UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE CIENCIAS DE LA DOCUMENTACIÓN
Departamento de Biblioteconomía y Documentación

APLICACIONES DE LA SINDICACIÓN PARA LA GESTIÓN DE CATÁLOGOS BIBLIOGRÁFICOS.

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

Manuel Blázquez Ochando

Bajo la dirección del doctor

Juan Antonio Martínez Comeche

Madrid, 2010

ISBN: 978-84-693-7622-5

UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE CIENCIAS DE LA DOCUMENTACIÓN

Departamento de Biblioteconomía y Documentación

APLICACIONES DE LA SINDICACIÓN PARA LA GESTIÓN DE CATÁLOGOS BIBLIOGRÁFICOS

Trabajo de Investigación presentado por el licenciado en Documentación D. Manuel Blázquez Ochando para la obtención del Título de Doctor en Documentación por la Universidad Complutense de Madrid bajo la dirección del Profesor Dr. Juan Antonio Martínez Comeche.

Madrid, Diciembre 2009

A mi familia y amigos

Índice

Capítulo 1. Introducción

Uno de los aspectos más relevantes en el desarrollo de las Ciencias de la Documentación es la mejora y evolución de los servicios documentales en las distintas unidades de información y documentación. El presente estudio incide en un nuevo enfoque para orientar los servicios bibliotecarios relativos al catálogo bibliográfico mediante nuevas técnicas de redifusión de la información y de los contenidos documentales.

El empleo de estas técnicas, también denominadas de sindicación de contenidos, mejoran cualitativamente los servicios de gestión catalográfica y bibliográfica que actualmente se emplean. Esto se debe en gran medida a la posibilidad de facilitar el acceso a los servicios bibliotecarios de una forma permanente y directa mediante un canal de información especial y personalizado tanto para el usuario de la biblioteca como para su administrador.

Para hacer efectivo el proceso de investigación e innovación ha sido necesario el estudio previo de la fisionomía de los catálogos bibliográficos de referencia en línea, los actuales servicios de sindicación de contenidos aplicados a bibliotecas, los lenguajes de marcado extensible fundamentales, los formatos de sindicación y el principal protocolo de interacción cliente-servidor para la recuperación de información bibliográfica Z39.50.

Teniendo en cuenta las características y singularidades de todos los elementos señalados, se procede al diseño de las especificaciones necesarias para crear las nuevas aplicaciones de sindicación de contenidos destinadas a cada caso y servicio concreto. De forma consecutiva, se explica el funcionamiento y estructura de las herramientas creadas para finalmente extraer diferentes resultados de carácter cuantitativo, metodológico y práctico.

1.1. Objeto de estudio

El objeto de estudio es el desarrollo y aplicación de las técnicas de sindicación de contenidos en la gestión de catálogos bibliográficos. Esto implica cómo se efectúa la redifusión, el procesamiento, la representación y la recuperación de la información bibliográfica sindicada. Por tanto, el problema planteado es cómo mejorar, ampliar o innovar nuevas capacidades de servicio y redifusión de información a partir del catálogo bibliográfico de una biblioteca o lo que es lo mismo, la sindicación de colecciones bibliográficas.

La justificación para llevar a cabo la presente investigación surge de la necesidad de mejorar los procesos de interacción en los catálogos bibliográficos, partiendo del empleo de técnicas de sindicación. Por otro lado, la necesidad de comprender mejor las posibilidades y capacidades de la sindicación de contenidos aplicada a otros campos distintos del concebido originalmente de redifusión de información.

Con esta investigación se demuestra que la sindicación de contenidos puede utilizarse como método alternativo al protocolo Z39.50. También se demuestra la posibilidad de sindicar catálogos bibliográficos que sirvan para confeccionar fácilmente catálogos colectivos, así como el control bibliográfico nacional y universal. Otros resultados importantes son los que se desprenden de la herramienta desarrollada para la investigación, ya que se demuestra que es un sistema útil para la edición, catalogación, búsqueda y exportación de registros y referencias bibliográficas a modo de sistema híbrido que integra funciones propias del protocolo Z39.50, de catálogo OPAC y de sistema de gestión de bibliotecas. También se demuestra la utilidad de la sindicación de contenidos no sólo para la redifusión de información bibliográfica, sino para transmitir los servicios bibliotecarios y de gestión del catálogo propiamente dicho. Además, una de las consecuencias directas del desarrollo técnico de la investigación es la acumulación de una experiencia importante en la manipulación y control de la información formateada en lenguajes de marcado extensibles, que puede resultar de gran utilidad para ampliar el desarrollo original presentado en este documento.

Todos estos avances pueden repercutir en la mejora de los actuales sistemas de gestión de bibliotecas, catálogos en línea, e incluso en el desarrollo de protocolos de sindicación para la recuperación de información bibliográfica. Por otro lado, se pueden definir aplicaciones específicas en la automatización de los procedimientos administrativos en la administración pública, en la compartición y confección de catálogos unificados de carácter poli-documental, conformados por fondos de archivos, bibliotecas y centros de documentación, así como la sindicación de los principales productos de la documentación como autoridades, lenguajes de clasificación, etc.

En definitiva, como se puede suponer, las posibilidades de aplicación, repercusión, mejora e investigación son múltiples dado que implica una nueva forma de tratar y procesar la información, así como por la relativa sencillez de adaptación a los marcos documentales actualmente establecidos.

1.2. Metodología

La metodología empleada para la consecución del objeto de estudio se conforma a partir del método documental que de forma sistemática permite crear el marco teórico necesario de la investigación. Por otro lado se emplean los métodos técnicos y experimentales para generar una demostración práctica de los principios y aplicaciones de la sindicación en relación al catálogo bibliográfico y su gestión.

Precisando mejor el método bibliográfico-documental se abordan los siguientes temas de estudio:

- En primer lugar, el funcionamiento, estructura y características de los catálogos bibliográficos en línea de cara a la prestación de servicios de referencia en las bibliotecas. Con ello se pretende obtener un análisis de cómo es un catálogo bibliográfico OPAC, qué servicios presta al usuario, cómo es soportado por los sistemas de gestión de bibliotecas, cuáles son sus limitaciones y qué aspectos podrían ser mejorables.

- En segundo lugar, la revisión del protocolo de recuperación y transmisión de información bibliográfica Z39.50. Este estudio se debe a la relación directa existente con los catálogos bibliográficos en línea y los sistemas de gestión de bibliotecas que utilizan este protocolo para facilitar la recuperación e importación de registros bibliográficos entre otros procesos de la pre-catalogación. Esto significa una alta implicación de dicho protocolo en las actividades de gestión y acceso a los catálogos. Por ese motivo se hace fundamental su conocimiento.

- En tercer lugar, el estudio de la componente tecnológica de la sindicación de contenidos y su concepción. Ello conlleva el conocimiento teórico, práctico y técnico de los lenguajes de marcado y rutinas programadas implicadas directamente en las técnicas de sindicación de contenidos y que consecuentemente se utilizarán para desarrollar el apartado práctico de la investigación.

- En cuarto lugar, el estudio de las aplicaciones de la sindicación de contenidos adaptadas al entorno y servicios de la biblioteca. Esto permite conocer qué aplicaciones existen actualmente, cuáles son sus objetivos, cómo interactúan con el usuario, en qué aspectos mejoran los servicios de la biblioteca y cuáles serían deseables por su utilidad.

Establecido el marco teórico, se procede a una fase creativa de las posibles utilidades de sindicación. En este punto se enuncian y argumentan cuales son las metas, objetivos y procesos concretos que se van a diseñar y desarrollar. Estos se resumen en la transmisión de registros bibliográficos entre bibliotecas, la sindicación del catálogo bibliográfico y la gestión del catálogo bibliográfico mediante servicios sindicados.

A continuación le sigue una fase de diseño en la que se plasman dichas aplicaciones mediante diagramas que facilitan la comprensión de su mecánica y funcionamiento. Para integrar todos los vectores de la investigación, ha sido necesario un diseño modular que a modo de plataforma ha permitido simular un entorno de aplicación realista sobre el que se disciernen todos los pasos del experimento.

Seguidamente se sucede una fase de construcción del modelo planteado. Ello conlleva la explicación de las herramientas desarrolladas, su código fuente y las soluciones técnicas que se han utilizado para cada caso.

Una vez preparado el sistema se da lugar a la fase de prueba basada en el empleo de colecciones bibliográficas, su transformación para ser objeto de sindicación, su transmisión, redifusión, recuperación y edición a partir de servicios sindicados. Los resultados de la prueba han sido obtenidos mediante una metodología experimental por cuanto puede ser medido y cuantificado, como es el caso de la medición de tiempos de ejecución de los programas. Pero no debemos considerarla pura dada su mayoritaria componente técnica. Esto se debe a dos factores, por un lado a la inexistencia de sistemas informáticos análogos al propuesto y por otro lado al afán de demostrar las capacidades y posibilidades reales de la sindicación de contenidos en el contexto bibliográfico. Por ello destaca un carácter fáctico destinado a la comprobación de hechos tanto por ausencia como por presencia de los fenómenos y propiedades características de la sindicación de contenidos en los casos planteados.

De esta forma queda asegurado un método de contraste sencillo que permite no sólo su comprobación técnica y experimental, sino bibliográfica y documental en relación a las fuentes de información utilizadas principalmente en el marco teórico de la investigación.

Obtenidos y fijados los resultados de la prueba, bajo un entorno y circunstancias de aplicación definidas, se lleva a cabo una fase de análisis e interpretación de los mismos. Este apartado también incluye su comparación con terceros sistemas y aplicaciones relacionadas, concretamente con los sistemas de gestión de bibliotecas, los catálogos bibliográficos en línea OPAC y el protocolo Z39.50.

Como consecuencia de la investigación, surge un nuevo apartado destinado a definir y proponer futuras investigaciones y líneas de trabajo relativas a la configuración de redes de catálogos archivísticos, bibliográficos y documentales sindicados, el desarrollo de procedimientos administrativos sindicados, redes de autoridades sindicadas, redes de tesauros sindicadas, alimentación automática de catálogos bibliográficos mediante sindicación, modelos de control bibliográfico nacional mediante sindicación, entre otros.

1.3. Fuentes y bibliografía

En relación al objeto de estudio planteado, existen multitud de fuentes de tipo electrónico que reflejan una importante actividad investigadora y de aplicación de la sindicación de contenidos con el objetivo de la redifusión de información. Tales aplicaciones son citadas principalmente en diversos estudios relativos a la Web2.0 (FRANGANILLO, J. and Catalán, M.A., 2005).

Por otro lado, existen algunas intromisiones en aspectos que originalmente son menos conocidos en la sindicación de contenidos. Estos resultan los más interesantes, pero también los menos numerosos, en los que se debe destacar las investigaciones sobre el portal de contenidos de revistas de biblioteconomía y documentación *Temaria* (RODRÍGUEZ GAIRÍN, J.M. et al., 2006), el empleo de la sindicación para la representación de los resultados de una búsqueda bibliográfica, tal como viene desempeñando la base de datos de medicina *PubMed* (GIGLIA, E., 2007), la utilización de la sindicación de contenidos a efectos de boletín de alerta bibliográfica, temáticamente filtrada como viene a determinar la *Biblioteca de la Universidad Nacional de Australia* (ANU, 2008) y a la postre el estudio sobre modelos semánticos de difusión selectiva de la información que se sirve de la sindicación como principal canal de transmisión y comunicación (PEIS, E. et al., 2008).

Son muy destacables los trabajos del investigador japonés Shoichi Taniguchi no sólo por su trayectoria relativa al desarrollo de catálogos bibliográficos, su automatización y modelado (TANIGUCHI, S., 2009) como por sus estudios sobre metadatos en RDF para la mejor identificación y descripción unívoca de registros bibliográficos en los catálogos (TANIGUCHI, S., 2005), (TANIGUCHI, S., 2006) que permitieran desarrollar un sistema para la catalogación y mejora de la representatividad y recuperación de los documentos (TANIGUCHI, S., 2007). Su aportación a esta investigación se centra en el modelo de interfaz y arquitectura tecnológica del sistema que desarrolló así como en el modelo de representación de las áreas de catalogación y organización estructural mediante el empleo de metadatos.

Sin embargo los estudios más relevantes para esta investigación son los presentados por los investigadores serbios Dusan Surla, Srdan Skrbic y Danijela Boberic (BOBERIC, D. and Surla, D., 2007), (SKRBIC, S. and Surla, D., 2008), (BOBERIC, D. and Surla, D., 2009) al desarrollar un editor de registros bibliográficos en un entorno XML nativo. Las ideas clave de su innovación se basa en el empleo de un formato XML propio, basado en Unimarc y MARC-XML, el empleo de bases de datos nativas en XML, la introducción de una metodología de transformación de los registros bibliográficos entre formatos y la incorporación del lenguaje de consulta extensible XQuery y esquemas XSD de Z39.50 para la interrogación de múltiples bases de conocimiento. Tales características desvelan una línea de investigación tendente a transformar el modo de trabajo con los catálogos bibliográficos utilizando unas tecnologías comunes a las empleadas en sindicación de contenidos. Es importante señalar, además, que tales investigaciones han corrido en paralelo con respecto a la presentada en este trabajo, por lo que aún partiendo de diferentes métodos y procesos, tratan de obtener en algunos casos objetivos muy similares, tal y como se desvelará más adelante.

En cuanto a las fuentes tecnológicas que han permitido el desarrollo de un sistema de sindicación para la gestión de colecciones bibliográficas, hay que señalar las especificaciones del consorcio de la web (W3C CONSORTIUM, 2009), ya que presenta todos los estándares de construcción y programación de la red. Son destacables igualmente las fuentes del W3School, (W3Schools Online Web Tutorials. Refsnes Data, 2009), que pretende dotar de una dimensión práctica las especificaciones del consorcio W3C. Por otro lado en materia de sindicación se han considerado muy especialmente las especificaciones de los formatos RSS2.0 (USERLAND; BERKMAN CENTER, 2003), RSS1.0 (BEGED DOV, G. et al., 2008), Atom enabled (The Atom syndication format, 2005) y MARC-XML (MARC-XML Schema, 2009) que dictan estructuras y etiquetas muy concretas para la construcción de los canales e ítems de que están compuestos. Finalmente se ha consultado de forma profusa la documentación oficial de php.net (PHP: Hypertext Preprocessor, 2009) para la programación del sistema de sindicación y sus funciones, así como la documentación oficial de la base de datos MySQL (SUN MICROSYSTEMS, 2009) como soporte intermedio de la base de conocimiento manejada en los canales de sindicación.

También se han utilizado fuentes institucionales, en especial documentación relativa a normas y estándares de catalogación y recuperación de información bibliográfica, tales como las normas MARC (MARC Standards, 2009) y el estándar NISO de Z39.50 (NISO; LOC, 2003) ambas participadas por la Library of Congress. Otras normas de importancia para la descripción bibliográfica son las ISBD (IFLA (INTERNATIONAL FEDERATION OF LIBRARY ASSOCIATIONS AND INSTITUTIONS), 2007), que han sido utilizadas para estudiar la capacidad de representación y adaptabilidad de los formatos de sindicación a la hora de contener información bibliográfica estructurada.

En cuanto a las fuentes utilizadas para la organización, estructuración del presente trabajo así como la aplicación de la metodología científica, se han tenido muy en cuenta manuales clásicos de investigación como por ejemplo (VALLES, M.S., 1999) (DIETERICH STEFFAN, H., 2001) (SIERRA BRAVO, R., 2002) (LÓPEZ YEPES, J. et al., 2005). No obstante no han sido las únicas referencias sobre metodología, dado que el desarrollo informático del sistema de sindicación ha requerido del empleo de metodología especial, concretamente la metodología Métrica versión 3 (MÉTRICA. VERSIÓN3: Metodología de Planificación, Desarrollo y Mantenimiento de sistemas de información, 2005), específicamente creada para la planificación, desarrollo y mantenimiento de sistemas de información.

1.4. Estado de la cuestión

La sindicación de contenidos presenta un desarrollo relativamente corto desde sus comienzos, ligados a los sistemas de publicación en red. Esto ha significado que la sindicación de contenidos se basara únicamente en la redifusión de información mediante canales específicamente diseñados para tal propósito. Bajo esta perspectiva se han ordenado todos los servicios que actualmente se proveen en las bibliotecas y centros de información en general. Concretamente destacan las aplicaciones de difusión selectiva de la información orientada al usuario, en la que se transmiten diversos canales de información bibliográfica de tipo temático. También se han desarrollado mejoras en materia de boletines de novedades bibliográficas, demostrándose en estos casos que la sindicación resulta un método eficaz para la notificación de las altas en los catálogos bibliográficos. Terceros desarrollos han permitido que la sindicación se aplique para suscribir y guardar los resultados de búsquedas y consultas efectuadas en un sistema de información o catálogo especializado, facilitando su posterior acceso o revisión. Pero tales conocimientos y aplicaciones de la sindicación a los servicios bibliotecarios han encontrado un límite en su evolución, debido a la dificultad de superar el método de comunicación unidireccional que caracteriza cualquier canal de información sindicada.

Por este motivo el actual papel de la sindicación en el catálogo bibliográfico es secundario sirviendo únicamente de complemento para la redifusión de contenidos delimitados por su temática y por su tamaño. A pesar de ello, bajo esta línea de investigación, existen estudios que han tratado de avanzar en materia de formatos de sindicación de contenidos en mayor medida adaptados a las necesidades bibliográficas, tomando como punto de partida metadatos y lenguajes de marcado extensibles especiales cuyo objetivo era lograr una representación catalográfica cada vez mayor que permitiera a la postre su transferencia y comunicación.

El objetivo planteado en esta investigación conlleva retomar la idea de aplicar un formato de descripción bibliográfica basado en lenguajes extensibles de marcado. Este avance permitirá no solo una mayor representatividad de la colección, sino la sindicación completa de los catálogos bibliográficos, favoreciendo su transmisión entre bibliotecas, así como la creación de catálogos colectivos. Como consecuencia de la sindicación bibliográfica, también se logrará la sindicación de los servicios bibliográficos que atañen a la gestión del catálogo como son la edición del catálogo, la recuperación de información y su ordenación.

Esto implica un importante avance en cuanto al empleo de la sindicación como alternativa a los servicios provistos por el protocolo Z39.50, ofreciendo de esta forma una continuidad en la evolución de la sindicación y la superación de sus barreras de comunicación unidireccional de difusión de información, por la interacción con el usuario y su acceso común a todas las fuentes de información de la biblioteca. Como conclusión, se puede afirmar que la sindicación de contenidos constituye una técnica de difusión e interacción de contenidos cuyas funcionalidades de recuperación de información, estructuración de los datos, representación de la información y proyección de nuevos servicios conllevarán un mejor y más amplio acceso a la información para los centros y unidades de información y documentación que los incorporen.

Capítulo 2. Los servicios bibliotecarios en la gestión bibliográfica

Las bases de cualquier desarrollo tecnológico aplicado a la gestión bibliográfica, implica el consabido estudio de los servicios bibliotecarios fundamentales, partiendo de su definición, funciones, objetivos y características más importantes. Una vez analizadas, el estudio se centra en los servicios web de la biblioteca, que con frecuencia resultan ser extensiones de los servicios originales para ampliar la accesibilidad a un público objetivo cada vez más interconectado. Este segmento constituirá el centro de atención del capítulo, destacando por ello los servicios de catálogo bibliográfico en línea OPAC y recuperación de información Z39.50. Ambos servicios se encuentran ampliamente integrados en los sistemas de gestión de bibliotecas modernos, por lo que se atenderá al estudio de sus funciones en correlación con la edición del catálogo bibliográfico.

2.1. El concepto de biblioteca y servicio bibliotecario

El concepto biblioteca antecede a la definición de servicio bibliotecario. Esto se debe a que una biblioteca es el centro de desarrollo y armonización de las colecciones documentales. Es decir, la biblioteca, en sentido genérico, está formada por múltiples tipos documentales y en relación a ellos ofrece una serie de servicios y prestaciones, destinadas a un público objetivo, perfectamente tipificado por sus necesidades informativas.

En torno a la definición de biblioteca, la norma ISO 5127 la define como *"Cualquier colección organizada de libros y publicaciones en serie, u otros tipos de documentos gráficos o audiovisuales, disponibles para préstamo o consulta."* (ISO 5127-1. Documentación e información. Vocabulario. Parte1: conceptos fundamentales, 1983). Tal explicación de biblioteca resulta escueta, pero suficiente para aportar las primeras ideas en relación a los servicios que de la biblioteca subyacen. Estos son fundamentalmente el préstamo y la consulta. El préstamo por ser considerado el método idóneo para el uso efectivo de los documentos y su mayor aprovechamiento. En cuanto a la consulta por ser la acción primigenia que antecede a cualquier préstamo y que permite la recuperación y repaso de las alternativas documentales que satisfacen las necesidades del usuario.

Según la norma UNE 50113-1 biblioteca es el *"Organismo o parte de él cuya función principal consiste en constituir bibliotecas, mantenerlas, actualizarlas y facilitar el uso de los documentos que precisen los usuarios para satisfacer sus necesidades de información de investigación, educativas o de esparcimiento, contando para ello con personal especializado."* (UNE 50113-1 Documentación e información. Vocabulario. Parte1: conceptos fundamentales, 1992). En esta definición se precisan mejor las necesidades del usuario en relación a los contenidos que la biblioteca mantiene y actualiza. Debemos señalar el concepto de mantenimiento y actualización de la colección documental, puesto que son tareas que implican la pre-catalogación, catalogación, clasificación, indexación, vaciado de contenidos y elaboración de puntos de acceso a tenor de otros trabajos. Según esta valoración, la biblioteca no sólo proporciona servicios, sino que tales servicios requieren de un mantenimiento para funcionar correctamente. Lo cual implica la gestión propiamente dicha de las colecciones bibliográficas, tema central propuesto en este estudio.

Finalmente, biblioteca según la norma UNE-EN ISO 2789 es aquella *"Organización o parte de ella cuya principal función consiste en mantener una colección y facilitar, mediante los servicios del personal, el uso de los documentos necesarios para satisfacer las necesidad de información, de investigación, educación y ocio de sus usuarios."* (UNE-EN ISO 2789 Información y documentación. Estadísticas de bibliotecas para uso internacional (ISO 2789:2003), 2004). Al igual que las definiciones mencionadas anteriormente, esta norma más actualizada del concepto de biblioteca, redunda en el factor usuario, destacándolo como el centro de atención al cual están destinados no sólo los fondos y colecciones documentales de la biblioteca, sino los servicios de acceso a los mismos. Esto implica un aspecto nada desdeñable, la experiencia del usuario y la satisfacción de sus necesidades. Tales variables son y han sido consideradas siempre la meta a perseguir por cualquier unidad de información y documentación en su camino por lograr la excelencia y la calidad de sus servicios.

Resumiendo lo citado, si la biblioteca es el centro de organización de las colecciones documentales, destinadas a un público objetivo, definido según sus necesidades investigadoras, informativas o documentales, los servicios bibliotecarios son todas aquellas actividades llevadas a cabo para asegurar la satisfacción de las mismas.

Entre tales actividades, según (ORERA ORERA, L., 2002), se distinguen dos ejes fundamentales: por un lado, aquellas que pertenecen a los servicios bibliotecarios relacionados con la información y la referencia al usuario y por otro el servicio de lectura en sala y préstamo. Sin ser la única distinción, en la definición de biblioteca, propuesta por (GARCÍA CAMARERO, E. and García Melero, L.A., 1999, pp.13-16) las integra en un marco conceptual mucho más completo, definiendo que *"Es un sistema de comunicación que pone en contacto la edición mundial con la comunidad a la que atiende mediante la realización de una serie de actividades que requieren la aplicación de unos conocimientos, códigos y normas para la ejecución de los procesos y funciones en que se descomponen… consisten básicamente en las siguientes: Selección y Adquisición, catalogación y clasificación, Información bibliográfica, Acceso a los documentos, Administración y gestión."* Significa que la prestación de servicios es una atención que se vale de la comunicación y la difusión de información a partir de unos códigos que rigen unas determinadas y actividades claves de la biblioteca. Si bien Orera se centraba en el servicio de información y referencia, así como en el servicio de lectura en sala y préstamo domiciliario, como principales servicios a partir de los que se desarrollan el resto, en este caso se ponen de manifiesto otras tareas que atañen a las actividades y funciones de preparación del fondo bibliográfico, como la selección y adquisición o la catalogación y clasificación. Dicho de otra manera, se conciben los servicios bibliotecarios como un todo comprendido en un sistema o cadena documental, sobre la que es posible observar su completa interrelación.

Si bien lo descrito atiende al concepto de biblioteca y servicio bibliotecario, no alude a su aplicación de los servicios web bibliotecarios en la red internet. En este sentido existen otros conceptos que han generado una gran ambigüedad como biblioteca virtual, biblioteca digital, biblioteca híbrida, biblioteca 2.0 o biblioteca integral. Esto obliga a esclarecer las diferencias y semejanzas entre estos conceptos, partiendo de los estudios realizados por otros autores, sobre los que poder establecer una posición teórica en esta investigación.

Una de las primeras definiciones de biblioteca digital según (TRAMULLAS SAZ, J., 1998, pp.263-282) estima que es *"...un sistema de tratamiento técnico, acceso y transferencia de información digital, estructurado alrededor del ciclo de vida de una colección de documentos digitales, sobre los cuales se ofrecen servicios interactivos de valor añadido para el usuario final."*

Esta definición supone un enfoque tecnológico centrado en las publicaciones electrónicas que una biblioteca digital puede ofrecer con cierto valor añadido, en su entorno o sede web, aunque explícitamente no queda mencionado. Ello supone que para hablar de biblioteca digital deben existir dos elementos: en primer lugar un sitio o sede web con soporte tecnológico y en segundo lugar un catálogo electrónico o publicaciones electrónicas para su difusión.

Según la definición de (GARCÍA MARCO, F.J., 2006, pp.651-670), *"... una biblioteca digital se puede definir de forma sencilla como un sistema de información que mantiene o proporciona acceso remoto a varias colecciones de publicaciones digitales […] la biblioteca digital no es una colección digital, aunque constituya su aspecto más visible, sino todos y cada uno de los elementos y procesos humanos, tecnológicos, normativos, económicos y materiales que hacen posible su existencia…"*

El aporte fundamental, en este caso es la implantación de otros ingredientes en la biblioteca digital: no sólo un sitio web con soporte tecnológico o una colección digital o electrónica de publicaciones o documentos, sino que incluye todos los elementos y procesos que hacen posible su existencia, lo cual explica un interés por abarcar más aspectos y generalizar su implicación al equipo humano que lo soporta. Resulta interesante analizar que la biblioteca digital, según (GARCÍA MARCO, F.J., 2006), establece dentro del apartado de las bibliotecas digitales una tipología que engloba otro tipo de bibliotecas, que corresponden con algunos términos señalados inicialmente por su ambigüedad; se trata de biblioteca digital *"... una organización bibliotecaria que proporciona sus servicios completos en la Internet. Es el caso de las colecciones de textos..."* , biblioteca digital institucional *"... es cada vez más frecuente que las instituciones que realizan algún tipo de actividad editorial o, sencillamente, que deben o desean publicar sus memorias periódicas, incluyan en su sede WWW una sección que proporciona sus publicaciones en su versión íntegra…"* y biblioteca digital universal *"…muchos teóricos plantean que la Internet y sus catálogos y buscadores conforman el germen de una futura biblioteca digital universal distribuida, que constituiría el horizonte final de la revolución digital en el campo de las bibliotecas, los archivos y la documentación…"*.

Pero los tipos de bibliotecas señalados no son los únicos con los que se configura el mapa de aplicación de los servicios web bibliotecarios. Existen otro tipo de bibliotecas en las que podría ser susceptible su asignación; es el caso de la denominada biblioteca integral, que es definida por Acebes Jiménez como *"...una organización humana y un sistema, con una estructura funcional dinámica, que gestiona recursos y servicios de información públicos e internos, independientemente de los soportes en los que se puedan difundir y de su localización [...] Esta organización proporciona acceso a estos recursos por diversos medios y en diferentes puntos, ya impliquen la interacción directa con personas y con locales e instalaciones o la mediación de ordenadores o de cualesquiera otros dispositivos y mecanismos..."* (ACEBES JIMÉNEZ, R. and Magán Walls, J.A., 2002, pp.46-60). Según la definición de biblioteca integral, es aquella que unifica la biblioteca tradicional y la biblioteca virtual que consta de los diversos medios, recursos y canales de difusión.

En cuanto a la biblioteca electrónica, como marco de servicios web bibliotecarios, supone un concepto más técnico que afina connotaciones como la necesidad de suministro eléctrico para la consulta de los documentos. En este sentido se explica la definición de Roy Tennant: *"biblioteca compuesta por un conjunto de materiales y servicios electrónicos. Los materiales electrónicos pueden incluir todos los tipos de materiales digitales, así como la extensa variedad de formatos analógicos que requieren de electricidad para su uso [...] Por ello el término biblioteca electrónica entronca o enlaza con todos los materiales que pueden ser enlazados por una biblioteca digital, siendo ésta denominación más inclusiva..."* (TENNANT, R., 1999) Pero la verdadera aportación de este autor se sitúa en la definición de biblioteca digital, que apunta de la siguiente manera: *"...una biblioteca digital es una biblioteca que está compuesta por un conjunto de materiales y servicios digitales. Los materiales digitales son objetos que pueden ser almacenados, procesados y transferidos por un canal binario de unidades y redes telemáticas. Los servicios digitales son servicios que han sido desarrollados sobre equipos y redes con soporte binario..."* tal y como sostiene en ensayos posteriores (TENNANT, R. et al., 2008) finalmente concluye en su trabajo que las bibliotecas electrónicas y digitales pueden ser bibliotecas virtuales, lo que significa una independencia y abstracción que hacen que no existan en un entorno físico.

Completando el paradigma de la terminología aplicada a los tipos de bibliotecas sobre las que se aplican los servicios web bibliotecarios, cabe destacar también el concepto de biblioteca híbrida. Según lo que aporta Jesús González Lorca y José Vicente Rodríguez Muñoz, basándose en el trabajo de autores que preceden en el tratamiento de este tipo de bibliotecas, como Odlyzko, Gilster, Clawford, Gorman y Rusbridge: *"...la biblioteca digital implica el concepto de biblioteca híbrida, erigiéndose éste último como el paradigma de integración entre la biblioteca tradicional y la digital."* (GONZÁLEZ LORCA, J. and Rodríguez Muñoz, J.V., 2002, p.160). Según los propios autores indican, se puede definir la biblioteca híbrida como una característica más de las que consta la biblioteca digital, siendo entonces considerado un término que pretende integrar la biblioteca tradicional y la biblioteca digital, tanto en sus funciones, como en sus mecanismos y sistemas de difusión de la información más relacionados con el entorno digital.

Una excelente visión de la biblioteca digital es la aportada por los profesores Ernesto García Camarero y Luís Ángel García Melero: *"...La biblioteca digital es un organismo encargado de seleccionar, adquirir, controlar bibliográficamente, almacenar, conservar y poner a disposición de los usuarios una colección de publicaciones electrónicas [...] Los procesos y servicios de la biblioteca digital o de la colección digital, si ésta forma parte de una biblioteca mixta... se estructuran en torno a las siguientes actividades*: *desarrollo de las colecciones, catalogación, organización y conservación del depósito digital y servicios bibliotecarios [...] Como se puede apreciar no hay tanta diferencia entre la organización funcional de la biblioteca convencional y de la electrónica, salvo la naturaleza de la información y las características de los documentos que inciden en la realización de los procesos y en la prestación de servicios..."* (GARCÍA CAMARERO, E. and García Melero, L.A., 2001, p.183). En estos párrafos se exponen las actividades, procesos y recursos en torno a los cuales gira la biblioteca digital que tiene unos objetivos concretos similares a los de la biblioteca convencional, tal y como se relata a lo largo de la definición, ya que la naturaleza de la información o las publicaciones, constituyen el aspecto que varía.

También existen puntos de vista uniformadores del concepto biblioteca digital y virtual. Según (RODRÍGUEZ BRAVO, B., 2002, p.223). *"...la existencia de documentos digitales-virtuales, permite la biblioteca digital-virtual. Ambos adjetivos se complementan, ya que si digital hace referencia a la composición o estructura de los documentos, y por extensión a las bibliotecas que los contienen, virtual se refiere a la accesibilidad documental, y con respecto a las bibliotecas, se aplica aquellas que ofertan servicios a distancia. En sentido estricto, una biblioteca digital sería una biblioteca sin papeles y una biblioteca digital, una biblioteca sin paredes que se contrapondría a la biblioteca presencial o biblioteca física."*

Finalmente es necesario tener en cuenta la última aportación terminológica relativa a los tipos de biblioteca relacionados con la red Internet. Se trata de la biblioteca 2.0. Según Michael Casey, introductor original del término, *"...La biblioteca 2.0 no es una manera de reemplazar los servicios más tradicionales de una biblioteca. Es una manera de extender dichos recursos dentro de las nuevas áreas de la red y comenzar en otras nuevas. No es un cambio para provocar el cambio."* (CASEY, M.E. and Savastinuk, L.C., 2006) Ante esto, Michael Habib, uno de los principales promotores de la terminología 2.0, añade: *"...La biblioteca 2.0 describe una serie de servicios diseñados para satisfacer las necesidades de los usuarios, causadas por los efectos directos o periféricos de la web 2.0..."* (HABIB, M.C. and Carr, D., 2006). En relación a este incipiente término, la biblioteca 2.0 es presentada como una ampliación de las características y servicios de la ya de por sí completa biblioteca digital, por lo que según se discierne de la lectura de dichos autores, se emplean las nuevas aplicaciones y avances multimedia para mejorar la interacción con el usuario. Un ejemplo de la biblioteca 2.0 lo constituyen las técnicas de sindicación de contenidos aplicadas a los servicios web, lo que constituye parte del objeto de investigación en este trabajo.

Como resultado de las explicaciones vertidas sobre los distintos tipos de biblioteca, el marco teórico elegido para el presente estudio será el de Biblioteca Digital, por considerarse el concepto más aproximado a la prestación de servicios web bibliotecarios. No obstante hay que aclarar que todos los conceptos definidos son válidos por aportar rasgos y características complementarias, que pueden ser aunadas en dicho marco teórico.

De hecho la extensión de la biblioteca a la red internet es sin duda la característica central que permite hablar de unos servicios bibliotecarios en línea, es decir, su acceso telemático. Estos servicios no serían posibles sin las infraestructuras necesarias para su correcto funcionamiento. Por ende estaríamos hablando de bibliotecas electrónicas, según los planteamientos anteriores, pero también de bibliotecas digitales, puesto que los catálogos colectivos son de libre consulta mediante catálogos electrónicos OPAC. Por lo tanto pueden o no incluir documentación electrónica o digital entre sus contenidos. Pero para permitir que la biblioteca pueda ofrecer todos los servicios expuestos de una manera integrada, se necesita una sede web oficial o institucional correspondiente a la biblioteca que permita su representación adecuada en la red. Al incluir un sitio web integrador, se incluye asimismo un equipo humano y técnico necesario para su mantenimiento y actualización, por lo que a su vez se cumple el concepto de biblioteca integral e híbrida, puesto que permite proporcionar servicios digitales en relación a los tradicionales ofrecidos en el espacio físico de la biblioteca. Pero además la biblioteca es 2.0, por el hecho que supone el avance en las técnicas de programación y representación de información que implican unas bases tecnológicas renovadas para una mayor interacción con el usuario. El caso de la biblioteca virtual supone una emulación digital de la biblioteca tradicional y de sus servicios en su espacio físico. Si bien la biblioteca tradicional está limitada por el espacio disponible entre estanterías, baldas y depósitos, una biblioteca virtual permite la descentralización del espacio mediante unidades de almacenamiento distribuidas. Dicho de otra forma, la biblioteca virtual es aquella compuesta de materiales documentales y servicios especializados en un entorno de red global distribuida, de forma tal que empleando las técnicas de programación, recuperación y representación de información es capaz de emular digitalmente, los servicios, recursos y fuentes de la misma. Con esta reflexión, se justifica definitivamente, que la biblioteca digital engloba las características de la biblioteca electrónica, integral, híbrida, 2.0 y virtual, siendo por ello el más completo y preciso para referirse a la prestación de servicios web bibliotecarios.

2.2. Evolución de los servicios bibliotecarios

Desde el comienzo de la biblioteca universitaria como centro de información y documentación para la investigación y el aprendizaje de las diversas áreas de conocimiento, hasta las más modernas bibliotecas digitales, se ha comprobado que la base para la prestación de cualquier servicio al usuario es la disposición de un fondo bibliográfico y una colección de fuentes de información en múltiples formatos, adecuada con las necesidades y demandas del público objetivo.

En esta evolución de los servicios bibliotecarios, se pueden distinguir tres periodos bien diferenciados a lo largo del siglo XX y comienzos del XXI, a saber: *Primer periodo*, de gestión, reorganización y asentamiento de los recursos documentales que establecen las bases de los servicios bibliotecarios tradicionales. *Segundo periodo*, de automatización de la gestión y adaptación de los recursos documentales a los sistemas de la informática-documental, que permiten su control y ampliación de su difusión mediante nuevos servicios de atención y recuperación de la información. *Tercer periodo*, de implantación de la biblioteca digital, generando servicios documentales, de información y referencia con valor añadido, convirtiéndola en un centro analítico de los contenidos.

El primer periodo de servicios se basa exclusivamente en torno al material bibliográfico, mediante métodos tradicionales de descripción y recuperación de información. Dicho periodo se puede determinar desde comienzos del siglo XX hasta la hasta la implantación de los primeros sistemas de automatización de bibliotecas, entre 1970 y 1980. Durante este gran periodo de tiempo, la biblioteconomía sufre importantes procesos de profesionalización, determinación y mejora de reglas de catalogación e instrumentos de descripción, gestación de los primeros programas y políticas de información y apoyo a los centros de documentación e investigación y la consolidación del control bibliográfico nacional mediante el sistema de depósito legal. Todas estas circunstancias tuvieron sus repercusiones en la prestación de servicios bibliotecarios básicos, si bien las circunstancias no siempre lo permitieron en España debido al marasmo generado después de la guerra civil. Posteriormente sí se consiguieron avances significativos, que continuaron y ampliaron con la llegada de la democracia moderna. Algunos ejemplos de estos avances fueron la gestación de los primeros catálogos provisionales de patrimonio bibliográfico español entre 1970 y 1980, que serían concretados y oficialmente conformados en cumplimiento de la *Ley 16/1985 del Patrimonio Histórico*

Español. Citado este caso fundamental e ineludible, se debe recordar que, pese a los problemas heredados en la inmensa mayoría de las bibliotecas, durante las últimas cuatro décadas de este periodo se procedió en gran cantidad de bibliotecas universitarias y de referencia a un proceso de reorganización, control y normalización de las herramientas descriptivas de su fondo bibliográfico, mediante la edición de múltiples catálogos que facilitaron a la postre una recuperación de los documentos en sala más eficaz, aunque no siempre fructífera, no exenta de problemas de localización topográfica. Dicho de otra manera, el servicio ofertado a los usuarios, y concretamente a los investigadores, consistía en la referencia bibliográfica, consulta de catálogos temáticos y listados bibliográficos de autoridades. La documentación era servida directamente en la sala de lectura, y sólo en las últimas décadas de este periodo fue posible la copia sencilla mediante sistemas reprográficos, que disminuyeron en cierta medida el tiempo que un usuario invertía a la hora de localizar los contenidos que deseaba.

El segundo periodo podría considerarse breve pero muy intenso, dentro de la escala cronológica de la evolución de los servicios bibliotecarios, pero ha sido fundamental para filtrar, subsanar y perfeccionar los catálogos bibliográficos y herramientas de descripción documental de las bibliotecas. Se trata del periodo de la automatización de los procesos de la cadena documental. Aproximadamente se puede establecer un rango temporal que incluiría desde mediados de la década de los 80 del siglo XX, hasta el final del año 2000. Durante este periodo de escasos 20 años, la biblioteca ha ganado la herramienta de la informática aplicada a la documentación. Se suceden desde estas fechas los proyectos de informatización y automatización de bibliotecas y archivos más ambiciosos jamás contemplados a nivel estatal, concretamente los planes de i+D (investigación y desarrollo) promovidos por el Ministerio de Cultura que logran informatizar y automatizar los primeros catálogos colectivos del país, empezando por la cabecera del sistema bibliotecario español, mediante "*...ARIADNA como resultado de la automatización y ampliación del catálogo original de 1945, quedando gestionado por el sistema de bases de datos SIRTEX y estando a plena disposición para su consulta ante los usuarios desde 1992, constando de medio millón de documentos registrados"* (HÍPOLA, P., 1994) atendiendo a los campos de descripción bibliográfica de *Ibermarc*[1]. (Formato IBERMARC para registros de fondos y localizaciones, 2004).

[1] **Ibermarc:** formato para la catalogación bibliográfica automatizada que emplea la Biblioteca Nacional de España, resultante de la adaptación del formato marc21.

El sector de la informática documental comienza a desarrollarse en España y a crear los primeros sistemas de gestión integral de bibliotecas, mediante módulos para el control de los servicios de préstamo domiciliario, inter-bibliotecario y catalogación. Sistemas como Absys, Sabini (GARRIDO ARILLA, M.R., 1993), Unicorn (GETHIN, P., 2001) tienen sus primeras experiencias a lo largo de los principales sistemas bibliotecarios universitarios y autonómicos, obteniendo un campo de pruebas y experiencias sin parangón. Esto se debe al fenómeno de discordancia de los sistemas de gestión ad-hoc, desarrollados en las bibliotecas que no atendían a ningún criterio de normalización. Dicho efecto generó una experiencia fundamental en la migración de datos bibliográficos que permitió asentar las bases de los estándares de descripción para su uso extensivo en el sistema bibliotecario español. Como consecuencia directa del impacto de estos sistemas de gestión bibliotecaria comenzarían a desarrollarse nuevos servicios de consulta y recuperación de información; concretamente la difusión de los sistemas *OPAC (An Online Public Access Catalogue, Catálogo en línea de acceso público)* (BRISCO, S., 2006, pp.56-57) para la consulta telemática de los catálogos bibliográficos y los primeros servicios de difusión selectiva de la información, basados en la gestión de sencillos listados documentales generados con dichas herramientas. Al tiempo que se normalizaban los asientos catalográficos, también se utilizan a los primeros listados electrónicos de autoridades, que empezaban a vincular y relacionar sus asientos con respecto a la documentación registrada.

El tercer periodo comprende desde el comienzo del siglo XXI hasta la fecha. En menos de siete años, los servicios bibliotecarios han sufrido una gran transformación en su expansión electrónica. De la perfección de los servicios principales, gracias a las herramientas de gestión bibliotecaria, se obtienen verdaderos OPACS descentralizados para la consulta de cualquier catálogo desde cualquier acceso a la red; se generalizan los sitios web oficiales de las bibliotecas y con ello se abre una nueva ventana a lo que se ha denominado la *biblioteca digital* y a una gran diversidad de servicios que no habrían sido posibles de no haberse consolidado unas normas o reglas de construcción normalizadas en este caso por el W3C Consortium. Se trata de un periodo de madurez tecnológica y estandarización de las técnicas para la prestación de cualquier servicio de información a un usuario remoto.

Estas circunstancias permiten a cualquier investigador, a diferencia de la primera etapa, evitar gran cantidad de desplazamientos e inversión de tiempo en las consultas de los catálogos. Esto se debe también a la disponibilidad de textos completos y obras de referencia que se encuentran a disposición de los usuarios, que si bien en teoría no deben reemplazar las fuentes primarias escritas, las están superando en popularidad y frecuencia de uso. Servicios como la biblioteca electrónica, buscadores integrados para la consulta de referencias dentro de los propios textos electrónicos, servicios de DSI (difusión selectiva de la información) soportados en sencillos formularios electrónicos automatizados incluyendo así el servicio de alerta y novedades bibliográficas, servicio de boletines y revistas electrónicas consolidadas, servicios de bases de datos en línea, servicios de reproducción digital a la carta, constituyen algunos ejemplos de la explosión técnico-documental gestada durante el periodo anterior y que durante la presente década está teniendo su máxima expresión y representatividad.

En los tres periodos de evolución de los servicios bibliotecarios existen varios factores clave, que son:

1. La correcta disposición de las fuentes documentales, bibliográficas primarias, secundarias y terciarias que constituyen la base de cualquier servicio ofertado.

2. La aplicación de las soluciones informáticas como factor dinamizador para el desarrollo de servicios telemáticos, generando un periodo de pruebas y experiencias para establecer los perfiles fundamentales de cualquier servicio electrónico.

3. La propagación de la red Internet que ha hecho posible los servicios de difusión de información des-localizados de la ubicación física de la biblioteca.

4. La madurez y normalización tecnológica aplicada a la automatización de los servicios, procesos y productos de la biblioteca, cuya trayectoria a lo largo de la historia ha podido ofrecer una madurez teórico-práctica suficiente como para tolerar y saber encauzar los cambios y avances propuestos para su modernización.

5. Mayor demanda de información y documentación y, en consecuencia, más investigadores y usuarios preparados para aprovechar la tradición de los servicios bibliotecarios, y en especial su expansión digital a través de la red.

La biblioteca digital es la extensión electrónica de la biblioteca tradicional. Se considera la existencia de una biblioteca digital, desde que se establece el principal y más común de los servicios a través de su sitio web oficial es decir, el servicio de información y referencia interna y externa. Así pues, el objetivo primario de una biblioteca digital es el desempeño de las labores de información y posteriormente de referencia.

Una biblioteca digital debe ser capaz de ofrecer los mismos servicios que una biblioteca tradicional, salvando las circunstancias técnicas. Si existe una característica que ha hecho que este tipo de extensiones bibliotecarias hayan tenido un importante éxito y fueran secundadas por los usuarios, es el hecho de que una biblioteca digital constituye un medio más fácilmente accesible para ellos, permitiéndoles hacer casi las mismas operaciones que en el espacio físico de la biblioteca, como por ejemplo reservar libros para su préstamo, consultarlos remotamente, recuperar referencias bibliográficas, consultar las incorporaciones de nuevos fondos en el catálogo de la biblioteca, entre otras posibilidades.

Todo ello es posible gracias a Internet y a la maduración de tecnologías de programación como los lenguajes de marcado, que como se expondrá más adelante, supone la base de la sindicación. Teniendo en cuenta este punto previo, se puede llegar a la sencilla conclusión de que esta libertad de acceso a la biblioteca digital mediante su disponibilidad en un espacio web, permite ampliar su público objetivo y potencial, de forma tal que se suceden y llevan a efecto las propiedades de la globalización de los recursos y servicios de información, como pone de manifiesto la famosa idea de la *aldea global* [2]. (MCLUHAN, M., 1989)

Por último es importante destacar otras circunstancias que atañen a la presentación de los servicios de una biblioteca digital. Según (MACÍAS GONZÁLEZ, J., 2005) la determinación del grado de acceso a Internet de la biblioteca, el apoyo institucional para el mantenimiento de la sede web, asumir la complementariedad de los recursos que la biblioteca dispone evitando

[2] **Aldea Global:** es un término acuñado por el sociólogo y profesor universitario, McLuhan, que se refiere al cambio en la sociedad humana a consecuencia de la inmediatez de las comunicaciones. En el contexto de este párrafo, se atiende al paradigma de la aldea global como consecuencia de la aplicación de las tecnologías de la información que han permitido el efecto de la globalización, interrelación y dependencia de la red.

la duplicidad e introduciendo las aplicaciones y recomendaciones de la *usabilidad* [3], *posicionamiento* [4], *accesibilidad* [5] y *arquitectura de la información* [6], permitir el mantenimiento diario y actualizado de los contenidos y novedades del servicio de información y proveer de los servicios correspondientes a los usuarios que accedan a la sede web. Además de estas características habría que añadir, que cada servicio ofertado, debe ser convenientemente descrito, en su finalidad, objetivos y prestaciones, de manera que todo usuario que acceda conozca su aplicación, y pueda determinar su complementariedad en la recuperación de la información relacionada con su investigación o estudio.

Como última reflexión del apartado, hay que afirmar la importancia del marco teórico de la biblioteca digital y los servicios web bibliotecarios para el desarrollo e implantación de las técnicas de sindicación de contenidos para la redifusión de la información.

---

[3] **Usabilidad:** en relación al grado de adaptación de los servicios con respecto al usuario, es además un término adoptado del inglés usability que tiene como objetivo determinar el grado de adaptación de los servicios de un sitio web o sistema informático a un determinado tipo de usuarios, en un contexto de interacción y retroalimentación. Un sistema ha de basar su usabilidad en la facilidad de aprendizaje, flexibilidad en sus servicios y pertinencia en sus resultados. El término usabilidad está siendo objeto de debate sobre su concepción y referencia. El padre del término es Jacob Nielsen, ingeniero de computación por la Universidad Técnica de Dinamarca y experto en diseño de interfaces de usuario.

[4] **Posicionamiento:** en referencia a la mejora de la visibilidad en la red, es además un término que en el contexto expresado se refiere a la técnica de optimización para motores de búsqueda, técnica procedente del inglés SEO Search Engine Optimization. El origen del posicionamiento web tiene su naturaleza en el pagerank del motor de búsqueda google. Consiste en emplear soluciones de programación, redirección y referenciación para obtener un emplazamiento más visible en la recuperación y consulta por una determinada cadena de caracteres

[5] **Accesibilidad:** en función de la correcta organización y empleo de la normalización de servicios y recursos web, Su desarrollo y normalización han sido promovidos desde sus inicios por los grupos de trabajo del W3C World Wide Web Consortium, mediante la WAI Web Accessibility Initiative o iniciativa de accesibilidad web que tiene como objetivo facilitar el acceso a la información y a los contenidos a personas con discapacidad. Esto se consigue mediante técnicas y esquemas de programación que favorecen la descripción y simplicidad estructural de los sitios web

[6] **Arquitectura de la Información:** o cómo estructurar y representar la información, sus servicios y los factores de interacción de los sitios web. Está directamente relacionado con la accesibilidad web, ya que uno de sus objetivos es la consecución de sitios web conformes a la normativa WAI. El origen del término se debe a Louis Rosenfeld y Peter Morville al publicar Information Architecture for the World Wide Web en 1998.

En este sentido, la implementación de estas técnicas, resulta factible en los servicios de información y referencia, consulta bibliográfica y los procedimientos normalizados del trabajo bibliotecario, interrelacionados en la cadena documental. Todos estos aspectos se estudiarán más detalladamente con la finalidad de obtener un diagnóstico de aplicación claro que favorezca la comprensión y el rol que puede desempeñar la sindicación.

2.3. Principales servicios bibliotecarios

Como se ha explicado anteriormente, se consideran servicios bibliotecarios a las prestaciones de acceso y difusión de los fondos y contenidos documentales de la biblioteca para satisfacer las necesidades educativas, científicas e informativas de diferentes tipos de usuario o público objetivo. Los servicios bibliotecarios también implican, necesariamente, una serie de procesos documentales derivados del análisis documental, la sistematización de los lenguajes documentales, la edición de materiales de consulta y referencia, la compilación y análisis de recursos de información y documentación o la planificación de actividades de extensión bibliotecaria. Además estos servicios bibliotecarios y procesos documentales, tienen su propio equivalente o aplicación en la red, conllevando una ampliación de sus posibilidades y una considerable automatización de los mismos. Por lo tanto la finalidad de este apartado es estudiar cuáles son los principales servicios bibliotecarios, susceptibles de automatización y sindicación, cuál es su aplicación o extensión para ser considerado un servicio web bibliotecario y de que procesos documentales se nutre para asegurar su funcionamiento.

2.3.1. Servicio de información y referencia

"...Consiste en una asesoría personalizada, que se presta a quienes acuden a la biblioteca en busca de información o necesitados de una orientación, con el objeto de que, auxiliados por el personal competente o solos, puedan seleccionar y acceder al material de su interés, mostrándoles, en su caso, cómo se manejan los instrumentos que pueden proporcionarles la información que necesitan." (LÓPEZ YEPES, J. (Coord.) and et.al., 2004, p.437)

El servicio de información y referencia bibliográfica es considerado una piedra angular del sistema bibliotecario. Su funcionamiento pone en relación directa al usuario con el bibliotecario y con los sistemas de información que la biblioteca desarrolla y contrata. Esta

interacción se puede generar mediante el trato personal con el bibliotecario o mediante sistemas de inteligencia artificial que aprovechando la técnica de cuestionarios estructurados, son capaces de guiar al usuario al recurso de información o información demandada. Pero los servicios de información y referencia tienen su extensión y definitiva solución a cualquier necesidad y demanda del usuario, en el servicio de consultas de bases de datos en línea, la difusión selectiva de la información, el servicio de búsquedas bibliográficas y alerta de novedades bibliográficas y documentales. Todos los servicios citados corresponden a algunas de las respuestas directas y operaciones que se efectúan para solucionar todo tipo de dudas, y problemas de acceso a la información que pueda presentar un usuario. Los servicios de recursos en línea están directamente relacionados con el servicio de información y referencia. Esto se materializa en las bases de datos para su consulta en línea, las colecciones de revistas electrónicas a las que una biblioteca se suscribe, los directorios de recursos electrónicos clasificados y el directorio de entidades y centros relacionados o vinculados. Los dos últimos servicios citados consisten en la agrupación de las referencias de espacios de recursos que son convenientemente descritos para su listado y organización. Los directorios de recursos electrónicos, están diseñados para la referenciación de aquellos espacios de información y documentación auxiliares que complementan los contenidos de la biblioteca en un área del conocimiento especializada. Los directorios de entidades y centros relacionados tienen la función de servir de testigo entre las relaciones de la biblioteca o unidad de información y documentación con respecto a centros y entidades públicas o privadas con las que colabora o tiene algún tipo de relación especial.

Bases de datos de consulta en línea

"Es la que permite que un centro de documentación pueda acceder a la casi totalidad del saber humano actualmente registrado, repertoriado o almacenado, a través de los datos bibliográficos que dan acceso a las referencias de los documentos." (LÓPEZ YEPES, J. (Coord.) and et.al., 2004, p.137)

En relación a las bases de datos de consulta en línea, constituyen colecciones de recursos organizadas de libre acceso (bases de datos documentales en línea cuya edición, acceso, confección y desarrollo corre a cargo del centro bibliotecario) o pago (bases de datos de recursos, elaboradas por otras instituciones externas a la biblioteca que suelen ser de acceso restringido), que permiten la obtención de complementos documentales e informativos en un

determinado área de conocimiento proporcionando al usuario toda referencia relacionada con su objeto de estudio. Suelen estar dispuestas en forma de directorio, descritas someramente y directamente enlazadas desde el sitio web de la biblioteca o unidad de información y documentación. En función de la información que contienen pueden clasificarse según el tipo de documento que describen y referencian, bases de datos de monografías, artículos y documentos científicos de revistas de un área de conocimiento concreto a texto completo, bases de datos de recursos electrónicos, bancos de imágenes, bases de datos especializadas en legislación, entre otros.

En el área de las bases de datos en línea, la sindicación de contenidos tiene un papel fundamental, ya que puede incidir directamente en la creación de fuentes de datos que estructuren la información y documentación que poseen. Esto permitiría el establecimiento de canales específicos de información en relación a las novedades, altas producidas y recuperación de información, pudiendo conocer su estado, sin necesidad de acceder al servicio de forma directa. Esto supone posibilidades de manipulación, exportación y distribución compartida de la información de manera directa sin necesidad de elementos intermediarios, entre el usuario y la fuente de información. Estas características entroncan con los servicios de difusión selectiva de la información que podrían ser aplicados junto con la sindicación de manera sistemática en este tipo de bases de datos.

DSI, difusión selectiva de la información

"... a través de este servicio, el acceso a la información, se realiza de forma acotada según el interés señalado previamente, por el usuario [...] El DSI puede realizarse de varias formas: individual, colectiva o estándar y personalizada [...] El DSI individual puede obtenerlo el usuario siempre que se produzca una demanda de información o de forma periódica mediante la consulta a un catálogo o bibliografía [...] El DSI colectivo o estándar se realiza cuando un mismo tipo de información interesa a un amplio colectivo de usuarios [...] El DSI personalizado necesita que el especialista prepare y envíe la información de forma continua y para un único usuario, según el perfil de acotación de la información que éste habrá determinado de forma concreta." (LÓPEZ YEPES, J. (Coord.) and et.al., 2004, pp.434-435)

Es el servicio especializado en cubrir las necesidades informativas del usuario, ante las cuales la biblioteca ofrece respuestas documentales a medida. Existen diferentes posibilidades de desarrollar este servicio: por regla general, el servicio se basa en la obtención de perfiles del

usuario en función a la demanda de información, temática, objeto de estudio o investigación del mismo. Esta información específica se obtiene, ya sea por la vía personal de entrevista o por la utilización de formularios de consulta para filtrar las novedades bibliográficas, documentales e informativas generando en ambos casos el citado perfil. A partir de un perfil establecido, la biblioteca mediante un sistema de gestión de contenidos para la publicación de novedades bibliográficas, o a través del propio sistema de gestión bibliotecaria, generan listas de distribución especializadas para cada tipo de usuario. Si bien las listas de distribución que difunden los boletines de novedades, información o noticias son transmitidas mediante listas de correos electrónicos, la sindicación puede considerarse como un método de distribución más inclusivo, flexible, especializado y escalable que las soluciones basadas en el tradicional correo. La sindicación de contenidos es capaz de crear fuentes de información propias de la biblioteca, que se actualizan con los contenidos y documentos que ésta edita, consiguiendo una actualización en cadena de las mismas con un tiempo de refresco que puede ser cifrado en cifras inferiores al segundo. Dicho de otra forma la sindicación permite generar tantos canales de distribución como perfiles o clasificaciones de intereses y necesidades de usuarios puedan darse en un entorno bibliotecario determinado, con las consiguientes ventajas de la sencillez de la suscripción de dichos usuarios a dichas fuentes de información, obteniendo respuesta inmediata. En consecuencia, se considera DSI a cualquier difusión de información tratada previamente por la biblioteca y sus profesionales, que por ende señala directamente a fuentes de noticias especializadas en un sector concreto, distribución de dosieres especializados e incluso la tan necesaria alerta de novedades bibliográficas relacionadas con el perfil del usuario. Todo ello indica que aun existiendo métodos de transmisión y difusión de datos avanzados, aún debe existir un tratamiento especializado para lograr el calificativo de información con valor añadido, que constituye una finalidad y objetivo primordial en la concepción de este tipo de servicios. En este sentido la sindicación también ofrece una respuesta clara a las necesidades de ampliación de la información y su correspondiente edición. Esto se consigue gracias a las posibilidades de extensibilidad de los formatos de sindicación, que permiten a la postre aumentar el número de campos y argumentos de descripción en el esquema original de descripción. También se ha de señalar que estas funciones y capacidades no están siendo explotadas en las actuales aplicaciones de sindicación para el entorno de servicios bibliotecarios, ya que los principales desarrollos de la sindicación están dirigidos a la visualización y recuperación de la información sindicada en los distintos canales o fuentes de datos en detrimento de las opciones de ampliación,

modificación y tratamiento de las fuentes de sindicación en sí mismas, descargadas previamente a modo de documento secundario.

Búsquedas bibliográficas

El servicio de consultas bibliográficas está inscrito también dentro de los servicios de información y referencia y comporta la recuperación de información, comúnmente bibliográfica así como aquella del ámbito documental ante la formulación de la consulta del usuario al bibliotecario o documentalista, así como a aquellos catálogos colectivos en línea disponibles. El funcionamiento del servicio consiste en la reformulación de la expresión del lenguaje natural del usuario, confeccionando una cadena de descriptores, palabras clave y operadores, que permiten recuperar la información que el usuario no logro recabar inicialmente. Para dicha recuperación pueden emplearse todo tipo de bases de datos de referencia y consulta en línea, el propio catálogo colectivo del centro de información y documentación, así como aquellos directorios y buscadores que procedan. Esto exige un conocimiento de la correcta organización y descripción de los recursos de recuperación y referencia bibliográfica, el empleo de lenguajes de consulta específicos, basados en técnicas booleanas o de programación como SQL o XQuery, según el entorno a interrogar. En este sentido, el servicio de búsquedas bibliográficas esta directamente basado en un servicio con entidad propia, denominado OPAC, que permite facilita la consulta automatizada, según campos descriptivos, mediante la reformulación de consultas en la base de datos del sistema de gestión de bibliotecas. No obstante la sindicación de contenidos puede mejorar de forma decisiva las posibilidades del presente servicio, pudiendo aportar técnicas de recuperación especial mediante la URL de la fuente de datos o canal sindicado. Esto se consigue gracias a las propiedades de estructuración y modulación en metadatos de los formatos de sindicación, pudiendo ser interpretados para la edición de consultas especiales mediante los campos de descripción con una sencilla sintaxis.

Alerta de novedades bibliográficas

"Mensaje que notifica una novedad bibliográfica. Dicho mensaje puede producirse por cualquier medio de comunicación – teléfono, carta, correo electrónico, etc. - , y puede ser ocasional o, más frecuentemente, realizarse de forma periódica, cada cierto tiempo, a modo de una suscripción." (LÓPEZ YEPES, J. (Coord.) and et.al., 2004, p.42)

Se puede inscribir dentro de los servicios de difusión selectiva de la información, no en vano, los sistemas de gestión de bibliotecas, pueden gestionar el envío de novedades bibliográficas en función de un sencillo perfil que se genera para aquellos usuarios interesados. También puede considerarse un servicio aparte dada la entidad e importancia que cobra al extrapolarse no sólo al ámbito de envío de correos mediante listas de distribución, sino en el más puramente informativo y de difusión general para todo un centro universitario o de investigación. En este sentido, la alerta de novedades bibliográficas se convierte en verdaderos boletines con los últimos documentos adquiridos por la biblioteca quedando registrados en formato impreso y electrónico. La sindicación tiene una aplicación directa, en este tipo de servicio, que se está llevando a cabo en algunas bibliotecas universitarias, como la correspondiente a la *Australian National University* (ANU, 2008), que ha configurado canales de sindicación para la difusión bibliográfica de las últimas novedades documentales incorporadas a los fondos, como se explicará en el capítulo 5 de la investigación. Una de las ventajas del empleo de canales específicos para alerta de novedades bibliográficas es la posibilidad de suscripción que la sindicación brinda al usuario pudiéndole remitir automáticamente sin necesidad de acceder al sitio web de origen todas las actualizaciones sobre la información publicada.

2.3.2. Servicio de consulta OPAC

"... es el instrumento que permite al público acceder consultar, de forma interactiva, los materiales que forman una colección a través de un Terminal de ordenador situado en la biblioteca o, utilizando las redes de de telecomunicación desde un Terminal remoto [...] en muchos casos se identifica con el catálogo ya que ofrece la información suministrada por el profesional en el proceso de la catalogación, [...] la disponibilidad de los ejemplares y el acceso al documento original." (PINTO MOLINA, M., 1994, p.510)

El servicio OPAC, consiste en la disposición del catálogo colectivo de la biblioteca que contiene las referencias bibliográficas a materiales documentales según sus correspondientes tipologías. Es un servicio de consulta libre para los usuarios, basado en un equipo conectado a la red de la biblioteca e Internet, que se alimenta de los contenidos del catálogo del sistema de gestión bibliotecario. Un servicio OPAC se caracteriza por la adopción de formularios de consulta que presentan los correspondientes campos catalográficos del registro de un documento. A su vez suelen incorporar diferentes operadores booleanos de proximidad, de

fecha, de restricción sintáctica e incluso de tipo de extensión y formato, para el refinamiento de las consultas a realizar. El resultado de tales consultas, siempre estará ordenado por criterios de ponderación en la pertinencia y exhaustividad con respecto a la cadena de consulta planteada inicialmente, generando listados de resultados, con descripciones someras que permiten la navegación sobre los mismos y un primer acercamiento sobre el objeto de conocimiento demandado. También deben proporcionar información más concreta, que permitan hacer efectivo el servicio en sí mismo, se trata de facilitar la ficha descriptiva completa en los diversos formatos de catalogación Marc, ISBD así como otros formatos para la exportación sencilla de la información. Finalmente y adjunto a esta información, han de informar sobre la localización exacta del documento, atendiendo al centro y dirección en la que se encuentran, planta, sección o departamento, estantería, balda y obligatoriamente signatura. De esta manera se puede concluir que el OPAC es el servicio de consulta bibliográfica del catálogo colectivo de un centro de información que permite consultar todas sus tipologías documentales mediante un sistema de recuperación basado en la filtración de los campos catalográficos con los principales puntos de acceso al documento y que devuelve ante las consultas efectuadas, la información catalográfica completa y localización de los ejemplares aproximados y demandados inicialmente. Se ha de añadir a todo lo explicado que los servicios de OPAC incluyen en muchos casos el servicio de reserva de libros que permiten a los usuarios de la biblioteca, hacer reservas con su código de usuario lector o referencia personal. Lo que representa la vinculación del servicio OPAC, con el préstamo domiciliario e incluso inter-bibliotecario, atendiendo a los reglamentos y convenios en las políticas de préstamo de materiales.

2.4. La recuperación de información bibliográfica Z39.50 y su integración en los OPACS

Los OPACS tienen una importante interrelación con los sistemas de gestión de bibliotecas y el sistema de recuperación Z39.50. Esto se debe a que los OPACS constituyen módulos del propio sistema de gestión de la biblioteca, para permitir el acceso público a los catálogos bibliográficos. De hecho "*Son sistemas de control bibliográfico que permiten leer y recuperar los datos almacenados en el ordenador por medio de varios puntos de acceso. Se trata de sistemas automatizados de acceso público que permiten la consulta y visualización de los*

registros que componen el fondo de una colección bibliotecaria o de varias de ellas" (GARCÍA LÓPEZ, G.L., 2007).

Como se viene explicando los objetivos del OPAC, consisten en facilitar la recuperación de un libro o un documento a partir del autor, título o materia, mostrar todos los contenidos que la biblioteca disponga sobre un tema y ayudar a la hora de elegir un libro. Para ello desarrollan una serie de estrategias de recuperación como la utilización de operadores booleanos, la recuperación mediante búsquedas en texto libre, la limitación de los resultados, la representación de los distintos puntos de acceso, entre otros, con el objetivo de permitir el préstamo de los documentos y facilitar la comunicación del usuario con el personal bibliotecario.

Si bien esto es así, existen limitaciones y dificultades que reducen la efectividad de los OPACS, como por ejemplo la disposición preferente para mostrar materiales disponibles en el propio centro, sin contar con las colecciones de terceros, la carencia de sistemas de ayuda para mejorar la experiencia del usuario, problemas de navegación entre pantallas, problemas para identificar las colecciones bibliográficas o dificultades para interpretar los registros bibliográficos, dado el empleo de esquemas de representación no normalizados o poco amigables para su lectura (MOYA ANEGÓN, F. and Fernández Molina, J.C., 1998).

Los componentes relativos al OPAC pueden dividirse en la interfaz de usuario que utilizar diversos formularios con las opciones de búsqueda y recuperación del catálogo. Ayuda al usuario a definir qué acciones desea llevar a cabo en el OPAC. Por otro lado el interfaz del sistema de gestión de la base de datos que de manera interna trabaja para la gestión y procesamiento de las consultas del usuario para recuperar la información, coordinación de distintos puntos de acceso, índices y ficheros. Otro componente sería el Sistema de gestión de bibliotecas propiamente dicho, encargado de la gestión integral del catálogo bibliográfico permitiendo añadir nuevos registros y crear sus índices, borrar registros, actualizar la información de los campos, recuperarlo y editarlos. Finalmente la base de datos es el componente que a modo de pilar soporta todo el sistema al ser la base sobre la que actúa el resto de componentes. Contiene físicamente el catálogo bibliográfico, índices para la recuperación y dispositivos de almacenamiento.

Algunas de las capacidades comprobadas de los últimos OPACS, consisten en el aumento de las funcionalidades hipertextuales para facilitar la navegación, la utilización de los operadores

booleanos de forma implícita en las búsquedas, mejoras de la accesibilidad por idioma, fecha de actualización de los registros bibliográficos, adyacencia o proximidad,

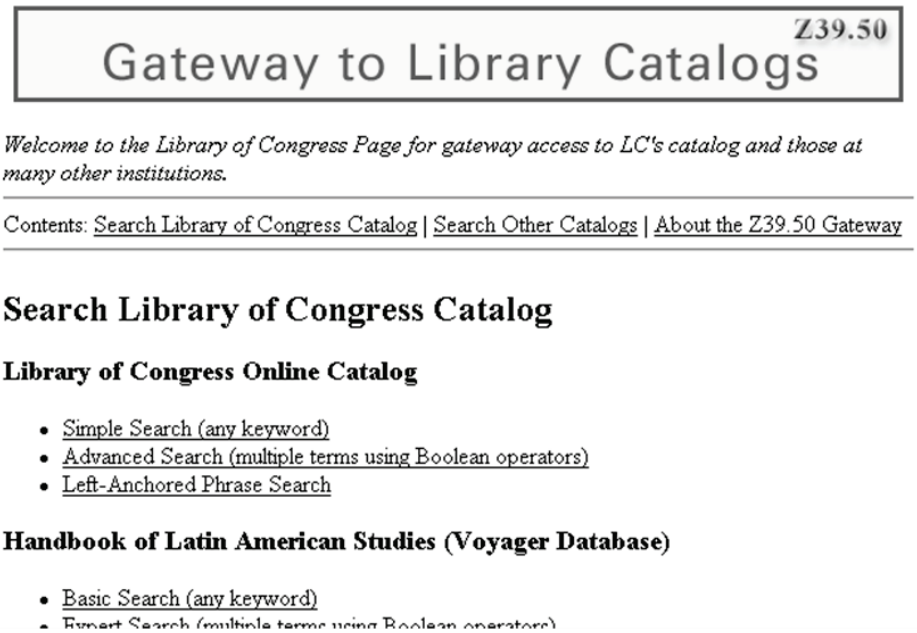
Son deseables mejoras en los apartados relativos a la consulta de listas de encabezamientos de materia, autoridades y sistemas de clasificación. El aspecto del interfaz gráfico de usuario y su experiencia en el acceso de las funciones del OPAC. Mejorar la implantación de los portales OPAC en la web, desarrollar mejores métodos de redifusión de los catálogos bibliográficos con técnicas hipertextuales entre otros aspectos. Conociendo estas problemáticas, en la presente investigación, se procura dar solución a cada uno de ellos con la aplicación de sindicación de los catálogos bibliográficos, favoreciendo su redifusión, y mejorando la experiencia del usuario, al poder comprobar que las autoridades también pueden ser visualizadas mediante estas técnicas, tal y como se explicará en el capítulo 7 de desarrollo de las aplicaciones de sindicación de contenidos para la gestión de catálogos bibliográficos.

Llegados a este punto, se concluye que existe una importante relación de los OPACS con los sistemas de gestión de bibliotecas, por constituir en sí mismos, módulos integrados, dependientes de las modificaciones que se efectúan en dicho sistema de gestión.

Pero no solamente se integran los OPACS en los sistemas de gestión de bibliotecas, sino también los sistemas de recuperación colectiva de registros bibliográficos que permiten los procesos de pre-catalogación. Se trata del protocolo Z39.50 (NISO; LOC, 2003), que forma parte activa de muchos OPACS, para recuperar a modo de catálogo colectivo unificado. De hecho según lo que se explica en las Jornadas Españolas de Documentación FESABID en 1998, se llega a afirmar en relación a los OPACS, que "*no hay dos sistemas cuyo interfaz tenga las mismas características y cada uno obliga al usuario a familiarizarse con una estructura distinta, además no suele ser posible pasar directamente los resultados obtenidos para emplearlos en otras aplicaciones, ni es fácil repetir la misma consulta en distintos servidores a través de Web y obtener resultados consistentes"*. Hecho que justifica la aplicación de Z39.50 en el OPAC como un elemento normalizador tanto de los accesos a las colecciones, como por los resultados que pueden conseguirse. Tal consideración es expresada de forma clara en el siguiente enunciado. "*Z39.50 es un estándar internacional, amplio, potente y muy difundido en el mundo bibliotecario entre otros. Presenta numerosas ventajas tanto para el usuario final como para el bibliotecario. Su implantación y desarrollo no va en detrimento del empleo del Web, sino que deben converger en pasarelas Web a servicios Z39.50"*. (LÓPEZ DE SOSOAGA TORIJA, A., 1998).

Otro ejemplo claro de la integración del protocolo Z39.50 en los OPACS, se encuentra en la red de catálogos de bibliotecas (Gateway to library catalogs Z39.50, 2009), que emplean el protocolo Z39.50 como sistema de recuperación principal de sus propias colecciones y del resto de catálogos que conforman dicha red.

Título: Guía de catálogos OPAC de bibliotecas con Z39.50
Fuente: http://www.loc.gov/z3950/
Referencia: figura1



Puede resumirse que el protocolo Z39.50, constituye una buena solución para la normalización de la recuperación de información en los catálogos bibliográficos, pero también es verdad que entraña un complejo sistema de comunicaciones para la transferencia de los resultados e incluso la formulación de las consultas. Esto se debe a que los orígenes del protocolo se remontan a 1988, sufriendo posteriormente diversas revisiones en 1990 por parte de grupos de trabajo anexos a la Library of Congress como el *Z39.50 Implementors Group* o el actual *Z39.50 Maintenance Agency* como consecuencia de la obsolescencia de los procedimientos utilizados fuera de los estándares web. De hecho no se logrará hasta el año 1997 alcanzar un grado de estandarización y compenetración con el protocolo TCP-IP que como bien es sabido rige el funcionamiento del modelo cliente servidor en la web. Tales

resultados fueron configurados en la norma ISO 23950 y posteriormente por la más actual ANSI/NISO Z39.50 del año 2003.

Una herencia de este desarrollo fue la creación de un lenguaje de comunicaciones interno entre las aplicaciones Cliente y Servidor de Z39.50, denominado Z-Speak. Dicho lenguaje pone en comunicación un programa cliente con un servidor especialmente configurado para traducir las consultas bibliográficas remitidas por el cliente. Esto implicaba que tales consultas fueran traducidas desde el cliente para utilizar el propio protocolo Z39.50, lo que supone la disposición de librerías especiales para su codificación. Algunas de las librerías y clientes más conocidos como YAZ (YAZ Index Data, 2009), JZKit (JZKit2: Z39.50 Server, 2009) o MercuryZ39.50 (Mercury Z39.50 Client, 2009), permiten implementar soluciones a nivel de usuario, que no siempre resulta fáciles de instalar, debido a que están diseñadas para funcionar bajo sistemas operativos Linux, a partir de terceros programas, o mediante entornos no visuales.

Dentro del apartado de especificaciones de Z39.50, contempla el empleo de operadores booleanos AND, OR y NOT, operadores de comparación, proximidad y truncamiento. También son características de Z39.50, la autenticación de los accesos al sistema y de las bases de datos, la conexión a bases de datos remotas, la interfaz de consulta a modo de OPAC, el empleo del formato MARC normalizado, la posibilidad de guardar registros para su posterior exportación, posibilidad de ordenación de los catálogos bibliográficos a partir de la fecha de actualización o modificación de los registros (DENENBERG, R. et al., 1996).

Tales características pondrán a prueba el desarrollo de las aplicaciones de sindicación para la gestión de catálogos bibliográficos, implicando en esta investigación un proceso de simplificación y renovación de los métodos que pueden emplearse para recuperar, ordenar e incluso editar los catálogos bibliográficos.

Pero Z39.50 también tiene aspectos mejorables como por ejemplo dificultades de integración del protocolo en las bibliotecas, debido a los requerimientos que plantea la instalación de dicho protocolo en los servidores, la carencia de soportes técnicos del protocolo que se emplea en terceros programas, incluso en los propios sistemas de gestión de bibliotecas o por ejemplo los problemas de colapso del servicio cuando múltiples usuarios tratan de acceder a la vez a los servicios del protocolo. Tales problemas han sido heredados precisamente por la aplicación de un modelo de protocolo que duplica el proceso de comunicación y transferencia tanto de las consultas como de los resultados. De hecho en esta investigación se demuestra

que la sindicación de contenidos mediante el empleo del protocolo HTTP resulta suficiente para efectuar todas estas operaciones. La mejor prueba de lo afirmado, es el hecho de que Z39.50 está tratando de implementar en mayor medida las tecnologías XML para la recuperación de registros bibliográficos como demuestra la más reciente modificación del sistema (Z39.50 Implementor Agreement: Requesting XML Records, 2009), por la que estaría capacitado para hacer peticiones de registros XML, sin embargo no para sindicarlos o efectuar una redifusión o un tratamiento como el que se demuestra en la presente investigación.

Más pruebas del interés de la comunidad científica por desarrollar las tecnologías XML para ser aplicadas en el entorno bibliográfico y de la recuperación de información en catálogos, puede observarse en recientes publicaciones de los autores Surla Dusan y Danijela Boberic, en el trabajo, *XML Schema of query language defined by Z39.50 standard* (BOBERIC, D. and Surla, D., 2007), en el cual tratan de desarrollar un archivo XSD schema para la elaboración de un formato de consulta basado en los mecanismos de recuperación de Z39.50. De esta forma tratan de representar las consultas a partir de Argumentos y Operadores, que combinados dan como resultado una serie de tipologías o rutinas de consulta predefinidas en dicho esquema. Éstas permitirían utilizar atributos para definir los puntos de acceso de las búsquedas por título, autor materia. Pero también el empleo de atributos de truncamiento, posición o estructura de una frase o palabra. No obstante, las teorías expuestas no logran llevarse a la práctica, quedando como un marco teórico para futuros desarrollos.

Posteriormente se elabora otro trabajo también participado por Surla Dusan, titulado *Bibliographic records editor in XML native environment* (SKRBIC, S. and Surla, D., 2008), en el que se trata de abordar la adaptabilidad de los registros bibliográficos en formatos XML nativos, concretamente tratando de adaptar UNIMARC a XML y modelar la introducción de las descripciones bibliográficas. Este paso si fue desarrollado en la práctica, traduciéndose en una aplicación basada en JAVA y en TAMINO, una base de datos nativa desarrollada específicamente para la ocasión.

Finalmente se trato de dar un paso más allá con el artículo *XML editor for search and retrieval of bibliographic records in the Z39.50 standard* (BOBERIC, D. and Surla, D., 2009), recientemente publicado, en el que se aborda el desarrollo de un editor XML para la recuperación de registros bibliográficos en Z39.50, similar a la última aportación efectuada a la Library of Congress anteriormente citada.

Todo ello viene a determinar que las tendencias de desarrollo de las tecnologías basadas en XML, como la sindicación de contenidos, pueden jugar un papel importante, en los procesos de recuperación y transmisión de información bibliográfica de los catálogos bibliotecarios. En los trabajos anteriormente citados estos desarrollos tienen como objetivo, su implementación en el protocolo Z39.50. En la presente investigación se aborda un enfoque nuevo que predispone un modelo integral basado exclusivamente en el protocolo HTTP, sin requerimientos especiales de bases de datos nativas o complejos esquemas de traducción de las consultas.

Capítulo 3. Bases tecnológicas de la sindicación de contenidos

La aplicación de la sindicación de contenidos en los servicios bibliotecarios conlleva el conocimiento de sus bases tecnológicas e infraestructura. Tal infraestructura consiste en la familia de lenguajes XML que permiten confeccionar los formatos de sindicación, pero también su representación, recuperación y filtrado. En el presente capítulo se explican las funciones y características de los lenguajes de marcado basados en XML para determinar finalmente que sus propiedades y características son las mismas que las dispuestas por los formatos de sindicación para el procesamiento y transmisión de información.

3.1. La familia de lenguajes XML

La sindicación de contenidos está ligada al lenguaje de marcado XML. Pero su origen procede de SGML *Standard General Markup Language*, lenguaje madre del cual se derivan todos los lenguajes de marcado que actualmente se emplean en la red. SGML fue creado durante la década de 1980 a 1990, a partir de las investigaciones que IBM *Internacional Business Machines* venía desarrollando para conseguir un lenguaje capaz de estructurar y formatear contenidos. El resultado de dicho esfuerzo se convirtió en un estándar reconocido por la organización internacional de normalización y estandarización ISO en 1986 (Gentle Introduction to SGML, 1994).

A partir de SGML, se fraguó una importante familia de lenguajes de marcado derivados, de los cuales *John Berners-Lee* fue el artífice principal. Se trata del actual *HTML Hypertext Markup Language* (BERNERS-LEE, T. and Fischetti, M., 2000), empleado en la por aquel entonces incipiente red Internet que permitía la visualización de información en los primeros visores y posteriormente navegadores web, y XML *Extensible Markup Language*, pensado para el procesamiento y tratamiento de información de manera automatizada, para la que HTML no estaba concebido. Téngase en cuenta que XML permite compartir información entre dispositivos y resulta estándar en cualquier navegador, editor y sistema que se utilice, dada su sencillez.

En 1996 HTML queda estandarizado y dos años después en 1998 será XML gracias al *consorcio W3C* [7] creado como respuesta a los problemas de programación y codificación de ambos lenguajes y que el propio Berners-Lee puso en funcionamiento desde el *Instituto tecnológico de Massachussets MIT*[8].

XML es el acrónimo de *Extensible Markup Language* es decir, lenguaje de marcado ampliable o extensible, que constituye por sí sólo uno de los pilares de la web. Téngase en cuenta que una proporción importante de toda la información representada en la red está recogida utilizando las bases y fundamentos de este lenguaje. Tal aseveración tiene su explicación y repercusiones en el continuo desarrollo que está sufriendo XML, desde su adopción por el W3C para convertirse en estándar de la construcción web a partir del 10 de febrero de 1998. Desde entonces hasta la fecha, XML ha generado una completa familia de lenguajes cuya principal misión es servir de soporte para representar la información, siendo descrita y organizada conforme a unas normas de estructuración, facilitando la recuperación de la misma mediante métodos de enrutamiento y de filtración y como aspecto fundamental de cualquier recurso o fuente de información, su transmisión y difusión. Entendiendo que la finalidad de XML es la mejora de las posibilidades de tratamiento o manipulación de la información y documentación electrónica, se hace necesario establecer un mapa o diagrama que permita aclarar un patrón de estudio sobre sus principales elementos y lenguajes derivados. Se debe recordar que aún no existiendo una clasificación que organice las relaciones funciones y dependencias de cada uno de los lenguajes, se pueden comprobar que existen las siguientes:

[7] **Consorcio W3C:** Consorcio del World Wide Web. Es el organismo internacional encargado de desarrollar tecnologías inter-operativas (especificaciones, líneas maestras, software y herramientas) para guiar la red a su potencialidad máxima a modo de foro de información, comercio, comunicación y conocimiento colectivo. *Véase* http://www.w3.org/

[8] **MIT:** Massachusetts Institute of Technology o Instituto tecnológico de Massachusetts, constituye uno de los centros de investigación y desarrollo más importantes del mundo. Una de sus especializaciones más reseñables es la investigación de sistemas de información, computación del web y recursos para la investigación, entre otros.

Tipología funcional de la familia de lenguajes XML

1. **Matriz XML:** Lenguaje matriz o madre, en el que se basan todos sus complementos incluyendo los lenguajes de sindicación de contenidos.

2. **Transformación de documentos XML:** Aquellos lenguajes destinados a generar un formato que afecta a la presentación y visualización de un documento XML que contiene información. En la transformación de los documentos, no hay que olvidar que se suceden breves funciones de recuperación de información, gestión de contenidos que permiten como el propio nombre indica, transformar el documento original en un documento secundario. Estos lenguajes son: XSL, XSLT y XSL-FO.

3. **Descripción y validación de documentos XML:** Contiene aquellos lenguajes empleados para generar una descripción del tipo de documento, su estructura e incluso el tipo de información contenida. Estos esquemas programados permiten validar el documento XML, considerándose en consecuencia un formato construido de acuerdo con las reglas de normalización expresadas en el W3C (World Wide Web Consortium). Estos lenguajes son: DTD y XSD Schema.

4. **Enrutamiento y recuperación de información:** Todos aquellos lenguajes basados en XML que permiten identificar las rutas de los elementos contenidos en un documento XML, ya sea apuntando a una referencia, o bien a un identificador. También se incluyen los lenguajes de recuperación basados en XML, ya que aprovechan todas las posibilidades de referencia de los primeros para obtener una respuesta con resultados de cada consulta. Estos lenguajes son: XPath, XLink, XPointer y XQuery.

5. **Manipulación de documentos XML:** Implica la base de programación en XML, extendiendo las posibilidades de tratamiento y manipulación a otras aplicaciones, aprovechando de esa forma las descripciones del tipo de documento y de los lenguajes de enrutamiento. Estos lenguajes son: XML-DOM y derivados.

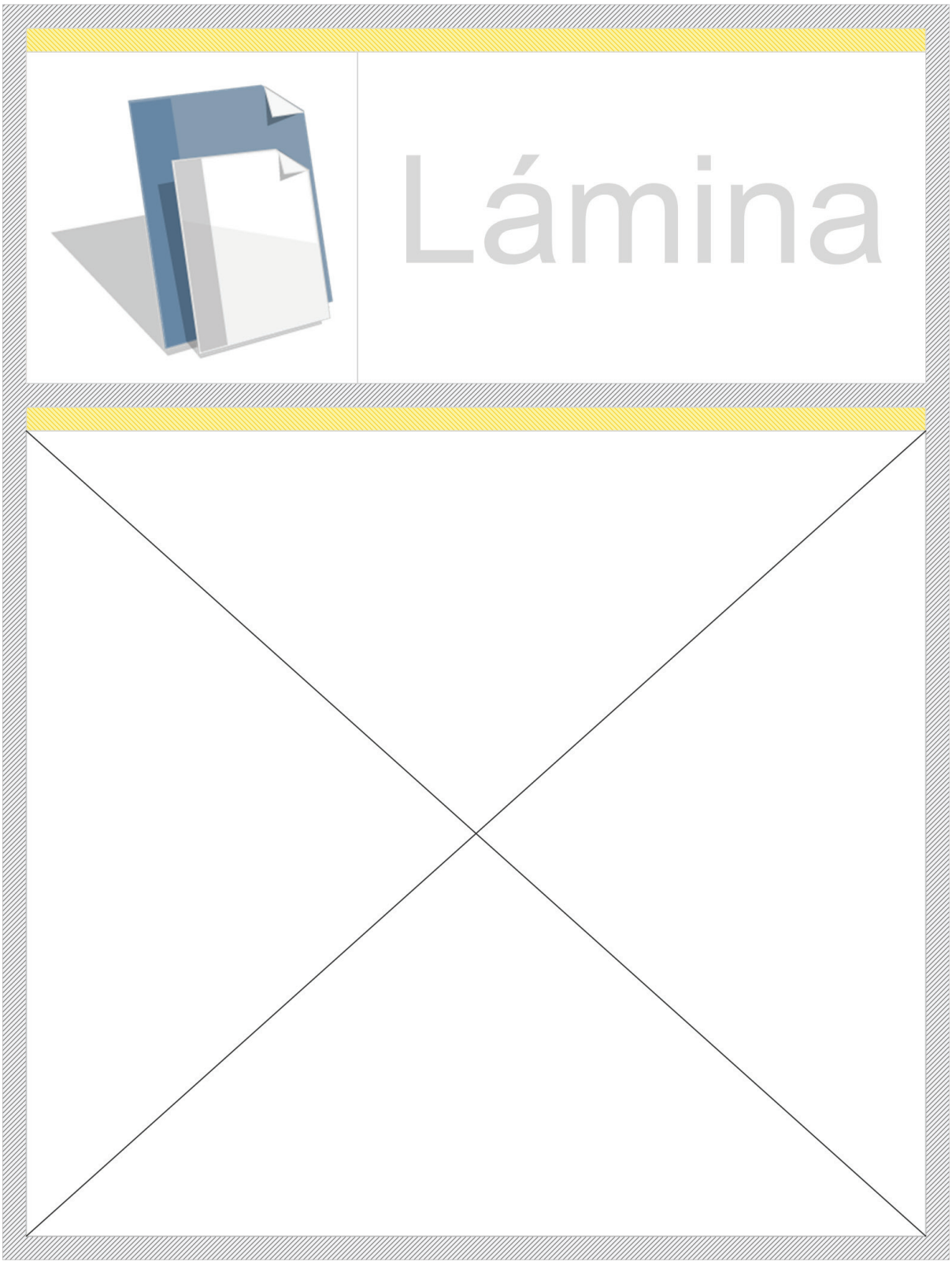
6. **Protocolos de comunicación y servicios:** Aquellos lenguajes dedicados a la transmisión de información y a la descripción de servicios telemáticos que empleen protocolos en la web para su funcionamiento. Estos lenguajes son: XML-RPC, SOAP y WSDL.

7. **Lenguajes de descripción "ad-hoc":** Son aquellos que han sido adoptados como estándares web por el W3C (World Wide Web Consortium) y que sirven para describir tipos de documentos muy específicos en entornos muy concretos. Estos lenguajes son: SMIL, SVG y WAP.

8. **Desarrollo de formularios:** Lenguajes basados en XML para la descripción y desarrollo de formularios de cara a su implantación en la web, siendo su principal finalidad, la entrada de datos en un entorno previamente estructurado. Estos lenguajes son: XForms y derivados.

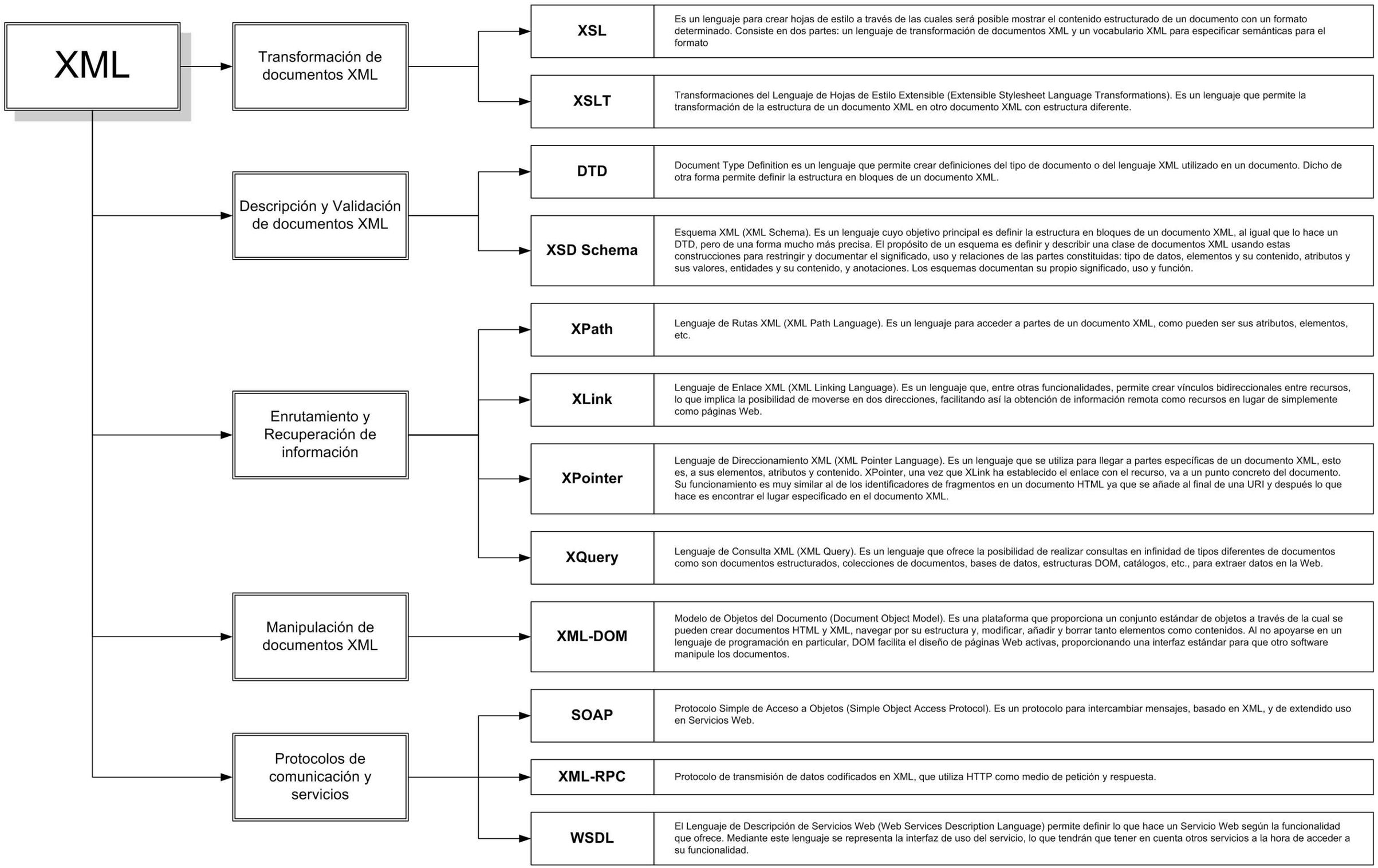
En esta tipología se advierten funciones específicas para cada lenguaje, lo cual no ha de llevar al error de pensar que su funcionamiento es independiente. Todos los lenguajes, y en especial aquellos destinados al enrutamiento y recuperación de información, protocolos de comunicación y servicios, así como descripción, validación de documentos y transformación están íntimamente relacionados. Se explicará y comprobará más adelante que la sintaxis definida para unos lenguajes también es válida para otros y viceversa; que a efectos de posibles aplicaciones para la documentación y su uso en servicios bibliotecarios, son muy extensas y ampliables. Dicha tipología puede ser consultada en el esquema gráfico de la familia de lenguajes XML, *véase la figura2*

Título: Tipología funcional de la familia de lenguajes XML
Fuente: http://www.w3c.es/Divulgacion/a-z/
Referencia: figura2



Tipología funcional de la familia de lenguajes XML

Referencia: figura2



Fuente: *El W3C de la A a la Z. (World Wide Web. Oficina Española). Disponible en: http://www.w3c.es/divulgacion/a-z/*

Teniendo en cuenta la disposición y tipología funcional de los lenguajes de marcado extensibles, se ha determinado pertinente para el presente estudio abordar los siguientes: XML, XSLT, DTD, XSD Schema, XPath, XLink y XPointer, por ser esenciales para el desarrollo de la sindicación de contenidos.

3.2. XML

Tal y como lo define el W3C (World Wide Web Consortium), XML es: *"...Lenguaje de Etiquetado Extensible (eXtensible Markup Language). Es un lenguaje con una importante función en el proceso de intercambio, estructuración y envío de datos en la Web. Describe los datos de tal manera que es posible estructurarlos utilizando para ello etiquetas, como lo hace HTML, pero que no están predefinidas, delimitando de esta manera los datos, a la vez que favoreciendo la interoperabilidad de los mismos…"* (W3C de la A a la Z, 2009)

Para llevar a cabo el estudio del apartado de XML, tanto en sus aspectos formales como técnicos, se han tenido en cuenta fuentes de información especializadas. Éstas proceden del fundamentalmente de las especificaciones oficiales del W3C Consortium (Extensible Markup Language (XML) 1.0 (Fifth Edition), 2008) y en menor medida del W3Schools (XML Tutorial, W3Schools, 2009). Se han seleccionado éstas fuentes, por ser oficiales y de referencia en el marco internacional de edición y programación web. Por otro lado pueden facilitar aún más si cabe la introducción y profundización en la investigación del objeto de estudio.

Ciertamente XML está diseñado para el soporte y descripción de contenidos e información de forma estructurada, mediante una serie de esquemas de elementos representados mediante etiquetas o marcas para ser estructurados mediante anidamientos que generan estructuras y relaciones de tipo jerárquico entre ellos. El paralelismo más sencillo se puede establecer respecto a los lenguajes documentales de tipo controlado como los tesauros, cuyas representaciones permiten un orden jerárquico lógico y semántico de sus términos. En XML es posible seguir el mismo modelo, con la diferencia de que lo que se establecen son estructuras, etiquetas, marcas o campos que describen la información que contendrán. Por ese motivo se explica que XML tiene una función fundamental a la hora de estructurar datos y por extensión la información.

El lenguaje XML no utiliza un modelo de etiquetas predefinidas, lo que permite al documentalista crear aquellas que necesite para describir un determinado documento. Esta propiedad es especialmente útil y versátil ya que posibilita desarrollar formatos especializados, para mejorar los servicios bibliotecarios en particular.

En XML se requiere siempre de DTD o XSD Schema adjunto que permita la descripción del tipo de contenidos en las etiquetas utilizadas. Como se ha explicado anteriormente, tanto las

DTD como los XSD Schema son lenguajes especializados en la descripción y validación de los documentos. Concretamente aportan una estructura válida que el documento XML, que contiene la información, ha de seguir y mantener para ser válido y ser considerado por ende un formato de descripción.

En relación a la validación mencionada, el resultado de la construcción de documentos XML, permite definirlos como *bien formados*[9] y *validados*[10]. Son bien formados cuando presentan una sintaxis correcta conforme a las normas de construcción y edición; son validados cuando existe un documento DTD o Schema que conforme la estructura del modelo XML y determine el tipo de datos que contiene en cada etiqueta, definiendo de esta manera la organización de las mismas y sus características, aportando cuáles son los elementos permitidos.

Se emplea XML para la importación y exportación de contenidos, incluso entre sistemas incompatibles, dada la simplicidad y definición de su estructura y elementos expresados mediante etiquetas (Más adelante se comprenderá este asunto, al explicar el funcionamiento del protocolo de transmisión de datos basado en XML, SOAP). Estas propiedades de compatibilidad permiten a la postre facilitar la lectura, tratamiento, modificación, gestión y visualización de los contenidos en bases de datos distintas y diferentes aplicaciones.

Siendo XML un estándar independiente del hardware que una máquina determinada pueda utilizar (ya que la interpretación de XML es la misma en cualquier sistema operativo) es posible incluir o agregar recursos de información comunes para ser utilizadas por diversos usuarios a la vez, incluso mediante diferentes programas de tratamiento y edición siendo considerados éstos como *agentes*[11] y los recursos de información como *fuentes de datos*[12].

---

[9] **Documentos XML bien formados:** Denominados Well Formed XML cumplen las características de construcción principales. Es decir, apertura y cierre de etiquetas correcto, empleo de caracteres aceptados, correcto anidamiento o raíz, correcta citación de los atributos de las etiquetas utilizadas, son establecidas en W3C como las principales.

[10] **Documentos XML validados:** Denominados Valid XML son aquellos que siendo bien formados, están construidos no sólo conforme a una sintaxis correcta sino a unas normas de de estructuración y definición de los contenidos de sus etiquetas. Por regla general esto se lleva a cabo mediante archivos DTD y Schema XSD.

[11] **Agentes:** Se entiende por agentes aquellos usuarios, sistemas, programas o aplicaciones que, aprovechando las fuentes de datos XML, realizan interacciones directas como su transformación, edición, visualización o representación.

Otra característica inherente a XML es que la información contenida en documentos confeccionados con su codificación normalizada puede ser visualizada y transformada mediante las técnicas de formateado con *CSS* [13] o *XSL* [14] , permitiendo una presentación más adaptada y accesible para los usuarios. Esta característica enlaza con los lenguajes de transformación de documentos que se explicaban en la tipología funcional de la familia XML.

Como se venía apuntando, XML también se emplea frecuentemente en el desarrollo y creación de nuevos lenguajes especializados ad-hoc, como SMIL, SVG o WAP. Inscritos pero aún no presentes en este cuadro se incluirían los lenguajes de sindicación de contenidos. Los principales son RSS, RDF, ATOM-XML, OAI y OWL que serán debidamente tratados en sucesivos capítulos, una vez sean conocidas las propiedades de XML y su familia de lenguajes más directos, ya que como se reitera en diversas ocasiones, los lenguajes de sindicación heredan todas las propiedades, características y requisitos que cualquier formato desarrollado en XML.

---

[12] **Fuentes de Datos:** Una de las características principales de XML, es su empleo como fuente de datos. Esto difiere en poco a la terminología acuñada en sindicación para delimitar los canales y medios de difusión, Feeds entendido como alimentación, o fuente de la que se alimentan los usuarios que recopilan su referencia. Así pues, se observa que los formatos de sindicación son la evolución directa y dirigida de XML para servir como canales de transmisión y difusión de datos e información, considerándose, ciertamente, una especialización del mismo.

[13] **CSS (Cascading Style Sheets):** Hojas de estilo en cascada, son empleadas de forma extensiva tanto en documentos XML, como en páginas web desarrolladas en HTML, PHP e incluso ASP. No están sujetas en su aplicación a un lenguaje de marcado o programación concreto, siendo de esta manera polivalentes, a diferencia de XSL. Su funcionamiento se basa en clases, grupos y capas de estilo que se aplican directamente en el documento previa referencia del elemento designado para su transformación. (No están basadas en XML)

[14] **XSL (Extensible Stylesheet Language):** Lenguaje extensible de hojas de estilo, empleado de forma específica en el formateado, representación y visualización de documentos XML en los principales navegadores web. A partir del documento XML, se aplica estilo para su transformación en forma, estilo y datos, representado en un documento HTML de salida.

XML puede ser empleado para generar *islas de datos*[15] en las páginas web independientemente del lenguaje con que estén programadas (HTML, PHP o ASP); para ello se incluye la referencia y el identificador del contenido que se desea introducir.

La transformación de un documento XML, así como su procesamiento, requiere de un *sistema parser*[16], que es un programa que varía entre unas breves líneas de código y extensas instrucciones que permiten la lectura y tratamiento del documento XML destinado a un entorno web. La tecnología de principal uso en este aspecto es *DOM*[17] , común a todos los navegadores; aunque este inicial punto en común tiene diferentes interpretaciones dependiendo de la plataforma de navegación. Por ejemplo, Internet Explorer emplea códigos Javascript, VBSscript y ASP, difiriendo de otros navegadores basados en Mozilla por la manera de llamar o cargar el documento XML, ya que emplean exclusivamente Javascript.

XML permite introducir scripts y otras líneas de código programadas fuera de las reglas de un DTD, Schema o DOM, mediante el empleo de notaciones específicas, *<![CDATA[...]]>* que evitan que un sistema parser evalúe dichos scripts o líneas de código, permitiendo a su vez la ejecución de los mismos durante la visualización o representación.

Como se ha descrito en estos párrafos, XML actúa como plataforma fundamental para la transferencia y estructuración de la información en la web. Para ello cuenta con importantes herramientas y lenguajes derivados que permiten su transformación, cambio de formato, descripción, modularidad y empleo como repositorio de alta disponibilidad.

---

[15] **Islas de datos:** Originalmente denominadas *Data Island* describen correctamente la finalidad de implantar información estructurada de un documento XML, precisamente también por ello pueden ser denominados injertos de información, no en vano supone la modificación del código fuente original del documento y la presentación de información en un fragmento de la página, procedente de otro documento.

[16] **Sistema parser:** Parser es un término anglosajón que significa Programa de Análisis y cuya principal finalidad en este contexto es la carga del documento XML, el análisis de la estructura de datos y su representación para facilitar su transformación o lectura en un navegador web o terceros programas o sistemas de bases de datos.

[17] **DOM:** Document Object Model, establece las normas para acceder y manipular documentos XML, de ahí su denominación Modelo de Objetos del documento. Para ello DOM presenta el documento como una estructura arborescente, constituida por nodos, que distribuyen los correspondientes elementos y atributos. Así de esta manera se determina una relación entre los elementos designados en las etiquetas del documento XML, como padres e hijos a modo de jerarquía.

Por ello conviene analizar algunos ejemplos en los que se confirman las propiedades anteriormente citadas.

Pruebas de las propiedades de XML como lenguaje de marcado

A continuación se incluyen diversos ejemplos que ilustran algunas de las características anteriormente citadas, concretamente el esquema de un documento XML, la vinculación con una hoja de estilo para su formateado, el empleo de islas de datos o injertos de información y el uso de parsers para la transformación, procesamiento y representación en entornos web.

Ejemplo de documento XML

En este ejemplo se muestran algunas de las propiedades de los documentos XML como el uso de etiquetas definidas por el usuario, para describir un artículo de divulgación científica. Se puede observar el uso de una hoja de estilo CSS al utilizar el código *<?xml-stylesheet type="text/css" href="kask.css"?>* que, referencia al archivo "kask.css" que a su vez presenta los estilos correspondientes para formatear la información contenida en los campos, como muestran las diversas anotaciones a lo largo del documento. Otra característica incluida en el ejemplo es la propiedad de anidamiento y jerarquía de la etiqueta *item*, conteniendo hasta tres niveles de profundidad en el campo de fecha de publicación.

Título: Ejemplo de documento XML
Archivo: articulo.xml
Referencia: tabla1

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="kask.css"?>

<item>
        <!-- Campo de Título -->
        <title>Entrevista a Juan Ignacio Cirac</title>

        <!-- Campo de Autor -->
        <author>Vicente María Desantes Fernández</author>

        <!-- Campo de Edición -->
        <edition>1ªed.</edition>

        <!-- Campo de Editor -->
        <publisher>El Documentalista Audaz</publisher>

        <!-- Campo de Fecha de Publicación -->
        <date>
                <year>2007</year>
                <month>10</month>
                <day>19</day>
        </date>

        <!-- Campo de Contenido -->
        <content>
        Calificar de mente privilegiada al Premio Príncipe de Asturias de Investigación Científica
        y Técnica 2006 es con toda seguridad una afirmación gratuita...
        </content>

        <!-- Campo de Enlace -->
        <permalink>
        http://documentalista-audaz.blogspot.com/entrevista-juan-ignacio-cirac.html
        </permalink>

</item>
```

Ejemplo de hoja de estilo aplicada al documento XML

En el ejemplo de la *tabla1* se utiliza el lenguaje CSS para la transformación del formato y representación de la información en el documento XML; pero se ha de tener presente que CSS no es un lenguaje de marcado, y que existen otros lenguajes como XSLT, basados en XML, que son capaces de realizar las mismas funciones. La diferencia básica entre unos y otros es que CSS está específicamente diseñado para aplicar estilos y XSLT, aplica estilos de manera más limitada pero permite transformar los resultados de datos en las representaciones, evitando cualquier posible problema de incompatibilidad de funciones.

Título: Ejemplo de hoja de estilo aplicada al documento XML
Archivo: kask.css
Referencia: tabla2

```
item {background-color: #FFFFFF;
      width: 100%;
      font-family: arial;
      font-size: 12pt;
      font-weight: lighter;
      color: #000000;
      padding: 10px;}

title, author, edition, publisher {display: block; margin: 10px;}

date {display: block; margin: 10px;}

year, month, day {font-weight: bold;}

content {display: block; margin: 10px;}

permalink {display: block; margin: 20px;}
```

Complementando al archivo ***articulo.xml***, el presente archivo de estilo ***kask.css*** formatea en los términos explicitados, la presentación de la información, dotando a la información de salida un determinado tipo de fuente, dimensiones, espacio, ubicación, entre otros aspectos.

Ejemplo de isla de datos XML

Este ejemplo demuestra que es posible realizar islas de datos en documentos HTML o similares, referenciando el documento XML, que resulta ser la fuente de datos, y estableciendo una identificación para la carga de los datos en aquellos elementos o bloques donde sea necesario. En este caso, se realizan sendas llamadas al elemento título y autor del documento *articulo.xml*. Al estar inscrito en una tabla correctamente definida con celdas extensible, en el momento de la representación en navegador, se generará un listado con todos los títulos y autores correspondientes de todos artículos incluidos en la fuente de información XML.

Título: Ejemplo de isla de datos XML
Archivo: isla-datos.css
Referencia: tabla3

```
<html>
<head><title>Ejemplo de isla de datos</title></head>
<body>

<!-- Código de instanciación del archivo XML al documento HTML-->
<xml id="item" src="articulo.xml"></xml>

<!-- Representación de los contenidos del archivo XML definiendo el origen de datos en los
     elementos item -->
<table border="1" datasrc="#item">
    <tr>
        <!-- Selección de los subelementos title y author de la etiqueta item, para mostrar el título y el
             autor -->
        <td><div datafld="title"></div></td>
        <td><div datafld="author"></div></td>
    </tr>
</table>

</body>
</html>
```

Hasta este punto los ejemplos para la demostración de las propiedades de XML, se han fundamentado en los principios de la estructuración de los datos, su representación básica y su implantación en otros documentos. A continuación se expone el ejemplo de transformación mediante parser, que supone un salto directo a uno de los principales usos y empleos de la sindicación, precisamente, la difusión de la información y su representación mediante *feeds*[18]. Esto tiene que alertar de que hasta la fecha las principales utilidades se están fundamentando en la capacidad de transferir y difundir información. Así pues el presente ejemplo pretende demostrar esta realidad, y para ello se ha generado un código en PHP, que permite la captación de una fuente de datos o documento XML, para ser representada de manera muy primaria en cualquier navegador web; probando los principios de tratamiento, transferencia y representación automática de la información basada en documentos XML, siendo éste el germen de la sindicación.

[18] **Feeds:** del inglés alimentar o dar de comer. Su equivalente en español es variado pudiendo denominarse como fuentes de datos, fuentes de información, fuentes de sindicación, canales, canales de información o canales de sindicación. En todos estos casos, se refiere a la dirección URL completa donde se aloja el documento XML que contiene la información objeto de sindicación.

Ejemplo de parser de datos XML

En este ejemplo, íntegramente desarrollado en PHP (BUYTAERT, D. and Wittens, S., 2007), se puede explicar la sencillez con que puede desarrollarse un programa parser para representar información en la web. El funcionamiento se basa en la carga de la fuente de datos XML, la declaración de las variables para distinguir los contenidos de las etiquetas empleadas en los elementos de tipo de datos de XML y los correspondientes ecos para imprimir los datos en pantalla. También incluye una breve instrucción que determina una condición para los casos de error en el tratamiento de los datos, que obtiene la línea en la que se produce el fallo.

Título: Ejemplo de parser de datos XML
Archivo: parser.php
Referencia: tabla4

```
// La variable $file toma el valor del nombre del archivo XML que analizará el programa parser
<?php    $file = "articulo.xml";

// Función para la impresión de los datos resultantes del análisis
function contenidos($parser, $datos) { echo $datos; }

// Función para imprimir las etiquetas de cada apartado en negrita
function inicioEtiqueta($parser, $datos) { echo "<b>"; }

// Función para cerrar las etiquetas de negrita de cada apartado de datos
function finalEtiqueta($parser, $datos){ echo "</b><br />"; }

// Función para crear un proceso de análisis XML. Esto es posible a las librerías PHP Expat.
$xml_parser = xml_parser_create();

// Función para controlar la selección de elementos de la estructura del archivo XML
xml_set_element ($xml_parser, "inicioEtiqueta", " finalEtiqueta");

// Función para controlar la selección de contenidos de los elementos del archivo XML
xml_set_content($xml_parser, "contenidos");

// Variable que incluye el control de apertura del archivo XML para su lectura
$fp = fopen($file, "r");

// Variable datos en la que se vuelca todo el contenido leído en el archivo XML
$datos = fread($fp, 80000);

// Programa parser propiamente dicho, en el que se controla la ejecución del parser hasta finalizar
// con el final de línea del archivo XML sujeto de análisis. El proceso finaliza en caso de error de
// lectura.
if(!(xml_parse($xml_parser, $datos, feof($fp)))){  die("Error en la línea " .
xml_obtiene_linea($xml_parser)); }

// Función para cerrar el programa parser
xml_close_parser($xml_parser);
fclose($fp);     ?>
```

La sindicación como técnica emplea formatos específicamente diseñados en XML. Para que un contenido pueda ser sindicado es necesario que esté correctamente estructurado, bien formado y posteriormente validado por un documento de descripción del tipo de contenido para determinar el modelo o esquema de estructura aceptada. Estos requisitos son cumplidos por los actuales estándares de sindicación, a saber RSS, RDF, ATOM-XML, OWL[19] y de manera experimental OAI[20]. Esta propiedad es generalizada para todo lenguaje que se desee desarrollar con una sintaxis diferente a la establecida, dando lugar, a nuevos formatos de descripción.

En el caso de la archivística, las posibilidades para desarrollar lenguajes de descripción avanzados, adaptados a las necesidades de control y gestión de fondos, secciones y archivos, así como a las normas de descripción ISAD-G e ISAAR-CPF, dieron como resultado la creación del formato de descripción archivística EAD.

En bibliotecas, el empleo de XML, aún no teniendo un principio de empleo masivo como formato de descripción bibliográfica en sistemas de gestión bibliotecarios, sí es posible gracias al empleo de los formatos *Marc-XML* (MARC-XML Schema, 2009) promovidos por la Library of Congress. Estos formatos, aun no siendo de sindicación, al estar preparados para la estructuración de la información bibliográfica en bases de datos, posiblemente pudieran modificarse en su estructura de *schema-XSD* (KEITH, C., 2002) y albergar opciones de sindicación, lo cual permitiría aplicar la sindicación al formato MARC y hacer extensivo el empleo de la técnica para la transmisión de los datos, dándose un importante debate en relación a considerar Z.39-50 y la sindicación de contenidos como dos opciones para el intercambio de registros bibliográficos. En el portal especializado de Marc-XML aparecen otras informaciones de importancia como el empleo de módulos de estilo para las diferentes representaciones de datos, según tipo de descripción extendida. Pero una de las principales características advertidas en este plano es la *aplicación de metadatos para la descripción bibliográfica, MODS* (Metadata Object Description Schema: MODS, 2009).

[19] **OWL:** Siglas de Web Ontology Language. Formato XML para desarrollar ontologías para definir los términos a utilizar y describir en un área de conocimiento.

[20] **OAI:** Siglas de Open Archives Initiative. Formato XML utilizado para especificar el protocolo de extracción de metadatos basados en Dublin Core. También existen especificaciones especializadas en la descripción y publicación de artículos científicos y académicos.

Se debe recordar, llegado a este punto que en el caso de aplicación de la sindicación de contenidos a un portal de revistas *Temaria* se empleaba esta técnica de metadatos para la descripción de publicaciones en revistas del área de la biblioteconomía y documentación. Esto quiere decir que existe un empleo secundado de los metadatos en la catalogación bibliográfica XML, concretamente en formato MARC.

Queda demostrado, en consecuencia que, XML supone una estructura principal a la hora de desarrollar formatos de sindicación y descripción bibliográfica, existiendo posibilidades de ampliación y combinación con otros métodos basados en las iniciativas de Dublin Core.

3.3. XSLT

XSL es un lenguaje que está compuesto de diferentes partes en su programación; por un lado XSLT se utiliza para la transformación de documentos XML, XPath para establecer los elementos de navegación, y XSL-FO, que permite dar formato a los mismos. Además pueden incluirse otros lenguajes como XQuery, XLink y XPointer que se emplean para el enrutamiento y recuperación de los datos almacenados y estructurados en los documentos XML. Por lo que el funcionamiento de todos los lenguajes de marcado extensibles está coordinado y relacionado de manera muy precisa aunque la definición que aporta el W3C (World Wide Web Consortium) sobre XSL no es tan explícita:

"…eXtensible Stylesheet Language es un lenguaje para crear hojas de estilo a través de las cuales será posible mostrar el contenido estructurado de un documento con un formato determinado. Consiste en dos partes: un lenguaje de transformación de documentos XML y un vocabulario XML para especificar semánticas para el formato (objetos de formato)..." (W3C de la A a la Z, 2009)

Para estudiar XSLT, ha sido necesario el empleo de dos fuentes electrónicas fundamentales, por un lado las especificaciones del propias del W3C Consortium (XSL Transformations XSLT Version 2.0, 2007) y d*ocumentación de referencia* del W3School (XSLT Tutoria, W3Schools, 2009) para abordar los aspectos más prácticos.

Siendo XSLT un lenguaje que permite la transformación de los documentos XML en documentos XHTML, o en otros documentos XML, con diferentes contenidos. De hecho es posible elegir qué elementos se desean agregar y eliminar en las visualizaciones o presentaciones de información, evitando que la representación de la información de un documento XML, sea un árbol jerárquico por defecto, tal y como se puede comprobar en cualquier navegador con el documento XML por defecto.

Construcción de un documento XSLT

El elemento *<xsl:stylesheet></xsl:stylesheet>* es utilizado para construir o generar plantillas XSLT aplicadas a documentos XML. Afecta a todos aquellos elementos definidos entre *<xsl:template></xsl:template>*. *Véase tabla5* y *7.*

Título: Declaración de un documento XSLT
Archivo: declaracion-xslt.txt
Referencia: tabla5

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

Para conocer el funcionamiento y construcción de los documentos XSLT, se toma un documento XML, para transformar. *Véase tabla6.*

Título: Ejemplo de catálogo XML afectado por un documento XSLT
Archivo: libros.xml
Referencia: tabla6

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="catalogo.xsl"?>

<biblioteca>
    <monografia>
        <titulo>El conde Lucanor</titulo>
        <autor>Juan Manuel, Infante de Castilla (1282-1348)</autor>
        <publicacion-lugar>Madrid</publicacion-lugar>
        <publicacion-editor>Jaime Ratés</publicacion-editor>
        <publicacion-fecha>1920</publicacion-fecha>
        <serie>Biblioteca Calleja</serie>
    </monografia>
</biblioteca>
```

En la *tabla7*, se puede apreciar la estructura del documento XSLT, atendiendo a la declaración de tipo de documento, al nombre de espacio del W3C donde se define el estándar, donde toma las reglas sintácticas y elementos que se emplean en su construcción. Además se incluye como se adelantaba anteriormente, entre *<xsl:template></xsl:template>* toda la construcción correspondiente a un documento html, que es el objetivo de salida para formatear el documento *libros.xml.* Ha de advertirse en el documento y concretamente en el elemento *<xsl:template>*, el empleo de un atributo especial denominado *match="/"* que es

empleado para asociar una plantilla con un elemento XML. Si se utiliza match con valor *backslass* **"/"** define que se aplicará a todo el documento completo; es decir, que la plantilla se aplicara a todos los elementos de que conste el documento XML. Una vez realizada la declaración de documento XSL, y establecido el elemento *xsl:template* con atributo match="/", para determinar la extensión de la aplicación al documento *libros.xml*, se procede a la extracción y representación concreta de los elementos de la sencilla ficha de la monografía de libros. Para ello es necesario acceder al elemento concreto mediante un acceso jerárquico, según el anidamiento del documento XML de origen de datos. En este caso se utilizan las sentencias que seleccionan el valor correspondiente al elemento que se desea visualizar:

Título: Ejemplo de documento XSLT para la transformación de libros.xml
Archivo: catalogo.xsl
Referencia: tabla7

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
    <html><body>
        <h3>catalogo de libros</h3>
        <table border="1">
            <tr bgcolor="#DDDDDD">
                <th align="left">título</th>
                <th align="left">autor</th>
                <th align="left">Lugar de Publicación</th>
                <th align="left">Editor</th>
                <th align="left">Fecha de Publicación</th>
                <th align="left">Serie</th>
            </tr>

        <xsl:for-each select="biblioteca/monografia">
            <tr>
                <td><xsl:value-of select="titulo"/></td>
                <td><xsl:value-of select="autor"/></td>
                <td><xsl:value-of select="publicacion-lugar"/></td>
                <td><xsl:value-of select="publicacion-editor"/></td>
                <td><xsl:value-of select="publicacion-fecha"/></td>
                <td><xsl:value-of select="serie"/></td>
            </tr>
        </xsl:for-each>
        </table>
    </body></html>
</xsl:template>

</xsl:stylesheet>
```

Pero en los casos en que existan más de un registro en el documento XML a transformar, hay que especificar en el documento de transformación XSLT que determinado fragmento de la estructura HTML, donde se representan los valores de los elementos, se va a repetir tantas veces como registros existan en el documento de fuente de datos XML.

Para ello se emplea el código *<xsl:for-each select="">* que permite especificar la ruta de acceso a los elementos *titulo, autor, publicacion-lugar, publicacion-editor, publicacion-fecha y serie* a partir de un determinado nodo, concretamente expresado en la ruta *biblioteca/monografia. Véase tabla7 y 8.*

Título: Ejemplo de sintaxis para el enrutamiento y selección de elementos en XSLT
Archivo: seleccionar-elementos-XSLT.txt
Referencia: tabla8

```
<xsl:for-each select="biblioteca/monografia">

    <xsl:value-of select="titulo"/>
    <xsl:value-of select="autor"/>

</xsl:for-each>
```

Para aplicar una plantilla XSLT a un documento XML es necesario referenciar el documento XML a la plantilla XSLT de la que tomará el formato apropiado. Para ello se introduce el siguiente código. *Véase tabla9*

Título: Declaración para vincular un documento XSLT a un documento XML
Archivo: referenciar-documento-XSLT.txt
Referencia: tabla9

```
<?xml-stylesheet type="text/xsl" href="catalogo.xsl"?>
```

Otras propiedades específicas de XSL, son algunas opciones de recuperación de información predeterminadas directamente en la plantilla. En dichos casos se utiliza la instrucción *for each* con operadores específicos que permiten filtrar los datos del documento XML original. *Véanse los operadores XSLT en la tabla10*

Título: Operadores XSLT
Referencia: tabla10

Operador	Significado
=	Igual
!=	Diferente a…
<	Menos que…
>	Más que…

De esta manera, cuando se quisiera filtrar la obra *El Conde Lucanor del Infante Juan Manuel*, aparecería concretamente la información o datos, concretos del título seleccionado, mediante una instrucción similar a la presentada en la *tabla11*

Título: Ejemplo de sintaxis de filtrado XSLT
Archivo: sintaxis-filtrado-xslt.txt
Referencia: tabla11

Ejemplo de filtrado

```
<xsl:for-each select="biblioteca/monografia[titulo=”El Conde Lucanor”] "></xsl:for-each>
```

Esquema sintáctico

```
<xsl:for-each><xsl:sort select="titulo"/></xsl:for-each>
```

Para organizar la información a partir de un campo concreto alfabéticamente, se utiliza el elemento <*xsl:sort*> sin terminación <*/xsl:sort*> ya que es un elemento unimembre.

XSLT también permite elementos para expresar condiciones para la representación de determinados datos del documento XML. Para ello se utiliza el elemento <*xsl:if*><*/xsl:if*>. Mediante el atributo *test*, y siendo susceptible de utilizar los operadores anteriormente mencionados, permite la recuperación de información de todos aquellos registros que cumplan una determinada condición. Por ejemplo: se establece que se visualicen todos los títulos, publicados posteriormente a 2002. *Véase tabla12*

Título: Ejemplo sintáctico para expresar condiciones en XSLT para la representación de información procedente de un documento XML.
Archivo: condicion-if-xslt.txt
Referencia: tabla12

```
<xsl:for-each select="biblioteca/monografia">
<xsl:if test="publicacion-fecha &gt; 2002">
                    <td><xsl:value-of select="titulo"/></td>
                    <td><xsl:value-of select="autor"/></td>
                    <td><xsl:value-of select="editorial"/></td>
</xsl:if>
</xsl:for-each>
```

En este ejemplo se puede empezar a conocer el verdadero potencial de consulta, recuperación y representación de información de los sistemas de información basados en la familia de lenguajes XML. Téngase en cuenta que XSLT es un lenguaje específico para transformar y presentar datos y tiene a su disposición herramientas de filtrado semejantes a las de un lenguaje de recuperación de información. Se podrá comprobar más adelante que estos cometidos están asignados principalmente al lenguaje XQuery.

XSLT, puede utilizar elementos para expresar condiciones más complejas, también denominadas *Tests de Condiciones Múltiples*. Para estos casos se emplea el elemento *<xsl:choose></xsl:choose>* entre *<xsl:for-each></xsl:for-each>*.

A su vez, entre *<xsl:choose></xsl:choose>* se emplean otros dos elementos inherentes a su funcionamiento, *<xsl:when test=””></xsl:when>* que permite realizar la condición que se debe de cumplir para visualizar la información de una determinada manera y *<xsl:otherwise></xsl:otherwise>*, empleado para determinar el resultado de no cumplir la condición formulada. Este sistema permite formular múltiples condiciones en las plantillas de XSL, para la representación de información. En el siguiente ejemplo colorea el fondo de la casilla que contengan aquellos títulos publicados posteriormente a 2002, dejando en blanco aquellos que no cumplen dicha condición. *Véase tabla13*

Título: Documento XSLT completo con funciones de filtrado y condiciones
Archivo: plantilla-xslt-completa.xsl
Referencia: tabla13

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">

    <html><body>
    <h3>catalogo de libros</h3>
    <table border="1">
        <tr bgcolor="#DDDDDD">
            <th align="left">titulo</th>
            <th align="left">autor</th>
            <th align="left">Lugar de Publicación</th>
            <th align="left">Editor</th>
            <th align="left">Fecha de Publicación</th>
            <th align="left">Serie</th>
        </tr>

    <xsl:for-each select="biblioteca/monografia">
        <tr>
        <xsl:choose>

        <xsl:when test="fecha &gt; 2002">
            <td bgcolor="red"><xsl:value-of select="titulo"/></td>
            <td><xsl:value-of select="autor"/></td>
            <td><xsl:value-of select="publicacion-lugar"/></td>
            <td><xsl:value-of select="publicacion-editor"/></td>
            <td><xsl:value-of select="publicacion-fecha"/></td>
            <td><xsl:value-of select="serie"/></td>
        </xsl:when>

        <xsl:otherwise>
        <tr>
            <td><xsl:value-of select="titulo"/></td>
            <td><xsl:value-of select="autor"/></td>
            <td><xsl:value-of select="publicacion-lugar"/></td>
            <td><xsl:value-of select="publicacion-editor"/></td>
            <td><xsl:value-of select="publicacion-fecha"/></td>
            <td><xsl:value-of select="serie"/></td>
        </tr>
        </xsl:otherwise>

        </xsl:choose>
    </tr>
    </xsl:for-each>
    </table>
    </body></html>

</xsl:template>

</xsl:stylesheet>
```

3.4. DTD

Como se ha explicado anteriormente, para validar el formato de un documento XML y por extensión generar un formato de sindicación es necesario que los elementos, nodos y atributos del documento estén conformados de acuerdo a unas *normas de definición del tipo de datos y organización* (W3C QA Recommended list of DTDs. W3C World Wide Web Consortium, 2009). Para estos efectos existen dos opciones posibles: Los documentos DTD, (Document Type Definition) que tienen ventajas en cuanto a su simplicidad de edición y declaración pero problemas de adaptación al protocolo de transmisión de datos SOAP (fundamental para la elaboración de sistemas y formatos de sindicación); y XSD Schema, que se emplea con las mismas intenciones y finalidades que DTD, con la ventaja de ser compatible con SOAP, pero complicado en cuanto a la declaración y sintaxis que emplea para la definición de los elementos de un documento XML. Sin lugar a dudas ya se puede observar que la sindicación es una tecnología que no sólo emplea un tipo de estructura, un formato o un documento para la declaración y validación de los contenidos, sino que requiere de otros elementos como protocolos de comunicación basados en la red como SOAP, y otros módulos que permiten mejorar y perfeccionar la definición del lenguaje que de sindicación que se pretenda desarrollar o utilizar.

En torno a la descripción del tipo de documento es necesario basarse en fuentes electrónicas especializadas procedentes del W3C Consortium. Concretamente en las *especificaciones oficiales de la DTD* (RAGGET, D. et al., 1999) y en el *manual de referencia del W3School* (DTD Tutorial, W3Schools, 2009) del cual se han tomado ejemplos que posteriormente han sido adaptados al presente estudio.

Una definición del tipo de documento, DTD, permite determinar la arquitectura de los elementos y datos que contiene un documento XML. Define la estructura del documento con una lista de elementos y atributos válidos. Un archivo DTD, puede ser declarado en línea, dentro de un documento XML, o como una referencia externa, siendo ésta la forma más recomendable. Con un archivo DTD, cada documento XML, incluye una descripción de su propio formato. Con un archivo DTD independiente o distribuido, se genera un formato de descripción de contenidos que puede ser compartido por diversos grupos de usuarios, que permite verificar los datos que se reciben y envían.

Definición de la construcción de bloques en DTD

La construcción de un documento DTD está sujeto a una serie de bloques de construcción, a saber: Elementos, Atributos, Entidades, PCDATA, CDATA.

- **Elementos - Elements:** Los elementos constituyen el bloque de construcción principal. Tanto en XML como en html, se corresponden con las etiquetas de un documento html básico o con las etiquetas de los nodos de un documento XML.

- **Atributos - Attributes:** Los atributos aportan información extra sobre los elementos. Siempre están situados dentro de una etiqueta y suelen estar compuestos por el nombre del atributo y su correspondiente valor.

- **Entidades - Entities:** Las entidades son algunos caracteres que tienen un significado especial en XML. Se trata de los signos de mayor, menos, comillas y "ampersand". Estas entidades son utilizadas en muchos casos como resultado del análisis de programas parser de XML.

- **PCDATA:** Acrónimo de *Parsed Character Data*. Se utiliza para expresar que un texto será analizado por un programa parser. El proceso de análisis consiste en la validación de entidades y marcas o etiquetas. De esta forma, las etiquetas que se encuentren en el texto, serán tratadas como marcas y las entidades serán utilizadas con el valor expandido.

- **CDATA:** Acrónimo de Character Data. A diferencia de PCDATA es aquel texto que no será analizado por un programa parser Existen determinados tipos de datos que no requieren de dicho análisis como los hipervínculos o las imágenes en un documento XML. Por ello se suele emplear CDATA.

Ejemplo de DTD interna

Título: Ejemplo de DTD interna para la descripción de un documento XML
Archivo: libros2.xml
Referencia: tabla14

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE biblioteca [<!ELEMENT biblioteca (monografia)>
                     <!ELEMENT monografia (titulo,autor,fecha)>
                     <!ELEMENT titulo     (#PCDATA)>
                     <!ELEMENT autor      (#PCDATA)>
                     <!ELEMENT fecha      (#PCDATA)>]>

<biblioteca xmlns:xlink="http://www.w3.org/1999/xlink">
<monografia categoria=”literatura española”>
  <titulo lang=”es” id=”entremeses”>Entremeses</titulo>
  <autor>Miguel de Cervantes Saavedra</autor>
  <fecha>1967</fecha>
</monografia>
</biblioteca>
```

Ejemplo de DTD externa

Título: Ejemplo de DTD externa para la descripción de un documento XML
Archivo: libros1.xml, libros.dtd
Referencia: tabla15

Documento XML: libros1.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE libros SYSTEM “libros.dtd”]>

<biblioteca xmlns:xlink="http://www.w3.org/1999/xlink">
<monografia categoria=”literatura española”>
    <titulo lang=”es” id=”entremeses”>Entremeses</titulo>
    <autor>Miguel de Cervantes Saavedra</autor>
    <fecha>1967</fecha>
</monografia>
</biblioteca>
```

Documento DTD: libros.dtd

```
  <!ELEMENT biblioteca (monografia)>
  <!ELEMENT monografia (titulo,autor,fecha)>
  <!ELEMENT titulo     (#PCDATA)>
  <!ELEMENT autor      (#PCDATA)>
  <!ELEMENT fecha      (#PCDATA)>
```

Declaración de elementos en DTD

Los elementos en un documento DTD, son declarados como *ELEMENT*. Por lo que la expresión sintáctica del elemento estaría formada por el nombre del elemento y su correspondiente categoría.

La sintaxis de descripción básica de un elemento consiste en declarar el nombre del elemento y si dispone de una relación jerárquica de hijos, incluirlos como contenidos dentro del paréntesis. También puede expresarse en dicho paréntesis si el contenido del elemento debe ser o no analizado mediante programa parser, este aspecto se declara mediante el valor *(#PCDATA)*. Para los casos en los que el elemento no dispone de contenido alguno, se emplea la categoría de elemento *EMPTY*, o vacío. En los casos en los que el elemento puede o no tener contenidos a diferencia de EMPTY que especifica un vacío absoluto, se emplea la categoría *ANY*. Finalmente para los casos en los que se planteen varios hijos, son declarados entre paréntesis, como contenidos del elemento a modo de categoría, significando esto, que deben ser declarados a parte como elementos que conforman la estructura del documento XML.

Título: Elementos de la declaración DTD.
Referencia: tabla16

Declaración del elemento en DTD
<!ELEMENT nombre-del-elemento categoria>

El elemento puede tener hijos u otros elementos que dependen jerárquicamente.
<!ELEMENT nombre-del-elemento (contenido-del-elemento)>

El elemento no dispone de contenido o información alguna.
<!ELEMENT nombre-del-elemento EMPTY>

El elemento será analizado mediante parser en todo su contenido.
<!ELEMENT nombre-del-elemento (#PCDATA)>

El elemento puede o no tener algunos contenidos
<!ELEMENT nombre-del-elemento ANY>

El elemento tiene tres hijos
<!ELEMENT monografia (titulo,autor,fecha)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT autor (#PCDATA)>
<!ELEMENT fecha (#PCDATA)>

Declarar una ocurrencia de un elemento

<!ELEMENT nombre-del-elemento (nombre-del-hijo)>
<!ELEMENT titulo (subtitulo)>

En el ejemplo se declara que un elemento hijo tendrá una única ocurrencia, sólo aparecerá una vez en el documento xml.

Declarar como mínimo una ocurrencia de un documento

<!ELEMENT nombre-del-elemento (nombre-del-hijo+)>
<!ELEMENT titulo (subtitulo+)>

Se declara con el signo *(+)* que el hijo del elemento *titulo* es decir, *subtitulo* debe tener una única ocurrencia como mínimo o más dentro del nodo *titulo*.

Declarar cero o más ocurrencias de un elemento

<!ELEMENT nombre-del-elemento (nombre-del-hijo*)>
<!ELEMENT titulo (subtitulo*)>

Se declara con el signo *(*)* para determinar que *subtitulo* puede tener cero o más ocurrencias dentro del nodo *titulo*.

Declarar cero o una ocurrencia de un elemento

<!ELEMENT nombre-del-elemento (nombre-del-hijo?)>
<!ELEMENT titulo (subtitulo?)>

Se declara con el signo *(?)* para determinar que *subtitulo* solo puede tener cero o una ocurrencia dentro del elemento o nodo *titulo*.

Declarar contenidos y cualquiera de los establecidos a parte

<!ELEMENT nombre-del-elemento (hijo1,hijo2,(hijo3|hijo4))>
<!ELEMENT titulo (subtitulo,(paralelo|tipo-documento))>

Se declara que el elemento *titulo* tiene un hijo *subtitulo* y pueden o no aparecer cualquiera de los establecidos en el segundo paréntesis es decir, el hijo *paralelo* o *tipo-documento*. En este tipo de declaración, tanto el hijo1 como el hijo2, son de aparición obligatoria en el documento xml.

Declarar mezcla de contenidos

<!ELEMENT nombre-del-elemento (hijo1|hijo2|hijo3)*>
<!ELEMENT monografia (#PCDATA|titulo|autor|fecha)*>

Se declara que el elemento *monografia* contiene cero o más ocurrencias que deben ser analizadas por el programa parser de los hijos correspondientes, *titulo*, *autor* y *fecha*.

Declarando atributos en DTD

Para declarar atributos en un documento DTD es necesario aplicar la siguiente sintaxis, expresada en la *tabla17.*

Título: Sintaxis de la declaración de atributos en DTD
Referencia: tabla 17

<!ATTLIST nombre-elemento nombre-atributo tipo-atributo valor-por-defecto >

Para la sintaxis expuesta existen unos atributos determinados y, en consecuencia unos valores específicos o por defecto. El atributo principal de cualquier documento DTD es *type* que permite determinar el tipo de información que contiene un elemento. Para ello existe una tipología específica que guarda una relación especial con sus correspondientes valores. (*Véase la tabla18)*

Título: Cuadro de atributos y valores en DTD
Referencia: tabla18

Tipo de Atributo	Valores Específicos	Valores por defecto
CDATA	El valor son caracteres de datos	**#REQUIRED**
(en1|en2|..)	El valor debe ser uno de una lista enumerada	
ID	El valor es un “id” único	
IDREF	El valor es el “id” de otro elemento	**#IMPLIED**

IDREFS	El valor es una lista de otros “id”	**#IMPLIED**
NMTOKEN	El valor es una etiqueta xml válida	
NMTOKENS	El valor es una lista de etiquetas XML válidas	
ENTITY	El valor es una entidad	**#FIXED**
ENTITIES	El valor es una lista de entidades	
NOTATION	El valor es el nombre de una Notación	
xml	Es un valor predefinido en xml	

Los valores por defecto, que muestra la *tabla18*, son utilizados para designar alguna característica especial en la validación de un elemento en el documento XML. Concretamente #REQUIRED es empleado para definir que el atributo es obligatorio y necesario. #IMPLIED determina que el atributo no es necesario u obligatorio y #FIXED, que el valor del atributo está fijado de manera obligatoria y predeterminada. *Véase tabla19*

Título: Ejemplo de valores Required, Implied y Fixed
Referencia: tabla19

#REQUIRED: es obligatorio y necesario el atributo.
<!ATTLIST titulo lang CDATA #REQUIRED>
<titulo lang=”es” id=”entremeses”>Entremeses</titulo> VÁLIDO
<titulo id=”entremeses”>Entremeses</titulo> INVÁLIDO

#IMPLIED: no es necesario u obligatorio el atributo.
<!ATTLIST titulo lang CDATA #IMPLIED>
<titulo lang=”es” id=”entremeses”>Entremeses</titulo> VÁLIDO
<titulo id=”entremeses”>Entremeses</titulo> VÁLIDO

#FIXED valor: el valor del atributo está fijado o predeterminado.
<!ATTLIST titulo lang CDATA #FIXED “es”>
<titulo lang=”es” id=”entremeses”>Entremeses</titulo> VÁLIDO
<titulo lang=”es”>Entremeses</titulo> VÁLIDO
<titulo lang=”en”>Entremeses</titulo> INVÁLIDO
<titulo id=”entremeses”>Entremeses</titulo> INVÁLIDO

Declarando entidades en DTD

Las entidades son utilizadas para definir atajos a un texto estándar o caracteres especiales. Las entidades pueden ser referencias a otras entidades. Las entidades pueden ser declaradas interna o externamente. La sintaxis de una entidad, se define mediante el nombre de la entidad, y su correspondiente valor entre comillas. *Véase la tabla20*

Título: Sintaxis de una entidad en DTD
Referencia: tabla20

```
<!ENTITY nombre de la entidad "valor de la entidad">
```

Así pues se pueden declarar entidades internas y externas siguiendo el modelo sintáctico determinado. Si bien esto se define fácilmente en el documento DTD, no es así en el documento XML, que permite también su empleo, mediante el uso del signo *ampersand (&)* delante del nombre de la entidad, seguidamente el nombre de la entidad y finalmente el *(;)* para cerrar su correspondiente declaración. *Véase tabla21*

Título: Ejemplo de entidades Internas y Externas
Referencia: tabla21

Entidades Internas

En el documento DTD

```
<!ENTITY editor "Sisebuto Ordaz.">
<!ENTITY prologuista "Valerio de la Bastilla">
```

En el documento XML

```
<autor>&editor;&prologuista;</autor>
```

Entidades Externas

En el documento DTD

```
<!ENTITY editor SYSTEM "http://localhost/entidades.dtd">
<!ENTITY prologuista SYSTEM "http://localhost/entidades.dtd">
```

En el documento XML

```
<autor>&editor;&prologuista;</autor>
```

3.5. XSD Schema

XSD (Schema Definition) es una alternativa al empleo de DTD para describir la estructura y datos contenidos en un documento XML. El propósito de Schema es definir los elementos, atributos, relaciones jerárquicas entre elementos, número y orden de elementos, condición y tipo de contenido de los elementos, tipo de datos para los elementos y sus atributos, definir valores por defecto y fijados para los elementos y sus atributos; casuísticas que pueden aparecer en un documento. XSD Schema está concebido como el lenguaje sucesor de DTD; esto se debe a varias razones como la flexibilidad a la hora de modificar y ampliar las estructuras del documento. S*chema es un lenguaje enteramente basado en XML, lo que le brinda una mayor versatilidad, soportando más tipos de datos que DTD, por lo que se definen mejor los contenidos del documento XML* (XML Schema Part 1: Structures Second Edition, 2004). Schema permite el empleo de etiquetas para definir las estructuras. Una de las grandes ventajas de XML es que admite más variedad de tipos de datos. Permite describir cualquier contenido a lo largo del documento XML, permite la correcta validación de los datos, sin necesidad de disponer de un parser ad-hoc para determinar errores de descripción, permite trabajar con datos directamente desde la base de datos, define facetas de datos o restricciones en los datos, permite definir patrones de datos (es decir formatos de datos), permite convertir datos entre diferentes tipos de datos. XSD Schema utiliza sintaxis xml, siendo ésta otra de las ventajas sobre los documentos DTD. Esto se traduce en, que es posible utilizar el mismo programa parser empleado en DTD, para analizar los schemas; la posibilidad de manipular el Schema con técnicas de XML DOM, así como transformar el Schema mediante técnica y lenguajes XSL-T. XSD Schema además asegura una comunicación segura de los datos, permitiendo el envío y recepción de los mismos. Con Schema el emisor puede describir los datos de una manera que el receptor pueda entenderlo y obtener de dicha forma la confirmación de su recepción. Esto supone una normalización en algunos tipos de datos, lo que permite que sean interpretables por cualquier usuario. Los XSD Schemas son extensibles o ampliables ya que están escritos en xml. *Esto permite que una definición de schema de un documento, pueda ser reutilizada en otros schemas, pudiendo crear un modelo de tipos de datos propio basado en los tipos de datos normalizados* (SPERBER-MACQUEEN, C.M. and Thompson, H., 2007). Otras posibilidades son la referenciación múltiple de schemas en un mismo documento, como se podrá comprobar en sucesivos *casos y ejemplos* (Schema Tutorial, W3Schools, 2007).

Acercamiento al XSD Schema

Los documentos xml, pueden tener una referencia de la XSD que les define, igual que ocurría con DTD. En el documento XSD que se presenta en la *tabla22*, denominado *esquemalibros.xsd* define los elementos del documento XML, *libros1.xml* anteriormente escrito. Se pueden observar los elementos <*complexType*> que sirven para contener a su vez otros elementos. Cuando se desciende en la jerarquía al nivel de *título*, *autor* y *fecha*, al ser elementos simples sin hijos o descendientes, se utilizan etiquetas simples para definir su tipo de datos; empleándose un código similar a la etiqueta <*xs:element*>.

Título: Ejemplo de Schema aplicado a la estructura del documento libros1.xml
Archivo: libros1.xml ; esquemalibros.xsd
Referencia: tabla22

libros1.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<biblioteca
xmlns="http://localhost/prueba"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://localhost/prueba esquemalibros.xsd">

<monografia categoria=”literatura española”>
  <titulo lang=”es” id=”entremeses”>Entremeses</titulo>
  <autor>Miguel de Cervantes Saavedra</autor>
  <fecha>1967</fecha>
</monografia>
</biblioteca>
```

esquemalibros.xsd

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace=" "
xmlns=" "
elementFormDefault="qualified">
<xs:element name="biblioteca">
   <xs:complexType>
     <xs:element name="monografia">
       <xs:complexType>
         <xs:sequence>
                <xs:element name="titulo" type="xs:string"/>
                <xs:element name="autor" type="xs:string"/>
                <xs:element name="fecha" type="xs:date"/>
         </xs:sequence>
       <xs:complexType>
     </xs:element>
   </xs:complexType>
</xs:element>
</xs:schema>
```

El elemento <schema> en XSD

El elemento <*schema*> es el elemento de ruta de cada schema xml. Si bien, siempre las estructuras de schemas, presentan las siguientes formas de construcción elemental. *Véase tabla23*

Título: Sintaxis y estructura básica de un Schema
Referencia: tabla 23

Estructura básica de Schema

```
<?xml version="1.0"?>
<xs:schema>…Resto del desarrollo del schema…</xs:schema>
```

Declaración y definición de espacios en XSD Schema

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://localhost/prueba"
           xmlns="http://localhost/prueba"  elementFormDefault="qualified">

…Resto del desarrollo del schema…

</xs:schema>
```

El elemento <*schema*> puede contener diferentes atributos que están insertos en la propia declaración de un documento XSD. Estos son:

- **Atributo, *xmlns:xs="http://www.w3.org/2001/XMLSchema"*:** indica que los elementos y tipos de datos son los utilizados en el schema de acuerdo con los señalados en el w3c. Esta declaración especifica que los elementos y tipos de datos determinados en la url del w3c, deben emplear el prefijo *xs* en su espacio de nombre.

- **Atributo, *targetNamespace="http://localhost/prueba"*:** indica que los elementos definidos por este schema están registrados, descritos y estructurados en la url que se señala correspondiente a su espacio de nombre.

- **Atributo, *xmlns="http://localhost/prueba"*:** Indica que el espacio de nombres o *namespace* es *localhost*.

- **Atributo, *elementFormDefault="qualified"*:** indica que algunos elementos son utilizados por las instancias de un documento xml, el cual fue declarado en el presente schema debiendo ser calificado en su espacio de nombres.

Referenciando un Schema en un documento XML

Para establecer una referencia a un documento XSD desde un documento XML, de manera que queden vinculados y afectados para su validación, se requiere de una declaración especial que ha de contener el nombre de espacio por defecto. Esta declaración nos describe el schema validador que todos los elementos del documento xml emplean o utilizan estando declarados en el nombre de espacio, de la url por ejemplo ***xmlns="http://localhost/prueba"***. Por otro lado, el atributo de schema está compuesto de dos valores. El primer valor es el nombre de espacio a utilizar, (en el caso de la *tabla24* es *xsi*). El segundo valor que toma es la localización del schema que utiliza ese nombre de espacio; ***xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance".*** Finalmente el último atributo es *schemaLocation* que también está compuesto a su vez de dos valores. El primer valor es el nombre de espacio que utiliza. El segundo valor es la localización del schema xml, que utiliza el nombre de espacio, por ejemplo: ***xsi:schemaLocation="http://localhost/pruebaesquemalibros.xsd"***

Título: Ejemplo de referenciación XSD en un documento XML
Archivo: referencia-xsd.xml
Referencia: tabla24

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<biblioteca
xmlns="http://localhost/prueba"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://localhost/prueba esquemalibros.xsd">
<monografia categoria=”literatura española”>
  <titulo lang=”es” id=”entremeses”>Entremeses</titulo>
  <autor>Miguel de Cervantes Saavedra</autor>
  <fecha>1967</fecha>
</monografia>
</biblioteca>
```

Definiendo elementos simples en XSD

Un elemento simple es un elemento xml que contiene solamente texto. No puede contener otros elementos o atributos. Sin embargo, las restricciones de sólo texto, no limitan a un único tipo de texto. El texto puede ser de diferentes tipos. Puede ser uno de los tipos incluidos en la definición de Schema del documento xml, siendo un operador booleano, una cadena de caracteres, fecha, o cualquier otro tipo de datos definido por el documentalista. Se puede además crear o añadir restricciones, también denominadas facetas a los tipos de datos para marcar los límites del contenido o facilitar la recuperación de datos y establecer un patrón específico. La sintaxis para definir un elemento simple es la siguiente: *Véase tabla25*

Título: Operadores para filtros XSLT
Referencia: tabla25

Sintaxis básica de los elementos simples en XSD

```
<xs:element name=”xxx” type=”yyy” />
```

Donde *xxx* es el nombre del elemento e *yyy* es el tipo de dato que contiene el elemento. En cuanto a las construcciones más frecuentes con esta sintaxis se encuentran los siguientes tipos de datos:

type=”yyy”	Definición
xs:string	Define un valor que es una cadena de texto.
xs:decimal	Define un valor numérico decimal.
xs:integer	Define un valor numérico entero positivo.
xs:time	Define la fecha y hora completa de una zona horaria.
xs:date	Define una fecha con formato yyyy-mm-dd
xs:boolean	Define valores booleanos como verdadero o falso 1 o 0.

Ejemplo de aplicación

```
<xs:element name="titulo" type="xs:string"/>
<xs:element name="autor" type="xs:string"/>
<xs:element name="fecha" type="xs:integer"/>
```

Valores fijados y por defecto en los elementos simples XSD

La diferencia del valor por defecto y del valor fijado en los elementos simples de un schema radica en que el valor por defecto es automáticamente asignado a un elemento cuando no existe otro valor que haya sido especificado en el documento XML. De esa forma todos los libros que no dispusieran de fecha de publicación tendrían automáticamente la fecha del año 2000. En el caso del valor fijado, el valor 2000 es automáticamente asignado al elemento sin posibilidad de anotar en el documento xml, otro valor que determine el contenido del elemento. *Véase tabla26*

Título: Ejemplo de valores por defecto y fijados en un documento XSD
Referencia: tabla26

Valor por defecto
<xs:element name="fecha" type="xs:integer" **default=”2000”**/>

Valor fijado
<xs:element name="fecha" type="xs:integer" **fixed=”2000”**/>

Definiendo atributos en XSD

Los elementos simples no pueden tener atributos. Si un elemento tiene atributos es considerado como un elemento complejo. No obstante los atributos son declarados como tipos simples, ateniéndose a las mismas características sintácticas que un elemento simple incluyendo los mismos tipos de datos que pueden configurarse como string, decimal, integer, boolean, date y time. La única diferencia es la denominación *attribute* que define al atributo. *Véase tabla27*

Título: Declaración de atributos en XSD
Referencia: tabla27

<xs:**attribute** name=”xxx” type=”yyy”/>

Ejemplo de atributo en documento XML.
<titulo lang=”es”>El Quijote</titulo>

Ejemplo de descripción de atributo en documento XSD.
<xs:attribute name=”lang” type=”xs:string” **use=”required”**/>

Como se puede observar en la *tabla27* al igual que los elementos simples, en los atributos, también se pueden aplicar las propiedades de valores por defecto y fijados con las mismas connotaciones y efectos. Además existe otra propiedad para su empleo en la descripción de atributos, se trata de *requiered* que permite especificar que el atributo es necesario y obligatorio, para ello se emplea el atributo *use* en la descripción del atributo

Aplicando restricciones y facetas en schemas XSD

Las restricciones son utilizadas para definir cuáles deben ser los valores aceptados por los elementos o atributos XML. Las restricciones son elementos XML denominados también *facetas* aunque no hay que confundir dicho término con el referido en la construcción y organización de los tesauros facetados. Aunque existe un paralelismo pues ambos tipos de facetas restringen un tipo de valores concreto en un caso y en el caso de los tesauros, una materia, concepto o área de conocimiento determinada. Véanse ejemplos de restricción de valores simple, de conjuntos de valores establecidos y de series de valores en la *tabla28*.

Título: Ejemplos de restricción de valores en XSD.
Archivo: restricciones-xsd.txt
Referencia: tabla28

Ejemplo de restricción en valores Simple: *Se determina que los únicos registros válidos son aquellos que se encuentran dentro del rango de fechas fijado 1957-2006*

```
<xs:element name=”titulo”>
<xs:simpleType>
        <xs:restriction base=”xs:string”>
                <xs:minInclusive value=”1957”/>
                <xs:maxInclusive value=”2006”/>
        </xs:restriction>
</xs:simpleType>
</xs:element>
```

Ejemplo de restricción en conjunto de valores establecidos: *En este ejemplo se determina que los únicos registros aceptados son aquellos cuyos valores sean el idioma español e inglés.*

```
<xs:attribute name=”lang”>
<xs:simpleType>
        <xs:restriction base=”xs:string”>
                <xs:enumeration value=”es”/>
                <xs:enumeration value=”en”/>
        </xs:restriction>
</xs:simpleType>
</xs:attribute>
```

Ejemplo de restricciones en series de valores: *Se determina que el atributo categoría debe tomar el valor literatura española o literatura universal para ser válido.*

```
<xs:attribute name=”categoria”>
<xs:simpleType name=”categoria”>
        <xs:restriction base=”xs:string”>
                <xs:pattern value=”literatura española|literatura universal”/>
        </xs:restriction>
</xs:simpleType>
</xs:attribute>
```

Definiendo elementos complejos en XSD

Un elemento es complejo cuando contiene otros elementos y o atributos. Existen diversas formas de elementos complejos. Son las siguientes: elementos vacíos, elementos que contienen un elemento o más, elementos que contienen solo texto, elementos que contienen tanto otros elementos como texto. Es importante no olvidar que todos estos elementos complejos pueden contener a su vez atributos, recuérdese que un elemento deja de ser simple cuando contiene un atributo, aunque la descripción del mismo en el documento XSD sea simple o sencilla. Para definir un elemento complejo en un documento XSD, existen dos formas de construcción sintáctica:

1. Declarar directamente el elemento, nombrándolo: Utilizando este método se puede especificar la complejidad de su contenido. Los elementos hijos que estén adscritos al elemento superior en el nivel jerárquico están contenidos entre las etiquetas *<sequence></sequence>*, que se trata de un indicador que permite determinar el orden en que deben aparecer los elementos descendientes o hijos, *véase tabla29.*

Título: Ejemplo de declaración de elementos complejos en XSD con hijos
Archivo: declaracion-elementos-complejos1.txt
Referencia: tabla29

```
<xs:element name=”monografia”>
        <xs:complexType>
                <xs:sequence>
                    <xs:element name="titulo" type="xs:string"/>
                    <xs:element name="autor" type="xs:string"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>
```

2. En el caso de que el elemento complejo a declarar, además tenga un atributo, se deberán anotar aquellos elementos que tengan como mismo referente el mismo tipo de atributo, a lo largo de todo el documento, *Véase tabla30.*

Título: Ejemplo de declaración de elementos complejos en XSD con atributos
Archivo: declaracion-elementos-complejos2.txt
Referencia: tabla30

```
<xs:element name=”revista” type=”categoria”>
<xs:element name=”diccionario” type=”categoria”>
<xs:element name=”test” type=”categoria”>

<xs:element name=”monografia” type=”categoria”>
        <xs:complexType>
                <xs:sequence>
                    <xs:element name="titulo" type="xs:string"/>
                    <xs:element name="autor" type="xs:string"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>
```

Definiendo elementos vacíos XSD

Un elemento vacío complejo, nunca tiene contenidos, solamente atributos. Esto quiere decir, que no contiene información entre sus etiquetas, exceptuando en su atributo, pudiendo o no contener a su vez, un valor fijo y por defecto. Véase el ejemplo de la *tabla31*, para comprobar que su descripción en el schema XSD se define con las mismas reglas o normas sintácticas que los atributos y los elementos complejos descritos anteriormente.

Título: Ejemplo de declaración de elementos vacíos en XSD
Referencia: tabla31

Ejemplo de elemento vacío en XML

```
<numero id=”0132” />
```

Definición de un elemento vacío en XSD

```
<xs:element name="número">
  <xs:complexType>
    <xs:attribute name="id" type="xs:positiveInteger"/>
  </xs:complexType>
</xs:element>
```

Definición de elementos complejos únicos en XSD

Son elementos complejos únicos aquellos que contienen un elemento que a su vez conlleva otros elementos inscritos, *Véase tabla32.*

Título: Ejemplo de definición de elementos complejos únicos.
Referencia: tabla32

Ejemplo de elemento complejo único en XML, representado por *monografia*

```
<monografia>
  <titulo>Entremeses</titulo>
  <autor>Miguel de Cervantes Saavedra</autor>
  <fecha>1967</fecha>
</monografia>
```

Definición del elemento *monografia*

```
<xs:element name="monografia">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="titulo" type="xs:string"/>
      <xs:element name="autor" type="xs:string"/>
      <xs:element name="fecha" type="xs:integer"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Definición de elementos complejos de tipo sólo texto en XSD

Un elemento complejo de tipo, *sólo texto* es aquel que puede contener texto y atributos. En el ejemplo de la *tabla33*, el elemento título se define como tipo complejo cuyo contenido *xs:extension* está definido como cadena de caracteres, que corresponde a *Entremeses*, especificando finalmente un atributo con un valor de *id* numérico, que está definido como número entero, mediante el valor *integer*.

Título: Ejemplo de definición de elementos solo-texto
Archivo: definicion-elementos-complejos-unicos-xsd.txt
Referencia: tabla33

Ejemplo de un elemento complejo del tipo sólo texto. *Recuérdese que para ser considerado complejo, ha de disponer de un atributo, según las reglas de definición de schema.*

```
<titulo id=”0132”>Entremeses</titulo>
```

Definición del elemento complejo del tipo sólo texto, *titulo*.

```
<xs:element name="titulo">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="id" type="xs:integer" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

Definiendo elementos complejos con mezcla de contenidos en XSD

Un elemento complejo con mezclas de contenido puede contener atributos, elementos y texto. En la *tabla34*, se muestra un ejemplo de ello. En el documento xml, concretamente el elemento monografia está compuesto por texto y elementos intercalados. Esto significa que el elemento complejo *monografia* tiene que ser definido mediante el atributo *mixed=”true”* en la definición o declaración del elemento en el schema XSD. El resto de la estructura es similar a los anteriores casos, manteniendo el orden de secuencia y determinado cada elemento simple que contiene, concretamente, título, autor y fecha.

Título: Ejemplo de definición de elementos complejos de contenido mixto
Archivo: elementos-complejos-mixtos-xsd.txt
Referencia: tabla34

Documento XML, con elemento complejo y de contenido mixto.

```
<monografia>
Campo de título: <titulo>Entremeses</titulo>
Campo de mención de responsabilidad: <autor>Miguel de Cervantes Saavedra</autor>
Campo de fecha de publicación: <fecha>1967</fecha>
</monografia>
```

Definición del elemento complejo y de contenido mixto en XSD.

```
<xs:element name="monografia">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element name="titulo" type="xs:string"/>
      <xs:element name="autor" type="xs:string"/>
      <xs:element name="fecha" type="xs:positiveInteger "/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Definiendo indicadores para tipos complejos en XSD

En XSD, se pueden controlar cuántos elementos están siendo utilizados en los documentos mediante el empleo de indicadores. Existe una clasificación de tres grupos de indicadores, clasificándose en indicadores de orden, ocurrencias y grupos. A su vez cada tipo de indicador contiene una breve familia de elementos que sirven para establecer parámetros de control y validación en el documento XML.

Indicadores de orden

Son indicadores utilizados para definir el orden de los elementos para la validación de la estructura del documento XML original. Forman parte de este grupo los indicadores *all*, *choice* y *sequence*, que constan de diferente significación y pueden ser de utilidad para definir alteraciones en el orden de aparición de elementos. Desde el punto de vista documental, se trata de una característica que permite facilitar la recuperación y tratamiento de la información estructurada originalmente, en especial en los registros de tipo bibliográfico e incluso materiales que no pueden tener un modelo de descripción y catalogación cerrada, *véase tabla35*.

Título: Indicadores de orden en XSD
Archivo: indicadores-orden-xsd.txt
Referencia: tabla35

- **Indicador de orden “ all ”**

Especifica que los elementos descendientes pueden aparecer en cualquier orden, siendo posible tener solamente una única ocurrencia.

```
<xs:element name="monografia">
  <xs:complexType>
    <xs:all>
      <xs:element name="titulo" type="xs:string"/>
      <xs:element name="autor" type="xs:string"/>
      <xs:element name="fecha" type="xs:positiveInteger"/>
    </xs:all>
  </xs:complexType>
</xs:element>
```

- **Indicador de orden “ choice ”**

Especifica que tanto un hijo como otro pueden tener una ocurrencia.

```
<xs:element name="monografia">
  <xs:complexType>
    <xs:choice>
      <xs:element name="titulo" type="xs:string"/>
      <xs:element name="autor" type="xs:string"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

- **Indicador de orden “sequence”**

Especifica que los elementos hijos deben aparecer en un orden específico, tal y como se emplaza en la descripción. Recuérdese que los hijos o descendientes suelen ser definidos entre estos dos indicadores.

```
<xs:element name="monografia">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="autor" type="xs:string"/>
      <xs:element name="titulo" type="xs:string"/>
      <xs:element name="fecha" type="xs:positiveInteger"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Indicadores de ocurrencias

Son utilizados para definir cuantas veces o con qué frecuencia debe aparecer un elemento es decir, establece el número de ocurrencias del mismo, *véase tabla36*. Esta característica permite la repetición de campos descriptivos en un documento XML, para la descripción de materiales bibliográficos o de cualquier otra índole. Su utilidad directa se sitúa en la repetición de campos de mención de responsabilidad, autores, editores principalmente. Esto favorece el desarrollo de asientos catalográficos con más puntos de acceso al documento de cara a su recuperación en catálogos colectivos.

Título: Indicadores de ocurrencias en XSD
Archivo: indicadores-ocurrencias-xsd.txt
Referencia: tabla36

- **Indicador de ocurrencias " maxOccurs "**

Especifica el máximo número de veces que se puede repetir u ocurrir un determinado elemento.

```
<xs:element name="monografia">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="titulo" type="xs:string"/>
      <xs:element name="autor" type="xs:string" maxOccurs="10"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

- **Indicador de ocurrencias " minOccurs "**

Especifica el mínimo número de veces que se puede repetir u ocurrir un determinado elemento.

```
<xs:element name="monografia">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="titulo" type="xs:string"/>
      <xs:element name="autor" type="xs:string"
                maxOccurs="10" minOccurs="1"/>
      <xs:element name="fecha" type="xs: positiveInteger"
                maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Grupos de indicadores

Son utilizados para definir relaciones de grupos entre los elementos que se deseen. Esto permite una vinculación de elementos simples descendientes de un elemento complejo, *véase tabla37*. Su aplicación en la documentación es la de poder especificar áreas de descripción dentro de la estructura de un registro catalográfico, pudiendo discernir aquellos elementos que por ejemplo forman parte del área de descripción física de los del área de de mención de responsabilidad.

Título: Indicadores de grupos en XSD
Archivo: indicadores-grupos-xsd.txt
Referencia: tabla37

- **Indicador de grupos " Group + name "**

Permite definir todos, parte o una secuencia de elementos dentro de la declaración de grupo.

```
<xs:group name="grupo-monografia">
  <xs:sequence>
    <xs:element name="titulo" type="xs:string"/>
    <xs:element name="autor" type="xs:string"/>
    <xs:element name="fecha" type="xs:positiveInteger"/>
  </xs:sequence>
</xs:group>
<xs:element name="monografia" type="cata-doc"/>
<xs:complexType name="cata-doc">
  <xs:sequence>
    <xs:group ref="grupo-monografia"/>
    <xs:element name="editorial" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
</xs:element>
```

- **Indicador de grupos " attributeGroup + name "**

Permite definir un grupo de atributos bajo una referencia de atributo de grupo.

```
<xs:attributeGroup name="meta-doc">
  <xs:attribute name="subtítulo" type="xs:string"/>
  <xs:attribute name="colaborador" type="xs:string"/>
  <xs:attribute name="fecha-edición" type="xs:date"/>
</xs:attributeGroup>
<xs:element name="titulo">
  <xs:complexType>
    <xs:attributeGroup ref="meta-doc"/>
  </xs:complexType>
</xs:element>
```

El elemento <any> en XSD

El elemento <*any*> permite extender el documento XML con elementos no especificados en el schema, originalmente. Este esquema que emplea <*any*> permitiría introducir documentos XML como el siguiente, en el que se observa un elemento no declarado en el schema anterior, concretamente <*edicion*>. Mediante <*any*> se permite introducir ese elemento y tantos sean necesarios para la descripción de la monografía sin ser necesaria su declaración, de ahí la importancia de la capacidad de extensibilidad, expansión o ampliación de XSD para adaptarse a las necesidades del objeto a describir, *véase tabla38*. Desde el punto de vista de la descripción y catalogación de materiales documentales, la posibilidad de extensibilidad que ofrece este elemento para la validación de documentos es fundamental para ofrecer las propiedades y flexibilidad que requieren muchas veces las normas y reglas de catalogación, incluso aquellas ediciones para la automatización descriptiva de asientos bibliográficos entre otros tipos. De alguna forma, se pueden establecer una serie de campos o elementos descriptivos y posibilitar la introducción de otros nuevos, sin que pueda existir conflicto de compatibilidad y validación del formato. En relación a la sindicación esta propiedad es fundamental para conseguir formatos de sindicación que se adapten mejor a los propósitos de la documentación y su descripción mediante catálogos.

Título: Ejemplo del elemento especial <any> como función extensible en XSD
Archivo: elemento-any-xsd.txt
Referencia: tabla38

Ejemplo del empleo <any> con propiedades extensibles.

```
<xs:element name="monografia">
  <xs:complexType>
    <xs:sequence>
      <xs:element name=”titulo” type=”xs:string"/>
      <xs:element name=”autor” type=”xs:string”/>
      <xs:any minOccurs=”0”/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Ejemplo de extensión permitida en un documento XML.

```
<monografia>
        <titulo></titulo>
        <autor></autor>
        <edicion></edición>
</monografia>
```

El elemento <anyAttribute> en XSD

Mediante el elemento <*anyAttribute*> al igual que <*any*> se permite extender o ampliar el documento XML, en cuanto a su estructura descriptiva, mediante atributos no especificados en el schema. Por lo tanto goza de similares características a las explicadas anteriormente, *véase tabla39*

Título: Ejemplo completo del funcionamiento del elemento <anyAttribute>
Archivo: principal.xsd, atributo.xsd y documento.xml
Referencia: tabla39

- **Schema Original. Establece el elemento </xs:anyAttribute/>**

En este esquema se emplea <*anyAttribute*> que permite la introducción de cualquier atributo en el elemento monografía que no haya sido declarado en el schema principal. Para ello se crea un nuevo Schema con la descripción concreta del atributo estando de esta manera presente en la validación del documento xml, ejecutándose los dos Schemas, principal y atributo.

Schema principal

```
<xs:element name="monografia">
  <xs:complexType>
    <xs:sequence>
      <xs:element name=”titulo” type=”xs:string"/>
      <xs:element name=”autor” type=”xs:string”/>
    </xs:sequence>
    </xs:anyAttribute/>
  </xs:complexType>
</xs:element>
```

- **Schema Atributo. Establece el Schema de un atributo determinado**

En el atributo que se define en este schema, se aplican una serie de restricciones en el valor de idioma o atributo *lang* que pueda contener hipotéticamente el documento XML. Esto permite ampliar tanto como se quiera las posibilidades descriptivas de los elementos de una estructura previamente definida. La extensibilidad permite definir aspectos que inicialmente no fueron planificados.

Schema atributo

```
<xs:attribute name="lang">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="es|en|fr"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
```

- **Documento XML afectado, con atributo declarado en " atributo.xsd "**

El documento XML, ha de incluir una doble referencia al schema principal y al schema atributo, para poder incluir correctamente la función *<anyAttribute>* que permite a su vez validar el atributo *lang="es"* que originalmente no está establecido en la definición de estructura del documento en *principal.xsd*.

Documento xml

```
<persons xmlns="http://localhost/prueba"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:SchemaLocation=" http://localhost/principal.xsd
                    http://localhost/atributo.xsd">

<monografia lang="es">
        <titulo></titulo>
        <autor></autor>
        <edicion></edición>
</monografia>
```

El elemento substitution en XSD

Schema permite que un elemento pueda sustituir a otro. Por ello se emplea el elemento Substitution. Un ejemplo muy característico de su uso, son los cambios de términos de un idioma a otro, concretamente en los nombres de los elementos de un documento XML.

Otros usos podrían ser la sustitución o cambio de los elementos de descripción de materiales, lo que permitiría en el ámbito documental obtener plantillas de etiquetas especializadas para diferentes tipos de documentos.

Para solucionar los problemas de sustitución, se define un atributo denominado *substitutionGroup* en el Schema XML. En este caso se declara primeramente una cabecera del elemento y después se declaran los otros documentos cuyo estado sea el de ser sustituidos por el elemento cabecera, *véase tabla40.*

Título: Ejemplo de sustitución con el atributo “substitutionGroup”
Archivo: substitutiongroup-xsd.txt
Referencia: tabla40

- **Sintaxis de “ substitutionGroup ”**

En este ejemplo se observa que el elemento *name* es la cabecera del elemento *name* y que el elemento *nombre* es sustituible por *name*, mediante el atributo substitutionGroup.

Definición básica de elementos de sustitución

```
<xs:element name="name" type="xs:string"/>
<xs:element name="nombre" substitutionGroup="name"/>
```

- **Schema de sustitución complejo, basado en “ substitutionGroup ”**

El resultado de aplicar esta técnica es que, el schema permite validar correctamente la denominación nombre y monografía, como nombres de elemento para formar parte de la estructura del documento XML.

Ejemplo de doble sustitución, del elemento nombre y monografía

```
<xs:element name="name" type="xs:string"/>
<xs:element name="nombre" substitutionGroup="name"/>

<xs:element name="monograph" type="mono"/>
<xs:element name="monografia" substitutionGroup=" monograph "/>

<xs:complexType name="mono">
  <xs:sequence>
    <xs:element ref="name"/>
  </xs:sequence>
</xs:complexType>
```

- **Documento XML con dualidad validad en la denominación de elementos**

En el documento XML es factible cualquiera de las dos formas en que se expresa la estructura monografía y nombre.

Ejemplo de documento xml, con dos formas válidas de denominación de elementos.

```
<monograph>
  <name>Entremeses</name>
</monograph>

<monografia>
  <nombre>Entremeses</nombre>
</monografia>
```

Bloquear un elemento de sustitución

Para prevenir que otros elementos sustituyan a un elemento específico, se utiliza el atributo de bloqueo *block=”substitution”*. Es especialmente útil para evitar elementos cuya traducción no esté autorizada, *véase tabla41*.

Título: Ejemplo de bloqueo de un elemento de sustitución
Archivo: bloqueo-elemento-sustitucion-xsd.txt
Referencia: tabla41

- **Schema de sustitución bloqueado**

En el schema se agrega el atributo *block=”substitution”* que permite establecer un bloqueo en todos los elementos *nombre* que tengan como objetivo sustituir al nombre de elemento *name*. Similar suerte, corre el término *monografia*.

```
<xs:element name="name" type="xs:string" block="substitution"/>
<xs:element name="nombre" substitutionGroup="name"/>

<xs:element name="monograph" type="mono" block="substitution"/>
<xs:element name="monografia" substitutionGroup=" monograph "/>

<xs:complexType name="mono">
  <xs:sequence>
    <xs:element ref="name"/>
  </xs:sequence>
</xs:complexType>
```

- **Documento XML, con el ejemplo de validación en relación al bloqueo**

Esto nos lleva a que el único documento válido sería aquel que se encuentra escrito en lengua inglesa. El objeto definido en español sería bloqueado por efecto del atributo *block=”substitution”*.

Validado

```
<monograph>
  <name>Entremeses</name>
</monograph>
```

No validado

```
<monografia>
  <nombre>Entremeses</nombre>
</monografia>
```

Los tipos de datos en XSD

Las cadenas de caracteres; string

Utilizando el tipo de dato *string*, define una cadena de caracteres normal, que incluye espacios, retornos de carro, saltos de línea, entre otros recursos de formato. Con el valor *normalizedString* una cadena de caracteres, puede ser posteriormente procesada directamente desde el documento XML, ya que elimina los saltos de línea, retornos de carro y los espacios que son sustituidos por tabulaciones entre caracteres. A efectos de visualización, se obtiene el mismo resultado que *string*, con la diferencia, de que la información queda preparada para trabajos de análisis y procesamiento de datos y recuperación de información que puede llevar a cabo un programa parser, *véase tabla42*.

Título: Ejemplo de definición de cadena de caracteres en XSD
Archivo: definicion-string-xsd.txt
Referencia: tabla42

Ejemplo del uso del tipo de datos *string*

```
<xs:element name="titulo" type="xs:string"/>
```

Ejemplo del uso del tipo de datos *normalizedString*

```
<xs:element name="titulo" type="xs:normalizedString"/>
```

Los datos de fechas; date y time

El valor *date* Define una cadena tipo fecha en la que la información se dispone mediante el orden establecido YYYY-MM-DD (Año, mes y día). Con el elemento time, se obtiene una cadena del tipo hora completa según el orden establecido hh:mm:ss (Horas, minutos y segundos), *véase tabla43*.

Título: Ejemplo de definición de datos de tipo fecha y hora en XSD
Archivo: definicion-date-time-xsd.txt
Referencia: tabla43

Ejemplo del uso del tipo de datos *string*

```
<xs:element name="start" type="xs:date"/>
```

Ejemplo del uso del tipo de datos *normalizedString*

```
<xs:element name="start" type="xs:time"/>
```

Los datos numéricos; decimal e integer

El valor *decimal*, permite definir datos con componente decimal en los elementos de la estructura de un documento XML dado. En cambio *integer*, define números enteros, sin componente fraccional, v*éase tabla44.*

Título: Ejemplo de definición de datos numéricos en XSD
Archivo: definicion-decimal-integer-xsd.txt
Referencia: tabla44

Ejemplo del uso del tipo de dato *decimal*

```
<xs:element name="xcm" type="xs:decimal"/>
```

Ejemplo del uso del tipo de dato *integer*

```
<xs:element name="fecha-publicacion" type="xs:integer"/>
```

Otros tipos de datos; URI para la definición de enlaces

Otros tipos de datos de importancia para su definición o declaración en los documentos XSD, son las URI correspondientes a una dirección absoluta, relativa de un determinado recurso o página web. Para ello se emplea el valor *anyURI*, *véase tabla45.*

Título: Ejemplo de definición de URIs en XSD
Archivo: definicion-uri-xsd.txt
Referencia: tabla45

Ejemplo de definición de URI

```
<xs:attribute name="src" type="xs:anyURI"/>
```

Ejemplo de URI en el atributo de un elemento de documento XML

```
<imagen src="http://localhost/imagenes/quijote.jpg" />
```

3.6. XPath

XPath es el lenguaje que permite la navegación en documentos XML, permitiendo el acceso a determinadas partes como pueden ser sus atributos o documentos. Para ello utiliza una sintaxis para definir las partes de un documento, considerándose un elemento fundamental de XSLT. XPath utiliza expresiones para seleccionar nodos en un documento xml. Por ello las expresiones empleadas son las rutas de acceso a los nodos y elementos del documento. XPath incluye funciones estándar embebidas en el propio lenguaje. Varían desde cadenas de texto, valores numéricos, operadores de fecha, comparación, manipulación de secuencias, funciones y valores boléanos, entre otros.

En el estudio de XPath, se han utilizado fuentes electrónicas especializadas cuyo origen procede del W3C Consortium. Se han seleccionado las *especificaciones del lenguaje XPath* (XML Path Language (XPath) W3C Consortium, 1999) y el correspondiente *manual de referencia*. (XPath Tutorial, W3Schools, 2009).

Para conocer la terminología utilizada en XPath, y en todos los lenguajes XML, para referirse a cada punto o elemento de la estructura, tomamos el siguiente ejemplo que ilustra su correspondiente jerarquía, *véase tabla46.*

Título: Documento libros.xml, ejemplo de estructura XML accesible por medio de XPath
Archivo: libros.xml
Referencia: tabla46

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="catalogo.xsl"?>

<biblioteca>
    <monografia>
        <titulo>El conde Lucanor</titulo>
        <autor>Juan Manuel, Infante de Castilla (1282-1348)</autor>
        <publicacion-lugar>Madrid</publicacion-lugar>
        <publicacion-editor>Jaime Ratés</publicacion-editor>
        <publicacion-fecha>1920</publicacion-fecha>
        <serie>Biblioteca Calleja</serie>
    </monografia>
</biblioteca>
```

Elementos de la estructura de un documento XML en XPath

1. **Nodes - Nodos:** En XPath, existen siete clases o tipos diferentes de nodos: Elementos, atributos, texto, nombre de espacio, instrucciones de procesamiento, comentarios y rutas de los nodos del documento. Los documentos xml son tratados como estructuras arborescentes o jerárquicas desarrolladas a partir de nodos. La ruta jerárquica es denominada *Ruta del Nodo* o *Root Node*.

2. **Atomic Values - Valores Atómicos:** Los valores atómicos son nodos sin hijos o padres. Suelen estar contenidos entre las etiquetas situadas al final de la jerarquía.

3. **Items - Objetos:** Los ítems pueden ser valores atómicos o nodos. Entre los nodos se establecen una serie de relaciones, de padres, hijos hermanos, ancestros y descendientes.

4. **Parent - Padres:** Relación dada entre cada elemento y atributo que tiene un nodo superior en la jerarquía, en consecuencia tiene un padre. Existe relación de padres entre *titulo* (hijo) y *monografia* (padre).

5. **Children - Hijos:** Relación en la que el nodo o elemento puede tener cero, uno o más hijos, por lo que es la relación inversa al padre. *autor* es hijo de *monografia*. Y *monografia* tiene seis hijos, *titulo*, *autor*, *publicacion-lugar*, *publicacion-editor*, *publicacion-fecha* y *serie*.

6. **Siblings - Hermanos:** Relación en la que un elemento o nodo depende, junto con otros situados al mismo nivel jerárquico, de un mismo padre o nodo superior. En este caso los tres hijos del padre *monografia* son hermanos.

7. **Ancestors - Ancestros:** Relación que se establece a partir del segundo nivel de profundidad en una jerarquía. En este caso, el nodo ancestro es biblioteca, del cual depende *monografía* como hijo respecto a él.

8. **Descendants - Descendientes:** Es la relación inversa con respecto a *ancestro*. Son descendientes los hijos de los hijos. En este caso respecto de *biblioteca*.

Sintaxis en XPath

XPath emplea expresiones para seleccionar nodos u otros elementos en un documento XML. El nodo es seleccionado mediante una serie de pasos:

- **Selección de Nodos:** Para seleccionar nodos en un documento xml, XPath emplea expresiones concretas; siendo estándares las siguientes:
  1. **nodo0**: Selecciona el nombre del nodo.
  2. **/nodo0**: Selecciona la ruta correspondiente al nodo de forma absoluta.
  3. **nodo0/nodo1**: Selecciona todos los elementos cuyo item sea nodo1, que sean hijos de nodo0.
  4. **//nodo1**: Selecciona todos los item de nodo0 que existan en el documento xml.
  5. **nodo0//nodo1**: Selecciona todos los nodos1 que encuentre en orden descendente después de niveles jerárquicos inferiores a nodo0.
  6. **//@lang**: selecciona todos los atributos de que se denominan lang, en todo el documento.

- **Construcción de predicados:** Los predicados son utilizados para encontrar un nodo específico o un nodo que contiene un valor concreto. Los predicados son embebidos o introducidos entre corchetes, quedando de esa forma correctamente definidos.

- **Selección de nodos desconocidos:** XPath también es capaz de utilizar comodines para recuperar elementos desconocidos en XML.

- **Seleccionar diferentes rutas:** Utilizando el operador "|" de barra vertical, en XPath se expresa la selección de múltiples rutas, pudiendo combinarlas según las necesidades.

Los ejes de XPath

Un eje define un nodo relativo en relación al nodo actual disponible. Basándose en las relaciones entre nodos determinadas anteriormente, se tienen en consideración los siguientes ejes:

1. **ancestor:** Selecciona todos los ancestros, padres, abuelos o nodos que se encuentran en los niveles superiores de la jerarquía, respecto de un nodo determinado.
2. **ancestor-or-self:** Selecciona todos los ancestros padres y abuelos que se encuentran en los niveles superiores de la jerarquía, respecto de un nodo determinado o cualquier otro que se encuentre su mismo nivel jerárquico. El resultado puede ser la selección de padres diferentes pero que se encuentran en un nivel jerárquico superior.
3. **attribute:** Selecciona todos los atributos del nodo actual
4. **child:** Selecciona todos los hijos del nodo actual
5. **descendant:** Selecciona todos los descendientes de un nodo.
6. **descendant-or-self:** Selecciona todos los descendientes de un nodo, y de aquellos que se encuentren en su mismo nivel jerárquico.
7. **following:** Selecciona todos los elementos en el documento, antes de cerrar la etiqueta del nodo actual en dicho nivel jerárquico.
8. **following-sibling:** Selecciona todos los hermanos del nodo actual.
9. **namespace:** Selecciona todas las etiquetas del nodo actual.
10. **parent:** Selecciona el padre del nodo actual.
11. **preceding:** Selecciona todos los elementos en el documento después de la etiqueta de inicio del nodo actual.
12. **preceding-sibling:** Selecciona todos los hermanos después del nodo actual.
13. **Self:** selecciona el nodo actual.

Expresión de ruta de localización en XPath

Una ruta de localización puede ser *absoluta* o *relativa*. Una ruta de localización es absoluta, cuando comienza con slash "*/*". Una ruta de localización es relativa cuando no utiliza *slash*. En ambos casos la localización de la ruta consiste en uno o más pasos, cada uno de ellos separados por slash, v*éase tabla47.*

Título: Ejemplo de ruta absoluta y relativa en XPath
Archivo: rutas-xpath.txt
Referencia: tabla47

Ejemplo de ruta relativa
biblioteca/monografia/titulo

Ejemplo de ruta absoluta
/biblioteca/monografia/titulo

Cada paso establecido por "/" es evaluado en función del nivel jerárquico del nodo. Por ello las rutas se generan mediante diferentes pasos jerárquicos que configuran un predicado determinado, que a su vez está circunscrito según la siguiente sintaxis: *véase tabla48.*

1. **axisname:** Un Eje que defina el árbol de relaciones jerárquicas entre los nodos seleccionados y el nodo actual.
2. **nodetest:** Un test del Nodo: que identifique un nodo sin un eje.
3. **[predicate]:** Cero o más predicados: para refinar la selección de los nodos.

Título: Esquema de sintaxis para la designación de rutas a elementos en XPath
Archivo: sintaxis-xpath.txt
Referencia: tabla48

Ejemplo de sintaxis de localización final
axisname::nodetest[predicate]

3.7. XLink

XLink define el estándar para la creación de hipervínculos en los documentos xml. XLink está directamente relacionado con XPointer que es el lenguaje responsable de apuntar a partes específicas de un documento xml. Por ello *existe una relación directa en la sintaxis que emplea XPointer y XPath* (XLink and Xpointer Tutorial, W3School., 2009).

XLink es similar a los enlaces de html, aunque con diferencias sustanciales que lo hacen más completo. Cualquier elemento en un documento xml, puede llegar a convertirse en un elemento de XLink. XLink soporta enlaces simples como los utilizados en html, o enlaces extendidos, que pueden enlazar a múltiples recursos a la vez unidos en un mismo hipervínculo. Con XLink los hipervínculos pueden ser definidos fuera de los archivos enlazados.

La relación de XLink con XPointer se basa en los siguientes aspectos; Al utilizar XPath, permite apuntar a determinadas partes de un documento xml y navegar a través de él. XLink emplea el sistema de enrutamiento determinado por XPointer y basado íntegramente en XPath.

Teniendo en cuenta la vinculación entre XLink y XPointer, las fuentes de información utilizadas para elaborar el estudio en este punto pueden ser compartidas indistintamente. No obstante, XLink consta de fuentes electrónicas particulares que son las *especificaciones de programación para hipervínculos* (XML Linking Language (XLink) Version 1.0, W3C Consortium, 2001), y la recomendación conjunta del W3C sobre enlaces y vinculación en documentos XML (THOMPSON, H.S., 2001).

Sintaxis de XLink

En html, se sabe que *<a>* es el elemento que define el hipervínculo, sin embargo xml, no funciona de la misma manera. La solución para crear enlaces en documentos xml es marcar los elementos correspondientes como hiperenlaces. Para conseguir acceso a los atributos de XLink y sus características debemos declarar la etiqueta de XLink en el documento, *véase tabla49.*

Título: Declaración de XLink y ejemplo de un hipervínculo o enlace en XML
Referencia: tabla49

Declaración: <homepages xmlns:xlink="http://www.w3.org/1999/xlink">

Enlace: <homepage xlink:type="simple" xlink:href="http://www.w3.org">
Sitio web</homepage>

Para definir un enlace en XLink, se atiende a incluir los siguientes atributos en la etiqueta:

- *xlink:type=”simple”*: Que permite definir el tipo de hipervínculo, en este caso simple (también puede tomar el valor *multidirectional*), que sólo hace mención una sola referencia, incluida en el siguiente atributo.
- *xlink:href=”http://...”*: Que permite definir el enlace del elemento.

Como se puede ver en el siguiente ejemplo, XLink consta del atributo *xlink:href=””* que permite especificar la url del enlace; además puede utilizar otros atributos para especificar que el enlace se abra en página nueva; para ello emplea *xlink:show=”new”*. Por otra parte también se especifica el tipo de hipervínculo o enlace que dispone, para ello emplea el atributo *xlink:type=””*, *véase tabla50.*

Título: Ejemplo de un documento XML con aplicación de hipervínculos definidos con XLink.
Referencia: tabla50

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<biblioteca xmlns:xlink="http://www.w3.org/1999/xlink">
<monografia categoria=”literatura española”>
  <titulo lang=”es”>Entremeses</titulo>
  <autor>Miguel de Cervantes Saavedra</autor>
  <fecha>1967</fecha>
  <portada
   xlink:type="simple"
   xlink:href="http://localhost/images/entremeses.jpg"
   xlink:show="new">Entremeses de Miguel de Cervantes…
  </portada>
</monografia>
```

```
<monografia categoría=”literatura española”>
  <titulo lang=”es”>El Quijote</titulo>
  <autor>Miguel de Cervantes Saavedra</autor>
  <fecha>1978</fecha>
  <portada
   xlink:type="simple"
   xlink:href="http://localhost/images/quijote.jpg"
   xlink:show="new">En un lugar de la Mancha, de cuyo nombre no
   quiero acordarme…
  </portada>
</monografia>
</biblioteca>
```

En este ejemplo se demuestran funciones sencillas o simples de los hipervínculos equivalentes a las utilizadas en html. Pero XLink tiene otras funciones como acceder remotamente a localizaciones y recursos, en vez de presentarlos directamente en páginas. En el elemento portada se puede comprobar que el atributo *xlink:show* con un valor *new* permitía abrir el recurso o archivo referenciado en la ruta url, en una nueva ventana, pero es posible utilizar atributos para dejar dicho documento embebido es decir, que el recurso, en este caso la imagen de la portada, sea procesada en línea con la página.

De esta manera con XLink se puede especificar cuando un recurso debe o no aparecer. Este aspecto es tratado por los atributos; *xlink:actuate=”onLoad”* para especificar que el recurso debe ser cargado y mostrado cuando el documento se cargue. Sin embargo de utilizar el mismo atributo, con valor *onRequest*; *xlink:actuate=”onRequest”*, se indica que el recurso no debe ser leído o mostrado sin antes haber interactuado o haber hecho clic sobre él.

3.8. XPointer

Conocida la interrelación entre XLink y XPointer, se debe concretar que XPointer utiliza los fundamentos de XPath para apuntar a un determinado lugar o localización del documento XML. Para ello emplea atributos *id* que permiten identificar un determinado punto o elemento en el documento. A su vez XPath actúa para determinar la posición y nivel jerárquico de los elementos. En relación a ello se han seguido las fuentes especializadas que aportan el *schema de XPointer* (DEROSE, S. et al., 2002) y las *especificaciones que recomienda el W3C* (DEROSE, S. et al., 2002).

Título: Ejemplo de un documento XML con aplicación de señalización XPointer
Referencia: tabla51

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<biblioteca xmlns:xlink="http://www.w3.org/1999/xlink">

<monografia categoria=”literatura española”>
    <titulo lang=”es” id=”entremeses”>Entremeses</titulo>
    <autor>Miguel de Cervantes Saavedra</autor>
    <fecha>1967</fecha>
</monografia>

<monografia categoría=”literatura española”>
    <titulo lang=”es” id=”quijote”>El Quijote</titulo>
    <autor>Miguel de Cervantes Saavedra</autor>
    <fecha>1978</fecha>
</monografia>

<monografia categoría=”literatura universal”>
    <titulo lang=”es” id=”palacio”>El Palacio de los Sueños</titulo>
    <autor>Ismail Kadare</autor>
    <fecha>1956</fecha>
</monografia>

</biblioteca>
```

Sintaxis de XPointer

Para apuntar al documento xml, anteriormente descrito en la *tabla 45*; se utiliza XLink en combinación con XPointer como se muestra en el siguiente código de ejemplo, marcando concretamente la *id* con el valor *entremeses* entre XLink y XPointer se utiliza el operador *“#”* similar al empleado en html, v*éase tabla52 sobre la sintaxis de XPointer*.

Título: Sintaxis XPointer aplicada a XLink.
Referencia: tabla52

Notación extendida
xlink:href="http://localhost /libros.xml#xpointer(id('entremeses'))"

Notación Reducida
xlink:href="http://localhost /libros.xml#entremeses"

Si bien la notación extendida utilizada en la *tabla 46* es la estructura formal recomendada, también existe una forma de acortar la notación, eliminando la instrucción XPointer y reduciéndolo al valor del *id*, mediante el empleo de *"#"* obteniendo así una notación reducida, igual de válida en ambos casos.

Capítulo 4. La sindicación de contenidos

La palabra sindicación en su acepción más común, tiene su origen en el vocablo griego *σύνδικος* (con Justicia) y del latín *syndĭcus* que en castellano se denomina *síndico*; entendiendo por síndico aquel hombre elegido por una comunidad o corporación para cuidar de sus intereses. De síndico proviene el verbo transitivo *sindicar* de entre cuyas acepciones se entiende *ligar varias personas de una misma profesión, o de intereses comunes, para formar un sindicato* y entendido como verbo pronominal el *entrar a formar parte de un sindicato*. Del verbo sindicar se obtiene el sustantivo sindicación como la acción o efecto de sindicar o sindicarse. Por tanto puede extraerse una connotación de afiliación o pertenencia a una entidad.

Pero la sindicación empleada en el contexto de esta investigación y más particularmente en el entorno web, adquiere una significación diferente, ya que es tomada del inglés. Esto es debido a que precisamente los pioneros e investigadores de dicha técnica y proceso son autores anglosajones. Teniendo en cuenta este aspecto se viene utilizando sindicación a modo de anglicismo de *syndication*; siendo éste un vocablo ambiguo en cuanto a su significado, puesto que es empleado para definir la transmisión de derechos de emisión en televisión, las licencias de impresión en prensa, así como las propias para la radiodifusión. Estos usos demuestran que su empleo tradicional viene de la mano de los medios de comunicación y más concretamente vinculados a la gestión de los derechos de propiedad intelectual de los contenidos.

Syndication tiene sus raíces a su vez, en el sustantivo *syndicate* o sindicato, tomando su connotación de afiliación o pertenencia a una entidad. De hecho se establece un paralelismo en referencia al vínculo existente entre el trabajador y el sindicato al que está afiliado, al igual que lo hace un medio de comunicación respecto de la fuente de información por la que se nutre habitualmente, para difundir una serie de contenidos.

En este sentido en los países anglosajones han utilizado este término desde las primeras ediciones escritas de los medios de comunicación modernos, adecuándose a los nuevos formatos con el transcurso del tiempo, hasta la llegada del medio fundamental de la sociedad de la información, la web. Así pues *syndication* en el presente trabajo, se aplica a la difusión de contenidos en la web a partir de una fuente determinada, a la cual el destinatario está afiliado o suscrito.

Aunque el empleo de sindicación de contenidos es la fórmula más extendida para enunciar la difusión de información a partir de canales de diversos medios, se tiene constancia de un esfuerzo por encontrar un término sinónimo que evite el empleo del anglicismo sindicación. Se trata de la *redifusión de contenidos*, mencionada en algunos trabajos relativos a la sindicación y a los sistemas de publicación web de la biblioteca digital (FRANGANILLO, J. and Catalán, M.A., 2005) y (PEIS, E. et al., 2008). Dicho concepto resulta válido por facilitar en sí mismo la explicación de una de las características básicas de la sindicación de contenidos, siendo esta la distribución, difusión y coparticipación de la información que ya fue publicada. Su empleo en esta investigación por tanto, se ceñirá para definir el concepto de sindicación y parte de su proceso documental situado entre el productor de la documentación y el receptor de la misma.

4.1. Orígenes de la sindicación

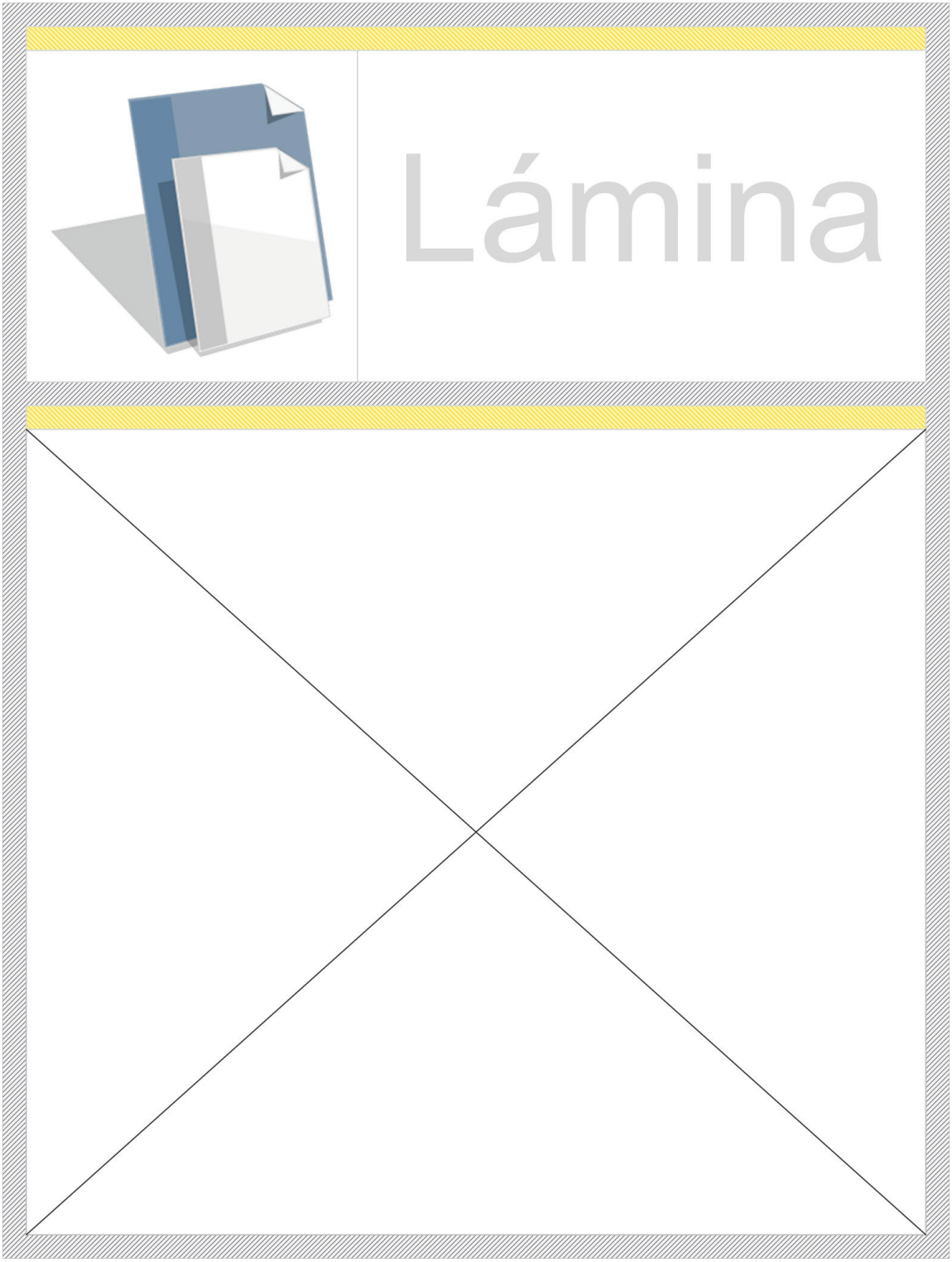
El origen de la sindicación de contenidos resulta complejo de analizar, dado que ha dependido en gran medida del desarrollo tecnológico de los lenguajes de marcado y de la generalización de su uso para transmitir información. Por tanto no se puede atribuir a un único autor la autoría de la sindicación de contenidos, puesto que han sido múltiples investigadores y desarrolladores los que confluyendo sus conocimientos han logrado crear esta técnica y proceso. La historia y evolución de la sindicación de contenidos puede resumirse en la implantación y mejora de los formatos de sindicación. Por ello sus orígenes hay que buscarlos en las investigaciones del primer lenguaje de marcado SGML (Standard Generalized Markup Language) y su método de estructuración de datos que, desembocó en el lenguaje XML, conocido como eXtensible Markup Language. La sindicación está soportada pues, por XML o lo que es lo mismo, los lenguajes extensibles de marcado, para estructurar, organizar y difundir la información vía web. Esto significa que la sindicación de contenidos es consustancial al desarrollo técnico de sus derivados como por ejemplo XSLT, XSD o XPath. Algunos de los responsables directos del desarrollo de la sindicación de contenidos son los investigadores Dave Winer, Ramanathan V. Guha, Dan Libby, Tim Bray, Dan Brickley o Sam Ruby considerados pioneros y padres de esta técnica. A Tim Bray se le atribuye una aportación esencial en el desarrollo de SGML desde 1987, destinada a la creación del lenguaje XML, considerado la base de cualquier formato de sindicación. Ramanathan V. Guha destaca por su investigación sobre los meta-contenidos MCF (Meta Content Framework) entre 1995 y

1997 (GUHA, R.V. and Bray, T., 1997) que dará lugar más tarde al conocido formato RDF Resource Description Frammework en 1999 (BRICKLEY, D. and Guha, R.V., 1999), o marco de descripción de recursos, que supone el punto de partida para el desarrollo del formato de sindicación RSS1.0. Por su parte Dave Winer es el primer autor que logra poner en práctica las técnicas de sindicación de contenidos, elaborando su propio formato de datos XML para su sitio web Scripting News en 1997 (WINER, D., 2001). Este logro fue el resultado de aplicar todos los conocimientos citados sobre los primeros metalenguajes. También se tiene constancia de que el primer navegador web en aplicar sistemáticamente las técnicas de sindicación de contenidos fue Netscape con el formato de sindicación RSS 0.91. Esto se debe al trabajo de Dan Libby y Ramanathan V. Guha autores de la versión de pruebas de dicho formato, denominada RSS 0.90 (JOHNSON, D., 2006) pp57 creada en 1999 hasta su posterior actualización a la versión RSS 0.91 (CADENHEAD, R., 2008) también publicada el mismo año. Por otro lado Dave Winer desarrolla en el año 2000 una segunda variante denominada RSS 0.92 (WINER, D., 2003), con nuevos elementos que amplían las capacidades originales del formato para la descripción de recursos y contenidos informativos. De hecho la sindicación se emplea para compartir tales contenidos, generados por los medios de comunicación, siendo estos los primeros en aprovechar las ventajas de esta técnica para transmitir y difundir en mayor medida su información. En la consecución de ese objetivo se fue haciendo necesaria la creación de programas capaces de codificar la información en los formatos recientemente creados, así como su correcta publicación de forma tal, que pudieran ser admitidos por Netscape. Este problema es solucionado en 1997 con la aparición del programa Manila (USERLAND, 2009), el primer sistema de publicación de contenidos vía web con sindicación activa, desarrollado por UserLand, la primera empresa especializada en este tipo de software, con Dave Winer a la cabeza de su dirección. De esta forma se logra materializar la difusión de noticias en los periódicos como el New York Times (New York Times News Service Syndicate, 2009), el Washington Post (Washington Post RSS news feed topics, 2009), así como en portales de noticias como Yahoo News (YAHOO NEWS, 2009). Esta primera revolución de la sindicación tiene lugar entre 1999 y 2002, fechas en las que aún resulta muy novedosa y poco extendida. Seguidamente se presenta un periodo de maduración que abarca desde el año 2002 hasta el 2004, según las fuentes consultadas (FESTA, P., 2003), periodo en el que se produce un perfeccionamiento de las técnicas, formatos y programas que utilizan la sindicación de contenidos. Dave Winer es considerado en este sentido, el responsable de gran parte de los avances de ese periodo de perfeccionamiento, puesto que materializa una de las primeras aplicaciones Weblog derivada del programa Manila y extiende

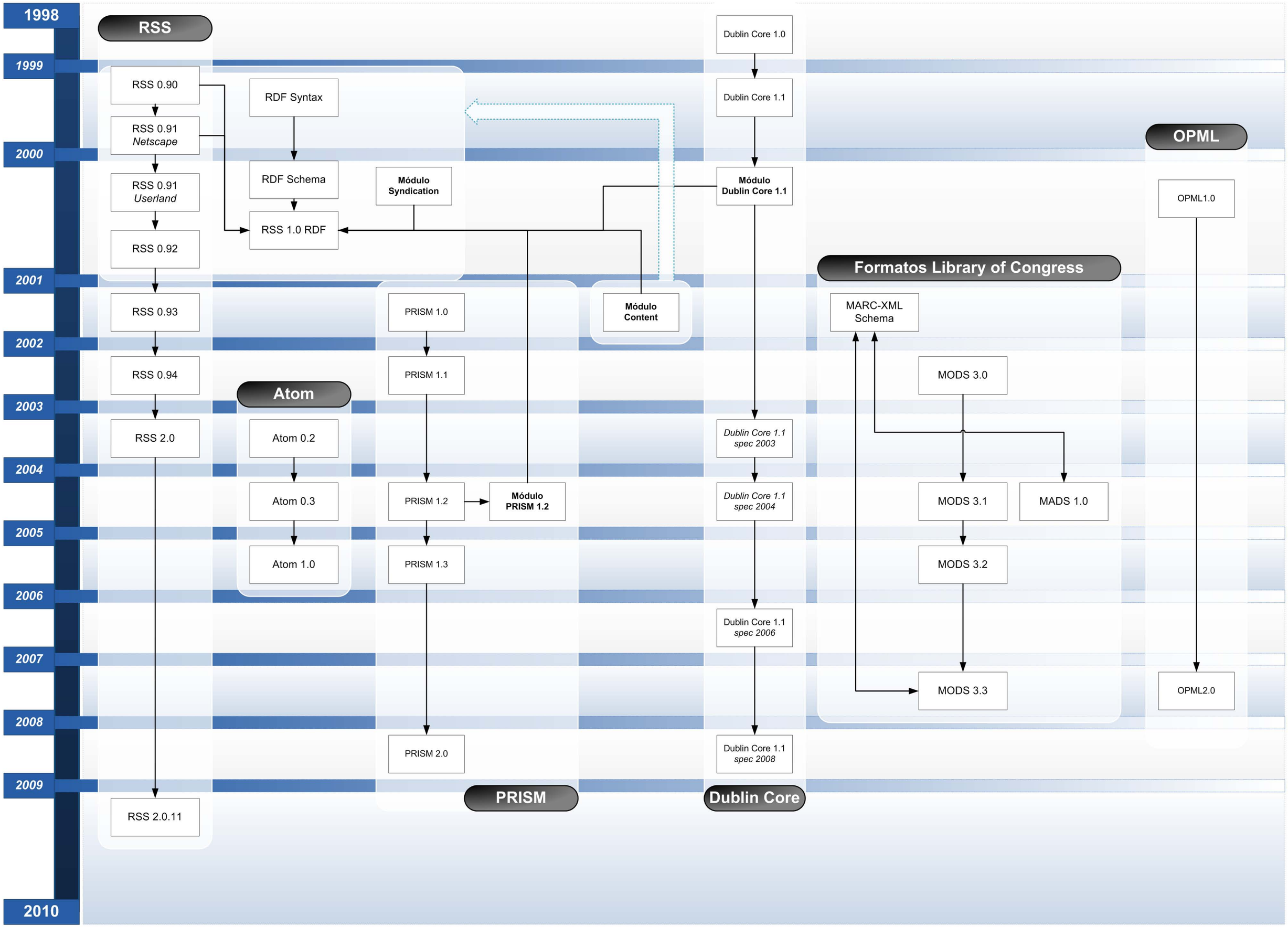
su utilización no sólo a los medios de comunicación, sino al entorno académico, concretamente en la Universidad de Harvard, tal como queda reflejado en el diario CNET News (FESTA, P., 2003), en el que se pone de manifiesto la iniciativa llevada a cabo para la gestión y publicación de contenidos informativos y académico-formativos. De esta forma Winer se adelanta a la gran explosión demográfica del fenómeno blog (BLOOD, R., 2000). El resultado de estas experiencias fue la ampliación y actualización paulatina del formato RSS a las versiones 0.93 y 0.94 respectivamente, que darán como resultado definitivo el actual formato RSS 2.0 en el año 2003 (USERLAND; BERKMAN CENTER, 2003). En esa misma fecha comenzaría el desarrollo de un formato de sindicación alternativo a RSS2.0, el formato Atom y su protocolo. La idea fue propuesta por Sam Ruby y pronto obtuvo los apoyos necesarios de otros desarrolladores como Mark Pilgrim, Aaron Swartz o el propio Dave Winer, junto con múltiples empresas y comunidades de desarrolladores (RUBY, S. and Hopkins, D., 2007). El objetivo del formato Atom era lograr una distribución neutral del mismo en los navegadores web, evitando problemas de monopolio, permitiendo que fuera implementado por cualquier usuario, dotándolo de extensibilidad y de una estructura limpia y ordenada. Como resultado de estas pautas se obtuvo una primera versión de pruebas Atom 0.2 y una versión definitiva Atom 0.3 que fue adoptada por Google para equipar sus principales servicios web como Blogger (Atom API Documentation for Blogger, 2006), Gmail o Google News (GData JavaScript Client 2.0 Class Hierarchy, 2009). Pero el formato Atom continuaría su desarrollo, dirigido por Tim Bray y Paul Hoffman, hacia una mayor normalización elaborándose la versión Atom 1.0 que actualmente es reconocida por el W3C Consortium y el IETF (Internet Engineering Task Force).

Como se desprende de este recorrido, la sindicación de contenidos es una técnica que ha tenido un desarrollo caótico e imperfecto, a tenor de la gran cantidad de formatos elaborados. También se advierte un continuo proceso de reinvención, lo que demuestra una importante componente tecnológica inherente a los lenguajes extensibles de marcado para los que la sindicación cumple su principal objetivo, transportar y transmitir información estructurada. A tal consideración se une el modo de empleo y utilización que de la sindicación se ha hecho a partir de los contenidos web y de forma más especializada, como los contenidos informativos, académicos, audiovisuales y documentales, todos ellos aspectos a considerar en una posible definición del término.

Título: Cronograma de la evolución de los formatos de sindicación
Referencia: figura3



Cronograma de la evolución de los formatos de sindicación



4.2. Definición del término

La sindicación de contenidos es un concepto relativamente novedoso si se tiene en cuenta que su generalización no llegó hasta la popularización de los sistemas de publicación web. Por otro lado el concepto *sindicación de contenidos* ha sido confundido con el de *formato de sindicación*, implicando a menudo definiciones parciales o explicaciones sobre el funcionamiento de un determinado formato, omitiendo una abstracción superior y reflexión de la sindicación como un proceso documental que se sirve de tales formatos, técnicas y tecnologías.

También resulta significativa la madurez tecnológica que se ha logrado alcanzar y el poco aprovechamiento logrado. Esto se debe a un ámbito de aplicación centrado en la redifusión de contenidos, variando únicamente el tipo documental y el sujeto productor de la información. Esta praxis tiene su eco en las definiciones de sindicación de contenidos, limitando con ello nuevos usos y ámbitos de aplicación. En consecuencia no se advierten un gran número de definiciones del concepto *sindicación de contenidos* emitidas desde las fuentes formales de información científica. Fuera de estos canales es posible encontrar gran cantidad de definiciones de todo tipo, faltas de revisión, exhaustividad y control, intoxicando la web de información poco fiable y carente de contraste en cuanto al objeto de estudio. Por ello, se ha procurado una consulta basada en documentación especializada, autores e investigadores de la sindicación, artículos y presentaciones del ámbito académico y científico.

Según (TARANKO, S., 2002) "*sindicación es la transmisión de activos informativos y documentales para su reutilización e integración en terceros recursos por medio de una relación canal-suscriptor, que hace que tales contenidos sean sindicados, a múltiples suscriptores.*" Desde el punto de vista formal, la definición abarca los principales actores que intervienen en el proceso de sindicación, incidiendo en el tipo de relación que se produce entre la información que es transmitida por medio del canal, también denominada activo informativo. Resulta muy válida esta consideración, puesto que un activo informacional representa información original que mana de una fuente o recurso. Si bien este extremo también resulta aceptable, no se precisa si dicha fuente puede ser a su vez un recurso primario o secundario. Como se precisará en sucesivos capítulos, la sindicación puede nutrirse, en efecto, tanto de fuentes primarias, como secundarias y terciarias. En consecuencia, aunque la condición de originalidad del contenido suele ser la más extendida, no siempre es así. Es muy

frecuente la reproducción de terceros contenidos, citas, copias y sindicación de terceros canales sindicados. Por último transmite correctamente la característica de múltiple suscripción, que posibilita compartir la información sindicada con los usuarios.

Según (HAMMERSLEY, B., 2003) "*La sindicación de contenidos hace que una parte o la totalidad del contenido de un sitio web esté disponible para su ulterior utilización en terceros servicios. El contenido sindicado puede ser la información propia del sitio o sencillamente sus metadatos… Un feed de sindicación puede ser cualquier contenido, titulares, enlaces a terceros contenidos, el cuerpo informativo del sitio, despojados de su formato visual, y de sus metadatos libremente aplicados… La sindicación de contenidos permite al usuario examinar toda la información de un sitio web y recibir la notificación de sus actualizaciones. Ésta puede variar desde una simple lista de enlaces de un sitio web a otro, hasta los fundamentos de interrelación de la web semántica.*" Si bien Ben Hammersley definió sindicación en el contexto de un formato de sindicación muy concreto, RSS, también es verdad que su definición aporta una connotación de metalenguaje, siendo la sindicación capaz de incorporar meta-contenidos. Esto es muy significativo para los procesos de actualización y notificación de novedades en las fuentes de sindicación, que denomina feeds. Otro aspecto destacable de la definición es una de sus últimas reflexiones en relación a las posibilidades de servicio de dichas fuentes de sindicación, variando desde simples listas de enlaces hasta complejas relaciones propias de la web semántica. Esta afirmación viene a vislumbrar que la tecnología de sindicación y de la web semántica parte de los mismos fundamentos técnicos, dando a entender que es posible su desarrollo desde la sindicación de contenidos y lo difusas que son las barreras que separan todas las técnicas que emplean lenguajes extensibles de marcado basados en XML.

Según (AYERS, D. and Watt, A., 2005) es "*La distribución de contenidos a múltiples usuarios. A menudo se utiliza para referirse a la utilización de contenidos en múltiples sitios web*" La definición se centra en el proceso de difusión de contenidos sin especificar su tipología, alcance, entorno de aplicación o actores que participan en el proceso de sindicación. Sí queda patente la afirmación de emplear la sindicación como método para embeber contenidos en diversos sitios web, en clara alusión a los agregadores y lectores de canales de sindicación.

Según (FRANGANILLO, J. and Catalán, M.A., 2005) en su trabajo *bitácoras y sindicación de contenidos: dos herramientas para difundir información*, explican que la sindicación hace posible tener constancia de la actualización de una gran cantidad de fuentes de información sin recurrir a la navegación, dando a entender una de sus principales características. A continuación exponen una definición elaborada por el Termcat, Centro de Normalización Terminológica de Cataluña que define sindicación como un "*Proceso por el cual un productor o distribuidor de contenidos en internet proporciona información en formato digital a uno o varios suscriptores, generalmente para que la integren en sus sitios web.*" (TERMCAT, Centre de Terminologia, 2009). En la misma línea se encuentra la definición de (RODRÍGUEZ GAIRÍN, J.M. et al., 2006), al modificar levemente dicha definición. "*La sindicación es el proceso mediante el cual un productor o un distribuidor de contenidos en internet los proporciona a un suscriptor, o a una red de suscriptores. Éstos pueden utilizar un agregador de noticias (lector de fuentes de sindicación) para suscribirse a diversos canales y recibir notificaciones del ámbito que les interese*". Ambas definiciones son correctas y concordantes en gran medida, al coincidir en los actores fundamentales del proceso de sindicación; el sujeto productor, la disposición de contenidos, su publicación y la transferencia a un público objetivo que suscrito al canal de noticias puede recibir cualquier actualización de la información demandada de una manera más personalizada. No obstante podría precisarse mejor el principal medio de representación de los canales de sindicación, el navegador web, así como especificar los aspectos que hacen clave que un formato XML, sea denominado, de sindicación, implicando una reflexión además de documental, técnica.

Según (JOHNSON, D., 2006) es "*Proporcionar una representación XML de noticias o titulares, de actualización constante en blogs, wikis o con cualquier otro tipo de dato que pueda ser distribuido como una colección de elementos discretos*" La definición de Johnson está orientada en un primer momento a la difusión de información periódica y por tanto al entorno de los medios de comunicación. No obstante vuelve a recalcar algunos aspectos ya señalados anteriormente, como la actualización de los contenidos, los sistemas de publicación web sobre los que se aplica más frecuentemente la sindicación ya sean blogs, wikis o CMS. A pesar de que resultan elementos sencillos de la sindicación, también aporta un rasgo no declarado hasta el momento que es la disposición de una colección de elementos discretos. Con ello determina la disposición de dos partes inseparables que permiten hablar de un feed y por tanto de sindicación de contenidos. Se trata del canal y los ítems, o lo que es lo mismo

aquellos elementos organizados que constituyen una colección, sin definir por ello la naturaleza de los datos.

Según las especificaciones del formato RSS1.0 (BEGED DOV, G. et al., 2008) "*Sindicación es la toma de datos disponibles en línea para la recuperación y posterior transmisión o agregación de la publicación en la red.*" Aunque la última fecha de actualización de las especificaciones de RSS1.0 data del año 2008, se tiene constancia de la presente definición en la primera versión de dicho documento en el año 2000. Resulta muy clarificadora la identificación de la sindicación con la transmisión y recuperación de la información y su relación con la publicación en la red. De hecho supone concebir los formatos de sindicación como meros medios de comunicación, considerando la sindicación de contenidos como aquel proceso que permite la transacción de datos e información con distintos propósitos ya sea su búsqueda, consulta, recopilación o re-publicación.

Según (ICSC, 2008) "*La sindicación es una estructura de negocios de gran alcance que facilita la distribución de contenidos, productos e información en la red. Ésta supera el problema de la fragmentación del mercado, permitiendo presentar un mismo contenido en múltiples sitios web con el objetivo de aumentar la capacidad de los usuarios para encontrarlo. Mediante el método de suscripción y agregación, se logran importantes beneficios en los cuatro componentes de la industria de los medios de comunicación: los creadores de contenidos, editores, anunciantes y usuarios… Sindicación es el posicionamiento controlado de un mismo contenido en múltiples destinos de la web*" La visión ofrecida por el Internet Content Syndication Council en esta definición resulta novedosa y a la vez particular; Se entiende la sindicación como parte del mecanismo de negocio de los medios de comunicación de masas en su estrategia de información global a través de la red. Tal aspecto resulta de interés para compañías dedicadas a la información, nuevas tecnologías y el desarrollo web como Google, AT&T, Reuter, NBC o CBS, que han respaldado el desarrollo de esta institución. En la definición también es destacable la concepción de la sindicación como un método de posicionamiento de noticias, que en sí mismo no deja de ser un hecho banal e implícito, si no fuera por la capacidad de controlar el contenido para ser posicionado en terceros recursos web. Esta enunciación significa que pueden llegar a controlarse los contenidos producidos a través de métodos de post-edición, pero también que la sindicación puede ser utilizada con propósitos SEO Search Engine Optimization. Comprendida la dimensión comercial de la sindicación, la definición carece de

un enfoque documental, tendiendo a relatar elementos más relacionados con el periodismo y el marketing.

De todas las definiciones analizadas existe una serie de afirmaciones claras en relación a la sindicación de contenidos; 1) La sindicación es un proceso de comunicación que permite transmitir un contenido de un productor a múltiples usuarios. 2) La sindicación es un conjunto de elementos que conforman una colección o canal de sindicación. 3) La sindicación está basada en XML. 4) La sindicación permite integrar contenidos en múltiples sitios web para su aprovechamiento. 5) Los contenidos que se transmiten mediante sindicación son actualizados de forma constante o periódica. 6) La naturaleza de los contenidos sindicados es variada, pudiendo ser cualquier tipo de dato o información. 7) Existe una clara relación entre los sistemas de publicación y la sindicación como su herramienta de difusión. 8) Existen herramientas que facilitan la lectura de ordenación de la información transmitida mediante sindicación, haciendo alusión a lectores y agregadores especializados en canales de sindicación.

Teniendo en cuenta lo dicho, sindicación de contenidos es el proceso de redifusión de información que permite la suscripción a una fuente de información alimentada por sujetos productores de contenidos informativos, documentales o procedimentales en el corpus de un canal y un formato de datos que lo estructura para su intercambio, servicio, recopilación, lectura y gestión por parte de administradores, editores y usuarios.

La sindicación también puede ser entendida como técnica de transmisión de datos a partir de archivos XML configurados como canales de información de actualización periódica, que posibilitan el intercambio de contenidos publicados o no en el entorno web, posibilitando su alerta, conocimiento, gestión, almacenamiento y tratamiento.

Desde el punto de vista técnico, la sindicación emplea una serie de elementos muy concretos, a saber; 1) el formato de sindicación propiamente dicho basado siempre en XML, 2) el archivo XML debe estar bien formado y validado, 3) La disposición de un canal o colección de ítems o elementos jerárquicamente embebidos dentro del mismo, 4) los esquemas de descripción de contenidos denominados SCHEMAS o DTD que lo definen, 5) las hojas de formato y estilo adaptadas y desarrolladas en XSL o CSS, para su correspondiente visualización, 6) y la utilización del protocolo básico de comunicación web HTTP para la transmisión de datos o el empleo de protocolos específicos de comunicación como SOAP o

XML-RPC que permiten la recepción de una fuente de información determinada por parte del usuario es decir su transferencia.

Atendiendo a un enfoque más documental, la sindicación es el proceso de transmisión de información que haciendo uso de técnicas de redifusión estructurada de los contenidos es capaz de representar una o diversas fuentes de información, también denominadas Feeds, para el intercambio y distribución de la documentación entre múltiples usuarios y terceros medios de comunicación, utilizando para ello un entorno web hipertextual e interactivo, que permite definir en todo momento, el origen, la fuente, la autoría y las dataciones cronológicas del ciclo vital del documento representado. Por ello la sindicación implica una cadena documental desde el momento en que se genera el contenido o el documento, procesado y editado hasta su inserción en el canal o fuente de información del sujeto productor, mediante diversos medios de publicación web como bitácoras, wikis, portales de contenidos, buscadores, directorios de noticias y agregadores.

4.3. Propiedades y características de la sindicación

Estructuración de los datos: Al emplear formatos extensibles basados en XML, la sindicación permite la transmisión de información estructurada. Ello depende de la sintaxis o arquitectura con la que se definen los formatos de sindicación o los lenguajes que se emplean para sindicar los contenidos. En el caso de los formatos tradicionales de sindicación como RSS o Atom éstos están preparados para la descripción de diversos tipos de recursos web, distinguiendo sus elementos básicos en el propio lenguaje. Por ejemplo en la descripción de un elemento se utilizan etiquetas para contener el título, el nombre y datos de filiación del autor, el contenido o descripción del mismo, el enlace al documento original, su identificador universal URI, entre otros. La posibilidad de estructurar la información de un documento y conocer de qué partes consta, así como qué función desempeñan en el conjunto, hacen que la sindicación sea un medio ideal para transmitir cualquier tipo de documentación o información, siempre que se emplee el lenguaje adecuado. A la postre esta estructuración de la información es la que permite que las aplicaciones de sindicación encargadas de la actualización continua de los contenidos o la lectura de los canales, sean capaces de reconocer las estructuras de cada formato. De esta forma, un lector de canales de sindicación especializado en RSS, puede que no sea capaz de captar otros formatos de sindicación como RDF o Atom. A raíz de esto, se

vislumbra también un problema de armonización de los formatos de sindicación, dada su amplia variedad y desarrollo, que se ha tratado de solucionar con aplicaciones parser de reconocimiento de formatos y lectura especializada. El ejemplo más claro de ello es el *UniversalFeedParser* (PILGRIM, M., 2006) desarrollado por Mark Pilgrim, que logra el análisis y reconocimiento de canales de sindicación en los formatos RSS1.0, RSS2.0 y Atom. No obstante no siempre existe una herramienta universal, teniendo en cuenta que los anteriores formatos son los más comunes. Esto ocurre cuando un parser tiene que enfrentarse a un formato para el que no está diseñado. El ejemplo más claro es el formato MARC-XML especializado en la descripción catalográfica de los documentos y en la consecución de registros bibliográficos, que consta de las mismas características de estructuración que los formatos de sindicación clásicos, pero sin el mismo soporte tecnológico. Por lo tanto puede concluirse que la estructuración de la información facilita la descripción de cualquier tipo de documento, identificando todos sus apartados correctamente mediante sus etiquetas y sintaxis. Si bien la elección de un formato u otro puede determinar la correcta descripción de un documento, no siempre posibilita su análisis en aplicaciones parser de lectura y aprovechamiento de los contenidos, lo que hace condicionar en gran medida el empleo de formatos poco adecuados para lograr una mayor difusión o alcance, como podría desprenderse del ejemplo de MARC-XML. Tal problema es resuelto en esta investigación, dado que MARC-XML es el principal formato elegido para elaborar los servicios de gestión catalográfica mediante sindicación.

Extensibilidad: La sindicación hereda la propiedad de extensibilidad propia de los lenguajes de marcado derivados de XML. Esto significa que la estructura original de un canal de sindicación puede ser ampliada según las necesidades de descripción, permitiendo la combinación de diversos formatos y una mayor polivalencia.

Modularidad y transformación de la información: La información estructurada puede representarse de forma modular, sin necesidad de someterse a un esquema fijo predeterminado. Esto se consigue mediante el empleo de hojas de estilo para la transformación de los contenidos del canal de sindicación, también denominadas hojas XSLT. Posibilitan la transformación, filtrado, ordenación y ejecución de operaciones de búsqueda y comparación relativamente sencillas, lo que a la postre permite transformar y tratar la información original. La modularidad de la información estructurada es esencial para llevar a cabo la visualización y representación de los datos y contenidos que sean precisos en cada

momento, facilitando su correcta representación. Esto es posible ya que cada elemento puede ser referenciado de manera particular e independiente, pudiendo ser recuperado conforme a los criterios que el documentalista establezca.

Capacidad de descripción del tipo de datos: El empleo de técnicas de sindicación supone que aparte de estructurar los datos, existen documentos que validan los tipos de datos que contiene la fuente de sindicación o de datos. Es decir, resulta posible definir qué tipo de datos han de recoger cada uno de los elementos que conforman la estructura o formato de un documento sindicado. Por ejemplo es posible definir cadenas de caracteres, campos numéricos, hipervínculos, entre otros. Esto permite un mejor control y normalización de los contenidos y en general del conjunto de elementos sindicados. Estas formulaciones del formato constituyen el denominado schema XSD o DTD, que permite su validación.

Actualización de alta capacidad: La sindicación tiene la propiedad de permitir una alta capacidad de actualización, en todos los suscriptores que acogen el canal original. Esto se debe al empleo de protocolos de transmisión de datos que facilitan la gestión de las peticiones de datos que se realizan desde los clientes suscritos, que reciben respuestas con las últimas publicaciones actualizadas de manera instantánea.

Compatibilidad de estándares: Cualquier formato de sindicación tiene una alta compatibilidad y está desarrollado de acuerdo con el estándar XML. Esto permite que puedan ser interpretados en cualquier equipo con acceso a la red, ya que su medio de difusión es la propia web, utilizando los mismos protocolos que marcan el funcionamiento de Internet, concretamente http.

Flexibilidad de aplicación: La sindicación es flexible, tanto en cuanto puede tener diversos usos. Se emplea principalmente como sistema de difusión de noticias, pero pueden generarse auténticos sistemas de información a partir de las fuentes de sindicación de otros medios de información primarios. Por lo tanto se puede apreciar que al igual que ocurre en la *teoría de la documentación* (LÓPEZ YEPES, J. and Desantes Guanter, J.M., 1978), en la que el documento primario era tratado y representado generando documentos secundarios, con la sindicación se puede llegar a conseguir los mismos efectos, pero en un cambio de escenario físico a otro virtual y electrónico. Se conocen más aplicaciones que demuestran la flexibilidad de la sindicación. Una de las más importantes es la agregación y los lectores de fuentes de

sindicación, para las que la sindicación se convierte en un recurso para la concentración de un objeto de estudio, investigación o temática especializada que pueda interesar a un determinado investigador.

Redifusión y agregación: Como se ha apuntado en el párrafo anterior, la sindicación se caracteriza principalmente por las propiedades de difusión universal desde la red de un determinado sitio web, a todos los usuarios y clientes de Internet, mediante su divulgación mediante motores de recuperación o buscadores y directorios de canales de sindicación. La otra propiedad fundamental es la agregación o la capacidad de ser recopilados tantos canales, como fuentes de datos y editores se encuentren bajo un determinado área del conocimiento.

4.4. Funcionamiento de la sindicación

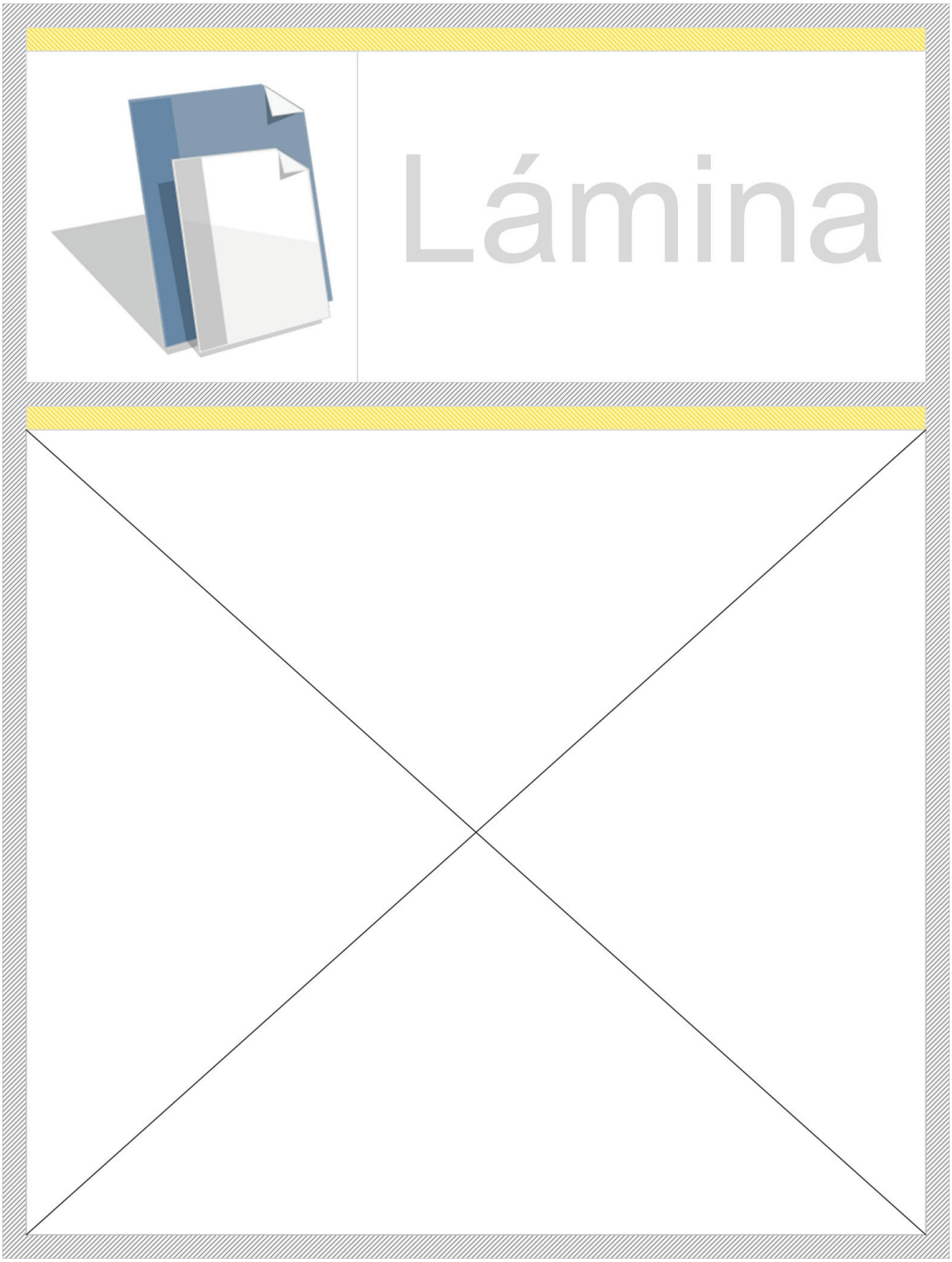
El funcionamiento de la sindicación de contenidos puede observarse a modo de cadena documental y comunicativa donde se suceden una serie de procesos bien definidos y caracterizados por las propiedades de la estructuración de los datos, la extensibilidad, modularidad y transformación de la información, descripción del tipo de datos, la actualización de la información, su redifusión y agregación.

Con la finalidad de desarrollar una idea del funcionamiento de la sindicación de contenidos, con respecto a la transmisión y desarrollo de los contenidos desde su inicio y gestación hasta su recepción por parte de los destinatarios y usuarios de las correspondientes fuentes de datos, se determina el modo de funcionamiento de la sindicación de contenidos en el entorno de red, explicando su entramado, gestión y tratamiento. La teoría y modo de funcionamiento de la sindicación de contenidos es la comunicación que se produce en el entorno de una red telemática global o local en la que los contenidos informativos, documentales, textuales y audiovisuales confeccionados por determinados editores, son emitidos, publicados y difundidos mediante un canal de transmisión de datos para ser recibidos y analizados por iguales, lectores, distribuidores y centros de información y documentación. Si bien la definición aportada resulta amplia y general, permite circunscribir mejor el entorno, los elementos y partes emisoras y receptoras de un determinado tipo de contenidos, que es publicado para su difusión, redifusión, análisis y lectura. (*Véase modo de funcionamiento en el entorno de red, en la figura 4)*. La trama de la teoría de la sindicación de contenidos es además en sí misma la descripción de su funcionamiento, por el cual los responsables de los contenidos, editores, autores de divulgación temática, profesional y científica e incluso instituciones, generan una producción documental con un grado de validez, calidad y representatividad de un área del conocimiento, que puede variar, según su disposición, profundidad en la que se analiza el objeto de estudio y referenciación bibliográfica, o aparato crítico para la corroboración y prueba de la veracidad de lo aportado. En este estadio, los principales responsables de los contenidos, publican la información producen y generan documentación cuyo objetivo particular es su novedad y aportación técnica, teórica, documental y científica, para ser publicada y difundida mediante sistemas de publicación que incorporan en sí mismos el soporte de representación de los contenidos. Es decir, sistemas como los portales de contenidos participativos wikis, los sistemas de gestión del conocimiento CMS y las bitácoras o blogs, incluyen en sí mismos la representación visual y el espacio de

publicación en la red Internet, por tanto en este caso global, en forma de sitio web, portal de noticias e incluso revistas electrónicas. No ha de pensarse que sólo se está restringido a esta tipología de recursos electrónicos, pero sí constituyen los más significativos que ostentan la mayor parte de los contenidos en la red. Además tanto el sistema de publicación como el soporte en muchos casos añade la función de sindicación automática de los contenidos, generando los correspondientes canales o fuentes de sindicación, en los formatos o lenguajes de sindicación RSS, RDF y Atom, que permiten la estructuración de la información publicada para su correspondiente difusión a través del citado canal o fuente creada. Llegados a este término la sindicación cumple ya su objetivo fundamental que es poner en representación global una fuente de datos o canal de sindicación que engloba todos los contenidos de un determinado autor, editor o institución. Pero la evolución de la cadena de sindicación de contenidos ha permitido ciertos controles en la manipulación de las fuentes o canales de sindicación. Esto se consigue mediante programas de sindicación, capaces de realizar la lectura de las fuentes originales. Desarrollados originalmente en el ámbito anglosajón y por ello denominándose *FeedReaders*, permiten captar y agrupar diversas fuentes para el seguimiento alerta y lectura de los contenidos sin necesidad de acceder a la página web original o soporte de representación. Estos sistemas son empleados por los usuarios de la red, consumidores de contenidos de las fuentes de sindicación de un determinado autor, editor o entidad, por lo que les permite tener en conocimiento instantáneo todas las novedades documentales de aquellos recursos que ellos mismos han seleccionado. De esta manera el usuario se convierte en un recolector de recursos y fuentes de forma directa, mediante las técnicas de sindicación. No obstante existen otros caminos para acceder al mismo resultado o al mismo receptor de la información y de los contenidos. Existen entornos en la red considerados de redifusión de los contenidos, conformados por los agregadores y portales redistribuidores de contenidos y fuentes de sindicación. En estos casos las fuentes de sindicación requieren de un tratamiento diferente al empleado por los lectores de fuentes al uso que un usuario determinado puede utilizar de manera local. En este sentido son terceros soportes de representación de contenidos que adaptando programas parser para el tratamiento de las fuentes de sindicación, son capaces de analizar los contenidos originales, y extraerlos para diversos usos de valor añadido ya sea por recuperación, agrupación de contenidos o mejor representación de la información que en el sitio web original. En virtud a estos aspectos, se han desarrollado sistemas de análisis de fuentes de sindicación, parsers especializados en la lectura, la transformación y la recuperación de la información sita en las fuentes. De esta manera terceros medios de redifusión son capaces de adaptar los contenidos a

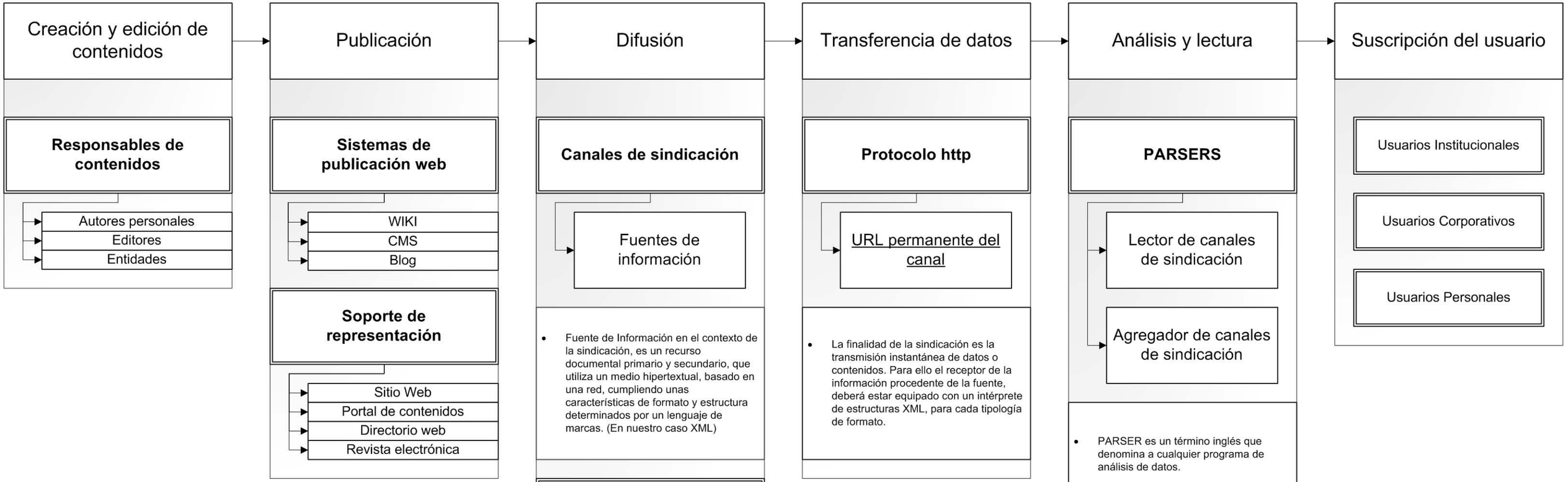
formatos de representación nuevos, organizar y agrupar fuentes o canales de sindicación para su organización temática y clasificación, que constituyen una segunda vía de alimentación para los usuarios de la red, consumidores de determinados tipos de contenidos. También se desprende de la teoría de la sindicación de contenidos, que existe una transformación de los documentos al igual que el fenómeno producido en las bibliotecas y centros de información y documentación, que desarrollan catálogos y productos bibliográficos para difundir y acceder a la información. En la sindicación de contenidos ocurre lo mismo si se otorga el tratamiento de documento a los contenidos producidos por una serie de autores, siendo documentos primarios, que mediante el canal o fuente de sindicación, son manipulados para su organización, distribución y descripción, generándose a la postre una documentación secundaria, referencial de los primeros.

Título: Esquema de funcionamiento de la sindicación de contenidos en el entorno web
Referencia: figura4



Esquema de funcionamiento de la sindicación de contenidos en el entorno web

Referencia: figura4



Paso previo a la sindicación de contenidos: la edición y publicación

- La sindicación de contenidos se circunscribe como proceso de difusión y transferencia de contenidos. Tales contenidos son generados por autores personales, editores, entidades públicas y privadas, con información de muy diverso tipo y condición. Por ello, hay que considerar una primera fase de génesis documental.
- La información o documentación generada puede ser de tipo primario, secundario o terciario, sin que ello afecte a la difusión mediante sindicación de contenidos.
- Por regla general antes de que los contenidos sean sindicados, éstos son publicados mediante algún sistema de publicación web en forma de sitio web, portal de contenidos, directorio o revista electrónica. Para ello existen aplicaciones como los CMS, WIKIS y Blogs que automatizan todo el proceso de edición de los contenidos.
- Cuando el contenido es publicado normalmente se ejecuta de manera automática en éstas aplicaciones, la generación o actualización de un canal de sindicación con todos los contenidos publicados. Por ese motivo se considera el proceso de difusión o redifusión de los contenidos publicados, aunque en ambos casos se suceda simultáneamente.

El canal de sindicación y su transferencia

- Como resultado de la creación del canal de sindicación, se materializa un archivo XML que contiene todos los contenidos publicados, correctamente etiquetados conforme a un formato de sindicación determinado.
- La ruta de acceso al canal de sindicación es precisamente la URL permanente y absoluta a dicho archivo XML, lo que permite su transmisión vía http, mediante el navegador web.

Suscripción, análisis y lectura

- Finalmente actúan los programas parser de análisis para efectuar la lectura y recopilación de los canales de sindicación que a un determinado usuario le pueden resultar pertinentes. Éste proceso es uno de los más importantes en sindicación dado que atañe un proceso de recuperación y representación de la información según la codificación del formato de sindicación.

4.5. Aplicaciones de la sindicación

Habiendo descrito los principales servicios bibliotecarios desde el entorno físico y virtual que puede ofrecer una biblioteca, se procede al análisis de las actuales aplicaciones de la sindicación de contenidos en los servicios bibliotecarios. Hay que apuntar, desde el inicio de este capítulo, que el fenómeno de la sindicación se inicio con el desarrollo de los blogs, también conocidos como bitácoras. La explosión demográfica de este sistema de publicación comienza a partir de 1999 teniendo su máxima expresión de desarrollo a partir del año 2003. Pero hay que anotar que no sólo las bitácoras fueron las únicas promotoras de esta tecnología. Existen otros sistemas que hicieron posible la gestación de la sindicación como los CMS para la gestión documental y del conocimiento o los sistemas WIKI de publicación instantánea de artículos y contenidos cooperativos. La prueba está en la necesidad de realizar seguimientos de la evolución de las publicaciones generadas por un autor mediante un sistema de alerta y actualización automática que desembocaron en el desarrollo de la tecnología de sindicación basada en XML. Por tanto se puede confirmar que la sindicación de contenidos está unida a un problema inherente a cualquier unidad de información y documentación, el problema de la organización y la recuperación de grandes volúmenes de contenidos en espacios globales saturados ya de por sí con grandes o ingentes cantidades de datos. La solución a dicho problema parece lógica, un sistema capaz de compartir y transmitir todos los contenidos publicados, permite algunas posibilidades de organizar el conocimiento y de facilitar las consultas sobre él, asimismo poder determinar qué contenidos pueden resultar accesorios de los realmente pertinentes y válidos.

Estado actual de la aplicación de la sindicación en los servicios bibliotecarios

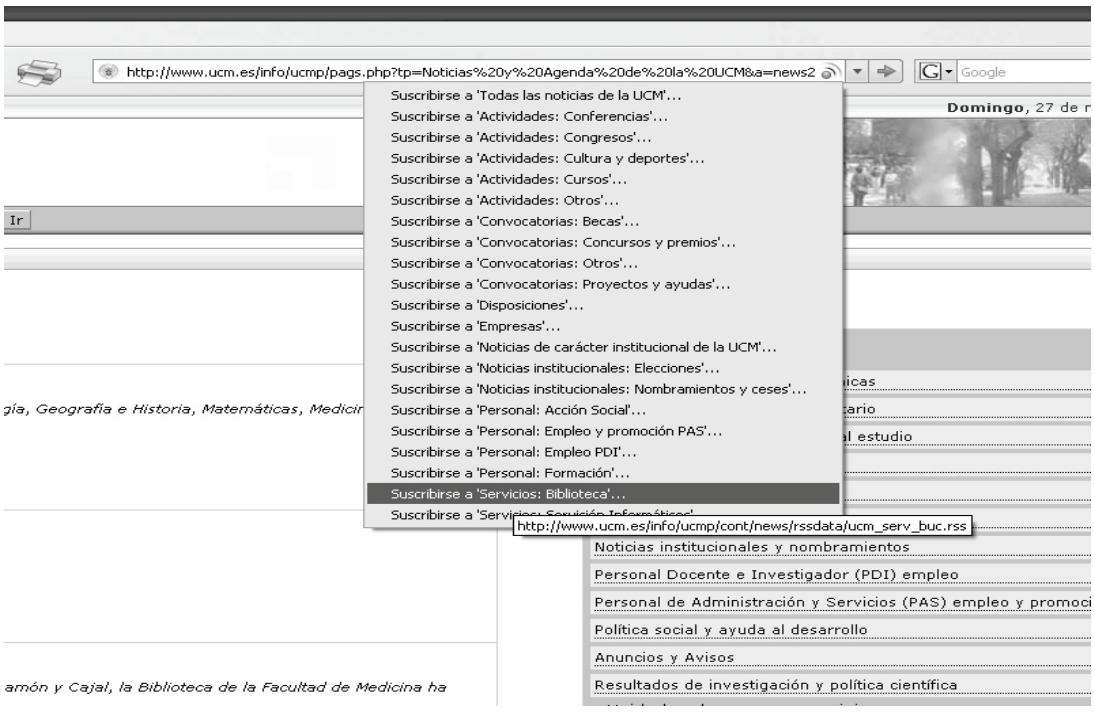
La sindicación aplicada a la biblioteconomía y la documentación está comenzando a dar sus frutos, aunque aún no existen patrones firmes de implantación; sí que se están realizando las primeras pruebas, intentos y casos en los que se comprueba la utilidad del empleo de estas técnicas. Principalmente las áreas más afectadas por la sindicación son todas aquellas en que la biblioteca, realiza algún tipo de publicación, comunicación o divulgación. Esto es debido a que son a su vez los principales objetivos de la técnica de sindicación, de aquí que su aplicación se adapte perfectamente a algunos servicios ofertados por las bibliotecas, principalmente desde su entorno virtual.

En la presente investigación se han encontrado usos concretos en los servicios de información externa de las bibliotecas, en la difusión de noticias y actividades de la biblioteca, en el caso de aplicación de sindicación en un portal de revistas temático, en la aplicación de bitácoras profesionales inscritas en los recursos bibliotecarios y finalmente y de manera fundamental en la difusión selectiva de la información.

Aplicación al servicio de información y referencia

Existen aplicaciones de la sindicación en los servicios de información de noticias, actividades y eventos en algunas bibliotecas universitarias. El servicio de información de una biblioteca se adapta a la perfección a las propiedades de la sindicación y concretamente a uno de los sistemas de publicación al cual siempre ha estado unido; las bitácoras y los CMS. Aunque pueda resultar sorprendente los patrones y modelos de publicación de noticias en los sitios web tienen más que ver con sistemas de gestión del conocimiento y bitácoras que con la gestión manual de la información. Un ejemplo de este supuesto está en el sistema de información de la Universidad Complutense que dispone de un área de sindicación de noticias que se desarrolla en el marco de la biblioteca general BUCM.

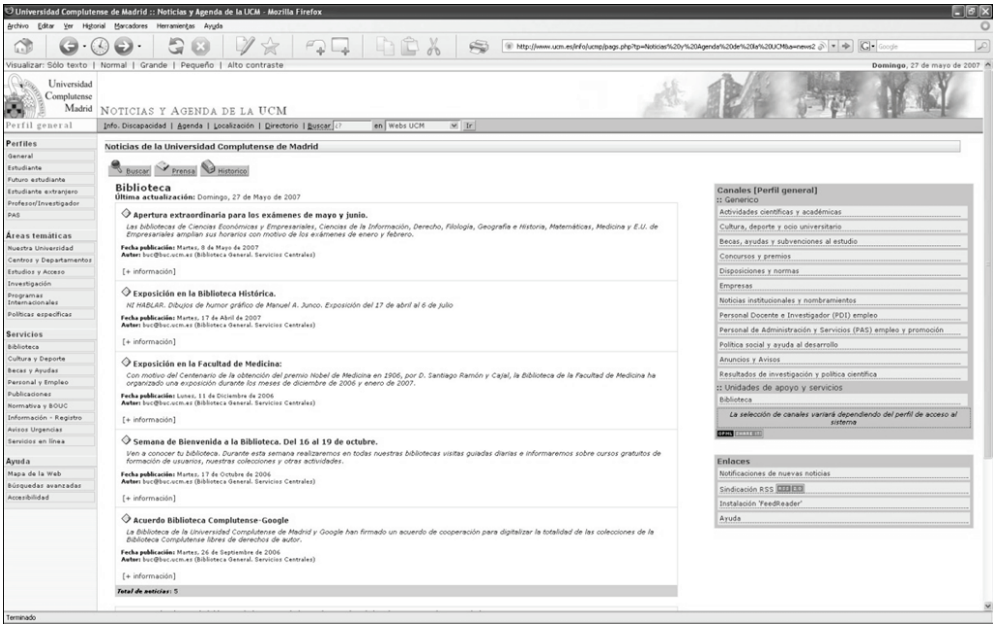
Título: Canales o fuentes de sindicación de la UCM
Referencia: figura5



La impresión de pantalla corresponde al sitio web oficial de la universidad complutense, sección de sindicación. Se puede comprobar que existen múltiples canales o fuentes de sindicación disponibles, que corresponden a los distintos departamentos y áreas que componen la universidad. En concreto está seleccionada la suscripción a la información de servicios de la biblioteca BUCM.

Accediendo directamente a la sección de noticias y sindicación de la universidad complutense se comprueba la disposición de un cuadro de contenidos donde se exponen las noticias sindicadas conforme a la fuente o canal seleccionado en el cuadro alojado en el margen derecho, tabla que contiene todas las referencias correspondientes a las noticias de los diferentes departamentos, que conforman la universidad. En este listado aparece la opción de la biblioteca BUCM que tiene su propia fuente de noticias para la sindicación. Por el momento y tras las pertinentes comprobaciones, aún no existen canales de sindicación particulares para las diferentes bibliotecas de la red complutense que automaticen el servicio de información, pero se puede considerar el esfuerzo conjunto de la administración de la universidad para desarrollar esta tecnología que previsiblemente aplicará a todas sus unidades y centros de información y documentación.

Título: Plano general de la sección de sindicación de la UCM.
Referencia: figura 6



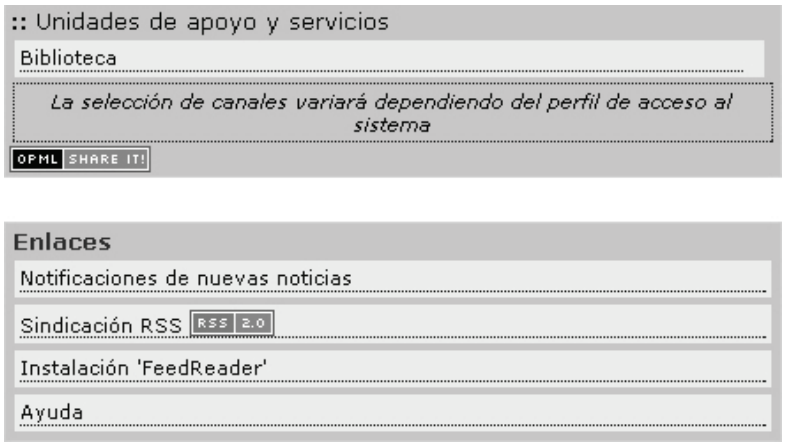
Este es el aspecto general que plantea la sección de sindicación de la universidad complutense. En relación a la arquitectura de la información está diseñada de modo funcional, con los artículos y noticias insertos en el centro de la página y en el lateral derecho los principales canales y fuentes de sindicación a visualizar directamente desde el sitio web.

Inspeccionando más a fondo la página dedicada al servicio de noticias de la universidad complutense se observan detalles de relevancia como el que se presenta en la *figura7*. El empleo de formatos para la agrupación de fuentes de sindicación como OPML[21] (WINER, D., 2007), formato de sindicación utilizado para agregar conjuntos de fuentes. En este caso tiene sentido, puesto que agrega las fuentes y canales de sindicación de todos los departamentos y áreas de la UCM en un solo enlace.

Además de estos aspectos de interés, se encuentra un enlace a *Sindicación RSS* que aparte de indicar el tipo de formato de sindicación empleado, permite acceder a una página de información con la explicación sencilla del formato y los principales lectores de noticias o fuentes de sindicación en RSS.

Finalmente llama la atención el enlace al sistema de lectura de feeds o fuentes, *FeedReader*, sistema de código libre y libre distribución, que constituye el soporte recomendado para los suscriptores y usuarios de los diferentes canales de información y noticias.

Título: Cuadro de opciones de sindicación de la UCM
Referencia: figura7

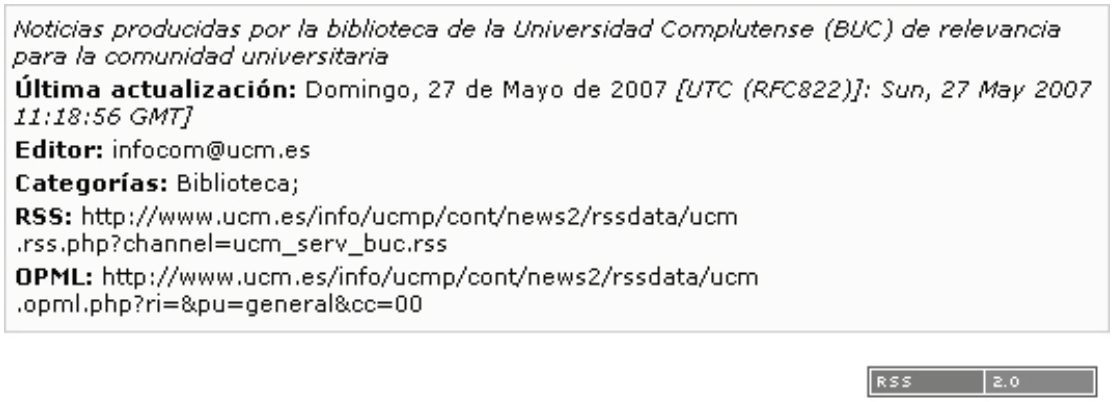


En el cuadro de opciones de sindicación de la UCM, aparece referenciado el canal o fuente de datos de la biblioteca, como unidad de apoyo y servicios. Por lo que se demuestra el uso de la sindicación en los servicios de información bibliotecarios.

[21] **OPML:** Siglas de *Outline Processor Markup Language*. Es un formato XML, utilizado para el procesamiento de contenidos textuales y principalmente la agrupación de fuentes de sindicación basadas en RSS para su posterior uso en sistemas gestores y agregadores.

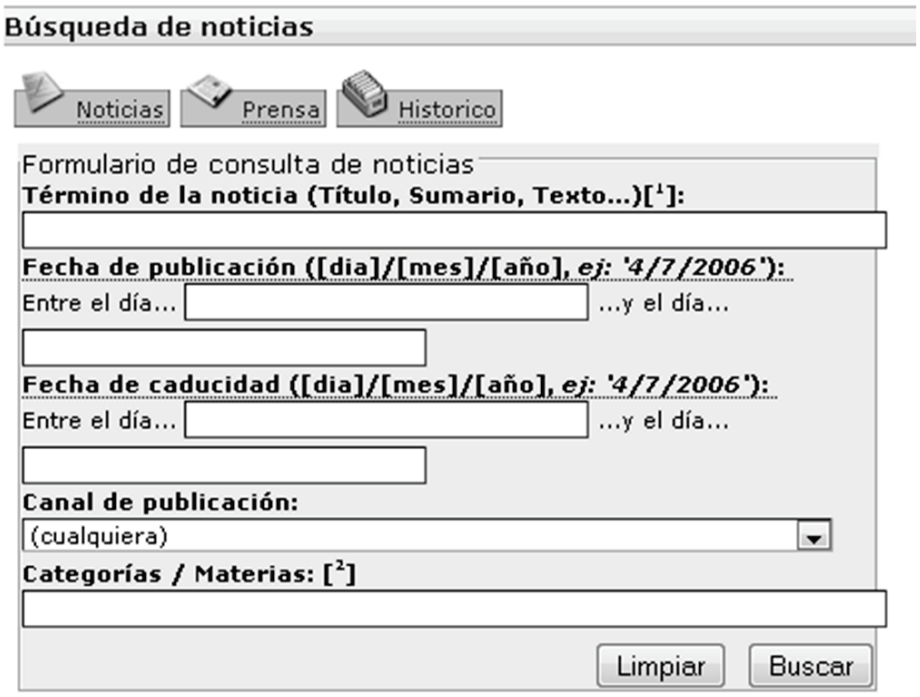
Otro detalle que se aprecia en el estudio de este caso es la correcta adopción de los créditos y características de los canales o fuentes de sindicación dispuestos en la sección. Atienden a la descripción de la fecha de actualización completa, editor, mención de responsabilidad, clasificación de la fuente de sindicación, vínculo al canal y al documento OPML que enlaza el resto de canales de sindicación disponibles, además del visualizado de la biblioteca, v*éase figura8.*

Título: Créditos del canal o fuente de sindicación de la UCM
Referencia: figura8



En este gráfico se puede observar la correcta descripción de la fuente de sindicación correspondiente a la biblioteca de la universidad complutense que atiende a la fecha de actualización, editor, categoría a la que pertenece, Fuente de datos particular y archivo de general de fuentes de datos de toda la UCM, en formato OPML, señalado anteriormente. Al igual que en el cuadro de opciones de la figura7 también aparece el icono que señala o indica el formato de sindicación empleado, concretamente RSS.2.0.

Título: Sistema de consulta de noticias sindicadas en la UCM
Referencia: figura9



En esta imagen se puede observar el sistema de consulta de noticias sindicadas, si bien atiende a los campos que el formato de sindicación RSS, permite aplicar en todos los contenidos publicados.

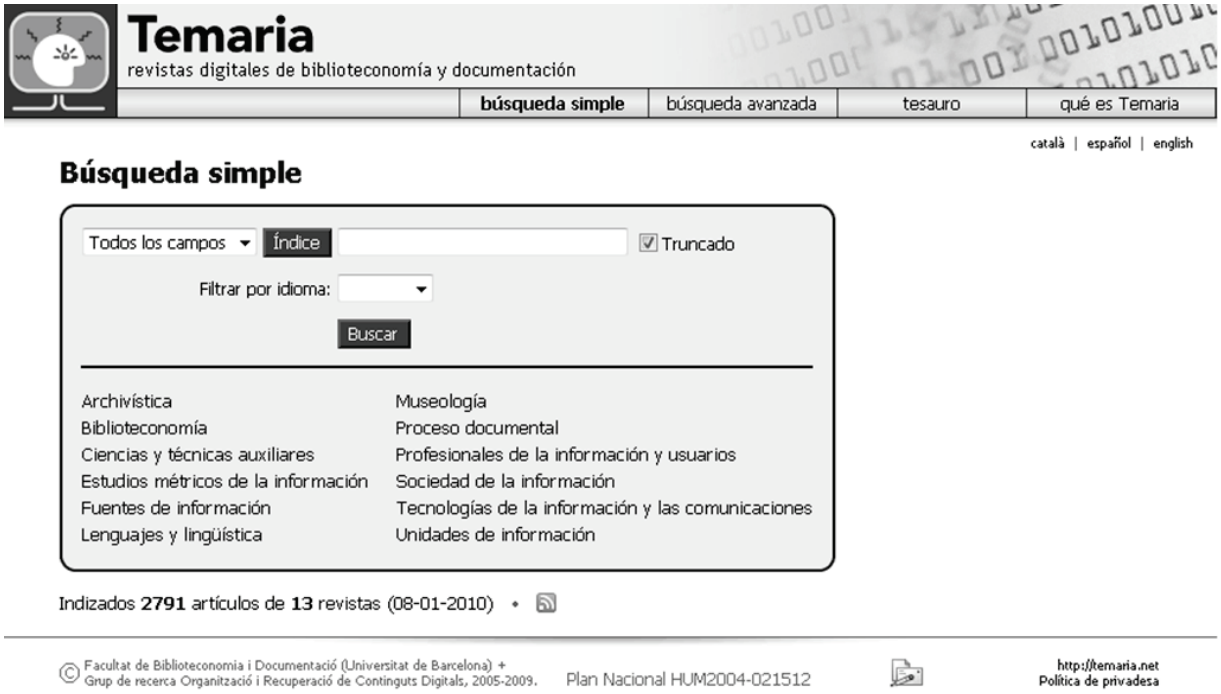
Finalmente se puede acceder al sistema de consulta de los canales o fuentes y artículos que se publican diariamente en todas las áreas de la universidad complutense. Se observan las posibilidades de recuperación de información mediante la filtración del canal, palabras que contiene el título, fecha de publicación y materias clasificatorias que proceden de un tesauro no visualizado, interno al sistema de gestión de contenidos que emplea. El resultado de las consultas es en todo caso completo y pertinente, pero con la carencia de no disponer del correspondiente vínculo sindicado. Aún así la aplicación muestra la información correspondiente de los artículos y noticias seleccionadas por el usuario, constituyendo un sistema de información ejemplar.

Sindicación en revistas electrónicas: Estudio de caso de Temaria

Una de las principales aplicaciones plenamente operativas a modo servicio de referencia para bibliotecas es la sindicación de revistas en el desarrollo de un portal temático especializado. Este servicio de referencia en línea que anteriormente se refería en el capítulo de descripción de servicios quedaría sindicado completamente, como se muestra en el caso del portal de revistas digitales de biblioteconomía y documentación *Temaria* (RODRÍGUEZ GAIRÍN, J.M. et al., 2006). Se trata de un trabajo reciente elaborado por el grupo de investigación *Organización y Recuperación de Contenidos Digitales* de la facultad de biblioteconomía y documentación de la Universidad de Barcelona, formado por Ernest Abadal, Assumpció Estivill, Jorge Franganillo, Jesús Gascón y Joseph Manuel Rodríguez Gairín en el que se concibe la sindicación, como una cadena de producción, compuesto por editores, autores y distribuidores, que generan unos contenidos periódicos que son sindicados por ellos mismos, o mediante el trabajo de gestión, importación y transformación de documentos del portal Temaria. El resultado de la captación de todas las fuentes de suscripción, ha dado como resultado un sistema capaz de recopilar al igual que un agregador, todas las noticias y artículos generados en tiempo real, clasificados y organizados semiautomáticamente, mediante un tesauro electrónico aplicado. Este sistema permite la recuperación de la documentación sindicada mediante parámetros como título, autor, materia, resumen o identificador, de la misma manera que en el caso anterior la universidad complutense trataba la información y contenidos correspondientes a las noticias y novedades de cada departamento y de la biblioteca en particular.

El caso de Temaria tiene un interés especial para los servicios bibliotecarios, ya que es una aplicación directa a los servicios de información y referencia bibliográfica, así como de consulta de revistas electrónicas en línea a las que una unidad de información y documentación puede estar adscrita. En este sentido la fórmula desarrollada en este caso supone un adelanto en la manera de gestionar la información bibliográfica de revistas y un paso adelante en la consecución de la biblioteca digital con funciones plenas.

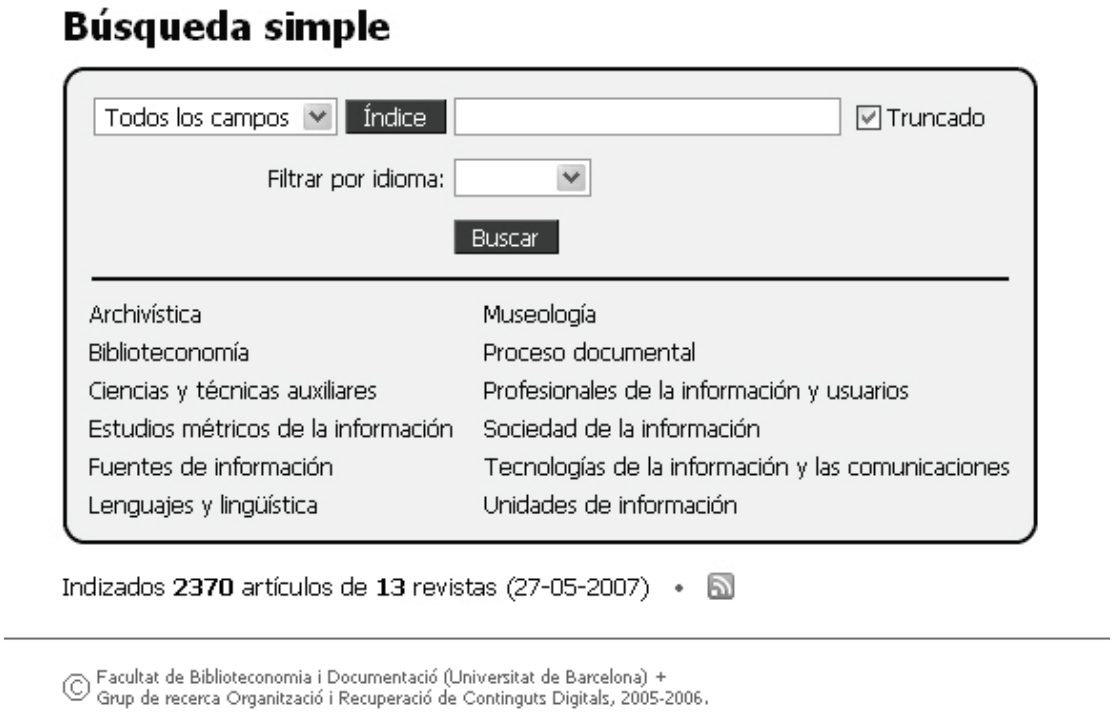
Título: Portada del Portal de Revistas Temaria
Referencia: figura10



Aspecto del portal de revistas digitales de biblioteconomía y documentación Temaria, desarrollado por el *Grupo de Investigación de Organización y Recuperación de Contenidos Digitales* de la facultad de biblioteconomía y documentación de la universidad de Barcelona, formado por Ernest Abadal, Assumpció Estivill, Jorge Franganillo, Jesús Gascón y Joseph Manuel Rodríguez Gairín.

Si existen aspectos y técnicas que interesan sobremanera en este proyecto, son: el desarrollo de una guía de estilos que determina las características del servicio de sindicación, la disposición de una herramienta documental como un tesauro especializado en documentación, para gestionar y clasificar los contenidos de las revistas, y las posibilidades de recuperación de fuentes sindicadas mediante el cambio de parámetros en la URL de la fuente de sindicación.

Título: Sistema de consulta de artículos de revistas sindicadas
Referencia: figura11



El portal de revistas Temaria es capaz de organizar los artículos de las principales revistas de biblioteconomía y documentación digitales, mediante la categorización correspondiente a un sencillo tesauro al uso.

En relación a la denominada, *Guía de estilo* (ESTIVILL RIUS, A., 2006) que emplea Temaria, salta a la luz el empleo de *metadatos Dublin Core*, para distinguir los elementos y calificadores de los artículos y contenidos publicados por las revistas. Esto supone atender a los campos de título, creador, materia, descripción, editor, colaborador, fecha, tipo de recurso, formato, identificador del recurso, fuente, lengua, relación, cobertura y derechos. Si bien Temaria no adapta exactamente igual todos los campos del estándar Dublin Core, si define unos concretos, lo que viene a justificar la edición de dicha guía de estilo que define las características de los elementos que emplea; que corresponden a title, *creator{personalName, corporateName, address}*, *subject, TBD (Tesauro de biblioteconomía y documentación), CDU2000*, *description{ abstract, tableOfContents, publisher{personalName, corporateName, address}*, *contributor{personalName, corporateName, address}*, *date {issued (W3CDTF)}*, *Type {text, format, extent, medium}*, *IMT (Internet Media Types o Tipos MIME)*, *identifier, bibliographicCitation*, *source*, *language(ISO639-2)*, *relation, {isVersionOf, hasVersion,*

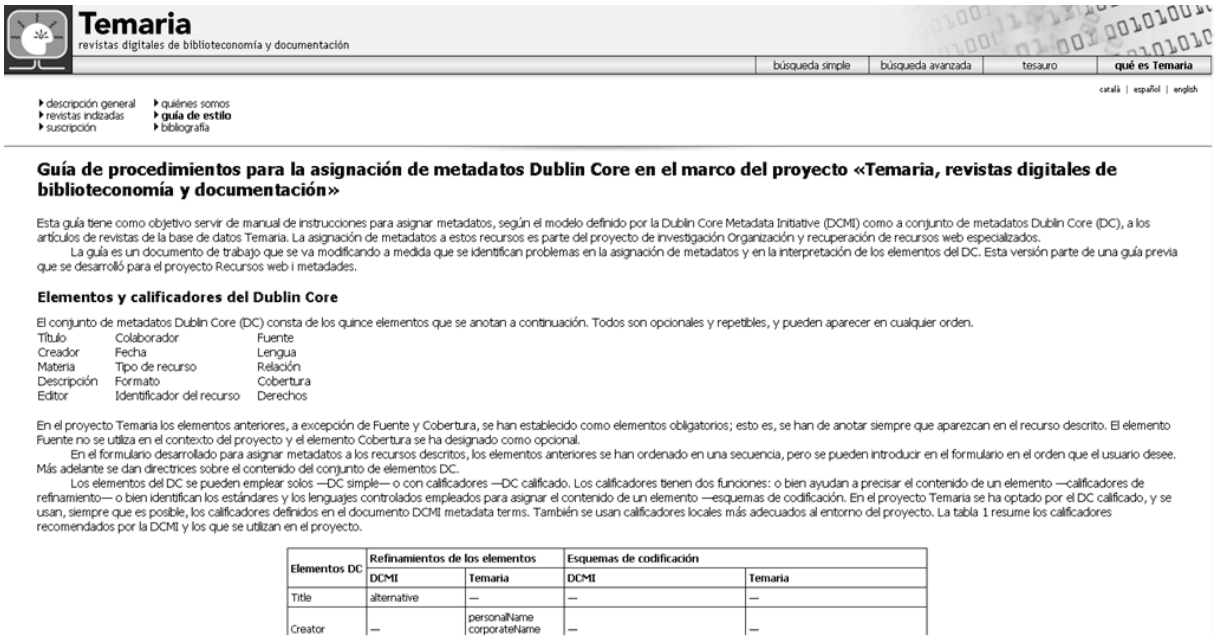
isPartOf, isFormatOf, hasFormat}, coverage [spatial, temporal, ISO3166, CATMARC043, W3CDTF], rights. Expresado en un archivo de metadatos Dublin Core, el formato Temaria para la descripción de los artículos de revistas tendría el siguiente aspecto, v*éase tabla53.*

Título: Esquema de descripción con Metadatos Dublin Core adaptados por Temaria
Referencia: tabla53

```
<meta name="DC.title" content="título">
<meta name="DC.creator.personalName" content="nombre y apellidos del autor">
<meta name="DC.creator.address" content="dirección de correo">
<meta name="DC.creator.corporateName" content="institución a la que pertenece">
<meta name="DC.subject" content="materia del tesauro" scheme="TBD">
<meta name="DC.subject" content="clasificación CDU" scheme="CDU2000">
<meta name="DC.description.abstract" content="resumen">
<meta name="DC.description.tableOfContents" content="índice de contenidos">
<meta name="DC.publisher.personalName" content="nombre del editor">
<meta name="DC.publisher.corporateName" content="institución a la que pertenece">
<meta name="DC.publisher.addressName" content="dirección de correo">
<meta name="DC.date.issued" content="fecha de publicación">
<meta name="DC.type" content="tipo de documento">
<meta name="DC.format" content="formato MIME">
<meta name="DC.identifier" content="identificador del recurso " scheme="URI">
<meta name="DCTERMS.bibliographicCitation" content="Cita bibliográfica">
<meta name="DC.language" content="idioma" scheme="ISO639-2">
<meta name="DC.relation.isVersionOf" content="version de">
<meta name="DC.relation.hasVersion" content="tiene versión">
<meta name="DC.relation.isPartOf" content="parte componente de" scheme="ISSN">
<meta name="DC.relation.isFormatOf" content="formato de" scheme="ISSN">
<meta name="DC.relation.hasFormat" content="tiene otros formatos">
<meta name="DC.coverage.spatial" content="cobertura espacial" scheme="ISO3166">
<meta name="DC.coverage.temporal" content="cobertura temporal" scheme="W3CDTF">
<meta name="DC.rights" content="Derechos">
```

Como se puede comprobar Temaria emplea un sistema de sindicación de contenidos para la difusión de los artículos y publicaciones de las revistas científicas desde el apartado del directorio de sindicación que habilita, *véase figura12*, pero en la recuperación de información los datos de los artículos de las revistas, así como la obtención de las correspondientes fichas bibliográficas se desarrollan soportadas en metadatos Dublin Core. La explicación de este fenómeno radica en la utilización de la sindicación en formato RSS, para la importación de los campos descriptivos principales de los artículos de las principales revistas y transformación en archivos de metadatos Dublin Core que resultan más versátiles para su tratamiento documental y recuperación de información en base de datos.

Título: Estilos y especificaciones de la sindicación de Temaria
Referencia: figura12



Temaria
revistas digitales de biblioteconomía y documentación

búsqueda simple | búsqueda avanzada | tesauro | qué es Temaria

català | español | english

descripción general | revistas indizadas | suscripción | quiénes somos | guía de estilo | bibliografía

Guía de procedimientos para la asignación de metadatos Dublin Core en el marco del proyecto «Temaria, revistas digitales de biblioteconomía y documentación»

Esta guía tiene como objetivo servir de manual de instrucciones para asignar metadatos, según el modelo definido por la Dublin Core Metadata Initiative (DCMI) como a conjunto de metadatos Dublin Core (DC), a los artículos de revistas de la base de datos Temaria. La asignación de metadatos a estos recursos es parte del proyecto de investigación Organización y recuperación de recursos web especializados.

La guía es un documento de trabajo que se va modificando a medida que se identifican problemas en la asignación de metadatos y en la interpretación de los elementos del DC. Esta versión parte de una guía previa que se desarrolló para el proyecto Recursos web i metadades.

Elementos y calificadores del Dublin Core

El conjunto de metadatos Dublin Core (DC) consta de los quince elementos que se anotan a continuación. Todos son opcionales y repetibles, y pueden aparecer en cualquier orden.

Título	Colaborador	Fuente
Creador	Fecha	Lengua
Materia	Tipo de recurso	Relación
Descripción	Formato	Cobertura
Editor	Identificador del recurso	Derechos

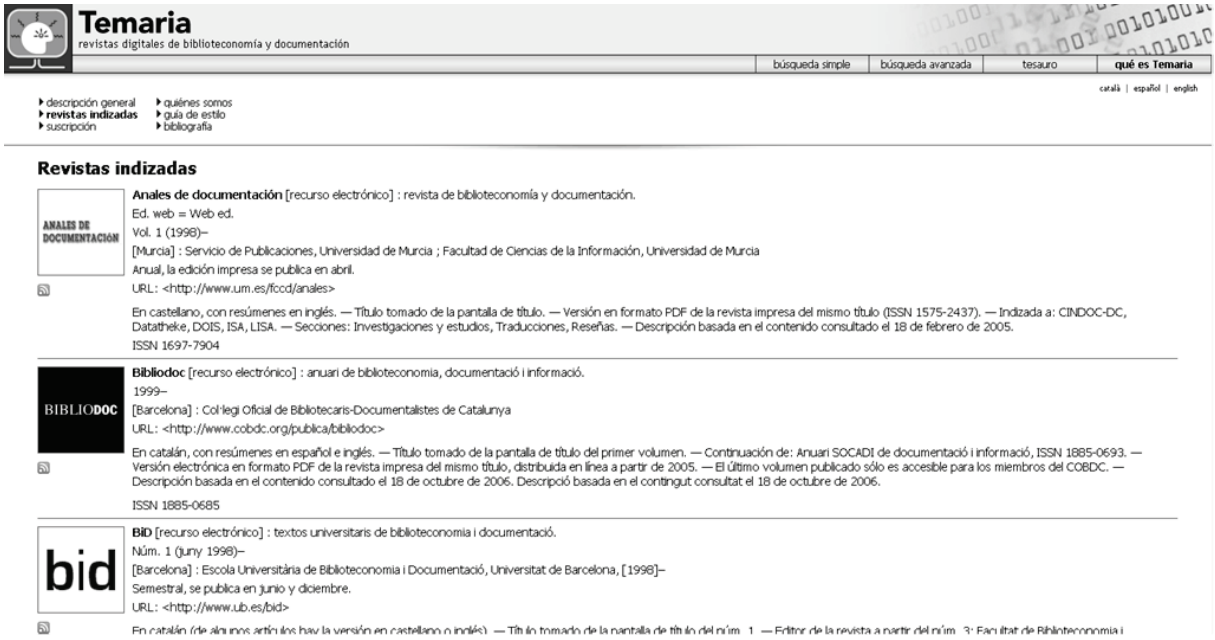
En el proyecto Temaria los elementos anteriores, a excepción de Fuente y Cobertura, se han establecido como elementos obligatorios; esto es, se han de anotar siempre que aparezcan en el recurso descrito. El elemento Fuente no se utiliza en el contexto del proyecto y el elemento Cobertura se ha designado como opcional.

En el formulario desarrollado para asignar metadatos a los recursos descritos, los elementos anteriores se han ordenado en una secuencia, pero se pueden introducir en el formulario en el orden que el usuario desee. Más adelante se dan directrices sobre el contenido del conjunto de elementos DC.

Los elementos del DC se pueden emplear solos —DC simple— o con calificadores —DC calificado. Los calificadores tienen dos funciones: o bien ayudan a precisar el contenido de un elemento —calificadores de refinamiento— o bien identifican los estándares y los lenguajes controlados empleados para asignar el contenido de un elemento —esquemas de codificación. En el proyecto Temaria se ha optado por el DC calificado, y se usan, siempre que es posible, los calificadores definidos en el documento DCMI metadata terms. También se usan calificadores locales más adecuados al entorno del proyecto. La tabla 1 resume los calificadores recomendados por la DCMI y los que se utilizan en el proyecto.

Elementos DC	Refinamientos de los elementos		Esquemas de codificación	
	DCMI	Temaria	DCMI	Temaria
Title	alternative	—	—	—
Creator	—	personalName corporateName	—	—

Título: Directorio de revistas sindicadas por Temaria
Referencia: figura13



Temaria
revistas digitales de biblioteconomía y documentación

búsqueda simple | búsqueda avanzada | tesauro | qué es Temaria

català | español | english

descripción general | revistas indizadas | suscripción | quiénes somos | guía de estilo | bibliografía

Revistas indizadas

Anales de documentación [recurso electrónico] : revista de biblioteconomía y documentación.
Ed. web = Web ed.
Vol. 1 (1998)–
[Murcia] : Servicio de Publicaciones, Universidad de Murcia ; Facultad de Ciencias de la Información, Universidad de Murcia
Anual, la edición impresa se publica en abril.
URL: <http://www.um.es/fccd/anales>

En castellano, con resúmenes en inglés. — Título tomado de la pantalla de título. — Versión en formato PDF de la revista impresa del mismo título (ISSN 1575-2437). — Indizada a: CINDOC-DC, Datatheke, DOIS, ISA, LISA. — Secciones: Investigaciones y estudios, Traducciones, Reseñas. — Descripción basada en el contenido consultado el 18 de febrero de 2005.
ISSN 1697-7904

Bibliodoc [recurso electrónico] : anuari de biblioteconomia, documentació i informació.
1999–
[Barcelona] : Col·legi Oficial de Bibliotecaris-Documentalistes de Catalunya
URL: <http://www.cobdc.org/publica/bibliodoc>

En catalán, con resúmenes en español e inglés. — Título tomado de la pantalla de título del primer volumen. — Continuación de: Anuari SOCADI de documentació i informació, ISSN 1885-0693. — Versión electrónica en formato PDF de la revista impresa del mismo título, distribuida en línea a partir de 2005. — El último volumen publicado sólo es accesible para los miembros del COBDC. — Descripción basada en el contenido consultado el 18 de octubre de 2006. Descripció basada en el contingut consultat el 18 de octubre de 2006.
ISSN 1885-0685

BiD [recurso electrónico] : textos universitaris de biblioteconomia i documentació.
Núm. 1 (juny 1998)–
[Barcelona] : Escola Universitària de Biblioteconomia i Documentació, Universitat de Barcelona, [1998]–
Semestral, se publica en junio y diciembre.
URL: <http://www.ub.es/bid>

En catalán (de algunos artículos hay la versión en castellano o inglés). — Título tomado de la pantalla de título del núm. 1. — Editor de la revista a partir del núm. 3: Facultat de Biblioteconomia i

Además de los aspectos comentados en torno al método de almacenamiento y procesamiento de la información correspondiente a los artículos de las revistas, también hay que recalcar una

importante aplicación, que dota de nuevas prestaciones a la recuperación de información en su modalidad. Se trata del uso de la URL, como buscador improvisado de información de una fuente o canal de sindicación dado, v*éase figura14*.

Título: Directorio de fuentes de sindicación de revistas
Referencia: figura14

Imagen del directorio de revistas sindicadas de Temaria. Obsérvese la organización por título, autor y materia, así como la aplicación de sintaxis adjunta a la url para la recuperación de parámetros concretos que el usuario requiera, como apellido del autor, o materia normalizada según tesauro al uso.

Analizando el documento y la bibliografía existente, se comprueba el uso de sintaxis específica para la recuperación del sumario del último número de una revista, el seguimiento del autor o un tema, *véase la tabla54*.

Título: Sintaxis de recuperación mediante url
Referencia: tabla 54

Autor determinado:
http://temaria.net/metadatos.php?format=rss&aulast=***autor***

Materia determinada:
http://temaria.net/metadatos.php?format=rss&subject=***materia***

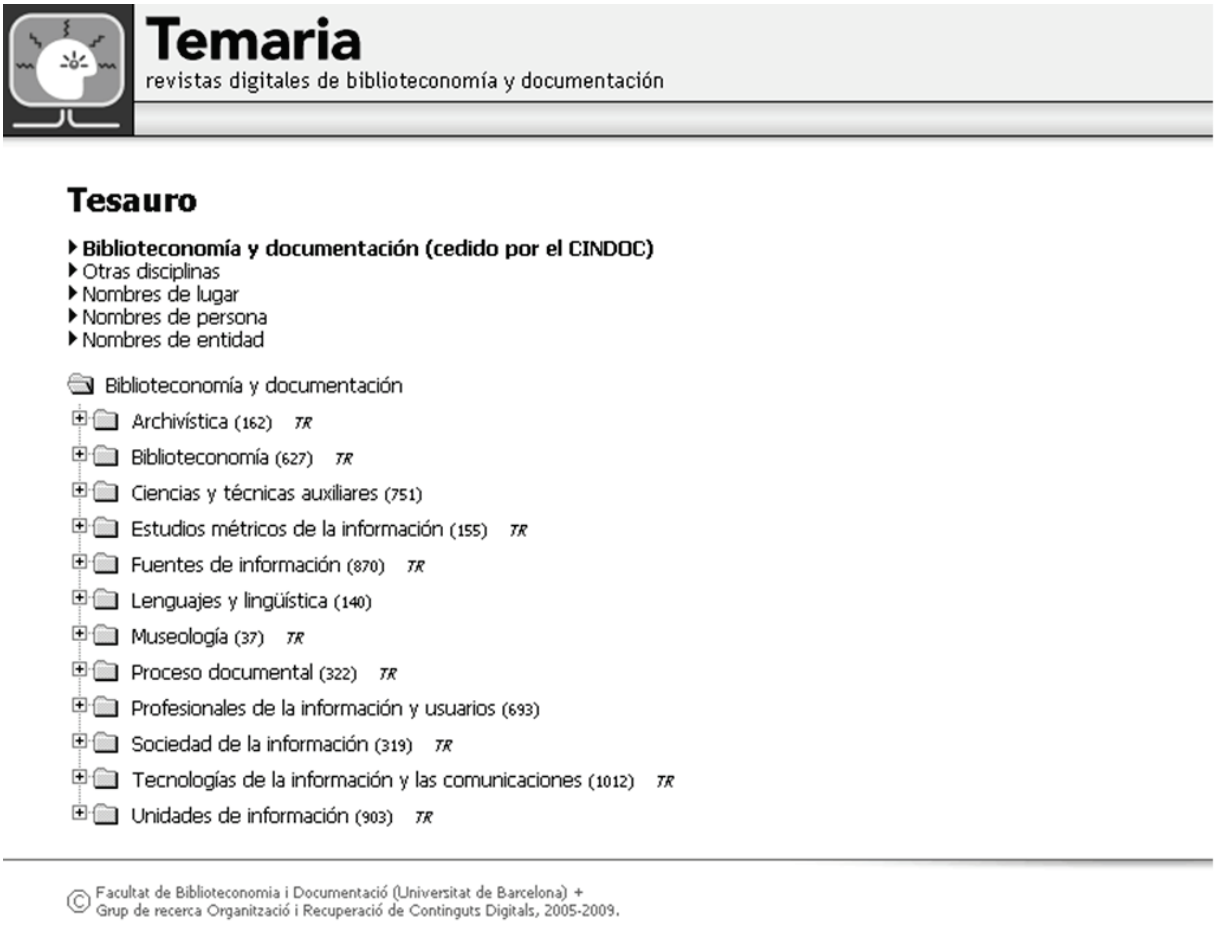
Sumario del último número de una revista:
http://temaria.net/metadatos.php?format=rss&issn=***nnnn-nnnn***&***actual***=***s***

Cadena diseñada por el usuario:
http://temaria.net/metadatos.php?format=rss&issn=***1697-7904***&author=***S***&subject=***bibliotecas***

La recuperación de información a través de URL, opción que permite el portal de revistas Temaria, además de ser una característica avanzada en cuanto a técnicas de recuperación se refiere, justifica en sí mismo, el empleo de metadatos Dublin Core, mediante los cuales se consigue definir el propio canal o fuente de sindicación, concretamente mediante su ISSN como principal campo de recuperación. No obstante es susceptible de utilizar cualquier otro dispuesto en el esquema de descripción señalado en la *tabla54*, mediante el operador *"&"* ampersand, que permite crear auténticas cadenas de consulta. Por ejemplo en la *tabla54* se puede comprobar en la cadena diseñada por el usuario que designa, el número de ISSN correspondiente a la revista en la que se quiere realizar la consulta concreta, buscando sobre todos los autores del sumario que hayan escrito artículos cuya materia clasificatoria sea bibliotecas. El funcionamiento de la consulta es sencillo, la información es enviada como consulta y filtrada directamente por medio de los metadatos Dublin Core. Por lo que se selecciona la revista cuyo ISSN coincida con el requerido, se filtra el sumario y la materia. El proceso más complicado en todo el filtrado corresponde precisamente a la materia. Téngase en cuenta que la información a filtrar no sólo se encuentra en los metadatos, los descriptores válidos para la clasificación se encuentran en el tesauro electrónico de biblioteconomía y documentación, en torno al cual están clasificados todas las publicaciones de la revista. Por tanto el filtrado puede ser sencillo si lo hace respecto del propio documento previamente

clasificado en cuyo caso podrá o no coincidir con la terminología en lenguaje natural del usuario, o llevar a cabo un proceso de aproximación de términos mediante el tesauro para su posterior recuperación en la colección de contenidos de la revista, *véase figura15*.

Título: Tesauro de Temaria
Referencia: figura15



Tesauro para la normalización de las categorías y materias en la clasificación de las revistas de biblioteconomía y documentación. También se encuentran normalizadas las entradas de nombres de persona y entidad, así como de lugar, pudiendo acceder a los artículos sindicados mediante la navegación de todos los elementos descritos.

Sindicación de contenidos en bitácoras profesionales

Otra aplicación de la sindicación unida a la creación de nuevos servicios y comunidades virtuales de profesionales, son los blogs o bitácoras especializadas. Si bien al inicio del desarrollo de las primeras bitácoras, eran consideradas sitios web con sistema de publicación automatizado, basado en un sencillo editor de textos, en la actualidad, se han convertido en verdaderas herramientas de gestión de la información, con características especiales para la clasificación de la información publicada, las técnicas de sindicación, la citación digital de determinados artículos y contenidos, entre otros. De hecho, salvando las distancias los CMS de gestión de contenidos, *herramientas específicas para la gestión y el enrutamiento de la producción documental de una corporación o entidad, han quedado equiparados a las posibilidades que pueden ofrecer los blogs o bitácoras* (FRANGANILLO, J. and Catalán, M.A., 2005). Otras propiedades como el valor añadido que el editor y autor de las mismas desarrolla no sólo con sus contenidos, investigaciones, artículos, sino mediante sus complementos de recursos especializados en un área del conocimiento, permiten el ya conocido fenómeno de las comunidades virtuales de profesionales especializados en un área concreta de conocimiento. El resultado de este fenómeno es la existencia de una importante cantidad de recursos con canales de sindicación, susceptibles de ser recopilados, agrupados y clasificados por las bibliotecas y unidades de información y documentación, sin olvidar de aplicar criterios de calidad en la forma y en el contenido de la información publicada. Conociendo estos recursos que están siendo generalizados para todas las instituciones y entidades por su sencillez de mantenimiento y gestión, hacen que bibliotecas sin sitios web oficiales puedan disponer de una representación directa mediante su bitácora, que se adapta a las necesidades de su entorno profesional. En España este fenómeno está siendo asumido de manera tardía, aunque se están dando pasos en el cambio por incluir a los blogs como un servicio más de la biblioteca digital según los últimos debates, que se sucedieron en las *X Jornadas Españolas de Documentación FESABID 2007*. En contraposición al panorama que se sucede en España, en los países anglosajones ya están empleando la bitácora como un sistema más de difusión de ideas, actividades, contenidos complementarios de sus correspondientes sedes web. Un ejemplo de ello es la bitácora de la cabecera bibliotecaria de Estados Unidos, la *Library of Congress* (Library of Congress Blog , 2006), que presenta su sitio web general que articula todos los servicios bibliotecarios, entre ellos la bitácora, que dispone de entidad propia.

Así pues las bitácoras están llamadas a convertirse en un servicio más de la biblioteca digital tanto por su desarrollo institucional como por la introducción de comunidades virtuales de profesionales que dan lugar a un nuevo tipo de fuentes de información que de otra manera sería imposible aglutinar en la biblioteca física, y se trata de las fuentes personales que representan dichas comunidades.

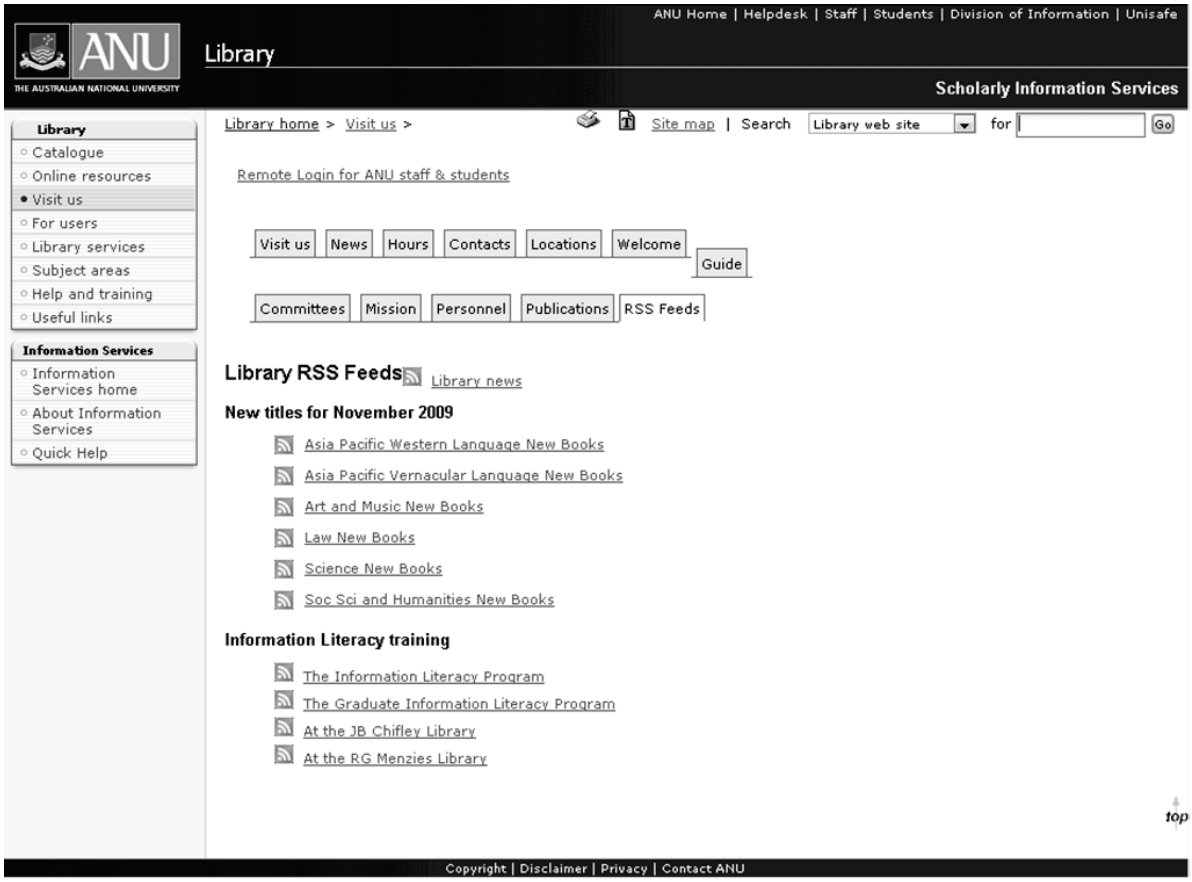
En el caso de la biblioteconomía y la documentación, así como la archivística, existe una comunidad, denominada generalmente *Blogosfera documental* (BECERRIL GONZÁLEZ, V. et al., 2006), arraigada en la publicación y difusión de contenidos especializados mediante sindicación, que podrían ser aprovechados por las principales bibliotecas especializadas en la materia, a modo de recursos anexionando cada vez más en anillos web. Esta referenciación electrónica y visual es necesaria para el establecimiento de las bases de una sólida comunidad que trabaje, investigue y produzca más información y documentación científica, retrospectiva y explicativa de los hechos y avances acontecidos.

Sindicación de novedades bibliográficas: El caso de ANU

Constituye una de las primeras incursiones de la sindicación en la representación de verdaderos boletines de novedades bibliográficas similares y paralelas a los existentes en cualquier panel de anuncios de una biblioteca física. Este servicio, dadas sus características entronca con la difusión selectiva de la información que se produce en las bibliotecas con este tipo de materiales. No obstante también hay que ser cautos, ya que las aplicaciones visibles hasta la fecha, se reducen a las novedades bibliográficas, no estando ampliadas a noticias y novedades que se publican en revistas y recursos como bitácoras profesionales sobre las cuales realizar vaciados y transmitirlas a los usuarios atendiendo a perfiles de interés y afinidad en las necesidades de información y documentación. El funcionamiento de estas aplicaciones se basa en generar la fuente de sindicación a partir del catálogo colectivo, que recibe las altas bibliográficas correspondientes. Si el sistema de gestión bibliotecario que gestiona el catálogo está adaptado, crea el canal de sindicación correspondiente siendo incluido en el sitio web oficial de la biblioteca. De esta manera los usuarios acceden a la información directamente pudiendo realizar un seguimiento de todas las novedades catalogadas por la biblioteca.

Existe un ejemplo fundamental para ver esta aplicación en funcionamiento. Se trata de la Biblioteca de la Universidad Nacional de Australia que aporta todos los canales o fuentes de sindicación de datos e información bibliográfica atendiendo a criterios de lengua y clasificación temática, *véase figura16.*

Título: Portal de novedades bibliográficas de la Biblioteca de la Universidad Nacional de Australia
Referencia: figura 16



Capítulo 5. Los formatos de sindicación

Los formatos de sindicación son lenguajes de marcado basados en XML con los que se conforma la estructura de un canal de sindicación y sus elementos. El objetivo de un formato de sindicación es representar un contenido de la mejor forma posible para ser transmitido y difundido. A este objetivo hay que añadir que los formatos de sindicación tradicionales como RSS1.0, RSS2.0 y Atom constan de estructuras jerárquicas sencillas puesto que fueron diseñados para transmitir noticias, artículos y contenidos publicados en la web. Para estos efectos inicialmente no eran necesarias etiquetas especializadas para distinguir datos especiales. Pero esta tendencia empezó a cambiar conforme se fue ampliando la complejidad de los tipos de documentales publicados y su entorno de aplicación. Un ejemplo claro de este cambio lo suponen las revistas electrónicas y la necesidad de controlar parámetros como la periodicidad de la publicación, las páginas en las que se encuentra un determinado artículo, los números de identificación de publicaciones seriadas, el volumen, número o ejemplar de la revista, su fecha de publicación, etc. Todos ellos constituían elementos que no habían sido tenidos en cuenta en tales formatos de sindicación y tal caso demostraba que conforme a la disposición de un tipo documental como el artículo científico, los formatos de sindicación no estaban preparados para su completa estructuración, evitando pérdidas de datos.

Si bien los formatos de sindicación por sí solos atendiendo a su estructura básica no eran capaces de representar toda la información, utilizaron una de las propiedades fundamentales de la sindicación basada en XML, la propiedad de extensibilidad, para solucionar dichos problemas. De hecho la extensibilidad de los formatos de sindicación de contenidos permite el empleo de terceros formatos de sindicación o módulos debidamente validados a través de la declaración de su namespace o espacio de nombres en el que se encuentra su schema XSD o lo que es lo mismo las reglas de construcción de dicho lenguaje, sus etiquetas, atributos y características propias.

Esta capacidad propia de los lenguajes de marcado derivados de XML ha sido ampliamente aprovechada por los formatos de sindicación que pueden emplear módulos especializados para resolver por ejemplo el problema de la descripción de artículos de revistas científicas.

Conforme a lo explicado, los formatos de sindicación originalmente no fueron concebidos para describir tipos documentales especializados, sino más bien una generalidad de tipos documentales enmarcados en la publicación web, lo cual ha supuesto múltiples cambios,

mejoras y avances conforme a los formatos originales. Un ejemplo claro de ello es el formato RSS que en total ha sumado ocho versiones distintas desde su creación en 1999 hasta el año 2003. Todos estos cambios de versión han propiciado un continuo proceso de adaptación por parte de los navegadores web que soportaban la sindicación como método de redifusión de la web, implicando constantes cambios en los mecanismos parser de interpretación, lectura y agregación de contenidos.

Por otro lado hay que advertir el empleo del concepto formato de sindicación como un término adscrito únicamente a unos formatos determinados RSS1.0, RSS2.0 y Atom, *véase tabla55 de formatos de sindicación*. Esto no es del todo correcto y resulta importante desentrañar que hay de verdad en hacer equivaler formato de sindicación a tales lenguajes.

Si bien un formato de sindicación es aquel lenguaje XML que permite llevar a cabo los procesos de la sindicación de contenidos, es decir, la transmisión de información estructurada a múltiples usuarios, tan formato de sindicación es RSS o Atom como MARC-XML o cualquier otro que validado. Por tanto la diferencia entre los formatos de sindicación tradicionales y aquellos menos conocidos radica precisamente en su presencia o ausencia en los navegadores web y por ende que su canal pueda ser actualizado con una frecuencia determinada, siendo legible y analizable por parte de las aplicaciones parser. Dicho esto se determinan unas características muy claras que distinguen qué formatos basados en XML pueden emplearse a efectos de sindicación:

1. Transmitir información y contenidos estructurados en XML bien formado y validado.
2. Disponer de un canal de información cuyo formato contemple un catálogo de entradas, elementos, ítems o registros.
3. Actualización periódica del canal de información.
4. Permitir la suscripción o representación del canal mediante lector parser, agregador de canales, o documento de estilo y transformaciones XSLT.
5. Posibilitar su integración web en terceros recursos.

Por lo tanto se consideran formatos de sindicación de contenidos aquellos basados en una sintaxis XML y cuyos archivos están bien formados, validados, correctamente estructurados, dando cabida a la información del canal y agrupando a modo de catálogo o colección una serie de elementos, ítems o registros unívocamente diferenciados y cuyo contenido informativo y documental resulta primario, secundario o terciario.

Según estos aspectos y a tenor de la investigación que se llevar a cabo, se demostrará que es posible utilizar formatos basados en XML para la sindicación de contenidos bibliográficos cumpliendo con los preceptos señalados, lo que implica la introducción del formato MARC-XML como formato de sindicación.

Título: Tabla de formatos de sindicación.
Fuente: (SHEPHERD, E., 2007)
Referencia: tabla55

Formato	Estado	Fecha	Base	Autor
RSS0.90	Obsoleto	1999	XML-RDF	Netscape
Disponible en: http://www.rssboard.org/rss-0-9-0				
RSS0.91	Obsoleto	1999	XML	Netscape
Disponible en: http://www.rssboard.org/rss-0-9-1-netscape				
RSS0.91	Obsoleto	1999	XML	UserLand
Disponible en: http://www.rssboard.org/rss-0-9-1				
RSS0.92	Obsoleto	2000	XML	UserLand
Disponible en: http://www.rssboard.org/rss-0-9-2				
RSS0.93	Obsoleto	2001	XML	UserLand
Disponible en: http://backend.userland.com/rss093				
RSS0.94	Obsoleto	2002	XML	UserLand
Especificación no disponible.				
RSS1.0	Estándar	2000	RDF	RSS-DEV Working Group
Disponible en: http://web.resource.org/rss/1.0/spec				
RSS2.0	Estándar	2003	XML	UserLand
Disponible en: http://www.rssboard.org/rss-2-0				
RSS2.0.11	Estándar	2009	XML	UserLand
Disponible en: http://www.rssboard.org/rss-specification				

Atom0.2	Obsoleto	2003	XML	APE(Atom/Pie/Echo) Working Group
Disponible en: http://wearehugh.com/public/2003/08/atom02spec.txt				
Atom0.3	Obsoleto	2003	XML	APE(Atom/Pie/Echo) Working Group
Disponible en: http://www.mnot.net/drafts/draft-nottingham-atom-format-02.html				
Atom1.0	Estándar	2005	XML	IETF AtomPub Working Group
Disponible en: http://www.ietf.org/rfc/rfc4287.txt				
OPML1.0	Obsoleto	2000	XML	Dave Winer
Disponible en: http://www.opml.org/spec				
OPML2.0	Estándar	2007	XML	Dave Winer
Disponible en: http://www.opml.org/spec2				
MARC-XML	Estándar	2001	XML	The Library of Congress' Network Development and MARC Standards Office
Disponible en: http://www.loc.gov/standards/marcxml/				

5.1. Atom

El formato Atom fue desarrollado a partir del año 2003 como resultado de una propuesta de Sam Ruby, por crear un formato de sindicación que mejorara la experiencia de descripción de los contenidos web. Es importante señalar que la iniciativa fue propiciada por la disconformidad con el formato RSS, que resultaba en muchos casos demasiado sencillo e ineficaz para representar contenidos publicados en una web cada vez más dinámica (RUBY, S., 2003), por ejemplo la posibilidad de incluir comentarios, las fuentes de origen, autores, colaboradores de los contenidos, etc. Esta idea fue expuesta en el portal *Intertwingly* del propio Sam Ruby, obteniendo una gran cantidad de adhesiones de desarrolladores para trabajar en el desarrollo de dicho formato. Se tiene constancia de la participación activa de algunos de los autores del formato de sindicación RSS1.0 y RSS2.0 como Aaron Swartz, Mark Pilgrim, Tim Bray o incluso el propio Dave Winer creando en conjunto como grupo de trabajo APE(Atom/Pie/Echo), la primera hoja de ruta para desarrollar el formato Atom (RUBY, S. and Hopkins, D., 2007). En este documento se expresan algunas de las directrices distintivas del nuevo formato: 1) Neutralidad de desarrollo con independencia de las compañías que suministren y proporcionen soporte al formato. 2) libre utilización del formato por todo el mundo. 3) Libre extensibilidad del formato por cualquier con cualquier otro formato o módulo. 4) Constituido por un lenguaje limpio y bien formado.

El grupo de trabajo APE, así denominado por las distintas líneas de desarrollo y concepción del formato (Atom, Pie, Echo), desembocó en el desarrollo de la versión 0.2 de prueba del formato Atom, todavía en un estado muy primario, carente de DTD, validación y estructuración clara a tenor de lo analizado en sus especificaciones (APE(ATOM/PIE/ECHO) WORKING GROUP, 2003). Este primer resultado no disponía de protocolos y herramientas de edición o actualización del canal de sindicación, tal y como fue delimitado en la hoja de ruta. Estos avances se conseguirán con una versión más normalizada, Atom0.3, cuyas especificaciones quedan perfectamente aclaradas (APE(ATOM/PIE/ECHO) WORKING GROUP, 2003). A partir de entonces Atom empezó a ganar importancia al ser adoptado como principal formato de sindicación adoptado por Google para la representación de los contenidos publicados en su división de blogs Blogger (Atom API Documentation for Blogger, 2006), extendiéndose su uso a terceras aplicaciones de importancia como Google News o incluso el propio Gmail, posibilitando la sindicación de noticias y correos electrónicos mediante canales en formato Atom.

A pesar de todo, Atom0.3 no era reconocido como lenguaje normalizado, por lo que se transfirió su desarrollo al IETF Atom Pub Working Group, constituido por los mismos miembros del equipo de trabajo anterior, dirigidos por Tim Bray y Paul Hoffman. Los objetivos esenciales de este nuevo equipo era la creación de un espacio de nombres propio sobre el asentar un formato de sindicación más maduro y completo. Ello se logró en el año 2005 con Atom1.0 logrando el reconocimiento del Internet Engineering Task Force y el W3C Consortium como un estándar de sindicación de contenidos (IETF ATOMPUB WORKING GROUP, 2005).

5.1.1. Estructura del formato Atom

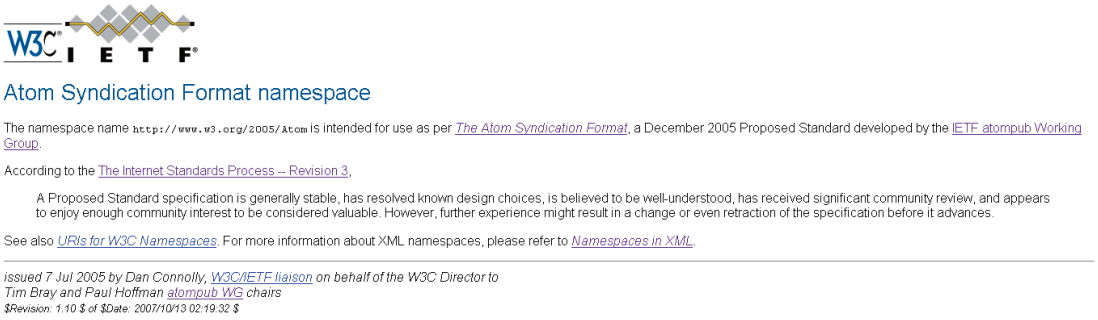
Atom es un formato basado en XML para la descripción general de contenido web que a partir del protocolo HTTP y sus métodos POST y GET permite la actualización de los contenidos publicados en el canal. Atom también cumple con las condiciones de cualquier formato de sindicación en relación a constituir un canal de sindicación con un archivo XML bien formado y validado. Además su identificación y enlace se efectúa con un tipo MIME especial "application/atom+xml", empleándose comúnmente en los vínculos de los sitios web de la siguiente forma, *véase tabla56.*

Título: Enlace predeterminado para un canal de sindicación Atom.
Referencia: tabla56

```
<link rel="alternate" type="application/atom+xml" title="titulo-del-canal" href="url-del-canal" />
```

Atom consta además de un espacio de nombres propio que normaliza el formato y los elementos que emplea en todas sus estructuras. Constituyen una parte importante de las especificaciones del lenguaje. Originalmente en la declaración de espacio de nombres namespace se utiliza por convención la siguiente URL del W3C Consortium http://www.w3.org/2005/Atom.

Título: Namespace del formato Atom
Referencia: figura17



Atom Syndication Format namespace

The namespace name `http://www.w3.org/2005/Atom` is intended for use as per *The Atom Syndication Format*, a December 2005 Proposed Standard developed by the IETF atompub Working Group.

According to the The Internet Standards Process -- Revision 3,

> A Proposed Standard specification is generally stable, has resolved known design choices, is believed to be well-understood, has received significant community review, and appears to enjoy enough community interest to be considered valuable. However, further experience might result in a change or even retraction of the specification before it advances.

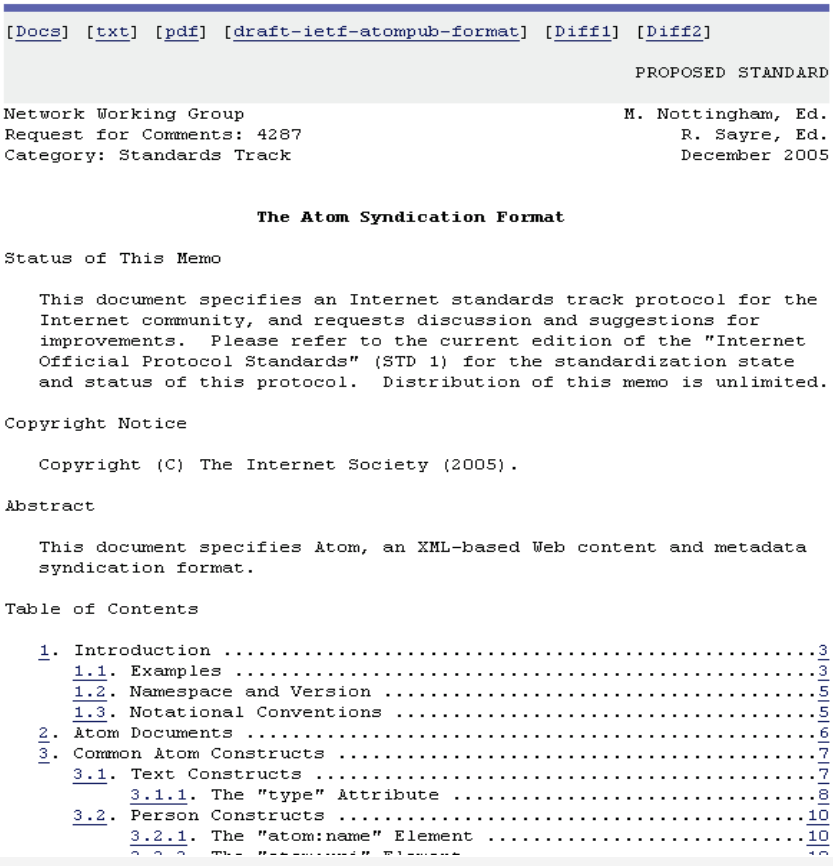
See also *URIs for W3C Namespaces*. For more information about XML namespaces, please refer to *Namespaces in XML*.

---

issued 7 Jul 2005 by Dan Connolly, W3C/IETF liaison on behalf of the W3C Director to
Tim Bray and Paul Hoffman atompub WG chairs
$Revision: 1.10 $ of $Date: 2007/10/13 02:19:32 $

Es importante señalar que de entrada, no contiene las especificaciones del formato a modo de Schema XSD como sería lógico esperar, pero sí dispone de un enlace que a su vez que redirige a las especificaciones oficiales del formato Atom http://tools.ietf.org/html/rfc4287.

Título: Especificaciones originales del formato Atom
Referencia: figura18



[Docs] [txt] [pdf] [draft-ietf-atompub-format] [Diff1] [Diff2]

PROPOSED STANDARD

Network Working Group — M. Nottingham, Ed.
Request for Comments: 4287 — R. Sayre, Ed.
Category: Standards Track — December 2005

The Atom Syndication Format

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2005).

Abstract

This document specifies Atom, an XML-based Web content and metadata syndication format.

Table of Contents

Tales especificaciones tampoco constituyen un schema XSD o una DTD correspondiente al formato, considerándose más un manual de construcción y edición del formato, que unas reglas sintácticas definidas para actuar en correlación con el formato. Dicho esto la principal conclusión que se extrae de este apartado es que la normalización del formato Atom es puramente teórica y oficial por el reconocimiento de las principales instituciones de normalización de la web como el W3C y IETF, pero no práctica como se podría comprobar mediante el empleo de un schema.

En cuanto a la utilización de estándares y normas añadidas, el formato Atom utiliza un formato de fecha y hora de internet normalizado por la norma RFC3339 (KLYNE, G. and Newman, C., 2002).

En relación al tipo de codificación del contenido, por regla general Atom utiliza siempre texto plano excepto cuando se delimita claramente en los atributos de cada elemento que el contenido está codificado. En general cualquier contenido del formato puede tener una codificación text, html o xhtml, según lo que establecen las especificaciones.

Apertura y cierre del formato

El formato Atom como cualquier XML, siempre comienza con una primera línea de declaración del lenguaje XML *<?xml version="1.0" encoding="UTF-8"?>* necesaria para identificar que el archivo utiliza las reglas de marcado de XML, así como el set de caracteres empleado. Aunque resulte una instrucción banal, como se explicará en sucesivos capítulos, esta declaración es necesaria para considerar un documento XML bien formado y validado, además de contener información esencial del set de caracteres que juega un papel fundamental para la correcta visualización de los contenidos y su posterior tratamiento.

A continuación se conforma la estructura del formato Atom propiamente dicho, compuesto por dos partes fundamentales; el *feed o canal* y el *entry o ítem*. El feed es el primer elemento del formato y acoge como descendientes a los elementos entry que componen los contenidos que se sindican. Se trata de una etiqueta de apertura y cierre *<feed></feed>* por lo que encabeza y finaliza el formato. Las entradas del canal de sindicación también emplean una solución de apertura y cierre *<entry></entry>* que embeben a su vez el contenido, configurándose cuantas sean necesarias.

Título: Esquema de apertura y cierre del formato
Referencia: tabla57

```
<?xml version="1.0" encoding="UTF-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
…
   <entry>
   …
   </entry>
…
</feed>
```

El elemento *feed* es por tanto el elemento padre del que dependen todos los demás y tiene como objetivo contener tanto la información propia del canal de sindicación como las entradas o ítems que constituyen los contenidos. También consta de un atributo obligatorio concretamente, el namespace o espacio de nombres correspondiente al formato Atom, que resulta obligatorio para la identificación de la estructura para el protocolo Atom o para cualquier parser de canales de sindicación, *<feed xmlns="http://www.w3.org/2005/Atom">*.

Definidos estos aspectos se consideran etiquetas del elemento feed aquellas que definen la información básica del canal de sindicación como su título, descripción, autor, fecha de publicación o enlace, configurándose los siguientes elementos:

Elementos de descripción del canal

- **Identificador – `<id></id>`.** El elemento *id* permite identificar de forma unívoca el sitio web del canal de sindicación que actúa a modo de fuente de información, para ello suele emplearse su URI (Universal Resource Identifier), por constituir un vínculo permanente de acceso directo.

- **Título – `<title></title>`.** *Title* es el título completo propiamente dicho del canal.

- **Fecha de actualización – `<updated></updated>`.** El elemento *updated* configura la fecha de actualización del canal, por lo que su valor varía según la regeneración de la fuente de información. Su formato es el establecido por la norma RFC3339 anteriormente referido.

- **Autor – <author></author>.** El elemento *author* está pensado para contener la información correspondiente a una autoridad personal responsable del canal de sindicación. Por ello, consta a su vez de una serie de elementos anidados para su descripción, a saber:
  - **Nombre – <name></name>.** forma completa del nombre de la persona.
  - **Correo – <email></email>.** Correo electrónico de la persona.
  - **Identificador – <uri></uri>.** identificador universal de la persona, su número de autoridad o más comúnmente la url de su sitio web.

  (Tales elementos son comunes para cualquier otro elemento que implique persona o autoridad como puede ser el elemento colaborador <contributor></contributor>)

- **Enlace – <link rel="" href="" type="" hreflang="" title="" length=""/>.** El elemento link lo constituye una sola etiqueta de apertura y cierre y su papel es fundamental en el canal ya que identifica la url completa al archivo XML del canal de sindicación y el tipo de relación del enlace con el propio canal. Estos aspectos se definen en sus atributos de la siguiente forma:
  - **Relación – rel="".** El atributo rel determina el tipo de relación que existe entre el enlace y el canal de sindicación. Por tanto si contiene una URL o enlace al canal de sindicación tomará el valor *self*, puesto que se identifica a sí mismo. El empleo del atributo rel es opcional pero importante en combinación con href.
  - **Dirección URL – href="".** El atributo href se emplea para definir el enlace o URL correspondiente al canal de sindicación Atom. Debe asegurarse que el contenido enlazado es el propio archivo XML empleado para la sindicación del contenido. La utilización del atributo href es obligatoria para definir el enlace.
  - **Tipo – type="".** El atributo type define el tipo de información que contiene el enlace href, pudiendo definir tipos MIME, en el caso de definir el propio canal de sindicación type toma el valor *application/atom+xml*, tal y como aclaran las especificaciones. Su empleo es
  - **Idioma – hreflang="".** Permite definir en qué idioma se encuentra el contenido enlazado por la URL. Por regla general suele emplearse la norma ISO 639-1 de códigos de idioma que los define con dos caracteres.
  - **Título – title="".** El atributo title define el título del enlace, facilitando su comprensión de cara a su representación y lectura.

- **Tamaño – length="".** El atributo length define el tamaño o extensión del contenido enlazado por la URL en bytes. Por tanto su información es de tipo numérico.

- **Temática – <category term="" scheme="" label=""/>.** Al igual que el elemento link, category está formado por una sola etiqueta que contiene los atributos necesarios para definir la temática del canal.
  - **Término – term="".** Contiene el identificador o identificadores temáticos que permiten clasificar el canal de sindicación. Normalmente lo constituye un descriptor temático o las palabras clave. Es un atributo de uso obligado si se emplea el elemento category.
  - **Esquema – scheme="".** Define la URI del esquema de categorías del que se ha tomado el identificador temático. Suele emplearse cuando existen lenguajes controlados como ontologías o tesauros basados en XML. El empleo de este atributo es opcional.
  - **Etiqueta – label="".** Define una etiqueta que o bien describe el término empleado para clasificar el canal o bien sirve como término legible y representable. Esto sucede cuando el atributo term, en vez de contener un texto como un descriptor, alberga un código de identificación numérico o alfanumérico correspondiente a tal término. El empleo de este atributo al igual que scheme, también es opcional.

- **Colaborador – <contributor></contributor>.** El elemento *contributor* identifica a los colaboradores del canal de sindicación. Al igual que el elemento *author*, dispone de una estructura de subelementos para definir el nombre completo, el correo electrónico y la URI correspondiente al identificador de la persona en una lista de autoridades o de su sitio web.
  - **Nombre – <name></name>.** forma completa del nombre de la persona.
  - **Correo – <email></email>.** Correo electrónico de la persona.
  - **Identificador – <uri></uri>.** identificador universal de la persona, su número de autoridad o más comúnmente la url de su sitio web.

- **Generador – <generator uri=" " version=""></generator>.** El elemento *generator* identifica el nombre del programa que generó el canal de sindicación. Normalmente este suele coincidir con los sistemas de publicación web que incorporar sus propios procesos de

edición y publicación de canales de sindicación. Además como atributos opcionales permite declarar la URI o URL del programa generador del canal y su versión.

- **Identificador – uri="".** Atributo que contiene el enlace identificador universal del programa generador de canales Atom.
- **Versión – version="".** Versión del programa empleado para generar el canal.

- **Icono – <icon></icon>.** Permite identificar un icono que se utiliza a modo de *favicon* del propio canal de sindicación. Esto se efectúa introduciendo la URL correspondiente a la imagen que configura el icono en formato *.ico* ó *.png*.

- **Logotipo – <logo></logo>.** El elemento logo contiene la URL a una imagen que actúa como logotipo identificador del canal, de forma que sea visible en los lectores de canales de sindicación

- **Derechos – <rights></rights>.** Contiene los derechos de autor o consideraciones de propiedad intelectual sobre los contenidos del canal de sindicación, de forma que permite definir una norma en el tratamiento de los contenidos por terceros.

- **Subtítulo – <subtitle></subtitle>.** El elemento *subtitle* no solo hace referencia al empleo de subtítulos, ya que también permite descripciones del canal de sindicación. Esta característica plasmada en las especificaciones del formato hace que resulte especialmente ambiguo su empleo, constituyendo un fallo importante en el diseño del formato. Por este motivo sigue siendo más aconsejable la introducción únicamente del subtítulo.

Elementos de entry

Una vez definido un canal con los elementos propios de *feed*, se suceden consecutivamente las entradas o contenidos propiamente dichos. Éstos están anidados en la estructura *<feed></feed>* conformando el cuerpo del canal de sindicación con los siguientes subelementos:

- **Identificador – <id></id>.** El elemento *id* enmarcado en una entrada es el identificador universal de la misma. Normalmente se utiliza la URL absoluta del contenido que se está

representando.

- **Título – <title></title>.** Contiene el título completo de la entrada o contenido.

- **Fecha de actualización – <updated></updated>.** El elemento updated al igual que en el canal representa la fecha de actualización de la entrada, que hay que diferenciar de la fecha de publicación de la misma. En todos los casos el formato de datos aceptado para estos elementos es el que viene establecido por la norma RFC3339.

- **Autor – <author></author>.** Contiene la autoridad personal responsable del contenido descrito en la entrada. Tal y como se ha especificado anteriormente el elemento consta de subelementos para definir el nombre completo, el correo electrónico y su uri de identificación como autoridad normalizada o la url correspondiente a su sitio web particular.

- **Contenido – <content type="" src=""></content>.** El elemento *content* permite introducir todo o parte del contenido que ha sido referenciado por la entrada. Resulta habitual definir un atributo *type* opcional que determina el tipo de codificación del contenido.
  - **Tipo – type="".** El atributo type es opcional pero recomendado si el texto contenido está codificado, permitiendo su correcta visualización e interpretación por cualquier parser. Permite definir la codificación del contenido como *text*, *html* y *xhtml*.
  - **Dirección URL – src="".** Atributo opcional que posibilita la introducción de una URL absoluta a modo de URI que representa la fuente original en la que fue encontrado el contenido. Si se emplea el atributo *src* junto con *type* el significado de *type* varía por el de tipo MIME del contenido del enlace.

- **Enlace – <link rel="" href="" type="" hreflang="" title="" length=""/>.** El elemento link lo constituye una sóla etiqueta de apertura y cierre y contiene la url relativa de la entrada, pudiendo ser una página web única propia del contenido o relacionada. De hecho puede emplearse tantos elementos *links* como sean necesarios para describir los enlaces del contenido. En relación a sus atributos operan de la misma forma que en el caso del *feed*. El atributo *rel* define el tipo de relación del contenido con el enlace, adquiriendo distintos valores predeterminados como *alternate, enclosure, related, self* y *via*. El valor *alternate*

define que el contenido enlazado es una representación alternativa de la entrada; El valor *enclosure* define que el enlace es un contenido de gran tamaño y extensión, fundamentalmente un archivo de audio o video, lo que implica un canal de broadcasting o podcasting, afectando al atributo *type* en el que se tendrá que definir obligatoriamente su tipo MIME; El valor *related* implica que el contenido enlazado está relacionado con la entrada; El valor *self* indicaría que el contenido enlazado es la propia entrada; El valor *via* indicaría la fuente original del contenido de la entrada. En relación al resto de atributos no constan de mayor complejidad pues el atributo *type* mantiene su objetivo de establecer el tipo MIME, *hreflang* el código de idioma del contenido enlazado, *title* el título de la página o contenido enlazado y *length* la extensión o tamaño de la página, archivo o contenido enlazado, siendo de especial utilidad en el caso de archivos audiovisuales.

- **Sumario – `<summary></summary>`.** El elemento *summary* puede contener diversos tipos de contenido, considerándose polivalente para definir tanto un sumario o relación de contenidos, un resumen o incluso un prólogo del contenido de la entrada. Al igual que ocurre con otros elementos del formato Atom como el caso del elemento *subtitle* su utilización resulta ambigua.

- **Temática – `<category term="" scheme="" label=""/>`.** El elemento *category* está formado por una sola etiqueta que contiene los atributos necesarios para definir la temática de la entrada. Al igual que su modo de empleo con el elemento *feed*, el atributo *term* define el término o descriptor propiamente dicho que clasifica la entrada; el atributo *scheme* define un enlace URI al lenguaje de clasificación si lo hubiera y finalmente label se emplea como etiqueta o descripción visible en relación al valor definido en *term*.

- **Colaborador – `<contributor></contributor>`.** El elemento *contributor* opera de la misma forma que se ha definido anteriormente con *feed.* Se refiere al colaborador del contenido de la entrada para el que se definen los mismos subelementos para identificar su nombre completo, correo electrónico e identificador URI como autoridad o URL de su sitio web personal.

- **Fecha de publicación – `<published></published>`.** El elemento *published* define la fecha de publicación del contenido, con el formato definido por la norma RFC3339.

- **Fuente – <source></source>.** Constituye un elemento importante desde el punto de vista documental por permitir definir e identificar con mayor precisión la fuente original de la que se ha tomado un determinado contenido que forma parte del canal de sindicación en la entrada correspondiente. Para ello se articulan los siguientes elementos anidados:
  - **Identificador – <id></id>.** Url absoluta a modo de URI de la fuente primaria del contenido de la entrada.
  - **Título – <title></title>.** Título de la fuente original.
  - **Fecha de actualización – <updated></updated>.** Fecha de actualización de la fuente original en formato RFC3339.
  - **Derechos – <rights></rights>.** Derechos de autor o propiedad intelectual que definen los usos de la fuente original del contenido.

- **Derechos – <rights></rights>.** El elemento *rights* define los derechos de autor o propiedad intelectual que rigen el uso de los contenidos de la entrada.

Conforme a las estructuras del canal *feed* y de las entradas de contenido *entry*, la estructura final se constituiría de la siguiente forma en la *tabla58*.

Título: Ejemplo de estructura de un archivo XML en formato Atom
Referencia: tabla58

```
<?xml version="1.0" encoding="UTF-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">

        <id>identificador del canal</id>
        <updated>fecha de actualización</updated>
        <title type="text">título</title>
        <subtitle type="html">subtítulo o descripción del canal</subtitle>
        <link href="url del canal"/>

        <author>
                <name>nombre y apellidos del autor del canal</name>
                <uri>url del perfil del autor del canal</uri>
                <email>correo electrónico del autor del canal</email>
        </author>

        <contributor>
                <name>nombre y apellidos del colaborador del canal</name>
                <uri>url del perfil del colaborador del canal</uri>
                <email>correo electrónico del colaborador del canal</email>
        </contributor>
```

```
    <generator uri="url del programa" version="version del programa">nombre</generator>

    <icon>url del icono del canal</icon>
    <logo>url del logo del canal</logo>
    <rights>derechos del canal</rights>

    <entry>
        <id>identificador de la entrada</id>
        <published>fecha de publicación</published>
        <updated>fecha de actualización</updated>
        <category term="categoría temática"/>
        <link href="url de la entrada"/>
        <title type="text">título de la entrada</title>
        <content type="html">contenido de la entrada</content>
        <summary>resumen o sumario de la entrada</summary>

        <author>
            <name>nombre y apellidos del autor</name>
            <uri>url del perfil del autor</uri>
            <email>correo electrónico del autor</email>
        </author>

        <contributor>
            <name>nombre y apellidos del colaborador</name>
            <uri>url del perfil del colaborador</uri>
            <email>correo electrónico del colaborador</email>
        </contributor>

        <source>
            <id>identificador de la fuente</id>
            <title>título de la fuente<title>
            <updated>fecha de actualización de la fuente</updated>
            <rights>derechos de la fuente</rights>
        </source>

        <rights>derechos de la entrada</rights>
    </entry>
</feed>
```

5.1.2. Extensibilidad de Atom

Al igual que sucede con el resto de formatos de sindicación, el formato Atom es extensible por estar definido como un lenguaje basado en XML. La propiedad de extensibilidad lo habilita para introducir elementos y etiquetas con nombres de espacio propio. Esto se consigue mediante la declaración del namespace en la etiqueta *feed* como un atributo *xmlns* añadido a los ya existentes. De esta forma pueden emplearse terceros formatos basados en XML que sirvan para ampliar la capacidad de descripción de Atom.

Título: Sintaxis para la declaración de namespace por extensibilidad
Referencia: tabla59

Sintaxis de declaración Namespace
xmlns:prefijo="URI del espacio de nombres"

Ejemplo de declaración Namespace
xmlns:dc="http://purl.org/dc/elements/1.1/"

xmlns	XML Namespace
dc	Prefijo del espacio de nombres que adquieren todos los elementos del formato dc (Dublin Core), por ejemplo: dc:title, dc:creator, dc:contributor, etc.
http://purl.org/dc/elements/1.1/	URI del schema XSD o DTD del formato que constituye el Namespace

Título: Ejemplo de extensibilidad de Atom
Referencia: tabla60

```
<?xml version="1.0" encoding="UTF-8"?>

<feed xmlns="http://www.w3.org/2005/Atom"
    xmlns:sy="http://purl.org/rss/1.0/modules/syndication/"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:prism="http://prismstandard.org/namespaces/1.2/basic/">
…
    <entry>
    …
    </entry>
…
</feed>
```

En la presente tabla se muestra un canal Atom, con la declaración de extensibilidad para los metadatos Dublin Core, Syndication y PRISM.

5.1.3. Ventajas e inconvenientes del formato Atom

El formato Atom tal y como es definido por sus propias especificaciones está diseñado para la descripción de recursos y contenidos web. Esto significa que su aplicación universal a cualquier tipo documental viene limitada por una estructura sencilla pensada para servir de soporte de contenidos con una descripción básica. Teniendo en cuenta este aspecto, desde el punto de vista documental existen elementos ventajosos en el empleo de Atom en la descripción de documentos. Fundamentalmente estos residen en la distinción entre la fecha de publicación y la fecha de actualización del canal de sindicación y su diferenciación con respecto al contenido, lo que permite determinar fechas precisas en relación a la edición de la información. Está bien pensado el empleo de elementos como *source* ya que permiten citar y referenciar los documentos originales y su localización absoluta, facilitando un orden y precedencia lógica de la información, su evolución cronológica e incluso su seguimiento hasta las fuentes primarias. Finalmente es destacable el intento por dotar de una estructura de autoridades a los autores y colaboradores tanto del canal de sindicación como de los contenidos. Si bien esta intencionalidad es clara por constituir una estructura común es insuficiente para ordenar y definir suficientemente una autoridad personal, por lo que su modo de empleo resulta muy laxo. En cuanto a los inconvenientes y factores negativos del formato se encuentran algunos elementos demasiado ambiguos que pueden dificultar la normalización de la información siendo éste el caso de *subtitle* y *summary*. En el primer caso *subtitle* permite la introducción indistinta o bien del subtítulo o bien de la descripción del canal de sindicación, lo que en términos documentales resulta radicalmente diferente. Por otro lado *summary* permite la introducción indistinta de un sumario de contenidos o resumen que también resultan dos conceptos diferentes que habría que diferenciar en el formato. Este tipo de elementos ambivalentes bien intencionados originalmente para obtener la máxima flexibilidad y sencillez del formato dificultan la correcta identificación de la información de un documento. En cuanto a los identificadores URI de una fuente original, las especificaciones dan pie a una posible duplicación de información, al permitir definirla con el atributo *href* del elemento *content* con respecto al elemento *source*. Si bien no suele ser definida desde el elemento *content* la URL absoluta de las fuentes debería ser únicamente declarable desde el elemento diseñado específicamente para ese propósito.

5.2. RSS 1.0 RDF

El format RSS1.0 se crea en el 6 de diciembre de 2000 como resultado de los trabajos del RSS-DEV Working Group (BEGED DOV, G. et al., 2008). Este grupo de trabajo continuó la línea de desarrollo original del formato RSS0.90 y RSS0.91 (NETSCAPE, 2001), primer formato de sindicación reconocido y utilizado, creado por Dan Libby y Ramanathan V. Guha para el navegador Netscape y su portal de noticias (KING, A., 2003). RSS1.0 aportaría la primera estructura modular, pensada para las propiedades extensibles de XML. A parte de esta característica RSS1.0 se diferenciará del resto de versiones de RSS por basarse en un nuevo modelo de lenguaje basado en XML, se trata de RDF (Resource Description Framework), que permite la descripción y estructuración de la información de los recursos de la web. Esta particularidad se debe a su capacidad para definir vocabularios de descripción, así como determinar relaciones semánticas de los contenidos mediante el empleo de relaciones sintácticas entre distintos objetos de descripción que hacen las veces de 1) sujetos, 2) predicados y 3) objetos, también denominadas como relaciones triples. El nuevo formato requería el empleo real de un DTD y posteriormente Schema XSD, por lo que se desarrolló uno compatible con los elementos básicos del formato y las propiedades de RDF (RDF/XML Syntax Specification (Revised), 2004). Aunque la propiedad de extensibilidad es común a cualquier formato de sindicación basado en XML, RSS1.0 será el primero en apostar por la introducción de módulos que completaran el formato original para ser capaz de describir mejor, según el caso cualquier tipo de recurso web. Esta concepción llevó al desarrollo de módulos específicamente diseñados para el formato RSS1.0, formando parte de las especificaciones originales. Según la última fecha de actualización de las especificaciones RSS1.0 reconocen tres módulos completamente reconocidos como parte del estándar en uso siendo el módulo *content, syndication y dublin core*. No obstante se desarrollaron otros muchos módulos que o bien se encuentran en desuso y desactualización o bien por motivos de desarrollo aún no son considerados estándar de uso junto con RSS1.0; es el caso de módulos como *prism, admin, aggregation, annotation, audio, cc, changedpage, company, context, email, event, link, richequiv, rss091, search, servicestatus, slash, streaming, subscription, taxonomy o threading* (BEGED DOV, G. et al., 2002).

5.2.1. Estructura del formato RSS1.0 RDF

RSS1.0 consta de una estructura modular pensada para la extensibilidad con terceros formatos y módulos. Al igual que Atom, para operar como formato de sindicación el archivo correspondiente al canal debe estar bien formado y validado. Para estos efectos dispone de un Schema XSD real correspondiente a su modularidad RDF http://www.w3.org/1999/02/22-rdf-syntax-ns# y un namespace sencillo que alude a la estructura básica de RSS heredada en gran medida de la versión 0.90 original, http://purl.org/rss/1.0/ que constituye el manual de referencia y especificaciones del formato de sindicación al igual que ocurre con Atom. Dicho de otra forma, RSS1.0 no requiere de DTD.

Título: Schema XSD de RSS1.0 RDF y Namespace de RSS1.0 RDF
Fuente: http://www.w3.org/1999/02/22-rdf-syntax-ns# ; http://purl.org/rss/1.0/
Referencia: figura19

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:dc="http://purl.org/dc/elements/1.1/">

 <owl:Ontology
     rdf:about="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   <dc:title>The RDF Vocabulary (RDF)</dc:title>
   <dc:description>This is the RDF Schema for the RDF vocabulary defined in the RDF namespace.</dc:description>
 </owl:Ontology>

<rdf:Property rdf:about="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">
  <rdfs:isDefinedBy rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
  <rdfs:label>type</rdfs:label>
  <rdfs:comment>The subject is an instance of a class.</rdfs:comment>
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:domain rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
</rdf:Property>

<rdfs:Class rdf:about="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property">
  <rdfs:isDefinedBy rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
  <rdfs:label>Property</rdfs:label>
  <rdfs:comment>The class of RDF properties.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
</rdfs:Class>

<rdfs:Class rdf:about="http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement">
  <rdfs:isDefinedBy rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
  <rdfs:label>Statement</rdfs:label>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
  <rdfs:comment>The class of RDF statements.</rdfs:comment>
</rdfs:Class>

<rdf:Property rdf:about="http://www.w3.org/1999/02/22-rdf-syntax-ns#subject">
  <rdfs:isDefinedBy rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
  <rdfs:label>subject</rdfs:label>
  <rdfs:comment>The subject of the subject RDF statement.</rdfs:comment>
  <rdfs:domain rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement"/>
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
</rdf:Property>
```

RDF Site Summary (RSS) 1.0

Official Specification

The RSS 1.0 specification was released on 2000-12-06.

It was published along with guidelines for the design of RSS Modules so that RSS can be easily extended.

Modules

The URI `http://purl.org/rss/1.0/modules/` can be used to represent the class of RSS modules.

The following modules are official modules of the RSS-DEV working group:

- Dublin Core
- Syndication
- Content

In addition there are a number of proposed modules.

Schemas

RSS has an RDF Schema available. A copy is also embedded in this document.

Leigh Dodds has created a Schematron Schema and validator for RSS 1.0

More Information

Development and discussion of RSS takes place on the RSS-DEV Mailing List. Feel free to email the list with any questions regarding RSS that you may have. This is also the place to mail your suggestions or corrections to this namespace.

Otras características reseñables que determinan la estructura del formato RSS1.0 es la disposición de un esquema de descripción más sencillo y menos controvertido de cuantos formatos de sindicación existen, permitiendo una total libertad para la introducción de formatos no consensuados pero validados por método de extensibilidad y declaración de espacio de nombres. Tal y como reza la especificación, está pensado para no reescribir el núcleo de elementos que componen el formato, no requiere consenso alguno la utilización de los elementos, dado que resulta la más sencilla y común a la práctica totalidad de los formatos de sindicación, como por ejemplo RSS2.0 o Atom. También destaca la alta compatibilidad con cualquier módulo puesto que no entra en conflicto por constar de elementos suficientemente básicos y finalmente se destaca la libertad de aplicación para describir cualquier tipo de documento, es decir una mayor flexibilidad de aplicación.

Apertura y cierre del formato

El formato RSS1.0 al igual que cualquier documento XML comienza con una primera línea de declaración del lenguaje XML *<?xml version="1.0" encoding="UTF-8"?>* necesaria para identificar que el archivo utiliza las reglas de marcado de XML, así como el set de caracteres empleado. En las especificaciones se recomienda el empleo del set de caracteres UTF-8 como valor preferente del atributo *encoding* de la declaración. Ello conlleva una mayor normalización de los caracteres y símbolos que pueden emplearse en un momento dado entre los ítems del canal de sindicación. Por otro lado la declaración de archivo XML es necesaria para considerar un documento XML bien formado y validado.

A continuación se conforma la estructura del formato RSS1.0 propiamente dicho, compuesto por un elemento raíz, *<rdf:RDF></rdf:RDF>*. Se trata de una etiqueta de apertura y cierre que articula y engloba todo el canal incluyendo sus dos partes esenciales, el *<channel></channel>* y los contenidos *<item></item>*. La etiqueta *rdf:RDF* está asociada de manera directa al espacio de nombres definido por defecto para este formato http://www.w3.org/1999/02/22-rdf-syntax-ns#, convirtiendo el canal de sindicación completo a todos los efectos en un objeto más de la web semántica. Resulta por tanto imperativo la declaración del namespace de RDF para su correcta validación, según se muestra en la *tabla61*.

Título: Esquema de apertura y cierre del formato
Referencia: tabla61

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns="http://purl.org/rss/1.0/">

   <channel rdf:about=""> …</channel>

   <item>…</item>

</rdf:RDF>
```

En relación a la etiqueta *channel* contiene todas las etiquetas de descripción del canal de sindicación. Por otro lado la etiqueta *item* contiene etiquetas de descripción de las entradas que conforman el contenido del canal. Los ítems son repetibles y pueden configurarse cuantos sean necesarios.

Elementos de descripción del canal, <channel></channel>

El elemento channel como se acaba de indicar contiene elementos que actúan como meta-descriptores del propio canal, definiéndolo y enlazándolo oportunamente. A parte de ello, la etiqueta channel requiere un atributo *rdf:about* que sirve para definir la URL absoluta o URI del propio canal de sindicación, es decir la ruta absoluta al archivo XML que contiene el canal. Si no se satisface esta declaración el formato RSS1.0 no estará bien formado y por ende validado. En cuanto a los elementos que componen *channel*, lo constituyen *title, link, description, image, items y textinput*.

- **Título – <title></title>.** Contiene un título descriptivo del canal de sindicación.

- **Enlace – <link></link>.** Contiene la URL absoluta de la página web que contenga una representación HTML del canal de sindicación. También resulta frecuente utilizar la URL del sitio web del canal de sindicación.

- **Descripción – <description></description>.** Contiene una descripción del contenido del canal, su función o fuentes que utiliza para nutrirse de información.

- **Recurso de imagen – `<image rdf:resource=""/>`.** El elemento *image* es opcional, no requiere de etiqueta de cierre y permite realizar una asociación RDF entre una imagen y el propio canal de sindicación. Esta asociación se efectúa mediante el empleo de un atributo *rdf:resource* que toma el valor de la URL absoluta del archivo de imagen asociado. Dado que la etiqueta *image* acompañada del atributo *rdf:resource* es una declaración de asociación, en RDF es necesario identificar o declarar el objeto de dicha relación, implicando otro elemento *image* situado en un nivel jerárquico superior, considerado hermano de *channel* pero con una estructura diferente:

  - **Descripción de imagen – `<image rdf:about=""> <title></title> <link></link> <url></url> </image>`.** Esta estructura responde a la declaración de un objeto, concretamente una imagen. Para ello la etiqueta *image* consta de un atributo *rdf:about* que permite establecer una relación de identificación con respecto al recurso definido anteriormente <*image rdf:resource*=""/>. La descripción de este objeto conlleva definir la misma URL absoluta o URI de la imagen definida anteriormente como valor del atributo *rdf:about*. Por otro lado supone la descripción del título de la imagen mediante el subelemento *title*, la introducción del vínculo que representa la imagen mediante *link* y la URL absoluta correspondiente a dicha imagen mediante *url*, que coincide normalmente con la URI del recurso u objeto que se describe.

- **Lista de elementos – `<items></items>`.** El elemento *items* es de interés para la organización de los contenidos del canal de sindicación, puesto que actúa a modo de índice o tabla de contenidos, permitiendo relacionar cada elemento *item* por medio de su URL absoluta o URI, principal identificador de cada apartado. Para ello *items* está compuesto por la siguiente estructura anidada: `<rdf:Seq><rdf:li resource=""/></rdf:Seq>`. La etiqueta *rdf:Seq* responde a una secuencia o listado de recursos declarados y que por lo tanto están presentes en el canal de sindicación a modo de *ítem.* Dentro de *rdf:Seq* se encuentra el elemento repetible *rdf:li* que contiene la referencia URI del contenido disponible en el canal, declarado a través del atributo *resource*. De esta forma se encontrarán tantas referencias como contenidos se traten dentro del canal de sindicación, manteniendo la integridad referencial del URI de acceso a los mismos.

- **Recurso de entrada de datos – `<textinput rdf:resource=""/>`.** El elemento *textinput* permite establecer una asociación mediante RDF con una página web o aplicación XML a la cual

puede remitirse una determinada información mediante el protocolo HTTP, método GET. La utilidad original de este elemento era servir de caja de texto para introducir cadenas de consulta y poder efectuar búsquedas o filtrados sobre los contenidos del canal de sindicación. De esta forma la aplicación de destino, directamente vinculada procesaría la información y ofrecería una respuesta a dicha demanda informativa. Al igual que sucede con el elemento *image*, en RSS1.0 al declarar una asociación se hace necesario describir el objeto de dicha relación. Por este motivo se emplea otro elemento textinput situado en un nivel jerárquico superior como si de un ítem se tratara, considerado hermano del elemento *channel* pero con una estructura diferente:

- **Descripción de entrada de datos – <textinput rdf:about=""> <title></title> <description></description> <name></name> <link></link> </textinput>.** Esta estructura adicional refleja la descripción del objeto de la relación, en este caso una página a la que se remiten datos por medio del protocolo HTTP. Para establecer una integridad referencial y con ello una asociación con el recurso, se emplea el atributo *rdf:about* que coincidirá con la URI o URL absoluta de la página o aplicación XML objeto. Por otro lado se contienen los subelementos *title* para identificar el título de dicha página u objeto, *description* para una descripción o resumen de su contenido y funciones, *name* para su denominación básica y *link* para declarar el enlace o URL del objeto que con frecuencia coincidirá con la URI o URL absoluta del recurso y de la asociación.

Elementos de item

El elemento *item* constituye la entrada a los contenidos del canal de sindicación en RSS1.0, pero también la declaración del objeto o descripción del mismo, anteriormente referenciado en el canal, concretamente en la ruta (*rdf:RDF →channel → items → rdf:Seq → rdf:li)*. El ítem forma parte dependiente de la etiqueta *rdf:RDF* al igual que channel por lo que se pueden considerar nodos hermanos, dependientes del mismo padre. Cada ítem consta a su vez de los siguientes elementos:

- **Título – <title></title>.** Contiene el título del contenido.

- **Enlace – <link></link>.** Contiene la URL absoluta de la página o URI correspondiente al contenido.

- **Descripción – <description></description>.** Contiene el contenido completo o parte del mismo. Engloba toda la información en bruto, incluyendo codificación html o xhtml si fuera necesario.

Conforme a las estructuras del canal *channel* y de las entradas de contenido *item*, la estructura final se constituiría de la siguiente forma:

Título: Ejemplo de estructura de un archivo XML en formato RSS1.0
Referencia: tabla62

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns="http://purl.org/rss/1.0/"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">

    <channel rdf:about="url del canal">
        <title>título del canal</title>
        <link>url del canal</link>
        <description>descripción del canal</description>
        <image rdf:resource="url del logotipo del canal" />
        <items>
            <rdf:Seq>
                <rdf:li resource="url de la entrada" />
            </rdf:Seq>
        </items>
    </channel>

    <image rdf:about="url del logotipo del canal">
        <title>título del canal</title>
        <link>url del canal para el enlace del logotipo</link>
        <url>url del logotipo del canal</url>
    </image>

     <item rdf:about="url de la entrada">
          <title>título de la entrada</title>
          <link>url de la entrada</link>
          <description>contenido de la entrada</description>
    </item>

</rdf:RDF>
```

5.2.2. Extensibilidad de RSS1.0 RDF

RSS1.0 es un formato diseñado originalmente para ser ampliado con diversos módulos o terceros formatos por medio de la propiedad de extensibilidad. Para utilizarlos es necesario declarar el espacio de nombres o namespace correspondiente al módulo o formato permitiendo en ese caso la introducción de nuevas etiquetas y elementos de descripción (BRAY, T. et al., 2006).

La sintaxis para la declaración de namespace es similar en todos los formatos, variando la ubicación de los atributos *xmlns*. En el caso de RSS1.0 se sitúan como atributos del elemento raíz, la etiqueta *rdf:RDF*.

Título: Ejemplo de extensibilidad de RSS1.0 RDF
Referencia: tabla63

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns="http://purl.org/rss/1.0/"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:sy="http://purl.org/rss/1.0/modules/syndication/"
         xmlns:content="http://purl.org/rss/1.0/modules/content/"
         xmlns:prism="http://prismstandard.org/namespaces/1.2/basic/">

    <channel>…</channel>

    <item>…</item>

</rdf:RDF>
```

Como se ha explicado anteriormente forman parte de las especificaciones de RSS1.0 los módulos con los que puede ser extendido o ampliado el formato de sindicación. En este apartado se analizan aquellos que han sido aceptados como estándar Dublin Core, Syndication, Content y aquellos que aún constando como propuesta, tienen una aplicación fundamental a efectos documentales, concretamente PRISM.

Módulo Dublin Core

Dublin Core (BEGED DOV, G. et al., 2000) es un estándar de metadatos que si bien resulta independiente del desarrollo de RSS1.0 RDF, también es verdad que ha sido incluido y aprobado como módulo estándar del mismo, según las especificaciones. Ateniéndonos a estas, el módulo consta de los elementos básicos de Dublin Core a saber:

Título: Metadatos básicos de Dublin Core
Fuente: (Dublin Core Metadata Element Set, Version 1.1, 2008)
Referencia: tabla64

Elemento	Descripción
<dc:title>	Definición del título propiamente dicho. *Disponible en:* http://purl.org/dc/terms/title
<dc:creator>	Definición del nombre del autor y sus datos de filiación. *Disponible en:* http://purl.org/dc/terms/creator
<dc:subject>	Materia o temática del documento o contenido. *Disponible en:* http://purl.org/dc/terms/subject
<dc:description>	Descripción o contenido completo. *Disponible en:* http://purl.org/dc/terms/description
<dc:publisher>	Nombre del editor y sus datos de filiación. *Disponible en:* http://purl.org/dc/terms/publisher
<dc:contributor>	Nombre del colaborador y sus datos de filiación. *Disponible en:* http://purl.org/dc/terms/contributor
<dc:date>	Fecha de publicación según formato ISO8601. *Disponible en:* http://purl.org/dc/terms/date
<dc:type>	Tipo de contenido en función a su género o naturaleza. *Disponible en:* http://purl.org/dc/terms/type
<dc:format>	Formato del documento, soporte físico e incluso su descripción física. *Disponible en:* http://purl.org/dc/terms/format
<dc:identifier>	Identificador universal del contenido, permite ISBN, ISSN, URI, número de registro, etc. *Disponible en:* http://purl.org/dc/terms/identifier
<dc:source>	Fuente de origen del contenido. Admite URL o definición de la fuente. *Disponible en:* http://purl.org/dc/terms/source
<dc:language>	Idioma o lengua en la que está escrito el contenido. *Disponible en:* http://purl.org/dc/terms/language
<dc:relation>	URL de páginas cuyos contenidos están relacionados. *Disponible en:* http://purl.org/dc/terms/relation
<dc:coverage>	Definición de la cobertura cronológica y geográfica. *Disponible en:* http://purl.org/dc/terms/coverage
<dc:rights>	Derechos de propiedad intelectual, autoría y explotación. *Disponible en:* http://purl.org/dc/terms/rights

Los metadatos fundamentales de Dublin Core permiten completar los formatos de sindicación, para describir de la mejor manera posible un documento atendiendo a sus aspectos fundamentales, que por otra parte RSS1.0 no contempla en su configuración básica. El tipo de recurso, autor, colaborador, editor, el formato, la fuente original de la información, contenidos relacionados, declaración de números de identificación universal ISBN, ISSN, derechos de propiedad y explotación así como la cobertura cronológica y geográfica, permiten una identificación más completa de cualquier material bibliográfico, digital, facilitando por otra parte su tratamiento documental.

Si bien tales los elementos de Dublin Core se consideran esenciales, el empleo de su Namespace habilita la utilización de otros que también pueden ser aprovechados para la descripción documental.

Título: Metadatos extendidos Dublin Core.
Fuente: (Dublin Core Metadata Element Set, Version 1.1, 2008)
Referencia: tabla65

Elemento	Descripción
<dc:alternative>	Define un contenido alternativo al descrito mediante su URI. *Disponible en: http://purl.org/dc/terms/alternative*
<dc:abstract>	Resumen del contenido. *Disponible en: http://purl.org/dc/terms/abstract*
<dc:created>	Fecha de creación del contenido. Disponible en: http://purl.org/dc/terms/created
<dc:bibliographicCitation>	Define una cita bibliográfica en el contenido que se describe. *Disponible en:http://purl.org/dc/terms/bibliographicCitation*
<dc:accrualMethod>	Define el método de introducción de los recursos en la colección. *Disponible en: http://purl.org/dc/terms/accrualMethod*
<dc:accrualPeriodicity>	Define la periodicidad de incorporación de los recursos en la colección. *Disponible en: http://purl.org/dc/terms/accrualPeriodicity*
<dc:BibliographicResource>	Define referencia bibliográfica o documental. *Disponible en: http://purl.org/dc/terms/BibliographicResource*
<dc:URI>	Define una URL identificadora universal del documento o recurso descrito. *Disponible en: http://purl.org/dc/terms/URI*
<dc:UDC>	Contiene la notación numérica de la clasificación decimal universal o su denominación. *Disponible en: http://purl.org/dc/terms/UDC*

Los metadatos de esta segunda tabla no constan como los principales entre Dublin Core, aunque sí forman parte de su schema XSD, empleado para la extensibilidad de RSS1.0 RDF. Tales metadatos posibilitarían la descripción de contenidos alternativos, delimitación específica de resúmenes, fecha de creación de contenido, citas y referencias bibliográficas e incluso la clasificación CDU. Resulta concluyente que ampliando la estructura con estos metadatos, el formato de sindicación está más cerca de representar un registro bibliográfico, dado que tiene en cuenta los aspectos de clasificación, publicación y referenciación bibliográfica propios de la catalogación de cualquier documento.

Para utilizar los metadatos Dublin Core como entidades de la estructura del formato de sindicación RSS1.0 es necesario declarar su Namespace correspondiente **xmlns:dc="**http://purl.org/dc/elements/1.1/**"**. Por otra parte, los metadatos Dublin Core pueden emplearse para la descripción del canal de sindicación o bien para la descripción de los ítems o contenidos documentales que constituyen la colección. Dicho de otra forma, está permitido su empleo en las entidades <*channel*> e <*item*>, constituyendo ésta una característica de flexibilidad esencial.

Módulo Syndication

El módulo *Syndication* (RDF Site Summary 1.0 Modules: Syndication, 2000), fue desarrollado por el mismo equipo de trabajo autor del formato de sindicación RSS1.0. Esto significa que su integración es muy frecuente, dado que permite definir la fecha de actualización, frecuencia y periodo de actualización. Aunque se considera un módulo con una cantidad de elementos reducida, resulta importante para indicar a cualquier lector parser o agregador cuando debe refrescar el contenido del canal de sindicación. Aunque tal especificación es solucionada por muchos programas mediante la actualización sistemática, predefinida por el usuario, aún se emplean las etiquetas de Syndication como resultado del proceso de generación de un canal de sindicación a efectos indicativos del grado de actualización del mismo.

Título: Elementos del módulo Syndication
Referencia: tabla66

Elemento	Descripción
<sy:updatePeriod>	Define el periodo de actualización del canal de sindicación a partir de valores predeterminados, concretamente: hourly, daily, weekly, monthly, yearly
<sy:updateFrequency>	Se determina con un número entero positivo la frecuencia de actualización en función a un periodo definido.
<sy:updateBase>	Fecha de actualización del canal de sindicación según formato ISO8601 http://www.w3.org/TR/NOTE-datetime o RFC3339

En relación al modo de empleo, primeramente es necesario declarar el Namespace de Syndication **xmlns:sy="http://purl.org/rss/1.0/modules/syndication"**. Esta instrucción habilita el empleo de los elementos del módulo Syndication, especificándose que su incorporación se limita y circunscribe a la entidad <*channel*></*channel*> dado que constituyen parte de la descripción del canal de sindicación.

Módulo Content

El módulo Content (BEGED DOV, G. et al., 2002), también desarrollado por el mismo equipo de trabajo de RSS1.0, viene a resolver uno de los problemas más importantes que plantea el uso de RSS1.0. Se trata del apartado de contenidos, dado que en la configuración básica del formato, solo cuenta con una etiqueta <*description*></*description*>, que a todas luces resultaba insuficiente para determinar con claridad qué tipo de información contendría, dada su ambigüedad, para definir o todo el contenido, o parte del mismo o bien inclusive un resumen. El módulo Content aporta el elemento especial <*content:encoded*>, que hace posible la incorporación de contenido codificado en formato HTML o XHTML sin necesidad de especificar en RDF tal condición. De esta forma pueden introducirse textos o contenidos completos editados para su visualización web (SHEPHERD, E., 2008).

Título: Elementos del módulo Content
Referencia: tabla67

Elemento	Descripción
<content:encoded> <![CDATA[ ]]> **</content:encoded>**	El elemento *content:encoded* se emplea para incorporar contenidos editados en HTML. No obstante siempre que esto ocurre, se utiliza la instrucción *<![CDATA[ ]]>* para marcar que el contenido está compuesto por código y datos.

El elemento *<content:encoded></content:encoded>* no se emplea solo. Requiere del uso de una etiqueta embebida que determina que el contenido está compuesto por código y datos, se trata de la etiqueta *<![CDATA[ ]]>*. Es importante señalar que toda información contenida en CDATA no es analizada por ningún parser, ni siquiera trasciende a la validación del Schema XSD de RSS1.0, RDF o del propio módulo Content. Esto se debe a que declara un espacio libre de análisis para permitir precisamente la ejecución del código HTML. Es precisamente esta característica la que hace que el módulo Content sea utilizado muy frecuentemente en formatos de sindicación RSS1.0 RDF. Por otro lado también es considerada por muchos autores como un agujero de seguridad. Si bien no es posible ejecutar código distinto de HTML, sí es posible ejecutar códigos programados en JAVA o Javascript que pueden contener programas maliciosos que escaparían al control de ejecución del navegador (PILGRIM, M., 2003).

Para utilizar el módulo Content, previamente es necesario declarar de su Namespace **xmlns:content="http://purl.org/rss/1.0/modules/content/".** Por otro lado *content:encoded* está pensado para ser utilizado en el marco de los ítems del canal *<item></item>* completando el esquema básico original.

Módulo PRISM

Las siglas PRISM corresponden a *Publishing Requirements for Industry Standard Metadata* y son un compendio de metadatos, desarrollados por el consorcio empresarial IDEAliance (IDEALLIANCE, 2008) especializados en la descripción y gestión de las publicaciones periódicas y sus contenidos.

El módulo PRISM (HAMMOND, T. et al., 2004) no forma parte de los módulos plenamente aceptados por las especificaciones de RSS1.0 RDF (BEGED DOV, G. et al., 2001), pero sí

constituye un modulo propuesto que está siendo empleado sistemáticamente debido a su actualización más continuada y en particular a las posibilidades de descripción que proporciona para los materiales seriados. Tal es así que probablemente sin PRISM no podrían definirse correctamente las numeraciones de los ejemplares de una revista digital que desarrollara su propio canal de sindicación, tampoco podrían conocerse la extensión de los artículos, su paginación, su ISSN propiamente tipificado, así como las distintas fechas que acompañan a la tramitación y publicación de un artículo científico como su fecha de recepción, aceptación y publicación. De esta forma, PRISM consta de más de medio centenar de elementos para describir estos conceptos.

Las entidades de PRISM se dividen en dos apartados; por un lado los elementos de la categoría *A-PRISM*, utilizados para la descripción y control de publicaciones periódicas y por otro *B-PRISM* compuesto por elementos especializados en la gestión de derechos, cuyo prefijo de namespace varía levemente de *prism* a *prismUsageRights* (HAMMOND, T., 2008).

En este análisis se abordan los elementos genuinos del módulo *A-PRISM,* clasificados según su objeto de descripción a saber: título, mención de responsabilidad, numeración, paginación, fechas, edición, clasificación, categorías, indización, resumen, secciones y apartados de publicación, identificadores, relaciones entre recursos, derechos y portada.

Título: Elementos básicos de PRISM 2.0 para la descripción de publicaciones seriadas
Referencia: tabla68

Elemento	Descripción
Título	
<prism:publicationName>	Título de la publicación o revista, así como del recurso que será publicado.
<prism:alternateTitle>	Variaciones del título o títulos alternativos al principal.
Mención de responsabilidad	
<prism:person>	Nombre completo a modo de autoridad personal relativa a un artículo o contenido.
<prism:corporateEntity>	Entidades corporativas editoriales de la publicación periódica.
<prism:organization>	Autoridad corporativa relativa a un artículo o contenido.
<prism:distributor>	Autoridad distribuidora de la publicación.

Numeración	
<prism:volume>	Volumen de la publicación.
<prism:number>	Número de ejemplar.
Paginación	
<prism:startingPage>	Página de inicio de un artículo o contenido.
<prism:endingPage>	Página de finalización de un artículo o contenido.
<prism:pageRange>	Rango de páginas de un artículo o contenido.
Fechas	
<prism:creationDate>	Fecha de creación de un artículo según formato ISO8601 http://www.w3.org/TR/NOTE-datetime o RFC3339
<prism:dateReceived>	Fecha de recepción del artículo según formato ISO8601 http://www.w3.org/TR/NOTE-datetime o RFC3339
<prism:modificationDate>	Fecha de modificación del artículo según formato ISO8601 http://www.w3.org/TR/NOTE-datetime o RFC3339
<prism:pulicationDate>	Fecha de publicación definitiva del artículo según formato ISO8601 http://www.w3.org/TR/NOTE-datetime o RFC3339
<prism:embargoDate>	Fecha de inicio del embargo del artículo, para el sometimiento de los contenidos bajo unas clausulas y derechos de utilización. Emplea formato ISO8601 http://www.w3.org/TR/NOTE-datetime o RFC3339
<prism:expirationDate>	Fecha de finalización del embargo del artículo según formato ISO8601 http://www.w3.org/TR/NOTE-datetime o RFC3339
<prism:killDate>	Fecha de eliminación del artículo o contenido en línea según formato ISO8601 http://www.w3.org/TR/NOTE-datetime o RFC3339
Edición	
<prism:edition>	Datos de la edición de la publicación, incluyendo identificadores especiales de su edición, tipo de edición, etc.
Clasificación, categorías, indización y resumen	
<prism:genre>	Descripción del género del contenido.

<prism:location>	Descripción de las localizaciones geográficas relativas al contenido.
<prism:timePeriod>	Descripción del periodo cronológico relativo al contenido.
<prism:keyword>	Relación de palabras clave para etiquetar el artículo o contenido.
<prism:metadataContainer>	Metadatos originales de un artículo o contenido. Este elemento actúa como una bolsa de terceros elementos o etiquetas, correspondientes a dichos metadatos.
<prism:category>	Categoría temática del contenido de la publicación o de los artículos contenidos.
<prism:teaser>	Resumen o breve descripción que presenta el contenido del artículo.
<prism:industry>	Categoría industrial en la que está orientado el recurso, artículo o contenido.
<prism:object>	Identificación y Descripción de objetos físicos o virtuales referidos en la temática del artículo o en su contenido.
Secciones y apartados de la publicación	
<prism:section>	Sección en la que está organizado un artículo de una publicación periódica. Responde a la subdivisión lógica del tema principal que aborda.
<prism:subsection1>	Responde a una subdivisión de una sección principal de la publicación. Responde al segundo nivel de clasificación.
<prism:subsection2>	Se trata de una subdivisión de tercer nivel.
<prism:subsection3>	Subdivisión de cuarto nivel.
<prism:subsection4>	Subdivisión de quinto nivel.
<prism:issueName>	Denominación de temas o apartados, diferentes de las secciones, utilizados en las publicaciones periódicas y seriadas para tratar asuntos destacados.
<prism:issueIdentifier>	Identificador adicional de un tema o apartado mediante URI.

Identificadores	
<prism:doi>	Document Object Identifier.
<prism:elssn>	ISSN de la versión electrónica.
<prism:isbn>	ISBN International Standard Book Number. Utilizado para contenidos seriados que también han sido publicados a modo de monografías.
<prism:issn>	ISSN International Standard Serial Number.
Relaciones entre recursos	
<prism:channel>	URI del sitio web del canal de sindicación correspondiente a la publicación.
<prism:hasAlternative>	URI de un recurso alternativo al contenido o artículo descrito.
<prism:hasCorrection>	URI de la corrección del recurso descrito. Permite remitir a la página con la corrección.
<prism:isCorrectionOf>	Determina la URI de la versión no corregida de una página o contenido. Esto significa que el ítem que emplea este elemento es la versión corregida.
<prism:hasPreviousVersion>	URI de una página con la versión anterior del contenido o del artículo.
<prism:hasTranslation>	URI de una página de traducción correspondiente al recurso o al contenido descrito.
<prism:isTranslationOf>	Identifica la URI del recurso original del que se ha efectuado una traducción.
<prism:url>	Determina la URL del recurso o artículo.
Derechos	
<prism:copyright>	Definición de los derechos de propiedad intelectual correspondientes al autor.
<prism:rightsAgent>	Nombre y datos de filiación del gestor de derechos y licencias de los contenidos del canal de sindicación de la publicación.

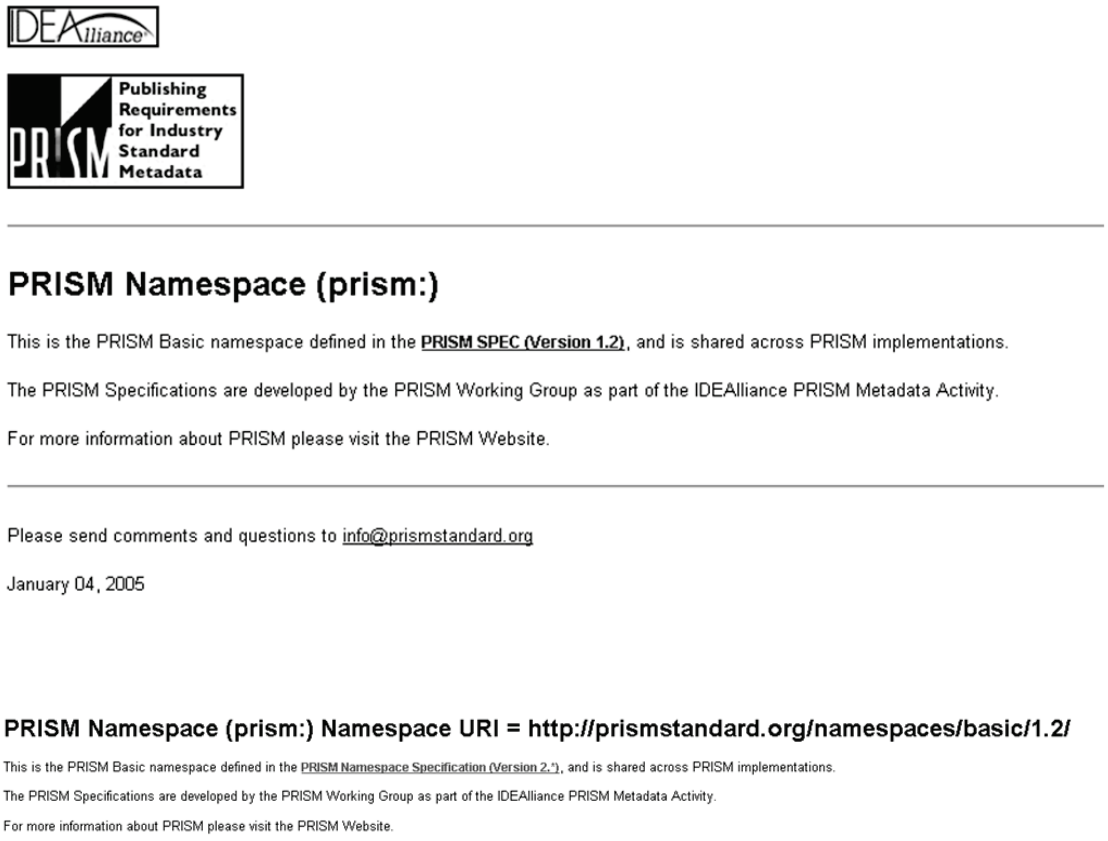
Portada	
<prism:coverDate>	Fecha de la portada de una revista en formato ISO8601 http://www.w3.org/TR/NOTE-datetime o RFC3339Normalmente se asocia a un número, volumen o ejemplar, utilizándose esta información en muchos casos como método de citación.
<prism:coverDisplayDate>	Fecha de la portada en modo escrito y legible.

Como puede observarse, el módulo PRISM permite describir una gran cantidad de casuísticas derivadas de la descripción de publicaciones periódicas y sus artículos o recursos, por ejemplo: El reenvío de contenidos traducidos, corregidos o alternativos, la disposición de identificadores específicos empleados por publicaciones seriadas como el ISSN, la distinción de las fechas en los artículos dependientes de un proceso de edición, revisión, aceptación y publicación, la delimitación cronológica de clausulas y derechos específicos del documento, la organización por secciones de los contenidos de la publicación periódica, el empleo de categorías temáticas, objetos o lugares relativos al contenido, etc. Todas estas características convierten a PRISM en un módulo especializado que permitiría especialmente en RSS1.0 RDF describir artículos de revista, sus contenidos y la propia revista en sí misma, logrando un símil de catalogación multinivel de este tipo documental.

Aunque PRISM no es un estándar reconocido en las especificaciones de RSS1.0 RDF, sí que es posible su utilización en el formato de sindicación. Esto se debe a que dispone de su propio Namespace que no Schema XSD. De hecho cuando se investiga a fondo el contenido del Namespace de las dos principales versiones de PRISM 1.2 (PRISM: Publishing Requirements for Industry Standard Metadata [version 1.2], 2004) y 2.0 (PRISM Namespace [version 2.0], 2008), se cargan páginas que remiten al manual básico del módulo, además de informar de la versión de Namespace correspondiente.

Suelen emplearse dos versiones de PRISM, la versión básica con menores capacidades de descripción, **xmlns:prism="http://prismstandard.org/namespaces/basic/1.2/"** y su versión extendida 2.0, **xmlns:prism="http://prismstandard.org/namespaces/basic/2.0/"** analizada en la tabla anterior. Es recomendable esta última por ser considerada la última edición estable, además de solucionar errores en la propia nomenclatura de la URI de su Namespace.

Título: Namespace de PRISM
Fuente: http://prismstandard.org/namespaces/basic/1.2/
Referencia: figura20



En relación al modo de empleo de los elementos PRISM, pueden utilizarse indistintamente tanto en el apartado <channel></channel> como en <item></item>. No obstante, existen limitaciones debido en parte por la propia finalidad descriptiva de las etiquetas, como por ejemplo las destinadas a la descripción de la fecha de la portada de la revista, que principalmente serán utilizadas para describir un ejemplar en el canal de sindicación pero no un artículo.

5.2.3. Ventajas e inconvenientes del formato RSS1.0

El formato RSS1.0 RDF resulta enormemente versátil debido a su capacidad de extensibilidad expresamente diseñada para incorporar módulos también basados en RDF como Syndication, Content o PRISM. Esto permite no sólo describir mejor un determinado tipo de documento,

sino vincularlo mejor a terceros recursos, calificadores, autoridades y demás elementos interrelacionados, dicho de otra forma, lograr una web semántica a pequeña escala dentro de un mismo canal de sindicación.

También resulta ventajosa la capacidad para describir documentación monográfica mediante Dublin Core y documentación seriada a través de PRISM logrando una máxima integración con el formato de sindicación. Por otra parte la posibilidad de introducir el contenido del documento completo o parcial con el mismo formato HTML que podría tener en origen embebido dentro del canal mediante el módulo Content, lo convierte en un formato muy polivalente.

En cuanto a los inconvenientes de RSS1.0 cabría destacar que la configuración básica del formato es la más reducida de todos los formatos de sindicación, dependiendo por completo de los módulos y la extensibilidad para mejorar su capacidad de descripción. Esto supone un inconveniente para los programas parser que deben adaptarse continuamente a los formatos de sindicación que cada productor de canales de sindicación genera. Dicho de otra forma, la ilimitada cantidad de posibilidades de combinación de todos los elementos de los módulos hace difícil y lenta la adaptación de los programas de lectura y análisis. De hecho puede atribuirse a esta causa la lentitud existente en el progreso de nuevos formatos de sindicación y la aprobación final de muchos módulos considerados actualmente como propuestas, tal y como ocurre con PRISM.

Otro inconveniente destacable es la concepción de los módulos y sus etiquetas para la descripción general de contenidos y tipos documentales que no sólo se encuentran en un soporte físico sino eminentemente digital. Queda patente que la conjugación de ambos conceptos de descripción no está del todo aclarada, dado que se echan en falta por ejemplo las fechas extremas de las publicaciones periódicas, no determinadas en ninguna etiqueta PRISM, que por otra parte se ocupa de las fechas que atañen fundamentalmente al artículo, contenido o recurso.

Finalmente es destacable una tendencia a no desarrollar schemas XSD que deberían estar enlazados a modo de Namespace según cada módulo y formato. Aunque a efectos formales la URI por sí sola, resulta suficiente, los casos citados no validan sus propias estructuras como queda demostrado con PRISM. De hecho la práctica común en estos casos es referir desde una página HTML al manual o guía de dicho Namespace o formato. Si bien esto resulta útil, no es del todo ortodoxo.

5.3. RSS 2.0

El formato RSS2.0 es un lenguaje de marcado basado en las especificaciones de XML1.0 (Extensible Markup Language (XML) 1.0 (Fifth Edition), 2008), cuyo principal elemento distintivo de otros formatos de sindicación es el empleo de la etiqueta inicial *<rss version="2.0"></rss>*. También es el resultado de la evolución del primer formato de sindicación empleado en un navegador web, concretamente RSS0.90 (NETSCAPE, 2001) desarrollado por Dan Libby y Ramanathan V. Guha. Comparte pues orígenes con el formato RSS1.0 RDF. El origen del desarrollo de RSS2.0 radica justamente en la versión RSS 0.91 que consta de dos interpretaciones distintas, por un lado el navegador Netscape (LIBBY, D.; NETSCAPE, 1999) con una visión tendente al sistema modular de RSS1.0 y por otro lado la empresa UserLand fundada y dirigida por Dave Winer, cuya perspectiva preveía un formato cuya estructura fuera más sencilla evitando la tendencia a implantar RDF (USERLAND, 2000), dado que dificultaba la composición y edición del mismo, así como su mayor difusión para la finalidad principal de transmitir noticias o contenidos web. Dicho de otra forma, Dave Winer optó por una interpretación más sencilla que revestía menor complejidad técnica, mayor facilidad de implantación en sitios web, más práctica a la hora de ser analizada por los programas parser y agregadores. Por estos motivos en el formato RSS0.91 se consolida una ruptura que desencadenará el actual formato de sindicación RSS1.0 RDF en el año 2000, originario de la corriente de desarrollo de Netscape. Ese mismo año, UserLand desarrolla el formato RSS0.92 en el que lleva a cabo leves modificaciones consistentes en las etiquetas de fecha de actualización y periodo de actualización del formato, tal y como indican sus especificaciones (USERLAND, 2000). Sucesivamente el formato evoluciona a las versiones RSS0.93 (WINER, D., 2001) y RSS 0.94 (WINER, D., 2003), hasta completar su transformación con el formato RSS2.0 en el año 2003. Se puede afirmar que las especificaciones de la versión RSS2.0 están plenamente vigentes (BERKMAN CENTER, 2003) aunque existe una versión posterior denominada RSS2.0.1 y una tercera más actualizada de tales especificaciones, publicada en 2009 (WINER, D., 2009).

Título: Historial de versiones de RSS2.0
Fuentes: (KING, A., 2003), **(WINER, D., 2003)**, **(PILGRIM, M., 2004) (SHEPHERD, E., 2007)**, **(HAMMERSLEY, B., 2003)**
Referencia: tabla69

Versión de RSS	Cambios
RSS 0.91 Userland	• Se parte de una estructura básica basada en dos apartados, <*channel*></*channel*> e <*ítem*></*ítem*> contenidos en el elemento <rss version=""></rss>. • Se determinan como subelementos del apartado channel *title, link, description, language, rating, textInput* y *skipHour*. • El apartado ítem, consta de los subelementos *title, link* y *description*.
RSS 0.92	• Define que todos los subelementos de la etiqueta <item></item> son opcionales. • Añade nuevos subelementos para el ítem, *source* (definición de la fuente original de información), *enclosure* (capacidad para alojar archivos multimedia), *category* (elemento para establecer categorías y descriptores temáticos). Además incorpora un sublemento *cloud* (Para incorporar protocolo SOAP que actualice los contenidos del canal de sindicación mediante HTTP y métodos GET y POST) al apartado <channel></channel> del canal. • También elimina los límites de caracteres en todos los elementos del canal de forma que pueda ser introducido cualquier extensión o volumen de texto.
RSS 0.93	• El elemento *enclosure* puede ser repetible indefinidamente dentro del apartado <item></item> también se añaden dos nuevos subelementos *pubDate* y *expirationDate* que indican la disponibilidad de un recurso desde su fecha de publicación hasta su eliminación.
RSS 0.94	• Versión incompatible con todas las anteriormente desarrolladas por eliminar el elemento *expirationDate*. • Por otro lado introduce el atributo *type* en el elemento *description* para definir el tipo MIME de la codificación del texto contenido.
RSS 2.0	• El elemento *language* permite la introducción de códigos de idioma ISO639 (dos caracteres indicativos de la lengua) e ISO3166 (tres caracteres indicativos del país). • El elemento *guid* cambia sus valores URL por URI. • El elemento *category* se habilita como repetible, tantas veces

	sea necesario tanto en el apartado <*channel*> como <*ítem*>. • Se añaden al apartado ítem, los nuevos subelementos *comments* (Dirección URI de los comentarios de un contenido), *author* (Define el autor, correo electrónico y sitio web) y *pubdate* (Indica la fecha de publicación, preferiblemente en formato RFC 2822 aunque posibilita el empleo de otros como el ISO8601, siempre que sean legibles por parsers y agregadores). • Se añaden al apartado channel, los nuevos subelementos *generator* (Define el programa generador del canal y su versión), *ttl* (Define el periodo de tiempo en minutos que tarda en refrescarse el canal) e *image* (Elemento opcional para introducir una imagen o logo identificativo del canal).
RSS 2.0.1	• A RSS 2.0 se le aplica la propiedad de extensibilidad de propia de los formatos basados en XML, de forma que pueda integrar cualquier módulo o tercer formato mediante su Namespace.
RSS 2.0.11	• RSS 2.0.11 se compatibiliza con RSS0.91 Userland al adoptar las mismas etiquetas para determinar el periodo de horas (elemento *skipHours*) y días (elemento *skipDays*) en los que el contenido del canal se actualiza, campos utilizados normalmente por agregadores y enmarcados en el apartado <channel>. • Se introduce un nuevo elemento *rating* para la valoración del canal mediante el sistema PICS (Platform for Internet Content Selection), método que será suprimido por el propio W3C en beneficio de POWDER (Protocol for Web Description Resources) aún en desarrollo.

No obstante, para el análisis del formato, se tendrá en cuenta la última versión RSS 2.0.11 a fin de presentar una mayor actualización de la información así como de los cambios acaecidos con respecto a las versiones anteriores.

5.3.1. Estructura del formato RSS 2.0

Apertura y cierre del formato

El formato RSS2.0 se identifica por su etiqueta de apertura inicial <rss version="2.0"></rss>. Ésta contiene dos apartados fundamentales en cualquier canal de sindicación; el elemento <channel></channel> correspondiente a la descripción del canal de sindicación, y el elemento

<item></item> repetible y que consta de los contenidos de dicho canal. Al contrario que el resto de formatos de sindicación RSS2.0 no dispone de Namespace, aspecto realmente revelador del bajo nivel de soporte al que están sometidos sus elementos y estructuras. Recuérdese que era condición fundamental que el documento esté validado y bien formado para poder ser considerado un formato de sindicación. En este caso particular, se tiene constancia de algún intento por incorporar un namespace propio del formato, de forma que fuera regulado y soportado por la IANA (IANA Uniform Resource Identifer (URI) Schemes per [RFC4395], 2009), *Corporación de Internet para la Asignación de Nombres y Números de Internet*, que también actúa como entidad normalizadora ofreciendo alojamiento a diversas especificaciones y Namespace, entre los que debería de encontrarse el formato RSS2.0. En cambio sí se encuentra el protocolo que emplea para la actualización de los contenidos XML-RPC (HAROLD, W., 2003), que en todo caso, actúa como elemento normalizador y cohesionador del formato. Salvo por esta peculiaridad, el formato de sindicación en sí mismo carece de espacio de nombres propio.

Elementos de channel

- **Título – <title>.** Título del canal propiamente dicho.

- **Enlace – <link>.** URI del canal de sindicación.

- **Descripción – <description>.** Descripción general de la temática y contenidos del canal de sindicación.

- **Idioma – <language>.** Idioma del canal de sindicación atendiendo al formato ISO639, ISO3166.

- **Derechos – <copyright>.** Declaración de derechos de autor, propiedad intelectual y explotación del canal de sindicación.

- **Editor – <managingEditor>.** Nombre y datos de filiación de la persona o entidad gestora de los contenidos del canal de sindicación. Normalmente se identifica únicamente mediante su correo electrónico.

- **Administrador – <webMaster>.** Nombre y datos de filiación de la persona o entidad responsable de la web de la que depende el canal de sindicación. Al igual que con el elemento <*managinEditor*>, suele identificarse mediante su correo electrónico.

- **Fecha de publicación – <pubDate>.** Fecha de publicación del canal de sindicación.

- **Fecha de actualización – <lastBuildDate>.** Fecha de última actualización del canal de sindicación.

- **Temática – <category>.** Permite la introducción de descriptores temáticos o categorías relativas a las secciones a las que pertenece el canal de sindicación.

- **Generador – <generator>.** Define el nombre y versión del programa que generó el canal de sindicación.

- **Especificación – <docs>.** Determina la URI correspondiente al documento con la especificación o guía del formato RSS que se está empleando. En este caso toma el valor http://www.rssboard.org/rss-specification , siempre correspondiente a la última versión del formato. Esto implica que el método alternativo al Namespace es el uso de este elemento.

- **Nube de cambios – <cloud domain="rpc.sys.com" port="80" path="/RPC2" registerProcedure="pingMe" protocol="soap"/>.** Determina un método de actualización mediante un protocolo. Normalmente mediante SOAP o en su defecto XML-RPC. Los valores de los atributos están predeterminados para la función de refresco mediante el proceso *pingMe* por medio del puerto 80, en el programa alojado en el dominio http://rpc.sys.com .

- **Tiempo de vida – <ttl>.** Contiene el periodo de tiempo en minutos, correspondiente al ciclo de refresco y actualización del canal.

- **Imagen – <image> <url></url> <link></link> <title></title> </image>.** Contiene la información de la imagen o logotipo del canal de sindicación. Para ello incorpora tres

subelementos, a saber: la URL de la página enlazada con la imagen, la URI correspondiente a la imagen propiamente dicha y su título.

- **Valoración – <rating>.** Contiene la URI correspondiente a la valoración del canal.

- **Entrada de datos – <textInput> <title></title> <description></description> <name></name> <link></link> </textinput>.** El elemento *textInput* corresponde forma parte de la herencia del formato RSS0.91, que al igual que en RSS1.0 permite remitir a una aplicación XML externa, información, contenidos y textos transmitidos mediante caja de texto presente en el canal de sindicación. Algunas de las aplicaciones XML vinculadas a este elemento son las derivadas de la recuperación y filtrado de la información del propio canal de sindicación. Esto se consigue mediante el protocolo HTTP y su método GET para el envío de los datos de la consulta. Para que el conjunto funcione correctamente, se debe definir una URI correspondiente al enlace de la aplicación XML a través del subelemento *link*. Por otro lado es necesario consignar correctamente el subelemento *name* que contiene el identificador de la caja de texto y por ende el valor de la consulta del usuario. Finalmente, el subelemento *title* y *description* hacen referencia a la etiqueta del botón de envío del formulario y a la explicación del formulario y de los datos a rellenar en la caja de texto.

Elementos del elemento, item

- **Título – <title>.** Contiene el título de la entrada.

- **Enlace – <link>.** Corresponde a la URL del contenido.

- **Descripción – <description>.** Comprende el contenido completo o parte del mismo.

- **Autor – <author>.** Contiene el nombre y filiación del autor del artículo o contenido de la entrada.

- **Temática – <category>.** Contiene descriptores temáticos, categorías o secciones en las que se enmarca el contenido del artículo o la entrada.

- **Comentarios – <comments>.** Contiene un enlace URI correspondiente a los comentarios que se han efectuado al respecto del contenido descrito.

- **Identificador – <guid>.** Contiene un enlace URI del contenido que será utilizado por los agregadores para determinar la existencia del contenido en su repositorio.

- **Fecha de publicación – <pubDate>.** Fecha de publicación del contenido.

- **Fuente – <source>.** Identifica la URL o URI del canal de sindicación, considerado fuente original del contenido descrito en el ítem.

Conforme a las estructuras del canal y de las entradas de contenido, se presenta una estructura básica del formato RSS2.0:

Título: Estructura de un archivo RSS2.0
Referencia: tabla70

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
<channel>

    <title>título del canal</title>
    <link>url del canal</link>
    <description>descripción del canal</description>
    <copyright>derechos del canal</copyright>
    <pubDate>fecha de publicación del canal</pubDate>
    <generator>programa generador del canal</generator>

    <item>
        <title>título de la entrada</title>
        <link>url de la entrada</link>
        <comments>comentarios de la entrada</comments>
        <pubDate>fecha de publicación de la entrada</pubDate>
        <category>categoría temática de la entrada</category>
        <guid>identificador de la entrada</guid>
        <description>contenido de la entrada</description>
    </item>

</channel>
</rss>
```

5.3.2. Extensibilidad de RSS 2.0

La propiedad de extensibilidad en RSS2.0 resulta relativamente novedosa, si se tiene en cuenta que fue introducida en la versión RSS2.0.1. Se puede afirmar que es uno de los últimos formatos de sindicación en hacer efectiva esta propiedad. Al igual que en el resto de formatos, el método de extensibilidad, se lleva a cabo mediante la declaración de Namespace, lo que no deja de ser paradójico dado que originalmente RSS2.0 carece de Namespace propio. Aún así es posible introducir espacios de nombre, concretamente en el elemento de apertura y cierre inicial del formato *<rss version="2.0"></rss>*, tal y como muestra el siguiente ejemplo:

Título: Ejemplo de extensibilidad de RSS2.0
Referencia: tabla71

```
<?xml version="1.0" encoding="UTF-8"?>
<rss xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#”
      xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:sy="http://purl.org/rss/1.0/modules/syndication/"
      xmlns:content="http://purl.org/rss/1.0/modules/content/"
      xmlns:prism="http://prismstandard.org/namespaces/1.2/basic/"  version="2.0">

       <channel>
       …
           <item>…</item>
       …
       </channel>

</rss>
```

Mediante extensibilidad al igual que el resto de formatos, puede emplear cualquier módulo o grupo de metadatos que disponga de espacio de nombres, schema XSD o DTD.

5.3.3. Ventajas e inconvenientes del formato RSS 2.0

Las ventajas de RSS2.0 son la simplicidad del formato de sindicación que hace sencilla su implementación en cualquier navegador, parser o agregador. Por otro lado tal simplicidad lo ha convertido en uno de los formatos más utilizados en los sistemas de publicación web, principalmente blogs, wikis y CMS, pese a que no dispone de un schema XSD, DTD o Namespace. La reciente incorporación de la propiedad de extensibilidad amplia las capacidades de descripción, lo que implica, que de ser utilizada RSS2.0 perdería una de sus principales razones de ser, la simplicidad de sus elementos estructurales. Esto puede ser

beneficioso si se desea emplear RSS2.0 para describir documentos y ofrecer servicios para los que no fue diseñado, pero también puede ser perjudicial para la legibilidad del lenguaje en los sistemas parser y agregadores. Esto se debe a que muchos de estos sistemas al estar adaptados a la estructura básica que en pocos casos se ha visto alterada con el paso de los años, podrían encontrar problemas de lectura al encontrar elementos del espacio de nombres referido en ubicaciones no definidas por la especificación de RSS2.0. Dicho de otra forma, no existe una modelo de aplicación para los distintos módulos y formatos que pueden ser aplicados, como sucede con el formato RSS1.0 RDF. En este sentido RSS1.0 RDF es el formato mejor preparado para la extensibilidad, dado que está caracterizado específicamente para ello y consta de tal documentación, así como del schema XSD de RDF que habilita el empleo de la extensibilidad de una forma explícita.

5.4. MARC-XML

El formato MARC-XML (MARC-XML Schema, 2009) aparece publicado como schema XSD en el año 2001, como una solución al cada vez más inflexible formato MARC tradicional, difícil de ampliar y adaptar a los diferentes centros de información y documentación. Si bien no es considerado un formato de sindicación tradicional como Atom, RSS1.0 RDF y RSS2.0. Ello se debe a que los formatos anteriormente mencionados fueron los primeros en ser incorporados en los navegadores web, obteniendo de esta forma un soporte fundamental para su lectura y actualización de contenidos. También se presupone, según la definición de sindicación de contenidos, la existencia de un productor o creador de documentos bien sean primarios o secundarios, que transmite mediante un canal de información basado en XML, a múltiples suscriptores. En el caso de MARC-XML los productores de los contenidos del canal lo constituyen las propias bibliotecas y centros de documentación que catalogan los fondos bibliográficos y documentales. Dado que MARC-XML está basado en XML, consta de schema XSD validador del formato y dispone de Namespace propio, puede considerarse al formato MARC-XML, habilitado para las mismas funciones que cualquier otro formato de sindicación. En relación al aspecto de la visualización, actualización y representación de los contenidos codificados por MARC-XML, es relevante señalar que la Library of Congress ha desarrollado diversas hojas de estilo y transformaciones XSLT que permiten visualizar y representar correctamente la información del formato MARC-XML en cualquier navegador web, como se explicará en esta parte del capítulo. Por otro lado hay que señalar la carencia de programas parser o agregadores especializados en MARC-XML que en principio podría echar por tierra la consideración de MARC-XML como un posible formato de sindicación utilizándolo como medio de difusión del catálogo bibliográfico. Este aspecto que en principio podría ser problemático es enfrentado y resuelto en esta investigación, mediante el desarrollo de diversos parsers y generadores especializados en el formato MARC-XML como podría haber en los distintos navegadores web si desearan emular con MARC al resto de formatos, por lo que el precedente de su modo de empleo se marca en el presente trabajo.

Argumentada la posibilidad de utilizar MARC-XML como formato de sindicación, se debe señalar su naturaleza plenamente documental que encaja con la disciplina de la catalogación. De hecho, MARC-XML se basa en las normas del formato MARC21 (MARC Standards, 2009) de la Library of Congress, profusamente divulgadas y adaptadas en la práctica totalidad

de las bibliotecas del mundo. El desarrollo del formato MARC-XML es compatible con el estándar ISO2709 permitiendo la interoperabilidad con cualquier adaptación MARC. El formato MARC-XML ha sido diseñado de forma tal que mantiene las mismas etiquetas, indicadores y códigos que el estándar MARC21. Para lograrlo ha sido concebido un schema XSD que actúa como validador estructural del formato que determina una estructura anidada basada en la colección, el registro bibliográfico, el campo o etiqueta bibliográfica y su sub-campo que por regla general es el contenedor de información o datos descriptivos del documento.

5.4.1. Estructura del formato MARC-XML

La estructura del formato MARC-XML está basada en la organización de los catálogos bibliográficos. Esto significa la disposición de un elemento cabecera de apertura y cierre denominado *<collection></collection>* que engloba directamente todos y cada uno de los registros bibliográficos etiquetados con el elemento *<record></record>* y que a su vez contienen las etiquetas correspondientes a los campos de catalogación MARC21. El modo de apertura incluye la declaración del Namespace propio de MARC-XML, su instancia con el schema XSD de XML y su instanciación. Para ello se necesita incluir las URIs especificadas en la siguiente tabla:

Título: Apertura del formato MARC-XML
Referencia: tabla72

```
<?xml version="1.0" encoding="UTF-8"?>
<collection xmlns:marc="http://www.loc.gov/MARC21/slim"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.loc.gov/MARC21/slim
            http://www.loc.gov/standards/marcxml/schema/MARC21slim.xsd">
…
   <record>…</record>
…
</collection>
```

Elementos de record

El elemento *record* delimita un registro bibliográfico y permite la incorporación de todas las etiquetas MARC21 especificadas por la norma (MARC Standards, 2009). En el presente análisis se explican todas aquellas de uso común en la descripción bibliográfica y que a la postre también son utilizadas en el sistema de sindicación desarrollado para la presente investigación.

- **Cabecera – <leader>.** Consiste en elementos de información que contienen valores numéricos o codificados que definen los parámetros para el procesamiento del registro. http://www.loc.gov/marc/bibliographic/bdleader.html

- **Número de control – <controlfield tag='001'>.** Número de control asignado al registro bibliográfico por la organización que lo crea, utiliza o distribuye. Disponible en: http://www.loc.gov/marc/bibliographic/bd001.html

- **Identificador del número de control – <controlfield tag='003'>**. El código MARC de la organización cuyo número de control se registra en el campo 001. Disponible en: http://www.loc.gov/marc/bibliographic/bd003.html

- **Fecha y hora de la última actualización – <controlfield tag='005'>**. Cadena de dieciséis caracteres que especifica la fecha y hora de la última actualización del registro y sirve como un identificador de la versión del registro. Se registran de acuerdo a la Representación de fechas y horas en formato ISO 8601. Disponible en: http://www.loc.gov/marc/bibliographic/bd005.html

- **Número de control LC – <datafield tag='010' ind1=" ind2="><subfield code='a'>.** Número de control LC Library of Congress. Disponible en: http://www.loc.gov/marc/bibliographic/bd010.html

- **Depósito legal <datafield tag='017' ind1=" ind2="><subfield code='a'>.** Número de control DL Depósito Legal. Disponible en: http://www.loc.gov/marc/bibliographic/bd017.html

- **ISBN – <datafield tag='020' ind1=" ind2="><subfield code='a'>.** ISBN Número Internacional Normalizado del libro e información calificativa entre paréntesis. Disponible en: http://www.loc.gov/marc/bibliographic/bd020.html

- **Idioma – <datafield tag='041' ind1=" ind2="><subfield code='a'>.** Código de idioma para el texto y contenido del documento en formato ISO 639-2. Disponible en: http://www.loc.gov/marc/bibliographic/bd041.html

- **Signatura – <datafield tag='050' ind1=" ind2="><subfield code='a'>.** Notación de la signatura del centro catalogador. Disponible en: http://www.loc.gov/marc/bibliographic/bd050.html

- **CDU – <datafield tag='080' ind1=" ind2="><subfield code='a'>.** Número de la Clasificación Decimal Universal. Disponible en: http://www.loc.gov/marc/bibliographic/bd080.html

- **Dewey – <datafield tag='082' ind1=" ind2="><subfield code='a'>.** Número de clasificación Decimal Dewey. Disponible en: http://www.loc.gov/marc/bibliographic/bd082.html

- **Entidad personal – <datafield tag='100' ind1=" ind2=">.**
  Disponible en: http://www.loc.gov/marc/bibliographic/bd100.html
  - **<subfield code='a'>.** Un nombre formado por apellido y nombre; letras, iniciales, abreviaturas, frases o números que se utilizan en lugar de un nombre; o un nombre de familia.
  - **<subfield code='0'>.** Número de control del registro de autoridad.

- **Entidad corporativa – <datafield tag='110' ind1=" ind2=">.**
  Disponible en: http://www.loc.gov/marc/bibliographic/bd110.html
  - **<subfield code='a'>.** El nombre de una entidad corporativa o de la primera entidad subordinada cuando hay alguna presente; el nombre de una jurisdicción bajo la cual se asientan: una entidad corporativa, la sección de una ciudad o el título de una obra.
  - **<subfield code='0'>.** Número de control del registro de autoridad.

- **Nombre de reunión – <datafield tag='111' ind1='' ind2=''>.**
  Disponible en: http://www.loc.gov/marc/bibliographic/bd111.html
  - **<subfield code='a'>.** El nombre de una reunión o de la primera entidad subordinada cuando hay alguna presente; o el nombre de una jurisdicción bajo la cual se asienta un nombre de reunión.
  - **<subfield code='0'>.** Número de control del registro de autoridad.

- **Título uniforme – <datafield tag='130' ind1='' ind2=''>.**
  Disponible en: http://www.loc.gov/marc/bibliographic/bd130.html
  - **<subfield code='a'>.** Título uniforme propiamente dicho, clarificador de obras con títulos confusos o poco claros para su distinción.
  - **<subfield code='0'>.** Número de control del registro de autoridad.

- **Mención de título – <datafield tag='245' ind1='' ind2=''>.**
  Disponible en: http://www.loc.gov/marc/bibliographic/bd245.html
  - **<subfield code='a'>.** Título propiamente dicho.
  - **<subfield code='b'>.** Parte restante del título, subtítulo o título paralelo.
  - **<subfield code='c'>.** Mención de responsabilidad.

- **Mención de edición – <datafield tag='250' ind1='' ind2=''>.**
  Disponible en: http://www.loc.gov/marc/bibliographic/bd250.html
  - **<subfield code='a'>.** Mención de edición propiamente dicha.
  - **<subfield code='b'>.** Resto de la mención de edición. Generalmente, una mención de responsabilidad personal o corporativa y/o una mención de edición paralela.

- **Publicación – <datafield tag='260' ind1='' ind2=''>.**
  Disponible en: http://www.loc.gov/marc/bibliographic/bd260.html
  - **<subfield code='a'>.** Lugar de publicación, distribución.
  - **<subfield code='b'>.** Nombre del editor, distribuidor.
  - **<subfield code='c'>.** Fecha de publicación, distribución.

- **Descripción física – <datafield tag='300' ind1='' ind2=''>.**
  Disponible en: http://www.loc.gov/marc/bibliographic/bd300.html
  - **<subfield code='a'>.** Extensión, número de páginas, volúmenes del documento descrito.
  - **<subfield code='b'>.** Otros detalles físicos.
  - **<subfield code='c'>.** Dimensiones.
  - **<subfield code='e'>.** Materiales adjuntos.

- **Mención de serie – <datafield tag='490' ind1='' ind2=''>.**
  Disponible en: http://www.loc.gov/marc/bibliographic/bd490.html
  - **<subfield code='a'>.** El título de una serie que también puede contener una mención de responsabilidad, otra información sobre el título, fechas o números de volumen que preceden al título o aparecen como parte del título.
  - **<subfield code='v'>.** Designación numérica o secuencial del volumen.

- **Nota general – <datafield tag='500' ind1='' ind2=''>.**
  Disponible en: http://www.loc.gov/marc/bibliographic/bd500.html
  - **<subfield code='a'>.** Información general de relevancia para la descripción del documento.

- **Temática – <datafield tag='650' ind1='' ind2=''>.**
  Disponible en: http://www.loc.gov/marc/bibliographic/bd650.html
  - **<subfield code='a'>.** Término temático como elemento de entrada.
  - **<subfield code='0'>.** Número de control del registro de autoridad.

- **Secundaria de entidad personal – <datafield tag='700' ind1='' ind2=''>.**
  Disponible en: http://www.loc.gov/marc/bibliographic/bd700.html
  - **<subfield code='a'>.** Un nombre formado por apellido y nombre; letras, iniciales, abreviaturas, frases o números que se utilizan en lugar de un nombre; o un nombre de familia.
  - **<subfield code='0'>.** Número de control del registro de autoridad.

- **Secundaria de entidad corporativa – <datafield tag='710' ind1='' ind2=''>.**
  Disponible en: http://www.loc.gov/marc/bibliographic/bd710.html
  - **<subfield code='a'>.** El nombre de una entidad corporativa o de la primera entidad subordinada cuando hay alguna presente; el nombre de una jurisdicción bajo la cual se asientan: una entidad corporativa, la sección de una ciudad o el título de una obra.
  - **<subfield code='0'>.** Número de control del registro de autoridad.

- **No MARC – <datafield tag='887' ind1='' ind2=''>.**
  Disponible en: http://www.loc.gov/marc/bibliographic/bd887.html
  - **<subfield code='a'>.** Campo libre que no está sujeto a las normas MARC, puede incluirse el texto completo de la ficha bibliográfica, metadatos, codificación en lenguajes de marcado, que complete la descripción del documento.

Título: Estructura de un archivo MARC-XML
Referencia: tabla73

```
<?xml version="1.0" encoding="UTF-8"?>
<collection xmlns:marc="http://www.loc.gov/MARC21/slim"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.loc.gov/MARC21/slim
        http://www.loc.gov/standards/marcxml/schema/MARC21slim.xsd">

<record>

<leader>Cabecera</leader>

<controlfield tag='001'>Número de control</controlfield>
<controlfield tag='003'>Identificador del número de control</controlfield>
<controlfield tag='005'>Fecha y hora de la última actualización</controlfield>

<datafield tag='010' ind1='' ind2=''>
    <subfield code='a'>Número de control de la biblioteca del congreso</subfield>
</datafield>

<datafield tag='017' ind1='' ind2=''>
    <subfield code='a'>Número de depósito legal</subfield>
</datafield>

<datafield tag='020' ind1='' ind2=''>
    <subfield code='a'>ISBN international standard book number</subfield>
</datafield>

<datafield tag='041' ind1='' ind2=''>
    <subfield code='a'>Código de idioma</subfield>
</datafield>
```

```
<datafield tag='050' ind1='1' ind2='0'>
    <subfield code='a'>Signatura topográfica</subfield>
</datafield>

<datafield tag='080' ind1='' ind2=''>
    <subfield code='a'>Número de CDU clasificación decimal universal</subfield>
</datafield>

<datafield tag='082' ind1='0' ind2=''>
    <subfield code='a'>Número de clasificación decimal DEWEY</subfield>
</datafield>

<datafield tag='100' ind1='0' ind2=''>
    <subfield code='a'>Asiento principal entidad personal</subfield>
    <subfield code='0'>Identificador de la entidad personal</subfield>
</datafield>

<datafield tag='110' ind1='2' ind2=''>
    <subfield code='a'>Asiento principal entidad corporativa</subfield>
    <subfield code='0'>Identificador de la entidad corporativa</subfield>
</datafield>

<datafield tag='111' ind1='2' ind2=''>
    <subfield code='a'>Asiento principal nombre de reunión</subfield>
    <subfield code='0'>Identificador de la reunión</subfield>
</datafield>

<datafield tag='130' ind1='0' ind2=''>
    <subfield code='a'>Asiento principal título uniforme</subfield>
    <subfield code='0'>Identificador de título uniforme</subfield>
</datafield>

<datafield tag='245' ind1='1' ind2='0'>
    <subfield code='a'>Título</subfield>
    <subfield code='b'>Subtítulo</subfield>
    <subfield code='c'>Mención de responsabilidad</subfield>
</datafield>

<datafield tag='250' ind1='' ind2=''>
    <subfield code='a'>Edición</subfield>
    <subfield code='b'>Mención de edición</subfield>
</datafield>

<datafield tag='260' ind1='' ind2=''>
    <subfield code='a'>Lugar de publicación</subfield>
    <subfield code='b'>Editorial</subfield>
    <subfield code='c'>Fecha de publicación</subfield>
</datafield>

<datafield tag='300' ind1='' ind2=''>
    <subfield code='a'>Extensión</subfield>
    <subfield code='b'>Otros detalles físicos</subfield>
    <subfield code='c'>Dimensiones</subfield>
    <subfield code='e'>Materiales adjuntos</subfield>
</datafield>

<datafield tag='490' ind1='0' ind2=''>
    <subfield code='a'>Título de la serie</subfield>
    <subfield code='v'>Número o volumen de la serie</subfield>
</datafield>
```

```
<datafield tag='500' ind1='' ind2=''>
    <subfield code='a'>Nota general</subfield>
</datafield>

<datafield tag='650' ind1='1' ind2='4'>
    <subfield code='a'>Asiento secundario de materia</subfield>
    <subfield code='0'>Número de control de materia</subfield>
</datafield>

<datafield tag='700' ind1='0' ind2=''>
    <subfield code='a'>Asiento secundario de entidad personal</subfield>
    <subfield code='0'>Número de control de entidad personal</subfield>
</datafield>

<datafield tag='710' ind1='2' ind2=''>
    <subfield code='a'>Asiento secundario de entidad corporativa</subfield>
    <subfield code='0'>Número de control de entidad corporativa</subfield>
</datafield>

<datafield tag='887' ind1='' ind2=''>
    <subfield code='a'>Registro bibliográfico completo</subfield>
</datafield>

</record>

</collection>
```

5.4.2. Extensibilidad y representación de MARC-XML

MARC-XML es un formato diseñado para ser extensible, como se determina en las especificaciones (MARC XML Architecture, 2004), de hecho está declarada esta propiedad de forma oficial mediante el schema XSD de XML (XML Schema, 2005), embebido a su vez, dentro del schema XSD del propio formato MARC-XML (KEITH, C., 2002), que lo habilita.

Las capacidades prescritas en materia de extensibilidad para MARC-XML son las transformaciones de los registros bibliográficos para su visualización y representación HTML, la conversión del set de caracteres mediante XSLT, la incorporación de metadatos Dublin Core, el formato MODS (Metadata Object Description Schema) y capacidad para introducir terceros formatos basados en XML, como por ejemplo módulos propios de RSS1.0 RDF.

Al igual que en el resto de formatos, se requiere la declaración de los espacios de nombres correspondientes en la etiqueta de apertura y cierre del formato *<collection></collection>* tal y como se muestra en la *tabla74*.

Título: Ejemplo de extensibilidad en MARC-XML
Referencia: tabla74

```
<?xml version="1.0" encoding="UTF-8"?>
<collection xmlns:marc="http://www.loc.gov/MARC21/slim"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.loc.gov/MARC21/slim
                                http://www.loc.gov/standards/marcxml/schema/MARC21slim.xsd"
            xmlns:dc="http://purl.org/dc/elements/1.1/"
            xmlns:sy="http://purl.org/rss/1.0/modules/syndication/"
            xmlns:content="http://purl.org/rss/1.0/modules/content/"
            xmlns:prism="http://prismstandard.org/namespaces/1.2/basic/" >

…
<record> … </record>
…

</collection>
```

En cuanto a la capacidad de visualización y representación de un archivo MARC-XML, está ampliamente desarrollada y activa por la Library of Congress. Esto se logra enlazando una hoja de estilo y transformaciones XSLT al archivo MARC-XML. De esta forma, cuando se accede al archivo XML, automáticamente la información contenida en él, toma el aspecto y características visuales y representativas que se hayan definido. Así es posible determinar, una representación de cita bibliográfica o convertir la información a un tercer formato mediante la alteración del orden y disposición de los datos.

Título: Hojas XSLT para MARC-XML
Referencia: tabla75

Conversiones y transformaciones	
Conversión de MARC-XML a MODS	http://www.loc.gov/standards/mods/v3/MARC21slim2MODS3-3.xsl
Conversión de MODS a MARC-XML	http://www.loc.gov/standards/marcxml/xslt/MODS2MARC21slim.xsl
Conversión de MARC-XML a RDF	http://www.loc.gov/standards/marcxml/xslt/MARC21slim2RDFDC.xsl
Conversión de MARC-XML a OAI	http://www.loc.gov/standards/marcxml/xslt/MARC21slim2OAIDC.xsl

Conversión de MARC-XML a SRW	http://www.loc.gov/standards/marcxml/xslt/MARC21slim2SRWDC.xsl
Conversión de Dublin Core a MARC-XML	http://www.loc.gov/standards/marcxml/xslt/DC2MARC21slim.xsl
Conversión de OAI a MARC-XML	http://www.loc.gov/standards/marcxml/xslt/OAIMARC2MARC21slim.xsl
Conversión de ONIX a MARC-XML	http://www.loc.gov/standards/marcxml/xslt/ONIX2MARC21slim.xsl
Visualización y representaciones	
Cita bibliográfica	http://www.loc.gov/standards/marcxml/xslt/MARC21slim2MARCBIG.xsl
Cita de autoridad	http://www.loc.gov/standards/marcxml/xslt/MARC21slim2MARCBIGA.xsl
Cita de autoridad y bibliográfica	http://www.loc.gov/standards/marcxml/xslt/MARCBIG2MARC21slim.xsl
Visualización de etiquetado MARC	http://www.loc.gov/standards/marcxml/xslt/MARC21slim2HTML.xsl

5.4.3. Ventajas e inconvenientes del formato MARC-XML

El formato MARC-XML es el formato idóneo para la descripción bibliográfica y documental, dada su amplísima y vasta extensión de etiquetas y casuísticas heredadas de una larga tradición biblioteconómica. Otra ventaja destacable es el gran nivel de normalización y oficialidad alcanzado por el formato, que resulta mucho mejor soportado y documentado que los propios formatos de sindicación tradicionales. La estructura de etiquetas y elementos resulta acertada por lograr adaptar el método de catalogación automatizada a los lenguajes de marcado. De hecho es posible representar cualquier documento MARC en MARC-XML y convertirlo a cualquier otro formato mediante las herramientas de conversión y transformación XSLT. Otra ventaja del formato es la posibilidad de representar y visualizar el contenido al igual que lo haría un navegador web con cualquier canal de sindicación, lo que demuestra que puede ser utilizado para la difusión de catálogos y colecciones bibliográficas, museísticas e incluso archivísticas. Como aspecto interesante de los destacados en las

especificaciones del formato, se detecta un apartado vacío, relativo a los servicios web de MARC-XML. Esto es meramente indicativo de que la redifusión aplicada a MARC será posible en breve, tal y como se demuestra en la presente investigación, desarrollando la primera aplicación web que actúa sobre canales de sindicación en MARC-XML.

5.5. OPML

El formato OPML fue creado por Dave Winer en el año 2000, y se considera uno de los desarrollos más interesantes en cuanto a sindicación de contenidos se refiere. Esto se debe a que originalmente OPML *Outline Processor Markup Language* fue diseñado para servir de medio de intercambio de recursos entre aplicaciones web XML. Esta aplicación se extendió cada vez más y fue incluida en los navegadores a modo de extensión API (FINKE, C., 2009) especializada en la lectura de este tipo de archivos basados completamente en XML. En realidad OPML permite la organización de canales de sindicación y sus sitios web de producción, pero los usos originales se han propagado y extendido a la agrupación de recursos y fuentes de información categorizadas y pre-clasificadas. Esto implica un importante avance en la ordenación de los contenidos y de las fuentes de información electrónicas que se ven beneficiadas por un formato pensado exclusivamente para tal fin, pudiendo reflexionar que OPML al igual que las bibliografías de bibliografías, resulta ser una fuente de fuentes y por lo tanto un documento secundario. La primera versión operativa fue OPML1.0 en el año 2001, que en aquel entonces era considerado un borrador. Sus primeras aplicaciones fueron destinadas a los sistemas de podcasting (Radio UserLand: Tune Into Radio, 2009) que empleaban el formato OPML nativo para las retransmisiones y posteriormente en el sistema de publicación web Manila (USERLAND, 2009). No será hasta el año 2007, cuando se desarrolle OPML2.0, la versión definitiva del formato, objeto de análisis en este apartado.

5.5.1. Estructura del formato OPML

La estructura del formato OPML es una de las más sencillas dentro de los formatos de sindicación. Se compone de una etiqueta de apertura y cierre del formato *<opml version="2.0"></opml>* que comprende dos apartados, la cabecera *<head></head>* que describe el canal de sindicación y el cuerpo *<body></body>* que contiene la lista de recursos y canales de sindicación para su suscripción. No emplea Namespace, por lo que no consta validación alguna del formato de sindicación, ni siquiera existe schema XSD.

Elementos de head

El apartado de cabecera *<head></head>* permite describir el canal de sindicación OPML de forma básica a partir de las siguientes elementos.

- **Título – <title>.** Título del canal de sindicación OPML.

- **Fecha de publicación – <dateCreated>.** Fecha de creación del canal, según formato ISO8601 http://www.w3.org/TR/NOTE-datetime o RFC3339

- **Fecha de modificación – <dateModified>.** Fecha de modificación o actualización del canal según formato ISO8601 http://www.w3.org/TR/NOTE-datetime o RFC3339

- **Administrador – <ownerName>.** Cadena de texto que contiene el nombre del propietario del canal.

- **Correo del administrador – <ownerEmail>.** Correo electrónico del propietario del canal.

- **Identificador del administrador – <ownerId>.** Identificador del propietario del canal. Normalmente URL o URI del perfil o sitio web que pueda ser utilizado para identificar al autor.

- **Especificaciones – <docs>.** Etiqueta similar a la empleada por RSS2.0 que sirve para identificar mediante URI las especificaciones utilizadas para confeccionar el canal de sindicación OPML.

- **Estado de expansión – <expansionState>.** Permite expresar una lista de número que se expande correspondiente a la posición de los titulares o ítems del canal de sindicación. Dicho de otra forma, permite determinar un orden de expansión de los contenidos, mostrando unos contenidos de forma preferente sobre otros.

- **Scroll vertical – <vertScrollState>.** Contiene un número entero positivo que indica el número de entradas que serán visualizadas, de forma que pueda calcularse la expansión del scroll vertical de la ventana.

- **Marco superior – <windowTop>.** Determina la distancia en pixeles con respecto al borde superior de la ventana. Se emplea este elemento para encajar y visualizar las entradas en un espacio determinado de la ventana.

- **Marco izquierdo – <windowLeft>.** Determina la distancia en pixeles con respecto al borde izquierdo de la ventana. Se emplea este elemento para encajar y visualizar las entradas en un espacio determinado de la ventana.

- **Marco inferior – <windowBottom>.** Determina la distancia en pixeles con respecto al borde inferior de la ventana. Se emplea este elemento para encajar y visualizar las entradas en un espacio determinado de la ventana.

- **Marco derecho – <windowRight>.** Determina la distancia en pixeles con respecto al borde derecho de la ventana. Se emplea este elemento para encajar y visualizar las entradas en un espacio determinado de la ventana.

Elementos de body, el outline

La etiqueta *<body></body>* contiene el equivalente a las entradas en los formatos de sindicación, en OPML se denominan *outline* o perfiles. Cada perfil es una sóla etiqueta lo que evita su homóloga de cierre. La información se consigna en atributos de la propia etiqueta, presentando la siguiente estructura:

- **Recurso – <outline type="" text="" xmlUrl="" description="" htmlUrl="" language="" title="" version=""/ >.** El elemento *outline* está compuesto por diversos atributos que permiten la descripción somera de cualquier recurso o canal de sindicación. El atributo *type* hace referencia al formato del canal de sindicación, por ejemplo *rss, rdf, atom,* etc. El atributo *xmlUrl* resulta esencial para definir la URL o URI del canal de sindicación que identifique a su archivo XML. El atributo *text* se emplea para determinar una categoría o sección a la que pertenece el recurso descrito en el outline. El atributo *description* permite elaborar un breve resumen o descripción del canal de sindicación. El atributo *htmlUrl* define el sitio web del que depende el canal de sindicación. El atributo *language* define el idioma del recurso según códigos de idioma ISO639 (dos caracteres

indicativos de la lengua) e ISO3166 (tre caracteres indicativos del país). El atributo *title* define el título del canal de sindicación.

- **Categoría – <outline text=""><outline xmlUrl="" type="" text=""/></outline>.** El elemento *outline* también puede ser utilizado para agrupar subelementos *outline*. El requisito indispensable para ello es dotarlo del atributo *text* que tomara el valor de la categoría temática correspondiente. Embebidos entre <*outline*></*outline*> se sucederán todos los subelementos <*outline*/> de etiqueta única.

5.5.2. Extensibilidad de OPML

Las especificaciones de OPML también hacen referencia a la posibilidad de aplicar la extensibilidad. Como en los formatos anteriores, es necesario precisar el espacio de nombres en la etiqueta de apertura de la siguiente forma:

Título: Extensibilidad en OPML
Referencia: tabla76

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<opml xmlns:dc="http://purl.org/dc/elements/1.1/"
        xmlns:sy="http://purl.org/rss/1.0/modules/syndication/"
        xmlns:content="http://purl.org/rss/1.0/modules/content/"
        xmlns:prism="http://prismstandard.org/namespaces/1.2/basic/" version="2.0">

<head>…</head>

<body>…<outline text="" xmlUrl=""/>…</body>

</opml>
```

5.5.3. Ventajas e inconvenientes del formato OPML

El formato OPML tiene la ventaja esencial de ser el único formato de sindicación diseñado expresamente para la agrupación de recursos, especialmente terceros canales de sindicación. La sencillez de su estructura lo convierte en un formato de rápida configuración y edición, lo que facilita su implementación en aplicaciones web. Finalmente la capacidad para categorizar empleando los mismos elementos con una configuración básica mediante el atributo *text*, resulta útil, para organizar gran cantidad de recursos con la mayor economía del lenguaje. En

cuanto a sus inconvenientes destacar la carencia de Namespace, schema XSD o DTD que permita validar las estructuras de un archivo OPML.

Capítulo 6. Los programas parser

6.1. Funcionamiento de un programa parser

Un parser es un analizador sintáctico de patrones o estructuras predefinidas, que actúa sobre un archivo, cadena de caracteres, códigos, formatos o texto, de forma tal que es capaz de generar una pila ordenada de los elementos coincidentes con dicho patrón según su jerarquía y posición original, para su posterior acceso, selección y recuperación. Con independencia del patrón y de la fuente de datos que el parser analiza, también existen otras características que definen su funcionamiento, como el recorrido ascendente *bottom-up-parsing* o descendente *top-down-parsing*, por derivación *LL left to right, leftmost derivation* o por ampliación *LR left to right, rightmost derivation*.

Un parser es de recorrido ascendente, cuando parte de los elementos básicos de una estructura jerárquica de tal forma que desconoce por completo sus posibles relaciones ascendentes, con terceros elementos padres o ancestros, por lo que su orden de inferencia se basa en la ampliación de tales estructuras con las de los niveles superiores, hasta alcanzar el primer elemento de la jerarquía. Por este motivo un parser de tipo ascendente *bottom-up-parsin*g, también será de tipo *LR left to right, rightmost derivation* (CHAPMAN, N.P, 1987).

Un parser es de recorrido descendente, cuando parte del primer elemento de la estructura jerárquica de tal forma que establece relaciones con los elementos hijos y nietos mediante la derivación del análisis en cada uno de ellos de forma recursiva, hasta alcanzar los niveles inferiores de la jerarquía. Por este motivo un parser de tipo descendente top-down-parsing, también será de tipo *LL left to right, leftmost derivation* (GRUNE, D. and Jacobs, C., 1998).

En el caso de la sindicación de contenidos, los analizadores sintácticos parten de las estructuras propias de XML como patrones conocidos de comparación y análisis. Dado que XML es un lenguaje estructurado y anidado, resulta eminentemente jerárquico y de contexto gramatical conocido. Por estos motivos, los parser aplicados a XML son del tipo *LL left to right, leftmost derivation*.

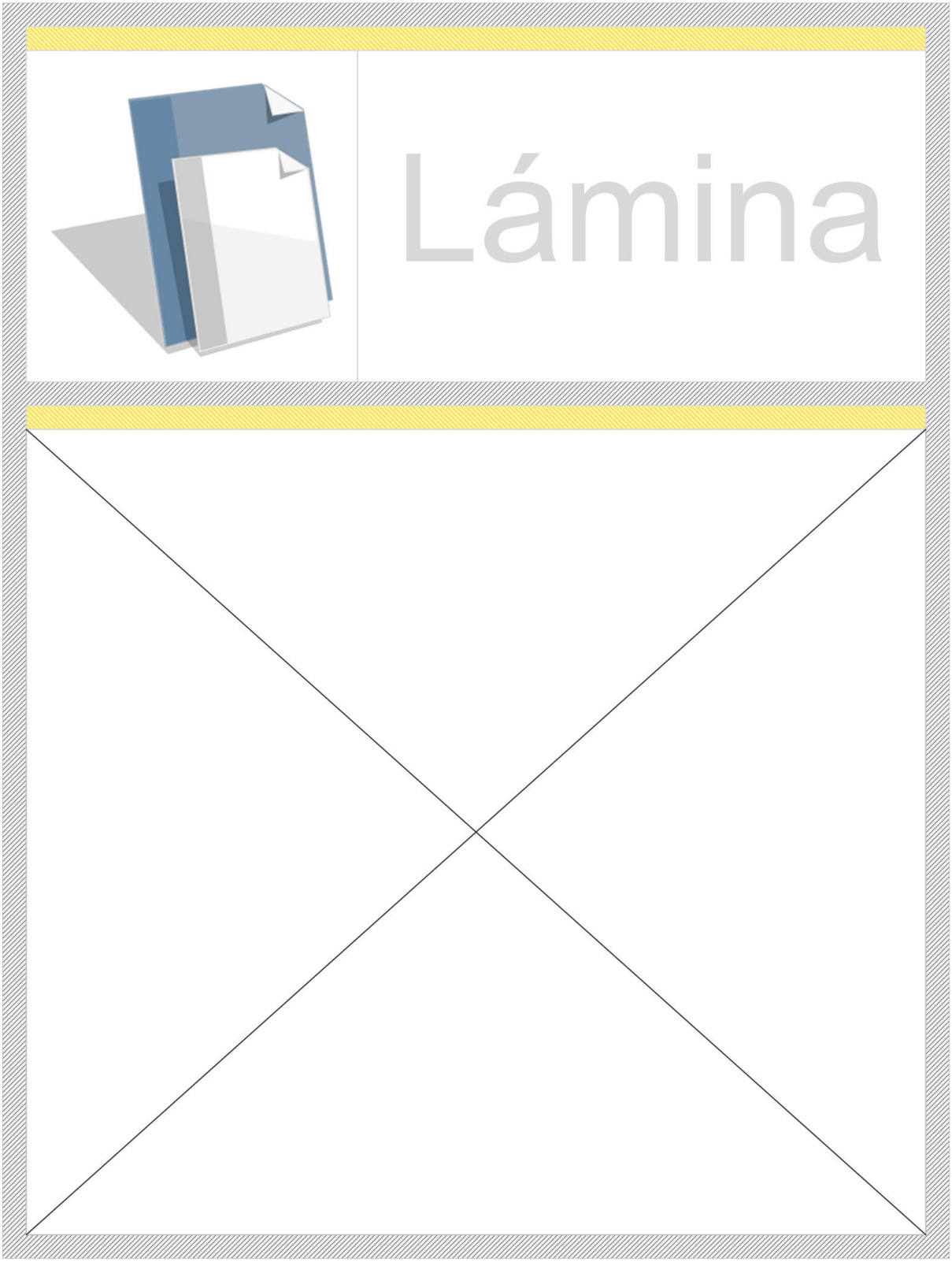
En esta investigación, se han desarrollado diversos parser en lenguaje PHP, dado que dispone de funciones específicas que abordan el proceso de análisis del lenguaje de marcado XML. De esta forma resulta más fácil su modificación, adaptación e implantación en el desarrollo de un sistema de sindicación para la gestión de colecciones bibliográficas.

Funcionamiento general de un parser XML

Un parser XML es un analizador sintáctico de estructuras anidadas de etiquetas. Ello significa que cualquier formato de sindicación es susceptible de ser analizado por este tipo de programas por el mero hecho de estar basados en XML. Por lo tanto el primer requisito para el funcionamiento de un parser XML es la disposición de un canal de sindicación que actúa como fuente de datos para el análisis. A continuación el sistema carga el archivo XML en búfer de memoria para empezar el análisis descendente de la estructura jerárquica. El primer paso es la detección de la cabecera XML, indicativa de que el archivo posee dicho dominio gramatical. Este paso resulta fundamental, puesto que determina la validación del lenguaje XML. A continuación se procede con un análisis descendente de la estructura jerárquica propia del formato de sindicación. Ello significa que tomará como punto de partida la primera etiqueta de apertura y cierre del formato.

En el caso de Atom *<feed></feed>*, en RSS1.0 *<rdf:RDF></rdf:RDF>*, en RSS2.0 *<rss></rss>*, en MARC-XML *<collection></collection>*, en OPML *<opml></opml>*. Esta primera confrontación también resulta clave, dado que las etiquetas de apertura del canal de sindicación contienen atributos *xmlns* para definir su propio namespace y el de los módulos que utilicen en todo caso, lo que puede facilitar la identificación de los juegos e etiquetas necesarios para interpretar el contenido del formato. No obstante, el parser por sí solo no entiende estas disquisiciones y únicamente anotará en su pila de elementos la existencia de atributos adscritos a la primera etiqueta del canal de sindicación. Recuérdese que al tratarse de un parser de análisis por derivación, comprobará las etiquetas de primer nivel jerárquico propias de la descripción del canal de sindicación y de las entradas de contenidos que lo conforman de forma secuencial y ordenada hasta agotar todas las alternativas posibles de derivación con el primer elemento del primer nivel jerárquico y sus sucesivos. De esta forma, el parser configura una pila de elementos que es esencialmente un *array de array* o una *matriz de matrices* que a modo de mapa de la estructura del canal de sindicación, permite acceder a sus contenidos, mediante expresiones de selección preferiblemente compuestas en lenguaje XPath o XQuery. Para almacenar los contenidos del canal de sindicación, también se necesita de un proceso denominado asignación de valores a variables, ello implica crear las variables propias de cada etiqueta para almacenar la información del formato. El método más ordenado y efectivo para lograrlo es la identificación de la variable a una función de selección expresada en XPath que actúa sobre el mapa estructural creado por el parser.

Título: Funcionamiento general de un parser XML
Referencia: figura21



Funcionamiento general de un parser XML

Referencia: figura21

Canal de sindicación

URL

http://www.ucm.es/articulos.xml

Archivo XML

INPUT

Parser XML

Carga del archivo XML

Análisis descendente de la estructura jerárquica

Identificación y validación de cabecera XML

<?xml version="1.0" encoding="UTF-8"?>

Identificación de la etiqueta de apertura y cierre

Atom	<feed></feed>
RSS1.0	<rdf:RDF></rdf:RDF>
RSS2.0	<rss></rss>
MARC-XML	<collection></collection>
OPML	<opml></opml>

Identificación de namespace

Atom	http://www.w3.org/2005/Atom
RSS1.0	http://purl.org/rss/1.0/
RDF	http://www.w3.org/1999/02/22-rdf-syntax-ns#
Syndication	http://purl.org/rss/1.0/modules/syndication/
Content	http://purl.org/rss/1.0/modules/content/
Dublin Core	http://purl.org/dc/elements/1.1/
PRISM	http://prismstandard.org/namespaces/1.2/basic/
MARC-XML	http://www.loc.gov/MARC21/slim
	http://www.loc.gov/standards/marcxml/schema/MARC21slim.xsd

Elementos de primer nivel jerárquico

Canal	Atom	• <id></id> • <title></title> • <updated></updated> • <author></author> • <link/> • <category/> • <contributor></contributor> • <generator></generator> • <icon></icon> • <logo></logo> • <rights></rights> • <subtitle></subtitle>
	RSS1.0	<channel></channel>
	RSS2.0	
	OPML	<head></head>
Entrada	Atom	<entry></entry>
	RSS1.0	<item></item>
	RSS2.0	<item></item>
	MARC-XML	<record></record>
Bolsa de entradas	OPML	<body></body>

Elementos de terceros niveles

Mapa en árbol de la estructura del archivo

Funcionamiento

- El modelo de funcionamiento de un parser XML es de tipo descendente (Top-Down-Parser), también conocido como analizador LL (Left to right, Lefmost derivation), debido a que la información está estructurada por una jerarquía de etiquetas que responde a un formato de sindicación.
- Ello significa que analiza sintácticamente toda cadena de caracteres que coincida con la forma de una etiqueta, desde la primera hasta la última, según se descienden niveles del anidamiento.
- Según se compara el archivo XML con el patrón básico de las etiquetas en XML, el nombre de dicha etiqueta queda registrado en una pila o array de arrays que configura el mapa estructural de los elementos del archivo.
- Este mapa posibilita el acceso a tales elementos mediante expresiones de selección como XPath o Xquery. Este método permite guardar los valores de cada elemento en nuevas variables con las que gestionar la información obtenida.

Leyenda

- Explicación del proceso
- Canal de sindicación
- Archivo XML
- Programa
- Función del programa
- Script o proceso
- Mapa jerárquico en árbol

Validación de archivo XML bien formado

- El parser XML lee obligatoriamente la primera línea del archivo, que deberá ser la declaración del lenguaje XML. En caso contrario el programa detectaría un error de lectura

Primera identificación del formato

- La primera etiqueta de apertura y cierre de un formato basado en XML, suele ser identificativa del mismo, tanto por su denominación, como por los atributos propios de los namespace de que consta. El mero hecho de acceder a esta primera etiqueta, permite conocer qué juego de etiquetas básicas se ha de utilizar para recuperar la información en el archivo.

Detección del juego de etiquetas del namespace

- Un Parser XML puede identificar a partir de la etiqueta de apertura y cierre, qué atributos xmlns se encuentran presentes en el archivo. De esta forma puede automatizarse todo el juego de etiquetas correspondientes a un determinado formato o módulo que trabaja en modo extensible y que por regla general suelen identificarse por su prefijo.

Selección y asignación de valores

- Una vez se crea el mapa de la estructura del archivo XML, se produce el proceso de selección de los elementos que lo constituyen. Esto se logra mediante el empleo de expresiones XPath o Xquery, con las que se navega a través del mapa estructural hasta obtener el valor del elemento deseado. Dicho valor es almacenado y asignado a una variable para usos posteriores.

Creación de variables de datos	Tantas como de elementos conste el formato analizado.
Creación de las expresiones de selección Xpath / XQuery	Nodos, valores atómicos, objetos, padres, hijos, hermanos, ancestros y descendientes.
Asignación de valores	A cada variable creada se le asigna una expresión de selección.

Esquema de asignación de valores a variables

Variable = Expresión Xpath → Mapa XML → Nodo / Elemento / Atributo → Valores

OUTPUT

Gestión de la información contenida en las variables

Lectura	• Impresión en pantalla de los contenidos de las variables.
Importación	• Importación de los contenidos de las variables a una base de datos para su explotación, edición y almacenamiento
Exportación	• Reorganización de los contenidos de las variables para generar terceros formatos, de cara a su exportación
Edición	• Es posible la edición de los contenidos antes de su exportación.

El resultado de su ejecución es la selección del contenido de dicha etiqueta y su asignación a la variable. Finalmente una vez asignados los valores a las variables, pueden emplearse para constituir un servicio de lectura del canal de sindicación, pueden importarse a una base de datos para su posterior edición o exportarse los contenidos a terceros formatos.

Patrones para el análisis de XML

Según lo explicado el analizador sintáctico parser, requiere de patrones que posibiliten la confrontación del canal de sindicación. Esto objetivo se logra de varias formas; por un lado determinando el inicio y final de la sintaxis de marcado en la estructura del formato de sindicación. Esto es posible debido a que los lenguajes de sindicación están diseñados en base a las normas y reglas principales de XML y sus complementos, permitiendo este tipo de opciones. El sistema detecta que todo elemento contenido entre *(<)* y *(>)* es una etiqueta correspondiente a un nivel jerárquico determinado en la estructura que no termina hasta encontrar el patrón *(</>)*. Por ende es factible esclarecer para el sistema que el contenido y la información principal se encuentra entre *(<>)* y *(</>)*. Este método tiene la ventaja de que es capaz de respetar completamente la estructura original del formato ya sea de sindicación o no, siempre y cuando mantenga el patrón definido.

Otra opción posible la constituye el uso de la especificación de documentos integrada en el código para la detección de la estructura determinada. Este método facilita la correspondencia entre elementos del documento y elementos designados dentro del parser. Aún siendo el método más sencillo, su principal desventaja es la imprevisión de las características y estructuras de los formatos de sindicación. De hecho no siempre se encuentran presentes las mismas etiquetas ni los mismos atributos, por lo que se recomienda su empleo en casos en los que la estructura del formato esté predefinida y no entrañe desconocimiento alguno para evitar la pérdida de información.

Finalmente el enfoque más exhaustivo lo constituye la carga de schemas XSD y DTD del formato, que ofrezcan la sintaxis y estructura completa del canal de sindicación. De esta manera el programa parser, asegura la correcta identificación de todos los formatos cuyo documento de descripción de tipos de datos esté referenciado. Aún así esto no es posible en todos los formatos como ya se ha podido comprobar en el capítulo de formatos de sindicación, ya que muchos de ellos aún teniendo un namespace, carecen del schema XSD, lo que dificulta su validación y en este caso su análisis.

Procesos auxiliares de depuración

Otro tipo de análisis que un sistema parser puede llegar a realizar es el análisis de la cadena de caracteres que se encuentra entre etiquetas, correspondiente a su contenido. Su razón de ser radica en la necesidad de evitar los problemas propios de la codificación, caracteres especiales y excepciones que se encuentran en los contenidos sitos entre las etiquetas de cada elemento de la estructura de la fuente de sindicación. Dicho de otra forma, el objetivo del análisis de caracteres es evitar la existencia de elementos no válidos que pueden producir errores de interpretación con respecto a las normas de descripción de datos y transformación en los lenguajes de marcado principalmente. Pero no es el único uso o empleo que tiene; es versátil en la depuración de la información correspondiente a campos o elementos sensibles como los enlaces y URI principalmente. En virtud a estas funciones y operaciones de análisis, se destaca que en este estadio el programa parser puede ser diseñado para la eliminación de caracteres o para la sustitución de los mismos, por otros que se designen como válidos y correctos.

Utilidades de un parser

Los parser son una de las aplicaciones más útiles para la sindicación de contenidos y para cualquier aplicación basada en XML. Ello se debe a la capacidad de manipular la información contenida en los archivos XML, para su posterior procesamiento, lectura, edición o consulta. De hecho los conocidos agregadores y lectores de feeds, están basados en los principios de los parser, para captar la información estructurada en los formatos de sindicación.

- **Lectura:** La función de lectura devuelve el contenido de las variables asignadas para cada elemento en el navegador. Esto permite construir esquemas de visualización para todos los campos de los que está compuesto el formato de sindicación. Los resultados pueden ser formateados por documentos XSLT, u hojas de estilo CSS en su defecto.

- **Consulta:** La función de consulta organiza la fuente de sindicación para su correspondiente filtrado según variables. Esto se consigue mediante el empleo de XPath o XQuery como lenguaje de recuperación embebido directamente en el código del parser. El resultado es la ejecución de la cadena de consulta introducida para interrogar un nodo concreto de la estructura del formato de sindicación, pudiendo enlazar a su vez con módulos de lectura de los resultados o de transformación.

- **Transformación:** Permite la transformación de la representación visual de la información, mediante un documento XSLT asociado u hoja de estilos que también es capaz de modificar el orden y los elementos que se desean visualizar. Esta aplicación suele estar unida a las funciones de lectura, aunque no se considera en el mismo punto, puesto que la transformación reporta cambios respecto al archivo original.

- **Importación:** Los valores almacenados en variables también pueden ser incorporados a una base de datos, de forma que puedan ser posteriormente editados, transformados o consultados.

- **Exportación:** Otra posibilidad se encuentra en la exportación de la información contenida en las variables. Ello implica el desarrollo de un programa generador de terceros formatos de sindicación que permita embeber el contenido extraído durante el proceso de análisis.

Todas estas posibilidades son explotadas en la aplicación de sindicación que se ha desarrollado para probar su validez de cara a la gestión de catálogos bibliográficos. Concretamente los parser son utilizados en los procesos de lectura e importación de canales de sindicación que como se podrá comprobar contienen el catálogo bibliográfico. Sin esta ayuda no se podrían sindicar colecciones bibliográficas, dado que no podrían ser interpretadas y aprovechadas por ninguna otra biblioteca.

6.2. Parser PHP Expat

El parser PHP Expat es un sistema de análisis sintáctico para procesar documentos XML mediante PHP. Su funcionamiento se basa en eventos prediseñados con los que se controla el funcionamiento del parser. Estos eventos permiten controlar la codificación del set de caracteres de las cadenas de texto, ejecutar un archivo XML, analizar datos de la estructura del documento, utilizar un namespace para controlar un conjunto de entidades externas o incluso controlar del procesamiento de los datos para su representación en pantalla. Por otra parte, PHP Expat automatiza el proceso de mapeado de la estructura del archivo XML que almacena en memoria, clasificándose así como un parser del tipo *top-down-parsing, LL left to right, leftmost derivation*.

La función *Expat* fue implementada por James Clark (CLARK, J., 2001), siendo utilizada inicialmente por Netscape y Perl como solución para el análisis de fuentes de sindicación. Su código fuente original fue escrito en C y adaptado posteriormente al lenguaje PHP. El verdadero potencial de Expat, reside en la posibilidad de generar tantos intérpretes y gestores de XML como se quieran, incluyendo el control de diferentes eventos en distintos archivos XML (PHP XML Expat Parser, 2007), pudiendo ajustarse a parámetros de extensión de cadenas de caracteres, control de ejecución desde las etiquetas de inicio y fin de formato, limitación del número de ítems, etc.

PHP Expat permite hacer las veces de parser mediante un código notablemente sencillo para leer y actualizar los canales de sindicación. La ventaja fundamental de su empleo es la facilidad de su instalación, puesto que únicamente requiere de un intérprete PHP, ya que las funciones de análisis están incorporadas en la librería original del lenguaje PHP.

Para comprender mejor el funcionamiento de este sistema parser, se ha seleccionado un código disponible en el directorio de código libre (FSF Free Software Directory, 2009). Procede del sistema de gestión de contenidos *typo3* (SKARHOJ, K., 2005). Dicho código está disponible en la *tabla4, página 58*.

En el código pueden observarse señaladas las funciones propias de PHP Expat (CLARK, J., 2007). PHP Expat emplea *funciones de intérprete XML, nativas en el propio lenguaje de programación PHP* (FETZER, T., 2004) siendo capaz de hacer las veces de un parser para el procesamiento de canales de sindicación. PHP Expat, desarrolla el siguiente análisis sobre el documento XML:

- **Análisis sintáctico:** Permite transformar un documento XML en una estructura en árbol. Analiza la mayor parte del documento y provee de acceso a sus correspondientes elementos. No analiza Schemas e ignora DTD.

- **Análisis de cadenas:** Permite visualizar documentos en XML, como una lista de eventos. Cuando un evento concreto sucede, llama directamente a una función de error y transformación para eliminar el fallo de la cadena o reemplazarlo por los caracteres válidos.

Si bien no asegura la validación de los canales de sindicación, garantiza su tratamiento y análisis para funciones de lectura, procesado y representación de la información. Para el caso de *lector1.php* presentado anteriormente, PHP-Expat, funciona de la siguiente manera:

1. Inicializa el programa parser al introducir la función *xml_parser_create()*
2. Crea funciones para usar los diferentes eventos, concretamente startTag, contents, endTag, que determinan el inicio y final de los contenidos, mediante patrones de comparación predefinidos.
3. Agrega la función *xml_set_element_handler()* para especificar que funciones deben ser ejecutadas cuando el parser encuentre abiertas y cerradas las etiquetas.
4. Agrega el gestor *xml_set_character_data_handler()* para especificar qué función se ejecutará cuando el parser encuentre la cadena de caracteres contenidos entre las etiquetas correspondientes al elemento; es decir los datos del contenido sindicado.
5. Analiza el archivo seleccionado en este caso, la fuente de sindicación elegida http://www.ucm.es/articulos.xml
6. En caso de error, con la función xml_error_string() se convierte el error XML en un error con descripción textual. En el caso concreto de *lector1.php*, se emplea una condición de

final de estructura *feof* y la función *die*, en la que se establece que el proceso sea parado para mostrar un valor determinado, en este caso un error en línea.

```
if(!(xml_parse($xml_parser, $data, feof($fp)))){die("Error en línea" .
xml_get_current_line_number($xml_parser));}
```

7. Finalmente se carga la función *xml_parser_free()* para liberar la memoria asignada a la función *xml_parser_create()*.

PHP Expat puede emplear además, otros eventos para el análisis canales de sindicación. Estudiando la documentación original disponible sobre la técnica Expat en PHP, se obtienen los siguientes eventos y funciones predeterminadas.

Título: Eventos del módulo nativo PHP-Expat
Referencia: tabla77

Funciones gestoras de PHP-Expat
xml_set_element_handler() Su empleo determina el comienzo y final de las etiquetas de los elementos que conforman la estructura de un documento XML.
xml_set_character_data_handler() Analiza la información disponible de caracteres es decir, todo aquel contenido que no está marcado en un documento XML. Esta función depende de la definición del tipo de documento DTD o schema tal y como esté definido, para conocer la orden de análisis de caracteres en un elemento dado.
xml_set_processing_instruction_handler() Función que permite asignar variables a todas las cadenas de caracteres que se encuentren entre las etiquetas de inicio y fin. Para ello emplea tres parámetros, a saber *int parser*, *string target*, *string data.*
xml_set_default_handler() Se considera el gestor de datos por defecto, que tiene como cometido el análisis de aquellos contenidos que no han sido analizados por el resto de gestores. Por ejemplo aquella información que queda fuera de las etiquetas de un elemento.
xml_set_unparsed_entity_decl_handler() Función gestora que asigna una denominación especial del tipo NDATA, a aquellos elementos que no contienen información es decir, están vacíos.

xml_set_notation_decl_handler()
Función para la gestión de elementos especiales en el documento XML, como anotaciones debidamente marcadas con los signos apropiados, quedando declarada como información a parte del contenido principal.

xml_set_external_entity_ref_handler()
Gestor de referencias externas, para incluir documentos XML, mediante URL.

Ventajas e inconvenientes del parser PHP Expat

El parser PHP Expat es un buen método para el análisis de canales de sindicación, dado que aporta múltiples posibilidades para controlar la ejecución del parser. No obstante, no se recomienda su empleo para el análisis de archivos XML complejos o extensos. La principal razón de esta afirmación es el almacenamiento en memoria de todas las estructuras y contenidos del archivo XML que analiza. Esta característica supone un problema si el canal de sindicación consta de miles de registros, tal y como se da el caso cuando se gestiona un catálogo bibliográfico. De cara a la investigación, PHP Expat no es lo suficientemente robusto como para poder ser utilizado con garantías, puesto que ralentiza la ejecución del propio script e incluso puede llegar a bloquear el propio navegador.

6.3. Parser PHP Simple XML y XPath

El parser PHP Simple XML también hace uso de las funciones de PHP para el análisis de archivos XML. La diferencia fundamental con respecto a PHP Expat es el método para analizar y almacenar la información. Si bien PHP Expat almacenaba toda la información del mapa estructural en memoria, PHP Simple XML permite almacenarlo como un objeto mediante DOM (Document Object Model) que en sí mismo constituye uno de los más completos analizadores en árbol que se han desarrollado para XML. La principal ventaja de este modelo es la accesibilidad de los contenidos mediante *array* y el empleo de expresiones XPath para seleccionar los contenidos de un determinado elemento de la estructura.

No obstante, para utilizar Simple XML, no siempre es necesario utilizar las funciones orientadas a objetos basadas en DOM. De hecho tan sólo es necesario utilizar la función *simplexml_load_file()* para cargar un canal de sindicación en memoria y permitir su selección y representación. Este es el caso del siguiente ejemplo:

Título: Ejemplo de parser simplexml_load_file()
Referencia: tabla78

```
// La variable $feed contiene la URL del archivo XML del canal de sindicación.
$feed = "url-canal-sindicación";

// La variable XML contiene la función de carga y lectura del parser Simplexml_load_file que actúa
// sobre el archivo definido en la variable $feed.
$xml = simplexml_load_file($feed);

// Para extraer los contenidos etiquetados se necesita emplear un bucle que actúa sobre todos los
// elementos disponibles en el canal.
for($i=0; $xml->item[$i]; $i++) {

    // La variable $title almacena la información del título de cada elemento del canal. Para ello, se
    // emplea el método de selección XPath explicado en el capítulo de bases tecnológicas de la
    // sindicación de contenidos.
    $title = $xml->item[$i]->xpath('//dc:title');

    // Una vez la variable $title contiene el valor del título, se imprime en pantalla.
    echo "<p><h3>Title:</h3> $title[$i]<br/></p>

}
```

Como puede observarse, la función *simplexml_load_file()* es suficiente para cargar el canal de sindicación en memoria que posteriormente será filtrado por la instrucción **`$title = $xml->item[$i]->xpath('//dc:title');`** que permite asignar a la variable *$title* el contenido de la etiqueta *<dc:title></dc:title>*. Al igual que en el caso de PHP Expat, el presente ejemplo peca del mismo problema de carga en memoria. Por lo tanto su empleo tiene que ser reducido a aquellos casos en los que el canal de sindicación no sea extenso ni complejo para evitar problemas de ejecución en el navegador web. A pesar de ello este modelo plantea ventajas con respecto al parser PHP Expat, concretamente la posibilidad de configurar de forma independiente, las variables y expresiones de selección de los contenidos del archivo XML. De esta forma, pueden configurarse parsers que respondan a las necesidades de cada sistema, definiendo qué contenidos del canal serán seleccionados y cuáles no.

No obstante el problema de la carga en memoria puede ser solucionado, utilizando un parser basado en la función *SimpleXMLElement(),* (Php: The SimpleXMLElement class Manual, 2009) que sí implica la ejecución de DOM y la creación de un mapa de nodos y elementos de la estructura del canal de sindicación. Esta operación no tiene los mismos requerimientos de memoria que los casos anteriores, en los que todo el contenido del canal de sindicación era cargado. Tal y como muestra el siguiente ejemplo, la función *new SimpleXMLElement()*

indica la creación de un nuevo objeto, lo que plantea un almacenamiento en *array de arrays*. Este tipo de estructuras permite un acceso más rápido a los elementos, puesto que dicho array se configura a partir de los nodos y elementos de los que está constituido el canal de sindicación. En cuanto al método de asignación de valores a variables, se mantiene el mismo planteamiento que en el parser anterior. Se defina una ruta XPath que filtra el contenido del elemento seleccionado.

Título: Ejemplo de parser *simpleXMLElement()*
Referencia: tabla79

```
// La variable $feed contiene la URL del archivo XML del canal de sindicación.
$feed = "url-canal-sindicación";

// La variable $sxe contiene la función parser DOM para analizar el canal de sindicación.
$sxe = new SimpleXMLElement($feed, NULL, TRUE);

// La variable $title ejecutar el parser simpleXMLElement sobre la etiqueta dc:title para extraer todos
// los títulos del canal de sindicación.
$title = $sxe->xpath("//dc:title");

// Para imprimirlos en pantalla se requiere de un bucle que funcione hasta agotar todos los
// contenidos de la pila de la variable $title.
foreach ($title as $control) {
      echo "<p><h3>Title:</h3> $control<br/></p>
}
```

6.4. Prueba de identificación de las estructuras básicas de los formatos de sindicación mediante el empleo del parser simpleXMLElement

El presente apartado tiene por objetivo determinar si las propiedades de extensibilidad de los diferentes formatos de sindicación, podrían ser reconocidas directamente por el principal parser XML del lenguaje PHP. Es decir, si la propiedad de extensibilidad puede hacer que las distinciones entre formatos sean borrosas, dado que cualquier formato puede ser aplicado sobre cualquier otro, tal y como se ha explicado en el capítulo 5 de formatos de sindicación.

Por otro lado, la prueba con el parser *simpleXMLElement* (Php: The SimpleXMLElement class Manual, 2009) permite comprobar cuál es su funcionamiento práctico, así como distinguir el mapa de nodos y elementos accesible mediante arrays.

Como introducción a esta prueba, hay que recordar que los formatos de sindicación de contenidos han sido desarrollados a partir de una sintaxis y etiquetas básicas para estructurar

diversos contenidos. Tales estructuras pueden ser ampliadas y completadas adaptándose a las necesidades de los recursos a describir. Esto se logra por medio de la propiedad de extensibilidad del lenguaje XML que permite la introducción de módulos y terceros lenguajes por medio de la declaración de Namespace, es decir, a partir de sus propios schemas XSD o DTD. Dado que en las especificaciones de los formatos de sindicación se contempla este supuesto, se hace necesario encontrar un método de comparación de los elementos y etiquetas básicas de cada uno de ellos. Esto se consigue con el empleo del parser anteriormente mencionado, resultando suficientemente amplio por ser utilizado por la comunidad de programadores de PHP, sencillo por no requerir más de tres líneas de código, normalizado al ser reconocido en el manual oficial del lenguaje y extendido para el entorno web, dado que es su medio de funcionamiento.

Como se viene explicando la clase *simpleXMLElement* contiene una librería de funciones que permiten la lectura y análisis de cualquier archivo XML y por ende de cualquier formato de sindicación. Las funciones de la clase *simpleXMLElement* convierten cualquier archivo XML en un objeto al que se puede acceder mediante arrays. Esta característica permite conocer de primera mano qué elementos de la estructura del archivo son reconocidos y cuáles no.

Título: Parser para identificar la configuración básica de los formatos de sindicación
Referencia: tabla80

```
1  $feed = "feed.xml";
2  $xml = new SimpleXMLElement(file_get_contents($feed));
3  print_r($xml);
```

El código del parser se resume en tres líneas:

- **Línea 1:** Define una variable *$feed* que contiene la URL del canal de sindicación.
- **Línea 2:** Introduce una variable *$xml* que incluye la función SimpleXMLElement que permite convertir el archivo XML del canal de sindicación en un objeto DOM. Para asegurar el proceso de conversión a objeto, la función embebida *file_get_contents* permite descargar toda la estructura del archivo XML correspondiente al canal de sindicación.

- **Línea 3:** La constituye la función *print_r* que permite imprimir en pantalla el array completo correspondiente a la variable *$xml*, visualizando de esta forma toda la estructura y elementos que es capaz de analizar el parser.

Empleando este parser como referencia y control de los canales de sindicación, el siguiente paso será someter a los formatos de sindicación a una prueba de análisis y lectura que efectuará automáticamente el programa. Los formatos seleccionados para la prueba son Atom, RSS1.0 RDF, RSS2.0 y MARC-XML.

Para que la prueba sea válida se ha optado por crear canales de sindicación lo más completos posibles en cuanto a su estructura, según sus especificaciones. Además se han introducido por extensibilidad en todos los casos los mismos módulos, manteniendo una estructura común definida en la *tabla81*.

Título: Estructuras extensibles comunes a todos los formatos
Referencia: tabla81

xmlns:sy="http://purl.org/rss/1.0/modules/syndication/"
xmlns:content="http://purl.org/rss/1.0/modules/content/"
xmlns:prism="http://prismstandard.org/namespaces/1.2/basic/"
xmlns:dc="http://purl.org/dc/elements/1.1/

<sy:updatePeriod>periodo de actualización**</sy:updatePeriod>**
<sy:updateFrequency>frecuencia de actualización**</sy:updateFrequency>**
<sy:updateBase>fecha de actualización**</sy:updateBase>**
<dc:identifier>identificador de la entrada**</dc:identifier>**
<dc:subject>temática o materia de la entrada**</dc:subject>**
<dc:creator>autor de la entrada**</dc:creator>**
<dc:contributor>colaborador de la entrada**</dc:contributor>**
<dc:date>fecha de publicación de la entrada**</dc:date>**
<dc:source>url de la fuente de origen de la entrada**</dc:source>**
<dc:language>lenguaje o idioma de la entrada**</dc:language>**
<dc:relation>url de un recurso relacionado con la entrada**</dc:relation>**
<dc:rights>derechos de la entrada**</dc:rights>**
<prism:volume>número de volumen**</prism:volume>**
<prism:number>número de la publicación**</prism:number>**
<prism:section>sección**</prism:section>**
<prism:startingPage>página de inicio**</prism:startingPage>**
<prism:endingPage>página de finalización**</prism:endingPage>**
<content:encoded>contenido codificado en xhtml</content:encoded>

En todo caso cada canal dispone de una descripción de cada etiqueta y elemento de su estructura, de forma que pueda ser visualizado en pantalla al ejecutar la función *print_r*.

La prueba también se encuentra accesible en red, para verificar que los resultados obtenidos se consiguen tanto en el entorno local como en un servidor remoto operativo. Para comprobarlo basta con cargar en el navegador la siguiente dirección, http://mblazquez.es/docs/prueba-formatos-sindicacion.php, dado que estas pruebas han sido verificadas en red.

Los resultados que se pueden consultar en las siguientes tablas, constituyen también la mejor demostración del funcionamiento de un parser creando un mapa de nodos y elementos.

Título: Mapa reconocido por el parser simpleXMLElement en el formato Atom
Fuente: http://mblazquez.es/docs/atom.xml
Referencia: tabla82

SimpleXMLElement Object (

[id] => identificador del canal
[updated] => fecha de actualización
[title] => título
[subtitle] => subtítulo o descripción del canal

[link] => SimpleXMLElement Object ([@attributes] => Array (**[href]** => url del canal))

[author] => SimpleXMLElement Object (
[name] => nombre y apellidos del autor del canal
[uri] => url del perfil del autor del canal
[email] => correo electrónico del autor del canal)

[contributor] => SimpleXMLElement Object (
[name] => nombre y apellidos del colaborador del canal
[uri] => url del perfil del colaborador del canal
[email] => correo electrónico del colaborador del canal)

[generator] => nombre del programa generador del canal
[icon] => url del icono del canal
[logo] => url del logo del canal
[rights] => derechos del canal

[entry] => SimpleXMLElement Object (
[id] => identificador de la entrada
[published] => fecha de publicación
[updated] => fecha de actualización
[category] => SimpleXMLElement Object ([@attributes] => Array (**[term]** =>
categoría temática))

[link] => SimpleXMLElement Object ([@attributes] => Array (**[href]** => url de la
entrada))

[title] => título de la entrada
[content] => contenido de la entrada
[summary] => resumen o sumario de la entrada
[author] => SimpleXMLElement Object (
[name] => nombre y apellidos del autor

[uri] => url del perfil del autor
[email] => correo electrónico del autor)

[contributor] => SimpleXMLElement Object (
[name] => nombre y apellidos del colaborador
[uri] => url del perfil del colaborador
[email] => correo electrónico del colaborador)))

Título: Mapa reconocido por el parser simpleXMLElement en el formato RSS1.0 RDF
Fuente: http://mblazquez.es/docs/rss1.xml
Referencia: tabla83

SimpleXMLElement Object (
[channel] => SimpleXMLElement Object (
[title] => título del canal
[link] => url del canal
[description] => descripción del canal
[image] => SimpleXMLElement Object ( )
[items] => SimpleXMLElement Object ( ))
[image] => SimpleXMLElement Object (
[title] => título del canal
[link] => url del canal para el enlace del logotipo
[url] => url del logotipo del canal)
[item] => SimpleXMLElement Object (
[title] => título de la entrada
[link] => url de la entrada
[description] => contenido de la entrada))

Título: Mapa reconocido por el parser simpleXMLElement en el formato RSS2.0
Fuente: http://mblazquez.es/docs/rss2.xml
Referencia: tabla84

SimpleXMLElement Object ([@attributes] => Array (
[version] => 2.0)

[channel] => SimpleXMLElement Object (
[title] => título del canal
[link] => url del canal
[description] => descripción del canal
[copyright] => derechos del canal
[pubDate] => fecha de publicación del canal
[generator] => programa generador del canal

[item] => SimpleXMLElement Object (
[title] => título de la entrada
[link] => url de la entrada
[comments] => comentarios de la entrada
[pubDate] => fecha de publicación de la entrada
[category] => categoría temática de la entrada
[guid] => identificador de la entrada
[description] => contenido de la entrada)))

Título: Mapa reconocido por el parser simpleXMLElement en el formato MARC-XML
Fuente: http://mblazquez.es/docs/marc.xml
Referencia: tabla85

SimpleXMLElement Object (

[record] => SimpleXMLElement Object (
[leader] => cabecera
[controlfield] => Array (
[0] => número de control
[1] => identificador del número de control
[2] => fecha y hora de la última actualización)

[datafield] => Array (
[0] => SimpleXMLElement Object ([@attributes] => Array (
[tag] => 010 **[ind1]** => **[ind2]** =>)
[subfield] => número de control de la biblioteca del congreso)

[1] => SimpleXMLElement Object ([@attributes] => Array (
[tag] => 017 **[ind1]** => **[ind2]** =>)
[subfield] => número de depósito legal)

[2] => SimpleXMLElement Object ([@attributes] => Array (
[tag] => 020 **[ind1]** => **[ind2]** =>)
[subfield] => isbn international standard book number)

[3] => SimpleXMLElement Object ([@attributes] => Array (
[tag] => 041 **[ind1]** => **[ind2]** =>)
[subfield] => código de idioma)
[4] => SimpleXMLElement Object ([@attributes] => Array (
[tag] => 050 **[ind1]** => 1 **[ind2]** => 0)
[subfield] => signatura topográfica)

[5] => SimpleXMLElement Object ([@attributes] => Array (
[tag] => 080 **[ind1]** => **[ind2]** =>)
[subfield] => número de CDU clasificación decimal universal)

[6] => SimpleXMLElement Object ([@attributes] => Array (
[tag] => 082 **[ind1]** => 0 **[ind2]** =>)
[subfield] => número de clasificación decimal DEWEY)

[7] => SimpleXMLElement Object ([@attributes] => Array (
[tag] => 100 **[ind1]** => 0 **[ind2]** =>)
[subfield] => Array (
[0] => asiento principal entidad personal
[1] => SimpleXMLElement Object ([@attributes] => Array (
[code] => 0))))

[8] => SimpleXMLElement Object ([@attributes] => Array (
[tag] => 110 **[ind1]** => 2 **[ind2]** =>)
[subfield] => Array (**[0]** => asiento principal entidad corporativa
[1] => SimpleXMLElement Object ([@attributes] => Array (
[code] => 0))))

[9] => SimpleXMLElement Object ([@attributes] => Array (
[tag] => 111 **[ind1]** => 2 **[ind2]** =>)
[subfield] => Array (
[0] => asiento principal nombre de reunión

[1] => SimpleXMLElement Object ([@attributes] => Array (
[code] => 0))))

[10] => SimpleXMLElement Object ([@attributes] => Array (
[tag] => 130 **[ind1]** => 0 **[ind2]** =>)
[subfield] => Array (
[0] => asiento principal título uniforme
[1] => SimpleXMLElement Object ([@attributes] => Array (
[code] => 0))))

[11] => SimpleXMLElement Object ([@attributes] => Array (
[tag] => 245 **[ind1]** => 1 **[ind2]** => 0)
[subfield] => Array (
[0] => título
[1] => subtítulo
[2] => mención de responsabilidad))

[12] => SimpleXMLElement Object ([@attributes] => Array (
[tag] => 250 **[ind1]** => **[ind2]** =>)
[subfield] => Array (
[0] => edición
[1] => mención de edición))

[13] => SimpleXMLElement Object ([@attributes] => Array (
[tag] => 260 **[ind1]** => **[ind2]** =>)
[subfield] => Array (
[0] => lugar de publicación
[1] => editorial
[2] => fecha de publicación))

[14] => SimpleXMLElement Object ([@attributes] => Array (
[tag] => 300 **[ind1]** => **[ind2]** =>)
[subfield] => Array (
[0] => extensión
[1] => otros detalles físicos
[2] => dimensiones
[3] => materiales adjuntos))

[15] => SimpleXMLElement Object ([@attributes] => Array (
[tag] => 490 **[ind1]** => 0 **[ind2]** =>)
[subfield] => Array (
[0] => título de la serie
[1] => número o volumen de la serie))

[16] => SimpleXMLElement Object ([@attributes] => Array (
[tag] => 500 **[ind1]** => **[ind2]** =>)
[subfield] => nota general)

[17] => SimpleXMLElement Object ([@attributes] => Array (
[tag] => 650 **[ind1]** => 1 **[ind2]** => 4)
[subfield] => Array (
[0] => asiento secundario de materia
[1] => número de control de materia))

[18] => SimpleXMLElement Object ([@attributes] => Array (
[tag] => 700 **[ind1]** => 0 **[ind2]** =>)
[subfield] => Array (
[0] => asiento secundario de entidad personal
[1] => número de control de entidad personal))

```
[19] => SimpleXMLElement Object ( [@attributes] => Array (
[tag] => 710 [ind1] => 2 [ind2] => )
        [subfield] => Array (
                [0] => asiento secundario de entidad corporativa
                [1] => número de control de entidad corporativa ) )

[20] => SimpleXMLElement Object ( [@attributes] => Array (
[tag] => 887 [ind1] => [ind2] => )
        [subfield] => registro bibliográfico completo ) ) ) )
```

Los resultados indican que el empleo de la extensibilidad en los formatos de sindicación si bien es posible, dificulta e imposibilita la lectura de las etiquetas en los parser básicos como el que se ha empleado con el método *simpleXMLElement()*. Esto significa que aquellos módulos y formatos que utilizan prefijos propios del espacio de nombres namespace, como Dublin Core (prefijo dc:*), Syndication (prefijo sy:*), Content (prefijo content:*), PRISM (prefijo prism:*), entre otros, no son legibles por los parser básicos. De hecho solo las etiquetas sin prefijo propias de cada formato de sindicación sí han sido en todo caso analizadas, lo que plantea un método válido para comparar los formatos de sindicación en su configuración básica.

Hay que precisar que si bien los módulos y formatos que emplean prefijos como identificación de su espacio de nombres sí pueden ser analizados convenientemente mediante la modificación del parser empleado, también es verdad que los resultados obtenidos responden a la configuración básica de los formatos de sindicación, tal y como estipulan sus especificaciones, con independencia del formato o módulo que a la postre se desee implementar.

También hay que analizar cómo el parser efectúa el proceso de reconocimiento y análisis de los elementos de cada formato. Como se puede observar, siempre se genera un objeto para la estructura de cada formato. Ello queda patente cuando se imprime el resultado del array de arrays *$xml*, visualizando en todo caso el indicativo *SimpleXMLElement Object*. Tal objeto almacena arrays asociativos que corresponden al nombre de las etiquetas entre corchetes que componen el canal, devolviendo a su vez el contenido de dicha etiqueta, indicado mediante una flecha, *[title] => título del canal*.

En los resultados obtenidos, también es posible apreciar la estructura jerárquica del formato y cómo es asimilada por el parser que devuelve una cadena anidada de arrays que coincide igualmente.

En el caso de MARC-XML, el nivel de jerarquización es mucho mayor y se da la posibilidad de comprobar que cuando se suceden varias cadenas de arrays, dicho término aparece explícitamente impreso en pantalla, como por ejemplo en la siguiente cadena: *Array ([0] => título [1] => subtítulo [2] => mención de responsabilidad)*.

En resumen y conclusión a este capítulo y apartado, se destaca la importancia de los programas parser para la sindicación de contenidos, puesto que constituyen el núcleo invisible que permite su funcionamiento en navegadores web, lectores de noticias, agregadores y demás aplicaciones que se alimentan de los canales de sindicación.

También queda patente que cualquier formato XML con información puede ser considerado un formato de sindicación si se emplea en el contexto de difusión de sus contenidos. Para permitir precisamente la difusión de sus contenidos, se necesitan mecanismos que permitan recuperar y visualizar la información de dicho formato. En este sentido la existencia de los parser es consustancial al mero hecho de sindicar contenidos, puesto que supone interpretar los formatos en los que se contiene la información. Este caso sucede con MARC-XML, que constituye un formato de descripción bibliográfica basado en XML que bien puede emplearse para la transmisión de registros bibliográficos producidos por los bibliotecarios y profesionales de la información, cuya información está destinada a un público objetivo, lector e investigador, potencial consultor del catálogo bibliográfico generado en dicho formato. Si bien no es un formato tradicional de sindicación, puede ser interpretado al igual que el resto de formatos Atom, RSS1.0 RDF y RSS2.0, mediante el empleo de un parser como el utilizado en la prueba de este apartado. Esto permite afirmar que el soporte tecnológico es clave para desarrollar nuevos formatos de sindicación aplicados a las bibliotecas y en especial a sus servicios documentales.

Capítulo 7. Aplicaciones de la sindicación para la gestión de catálogos bibliográficos

7.1. Aplicaciones y necesidad de desarrollo

En esta investigación se proponen nuevas aplicaciones para la sindicación, que rompen de forma radical el modo de empleo tradicional de esta técnica. Concretamente se plantea la difusión e intercambio de catálogos bibliográficos completos mediante sindicación de contenidos, de forma tal que entre diversas bibliotecas puedan crearse catálogos colectivos extensibles y libremente accesibles desde cualquier navegador web. Esto implicaría mejoras en los procesos de migración de datos, la creación de repertorios universales y la deslocalización de las colecciones bibliográficas, obteniendo la categoría de catálogo electrónico propiamente dicho, puesto que digitalmente resulta tangible en una serie archivos XML que configurarían el canal de sindicación. Por otro lado permitiría que los usuarios de una biblioteca pudieran suscribirse a las secciones temáticas delimitadas por la clasificación, o por el contrario la suscripción a la colección completa. La sindicación en este caso, si bien mantendría el principio de redifusión de contenidos, no utilizaría el mismo volumen de datos y como se explicará más adelante, tampoco empleará los mismos formatos de sindicación. Por otro lado esta aplicación lograría resolver el problema de la difusión de los fondos bibliográficos y sus contenidos, muchas veces desaprovechados por desconocimiento, ampliando su visibilidad y favoreciendo la cooperación bibliográfica entre bibliotecas que pudiendo suscribirse a un canal puedan ser capaces de efectuar una pre-catalogación de los documentos que se adquieran. También supone una mejora en los servicios de información y referencia, al poder aplicarse métodos de filtrado y recuperación de información sobre las colecciones sindicadas a gran escala.

La segunda utilidad que se va a exponer versa sobre el modo de empleo de la sindicación como un canal bidireccional de redifusión de información. Dicho de otra forma, transmitir información estructurada a un usuario y recibir sus peticiones, logrando canales de sindicación capaces de interactuar según las necesidades de información del usuario, ejecutando servicios documentales e información simultáneamente. Esta aplicación se puede aprovechar sobre los catálogos bibliográficos sindicados, permitiendo su gestión, edición, catalogación, consulta y ordenación desde un canal de sindicación. Esta utilidad implicaría que los servicios propios de la biblioteca podrían resumirse en un canal de sindicación que en

sí mismo los engloba junto con el fondo bibliográfico también sindicado. La aplicación de estas utilidades puede suponer una simplificación de los mecanismos y soportes tecnológicos empleados en las bibliotecas para llevar a cabo todas las funciones mencionadas, entre ellas el protocolo Z39.50 como método de recuperación bibliográfico, que podría ser sustituido por una alternativa de sindicación presumiblemente más sencilla de implantar.

7.2. Especificaciones del sistema

Especificación de funciones

- Sindicar catálogos bibliográficos en un formato de sindicación que permita la completa descripción del contenido.
  - Analizar formalmente la sindicación de contenidos y sus formatos.
  - Analizar la estructura de los formatos extensibles para la descripción y transmisión de registros bibliográficos.
  - Crear colecciones bibliográficas de prueba de diferentes volúmenes y tamaños de forma que puedan ser sindicadas en los formatos extensibles de sindicación que se pretenden analizar.
  - Procesar las colecciones bibliográficas en base de datos y generar colecciones bibliográficas en los distintos formatos extensibles de sindicación.
  - Suscribir e importar las colecciones bibliográficas sindicadas en los distintos formatos extensibles para verificar la capacidad de lectura y transferencia de registros bibliográficos utilizando el método de sindicación de contenidos.

- Importar y exportar colecciones bibliográficas en formatos de sindicación para demostrar las capacidades de transmisión de datos bibliográficos.
  - Efectuar mediciones de tiempo de ejecución en las operaciones señaladas anteriormente con los distintos formatos extensibles de sindicación para cada una de las colecciones bibliográficas sindicadas.
  - Determinar en virtud de las pruebas efectuadas y de la estructura descriptiva de cada formato, cuál es el más adecuado para servir de medio de transmisión bibliográfica.

- Sindicar servicios de gestión, edición y consulta similares a los proporcionados por Z39.50.
  - Desarrollar un sistema análogo a Z39.50 capaz de operar los servicios básicos de recuperación, ordenación y edición sobre catálogos bibliográficos, utilizando como medio de comunicación fundamental canales de sindicación de contenidos.
  - Permitir la suscripción a un canal de sindicación de servicios, para efectuar las funciones propias de la edición, consulta y actualización de los catálogos bibliográficos.
  - Crear un canal de sindicación de respuestas a las peticiones que efectúe un eventual usuario desde el canal de sindicación de servicios.
  - Crear un interfaz que permita representar en cualquier navegador y sistema la información bibliográfica estructurada en los formatos de sindicación, transmitidos al usuario.

- Conseguir una comunicación bidireccional entre las peticiones y las respuestas del cliente y servidor mediante sindicación y XML.
  - Lograr un método de comunicación basado en el binomio XML y http desde los canales de sindicación hacia el servidor y viceversa.
  - Conseguir remitir parámetros e información del lado del usuario a partir de las respuestas XML que visualiza con el sistema.
  - Manipular información bibliográfica utilizando tal método de comunicación.

Especificaciones técnicas

- El sistema requerirá instalación por parte de la biblioteca pero no por parte del usuario, clientes de los servicios, de forma tal que sólo sea necesario el empleo de un navegador web para acceder a los servicios que tradicionalmente oferta Z39.50.

- El sistema dispondrá de una interfaz gráfica capaz de representar los servicios, las peticiones y las respuestas, de forma transparente y directa.

- Los requerimientos de instalación del lado del servidor serán los mínimos exigibles en cualquier distribución Apache, PHP y MySQL para simplificar al máximo su empleo y reducir los costes de configuración y adaptación.

- Se evitará en todo caso la instalación de librerías y clases de funciones de programación, que no hayan sido incluidas de serie en las principales distribuciones autoinstalables disponibles en código abierto, para asegurar su funcionamiento bajo cualquier sistema operativo y distribución de servidor básica.

- El sistema ha de ser capaz de gestionar colecciones bibliográficas pequeñas de entre 1000 y 5000 registros, hasta grandes colecciones circunscritas entre 500000 y 1000000 de registros.

- El sistema ha de ser capaz de gestionar colecciones bibliográficas utilizando métodos de comunicación de las peticiones y respuestas basados únicamente en el protocolo http y en el lenguaje XML.

Especificaciones documentales

- El sistema debe ser capaz de generar catálogos y colecciones bibliográficas en distintos formatos extensibles para su sindicación a saber: RSS1.0 RDF, RSS2.0, ATOM y MARC-XML.

- El sistema permitirá la transferencia, exportación e importación de colecciones bibliográficas sindicadas.

- Se permitirá la recuperación y sindicación de los resultados de las consultas efectuadas.

- El sistema facilitará los servicios documentales básicos de edición de registros bibliográficos del catálogo, la inserción de otros nuevos, la consulta y refinamiento de la misma, la capacidad de selección bibliográfica, la exportación y representación de los resultados, así como la actualización de los catálogos bibliográficos, utilizando únicamente canales de sindicación de contenidos.

Especificaciones de diseño

- El sistema contemplará dos aplicaciones diferenciadas. La primera, denominada SYNC, actuará como plataforma de contenidos sindicados. La segunda, denominada SYNCORE, contendrá el sistema de comunicación, procesamiento de peticiones y respuestas mediante sindicación.

- La plataforma de contenidos sindicados SYNC, constituye la parte del sistema encargada de la creación de colecciones bibliográficas sindicadas, así como su exportación, importación, pruebas de transferencia, lectura, así como las mediciones pertinentes de los tiempos de ejecución de tales funciones. Aportará un área de trabajo para las distintas colecciones bibliográficas en los diferentes formatos extensibles de sindicación, así como el procesamiento de la información en base de datos, facilitando su acceso y organización.

- La aplicación SYNCORE, también denominada programa núcleo, se diseñará como un sistema autónomo que comprende el mecanismo de comunicación mediante petición y respuesta basado, en http y XML, que posibilita operar los distintos servicios de gestión bibliográfica a partir de canales de sindicación de contenidos. SYNCORE será capaz de contactar con cualquier plataforma SYNC, que mantenga las especificaciones con las que están diseñadas, procesando y devolviendo respuestas a un usuario remoto. SYNCORE incorporará en su esquema las funciones de consulta, edición, selección, actualización y exportación de registros, catálogos y autoridades, por lo que el acceso a cada una de ellas deberá ser completamente simplificado. SYNCORE se diseñará bajo un esquema modular en cuanto a su estructura, siendo clave para su gestión un método de redirección de peticiones de servicios, con el que se han concebido el orden de ejecución de sus funciones.

Especificaciones del diseño modular

- El diseño del sistema en conjunto es de tipo modular, especialmente en el caso de SYNCORE, el cual está preparado para acoger módulos externos, terceros desarrollos o mejoras sin instalación previa, de forma que el empleo de cada nueva funcionalidad o servicio bibliográfico sea instantáneo.

- En relación a la plataforma de sindicación SYNC consta de una estructura modular en cuanto a la distribución de los programas disponibles.

Especificación de estándares

- Con la finalidad de facilitar la adaptación e implementación en el entorno bibliotecario a efectos de ser una alternativa a Z39.50, se cuidarán los estándares a utilizar, en cuanto a formatos de descripción, normas ISO, metodología de desarrollo, etc.
  - Se asumirán los estándares: ISO 2709 Documentation - Format for Bibliographic Information Interchange on Magnetic Tape, MARC21 Machine Readable Cataloging e ISO8601 Data elements and interchange formats.

  - Se revisarán los estándares: ANSI/NISO Z39.53-2001 *Codes for the Representation of Languages for Information Interchange*, de forma que puedan ser comparados sus servicios principales.

  - Se revisarán, para su mejora e innovación, los estándares: MADS *Metadata Authority Description Schema*, MODS *Metadata Object Description Schema*.

Especificaciones tecnológicas

- Los lenguajes de programación a utilizar facilitarán la implantación del sistema en cualquier distribución LAMP o WAMP. Esto supone el empleo de los lenguajes de marcado XML, RSS1.0, RSS2.0, Atom, MARC-XML, XSLT, XPath, XPointer, XLink. Por otro lado el empleo de lenguajes de programación PHP y Javascript y el lenguaje de interrogación y consulta SQL. Por otro lado la base de datos que se empleará a modo de plataforma de datos, será MySQL.

Especificaciones de interfaz

- El interfaz general del sistema pone al alcance de cualquier usuario los procesos de sindicación descritos de forma tal que pueda seguirse una línea de desarrollo de la cadena de sindicación desde la preparación de las colecciones bibliográficas, hasta su sindicación, interrogación, consulta y redifusión de los resultados empleando dicha técnica.

- En el caso de la plataforma de sindicación, se dispone de un interfaz a nivel de administrador que permite un control completo de todo el proceso. No requiere de configuración o programación previa para el funcionamiento de los programas y las funciones del sistema, estando todos los procesos automatizados.

- Por otro lado en el caso del programa núcleo, la interfaz se ha diseñado de cara al usuario, de manera que puedan ser visibles los efectos de la sindicación de contenidos de forma constante en el flujo de trabajo de la herramienta, quedando patente este concepto en la incorporación de un canal de sindicación de servicios en el que queda embebido un canal de respuestas, ambos sindicados y actualizables instantáneamente ante cualquier petición de servicio del usuario. En este sentido el interfaz ha cuidado la disposición de toda la información catalográfica de referencia necesaria en cada caso para comprender los formatos de edición e inserción de registros bibliográficos, así como de resultados.

7.3. Arquitectura y funcionamiento de la plataforma SYNC

La plataforma de sindicación SYNC es la primera parte del sistema que se ha desarrollado. Esto se debe a la necesidad de someter a diversas pruebas de funcionamiento la sindicación de contenidos a través de sus formatos, para conocer sus capacidades y viabilidad para sostener colecciones bibliográficas.

Por tanto SYNC se diseña para resolver las especificaciones de diseño que exigen capacidades de conversión de datos, transferencia, importación y exportación a través de los métodos de sindicación de contenidos y sus formatos, de forma que una colección bibliográfica mantenga la mayor cantidad de información correctamente etiquetada.

Teniendo en cuenta estas consideraciones la plataforma SYNC actuará como un sistema capaz de convertir y transferir colecciones bibliográficas que se encuentren en formato CSV directamente a su base de datos. Una vez la información haya sido procesada por el sistema, permitirá la exportación de los datos de la colección bibliográfica en forma de canal de sindicación en formatos RSS1.0, RSS2.0, Atom, XML-Marc Abreviado y Extendido. Una vez los catálogos bibliográficos han sido exportados, SYNC permitirá su importación, esta vez teniendo en cuenta las características y singularidades de cada formato, de forma que pueda obtenerse una experiencia de los tiempos de importación, exportación, lectura y análisis de las colecciones y su comportamiento con el resto de formatos de sindicación.

Se trata en todo caso de que la plataforma SYNC de respuesta a las siguientes preguntas:

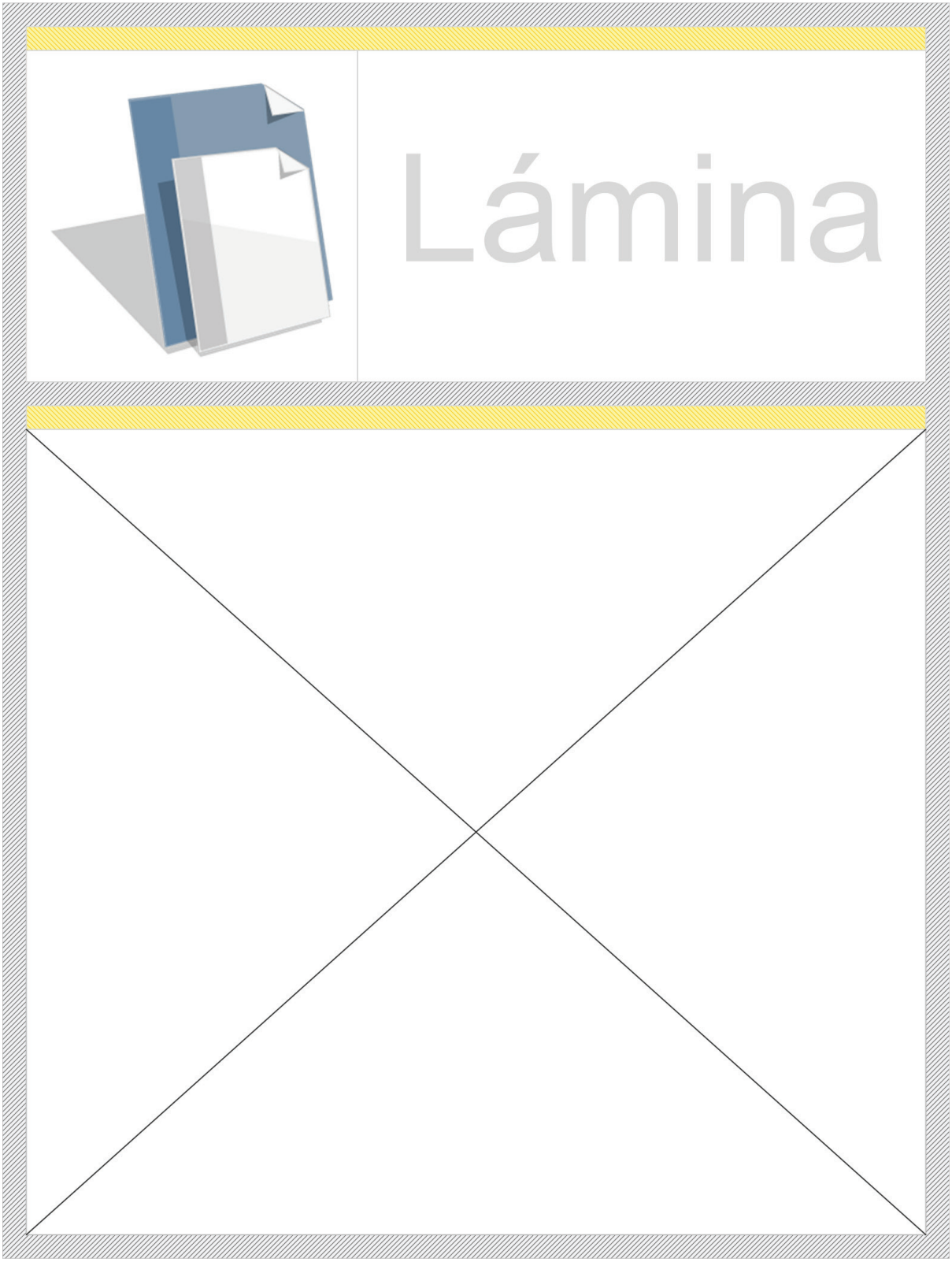
- ¿Es posible sindicar colecciones bibliográficas?
- ¿Es posible formatear una colección bibliográfica en los formatos de sindicación tradicionales?
- ¿Es posible utilizar formatos de sindicación no reconocidos como MARC-XML y llevar a cabo los mismos procesos de exportación, importación y transferencia que con los formatos de sindicación tradicionales RSS1.0, RSS2.0 y Atom?
- ¿Es capaz la sindicación de contenidos de crear canales bibliográficos de grandes dimensiones? ¿Cuál es la capacidad máxima de la sindicación para representar colecciones bibliográficas? ¿Cuál es el volumen óptimo de un catálogo bibliográfico para ser sindicado? ¿Qué soluciones hay para sindicar grandes colecciones bibliográficas?

- ¿Qué formato de sindicación es el más adecuado para representar información bibliográfica?
- ¿Qué formato de sindicación es el más eficiente y rápido en los procesos de exportación, importación y transferencia?

Todas las pruebas de exportación e importación representan un símil real del funcionamiento de la sindicación en un entorno de red local, que permite obtener conclusiones sobre cuál es el formato más adecuado para trabajar con colecciones bibliográficas, así como la reflexión de las dificultades que unos casos u otros pueden representar. No obstante la plataforma SYNC ha sido diseñada también para ser instalada en servidores en red, de hecho las pruebas del sistema de gestión de las colecciones bibliográficas SYNCORE dado que ambos trabajan en conjunción. Por este motivo las estructuras de datos, campos y tablas son compartidas y basadas en el estándar MARC como se explicará más adelante.

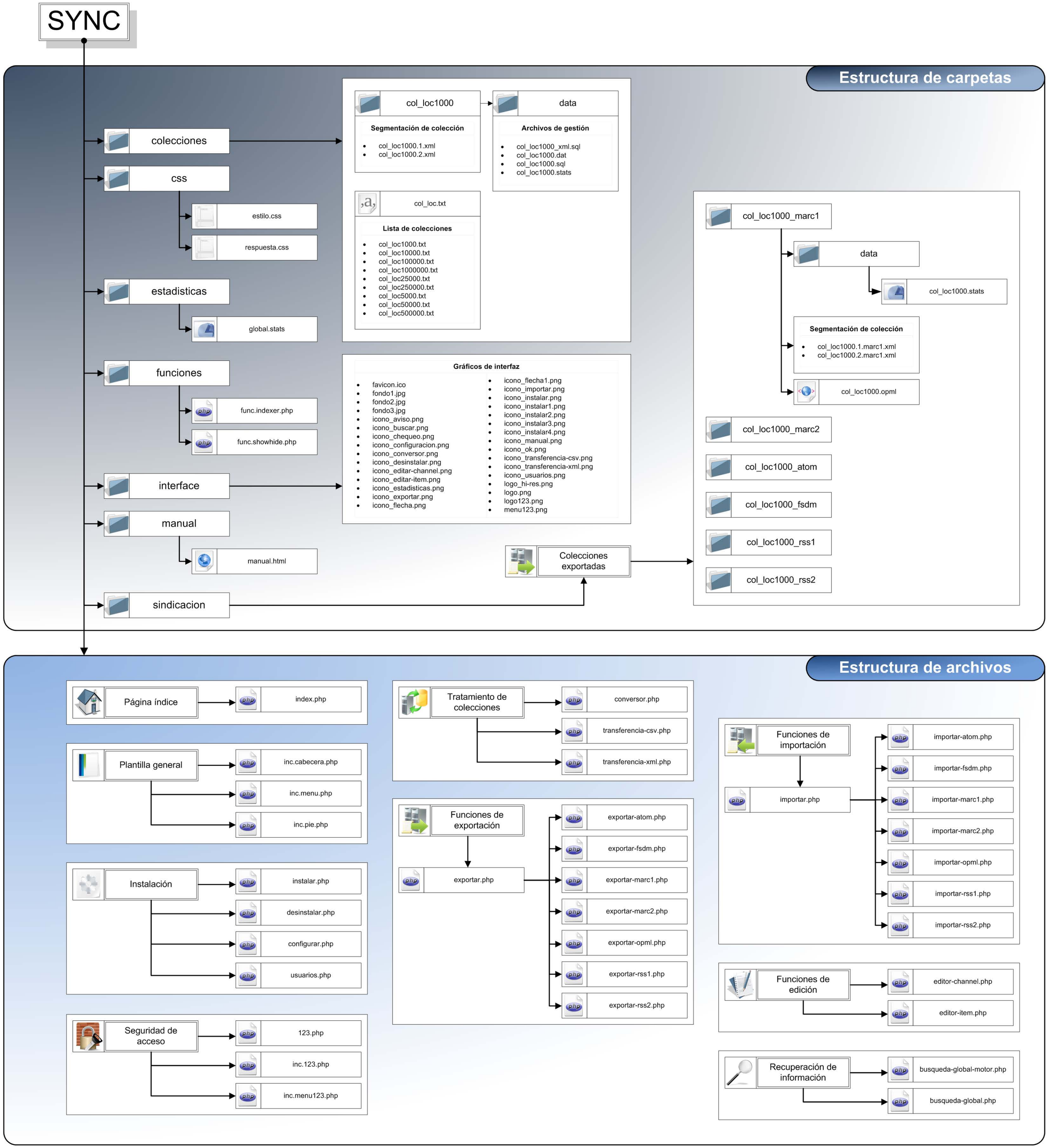
Título: Impresión de pantalla de la plataforma SYNC
Referencia: figura22

Título: Arquitectura de la plataforma SYNC
Referencia: figura23



Arquitectura de la plataforma SYNC

Referencia: figura23



7.4. Construcción de SYNC

La plataforma SYNC se ha construido sobre un paradigma de programación Apache, PHP y MySQL, mediante el empleo del editor Eclipse-PDT. Para organizar su construcción se concretó la elaboración nuclear de un sistema de instalación de SYNC, un programa de conversión y análisis CSV, un programa de transferencia XML Raw y CSV, un programa de exportación de formatos de sindicación, un programa de importación de canales de sindicación, concretamente los relativos a los catálogos bibliográficos exportados.

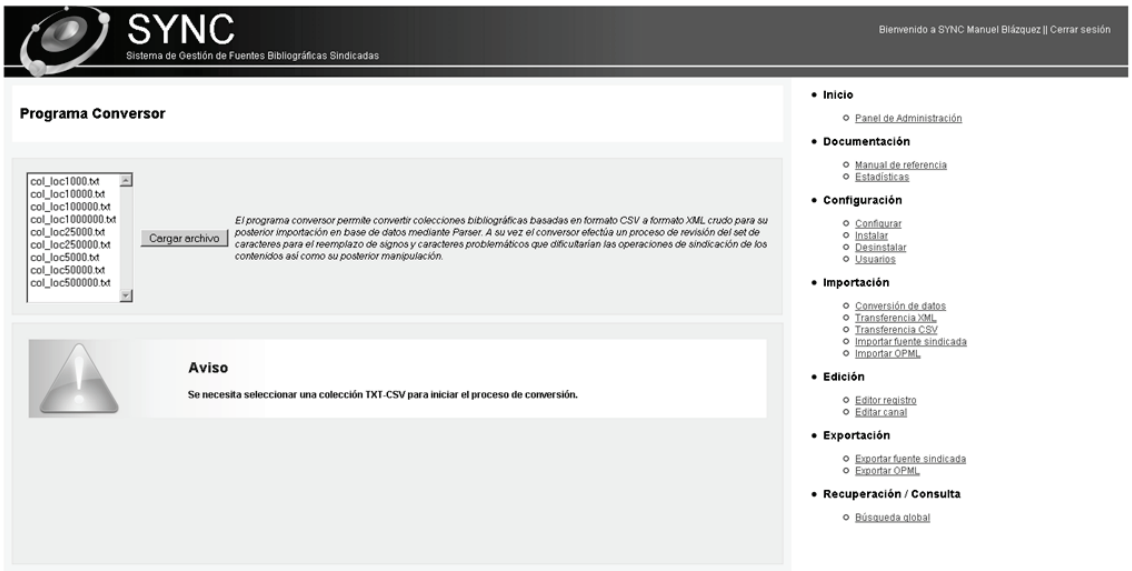
Título: Resumen de programas esenciales de la plataforma SYNC
Referencia: tabla86

Programa	Nombre de archivo/s	Función
Conversión	conversor.php	Efectúa un proceso de conversión y transformación de los catálogos bibliográficos en formato CSV, para su transferencia posterior a la plataforma SYNC.
Transferencia	transferencia-xml.php transferencia-csv.php	El programa transferencia está compuesto por dos métodos de transferencia directa de la información a la plataforma SYNC. Por un lado a partir de XML Raw transformado por el programa conversor.php. Por otro mediante transferencia directa en CSV original.
Exportación	exportar.php exportar-atom.php exportar-rss1.php exportar-rss2.php exportar-marc1.php exportar-marc2.php	El programa de exportación está compuesto por generadores especializados en cada formato de sindicación. Su función es crear canales de sindicación correspondientes a las colecciones transferidas a la plataforma SYNC. También lleva a cabo la prueba de medición de tiempos de exportación distinguiendo cada colección y su formato.
Importación	importar.php importar-atom.php importar-rss1.php importar-rss2.php importar-marc1.php importar-marc2.php	El programa de importación está compuesto por programas parser especializados en los formatos de sindicación Atom, RSS1.0, RSS2.0, MAR-XML abreviado y extendido. Al igual que el programa de exportación, se encarga de la medición de tiempos de la importación a la plataforma SYNC de las colecciones sindicadas.

7.4.1. Programa de conversión

El programa de conversión es un compendio de rutinas que permiten analizar una colección bibliográfica en formato CSV y prepararla para su transferencia a la base de datos de soporte de la plataforma SYNC. Conversión actúa en correlación con el programa de transferencia, por lo que los datos y rutinas generados son utilizados durante la migración de los datos del catálogo bibliográfico a la base de datos de la plataforma SYNC. Por este motivo el programa de conversión y de transferencia son enmarcados dentro del marco de tratamiento de colecciones.

Título: Impresión de pantalla del programa de conversión
Referencia: figura24



Selección de colecciones en formato CSV

Se trata de una rutina que emplea expresiones regulares para filtrar los archivos que forman parte de una carpeta denominada *colecciones* sita en el directorio raíz del sistema SYNC y en la que se alojan todas las colecciones bibliográficas de prueba. Concretamente compara aquellos archivos cuya extensión es *.txt* cuyo contenido ya ha sido preparado con el patrón de valores separados por comas. De esta forma se visualizan a modo de lista desplegable.

Título: Selección de colecciones en formato CSV
Referencia: tabla87

```
// La variable directorio contiene la función de apertura del directorio colecciones en el que se
// encuentran las colecciones bibliográficas de prueba en formato CSV y extensión .txt
$directorio = opendir("colecciones");

// El código incluye un bucle para extraer todos los nombres de los archivos cuya extensión sea .txt,
// mostrándola a modo de opción en una caja de selección dentro de un formulario.
while ($archivo = readdir($directorio)) {

        // Condición con expresión regular para seleccionar todos los archivos acabados en txt.
        if (preg_match("/.txt$/", "$archivo")) {
        // Convierte en minúsculas el nombre del archivo para representarlo correctamente.
        $nombreArch = strtolower($archivo);
        // Impresión en pantalla del nombre del archivo.
        echo "<option value='$nombreArch'>$nombreArch</option>";
        }
}
// Cierre del directorio.
closedir($directorio);
```

Cálculo del número de campos de la colección

Para permitir la transferencia del catálogo bibliográfico a la base de datos MySQL es necesario calcular el número de campos de que consta la colección en formato CSV. Por ese motivo se efectúa un cálculo del número máximo de campos de que dispone a partir de la contabilización de valores separados por coma línea a línea. Se debe recordar que cada línea representa un registro bibliográfico.

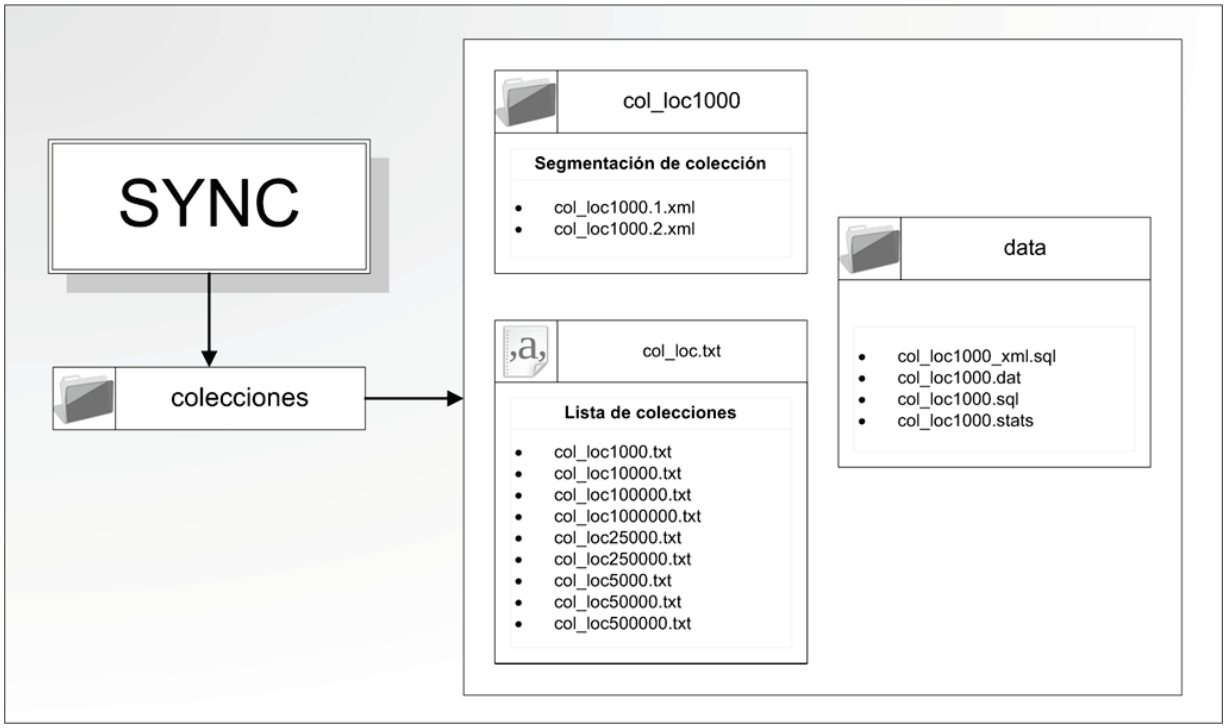
Título: Cálculo del número de campos de la colección
Referencia: tabla88

```
$numax = 1; // Contador de referencia

// Apertura de la colección bibliográfica de prueba en modo lectura
$file1 = fopen("colecciones/$_POST[txtselect]","r");

// Bucle de lectura de campos separados por comas, en líneas de 1024 caracteres
while ($data=fgetcsv($file1, 1024, ","))  {
      // La variable $num almacena el número de campos de la línea
      $num = count($data);
      // Condición para sustituir el valor de la variable si el número de campos aumenta
      if ($num > $numax) { $numax = $num; }
}
// Cierre del archivo
fclose($file1);
```

Conversor CSV a XML

Constituye la función principal del programa. Su función es generar una estructura de carpetas que organice la información y datos del proceso de conversión. Todas son alojadas en el directorio *colecciones* de SYNC, donde se identifican fácilmente dado que toman el nombre del archivo CSV de la colección elegida. En tales carpetas se almacenará la colección bibliográfica en formato *XML Raw*[22], la sentencia SQL necesaria para crear una tabla en la base de datos MySQL para la colección bibliográfica y las estadísticas correspondientes al proceso.

Título: Ejemplo de estructuración de los datos en carpetas
Referencia: figura25



[22] **XML Raw:** Expresión inglesa también expresada como Raw XML se emplea para definir que un archivo XML carece de formato reconocido y validación porque es un mero contenedor de información en bruto. Una expresión similar en español sería XML Crudo.

La función conversora, una vez creadas las carpetas necesarias, comienza a escribir los archivos XML a los que se traspasará la información de CSV, para ello introduce las cabeceras y estructuras básicas necesarias para que se configure como un XML bien formado pero no validado. Antes de escribir ningún dato de la colección bibliográfica se produce un proceso de lectura línea a línea para detectar caracteres del texto que sean erróneos y que puedan dar lugar a fallos durante el proceso. Para ello se somete toda la información a un proceso de depuración y filtrado de caracteres que evita este problema. Una vez depurada cada línea se escribe cada valor o contenido separado por comas, entre etiquetas, constituyendo, campo a campo cada registro y por ende el catálogo completo. Si bien este proceso daría como resultado un único archivo, la experiencia ha demostrado la necesidad de fragmentar las colecciones de gran tamaño para agilizar el proceso. Para ello se ha determinado un límite en el volumen de registros por archivo XML que no excederá de 1000. De esta forma la función genera tantos archivos XML numerados por orden correlativo, como sean necesarios hasta cubrir el volumen total de la colección bibliográfica.

Título: Conversor CSV a XML
Referencia: tabla89

```
// Imprime en pantalla el nombre del archivo que se desea convertir
echo "$archivoName<br/>";

// Crea un directorio con el nombre de la colección de prueba y dentro de este una carpeta de
// datos, en la que se almacenarán archivos y códigos necesarios para efectuar la conversión
mkdir("colecciones/$archivoName", 0777);
mkdir("colecciones/$archivoName/data", 0777);

// Contador de referencia de número de archivo
$contadorA=1;

// Contador de referencia para la contabilización del número de registros
$contadorB=0;

// La variable $maxitems contiene el número máximo de registros que se pueden guardar en un
// archivo XML. Este valor puede ser modificado según las necesidades de cada experimento
$maxitems=1000;

// Apertura de la colección de prueba en formato CSV en modo de lectura
$file1 = fopen("colecciones/$archivoName.txt","r");

// Bucle para la lectura del archivo hasta finalizar la última cadena de caracteres
while(!feof($file1)) {

// La variable $archivoActual conforma el nombre de los archivos que serán convertidos al formato
// XML Raw. En dicha denominación se incluye la numeración de cada uno de ellos.
$archivoActual = $archivoName."$contadorA".".xml";
```

```
// La variable $file2 crea un archivo con la denominación anterior en modo escritura
$file2 = fopen("colecciones/$archivoName/$archivoActual","w");

// Comienza la escritura del archivo XML con la declaración de XML y las primeras etiquetas
fwrite($file2, "<?xml version='1.0' encoding='UTF-8'?>\n");
fwrite($file2, "<catalogo>\n");


// Se imprime en pantalla el número de archivo y el número de registros
echo "PARTE: $contadorA // Registro de Inicio: $contadorB<br/>";
echo "<div class='detalle'>";

$row = 0; // Contador de líneas o registros

// Relación de caracteres y signos que se reemplazarán según se proceda a la lectura de la
// colección bibliográfica
$signo1a = "&"; $signo1b = "Y";
$signo1c = """; $signo1d = " ";
$signo1e = "/"; $signo1f = " ";
$signo1g = "<"; $signo1h = " ";
$signo1i = ">"; $signo1j = " ";
$signo2a = â; $signo2b = a;
$signo2c = ê; $signo2d = e;
$signo2e = î; $signo2f = i;
$signo2g = ô; $signo2h = o;
$signo2i = û; $signo2j = u;
$signo3a = Â; $signo3b = A;
$signo3c = Ê; $signo3d = E;
$signo3e = Î; $signo3f = I;
$signo3g = Ô; $signo3h = O;
$signo3i = Û; $signo3j = U;

// Bucle para la lectura de las líneas de la colección en formato CSV
while($row != $maxitems && $data = fgetcsv ($file1, 1024, ",")) {

$row++;  // Incremento para el contador de líneas
$contadorB++; //  Incremento para el contador del número de registros

// Impresión en pantalla del recuento de registros
echo " || $num - $contadorB";

// Escritura del elemento <item>
fwrite ($file2, "<item>\n");

// Bucle para reemplazar caracteres y signos de todos los registros del catálogo
for ($c=0; $c< $numax; $c++) {

        if ($data[$c]) {
        $campomod1 = ereg_replace ($signo1a, $signo1b, $data[$c]);
        $campomod2 = ereg_replace ($signo1c, $signo1d, $campomod1);
        $campomod3 = ereg_replace ($signo1e, $signo1f, $campomod2);
        $campomod4 = ereg_replace ($signo1g, $signo1h, $campomod3);
        $campomod5 = ereg_replace ($signo1i, $signo1j, $campomod4);
        $campomod6 = ereg_replace ($signo2a, $signo2b, $campomod5);
        $campomod7 = ereg_replace ($signo2c, $signo2d, $campomod6);
        $campomod8 = ereg_replace ($signo2e, $signo2f, $campomod7);
        $campomod9 = ereg_replace ($signo2g, $signo2h, $campomod8);
        $campomod10 = ereg_replace ($signo2i, $signo2j, $campomod9);
        $campomod11 = ereg_replace ($signo3a, $signo3b, $campomod10);
```

```
            $campomod12 = ereg_replace ($signo3c, $signo3d, $campomod11);
            $campomod13 = ereg_replace ($signo3e, $signo3f, $campomod12);
            $campomod14 = ereg_replace ($signo3g, $signo3h, $campomod13);
            $campomod15 = ereg_replace ($signo3i, $signo3j, $campomod14);

            // Cambio del set de caracteres a utf-8 para la normalización de los contenidos
            $campomod16 = iconv("utf-8", "utf-8", $campomod15);


            // Impresión en pantalla de una cruz que representa la línea del registro que ha sido tratado
            // y será escrito en el archivo XML y por ende correctamente codificado
            echo " + ";

            // Escritura del campo bibliográfico que ha sido normalizado y depurado anteriormente,
            // hasta configurar el registro
            fwrite($file2, "<a$c><![CDATA[".$campomod16."]]></a$c>\n");
            } else { fwrite ($file2, "<a$c></a$c>\n"); }
}

// Finalización de la escritura del elemento <item>
fwrite($file2, "</item>\n");
}

// Finalización de la escritura del archivo XML Raw
fwrite($file2, "</catalogo>\n");
echo "</div><br/>";

// Cierre del archivo XML
fclose($file2);
$contadorA++;  // Incremento del contador de archivos
}

// Cierre del archivo CSV
fclose($file1);

// Impresión en pantalla de las estadísticas de la conversión como el número total de registros de la
// colección y el número máximo de campos bibliográficos detectados
echo "Total Registros: $contadorB<br/>";
echo "Nº Máximo de Campos: $numax <br/>";
```

Generador de rutina SQL create table

La función *SQL create table* ha sido diseñada para crear una rutina SQL que permita crear una tabla adaptada a las características de la colección bibliográfica que se desea convertir. Para ello se sirve del número de campos contabilizado anteriormente para crear tantos como fueren necesarios y cuya denominación consiste en el prefijo *(a)* seguido del número de campo correspondiente.

También se determinan que los campos bibliográficos en todo caso serán de tipo texto con set de caracteres utf-8 para facilitar su recuperación, así como por *método fulltext*[23] de MySQL. Finalmente todas estas características son escritas en un archivo de extensión *.sql* sito en la carpeta de datos *data* de la colección bibliográfica.

Título: Generador de rutina SQL create table
Referencia: tabla90

```
// La variable $lineant contiene la primera instrucción para la creación del campo genérico “a0”
$lineant = "a0 TEXT";

// Bucle para generar tantos campos genéricos correlativos como campos bibliográficos han sido
// detectados en cada registro y línea del archivo CSV
for ($c=1; $c< $numax; $c++) {
        $linea = "$lineant,a$c TEXT";
        $lineant = "$linea";
}

// La variable $defcampos construye la instrucción parcial de SQL, completada con la opción fulltext
// que permitirá la posterior indexación y recuperación de los contenidos en los campos
// especificados
$defcampos = '('.$linea.', FULLTEXT(a7,a9,a16,a18,a19,a20,a32,a41,a42,a45))';

// La variable tablename1 construye el nombre de la tabla en la base de datos
$tablename1 = $archivoName."_xml";
echo "Nombre de la tabla: $tablename1<br/>";

// La variable $file6 contiene la función que genera el archivo de datos en modo escritura para
// agregar la instrucción SQL que permitirá crear la tabla en la base de datos SYNC
$file6 = fopen("colecciones/$archivoName/data/$archivoName.sql","w");

// Escritura del archivo de datos y de la instrucción completa
fwrite($file6, "<?php \n");
fwrite($file6, "\$sqr = 'CREATE TABLE $tablename1 $defcampos CHARACTER SET utf8';");
fwrite($file6, "\n ?>");

// Cierre del archivo de datos
fclose($file6);
```

[23] **Método fulltext:** Consiste en el empleo de las funciones de consulta a texto completo disponibles en la base de datos MySQL denominadas Fulltext. Implica la automatización del proceso de indexación de todos los campos que se designen con esta característica. En MySQL cualquier campo de tipo texto y varchar son susceptibles de ser tratados con dicha opción. En la plataforma SYNC se aprovechan estas características para facilitar posteriormente la recuperación de los registros bibliográficos en la base de datos.

Generador de rutina SQL insert into campos

Creado el archivo SQL anterior, necesario para generar una tabla en la base de datos MySQL que contenga la colección bibliográfica, se necesita preparar otra rutina SQL que permita la introducción de los registros bibliográficos en la tabla de la base de datos. La instrucción de inserción está diseñada para contener variables con los valores definitivos a insertar. Dicho de otra forma su construcción es variable dependiendo del número de campos con que haya sido definida la colección, lo que define el orden de introducción de los datos.

Título: Generador de rutina SQL insert into campos
Referencia: tabla91

```
// La variable $lineant1 contiene el primer campo genérico “a0” para la instrucción SQL de inserción
// de datos en los campos de la tabla de cada colección
$lineant1 = "a0";

// Bucle para generar todos los campos de inserción de datos que serán utilizados en la instrucción
for ($c=1; $c< $numax; $c++) {
        $linea1 = "$lineant1,a$c";
        $lineant1 = "$linea1";
}

// La variable $campos contiene la línea de campos definitiva
$campos = "($linea1)";

// La variable $file3 crea un archivo de datos en modo de escritura, que contendrá la instrucción
// SQL completa
$file3 = fopen("colecciones/$archivoName/data/$tablename1.sql","w");
fwrite($file3, "<?php\n\$sql = ");

// Se escribe la instrucción de inserción de datos en la tabla y campos definidos anteriormente
fwrite($file3, "\"INSERT DELAYED INTO $tablename1 $campos \nVALUES (");
$numax1= $numax-1;
for ($c=0; $c< $numax1; $c++) {
        fwrite($file3, "'\$a$c',");
}
fwrite($file3, "'\$a$numax1')\"");
fwrite($file3, "\n?>");

// Se cierra el archivo de datos
fclose($file3);
```

Generador de rutina XPath para la inserción de datos

Si bien las rutinas anteriores daban como resultado dos archivos de extensión *.sql* en los que se define la creación de la tabla y la rutina de inserción de datos de la colección en sus campos correspondientes, el generador de rutina XPath viene a completar dicho proceso. Concretamente se crean las variables de datos correspondientes a los campos que se utilizan en la inserción de registros bibliográficos en el programa de transferencia XML. Esta rutina resulta compleja puesto que implica el uso de programación XPath que permita el acceso a un determinado nodo o entidad de la estructura XML Raw creada anteriormente, que representa a un dato concreto de un campo en especial.

Título: Generador de rutina XPath para la inserción de datos
Referencia: tabla92

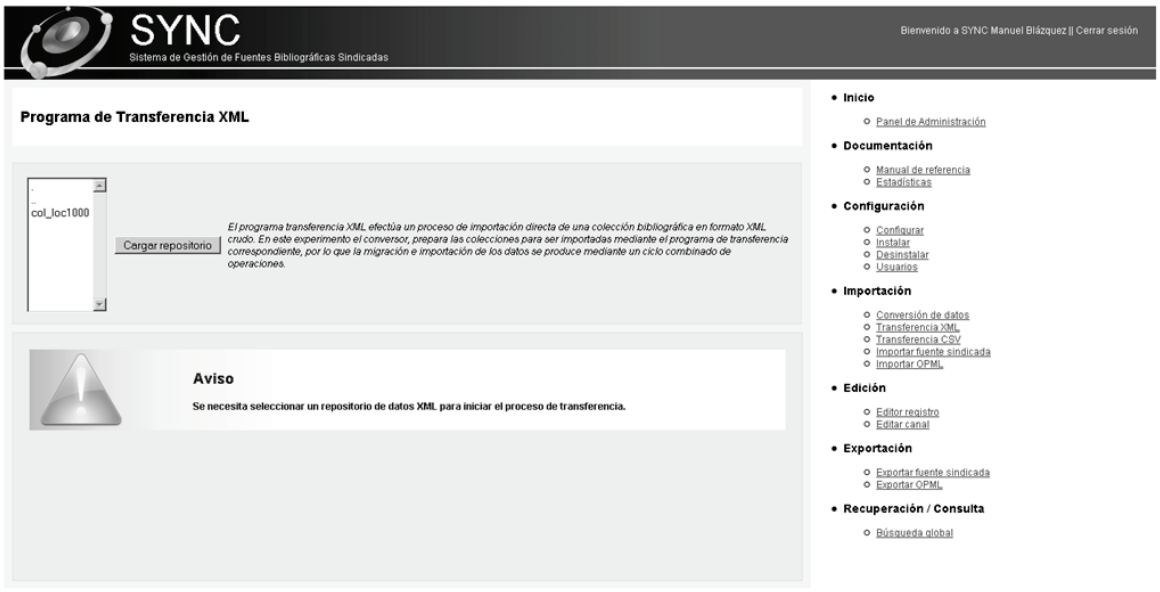
```
// La variable $file4 crea un archivo de datos con las instrucciones XPath de selección de las
// etiquetas de los campos genéricos creados en el archivo XML Raw. Tales instrucciones serán
// utilizadas a posteriori por un parser especialmente diseñado para la importación de los datos
$file4 = fopen("colecciones/$archivoName/data/$archivoName.dat","w");
fwrite($file4, "<?php \n");

// Bucle para escribir todas las instrucciones XPath para cada campo bibliográfico
for ($c=0; $c< $numax; $c++) {

        // Escritura de la variable del campo y su función de selección XPath en XML Raw
        fwrite($file4, "\$a$c = \$xml->item[\$i]->a$c;\n");

}
fwrite($file4, "\n?>");

// Cierre del archivo de datos
fclose($file4);
```

Para comprender mejor el porqué de esta función es necesario recordar que los datos de la colección bibliográfica han sido convertidos a un formato XML y que la mejor manera de acceder a ellos es mediante el empleo de un parser. Dicho parser carece de las instrucciones de acceso a datos de la estructura XML, que le son proporcionados a posteriori por el generador de rutina XPath que se está explicando. Esto es así dado el desconocimiento inicial del número de campos de que constará la colección, obligando a depender este tipo de instrucciones del análisis inicial del programa de conversión.

7.4.2. Programa de transferencia

El programa de transferencia es clave para migrar las colecciones bibliográficas a la base de datos de la plataforma SYNC. Su función es crear las tablas y campos necesarios para alojar las colecciones bibliográficas e insertar todos sus registros. Para ello utiliza los archivos creados anteriormente por el programa de conversión. De esta forma las rutinas SQL, XPath y los propios archivos XML que contienen la información de la colección son empleados con este propósito.

Título: Impresión de pantalla del programa de transferencia
Referencia: figura26



En el diseño original de la plataforma SYNC, el programa de transferencia se ha diseñado con un doble método de migración. Por un lado, empleando el propio archivo CSV de la colección original. Por otro, utilizando los archivos XML Raw creados desde el conversor. El método preferido en todas las pruebas por su fiabilidad, estabilidad y rapidez es el que emplea XML como punto de partida para la migración de datos. Esto se debe a que CSV exige más tiempo de ejecución de lectura que un archivo XML como se podrá comprobar en las pruebas efectuadas. También es cierto que la posibilidad de emplear un parser, supone la seguridad añadida de que todos los datos han sido leídos y carecen de error alguno, cuestión que un sistema de migración CSV no siempre puede asegurar, dado que el parser está diseñado para

la validación de las etiquetas de un archivo XML pero también de sus contenidos. Finalmente la última argumentación a favor del método XML es un mejor control y acceso a los campos bibliográficos mediante XPath, control que no puede ser ejercido sobre CSV, al no distinguir filas ni columnas etiquetadas.

- **Método de transferencia XML**. El método de transferencia XML se sirve de los archivos creados durante el proceso de conversión, especialmente los que contienen las rutinas de creación de tabla, inserción de datos y definición de variables de campos en XPath para el parser. De hecho son cargados a lo largo de todo el proceso de migración que se inicia de la siguiente forma. En principio se generan los nombres de los archivos de datos, ya que pueden variar de una colección bibliográfica a otra. A continuación se carga el archivo de configuración que contiene los datos de conexión a la base de datos en MySQL y se introduce la rutina SQL para la creación de la tabla de la colección. Creada la tabla en la base de datos, se procede a la inserción de los registros bibliográficos, para ello se hace uso de un parser que permite analizar la estructura de los archivos XML Raw de forma secuencial, dado que está inserto en un *bucle while*[24] cuya condición de control es la existencia de los propios archivos XML. Esto se hace para cargar más de un archivo. Recuérdese que el programa de conversión crea un archivo XML por cada 1000 registros, obligando ello a introducir este tipo soluciones automáticas. Volviendo a la explicación del parser, hay que destacar que su estructura se completa con otro bucle embebido en el primero. Se trata de un *bucle for*[25] que actúa mientras existan registros por leer en el archivo XML. Esto obliga a un control de lectura registro por registro de cada archivo XML, permitiendo extraer mediante la rutina XPath generada durante el proceso de conversión, los datos de cada campo uno a uno. Eso facilita sobremanera la inserción de los registros en la base de datos.

[24] **Bucle while:** Estructura de control del lenguaje PHP que permite hacer tantas iteraciones de un código o un proceso mientras se cumpla la condición lógica o de comparación, expresada para tal bucle. En el caso del programa de transferencia, el bucle while se emplea para distinguir cada archivo que constituye el canal de sindicación, con la condición *mientras exista el archivo while(file_exists())*.

[25] **Bucle for:** Estructura de control del lenguaje PHP que permite hacer tantas iteraciones de un código o un proceso mientras se evalúa una condición basada en un contador. En el caso del programa de transferencia, el bucle está limitado por el número de registros del archivo. Mientras puedan contarse ítems con la variable $i, el bucle continuará ejecutándose, *for ($i=0; $xml->item[$i]; $i++)*.

Título: Método de transferencia xml
Referencia: tabla93

```
// La variable $tablename0 toma el valor del nombre del archivo de datos que contiene la
// instrucción SQL “create table” que permite crear la tabla correspondiente a la colección
// bibliográfica
$tablename0 = "$_POST[xmlselect]".".sql";
echo "Archivo SQL Crear Tabla: $tablename0<br/>";

// La variable $tablename1 toma el valor del nombre del archivo de datos que contiene la
// instrucción XPath para seleccionar los contenidos de cada etiqueta
$tablename1 = "$_POST[xmlselect]".".dat";
echo "Archivo DAT Parser: $tablename1<br/>";

// La variable $tablename2 toma el valor del nombre del archivo de datos que contiene la
// instrucción SQL “insert into” que permite la introducción de datos en la tabla
$tablename2 = "$_POST[xmlselect]"."_xml".".sql";
echo "Archivo SQL Insertar Datos: $tablename2<br/>";

// Se imprime en pantalla la colección que se desea transferir a la base de datos SYNC
echo "formulario POST: $_POST[xmlselect]<br/>";

// Incorpora el archivo “config.php” con los datos de conexión a MySQL
include ("config.php");

// Selección de la base de datos SYNC
mysql_select_db("$database", $con);

// Incorpora el archivo de datos “create table” ejecutando su código
include ("colecciones/$_POST[xmlselect]/data/$tablename0");
mysql_query($sqr,$con);

$contadorA=1;  // Contador de archivos
$contadorB=0;  // Contador de registros

// Bucle para comprobar la existencia de los archivos XML desde los que se transferirán los
// registros bibliográficos a la base de datos
while(file_exists("colecciones/$_POST[xmlselect]/$_POST[xmlselect].$contadorA.xml")) {

// La variable $file1 construye la URL de acceso al archivo XML para que pueda ser accesible al
// parser
$file1 = "colecciones/$_POST[xmlselect]/$_POST[xmlselect].$contadorA.xml";

// Parser simplexml_load_file() que actúa sobre el archivo XML para su transferencia a la base de
// datos SYNC. Para ello se incorporarán los archivos de datos con las instrucciones de selección
// XPath y de inserción de los contenidos extraídos
$xml=@simplexml_load_file($file1);
echo "$contadorA"; // Impresión del número de archivo que se está tratando
echo "<div class='detalle'>";

// Inicio del bucle para la selección e inserción de los contenidos de los campos de los registros
// bibliográficos. El final del bucle está dispuesto en el archivo de datos “insert into”
for ($i=0; $xml->item[$i]; $i++) {
$contadorB++; // Incremento del contador de registros

// Incorpora el archivo de datos XPath para seleccionar los contenidos de los campos bibliográficos
// a través del parser que se está ejecutando
include ("colecciones/$_POST[xmlselect]/data/$tablename1");
```

```
// Incorpora el archivo de datos “insert into” para introducir los valores de cada campo obtenidos por
// el parser mediante las instrucciones XPath anteriormente ejecutadas
include ("colecciones/$_POST[xmlselect]/data/$tablename2");
mysql_query($sql);

// Control de fallos en la ejecución basado en el contador de registros $contadorB
if ($sql) { echo " $contadorB + "; } else { echo " Fallo"; }
}
echo "</div><br/>";
$contadorA++;
}
```

- **Método de transferencia CSV**. Como se ha explicado, el método de transferencia CSV resulta más costoso en cuanto a computación. Esto se debe a que todo el proceso de identificación de campos y valores ha de ser procesado en el acto, ya que no consta de almacenamiento intermedio alguno, como sucede con el método XML. A pesar de esto, los pasos que se siguen son los mismos que en XML, a excepción del proceso de inserción de datos. Si bien se mantiene la creación de archivos SQL con las rutinas necesarias, no es así en cuanto a la forma de introducción. Cuando el programa revisa línea a línea los fallos en el set de caracteres del archivo en CSV, se produce la inserción de la línea de datos. Esto es posible debido a que la introducción de los datos es secuencial, no guiada como sucede en XML, lo que podría conllevar problemas de discordancia entre los datos y los campos (Aunque este extremo no ha sucedido en las pruebas realizadas).

Título: Método de transferencia CSV
Referencia: tabla94

```
// PROCESO DE CONTABILIZACIÓN DE CAMPOS

$nummax=1; // Contador del número máximo de campos

// La variable $file1 abre la colección CSV seleccionada en modo lectura
$file1 = fopen("colecciones/$_POST[csvselect]","r");

// Bucle de lectura de las líneas del archivo de la colección de 1024 caracteres, extrayendo los
// campos bibliográficos separados por comas
while ($data1=fgetcsv($file1, 1024, ",")) {

// La variable $num contiene la función de recuento del número total de campos en cada línea
$num = count($data1);

// Condición para establecer el número de campos máximo durante todo el proceso de lectura
if ($num > $nummax) {$nummax=$num;} }
fclose($file1);
```

```
// PROCESO DE DENOMINACIÓN DE ARCHIVOS

// La variable $tablename1 toma el valor del nombre del archivo de datos que contiene la
// instrucción SQL “insert into” para insertar los registros bibliográficos en la tabla correspondiente a
// la colección bibliográfica
$tablename1 = "$archivoName"."_csv".".sql";
echo "Archivo SQL Insertar Datos: $tablename1<br/>";

// La variable $tablename2 construye el nombre de la tabla correspondiente a la colección
// bibliográfica que ha sido importada mediante método de transferencia CSV
$tablename2 = "$archivoName"."_csv";
echo "Nombre de la Tabla: $tablename2<br/>";

// La variable $tablename3 toma el valor del nombre del archivo CSV resultante de la depuración
// directa de la colección bibliográfica original y que será utilizado a modo de buffer de datos para la
// posterior transferencia e importación a la base de datos SYNC
$tablename3 = "$archivoName".".csv";
echo "Archivo Temporal Buffer de datos: $tablename3<br/>";

// CREACIÓN DEL ARCHIVO DE DATOS “INSERT INTO”

// La variable $lineaant1 contiene el primer campo genérico “a0”
$lineaant1= "a0";

// Bucle para la generación del resto de campos con notación genérica
for ($c=1; $c<$nummax; $c++) {
        $linea1 = "$lineaant1,a$c";
        $lineaant1 = "$linea1";
        }
// La variable $campos contiene todos los campos que se utilizaran en la instrucción “insert into”
$campos = "($linea1)";

// Contador para controlar el número de campos a escribir en el archivo de datos
$numax2= $nummax-1;

// La variable $file2 crea un archivo de datos en modo de escritura que contendrá la sentencia de
// inserción SQL completa
$file2 = fopen("colecciones/$archivoName/data/$tablename1","w");
fwrite($file2, "<?php\n\$sql = ");

// Escritura en el archivo de la consulta de inserción, el nombre de la tabla y los campos afectados
fwrite($file2, "\"INSERT DELAYED INTO $tablename2 $campos \n");
fwrite($file2, "VALUES (");
        for ($c=0; $c<$numax2; $c++) {
        fwrite($file2, "'\$a$c',");
        }
fwrite($file2, "'\$a$numax2'");
fwrite($file2, ")\"");
fwrite($file2, "\n?>");

// Cierre del archivo de datos
fclose($file2);
```

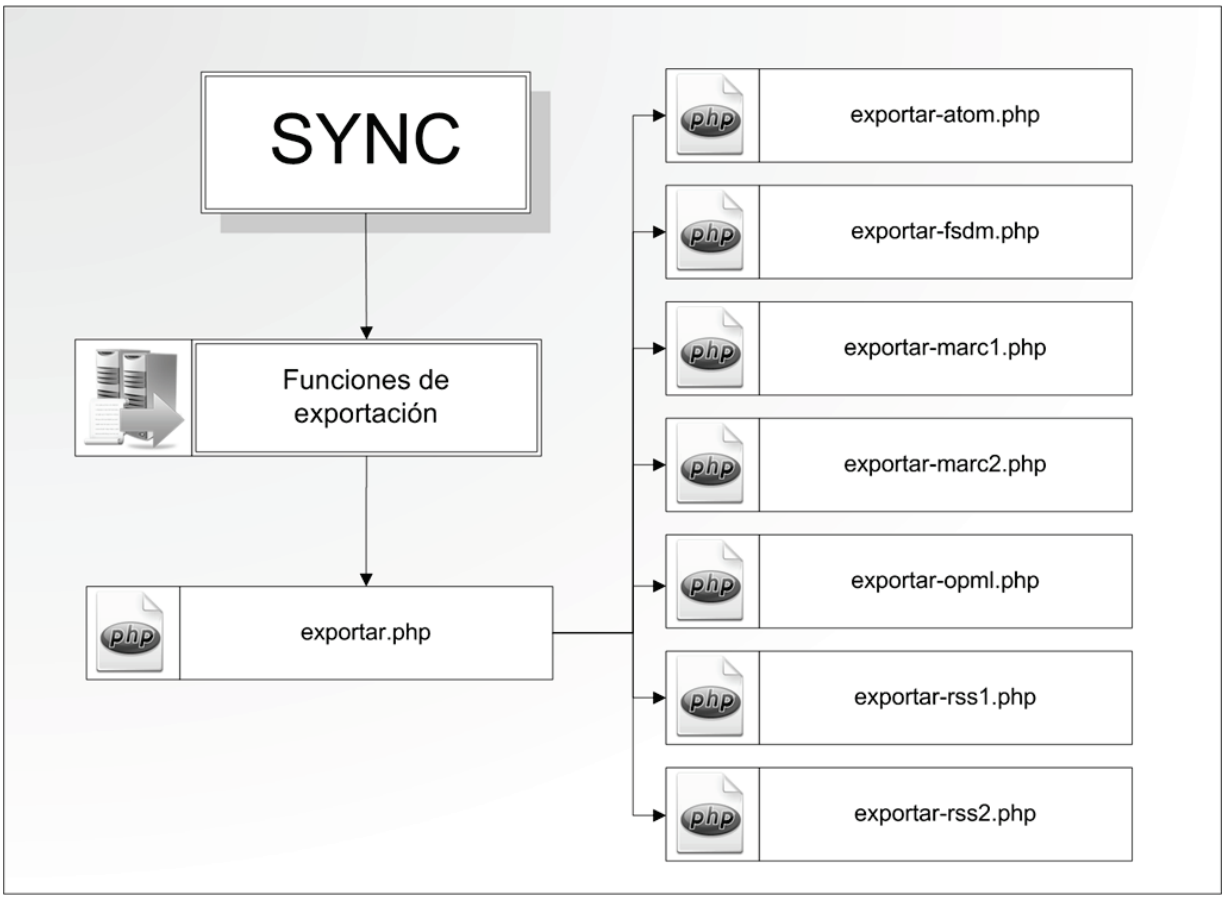
```
// CREACIÓN DE LA TABLA CORRESPONDIENTE A LA COLECCIÓN

// La variable $lineaant2 contiene la instrucción para crear el primer campo de la tabla
$lineaant2= "a0 TEXT";

// Bucle para completar la instrucción de creación de campos de la tabla
for ($c=1; $c<$nummax; $c++) {
$linea2 = "$lineaant2,a$c TEXT";
$lineaant2 = "$linea2";
}
// La variable $defcampos compone la sentencia final para crear la tabla
$defcampos = '('.$linea2.', FULLTEXT(a7,a9,a16,a18,a19,a20,a32,a41,a42,a45))';

// Introducción del archivo config.php con los datos de conexión a MySQL
include("config.php");

// Selección de la base de datos SYNC sobre la que se pretende actuar
mysql_select_db("$database", $con);

// Ejecución de la consulta para crear la tabla correspondiente a la colección bibliográfica
// seleccionada por el usuario
$sqr = "CREATE TABLE $tablename2 $defcampos CHARACTER SET utf8";
mysql_query($sqr,$con);

echo "<div class='detalle'>";

// CREACIÓN DE ARCHIVO CSV SOPORTE Y DEPURACIÓN DE CARACTERES

// Relación de caracteres y signos que se reemplazarán según se proceda a la lectura de la
// colección bibliográfica
$signo1a = "&"; $signo1b = "Y";
$signo1c = """; $signo1d = " ";
$signo1e = "/"; $signo1f = " ";
$signo1g = "<"; $signo1h = " ";
$signo1i = ">"; $signo1j = " ";
$signo2a = â; $signo2b = a;
$signo2c = ê; $signo2d = e;
$signo2e = î; $signo2f = i;
$signo2g = ô; $signo2h = o;
$signo2i = û; $signo2j = u;
$signo3a = Â; $signo3b = A;
$signo3c = Ê; $signo3d = E;
$signo3e = Î; $signo3f = I;
$signo3g = Ô; $signo3h = O;
$signo3i = Û; $signo3j = U;

// La variable $file3 abre el archivo de la colección en formato CSV en modo lectura para proceder
// a su depuración y reemplazo de caracteres problemáticos
$file3=fopen("colecciones/$_POST[csvselect]","r");

// Bucle para la lectura y extracción de los contenidos separados por comas en cada registro
// bibliográfico, que constituirán los valores que se transferirán a la tabla
while ($data2=fgetcsv($file3, 1024, ",")) {

// La variable $file4 crea un archivo CSV de soporte en el que se copiarán todos los datos de la
// colección en formato CSV, depurados y corregidos en su set de caracteres
$file4=fopen("colecciones/$archivoName/data/$tablename3","w");
fwrite($file4, "<?php \n");
```

```
// Bucle para afectar todos los reemplazos de caracteres problemáticos para el proceso de
// transferencia
for ($c=0; $c< $nummax; $c++) {
$lineamod = ereg_replace ($signo1a, $signo1b, $data2[$c]);
$lineamod1= ereg_replace ($signo1c, $signo1d, $lineamod);
$lineamod2= ereg_replace ($signo1e, $signo1f, $lineamod1);
$lineamod3= ereg_replace ($signo1g, $signo1h, $lineamod2);
$lineamod4= ereg_replace ($signo1i, $signo1j, $lineamod3);
$lineamod5= ereg_replace ($signo2a, $signo2b, $lineamod4);
$lineamod6= ereg_replace ($signo2c, $signo2d, $lineamod5);
$lineamod7= ereg_replace ($signo2e, $signo2f, $lineamod6);
$lineamod8= ereg_replace ($signo2g, $signo2h, $lineamod7);
$lineamod9= ereg_replace ($signo2i, $signo2j, $lineamod8);
$lineamod10= ereg_replace ($signo3a, $signo3b, $lineamod9);
$lineamod11= ereg_replace ($signo3c, $signo3d, $lineamod10);
$lineamod12= ereg_replace ($signo3e, $signo3f, $lineamod11);
$lineamod13= ereg_replace ($signo3g, $signo3h, $lineamod12);
$lineamod14= ereg_replace ($signo3i, $signo3j, $lineamod13);

// Conversión y normalización de los datos al set de caracteres utf-8
$lineamod15 = iconv("utf-8", "utf-8", $campomod14);

// Escritura de los datos depurados en el archivo CSV de soporte
fwrite($file4, "\$a$c = '$lineamod15';\n");
}
fwrite($file4, "?>");

// Cierre del archivo CSV de soporte
fclose($file4);

// INSERCIÓN DE DATOS EN LA TABLA

// Incorporación y ejecución de las instrucciones del archivo CSV de soporte
include("colecciones/$archivoName/data/$tablename3");

// Incorporación y ejecución de las instrucciones del archivo de datos SQL para la inserción de los
// contenidos depurados anteriormente
include("colecciones/$archivoName/data/$tablename1");

// Ejecución de la consulta SQL
mysql_query($sql);
echo " + ";
}
echo "</div>";
```

7.4.3. Programa de exportación

El programa de exportación ha sido diseñado para generar canales de sindicación a partir de las colecciones bibliográficas que fueron transferidas a la plataforma SYNC. Para ello el programa de exportación está constituido por funciones especializadas en cada formato de sindicación. Cada una de ellas efectúa consultas SQL sobre el catálogo bibliográfico que haya sido seleccionado, componiendo un canal de sindicación con todos sus registros, de tal forma que guarda la estructura y composición que dictan las especificaciones de cada formato.

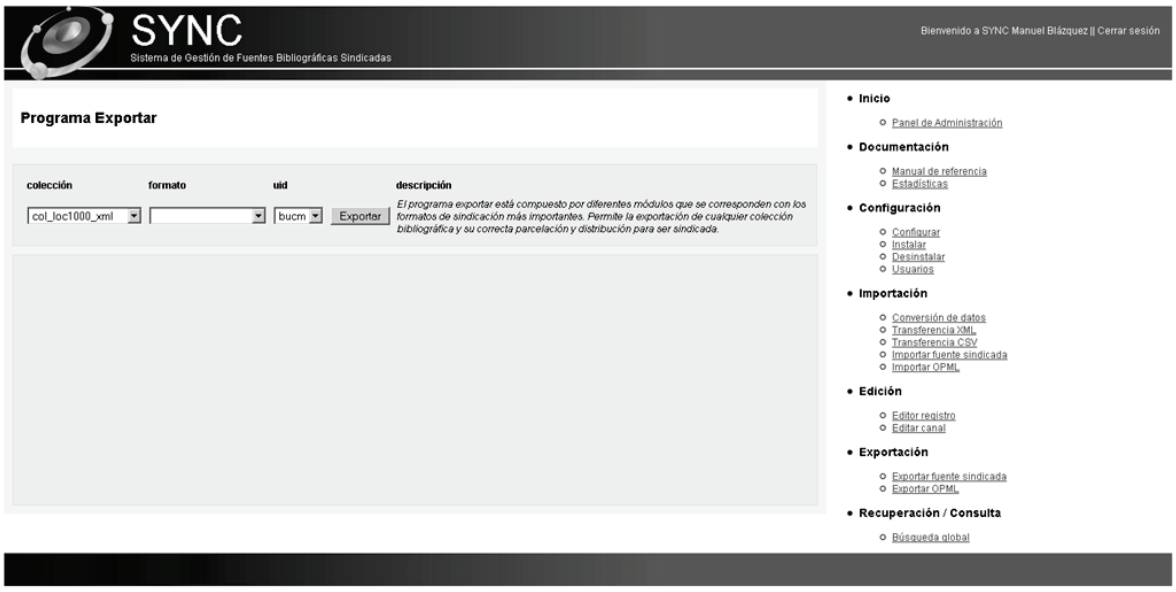
Título: Diagrama del programa de exportación
Referencia: figura27



Archivo exportar.php

Constituye la cabecera del programa de exportación. Su función es proporcionar un formulario de opciones de exportación entre las que se puede seleccionar el catálogo bibliográfico, el formato de sindicación y el identificador del centro bibliotecario. Esta información es vital para redirigir la consulta de la colección bibliográfica a la función adecuada de exportación en el formato de sindicación elegido.

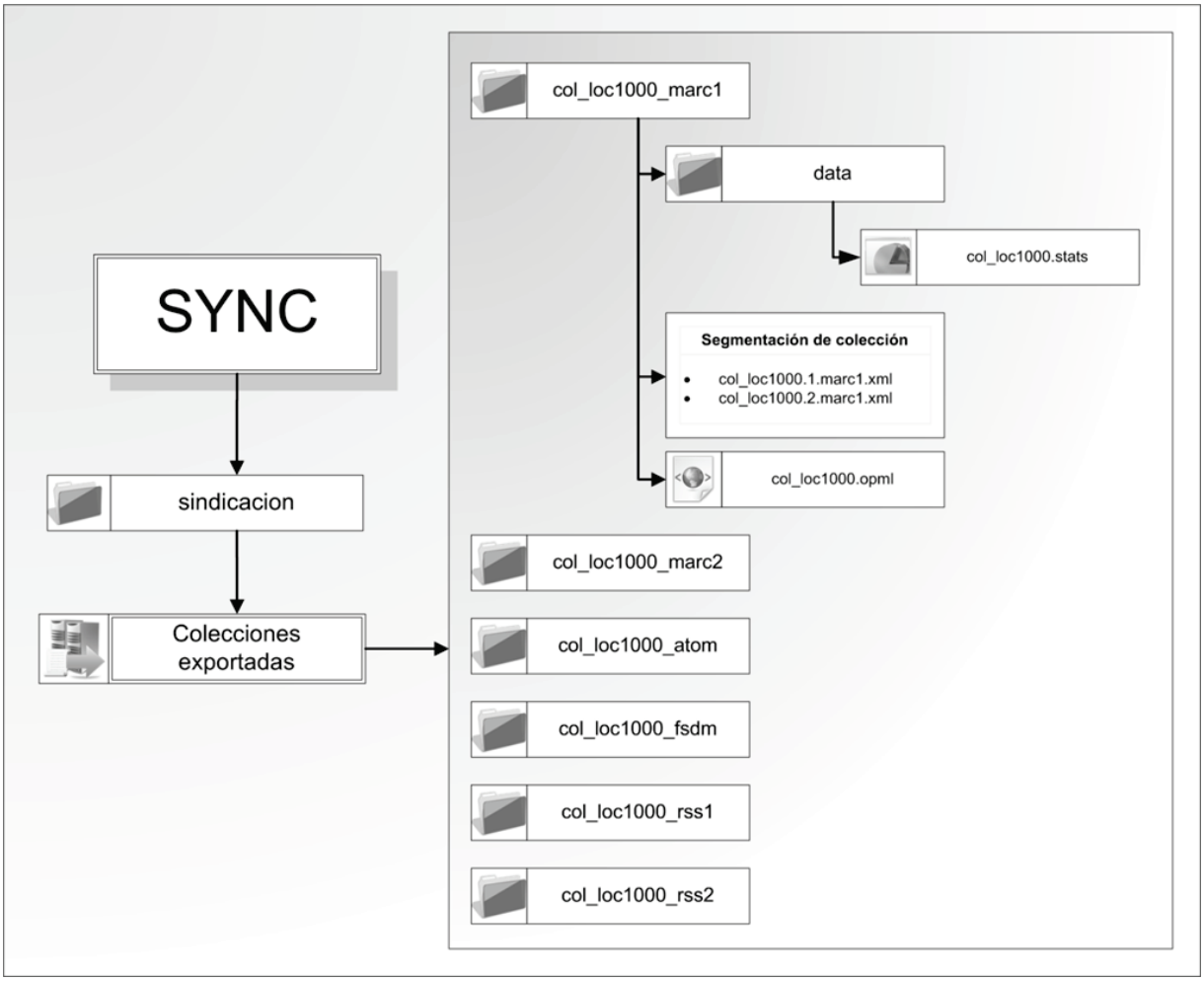
Título: Impresión de pantalla del programa de exportación
Referencia: figura28



Archivo exportar-atom.php

Contiene la función de exportación propia del formato de sindicación Atom. Su funcionamiento comienza a partir de la consulta de la tabla correspondiente a la colección seleccionada. Concretamente se contabiliza el número de registros bibliográficos que servirá para limitar el bucle de creación de registros en el formato Atom. A continuación se crea una estructura de carpetas en la que se guardarán todos los archivos de exportación, el canal de sindicación propiamente dicho y los datos estadísticos del proceso.

Título: Diagrama de la estructura de carpetas en la exportación
Referencia: figura29



La función de exportación tiene en cuenta los casos de colecciones bibliográficas de gran volumen. En tales casos, al igual que en el proceso de conversión, el canal de sindicación se divide en archivos XML de 1000 registros cada uno hasta completar el total de la colección. Esta solución permite distribuir equitativamente el tamaño de los archivos y su peso en cualquier servidor en el que estén alojados. Otro motivo por el que se plantea esta solución es que los archivos de mayor tamaño tardan más tiempo en ser leídos por lo tanto ese aspecto repercute negativamente en el proceso de importación, propio de los parser, dado que requieren de la carga del archivo y su lectura previa para efectuar el análisis, extracción de datos o acceso a los nodos, atributos y entidades correspondientes del formato de sindicación. De esta forma un archivo XML posee un tamaño razonable resultando mucho más práctico y flexible a la hora de ser manipulado para su tratamiento y proceso importación.

No obstante se plantea una dificultad en el desarrollo de esta técnica de partición de archivos XML. Téngase en cuenta que no se trata de un formato XML Raw, en el cual no importaba que no estuviera validado por ningún XSD o DTD. En este caso de los formatos de sindicación la singularidad de la validación resulta tan importante como que el archivo XML esté correctamente formado. Además un canal de sindicación particionado plantea el problema de ser poco accesible, puesto que implicaría la introducción de tantos canales en el sitio web como miles de registros tuviesen la colección. Esta situación haría insostenible cualquier intento de sindicación de colecciones bibliográficas.

Ante este importante problema, se ha desarrollado una solución sencilla y efectiva que posibilita la agrupación de todos los archivos XML que forman parte del mismo canal de sindicación. Se trata del empleo de OPML un formato basado en XML que permite el intercambio de información de esquema estructurado entre las aplicaciones que se ejecutan en diferentes sistemas operativos y entornos. En este caso concreto OPML se utiliza para agrupar todos los archivos XML del canal de sindicación, mediante las etiqueta *<outline type=’’ title=’’ text=’’ xmlUrl=’’ htmlUrl=’’/>*. Esta estructura posibilita la descripción de todos los archivos, definiendo su formato de sindicación, título, resumen de contenidos, URL del archivo e incluso URL del sitio web del centro catalogador. El resultado de aplicar este proceso se puede comprobar en el siguiente ejemplo práctico de un archivo OPML unificando todos los archivos de un canal de sindicación Atom.

Título: Ejemplo práctico de un archivo OPML
Referencia: tabla95

```
<?xml version='1.0' encoding='UTF-8'?>
<opml version='1.0'>
<head>
        <title></title>
        <dateCreated></dateCreated>  <dateModified></dateModified>
        <ownerName></ownerName>   <ownerEmail></ownerEmail>
</head>
<body>

<outline type='atom' title='col_loc2000-atom Parte: 1'
xmlUrl='sindicacion/col_loc2000_atom/col_loc2000.1.atom.xml' />

<outline type='atom' title='col_loc2000-atom Parte: 2'
xmlUrl='sindicacion/col_loc2000_atom/col_loc2000.2.atom.xml' />

</body>
</opml>
```

La técnica de OPML combinada con los demás formatos de sindicación de contenidos desempeña un papel determinante para organizar todas las partes de las que consta una colección bibliográfica, ya que permite gestionarlas con un único archivo, manteniendo la unicidad del canal. Teniendo en cuenta todas estas consideraciones, el desarrollo de todas las funciones de exportación ha contemplado la creación automática de un archivo OPML que enlaza toda la colección bibliográfica.

Continuando con el proceso de exportación Atom, se emplea un bucle de control del número total de registros bibliográficos de cara a parcelar el número de archivos XML necesarios para el canal. A continuación se crea el archivo OPML con la referencia de los archivos XML que se crearán en formato Atom. Seguidamente se produce una consulta completa de la tabla correspondiente a la colección, de la que se obtienen todos los registros con todos sus campos, mediante un array o matriz asociativa que permite la identificación de los mismos por el nombre de sus etiquetas o campos. De esta forma, se identifican qué campos han de ocupar qué etiquetas de la estructura Atom según las especificaciones (IETF ATOMPUB WORKING GROUP, 2005).

Título: Código del archivo exportar-atom.php
Referencia: tabla96

```
// Se imprime en pantalla la variable $archivoName que es el nombre de la colección elegida para
// la exportación. Por otro lado se imprime la comprobación de la selección mediante la variable
// $_POST[colexport] enviada por formulario vía http POST
echo "Valor de archivoName: $archivoName<br/>";
echo "Valor POST: $_POST[colexport]<br/>";

// Incorporación del archivo “config.php” con los datos de conexión a MySQL
include("config.php");

// Selección de la base de datos SYNC sobre la que se opera
mysql_select_db("$database", $con);

// Se ejecuta una consulta SQL para contabilizar el número de registros disponibles en la tabla
$sql="SELECT COUNT(*) FROM $_POST[colexport]";
$consulta=mysql_query($sql); $rcount=mysql_result($consulta,0);
echo "Nº Total de Registros en tabla $_POST[colexport]: $rcount<br/>";

// Se crea una carpeta en el directorio “sindicacion” con el nombre de la colección y el sufijo atom
// que identifica el proceso de exportación a este formato de sindicación
mkdir("sindicacion/".$archivoName."_atom", 0777);

// Se crea la carpeta “data” dentro del directorio anterior de forma tal que puedan ser guardados los
// archivos de datos necesarios, así como las estadísticas parciales del proceso
mkdir("sindicacion/".$archivoName."_atom/data", 0777);
```

```
$contadorA=1;  // Contador de archivos
$contadorB=0;  // Contador de registros
$maxitems=1000;  // Límite de registros por archivo XML en formato Atom exportado

// La variable $file1 crea un archivo OPML en el directorio de la colección exportada que contendrá
// todas las URL a los archivos XML con la colección bibliográfica en formato Atom.
$file1 = fopen("sindicacion/".$archivoName."_atom/$archivoName.opml","w");

// Se escribe la cabecera y el canal del archivo OPML
fwrite($file1,"<?xml version='1.0' encoding='UTF-8'?>
<opml version='1.0'>
<head>
<title></title>
<dateCreated></dateCreated>
<dateModified></dateModified>
<ownerName></ownerName>
<ownerEmail></ownerEmail>
</head>
<body>
");

// Bucle principal que permite ejecutar el código debajo descrito, mientras el contador de registros
// no supere el total definido de la colección, definido por la variable $rcount
while($contadorB != $rcount) {

// La variable $archivoActual construye el nombre de cada archivo que será exportado a partir del
// nombre de la colección y el valor del contador de archivos
$archivoActual = $archivoName.".$contadorA";

// Se crea un archivo que contendrá la codificación de la colección bibliográfica en formato Atom.
// Para ello se construye un nombre de archivo diferenciador basado en el contador correlativo de
// archivos anteriormente mencionado
$file = fopen("sindicacion/".$archivoName."_atom/$archivoActual.atom.xml","w");

// Escritura del elemento “Outline” del archivo OPML que incluye la URL de los archivos creados
// durante el proceso de exportación. Este proceso es necesario para su reagrupación y posterior
// lectura desde un parser
fwrite($file1,"<outline type='atom' title='$archivoName-atom Parte: $contadorA'
xmlUrl='sindicacion/".$archivoName."_atom/$archivoActual.atom.xml'/>");

// Dentro del mismo bucle se comienza a escribir la cabecera del archivo Atom
fwrite($file,"<?xml version='1.0' encoding='utf-8'?>
<feed>
");

// Se ejecuta la consulta de la tabla channel en correspondencia con la colección seleccionada, de
// forma que puedan ser escritos los datos de descripción de cada canal de sindicación que será
// exportado
$sql1 = "SELECT * FROM channel WHERE colname LIKE '$_POST[colexport]'";
$resultado1 = mysql_query($sql1);
$row1 = mysql_fetch_assoc($resultado1);

// Escritura de los valores obtenidos en la consulta en el archivo XML
fwrite($file,"
<id>$row1[url]</id> // identificador
<title>$row1[title]</title> // título del canal
<subtitle>$row1[subtitle]</subtitle> // Subtítulo o breve descripción del canal
<updated>$row1[lastDate]</updated> // Fecha de actualización del canal
```

```
<author>
<name>$row1[author]</name>  // Nombre completo del autor del canal
<email>$row1[authorMail]</email>  // Correo electrónico del autor
</author>

<contributor>
<name>$row1[liaison]</name>  // Nombre completo del colaborador del canal
<email>$row1[liaisonMail]</email>  // Correo electrónico del colaborador
</contributor>

<link rel='self' href='$row1[link]'/>  // Enlace del canal de sindicación o sitio web del mismo
<category term='$row1[category]'/>  // Categoría temática del canal
<generator uri='$row1[generatorLink]'>$row1[generator]</generator>  // Programa generador
<icon>$row1[icon]</icon>  // URL del favicon del canal
<logo>$row1[logo]</logo>  // URL del logotipo del canal
<rights>$row1[rights]</rights>  // Derechos de explotación de los contenidos del canal
");

// Impresión en pantalla del número de archivo que se está tratando
echo "Nº de Archivo: $contadorA ";

// Impresión en pantalla del número de registro bibliográfico por el que se comienza a trabajar
echo "Nº de Registro: $contadorB ";
echo "<div class='detalle'>";

// Se ejecuta la consulta SQL para extraer todos los registros de la colección atendiendo al
// contador de registros límite $maxitems
$sql2 = "SELECT * FROM $_POST[colexport] LIMIT $contadorB,$maxitems";
$resultado2 = mysql_query($sql2);

// Segundo bucle que permite la escritura de un registro bibliográfico codificado en formato Atom,
// dentro del archivo de exportación anteriormente creado
while ($row2 = mysql_fetch_assoc($resultado2)) {
fwrite($file,"
<entry>  // Apertura del registro
<id>$row2[a0]</id>  // Identificador del registro bibliográfico
<title>$row2[a9]</title>  // Título propiamente dicho
<author><name>$row2[a7]</name></author>  // Mención de responsabilidad
<updated></updated>  // Fecha de actualización del registro
<content>$row2[a7] $row2[a9] $row2[a15] $row2[a18] $row2[a19] $row2[a20] $row2[a26]
$row2[a32] $row2[a33] $row2[a41] $row2[a42] $row2[a44] $row2[a44]</content>  // Ficha completa
<link rel='alternate' href=''/>  // Contenido adjunto o alternativo
<summary>$row2[a42]</summary>  // Notas de contenidos
<category term='$row2[a45]'/>  // Categoría temática del documento
<contributor><name>$row2[a16]</name></contributor>  // Otras menciones de responsabilidad
<published></published>  // Fecha de publicación
<source></source>  // Fuente de información citada
<rights></rights>  // Derechos del documento
</entry>  // Cierre del registro
");
$contadorB++;  // Incremento del contador de registros
echo " + "; // Impresión en pantalla de un signo que representa el registro escrito
}
echo "</div><br/>";

// Escritura del final del archivo de exportación cuando llega al límite de 1000 registros planteado
// anteriormente en la variable $maxitems
fwrite($file,"</feed>");
```

```
// Cierre del archivo de exportación
fclose($file);
$contadorA++; // Incremento del contador de archivos
}

// Escritura del final del archivo OPML
fwrite($file1,"</body></opml>");

// Cierre del archivo OPML
fclose($file1);
```

Archivo exportar-rss1.php

El archivo exportar-rss1.php contiene la función generadora del canal de sindicación RSS1.0. Su funcionamiento es similar al relatado anteriormente en Atom a diferencia de las especificaciones empleadas (BEGED DOV, G. et al., 2008).

Título: Código del archivo exportar-rss1.php
Referencia: tabla97

```
// Se imprime en pantalla la variable $archivoName que es el nombre de la colección elegida para
// la exportación. Por otro lado se imprime la comprobación de la selección mediante la variable
// $_POST[colexport] enviada por formulario vía http POST
echo "Valor de archivoName: $archivoName<br/>";
echo "Valor POST: $_POST[colexport]<br/>";

// Incorporación del archivo “config.php” con los datos de conexión a MySQL
include("config.php");

// Selección de la base de datos SYNC sobre la que se opera
mysql_select_db("$database", $con);

// Se ejecuta una consulta SQL para contabilizar el número de registros disponibles en la tabla
$sql="SELECT COUNT(*) FROM $_POST[colexport]"; $consulta=mysql_query($sql);
$rcount=mysql_result($consulta,0);
echo "Nº Total de Registros en tabla $_POST[colexport]: $rcount<br/>";

// Se crea una carpeta en el directorio “sindicacion” con el nombre de la colección y el sufijo RSS1.0
// que identifica el proceso de exportación a este formato de sindicación
mkdir("sindicacion/".$archivoName."_rss1", 0777);

// Se crea la carpeta “data” dentro del directorio anterior de forma tal que puedan ser guardados los
// archivos de datos necesarios, así como las estadísticas parciales del proceso
mkdir("sindicacion/".$archivoName."_rss1/data", 0777);

$contadorA=1;  // Contador de archivos
$contadorB=0;  // Contador de registros
$maxitems=1000;  // Límite de registros por archivo XML en formato RSS1.0 RDF exportado

// La variable $file1 crea un archivo OPML en el directorio de la colección exportada que contendrá
// todas las URL a los archivos XML con la colección bibliográfica en formato RSS1.0 RDF
$file1 = fopen("sindicacion/".$archivoName."_rss1/$archivoName.opml","w");
```

```
// Se escribe la cabecera y el canal del archivo OPML
fwrite($file1,"<?xml version='1.0' encoding='UTF-8'?>
<opml version='1.0'>
<head>
<title></title>
<dateCreated></dateCreated>
<dateModified></dateModified>
<ownerName></ownerName>
<ownerEmail></ownerEmail>
</head>
<body>
");

// Bucle principal que permite ejecutar el código debajo descrito, mientras el contador de registros
// no supere el total definido de la colección, definido por la variable $rcount
while($contadorB != $rcount)  {

// La variable $archivoActual construye el nombre de cada archivo que será exportado a partir del
// nombre de la colección y el valor del contador de archivos
$archivoActual = $archivoName.".$contadorA";

// Se crea un archivo que contendrá la codificación de la colección bibliográfica en formato RSS1.0.
// Para ello se construye un nombre de archivo diferenciador basado en el contador correlativo de
// archivos anteriormente mencionado
$file = fopen("sindicacion/".$archivoName."_rss1/$archivoActual.rss1.xml","w");

// Escritura del elemento “Outline” del archivo OPML que incluye la URL de los archivos creados
// durante el proceso de exportación. Este proceso es necesario para su reagrupación y posterior
// lectura desde un parser
fwrite($file1,"<outline type='rss1' title='$archivoName-rss1 Parte: $contadorA'
xmlUrl='sindicacion/".$archivoName."_rss1/$archivoActual.rss1.xml' />");

// Dentro del mismo bucle se comienza a escribir la cabecera del archivo RSS1.0 RDF
fwrite($file,"<?xml version='1.0' encoding='utf-8'?>
<?xml-stylesheet href='.xsl' type='text/xslt' media='screen'?>
<?xml-stylesheet href='.css' type='text/css' media='screen'?>
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
            xmlns:dc='http://purl.org/dc/elements/1.1/'
            xmlns='http://purl.org/rss/1.0/'
            xmlns:sy='http://purl.org/rss/1.0/modules/syndication/'
            xmlns:content='http://purl.org/rss/1.0/modules/content/'
            xmlns:prism='http://prismstandard.org/namespaces/1.2/basic/'
            xmlns:skos='http://www.w3.org/2004/02/skos/core#'>
");

// Se ejecuta la consulta de la tabla channel en correspondencia con la colección seleccionada, de
// forma que puedan ser escritos los datos de descripción de cada canal de sindicación que será
// exportado
$sql1 = "SELECT * FROM channel WHERE colname LIKE '$_POST[colexport]'";
$resultado1 = mysql_query($sql1);
$row1 = mysql_fetch_assoc($resultado1);

// Escritura de los valores obtenidos en la consulta en el archivo XML
fwrite($file,"
<channel rdf:about='enlace directo al canal'>
<title>$row1[title]</title> // Título del canal
<link>$row1[link]</link> // URL del canal a modo de identificador
<description>$row1[description]</description>  // Descripción somera del contenido del canal
<language>es</language>  // Idioma del canal
<sy:updatePeriod>$row1[syupdatePeriod]</sy:updatePeriod>  // Periodo de actualización
```

```
<sy:updateFrequency>$row1[syupdateFrequency]</sy:updateFrequency>  // Frecuencia de act.
<sy:updateBase>$row1[syupdateBase]</sy:updateBase>  // Fecha de actualización
</channel>
");

// Impresión en pantalla del número de archivo que se está tratando
echo "Nº de Archivo: $contadorA ";

// Impresión en pantalla del número de registro bibliográfico por el que se comienza a trabajar
echo "Nº de Registro: $contadorB ";

echo "<div class='detalle'>";

// Se ejecuta la consulta SQL para extraer todos los registros de la colección atendiendo al
// contador de registros límite $maxitems
$sql2 = "SELECT * FROM $_POST[colexport] LIMIT $contadorB,$maxitems";
$resultado2 = mysql_query($sql2);

// Segundo bucle que permite la escritura de un registro bibliográfico codificado en formato RSS1.0,
// dentro del archivo de exportación anteriormente creado
while ($row2 = mysql_fetch_assoc($resultado2)) {
fwrite($file,"
<item>
<dc:title>$row2[a9]</dc:title> // Título propiamente dicho
<dc:creator>$row2[a7]</dc:creator>  // Mención de responsabilidad principal
<dc:contributor>$row2[a16]</dc:contributor>  // Otras menciones de responsabilidad
<dc:publisher>$row2[a18]: $row2[a19]</dc:publisher>  // Área de publicación
<dc:date>$row2[a20]</dc:date>  // Fecha de publicación
<dc:type></dc:type>  // Tipo documental
<dc:format></dc:format>  // Soporte o formato
<dc:identifier>$row2[a41]</dc:identifier>  // Identificador del documento
<dc:subject>$row2[a44] - $row2[a45]</dc:subject>  // Categoría temática
<dc:source></dc:source>  // Fuente de información citada
<dc:language></dc:language>  // Idioma
<dc:relation></dc:relation>  // Contenidos adjuntos relacionados
<dc:coverage></dc:coverage>  // Cobertura temporal y espacial
<dc:rights></dc:rights>  // Derechos del documento
<skos:note>$row2[a42]</skos:note>  // Área de notas
<content:encoded>$row2[a0] $row2[a9] $row2[a7] $row2[a15] $row2[a16] $row2[a18] $row2[a19]
$row2[a20] $row2[a26] $row2[a32] $row2[a33] $row2[a41] $row2[a42] $row2[a44]
$row2[a45]</content:encoded>  // Ficha completa
</item>
");

$contadorB++;  // Incremento del contador de registros
echo " + "; // Impresión en pantalla de un signo que representa el registro escrito
}

echo "</div><br/>";

// Escritura del final del archivo de exportación cuando llega al límite de 1000 registros planteado
// anteriormente en la variable $maxitems
fwrite($file,"</rdf:RDF>");


// Cierre del archivo de exportación
fclose($file);
$contadorA++; // Incremento del contador de archivos
}
```

```
// Escritura del final del archivo OPML
fwrite($file1,"</body></opml>");

// Cierre del archivo OPML
fclose($file1);
```

Archivo exportar-rss2.php

El archivo exportar-rss2.php contiene la función generadora del canal de sindicación RSS2.0. Su funcionamiento es similar al relatado anteriormente en Atom a diferencia de las especificaciones empleadas (WINER, D., 2009).

Título: Código del archivo exportar-rss2.php
Referencia: tabla98

```
// Se imprime en pantalla la variable $archivoName que es el nombre de la colección elegida para
// la exportación. Por otro lado se imprime la comprobación de la selección mediante la variable
// $_POST[colexport] enviada por formulario vía http POST
echo "Valor de archivoName: $archivoName<br/>";
echo "Valor POST: $_POST[colexport]<br/>";

// Incorporación del archivo “config.php” con los datos de conexión a MySQL
include("config.php");

// Selección de la base de datos SYNC sobre la que se opera
mysql_select_db("$database", $con);

// Se ejecuta una consulta SQL para contabilizar el número de registros disponibles en la tabla
$sql="SELECT COUNT(*) FROM $_POST[colexport]";
$consulta=mysql_query($sql); $rcount=mysql_result($consulta,0);
echo "Nº Total de Registros en tabla $_POST[colexport]: $rcount<br/>";

// Se crea una carpeta en el directorio “sindicacion” con el nombre de la colección y el sufijo RSS2.0
// que identifica el proceso de exportación a este formato de sindicación
mkdir("sindicacion/".$archivoName."_rss2", 0777);

// Se crea la carpeta “data” dentro del directorio anterior de forma tal que puedan ser guardados los
// archivos de datos necesarios, así como las estadísticas parciales del proceso
mkdir("sindicacion/".$archivoName."_rss2/data", 0777);

$contadorA=1; // Contador de archivos
$contadorB=0; // Contador de registros
$maxitems=1000; // Límite de registros por archivo XML en formato RSS2.0 exportado

// La variable $file1 crea un archivo OPML en el directorio de la colección exportada que contendrá
// todas las URL a los archivos XML con la colección bibliográfica en formato Atom.
$file1 = fopen("sindicacion/".$archivoName."_rss2/$archivoName.opml","w");
```

```
// Se escribe la cabecera y el canal del archivo OPML
fwrite($file1,"<?xml version='1.0' encoding='UTF-8'?>
<opml version='1.0'>
<head>
<title></title>
<dateCreated></dateCreated>
<dateModified></dateModified>
<ownerName></ownerName>
<ownerEmail></ownerEmail>
</head>
<body>
");

// Bucle principal que permite ejecutar el código debajo descrito, mientras el contador de registros
// no supere el total definido de la colección, definido por la variable $rcount
while($contadorB != $rcount)  {

// La variable $archivoActual construye el nombre de cada archivo que será exportado a partir del
// nombre de la colección y el valor del contador de archivos
$archivoActual = $archivoName.".$contadorA";

// Se crea un archivo que contendrá la codificación de la colección bibliográfica en formato RSS2.0.
// Para ello se construye un nombre de archivo diferenciador basado en el contador correlativo de
// archivos anteriormente mencionado
$file = fopen("sindicacion/".$archivoName."_rss2/$archivoActual.rss2.xml","w");

// Escritura del elemento “Outline” del archivo OPML que incluye la URL de los archivos creados
// durante el proceso de exportación. Este proceso es necesario para su reagrupación y posterior
// lectura desde un parser
fwrite($file1,"<outline type='rss2' title='$archivoName-rss2 Parte: $contadorA'
xmlUrl='sindicacion/".$archivoName."_rss2/$archivoActual.rss2.xml' />");

// Dentro del mismo bucle se comienza a escribir la cabecera del archivo RSS2.0
fwrite($file,"<?xml version='1.0' encoding='UTF-8'?>
<?xml-stylesheet href='.xsl' type='text/xslt' media='screen'?>
<?xml-stylesheet href='.css' type='text/css' media='screen'?>
<rss version='2.0' xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
xmlns:sy='http://purl.org/rss/1.0/modules/syndication/'>");

// Se ejecuta la consulta de la tabla channel en correspondencia con la colección seleccionada, de
// forma que puedan ser escritos los datos de descripción de cada canal de sindicación que será
// exportado
$sql1 = "SELECT * FROM channel WHERE colname LIKE '$_POST[colexport]'";
$resultado1 = mysql_query($sql1);
$row1 = mysql_fetch_assoc($resultado1);

// Escritura de los valores obtenidos en la consulta en el archivo XML
fwrite($file,"
<channel rdf:about='enlace directo al canal'>
<title>$row1[title]</title>  // Título del canal
<link>$row1[link]</link>  // URL del canal
<description>$row1[description]</description>  // Descripción del contenido del canal
<language>es</language>  // Idioma
<category></category>  // Categoría temática del canal
<sy:updatePeriod>$row1[syupdatePeriod]</sy:updatePeriod>  // Periodo de actualización
<sy:updateFrequency>$row1[syupdateFrequency]</sy:updateFrequency>  // Frecuencia act.
<sy:updateBase>$row1[syupdateBase]</sy:updateBase> // Fecha de actualización
");
```

```
// Impresión en pantalla del número de archivo que se está tratando
echo "Nº de Archivo: $contadorA ";

// Impresión en pantalla del número de registro bibliográfico por el que se comienza a trabajar
echo "Nº de Registro: $contadorB ";

echo "<div class='detalle'>";

// Se ejecuta la consulta SQL para extraer todos los registros de la colección atendiendo al
// contador de registros límite $maxitems
$sql2 = "SELECT * FROM $_POST[colexport] LIMIT $contadorB,$maxitems";
$resultado2 = mysql_query($sql2);

// Segundo bucle que permite la escritura de un registro bibliográfico codificado en formato RSS2.0,
// dentro del archivo de exportación anteriormente creado
while ($row2 = mysql_fetch_assoc($resultado2)) {
fwrite($file,"
<item>
<title>$row2[a9]</title>  // Título propiamente dicho
<author>$row2[a7]</author>  // Mención de responsabilidad
<pubDate>$row2[a20]</pubDate>  // Fecha de publicación
<source></source>  // Fuente de información citada
<guid>$row2[a0]</guid>  // Identificador del documento
<link></link>  // URL del registro bibliográfico
<description>$row2[a9] $row2[a7] $row2[a15] $row2[a16]
 $row2[a18] $row2[a19] $row2[a20] $row2[a26] $row2[a32]
 $row2[a33] $row2[a41] $row2[a42] $row2[a44] $row2[a45]</description>  // Ficha completa
<enclosure></enclosure> // URL de archivo o material adjunto
<comments></comments>  // Comentarios
<category>$row2[a45]</category>  // Categoría temática
</item>
");

$contadorB++;  // Incremento del contador de registros
echo " + "; // Impresión en pantalla de un signo que representa el registro escrito
}

echo "</div><br/>";

// Escritura del final del archivo de exportación cuando llega al límite de 1000 registros planteado
// anteriormente en la variable $maxitems
fwrite($file,"</channel></rss>");

// Cierre del archivo de exportación
fclose($file);
$contadorA++; // Incremento del contador de archivos
}
// Escritura del final del archivo OPML
fwrite($file1,"</body></opml>");

// Cierre del archivo OPML
fclose($file1);
```

Archivo exportar-marc1.php

El archivo exportar-marc1.php contiene la función generadora del canal de sindicación MARC-XML abreviado. Su funcionamiento es similar al relatado anteriormente en Atom a diferencia de las especificaciones empleadas (MARC-XML Schema, 2009).

Título: Exportación MARC-XML abreviado del archivo exportar-marc1.php
Referencia: tabla99

```
// Se imprime en pantalla la variable $archivoName que es el nombre de la colección elegida para
// la exportación. Por otro lado se imprime la comprobación de la selección mediante la variable
// $_POST[colexport] enviada por formulario vía http POST
echo "Valor de archivoName: $archivoName<br/>";
echo "Valor POST: $_POST[colexport]<br/>";

// Incorporación del archivo “config.php” con los datos de conexión a MySQL
include("config.php");

// Selección de la base de datos SYNC sobre la que se opera
mysql_select_db("$database", $con);

// Se ejecuta una consulta SQL para contabilizar el número de registros disponibles en la tabla
$sql="SELECT COUNT(*) FROM $_POST[colexport]";
$consulta=mysql_query($sql); $rcount=mysql_result($consulta,0);
echo "Nº Total de Registros en tabla $_POST[colexport]: $rcount<br/>";

// Se crea una carpeta en el directorio “sindicacion” con el nombre de la colección y el sufijo Marc1
// que identifica el proceso de exportación a este formato de sindicación
mkdir("sindicacion/".$archivoName."_marc1", 0777);

// Se crea la carpeta “data” dentro del directorio anterior de forma tal que puedan ser guardados los
// archivos de datos necesarios, así como las estadísticas parciales del proceso
mkdir("sindicacion/".$archivoName."_marc1/data", 0777);

$contadorA=1; // Contador de archivos
$contadorB=0; // Contador de registros
$maxitems=1000; // Límite de registros por archivo XML en formato Marc1 exportado

// La variable $file1 crea un archivo OPML en el directorio de la colección exportada que contendrá
// todas las URL a los archivos XML con la colección bibliográfica en formato Marc1.
$file1 = fopen("sindicacion/".$archivoName."_marc1/$archivoName.opml","w");

// Se escribe la cabecera y el canal del archivo OPML
fwrite($file1,"<?xml version='1.0' encoding='UTF-8'?>
<opml version='1.0'>
<head>
<title></title>
<dateCreated></dateCreated>
<dateModified></dateModified>
<ownerName></ownerName>
<ownerEmail></ownerEmail>
</head>
<body>
");
```

```
// Bucle principal que permite ejecutar el código debajo descrito, mientras el contador de registros
// no supere el total definido de la colección, definido por la variable $rcount
while($contadorB != $rcount)  {

// La variable $archivoActual construye el nombre de cada archivo que será exportado a partir del
// nombre de la colección y el valor del contador de archivos
$archivoActual = $archivoName.".$contadorA";

// Se crea un archivo que contendrá la codificación de la colección bibliográfica en formato Atom.
// Para ello se construye un nombre de archivo diferenciador basado en el contador correlativo de
// archivos anteriormente mencionado
$file = fopen("sindicacion/".$archivoName."_marc1/$archivoActual.marc1.xml","w");

// Escritura del elemento “Outline” del archivo OPML que incluye la URL de los archivos creados
// durante el proceso de exportación. Este proceso es necesario para su reagrupación y posterior
// lectura desde un parser
fwrite($file1,"<outline type='marc1' title='$archivoName-marc1 Parte: $contadorA'
xmlUrl='sindicacion/".$archivoName."_marc1/$archivoActual.marc1.xml' />");

// Dentro del mismo bucle se comienza a escribir la cabecera del archivo Marc1
fwrite($file,"<?xml version='1.0' encoding='utf-8'?>");
fwrite($file,"<collection>");

// Impresión en pantalla del número de archivo que se está tratando
echo "Nº de Archivo: $contadorA ";

// Impresión en pantalla del número de registro bibliográfico por el que se comienza a trabajar
echo "Nº de Registro: $contadorB ";

echo "<div class='detalle'>";

// Se ejecuta la consulta SQL para extraer todos los registros de la colección atendiendo al
// contador de registros límite $maxitems
$sql2 = "SELECT * FROM $_POST[colexport] LIMIT $contadorB,$maxitems";
$resultado2 = mysql_query($sql2);

// Segundo bucle que permite la escritura de un registro bibliográfico codificado en formato Atom,
// dentro del archivo de exportación anteriormente creado
while ($row2 = mysql_fetch_assoc($resultado2)) {

fwrite($file,"
<record>

<leader>00000na a22000005i 4500 </leader>  // Cabecera

<controlfield tag='001'>$row2[a0]</controlfield>  // Número de control
<controlfield tag='003'>$_POST[uidcode]</controlfield> // Identificador del número de control
<controlfield tag='005'>$dateiso8601</controlfield> // Fecha y hora de la última actualización

<datafield tag='010' ind1=" ind2="">
<subfield code='a'>$row2[a42]</subfield>  // Número de control de la Library of Congress
</datafield>

<datafield tag='017' ind1=" ind2="">
<subfield code='a'>$row2[a41]</subfield>  // Número de depósito legal
</datafield>

<datafield tag='020' ind1=" ind2="">
<subfield code='a'>$row2[a41]</subfield>  // ISBN International Standard Book Number
</datafield>
```

```
<datafield tag='041' ind1='' ind2=''>
<subfield code='a'></subfield>  // Código de idioma
</datafield>

<datafield tag='050' ind1='1' ind2='0'>
<subfield code='a'>$row2[a44]</subfield>  // Signatura topográfica
</datafield>

<datafield tag='080' ind1='' ind2=''>
<subfield code='a'></subfield>  // CDU Clasificación Decimal Universal
</datafield>

<datafield tag='082' ind1='0' ind2=''>
<subfield code='a'>$row2[a44]</subfield>  // Clasificación Decimal Dewey
</datafield>

<datafield tag='100' ind1='0' ind2=''>
<subfield code='a'>$row2[a7]</subfield>  // Asiento principal, entidad personal
<subfield code='0'></subfield>  // Identificador de la entidad personal
</datafield>

<datafield tag='110' ind1='2' ind2=''>
<subfield code='a'></subfield>  // Asiento principal, entidad corporativa
<subfield code='0'></subfield>  // Identificador de la entidad corporativa
</datafield>

<datafield tag='111' ind1='2' ind2=''>
<subfield code='a'></subfield>  // Asiento principal de reuniones
<subfield code='0'></subfield> // Identificador de la entidad de reuniones
</datafield>

<datafield tag='130' ind1='0' ind2=''>
<subfield code='a'></subfield>  // Asiento principal de título uniforme
<subfield code='0'></subfield>  // Identificador de título uniforme
</datafield>

<datafield tag='245' ind1='1' ind2='0'>
<subfield code='a'>$row2[a9]</subfield>  // Título propiamente dicho
<subfield code='b'></subfield>  // Subtítulo
<subfield code='c'>$row2[a7]</subfield>  // Mención de responsabilidad
</datafield>

<datafield tag='250' ind1='' ind2=''>
<subfield code='a'>$row2[a15]</subfield> // Edición
<subfield code='b'>$row2[a16]</subfield> // Mención de edición
</datafield>

<datafield tag='260' ind1='' ind2=''>
<subfield code='a'>$row2[a18]</subfield> // Lugar de publicación
<subfield code='b'>$row2[a19]</subfield> // Editorial
<subfield code='c'>$row2[a20]</subfield> // Fecha de publicación
</datafield>

<datafield tag='300' ind1='' ind2=''>
<subfield code='a'>$row2[a26]</subfield> // Extensión
<subfield code='b'>$row2[a27]</subfield> // Otros detalles físicos
<subfield code='c'></subfield> // Dimensiones
<subfield code='e'></subfield> // Materiales adjuntos
</datafield>
```

```
<datafield tag='490' ind1='0' ind2=''>
<subfield code='a'>$row2[a32]</subfield>  // Título de la serie
<subfield code='v'>$row2[a33]</subfield>  // Número o volumen de la serie
</datafield>

<datafield tag='500' ind1='' ind2=''>
<subfield code='a'>$row2[a43]</subfield> // Nota general
</datafield>

<datafield tag='650' ind1='1' ind2='4'>
<subfield code='a'></subfield>  // Asiento secundaria de materia
<subfield code='0'></subfield> // Identificador o número de control de materia
</datafield>

<datafield tag='700' ind1='0' ind2=''>
<subfield code='a'></subfield> // Asiento secundaria de entidad personal
<subfield code='0'></subfield> // Identificador de entidad personal
</datafield>

<datafield tag='710' ind1='2' ind2=''>
<subfield code='a'></subfield> // Asiento secundaria de entidad corporativa
<subfield code='0'></subfield> // Identificador de entidad corporativa
</datafield>

<datafield tag='887' ind1='' ind2=''> // Campo No Marc
<subfield code='a'></subfield> // Ficha bibliográfica completa
</datafield>

</record>
");

$contadorB++;  // Incremento del contador de registros
echo " + ";  // Impresión en pantalla de un signo que representa el registro escrito
}

echo "</div><br/>";

// Escritura del final del archivo de exportación cuando llega al límite de 1000 registros planteado
// anteriormente en la variable $maxitems
fwrite($file,"</collection>");

// Cierre del archivo de exportación
fclose($file);

$contadorA++; // Incremento del contador de archivos
}

// Escritura del final del archivo OPML
fwrite($file1,"</body></opml>");

// Cierre del archivo OPML
fclose($file1);
```

Archivo exportar-marc2.php

El archivo exportar-marc2.php contiene la función generadora del canal de sindicación MARC-XML abreviado. Su funcionamiento es similar al relatado anteriormente en Atom a diferencia de las especificaciones empleadas (MARC-XML Schema, 2009).

Título: Exportación MARC-XML Extendido
Referencia: tabla100

```
// Se imprime en pantalla la variable $archivoName que es el nombre de la colección elegida para
// la exportación. Por otro lado se imprime la comprobación de la selección mediante la variable
// $_POST[colexport] enviada por formulario vía http POST
echo "Valor de archivoName: $archivoName<br/>";
echo "Valor POST: $_POST[colexport]<br/>";

// Incorporación del archivo “config.php” con los datos de conexión a MySQL
include("config.php");

// Selección de la base de datos SYNC sobre la que se opera
mysql_select_db("$database", $con);

// Se ejecuta una consulta SQL para contabilizar el número de registros disponibles en la tabla
$sql="SELECT COUNT(*) FROM $_POST[colexport]";
$consulta=mysql_query($sql); $rcount=mysql_result($consulta,0);
echo "Nº Total de Registros en tabla $_POST[colexport]: $rcount<br/>";

// Se crea una carpeta en el directorio “sindicacion” con el nombre de la colección y el sufijo Marc2
// que identifica el proceso de exportación a este formato de sindicación
mkdir("sindicacion/".$archivoName."_marc2", 0777);

// Se crea la carpeta “data” dentro del directorio anterior de forma tal que puedan ser guardados los
// archivos de datos necesarios, así como las estadísticas parciales del proceso
mkdir("sindicacion/".$archivoName."_marc2/data", 0777);

$contadorA=1; // Contador de archivos
$contadorB=0; // Contador de registros
$maxitems=1000; // Límite de registros por archivo XML en formato Marc2 exportado

// La variable $file1 crea un archivo OPML en el directorio de la colección exportada que contendrá
// todas las URL a los archivos XML con la colección bibliográfica en formato Marc2.
$file1 = fopen("sindicacion/".$archivoName."_marc2/$archivoName.opml","w");

// Se escribe la cabecera y el canal del archivo OPML
fwrite($file1,"<?xml version='1.0' encoding='UTF-8'?>
<opml version='1.0'>
<head>
<title></title>
<dateCreated></dateCreated>
<dateModified></dateModified>
<ownerName></ownerName>
<ownerEmail></ownerEmail>
</head>
<body>
");
```

```
// Bucle principal que permite ejecutar el código debajo descrito, mientras el contador de registros
// no supere el total definido de la colección, definido por la variable $rcount
while($contadorB != $rcount)  {

// La variable $archivoActual construye el nombre de cada archivo que será exportado a partir del
// nombre de la colección y el valor del contador de archivos
$archivoActual = $archivoName.".$contadorA";

// Se crea un archivo que contendrá la codificación de la colección bibliográfica en formato Marc2.
// Para ello se construye un nombre de archivo diferenciador basado en el contador correlativo de
// archivos anteriormente mencionado
$file = fopen("sindicacion/".$archivoName."_marc2/$archivoActual.marc2.xml","w");

// Escritura del elemento “Outline” del archivo OPML que incluye la URL de los archivos creados
// durante el proceso de exportación. Este proceso es necesario para su reagrupación y posterior
// lectura desde un parser
fwrite($file1,"<outline type='marc2' title='$archivoName-marc2 Parte: $contadorA'
xmlUrl='sindicacion/".$archivoName."_marc2/$archivoActual.marc2.xml' />");

// Dentro del mismo bucle se comienza a escribir la cabecera del archivo Marc2
fwrite($file,"<?xml version='1.0' encoding='utf-8'?>");
fwrite($file,"<marc:collection xmlns:marc='http://www.loc.gov/MARC21/slim'
                               xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
                               xsi:schemaLocation='http://www.loc.gov/MARC21/slim
                               http://www.loc.gov/standards/marcxml/schema/MARC21slim.xsd'>");

// Impresión en pantalla del número de archivo que se está tratando
echo "Nº de Archivo: $contadorA ";

// Impresión en pantalla del número de registro bibliográfico por el que se comienza a trabajar
echo "Nº de Registro: $contadorB ";

echo "<div class='detalle'>";

// Se ejecuta la consulta SQL para extraer todos los registros de la colección atendiendo al
// contador de registros límite $maxitems
$sql2 = "SELECT * FROM $_POST[colexport] LIMIT $contadorB,$maxitems";
$resultado2 = mysql_query($sql2);

// Segundo bucle que permite la escritura de un registro bibliográfico codificado en formato Marc2,
// dentro del archivo de exportación anteriormente creado
while ($row2 = mysql_fetch_assoc($resultado2)) {
fwrite($file,"
<marc:record>

<marc:leader>00000na a22000005i 4500 </ marc:leader>  // Cabecera

<marc:controlfield tag='001'>$row2[a0]</ marc:controlfield>  // Número de control
<marc:controlfield tag='003'>$_POST[uidcode]</ marc:controlfield> // Identificador de control
<marc:controlfield tag='005'>$dateiso8601</ marc:controlfield> // Fecha de la última actualización

<marc:datafield tag='010' ind1=" ind2="">
<marc:subfield code='a'>$row2[a42]</ marc:subfield>  // Número de control de la LC
</marc:datafield>

<marc:datafield tag='017' ind1=" ind2="">
<marc:subfield code='a'>$row2[a41]</marc:subfield>  // Número de depósito legal
</marc:datafield>

<marc:datafield tag='020' ind1=" ind2="">
```

```
<marc:subfield code='a'>$row2[a41]</marc:subfield>  // ISBN International Standard Book Number
</ marc:datafield>

<marc:datafield tag='041' ind1='' ind2=''>
<marc:subfield code='a'></marc:subfield>  // Código de idioma
</marc:datafield>

<marc:datafield tag='050' ind1='1' ind2='0'>
<marc:subfield code='a'>$row2[a44]</marc:subfield>  // Signatura topográfica
</marc:datafield>

<marc:datafield tag='080' ind1='' ind2=''>
<marc:subfield code='a'></marc:subfield>  // CDU Clasificación Decimal Universal
</marc:datafield>

<marc:datafield tag='082' ind1='0' ind2=''>
<marc:subfield code='a'>$row2[a44]</marc:subfield>  // Clasificación Decimal Dewey
</marc:datafield>

<marc:datafield tag='100' ind1='0' ind2=''>
<marc:subfield code='a'>$row2[a7]</marc:subfield>  // Asiento principal, entidad personal
<marc:subfield code='0'></marc:subfield>  // Identificador de la entidad personal
</marc:datafield>

<marc:datafield tag='110' ind1='2' ind2=''>
<marc:subfield code='a'></marc:subfield>  // Asiento principal, entidad corporativa
<marc:subfield code='0'></marc:subfield>  // Identificador de la entidad corporativa
</marc:datafield>

<marc:datafield tag='111' ind1='2' ind2=''>
<marc:subfield code='a'></marc:subfield>  // Asiento principal de reuniones
<marc:subfield code='0'></marc:subfield> // Identificador de la entidad de reuniones
</marc:datafield>

<marc:datafield tag='130' ind1='0' ind2=''>
<marc:subfield code='a'></marc:subfield>  // Asiento principal de título uniforme
<marc:subfield code='0'></marc:subfield>  // Identificador de título uniforme
</marc:datafield>

<marc:datafield tag='245' ind1='1' ind2='0'>
<marc:subfield code='a'>$row2[a9]</marc:subfield>  // Título propiamente dicho
<marc:subfield code='b'></marc:subfield>  // Subtítulo
<marc:subfield code='c'>$row2[a7]</marc:subfield>  // Mención de responsabilidad
</marc:datafield>

<marc:datafield tag='250' ind1='' ind2=''>
<marc:subfield code='a'>$row2[a15]</marc:subfield> // Edición
<marc:subfield code='b'>$row2[a16]</marc:subfield> // Mención de edición
</ marc:datafield>

<marc:datafield tag='260' ind1='' ind2=''>
<marc:subfield code='a'>$row2[a18]</marc:subfield> // Lugar de publicación
<marc:subfield code='b'>$row2[a19]</marc:subfield> // Editorial
<marc:subfield code='c'>$row2[a20]</marc:subfield> // Fecha de publicación
</ marc:datafield>


<marc:datafield tag='300' ind1='' ind2=''>
<marc:subfield code='a'>$row2[a26]</marc:subfield> // Extensión
<marc:subfield code='b'>$row2[a27]</marc:subfield> // Otros detalles físicos
```

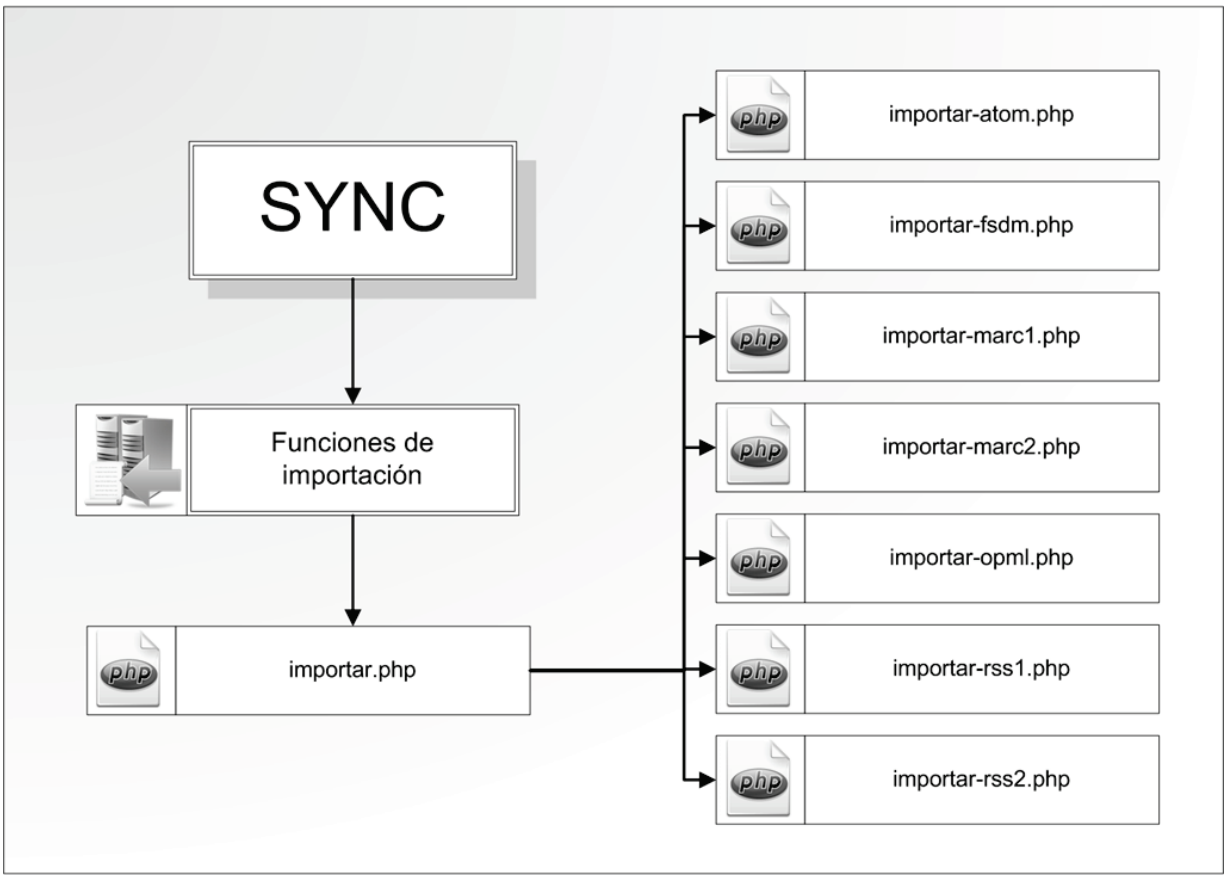
```
<marc:subfield code='c'></marc:subfield> // Dimensiones
<marc:subfield code='e'></marc:subfield> // Materiales adjuntos
</marc:datafield>

<marc:datafield tag='490' ind1='0' ind2=''>
<marc:subfield code='a'>$row2[a32]</marc:subfield>  // Título de la serie
<marc:subfield code='v'>$row2[a33]</marc:subfield>  // Número o volumen de la serie
</marc:datafield>

< marc:datafield tag='500' ind1='' ind2=''>
< marc:subfield code='a'>$row2[a43]</ marc:subfield> // Nota general
</ marc:datafield>

<marc:datafield tag='650' ind1='1' ind2='4'>
<marc:subfield code='a'></marc:subfield>  // Asiento secundaria de materia
<marc:subfield code='0'></marc:subfield> // Identificador o número de control de materia
</marc:datafield>

<marc:datafield tag='700' ind1='0' ind2=''>
<marc:subfield code='a'></marc:subfield> // Asiento secundaria de entidad personal
<marc:subfield code='0'></marc:subfield> // Identificador de entidad personal
</marc:datafield>

<marc:datafield tag='710' ind1='2' ind2=''>
<marc:subfield code='a'></marc:subfield> // Asiento secundaria de entidad corporativa
<marc:subfield code='0'></marc:subfield> // Identificador de entidad corporativa
</marc:datafield>

<marc:datafield tag='887' ind1='' ind2=''> // Campo No Marc
<marc:subfield code='a'></marc:subfield> // Ficha bibliográfica completa
</marc:datafield>

</marc:record>
");


$contadorB++;  // Incremento del contador de registros
echo " + "; // Impresión en pantalla de un signo que representa el registro escrito
}

echo "</div><br/>";

// Escritura del final del archivo de exportación cuando llega al límite de 1000 registros planteado
// anteriormente en la variable $maxitems
fwrite($file,"</marc:collection>");

// Cierre del archivo de exportación
fclose($file);
$contadorA++; // Incremento del contador de archivos
}

// Escritura del final del archivo OPML
fwrite($file1,"</body></opml>");

// Cierre del archivo OPML
fclose($file1);
```

7.4.4. Programa de importación

El programa de importación ha sido concebido para la incorporación de catálogos bibliográficos sindicados en los formatos RSS1.0, RSS2.0, Atom, MARC-XML abreviado y extendido. De esta forma las colecciones que fueron exportadas, serán importadas a la base de datos de la plataforma SYNC, como si de un sistema agregación o de lectura se tratara, simulando un proceso de suscripción a un catálogo bibliográfico sindicado, experiencia de la que se podrán extraer respuestas a las preguntas planteadas al comienzo de este capítulo, como por ejemplo si es posible sindicar un catálogo bibliográfico. Sin embargo no son las únicas cuestiones u objetivos a tener en cuenta. Como consecuencia evidente del proceso de importación, se somete bajo observación el funcionamiento de la solución OPML de agrupación de archivos y su implicación en el desarrollo de programas parser, que por otra parte se ve condicionada por ello. Todo esto también ha implicado la investigación y prueba de parsers desarrollados con la función *simplexml_load_file()* y *simpleXMLElement()* de forma que se pueda determinar que cuál es la más eficiente y en qué casos se debe aplicar. En este programa también se han estudiado las dificultades que plantean los formatos de sindicación para el análisis y lectura de los canales de sindicación a través de los parser desarrollados, dificultades que han sido solucionadas.

En relación a la estructura del programa de importación, está compuesto por un conjunto de archivos que contienen las funciones y programas parser especializados en cada formato de sindicación. Por ello su organización resulta muy parecida al programa de exportación, pero no en su desempeño.

Título: Diagrama del programa de importación
Referencia: figura30



- **Archivo importar.php.** El archivo importar.php es la cabecera del programa de importación. Su función principal consiste en proporcionar al usuario un formulario con las opciones de importación posibles. Éstas son el campo de URL del canal de sindicación, el formato del canal de sindicación, la modalidad de agrupación del canal y el nombre atribuido a la colección que se importará.

- **Campo de URL.** El campo de URL ha sido automatizado para reconocer todas las colecciones que fueron sindicadas. Esto es posible dado que están perfectamente localizadas en la carpeta *sindicacion*.

- **Formato de sindicación.** El programa de importación requiere precisar el formato del canal de sindicación, para redireccionar a la función parser adecuada.

- **Modalidad de agrupación.** Esta opción pregunta si el canal de sindicación está fragmentado con método OPML o no. Por defecto la opción fragmentada siempre está marcada. Esto se debe a que con todas las exportaciones que se efectúan se crea un archivo OPML obligatorio. Esto incluye, en consecuencia, a las colecciones de menos de 1000 registros, aunque no estén fragmentadas.

- **Nombre del canal.** Se trata de un campo de texto, en el que el usuario puede atribuir un nombre al catálogo que va a importar. De esta forma y bajo ese nombre quedará registrado en la plataforma SYNC.

Título: Impresión de pantalla del programa de importación
Referencia: figura31

Modelo de parser desarrollado

Los procesos de importación y transferencia basados en archivos XML, requieren de programas parser para su lectura, tal y como se ha explicado en capítulos anteriores. Para esta investigación se han desarrollado dos tipos de parser que permiten efectuar análisis y lectura de archivos XML para posteriormente poder acceder a su contenido, tratarlo o sencillamente importarlo. El primer tipo de parser, emplea la función php *simplexml_load_file()*. Como se ha explicado en el capítulo 6, dicha función como su propio nombre indica, procede a la carga completa del archivo XML indicado por la variable *$feed*. Dado que el proceso no implica la

declaración de ningún elemento u objeto, la información analizada por el parser es soportada en la memoria cache. Por este motivo la función *simplexml_load_file()* no es válida para ser empleada en tandas de archivos XML de gran volumen, como los catálogos bibliográficos. De hecho su empleo debe ser cuidadoso para evitar cargar en exceso la sesión del navegador pudiendo llegar incluso a bloquearlo. Por el contrario su empleo con pequeños archivos XML o canales de sindicación de entre 10 y 20 registros de extensión variable, puede resultar útil y práctico, evitando el empleo de un parser más complejo.

Título: Ejemplo de parser *simplexml_load_file()*
Referencia: tabla101

```
$feed = "url-canal-sindicacion";
$xml = simplexml_load_file($feed);
for($i=0; $xml->item[$i]; $i++) {

    $title = $xml->item[$i]->xpath('//dc:title');
    echo "<p><h3>Title:</h3> $title[$i]<br/></p>

}
```

Teniendo en cuenta todas estas consideraciones, este tipo de parser se emplea en el programa de importación para el análisis de los archivos OPML (de tamaño o peso muy ligero) que contienen las URL de cada archivo XML que configura el canal de sindicación correspondiente al catálogo bibliográfico. Las URL extraídas son tratadas por un parser más potente. Se trata del segundo tipo de parser, que emplea la función php *simpleXMLElement()*.

Título: Ejemplo de parser *simpleXMLElement()*
Referencia: tabla102

```
$feed = "url-canal-sindicacion";
$sxe = new SimpleXMLElement($feed, NULL, TRUE);
$title = $sxe->xpath("//dc:title");

$count=0;
foreach ($title as $control) {
     echo "<p><h3>Title:</h3> $control<br/></p>
}
```

La función *simpleXMLElement()* está diseñada para crear un objeto accesible mediante array de arrays o lo que es lo mismo una matriz de matrices en las que se almacenan todos los contenidos del archivo XML analizado. Este método de procesamiento es mucho más efectivo a la hora de tratar grandes cantidades de registros, debido a que exige un menor esfuerzo de computación interna al módulo de PHP en el servidor Apache; también ello implica que no carga el trabajo de análisis a la memoria cache del navegador, lo que evita bloqueos inesperados durante su ejecución. Además la función *simpleXMLElement()*, sí se considera manipulable como objeto, de hecho hace uso de la clase DOM (Document Object Model) utilizada ampliamente para el análisis de nodos en archivos XML. Estas características hacen que resulte más rápido el acceso mediante XPath a un determinado elemento del canal de sindicación.

De esta forma, los parser basados en *simpleXMLElement()* analizan archivos XML de 1000 registros cada uno de forma secuencial a partir de las URL extraídas por Parsers más sencillos, logrando una combinación equilibrada entre rendimiento y fiabilidad del proceso de parseo.

Ajustes en el desarrollo de parser por el método de acceso a nodos XPath

Uno de los aspectos más complejos de solucionar en el desarrollo de un parser es el método y rutina de acceso a los nodos o elementos de un archivo XML. Si bien esta labor con entidades simples resulta una tarea sencilla, este proceso puede complicarse al emplear los prefijos del espacio de nombres de un determinado formato o módulo de sindicación. Estos casos son frecuentes con el formato de sindicación RSS1.0 especialmente diseñado para su compatibilidad con RDF y su extensibilidad con módulos Dublin Core, Syndication, Content o PRISM. No obstante el caso más complejo lo brinda el formato MARC-XML extendido que consta de diversos niveles de anidamiento que en todo caso emplean prefijos de su namespace, véase el siguiente ejemplo.

Título: Ejemplo de rutas de acceso a nodos mediante XPath
Referencia: tabla103

```
// Acceso a elementos simple
$title = $sxe->xpath("//entry/title");

// Acceso a elementos con prefijos propios de su Namespace
$tag245 = $sxe->xpath("//marc:datafield[@tag='245']/marc:subfield[@code='a']");
```

Archivo importar-atom.php

Contiene la función y parser especializado en el formato de sindicación Atom. El proceso de importación comienza por la creación de una estructura de carpetas en el directorio *sindicacion* que denominará con el mismo nombre que el usuario haya atribuido a la colección que se desea importar. En estas carpetas se guardará únicamente información estadística del proceso. A continuación, se configura el nombre definitivo para la colección que se importará, que se configura con prefijo *col_* y sufijo *_atom (col_nombre-atribuido_atom)*. Esta denominación es importante, ya que será utilizada para el nombre de la tabla en la base de datos de la plataforma SYNC, resultando clave para identificar las colecciones bibliográficas sindicadas que han sido importadas de las que no. Seguidamente, la función de importación crea una tabla en la base de datos con dicha denominación de la colección y los campos propios del formato Atom coincidentes con los tratados en las especificaciones oficiales del formato.

Una vez creada la tabla en la que se alojarán los registros de la colección bibliográfica, se pone en funcionamiento el parser de OPML basado en la función *simplexml_load_file(),* que obtendrá las URL de los archivos XML del canal de sindicación en el que se encuentra formateado el catálogo bibliográfico. Obtenidas una a una, éstas son analizadas por un segundo parser basado en la función *simpleXMLElement()* capaz de leer y extraer cada valor recogido en cada entidad o elemento del archivo XML más voluminoso. Consecuentemente registro a registro es incorporado en la tabla creada anteriormente, hasta finalizar con todos los archivos que componen el canal de sindicación

En referencia al parser del formato Atom, la estructura de acceso a los elementos y nodos en XPath no resulta compleja si se indica una ruta absoluta (utilizando doble slash o stroke //) a partir del elemento *entry* hasta la entidad correspondiente *($title = $sxe >xpath("//entry/title");)*.

Título: Importación Atom
Referencia: tabla104

```
echo "<h2>LECTOR ATOM</h2><br/>";

// La variable $channelname construye el nombre de la tabla en la que se importarán los contenidos
// del canal de sindicación en formato Atom. La denominación comprende un prefijo “col_”, el
// nombre de la colección o del canal de sindicación seleccionado y su sufijo correspondiente al
// formato “_atom”
$channelname = "col_".$_POST['channelname']."_atom";

// Incorporación del archivo “config.php” con los datos de conexión a MySQL
include("config.php");

// Selección de la base de datos SYNC sobre la que se opera
mysql_select_db("$database", $con);

// Creación de tabla en la base de datos SYNC para la importación del canal
mysql_query("CREATE TABLE $channelname (
id      INT NOT NULL AUTO_INCREMENT, PRIMARY KEY(id),
title      TEXT CHARACTER SET utf8,
author      TEXT CHARACTER SET utf8,
updated      TEXT CHARACTER SET utf8,
content      TEXT CHARACTER SET utf8,
link      TEXT CHARACTER SET utf8,
summary      TEXT CHARACTER SET utf8,
category      TEXT CHARACTER SET utf8,
contributor      TEXT CHARACTER SET utf8,
published      TEXT CHARACTER SET utf8,
issued      TEXT CHARACTER SET utf8,
modified      TEXT CHARACTER SET utf8,
source      TEXT CHARACTER SET utf8,
rights      TEXT CHARACTER SET utf8,
FULLTEXT(title, author, content))CHARACTER SET utf8",$con);

// El parser que se muestra a continuación está diseñado para trabajar con canales de sindicación
// fragmentados en múltiples archivos XML y agrupados por medio de OPML, por ese motive se
// utiliza el función switch para distinguirlo de los casos en los que los canales de sindicación son
// simples
switch($_POST['fragmentado']) {
case si:

// La variable $feed contiene la URL del canal de sindicación en formato OPML que enlaza todos
// los archivos XML de la colección bibliográfica en formato Atom
$feed = $_POST['channelfeed'];

// La variable $xml carga el parser simple que actúa sobre el archivo OPML para extraer las URL de
// los archivos Atom que constituyen la colección. Para este proceso se selecciona mediante Xpath
// el atributo xmlUrl del elemento “outline”
$xml = simplexml_load_file($feed);

        $ii=0; // Contador para la automatización de la extracción de las URL de los atributos

        $contadorB=0; // Contador de registros del archivo OPML

        // Bucle para selección de todos los elementos “outline”
        for($i=0; $xml->body->outline[$i]; $i++) {
```

```
            // La variable $outpath contiene la instrucción XPath para seleccionar el atributo xmlUrl que
            // contiene la URL de cada archivo XML que compone la colección bibliográfica sindicada
            // en formato Atom
            $outpath = $xml->body->outline[$i]->xpath('//@xmlUrl');

// La variable $feed1 toma el valor de la URL de cada archivo XML que compone la colección
// bibliográfica
$feed1 = $outpath[$ii];

// Impresión de pantalla del número de archivo y el registro de inicio
echo "Archivo: $i // Registro de Inicio: $contadorB<br/>";

echo "<div class='detalle'>";

// La variable $sxe inicia el parser del formato Atom
$sxe = new SimpleXMLElement($feed1, NULL, TRUE);
$title = $sxe->xpath("//entry/title");
$author = $sxe->xpath("//entry/author/name");
$updated = $sxe->xpath("//entry/updated");
$content = $sxe->xpath("//entry/content");
$link = $sxe->xpath("//entry/link/@href");
$summary = $sxe->xpath("//entry/summary");
$category = $sxe->xpath("//entry/category/@term");
$contributor = $sxe->xpath("//entry/contributor/name");
$published = $sxe->xpath("//entry/published");
$issued = $sxe->xpath("//entry/issued");
$modified = $sxe->xpath("//entry/modified");
$source = $sxe->xpath("//entry/source/id");
$rights = $sxe->xpath("//entry/rights");

$count = 0; // Contador de registros de cada archivo XML en formato Atom

// Bucle para la importación de los contenidos de cada campo bibliográfico en la tabla de la
// colección anteriormente creada
foreach ($title as $control)  {
mysql_query("INSERT DELAYED INTO $channelname (id, title, author, updated, content, link,
summary, category, contributor, published, issued, modified, source, rights) VALUES (NULL,
'$control', '$author[$count]', '$updated[$count]', '$content[$count]', '$link[$count]',
'$summary[$count]', '$category[$count]', '$contributor[$count]', '$published[$count]',
'$issued[$count]', '$modified[$count]', '$source[$count]', '$rights[$count]')");

echo " + ";
$contadorB++; // Incremento en el contador de registros OPML
$count++; // Incremento en el contador de registros del archivo XML en formato Atom
}
echo "</div><br/>";
$ii = $i+1; // Incremento para el contador de atributos xmlUrl de OPML
}
break;
}
```

Archivo importar-rss1.php

Contiene la función y parser especializados en el análisis e importación de canales de sindicación en RSS1.0 RDF. Su funcionamiento es idéntico al definido para Atom. La diferencia más notable se encuentra en la expresión XPath utilizada para acceder a entidades con prefijo namespace. Este tipo de casos son solucionados con el empleo de comillas dobles en la expresión xpath que contendrá el prefijo junto con la puntuación que separa el nombre del elemento *xpath("//dc:title");*

Título: Importación RSS1.0 RDF del archivo importar-rss1.ph
Referencia: tabla105

```
echo "<h1>LECTOR RSS 1.0</h1><br/>";

// La variable $channelname construye el nombre de la tabla en la que se importarán los contenidos
// del canal de sindicación en formato RSS1.0. La denominación comprende un prefijo “col_”, el
// nombre de la colección o del canal de sindicación seleccionado y su sufijo correspondiente al
// formato “_rss1”
$channelname = "col_".$_POST['channelname']."_rss1";

// Incorporación del archivo “config.php” con los datos de conexión a MySQL
include("config.php");

// Selección de la base de datos SYNC sobre la que se opera
mysql_select_db("$database", $con);

// Creación de tabla en la base de datos SYNC para la importación del canal
mysql_query("CREATE TABLE $channelname (
id      INT NOT NULL AUTO_INCREMENT, PRIMARY KEY(id),
title      TEXT CHARACTER SET utf8,
creator      TEXT CHARACTER SET utf8,
contributor      TEXT CHARACTER SET utf8,
publisher      TEXT CHARACTER SET utf8,
date      TEXT CHARACTER SET utf8,
type      TEXT CHARACTER SET utf8,
format      TEXT CHARACTER SET utf8,
identifier      TEXT CHARACTER SET utf8,
subject      TEXT CHARACTER SET utf8,
source      TEXT CHARACTER SET utf8,
language      TEXT CHARACTER SET utf8,
relation      TEXT CHARACTER SET utf8,
coverage      TEXT CHARACTER SET utf8,
rights      TEXT CHARACTER SET utf8,
prismPublicationName      TEXT CHARACTER SET utf8,
prismEdition      TEXT CHARACTER SET utf8,
prismPublisher      TEXT CHARACTER SET utf8,
prismPublicationDate      TEXT CHARACTER SET utf8,
prismIssn      TEXT CHARACTER SET utf8,
skosNote      TEXT CHARACTER SET utf8,
contentEncoded      TEXT CHARACTER SET utf8,
FULLTEXT (title, creator, publisher, contentEncoded))CHARACTER SET utf8",$con);
```

```
// El parser que se muestra a continuación está diseñado para trabajar con canales de sindicación
// fragmentados en múltiples archivos XML y agrupados por medio de OPML, por ese motive se
// utiliza el función switch para distinguirlo de los casos en los que los canales de sindicación son
// simples
switch($_POST['fragmentado']) {
case si:

// La variable $feed contiene la URL del canal de sindicación en formato OPML que enlaza todos
// los archivos XML de la colección bibliográfica en formato RSS1.0
$feed = $_POST['channelfeed'];

// La variable $xml carga el parser simple que actúa sobre el archivo OPML para extraer las URL de
// los archivos RSS1.0 que constituyen la colección. Para este proceso se selecciona mediante
// Xpath el atributo xmlUrl del elemento “outline”
$xml = simplexml_load_file($feed);

        $ii=0;  // Contador para la automatización de la extracción de las URL de los atributos

        $contadorB=0;  // Contador de registros del archivo OPML

        // Bucle para selección de todos los elementos “outline”
        for($i=0; $xml->body->outline[$i]; $i++) {

        // La variable $outpath contiene la instrucción XPath para seleccionar el atributo xmlUrl que
        // contiene la URL de cada archivo XML que compone la colección bibliográfica sindicada
        // en formato RSS1.0
        $outpath = $xml->body->outline[$i]->xpath('//@xmlUrl');

// La variable $feed1 toma el valor de la URL de cada archivo XML que compone la colección
// bibliográfica
$feed1 = $outpath[$ii];

// Impresión de pantalla del número de archivo y el registro de inicio
echo "Archivo: $i // Registro de Inicio: $contadorB<br/>";

echo "<div class='detalle'>";

// La variable $sxe inicia el parser del formato RSS1.0
$sxe = new SimpleXMLElement($feed1, NULL, TRUE);
$title = $sxe->xpath("//dc:title");
$creator = $sxe->xpath("//dc:creator");
$contributor = $sxe->xpath("//dc:contributor");
$publisher = $sxe->xpath("//dc:publisher");
$date = $sxe->xpath("//dc:date");
$type = $sxe->xpath("//dc:type");
$format = $sxe->xpath("//dc:format");
$identifier = $sxe->xpath("//dc:identifier");
$subject = $sxe->xpath("//dc:subject");
$source = $sxe->xpath("//dc:source");
$language = $sxe->xpath("//dc:language");
$relation = $sxe->xpath("//dc:relation");
$coverage = $sxe->xpath("//dc:coverage");
$rights = $sxe->xpath("//dc:rights");
$prismPublicationName = $sxe->xpath("//prism:publicationName");
$prismEdition = $sxe->xpath("//prism:edition");
$prismPublisher = $sxe->xpath("//prism:publisher");
$prismPublicationDate = $sxe->xpath("//prism:publicationDate");
$prismIssn = $sxe->xpath("//prism:issn");
$skosNote = $sxe->xpath("//skos:note");
$contentEncoded = $sxe->xpath("//content:encoded");
```

```
$count=0; // Contador de registros de cada archivo XML en formato RSS1.0

// Bucle para la importación de los contenidos de cada campo bibliográfico en la tabla de la
// colección anteriormente creada
foreach ($title as $control) {
mysql_query("INSERT DELAYED INTO $channelname (id, title, creator, contributor, publisher,
date, type, format, identifier, subject, source, language, relation, coverage, rights,
prismPublicationName, prismEdition, prismPublisher, prismPublicationDate, prismIssn, skosNote,
contentEncoded) VALUES (NULL, '$control', '$creator[$count]', '$contributor[$count]',
'$publisher[$count]', '$date[$count]', '$type[$count]', '$format[$count]', '$identifier[$count]',
'$subject[$count]', '$source[$count]', '$language[$count]', '$relation[$count]', '$coverage[$count]',
'$rights[$count]', '$prismPublicationName[$count]', '$prismEdition[$count]',
'$prismPublisher[$count]', '$prismPublicationDate[$count]', '$prismIssn[$count]',
'$skosNote[$count]', '$contentEncoded[$count]')");

echo " + ";
$contadorB++; // Incremento en el contador de registros OPML
$count++; // Incremento en el contador de registros del archivo XML en formato RSS1.0
}

echo "</div><br/>";
$ii = $i+1; // Incremento para el contador de atributos xmlUrl de OPML
}
break;
}
```

Archivo importar-rss2.php

Contiene la función y parser especializados en el análisis e importación de canales de sindicación en RSS2.0. Su funcionamiento es idéntico al definido para Atom.

Título: Importación RSS2.0 del archivo importar-rss2.php
Referencia: tabla106

```
echo "<h1>LECTOR RSS 2.0</h1><br/>";

// La variable $channelname construye el nombre de la tabla en la que se importarán los contenidos
// del canal de sindicación en formato RSS2.0. La denominación comprende un prefijo “col_”, el
// nombre de la colección o del canal de sindicación seleccionado y su sufijo correspondiente al
// formato “_rss2”
$channelname = "col_".$_POST['channelname']."_rss2";

// Incorporación del archivo “config.php” con los datos de conexión a MySQL
include("config.php");

// Selección de la base de datos SYNC sobre la que se opera
mysql_select_db("$database", $con);

// Creación de tabla en la base de datos SYNC para la importación del canal
mysql_query("CREATE TABLE $channelname (
id      INT NOT NULL AUTO_INCREMENT, PRIMARY KEY(id),
title      TEXT CHARACTER SET utf8,
```

```
author      TEXT CHARACTER SET utf8,
pubDate      TEXT CHARACTER SET utf8,
source      TEXT CHARACTER SET utf8,
guid      TEXT CHARACTER SET utf8,
link      TEXT CHARACTER SET utf8,
description      TEXT CHARACTER SET utf8,
enclosure      TEXT CHARACTER SET utf8,
comments      TEXT CHARACTER SET utf8,
category      TEXT CHARACTER SET utf8,
FULLTEXT(title,author,description,category))CHARACTER SET utf8",$con);

// El parser que se muestra a continuación está diseñado para trabajar con canales de sindicación
// fragmentados en múltiples archivos XML y agrupados por medio de OPML, por ese motive se
// utiliza el función switch para distinguirlo de los casos en los que los canales de sindicación son
// simples
switch($_POST['fragmentado']) {
case si:

// La variable $feed contiene la URL del canal de sindicación en formato OPML que enlaza todos
// los archivos XML de la colección bibliográfica en formato RSS2.0
$feed = $_POST['channelfeed'];

// La variable $xml carga el parser simple que actúa sobre el archivo OPML para extraer las URL de
// los archivos RSS2.0 que constituyen la colección. Para este proceso se selecciona mediante
// Xpath el atributo xmlUrl del elemento “outline”
$xml = simplexml_load_file($feed);

$ii=0; // Contador para la automatización de la extracción de las URL de los atributos
$contadorB=0; // Contador de registros del archivo OPML

// Bucle para selección de todos los elementos “outline”
for($i=0; $xml->body->outline[$i]; $i++) {

// La variable $outpath contiene la instrucción XPath para seleccionar el atributo xmlUrl que
// contiene la URL de cada archivo XML que compone la colección bibliográfica sindicada
// en formato RSS2.0
$outpath = $xml->body->outline[$i]->xpath('//@xmlUrl');

// La variable $feed1 toma el valor de la URL de cada archivo XML que compone la colección
// bibliográfica
 $feed1 = $outpath[$ii];

// Impresión de pantalla del número de archivo y el registro de inicio
echo "Archivo: $i // Registro de Inicio: $contadorB<br/>";

echo "<div class='detalle'>";

// La variable $sxe inicia el parser del formato RSS2.0
$sxe = new SimpleXMLElement($feed1, NULL, TRUE);
$title = $sxe->xpath("//channel/item/title");
$author = $sxe->xpath("//channel/item/author");
$pubDate = $sxe->xpath("//channel/item/pubDate");
$source = $sxe->xpath("//channel/item/source");
$guid = $sxe->xpath("//channel/item/guid");
$link = $sxe->xpath("//channel/item/link");
$description = $sxe->xpath("//channel/item/description");
$enclosure = $sxe->xpath("//channel/item/enclosure");
$comments = $sxe->xpath("//channel/item/comments");
$category = $sxe->xpath("//channel/item/category");
```

```
$count=0; // Contador de registros de cada archivo XML en formato RSS2.0

// Bucle para la importación de los contenidos de cada campo bibliográfico en la tabla de la
// colección anteriormente creada
foreach ($title as $control) {
mysql_query("INSERT DELAYED INTO $channelname (id, title, author, pubDate, source, guid, link,
description, enclosure, comments, category) VALUES (NULL, '$control', '$author[$count]',
'$pubDate[$count]', '$source[$count]', '$guid[$count]', '$link[$count]', '$description[$count]',
'$enclosure[$count]', '$comments[$count]', '$category[$count]')");

echo " + ";
$contadorB++; // Incremento en el contador de registros OPML
$count++; // Incremento en el contador de registros del archivo XML en formato RSS2.0
}

echo "</div><br/>";
$ii = $i+1; // Incremento para el contador de atributos xmlUrl de OPML
}
break;
}
```

Archivo importar-marc1.php

Contiene la función y parser especializados en el análisis e importación de canales de sindicación en MARC-XML abreviado. Su funcionamiento es idéntico al definido para Atom, aunque son destacables diferencias de bulto, como una tabla con mayor número de campos e instrucciones XPath mucho más elaboradas que implican doble consulta de atributos para obtener la información correspondiente a un campo bibliográfico.

Título: Importación MARC-XML Abreviado del archivo importar-marc1.php
Referencia: tabla107

```
echo "<h1>LECTOR MARC1</h1><br/>";

// La variable $channelname construye el nombre de la tabla en la que se importarán los contenidos
// del canal de sindicación en formato Marc1. La denominación comprende un prefijo “col_”, el
// nombre de la colección o del canal de sindicación seleccionado y su sufijo correspondiente al
// formato “_marc1”
$channelname = "col_".$_POST['channelname']."_marc1";

// Incorporación del archivo “config.php” con los datos de conexión a MySQL
include("config.php");

// Selección de la base de datos SYNC sobre la que se opera
mysql_select_db("$database", $con);
```

```
// Creación de tabla en la base de datos SYNC para la importación del canal
mysql_query("CREATE TABLE $channelname (
id          INT NOT NULL AUTO_INCREMENT, PRIMARY KEY(id),
leader              VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag001              VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag003              VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag005              VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag010a             VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag017a             VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag020a             VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag041a             VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag050a             VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag080a             VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag082a             VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag100a             TEXT CHARACTER SET utf8 COLLATE utf8_general_ci,
tag100o             VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag110a             TEXT CHARACTER SET utf8 COLLATE utf8_general_ci,
tag110o             VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag111a             TEXT CHARACTER SET utf8 COLLATE utf8_general_ci,
tag111o             VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag130a             TEXT CHARACTER SET utf8 COLLATE utf8_general_ci,
tag130o             VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag245a             TEXT CHARACTER SET utf8 COLLATE utf8_general_ci,
tag245b             TEXT CHARACTER SET utf8 COLLATE utf8_general_ci,
tag245c             TEXT CHARACTER SET utf8 COLLATE utf8_general_ci,
tag250a             VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag250b             VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag260a             VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag260b             VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag260c             VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag300a             VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag300b             VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag300c             VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag300e             VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag490a             VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag490v             VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag500a             TEXT CHARACTER SET utf8 COLLATE utf8_general_ci,
tag650a             TEXT CHARACTER SET utf8 COLLATE utf8_general_ci,
tag650o             VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag700a             TEXT CHARACTER SET utf8 COLLATE utf8_general_ci,
tag700o             VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag710a             TEXT CHARACTER SET utf8 COLLATE utf8_general_ci,
tag710o             VARCHAR(500) CHARACTER SET utf8 COLLATE utf8_general_ci,
tag887a             TEXT CHARACTER SET utf8 COLLATE utf8_general_ci,
indexer TEXT CHARACTER SET utf8 COLLATE utf8_general_ci,
FULLTEXT(indexer)
) CHARACTER SET utf8 COLLATE utf8_general_ci",$con);


// El parser que se muestra a continuación está diseñado para trabajar con canales de sindicación
// fragmentados en múltiples archivos XML y agrupados por medio de OPML, por ese motive se
// utiliza el función switch para distinguirlo de los casos en los que los canales de sindicación son
// simples
switch($_POST['fragmentado']) {
case si:

// La variable $feed contiene la URL del canal de sindicación en formato OPML que enlaza todos
// los archivos XML de la colección bibliográfica en formato Marc1
$feed = $_POST['channelfeed'];
```

```
// La variable $xml carga el parser simple que actúa sobre el archivo OPML para extraer las URL de
// los archivos Marc1 que constituyen la colección. Para este proceso se selecciona mediante Xpath
// el atributo xmlUrl del elemento “outline”
$xml = simplexml_load_file($feed);

$ii=0; // Contador para la automatización de la extracción de las URL de los atributos
$contadorB=0;  // Contador de registros del archivo OPML

// Bucle para selección de todos los elementos “outline”
for($i=0; $xml->body->outline[$i]; $i++) {

// La variable $outpath contiene la instrucción XPath para seleccionar el atributo xmlUrl que
// contiene la URL de cada archivo XML que compone la colección bibliográfica sindicada
// en formato Marc1
$outpath = $xml->body->outline[$i]->xpath('//@xmlUrl');

// La variable $feed1 toma el valor de la URL de cada archivo XML que compone la colección
// bibliográfica
$feed1 = $outpath[$ii];

// Impresión de pantalla del número de archivo y el registro de inicio
echo "Archivo: $i // Registro de Inicio: $contadorB<br/>";

echo "<div class='detalle'>";

// La variable $sxe inicia el parser del formato Marc1
$sxe = new SimpleXMLElement($feed1, NULL, TRUE);
$leader = $sxe->xpath("//record/leader");
$tag001 = $sxe->xpath("//record/controlfield[@tag='001']");
$tag003 = $sxe->xpath("//record/controlfield[@tag='003']");
$tag005 = $sxe->xpath("//record/controlfield[@tag='005']");
$tag010a = $sxe->xpath("//record/datafield[@tag='010']/subfield[@code='a']");
$tag017a = $sxe->xpath("//record/datafield[@tag='017']/subfield[@code='a']");
$tag020a = $sxe->xpath("//record/datafield[@tag='020']/subfield[@code='a']");
$tag041a = $sxe->xpath("//record/datafield[@tag='041']/subfield[@code='a']");
$tag050a = $sxe->xpath("//record/datafield[@tag='050']/subfield[@code='a']");
$tag080a = $sxe->xpath("//record/datafield[@tag='080']/subfield[@code='a']");
$tag082a = $sxe->xpath("//record/datafield[@tag='082']/subfield[@code='a']");
$tag100a = $sxe->xpath("//record/datafield[@tag='100']/subfield[@code='a']");
$tag100o = $sxe->xpath("//record/datafield[@tag='100']/subfield[@code='o']");
$tag110a = $sxe->xpath("//record/datafield[@tag='110']/subfield[@code='a']");
$tag110o = $sxe->xpath("//record/datafield[@tag='110']/subfield[@code='o']");
$tag111a = $sxe->xpath("//record/datafield[@tag='111']/subfield[@code='a']");
$tag111o = $sxe->xpath("//record/datafield[@tag='111']/subfield[@code='o']");
$tag130a = $sxe->xpath("//record/datafield[@tag='130']/subfield[@code='a']");
$tag130o = $sxe->xpath("//record/datafield[@tag='130']/subfield[@code='o']");
$tag245a = $sxe->xpath("//record/datafield[@tag='245']/subfield[@code='a']");
$tag245b = $sxe->xpath("//record/datafield[@tag='245']/subfield[@code='b']");
$tag245c = $sxe->xpath("//record/datafield[@tag='245']/subfield[@code='c']");
$tag250a = $sxe->xpath("//record/datafield[@tag='250']/subfield[@code='a']");
$tag250b = $sxe->xpath("//record/datafield[@tag='250']/subfield[@code='b']");
$tag260a = $sxe->xpath("//record/datafield[@tag='260']/subfield[@code='a']");
$tag260b = $sxe->xpath("//record/datafield[@tag='260']/subfield[@code='b']");
$tag260c = $sxe->xpath("//record/datafield[@tag='260']/subfield[@code='c']");
$tag300a = $sxe->xpath("//record/datafield[@tag='300']/subfield[@code='a']");
$tag300b = $sxe->xpath("//record/datafield[@tag='300']/subfield[@code='b']");
$tag300c = $sxe->xpath("//record/datafield[@tag='300']/subfield[@code='c']");
$tag300e = $sxe->xpath("//record/datafield[@tag='300']/subfield[@code='e']");
$tag490a = $sxe->xpath("//record/datafield[@tag='490']/subfield[@code='a']");
$tag490v = $sxe->xpath("//record/datafield[@tag='490']/subfield[@code='v']");
```

```
$tag500a = $sxe->xpath("//record/datafield[@tag='500']/subfield[@code='a']");
$tag650a = $sxe->xpath("//record/datafield[@tag='650']/subfield[@code='a']");
$tag650o = $sxe->xpath("//record/datafield[@tag='650']/subfield[@code='o']");
$tag700a = $sxe->xpath("//record/datafield[@tag='700']/subfield[@code='a']");
$tag700o = $sxe->xpath("//record/datafield[@tag='700']/subfield[@code='o']");
$tag710a = $sxe->xpath("//record/datafield[@tag='710']/subfield[@code='a']");
$tag710o = $sxe->xpath("//record/datafield[@tag='710']/subfield[@code='o']");
$tag887a = $sxe->xpath("//record/datafield[@tag='887']/subfield[@code='a']");

$count = 0; // Contador de registros de cada archivo XML en formato Marc1

// Bucle para la importación de los contenidos de cada campo bibliográfico en la tabla de la
// colección anteriormente creada
foreach ($tag001 as $control) {
$texto = "$tag020a[$count] $tag245a[$count] $tag245b[$count] $tag245c[$count] $tag250b[$count]
$tag260a[$count] $tag260b[$count] $tag260c[$count] $tag500a[$count] $tag650a[$count]
$tag700a[$count] $tag710a[$count]";

mysql_query("INSERT DELAYED INTO $channelname (id, leader, tag001, tag003, tag005,
tag010a, tag017a, tag020a, tag041a, tag050a, tag080a, tag082a, tag100a, tag100o, tag110a,
tag110o, tag111a, tag111o, tag130a, tag130o, tag245a, tag245b, tag245c, tag250a, tag250b,
tag260a, tag260b, tag260c, tag300a, tag300b, tag300c, tag300e, tag490a, tag490v, tag500a,
tag650a, tag650o, tag700a, tag700o, tag710a, tag710o, tag887a, indexer) VALUES (NULL,
'$leader[$count]', '$tag001[$count]', '$tag003[$count]', '$tag005[$count]', '$tag010a[$count]',
'$tag017a[$count]', '$tag020a[$count]', '$tag041a[$count]', '$tag050a[$count]', '$tag080a[$count]',
'$tag082a[$count]',  '$tag100a[$count]', '$tag100o[$count]', '$tag110a[$count]', '$tag110o[$count]',
'$tag111a[$count]', '$tag111o[$count]', '$tag130a[$count]', '$tag130o[$count]',  '$tag245a[$count]',
'$tag245b[$count]', '$tag245c[$count]', '$tag250a[$count]', '$tag250b[$count]', '$tag260a[$count]',
'$tag260b[$count]', '$tag260c[$count]', '$tag300a[$count]', '$tag300b[$count]', '$tag300c[$count]',
'$tag300e[$count]', '$tag490a[$count]', '$tag490v[$count]', '$tag500a[$count]', '$tag650a[$count]',
'$tag650o[$count]', '$tag700a[$count]', '$tag700o[$count]', '$tag710a[$count]', '$tag710o[$count]',
'$indexer', '$indexer')");

echo " + ";
$contadorB++; // Incremento en el contador de registros OPML
$count++; // Incremento en el contador de registros del archivo XML en formato Marc1
}

echo "</div><br/>";
$ii = $i+1; // Incremento para el contador de atributos xmlUrl de OPML
}
break;
}
```

Archivo importar-marc2.php

Contiene la función y parser especializados en el análisis e importación de canales de sindicación en MARC-XML extendido. Su funcionamiento es idéntico al definido para Atom, destacando que contiene la fórmula XPath de acceso a entidades más compleja, puesto que combina dos elementos con prefijo de namespace y la selección por atributo.

Título: Importación MARC-XML Extendido del archivo importar-marc2.php
Referencia: tabla108

```
echo "<h1>LECTOR MARC2</h1><br/>";

// La variable $channelname construye el nombre de la tabla en la que se importarán los contenidos
// del canal de sindicación en formato Marc2. La denominación comprende un prefijo “col_”, el
// nombre de la colección o del canal de sindicación seleccionado y su sufijo correspondiente al
// formato “_marc2”
$channelname = "col_".$_POST['channelname']."_marc2";

// Incorporación del archivo “config.php” con los datos de conexión a MySQL
include("config.php");

// Selección de la base de datos SYNC sobre la que se opera
mysql_select_db("$database", $con);

// Creación de tabla en la base de datos SYNC para la importación del canal
mysql_query("CREATE TABLE $channelname (
id          INT NOT NULL AUTO_INCREMENT, PRIMARY KEY(id),
tag001      TEXT CHARACTER SET utf8,
tag002      TEXT CHARACTER SET utf8,
tag017      TEXT CHARACTER SET utf8,
tag020      TEXT CHARACTER SET utf8,
tag022      TEXT CHARACTER SET utf8,
tag035      TEXT CHARACTER SET utf8,
tag041      TEXT CHARACTER SET utf8,
tag043      TEXT CHARACTER SET utf8,
tag080      TEXT CHARACTER SET utf8,
tag100      TEXT CHARACTER SET utf8,
tag245      TEXT CHARACTER SET utf8,
tag250a     TEXT CHARACTER SET utf8,
tag250b     TEXT CHARACTER SET utf8,
tag260a     TEXT CHARACTER SET utf8,
tag260b     TEXT CHARACTER SET utf8,
tag260c     TEXT CHARACTER SET utf8,
tag300      TEXT CHARACTER SET utf8,
tag310      TEXT CHARACTER SET utf8,
tag490a     TEXT CHARACTER SET utf8,
tag490v     TEXT CHARACTER SET utf8,
tag500      TEXT CHARACTER SET utf8,
tag654      TEXT CHARACTER SET utf8,
FULLTEXT(tag020, tag022, tag100, tag245, tag250b, tag260a, tag260b, tag260c, tag490a,
tag654))CHARACTER SET utf8",$con);

// El parser que se muestra a continuación está diseñado para trabajar con canales de sindicación
// fragmentados en múltiples archivos XML y agrupados por medio de OPML, por ese motive se
// utiliza el función switch para distinguirlo de los casos en los que los canales de sindicación son
// simples
switch($_POST['fragmentado']) {
case si:

// La variable $feed contiene la URL del canal de sindicación en formato OPML que enlaza todos
// los archivos XML de la colección bibliográfica en formato Marc2
$feed = $_POST['channelfeed'];
```

```
// La variable $xml carga el parser simple que actúa sobre el archivo OPML para extraer las URL de
// los archivos Marc2 que constituyen la colección. Para este proceso se selecciona mediante Xpath
// el atributo xmlUrl del elemento “outline”
$xml = simplexml_load_file($feed);

$ii=0; // Contador para la automatización de la extracción de las URL de los atributos
$contadorB=0; // Contador de registros del archivo OPML

// Bucle para selección de todos los elementos “outline”
for($i=0; $xml->body->outline[$i]; $i++) {

// La variable $outpath contiene la instrucción XPath para seleccionar el atributo xmlUrl que
// contiene la URL de cada archivo XML que compone la colección bibliográfica sindicada
// en formato Marc2
$outpath = $xml->body->outline[$i]->xpath('//@xmlUrl');

// La variable $feed1 toma el valor de la URL de cada archivo XML que compone la colección
// bibliográfica
$feed1 = $outpath[$ii];

// Impresión de pantalla del número de archivo y el registro de inicio
echo "Archivo: $i // Registro de Inicio: $contadorB<br/>";

echo "<div class='detalle'>";

// La variable $sxe inicia el parser del formato Marc2
$sxe = new SimpleXMLElement($feed1, NULL, TRUE);
$tag001 = $sxe->xpath("//marc:record/marc:controlfield[@tag='001']");
$tag002 = $sxe->xpath("//marc:record/marc:controlfield[@tag='003']");
$tag017 = $sxe->xpath("//marc:record/marc:datafield[@tag='017']/marc:subfield[@code='a']");
$tag020 = $sxe->xpath("//marc:record/marc:datafield[@tag='020']/marc:subfield[@code='a']");
$tag022 = $sxe->xpath("//marc:record/marc:datafield[@tag='022']/marc:subfield[@code='a']");
$tag035 = $sxe->xpath("//marc:record/marc:datafield[@tag='035']/marc:subfield[@code='a']");
$tag041 = $sxe->xpath("//marc:record/marc:datafield[@tag='041']/marc:subfield[@code='a']");
$tag043 = $sxe->xpath("//marc:record/marc:datafield[@tag='043']/marc:subfield[@code='c']");
$tag080 = $sxe->xpath("//marc:record/marc:datafield[@tag='080']/marc:subfield[@code='a']");
$tag100 = $sxe->xpath("//marc:record/marc:datafield[@tag='100']/marc:subfield[@code='a']");
$tag245 = $sxe->xpath("//marc:record/marc:datafield[@tag='245']/marc:subfield[@code='a']");
$tag250a = $sxe->xpath("//marc:record/marc:datafield[@tag='250']/marc:subfield[@code='a']");
$tag250b = $sxe->xpath("//marc:record/marc:datafield[@tag='250']/marc:subfield[@code='b']");
$tag260a = $sxe->xpath("//marc:record/marc:datafield[@tag='260']/marc:subfield[@code='a']");
$tag260b = $sxe->xpath("//marc:record/marc:datafield[@tag='260']/marc:subfield[@code='b']");
$tag260c = $sxe->xpath("//marc:record/marc:datafield[@tag='260']/marc:subfield[@code='c']");
$tag300 = $sxe->xpath("//marc:record/marc:datafield[@tag='300']/marc:subfield[@code='a']");
$tag310 = $sxe->xpath("//marc:record/marc:datafield[@tag='310']/marc:subfield[@code='a']");
$tag490a = $sxe->xpath("//marc:record/marc:datafield[@tag='490']/marc:subfield[@code='a']");
$tag490v = $sxe->xpath("//marc:record/marc:datafield[@tag='490']/marc:subfield[@code='v']");
$tag500 = $sxe->xpath("//marc:record/marc:datafield[@tag='500']/marc:subfield[@code='a']");
$tag654 = $sxe->xpath("//marc:record/marc:datafield[@tag='654']/marc:subfield[@code='a']");

$count = 0; // Contador de registros de cada archivo XML en formato Marc2

// Bucle para la importación de los contenidos de cada campo bibliográfico en la tabla de la
// colección anteriormente creada
foreach ($tag001 as $control) {
mysql_query("INSERT DELAYED INTO $channelname (id, tag001, tag002, tag017, tag020, tag022,
tag035, tag041, tag043, tag080, tag100, tag245, tag250a, tag250b, tag260a, tag260b, tag260c,
tag300, tag310, tag490a, tag490v, tag500, tag654) VALUES (NULL, '$control', '$tag002[$count]',
'$tag017[$count]', '$tag020[$count]', '$tag022[$count]', '$tag035[$count]', '$tag041[$count]',
'$tag043[$count]', '$tag080[$count]', '$tag100[$count]', '$tag245[$count]', '$tag250a[$count]',
```

```
'$tag250b[$count]', '$tag260a[$count]', '$tag260b[$count]', '$tag260c[$count]', '$tag300[$count]',
'$tag310[$count]', '$tag490a[$count]', '$tag490v[$count]', '$tag500[$count]', '$tag654[$count]')");

echo " + ";
$contadorB++; // Incremento en el contador de registros OPML
$count++; // Incremento en el contador de registros del archivo XML en formato Marc2
}

echo "</div><br/>";
$ii = $i+1; // Incremento para el contador de atributos xmlUrl de OPML
}

break;
}
```

7.5. Experimentación con SYNC

La plataforma SYNC constituye la primera parte del sistema de sindicación que se ha desarrollado. Su principal función consiste en transformar colecciones bibliográficas de formato CSV a colecciones bibliográficas sindicadas. De tal forma que puedan ser utilizadas por cualquier biblioteca como un canal de sindicación más. Teniendo en cuenta este objetivo principal, se ha preparado un plan de experimentación, que no sólo trata de resolver las preguntas planteadas relativas al uso de la sindicación bibliográfica, sino que también aporta pruebas medibles y cuantificables de cada proceso llevado a cabo por el sistema.

Las pruebas efectuadas en la plataforma SYNC son los presentados en la siguiente *tabla109.*

Título: Pruebas efectuadas en la plataforma SYNC
Referencia: tabla109

Pruebas de la plataforma SYNC	Descripción
Tiempos del proceso de conversión de CSV a XML Raw.	El tiempo de conversión implica el tiempo dedicado a la depuración de la colección bibliográfica CSV y su conversión a un formato XML Raw.
Tiempos del proceso de transferencia de XML Raw a MySQL	Los tiempos del proceso de transferencia implican el tiempo de lectura y el tiempo de inserción de las colecciones bibliográficas de XML Raw a la base de datos.
Tiempos de exportación de MySQL a los distintos formatos de sindicación, Atom, RSS1.0, RSS2.0, MARC-XML abreviado y extendido.	Los tiempos de exportación determinan el proceso de creación de canales de sindicación para las colecciones bibliográficas transferidas anteriormente.
Tiempos de importación de los formatos de sindicación Atom, RSS1.0, RSS2.0, MARC-XML abreviado y extendido a MySQL.	Los tiempos de importación implican el tiempo de lectura del canal de sindicación por parte del parser y por otro lado el tiempo de inserción de los contenidos analizados.

Para poder medir los tiempos de ejecución del sistema para los procesos de conversión, transferencia, exportación e importación, se ha utilizado un método de medición de tiempos basado en la función cronómetro de PHP *microtime()*, elegida precisamente por su mínima inferencia en memoria y debido a su precisión en micro-segundos. Esta función ha permitido cronometrar el tiempo de ejecución de los script de exportación y generación de las

colecciones bibliográficas sindicadas, pero también la capacidad de los parser para leer tales colecciones e importarlas a la base de datos.

Título: Método microtime() para la medición de tiempos de ejecución
Referencia: tabla110

```
// La variable de sesión permite conocer el momento exacto en el que se carga la página en el
// navegador del usuario. La función microtime permite iniciar el cronómetro para contabilizar todo
// el proceso.
$_SESSION['inicio']=microtime(true);

// La variable $tmplecturafin se ubica en la zona del código en la que se finalice por completo el
// proceso que se desea medir
$tmplecturafin = (microtime(true) - $_SESSION['inicio']);
```

Por otra parte existen otros condicionantes de la prueba, que han de ser tenidos en cuenta. La plataforma SYNC ha sido probada en un equipo portátil Intel Core 2 Duo T7500 a 2,20Ghz con 2GB de memoria RAM con un espacio dedicado para la prueba de 60GB. El sistema operativo utilizado fue Windows Vista, y la distribución WAMP (Windows Apache MySQL PHP) empleada fue AppServ. También fue necesario configurar el archivo php.ini, para ampliar el tiempo de ejecución de los sripts en PHP. Tales parámetros son el tiempo máximo de ejecución, el tiempo máximo de lectura de archivos y el límite de memoria.

Título: Configuración del archivo php.ini
Referencia: tabla111

```
max_execution_time = 4000 ; Maximum execution time of each script, in seconds
max_input_time = 4000 ; Maximum amount of time each script may spend parsing request data
memory_limit = 128M ; Maximum amount of memory a script may consume (8MB)
```

7.5.1. Colecciones bibliográficas de prueba

Para llevar a cabo las pruebas con la plataforma SYNC es necesario crear una serie de colecciones bibliográficas en formato CSV. Se ha elegido este formato por considerarse uno de los más sencillos y extendidos tanto en bases de datos como en sistemas de gestión de bibliotecas. CSV significa *comma separated values* o valores separados por comas, por lo que

resulta sencillo distinguir los contenidos de cada campo bibliográfico. Por otro lado, la confección de las colecciones se ha hecho a partir de la consulta sistemática del catálogo OPAC de la Library of Congress (Library of Congress Online Catalog, 2009), de forma tal que uniendo los archivos de exportación CSV que constituyen los resultados de las consultas en el catálogo se conforman colecciones de diversos volúmenes que varían desde los mil registros hasta el millón. Para constituir colecciones lo más realistas posibles, se procuró un procedimiento de recuperación a partir de la lista de términos del tesauro multidisciplinar de la oficina de publicaciones de la Unión Europea (Eurovoc Thesaurus, 2008).

Título: Términos Eurovoc, empleados en la búsqueda OPAC de la Library of Congress.
Referencia: tabla112

Término general	Términos específicos utilizados
04 POLITICS	0406 political framework 0411 political party 0416 electoral procedure and voting 0421 parliament 0426 parliamentary proceedings 0431 politics and public safety 0436 executive power and public service
08 INTERNATIONAL RELATIONS	0806 international affairs 0811 cooperation policy 0816 international balance 0821 defence
12 LAW	1206 sources and branches of the law 1211 civil law 1216 criminal law 1221 justice 1226 organisation of the legal system 1231 international law 1236 rights and freedoms
16 ECONOMICS	1606 economic policy 1611 economic growth 1616 regions and regional policy 1621 economic structure 1626 national accounts 1631 economic analysis
20 TRADE	2006 trade policy 2011 tariff policy 2016 trade 2021 international trade 2026 consumption 2031 marketing

	2036 distributive trades
24 FINANCE	2406 monetary relations 2411 monetary economics 2416 financial institutions and credit 2421 free movement of capital 2426 financing and investment 2431 insurance 2436 public finance and budget policy 2441 budget 2446 taxation 2451 prices
28 SOCIAL QUESTIONS	2806 family 2811 migration 2816 demography and population 2821 social framework 2826 social affairs 2831 culture and religion 2836 social protection 2841 health 2846 construction and town planning
32 EDUCATION AND COMMUNICATIONS	3206 education 3211 teaching 3216 organisation of teaching 3221 documentation 3226 communications 3231 information and information processing 3236 information technology
64 PRODUCTION, TECHNOLOGY AND RESEARCH	6406 production 6411 technology and technical regulations 6416 research and intellectual property
68 INDUSTRY	6806 industrial structures and policy 6811 chemistry 6816 iron, steel and other metal industries 6821 mechanical engineering 6826 electronics and electrical engineering 6831 building and public works 6836 wood industry 6841 leather and textile industries 6846 miscellaneous industries

El resultado de este proceso es la creación de colecciones con tamaños y volúmenes similares a las de las pequeñas bibliotecas (entre 1000 y 25000 volúmenes), pasando por las medianas (entre 25000 y 250000), hasta las grandes bibliotecas (entre 250000 y 1000000 en adelante). En la siguiente tabla se muestra una relación de sus características.

Título: Características de las colecciones bibliográficas en CSV
Referencia: tabla113

Colección	Tamaño en disco (MB)	Nº Registros
1000_reg	0.77	1001
5000_reg	2.68	5002
10000_reg	5.05	10004
25000_reg	13.33	25008
50000_reg	28.34	50036
100000_reg	54.95	100054
250000_reg	144.00	250146
500000_reg	280.49	500309
1000000_reg	561.39	1000039

7.5.2. Prueba de conversión

La prueba de conversión está basada en el programa conversor. El objetivo de la prueba es cronometrar el tiempo que se tarda en convertir una colección bibliográfica de formato CSV a XML Raw, incluyendo su proceso de depuración y tratamiento de caracteres. Los resultados obtenidos se muestran en la siguiente *tabla114*.

Título: Tiempos de conversión de CSV a XML Raw
Referencia: tabla114

Colección	Tiempo de conversión (Segundos)
1000_reg	1.21
5000_reg	4.93
10000_reg	9.45
25000_reg	24.54
50000_reg	50.11
100000_reg	99.13
250000_reg	251.91
500000_reg	504.23
1000000_reg	992.36

7.5.3. Prueba de transferencia XML Raw a MySQL

La prueba de transferencia XML Raw a MySQL tiene como objetivo contabilizar el tiempo de que se tarda en transmitir una colección bibliográfica de un archivo XML sin formato a la base de datos SYNC en MySQL. En este proceso intervienen diversos factores como el método de introducción de datos en MySQL, concretamente mediante la función *insert delayed into* para cargar en búfer de memoria aquellos registros bibliográficos que se agolpan durante el proceso de inserción. Por otro lado el factor del parser empleado, que es de tipo *simplexml_load_file()*. También el tamaño de las colecciones y el número de registros por archivo que la constituyen (1 archivo XML por cada 1000 registros en la colección) configuran factores reseñables para la obtención de los resultados consultables en la siguiente *tabla115*.

Título: Tiempos de transferencia de XML Raw a MySQL
Referencia: tabla115

Colección	Tiempo de transferencia (Segundos)
1000_reg	4.4
5000_reg	17.44
10000_reg	32.05
25000_reg	79,57
50000_reg	159.38
100000_reg	336.34
250000_reg	746.33
500000_reg	1994.86
1000000_reg	4500.13

7.5.4. Prueba de exportación de colecciones a formatos de sindicación

La prueba de exportación de colecciones se ha efectuado una vez han sido transferidas las colecciones bibliográficas de XML Raw a las tablas correspondientes en la base de datos SYNC de MySQL. Para ello se emplea el programa de exportación diseñado para generar colecciones bibliográficas sindicadas en diversos formatos, concretamente Atom, RSS1.0 RDF, RSS2.0, MARC-XML abreviado y extendido. En este proceso se ha contabilizado el tiempo que se tarda en crear un canal de sindicación con los formatos citados y las colecciones transferidas.

Dentro de las posibilidades de cada formato de sindicación, se han escogido todas aquellas etiquetas que, siendo las habituales en la práctica, sean útiles para describir registros bibliográficos textuales (no multimedia), evitando en la medida de lo posible pérdidas de información. En el caso concreto de los formatos MARC, se ha considerado un corpus bibliográfico formado mayoritariamente por monografías y se ha empleado la Clasificación Decimal Dewey, dado que la colección fuente procede de la Library of Congress.

La estructura elegida para el formato ATOM (IETF ATOMPUB WORKING GROUP, 2005) es la siguiente:

Título: Estructura del registro bibliográfico en formato ATOM
Referencia: tabla116

<entry>

<id>Número de identificación**</id>**
<title>Área de título**</title>**
<author><name>Área de mención de responsabilidad**</name></author>**
<updated>Fecha de actualización del registro bibliográfico**</updated>**
<content>Registro bibliográfico completo**</content>**
<link rel='alternate' href= 'URL permanente del registro bibliográfico**'/>**
<summary>Resumen de contenido**</summary>**
<category term='Temática del documento, mediante palabras clave o clasificaciones**'/>**
<contributor><name>Otras menciones de responsabilidad**</name></contributor>**
<published>Área de publicación**</published>**
<source>URL del canal origen del registro**</source>**
<rights>Derechos sobre el registro bibliográfico**</rights>**

</entry>

Las etiquetas escogidas para configurar el canal de sindicación en formato RSS 1.0 RDF (BEGED DOV, G. et al., 2008), junto con las especificaciones de los módulos incluidos (BEGED DOV, G. et al., 2001), (BEGED DOV, G. et al., 2000), (BEGED DOV, G. et al., 2000), (BEGED DOV, G. et al., 2002), (PRISM: Publishing Requirements for Industry Standard Metadata [version 1.2], 2004) son las que se muestran en la *tabla117*.

Título: Estructura del registro bibliográfico en formato RSS 1.0
Referencia: tabla117

<item>
<dc:title>Área de título**</dc:title>**
<dc:creator>Área de mención de responsabilidad**</dc:creator>**
<dc:contributor>Otras menciones de responsabilidad**</dc:contributor>**
<dc:publisher>Área de publicación**</dc:publisher>**
<dc:date>Fecha de publicación**</dc:date>**
<dc:type>Tipo de contenido (texto, imagen, sonido) y género (conforme a un vocabulario previo, por ejemplo: monografías, publicaciones periódicas)**</dc:type>**
<dc:format>Denominación del formato del documento original, tamaño o duración**</dc:format>**
<dc:identifier>Identificador unívoco del documento o URL permanente del registro bibliográfico **</dc:identifier>**
<dc:subject>Temática del documento, mediante palabras clave o clasificaciones**</dc:subject>**
<dc:source>URL del canal origen del registro**</dc:source>**
<dc:language>Idioma del documento**</dc:language>**
<dc:relation>Contenidos relacionados (por ejemplo: colección a la que pertenece el documento, otras fuentes y recursos relacionados)**</dc:relation>**
<dc:coverage>Cobertura espacial o temporal del contenido del documento (por ejemplo siglo XIX, España)**</dc:coverage>**
<dc:rights>Derechos sobre el registro bibliográfico**</dc:rights>**
<prism:publicationName>Denominación de la publicación periódica**</prism:publicationName>**
<prism:edition>Área de edición de la revista**</prism:edition>**
<prism:publisher>Editor de la revista**</prism:publisher>**
<prism:publicationDate>Fecha de publicación del documento**</prism:publicationDate>**
<prism:issn>ISSN**</prism:issn>**
<content:encoded>Registro bibliográfico completo**</content:encoded>**
</item>

A partir de las especificaciones del formato RSS 2.0 (WINER, D., 2009), se ha configurado el registro de la siguiente forma en la *tabla118*.

Título: Estructura del registro bibliográfico en formato RSS 2.0
Referencia: tabla118

<item>
<title>Área de título**</title>**
<author>Área de mención de responsabilidad**</author>**
<pubDate>Fecha de publicación**</pubDate>**
<source>Fuente de procedencia del item**</source>**
<guid>Identificador unívoco del item**</guid>**
<link>URL permanente del registro bibliográfico**</link>**
<description>Resumen de contenido**</description>**
<enclosure>URL, longitud y tipo MIME del documento original**</enclosure>**
<comments>URL de la página de comentarios y valoraciones del documento original**</comments>**
<category>Temática del documento, mediante palabras clave o clasificaciones**</category>**
</item>

Para la configuración de los registros en MARC-XML se ha partido de las especificaciones de la Library of Congress y de la MARC Standards Office (MARC-XML Schema, 2009). En las *tablas 119 y 120* se recogen respectivamente las estructuras de las versiones abreviada y extendida):

Título: Estructura del registro bibliográfico en formato MARC-XML abreviado
Referencia: tabla119

<record>

<controlfield tag='001'>Número de Control Interno (Por ejemplo Registro o código de barras)**</controlfield>**
<controlfield tag='003'>Número de Control para la identificación del documento**</controlfield>**

<datafield tag='017' ind1=" ind2="">
<subfield code='a'>Depósito Legal o Copyright**</subfield>**
</datafield>

<datafield tag='020' ind1=" ind2="">
<subfield code='a'>ISBN**</subfield>**
</datafield>

<datafield tag='022' ind1='0' ind2="">
<subfield code='a'>ISSN**</subfield>**
</datafield>

<datafield tag='035' ind1=" ind2="">
<subfield code='a'>Número de Control del Sistema</subfield>
</datafield>

<datafield tag='041' ind1='0' ind2="">
<subfield code='a'>Código del idioma del documento original**</subfield>**
</datafield>

<datafield tag='043' ind1=" ind2="">
<subfield code='c'>Código geográfico del documento original**</subfield>**
</datafield>

<datafield tag='082' ind1=" ind2="">
<subfield code='a'>Clasificación Decimal Dewey**</subfield>**
</datafield>

<datafield tag='100' ind1='1' ind2="">
<subfield code='a'>Autor personal**</subfield>**
</datafield>

<datafield tag='245' ind1='1' ind2="">
<subfield code='a'>Área de título**</subfield>**
</datafield>

<datafield tag='250' ind1='' ind2=''>
<subfield code='a'>Nº de edición</subfield>
<subfield code='b'>Mención de edición</subfield>
</datafield>

<datafield tag='260' ind1='' ind2=''>
<subfield code='a'>Lugar de publicación</subfield>
<subfield code='b'>Editorial</subfield>
<subfield code='c'>Año de publicación</subfield>
</datafield>

<datafield tag='300' ind1='' ind2=''>
<subfield code='a'>Área de descripción física</subfield>
</datafield>

<datafield tag='310' ind1='' ind2=''>
<subfield code='a'>Periodicidad</subfield>
</datafield>

<datafield tag='490' ind1='0' ind2=''>
<subfield code='a'>Serie o colección</subfield>
<subfield code='v'>Nº de serie o colección</subfield>
</datafield>

<datafield tag='500' ind1='' ind2=''>
<subfield code='a'>Área de notas</subfield>
</datafield>

<datafield tag='654' ind1='0' ind2=''>
<subfield code='a'>Temática del documento, mediante palabras clave o clasificaciones</subfield>
</datafield>

</record>

Título: Estructura del registro bibliográfico en formato MARC-XML extendido
Referencia: tabla120

<marc:record>

<marc:controlfield tag='001'>Número de Control Interno (Por ejemplo Registro o código de barras)</marc:controlfield>
<marc:controlfield tag='003'>Número de Control para la identificación del documento
</marc:controlfield>

<marc:datafield tag='017' ind1='' ind2=''>
<marc:subfield code='a'>Depósito Legal o Copyright</marc:subfield>
</marc:datafield>

<marc:datafield tag='020' ind1='' ind2=''>
<marc:subfield code='a'>ISBN</marc:subfield>
</marc:datafield>

<marc:datafield tag='022' ind1='0' ind2=''>
<marc:subfield code='a'>ISSN</marc:subfield>
</marc:datafield>

<marc:datafield tag='035' ind1='' ind2=''>
<marc:subfield code='a'>Número de Control del Sistema</marc:subfield>
</marc:datafield>

<marc:datafield tag='041' ind1='0' ind2=''>
<marc:subfield code='a'>Código del idioma del documento original</marc:subfield>
</marc:datafield>

<marc:datafield tag='043' ind1='' ind2=''>
<marc:subfield code='c'>Código geográfico del documento original</marc:subfield>
</marc:datafield>

<marc:datafield tag='082' ind1='' ind2=''>
<marc:subfield code='a'>Clasificación Decimal Dewey</marc:subfield>
</marc:datafield>

<marc:datafield tag='100' ind1='1' ind2=''>
<marc:subfield code='a'>Autor personal</marc:subfield>
</marc:datafield>

<marc:datafield tag='245' ind1='1' ind2=''>
<marc:subfield code='a'>Área de título</marc:subfield>
</marc:datafield>

<marc:datafield tag='250' ind1='' ind2=''>
<marc:subfield code='a'>Nº de edición</marc:subfield>
<marc:subfield code='b'>Mención de edición</marc:subfield>
</marc:datafield>

<marc:datafield tag='260' ind1='' ind2=''>
<marc:subfield code='a'>Lugar de publicación</marc:subfield>
<marc:subfield code='b'>Editorial</marc:subfield>
<marc:subfield code='c'>Año de publicación</marc:subfield>
</marc:datafield>

<marc:datafield tag='300' ind1='' ind2=''>
<marc:subfield code='a'>Área de descripción física</marc:subfield>
</marc:datafield>

<marc:datafield tag='310' ind1='' ind2=''>
<marc:subfield code='a'>Periodicidad</marc:subfield>
</marc:datafield>

<marc:datafield tag='490' ind1='0' ind2=''>
<marc:subfield code='a'>Serie o colección</marc:subfield>
<marc:subfield code='v'>Nº de serie o colección</marc:subfield>
</marc:datafield>

<marc:datafield tag='500' ind1='' ind2=''>
<marc:subfield code='a'>Área de notas</marc:subfield>
</marc:datafield>

<marc:datafield tag='654' ind1='0' ind2=''>
<marc:subfield code='a'>Temática del documento, mediante palabras clave o clasificaciones</marc:subfield>
</marc:datafield>

</marc:record>

Teniendo en cuenta que cada formato de sindicación es capaz de alojar una serie de etiquetas básicas según las especificaciones, el proceso de exportación arroja los siguientes resultados disponibles en la *tabla121*.

Título: Tiempos de creación de canales de sindicación por formato
Referencia: tabla121

Formato	Colección	Tiempo de creación (Segundos)
Atom	1000_reg	0.19
	5000_reg	0.74
	10000_reg	1.32
	25000_reg	3.64
	50000_reg	8.52
	100000_reg	20.90
	250000_reg	90.89
	500000_reg	287.61
	1000000_reg	1032.25
RSS1.0 RDF	1000_reg	0.17
	5000_reg	0.65
	10000_reg	1.65
	25000_reg	4.34
	50000_reg	8.80
	100000_reg	24.59
	250000_reg	100.34
	500000_reg	312.98
	1000000_reg	1068.93
RSS2.0	1000_reg	0.11
	5000_reg	0.94
	10000_reg	1.21
	25000_reg	3.37
	50000_reg	8.13
	100000_reg	20.28
	250000_reg	89.51
	500000_reg	290.36
	1000000_reg	1036.30
MARC-XML abreviado	1000_reg	0.24
	5000_reg	0.81
	10000_reg	1.75
	25000_reg	4.82
	50000_reg	11.78
	100000_reg	26.68
	250000_reg	105.28
	500000_reg	321.08
	1000000_reg	1095.83

MARC-XML extendido	1000_reg	0.20
	5000_reg	0.84
	10000_reg	1.70
	25000_reg	4.50
	50000_reg	10.51
	100000_reg	24.77
	250000_reg	99.03
	500000_reg	326.96
	1000000_reg	1091.31

7.5.5. Prueba de importación y lectura de colecciones bibliográficas sindicadas

Una vez exportadas las colecciones bibliográficas, éstas conforman canales de sindicación OPML que agrupan archivos de 1000 registros cada uno, hasta completar la colección correspondiente. Tales archivos se encuentran en los formatos Atom, RSS1.0 RDF, RSS2.0, MARC-XML abreviado y extendido. Tomando estas consideraciones como punto de partida para la prueba de importación, se trata de simular el proceso de difusión de los datos de cada colección, presentes en cada canal de sindicación, desde un servidor hasta el equipo cliente. Durante esta prueba se contabiliza el proceso que comprende los siguientes puntos:

- Detectar el tipo de formato del canal a través de su extensión.
- Crear una tabla de datos en MySQL cuya estructura se amolde a la del formato de sindicación correspondiente.
- Leer secuencialmente cada registro bibliográfico. El tiempo ocupado en este proceso representa el tiempo de transferencia entre el equipo servidor y el equipo cliente. Este tiempo de transferencia depende en la práctica de muchos factores, entre los que destaca el ancho de banda de la red, la velocidad de proceso del equipo cliente o la memoria del sistema. En consecuencia, no consideraremos en este estudio el tiempo de transferencia obtenido.
- Insertar cada registro bibliográfico leído (en grupos de mil, por haberse comprobado que es el nivel de agrupación que minimiza los tiempos de inserción) en la tabla de datos de destino.

Los tiempos obtenidos en el proceso de importación de fuentes sindicadas, considerando las distintas colecciones y los diferentes formatos, se resumen en la *tabla122*.

Título: Tiempo de importación
Referencia: tabla122

Formato	Colección	Tiempo de importación (Segundos)
Atom	1000_reg	0,75
	5000_reg	3,10
	10000_reg	5,27
	25000_reg	17,24
	50000_reg	42,35
	100000_reg	86,27
	250000_reg	284,09
	500000_reg	630,23
	1000000_reg	1832,74
RSS1.0 RDF	1000_reg	1,05
	5000_reg	3,45
	10000_reg	6,55
	25000_reg	21,27
	50000_reg	54,73
	100000_reg	113,13
	250000_reg	351,60
	500000_reg	930,99
	1000000_reg	2229,34
RSS2.0	1000_reg	0,45
	5000_reg	3,45
	10000_reg	5,83
	25000_reg	20,21
	50000_reg	50,41
	100000_reg	105,17
	250000_reg	363,52
	500000_reg	794,03
	1000000_reg	2519,13
MARC-XML abreviado	1000_reg	1,68
	5000_reg	8,36
	10000_reg	16,85
	25000_reg	42,63
	50000_reg	92,88
	100000_reg	184,64
	250000_reg	510,92
	500000_reg	1034,99
	1000000_reg	2857,61
MARC-XML extendido	1000_reg	4,05
	5000_reg	8,45
	10000_reg	17,21
	25000_reg	43,11
	50000_reg	92,56
	100000_reg	188,49
	250000_reg	508,32
	500000_reg	1125,76
	1000000_reg	2749,38

7.5.6. Análisis de los resultados de la plataforma SYNC.

Análisis de la adaptabilidad de los formatos a la descripción bibliográfica

El primer análisis claro de la plataforma SYNC constituye el hecho de las capacidades de representación y adaptabilidad de los formatos de sindicación, puesto que no todos son capaces de ofrecer las mismas posibilidades de descripción bibliográfica. Esta distinta adaptabilidad se debe a que poseen estructuras internas dispares, permitiendo o impidiendo la inserción de determinados tipos de información del registro bibliográfico.

Desde el punto de vista documental, todo registro bibliográfico consta de las siguientes áreas de información esenciales: Área de título y mención de responsabilidad, edición, clase de documento, publicación, descripción física, serie y notas (IFLA (INTERNATIONAL FEDERATION OF LIBRARY ASSOCIATIONS AND INSTITUTIONS), 2007). A ellos añadiremos la posibilidad de incluir el registro bibliográfico completo, porque favorece la recuperación de información mediante cualquier aspecto o punto de acceso secundario que no haya sido tenido en cuenta en la estructura original del formato.

En la siguiente tabla se exponen las áreas de descripción que es posible introducir en cada formato atendiendo a las especificaciones particulares de cada uno de ellos:

Título: Adaptabilidad bibliográfica de los formatos de sindicación
Referencia: tabla123

	Atom	RSS1.0 RDF	RSS2.0	MARC-XML abreviado	MARC-XML extendido
Área de título y mención de responsabilidad	X	X	X	X	X
Área de edición				X	X
Área de clase de documento		X		X	X
Área de publicación	X	X		X	X

Área de descripción física		X		X	X
Área de serie				X	X
Área de notas		X		X	X
Registro bibliográfico completo	X	X	X	X	X

De la tabla se desprende que el formato más adecuado para la difusión de registros bibliográficos es la familia de formatos MARC, pues permite incluir todas las áreas básicas de descripción bibliográfica con el nivel de exhaustividad deseado. También destaca la posibilidad de exhaustividad completa en la descripción bibliográfica, permitiendo la introducción de la ficha completa del registro bibliográfico mediante la etiqueta 887. Su presencia favorece la visualización del registro en su integridad en caso necesario y el acceso a cualquier dato en un solo campo durante los procesos de recuperación de información.

De los formatos de sindicación propiamente dichos, RSS1.0 RDF es el que mejor se adapta a un registro bibliográfico tanto monográfico como seriado, debido a la posibilidad de incluir módulos de Dublin Core y PRISM. Aún así, se detectan deficiencias importantes, como la imposibilidad de incluir las áreas de edición y serie. Estos problemas pueden ser solventados gracias a la posibilidad de incluir un campo *content:encoded* para introducir el registro bibliográfico completo en el que añadir dichos datos.

A un nivel similar de adaptabilidad se sitúan Atom y RSS2.0. Ambos presentan una capacidad baja de representación bibliográfica, básicamente limitada al área de título y mención de responsabilidad. RSS2.0 presenta la desventaja añadida de no incluir el registro bibliográfico completo. Este problema puede ser solucionado, aunque de manera poco ortodoxa, agregando módulos diseñados originalmente para el formato RSS1.0 (Dublin Core y PRISM) previa introducción de los *namespace* correspondientes según las propiedades de extensibilidad. Esta solución originaría un formato híbrido que dista bastante del original, convirtiéndose en un

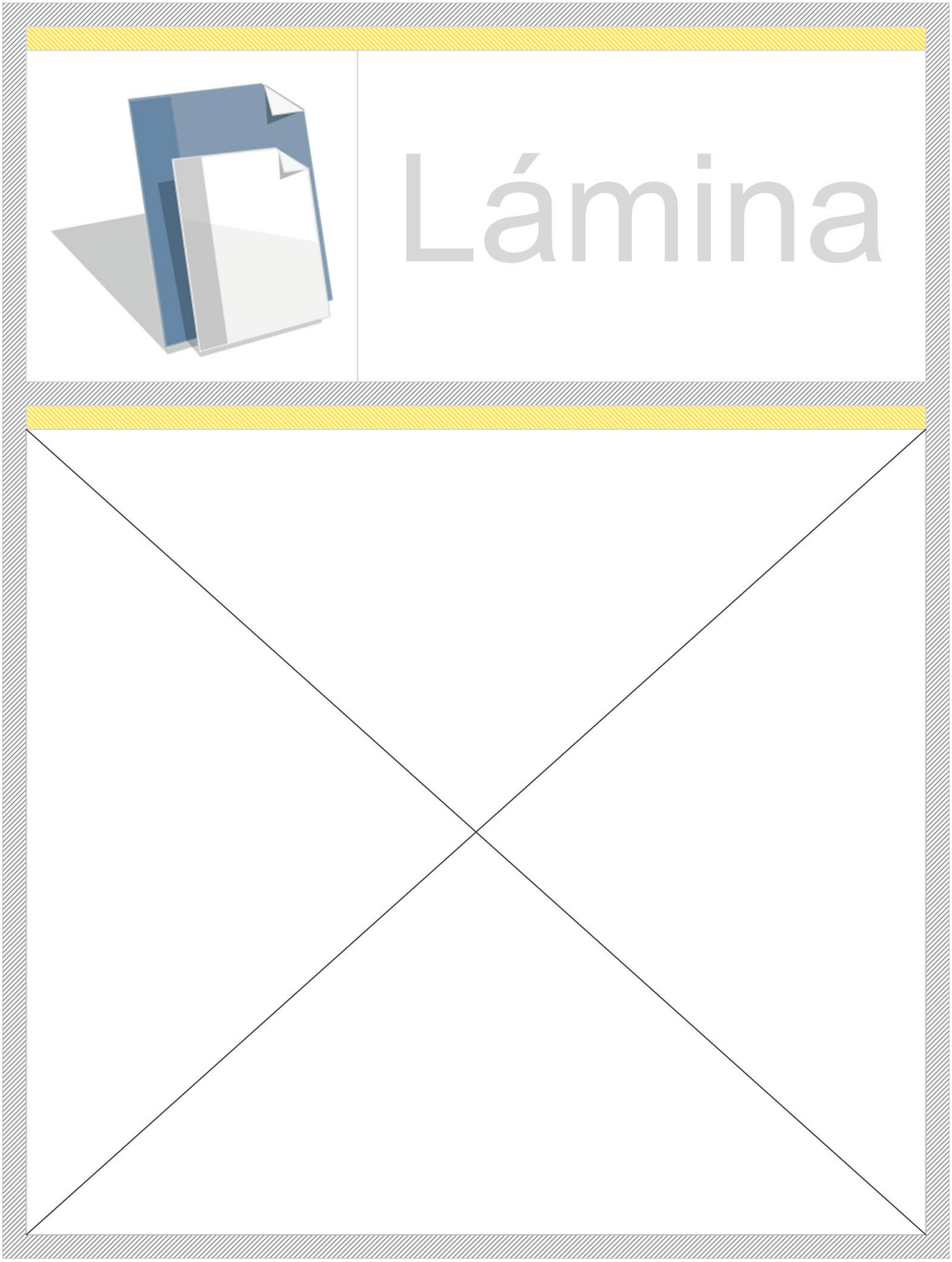
sucedáneo de RSS1.0 RDF, lo cual no permitiría establecer comparación alguna entre formatos, tal y como se explica en el capítulo 6, apartado 4 referente a la configuración básica de los formatos de sindicación de contenidos.

Evolución del tamaño de las colecciones

La evolución del tamaño de las colecciones es uno de los hechos que queda patente en las pruebas efectuadas. Bajo este punto de análisis, hay que distinguir el tamaño de las colecciones originales en formato CSV con respecto a XML Raw y por otra parte, los tamaños de las colecciones en los distintos formatos de sindicación.

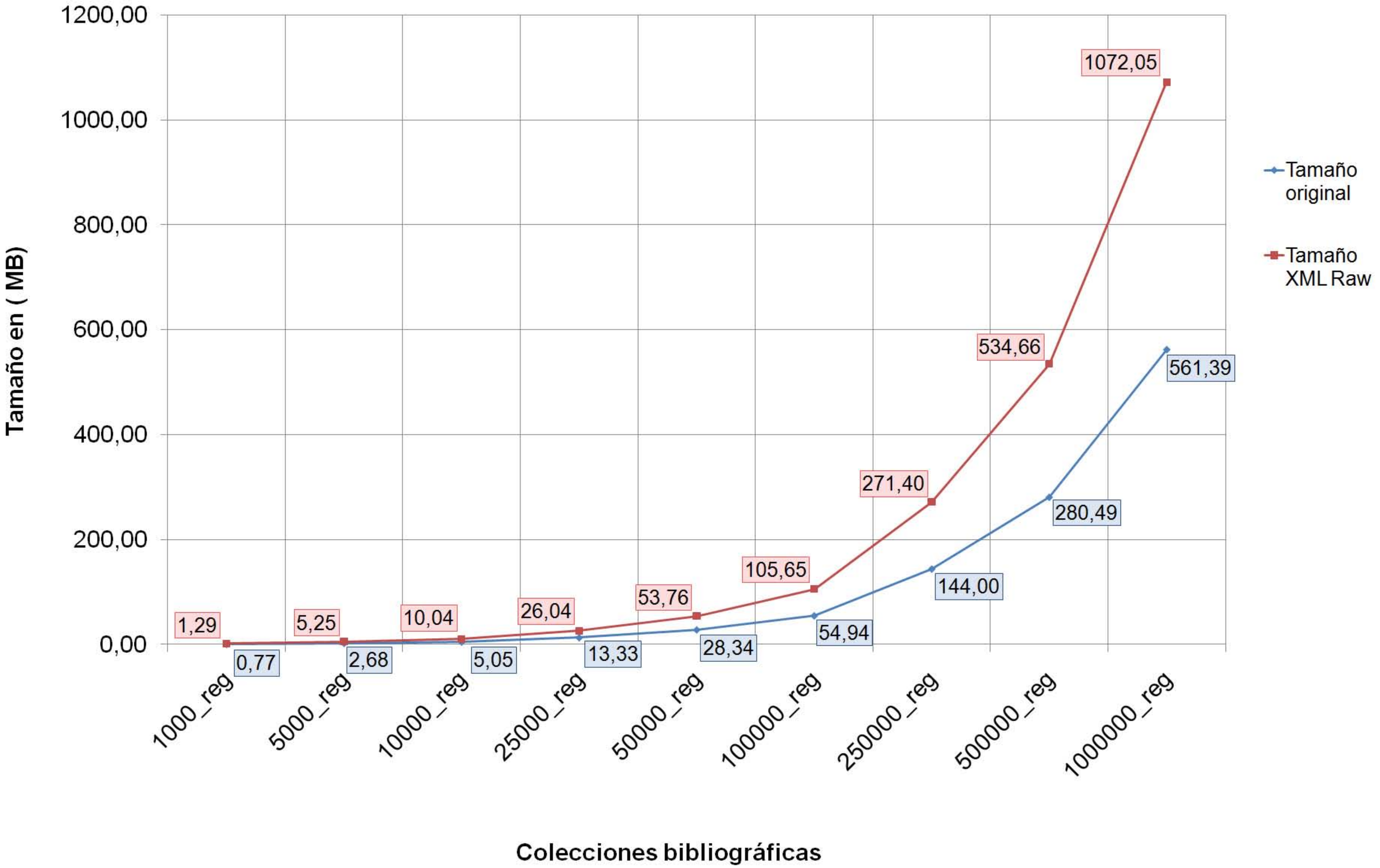
Se detecta que el proceso de conversión de las colecciones a formato XML Raw, aumenta el tamaño de las colecciones con respecto a los archivos originales en CSV. Ello puede comprobarse claramente, a partir del rango de colecciones con más de 50.000 registros. El ejemplo más significativo se encuentra al abordar un millón de registros en la que la diferencia entre la colección XML Raw y CSV, supera los 500 MB.

Título: Estadística evolución del tamaño de las colecciones en XML Raw
Referencia: figura32



Evolución del tamaño de las colecciones en XML Raw

Referencia: figura32



Esto es debido a la forma en la que se almacenan los datos en CSV y XML Raw. Si bien CSV tan solo requiere de una separación por comas de los valores correspondientes a cada campo, XML Raw requiere de etiquetas de apertura y cierre para definirlos, así como de etiquetas que identifiquen el registro completo. Para corroborar este análisis se toman dos registros en CSV y XML Raw, con el mismo número de campos y con el mismo contenido y se contabiliza el número de caracteres que ocupa uno y otro. Tal y como se muestra en la siguiente tabla, se demuestra que CSV ocupa 265 caracteres frente a los 627 de XML Raw, lo que significa que en el espacio de un registro XML Raw, cabrían dos en CSV.

Título: Modelo de registro en CSV y su comparación con XML Raw
Referencia: tabla124

Ejemplo de registro en CSV: 265 caracteres.

"a0","a1","a2","a3","a4","a5","a6","a7","a8","a9","a10","a11","a12","a13","a14","a15","a16","a17","a18","a19","a20","a21","a22","a23","a24","a25","a26","a27","a28","a29","a30","a31","a32","a33","a34","a35","a36","a37","a38","a39","a40","a41","a42","a43","a44","a45"

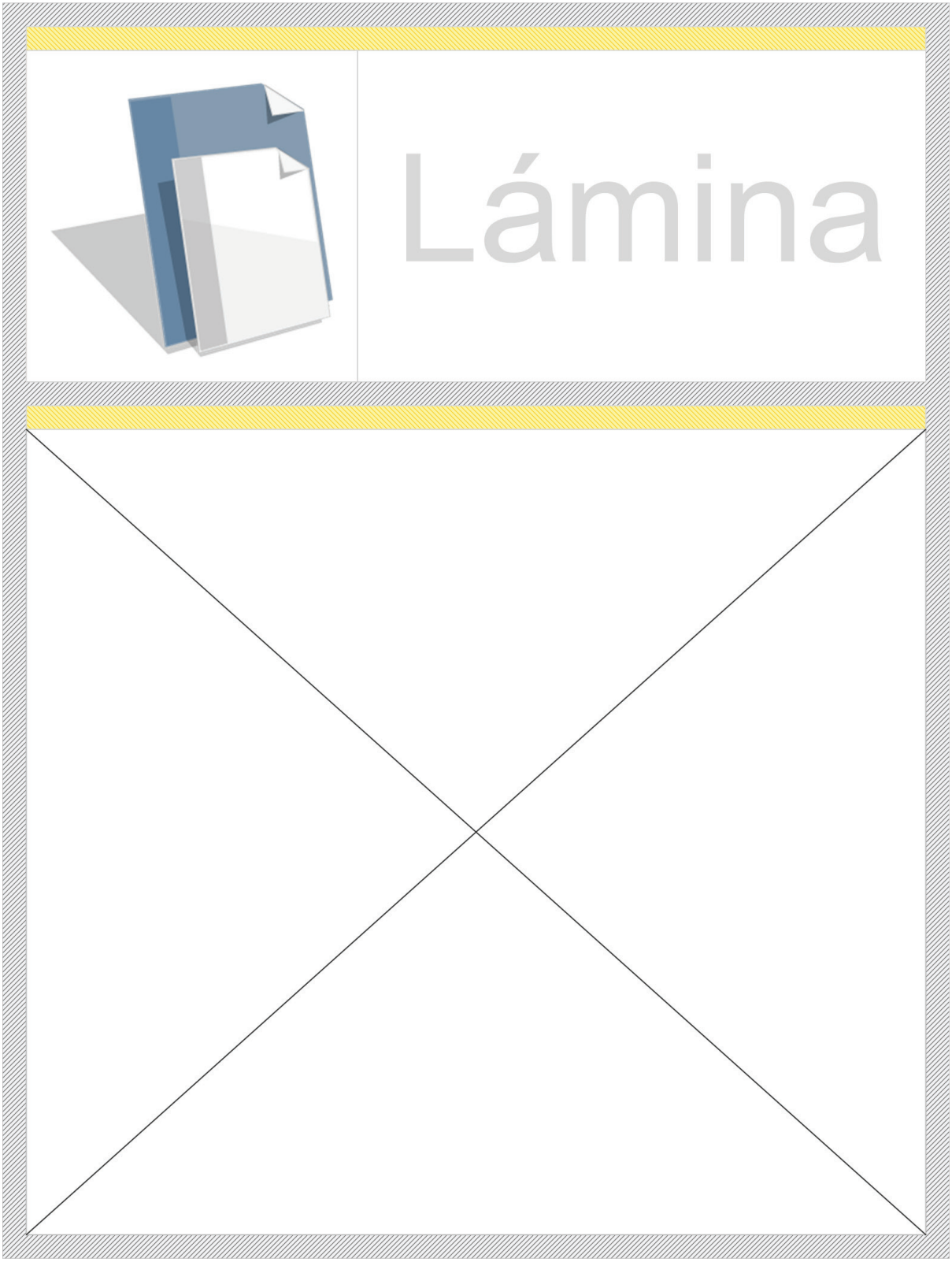
Ejemplo de registro en XML Raw: 627 caracteres.

<item>
<a0>a0</a0><a1>a1</a1><a2>a2</a2><a3>a3</a3><a4>a4</a4><a5>a5</a5><a6>a6</a6><a7>a7</a7><a8>a8</a8><a9>a9</a9><a10>a10</a10><a11>a11</a11><a12>a12</a12><a13>a13</a13><a14>a14</a14><a15>a15</a15><a16>a16</a16><a17>a17</a17><a18>a18</a18><a19>a19</a19><a20>a20</a20><a21>a21</a21><a22>a22</a22><a23>a23</a23><a24>a24</a24><a25>a25</a25><a26>a26</a26><a27>a27</a27><a28>a28</a28><a29>a29</a29><a30>a30</a30><a31>a31</a31><a32>a32</a32><a33>a33</a33><a34>a34</a34><a35>a35</a35><a36>a36</a36><a37>a37</a37><a38>a38</a38><a39>a39</a39><a40>a40</a40><a41>a41</a41><a42>a42</a42><a43>a43</a43><a44>a44</a44><a45>a45</a45>
</item>

A pesar de que CSV resulta un formato más óptimo en cuanto a la gestión del espacio y el tamaño se refiere, también tiene un punto débil, que es precisamente la imposibilidad de definir el nombre de los campos, solamente posible mediante el empleo de las etiquetas de marcado XML.

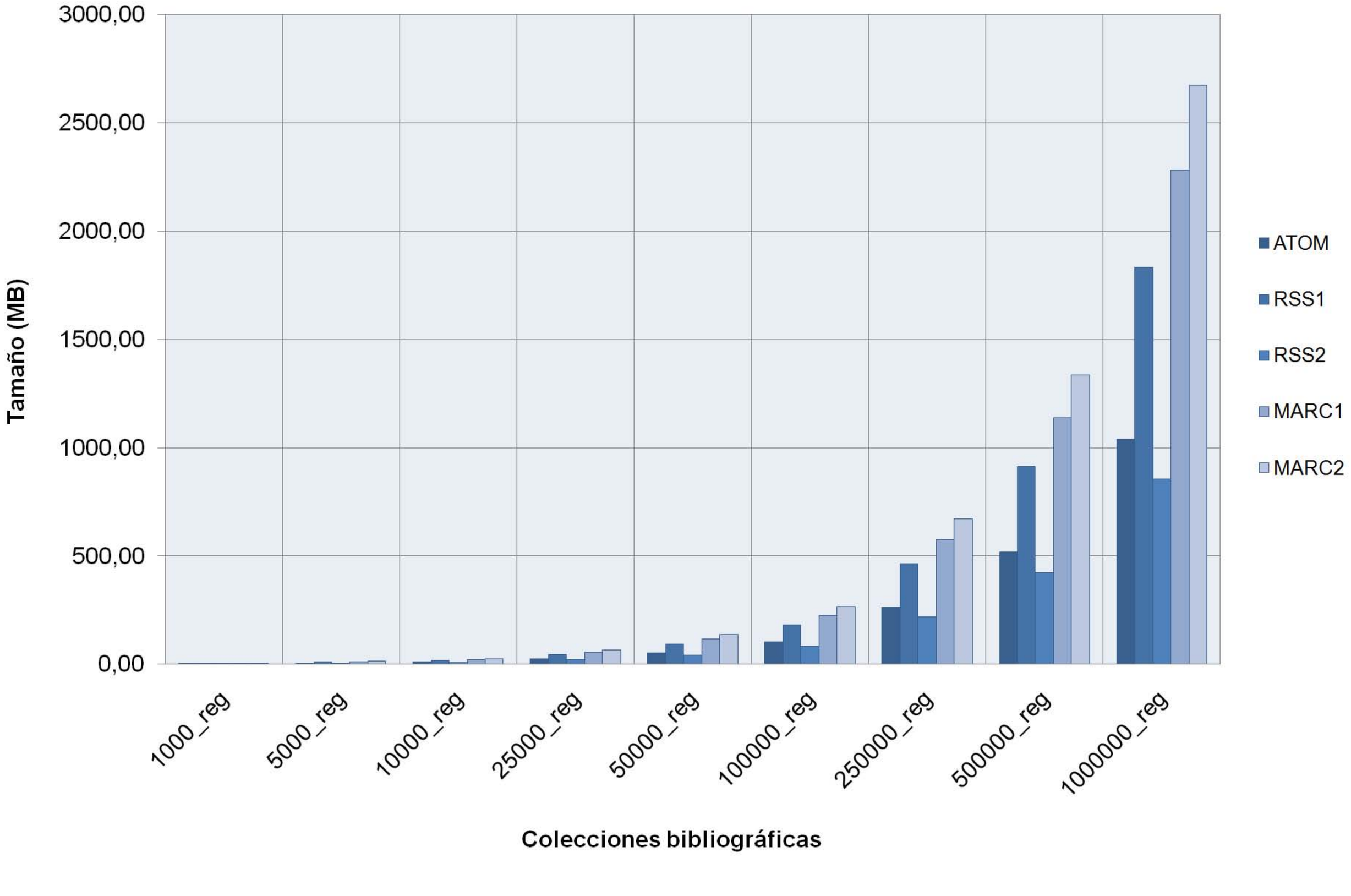
A parte de este análisis, resulta también importante comprobar cuál es la variación de tamaños entre las colecciones bibliográficas sindicadas, tal y como muestra la siguiente *figura33*.

Título: Estadística del tamaño de las colecciones bibliográficas sindicadas
Referencia: figura33



Tamaño de las colecciones bibliográficas sindicadas

Referencia: figura33

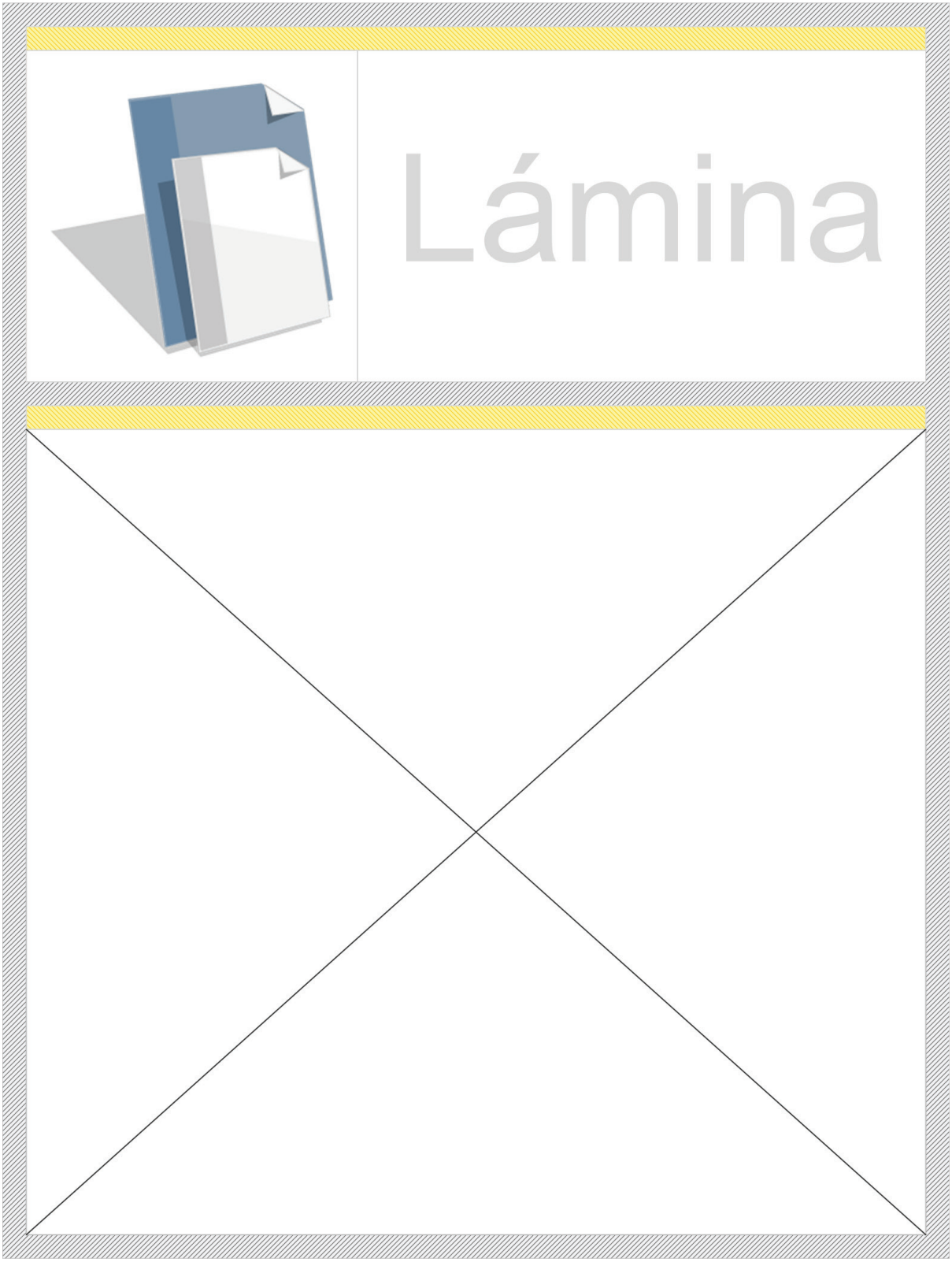


De los datos obtenidos, se deduce que las diferencias entre las estructuras de los distintos formatos influyen en el tamaño de los canales sindicados. Lógicamente cuanto más extensa y compleja es la estructura, más ocupa el canal sindicado correspondiente. Los formatos oscilan desde el más sencillo, el RSS2.0, hasta el más complejo, el MARC-XML extendido, cuyo tamaño es tres veces mayor que el anterior. No obstante, también hay que destacar el hecho de que con colecciones cuyo intervalo sitúa entre los 1000 y los 100.000 registros, apenas resulta significativa la diferencia de tamaños entre formatos, así mismo sucede hasta la colección de 250.000 registros. A partir de este punto en adelante sí se empieza a percibir una diferencia notable entre los formatos más ligeros como Atom y RSS2.0 con respecto a los más pesados RSS1.0 RDF y las dos variedades del formato MARC-XML.

Sin embargo, las diferencias observadas en el tamaño de las colecciones sindicadas no implican unas grandes diferencias en cuanto al tiempo de creación de dichos canales, como se muestra en la *figura33*. Aunque la colección sindicada de un millón de registros bibliográficos en MARC-XML extendido ocupa 2673 MB y en RSS2 ocupa 854 MB, los tiempos de creación de los canales correspondientes son respectivamente de 1091 y 1036 segundos, lo que supone apenas un minuto de diferencia entre ambas. Más aún, conforme disminuye el tamaño de la colección inicial, las diferencias también disminuyen, como puede comprobarse en la *tabla124* de resultados. Por tanto, el formato (a pesar de las diferencias estructurales entre ellas) no tiene una incidencia relevante sobre el tiempo de creación de los canales sindicados correspondientes.

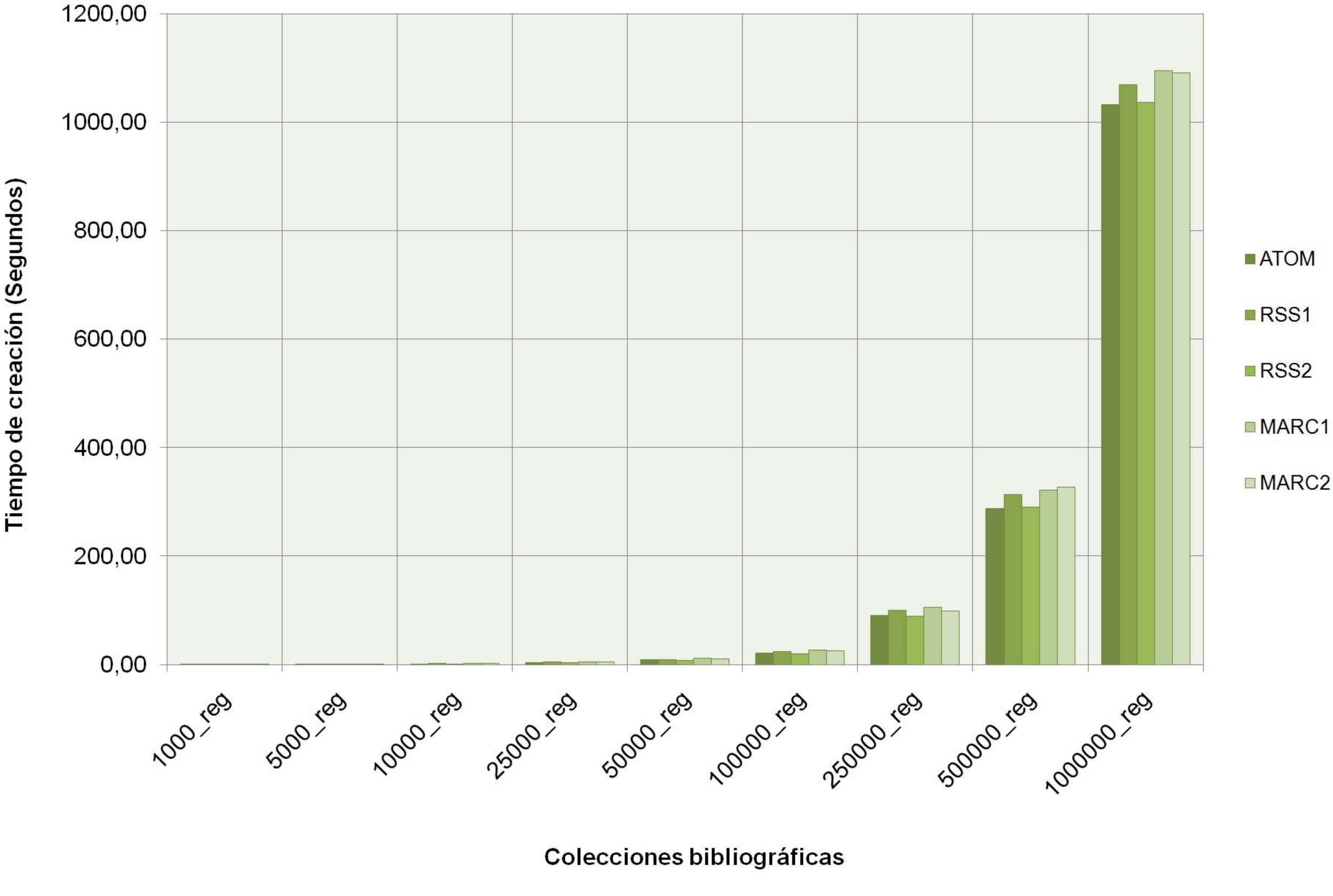
El gráfico del tiempo de exportación, junto a los datos obtenidos, destaca que el proceso de creación de los canales de sindicación bibliográficos es bastante rápido. Un dato revelador de ello es que todas las colecciones bibliográficas hasta 500.000 registros se procesan en menos de 5 minutos con cualquier formato de sindicación. Según decrece el tamaño de la colección se reduce a menos de la mitad el tiempo de ejecución para dicho proceso. Es el caso de la colección de 250.000 registros procesada en todos los formatos en menos de 2 minutos. Finalmente el tiempo de exportación más reducido es el correspondiente a la colección de 1000 registros, procesada entre 0,11 y 0,24 segundos.

Título: Estadística de tiempos de creación de los canales de sindicación
Referencia: figura34



Estadística de tiempos de creación de los canales de sindicación

Referencia: figura34

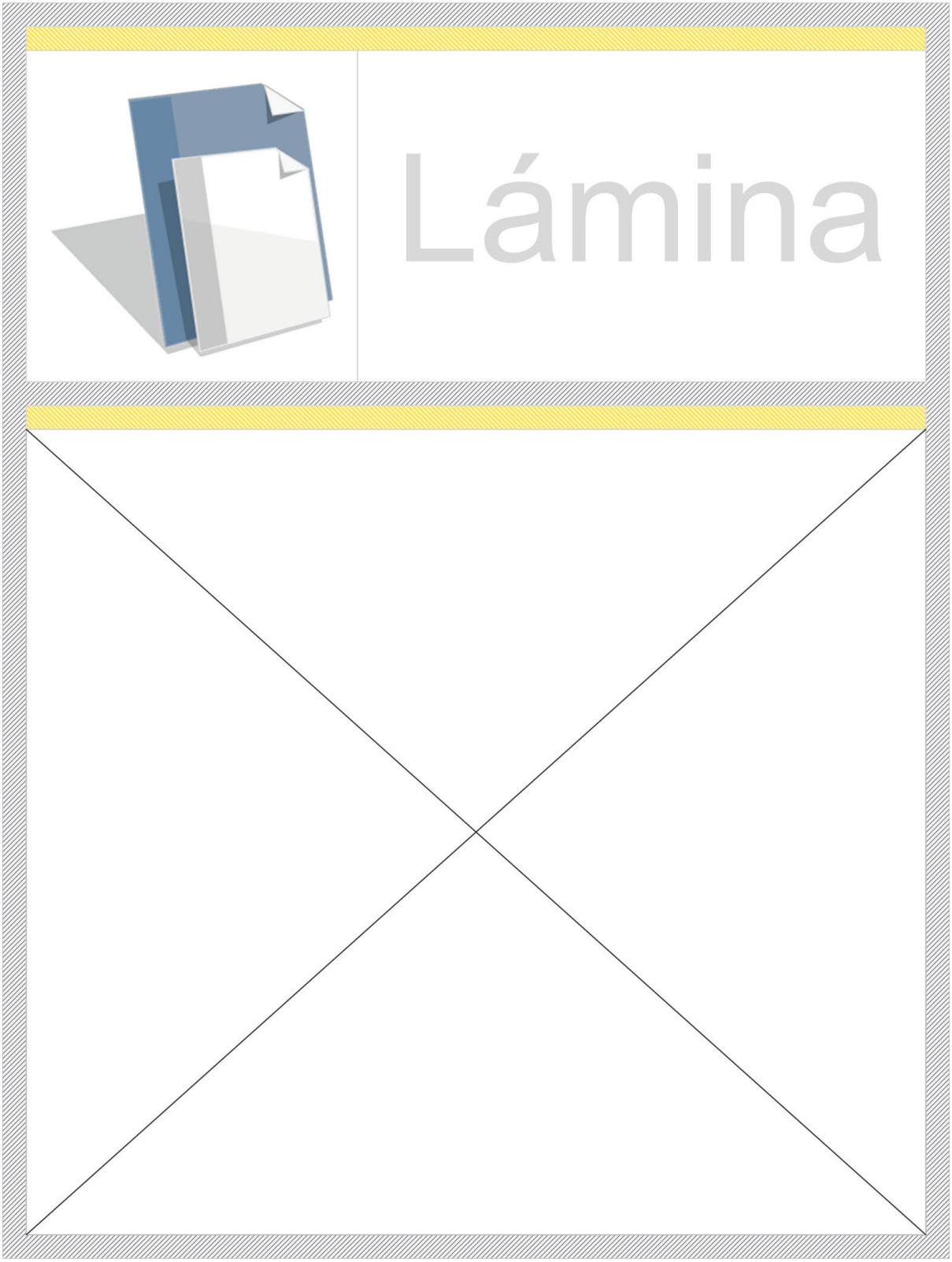


En cuanto a los tiempos de importación (considerando la lectura y el proceso de inserción en la base de datos), los resultados obtenidos en la *tabla122* permiten deducir que la diferencia en el tiempo de inserción entre los distintos formatos es elevada cuando las colecciones originales son grandes, pero disminuye conforme las colecciones son menores. En el caso de las colecciones de un millón de registros, la diferencia máxima obtenida (entre los formatos MARC extendido y el formato ATOM) ronda los 15 minutos. En las colecciones de medio millón de registros, la máxima diferencia obtenida es de aproximadamente 8 minutos entre los mismos formatos. Las colecciones de 250000 registros presentan una diferencia máxima que no llega a los 4 minutos. Esta diferencia máxima es de 1.7 minutos en el caso de 100000 registros y es inferior al minuto (50.53 segundos) en el caso de 50000 registros, disminuyendo hasta alcanzar los 3.3 segundos en el caso de 1000 registros.

Si bien el tiempo de creación de los canales no es relevante en relación al tamaño de las colecciones, sí lo es el tiempo de importación. De los datos analizados se deduce que la importación de datos sindicados es inferior al minuto, independientemente del formato elegido, solo cuando la colección no supera los 25000 registros, presentando unos tiempos considerables para colecciones muy grandes, superiores a los 250000 registros.

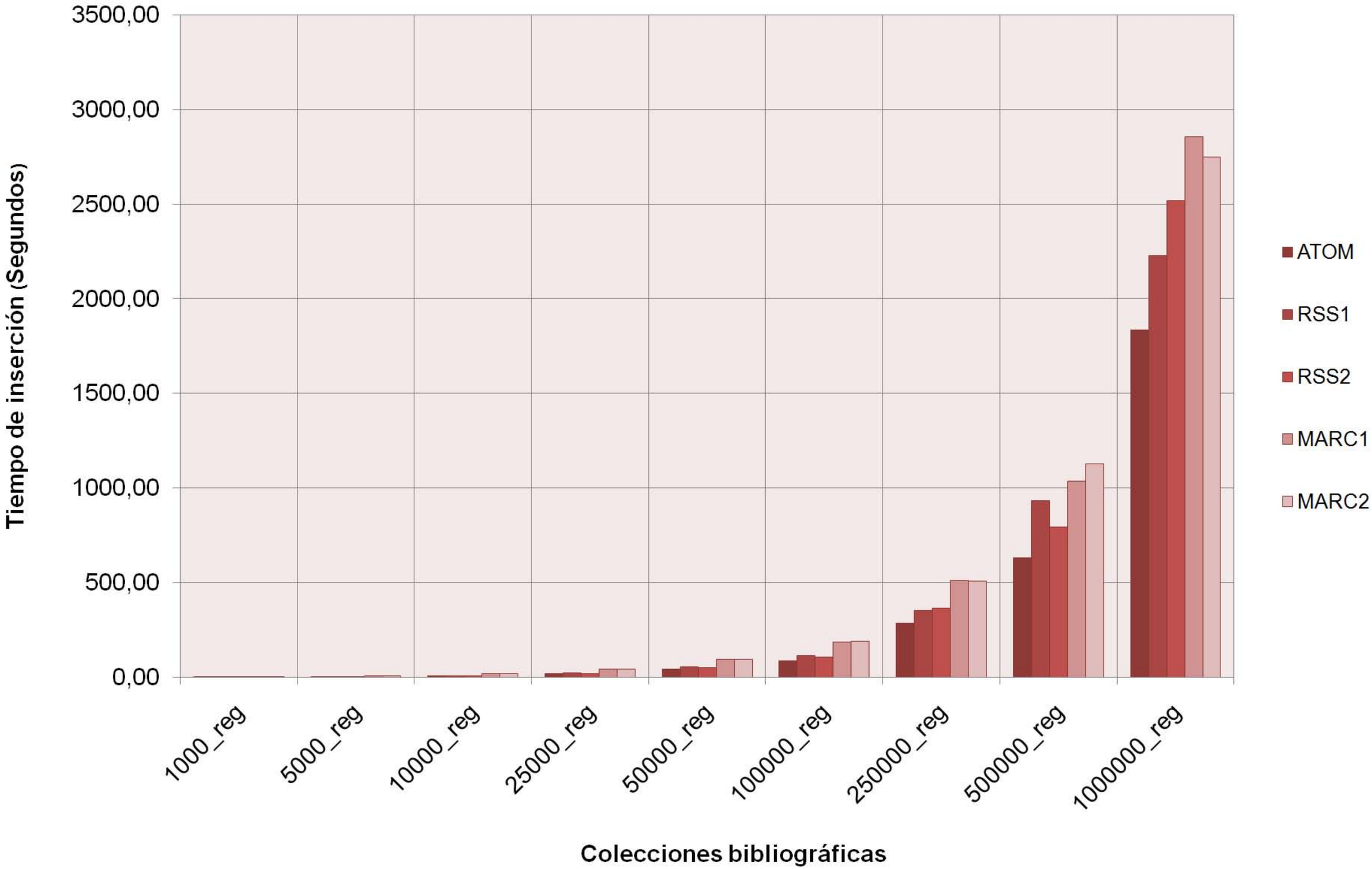
Finalmente también es reseñable la comparación entre el parser *simpleXMLElement()* y *simplexml_load_file()*. El primero ha sido empleado en los procesos de importación de los canales de sindicación y el segundo en el proceso de transferencia de XML Raw. Para efectuar una valoración efectiva entre ambos parser, se ha tomado el tiempo que tarda el parser del programa de transferencia en leer e insertar los contenidos con la base de datos y por otro lado se ha hecho la media de los tiempos de importación de los distintos canales de sindicación, a fin de ser comparados. El resultado ante colecciones de tamaño y volumen similares, destaca que los tiempos de lectura e inserción de datos en la base de datos SYNC son notablemente mejores empleando el parser *simpleXMLElement()* del programa de importación. De hecho en todos los casos la media de importación de los canales de sindicación es siempre la mitad o menos de la mitad del tiempo que necesita el programa de transferencia para hacer una operación similar con archivos en XML Raw. Un claro ejemplo de ello se encuentra en la colección de 500.000 registros que el programa de transferencia tarda más de 30 minutos en procesar frente a los poco más de 10 obtenidos en la media de tiempos del programa de importación para los distintos formatos de sindicación.

Título: Tiempo de inserción de datos
Referencia: figura35

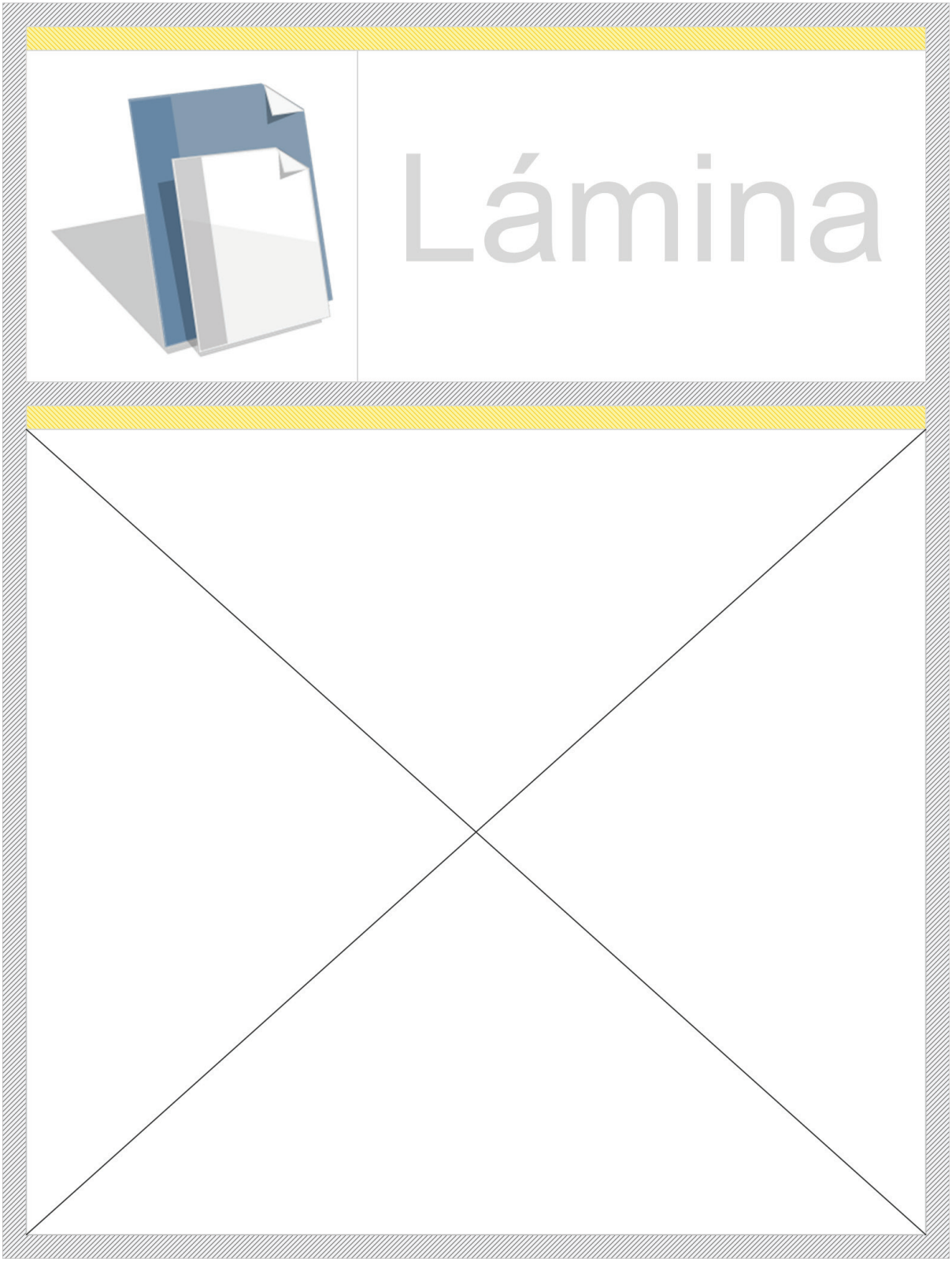


Tiempo de importación de colecciones sindicadas

Referencia: figura35

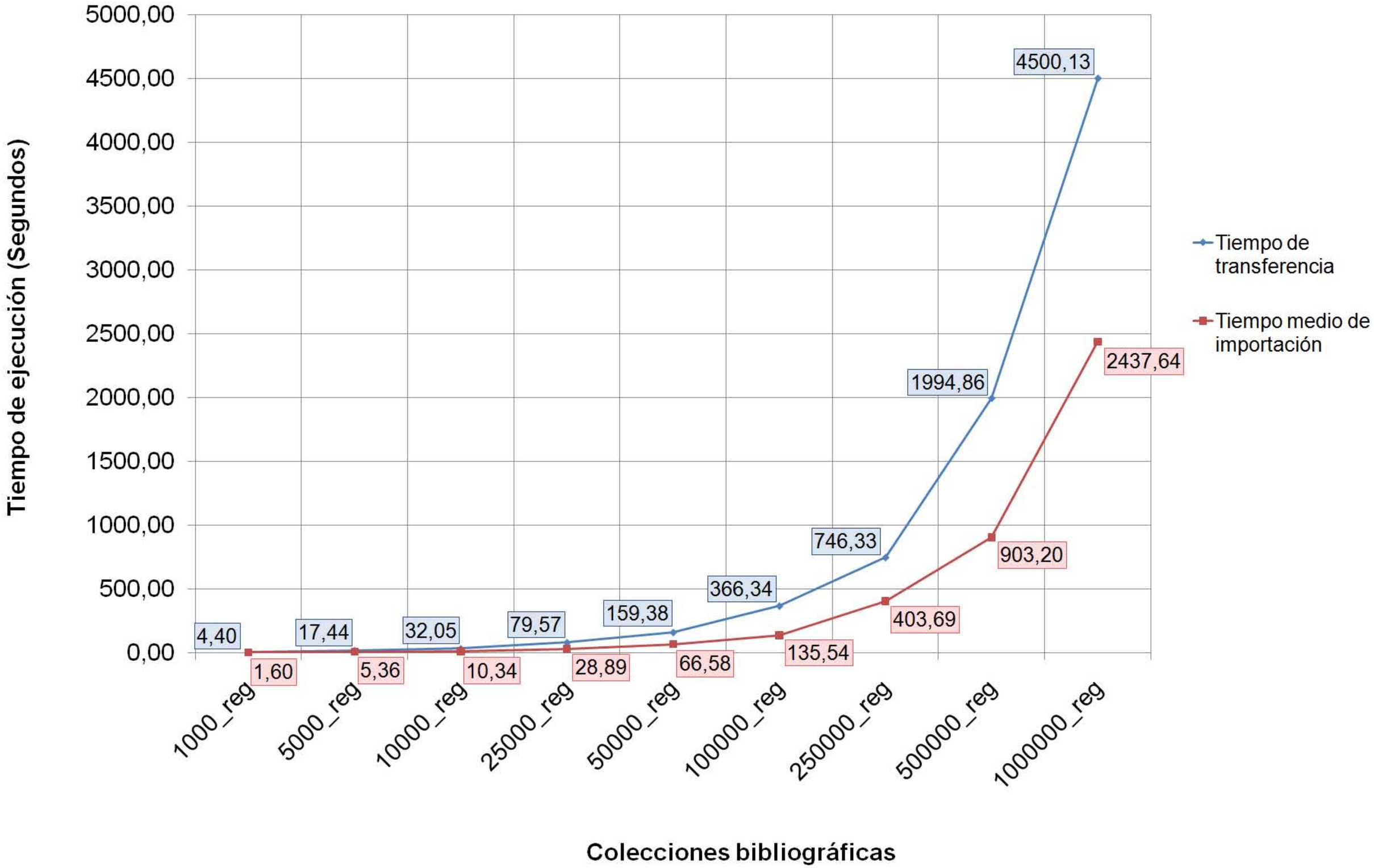


Título: Comparación de los tiempos de transferencia y la media de importación
Referencia: figura36



Comparación de los tiempos de transferencia y la media de importación

Referencia: figura36



Para concluir este apartado de análisis de los resultados, se está en disposición de resolver las preguntas planteadas al comienzo del desarrollo del sistema SYNC.

- **¿Es posible sindicar colecciones bibliográficas?** Es la principal pregunta que hay que plantear en relación a la sindicación y su aplicación a los servicios bibliotecarios. Como se ha podido comprobar, se han creado canales de sindicación en todos los formatos y volúmenes propuestos, por lo tanto hay que responder que sí es posible sindicar colecciones bibliográficas, quedando este hecho ampliamente demostrado.

- **¿Es posible formatear una colección bibliográfica en los formatos de sindicación tradicionales?** En relación a la posibilidad de utilizar los formatos de sindicación tradicionales para sindicar las colecciones, también hay que responder afirmativamente y precisar que tales formatos no son los más apropiados para la representación de registros bibliográficos.

- **¿Es posible utilizar formatos de sindicación no reconocidos como MARC-XML y llevar a cabo los mismos procesos de exportación, importación y transferencia que con los formatos de sindicación tradicionales RSS1.0, RSS2.0 y Atom?** Es perfectamente posible por cuanto supone el proceso de generación de los canales de sindicación, su exportación, lectura y transferencia mediante parser, al igual que los formatos RSS1.0, RSS2.0 y Atom. De hecho MARC-XML puede ser utilizado a los mismos efectos de redifusión de contenidos, lo que lo convierte en un formato útil para la sindicación de colecciones y registros bibliográficos.

- **¿Es capaz la sindicación de contenidos de crear canales bibliográficos de grandes dimensiones? ¿Cuál es la capacidad máxima de la sindicación para representar colecciones bibliográficas? ¿Cuál es el volumen óptimo de un catálogo bibliográfico para ser sindicado?** Sí es posible como se ha podido comprobar creando canales de sindicación que oscilan entre mil y un millón de registros, quedando representadas en estas pruebas las bibliotecas con pequeños, medianos y grandes fondos bibliográficos. Aunque también es reseñable que conforme la cifra de registros supera el medio millón, el proceso de lectura e importación de los parser se ralentiza. Esto implica que el tamaño óptimo para la gestión de colecciones bibliográficas se estipule entre los 1000 y los 250.000 registros.

- **¿Qué soluciones hay para sindicar grandes colecciones bibliográficas?** La principal solución pasa por la partición de la colección bibliográfica en archivos de tamaño y volumen de registros asumible para los programas parser. Este volumen se ha situado en torno a los 1000 registros, de forma que tanto el proceso de lectura como de inserción de datos en SYNC, no se vieran colapsados cuando se tratan grandes colecciones bibliográficas. Finalmente, para recuperar todos los archivos que conforman el canal de sindicación, se emplea el método OPML de agrupación de recursos, de forma que se puedan acceder a partir de un archivo de tamaño muy liviano y sencillo de manejar.

- **¿Qué formato de sindicación es el más adecuado para representar información bibliográfica?** El formato más adecuado para la representación de información bibliográfica es MARC-XML, como queda refrendado en el análisis de adaptabilidad anteriormente explicado, por contemplar todas las áreas de descripción del documento. Su comportamiento de exportación es similar al del resto de formatos y tan solo obtiene peores resultados en los procesos de importación de colecciones de entre medio millón y un millón de registros.

- **¿Qué formato de sindicación es el más eficiente y rápido en los procesos de exportación, importación y transferencia?** El formato más eficiente es Atom si se tiene en cuenta que obtiene mejores resultados de adaptabilidad bibliográfica y resultados similares en cuanto a tiempos de exportación e importación comparados con el mejor RSS2.0. Aún así siempre será muy inferior si se compara con MARC-XML en cuanto a descripción bibliográfica se refiere.

7.6. Diseño y arquitectura del sistema de gestión de servicios sindicados SYNCORE

Hasta el momento se ha explicado el desarrollo y funcionamiento de la plataforma de sindicación de contenidos SYNC, capaz de generar canales de sindicación con diversas colecciones bibliográficas. El siguiente paso es permitir la gestión bibliográfica de las colecciones mediante sindicación de contenidos, problema para el que se ha desarrollado el programa SYNCORE.

Tal y como se ha explicado en el primer apartado del capítulo, la sindicación de contenidos puede emplearse como una alternativa al protocolo Z39.50. Esta tesis, sostiene que ello es posible dadas las características de la sindicación para transmitir contenidos bibliográficos en múltiples formatos y al existir programas parser, capaces de interpretar y analizar sus estructuras para extraer la información. Dicho de otra forma, se ha demostrado que es posible controlar la generación de canales de sindicación, su lectura y manipulación posterior. Por ende no es descabellado que puedan emplearse métodos de edición, búsqueda y ordenación de las colecciones bibliográficas de la plataforma SYNC mediante sindicación de contenidos. De esta forma se convertiría en un sistema híbrido entre un OPAC y un protocolo Z39.50, que en todo caso persigue una menor cantidad de requerimientos para su instalación y su funcionamiento compartido con terceros centros de información.

Por tanto los interrogantes que se esperan resolver con el programa SYNCORE son los siguientes:

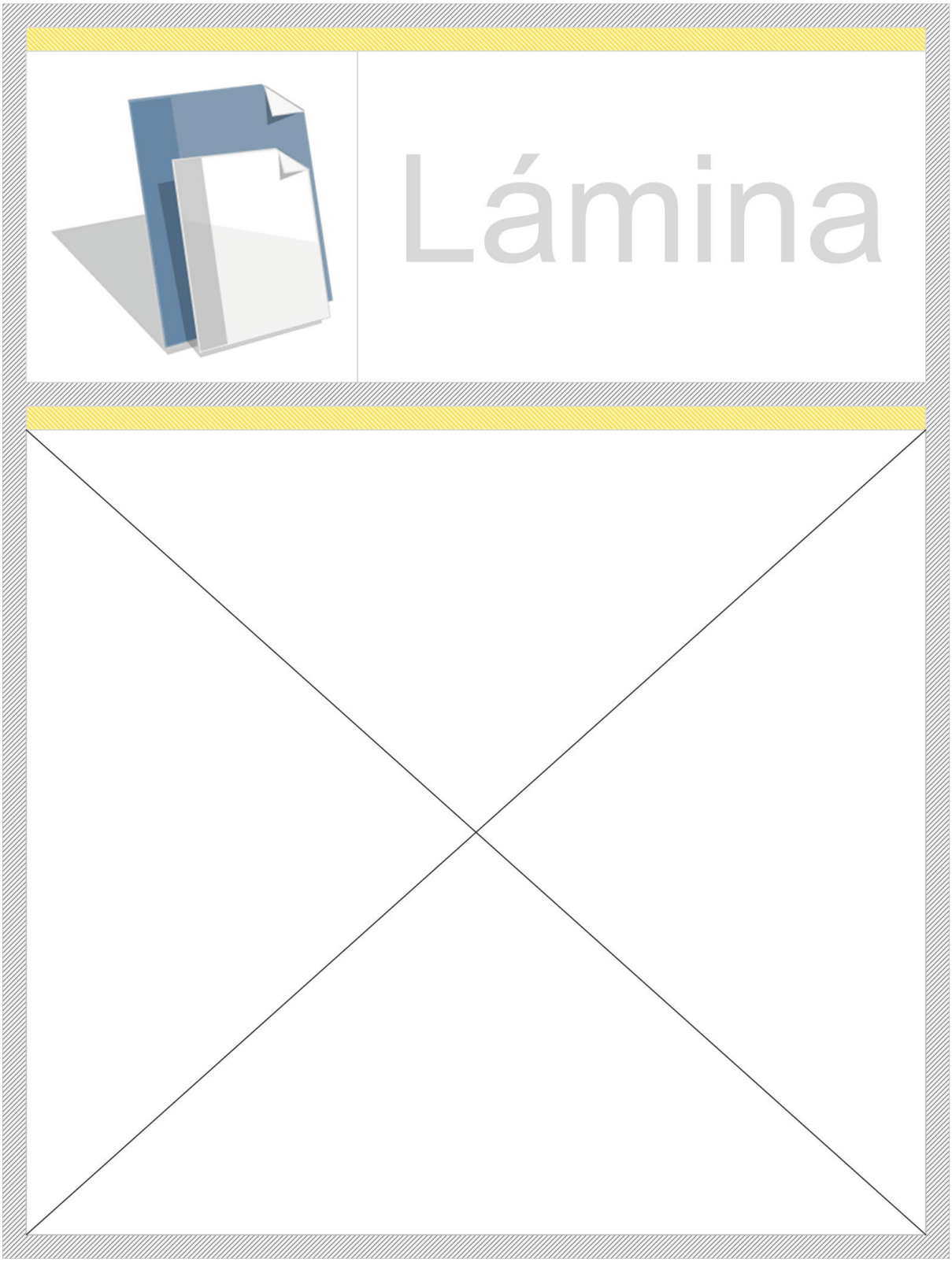
- ¿Es posible gestionar catálogos bibliográficos mediante técnicas de sindicación de contenidos?
- ¿Se pueden sindicar los servicios bibliotecarios relativos al catálogo?
- ¿Qué formatos se pueden emplear para la sindicación de los servicios y la gestión de catálogos bibliográficos?
- ¿Realmente el programa SYNCORE puede recuperar registros bibliográficos y seleccionar sus resultados como haría un cliente Z39.50?
- ¿Es realmente la sindicación en SYNCORE una alternativa a Z39.50 y en qué medida?

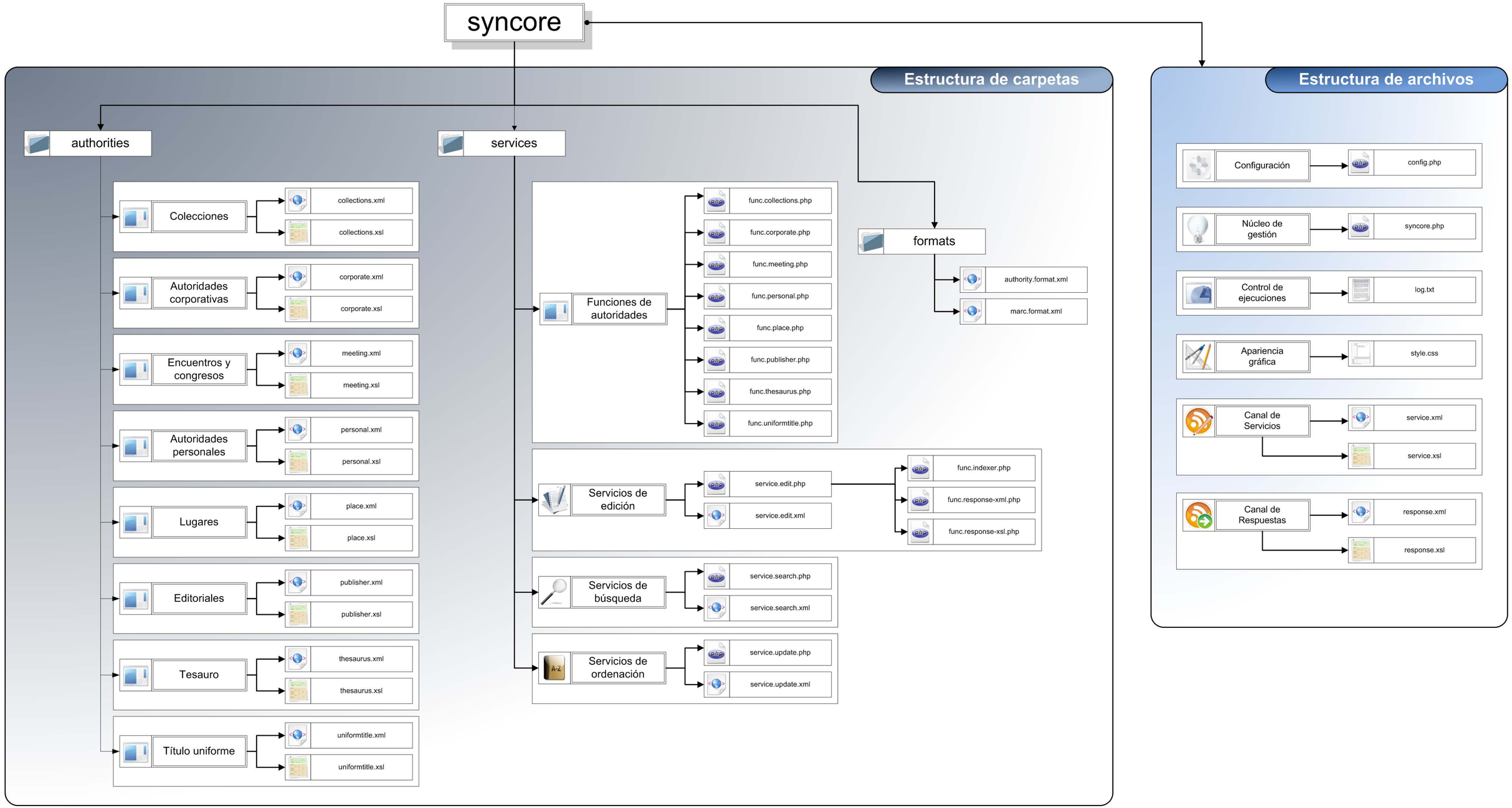
En cuanto a los requerimientos de SYNCORE, ha sido diseñado para estar alojado en un servidor Apache con PHP y MySQL, al igual que la plataforma SYNC. Su misión principal consiste en conectar con una base de datos bibliográfica basada en SYNC y sindicar los servicios correspondientes a la gestión de dicho catálogo. Desde el punto de vista del usuario, en cambio, únicamente se precisan aquellos elementos que permitan visualizar el canal generado por SYNCORE, esto es, un navegador que soporte XML, XSLT, XPath y Javascript. Afortunadamente, todos los navegadores conocidos satisfacen estos requerimientos.

En relación a la arquitectura del programa SYNCORE, se distingue la estructura de carpetas compuesta por *authorities*, *services* y *formats*. La carpeta *authorities* guarda todas las autoridades de las colecciones que están siendo consultadas. De hecho se encuentran autoridades corporativas, personales, encuentros, congresos y jornadas, lugares, editoriales, título uniforme e incluso un tesauro. En la carpeta *services* se guardan los módulos de los servicios que ofrece SYNCORE, por un lado las funciones de autoridades capaces de regenerar los contenidos de la carpeta *authorities*, las funciones de edición, consulta y ordenación. Finalmente, la carpeta *formats* contiene la descripción de los formatos de sindicación empleados. En este caso, como se explicará, las etiquetas de MARC-XML y el formato específico utilizado en las autoridades.

A parte de la estructura de carpetas, se encuentran los archivos del propio programa SYNCORE, compuestos por un archivo de configuración, el programa núcleo de gestión, un archivo de registro para el control de ejecución del programa, archivos de apariencia gráfica y los archivos propios del canal de sindicación de servicios y respuestas que se explicarán a continuación.

Título: Arquitectura de SYNCORE
Referencia: figura37




syncore
Estructura de carpetas
Estructura de archivos
authorities
Colecciones
collections.xml
collections.xsl
Autoridades corporativas
corporate.xml
corporate.xsl
Encuentros y congresos
meeting.xml
meeting.xsl
Autoridades personales
personal.xml
personal.xsl
Lugares
place.xml
place.xsl
Editoriales
publisher.xml
publisher.xsl
Tesauro
thesaurus.xml
thesaurus.xsl
Título uniforme
uniformtitle.xml
uniformtitle.xsl
services
Funciones de autoridades
func.collections.php
func.corporate.php
func.meeting.php
func.personal.php
func.place.php
func.publisher.php
func.thesaurus.php
func.uniformtitle.php
Servicios de edición
service.edit.php
service.edit.xml
func.indexer.php
func.response-xml.php
func.response-xsl.php
Servicios de búsqueda
service.search.php
service.search.xml
Servicios de ordenación
service.update.php
service.update.xml
formats
authority.format.xml
marc.format.xml
Configuración
config.php
Núcleo de gestión
syncore.php
Control de ejecuciones
log.txt
Apariencia gráfica
style.css
Canal de Servicios
service.xml
service.xsl
Canal de Respuestas
response.xml
response.xsl

7.7. Construcción y funcionamiento de SYNCORE

El programa SYNCORE resulta mucho menos voluminoso que la plataforma SYNC, pero también mucho más complejo en cuanto a su funcionamiento. Los programas esenciales de SYNCORE son los siguientes:

Título: Resumen de programas esenciales de SYNCORE
Referencia: tabla125

Programa	Nombre de archivo/s	Función
Instalación y Configuración	config.php	Contiene los datos de conexión a una plataforma SYNC.
Programa núcleo de redirección y ejecución de servicios	syncore.php	Es el núcleo del programa SYNCORE, que recibe las peticiones de servicio de los usuarios, ejecutando los servicios demandados y devolviendo canales de sindicación renovados con las respuestas a cada petición.
Servicio de edición	service.edit.php service.edit.xml func.indexer.php func.response-xml.php func.response-xsl.php	Constituye un módulo de servicio, instalado por defecto en el sistema SYNCORE. Su objetivo es proporcionar canales de sindicación para la edición de las colecciones bibliográficas.
Servicio de búsqueda	service.search.php service.search.xml	Se trata de un módulo de servicio, también instalado por defecto en el sistema SYNCORE, cuyo objetivo, es efectuar las búsquedas de los usuarios desde un canal de sindicación y generar un canal de sindicación con las respuestas.
Servicio de ordenación	service.update.php service.update.xml	Es un módulo de servicio original para SYNCORE, desarrollado con el objetivo de ordenar las colecciones bibliográficas sindicadas que el usuario desee en un canal de sindicación.

Como se explicará a continuación, el modelo de funcionamiento general se basa en la redirección de las peticiones del usuario cuando requiere del funcionamiento de un servicio bibliotecario. Para ello el canal de sindicación de servicios permite remitir al programa núcleo *syncore.php* una cabecera de datos que identifica el servicio pedido por el usuario. Dicho

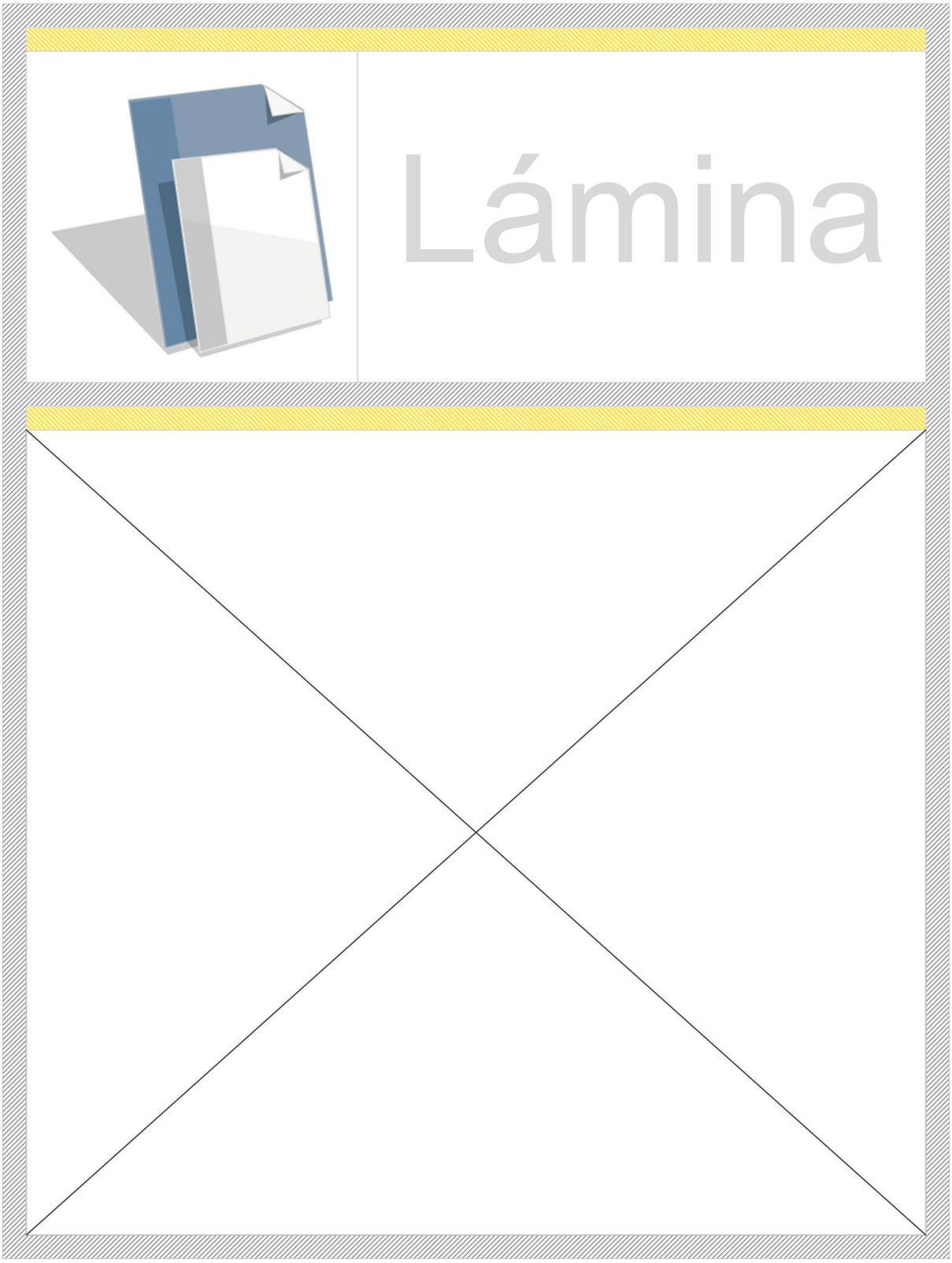
servicio es identificado entre los existentes en la carpeta *services*, concretamente el servicio de edición, búsqueda u ordenación. Una vez confrontado, el servicio es cargado y finalmente ejecutado por SYNCORE. Durante el proceso de ejecución el servicio está diseñado para generar una respuesta completa al usuario que se traduce en la regeneración del canal de servicios al completo, lo que origina un cambio en la información y contenidos que el usuario estaba visualizando con el navegador web.

Este funcionamiento de la sindicación, supone un concepto unidireccional tanto en cuanto los canales de servicios y respuestas se regeneran para mantener actualizados los contenidos demandados por el usuario. Dicho de otra forma, se transmiten contenidos que varían y se actualizan de la misma manera que cualquier canal de sindicación, pero permiten a la par la capacidad de interactuar con el sistema SYNCORE para operar con dichos contenidos, ya sea para editar el catálogo bibliográfico, recuperar un registro u ordenar la colección para conocer cuáles son las últimas modificaciones efectuadas.

Esta capacidad hace que SYNCORE aporte nuevas posibilidades de aplicación de la sindicación de contenidos, que alcanza una dimensión bidireccional, al tener en cuenta las necesidades del usuario. Este planteamiento responde a los principios biblioteconómicos más elementales de centrar los servicios bibliotecarios en torno al usuario, dado que es la razón de ser de los centros de información y documentación.

Otro aspecto decisivo en el diseño y funcionamiento de SYNCORE es la tecnología empleada en los canales de sindicación de servicios y respuestas, ya que aglutinan XML, XSL, XPath y Javascript. Con estos lenguajes es posible generar un interfaz gráfico tan completo, como el de cualquier sitio web, y permitir la transformación de los contenidos en XML para su correcta visualización. Pero también la ejecución de peticiones y procesos de almacenamiento y transmisión de variables sin necesidad de PHP, únicamente mediante el empleo de Javascript. Esta característica hace que los canales de sindicación se ejecuten en todo momento en el navegador web del usuario y no en el servidor, permitiendo claramente distinguirlo.

Título: Funcionamiento general de SYNCORE
Referencia: figura38



Funcionamiento general de SYNCORE

Referencia: figura38

Petición de servicio

Usuarios

Interacción con el servicio

Canal de sindicación de servicios bibliográficos

URL de acceso al canal: http://localhost/syncore/service.xml

Canal de servicios

service.xml

- crear nuevo registro bibliográfico
- editar registro bibliográfico existente
- consulta bibliográfica
- relación de últimas actualizaciones

service.xsl

<iframe src="http://localhost/syncore/response.xml"></iframe>

Canal de respuestas

response.xml

response.xsl

http

method POST

Cabecera del servicio

- Código de servicio
- Nombre del servicio
- archivo del módulo

Interacción

- Identificador de función
- Datos transmitidos

Transmisión de peticiones, servicios y datos

- El sistema SYNCORE entiende el proceso de interacción por medio del envío de cabeceras de servicio, resueltas en primer lugar, y a partir de las funciones de interacción de cada uno de ellos que se ejecutan para tratar los datos que sean transmitidos.

Programa núcleo

syncore.php

Redirección y ejecución

- El programa núcleo, identifica las peticiones vía http POST, identificando en primer lugar el servicio, para dar lugar a su ejecución y procesamiento de las funciones pertinentes.
- Este proceso de identificación tiene su base en los archivos service.*.xml situados en la carpeta services, que son inherentes a cada módulo instalado en SYNCORE. Contienen todos los datos identificativos, necesarios para contrastar la situación del servicio, su actualización, así como su existencia o no en el servidor, sin necesidad de efectuar complejos procesos de consulta como ocurre en Z39.50

Identificación de servicio

services/service.*.xml

- service.edit.xml
- service.search.xml
- service.update.xml
- ...

Modelo de archivo service.*.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<sync>
  <service>
    <id>Identificador del servicio</id>
    <update>Fecha de actualización del servicio</update>
    <name>Nombre del servicio</name>
    <file>Nombre completo del archivo</file>
    <url>URL completa del servicio</url>
    <description>Descripción del servicio</description>
  </service>
</sync>
```

Cargador de servicio

services

- Servicios de edición → service.edit.php
- Servicios de búsqueda → service.search.php
- Servicios de ordenación → service.update.php

Ejecución de servicio

Regeneración de los canales de servicio y respuestas

Modificación del canal de servicios

Actualización de servicios instalados

- Comprobación de servicios en services/ service.*.xml y reescritura del archivo service.xml

Modificación del canal de respuestas

Respuesta del servicio demandado

- Regeneración del canal de respuesta con los datos o contenidos demandados, así como los resultados de las consultas efectuadas.

Proceso de regeneración

- El proceso de regeneración de los canales se sucede siempre que existe cualquier tipo de interacción del usuario. Implica la modificación de los canales para ofrecer los contenidos demandados en cada momento. Para ello SYNCORE refresca y recarga los contenidos durante la sesión, de forma que puedan observarse los cambios.
- En el caso del canal de servicios se reescribe y actualiza para comprobar la instalación de nuevos servicios. Este proceso permite que el usuario siempre tenga información actualizada de los servicios disponibles, como haría cualquier canal de sindicación.
- En el caso del canal de respuestas, puede regenerarse en forma de canal de información con formulario o canal de sindicación MARC-XML cuando incluye contenidos bibliográficos.

Leyenda

- Explicación del proceso
- Carpeta
- Canal de sindicación
- Servicio de edición
- Archivo XML
- Servicio de búsqueda
- Archivo XSL
- Servicio de ordenación
- Petición o demanda
- Archivo PHP
- Script o proceso
- Usuarios
- Actualización y refresco

Especificaciones de formatos

Probablemente uno de los aspectos más complejos al que se ha enfrentado la investigación, ha sido la elección de los formatos XML y de sindicación con que el sistema SYNCORE iba a funcionar.

Las necesidades de un sistema capaz de interactuar con el usuario, implican que los formatos tradicionales de sindicación no siempre podrían asegurar la descripción de los servicios utilizados en SYNCORE, o el modo más apropiado de transmitir un sencillo mensaje al usuario que lo utiliza.

Si bien estaba claro que el formato ideal para la transmisión de los resultados bibliográficos era el propio MARC-XML abreviado por su relativa sencillez sintáctica y amplia capacidad de descripción, no estaba claro el empleo de los formatos sucedáneos aún en desarrollo por parte de la Library of Congress especializados en la descripción de autoridades como el propio MADS (Metadata Authority Description Schema: MADS, 2008). Esto se debe a los problemas que plantea dicho formato para distinguir muy diversos tipos de autoridades de forma clara y precisa, debido en parte al tipo de etiquetas empleado y a la mezcla necesaria con el formato MODS (Metadata Object Description Schema: MODS, 2009) para completar las descripciones de las autoridades. Este hecho hace que resulte difícil de aplicar y poco práctico, máxime cuando todavía son considerados borradores para el desarrollo de una nueva norma de descripción de objetos y autoridades en XML. Como tarea añadida, también surgió el problema de elaborar un tesauro de prueba para comprobar el comportamiento de SYNCORE con los lenguajes documentales. Este problema bien podría resolverse con una ontología, pero tal idea se desechó por requerir hojas XSL excesivamente complejas, que dificultaba el reconocimiento rápido de los términos de un posible tesauro.

Teniendo en cuenta todas estas circunstancias y dado que no se conocían todas las implicaciones que podrían surgir del desarrollo de una aplicación como SYNCORE, se decidió en primer lugar seleccionar MARC-XML abreviado como principal formato de sindicación bibliográfico. En segundo lugar desarrollar un formato para la definición de los servicios bibliográficos, las respuestas de referencia e informativas al usuario, por otro lado un formato de autoridades inspirado en el borrador de las normas MADS, de forma que tomará sus mejores ideas de definición de autoridades con una estructura más sencilla y lógica y finalmente un formato especial para el desarrollo de tesauros XML ultra-sencillos que debería reconocer una jerarquía de tres niveles con los términos empleados.

El resultado del formato para la descripción de servicios y respuestas de SYNCORE, se terminó denominando <*sync*> y se ha traducido en la siguiente tabla de especificaciones:

Título: Especificaciones del formato <sync>
Referencia: tabla126

Etiqueta y atributos	Aplicación	Descripción
<sync>	General	Etiqueta de apertura y cierre del formato SYNC, común a todos los usos de descripción de servicios, soporte y comunicación. Es el contenedor principal del formato.
<service> </service>	Descripción de servicios	Subelemento de *sync*. La etiqueta *service* permite agrupar todos los elementos de descripción de un servicio en SYNCORE.
<id> </id>		Subelemento de *service*. Permite establecer un código de identificación del servicio utilizado para generar las cabeceras de petición.
<update> </update>		Subelemento de *service*. Contiene la fecha de actualización del servicio en formato completo ISO 8601.
<name> </name>		Subelemento de *service*. Contiene la denominación del servicio.
<file> </file>		Subelemento de *service*. Define el nombre del archivo que efectúa el servicio bibliográfico.
<url> </url>		Subelemento de *service*. Define la URL completa o parcial del servicio.
<description> </description>		Subelemento de *service*. Permite describir el servicio y sus prestaciones.
<response> </response>	General de respuestas	Subelemento de *sync*. Permite identificar que todos los contenidos anidados entre *response* constituyen una respuesta del sistema SYNCORE. Engloba por tanto, las etiquetas *tag y message*.
<tag> </tag>	Respuestas de soporte	Subelemento de *response*. Permite definir el nombre de una etiqueta para su posterior descripción. Dentro de la etiqueta *tag* se incluyen los subelementos *description* y *reference*.

<description> </description>	Respuestas de soporte	Subelemento de *tag*. Permite describir una etiqueta de un tercer formato, como por ejemplo las etiquetas de MARC-XML.
<reference> </reference>		Subelemento de *tag*. Permite introducir una URL de referencia donde se explica el funcionamiento de un elemento de un tercer formato. En el caso de SYNCORE, se ha empleado para redirigir a las normas MARC en línea.
<message> </message>	Respuestas de comunicación	Subelemento de *response.* Permite definir un mensaje predeterminado del programa SYNCORE para el usuario que lo utiliza.

El formato de autoridades de SYNCORE aún carece de denominación alguna, pero bien podría considerarse en el apartado de recomendaciones e interpretaciones que podrían ayudar a desarrollar el próximo formato MADS de la Library of Congress. El desarrollo del trabajo ha llevado igualmente a crear la siguiente tabla de especificaciones:

Título: Especificaciones del formato de autoridades en SYNCORE
Referencia: tabla127

Etiqueta y atributos	Descripción
<collection>	Etiqueta de apertura y cierre del formato de autoridades.
<authority type='personal, corporate, meeting, place, uniformtitle, publisher'> </authoritiy>	Etiqueta para definir una autoridad de tipo personal, corporativa, de encuentros, lugares, título uniforme o editorial. Tales calificadores se determinan en el atributo *type.*
<identifier code='PER, CORP, MEE, UTI, PLA, PUB'>número de autoridad</identifier>	Subelemento de <authority>. Permite definir un código de autoridad compuesto por el atributo *code* y un número entero positivo, identificador de la autoridad. Dependiendo del tipo de autoridad, el atributo *code,* toma los valores PER (personas), CORP (entidades), MEE (encuentros, congresos, jornadas), UTI (título uniforme), PLA (lugares) y PUB (editoriales).
<fullname type='main, secondary'>nombre de la autoridad</fullname>	Subelemento de <authority>. Designa la forna normalizada del nombre de la autoridad propiamente dicha. Permite la introducción del atributo *type* para definir si esa forma de autoridad es la principal (toma el valor *main*) o secundaria (toma el valor *secondary*).

<date></date>	Subelemento de <authority>. Define una fecha o intervalo de fechas relativas a la autoridad, tal como las fechas de producción, vida y obra de autores, periodos de tiempo, etc.
<affiliation type='organization, occupation, email, website, geographic'></affiliation>	Subelemento de <authority>. Permite establecer una relación de la autoridad con cualquier otro elemento de tipo organizativo, ocupacional, correo electrónico, sitio web, geográfico, etc. Para definir estas características se emplea el atributo *type*.
</authority>	Cierre de la autoridad.
</collection>	Cierre de la colección.

En cuanto al formato de tesauro XML, se denominó <*thesaurus*>, dando lugar a un lenguaje muy sencillo en cuanto a sus elementos, basado en el anidamiento de las etiquetas que definen los términos del mismo. El resultado de dicho formato se puede comprobar en la *tabla128* de especificaciones.

Título: Especificaciones del formato <thesaurus> en SYNCORE
Referencia: tabla128

Etiqueta y atributos	Descripción
<thesaurus>	Etiqueta de apertura y cierre del tesauro
<term type='cat1, cat2, cat3…'> </term>	Subelemento de *thesaurus*. La etiqueta *term* permite definir un término en el tesauro y su nivel jerárquico mediante el atributo *type*. Así pues es posible definir todos los niveles que sean necesarios dentro de la estructura jerárquica. Otra característica fundamental es que resulta un elemento repetible que puede contener a su vez, tantos elementos *term* como fueren necesarios. De esta manera, un término de tipo *cat1*, podrá contener múltiples términos de tipo *cat2* y un término de tipo cat2, múltiples de tipo *cat3*.
<id>	Subelemento de *term.* Permite definir el número de identificación del término. Normalmente será correlativo, sin mezclar los términos correspondientes a terceras categorías. Por ejemplo una cuenta de identificadores correlativa a términos de primer nivel *cat1*, otra independiente para términos de segundo nivel *cat2* y así sucesivamente.

<name>	Subelemento de *term*. Permite establecer el nombre del término que se está describiendo.

Como conclusión al sub-apartado de formatos empleados en SYNCORE, la experiencia demuestra que los formatos empleados funcionan correctamente en la aplicación, permitiendo la sindicación y redifusión de los contenidos de sus entradas. No obstante en cualquier caso pueden ser reemplazados si así fuera necesario por cualquier otro formato de sindicación que respete la información básica que se debe transmitir para identificar cada servicio bibliográfico del sistema.

Esto significa que SYNCORE es una aplicación compatible con cualquier otro formato ya sean o no de sindicación, puesto que como se explicará a continuación genera sus propios recursos de visualización para sus propios canales. Por tanto la capacidad de adaptación a cualquier necesidad bibliográfica y documental está asegurada.

7.7.1. Instalación y configuración

El programa SYNCORE no requiere apenas de instalación. Tan solo es necesario alojarlo en el servidor y configurar los datos de conexión a una plataforma SYNC en el archivo *config.php*, lo que implica el nombre del servidor, usuario y contraseña de MySQL correspondiente. Una vez efectuado este paso, el sistema carece de los canales de sindicación iniciales, de forma que se necesita ejecutar el programa núcleo *syncore.php* para generarlos automáticamente. Basta con cargar *syncore.php* una vez, para empezar a funcionar con el sistema SYNCORE.

Este proceso de instalación y configuración, se ha diseñado de forma sencilla para permitir la conexión a una sóla plataforma SYNC. Esto se ha hecho para efectuar pruebas de funcionamiento de manera segura, con la intención de demostrar el concepto SYNCORE de sindicación de servicios bibliográficos. No obstante, sí es posible técnicamente añadir más servidores que utilicen la plataforma SYNC, tantos como fueran necesarios al igual que haría un servidor Z39.50.

7.7.2. Programa núcleo de redirección y ejecución de servicios

Enlazando con el apartado de instalación y configuración, el programa núcleo de redirección y ejecución de servicios es el centro neurálgico de todas las operaciones que se efectúan en el programa. Sus funciones son múltiples y resulta necesario analizarlas con detenimiento en la *tabla129*.

Título: Funciones del programa núcleo syncore.php
Referencia: tabla129

Funciones	Descripción
Generador del canal de servicios	Crea el canal de sindicación de servicios básico, siempre que se inicializa o actualiza la aplicación.
Generador del canal de respuestas	Crea el canal de sindicación de respuestas básico, siempre que se inicializa o actualiza la aplicación.
Cargador de servicios	Recibe las peticiones de servicio del usuario y las resuelve ejecutando en todo momento el adecuado.
Finalizador de servicios	Permite cortar o terminar la ejecución de un servicio o su sesión cuando esta acaba.

Para explicar desde el principio *syncore.php* es necesario comprender cómo genera los canales de sindicación de servicios bibliográficos. En realidad SYNCORE genera dos canales de sindicación, por un lado el canal de servicios propiamente dicho en el que se visualizan todos los servicios bibliográficos disponibles y por otro lado, el canal de respuestas que contiene los resultados de la interactuación del usuario con el canal. Dicho canal de respuestas está embebido en el canal de servicios, permitiendo al usuario visualizar ambos sin necesidad de cargar cada uno en una pestaña diferente del navegador. Esto es posible mediante el empleo de <*iframe*></*iframe*> propio de la técnica AJAX[26] que posibilita la introducción de un marco web, que en este caso es relleno con el canal de sindicación de respuestas.

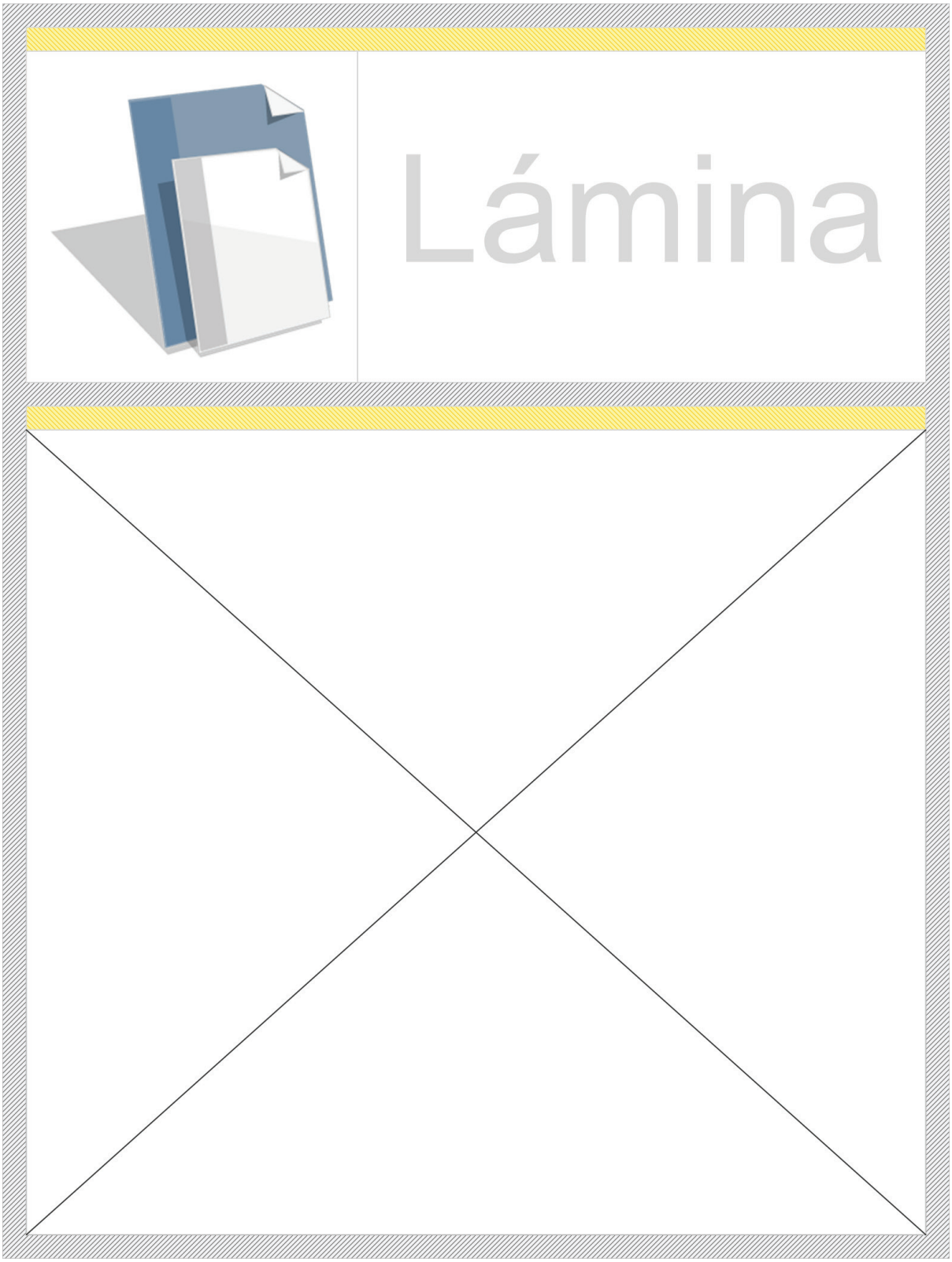
[26] **AJAX:** Asynchronous JavaScript And XML es una técnica de programación web basada en el empleo de los lenguajes Javascript y XML principalmente, para operar con independencia del servidor, funcionando y ejecutando todos los scripts desde el navegador del cliente.

Además ello conlleva ventajas de actualización de las ventanas, ya que tal operación se efectúa en bloque.

Como se ha relatado anteriormente, si es la primera vez que se instala el programa SYNCORE, será necesario ejecutar una única vez el archivo *syncore.php*, ya que generará todos los canales de sindicación y sus hojas XSL para visualizarlos correctamente. De hecho una vez generados los canales de sindicación SYNCORE se convierte en un programa autónomo que no requiere de atención alguna, puesto que está diseñado para mantenerse oculto tanto del usuario como del administrador, adquiriendo una función esclava de cualquier demanda o petición del usuario. Esto significa que desde ese momento, será necesario acceder siempre al canal de sindicación de servicios, para interactuar y operar con SYNCORE, no existiendo ningún otro método paralelo. La ruta de acceso a SYNCORE en una instalación local, sería muy similar a la ruta expuesta a continuación: http://localhost/syncore/service.xml

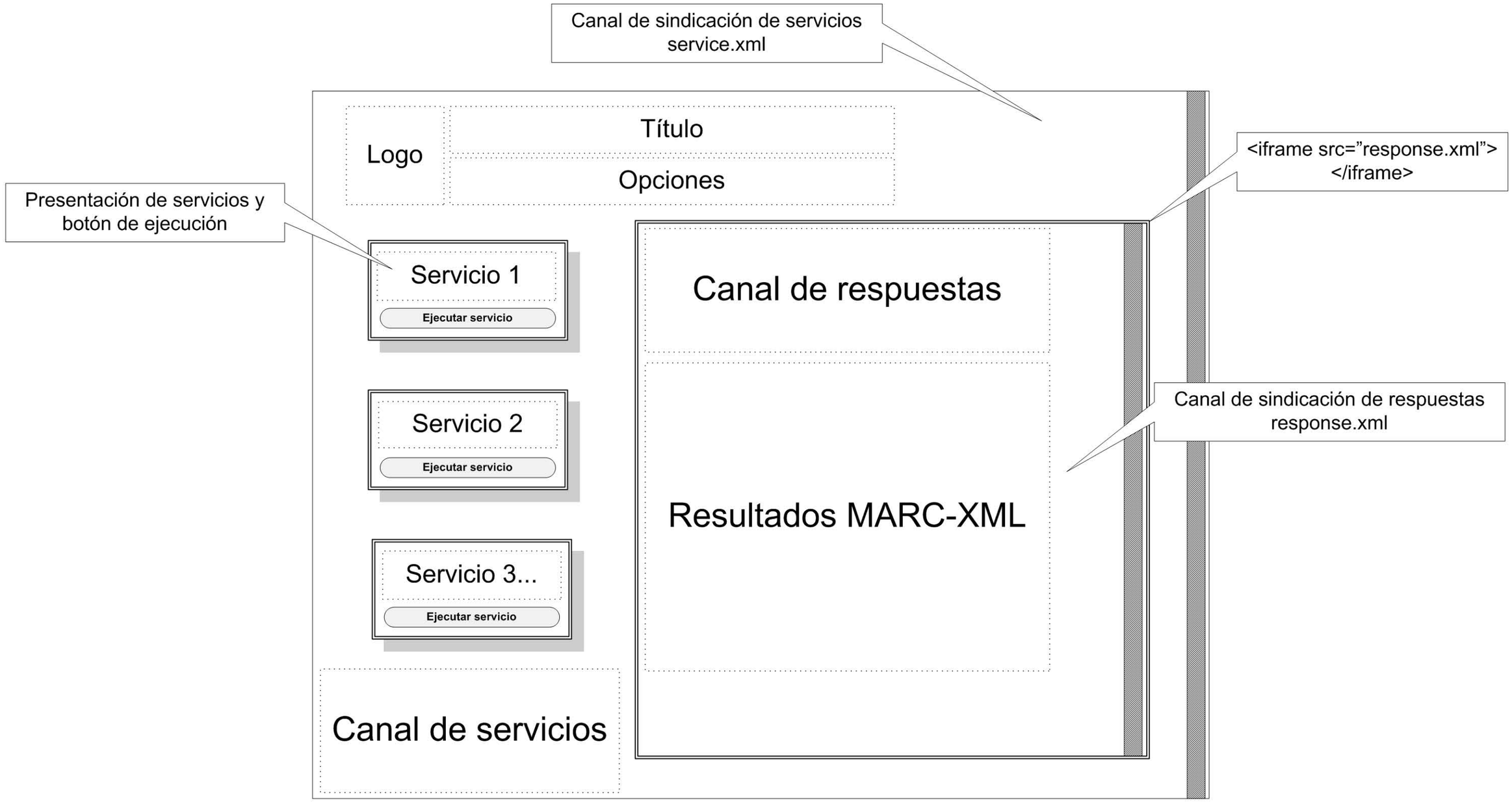
En cuanto a la fisionomía del canal de sindicación de servicios bibliográficos, puede apreciarse en la *figura39*.

Título: Fisionomía del canal de sindicación de servicios bibliográficos
Referencia: figura39



Fisionomía del canal de sindicación de servicios bibliográficos

Referencia: figura39



Generador del canal de servicios

La función especializada en la generación del canal de servicios sindicados está diseñada para comprobar automáticamente por medio de un parser XML qué módulos y servicios están instalados en SYNCORE. Esto es posible por la existencia de unos archivos XML que definen cada servicio y módulo disponible en la carpeta *services*. Una vez recopilados todos los registros correspondientes a los servicios presentes en el sistema, la función crea el canal service.xml, precisamente con la información recopilada. Para hacerlo visible y legible se genera a continuación la hoja XSL enlazada desde el canal de servicios como hoja de estilo predeterminada. Dicha hoja de estilo contiene las opciones de suscripción en favoritos y actualización manual del canal. También contiene los formularios con los botones de ejecución de cada servicio. Tales formularios son rellenos con las cabeceras y datos recopilados por el canal de sindicación XML que se ha generado.

Título: Función generadora del canal de sindicación de servicios
Referencia: tabla130

```
// GENERADOR DE CANAL DE SERVICIOS: service.xml
// Comprueba los archivos de información XML de cada servicio y los incorpora al canal

$file = fopen('service.xml','w');
fwrite ($file, "<?xml version='1.0' encoding='UTF-8'?>
                <?xml-stylesheet type='text/xsl' href='service.xsl'?><sync>");

$dir = opendir("services"); // Selección de la carpeta de servicios
while ($serv = readdir($dir)) {
        if (preg_match("/.xml$/", "$serv")) { // Selección de archivos de información XML
                $xml = simplexml_load_file("services/$serv");    // Parser de lectura
                for($i=0; $xml->service[$i]; $i++) {
                        $id = $xml->service[$i]->id;
                        $update = $xml->service[$i]->update;
                        $name = $xml->service[$i]->name;
                        $filename = $xml->service[$i]->file;
                        $description = $xml->service[$i]->description;

                        // Estructura de un registro del canal de servicios
                        fwrite ($file,"
                        <service>
                        <id>$id</id>
                        <update>$update</update>
                        <name>$name</name>
                        <file>$filename</file>
                        <url>services/$filename</url>
                        <description>$description</description>
                        </service>
                        ");
                }
        }
}
```

```
closedir($dir);
fwrite ($file,"</sync>");
fclose($file);

// GENERADOR DE HOJA XSL PARA SERVICIOS: service.xsl
// Genera un interfaz de interacción con los servicios bibliográficos

$file = fopen('service.xsl','w');
fwrite ($file,"<?xml version='1.0' encoding='UTF-8'?>
<xsl:stylesheet version='1.0' xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>

// Inicio de la plantilla XSL
<xsl:template match='/'>

<html>
<head>
<title>syncore service channel</title>
<meta http-equiv='content-type' content='text/html; charset=UTF-8'/>
<link rel='stylesheet' href='style.css' type='text/css'/>

// Código Javascript para agregar el canal a favoritos
<script type=\"text/javascript\" language=\"javascript\">
function add(){ if (navigator.appName == \"Microsoft Internet Explorer\") {
window.external.AddFavorite('http://127.0.0.1/syncore/service.xml','syncore service channel'); }
if (navigator.appName == \"Netscape\") {
window.sidebar.addPanel('syncore service channel','http://127.0.0.1/syncore/service.xml',''); }
}
</script>

</head>
<body>

<div class='sidegeneral'>
<div class='title1'><img src='logosync.png' alt='Sindicación activa' title='Sindicación activa'/> Canal
de servicios SYNCORE</div>
<div class='block1'>
<form action='syncore.php' method='post'>

// Botón favoritos
<input type='button' onclick='javascript:add()' value='suscribirse a este canal'/> -

// Botón para la actualización del canal de sindicación completo
<input type='submit' name='update' value='actualizar servicios disponibles'/>

</form>
</div>
</div>

<div class='sideservice'>

// Formateado de los contenidos del archivo service.xml correspondiente al canal
<p class='title2'>Lado del canal de servicios</p>
<xsl:for-each select='//service'>
<xsl:sort select='id'/>
<div class='itemservice'>
<form action='syncore.php' name='form' method='post'>
<p><xsl:value-of select='id'/></p>
<p><xsl:value-of select='update'/></p>
<p><xsl:value-of select='name'/></p>
<p><xsl:value-of select='description'/></p>
```

```
<input type='hidden' name='id' value='{id}'/>
<input type='hidden' name='file' value='{file}'/>
<input type='submit' name='launch' value='ejecutar servicio'/>
</form>
</div>
</xsl:for-each>
</div>

<div class='sideresponse'>
<p class='title2'>Lado del canal de respuestas</p>

// Iframe en el que se anida el canal de sindicación de respuestas
<iframe src='http://127.0.0.1/syncore/response.xml' class='frameresponse'
frameborder='0'></iframe>

</div>

</body>
</html>

</xsl:template>

</xsl:stylesheet>
");
fclose($file);
```

Generador del canal de respuestas

En cuanto al canal de respuestas, se genera con un código más sencillo dado que sólo introduce un mensaje de estado. También se incluye la función de actualización, por generar un archivo *response.xml*, muy similar teniendo en cuenta dicha consideración. En ambos casos una vez generados, ejecutan una instrucción Javascript, para actualizar y recargar el canal de sindicación *service.xml*.

Título: Función generadora del canal de sindicación de respuesta
Referencia: tabla131

```
// GENERADOR DEL CANAL DE RESPUESTA: response.xml

if (file_exists("response.xml")) { } else {
        $file = fopen('response.xml','w');
        fwrite ($file,"<?xml version='1.0' encoding='UTF8'?>
                <sync>
                <response>
                <message>Canal de respuesta esperando petición de servicio...</message>
                </response>
                </sync>");
                fclose($file);

// Instrucción Javascript para recargar el canal de sindicación, creada la respuesta.
```

```
        echo "<script language='javascript'>window.location.href='service.xml'</script>";
}

// FUNCIÓN DE ACTUALIZACIÓN: response.xml

if ($_POST[update]) {
        $file = fopen('response.xml','w');
        fwrite ($file,"<?xml version='1.0' encoding='UTF-8'?>
                <sync>
                <response>
                <message>Canal de servicios actualizado. Esperando petición de
                        servicio...</message>
                </response>
                </sync>");
                fclose($file);

// Instrucción Javascript para recargar el canal de sindicación, creada la respuesta.
        echo "<script language='javascript'>window.location.href='service.xml'</script>";

}
```

Cargador de servicios

El cargador de servicios está diseñado para recoger las peticiones de servicio del usuario mediante el protocolo http, método POST. Esta característica no resulta novedosa, puesto que fue empleada por primera vez en los formatos de sindicación RSS0.91 y RSS1.0, así como posteriormente en RSS2.0 aplicando en la estructura del canal el elemento *<textinput></textinput>*, *véase página 187*.

En SYNCORE a diferencia de los formatos de sindicación mencionados, habilita en los canales de sindicación y más concretamente en sus hojas XSL, simples formularios rellenos con la información del canal de sindicación. De esta forma, resulta más sencillo y práctico el proceso de envío de datos a syncore.php alojado en el servidor.

Originalmente esta función ha sido desarrollada para recibir una cabecera de datos compuesta por la acción de petición *$_POST[launch]*, el identificador del servicio *$_POST[id]* y el nombre del archivo del servicio *$_POST[file]*. Se han de dar estas condiciones de envío para que la función de carga de servicios funcione. Además efectúa una comprobación de servicios pedidos, con servicios existentes mediante parser XML de los módulos y servicios instalados en ese preciso momento. Si se cumplen todos los requisitos, el cargador de servicios incluye el código fuente del servicio demandado para que se ejecute a todos los efectos, según se puede comprobar en la siguiente tabla:

Título: Función cargador de servicios
Referencia: tabla132

```
// CARGADOR DE SERVICIOS
// Permite procesar la petición del usuario y cargar el servicio que demanda.

if ($_POST[launch]) {
        $xml = simplexml_load_file("service.xml");
        for($i=0; $xml->service[$i]; $i++) {
                $idpar = $xml->service[$i]->id;
                if ($idpar != "$_POST[id]") {} else { include ("services/$_POST[file]");
                }
        }
}
```

Finalizador de servicios

La función finalizadora de servicios tiene como objetivo cortar la ejecución de cualquier servicio siempre que éste lo requiera. A veces existen procedimientos de ejecución que deben ser cancelados para evitar bucles infinitos. Esta estructura asegura un correcto funcionamiento de los servicios y evita ese problema durante cualquier proceso de ejecución de SYNCORE. La finalización del servicio se efectúa mediante la instrucción Javascript *window.location.href* que recarga el canal de sindicación, invalidando de esa forma el proceso de ejecución presente hasta el momento.

Título: Función finalizadora de servicios
Referencia: tabla133

```
// FINALIZADOR DE SERVICIOS
// Permite finalizar todos los procesos de un servicio.

if ($_GET[func] == 'terminate1')  {
echo "<script language='javascript'>window.location.href='service.xml'</script>";
$file = fopen('response.xml','w');
fwrite ($file,"<?xml version='1.0' encoding='UTF-8'?>
<sync>
<response><message>El proceso ha finalizado correctamente.</message></response>
</sync>");
fclose($file);  }

if ($_GET[func] == 'terminate2')  {
echo "<script language='javascript'>window.location.href='service.xml'</script>";
}
```

7.7.3. El canal de servicios sindicados

El principal secreto del funcionamiento de SYNCORE reside en su canal de sindicación de servicios bibliográficos y su hoja XSL enlazada para proporcionar un interfaz gráfico, ambas generadas por el programa núcleo *syncore.php*. Éste canal ha sido desarrollado en un formato XML de prueba que se ha denominado <*sync*>, específicamente diseñado para describir los elementos identificadores básicos de cualquier servicio digital. Tales elementos son el <*id*> que es el identificador del servicio, <*update*> la fecha de actualización del servicio, <*name*> el nombre del servicio, <*file*> el archivo en el que está programado el servicio, <*url*> la url completa o relativa del servicio y <*description*> que incluye un breve resumen del funcionamiento y prestaciones del servicio.

El archivo propio del canal de servicios puede ser consultado en la *tabla134*.

Título: Canal de servicios service.xml
Referencia: tabla134

```
<?xml version='1.0' encoding='UTF-8'?>
<?xml-stylesheet type='text/xsl' href='service.xsl'?>
<sync>

<service>
<id>S1M1</id>
<update>2009-07-19 21:11:07</update>
<name>crear nuevo registro bibliográfico</name>
<file>service.edit.php</file>
<url>services/service.edit.php</url>
<description>permite crear un nuevo registro bibliográfico a partir de... </description>
</service>

<service>
<id>S1M2</id>
<update>2009-07-19 21:11:07</update>
<name>editar registro bibliográfico existente</name>
<file>service.edit.php</file>
<url>services/service.edit.php</url>
<description>posibilita la edición completa de un registro bibliográfico…</description>
 </service>

<service>
<id>S2M1</id>
<update>2009-07-19 21:11:07</update>
<name>consulta bibliográfica</name>
<file>service.search.php</file>
<url>services/service.search.php</url>
<description>permite efectuar consultas bibliográficas de tipo general....</description>
</service>
```

```
<service>
<id>S3M1</id>
<update>2009-07-19 21:11:07</update>
<name>relación de últimas actualizaciones</name>
<file>service.update.php</file>
<url>services/service.update.php</url>
<description>genera un repertorio ordenado de últimas…</description>
</service>

</sync>
```

El canal de servicios como se viene explicando, también consta de una hoja XSL especializada en proporcionar más que un estilo, una interfaz gráfica e interactiva destinada al usuario. Para ello el proceso de elaboración tiene en cuenta dos fases; por un lado la concreción de las funciones XSL que se utilizarán para representar la información del canal en formato XML. Por otro el diseño de la interfaz gráfica web, que contendrá los formularios y scripts necesarios para funcionar autónomamente sin requerimientos del servidor. En la siguiente tabla, se especifican las instrucciones fundamentales con las que se configura el archivo XSL para representar la información del canal de sindicación de servicios:

Título: Funciones XSL del canal de servicios.
Referencia: tabla135

```
// Apertura del documento XSL
<xsl:stylesheet version='1.0' xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>

// Instrucción de plantilla para los elementos del formato <sync>
<xsl:template match='/'>

// Selección de las estructuras “service” de forma secuencial hasta su finalización
<xsl:for-each select='//service'>

// Ordenación de los servicios según su identificador
<xsl:sort select='id'/>
        <xsl:value-of select='id'/> // Mostrar el valor del elemento id
        <xsl:value-of select='update'/> // Mostrar el valor del elemento update
        <xsl:value-of select='name'/> // Mostrar el valor del elemento name
        <xsl:value-of select='description'/> // Mostrar el valor del elemento description
</xsl:for-each>

// Final de la plantilla
</xsl:template>

// Cierre del documento XSL
</xsl:stylesheet>
```

Como se puede apreciar la base de un archivo XSL, siempre constará de los elementos básicos de delimitación del documento XSL *<xsl:stylesheet>*, de apertura y cierre de plantilla *<xsl:template>*, de selección de estructuras *<xsl:for-each>*, de ordenación de elementos por su contenido *<xsl:sort>* y de selección de los valores de cada elemento para su representación en pantalla <xsl:value-of select="/>.

En cuanto al apartado de formularios de la plantilla XSL, se configura creando campos ocultos que almacenan los datos de la cabecera de cada servicio representado. De esa forma cuando se remite la información a SYNCORE, se puede identificar cada servicio pedido. *Véase la tabla136*.

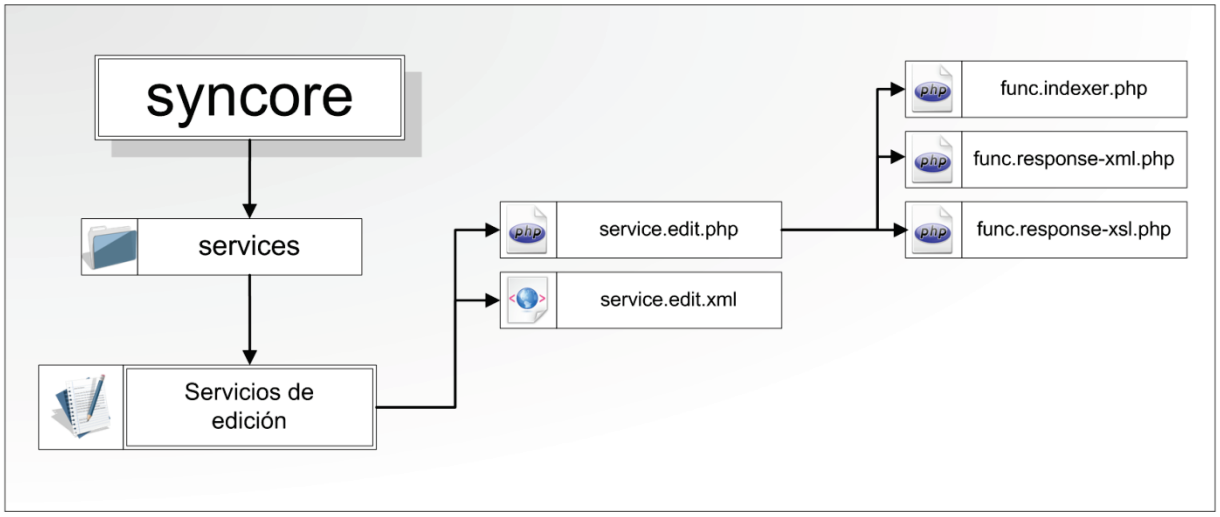
Título: Formulario XSL del canal de servicios
Referencia: tabla136

```
// Formulario con método http POST
<form action='syncore.php' name='form' method='post'>
<p><xsl:value-of select='id'/></p> // Identificador del servicio
<p><xsl:value-of select='update'/></p>// Fecha de actualización
<p><xsl:value-of select='name'/></p> // Nombre del servicio
<p><xsl:value-of select='description'/></p> // Descripción del servicio

// Campo oculto con el valor del identificador del servicio {id}
<input type='hidden' name='id' value='{id}'/>

// Campo oculto con el valor del nombre de archivo del servicio {file}
<input type='hidden' name='file' value='{file}'/>

// Botón de ejecución del servicio, en SYNCORE se detecta su pulsación.
<input type='submit' name='launch' value='ejecutar servicio'/>
</form>
```

7.7.4. Servicio de edición

El servicio de edición al igual que el resto de servicios se aloja en la carpeta *services*, y resulta el más complejo de todos ellos, debido a que contempla el caso de la incorporación de nuevos registros en el catálogo bibliográfico y la edición de un registro existente en alguno de los catálogos disponibles.

Título: Estructura de archivos del servicio de edición
Referencia: figura40



Estas dos opciones del servicio están identificadas en el canal de servicios por su identificador S1M1 y S1M2 que describen S1 (servicio 1) y M* (módulo 1 o 2). De esta forma se indica al programa que instrucciones debe ejecutar en cada momento.

Crear nuevo registro bibliográfico

La opción de creación de un nuevo registro bibliográfico, implica la ejecución de una serie de archivos que generan un completo formulario de edición para la catalogación en formato MARC-XML. Concretamente, se cargan las colecciones disponibles en las plataformas SYNC que han sido conectadas a SYNCORE. Esta operación la efectúa el programa *func.collections.php*, que genera un listado XML de dichas colecciones, que se almacena en la carpeta *authorities*. Así pues se pueden elegir las colecciones en las que se desea introducir un nuevo registro bibliográfico o efectuar una catalogación puntual. A continuación se ejecuta el generador del canal de respuestas para la edición y en especial para la creación de un nuevo

registro. Básicamente genera un canal que contiene información, descripción y referencias útiles para la catalogación en relación a las etiquetas MARC que se emplean en el formulario de edición. Al generador del canal de respuesta, le sigue la ejecución del generador de la hoja XSL correspondiente, que en este caso origina un formulario completo para la catalogación de documentos, con los elementos constituyentes del formato de sindicación MARC-XML. Una vez generados los canales de respuesta, se procede a la actualización completa del canal de servicios para hacer efectivos los cambios en el navegador web.

Para completar el funcionamiento del servicio de creación de nuevos registros bibliográficos, ha sido necesario contemplar el procedimiento de inserción y procesamiento de la catalogación que se efectúa desde el canal de sindicación de respuestas. Esto se logra mediante la identificación tanto del servicio como de la operación que se requiere para con los datos del formulario, de forma tal que la cabecera permita redirigir a SYNCORE al programa adecuado.

En cuanto al mecanismo de procesamiento de la información, se ha procurado respetar y mantener la codificación MARC, incluyendo su propia cabecera, que es reconocida y compuesta automáticamente por el servicio. Además procesa todos los puntos de accesos primarios y secundarios del documento para que sean recogidos e indexados en un campo especial de indexación que permita la elaboración de un fichero inverso depurado en la plataforma SYNC, de la que es originario el catálogo bibliográfico que se está explotando. De esta forma su posterior recuperación se ve simplificado por la interrogación en un único campo de consulta.

Finalmente, una vez preparados todos los contenidos, se dan de alta en el catálogo, respetando como puede observarse en la *tabla137*, todas sus denominaciones de etiqueta, elementos y calificadores.

Título: Funciones de creación de un nuevo registro bibliográfico
Archivo: service.edit.php
Referencia: tabla137

Función de creación de un nuevo registro

- Genera una respuesta que incluye un formulario de catalogación MARC completo para lo que elabora un listado de colecciones en las que puede ser incorporado el nuevo registro y documentación de referencia para la catalogación.

```
case 'S1M1': // OPCIÓN 1: CREAR UN NUEVO REGISTRO

$idservice = "$_POST[id]"; // Guarda el identificador del servicio.
$fileservice = "$_POST[file]"; // Guarda el nombre del archivo del servicio.

// Carga las colecciones disponibles.
include ("services/func.collections.php");

// Ejecuta el generador del canal de respuesta específico para la edición.
include ("services/func.response-xml.php");

// Ejecuta el generador de la hoja XSL para el canal de respuesta en modo de edición.
include ("services/func.response-xsl.php");

// Actualización completa del canal de servicios.
echo "<script language='javascript'>window.parent.location.href='service.xml'</script>";

break; // FIN DEL CASO 1
```

Función de procesamiento de un nuevo registro

- Permite incorporar el registro bibliográfico a la colección, para lo que efectúa un procesamiento adecuado de la información como la correcta disposición de las signaturas topográficas, el área de título y mención de responsabilidad e incluso la cabecera del registro en MARC.

```
// FUNCIÓN INSERTAR O PROCESAR UN NUEVO REGISTRO
// Permite procesar todas las variables correspondientes a los campos bibliográficos de un
// nuevo registro en la colección.

if($_POST[insertnew]) {

// Procesa el campo 050$a de signatura topográfica.
$instag050a = "$_POST[tag050a1] ; $_POST[tag050a2] ; $_POST[tag050a3]";

// Procesa el campo 245$c de mención de responsabilidad del área de título.
if($_POST[tag100a] != '') { $instag245c = "$_POST[tag100a]"; }
elseif ($_POST[tag110a] != '') { $instag245c = "$_POST[tag110a]"; }
elseif ($_POST[tag111a] != '') { $instag245c = "$_POST[tag111a]"; }
else { $instag245c = "$_POST[tag130a]"; }
$instag245c = "$instag245c $_POST[tag700a] $_POST[tag710a]";

// Procesa el campo 650$a de acceso temático.
$instag650a = "$_POST[tag650a1] ; $_POST[tag650a2] ; $_POST[tag650a3]";
```

```
// Procesa la ficha bibliográfica completa del documento.
$instag887a = "$insleader .- $_POST[tag001] .- $_POST[tag003] .- $_POST[tag005] .-
$_POST[tag010a] .- $_POST[tag017a] .- $_POST[tag020a] .- $_POST[tag041a] .- $instag050a .-
$_POST[tag080a] .- $_POST[tag082a] .- $_POST[tag100a] .- $_POST[tag110a] .- $_POST[tag111a]
.- $_POST[tag130a] .- $_POST[tag245a] .- $_POST[tag245b] .- $instag245c .- $_POST[tag250a] .-
$_POST[tag250b] .- $_POST[tag260a] .- $_POST[tag260b] .- $_POST[tag260c] .- $_POST[tag300a]
.- $_POST[tag300b] .- $_POST[tag300c] .- $_POST[tag300e] .- $_POST[tag490a] .-
$_POST[tag490v] .- $_POST[tag500a] .- $instag650a .- $_POST[tag700a] .- $_POST[tag710a]";

// cuenta la longitud de caracteres del registro bibliográfico.
$insleader1 = strlen($instag887a);

// Procesa los caracteres del campo cabecera leader.
$insleader =
$insleader1.$_POST[leader2].$_POST[leader3].$_POST[leader4].$_POST[leader5].$_POST[leader
6].$_POST[leader7].$_POST[leader8].$_POST[leader9].$_POST[leader10].$_POST[leader11].$_PO
ST[leader12].$_POST[leader13].$_POST[leader14];

// Procesa los puntos de acceso al documento para su normalización, futuro indexado y
recuperación.
$texto = "$_POST[tag020a] $_POST[tag100a] $_POST[tag110a] $_POST[tag111a]
$_POST[tag130a] $_POST[tag245a] $_POST[tag245b] $_POST[tag260a] $_POST[tag260b]
$_POST[tag260c] $_POST[tag490a] $_POST[tag490v] $_POST[tag500a] $instag650a
$_POST[tag700a] $_POST[tag710a]";
include("func.indexer.php");

// Crea un nuevo registro con los datos de todos los campos catalográficos cumplimentados.
mysql_select_db("$database", $con);
$sql = "INSERT INTO $_POST[coldestiny] (leader, tag001, tag003, tag005, tag010a, tag017a,
tag020a, tag041a, tag050a, tag080a, tag082a, tag100a, tag100o, tag110a, tag110o, tag111a,
tag111o, tag130a, tag130o, tag245a, tag245b, tag245c, tag250a, tag250b, tag260a, tag260b,
tag260c, tag300a, tag300b, tag300c, tag300e, tag490a, tag490v, tag500a, tag650a, tag650o,
tag700a, tag700o, tag710a, tag710o, tag887a, indexer)  VALUES ('$insleader', '$_POST[tag001]',
'$_POST[tag003]', '$_POST[tag005]', '$_POST[tag010a]', '$_POST[tag017a]', '$_POST[tag020a]',
'$_POST[tag041a]', '$instag050a', '$_POST[tag080a]', '$_POST[tag082a]', '$_POST[tag100a]',
'$_POST[tag100o]', '$_POST[tag110a]', '$_POST[tag110o]', '$_POST[tag111a]', '$_POST[tag111o]',
'$_POST[tag130a]', '$_POST[tag130o]', '$_POST[tag245a]', '$_POST[tag245b]', '$instag245c',
'$_POST[tag250a]', '$_POST[tag250b]', '$_POST[tag260a]', '$_POST[tag260b]', '$_POST[tag260c]',
'$_POST[tag300a]', '$_POST[tag300b]', '$_POST[tag300c]', '$_POST[tag300e]', '$_POST[tag490a]',
'$_POST[tag490v]', '$_POST[tag500a]', '$instag650a', '$_POST[tag650o]', '$_POST[tag700a]',
'$_POST[tag700o]', '$_POST[tag710a]', '$_POST[tag710o]', '$instag887a', '$indexer')";

mysql_query($sql,$con);

// Carga el finalizador de ejecución del servicio.
echo "<script language='javascript'>window.parent.location.href='$url?func=terminate1' </script>";

}
```

Editar registro bibliográfico existente

El servicio de edición de un registro bibliográfico existente emplea el mismo canal de respuesta XML que en el caso de creación de un nuevo registro, puesto que conlleva el uso del mismo formulario, solo que esta vez relleno con los datos del registro que se desea editar. Por este motivo el servicio consta de tres funciones esenciales; 1) Generar el formulario de consulta del registro, 2) Cargar la información del registro consultado en el formulario de edición, 3) Una vez hechas las modificaciones sobre el formulario, actualizar los datos en el catálogo.

Título: Funciones de edición de un registro bibliográfico existente
Archivo: service.edit.php
Referencia: tabla138

Función de consulta de un registro existente para su edición

- Se genera un canal de respuesta XML en el que se pide al usuario la introducción de un número de control o ISBN del registro que desee editar. Para ello se proporciona un formulario de consulta en el archivo XSL, en el que se permite también la selección de la colección correspondiente.

```
case 'S1M2': // OPCIÓN 2: EDITAR UN REGISTRO EXISTENTE

$idservice = "$_POST[id]"; // Guarda el identificador del servicio.
$fileservice = "$_POST[file]"; // Guarda el nombre del archivo del servicio.

$file = fopen('response.xml','w');
fwrite ($file,"<?xml version='1.0' encoding='UTF-8'?>
<?xml-stylesheet type='text/xsl' href='response.xsl'?>
<sync>
<response>
<message>Introduzca el nº de control o ISBN asignado al registro bibliográfico</message>
</response>
</sync>");

fclose($file);

$file = fopen('response.xsl','w');
fwrite ($file,"<?xml version='1.0' encoding='UTF-8'?>
<xsl:stylesheet version='1.0' xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
<xsl:template match='/'>

<html>
<head>
<title>syncore response channel</title>
<meta http-equiv='content-type' content='text/html; charset=UTF-8'/>
<link rel='stylesheet' href='style.css' type='text/css'/>
</head>

<body>
<p><h3>Editar registro bibliográfico existente</h3></p>
```

```
<form action='$url' method='post'>
	<input type='hidden' name='id' value='$idservice'/>
	<input type='hidden' name='file' value='$fileservice'/>
	<input type='hidden' name='launch' value='launch'/>
	<select name='coldestiny'>");

		mysql_select_db("$database", $con);
		$sql1 = "SHOW TABLE STATUS FROM $database WHERE Name REGEXP
		'^col.*.marc1$'"
		$resultado1 = mysql_query($sql1);

		while($array = mysql_fetch_array($resultado1)) { fwrite ($file,"
			<option value='$array[Name]'>$array[Name]</option>"); }

	fwrite ($file,"  </select>
	<input type='text' name='idisbn' size='42'/>
	<input type='submit' name='searchedit' value='buscar y editar'/>
</form>

<ul>
<xsl:for-each select='//response'>
<xsl:value-of select='message'/>
</xsl:for-each>
</ul>

</body>
</html>

</xsl:template>
</xsl:stylesheet>");

fclose($file);
echo "<script language='javascript'>window.parent.location.href='service.xml'</script>";

break;
// FIN CASO 2
}
```

Función de carga del registro

- La función consulta el ISBN o número de control en la colección seleccionada, devolviendo todos los datos del registro al formulario de edición que visualizará el usuario en el canal de respuesta.

```
// FUNCIÓN CARGAR UN REGISTRO EXISTENTE
// Permite visualizar un registro seleccionado por el usuario para su edición

if($_POST[searchedit]) {

$idservice = "$_POST[id]"; // Guarda el identificador del servicio.
$fileservice = "$_POST[file]"; // Guarda el nombre del archivo del servicio.
$coldestiny = "$_POST[coldestiny]"; // Guarda la colección MARC sobre la que se está operando.

mysql_select_db("$database", $con);
$sql = "SELECT * FROM $_POST[coldestiny] WHERE id LIKE '$_POST[idisbn]' OR tag001 LIKE
'$_POST[idisbn]' OR tag020a LIKE '$_POST[idisbn]'";
$resultado = mysql_query($sql,$con);
```

```
if ($fila = mysql_fetch_assoc($resultado)) {

// Proceso de split o división de los elementos que conforman la cabecera del registro MARC

        $stringtag050a = $fila[tag050a];
        $v1 = split(";", $stringtag050a);
        $nstr = count($v1);
        for ($i=0; $i<$nstr; $i++) { if($v1[$i] != '') { $tag050a[] = trim($v1[$i]); } else {} }

        $stringtag650a = $fila[tag650a];
        $v2 = split(";", $stringtag650a);
        $nstr = count($v2);
        for ($i=0; $i<$nstr; $i++) { if($v2[$i] != '') { $tag650a[] = trim($v2[$i]); } else {} }

        $stringleader = $fila[leader];
                $numero = 5; $leader1 = substr($stringleader,0,$numero);
                $numero = 1; $leader2 = substr($stringleader,5,$numero);
                $numero = 1; $leader3 = substr($stringleader,6,$numero);
                $numero = 1; $leader4 = substr($stringleader,7,$numero);
                $numero = 1; $leader5 = substr($stringleader,8,$numero);
                $numero = 1; $leader6 = substr($stringleader,9,$numero);
                $numero = 1; $leader7 = substr($stringleader,10,$numero);
                $numero = 5; $leader8 = substr($stringleader,11,$numero);
                $numero = 1; $leader9 = substr($stringleader,16,$numero);
                $numero = 1; $leader10 = substr($stringleader,17,$numero);
                $numero = 1; $leader11 = substr($stringleader,18,$numero);
                $numero = 1; $leader12 = substr($stringleader,19,$numero);
                $numero = 4; $leader13 = substr($stringleader,20,$numero);
                $numero = 1; $leader14 = substr($stringleader,24,$numero);

        include ("services/func.collections.php");
        include ("services/func.response-xml.php");
        include ("services/func.response-xsl.php");

echo "<script language='javascript'>window.parent.location.href='service.xml'</script>";
die();
        }
}
```

Función de actualización del registro

- Una vez efectuados los cambios del registro, la función de actualización recibe todos los datos del mismo y los actualiza en la colección bibliográfica. Al igual que cuando se introduce un nuevo registro, efectúa los procesos de preparación de los datos, cabeceras MARC e indexación de puntos de acceso al documento.

```
// FUNCIÓN DE ACTUALIZACIÓN O MODIFICACIÓN UN REGISTRO EXISTENTE
// Permite procesar todas las variables correspondientes a los campos bibliográficos de un
// nuevo registro en la colección.

if($_POST[editexist]) {

// Procesa el campo 050$a de signatura topográfica.
$instag050a = "$_POST[tag050a1] ; $_POST[tag050a2] ; $_POST[tag050a3]";

// Procesa el campo 245$c de mención de responsabilidad del área de título.
if($_POST[tag100a] != '') { $instag245c = "$_POST[tag100a]"; }
```

```
elseif ($_POST[tag110a] != '') { $instag245c = "$_POST[tag110a]"; }
elseif ($_POST[tag111a] != '') { $instag245c = "$_POST[tag111a]"; }
else { $instag245c = "$_POST[tag130a]"; }
$instag245c = "$instag245c $_POST[tag700a] $_POST[tag710a]";

// Procesa el campo 650$a de acceso temático.
$instag650a = "$_POST[tag650a1] ; $_POST[tag650a2] ; $_POST[tag650a3]";

// Procesa la ficha bibliográfica completa del documento.
$instag887a = "$insleader .- $_POST[tag001] .- $_POST[tag003] .- $_POST[tag005] .-
$_POST[tag010a] .- $_POST[tag017a] .- $_POST[tag020a] .- $_POST[tag041a] .- $instag050a .-
$_POST[tag080a] .- $_POST[tag082a] .- $_POST[tag100a] .- $_POST[tag110a] .- $_POST[tag111a]
.- $_POST[tag130a] .- $_POST[tag245a] .- $_POST[tag245b] .- $instag245c .- $_POST[tag250a] .-
$_POST[tag250b] .- $_POST[tag260a] .- $_POST[tag260b] .- $_POST[tag260c] .- $_POST[tag300a]
.- $_POST[tag300b] .- $_POST[tag300c] .- $_POST[tag300e] .- $_POST[tag490a] .-
$_POST[tag490v] .- $_POST[tag500a] .- $instag650a .- $_POST[tag700a] .- $_POST[tag710a]";

$insleader1 = strlen($instag887a);

// Procesa los caracteres del campo cabecera leader.
$insleader =
$insleader1.$_POST[leader2].$_POST[leader3].$_POST[leader4].$_POST[leader5].$_POST[leader
6].$_POST[leader7].$_POST[leader8].$_POST[leader9].$_POST[leader10].$_POST[leader11].$_PO
ST[leader12].$_POST[leader13].$_POST[leader14];

// Procesa los puntos de acceso al documento para su normalización, futuro indexado y
recuperación.
$texto = "$_POST[tag020a] $_POST[tag100a] $_POST[tag110a] $_POST[tag111a]
$_POST[tag130a] $_POST[tag245a] $_POST[tag245b] $_POST[tag260a] $_POST[tag260b]
$_POST[tag260c] $_POST[tag490a] $_POST[tag490v] $_POST[tag500a] $instag650a
$_POST[tag700a] $_POST[tag710a]";
        include("func.indexer.php");

// Modifica el registro con los datos de todos los campos catalográficos cumplimentados.
mysql_select_db("$database", $con);
$sql = "UPDATE $_POST[coldestiny] SET leader = '$insleader', tag001 = '$_POST[tag001]', tag003
= '$_POST[tag003]', tag005 = '$_POST[tag005]', tag010a = '$_POST[tag010a]', tag017a =
'$_POST[tag017a]', tag020a = '$_POST[tag020a]', tag041a = '$_POST[tag041a]', tag050a =
'$instag050a', tag080a = '$_POST[tag080a]', tag082a = '$_POST[tag082a]', tag100a =
'$_POST[tag100a]', tag100o = '$_POST[tag100o]', tag110a = '$_POST[tag110a]', tag110o =
'$_POST[tag110o]', tag111a = '$_POST[tag111a]', tag111o = '$_POST[tag111o]', tag130a =
'$_POST[tag130a]', tag130o = '$_POST[tag130o]', tag245a = '$_POST[tag245a]', tag245b =
'$_POST[tag245b]', tag245c = '$instag245c', tag250a = '$_POST[tag250a]', tag250b =
'$_POST[tag250b]', tag260a = '$_POST[tag260a]', tag260b = '$_POST[tag260b]', tag260c =
'$_POST[tag260c]', tag300a = '$_POST[tag300a]', tag300b = '$_POST[tag300b]', tag300c =
'$_POST[tag300c]', tag300e = '$_POST[tag300e]', tag490a = '$_POST[tag490a]', tag490v =
'$_POST[tag490v]', tag500a = '$_POST[tag500a]', tag650a = '$instag650a', tag650o =
'$_POST[tag650o]', tag700a = '$_POST[tag700a]', tag700o = '$_POST[tag700o]', tag710a =
'$_POST[tag710a]', tag710o = '$_POST[tag710o]', tag887a = '$instag887a', indexer = '$indexer'
WHERE id = '$_POST[tag001]'";
mysql_query($sql,$con);

// Carga el finalizador de ejecución del servicio.
echo "<script language='javascript'>window.parent.location.href='$url?func=terminate1'
</script>";

}
```

El formulario del archivo XSL que se genera tanto en el servicio de edición como en el de creación de un nuevo registro es muy similar al que se muestra en la *tabla139*. Una de las ventajas del diseño de este formulario reside en la inclusión y enlace con terceros archivos XML de autoridades mediante la técnica *<iframe></iframe>* mencionada anteriormente. De hecho es posible visualizar en el formulario listados de autoridades según el área de descripción en la que se encuentre el usuario catalogando, por ejemplo las autoridades personales, corporativas, título uniforme, congresos y jornadas, editoriales, lugares y tesauro. El formulario también automatiza la incorporación de los datos de las autoridades a la catalogación, mediante el empleo de instrucciones Javascript para el control de eventos en formularios, de forma que siempre está controlada la información relativa a los puntos de acceso principal. Por otro lado el formulario XSL formatea la información de referencia de cada etiqueta y área de descripción disponible en el canal de respuestas que se genera a la par.

Título: Archivo de respuesta de edición func.response-xsl.php
Archivo: func.response-xsl.php
Referencia: tabla139

Inicio del archivo response.xsl

- Destaca por incluir un código Javascript para expandir y colapsar los elementos correspondientes a las áreas de descripción. Esto permite organizar la visualización del formulario según las necesidades del usuario.

```
<?xml version='1.0' encoding='UTF-8'?>
<xsl:stylesheet version='1.0' xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
<xsl:template match='/'>

<html>
<head>
<title>syncore response channel</title>
<meta http-equiv='content-type' content='text/html; charset=UTF-8'/>
<link rel='stylesheet' href='style.css' type='text/css'/>

<script type='text/javascript' language='javascript'>
function HideContent(d) { document.getElementById(d).style.display = 'none'; }
function ShowContent(d) { document.getElementById(d).style.display = 'block'; }
function ReverseDisplay(d) {
if(document.getElementById(d).style.display == 'none') { document.getElementById(d).style.display
= 'block'; } else { document.getElementById(d).style.display = 'none'; } }
</script>

</head>
<body>
```

Formulario de selección de colección e identificación de servicio

- Constituye la primera parte del formulario XSL, que permite identificar el servicio y la colección en la que se desea operar. En el caso de edición de un registro ya existente, la colección ya viene marcada por defecto.

```
<xsl:for-each select='//response'>
<form action='http://localhost/syncore/syncore.php' method='post'>

<div class='blockhead'>
<input type='hidden' name='id' value='S1M1'/>
<input type='hidden' name='file' value='service.edit.php'/>
<input type='hidden' name='launch' value='launch'/>

<div style='position: relative; float: left;'>
<input type='text' name='coldestiny' id='coldestiny' size='32' value=''/>
<a href="javascript:ShowContent('collection')">+</a>
<a href="javascript:HideContent('collection')">- colección</a>
<div id='collection' style='display: block;'>

<iframe src='authorities/collections.xml' scrolling='yes' class='blockframe'></iframe>
</div>

</div>
<input type='submit' name='insertnew' value='grabar registro bibliográfico'/> - S1M1
<input type='reset' name='reset' value='borrar datos'/>
</div>

<br/>
```

Formulario de la cabecera MARC del registro

- Está formado por todos los elementos de la cabecera del registro MARC, denominado *leader* que definen el tipo de documento, extensión, nivel de descripción, etc.

```
<a href="javascript:ShowContent('leader')">+ cabecera</a>
<div id='leader' style='display: block;'>

<b><a href='{//tag[@name="leader"]/reference}' target='_blank'
title='{//tag[@name="leader"]/description}'>leader</a></b>

<input type='text' maxlength='5' size='3' name='leader1' value='' readonly='readonly'/>
<input type='text' maxlength='1' size='1' name='leader2' value='n' readonly='readonly'/>

<select maxlength='1' name='leader3' size='1'>
        <option value='a'>a</option>
</select>

<input type='text' maxlength='1' size='1' name='leader4' value=' ' readonly='readonly'/>
<input type='text' maxlength='1' size='1' name='leader5' value='a' readonly='readonly'/>
<input type='text' maxlength='1' size='1' name='leader6' value='2' readonly='readonly'/>
<input type='text' maxlength='1' size='1' name='leader7' value='2' readonly='readonly'/>
<input type='text' maxlength='5' size='1' name='leader8' value='00000' readonly='readonly'/>
<input type='text' maxlength='1' size='1' name='leader9' value='2' readonly='readonly'/>
```

```
<select name='leader10'>
        <option value=''>#</option>
        <option value='1'>1</option>
        <option value='2'>2</option>
        <option value='3'>3</option>
        <option value='4'>4</option>
        <option value='5'>5</option>
        <option value='7'>7</option>
        <option value='8'>8</option>
        <option value='u'>u</option>
        <option value='z'>z</option>
</select>

<input type='text' maxlength='1' size='1' name='leader11' value='i' readonly='readonly'/>
<input type='text' maxlength='1' size='1' name='leader12' value=' ' readonly='readonly'/>
<input type='text' maxlength='4' size='3' name='leader13' value='4500' readonly='readonly'/>
<input type='text' maxlength='1' size='1' name='leader14' value=' ' readonly='readonly'/>

<a href="javascript:HideContent('leader')">- cabecera</a>
</div>
```

Formulario de la campos de control

- Contiene los campos de control correspondientes al número de control de la biblioteca, el identificador del centro y la fecha de creación o actualización del registro.

```
<a href="javascript:ShowContent('controlfields')">+ campos de control</a>
<div id='controlfields' style='display: block;'>

// Número de control de la biblioteca
<a href='{//tag[@name="001"]/reference}' target='_blank'
title='{//tag[@name="001"]/description}'>tag001</a>

// Identificador del centro
<a href='{//tag[@name="003"]/reference}' target='_blank'
title='{//tag[@name="003"]/description}'>tag003</a>

// Carga Fecha de última actualización
<a href='{//tag[@name="005"]/reference}' target='_blank'
title='{//tag[@name="005"]/description}'>tag005</a>
<input type='text' name='tag001' id='tag001' size='22' value='' readonly='readonly'/>
<input type='text' name='tag003' size='22' value=''/>
<input type='text' name='tag005' size='25' value='2009-11-07T12:22:48+01:00'
readonly='readonly'/>

<a href="javascript:HideContent('controlfields')">- campos de control</a>
</div>
```

Formulario para campos de códigos

- Proporciona los campos de códigos necesarios para la identificación general del registro bibliográfico, como el número de Library of Congress, el depósito legal, el ISBN, el código de idioma, la signatura topográfica, el número de Clasificación Decimal Universal y Dewey.

```
<a href="javascript:ShowContent('0XXfields')">+ campos de códigos</a>
<div id='0XXfields' style='display: block;'>

// Número de identificación de la Library of Congress
<a href='{//tag[@name="010"]/reference}' target='_blank'
title='{//tag[@name="010"]/description}'>tag010</a>
<a href='{//tag[@name="010a"]/reference}' target='_blank'
title='{//tag[@name="010a"]/description}'>$a</a>
<input type='text' name='tag010a' size='22' value=''/>

// Número de depósito legal
<a href='{//tag[@name="017"]/reference}' target='_blank'
title='{//tag[@name="017"]/description}'>tag017</a>
<a href='{//tag[@name="017a"]/reference}' target='_blank'
title='{//tag[@name="017a"]/description}'>$a</a>
<input type='text' name='tag017a' size='22' value=''/>

// Número de ISBN
<a href='{//tag[@name="020"]/reference}' target='_blank'
title='{//tag[@name="020"]/description}'>tag020</a>
<a href='{//tag[@name="020a"]/reference}' target='_blank'
title='{//tag[@name="020a"]/description}'>$a</a>
<input type='text' name='tag020a' size='22' value=''/>

// Código de Idioma según norma ISO639-2
<a href='{//tag[@name="041"]/reference}' target='_blank'
title='{//tag[@name="041"]/description}'>tag041</a>
<a href='{//tag[@name="041a"]/reference}' target='_blank'
title='{//tag[@name="041a"]/description}'>$a</a>

<select name='tag041a'>
        <option value=''></option>
        <option value='dan'>dan</option>
        <option value='deu'>deu</option>
        <option value='eng'>eng</option>
        <option value='fra'>fra</option>
        <option value='gre'>gre</option>
        <option value='ita'>ita</option>
        <option value='jpn'>jpn</option>
        <option value='lat'>lat</option>
        <option value='por'>por</option>
        <option value='rus'>rus</option>
        <option value='spa'>spa</option>
        <option value='spa'>swe</option>
        <option value='zho'>zho</option>
</select>

// Signatura topográfica
<a href='{//tag[@name="050"]/reference}' target='_blank'
title='{//tag[@name="050"]/description}'>tag050</a>
<a href='{//tag[@name="050a"]/reference}' target='_blank'
```

```
title='{//tag[@name="050a"]/description}'>$a</a>
<input type='text' name='tag050a1' size='22' value=''/>

<a href="javascript:ShowContent('tag050a2')" title='agregar campo'>+</a>
<div id='tag050a2' style='display: none;'>
<a href='{//tag[@name="050a"]/reference}' target='_blank'
title='{//tag[@name="050a"]/description}'>$a</a>
<input type='text' name='tag050a2' size='22' value=''/>
<a href="javascript:ShowContent('tag050a3')" title='agregar campo'>+</a>
<a href="javascript:HideContent('tag050a2')" title='quitar campo'>-</a>
</div>

<div id='tag050a3' style='display: none;'>
<a href='{//tag[@name="050a"]/reference}' target='_blank'
title='{//tag[@name="050a"]/description}'>$a</a>
<input type='text' name='tag050a3' size='22' value=''/>
<a href="javascript:HideContent('tag050a3')" title='quitar campo'>-</a>
</div>

// Clasificación CDU
<a href='{//tag[@name="080"]/reference}' target='_blank'
title='{//tag[@name="080"]/description}'>tag080</a>

<a href='{//tag[@name="080a"]/reference}' target='_blank'
title='{//tag[@name="080a"]/description}'>$a</a>
<input type='text' name='tag080a' size='80' value=''/>

// Clasificación DEWEY
<a href='{//tag[@name="082a"]/reference}' target='_blank'
title='{//tag[@name="082a"]/description}'>tag082</a>

<a href='{//tag[@name="082a"]/reference}' target='_blank'
title='{//tag[@name="082a"]/description}'>$a</a>
<input type='text' name='tag082a' size='80' value=''/>

<a href="javascript:HideContent('0XXfields')">- campos de códigos</a>
<div>
```

Formulario de asientos principales

- Contiene las autoridades personales, corporativas, congresos y jornadas y título uniforme. La introducción de datos es automática y contempla no solo la incorporación de la denominación propia de la autoridad en $a, sino también su número de control en $0.

```
<a href="javascript:ShowContent('1XXfields')">+ campos de asiento principal</a>
<div id='1XXfields' style='display: block;'>

// Autoridades personales
<a href='{//tag[@name="100"]/reference}' target='_blank'
title='{//tag[@name="100"]/description}'>tag100</a>
<a href='{//tag[@name="100a"]/reference}' target='_blank'
title='{//tag[@name="100a"]/description}'>$a</a>
<input type='text' name='tag100a' id='tag100a' size='40' value=''/>
<a href='{//tag[@name="100o"]/reference}' target='_blank'
title='{//tag[@name="100o"]/description}'>$0</a>
<input type='text' name='tag100o' id='tag100o' size='12' value=''/>
```

```
<a href="javascript:ShowContent('listpersonal0')">ver autoridades personales</a>
<div id='listpersonal0' style='display: none;'>
<iframe src='authorities/personal.xml' width='100%' height='200px'></iframe>
<a href="javascript:HideContent('listpersonal0')">ocultar autoridades personales</a>
</div>

// Autoridades corporativas
<a href='{//tag[@name="110"]/reference}' target='_blank'
title='{//tag[@name="110"]/description}'>tag110</a>
<a href='{//tag[@name="110a"]/reference}' target='_blank'
title='{//tag[@name="110a"]/description}'>$a</a>
<input type='text' name='tag110a' id='tag110a' size='40' value=''/>
<a href='{//tag[@name="110o"]/reference}' target='_blank'
title='{//tag[@name="110o"]/description}'>$0</a>
<input type='text' name='tag110o' id='tag110o' size='12' value=''/>

<a href="javascript:ShowContent('listcorporate0')">ver autoridades corporativas</a>
<div id='listcorporate0' style='display: none;'>
<iframe src='authorities/corporate.xml' width='100%' height='200px'></iframe>
<a href="javascript:HideContent('listcorporate0')">ocultar autoridades corporativas</a>
</div>

// Autoridades de reuniones, congresos y jornadas
<a href='{//tag[@name="111"]/reference}' target='_blank'
title='{//tag[@name="111"]/description}'>tag111</a>
<a href='{//tag[@name="111a"]/reference}' target='_blank'
title='{//tag[@name="111a"]/description}'>$a</a>
<input type='text' name='tag111a' id='tag111a' size='40' value=''/>
<a href='{//tag[@name="111o"]/reference}' target='_blank'
title='{//tag[@name="111o"]/description}'>$0</a>
<input type='text' name='tag111o' id='tag111o' size='12' value=''/>

<a href="javascript:ShowContent('listmeeting0')">ver autoridades de reuniones, congresos</a>
<div id='listmeeting0' style='display: none;'>
<iframe src='authorities/meeting.xml' width='100%' height='200px'></iframe>
<a href="javascript:HideContent('listmeeting0')">ocultar autoridades de reuniones…</a>
</div>

// Autoridades de título uniforme
<a href='{//tag[@name="130"]/reference}' target='_blank'
title='{//tag[@name="130"]/description}'>tag130</a>
<a href='{//tag[@name="130a"]/reference}' target='_blank'
title='{//tag[@name="130a"]/description}'>$a</a>
<input type='text' name='tag130a' id='tag130a' size='40' value=''/>
<a href='{//tag[@name="130o"]/reference}' target='_blank'
title='{//tag[@name="130o"]/description}'>$0</a>
<input type='text' name='tag130o' id='tag130o' size='12' value=''/>

<a href="javascript:ShowContent('listuniformtitle0')">ver autoridades de título uniforme</a>
<div id='listuniformtitle0' style='display: none;'>
<iframe src='authorities/uniformtitle.xml' width='100%' height='200px'></iframe><br/><a
href="javascript:HideContent('listuniformtitle0')">ocultar autoridades de título uniforme</a>
</div>

<a href="javascript:HideContent('1XXfields')">- campos de asiento principal</a>
</div>
```

Formulario de campos del área de título, edición y publicación

- Contiene los campos de título propiamente dicho, otras formas del título, mención de edición, lugar de publicación, editorial y fecha de publicación.

```
<a href="javascript:ShowContent('2XXfields')">+ campos de título, edición y publicación</a>
<div id='2XXfields' style='display: block;'>

// Campos del área de título
<a href='{//tag[@name="245"]/reference}' target='_blank'
title='{//tag[@name="245"]/description}'>tag245</a>
<a href='{//tag[@name="245a"]/reference}' target='_blank'
title='{//tag[@name="245a"]/description}'>$a</a>
<input type='text' name='tag245a' size='80' value=''/>
<a href='{//tag[@name="245b"]/reference}' target='_blank'
title='{//tag[@name="245b"]/description}'>$b</a>
<input type='text' name='tag245b' size='80' value=''/>

// Campos del área de edición
<a href='{//tag[@name="250"]/reference}' target='_blank'
title='{//tag[@name="250"]/description}'>tag250</a>
<a href='{//tag[@name="250a"]/reference}' target='_blank'
title='{//tag[@name="250a"]/description}'>$a</a>
<input type='text' name='tag250a' size='22' value=''/>
<a href='{//tag[@name="250b"]/reference}' target='_blank'
title='{//tag[@name="250b"]/description}'>$b</a>
<input type='text' name='tag250b' size='22' value=''/>

// Campos del área de publicación
<a href='{//tag[@name="260"]/reference}' target='_blank'
title='{//tag[@name="260"]/description}'>tag260</a>
<a href='{//tag[@name="260a"]/reference}' target='_blank'
title='{//tag[@name="260a"]/description}'>$a</a>
<input type='text' name='tag260a' id='tag260a' size='22' value=''/>

<a href="javascript:ShowContent('listplace0')">ver autoridades de lugar</a>
<a href='{//tag[@name="260b"]/reference}' target='_blank'
title='{//tag[@name="260b"]/description}'>$b</a>
<input type='text' name='tag260b' id='tag260b' size='22' value=''/>
<div id='listplace0' style='display: none;'>
<iframe src='authorities/place.xml' width='100%' height='200px'></iframe>
<a href="javascript:HideContent('listplace0')">ocultar autoridades de lugar</a>
</div>

<a href="javascript:ShowContent('listpublisher0')">ver autoridades de editoriales</a>
<a href='{//tag[@name="260c"]/reference}' target='_blank'
title='{//tag[@name="260c"]/description}'>$c</a>
<input type='text' name='tag260c' size='22' value=''/>
<div id='listpublisher0' style='display: none;'>
<iframe src='authorities/publisher.xml' width='100%' height='200px'></iframe>
<a href="javascript:HideContent('listpublisher0')">ocultar autoridades de editoriales</a>
</div>

<a href="javascript:HideContent('2XXfields')">- campos de título, edición y publicación</a>
</div>
```

Formulario de campos del área de descripción física

- Contiene campos de extensión, otros detalles físicos, dimensiones y materiales adjuntos.

```
<a href="javascript:ShowContent('3XXfields')">+ campos de descripción física</a>
<div id='3XXfields' style='display: block;'>

// Campos de descripción física
<a href='{//tag[@name="300"]/reference}' target='_blank'
title='{//tag[@name="300"]/description}'>tag300</a>

// Extensión
<a href='{//tag[@name="300a"]/reference}' target='_blank'
title='{//tag[@name="300a"]/description}'>$a</a>
<input type='text' name='tag300a' size='22' value=''/>

// Otros detalles de la descripción física como ilustraciones
<a href='{//tag[@name="300b"]/reference}' target='_blank'
title='{//tag[@name="300b"]/description}'>$b</a>
<input type='text' name='tag300b' size='22' value=''/>

// Dimensiones del documento
<a href='{//tag[@name="300c"]/reference}' target='_blank'
title='{//tag[@name="300c"]/description}'>$c</a>
<input type='text' name='tag300c' size='22' value=''/>

// Materiales adjuntos
<a href='{//tag[@name="300e"]/reference}' target='_blank'
title='{//tag[@name="300e"]/description}'>$e</a>
<input type='text' name='tag300e' size='22' value=''/>

<a href="javascript:HideContent('3XXfields')">- campos de descripción física</a>
</div>
```

Formulario de campos del área de serie

- Contiene los campos de título de la serie y número o volumen de la misma.

```
<a href="javascript:ShowContent('4XXfields')">+ campos de serie</a>
<div id='4XXfields' style='display: block;'>

<a href='{//tag[@name="490"]/reference}' target='_blank'
title='{//tag[@name="490"]/description}'>tag490</a>

// Título de la serie
<a href='{//tag[@name="490a"]/reference}' target='_blank'
title='{//tag[@name="490a"]/description}'>$a</a>
<input type='text' name='tag490a' size='22' value=''/>

// Número de la serie
<a href='{//tag[@name="490v"]/reference}' target='_blank'
title='{//tag[@name="490v"]/description}'>$v</a>
<input type='text' name='tag490v' size='22' value=''/>

<a href="javascript:HideContent('4XXfields')">- campos de serie</a>
</div>
```

Formulario del campo de notas

- Campo de nota general.

```
<a href="javascript:ShowContent('5XXfields')">+ campos de notas</a>
<div id='5XXfields' style='display: block;'>

<a href='{//tag[@name="500"]/reference}' target='_blank'
title='{//tag[@name="500"]/description}'>tag500</a>

<a href='{//tag[@name="500a"]/reference}' target='_blank'
title='{//tag[@name="500a"]/description}'>$a</a>
<textarea name='tag500a' cols='80' rows='8'></textarea>

<a href="javascript:HideContent('5XXfields')">- campos de notas</a>
```

Formulario de campos temáticos

- Lo conforman los puntos de acceso temáticos en las normas MARC, concretamente las etiquetas 650. En este caso son empleadas para almacenar los términos generales y específicos que se marquen en el tesauro.

```
<a href="javascript:ShowContent('6XXfields')">+ campos de acceso temático</a>
<div id='6XXfields' style='display: block;'>

<a href='{//tag[@name="650"]/reference}' target='_blank'
title='{//tag[@name="650"]/description}'>tag650</a>

<a href='{//tag[@name="650a"]/reference}' target='_blank'
title='{//tag[@name="650a"]/description}'>$a</a>

// Término general
<input type='text' name='tag650a1' id='tag650a1' size='82' value=''/>
// Término específico 1
<input type='text' name='tag650a2' id='tag650a2' size='82' value=''/>
// Término específico 2
<input type='text' name='tag650a3' id='tag650a3' size='82' value=''/>

<a href='{//tag[@name="650o"]/reference}' target='_blank'
title='{//tag[@name="650o"]/description}'>$0</a>
<input type='text' name='tag650o' id='tag650o' size='22' value=''/>

<a href="javascript:ShowContent('thesaurus')">ver tesauro</a>

<div id='thesaurus' style='display: none;'>
<iframe src='authorities/thesaurus.xml' width='100%' height='200px'></iframe>
<a href="javascript:HideContent('thesaurus')">ocultar tesauro</a>
</div>

<a href="javascript:HideContent('6XXfields')">- campos de acceso temático</a>
</div>
```

Formulario de campos de asientos secundarios

- Contiene los campos de asientos secundarios para autoridades personales y entidades corporativas. Su funcionamiento es análisis al apartado de asientos principales.

```
<a href="javascript:ShowContent('7XXfields')">+ campos de asiento secundario</a>
<div id='7XXfields' style='display: block;'>

// Asiento secundario de autoridad personal
<a href='{//tag[@name="700"]/reference}' target='_blank'
title='{//tag[@name="700"]/description}'>tag700</a>
<a href='{//tag[@name="700a"]/reference}' target='_blank'
title='{//tag[@name="700a"]/description}'>$a</a>
<input type='text' name='tag700a' id='tag700a' size='82' value=''/>
<a href='{//tag[@name="700o"]/reference}' target='_blank'
title='{//tag[@name="700o"]/description}'>$0</a>
<input type='text' name='tag700o' id='tag700o' size='22' value=''/>

<a href="javascript:ShowContent('listpersonal1')">ver autoridades personales</a>
<div id='listpersonal1' style='display: none;'>
<iframe src='authorities/personal.xml' width='100%' height='200px'></iframe>
<a href="javascript:HideContent('listpersonal1')">ocultar autoridades personales</a>
</div>

// Asiento secundario de autoridad corporativa
<a href='{//tag[@name="710"]/reference}' target='_blank'
title='{//tag[@name="710"]/description}'>tag710</a>
<a href='{//tag[@name="710a"]/reference}' target='_blank'
title='{//tag[@name="710a"]/description}'>$a</a>
<input type='text' name='tag710a' id='tag710a' size='82' value=''/>
<a href='{//tag[@name="710o"]/reference}' target='_blank'
title='{//tag[@name="710o"]/description}'>$0</a>
<input type='text' name='tag710o' id='tag710o' size='22' value=''/>

<a href="javascript:ShowContent('listcorporate1')">ver autoridades corporativas</a>
<div id='listcorporate1' style='display: none;'>
<iframe src='authorities/corporate.xml' width='100%' height='200px'></iframe>
<a href="javascript:HideContent('listcorporate1')">ocultar autoridades corporativas</a>
</div>

<a href="javascript:HideContent('7XXfields')">- campos de asiento secundario</a>
</div>
```

7.7.5. Tratamiento de las autoridades

Las autoridades en el sistema SYNCORE constituyen una parte esencial del servicio de edición. Esto se debe a que están integradas en los formularios de edición con el objetivo de facilitar la incorporación de datos normalizados, incluyendo no solo la denominación propiamente dicha, sino también el número de autoridad que les corresponde (JIMÉNEZ PELAYO, J. and García Blanco, R., 2002).

En las autoridades personales, corporativas, jornadas y congresos, título uniforme, lugares y editoriales, se ha empleado un mismo esquema o formato XML de descripción, inspirado en las normas MADS (Metadata Authority Description Schema: MADS, 2008), de las que se ha tomado la idea de organizar la descripción de las autoridades por medio de colecciones y la definición de los elementos según el tipo de autoridad. No comparten por otra parte la denominación de las etiquetas, puesto que se ha considerado que resultaba demasiado compleja y difícil de implementar en el sistema SYNCORE. En beneficio de una mejor legibilidad del formato de descripción de autoridades se han desarrollado todos los elementos principales comunes, tal como sucede para abrir una colección de autoridades <collection>, o para identificar una autoridad <authority> dentro de la colección. La forma de distinguir las autoridades de un tipo o de otro dentro de una misma colección es mediante el atributo *type*=" de la etiqueta <authority> que define valores predeterminados para la autoridad personal *personal*, la autoridad corporativa *corporate*, las jornadas y congresos con el valor *meeting*, los títulos uniformes con *uniformtitle*, los lugares utilizando *place* y las editoriales empleando *publisher*. Estas directrices de desarrollo pueden verse aplicadas en la *tabla140*.

Título: Esquemas de autoridades XML
Referencia: tabla140

Esquema de autoridad personal

```
<?xml version='1.0' encoding='UTF-8'?>
<?xml-stylesheet type='text/xsl' href='personal.xsl'?>
<collection>
<authority type='personal'>
        <identifier code='PER'>1</identifier>
        <fullname type='main'>Abbott, Joan</fullname>
        <date></date>
        <affiliation type='organization'></affiliation>
        <affiliation type='occupation'>Research Electro-Biology Phd</affiliation>
        <affiliation type='email'></affiliation>
        <affiliation type='website'></affiliation>
</authority>
<collection>
```

Esquema de autoridad corporativa

```
<?xml version='1.0' encoding='UTF-8'?>
<?xml-stylesheet type='text/xsl' href='corporate.xsl'?>

<collection>

<authority type='corporate'>
        <identifier code='CORP'>0</identifier>
        <fullname type='main'>Ministerio de Trabajo y Asuntos Sociales</fullname>
        <date></date>
        <affiliation type='organization'>España</affiliation>
        <affiliation type='occupation'></affiliation>
        <affiliation type='email'></affiliation>
        <affiliation type='website'></affiliation>
</authority>

<collection>
```

Esquema de autoridad de encuentros, jornadas y congresos

```
<?xml version='1.0' encoding='UTF-8'?>
<?xml-stylesheet type='text/xsl' href='meeting.xsl'?>

<collection>

<authority type='meeting'>
        <identifier code='MEE'>0</identifier>
        <fullname type='main'>X Jornadas de Biblioteconomía y Documentación</fullname>
        <date></date>
        <affiliation type='organization'>Facultad de Ciencias de la Documentación</affiliation>
        <affiliation type='occupation'></affiliation>
        <affiliation type='email'></affiliation>
        <affiliation type='website'></affiliation>
</authority>

<collection>
```

Esquema de autoridad de título uniforme

```
<?xml version='1.0' encoding='UTF-8'?>
<?xml-stylesheet type='text/xsl' href='uniformtitle.xsl'?>

<collection>

<authority type='uniformtitle'>
        <identifier code='UTI'>0</identifier>
        <fullname type='main'>Boletín Oficial del Estado</fullname>
</authority>

<collection>
```

Esquema de autoridades de lugar, geográficas

```
<?xml version='1.0' encoding='UTF-8'?>
<?xml-stylesheet type='text/xsl' href='place.xsl'?>

<collection>

<authority type='place'>
        <identifier code='PLA'>0</identifier>
        <fullname type='main'>Madrid</fullname>
        <affiliation type='geographic'>España</affiliation>
</authority>

<collection>
```

Esquema de autoridades editoriales

```
<?xml version='1.0' encoding='UTF-8'?>
<?xml-stylesheet type='text/xsl' href='publisher.xsl'?>

<collection>

<authority type='publisher'>
        <identifier code='PUB'>0</identifier>
        <fullname type='main'>Anaya</fullname>
        <date></date>
        <affiliation type='organization'></affiliation>
        <affiliation type='occupation'></affiliation>
        <affiliation type='email'></affiliation>
        <affiliation type='website'></affiliation>
</authority>

<collection>
```

Como se puede comprobar en la *tabla140*, según el tipo de autoridad se emplea una serie de etiquetas de descripción u otras, pero en todo caso resultan en su mayor parte comunes y de fácil identificación. El formato de autoridades desarrollado, se puede consultar en la *tabla141*.

En cuanto al modelo de archivo XSL para la representación de las autoridades, se procuró introducir un método de transmisión de datos entre formularios, de forma tal que el formulario principal del servicio de edición captara la forma normalizada de la autoridad correspondiente y su número de registro e identificación. Para hacer posible este proceso se embebió un código Javascript en todos los enlaces de las autoridades, que a modo de evento *onclick* permite transmitir a un campo identificado en el formulario principal un determinado valor, correspondiente a la autoridad, todo se expresa con una sentencia similar a:

autoridad

El resultado de aplicar esta técnica puede comprobarse en todos los archivos XSL de autoridades y tesauros.

Título: Modelo de archivo XSL para la representación de autoridades
Referencia: tabla141

```
<?xml version='1.0' encoding='UTF-8'?>
<xsl:stylesheet version='1.0' xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
<xsl:template match='/'>

<html>
<head>
        <title>modelo de archivo XSL de autoridades</title>
        <meta http-equiv='content-type' content='text/html; charset=UTF-8'/>
        <link rel='stylesheet' href='../style.css' type='text/css'/>
</head>
<body>

<table>
<tr><th>id</th><th>fullname</th><th>info</th><th>X</th><th>Y</th></tr>
<xsl:for-each select='//collection/authority'>
<tr><th><xsl:value-of select='identifier'/></th>
<th><xsl:value-of select='fullname'/></th>
<th><a href='#' title='Organización: {affiliation[@type="organization"]} .- Ocupación:
{affiliation[@type="occupation"]}'>+ info</a></th>

<th><a href='#' onclick="parent.document.getElementById('tagZZZa').value='{fullname}';
parent.document.getElementById('tagZZZo').value='{identifier}'; return false;">asignar entrada
principal</a></th>

<th><a href='#' onclick="parent.document.getElementById('tagYYYa').value+='{fullname} ; ';
parent.document.getElementById('tagYYYo').value+='{identifier} ; '; return false;">asignar entrada
secundaria</a></th></tr>

</xsl:for-each>
</table>

</body>
</html>

</xsl:template>
</xsl:stylesheet>
```

Además de las autoridades expuestas, también se han considerado las autoridades temáticas, aprovechando para la ocasión el desarrollo de un tesauro básico en formato XML a efectos de poder probar la incorporación de lenguajes documentales al canal de sindicación. Para hacer dicha prueba se ha tomado una sección del tesauro EUROVOC (Eurovoc Thesaurus, 2008) para representarla en el tesauro de SYNCORE, *véase tabla 142*.

Título: Tesauro XML de prueba utilizado en SYNCORE
Referencia: tabla142

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type='text/xsl' href='thesaurus.xsl'?>

<thesaurus>

…

<term type='cat1'><id>1</id><name>derecho civil</name>
	<term type='cat2'><id>1</id><name>abuso del derecho</name></term>
	<term type='cat2'><id>2</id><name>contrato</name>
		<term type='cat3'><id>1</id><name>cláusula contractual</name></term>
		<term type='cat3'><id>2</id><name>cláusula abusiva</name></term>
		<term type='cat3'><id>3</id><name>cláusula compromisoria</name></term>
		<term type='cat3'><id>4</id><name>garantía</name></term>
		<term type='cat3'><id>5</id><name>solvencia</name></term>
		<term type='cat3'><id>6</id><name>hipoteca</name></term>
		<term type='cat3'><id>7</id><name>mandato</name></term>
		<term type='cat3'><id>8</id><name>recesión contractual</name></term>
	</term>

	<term type='cat2'><id>3</id><name>derecho de las obligaciones</name>
		<term type='cat3'><id>9</id><name>créditos por cobrar</name></term>
		<term type='cat3'><id>10</id><name>deuda</name></term>
	</term>

	<term type='cat2'><id>4</id><name>estatuto jurídico</name>
		<term type='cat3'><id>11</id><name>capacidad jurídica</name></term>
		<term type='cat3'><id>12</id><name>disfrute de los derechos</name></term>
		<term type='cat3'><id>13</id><name>domicilio legal</name></term>
		<term type='cat3'><id>14</id><name>estado civil</name></term>
		<term type='cat3'><id>15</id><name>persona física</name></term>
		<term type='cat3'><id>16</id><name>persona jurídica</name></term>
	</term>

	<term type='cat2'><id>5</id><name>responsabilidad</name>
		<term type='cat3'><id>17</id><name>daño</name></term>
		<term type='cat3'><id>18</id><name>indemnización</name></term>
		<term type='cat3'><id>19</id><name>responsabilidad civil</name></term>
		<term type='cat3'><id>20</id><name>responsab. contractual</name></term>
		<term type='cat3'><id>21</id><name>responsab. fabricante</name></term>
	</term>
</term>

<term type='cat1'><id>2</id><name>ciencia del espacio</name>
	<term type='cat2'><id>6</id><name>astronáutica</name></term>
	<term type='cat2'><id>7</id><name>astronomía</name></term>
	<term type='cat2'><id>8</id><name>cosmología</name></term>
</term>

…

</thesaurus>
```

Uno de los aspectos más difíciles de desarrollar en relación al tesauro fue el archivo XSL que permite visualizar todos los términos manteniendo la correspondiente jerarquía elaborada en el formato. Para ello se tuvo que recurrir a la instrucción *<xsl:if test='condición'>* para expresar una condición que permitiera mostrar los términos de cada nivel definidos en el atributo *type='cat1, cat2, cat3'*. Seleccionados los términos de cada nivel se representarían normalmente mediante la instrucción *<xsl:value-of select='name'/>*. Aún así se encontraron problemas en la hoja XSL a la hora de representar los valores propios de los elementos padre y abuelo de los términos de tercer nivel jerárquico. Esta necesidad responde a que siempre que se asignan una serie de temáticas del tesauro, siempre se rellenan los tres campos correspondientes a los tres niveles jerárquicos del formulario principal de edición. Esto implica que de utilizar un término de segundo o tercer nivel jerárquico, será necesario recordar sus niveles superiores para asegurar la introducción de todos los elementos en el formulario principal. La solución a este problema fue la utilización de sentencias XPath del tipo *parent::term/name* y *ancestor::term/name* que recuperan el nombre del término padre o del ancestro o abuelo cuya estructura coincida con *term/name*.

Título: Archivo XSL para la representación del tesauro XML
Referencia: tabla143

```
<?xml version='1.0' encoding='UTF-8'?>
<xsl:stylesheet version='1.0' xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
<xsl:template match='/'>

<html>
<head>

<title>tesauro</title>
<meta http-equiv='content-type' content='text/html; charset=UTF-8'/>
<link rel='stylesheet' href='../style.css' type='text/css'/>

</head>
<body>

<ul>
<xsl:for-each select='//term'>

<xsl:if test='@type="cat1"'>
<li><xsl:value-of select='name'/> -
<a href='#' onclick="
parent.document.getElementById('tag650a1').value='{name}';
parent.document.getElementById('tag650a2').value='';
parent.document.getElementById('tag650a3').value='';
parent.document.getElementById('tag650o').value='{id};0;0';
return false;">asignar</a>
</li>
</xsl:if>
```

```
<xsl:if test='@type="cat2"'>
<ul>
<li><xsl:value-of select='name'/> -
<a href='#' onclick="
parent.document.getElementById('tag650a1').value='{ancestor::term/name}';
parent.document.getElementById('tag650a2').value='{name}';
parent.document.getElementById('tag650a3').value='';
parent.document.getElementById('tag650o').value='{ancestor::term/id};{id};0';
return false;">asignar</a>
</li>
</ul>
</xsl:if>

<xsl:if test='@type="cat3"'>
<ul><ul>
<li><xsl:value-of select='name'/> -
<a href='#' onclick="
parent.document.getElementById('tag650a1').value='{ancestor::term/name}';
parent.document.getElementById('tag650a2').value='{parent::term/name}';
parent.document.getElementById('tag650a3').value='{name}';
parent.document.getElementById('tag650o').value='{ancestor::term/id};{parent::term/id};{id}'; return
false;">asignar</a></li>
</ul></ul>
</xsl:if>

</xsl:for-each>
</ul>
</body>
</html>

</xsl:template>
</xsl:stylesheet>
```

7.7.6. Servicio de búsqueda

El servicio de búsqueda se ha desarrollado pensando en demostrar las posibilidades de aplicar la sindicación de forma integral desde el formulario de consulta, hasta la presentación de los resultados y su selección.

La primera operación que se lleva a cabo es el despliege del formulario de consulta, que consta de dos elementos, por un lado el selector de colecciones disponibles y por otro lado la caja de texto para escribir las consultas. El sistema está diseñado para buscar en una colección concreta que el usuario especifique. No obstante esta característica puede ser mejorada fácilmente, de forma que elabore una consulta secuencial sobre todos los catálogos disponibles en la plataforma SYNC. A su vez la automatización de consultas en múltiples catálogos de múltiples plataformas SYNC también sería posible. Siendo conscientes de esta

limitación el objetivo que persigue SYNCORE es demostrar la posibilidad de funcionar en modo búsqueda empleando canales de sindicación y así como cualquier otro formato XML.

Una vez enviada la consulta, ésta se resuelve a través de syncore.php, mediante las cabeceras del servicio y ejecuta en *service.search.php*, la función de resolución de la consulta. Para ello efectúa una búsqueda de tipo *FULLTEXT IN BOOLEAN MODE* sobre el campo de indexación de la colección. El resultado de la consulta es la regeneración del canal de sindicación de servicios y respuestas, de forma tal que el archivo *response.xml*, contiene todos los registros bibliográficos resultantes, en formato MARC-XML. Además se genera un archivo *response.xsl* que permite preparar una representación de los resultados de la consulta, expresados ahora en el formato MARC-XML. La visualización de los registros ha sido preparada para mostrar únicamente los campos principales, su número de control y algunas opciones que permiten al usuario marcar los registros que considere relevantes para guardarlos posteriormente en un nuevo canal de sindicación, lo que a la postre permite la exportación y redifusión de los contenidos seleccionados. De hecho una vez que el usuario recibe la respuesta de la consulta, ya dispone de los resultados completos para su sindicación y exportación puesto que la URL del canal de respuesta contiene el archivo MARC-XML, que en el caso de un entorno local es http://localhost/syncore/response.xml

Otro aspecto desarrollado en este formulario es la posibilidad de filtrar los resultados mediante términos o frases expresadas en una caja de filtrado que también aparece al comienzo de la pantalla. Esta opción posibilita remitir tales términos para aplicar un filtro sobre el archivo XSL, empleado para representar los contenidos de los resultados. Esto significa que no efectúa cambio alguno en el archivo *response.xml,* de forma que los resultados anteriormente presentados nunca se pierden, aunque el usuario obtenga un listado más reducido que el original. Este mecanismo permite la reversibilidad de los procesos de filtrado, tantas veces como fueren necesarios.

A continuación se expone en la *tabla144*, el código fuente empleado para el servicio de búsquedas de SYNCORE.

Título: Código del servicio de búsqueda de SYNCORE
Archivo: service.search.php
Referencia: tabla144

Función de despliege del formulario de consulta

- La primera operación que se lleva a cabo cuando se demanda el servicio de búsqueda es el despliege del formulario de consulta que está compuesto por cuadro de selección de colecciones disponibles sobre las que realizar la búsqueda, y por otro lado, la adopción de caja de texto para escribir la cadena de caracteres frases o palabras que se utilizarán para la recuperación de registros bibliográficos.

```
// FUNCIÓN DE DESPLIEGE DEL FORMULARIO DE CONSULTA

switch($_POST[id]) {

case 'S2M1':

// Generación del canal de respuesta response.xml

$file = fopen('response.xml','w');
fwrite ($file,"<?xml version='1.0' encoding='UTF-8'?>
<?xml-stylesheet type='text/xsl' href='response.xsl'?>
<sync>
<response>
<message>servicio de búsqueda</message>
<response>
</sync>");
fclose($file);

// Generación del archivo XSL del canal de respuesta

$file = fopen('response.xsl','w');
fwrite ($file,"<?xml version='1.0' encoding='UTF-8'?>
<xsl:stylesheet version='1.0' xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
<xsl:template match='/'>
<html>
<head>
<title>syncore response channel</title>
<meta http-equiv='content-type' content='text/html; charset=UTF-8'/>
<link rel='stylesheet' href='style.css' type='text/css'/>
<script type='text/javascript' language='javascript'>
function HideContent(d) { document.getElementById(d).style.display = 'none'; }
function ShowContent(d) { document.getElementById(d).style.display = 'block'; }
function ReverseDisplay(d) {
if(document.getElementById(d).style.display == 'none') { document.getElementById(d).style.display
= 'block'; }
else { document.getElementById(d).style.display = 'none'; } }
</script>
</head>
<body>
<p><h3>Búsqueda</h3></p>
<form action='$url' method='post'>
<input type='hidden' name='id' value='$idservice'/>
<input type='hidden' name='file' value='$fileservice'/>
<input type='hidden' name='launch' value='launch'/>
<select name='coldestiny'>");
```

```
// Edición del formulario de búsqueda: Caja de selección de colección
mysql_select_db("$database", $con);
$sql1= "SHOW TABLE STATUS FROM $database WHERE Name REGEXP '^col.*.marc1$'";
$resultado1 = mysql_query($sql1);
while($array = mysql_fetch_array($resultado1)) {
        fwrite ($file,"<option value='$array[Name]'>$array[Name]</option>");
}

fwrite ($file,"</select>
// Edición del formulario de búsqueda: Caja de consulta
<input type='text' name='searchgeneral' size='55'/>
<input type='submit' name='search1' value='buscar'/>
</form>
<body>
</html>
</xsl:template>
</xsl:stylesheet>");

fclose($file);
echo "<script language='javascript'>window.parent.location.href='service.xml'</script>";
break;
}
```

Función de ejecución de consulta

- Cuando el usuario remite la consulta, se ejecuta y procesa con el siguiente script, que efectúa una búsqueda en el catálogo bibliográfico utilizando la función SQL FULLTEXT IN BOOLEAN MODE, de tipo booleano sobre el campo de indexación de la base datos. El resultado es la regeneración del canal de respuestas esta vez con los resultados de la búsqueda en formato MARC-XML, sobre los que se puede interactuar mediante filtros XPath y opciones de selección o marcado para su posterior exportación o sindicación.

```
// FUNCIÓN DE EJECUCIÓN DE CONSULTA

if($_POST[search1]) {


$file = fopen('response.xml','w');
fwrite ($file,"<?xml version='1.0' encoding='UTF-8'?>
<?xml-stylesheet type='text/xsl' href='response.xsl'?>
<collection>");

// Se efectúa una consulta SQL IN BOOLEAN MODE, con los términos de búsqueda del
// usuario $_POST[searchgeneral], así como en la colección elegida a todos los efectos
// $_POST[coldestiny].

mysql_select_db("$database", $con);
$sql = "SELECT * FROM $_POST[coldestiny] WHERE MATCH(indexer) AGAINST
('%$_POST[searchgeneral]%' IN BOOLEAN MODE) LIMIT 0,100";
$resultado = mysql_query($sql,$con);
while ($fila = mysql_fetch_assoc($resultado)) {

// Generación del canal de respuestas en formato MARC-XML con los resultados de la
// consulta
```

```
fwrite ($file,"
<record>
<leader>$fila[leader]</leader>
<controlfield tag='001'>$fila[tag001]</controlfield>
<controlfield tag='003'>$fila[tag003]</controlfield>
<controlfield tag='005'>$fila[tag005]</controlfield>

<datafield tag='010' ind1=" ind2="">
<subfield code='a'>$fila[tag010a]</subfield>
</datafield>

<datafield tag='017' ind1=" ind2="">
<subfield code='a'>$fila[tag017a]</subfield>
</datafield>

<datafield tag='020' ind1=" ind2="">
<subfield code='a'>$fila[tag020a]</subfield>
</datafield>

<datafield tag='041' ind1=" ind2="">
<subfield code='a'>$fila[tag041a]</subfield>
</datafield>

<datafield tag='050' ind1='1' ind2='0'>
<subfield code='a'>$fila[tag050a]</subfield>
</datafield>

<datafield tag='080' ind1=" ind2="">
<subfield code='a'>$fila[tag080a]</subfield>
</datafield>

<datafield tag='082' ind1='0' ind2="">
<subfield code='a'>$fila[tag082a]</subfield>
</datafield>

<datafield tag='100' ind1='0' ind2="">
<subfield code='a'>$fila[tag100a]</subfield>
<subfield code='0'>$fila[tag100o]</subfield>
</datafield>

<datafield tag='110' ind1='2' ind2="">
<subfield code='a'>$fila[tag110a]</subfield>
<subfield code='0'>$fila[tag110o]</subfield>
</datafield>

<datafield tag='111' ind1='2' ind2="">
<subfield code='a'>$fila[tag111a]</subfield>
<subfield code='0'>$fila[tag111o]</subfield>
</datafield>

<datafield tag='130' ind1='0' ind2="">
<subfield code='a'>$fila[tag130a]</subfield>
<subfield code='0'>$fila[tag130o]</subfield>
</datafield>

<datafield tag='245' ind1='1' ind2='0'>
<subfield code='a'>$fila[tag245a]</subfield>
<subfield code='b'>$fila[tag245b]</subfield>
<subfield code='c'>$fila[tag245c]</subfield>
</datafield>
```

```
<datafield tag='250' ind1='' ind2=''>
<subfield code='a'>$fila[tag250a]</subfield>
<subfield code='b'>$fila[tag250b]</subfield>
</datafield>

<datafield tag='260' ind1='' ind2=''>
<subfield code='a'>$fila[tag260a]</subfield>
<subfield code='b'>$fila[tag260b]</subfield>
<subfield code='c'>$fila[tag260c]</subfield>
</datafield>

<datafield tag='300' ind1='' ind2=''>
<subfield code='a'>$fila[tag300a]</subfield>
<subfield code='b'>$fila[tag300b]</subfield>
<subfield code='c'>$fila[tag300c]</subfield>
<subfield code='e'>$fila[tag300e]</subfield>
</datafield>

<datafield tag='490' ind1='0' ind2=''>
<subfield code='a'>$fila[tag490a]</subfield>
<subfield code='v'>$fila[tag490v]</subfield>
</datafield>

<datafield tag='500' ind1='' ind2=''>
<subfield code='a'>$fila[tag500a]</subfield>
</datafield>

<datafield tag='650' ind1='1' ind2='4'>
<subfield code='a'>$fila[tag650a]</subfield>
<subfield code='0'>$fila[tag650o]</subfield>
</datafield>

<datafield tag='700' ind1='0' ind2=''>
<subfield code='a'>$fila[tag700a]</subfield>
<subfield code='0'>$fila[tag700o]</subfield>
</datafield>

<datafield tag='710' ind1='2' ind2=''>
<subfield code='a'>$fila[tag710a]</subfield>
<subfield code='0'>$fila[tag710o]</subfield>
</datafield>

<datafield tag='887' ind1='' ind2=''>
<subfield code='a'>$fila[tag887a]</subfield>
</datafield>
</record>
");
}

fwrite ($file,"</collection>");
fclose($file);

// Generación del archivo XSL correspondiente al canal de respuestas.
$file = fopen('response.xsl','w');
fwrite ($file,"<?xml version='1.0' encoding='UTF-8'?>
<xsl:stylesheet version='1.0' xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
<xsl:template match='/'>
<html>
<head>
```

```
<title>syncore response channel</title>
<meta http-equiv='content-type' content='text/html; charset=UTF-8'/>
<link rel='stylesheet' href='style.css' type='text/css'/>

// Función Javascript para mostrar y ocultar capas <div></div>
<script type='text/javascript' language='javascript'>
function HideContent(d) { document.getElementById(d).style.display = 'none'; }
function ShowContent(d) { document.getElementById(d).style.display = 'block'; }
function ReverseDisplay(d) { if(document.getElementById(d).style.display == 'none') {
document.getElementById(d).style.display = 'block'; } else {
document.getElementById(d).style.display = 'none'; } }
</script>

</head>
<body>

// Cuadro de filtrado con método XPath para refinar los resultados de la consulta efectuada.
<form action='$url' method='post'>
<input type='text' name='filterm' size='22'/>
<input type='submit' name='filter' value='Filtrar Resultados'/>
<input type='hidden' name='varid' id='varid' size='42'/>

// Botón para guardar la selección de resultados marcados por el usuario.
<div style='height: 20px; width: 100%; background-color: #FFFFFF;'></div>
<div style='width: 100%; height: 30px; background-color: #eeeeee; text-align: center;'>
<input type='submit' name='selection' value='guardar selección'/>
</div>

// Inicio del formateado de los resultados en XSL
<div style='height: 20px; width: 100%; background-color: #FFFFFF;'></div>
<xsl:for-each select='//collection/record'> // Selección del registro
<div class='block'>

<input type='hidden' name='id' id='idserv' value='$idservice'/>
<input type='hidden' name='file' id='fileserv' value='$fileservice'/>
<input type='hidden' name='coldestiny' value='$coldestiny'/>
<input type='hidden' name='launch' value='launch'/>

// Generador automático de identificadores de recursos en XSL , empleado para guardar en
// sesión autónoma del navegador web, los códigos de los registros seleccionados.
<div style='display: table-cell; padding: 5px;'>
<input type='hidden' name='{generate-id()}' id='{generate-id()}'/></div>

// Visualización del número de control de la biblioteca.
<div style='display: table-cell; padding: 5px;'>
<xsl:value-of select='controlfield[@tag=\"001\"]'/></div>

// Visualización del identificador del centro catalogador.
<div style='display: table-cell; padding: 5px;'>
<xsl:value-of select='controlfield[@tag=\"003\"]'/></div>

// Representación del registro bibliográfico mediante la selección de los campos principales.
<div style='display: table-cell; padding: 5px;'>
<xsl:value-of select='datafield[@tag=\"245\"]/subfield[@code=\"a\"]'/> :
<xsl:value-of select='datafield[@tag=\"245\"]/subfield[@code=\"b\"]'/> /
<xsl:value-of select='datafield[@tag=\"245\"]/subfield[@code=\"c\"]'/> .-
<xsl:value-of select='datafield[@tag=\"260\"]/subfield[@code=\"a\"]'/> :
<xsl:value-of select='datafield[@tag=\"260\"]/subfield[@code=\"b\"]'/> ,
<xsl:value-of select='datafield[@tag=\"260\"]/subfield[@code=\"c\"]'/>
</div>
```

```
// Mecanismo de selección del registro bibliográfico por su identificador XSL.
// Cuando se marca por primera vez un registro aparece una capa de color verde
// Cuando se desmarca, el color varía a rojo.
<a href='#' onclick=\"javascript:ShowContent('DIV{generate-id()}');
self.document.getElementById('varid').value+='{generate-id()};';
self.document.getElementById('{generate-id()}').value='{controlfield[@tag=001]}'; return
false;\">marcar</a>
<div id='DIV{generate-id()}' style='display: none; width: 100%; height: 20px; background-color:
#40ff40;'> registro <xsl:value-of select='generate-id()'/> marcado

// Botón de desmarcar registro.
<a href='#' onclick=\"javascript:HideContent('DIV{generate-id()}');
self.document.getElementById('{generate-id()}').value='X'; return false;\">desmarcar</a>
 </div>

</div>

</xsl:for-each>

// Botón complementario para guardar los registros bibliográficos que se han seleccionado.
<div style='width: 100%; height: 30px; background-color: #eeeeee; text-align: center;'>
<input type='submit' name='selection' value='guardar selección'/>
</div>
</form>
<body></html>
</xsl:template></xsl:stylesheet>");
fclose($file);

// Actualización del canal de servicios completo para visualizar los cambios efectuados.
echo "<script language='javascript'>window.parent.location.href='$url?func=terminate2'
</script>";
die();
}
```

Función de filtrado

- La función de filtrado actúa cuando se seleccionan los registros bibliográficos. Esta operación supone únicamente la modificación del archivo XSL del canal de respuestas, que adopta un filtro establecido por el usuario mediante sintaxis XPath. Dicho proceso de filtrado se efectúa en el campo 887 de MARC-XML que incluye todos los puntos de acceso a cada documento, permitiendo de esta forma simplificar el proceso de recuperación.

```
// FUNCIÓN DE FILTRADO
// Permite filtrar los resultados de la consulta a partir de un término utilizando XPATH

if($_POST[filter]) {

$file = fopen('response.xsl','w');
fwrite ($file,"<?xml version='1.0' encoding='UTF-8'?>
<xsl:stylesheet version='1.0' xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
<xsl:template match='/'>
<html>
<head>
<title>syncore response channel</title>
<meta http-equiv='content-type' content='text/html; charset=UTF-8'/>
<link rel='stylesheet' href='style.css' type='text/css'/>
```

```
// Función Javascript para mostrar y ocultar capas <div></div>
<script type='text/javascript' language='javascript'>
function HideContent(d) { document.getElementById(d).style.display = 'none'; }
function ShowContent(d) { document.getElementById(d).style.display = 'block'; }
function ReverseDisplay(d) {
if(document.getElementById(d).style.display == 'none') { document.getElementById(d).style.display
= 'block'; }
else { document.getElementById(d).style.display = 'none'; } }
</script>

</head>
<body>

// Cuadro de filtrado con método XPath para refinar los resultados de la consulta efectuada.
<form action='$url' method='post'>
<input type='text' name='filterm' size='22'/>
<input type='submit' name='filter' value='Filtrar Resultados'/>
<input type='hidden' name='varid' id='varid' size='42'/>

// Botón para guardar la selección de resultados marcados por el usuario.
<div style='height: 20px; width: 100%; background-color: #FFFFFF;'></div>
<div style='width: 100%; height: 30px; background-color: #eeeeee; text-align: center;'>
<input type='submit' name='selection' value='guardar selección'/>
</div>

// Botón de desmarcar registro.
<xsl:for-each select='//collection/record'>
<xsl:if test=\"contains(datafield[@tag='887']/subfield[@code='a'],'$_POST[filterm]')\">
<div class='block'>

// Inicio del formateado de los resultados en XSL
<input type='hidden' name='id' id='idserv' value='$idservice'/>
<input type='hidden' name='file' id='fileserv' value='$fileservice'/>
<input type='hidden' name='coldestiny' value='$coldestiny'/>
<input type='hidden' name='launch' value='launch'/>

// Generador automático de identificadores de recursos en XSL , empleado para guardar en
// sesión autónoma del navegador web, los códigos de los registros seleccionados.
<div style='display: table-cell; padding: 5px;'>
<input type='text' name='{generate-id()}' id='{generate-id()}'/>
</div>

// Visualización del número de control de la biblioteca.
<div style='display: table-cell; padding: 5px;'>
<xsl:value-of select='controlfield[@tag=\"001\"]'/>
</div>

// Visualización del identificador del centro catalogador.
<div style='display: table-cell; padding: 5px;'>
<xsl:value-of select='controlfield[@tag=\"003\"]'/>
</div>

// Representación del registro bibliográfico mediante la selección de los campos principal.
<div style='display: table-cell; padding: 5px; background-color: #eeeeee;'>
<xsl:value-of select='datafield[@tag=\"245\"]/subfield[@code=\"a\"]'/> :
<xsl:value-of select='datafield[@tag=\"245\"]/subfield[@code=\"b\"]'/> /
<xsl:value-of select='datafield[@tag=\"245\"]/subfield[@code=\"c\"]'/> .-
<xsl:value-of select='datafield[@tag=\"260\"]/subfield[@code=\"a\"]'/> :
<xsl:value-of select='datafield[@tag=\"260\"]/subfield[@code=\"b\"]'/> ,
```

```
<xsl:value-of select='datafield[@tag=\"260\"]/subfield[@code=\"c\"]'/>
</div>

// Mecanismo de selección del registro bibliográfico por su identificador XSL.
// Cuando se marca por primera vez un registro aparece una capa de color verde
// Cuando se desmarca, el color varía a rojo.
<a href='#' onclick=\"javascript:ShowContent('DIV{generate-id()}');
self.document.getElementById('varid').value+='{generate-id()};';
self.document.getElementById('{generate-id()}').value='{controlfield[@tag=001]}'; return
false;\">marcar</a>
<div id='DIV{generate-id()}' style='display: none; width: 100%; height: 20px; background-color:
#40ff40;'> registro <xsl:value-of select='generate-id()'/> marcado

// Botón de desmarcar registro.
<a href='#' onclick=\"javascript:HideContent('DIV{generate-id()}');
self.document.getElementById('{generate-id()}').value='X'; return false;\">desmarcar</a>
</div>

</xsl:if>
</xsl:for-each>

// Botón complementario para guardar los registros bibliográficos que se han seleccionado.
<div style='width: 100%; height: 30px; background-color: #eeeeee; text-align: center;'>
<input type='submit' name='selection' value='guardar selección'/>
</div>

</form>
</body></html></xsl:template></xsl:stylesheet>");

fclose($file);
// Actualización del canal de servicios completo para visualizar los cambios efectuados.
echo "<script language='javascript'>window.parent.location.href='$url?func=terminate2
</script>
";
die();
}
```

Función de selección

- La función de selección se encarga de guardar los registros bibliográficos que han sido marcados para su posterior exportación y sindicación. Ello se consigue recuperando las variables codificadas por el generador de identificadores XSL y obteniendo el número de control de los registros bibliográficos marcados. El resultado de este proceso es la regeneración del canal de respuesta *response.xml* con los resultados que se habían seleccionado en formato MARC-XML.

```
// FUNCIÓN DE SELECCIÓN.

if($_POST[selection]) {

// Proceso de extracción de variables codificadas por el generador de identificadores XSL.
$stringpost = $_POST[varid];
$v1 = split(";", $stringpost);
$nstr = count($v1);

// Generación del canal de respuestas
$file = fopen('response.xml','w');
fwrite($file,"<?xml version='1.0' encoding='UTF-8'?>
```

```
<?xml-stylesheet type='text/xsl' href='response.xsl'?>
<collection>");

// Extracción de los números de control de los registros bibliográficos seleccionados.
for ($i=0; $i<$nstr; $i++) {
if($v1[$i] != '') {
$postvar[0] = trim($v1[$i]);
$varpost = $postvar[0];
$numvar = "$_POST[$varpost]";

// Control de registros desmarcados
if($numvar == 'X') {} else {
// Consulta de registros marcados
mysql_select_db("$database", $con);
$sql = "SELECT * FROM $_POST[coldestiny] WHERE tag001 = '$numvar' ORDER BY tag005
DESC";
$resultado = mysql_query($sql,$con);

// Escritura de los registros en formato MARC-XML
while ($fila = mysql_fetch_assoc($resultado)) {
fwrite($file,"
<record>
<leader>$fila[leader]</leader>
<controlfield tag='001'>$fila[tag001]</controlfield>
<controlfield tag='003'>$fila[tag003]</controlfield>
<controlfield tag='005'>$fila[tag005]</controlfield>
<datafield tag='010' ind1=" ind2=""><subfield code='a'>$fila[tag010a]</subfield></datafield>
<datafield tag='017' ind1=" ind2=""><subfield code='a'>$fila[tag017a]</subfield></datafield>
<datafield tag='020' ind1=" ind2=""><subfield code='a'>$fila[tag020a]</subfield></datafield>
<datafield tag='041' ind1=" ind2=""><subfield code='a'>$fila[tag041a]</subfield></datafield>
<datafield tag='050' ind1='1' ind2='0'><subfield code='a'>$fila[tag050a]</subfield></datafield>
<datafield tag='080' ind1=" ind2=""><subfield code='a'>$fila[tag080a]</subfield></datafield>
<datafield tag='082' ind1='0' ind2=""><subfield code='a'>$fila[tag082a]</subfield></datafield>

<datafield tag='100' ind1='0' ind2=''>
<subfield code='a'>$fila[tag100a]</subfield>
<subfield code='0'>$fila[tag100o]</subfield>
</datafield>

<datafield tag='110' ind1='2' ind2=''>
<subfield code='a'>$fila[tag110a]</subfield>
<subfield code='0'>$fila[tag110o]</subfield>
</datafield>

<datafield tag='111' ind1='2' ind2=''>
<subfield code='a'>$fila[tag111a]</subfield>
<subfield code='0'>$fila[tag111o]</subfield>
</datafield>

<datafield tag='130' ind1='0' ind2=''>
<subfield code='a'>$fila[tag130a]</subfield>
<subfield code='0'>$fila[tag130o]</subfield>
</datafield>

<datafield tag='245' ind1='1' ind2='0'>
<subfield code='a'>$fila[tag245a]</subfield>
<subfield code='b'>$fila[tag245b]</subfield>
<subfield code='c'>$fila[tag245c]</subfield>
</datafield>
```

```
<datafield tag='250' ind1='' ind2=''>
<subfield code='a'>$fila[tag250a]</subfield>
<subfield code='b'>$fila[tag250b]</subfield>
</datafield>

<datafield tag='260' ind1='' ind2=''>
<subfield code='a'>$fila[tag260a]</subfield>
<subfield code='b'>$fila[tag260b]</subfield>
<subfield code='c'>$fila[tag260c]</subfield>
</datafield>

<datafield tag='300' ind1='' ind2=''>
<subfield code='a'>$fila[tag300a]</subfield>
<subfield code='b'>$fila[tag300b]</subfield>
<subfield code='c'>$fila[tag300c]</subfield>
<subfield code='e'>$fila[tag300e]</subfield>
</datafield>

<datafield tag='490' ind1='0' ind2=''>
<subfield code='a'>$fila[tag490a]</subfield>
<subfield code='v'>$fila[tag490v]</subfield>
</datafield>

<datafield tag='500' ind1='' ind2=''>
<subfield code='a'>$fila[tag500a]</subfield>
</datafield>

<datafield tag='650' ind1='1' ind2='4'>
<subfield code='a'>$fila[tag650a]</subfield>
<subfield code='0'>$fila[tag650o]</subfield>
</datafield>

<datafield tag='700' ind1='0' ind2=''>
<subfield code='a'>$fila[tag700a]</subfield>
<subfield code='0'>$fila[tag700o]</subfield>
</datafield>

<datafield tag='710' ind1='2' ind2=''>
<subfield code='a'>$fila[tag710a]</subfield>
<subfield code='0'>$fila[tag710o]</subfield>
</datafield>

<datafield tag='887' ind1='' ind2=''>
<subfield code='a'>$fila[tag887a]</subfield>
</datafield>
</record>");
}
}
} else {}
}
fwrite($file,"</collection>");

fclose($file);
// Actualización del canal de servicios completo para visualizar los cambios efectuados.
echo "<script language='javascript'>window.parent.location.href='$url?func=terminate2'
</script>";
die();
}
```

7.7.7. Servicio de ordenación de la colección

El servicio de ordenación de la colección ha sido desarrollado para probar el proceso de ordenación y sindicación completa de la colección bibliográfica en formato MARC-XML. De hecho cuando se ejecuta este proceso, todos los registros que conforman la colección son ordenados en función de su fecha de modificación y actualización.

El procedimiento general contempla el despliege inicial de un formulario de selección de la colección y un botón de ejecución del proceso de ordenación que regenera el canal de sindicación de respuestas y su archivo XSL para la representación de los principales campos de descripción. Una vez terminado el proceso, el archivo *response.xml*, queda listo para su sindicación y redifusión.

Como puede observarse el servicio es limitado puesto que sólo permite efectuar la ordenación según actualización de registros bibliográficos de un catálogo. No obstante, el sistema puede llegar a ampliarse fácilmente para adaptar la ordenación de los registros según campos de descripción bibliográfica y autoridades utilizando la técnica de filtrado XPath, explicada en el apartado del servicio de búsqueda de SYNCORE.

Título: Código del servicio de ordenación de SYNCORE
Archivo: service.update.php
Referencia: tabla145

Función de despliege del formulario de ordenación

- La primera acción cuando se inicializa el servicio de ordenación es el despliege del formulario principal que permite elegir las colecciones disponibles que se desean ordenar según la fecha de actualización y de últimas modificaciones de sus registros.

```
switch($_POST[id]) {

case 'S3M1':

// Creación del canal de respuesta, response.xml
$file = fopen('response.xml','w');
fwrite ($file,"<?xml version='1.0' encoding='UTF-8'?>
<?xml-stylesheet type='text/xsl' href='response.xsl'?>
<sync>
<response>
<message>Seleccione una colección</message>
</response>
</sync>");
fclose($file);
```

```
include ("services/func.collections.php");

// Creación del XSL del canal de respuesta que incluye el formulario de colecciones
$file = fopen('response.xsl','w');
fwrite ($file,"<?xml version='1.0' encoding='UTF-8'?>
<xsl:stylesheet version='1.0' xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
<xsl:template match='/'>
<html>
<head>
<title>syncore response channel</title>
<meta http-equiv='content-type' content='text/html; charset=UTF-8'/>
<link rel='stylesheet' href='style.css' type='text/css'/>
</head>
<body>
<p><h3>Informe de últimas actualizaciones</h3></p>
<form action='$url' method='post'>

<input type='hidden' name='id' value='$idservice'/>
<input type='hidden' name='file' value='$fileservice'/>
<input type='hidden' name='launch' value='launch'/>

<div style='position: relative; float: left;'>
<input type='text' name='coldestiny' id='coldestiny' size='32' value='$coldestiny'/>
        <div id='collection' style='display: block;'>
         // Marco iframe con las colecciones disponibles
        <iframe src='authorities/collections.xml' scrolling='yes' class='selectframe'></iframe>
</div>
</div>
// Botón de ordenación.
<input type='submit' name='updateinfo' value='ordenar según fecha de actualización'/>
</form>
</body></html>
</xsl:template></xsl:stylesheet>");
fclose($file);

// Actualización del canal de servicios completo para visualizar los cambios efectuados.
echo "<script language='javascript'>window.parent.location.href='service.xml'</script>";
break;
}
```

Función de ordenación según fecha de actualización

- Enviada la petición de ordenación de una colección, la presente función, efectúa una consulta SQL sencilla de ordenación descendente de los registros de dicha colección en función al campo *tag005* correspondiente a la fecha en formato ISO8601. Como resultado de esta operación, se genera un canal de respuesta, con toda la colección sindicada en formato MARC-XML.

```
if($_POST[updateinfo]) {

// Creación del canal de respuesta, response.xml
$file = fopen('response.xml','w');
fwrite ($file,"<?xml version='1.0' encoding='UTF-8'?>
<?xml-stylesheet type='text/xsl' href='response.xsl'?>
<collection>");

// Consulta completa de la colección pedida según etiqueta tag005 de fecha de actualización
mysql_select_db("$database", $con);
```

```
$sql = "SELECT * FROM $_POST[coldestiny] ORDER BY tag005 DESC";
$resultado = mysql_query($sql,$con);

// Escritura de todos los registros bibliográficos de la colección en formato MARC-XML
while ($fila = mysql_fetch_assoc($resultado)) {
fwrite ($file,"
<record>
<leader>$fila[leader]</leader>
<controlfield tag='001'>$fila[tag001]</controlfield>
<controlfield tag='003'>$fila[tag003]</controlfield>
<controlfield tag='005'>$fila[tag005]</controlfield>
<datafield tag='010' ind1=" ind2=""><subfield code='a'>$fila[tag010a]</subfield></datafield>
<datafield tag='017' ind1=" ind2=""><subfield code='a'>$fila[tag017a]</subfield></datafield>
<datafield tag='020' ind1=" ind2=""><subfield code='a'>$fila[tag020a]</subfield></datafield>
<datafield tag='041' ind1=" ind2=""><subfield code='a'>$fila[tag041a]</subfield></datafield>
<datafield tag='050' ind1='1' ind2='0'><subfield code='a'>$fila[tag050a]</subfield></datafield>
<datafield tag='080' ind1=" ind2=""><subfield code='a'>$fila[tag080a]</subfield></datafield>
<datafield tag='082' ind1='0' ind2=""><subfield code='a'>$fila[tag082a]</subfield></datafield>

<datafield tag='100' ind1='0' ind2=''>
<subfield code='a'>$fila[tag100a]</subfield>
<subfield code='0'>$fila[tag100o]</subfield>
</datafield>

<datafield tag='110' ind1='2' ind2=''>
<subfield code='a'>$fila[tag110a]</subfield>
<subfield code='0'>$fila[tag110o]</subfield>
</datafield>

<datafield tag='111' ind1='2' ind2=''>
<subfield code='a'>$fila[tag111a]</subfield>
<subfield code='0'>$fila[tag111o]</subfield>
</datafield>

<datafield tag='130' ind1='0' ind2=''>
<subfield code='a'>$fila[tag130a]</subfield>
<subfield code='0'>$fila[tag130o]</subfield>
</datafield>

<datafield tag='245' ind1='1' ind2='0'>
<subfield code='a'>$fila[tag245a]</subfield>
<subfield code='b'>$fila[tag245b]</subfield>
<subfield code='c'>$fila[tag245c]</subfield>
</datafield>

<datafield tag='250' ind1=" ind2=''>
<subfield code='a'>$fila[tag250a]</subfield>
<subfield code='b'>$fila[tag250b]</subfield>
</datafield>

<datafield tag='260' ind1=" ind2=''>
<subfield code='a'>$fila[tag260a]</subfield>
<subfield code='b'>$fila[tag260b]</subfield>
<subfield code='c'>$fila[tag260c]</subfield>
</datafield>

<datafield tag='300' ind1=" ind2=''>
<subfield code='a'>$fila[tag300a]</subfield>
<subfield code='b'>$fila[tag300b]</subfield>
<subfield code='c'>$fila[tag300c]</subfield>
```

```
<subfield code='e'>$fila[tag300e]</subfield>
</datafield>

<datafield tag='490' ind1='0' ind2=''>
<subfield code='a'>$fila[tag490a]</subfield>
<subfield code='v'>$fila[tag490v]</subfield>
</datafield>

<datafield tag='500' ind1='' ind2=''>
<subfield code='a'>$fila[tag500a]</subfield>
</datafield>

<datafield tag='650' ind1='1' ind2='4'>
<subfield code='a'>$fila[tag650a]</subfield>
<subfield code='0'>$fila[tag650o]</subfield>
</datafield>

<datafield tag='700' ind1='0' ind2=''>
<subfield code='a'>$fila[tag700a]</subfield>
<subfield code='0'>$fila[tag700o]</subfield>
</datafield>

<datafield tag='710' ind1='2' ind2=''>
<subfield code='a'>$fila[tag710a]</subfield>
<subfield code='0'>$fila[tag710o]</subfield>
</datafield>

<datafield tag='887' ind1='' ind2=''>
<subfield code='a'>$fila[tag887a]</subfield>
</datafield>

</record> ");     }

fwrite ($file,"</collection>");
fclose($file);

// Creación del XSL del canal de respuesta
$file = fopen('response.xsl','w');
fwrite ($file,"<?xml version='1.0' encoding='UTF-8'?>
<xsl:stylesheet version='1.0' xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
<xsl:template match='/'>

<html>
<head>
<title>syncore response channel</title>
<meta http-equiv='content-type' content='text/html; charset=UTF-8'/>
<link rel='stylesheet' href='style.css' type='text/css'/>
</head>
<body>

<form action='$url' method='post'>
// Representación filtrada de todos los registros bibliográficos.
<xsl:for-each select='//collection/record'>

// Modelo de representación bibliográfica.
<div class='block'>
<div class='blockpart1'>
// Representación del número de control de registro bibliográfico.
<xsl:value-of select='controlfield[@tag=\"001\"]'/> -
// Identificador del centro catalogador.
```

```
<xsl:value-of select='controlfield[@tag=\"003\"]'/> -
// Última fecha de actualización
<xsl:value-of select='controlfield[@tag=\"005\"]'/>
</div>

<div class='blockpart2'>
// Título propiamente dicho
<xsl:value-of select='datafield[@tag=\"245\"]/subfield[@code=\"a\"]'/>:
// Otras formas del título
<xsl:value-of select='datafield[@tag=\"245\"]/subfield[@code=\"b\"]'/> /
// Mención de responsabilidad
<xsl:value-of select='datafield[@tag=\"245\"]/subfield[@code=\"c\"]'/> .-
// Lugar de publicación
<xsl:value-of select='datafield[@tag=\"260\"]/subfield[@code=\"a\"]'/>:
// Editorial
<xsl:value-of select='datafield[@tag=\"260\"]/subfield[@code=\"b\"]'/>,
// Fecha de publicación
<xsl:value-of select='datafield[@tag=\"260\"]/subfield[@code=\"c\"]'/>
</div>
</div>

</xsl:for-each>
</form>
</body></html>
</xsl:template></xsl:stylesheet>");

fclose($file);
// Actualización del canal de servicios completo para visualizar los cambios efectuados.
echo "<script language='javascript'>window.parent.location.href='$url?func=terminate2'
</script>";
die();   }
```

7.8. Experiencia de usuario en SYNCORE

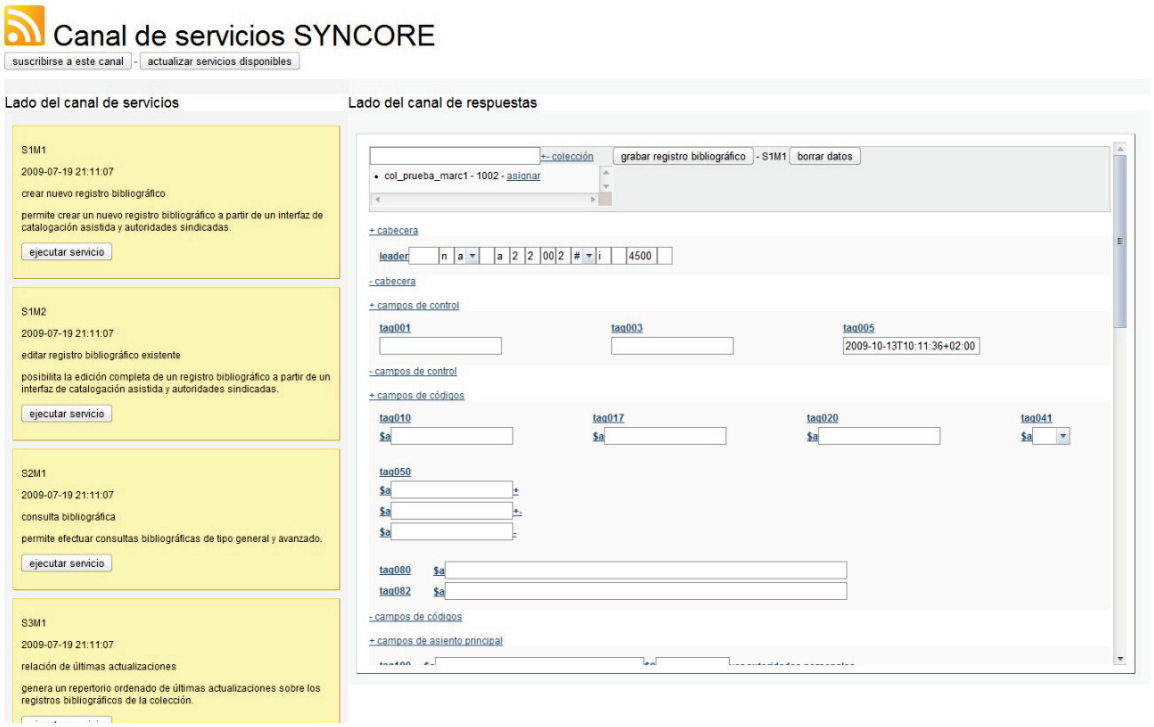
Uno de los objetivos planteados para la aplicación SYNCORE es permitir un fácil manejo de la misma, de manera que la experiencia del usuario sea lo más accesible posible. Por este motivo se explica el funcionamiento integral atendiendo a las pantallas que el usuario visualiza en cada momento. Cuando el usuario visualiza SYNCORE, obtiene una interfaz gráfica compuesta por dos partes. A la izquierda encontrará lo relativo a los servicios disponibles, y a la derecha un segundo canal de sindicación embebido con las respuestas del sistema. La parte izquierda consta básicamente de un archivo XML (el canal sindicado en sí) con el texto de los servicios y de las cabeceras identificativas de cada uno de ellos, más un archivo XSL adscrito al anterior, que muestra la cabecera correspondiente al servicio seleccionado por el usuario y lo envía a SYNCORE como si se tratase de un formulario. Dicha cabecera permite a SYNCORE ejecutar el módulo del programa relativo al servicio solicitado. Originada una respuesta, el programa regenera el segundo canal de sindicación con

la respuesta y actualiza el primero y principal para incorporar las novedades de la parte derecha. Este segundo canal con la respuesta consta esencialmente, a su vez, de otros dos archivos: un archivo XML (el canal sindicado) con el texto e información relativa al servicio elegido, más otro archivo XSL adscrito que incorpora los formularios con los datos que deba aportar el usuario, para culminar con la ejecución del servicio.

El sistema, a partir de una determinada colección, permite al usuario que se haya sindicado la realización de las funciones de incorporación de registros bibliográficos a la colección, edición o modificación de registros bibliográficos, consulta del fondo y recuperación de registros y visualización del fondo ordenado por fecha de última modificación.

Si el usuario selecciona la opción de incorporación de registros, en la ventana de respuesta visualizará un detallado formulario que incluye las principales áreas y etiquetas del formato MARC y sus definiciones, junto con listas de autoridades que ayudarán durante el proceso de descripción bibliográfica: personales, corporativas, temáticas, de títulos uniformes, de jornadas y congresos, de lugares y de editoriales.

Título: Formulario de incorporación de un nuevo registro en la colección
Referencia: figura41



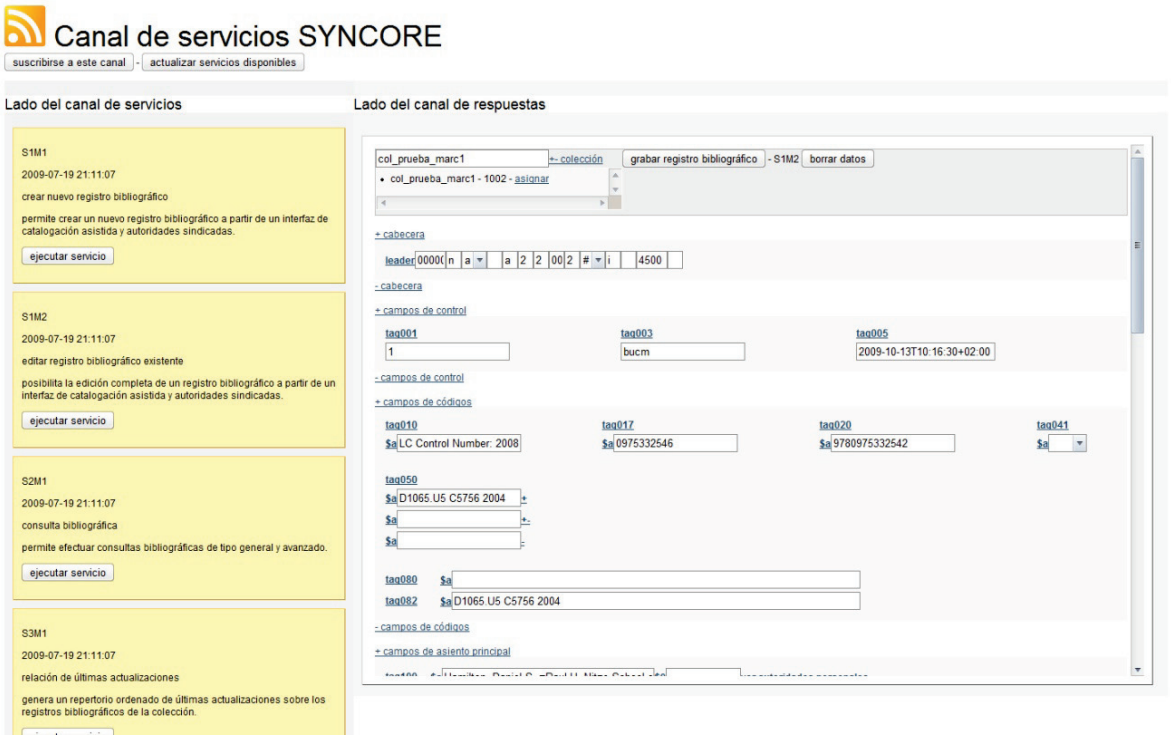
Una vez completado el formulario, el usuario remite la información a SYNCORE, que se encarga de incorporar el nuevo registro en la base de datos.

El usuario puede, igualmente, modificar o editar un registro existente en el catálogo. Una vez elegida esta función, se visualiza en la parte derecha una caja de texto donde introducir el número de control tipificado por la etiqueta 001 del formato MARC, o bien el ISBN asignado al registro.

Título: Formulario de consulta de un registro bibliográfico existente para su edición
Referencia: figura42

A continuación, este servicio visualizará el mismo formulario que en el servicio de incorporación, con la salvedad de que muestra todos los datos del registro requerido. Una vez modificado el formulario, el usuario remite la información a SYNCORE, que se encarga de incorporar el registro actualizado en la base de datos.

Título: Registro bibliográfico cargado en el formulario de edición
Referencia: figura43

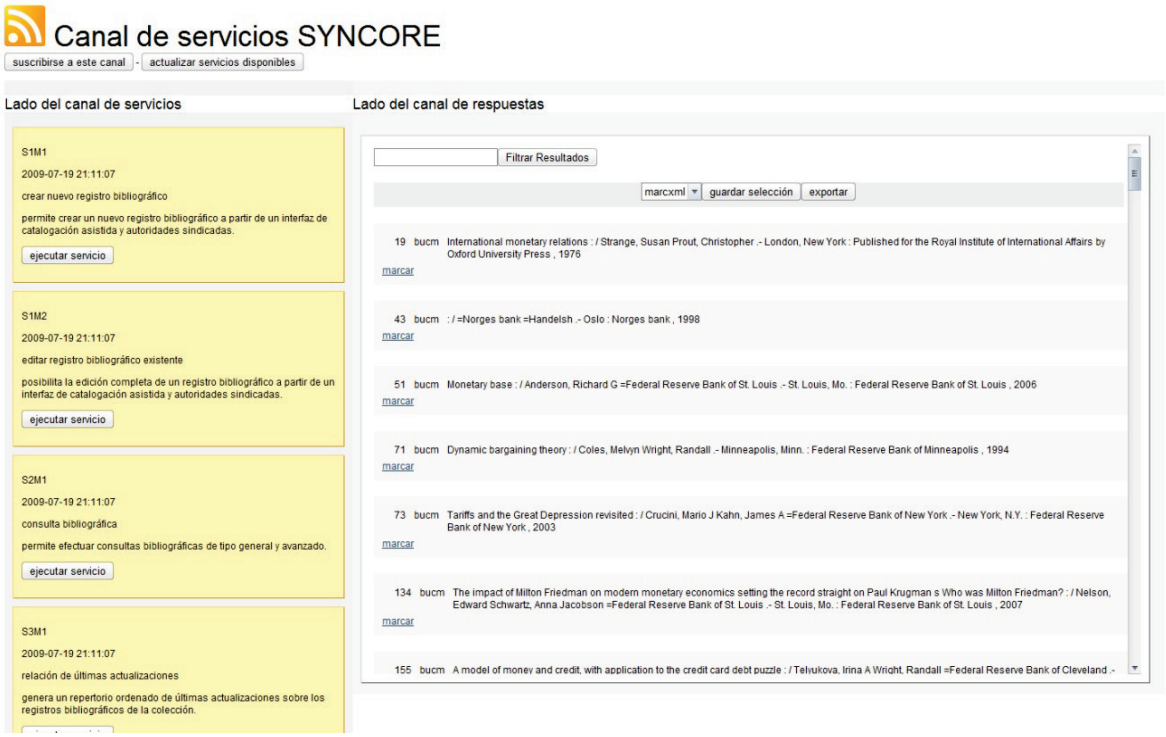


El servicio de consulta permite efectuar búsquedas sobre la colección bibliográfica. Cuando el usuario elige esta opción, visualiza en el canal de respuesta una caja de texto donde introducir la cadena de consulta.

Título: Formulario de consulta del catálogo bibliográfico
Referencia: figura44

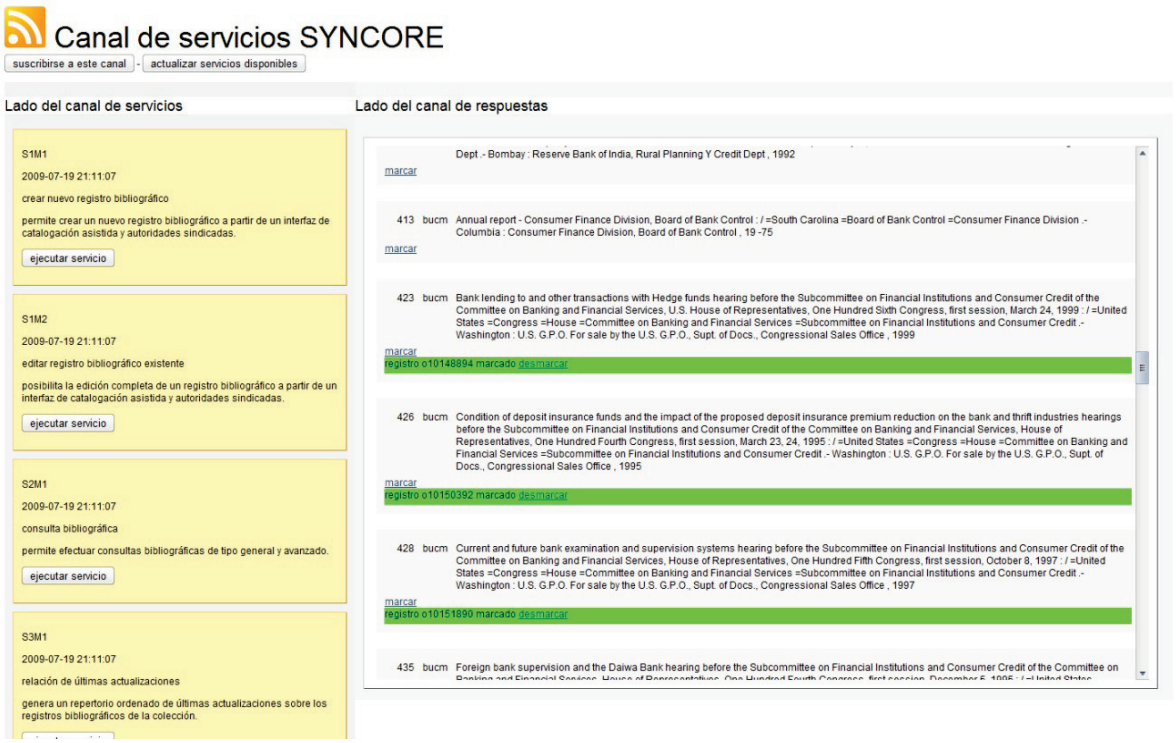
El texto es transmitido a SYNCORE, cuyo módulo correspondiente a este servicio se encarga de normalizar los caracteres introducidos, para a continuación construir la consulta en SQL y ejecutarla en la base de datos. El resultado es incorporado a un archivo MARC-XML, que junto a un nuevo archivo XSL, configuran el canal de respuesta. El archivo XSL, además de ocuparse de la fácil visualización de los registros de la respuesta, incluye un formulario que permite realizar una nueva consulta limitada a los registros previamente obtenidos. La búsqueda en este caso se efectúa mediante XPath, ya que se realiza directamente en el archivo MARC-XML.

Título: Resultados del proceso de búsqueda en SYNCORE
Referencia: figura45



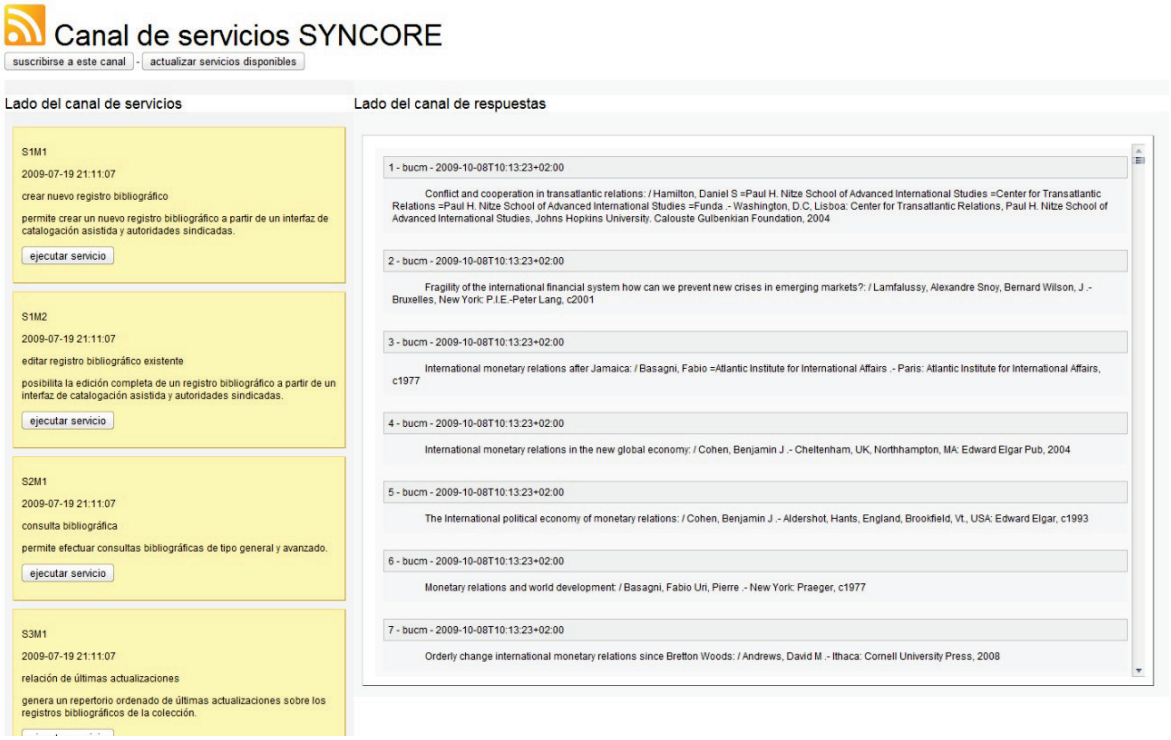
El usuario puede marcar aquellos registros que sean más relevantes en relación a su consulta. A continuación, SYNCORE se encargará de generar un nuevo canal de respuesta cuyo archivo de datos MARC-XML incluye exclusivamente los registros previamente seleccionados.

Título: Ejemplo de selección de resultados en SYNCORE
Referencia: figura46



El servicio de ordenación por su parte, organiza la colección bibliográfica por fecha de última actualización de cada registro, mostrando antes los incorporados o modificados más recientemente. También incorpora la función de sindicación del catálogo bibliográfico en su integridad, facilitando su difusión y utilización por parte de terceras unidades de información y documentación.

Título: Resultados del servicio de ordenación según fecha de actualización
Referencia: figura47



7.9. Análisis de resultados obtenidos con SYNCORE

Antes de proceder al análisis de los resultados obtenidos con el programa SYNCORE, se ha efectuado una última prueba que demuestra que todos los planteamientos escritos en relación al funcionamiento de este programa son perfectamente válidos. La mejor prueba de ello es que puede ser utilizado en Internet por cualquier usuario, dado que ha sido alojado en un servidor web, quedando a disposición en la siguiente dirección:

http://mblazquez.es/testbench/syncore/service.xml

Como consecuencia de la instalación y funcionamiento en línea de SYNCORE, también se ha instalado la plataforma SYNC, utilizada para las pruebas de sindicación bibliográfica. También puede ser consultada en la siguiente dirección:

http://mblazquez.es/testbench/sync/index.php

Para analizar los resultados obtenidos, se responde a las preguntas formuladas en el apartado de diseño y arquitectura de SYNCORE:

- **¿Es posible gestionar catálogos bibliográficos mediante técnicas de sindicación de contenidos?** Sí es posible gestionar los catálogos bibliográficos mediante sindicación de contenidos. Hay que precisar que la sindicación de contenidos se encarga de transmitir no solo los resultados de las consultas, sino también los formularios con los que interactúa con el usuario, incluso mensajes y comunicaciones oportunas en cada momento, de forma que se puede afirmar que se ha logrado una interactividad a base de regenerar y actualizar los canales de sindicación, a base de las peticiones del usuario.

- **¿Se pueden sindicar los servicios bibliotecarios relativos al catálogo?** Sí se pueden sindicar los servicios bibliotecarios, como se ha demostrado durante el proceso de construcción de SYNCORE, de hecho cada servicio que se integra en la carpeta *services*, contiene un archivo XML de descripción del programa, que es tomado por SYNCORE para construir una lista de servicios disponibles que conformará el canal de sindicación de servicios *service.xml*. Por lo tanto sí se pueden transmitir los servicios y sus cabeceras de identificación para su posterior ejecución.

- **¿Qué formatos se pueden emplear para la sindicación de los servicios y la gestión de catálogos bibliográficos?** Aunque SYNCORE se ha basado fundamentalmente en el formato MARC-XML para la transmisión de los registros bibliográficos de los catálogos, también se han empleado nuevos formatos en XML como <*sync*> para la descripción de los servicios, los mensajes de comunicación al usuario y las respuestas de referencia e información de los formularios. Por otro lado el formato <*thesaurus*> para la creación de un pequeño tesauro de prueba y finalmente un formato <*collection*> de autoridades basado en MADS. El empleo de tales formatos no reconocidos en los navegadores web, no ha impedido su correcta visualización, actualización, lectura o análisis parser, por lo que puede comprobarse que cualquier formato basado en XML puede ser empleado en SYNCORE para transmitir la misma información. Esto significa que podrían configurarse diversas combinaciones de formatos de sindicación trabajando en el mismo sistema.

- **¿Realmente el programa SYNCORE puede recuperar registros bibliográficos y seleccionar sus resultados como haría un cliente Z39.50?** El programa SYNCORE ha demostrado su capacidad para recuperar registros bibliográficos de una colección y también como partiendo de una colección en formato MARC-XML logra filtrar los términos de consulta que el usuario especifique. En este sentido SYNCORE tiene propiedades de recuperación muy similares a Z39.50. No obstante, como se ha aclarado al comienzo del apartado de este programa, aún no efectúa consultas secuenciales automáticas en múltiples catálogos de múltiples plataformas SYNC, como haría Z39.50. A pesar de ello, también hay que aclarar que si bien esto es así, no resulta un problema técnico imposible, ya que los principios de funcionamiento demostrados en SYNCORE son reproducibles con múltiples colecciones y servidores. En cuanto a la capacidad de selección de los resultados, queda demostrado que el programa SYNCORE puede almacenar los resultados que el usuario marca para posteriormente regenerar el canal de respuestas con ellos, permitiendo su sindicación o exportación.

- **¿Es realmente la sindicación en SYNCORE una alternativa a Z39.50 y en qué medida?** La sindicación de contenidos puede ser en realidad una alternativa al protocolo Z39.50, como ha demostrado en la plataforma SYNC y SYNCORE. De hecho se logran las principales funciones que desempeña dicho protocolo, con la ventaja sustancial que supone no tener los mismos requerimientos técnicos de instalación o interpretación. En este sentido SYNCORE es una herramienta más fácil de implementar y más accesible a

cualquier biblioteca y usuario. Ahora bien, aún no se puede determinar que SYNCORE constituya un sustituto de Z39.50, dada la importante cantidad de mejoras que son necesarias para llegar a un nivel de desarrollo similar.

Comparación de SYNCORE con Z39.50

En este apartado se aborda la comparación de las principales características del protocolo Z39.50 (NISO; LOC, 2003) con las de SYNCORE.

Hay que destacar que ambos parten de un modelo de comunicación tipo cliente-servidor. La diferencia estriba en que SYNCORE requiere del lado cliente un simple navegador web, mientras que Z39.50 exige un servidor Z capaz de resolver las peticiones, además de otro servidor del lado cliente para efectuarlas correctamente.

En segundo lugar, en Z39.50 no siempre se da una correspondencia entre los servicios disponibles en el lado del servidor y en el lado del cliente, lo que obliga a implementar un sistema de comprobación de servicio tras la petición del usuario. En cambio, el empleo de la sindicación no conlleva esta acción de comprobación, pues todos los servicios en el lado del servidor son visualizados en el canal de sindicación y están con absoluta seguridad disponibles para el usuario.

En cuanto a los servicios ofertados, ambos disponen de las capacidades de consulta y recuperación de registros bibliográficos, así como de ordenación del catálogo por fecha de actualización. Z39.50 permite, además, detectar registros duplicados, incorporar diversos criterios suplementarios de ordenación del catálogo, así como la posibilidad de efectuar préstamos inter-bibliotecarios. SYNCORE, por su parte, incluye los servicios de incorporación y edición de registros bibliográficos, además de permitir la sindicación del catálogo en su integridad.

El protocolo Z39.50 es capaz de establecer conexiones con múltiples bases de datos de manera secuencial, efectuando la consulta en todas ellas. Sin embargo, SYNCORE, en su configuración actual, solo permite la consulta de una única colección en una única base de datos. No obstante, es capaz de reconocer diversas colecciones, siempre que formen parte de la misma base MySQL.

Z39.50 requiere de un lenguaje denominado ZSpeak para enviar cada petición del usuario al servidor. El servidor, a su vez, se comunica con las bases de datos utilizando el mismo lenguaje. Y lo mismo sucede con las respuestas remitidas al cliente, que deben ser traducidas en última instancia del lenguaje Zspeak para poder ser visualizados en pantalla. SYNCORE, en cambio, no requiere del concurso de ningún lenguaje intermedio para efectuar la comunicación de las peticiones del usuario y las respuestas generadas. Los aspectos analizados previamente entre el protocolo Z39-50 y el programa SYNCORE de sindicación de catálogos bibliográficos pueden resumirse en la *tabla146*.

Título: Tabla comparativa del sistema Z39.50 y SYNCORE
Referencia: tabla146

Concepto	SYNCORE	Z39.50
Modelo de comunicación	Cliente-Servidor	Cliente-Servidor
Requisitos del lado cliente	Navegador web	Cliente Z39.50
Comprobación de servicio	Acción no necesaria	Acción necesaria
Lenguaje intermedio	No necesita	Necesita ZSpeak
Servicios ofertados	Consulta. Recuperación. Ordenación del catálogo por fecha de actualización. Incorporación de registros. Edición de registros. Sindicación del catálogo bibliográfico.	Consulta. Recuperación. Ordenación del catálogo por fecha de actualización. Detección de duplicados. Criterios suplementarios de ordenación. Préstamo inter-bibliotecario.
Bases de datos recuperables	Una base de datos MYSQL	Múltiples bases de datos
Detección de colecciones	Múltiples colecciones	Múltiples colecciones

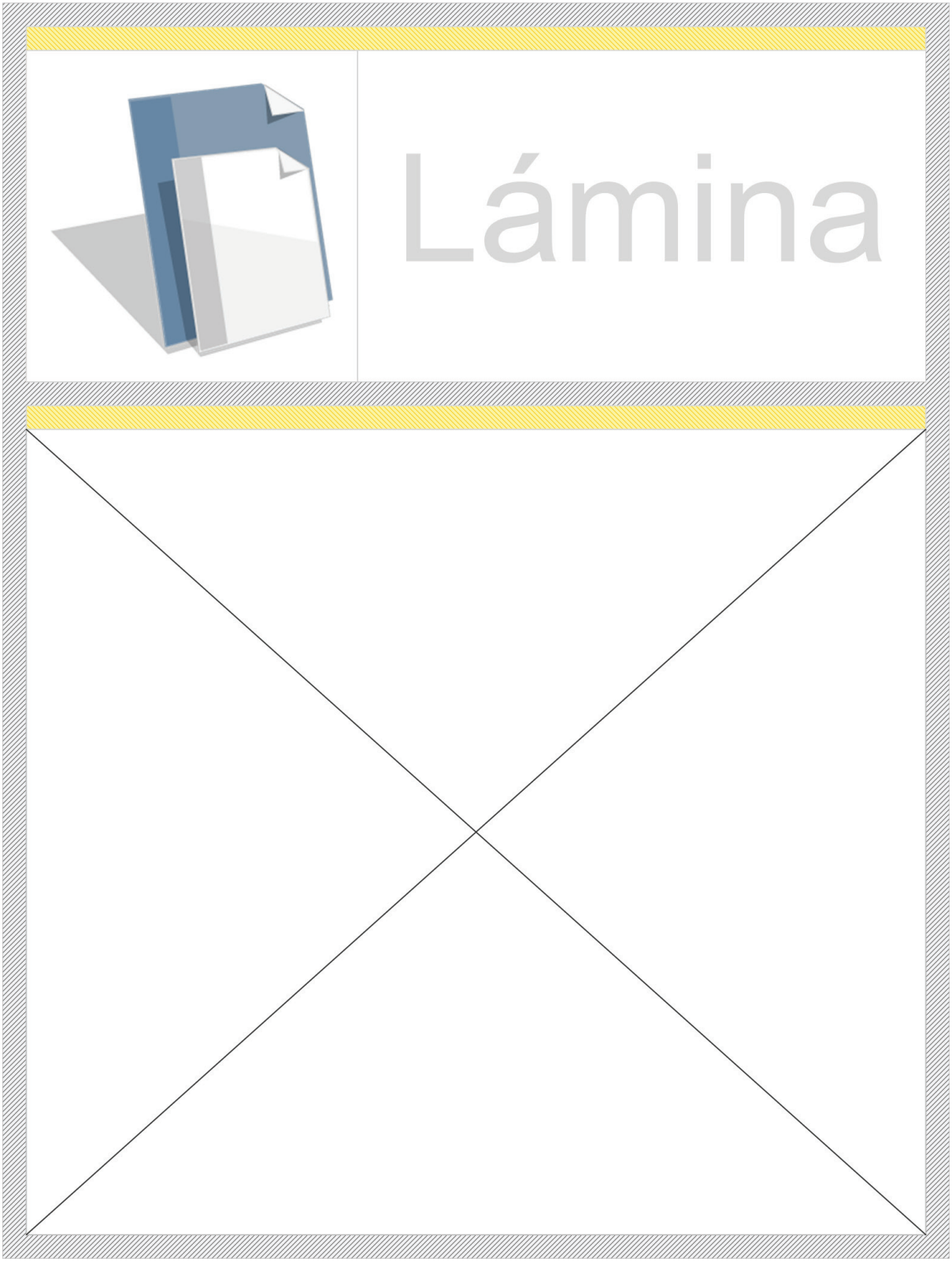
7.10. Aplicaciones del sistema desarrollado

La técnica de sindicación empleada en la plataforma SYNC y SYNCORE puede ser aprovechada en diversos campos de la documentación. Por ejemplo el desarrollo de catálogos colectivos sindicados, la unificación de acceso a colecciones bibliográficas, documentales, archivísticas y museísticas, la integración de listas de autoridades sindicadas, el desarrollo de la web semántica en colecciones bibliográficas sindicadas e incluso la sindicación de procedimientos administrativos, a continuación se describen los siguientes:

Desarrollo de catálogos colectivos sindicados.

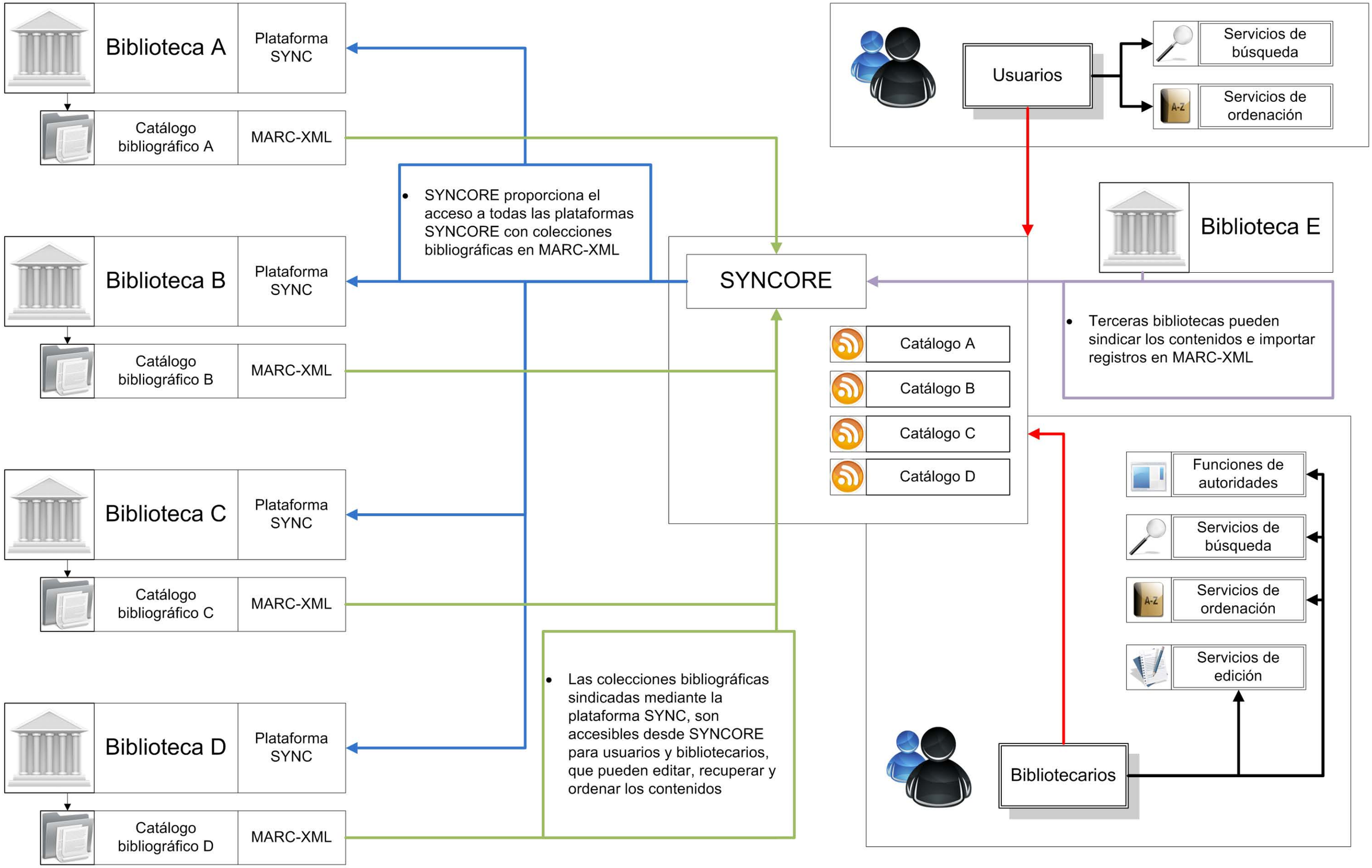
El desarrollo de catálogos colectivos mediante sindicación es un ejemplo claro de lo que se puede conseguir con la plataforma SYNC y SYNCORE. Si bien es posible sindicar las colecciones bibliográficas en formatos MARC-XML, también es posible compartirlas con terceras bibliotecas que empleen la plataforma SYNC. De esta forma, se pueden generar verdaderas redes de catálogos que sean accesibles desde una misma aplicación SYNCORE. A su vez el acceso a todas las colecciones queda unificado manteniendo la procedencia de las mismas y posibilitando su edición, ordenación, gestión de autoridades por parte de bibliotecarios y recuperación por parte del resto de usuarios. Esta aplicación solucionaría los problemas de formato existentes al emplear diversos sistemas de gestión de bibliotecas que codifican en cada caso de forma diferente, los registros bibliográficos disponibles desde el OPAC. Esta aplicación puede ser consultada en la *figura48*.

Título: Desarrollo de catálogos colectivos sindicados
Referencia: figura48



Desarrollo de catálogos colectivos sindicados

Referencia: figura48



Unificación de acceso a colecciones bibliográfico-documentales, archivísticas y museísticas

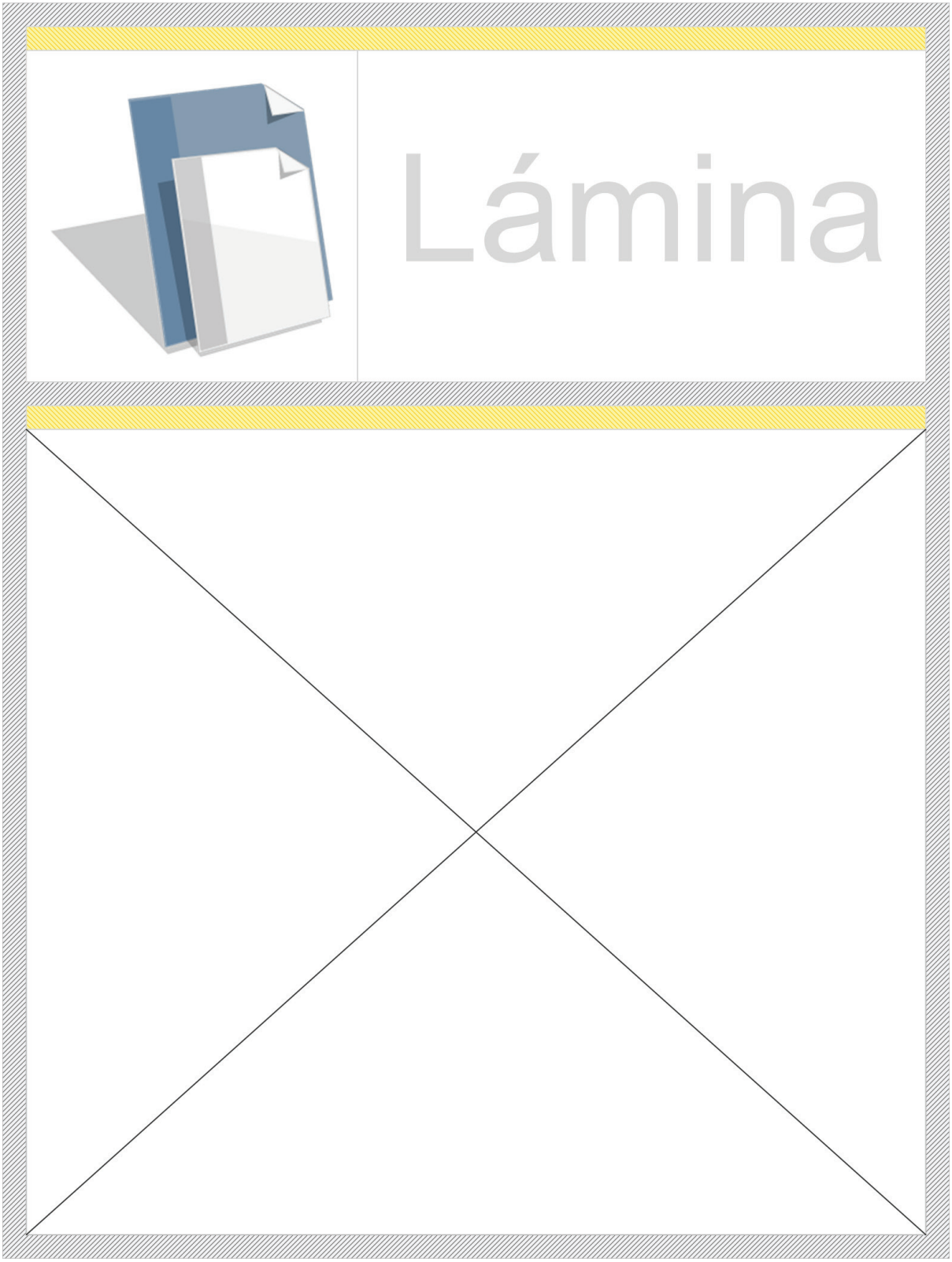
Las Bibliotecas que adoptan SYNC así como los participantes de una red de bibliotecas podrían generar su propio canal de sindicación MARC-XML correspondiente al conjunto o selección del fondo bibliográfico que disponen. Estos canales de sindicación son accesibles directamente por una aplicación SYNCORE de una biblioteca central que unificándolas consigue la recuperación de registros bibliográficos completos de manera inmediata, pudiendo ser visualizado cada asiento para su posterior corrección. Otras modalidades de sindicación, son aquellas que filtran o restringen los documentos sindicados a los que un usuario concreto demanda o según el depósito, archivo o centro de documentación requiere.

Este mismo método es aplicable a los depósitos pasivos de documentos, pudiendo ser identificados por su origen, y conocer el historial de cada documento según su trámite en biblioteca, fechas de ingreso en los mismos, hasta su almacenamiento. Un depósito podría compartir su registro y aparecer como tal en los catálogos bibliográficos sindicados de la biblioteca Central o cualquier otro nodo de la red.

Los Centros de Documentación, Editoriales y demás entes conexos con la documentación podrían nutrirse de información actualizada sobre la adquisición de publicaciones en bibliotecas, proceso de obsolescencia de la información (previa consulta de las altas en fuentes de sindicación de Registros Pasivos) o inclusive conocer la relación de materiales anejos entre de mediatecas con respecto a los fondos bibliográficos de las bibliotecas.

Los Archivos podrían tener un vínculo directo con las bibliotecas al poder incorporar su fuente de sindicación que a menudo está relacionada con la producción documental de la entidad, personas físicas o jurídicas que pueden ser vinculadas de manera directa al fondo bibliográfico de las bibliotecas, obteniendo en la sindicación, la plataforma ideal para la agrupación de asientos relacionados.

Título: Unificación de acceso a colecciones bibliográfico-documentales, archivísticas y museísticas
Referencia: figura49



Unificación de acceso a colecciones bibliográfico-documentales, archivísticas y museísticas

Referencia: figura49

Bibliotecas en Red

- Las Bibliotecas que adoptan SYNC así como los participantes de una red de bibliotecas podrían generar su propio canal de sindicación MARC-XML correspondiente al conjunto o selección del fondo bibliográfico que disponen. Estos canales de sindicación son accesibles directamente por una aplicación SYNCORE de una biblioteca central que unificándolas consigue la recuperación de registros bibliográficos completos de manera inmediata, pudiendo ser visualizado cada asiento para su posterior corrección. Otras modalidades de sindicación, son aquellas que filtran o restringen los documentos sindicados a los que un usuario concreto demanda o según el depósito, archivo o centro de documentación requiere.

Biblioteca Facultad 6 | *Biblioteca Facultad 5* | *Biblioteca Facultad 4* | *Biblioteca Facultad 3* | *Biblioteca Facultad 2* | *Biblioteca Facultad 1*

Leyenda (Fuentes Sindicadas)

- Fuente de Biblioteca
- Fuente de Depósito
- Fuente de Editorial
- Fuente de Archivo
- Fuente de Hemerotecas y Mediatecas

Fondos Bibliográficos 6 | **Fondos Bibliográficos 5** | **Fondos Bibliográficos 4** | **Fondos Bibliográficos 3** | **Fondos Bibliográficos 2** | **Fondos Bibliográficos 1**

Depósitos Documentales

- Este mismo método es aplicable a los depósitos pasivos de documentos, pudiendo ser identificados por su origen, y conocer el historial de cada documento según su trámite en biblioteca, fechas de ingreso en los mismos, hasta su almacenamiento.
- Un depósito podría compartir su registro y aparecer como tal en los catálogos bibliográficos sindicados de la biblioteca Central o cualquier otro nodo de la red.

Depósito B

Depósito A

Registro Pasivo B

Registro Pasivo A

SYNCORE

Colección de Pub. Periódicas

Colección de Docs. Audiovisuales

Hemerotecas y Mediatecas

Hemeroteca Anexa

Mediatica Anexa

- Los Centros de Documentación, Editoriales y demás entes conexos con la documentación podrían nutrirse de información actualizada sobre la adquisición de publicaciones en bibliotecas, proceso de obsolescencia de la información (previa consulta de las altas en fuentes de sindicación de Registros Pasivos) o inclusive conocer la relación de materiales anejos entre de mediatecas con respecto a los fondos bibliográficos de las bibliotecas.

Registro Administrativo

Fondos, Secciones, Series, Expedientes

Registro de Publicaciones

Centros de Documentación

- Los Centros de Documentación, Editoriales y demás entes conexos con la documentación podrían nutrirse de información actualizada sobre la adquisición de publicaciones en bibliotecas, proceso de obsolescencia de la información (previa consulta de las altas en fuentes de sindicación de Registros Pasivos) o inclusive conocer la relación de materiales anejos entre de mediatecas con respecto a los fondos bibliográficos de las bibliotecas.

Editorial Anexa

Archivos

Archivo 2

Archivo 1

- Los Archivos podrían tener un vínculo directo con las bibliotecas al poder incorporar su fuente de sindicación que a menudo está relacionada con la producción documental de la entidad, personas físicas o jurídicas que pueden ser vinculadas de manera directa al fondo bibliográfico de las bibliotecas, obteniendo en la sindicación, la plataforma ideal para la agrupación de asientos relacionados.

Integración de listas de autoridades sindicadas.

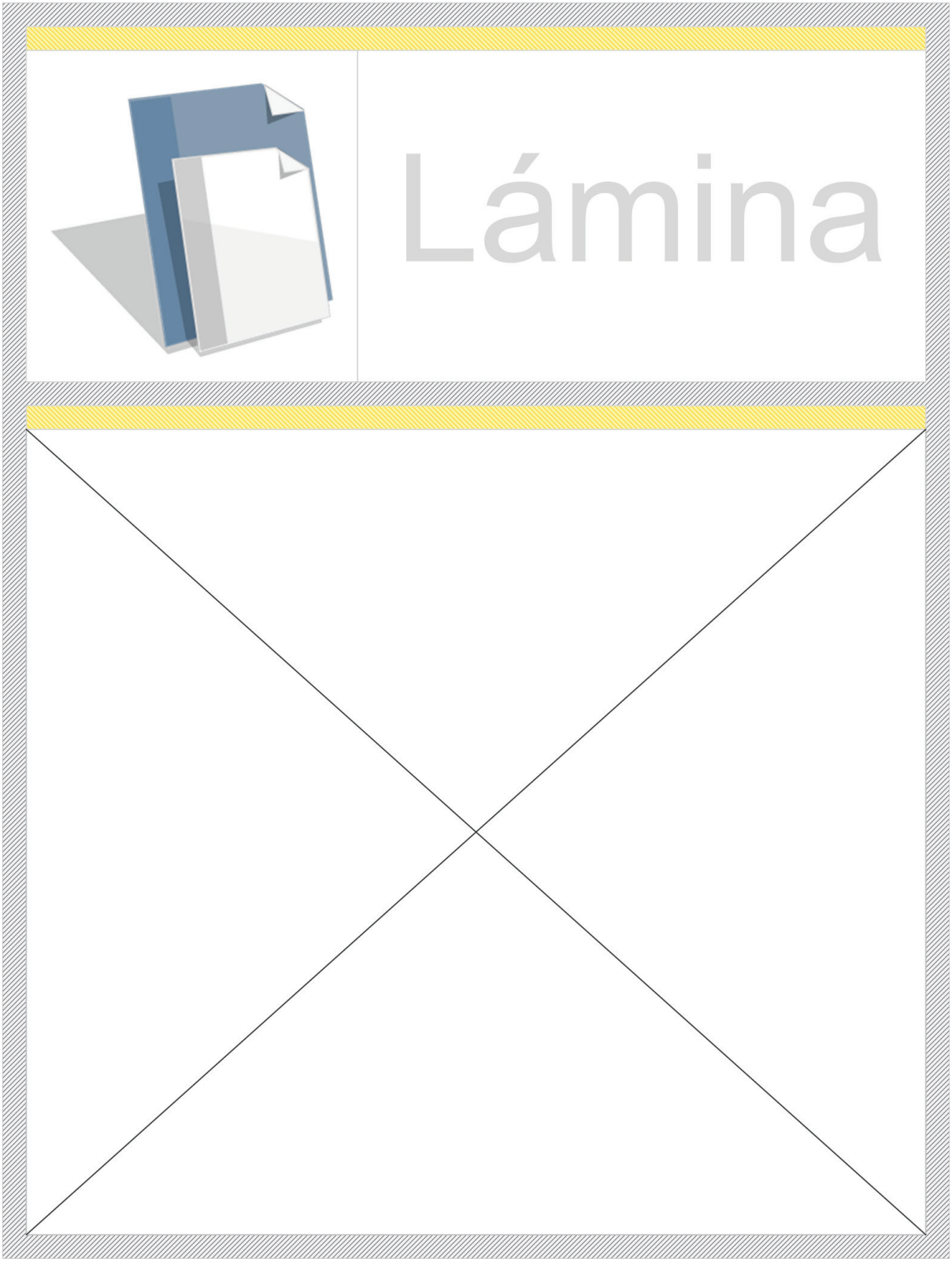
La idea de sindicar listas de autoridades no es descabellada, después de demostrar lo que puede hacer con las colecciones bibliográficas. La principal aplicación de la sindicación de autoridades sería la posibilidad de compartir con terceras bibliotecas la tarea de normalización de las autoridades personales, corporativas, títulos uniformes, editoriales, lugares, etc. Éstas podrían ser suscritas y empleadas en cualquier otro catálogo mediante la aplicación SYNCORE. Tal es así que manteniendo un formato común de descripción de autoridades, cualquier biblioteca suscrita a los diferentes canales de sindicación de autoridades, podría conocer las últimas modificaciones efectuadas en dichos listados.

Esta idea de integrar las listas de autoridades mediante sindicación de contenidos, también es extrapolable al caso de los tesauros y los lenguajes de clasificación. De hecho podrían compartirse por medio de canales de sindicación todas las casuísticas de clasificación desarrolladas por las bibliotecas que adoptan la plataforma SYNC y SYNCORE. De esta forma se facilitaría su corrección y mejora de la misma manera que con el resto de autoridades.

En el caso de los tesauros las ventajas que brinda la sindicación son evidentes, dado que podrían compartirse y desarrollarse grandes estructuras jerárquicas para mejorar la descripción de los recursos bibliográficos. De algún modo el concepto de redes de tesauros podría ser factible mediante su sindicación y aplicación en SYNCORE.

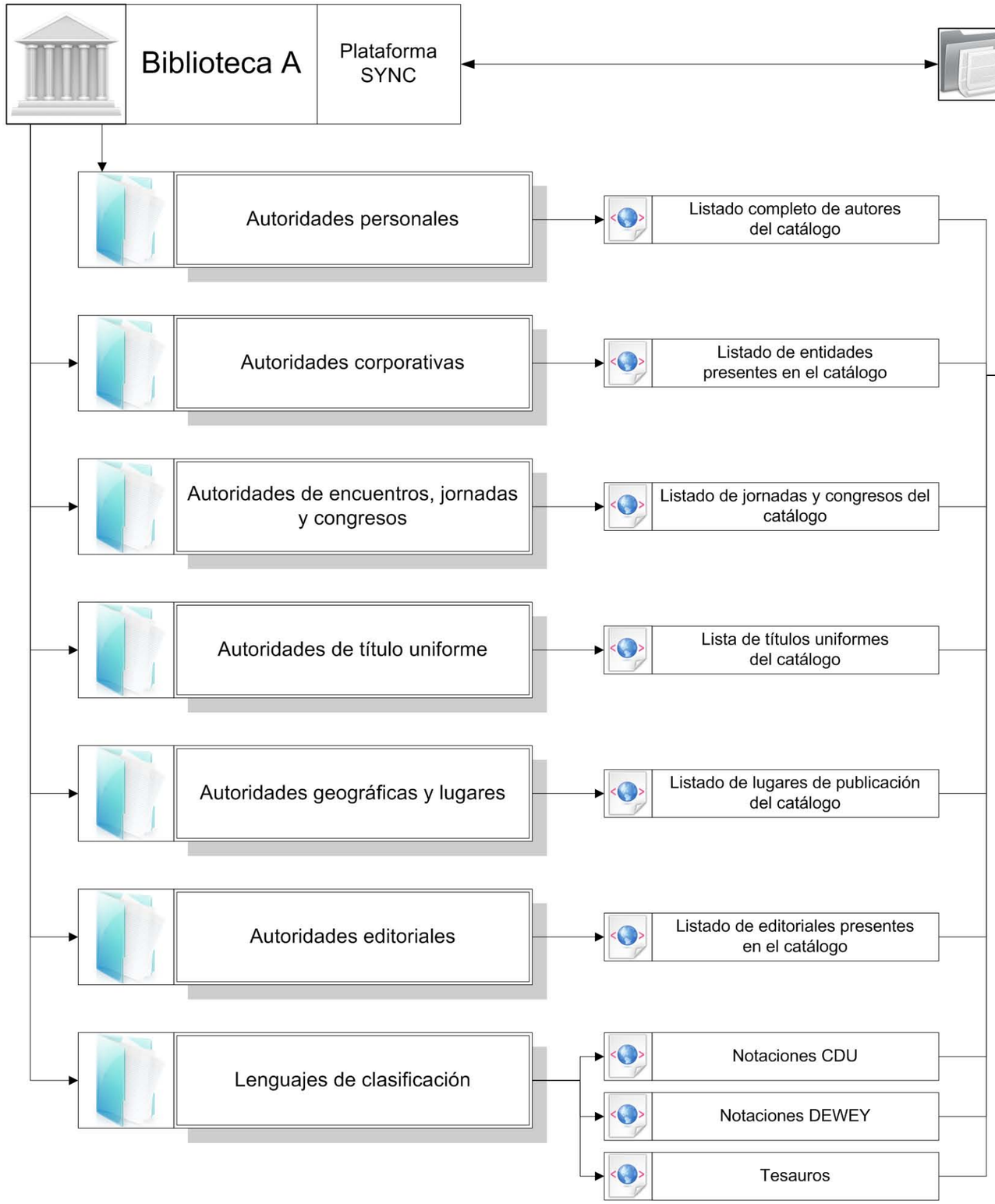
La principal prueba de que todos estos avances son posibles es que la carpeta *authorities* contiene listados de autoridades que podrían ser compartidos con terceras bibliotecas. Los desarrollos necesarios para llevarlo del todo a efecto pasan por el desarrollo de los programas parser necesarios para la importación de dichos listados de autoridades y por otro lado de su instalación en el propio SYNCORE.

Título: Integración de listas de autoridades sindicadas
Referencia: figura50



Integración de listas de autoridades sindicadas.

Referencia: figura50



Referencia: figura48

- Los catálogos bibliográficos sindicados en MARC-XML pueden ser editados incluso en cuanto a las autoridades se refiere. Permitiendo la sindicación de las listas de autoridades, se permite que terceras bibliotecas puedan participar en la mejora de estos instrumentos que termina repercutiendo en la mejor edición de los catálogos bibliográficos.
- El programa SYNCORE, permitiría por tanto no sólo difundir mediante sindicación las autoridades, sino permitir que terceras bibliotecas puedan incorporarlas para mejorar las descripciones de sus propias colecciones.
- En el caso de los tesauros se pueden generar verdaderas redes de sindicación de tesauros, que compartidos con el resto de bibliotecas puede dar lugar a un lenguaje documental más dinámico por cuantas modificaciones pueda sufrir y más completo por incorporar los puntos de vista de la comunidad de bibliotecarios y profesionales de los centros de información y documentación.

Capítulo 8. Conclusiones

1. La primera conclusión de la investigación es que en efecto, pueden gestionarse catálogos bibliográficos mediante sindicación de contenidos, ya que para ello se ha demostrado que es posible la sindicación de las colecciones bibliográficas, su manipulación, tratamiento y conversión a formatos de sindicación. Por otro lado también se han desarrollado las aplicaciones necesarias para editar y recuperar los registros bibliográficos que componen los catálogos de forma tal que en todo caso se utilizan la redifusión de contenidos MARC-XML específica del ámbito bibliotecario.

2. El análisis sugiere que la sindicación de contenidos es una técnica útil para la transferencia de datos bibliográficos, siendo un método alternativo al uso del protocolo Z39-50, más complejo (pues necesita comprobar los servicios antes de ejecutarlos y emplea un lenguaje intermedio ZSpeak en la comunicación), difícil de implementar (debido a la necesidad de un servidor en el lado cliente) y difícil de utilizar (pues SYNCORE precisa exclusivamente de un navegador web).

3. Se demuestra que el formato más completo para la transmisión de registros bibliográficos es MARC-XML, pudiendo utilizarse a todos los efectos como un formato de sindicación más. También RSS1.0 se demuestra útil y versátil, debido a la posibilidad de incluir diversos módulos de descripción especializados, como Dublin Core o PRISM mediante extensibilidad. No obstante su configuración básica, a tenor de las especificaciones, hace que carezca de las áreas de edición, descripción física y serie. Conforme a estos criterios, los formatos de sindicación menos aptos para la transferencia de datos bibliográficos son Atom y RSS2.0

4. Del análisis se desprende que no existen diferencias apreciables en cuanto a los tiempos de creación del canal de sindicación, sea cual sea el formato elegido y la complejidad de su estructura. Sin embargo, se ha comprobado que el tamaño del canal aumenta conforme aumenta la complejidad, implicando un incremento en los tiempos de importación. En consecuencia, si bien es técnicamente factible la sindicación de colecciones de cualquier tamaño, solamente con colecciones inferiores a 25000 registros nunca se alcanza un límite máximo de un minuto en la importación del canal sindicado.

5. El análisis sugiere que el rendimiento de la sindicación de contenidos en la transferencia de datos bibliográficos es mayor cuanto menor es el tamaño de la colección sindicada, con tiempos de importación inferiores al minuto, independientemente del formato, cuando la colección consta de menos de 25000 registros. En consecuencia, esta técnica sería muy apropiada para la actualización de catálogos en bibliotecas o para el mantenimiento de grandes bases de datos bibliográficas que se nutran de múltiples fuentes.

6. El programa SYNCORE muestra su utilidad en el área de la gestión de catálogos bibliográficos, aspecto no incluido en el protocolo Z39.50, posibilitando tanto la catalogación cooperativa como la creación de catálogos colectivos. Es verdad que Z39.50 posee servicios de los que carece actualmente SYNCORE (detección de duplicados, criterios suplementarios de ordenación de registros y préstamo inter-bibliotecario) y que puede conectarse a múltiples bases de datos, aunque su incorporación al programa SYNCORE no es técnicamente muy complicada.

Anexo 1. Bibliografía

- ACEBES JIMÉNEZ, R. and J.A. MAGÁN WALLS. 2002. La biblioteca electrónica y la sociedad virtual: volver a inventar la biblioteca. Madrid: Universidad Complutense, pp.46-60.
- AL-FADHLI, M. 2008. *Web 2.0 + Library = Library 2.0 / What is Library 2.0?* Sheffield.
- ANU. 2008. *The Australian National University (ANU) Library Home Page*. [Recurso electrónico]. [Revisado el 10 Marzo 2009]. Disponible en: <http://anulib.anu.edu.au/lib_home.html>
- APE(ATOM/PIE/ECHO) WORKING GROUP. 2003. *Atom 0.2 spec*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://wearehugh.com/public/2003/08/atom02spec.txt>
- APE(ATOM/PIE/ECHO) WORKING GROUP. 2003. *The Atom Syndication Format 0.3 (PRE-DRAFT)*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.mnot.net/drafts/draft-nottingham-atom-format-02.html>
- ARÉVALO MOLINA, J., J.L. RIVERA HERNÁNDEZ, O. FERNÁNDEZ OLALDE, y A.L. GALÁN GALL. 2003. Servicios bibliotecarios en las bibliotecas digitales. *In*: AAB ASOCIACIÓN ANDALUZA DE BIBLIOTECARIOS, (ed). *Los nuevos retos en los servicios bibliotecarios*, Málaga, pp.227-233.
- ARMS, W. 2000. *Digital Libraries*. Cambridge: MIT Press.
- *Atom API Documentation for Blogger*. 2006. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://code.blogger.com/archives/atom-docs.html>
- ATOMENABLED. 2005. *The Atom Syndication Format*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.atomenabled.org/developers/syndication/atom-format-spec.php>
- ATOMENABLED. 2007. *Atom Publishing Protocol: Introduction*. [Recurso electrónico]. [Revisado el 10 May 2009]. Disponible en: <http://www.atomenabled.org/developers/protocol/>
- ATOMENABLED. 2007. *Atom Syndication Format: Introduction*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.atomenabled.org/developers/syndication/>
- AYERS, D. y A. WATT. 2005. *Beginning RSS and Atom Programming*. Indianapolis: Wiley.
- BALNAVES, E. 2005. Systematic Approaches to Long Term Digital Collection Management. *Literary and Linguistic Computing*. 20(4), pp.399-413.
- BECERRIL GONZÁLEZ, V., L. RODRÍGUEZ YUNTA, y J. IGUALADOR OSORO. 2006. Blogs de biblioteconomía y documentación en España. *Revista Española de Documentación Científica*. 4(29), pp.603-627.
- BEGED DOV, G., D. BRICKLEY, R. DORNFEST et al. 2000. *RDF Site Summary 1.0 Modules: Dublin Core*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://purl.org/rss/1.0/modules/dc/>
- BEGED DOV, G., D. BRICKLEY, R. DORNFEST et al. 2000. *RDF Site Summary 1.0 Modules: Syndication*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://purl.org/rss/1.0/modules/syndication/>

- BEGED DOV, G., D. BRICKLEY, R. DORNFEST et al. 2001. *RDF Site Summary 1.0 Modules*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://web.resource.org/rss/1.0/modules/>

- BEGED DOV, G., D. BRICKLEY, R. DORNFEST et al. 2008. *RDF Site Summary (RSS) 1.0*. [Recurso electrónico]. [Revisado el 10 Mar 2009]. Disponible en: <http://web.resource.org/rss/1.0/spec>

- BEGED DOV, G., D. BRICKLEY, R. DORNFEST et al. 2000. *RSS 1.0 Modules: RSS091*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://web.resource.org/rss/1.0/modules/rss091/>

- BEGED DOV, G., A. SWARTZ, y E. VLIST. 2002. *RDF Site Summary 1.0 Modules: Content*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://purl.org/rss/1.0/modules/content/>

- BERKMAN CENTER. 2003. *RSS Advisory Board: RSS 2.0 Specification (version 2.0.1)*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.rssboard.org/rss-2-0-1>

- BERNERS-LEE, T. y M. FISCHETTI. 2000. *Weaving the Web: the past, present and future of the World Wide Web by its inventor*. Londres: Texere.

- BLÁZQUEZ OCHANDO, M. 2009. *Prueba de Formatos de Sindicación de Contenidos*. [Recurso electrónico]. [Revisado el 10 Noviembre 2009]. Disponible en: <http://mblazquez.es/docs/prueba-formatos-sindicacion.php>

- BLOOD, R. 2000. *weblogs: a history and perspective*. [Recurso electrónico]. [Revisado el 10 May 2009]. Disponible en: <weblogs: a history and perspective>

- BLOOD, R. 2002. *The weblog handbook: practical advice on creating and maintaining your blog*. Cambridge: Perseus.

- BOBERIC, D. y D. SURLA. 2007. XML Schema of query language defined by Z39.50 standard. *In*: D. HERCEG y H. ZARIN, (eds). *XVII Conference on Applied Mathematics PRIM 2006*. Novi Sad: University of Novi Sad, Faculty of Science and Mathematics, Department of mathematics and informatics, pp.107-114.

- BOBERIC, D. y D. SURLA. 2009. XML editor for search and retrieval of bibliographic records in the Z39.50 standard. *Electronic Library*. 27(3), pp.474-495.

- BRAY, T., D. HOLLANDER, A. LAYMAN, y R. TOBIN. 2006. *Namespaces in XML 1.0 (Second Edition)*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.w3.org/TR/xml-names/>

- BRAY, T. y D. WINER. 2003. *Ongoing: I Like Pie*. [Recurso electrónico]. [Revisado el 10 May 2009]. Disponible en: <http://www.tbray.org/ongoing/When/200x/2003/06/23/SamsPie>

- BRICKLEY, D. y R.V. GUHA. 1999. *Resource Description Framework (RDF) Schema Specification*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.w3.org/TR/1999/PR-rdf-schema-19990303/>

- BRICKLEY, D. y R.V. GUHA. 2004. *RDF Vocabulary Description Language 1.0: RDF Schema.* [Recurso electrónico]. [Revisado el 10 Marzo 2009]. Disponible en: <http://www.w3.org/TR/rdf-schema/>

- BRISCO, S. 2006. Visual OPACs. *Library Media Connection*. 25(3).
- BUYTAERT, D. y S. WITTENS. 2007. *Drupal Content management discusion community engine. Free software directory.* [Recurso electrónico]. [Revisado el 28 Abril 2008]. Disponible en: <http://directory.fsf.org/drupal.html>
- CADENHEAD, R. 2008. *RSS Advisory Board: RSS 0.90, 0.91 Moving to RSS Advisory Board*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.rssboard.org/news/181/rss-090-091-moving-rss-advisory-board>
- CADENHEAD, R., J. HOLDERNESS, R.C. MORIN, y G. SNEDDON. 2007. *RSS Advisory Board: Really Simple Syndication Best Practices Profile*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.rssboard.org/rss-profile>
- CASEY, M.E. y L.C. SAVASTINUK. 2006. *Library 2.0: Service for the Next-Generation Library.* [Recurso electrónico]. [Revisado el 3 Noviembre 2006]. Disponible en: <http://www.libraryjournal.com/article/CA6365200.html>
- CASTILLO BLASCO, L. 2002. *Introducción a la información científica y técnica*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.uv.es/macas/IICyT.htm>
- CHANG, N. 2008. Low-Cost Provision of XML Bibliographic Records. *Information Development*. 24(3), pp.237-244.
- CHAPMAN, N.P. 1987. *LR parsing: theory and practice*. Cambridge University Press.
- CLARK, J. 2001. *James Clark's Home Page.* [Recurso electrónico]. [Revisado el 29 Abril 2007]. Disponible en: <http://www.jclark.com/>
- CLARK, J. 2007. *Expat XML Parser Toolkit. Thai Open Source Software Center.* [Recurso electrónico]. [Revisado el 29 Abril 2007]. Disponible en: <http://www.jclark.com/xml/expat.html>
- DENENBERG, R., J. KUNZE, y D. LYNCH. 1996. *Uniform Resource Locators for Z39.50*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.rfc-editor.org/rfc/rfc2056.txt>
- DEROSE, S., E. MALER, y R. DANIEL. 2002. *Xpointer xpointer() Scheme. W3C World Wide Web Consortium*. [Recurso electrónico]. [Revisado el 10 Febrero 2007]. Disponible en: <http://www.w3.org/TR/xptr-xpointer/>
- DEROSE, S., E. MALER, R. DANIEL, y P. GROSSO. 2002. *XML Pointer Language (XPointer). W3C Consortium*. [Recurso electrónico]. [Revisado el 27 Marzo 2007]. Disponible en: <http://www.w3.org/TR/xptr/>
- DIETERICH STEFFAN, H. 2001. *Nueva guía para la investigación científica*. Ciudad de México: Ariel.
- *DTD Tutorial, W3Schools*. 2009. [Recurso electrónico]. [Revisado el 30 Marzo 2007]. Disponible en: <http://www.w3schools.com/dtd/default.asp>
- *Dublin Core Metadata Element Set, Version 1.1*. 2008. [Recurso electrónico]. [Revisado el 10 May 2009]. Disponible en: <http://dublincore.org/documents/dces/>
- ESTIVILL RIUS, A. 2006. *Guía de procedimientos para la asignación de metadatos Dublin Core en el marco del proyecto "Temaria, revistas digitales de biblioteconomía y documentación".*

Portal Temaria. [Recurso electrónico]. [Revisado el 9 Agosto 2006]. Disponible en: <http://www.temaria.net/guiaestilo.php>

- *Eurovoc Thesaurus*. 2008. [Recurso electrónico]. [Revisado el 10 Diciembre 2009]. Disponible en: <http://europa.eu/eurovoc/sg/sga_doc/eurovoc_dif!SERVEUR/menu!prod!MENU?langue=EN>

- *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. 2008. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.w3.org/TR/xml/>

- *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. 2008. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.w3.org/TR/xml/>

- FESTA, P. 2003. *Dispute exposes bitter power struggle behind Web logs*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://news.cnet.com/2009-1032_3-5059006.html>

- FESTA, P. 2003. *Newsmaker: Blogging comes to Harvard*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://news.cnet.com/Blogging-comes-to-Harvard/2008-1082_3-985714.html>

- FETZER, T. 2004. *XML mit PHP und Expat parser. Dr. Web Webdesign Magazin.* [Recurso electrónico]. [Revisado el 29 Abril 2007]. Disponible en: <http://www.drweb.de/programmierung/xml-php-expat.shtml>

- FINKE, C. 2009. *Complementos para Firefox: OPML Support 1.5*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <https://addons.mozilla.org/es-ES/firefox/addon/2625>

- FIUMARA, G. 2007. *Automated Information Extraction from Web Sources: a Survey*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-312/bof07-fiumara.pdf>

- FIUMARA, G., M.L. ROSA, y T. PIMPO. 2007. *(X)querying RSS/Atom Feeds Extracted from News Web Sites: a Cocoon-based Portal*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-312/bof07-fiumara-larosa-pimpo.pdf>

- *Formato IBERMARC para registros de fondos y localizaciones*. 2004. Madrid: Biblioteca Nacional.

- FRANGANILLO, J. y M.A. CATALÁN. 2005. Bitácoras y sindicación de contenidos: dos herramientas para difundir información. *bid textos universitaris de biblioteconomia i documentació*.

- *FSF Free Software Directory*. 2009. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://directory.fsf.org/>

- FUENTES, J.J. 1999. *Evaluación de bibliotecas y centros de documentación e información*. Gijón: Trea.

- GARCÍA CAMARERO, E. y L.A. GARCÍA MELERO. 1999. *Automatización de bibliotecas*. Madrid: Arco Libros.

- GARCÍA CAMARERO, E. y L.A. GARCÍA MELERO. 2001. *La biblioteca digital*. Madrid: Arco Libros.

- GARCÍA LÓPEZ, G.L. 2007. *Los sistemas automatizados de acceso a la información bibliográfica: Evaluación y tendencias en la era de internet*. Universidad de Salamanca.

- GARCÍA MARCO, F.J. 2006. La biblioteca digital. *In*: *Manual de Ciencias de la Documentación*, Madrid: Pirámide, pp.651-670.

- GARRIDO ARILLA, M.R. 1993. Estudio de la aplicación de las nuevas tecnologías de la información en las bibliotecas españolas: 1987-1991. *Revista General de Información y Documentación*. 3(2), pp.237-242.

- *Gateway to library catalogs Z39.50*. 2009. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.loc.gov/z3950/>

- *GData JavaScript Client 2.0 Class Hierarchy*. 2009. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://code.google.com/intl/es-ES/apis/gdata/jsdoc/2.0/overview-tree.html>

- *Gentle Introduction to SGML*. 1994. [Recurso electrónico]. [Revisado el 23 Enero 2007]. Disponible en: <http://www.isgmlug.org/sgmlhelp/g-index.htm>

- GETHIN, P. 2001. Automatización de bibliotecas. *El profesional de la información*. 10(11), pp.26-31.

- GIGLIA, E. 2007. PubMed at its best: Useful skills to perform an effective search, save it and automatically receive updates. *Europa MedicoPhysica*. 43(3), pp.427-437.

- GILCHRIST, A. 2007. Can Web 2.0 be Used Effectively Inside Organisations? *Bilgi Dünyası*. 8(1), pp.123-139.

- GONZÁLEZ LORCA, J. y J.V. RODRÍGUEZ MUÑOZ. 2002. La tecnología de flujo de trabajo en el contexto de la biblioteca digital. *Anales de Documentación*., pp.157-175.

- GRUNE, D. y C. JACOBS. 1998. *PARSING TECHNIQUES A Practical Guide*. Chichester: Ellis Horwood.

- GUHA, R.V. y T. BRAY. 1997. *Meta Content Framework Using XML*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.w3.org/TR/NOTE-MCF-XML-970624/>

- GUTIÉRREZ, J.J., M.J. ESCALONA, M. MEJÍAS et al. 2005. *XQuery*. Sevilla: Departamento de Lenguajes y Sistemas Informáticos de la Escuela Técnica Superior de Ingeniería de la Universidad de Sevilla.

- HABIB, M.C. y D. CARR. 2006. *Toward Academic Library 2.0: Development and Application of a Library 2.0 Methodology*. [Recurso electrónico]. [Revisado el 11 Marzo 2007]. Disponible en: <http://etd.ils.unc.edu/dspace/bitstream/1901/356/1/michaelhabib.pdf>

- HAMMERSLEY, B. 2003. *Content syndication with RSS*. Sebastopol: OReilly.

- HAMMOND, T. 2008. *RDF Site Summary 1.0 Modules: PRISM*. [Recurso electrónico]. [Revisado el 10 May 2009]. Disponible en: <http://nurture.nature.com/rss/modules/mod_prism.html>

- HAMMOND, T., T. HANNAY, y B. LUND. 2004. *RDF Site Summary 1.0 Modules: PRISM*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.prismstandard.org/resources/mod_prism.html>

- HAMMOND, T., T. HANNAY, E. NEYLON, y H. SOMPEL. 2004. *RDF Site Summary 1.0 Modules: Context*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://nurture.nature.com/rss/modules/mod_context.html>

- HAROLD, W. 2003. *Using Extensible Markup Language-Remote Procedure Calling (XML-RPC) in Blocks Extensible Exchange Protocol (BEEP)*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.rfc-editor.org/rfc/rfc3529.txt>

- HILL, L.L., G. JANÉE, R. DOLIN et al. 1999. Collection Metadata Solutions for Digital Library Applications. *Journal of the American Society for Information Science*. 50(13), pp.1169-1181.

- HÍPOLA, P. 1994. Las nuevas tecnologías en la Biblioteca Nacional. *El profesional de la información*. 1(7-8).

- *IANA Uniform Resource Identifer (URI) Schemes per [RFC4395]*. 2009. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.iana.org/assignments/uri-schemes.html>

- ICSC. 2008. *Internet Content Syndication: content creation and syndication in an expanding internet universe*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.internetcontentsyndication.org/downloads/whitepapers/content_creation.pdf>

- IDEALLIANCE. 2008. *PRISM: Publishing Requirements for Industry Standard Metadata*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.idealliance.org/industry_resources/intelligent_content_informed_workflow/prism>

- IETF. 2009. *The Internet Engineering Task Force*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.ietf.org/>

- IETF ATOMPUB WORKING GROUP. 2005. *The Atom Syndication Format*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.ietf.org/rfc/rfc4287.txt>

- IFLA (INTERNATIONAL FEDERATION OF LIBRARY ASSOCIATIONS AND INSTITUTIONS). 2007. *International Standard Bibliographic Desciption*. [Recurso electrónico]. [Revisado el 28 Mayo 2008]. Disponible en: <http://www.ifla.org/files/cataloguing/isbd/isbd-cons_2007-en.pdf>

- *International Standard Bibliographic Desciption*. 2009. [Recurso electrónico]. [Revisado el 10 Marzo 2009]. Disponible en: <http://www.ifla.org/isbd-rg>

- *ISO 5127-1. Documentación e información. Vocabulario. Parte1: conceptos fundamentales*. 1983. 50, Comité Técnico AEN CTN.

- JAKSIC, M. 2004. Mapping of bibliographical standards into XML. *Software - Practice and Experience*. 34(11), pp.1051-1064.

- JIMÉNEZ PELAYO, J. y R. GARCÍA BLANCO. 2002. *El catálogo de autoridades: creación y gestión en unidades documentales*. Gijón: Trea.

- JOHNSON, D. 2006. *RSS and Atom in Action: Web 2.0 building blocks*. Greenwich: Manning.

- *JZKit2: Z39.50 Server*. 2009. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://developer.k-int.com/projects.php?page=jzkit2>

- KEITH, C. 2002. *MARCXML: The MARC 21 XML Schema*. [Recurso electrónico]. [Revisado el 10 May 2009]. Disponible en: <http://www.loc.gov/standards/marcxml/schema/MARC21slim.xsd>

- KING, A. 2003. *The evolution of RSS*. [Recurso electrónico]. [Revisado el 22 Agosto 2007]. Disponible en: < http://www.webreference.com/authoring/languages/xml/rss/1/>

- KLYNE, G. y C. NEWMAN. 2002. *RFC3339 - Date and Time on the Internet: Timestamps*. [Recurso electrónico]. [Revisado el 10 May 2009]. Disponible en: <http://www.faqs.org/rfcs/rfc3339.html>

- LIBBY, D. 1999. *RSS 0.91 Spec, revision 3*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.scripting.com/netscapeDocs/RSS%200_91%20Spec,%20revision%203.html>

- LIBBY, D.; NETSCAPE. 1999. *RSS Advisory Board: RSS 0.91 Specification*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.rssboard.org/rss-0-9-1-netscape>

- LIBBY, D. y R.V. GUHA. 2008. *RSS Advisory Board: RSS 0.90 Specification*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.rssboard.org/rss-0-9-0>

- *Library of Congress*. 2009. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.loc.gov/index.html>

- *Library of Congress Blog*. 2006. [Recurso electrónico]. [Revisado el 10 Abril 2007]. Disponible en: <http://blogs.loc.gov/loc/>

- *Library of Congress Online Catalog*. 2009. [Recurso electrónico]. [Revisado el 10 Marzo 2009]. Disponible en: <http://catalog.loc.gov/>

- LÓPEZ DE SOSOAGA TORIJA, A. 1998. OPAC-Web-Z39.50: ¿Redundantes o complementarios? la realidad es multilingüe. *In*: *Jornadas Españolas de Documentación FESABID98*. Valencia.

- LÓPEZ YEPES, J. y J.M. DESANTES GUANTER. 1978. *Teoría de la documentación*. Pamplona: Universidad de Navarra.

- LÓPEZ YEPES, J. (Coord.) et.al. 2004. *Diccionario enciclopédico de ciencias de la documentación*. Madrid: Síntesis.

- LÓPEZ YEPES, J., M.T. FERNÁNDEZ BAJÓN, y J. PRAT SEDEÑO. 2005. Las tesis doctorales en Biblioteconomía y Documentación: Diagnóstico y propuesta de criterios de evaluación. *Documentación de las Ciencias de la Información*. 28, pp.173-187.

- MACÍAS GONZÁLEZ, J. 2005. *Servicios bibliotecarios a través de internet*. [Recurso electrónico]. [Revisado el 10 Mayo 2007]. Disponible en: <http://www.sedic.es/autoformacion/servicios_bibliotecarios/>

- MACÍAS GONZÁLEZ, J. 2005. *Servicios bibliotecarios a través de Internet. Unidad de Autoformación.* [Recurso electrónico]. [Revisado el 17 Diciembre 2006]. Disponible en: <http://www.sedic.es/autoformacion/servicios_bibliotecarios/1_bibliotecas_virtuales.htm>

- *Manual de usuario ISBN*. 2002. [Recurso electrónico]. [Revisado el 16 Diciembre 2006]. Disponible en: <http://www.isbninternational.org/esp/download/ISBNmanualesp.pdf>

- *MARC Standards*. 2009. [Recurso electrónico]. [Revisado el 10 Marzo 2009]. Disponible en: <http://www.loc.gov/marc/marcdocz.html>

- *MARC XML Architecture*. 2004. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.loc.gov/standards/marcxml/marcxml-architecture.html>

- *MARC-XML Schema*. 2009. [Recurso electrónico]. [Revisado el 10 Marzo 2009]. Disponible en: <http://www.loc.gov/standards/marcxml/>

- MARTENS, W., F. NEVEN, y M. GYSSENS. 2008. Typechecking top-down XML transformations: Fixed input or output schemas. *Information and Computation*. 206(7), pp.806-827.

- MCLUHAN, M. 1989. *The global village: transformations in World life and media in the 21st century*. Nueva York: Oxford University Press.

- *Mercury Z39.50 Client*. 2009. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.basedowinfosys.com/projects/mzc>

- *Metadata Authority Description Schema: MADS*. 2008. [Recurso electrónico]. [Revisado el 10 Mayo 2008]. Disponible en: <http://www.loc.gov/standards/mads/>

- *Metadata Object Description Schema: MODS*. 2009. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.loc.gov/standards/mods/>

- *MÉTRICA. VERSIÓN3: Metodología de Planificación, Desarrollo y Mantenimiento de sistemas de información*. 2005. [Recurso electrónico]. [Revisado el 19 Diciembre 2006]. Disponible en: <http://www.csae.map.es/csi/metrica3/index.html>

- MOYA ANEGÓN, F. y J.C. FERNÁNDEZ MOLINA. 1998. *Los catálogos de acceso públicos en línea: El futuro de la recuperación de información bibliográfica*. Málaga: Asociación Andaluza de Bibliotecarios.

- MOZILLA GUMI, JAPANESE MOZILLA USERS GROUP. 2006. *Mozilla developer center: RSS 0.90*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <https://developer.mozilla.org/En/RSS/Version/0.90>

- NETSCAPE. 2001. *My Netscape Network: RSS 0.90 Specification*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.purplepages.ie/RSS/netscape/rss0.90.html>

- *New York Times News Service Syndicate*. 2009. [Recurso electrónico]. [Revisado el 10 Marzo 2009]. Disponible en: <https://www.nytsyn.com/>

- NICHOLSON, S. 2006. The Basis for Bibliomining: Frameworks for Bringing Together Usage-Based Data Mining and Bibliometrics through Data Warehousing in Digital Library Services. *Information Processing and Management*. 42(3), pp.785-804.

- NISO; LOC. 2003. *Information Retrieval (Z39.50): Application Service Definition and Protocol Specification*. [Recurso electrónico]. [Revisado el 10 Marzo 2009]. Disponible en: <http://www.loc.gov/z3950/agency/Z39-50-2003.pdf>

- ORERA ORERA, L. 2002. *Manual de Biblioteconomía*. Madrid: Síntesis.

- ORERA ORERA, L. 2005. *La biblioteca universitaria: Análisis en su entorno híbrido*. Madrid: Síntesis.

- PEIS, E., E. HERRERA-VIEDMA, y J.M. MORALES-DEL-CASTILLO. 2008. Modelo de servicio semántico de difusión selectiva de información (DSI) para bibliotecas digitales. *El profesional de la información*. 17(5), pp.519-525.

- PEÑA HUERTAS, M.J., A. MORENO REQUES, y A.M. RODRIGO ECHALECU. 2002. Servicios bibliotecarios. *In*: *Temario para facultativo de biblioteca*, Madrid: Estudio de Técnicas Documentales.

- PEÑA HUERTAS, M.J., A. MORENO REQUES, y A.M. RODRIGO ECHALECU. 2002. Teoría, concepto y función de biblioteca. *In*: *Temario para facultativo de biblioteca*, Madrid: Estudio de Técnicas Documentales.

- *PHP XML Expat Parser*. 2007. [Recurso electrónico]. [Revisado el 29 Abril 2007]. Disponible en: <http://www.w3schools.com/php/php_xml_parser_expat.asp>

- *PHP: Hypertext Preprocessor*. 2009. [Recurso electrónico]. [Revisado el 28 Octubre 2008]. Disponible en: <http://php.net/index.php>

- *Php: The SimpleXMLElement class Manual*. 2009. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.php.net/manual/en/class.simplexmlelement.php>

- PILGRIM, M. 2003. *How to consume RSS safely*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://diveintomark.org/archives/2003/06/12/how_to_consume_rss_safely>

- PILGRIM, M. 2004. *The myth of RSS compatibility*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://diveintomark.org/archives/2004/02/04/incompatible-rss>

- PILGRIM, M. 2006. *Universal Feed Parser*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.feedparser.org/>

- PINTO MOLINA, M. 1994. *Catalogación de documentos: teoría y práctica*. Madrid: Síntesis.

- PRABOWO, R. y M. THELWALL. 2006. A comparison of feature selection methods for an evolving RSS feed corpus. *Information Processing and Management*. 42(6), pp.1491-1512.

- PRATS, J. y C. TREPAT. 2004. *Técnicas y recursos para la elaboración de tesis doctorales: bibliografía y orientaciones metodológicas*. Barcelona: Universitat de Barcelona, Departament de Didàctica de les Ciències Socials.

- *PRISM Namespace [version 2.0]*. 2008. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.prismstandard.org/specifications/2.0/PRISM_prism_namespace_2.0.pdf>

- *PRISM: Publishing Requirements for Industry Standard Metadata [version 1.2]*. 2004. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.prismstandard.org/specifications/PRISM1%5B1%5D.2.pdf>

- *Radio UserLand: Tune Into Radio*. 2009. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://radio.userland.com/>

- RAGGET, D., A. HORS, y I. JACOBS. 1999. *Document Type Definition*. [Recurso electrónico]. [Revisado el 20 Apr 2007]. Disponible en: <http://www.w3.org/TR/html401/sgml/dtd.html>

- *RDF Site Summary (RSS) 1.0*. 2008. [Recurso electrónico]. [Revisado el 10 Marzo 2009]. Disponible en: <http://web.resource.org/rss/1.0/spec>

- *RDF Site Summary 1.0 Modules: Content*. 2002. [Recurso electrónico]. [Revisado el 10 Marzo 2009]. Disponible en: <http://web.resource.org/rss/1.0/modules/content/>

- *RDF Site Summary 1.0 Modules: PRISM*. 2004. [Recurso electrónico]. [Revisado el 10 Marzo 2009]. Disponible en: <http://www.prismstandard.org/resources/mod_prism.html>

- *RDF Site Summary 1.0 Modules: Syndication*. 2000. [Recurso electrónico]. [Revisado el 10 Marzo 2009]. Disponible en: <http://web.resource.org/rss/1.0/modules/syndication/>

- *RDF/XML Syntax Specification (Revised)*. 2004. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.w3.org/TR/rdf-syntax-grammar/>

- RODRÍGUEZ BRAVO, B. 2002. *El documento entre la tradición y la renovación*. Gijón: Trea.

- RODRÍGUEZ GAIRÍN, J.M., J. FRANGANILLO, E. ABADAL et al. 2006. Sindicación de contenidos en un portal de revistas: Temaria. *El profesional de la información*. 15(3), pp.214-221.
- *RSS 2.0 Specification (version 2.0.11)*. 2009. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.rssboard.org/rss-specification>
- *RSS 2.0 Specification*. 2003. [Recurso electrónico]. [Revisado el 10 Marzo 2009]. Disponible en: <http://cyber.law.harvard.edu/rss/rss.html>
- RUBY, S. 2003. *Anatomy of a Well Formed Log Entry*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.intertwingly.net/blog/1472.html>
- RUBY, S. 2008. *Rss20AndAtom10Compared*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.intertwingly.net/wiki/pie/Rss20AndAtom10Compared>
- RUBY, S. y D. HOPKINS. 2007. *Road Map Atom Wiki*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.intertwingly.net/wiki/pie/RoadMap>
- RUSTY HAROLD, E. 2003. *Processing XML with Java: a guide to SAX, DOM, JDOM, JAXP, and TrAX*. Boston: Pearson.
- SÁNCHEZ DÍAZ, M. y J.C. VEGA VALDÉS. 2002. Bibliotecas electrónicas, digitales y virtuales: tres entidades por definir. *ACIMED*. 10(6).
- SÁNCHEZ TARRAGÓ, N. 2007. Sindicación de contenidos con canales RSS: aplicaciones actuales y tendencias. *Acimed*. 15(3).
- *Schema Tutorial, W3Schools*. 2007. [Recurso electrónico]. [Revisado el 6 Abril 2007]. Disponible en: <http://www.w3schools.com/schema/default.asp>
- SHEPHERD, E. 2007. *Mozilla developer center: RSS Versions*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <https://developer.mozilla.org/en/RSS/Version>
- SHEPHERD, E. 2008. *Why RSS Content Module is Popular - Including HTML Contents*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <https://developer.mozilla.org/En/RSS/Article/Why_RSS_Content_Module_is_Popular_-_Including_HTML_Contents>
- SIERRA BRAVO, R. 2002. *Tesis doctorales y trabajos de investigación científica*. Madrid: Thomson Paraninfo.
- SKARHOJ, K. 2005. *Typo3 Content management system. Free software directory*. [Recurso electrónico]. [Revisado el 28 Abril 2007]. Disponible en: <http://directory.fsf.org/Typo3.html>
- SKRBIC, S. y D. SURLA. 2008. Bibliographic records editor in XML native environment. *Software Practice and Experience*., pp.471-491.
- SOKÓL, T. 2006. *Mozilla developer center: Netscape's RSS 0.91 (Revision 1)*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <https://developer.mozilla.org/en/RSS/Version/0.91/Netscape/Revision_1>
- SPERBER-MACQUEEN, C.M. y H. THOMPSON. 2007. *XML Schema. W3Consortium*. [Recurso electrónico]. [Revisado el 6 Abril 2007]. Disponible en: <http://www.w3.org/XML/Schema>
- STAYTON, B. 2007. *DocBook XSL: The Complete Guide*. Santa Cruz: Sagehill.

- SUBIRATS COLL, I. y J.M. BARRUECO CRUZ. 2002. RcLIS: una biblioteca digital distribuida para documentación. *In*: *III Jornadas de Bibliotecas Digitales*. Madrid: Universidad Politécnica de Madrid.

- SUN MICROSYSTEMS. 2009. *MySQL 5.0 Reference Manual*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://dev.mysql.com/doc/refman/5.0/en/index.html>

- TANIGUCHI, S. 2005. Recording Evidence in Bibliographic Records and Descriptive Metadata. *Journal of the American Society for Information Science and Technology*. 56(8), pp.872-882.

- TANIGUCHI, S. 2006. A System for Supporting Evidence Recording in Bibliographic Records. *Journal of the American Society for Information Science and Technology*. 57(9), pp.1249-1262.

- TANIGUCHI, S. 2007. A System for Supporting Evidence Recording in Bibliographic Records, Part II: What Is Valuable Evidence for Catalogers? *Journal of the American Society for Information Science and Technology*. 58(6), pp.823-841.

- TANIGUCHI, S. 2009. *Shoichi Taniguchi's home page*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.slis.tsukuba.ac.jp/~taniguch/index_e.htm>

- TARANKO, S. 2002. *Content Syndication: Haupseminar Multimediatechnik = Web Content Managment*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http:tudresden.de/Lehre/Archiv/Sommersemester_02/Hauptseminar/Praesentation_Taranko.pdf >

- TENNANT, R. 1999. *Digital v. Electronic v. Virtual Libraries*. [Recurso electrónico]. [Revisado el 19 Octubre 2006]. Disponible en: <http://sunsite.berkeley.edu/mydefinitions.html>

- TENNANT, R., K. SCHNEIDER, H. HAYES, y E. MELTZER. 2008. *The Teaching Library: Rethinking Library Services*. Berkeley: Lulu.com.

- *TERMCAT, Centre de Terminologia*. 2009. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.termcat.cat/>

- *The Atom syndication format*. 2005. [Recurso electrónico]. [Revisado el 10 Marzo 2009]. Disponible en: <http://www.atomenabled.org/developers/syndication/atom-format-spec.php>

- THELWALL, M. 2005. *Web Order Forms with JavaScript*. Wolverhampton: University of Wolverhampton, School of Computing & Information Technology.

- THELWALL, M. y R. PRABOWO. 2007. Identifying and Characterizing Public Science-Related Fears From RSS Feeds. *JASIST Journal of the American Society for Information Science and Technology*. 58(3), pp.379-390.

- THELWALL, M., R. PRABOWO, y R. FAIRCLOUGH. 2006. Are Raw RSS Feeds Suitable for Broad Issue Scanning? A Science Concern Case Study. *JASIST Journal of the American Society for Information Science and Technology*. 57(12), pp.1644-1654.

- THOMPSON, H.S. 2001. *W3C XML Pointer, XML Base and XML Linking*. [Recurso electrónico]. [Revisado el 10 Marzo 2007]. Disponible en: <http://www.w3.org/XML/Linking>

- TRAMULLAS SAZ, J. 1998. Una propuesta de concepto y definición de documentación automatizada. *Revista General de Información y Documentación*. 8(1), pp.263-282.

- TRAMULLAS SAZ, J. 2002. Propuestas de concepto y definición de la biblioteca digital. *In*: *III Jornadas de Bibliotecas Digitales*. Madrid: Universidad Politécnica de Madrid, pp.11-20.

- *UNE 50113-1 Documentación e información. Vocabulario. Parte1: conceptos fundamentales*. 1992. Comité Técnico: AEN CTN 50.
- *UNE-EN ISO 2789 Información y documentación. Estadísticas de bibliotecas para uso internacional (ISO 2789:2003)*. 2004. Comité Técnico: AEN CTN 50.
- USERLAND. 2000. *RSS Advisory Board: RSS 0.91 Specification*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.rssboard.org/rss-0-9-1>
- USERLAND. 2000. *RSS Advisory Board: RSS 0.92 Specification*. [Recurso electrónico]. [Revisado el 25 Diciembre 2009]. Disponible en: <http://www.rssboard.org/rss-0-9-2>
- USERLAND. 2002. *RSS Advisory Board: RSS 2.0 Specification*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.rssboard.org/rss-2-0>
- USERLAND. 2009. *Manila Web Publishing System*. [Recurso electrónico]. [Revisado el 10 May 2009]. Disponible en: <http://manila.userland.com/>
- USERLAND; BERKMAN CENTER. 2003. *RSS 2.0 at Harvard Law: RSS 2.0 Specification*. [Recurso electrónico]. [Revisado el 10 Marzo 2009]. Disponible en: <http://cyber.law.harvard.edu/rss/rss.html>
- USERLAND; BERKMAN CENTER. 2003. *RSS 2.0 at Harvard Law: SOAP Meets RSS*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://cyber.law.harvard.edu/rss/soapMeetsRss.html>
- USERLAND; BERKMAN CENTER. 2004. *RSS2.0 at Harvard Law: RSS History*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://cyber.law.harvard.edu/rss/rssVersionHistory.html>
- USERLAND; BERKMAN CENTER. 2007. *RSS 2.0 at Harvard Law: Relative links*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://cyber.law.harvard.edu/rss/relativeURI.html>
- VALLES, M.S. 1999. *Técnicas cualitativas de investigación social: Reflexión metodológica y práctica profesional*. Madrid: Síntesis.
- W3C CONSORTIUM. 2009. *World Wide Web Consortium (W3C)*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.w3.org/>
- *W3C de la A a la Z*. 2009. [Recurso electrónico]. [Revisado el 4 Septiembre 2006]. Disponible en: <Disponible en: http://www.w3c.es/divulgacion/a-z/>
- *W3C QA Recommended list of DTDs. W3C World Wide Web Consortium*. 2009. [Recurso electrónico]. [Revisado el 30 Marzo 2007]. Disponible en: <http://www.w3.org/QA/2002/04/valid-dtd-list.html>
- *W3Schools Online Web Tutorials. Refsnes Data*. 2009. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.w3schools.com/>
- WALSH, N., L. MUELLNER, y B. STAYTON. 2006. Getting Startedwith SGML/XML. *In*: *DocBook: The Definitive Guide*, Sebastopol: OReilly.
- *Washington Post RSS news feed topics*. 2009. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.washingtonpost.com/wp-dyn/rss/>

- WINER, D. 2001. *Backend Userland: RSS 0.93*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://backend.userland.com/rss093>
- WINER, D. 2001. *Scripting News*. [Recurso electrónico]. [Revisado el 22 Agosto 2007]. Disponible en: <http://www.scripting.com/2001/01/11.html>
- WINER, D. 2003. *Backend Userland: RSS 0.94*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <backend.userland.com/rss094>
- WINER, D. 2003. *Backend Userland: RSS change notes*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://backend.userland.com/rssChangeNotes>
- WINER, D. 2003. *Backend.Userland.Com: RSS 0.92*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://backend.userland.com/rss092>
- WINER, D. 2003. *Backend.Userland.Com: RSS 2.0*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://backend.userland.com/rss094>
- WINER, D. 2007. *OPML 2.0*. [Recurso electrónico]. [Revisado el 10 Mayp 2008]. Disponible en: <http://www.opml.org/spec2>
- WINER, D. 2009. *RSS Advisory Board: RSS 2.0 Specification (version 2.0.11)*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.rssboard.org/rss-specification>
- *XLink and Xpointer Tutorial, W3School.* 2009. [Recurso electrónico]. [Revisado el 26 Marzo 2007]. Disponible en: <http://www.w3schools.com/xlink/default.asp>
- *XML Linking Language (XLink) Version 1.0, W3C Consortium*. 2001. [Recurso electrónico]. [Revisado el 10 Abril 2007]. Disponible en: <http://www.w3.org/TR/xlink/>
- *XML Path Language (XPath) W3C Consortium*. 1999. [Recurso electrónico]. [Revisado el 23 Febrero 2007]. Disponible en: <http://www.w3.org/TR/xpath>
- *XML Schema*. 2005. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.w3.org/2001/XMLSchema>
- *XML Schema Part 1: Structures Second Edition*. 2004. [Recurso electrónico]. [Revisado el 5 Octubre 2008]. Disponible en: <http://www.w3.org/TR/xmlschema-1/>
- *XML Tutorial, W3Schools*. 2009. [Recurso electrónico]. [Revisado el 21 Jan 2007]. Disponible en: <http://www.w3schools.com/xml/default.asp>
- *XPath Tutorial, W3Schools*. 2009. [Recurso electrónico]. [Revisado el 27 Febrero 2007]. Disponible en: <http://www.w3schools.com/xpath/default.asp>
- *XQuery Tutorial, W3Schools*. 2009. [Recurso electrónico]. [Revisado el 17 Marzo 2007]. Disponible en: <http://www.w3schools.com/xquery/default.asp>
- *XSL Transformations (XSLT) W3C Standard*. 1999. [Recurso electrónico]. [Revisado el 10 Mayo 2007]. Disponible en: <http://www.w3.org/TR/xslt>
- *XSL Transformations XSLT Version 2.0*. 2007. [Recurso electrónico]. [Revisado el 10 Marzo 2009]. Disponible en: <http://www.w3.org/TR/xslt20/>
- *XSL-FO Tutorial. 2007. W3Schools.* 2009. [Recurso electrónico]. [Revisado el 3 Mayo 2007]. Disponible en: <http://www.w3schools.com/xslfo/default.asp>

- *XSLT Tutoria, W3Schools*. 2009. [Recurso electrónico]. [Revisado el 23 Febrero 2007]. Disponible en: <http://www.w3schools.com/xsl/default.asp >
- YAHOO NEWS. 2009. *Yahoo News RSS*. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://news.yahoo.com/rss>
- *YAZ Index Data*. 2009. [Recurso electrónico]. [Revisado el 10 Mayo 2009]. Disponible en: <http://www.indexdata.com/yaz>
- *Z39.50 Implementor Agreement: Requesting XML Records*. 2009. [Recurso electrónico]. [Revisado el 10 Septiembre 2009]. Disponible en: <http://www.loc.gov/z3950/agency/proposals/resolution/request-xml-revised-2009.html>

Anexo 2. Índice de tablas

Anexo 3. Índice de figuras