Sistemas Informáticos
Curso 2003-2004

<e-Aula>: Desarrollo de un sistema e-learning basado en estándares IMS

Javier López Moratalla
Iván Martínez Ortiz
Pablo Moreno Ger

Dirigido por:
Prof. Baltasar Fernández Manjón
Dpto. Sistemas Informáticos y Programación

Facultad de Informática
Universidad Complutense de Madrid

ÍNDICE DE CONTENIDOS

Sección I: Introducción y Motivación

1.1.- Acerca de este documento

Este documento se entrega como informe final del trabajo realizado en el ámbito de la asignatura de Sistemas Informáticos correspondiente a la titulación de Ingeniero en Informática de la Universidad Complutense de Madrid en el curso 2003-2004.

El trabajo ha consistido en la participación en un proyecto "e-learning" previamente existente, el sistema <e-Aula>. Este proyecto tiene una doble vertiente de investigación y de desarrollo de productos software para educación en la red y está encuadrado en el Departamento de Sistemas Informáticos y Programación de la Universidad Complutense de Madrid, bajo la dirección del Profesor Baltasar Fernández-Manjón.

Como proyecto de investigación, <e-aula> tiene como objetivos prioritarios el desarrollo de una infraestructura y de un entorno que permita evaluar diferentes propuestas de estandarización. Más concretamente, se pretende analizar el potencial de los estándares educativos en la construcción de entornos de aprendizaje personalizado, mediante un conjunto de sistemas que implementen la infraestructura básica (e.g. creación, acceso y reutilización de los contenidos).

Hasta el inicio del presente curso, los alumnos de este proyecto de Sistemas Informáticos estuvieron asociados al proyecto en calidad de becarios de colaboración. Para el presente curso, se marcaron unos nuevos objetivos de modo que todo el peso del desarrollo y la planificación recayese sobre dichos alumnos.

Este trabajo de Sistemas Informáticos parte con la ambición de presentar una apuesta clara por la que aparentemente es la iniciativa de estandarización dominante, la propuesta del IMS Global Consortium (según las conclusiones obtenidas en los estudios del estado del arte realizados por otros miembros del proyecto <e-Aula>). Cabe destacar que este es un trabajo novedoso en el área del e-learning ya que a día de hoy no hay ningún sistema comercial ni de investigación que ofrezca una implementación completa de las especificaciones IMS. Es decir no existe una implementación de referencia ni un número suficiente de experiencias previas como para poder abordar el trabajo con una metodología definida a priori.

Otro objetivo principal del desarrollo, y a la vez el mayor reto, es conseguir una base arquitectónica estable, robusta y modular. Dicha base deberá permitir a corto plazo implementar y desarrollar el potencial de los estándares IMS y a largo plazo deberá permitir que el sistema se adapte a las modificaciones de los estándares con facilidad (incluso ante el escenario de cambios muy bruscos, suceso posible dada la falta de madurez del dominio). Así, el legado del presente trabajo al proyecto <e-Aula> será un sistema funcional y a la vez una base sobre la que asentar el trabajo futuro del proyecto.

El trabajo realizado dentro de la iniciativa <e-Aula> anteriormente al presente curso ha sentado las bases sobre las que construir un sistema de este tipo y la experiencia acumulada será muy valiosa para acometer el proyecto. Pese a ello, los sistemas construidos anteriormente se basaban en estándares muy distintos a los actuales y, como prototipos de aplicación que eran, no estaban concebidos para permitir sencillas migraciones a nuevas iniciativas. Por ello, en el ámbito del presente proyecto ha sido necesario acometer una re-implementación completa de los trabajos existentes, prestando por supuesto especial atención al diseño de la arquitectura del sistema.

El trabajo realizado ha consistido en la ejecución completa del proceso de desarrollo de un sistema como el anteriormente especificado, incluyendo tomar decisiones sobre la funcionalidad deseada para el sistema, planificar el desarrollo, proponer un nuevo diseño para la arquitectura y re-implementar el sistema por completo.

En este documento se describirá en qué consiste el proyecto <e-Aula>, se detallará el trabajo realizado por los alumnos y se expondrán los resultados obtenidos.

1.2.- About this Document

This document is being delivered as a final report describing the work developed in the scope of the subject *Sistemas Informáticos* under the *Ingeniero en Informática* degree delivered by the Complutense University at Madrid during the 2003-2004 terms.

The work performed has consisted in assisting in an already existing research project in the field of e-learning, the <e-Aula> system. Besides a strong research vocation, the project is also involved in the development of software products related to virtual class environments. The project exists within the SIP Department from the Complutense University at Madrid, under the direction of professor Baltasar Fernández-Manjón.

The main objective of the <e-Aula> initiative is the development of environments and infrastructures to allow the evaluation of diverse standardization proposals. More precisely, the pretension is to evaluate the potential benefits of standardization in the construction of personalized e-learning environments by means of systems that implement basic services (e.g. creation, navigation and reuse of content).

Before the start of the current year, the students developing this project were associated with the <e-Aula> initiative under the role of helper scholarship holders. For the present year new objectives were planned so that all the development weight would be carried by the aforementioned students.

This *Sistemas Informáticos* project sets sail with the ambition of presenting a clear bet in favour of the (apparently) leading initiative in standardization, the IMS Global Consortium initiative (as deduced from the research about the State of the Art conducted by other <e-Aula> contributors). It should be mentioned that this task is full of novelty in the e-learning field because nowadays there are no commercial or research projects offering a full implementation of IMS specifications. That is, there are no reference implementations nor there are enough previous experiences so as to approach the task with a predefined methodology.

Another main objective of this development (and the biggest challenge) is to obtain a stable, modular and robust architectural core. Such architecture should allow in the short term the implementation of IMS standards to their full potential while in the long term it should permit an easy adaptation of the system to whichever modifications the standards might suffer (that must include scenarios with severe modifications of the standards. Such scenarios are possible because the field isn't mature yet). Thus, the present task should leave as a legacy for <e-Aula> both a functioning e-learning system and a solid base on which future work can built on.

Previous experiences within the <e-Aula> initiative have settled the foundations on which a system like this could be built and the collected experience will be very valuable in the development of the system. That said, the implementations of previous <e-Aula> systems were based on standards which were completely different from the standards managed nowadays and, since they were constructed merely as prototypes, they were never meant to allow easy migrations towards new initiatives. As a consequence, given the objectives of the present development, it's been necessary to undertake a full re-implementation of the existing developments, this time with special attention to the design of the architecture so that the new flexibility requisites are met.

The work has consisted in the complete execution of the whole development process of a system with the specified characteristics, including the decision making process regarding the desired functionality, development planification, the proposal of the new architectural design and the full implementation of the system.

This document will describe what the <e-Aula> initiative is about and which tasks were performed by the students. Finally, the obtained results will be discussed.

1.3.- El campo del e-learning

1.3.1.- Los modelos clásicos de aprendizaje

El concepto de proceso clásico de educación es tremendamente amplio pues incluye todas las técnicas empleadas tradicionalmente en los distintos modelos de enseñanza y aprendizaje y a demostrado su éxito a la largo del tiempo. Esto abarcaría un amplio abanico de modelos como pueden ser el modelo presencial clásico (típico en escuelas y universidades), la enseñanza particular (un tutor frente a un único alumno) o la enseñanza a distancia con medios tradicionales (e.g. correo, radio, televisión).

Hoy en día los avances tecnológicos y sobre todo los cambios sociales han hecho que surjan nuevas demandas (e.g. formación continua en la empresa, formación semipresencial) para las que los modelos clásicos presentan carencias e inconvenientes que, como veremos, son salvables mediante el uso de técnicas de e-learning. Por ejemplo, la enseñanza clásica maneja como escenario más típico la figura de un tutor que transmite sus conocimientos a unos alumnos (su clase). Este escenario presenta como inconveniente la enorme dificultad de adaptar el conocimiento transmitido a las necesidades (y capacidades) individuales de cada alumno así como el hecho de que no es aplicable en muchos casos debido a la falta de compatibilidad con el trabajo o a factores de distancia geográfica o tiempo

Otra salvedad que puede observarse en el modelo de un aula y, sobre todo, en el modelo de educación a distancia tradicional (mediante textos específicos con poco o ningún contacto con tutores) es el problema de la falta de interactividad así como de comunicación con otros alumnos y con los tutores. La interactividad con el material estudiado permite al alumno una mayor familiarización con los conceptos, dando lugar a un aprendizaje más eficaz.

1.3.2.- Los modelos e-learning

En la actualidad existen diversos sistemas de enseñanza a distancia a través de Internet desarrollados por todo tipo de instituciones y de empresas comerciales. Estos sistemas se englobarían dentro de la disciplina denominada "e-learning", en la cual se incluirían todos aquellos procesos de aprendizaje en los que las nuevas tecnologías participen directa o indirectamente. Pese a ello, la atención se centra fundamentalmente en procesos de enseñanza en los que las nuevas tecnologías actúan como medio de transmisión sustituyendo así a los medios de transmisión tradicionales y, muy particularmente, en aquellos en los que la red Internet actúa como medio de transmisión principal.

Durante los últimos años se ha producido una revolución en las aplicaciones educativas debido a la adopción generalizada de Internet como plataforma de distribución. La informática educativa ha vuelto a cobrar actualidad y existe un gran interés, tanto de investigación como comercial, ya que el desarrollo tecnológico posibilita, generaliza y simplifica el uso de estos sistemas. Tanto las empresas cómo las universidades y entes públicos, tratan de hacer llegar al alumno una enseñanza de calidad en una competición sin precedentes. Todo esto se debe a la consideración (o suposición) de que este nuevo medio de transmisión es capaz de revolucionar cualquier campo al que se aplique.

Dicha consideración se apoya en el enorme impacto que las nuevas tecnologías han tenido en todos los campos donde existe una transmisión de información. Quizá el ejemplo más notorio sería el impacto que estas tecnologías han tenido sobre los diversos

medios de comunicación como puedan ser la televisión, la prensa o la radio. La entrada de las nuevas tecnologías (y muy especialmente de Internet) en estos medios ha supuesto una revolución a la hora de manejar, clasificar y transmitir la información.

La observación de estos fenómenos indica la posibilidad de que aplicar las nuevas tecnologías al medio de transmisión en los procesos de enseñanza pueda tener un impacto dramático, teniendo como consecuencia un enorme aumento en la flexibilidad, sencillez e incluso en la calidad del proceso de aprendizaje.

1.3.3.- El e-learning en la actualidad

La tendencia actual trata de, aprovechando las nuevas características del medio de distribución, evolucionar hacia modelos de enseñanza más centrados en el alumno de modo que se le pueda ofrecer contenidos de gran calidad y adaptados a sus necesidades, así como un entorno de enseñanza adecuado en el que poder asimilarlos.

La idea generalizada es que los nuevos medios de transmisión facilitan ese tipo de personalización y adaptación lo cual los hace muy deseables. No obstante, siguen identificándose problemas en este tipo de entornos educativos.

Un problema que no debe ser despreciado es la alta tasa de abandono que han encontrado iniciativas anteriores. Esto se puede deber a diversos motivos, uno de ellos que la falta de comunicación con compañeros y profesores erosiona el compromiso del alumno con el proceso educativo.

Otro de los problemas fundamentales es el alto coste de desarrollo de cursos para estos sistemas. El proceso de creación de contenidos educativos de calidad es una labor ardua que requiere la colaboración de expertos en diversos temas (e.g. contenidos, tecnología, didáctica). Y, en un sistema e-learning, la calidad de los contenidos es fundamental. En particular, disponer de contenidos de alta calidad y muy adaptados a las necesidades particulares de cada alumno puede ayudar a reducir el problema anterior de la baja capacidad de retención de estudiantes.

Esto se une a la baja posibilidad de reutilización/adaptación de contenidos cuando cambia algún factor como, por ejemplo, la plataforma o el contexto educativo. El problema puede convertirse en crítico cuando, como hasta ahora, ha sido habitual que contenidos educativos excelentes, desarrollados con enorme esfuerzo económico para una tecnología concreta, se hayan perdido cuando se ha cambiado de plataforma o se ha producido un cambio tecnológico (por ejemplo la evolución desde el video disco interactivo al CD-ROM y, posteriormente, a Internet).

Para paliar este problema, los productores de contenidos educativos tratan de sistematizar la producción de materiales educativos de calidad que puedan ser actualizados, reutilizados y mantenidos a lo largo del tiempo. Para ello hay que vencer las dependencias tecnológicas aunando los formatos de la información para que puedan ser utilizados en cualquier plataforma de aprendizaje.

De estas necesidades básicas surge un nuevo modelo para el diseño de los cursos que tiene su origen en el paradigma de orientación a objetos en programación: el llamado *Modelo de Objetos Educativos (Learning Object Model)*. El modelo consiste, básicamente, en diseñar los cursos como agregados de objetos educativos independientes y reutilizables, a la manera de un puzzle o un mecano.

La combinación de Internet como plataforma con el desarrollo de estos nuevos modelos de diseño de los cursos y la utilización de las tecnologías de marcado (e.g. XML y estándares relacionados), simplifican la creación de nuevos sistemas de enseñanza y proponen potenciar el paradigma educativo basado en la máxima "el alumno es el centro". Se trata, en esencia, de que el proceso de aprendizaje pueda adaptarse a las

características de cada alumno, y no a la inversa como es habitual en los métodos de enseñanza tradicionales.

En resumen, para solventar los problemas de los sistemas e-learning anteriormente descritos, los contenidos y los propios sistemas deben buscar un equilibro entre dos características clave:

1. Contenidos educativos de calidad y *reutilizables*. Hay que incrementar la eficiencia de los mecanismos de reutilización. Existiendo, como existe, una enorme cantidad de material educativo disponible en la red, es absurdo empezar cada curso desde cero. Ser capaces de identificar qué contenidos ya existentes son adecuados para emplearse en un contexto concreto redunda evidentemente en un incremento de la calidad de los cursos.
2. Sistemas de aprendizaje personalizado. Una enseñanza adaptada a las necesidades bajo tres ángulos diferentes: El nivel de conocimiento inicial del alumno, los objetivos de conocimiento del alumno y el método de aprendizaje preferido por el alumno.

Pero estas características son difíciles de alcanzar debido a la heterogeneidad de plataformas educativas y de los sistemas de enseñanza en línea. Esto hace necesaria la aparición de recomendaciones y estándares que permitan el uso e intercambio eficientes de los contenidos educativos. La estandarización de las tecnologías aplicadas al aprendizaje pretende posibilitar la reutilización de recursos educativos y la interoperabilidad entre diferentes sistemas.

1.3.4.- Estandarización en e-Learning

Siendo el e-Learning un campo tremendamente amplio, hay muchos aspectos en los que se podrían centrar los esfuerzos de estandarización. En particular existen dos aspectos fundamentales en los que los procesos de estandarización vienen centrándose en los últimos tiempos.

Un primer aspecto sería la estandarización del modo de funcionamiento de los sistemas e-learning del mercado. Todos los sistemas deberían ofrecer los servicios de una determinada manera. Este aspecto viene a englobar tanto a los esfuerzos por conseguir que todos los sistemas funcionen del mismo modo como a aquellos esfuerzos orientados a fijar ciertos criterios y definiciones como pueden ser el significado exacto de las palabras "Curso", "Tutor" o "Alumno".

Un segundo aspecto es la estandarización del modo en que se crean y organizan los contenidos. En la actualidad este es el aspecto en que más se ha avanzado, fundamentalmente debido a que es más sencillo llegar a puntos de acuerdo en la industria en este dominio que en el anterior.

El ideal que se persigue con los procesos de estandarización de contenidos es conseguir una total interoperabilidad entre todas las iniciativas que se acojan a estos estándares. Esto daría lugar, por ejemplo, a que el sistema de e-learning de la Universidad Complutense de Madrid pudiera ofrecer cursos originalmente publicados en el sistema equivalente de la Universidad de Carnegie Melon. Y, avanzando un paso más, podría darse pie a la aparición de repositorios de contenidos en los que profesionales de la enseñanza publican sus cursos que luego son presentados a través de los distintos sistemas del mercado. En este paradigma, los contenidos podrían ser elaborados por organizaciones profesionales en dedicación exclusiva dando lugar a un incremento en la calidad de los mismos.

Para que sea posible reutilizar de manera global los contenidos educativos entre distintos sistemas y plataformas (interoperabilidad de los cursos), es necesario llegar a un consenso sobre un conjunto de características relativas a los objetos educativos. Se identifican 8 capas sobre las que es necesario establecer estándares para lograr la total interoperabilidad de contenidos:

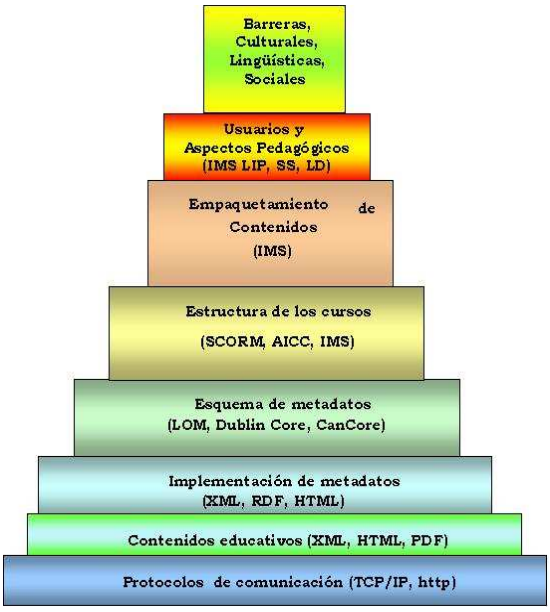


Fig. 1: Esquema representativo de las capas y las iniciativas más relevantes para llegar a la interoperabilidad de contenidos en *e-learning*.

La capa más baja hace referencia a aspectos puramente tecnológicos para las que ya existen estándares aceptados. TCP/IP y HTTP son los protocolos estándar de intercambio de información en la red.

La segunda capa busca homogeneidad en la creación de los contenidos educativos. En este punto no existe aún un consenso claro de qué lenguaje utilizar: XML y HTML son los principales candidatos.

En el caso de los metadatos, XML es la tecnología de implementación más frecuente, siendo considerada ya estándar de facto para esta capa. Entre las características que han convertido a XML en la tecnología más utilizada, vale la pena destacar: la validación automática de documentos, la separación entre contenido y procesamiento, y la independencia de herramientas o plataformas concretas.

En la cuarta capa, los esquemas de metadatos, se decide qué información es relevante para los objetivos del modelo, se agrupa de acuerdo a una serie de categorías, que por lo general tienen carácter jerárquico, y por último, se adjunta al objeto como metadatos (implementados habitualmente con XML).

Las capas quinta y sexta hacen referencia a la necesidad de estructurar los objetos en unidades superiores de contenido (los cursos) y asegurar su portabilidad a través de la red en forma de fichero, aportando toda la información para que sea posible su reconstrucción exacta en el sistema destinatario.

La séptima capa busca la homogeneidad en la estructuración de los perfiles de aquellos implicados en el proceso de enseñanza y en la forma de utilizar didácticamente los recursos educativos.

Por último, la capa de nivel superior aborda los aspectos de adecuación lingüística, cultural y social a distintos contextos. Esta última capa tiene un gran nivel de dificultad y todavía no hay trabajos significativos al respecto.

Observando la figura se puede deducir que de todas las propuestas existentes la que actualmente tiene mas auge es la propuesta de IMS debido a que numerosas empresas e instituciones se han reunido en el consorcio aportando su experiencia en el desarrollo teórico y de herramientas propias para el tratamiento de automatización de enseñanza por computadora. Estas iniciativas se han concretado normalmente con unas descripciones en formato XML que permiten concretar la especificación simplificando el desarrollo de contenidos y sistemas acordes a estándares. De hecho el uso extensivo de estas tecnologías de marcado (e.g. XML) permite también ampliar la funcionalidad y reutilización de la información contenida en el sistema.

1.4.- Otras Iniciativas de estandarización e-learning existentes

Además de la iniciativa de estandarización e-learning IMS, en la que se basa principalmente el sistema <e-Aula>, existen otras entre las que cabe destacar:

ADL (Advanced Distributed Learning): Propone un marco de trabajo y una referencia de implementación detallada en la que utilizar las diferentes propuestas de IMS. Pese a ello, el sistema propuesto por ADL no sigue de una manera estricta los criterios propuestos por IMS.

AICC (Aviation Industry CBT Comitee): Los trabajos del AICC contemplan, entre otros, la definición de requisitos hardware y software para los ordenadores de los alumnos, los periféricos necesarios, los formatos aceptados para los elementos multimedia que componen los cursos, así como recomendaciones para las interfaces de usuario. Otra de sus principales aportaciones es su propuesta para entornos de ejecución.

MIT/OKI/OCW (Open Knowledge Initiative, Open CourseWare): OCW es una iniciativa del MIT para poner todos los contenidos de sus cursos en Internet de forma gratuita. Esta iniciativa también pretende integrar dichos contenidos educativos en una plataforma de enseñanza. Para ello, colabora con la iniciativa Open Knowledge Iniciative (también del MIT), que busca la creación de sistemas de enseñanza abiertos e interoperables. En una fase posterior de este proyecto se pretende que dichos contenidos sigan los estándares propuestos por IMS, lo cual vendrá a potenciar aún más la importancia de dichos estándares.

Los ejemplos anteriormente citados representan los esfuerzos más importantes realizados en el ámbito de la investigación en sistemas e-learning con una cierta orientación hacia los procesos de estandarización.

Cabe señalar que las especificaciones de IMS son muy amplias y no se acompañan de ningún tipo de implementación de referencia. Aunque es posible que en un futuro se proponga una implementación de este tipo, a día de hoy los proyectos de desarrollo de sistemas de e-learning se encuentran con problemas a la hora de decidir como implementar los servicios propuestos en los estándares.

Existen, eso sí, diversos sistemas desarrollados por distintos grupos que, con mayor o menor éxito, implementan alguna parte del estándar. Como hemos mencionado previamente, entre ellos cabe destacar la implementación SCORM propuesta por ADL. Dicha implementación sería lo más parecido a una implementación de referencia que existe en la actualidad, pero su adhesión a los estándares es solamente parcial puesto que en la implementación se toman decisiones de diseño no completamente especificadas por IMS y que otras implementaciones alternativas podrían no ser compatibles. Esto puede provocar una perdida indeseada de información en la transición entre sistemas (aunque, bien es cierto, se sigue simplificando la reutilización aunque no se consigue el objetivo final de una total interoperabilidad).

Por tanto, la elaboración del sistema <e-Aula> debe afrontar la dificultad de ofrecer una interpretación propia del sentido de los estándares, sin poder apoyarse en trabajos previos o implementaciones de referencia. Se puede, eso sí, aprovechar la experiencia en el dominio aportada por diversas iniciativas ajenas a IMS, muy especialmente las versiones anteriores del propio sistema <e-Aula>.

Sección II: El proyecto <e-Aula>

2.1.- Descripción del sistema

Como se ha mencionado en la introducción, el proyecto de investigación <e-aula> tiene como uno de sus objetivos prioritarios el desarrollo de un entorno que permita evaluar diferentes propuestas de estandarización. Más concretamente, se pretende analizar el potencial de los estándares educativos en la construcción de entornos de aprendizaje personalizado, mediante un conjunto de sistemas que implementen la infraestructura básica (e.g. creación, acceso y reutilización de los contenidos). Para lograr estos objetivos se han desarrollado distintos entornos y varios cursos orientados a estudiantes universitarios. Se ha realizado una evaluación formativa tanto de los sistemas como de los cursos con estudiantes de informática de doctorado y de licenciatura

El sistema <e-aula> sirvió en su comienzo cómo evaluador de las especificaciones IMS y EML mediante la implementación de funciones básicas de cada uno de ellos. Actualmente, IMS ha englobado a EML, por lo que <e-aula> se ha centrado únicamente en IMS y ha aumentado el número de funciones implementadas, poniendo así a prueba la propuesta IMS en un sistema real de enseñanza con aplicación en la universidad.

En el presente curso, el objetivo del proyecto <e-Aula> es la planificación, diseño e implementación de un sistema de e-learning (habitualmente denominados *Learning Management Systems* o LMS) accesible desde un navegador web a través de la red Internet. El sistema ofrecería los servicios típicos en una universidad como pueden ser la matriculación, el proceso de inscribirse en un curso, la docencia, el apoyo al alumno por parte de los tutores, etc.

Toda la interacción se realiza a través de la página web del proyecto. Esto incluye las gestiones administrativas, el proceso de estudio de los alumnos, la elaboración de los contenidos por parte de los tutores, la evaluación de los conocimientos adquiridos y la comunicación entre tutores y alumnos.

La web sirve como herramienta de enseñanza y como punto de encuentro de la comunidad de la Universidad Virtual, entendiendo como tal a los alumnos y tutores. Con este fin, junto con las funcionalidades puramente docentes, el sistema incorpora herramientas de comunicación asíncrona (e.g. foros) y síncrona (e.g. Chat).

2.1.1.- Puntos clave

En general existe una gran variedad de sistemas educativos a distancia y, en particular, algunos cuyo concepto se aproxima al del LMS en desarrollo por el proyecto <e-Aula>. Pero, dentro de dicha variedad, el nuevo sistema <e-Aula> ofrece una serie de innovaciones y puntos clave que le destacan:

- Se presta una especial atención al uso de estándares e-learning sobre desarrollo y gestión de contenidos, llegando al extremo de que el núcleo de la arquitectura es una implementación detallada de los procesos definidos en dichos estándares. Como ya se ha mencionado, el uso de estándares permitirá un sencillo intercambio de contenidos con otros LMS, siendo el objetivo final la creación de grandes repositorios de contenidos que serían accedidos por las distintas universidades. El proyecto <e-Aula> se compromete muy estrechamente con los estándares propuestos por el IMS Global Consortium (descritos en mayor detalle en la siguiente sección) en un intento de impulsar el uso de dichos estándares a favor de la noción de una total interoperabilidad.

- Aunque los mencionados estándares emplean XML para determinadas tareas, el LMS desarrollado por <e-Aula> hace un uso intensivo de las tecnologías de marcado en todos los aspectos, ampliando así la flexibilidad del sistema. Estas tecnologías, descritas en mayor detalle más adelante, permiten separar el

contenido de la presentación, facilitando así enormemente el proceso de adaptación del contenido a las necesidades de los distintos usuarios.

- Se persigue conseguir una arquitectura base que sea robusta, estable y modular. Dichas características permitirán en un futuro añadir funcionalidad al sistema y adaptarse a los cambios que pudieran sufrir los estándares.
- Se busca implementar las mayores facilidades de adaptabilidad y comunicación para aprovechar al máximo las posibilidades ofrecidas por la aplicación de las nuevas tecnologías al campo de la enseñanza.
- Se desarrolla íntegramente empleando software abierto. Partes del sistema se distribuirán con licencia GNU quedando el trabajo realizado a disposición de toda la comunidad sin restricciones.
- El sistema se puede gestionar completamente a través de su interfaz web. Todos los procesos de gestión administrativa se pueden realizar mediante un navegador web de forma remota. Así mismo, los contenidos de los cursos y su estructura también pueden ser gestionados por parte de los tutores mediante el mismo mecanismo, aunque no se pretende que los contenidos ofrecidos deban ser creados y redactados de esta forma.

2.2.- Características del sistema

Aparte de los puntos clave tecnológicos y técnicos, se plantean una serie de características (o, si se prefiere, reglas de negocio) que le aportan un estilo propio bien definido. Comentamos algunas de ellas.

2.2.1.- Organización de los cursos

En el sistema <e-Aula> los cursos se organizan según el estándar Content Packaging del IMS Global Consortium (los detalles sobre dicho estándar pueden consultarse en el capítulo de Fundamentos Técnicos de esta misma memoria).

Aunque dicho estándar sólo sugiere una estructuración de los cursos para el proceso de importación/exportación entre sistemas distintos, en este proyecto se ha decidido llevar el seguimiento del estándar hasta el extremo de que los cursos mantengan una estructura estandarizada durante todo su periodo de vida en el sistema. Vamos a explicar brevemente el modelo de estructuración propuesto por dicho estándar.

Los ficheros

Los contenidos del curso se almacenan en un conjunto de ficheros (texto, imágenes, documentos HTML, ficheros XML, etc.). Los ficheros se agrupan sin ningún orden predeterminado, todos juntos. Posteriormente se introduce la noción de recurso.

Los recursos

Un recurso es una estructura compuesta por un fichero principal (el contenido de dicho recurso) y varios (posiblemente ninguno) ficheros secundarios. Dicha estructura posee también un identificador que debe ser único.

Los ficheros referenciados desde el recurso pueden ser locales al curso (se incluyen dentro del paquete y se alojan en el servidor de la web del LMS) o externos (una

dirección URL). Un mismo fichero puede ser referenciado por distintos recursos, como podría ser el caso típico de una figura (fichero secundario) que aparece en distintos recursos.

Los ítems

Los recursos también se almacenan todos juntos, sin definir ningún orden. Para definir la estructura del curso se recurre al planteamiento de un árbol de ítems. Un ítem es una estructura compuesta por un identificador y, opcionalmente, una referencia a un recurso.

Por otro lado, los ítems pueden contener a modo de hijos nuevos ítems, dando lugar así a la mencionada estructura en forma de árbol. Puesto que los ítems pueden referenciar recursos, el árbol de ítems representa efectivamente una estructura en forma de árbol de los recursos del curso.

Las organizaciones

Ya hemos visto que un árbol de ítems da lugar a una estructuración del curso. También se ha mencionado con anterioridad que una de las grandes ventajas del campo del e-learning es la posibilidad de personalización del contenido. Las organizaciones son uno de los mecanismos posibles para dicha personalización.

Una organización es un árbol de ítems. Un mismo curso puede tener varias organizaciones, lo cual implica que puede tener varias estructuraciones distintas del mismo contenido. Esto viene a significar que las distintas organizaciones de un curso son distintas "vistas" del mismo contenido, dando así la posibilidad de modificar el enfoque didáctico con sencillez.

2.2.2.- Uso de tecnologías de marcado

Todos los contenidos de los cursos creados dentro del sistema <e-Aula> se almacenan en ficheros XML separando así la vista del contenido. Cuando se solicita que se muestre por pantalla un determinado recurso que está almacenado en formato XML el primer paso es aplicar una transformación XSL que prepare el contenido para ser mostrado en el navegador del usuario.

Durante el proceso de transformación se puede adaptar el contenido del fichero según las necesidades del usuario, bien por características especiales de su navegador o por características del perfil del usuario.

Así mismo, los procesos de gestión, configuración, implementación, etc. hacen uso intensivo de XML demostrando así una apuesta clara por dichas tecnologías.

2.2.3.- Tipos específicos de contenido

Cada recurso de un curso tiene asociado un tipo. Dicho tipo permite especificar el tipo de tratamiento que hay que dar a cada recurso antes de mostrarlo por pantalla. Los estándares definen ciertos tipos predefinidos pero ofrecen la posibilidad de añadir nuevos tipos mediante mecanismos de extensión del estándar.

En el sistema <e-Aula> se definen cuatro tipos de contenido a modo de extensiones a los estándares: "Portada", "Glosario", "FAQ" y "Página". Esto permite a <e-Aula> definir una estructura propia e imprimir un estilo propio a los cursos elaborados dentro del sistema sin comprometer la portabilidad y compatibilidad.

Las tres primeras extensiones son tres ficheros XML (referenciados por sus correspondientes recursos) que al mostrarse por pantalla (previa transformación XSL

específica de cada tipo) ofrecen funcionalidades específicas como pueden ser un indexado en el caso del glosario.

El cuarto tipo es la definición de una estructura XML de gran potencia expresiva que permite plantear los contenidos del curso en dicho formato.

2.2.4.- Adaptabilidad a las necesidades del alumno

El nuevo sistema del proyecto <e-Aula> trata de ofrecer mecanismos que adapten el contenido y la forma del mismo a las necesidades del alumno. Los principales mecanismos de adaptabilidad actualmente ofrecidos por el sistema son:

- La propia presencia del concepto de organización definido por los estándares IMS descrita anteriormente ofrece una primera alternativa para presentar el contenido de los cursos. Disponiendo cada curso de diversas organizaciones se pueden presentar sus contenidos desde diversos puntos de vista, bien presentándole directamente al alumno la lista de organizaciones para que él mismo elija la presentación más adecuada a sus objetivos o bien siendo el sistema quien selecciona automáticamente la más adecuada en función de la información disponible sobre el alumno (normalmente en base a su nivel de conocimiento).

- Los estándares de IMS contemplan también la definición de prerrequisitos de acceso a determinados recursos. En este sentido es habitual requerir el acceso previo a algún otro recurso, o demostrar un nivel de conocimiento determinado en función de las calificaciones obtenidas por medio de algún sistema de evaluación. Esto permite generar dinámicamente la estructura presentada al alumno en función de su interacción con el entorno educativo.

- El tipo página definido en el sistema <e-Aula> introduce algunas mejoras al modelo educativo, siendo una de las más destacadas la personalización del nivel educativo. Cuando se elabora un documento de tipo página, a cada elemento del documento se le puede asignar un nivel educativo (bajo, medio o alto). El sistema permite mostrar la información con distinto nivel de detalle en función del conocimiento especificado por el usuario. A la hora de mostrar los cursos, al alumno se le ofrece un selector de nivel donde puede expresar el nivel de detalle con el que quiere estudiar el curso. En el paso de transformación y adaptación del contenido se filtrarán aquellos contenidos del documento que el creador del curso haya designado como de un nivel superior al deseado por el alumno. Por el momento, la elección del nivel de conocimiento se realiza manualmente por el alumno, aunque el sistema podría ser adaptado modificado para adaptar automáticamente el contenido mostrado en función de los resultados obtenidos en una evaluación previa.

- Las necesidades de adaptabilidad relacionadas con el alumno en sí y no con su perfil educativo se pueden satisfacer puesto que todos los contenidos almacenados en XML pueden ser adaptados justo antes de ser enviados al cliente. Las mencionadas necesidades podrían incluir desde el simple hecho de que el alumno acceda al curso empleando un navegador no estándar hasta necesidades mucho más complejas como que el alumno requiera mecanismos especiales de accesibilidad.

2.2.5.- Metodología de evaluación

El sistema desarrollado incluye la posibilidad de evaluar el rendimiento de los alumnos mediante la realización de tests de evaluación por parte de los alumnos cuyos resultados

quedan registrados en el sistema. Para este proceso se ha seguido el estándar QTI propuesto por IMS para la elaboración, almacenamiento e interoperabilidad de tests.

El estándar QTI

Esta especificación contempla una estructura básica que describe la forma de representar y gestionar evaluaciones (*assessments*). Su objetivo es conseguir que tanto las evaluaciones cómo los resultados sean intercambiables entre los diferentes LMS. Así, podríamos disponer de almacenes de preguntas y bases de datos con los resultados obtenidos por los alumnos a los que cualquier sistema de enseñanza electrónica podría acceder.

QTI ofrece una descripción y representación muy detallada de los distintos tipos de preguntas que se pueden incorporar en una prueba de evaluación. Define el elemento pregunta, que contiene toda la información referente a una cuestión (e.g. la pregunta en sí, posibles respuestas, pistas para los alumnos y respuesta correcta). Las preguntas se catalogan según el tipo de respuesta que ofrecen (no en contenido, sino teniendo en cuenta la estructura de la respuesta). Por ejemplo, varias respuestas con una sola de ellas válida, o respuestas de ordenación de elementos.

La especificación ofrece una gran cantidad de tipos de respuestas, lo que la hace muy versátil, aunque presenta algunos problemas a la hora de corregir preguntas cuyas respuestas son de tipo texto libre. Hoy por hoy, ese tipo de preguntas sólo pueden ser evaluadas mediante la intervención de un tutor.

Integración de QTI en el LMS

Los exámenes QTI se incorporan a los curso como nuevos recursos y son indexados en las organizaciones. Por tanto, en el flujo normal del estudio de un curso siguiendo una determinada organización los tests son presentados al alumno como un contenido más.

Ya se ha mencionado que cada recurso tiene asociado un tipo. Los tests tienen como tipo asociado "tipo-QTI" lo cual indica al sistema que el recurso requiere un tratamiento especial. En particular, requiere que al finalizar la visita al recurso se debe ejecutar la corrección del test y almacenar la calificación obtenida por el alumno.

Cabe señalar que en <e-aula> el estudio e implementación del estándar QTI no se limita a la incorporación de evaluaciones sino que además se ha creado una herramienta de autoría de exámenes que se puede integrar en la interfaz web del LMS o distribuir independientemente.

2.2.6.- Mecanismos de comunicación

Un proceso educativo donde no existe interactividad entre alumnos y tutores plantea enormes dificultades que se intentan solventar proponiendo una serie de mecanismos que facilitan la comunicación entre los miembros de la comunidad del LMS. Los mecanismos inicialmente facilitados serían:

Foros

Cada curso tiene asociado un foro. En el foro pueden publicar mensajes tanto los alumnos como los profesores, con lo que el foro se convierte en un mecanismo de debate diferido para discutir asuntos relacionados con el material docente del curso. El foro es, por tanto, un medio de comunicación asíncrono, donde la comunicación no es inmediata pero no requiere la presencia simultánea de todos los participantes.

Tablones

Además del foro, cada curso tiene asociado un tablón. El tablón funciona de un modo similar al foro, con la salvedad de que sólo los tutores del curso en cuestión pueden publicar en él. Es por tanto un mecanismo unidireccional de comunicación entre tutores y alumnos.

Chat

En ciertos casos, puede resultar interesante llevar a cabo un debate (entre alumnos o entre alumnos y tutores) más ágil que el permitido por los foros. A tal efecto, se dispone de un medio de comunicación por paso instantáneo de mensajes conocido como "Chat". Este medio permite debatir en tiempo real los asuntos que lo requieran, siendo por tanto un medio de comunicación síncrono.

2.3.- Estructura del LMS

Para la implementación del sistema se ha optado por una arquitectura de sistema web divido en capas. Las características de dicho modelo, así como sus ventajas e inconvenientes se discuten más adelante en la sección de descripción de la arquitectura.

Pero si cabe señalar que el objetivo es conseguir una separación entre los datos (ficheros, bases de datos…), la interfaz mostrada a los clientes y las *reglas de negocio*. Por reglas de negocio se entiende el conjunto de los procesos de interacción del sistema. El usuario interactúa con la interfaz siguiendo los procesos implementados en las reglas de negocio mientras que dichas reglas basan en ocasiones su comportamiento en los datos almacenados en el sistema.

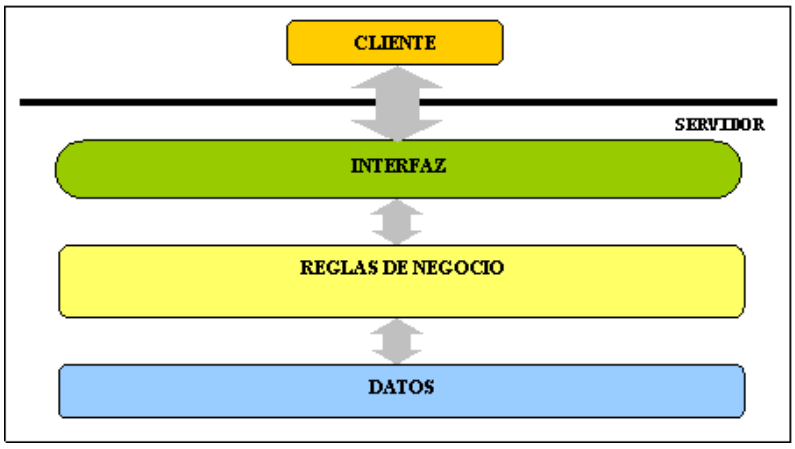


Fig. 2: Modelo de capas para aplicaciones web.

Las distintas características del sistema <e-Aula> mencionadas anteriormente se traducen en módulos situados en la capa de reglas de negocio junto con sus correspondientes elementos de interfaz y datos.

Se consigue así un sistema modular que, inicialmente, contendría los siguientes módulos:

- Administración de usuarios: Sistema de registro de alumnos y profesores, administración de acceso al sistema, registro de actividad de los usuarios, etc.

- Administración de cursos: Asignación de tutores a cursos, publicación de cursos, dar cursos de alta o de baja.
- Importación de cursos: Importación de cursos que sigan el estándar Content Packaging para su publicación en el sistema.
- Autoría de cursos: Gestión de organizaciones, ítems, recursos y ficheros. Edición de los recursos de tipos <e-Aula> secundarios (Portada, Glosario, FAQ)
- Delivery System (explicado en detalle en las próximas secciones): Gestiona el tratamiento de los recursos desde que el usuario selecciona un ítem hasta que es mostrado por pantalla.
- Servicios web: Todos aquellos subsistemas que no están directamente relacionados con el proceso educativo y que son accedidos a través de la propia web del LMS. Ejemplos de servicios web podrían ser los foros o el Chat.
- Servicios comunes: Todos los módulos anteriores emplean un cierto conjunto de funcionalidades (registro de eventos, tratamiento de XML, etc.) que quedan recogidas en la librería de servicios comunes.

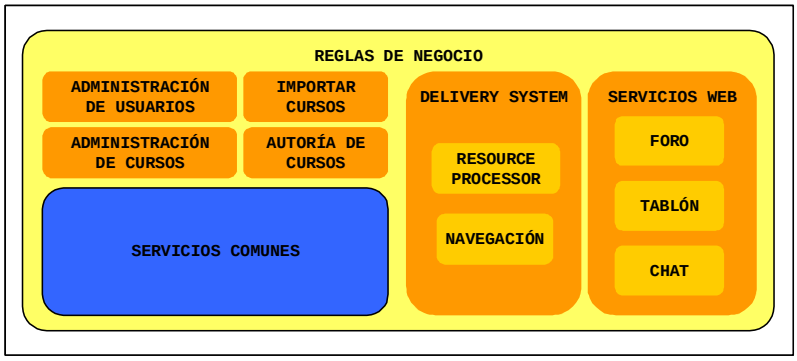


Fig. 3: Los módulos de la capa de reglas de negocio

Sección III: Fundamentos del Proyecto

3.1.- Fundamentos Tecnológicos

En esta sección se describen las tecnologías empleadas en el desarrollo de proyecto. Debido a la noción de que el sistema debe figurar como una clara aportación a la comunidad investigadora en e-learning, uno de los criterios principales a la hora de optar por una determinada tecnología u otra ha sido el requisito de utilizar tecnologías abiertas, preferiblemente ligadas a la iniciativa GNU. Esto se debe a que se pretende publicar aquellas partes del sistema susceptibles de ello bajo la licencia GPL.

3.1.1.- La plataforma J2EE

La plataforma J2EE es una iniciativa de Sun Microsystems consensuada con numerosas empresas e instituciones, que busca ofrecer a los desarrolladores de servicios web una serie de herramientas de trabajo para afrontar las diversas dificultades habituales en este tipo de sistemas.

Pero la iniciativa J2EE no se limita a proporcionar una serie de herramientas sino que también ofrece directrices para diseñar la arquitectura de dichos sistemas. A la especificación J2EE le debemos las definiciones del sistema de capas (capa de presentación, capa de reglas de negocio y capa de datos) que se sigue en el desarrollo del sistema. Estas nociones se explican en mayor detalle más adelante, en la sección de arquitectura del sistema.

API de Java

La plataforma J2EE propone como lenguaje de programación base para los sistemas J2EE el lenguaje Java. El lenguaje java consiste en un núcleo de funcionalidad más o menos estable (tipos básicos, sintaxis…) y por construidos sobre ello, una serie de interfaces para tareas comunes conocidas como *Application Programming Interfaces* (API's).

En realidad, la parte más interesante del lenguaje son precisamente sus API's. Éstas ofrecen al desarrollador toda una serie de funcionalidades ya implementadas simplificando enormemente los procesos de desarrollo. En general, las API's ofrecen implementaciones de las estructuras de datos más comúnmente empleados en programación así como de algoritmos y tareas comunes, funciones de entrada salida, generación de interfaces gráficas de usuario, etc.

Las API's están en continúa evolución y cada nueva versión implementa nuevos elementos dotando al lenguaje cada vez de más potencia. En particular, el LMS de <e-Aula> se ha implementado usando la versión 1.4.2 del lenguaje Java.

Servlets

La plataforma J2EE añade a las API's de Java clases para el manejo de Servlets en el desarrollo de aplicaciones web. Los Servlets son la alternativa propuesta por J2EE para la generación de HTML dinámico generado a partir de código ejecutado en el servidor.

Los Servlets proporcionan un método basado en componentes e independiente de la plataforma para construir aplicaciones web, sin las limitaciones de eficiencia de los programas CGI y, a diferencia de los mecanismos de extensión propietarios de algunos servidores (como la Netscape Server API o los módulos del servidor Apache), los Servlets son independientes del servidor.

Los Servlets son clases Java que se ejecutan en un servidor. Dicho servidor atiende las peticiones web de los clientes redirigiéndolas al denominado contenedor. El contenedor asocia las peticiones web a los distintos Servlets. De este modo, cuando el usuario hace una petición al servidor, el contenedor le pasa el control a un Servlet.

Dicho Servlet ejecutará el código Java escrito por el programador y producirá como salida el texto que el servidor debe devolver al usuario en respuesta a la petición.

La tecnología Java Servlet consiste en dos partes:

- Una API para el lenguaje de programación Java que encapsula las peticiones y respuestas y sus subobjetos y los métodos para manejar dichos objetos. De esta manera una petición a un servidor se traduce a una llamada a un método de un Servlet.
- Un mecanismo declarativo para especificar las propiedades de una aplicación web que son externas al código en sí y que son modificables en tiempo de despliegue (es decir, configurable cuando se instala en el servidor de aplicaciones). De esta manera el mapeo entre peticiones al servidor y los Servlets que responden a ellas se puede modificar en tiempo de despliegue.

Al ser código Java, los Servlets tienen completo acceso a toda la API de Java, incluyendo la JDBC API para acceder a las bases de datos, además pueden acceder a una librería específica de llamadas específicas de HTTP y recibir todos los beneficios del lenguaje Java como son la reutilización de código, la portabilidad, la eficiencia y el control de errores.

Al igual que el lenguaje Java, la tecnología de Servlets está en continua evolución. Por tanto, para mantener un cierto orden y garantizar que las implementaciones de los contenedores de Servlets ofrecen unos servicios mínimos al desarrollador de aplicaciones, Sun define la especificación de Java Servlets, cuya versión 2.3 ha sido utilizada para desarrollar el LMS de <e-Aula>.

Java Server Pages

La programación con Servlets, pese a ser muy cómoda en ciertos aspectos, es muy tediosa y difícilmente mantenible en otros casos.

Como ya se ha dicho, para escribir la respuesta en html desde un Servlet lo que hay que hacer es escribir en un flujo de salida de Java. Por ejemplo:

```
out.println("<html>");
out.println("<head><title>Página de Login</title></head>");
out.println("<body><p>El login se ha realizado
      satisfactoriamente</p></body>");
out.println("</html>");
```

Escribir el código así es sumamente incómodo, más aún cuando el html va acompañado de javascript que hay que depurar. Por ello en Sun se desarrollaron los JSP (Java Server Pages) que, simplificando, no son más que Servlets en los que se asume que, si no se indica lo contrario, todo el código escrito va dentro de un `out.println()`. De esta manera el código anterior se escribiría:

```
<html>
    <head><title>Página de Login</title></head>
    <body><p>El login se ha realizado satisfactoriamente</p></body>
</html>
```

y el efecto sería el mismo, dado que se asume que cada línea va incluida dentro un println(). Para ejecutar código Java que no debería ir dentro de un `println()` se utilizan unos símbolos delimitadores.

De esta manera, la tecnología JSP es una extensión de la tecnología de Servlets creada para soportar la autoría de páginas HTML y XML. Esta tecnología hace más fácil combinar contenido estático (más propenso al uso de plantillas) con contenido dinámico.

Esto agiliza la interacción entre los diferentes roles que intervienen en la creación de una aplicación web, puesto que permite una división clara, dentro del propio código de

un JSP, entre la parte estática (código HTML) y la parte dinámica (código Java entre los símbolos de escape), y por tanto facilita que el diseñador web modifique el código HTML sin necesidad de conocer ni una sola palabra de java.

La tecnología JSP viene acompañada por tecnologías auxiliares como las JSP Tags, que permiten la creación de etiquetas XML propias con código Java asociado. Estas etiquetas pueden ser incluidas en cualquier JSP y producen la ejecución del código Java en tiempo de ejecución del JSP. La principal ventaja es que permiten la creación de pequeñas rutinas reutilizables entre varios JSP's facilitando la mantenibilidad de la aplicación en su conjunto.

Manejo de XML

El manejo de los documentos XML se ha realizado mediante la *Java API for XML Processing* (JAXP).

JAXP hace más fácil el procesamiento de datos XML usando aplicaciones escritas en el lenguaje de programación Java. JAXP soporta los dos estándares para el procesamiento de documentos:

- SAX (*Simple API for XML Processing*): Realiza el procesamiento del documento XML mediante la generación de eventos durante su recorrido.
- DOM (*Document Object Model*): Construye el árbol asociado al documento XML proporcionando al programador los métodos necesarios para su posterior modificación.

Las últimas versiones de JAXP (como la utilizada en <e-Aula>) soportan además el uso del estándar XSLT (XML Stylesheet Language Transformations), permitiendo fácilmente la transformación de datos XML a otros formatos como HTML. JAXP también proporciona soporte para espacios de nombres, permitiendo trabajar con *schemas* que podrían tener conflictos de nombres.

Diseñado para ser flexible, JAXP permite el uso de cualquier parser *XML-compliant* mediante la llamada capa de conectividad. Esta capa permite el uso de cualquier implementación de SAX y DOM facilitando la modularidad del sistema.

En el sistema <e-Aula> se han utilizado como implementación subyacente a JAXP, Xerces para el parser de XML y Xalan como procesador XSL. Siendo los dos proyectos de la Apache Software Foundation. La elección de estas implementaciones se ha realizado siguiendo el doble criterio, mantenido en todo el proceso de desarrollo del sistema, de utilizar componentes que sigan los estándares al pie de la letra y que además sean de libre distribución.

3.1.2.- Apache Server

Apache Tomcat

Como ya se ha dicho anteriormente, para la ejecución de aplicaciones web basadas en Servlets es necesario contar con un contenedor de aplicaciones que es el encargado de transformar todas las peticiones HTTP en llamadas a métodos en las instancias de los Servlets.

El servidor de aplicaciones elegido para la implementación de <e-Aula> ha sido Apache Tomcat, desarrollado por el proyecto Jakarta de la Apache Software Foundation.

El criterio utilizado para la elección de este contenedor de aplicaciones se ha basado en dos motivos fundamentales:

- Tomcat es la implementación oficial de referencia propuesta por Sun para la especificación de contenedor de aplicaciones. La versión 4.1.24 utilizada para el desarrollo del proyecto soporta la especificación 2.3 de Servlets y 1.2 de JSP's. Al utilizar Tomcat sabemos que nuestro código es completamente estándar y debería ser portable a cualquier contenedor de aplicaciones que cumpliera con los estándares J2EE.
- Como el resto de la tecnología utilizada en <e-Aula> Tomcat es software de libre distribución.

Apache HTTP Server

Pese a que con el servidor Apache Tomcat podría ser suficiente para ejecutar el sistema <e-Aula>, por razones de eficiencia hemos decidido combinarlo con un servidor de contenido estático.

La idea de la combinación de servidores es que todo el contenido estático sea servido por el servidor HTTP, a todas luces más eficiente, y todo el contenido dinámico sea servido por el contenedor de aplicaciones. Para configurar este sistema de funcionamiento, se debe definir un *path* (también denominado ruta) a partir del cual se asume que las peticiones son dinámicas (este path es el llamado contexto de Tomcat), y se configura el servidor HTTP para servir todas la peticiones que se le hagan excepto aquellas que vayan dirigidas hacia ese path, que serán reenviadas hacia el servidor de aplicaciones.

El servidor HTTP elegido para la implementación del sistema <e-Aula> ha sido Apache HTTP Server, por motivos similares a los que nos han llevado a la elección de Tomcat como servidor de aplicaciones: Apache HTTP Server está distribuido como código libre, implementa el estándar HTTP 1.1 y además está demostrado que es uno de los servidores HTTP más fiables existentes ahora mismo en el mercado.

3.1.3.- Apache Struts

Pese a que el uso de Servlets y JSP's supuso un gran avance para el desarrollo de aplicaciones web modulares, fácilmente mantenibles y extensibles e independientes de la plataforma (debido al uso de Java), estas tecnologías todavía presentan algunos problemas que es necesario subsanar:

Como ya se ha dicho anteriormente, el uso de Servlets para generar directamente código html es un proceso tedioso muy propenso a errores, y muy difícil de depurar sobre todo cuando el html generado incluye código JavaScript o similar. Esta desventaja parece estar resuelta con la aparición de los JSP's puesto que permiten escribir código HTML de manera limpia y ordenada con todas las ventajas del uso de los Servlets.

Sin embargo el uso indiscriminado de JSP's convierte rápidamente las aplicaciones web en un conjunto de documentos html con código java incrustado independientes entre sí, dificultando el control del flujo de ejecución y reduciendo la reutilización, mantenibilidad y robustez del resultado final.

Debido a estos problemas (que están sobre todo en el uso que se hace de la tecnología y no en la tecnología misma) se han intentando desarrollar diferentes frameworks para aprovechar la potencia de la tecnología Servlet de Java evitando los problemas derivados de un uso incorrecto de ella.

El framework que hemos elegido para el desarrollo de <e-Aula> ha sido Struts, desarrollado por el proyecto Jakarta de la Apache Software Foundation (http://struts.apache.org/) debido la experiencia con otros proyectos relacionados de

Jakarta y a que, como el resto de tecnología utilizada en <e-Aula>, es de libre distribución.

La primera conclusión a la que se ha llegado en Struts, es que para desarrollar aplicaciones web es necesario utilizar tanto JSP's como Servlets de manera conjunta. Los Servlets son apropiados para realizar el control del flujo de ejecución, mientras que los JSP's se deben centrar en las tareas de visualización (que es donde se puede aprovechar su potencia para generar html). Esta división del trabajo entre Servlets y JSP's es conocida como Arquitectura Modelo 2 (en contraposición a la arquitectura centrada en los JSP's que sería el Modelo 1). De hecho, esta Arquitectura Modelo 2 no es más que la aplicación del patrón clásico Modelo-Vista-Controlador al dominio especifico de las aplicaciones web.

En el patrón MVC el flujo de ejecución de la aplicación es gestionado por un Controlador central. El Controlador delega las peticiones (en nuestro caso peticiones HTTP) al manejador apropiado. Los manejadores están colocados directamente sobre el Modelo y cada uno de ellos hace de adaptador entre la petición y el Modelo en sí. El Modelo representa, o encapsula, el estado o la lógica de negocio de la aplicación. una vez realizado el acceso al modelo, el control vuelve hacia el Controlador, que a su vez lo redirige hacia la Vista apropiada. La redirección hacia la vista se suele determinar mediante un conjunto de *mappings* (relaciones), normalmente cargados desde un archivo de configuración o desde una base de datos. Esto independiza completamente el Modelo de la Vista produciendo aplicaciones fáciles de crear, extender y mantener.

La implementación del patrón MVC en Struts presenta las siguientes características:

- Se define la clase `Action` que representa las diferentes acciones que se pueden realizar sobre el modelo. Esta clase tiene un método `execute()` mediante el cual se realiza la acción en si misma.
- Se define la clase `ActionForm`, que representa a los datos enviados por un usuario a través de un formulario HTTP. Esta clase contiene métodos para validar los datos que serán llamados automáticamente por el motor de struts.
- Se define la clase `ActionForward` que representa la redirección desde un Action hacia otro Action o hacia un elemento de la vista. Estas redirecciones se hacen de manera lógica, no física, es decir, que representan abstracciones de redirecciones sin indicar la URL concreta.
- Se define la clase `ActionMapping`. Esta clase representa la relación entre un cierto `ActionForward` y un recurso físico.
- Se define un único Servlet que hace de Controlador.

El funcionamiento de Struts es el siguiente:

1. Cada vez que llega una petición es atendida por el Servlet Controlador, que se encarga de relacionar la URL de la petición con un Action determinado. Esto lo hace mediante un archivo de configuración en el que se encuentran definidos todos los `ActionMapping` del sistema, relacionando las URL's con las clases java que implementan los Actions.
2. Una vez conocido el Action que procesará la petición se crea un objeto de la clase `ActionForm` asociada a ese Action en particular, se rellena con los datos enviados por el usuario y se llama al método `validate()`, que devuelve una lista de errores de validación. Si la lista no es vacía se devuelve el control al JSP o html que generó la petición, en otro caso se continúa.
3. Una vez averiguada la clase del Action que procesará la petición se consigue una instancia de esa clase (los Action son objetos sin estado así que es posible tener un pool de Actions o seguir una política similar) y se llama al método

`execute()`, pasando como parámetros del método todos aquellos parámetros de la propia petición HTTP.

4. La llamada al método `execute()` devuelve un objeto de la clase `ActionForward`, que indica hacia donde se dirigirá el flujo de la ejecución tras la ejecución del Action.
5. Para traducir el `ActionForward` a una URL válida se vuelve a consultar el archivo de configuración.

Como se puede ver, el framework de Struts automatiza tareas rutinarias como la validación de los datos de usuario y facilita enormemente la tarea de control del flujo de ejecución, puesto que esta se define en un archivo de configuración independiente del código. Además favorece la separación entre la vista y los Action que modifican el modelo al obligar a concentrar toda la lógica de negocio en los Action y separarlos de los JSP's mediante los ActionForward.

La distribución de Struts incluye implementaciones por defecto para todas las funciones incluidas anteriormente, incluida la validación de datos de usuarios, aunque es posible (y en el caso de <e-Aula> lo hemos hecho) extender la funcionalidad básica para personalizar el comportamiento de ciertas partes del framework.

3.2.- Fundamentos Técnicos

3.2.1.- Estándares IMS

IMS es un consorcio internacional que ha propuesto un conjunto de especificaciones sobre distintos aspectos que intervienen en el modelado del aprendizaje que utiliza Internet como vía de distribución de información. Su preocupación principal es el problema de la *interoperabilidad* entre distintas plataformas y sistemas de e-learning (Learning Management Systems, LMS).

El núcleo central es la especificación IMS Content Packaging en la cual se describe el modo en el se debe *empaquetar* el contenido educativo para que pueda ser procesado por otro LMS. Esta especificación, junto con el resto de especificaciones del IMS Consortium, pueden consultarse su WEB:

`http://www.imsglobal.org`

En los dos siguientes apartados se dará una descripción de las principales especificaciones de IMS usadas en el sistema <e-Aula>: IMS Content Packaging, IMS QTI, IMS Metadata e IMS Learning Information Package.

IMS Content Packaging

En la especificación IMS Content Packaging se propone el modo en el que deben crear los *paquetes* para distribuir el material educativo, definiendo la estructura del mismo y el formato del contenido. La estructura se plasma en un archivo en formato XML denominado *Manifiesto*[1], donde además se indica el formato del contenido (aunque en general se habla de contenido visualizable en un navegador, se recomienda que dicho contenido esté principalmente en formato HTML).

La especificación IMS Content Packaging propuesta en es lo suficientemente flexible para que pueda ser adaptada a las necesidades particulares de un dominio concreto al que esté orientado un curso o a las necesidades del LMS en particular. ADL-SCORM es un ejemplo de uso y adaptación concreta del IMS Content Packaging para dar soporte y servir como un cierto modelo de referencia tanto a la comunidad que desarrolla LMS, como al os desarrolladores de cursos que siguen estos estándares.

Application Profiles

La excesiva complejidad, generalidad y amplitud de las especificaciones IMS han hecho que se planteen algunas particularizaciones para simplificar su empleo en dominios o aplicaciones concretas. Es lo que se ha denominado perfiles de aplicación (A*pplication Profiles*).

También cabe destacar que los mismos mecanismos que permiten la introducción de los perfiles de aplicación han sido usados para incluir la información necesaria para dar soporte a nuevas especificaciones del propio IMS dentro del IMS Content Packaging. Por tanto, dicha información se incluye dentro del propio manifiesto. Entre estas especificaciones se encuentran, por ejemplo, IMS Simple Sequencing e IMS Learning Design.

El Manifiesto de IMS

Como ya se ha mencionado antes, al distribuir una serie de contenidos en un paquete en el estándar Content Packaging de IMS, existe un documento fundamental que es el Manifiesto. Dicho documento es un fichero XML en el que se describe la estructura de los contenidos incluidos en el paquete. Dicha descripción se realiza a dos niveles diferentes.

Por un lado, se describe cada uno de los *Recursos* del paquete. En una primera aproximación se puede hacer una relación casi directa entre un Recurso y un fichero con contenidos visualizables. En realidad, en cada Recurso se puede incluir información sobre los ficheros que componen dicho Recurso, el tipo de los mismos (que puede ser uno de los tipos ya definidos por el estándar o una extensión de los propuestos) y, opcionalmente, *metadatos* con información adicional sobre el Recurso.

Por otro lado, en el Manifiesto se describe como están organizados dichos Recursos, de modo que se estructura el contenido del paquete. Esto se implementa mediante las *Organizaciones*. Una organización es una vista (o recorrido) de una posible ordenación jerárquica (actualmente en forma de árbol) de los Recursos de un paquete. El estándar permite que un Manifiesto contenga distintas organizaciones sobre los Recursos del paquete, dando así lugar a distintas vistas o "cursos" a partir de los mismos contenidos. El elemento básico de estructuración que se usa al definir las organizaciones son los *Ítems*. A cada Ítem se le puede asociar un Recurso, de modo que el árbol de Ítems es, efectivamente, una estructuración de los Recursos del paquete.

[1] Se trata de un archivo llamado imsmanifest.xml que hay que incluir de manera obligatoria en el paquete.

En resumen, el Manifiesto es un fichero XML que describe, clasifica y organiza los contenidos de un paquete, añadiendo información adicional en forma de metadatos que pueden ser procesados y aprovechados en tareas de catalogación de contenidos.

IMS Question & Test Interoperability (QTI)

La especificación de IMS Questions & Test Interoperability (QTI) describe la estructura básica para la representación de exámenes on-line y su correspondiente informe de resultados. Estos tests siguen la especificación ASI que son las siglas de: *Assessment*, *Section* e *Item*, que son las distintas partes en las que se compone un examen QTI.

A continuación se define cada una de estas partes:

- **Item**: Se usan para la representación de las preguntas individuales. Contienen además de la pregunta en sí, la información para poder presentarla, información para la evaluación de las posibles respuestas, pistas para ayudar al alumno a responderla y metadatos sobre la misma. Los ítems
- **Section**: Se usan para estructurar los exámenes y para ayudar posteriormente al secuenciamiento de las preguntas. Cada sección puede estar compuesta por ítems y otras secciones.
- **Assessment**: Se usa como contenedor de los anteriores y su equivalente sería el examen que trata de determinar los conocimientos del que lo realiza. Tienen que contener una sección como mínimo ya que no pueden albergar ítems directamente. Poseen la información de secuenciamiento de las secciones y gracias a la puntuación que cada ítem lleva asociada, son capaces de genera la puntuación total del test.

También existe el **Object Bank** que define cualquier agrupación ítems o secciones que pueden ser empaquetadas. La definición de este nuevo elemento responde exclusivamente a la idea de facilitar la organización de los exámenes en repositorios.

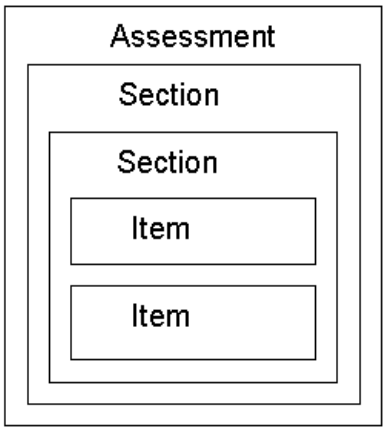


Fig. 4: Estructura de un test QTI

IMS Metadata

La gran complejidad tanto de las estructuras cómo de los contenidos de los recursos de aprendizaje hacen necesaria la aparición de información adicional sobre ambos. Estos son los metadatos, o información sobre los datos. Son etiquetas descriptivas que aportan información orientada a hacer más eficiente la búsqueda y utilización de los recursos. Estas etiquetas se encuentran agrupadas y ordenadas en un conjunto de estructuras, por lo que también pueden intercambiarse entre sistemas que se ajusten a esta especificación.

Esta especificación nos permite realizar una descripción muy detallada de cualquier recurso. Gracias a la cantidad de campos que se nos brindan, obviamente todos opcionales por tratarse de información adicional, podemos añadir una gran cantidad de datos que pueden facilitar las búsquedas, las clasificaciones y la comprensión de todos los elementos implicados en el proceso de enseñanza.

A pesar de las buenas intenciones de la especificación, la flexibilidad que propone conlleva una falta de rigor en todos sus campos, tanto elementos como atributos, que dificulta enormemente el procesamiento automático de los elementos.

IMS Learning Information Package

Esta especificación nos indica la forma de almacenar la información referente a un alumno (o grupo de alumnos) o incluso a un productor de contenido educativo. El objetivo de la misma es definir una estructura que permita el intercambio de paquetes con información relativa a cualquiera de los implicados en el sistema de enseñanza. Junto a esto se conseguiría:

- Llevar un registro del historial, objetivos y logros.
- Conseguir un registro de alumnos y profesores en el sistema.
- Ajustar las oportunidades de aprendizaje a las necesidades del alumno.

3.2.2.- Lenguajes de marcado

XML es el acrónimo de "eXtensible Markup Language", lenguaje de marcado extensible, ¿pero qué significa lenguaje de marcado?

"Un lenguaje de marcas es un conjunto de notas que son usadas para añadir metadatos. Este conjunto de notas que definen una sintaxis y una gramática las cuales pueden considerarse como un lenguaje"

XML

XML es un lenguaje relativamente nuevo, pero está basado sobre un lenguaje de marcas más antiguo denominado *Standard Generalized Markup Language* (SGML). El *Hypertext Markup Language* (HTML) está también basado en SGML, existiendo una nueva versión denominada *Extensible Hypertext Markup Language* (XHTML), que es muy similar a XML. Todos estos lenguajes son usados para definir metadatos, pero SGML y XML pueden ser además considerados como meta-lenguajes, puesto que ellos pueden ser usados para crear otros lenguajes de metadatos.

Los lenguajes de marcado basados en SGML usan cadenas de caracteres denominadas etiquetas (o *tags*) que delimitan componentes de metadatos. Estos tags representan objetos delimitadores y otras marcas.

El World Wide Web Consoritum (W3C) comenzó el proceso de diseño de un lenguaje que combinara la flexibilidad de SGML y la alta aceptación de HTML, este lenguaje es XML. W3C tuvo en cuenta 10 puntos como objetivos de diseño para XML:

1. XML debería ser un lenguaje que pudiera ser usado ampliamente en Internet.
2. XML debería dar soporte a un conjunto amplio de aplicaciones.
3. XML debería de ser compatible con SGML.
4. Debería ser simple escribir programas que procesaran documentos XML.
5. El número de características opcionales en XML debería de mantenerse al mínimo (idealmente cero).

6. Los documentos XML deberían de ser legibles por un ser humano.
7. La preparación del diseño de XML debe ser rápida.
8. El diseño de XML debe ser formal y conciso.
9. Debe ser posible crear documentos XML de manera simple.
10. La longitud de las marcas en XML tiene mínima importancia.

Para obtener mayor información a cerca del estándar que define XML consulte página oficial de W3C en:

http://www.w3.org

y en particular las páginas referentes a XML:

http://www.w3.org/XML

Como características principales del XML tenemos:

- **Extensibilidad**: SGML es altamente extensible y abierto, por tanto XML también lo es. Con XML es posible añadir nuestros propios tags al documento y por tanto incluir datos. XML proporciona una sintaxis básica pero no define los tags a usar, de manera que los tags pueden ser extendidos por cualquiera para sus propios propósitos.
- **Separación de la semántica y de la presentación:** HTML es un lenguaje que define datos y presentación, sin embargo con XML sólo damos una descripción de los datos sin decir nada a cerca de la presentación.

La especificación de XML no solo describe los formatos de los datos sino que también nos proporciona una arquitectura en 2 capas para manejar datos en XML. La primera capa, denominada *XML Processor*, comúnmente denominada XML parser, asegura que los datos en XML están bien formados (tiene la sintaxis correcta con respecto a la especificación de XML) y también puede ser usado para verificar los datos XML con respecto a la estructura definida por el usuario.

El parser debe ser compatible con la especificación de XML, además también es necesario que el parser suministre el contenido y la estructura de los datos XML a la segunda capa (capa de aplicación) siguiendo un formato específico.

El parser XML puede usar un documento separado, comúnmente denominado *schema*, usado para describir y validar un documento XML. Un tipo de esquema que ha sido utilizado es el denominado *Document Type Definition* (DTD) que está incluido en la propia definición del estándar XML.

Transformaciones XSL

La presentación de los datos en marcados con XML son definidos en un documento separado denominado "**hoja de estilo**", la cual usa el lenguaje *Extensible Stylesheet Language* (XSL). Mediante este lenguaje podemos definir reglas para transformar mediante el uso de reglas que asocian un patrón con una plantilla. El funcionamiento de estas reglas es que el patrón es aplicad los nodos del árbol XML origen y si encaja dicho patrón se ejecuta, generando parte del árbol resultante.

Podemos considerar la relación de XSL con XML como la misma relación que tiene las CSS con el HTML.

Los documentos XML pueden ser transformados para ser presentados usando herramientas de conversión denominadas XSL Transformations (XSLT). Estas

herramientas pueden ser usadas en aplicaciones web para transformar los datos en XML en una combinación de HTML y CSS para que pueda ser visualizado en un navegador.

Una hoja de estilo asegura una presentación uniforme con respecto al estilo, que puede ser usada sobre múltiples documentos XML. Por el contrario, un único documento XML puede ser presentado de múltiples maneras, usando distintas hojas de estilo.

Modelos de programación de XML

Existen dos modelos que son ampliamente usados para acceder a datos XML: DOM y SAX.

Un documento en formato XML bien formado se puede interpretar como una vista jerárquica de sus elementos en forma de árbol. Por tanto, podría ser accedido usando rutinas típicas de manejo de árboles. El estándar que define el modelo estructural en forma de árbol para datos en XML se denomina *Document Object Model* (DOM). La API DOM es implementada en diversos lenguajes de programación como C++, Java, etc.

Otra aproximación es que el procesador lea los datos XML secuencialmente y que genere eventos para cada nodo individual, de manera que estos eventos pueden ser usados por la aplicación para procesar los datos. Esta aproximación dirigida por eventos es similar al manejo de eventos en las interfaces gráficas de usuario, y puede manejar documentos XML de gran tamaño. *Simple API for XML* (SAX) define la interfaz estándar para los procesadores que estén dirigidos por eventos.

Sección IV: Especificación del sistema

4.1.- Requisitos del sistema

Los principales requisitos del sistema están directamente relacionados con los elementos que ya se ha destacado anteriormente como puntos clave del sistema. Recordamos brevemente los objetivos principales:

- Dada la línea de investigación principal del proyecto en que se engloba este sistema, se presta una especial atención al uso de estándares e-learning sobre desarrollo y gestión de contenidos.
- Como apuesta del proyecto, se combina el uso de los mencionados estándares con un uso intensivo de tecnologías de marcado (e.g. XML).
- Se busca implementar las mayores facilidades de adaptabilidad y comunicación para aprovechar al máximo las posibilidades ofrecidas por la aplicación de las nuevas tecnologías al campo de la enseñanza.
- El sistema debe poder gestionarse completamente a través de su interfaz web.

Y, por supuesto, no hay que olvidar el papel que el actual desarrollo de este LMS juega en el proyecto <e-Aula>. El objetivo principal para el desarrollo en el presente curso es implementar una arquitectura central estable sobre la que se puedan construir futuras versiones y se puedan evaluar futuras especificaciones.

Para ello es fundamental, en primer lugar, que la arquitectura sea esencialmente modular. Sin ese criterio de modularidad, los futuros cambios que se quisieran hacer debido a modificaciones que pudieran sufrir los estándares serían muy complicados.

Por otro lado, dado que se espera que dicha arquitectura base sea empleada durante mucho tiempo, es fundamental que la implementación sea robusta (poco propensa a errores).

Finalmente, cabe señalar que la continuidad en el tiempo que se espera de esta arquitectura implicará necesariamente que distintos grupos de trabajo tendrán que trabajar con ella por lo que es imprescindible que el código desarrollado sea altamente mantenible.

Manteniendo estos objetivos en mente, procederemos a describir en detalle y de manera formal la funcionalidad del sistema mediante la descripción de los actores y casos de uso del sistema. La implementación de dichas funcionalidades deberá realizarse teniendo siempre en cuenta los conceptos anteriormente expuestos.

4.2.- Actores

Describimos los distintos roles que puede tener un usuario en nuestro sistema, usualmente denominados actores en la terminología de la Ingeniería del Software.

4.2.1.- Alumno

La mayoría de los usuarios del sistema asumen el rol de los alumnos. El objetivo fundamental del alumno es el aprendizaje de los contenidos presentados por el sistema. Como parte de ese proceso un alumno accede al sistema, se matrícula en los cursos que desea y se conecta periódicamente a la web para estudiar los cursos allí publicados.

Además, los alumnos disponen de una serie de medios que les permiten comunicarse con otros alumnos y con sus tutores.

Tal y como se ha concebido el sistema inicialmente, los alumnos no tienen una obligación de conectarse periódicamente a los cursos ni de avanzar en ellos a un determinado ritmo. Así mismo, la evaluación del rendimiento del alumno se realiza mediante la realización de tests en los momentos en que el alumno lo desee.

Dichas libertades de los alumnos se complementan con el rol del tutor quién, sin poseer mecanismos para obligar a un alumno a trabajar, sí posee toda la información sobre el rendimiento del alumno y puede tratar de ayudarle y motivarle.

4.2.2.- Tutor

Los tutores ejercen las funciones del profesor universitario en los modelos clásicos de docencia. Un profesor es responsable de la elaboración o mantenimiento de una serie de cursos. Estos cursos les son asignados por los administradores y cada curso puede tener asignados uno o varios profesores.

Las responsabilidades del profesor con respecto a sus cursos incluyen asegurar la calidad de los contenidos del curso, controlar el progreso de los alumnos matriculados en el curso, ofrecer ayuda a aquellos alumnos que la necesiten y evaluar el rendimiento de los distintos alumnos del curso mediante las herramientas proporcionadas por el sistema.

4.2.3.- Administrador

La figura de los administradores representa la dirección de la Universidad Virtual. Sus responsabilidades tienen dos vertientes diferenciadas similares a las responsabilidades habituales de los organismos de dirección de los centros de enseñanza tradicionales.

Por un lado, un administrador es responsable del funcionamiento del sistema. Esto incluye el proceso de gestión de la web principal. El sistema ofrece unas ciertas posibilidades de configuración que deben ser escogidas por los administradores.

Por otro lado, y con mayor importancia, un administrador es responsable de la organización de los recursos de la Universidad Virtual. Esto implica que los administradores se encargan de decidir que cursos se ofertan en el sistema y designan los profesores responsables de dichos cursos. Dentro de esta vertiente también se incluye la responsabilidad de gestionar los procesos de importación de contenidos provenientes de otras universidades o de repositorios de contenidos.

4.3.- Casos de uso

A continuación se presenta una descripción formal de la funcionalidad del sistema mediante los casos de usos contemplados para la presente versión del sistema <e-Aula>. Dichos casos de uso se presentan resumidos brevemente y, en su conjunto, sirven como descripción con un alto nivel de detalle de la funcionalidad general del sistema.

4.3.1.- Matricularse en la Universidad

Actores implicados: Alumnos

Los alumnos acceden al portal principal del sistema y ahí encuentran un enlace para matricularse en la Universidad Virtual. Al seguir dicho enlace se muestra una página con campos para que el alumno introduzca sus datos personales así como un nombre de usuario y una contraseña. Estos últimos serán empleados para identificar al alumno al acceder al sistema..

4.3.2.- Acceder al sistema

Actores implicados: Alumnos, Tutores y Administradores

Independientemente de su rol, alumnos, tutores y administradores tienen que acceder al sistema como primera acción al visitar la web de <e-Aula>. En la portada principal hay un enlace de acceso. Siguiéndolo aparece un recuadro para introducir el nombre de usuario y la contraseña. Si éstos son válidos, se registra en la base de datos el acceso y se redirige al usuario a su portal inicial correspondiente según sea Alumno, Tutor o Administrador.

4.3.3.- Cambiar datos personales

Actores implicados: Alumnos, Tutores y Administradores

Alumnos, tutores y administradores pueden modificar sus datos personales almacenados en el sistema. Para ello, una vez que han accedido al sistema pueden acceder a la sección de "Modificación de Datos Personales" y allí se les ofrece un mecanismo para modificar dichos datos.

4.3.4.- Matricularse en un curso

Actores implicados: Alumnos

El alumno, tras acceder al sistema, puede acceder a la sección de "Matriculación" donde se le presentará una lista con todos los cursos activos del sistema. Ya que se ha planteado como un sistema abierto, el alumno puede matricularse en cualquiera de los cursos activos. La matrícula se hace efectiva a partir del momento en que el alumno selecciona un curso de la lista. Tras esto, el alumno puede acceder al curso con normalidad.

4.3.5.- Acceder a un curso para su visualización

Actores implicados: Alumnos

Tras acceder al sistema el alumno puede solicitar un listado de los cursos en los que está matriculado. Si el alumno selecciona cualquiera de dichos cursos, automáticamente será redirigido a la página inicial del mismo.

Internamente se registra el hecho de que el alumno ha accedido al curso y se monitoriza su actividad dentro del curso. Opcionalmente existiría la posibilidad de mostrar, en lugar de la página principal, el último recurso visitado por el alumno.

4.3.6.-Visitar un recurso

Actores implicados: Alumnos

Cada curso tiene sus contenidos organizados en una serie de recursos. Una vez que el alumno ha accedido a un curso, se le ofrece un árbol de navegación para desplazarse por los contenidos del curso. El alumno puede ir seleccionando los distintos elementos del curso en dicho menú y se le irán mostrando en el centro de la pantalla los elementos seleccionados. También es posible emplear unas flechas "Siguiente-Anterior" que ofrecen un recorrido lineal del curso.

Internamente, cada vez que el alumno visita un recurso, se registra el hecho de que el alumno lo ha visitado y cuanto tiempo permanece visitándolo. Esto se hace por un lado con fines estadísticos acerca de la utilización del sistema y por otro lado para permitir al tutor de un curso el tiempo empleado por cada alumno en visitar cada recurso.

El proceso de evaluación del rendimiento de los alumnos también se incluye dentro de este caso de uso puesto que los exámenes también son recursos visitables. Al finalizar la visita de dichos recursos, se guarda el tiempo empleado por el alumno en rellenar el cuestionario y la calificación obtenida. Ambos datos pueden ser consultados por los tutores responsables del curso.

4.3.7.- Acceder a un curso para su gestión

Actores implicados: Tutores

Tras acceder al sistema, un tutor puede solicitar la lista de aquellos cursos de los que es responsable. Si el tutor selecciona cualquiera de los cursos accederá al mismo. Dicho acceso queda registrado en la base de datos y al tutor se le ofrece un menú para poder gestionar el curso (modificar su contenido, comprobar los datos de los alumnos, interactuar con ellos, etc).

4.3.8.- Acceder al foro de un curso

Actores implicados: Alumnos y Tutores

Cada curso tiene asociado un foro donde se pueden discutir cuestiones relacionadas con el curso. Los alumnos pueden acceder al foro del curso desde la pantalla de visualización del contenido de un curso mientras que los tutores pueden acceder mediante la pantalla de gestión del curso.

Una vez dentro del foro, la funcionalidad ofrecida es similar a la vista habitualmente en foros de debate en Internet, pudiéndose publicar nuevos mensajes, responder a los ya publicados, etc.

Los tutores de un determinado curso son también moderadores de su foro correspondiente, teniendo a su disposición por tanto herramientas adicional para el ejercicio de la moderación (eliminación de mensajes, edición de mensajes publicados, etc.)

4.3.9.- Acceder al tablón de un curso

Actores implicados: Alumnos y tutores

De manera similar a los foros, cada curso tiene un tablón de anuncios. Los alumnos pueden acceder al tablón desde la pantalla de visualización del contenido del curso mientras que los tutores acceden desde la pantalla de gestión del curso.

Una vez dentro, la funcionalidad es similar a la descrita para los foros, con la salvedad de que sólo los tutores pueden publicar mensajes. Lo único que pueden hacer los alumnos es leer los mensajes publicados.

4.3.10.- Acceder al Chat

Actores implicados: Alumnos y tutores

El acceso al Chat es inmediato puesto que los usuarios ya están registrados y autenticados en el sistema por lo que no es necesario escoger un nombre sino que se accede directamente a las salas de Chat.

4.3.11.- Dar de alta un curso

Actores implicados: Tutores

Cuando un tutor está satisfecho con el contenido de un determinado curso del que es responsable, puede darlo de alta. Cuando un curso es dado de alta pasa a ser visible para los alumnos, que se pueden matricular, acceder a él, examinarse etc.

Cuando un curso está dado de alta, los tutores aún pueden realizar pequeñas modificaciones en su contenido, pero con una serie de restricciones que garantizan que no se creará confusión entre los alumnos matriculados.

4.3.12.- Dar de baja un curso

Actores implicados: Tutores

Cuando un curso ya dado de alta requiere modificaciones severas, existe la posibilidad de darlo de baja antes de acometer los cambios.

4.3.13.- Editar la descripción de un curso

Actores implicados: Tutores

El sistema guarda un título y una breve descripción de cada curso que pueden ser consultados sin estar matriculado. Estos datos se muestran cada vez que se solicita un listado de cursos permitiendo saber de un solo vistazo la temática del curso.

Desde la sección de gestión de cursos, un tutor puede editar estos datos independientemente del resto de los contenidos del curso de modo que no se interfiere con los procesos normales de visualización y gestión por parte de alumnos y tutores respectivamente.

4.3.14.- Editar los elementos secundarios del curso

Actores implicados: Tutores

Los cursos en el sistema <e-Aula> contienen tres elementos especiales definidos como extensiones a los estándares seguidos. Estos elementos, al recibir un tratamiento especial, no se editan del mismo modo que el resto de los contenidos del curso, teniendo cada uno un sistema de edición separado y específico.

Editar la portada del curso

Desde la sección de gestión de un curso se puede acceder al mecanismo de edición de la portada del curso. Desde aquí el tutor puede redactar una breve introducción al curso, gestionar una sección de noticias acerca del curso, editar la bibliografía, artículos relacionados, etc.

Editar el glosario de un curso

Desde la sección de gestión de un curso se puede acceder al mecanismo de edición del glosario del curso. Desde aquí el tutor puede añadir nuevas entradas al glosario, modificar las existentes o eliminar algunas de ellas.

Editar las preguntas frecuentes de un curso

Desde la sección de gestión de un curso se puede acceder al mecanismo de edición de las preguntas frecuentes del curso. Las preguntas se dividen en categorías. Tanto las preguntas como las categorías pueden ser añadidas, modificadas y eliminadas.

4.3.15.- Editar la estructura de un curso (Añadir y eliminar organizaciones)

Actores implicados: Tutores

Ya se ha dicho que en el sistema <e-Aula> los contenidos de los cursos se estructuran en organizaciones. El proceso de edición de un curso se divide por tanto en la gestión de organizaciones y la gestión de los recursos organizados en dichas organizaciones.

Para añadir y eliminar organizaciones el tutor debe acceder a la página de gestión del curso y allí acceder a su vez a la sección de edición de la estructura del curso.

4.3.16.- Editar la estructura de una organización

Actores implicados: Tutores

Ya se ha mencionado que las organizaciones se definen como estructuras en forma de árbol compuestas por diversos ítems. Por tanto el diseño de las organizaciones se basa en las siguientes operaciones:

Añadir Item

Desde la sección de edición de organizaciones, una vez activada la organización deseada, el tutor puede seleccionar cualquier nodo del árbol. Una vez elegido el nodo del que va a "colgar" el nuevo ítem el tutor escoge el emplazamiento exacto entre sus hermanos del nuevo ítem y lo crea en ese lugar.

En este punto se puede dejar el ítem tal cual (como ítem exclusivamente estructural) o se le puede asignar un recurso dotando así al ítem de contenido.

Modificar Item

Modificar un ítem puede consistir en cambiar su nombre, cambiar su identificador o modificar el recurso asociado a dicho ítem. Para realizar esta acción, el tutor accede a la sección de edición de organizaciones y escoge el ítem que desea modificar. Cada ítem tiene asociado un botón de edición. Pulsando en él, se accede a la pantalla de edición de un ítem donde se ofrecen los mecanismos para realizar las operaciones anteriormente mencionadas.

Eliminar Ítem

Para realizar esta acción el tutor accede a la sección de edición de organizaciones y escoge el ítem que desea eliminar. Cada ítem tiene asociado un botón para eliminarlo. Pulsando en él, el ítem correspondiente es eliminado de la organización activa. Si el ítem

tuviese otros ítems como hijos, todo el subárbol correspondiente a dicho ítem será eliminado también.

4.3.17.- Gestionar los ficheros del curso

Actores implicados: Tutores

En última instancia los datos de un curso se almacenan en ficheros. Cada curso contiene una serie de ficheros que posteriormente se enlazarán con los recursos para posteriormente aparecer en las organizaciones. La gestión de los ficheros se basa en dos operaciones básicas.

Añadir un fichero

En la sección de gestión de ficheros se le ofrece al tutor la posibilidad de añadir un fichero a un curso. Dicho fichero será cargado en el servidor desde el ordenador del usuario, mediante la apertura de un cuadro de diálogo que permite al tutor localizar el fichero en el disco de su computadora.

Eliminar un fichero

Si un tutor decide que un fichero no es necesario dentro de un curso específico, puede eliminarlo de la lista de ficheros (y por tanto del servidor) mediante la sección de eliminación de ficheros. Si el fichero está referenciado por algún recurso existente, el sistema impedirá la eliminación.

4.3.18.- Editar un recurso

Actores implicados: Tutores

Como ya se ha dicho, los contenidos del curso se presentan como recursos, los cuales identifican (o enlazan) una serie de ficheros (un fichero principal y una serie de ficheros secundarios) asociándolos con un identificador único y un tipo. Los ficheros pueden ser internos al curso o URL's a ficheros externos. El tipo de un recurso indica al sistema el tipo de tratamiento que hay que darle antes de mostrárselo al alumno. La gestión de los recursos de un curso se basa en las siguientes operaciones:

Crear un recurso

Los recursos de un curso se añaden sin importar el orden puesto que éste ya será descrito por la organización correspondiente. Por tanto, desde la sección de gestión de recursos, se ofrece al tutor un simple botón para crear recursos nuevos.

Tras pulsar dicho botón se crea un nuevo recurso (de momento vacío) que pasa a ser parte del curso y estará disponible en el banco de recursos para enlazarlo con los ítems. Este proceso desemboca directamente en la pantalla de edición de un recurso para que el tutor lo inicialice correspondientemente.

Modificar un recurso

En la pantalla de edición de recursos cada recurso tiene asociado un botón para acceder a la pantalla de modificación de un recurso. Desde esta pantalla se pueden seleccionar los ficheros asociados al recurso (principal y secundarios) así como el identificador del recurso. Dicho identificador debe ser único dentro de cada curso por lo que si ya existe, el sistema lo rechazará y solicitará al usuario que introduzca uno nuevo.

Eliminar un recurso

En la pantalla de edición de recursos, cada recurso del curso tiene junto a él un botón para su eliminación. Pulsando dicho botón el recurso seleccionado es eliminado. Puesto que los ficheros del curso pueden estar asociados a varios recursos, la eliminación de un recurso nunca implica la retirada de un determinado fichero del curso.

4.3.19.- Dar de alta a un tutor

Actores implicados: Administradores

Aunque el sistema es abierto para los alumnos (cualquier alumno puede acceder a la web del proyecto y matricularse) los profesores deben ser aprobados e inscritos por parte de los administradores. Para ello, desde la página principal de administración se puede acceder a la sección de Personal, donde se pueden dar de alta nuevos perfiles de tutor.

Al realizar este proceso, el administrador asigna al nuevo tutor un nombre de usuario y una contraseña. Esta última es de carácter temporal y deberá ser cambiada por el tutor cuanto antes mediante el sistema de modificación de datos personales descrito anteriormente.

4.3.20.- Dar de baja a un usuario

Actores implicados: Alumnos, Tutores y Administradores

Tanto alumnos como profesores tienen la posibilidad de darse de baja voluntariamente de la base de datos del sistema mediante los correspondientes enlaces en las interfaces de ambos roles. Por otro lado, en caso de necesidad, un administrador puede dar de baja a cualquier usuario mediante la sección de gestión de usuarios disponible en la interfaz del administrador.

4.3.21.- Crear un nuevo curso

Actores implicados: Administradores

Es responsabilidad de los administradores dar de alta los nuevos cursos que vayan a estar disponibles en el sistema. Esta acción se lleva a cabo mediante el correspondiente acceso desde la interfaz del administrador. La creación de un curso incluye asignar al nuevo curso un nombre y una breve descripción, así como la designación de los tutores responsables del mantenimiento/docencia de dicho curso.

Al crear un nuevo curso en blanco, se le añaden directamente los elementos complementarios ya mencionados (portada, glosario y FAQ). Dichos documentos se añaden inicialmente en blanco para su posterior edición por parte de los tutores asignados al curso.

4.3.22.- Importar un nuevo curso

Actores implicados: Administradores

Análogamente a la creación de un nuevo curso en blanco, el administrador puede añadir un nuevo curso a la oferta de la Universidad Virtual mediante un proceso de importación de cursos. Dicho proceso se basa en el estrecho seguimiento de los estándares de contenidos que permiten una alta interoperabilidad.

Para importar un curso, el administrador puede acceder a la correspondiente sección y allí abrir un cuadro de diálogo para buscar en el disco de su ordenador un fichero ZIP que contenga un curso empaquetado siguiendo los estándares de IMS.

Tras esto, el sistema comprobará si el curso es correcto, si sigue los estándares y si todos sus contenidos son compatibles con el sistema. Tras este proceso, que puede ir acompañado de interactividad con el usuario para resolver conflictos, el curso queda añadido a la oferta del sistema como si de un nuevo curso se tratase.

4.3.23.- Modificar los tutores responsables de un curso

Actores implicados: Administradores

Un administrador puede modificar en cualquier momento la lista de tutores designados como responsables de un curso. Esto incluye añadir nuevos tutores a la lista o eliminar alguno de los tutores presentes. Para realizar esta operación el administrador accede a un listado de todos los cursos presentes en el sistema, desde la cual puede acceder a la pantalla de gestión de tutores asociados de cada curso.

4.3.24.- Dar de baja un curso

Actores implicados: Administradores

En un momento dado es posible que se tome la decisión de eliminar un determinado curso de la oferta de la Universidad Virtual. En tal caso, un administrador puede acceder al listado de todos los cursos ofertados donde encontrará un botón para eliminar cada uno de los posibles cursos. Como parte de este proceso se puede enviar a los tutores o alumnos inscritos en el curso un mensaje de notificación con, quizá, una explicación de los motivos de la retirada.

Sección V: Arquitectura del LMS

5.1.- Introducción

En este apartado se presenta la arquitectura adoptada en la implementación del sistema <e-Aula> y algunas de las ideas que están detrás del diseño del mismo.

Actualmente existe una tendencia en los LMS hacia arquitecturas basadas en servicios. Estos servicios pueden estar implementados de manera natural con los "web services" o mediante el uso de subsistemas que están integrados dentro la aplicación. En nuestro caso hemos optado por la segunda opción, ya que considerábamos que los servicios que implementamos son internos al sistema <e-Aula> y no se contemplaba el interactuar con otros sistemas.

Como ya se ha destacado, uno de los problemas en el diseño del sistema ha sido la falta algún modelo de referencia para las especificaciones con las que estábamos trabajando, de manera que hemos tenido que desarrollar un proceso iterativo del diseño del sistema para adaptarlo a nuestras necesidades. Sin embargo si que es de destacar que en el diseño hemos podido reutilizar la experiencia obtenida en el tratamiento de especificaciones de sistemas de enseñanza, ya sean del consorcio IMS o de la iniciativa propuesta por la universidad Open University of the Netherlands plasmada en la iniciativa EML, por parte de versiones anteriores del sistema intentando incluir sus aciertos y evitar los problemas encontrados en los mismos.

En el siguiente apartado se da una visión general de los modelos de arquitectura de las aplicaciones WEB basadas en J2EE y en particular las que usan Java Servlets y Java Server Pages. Con esta visión general queremos hacer constar los problemas de implementar una aplicación WEB de escala media-grande como el sistema <e-Aula> y las razones de la elección de nuestra arquitectura y la adopción de Apache Struts como solución tecnológica.

5.2.- Aplicaciones WEB basadas en Java Servlets y Java Server Pages (JSP)

En el desarrollo de aplicaciones WEB basadas en Java J2EE pueden usarse diversos componentes para la implementación de las mismas, algunos de ellos pueden ser: Java Servlets, Java Server Pages, JSP Tag Libraries, Java Beans o incluso usar componentes como los Enterprise Java Beans. Como ya se ha mencionado en la sección de Fundamentos Tecnológicos, los dos modelos de arquitecturas comúnmente usados son el **Modelo 1** y el **Modelo 2**.

5.2.1.- Arquitectura Modelo 1

Una aplicación web basada en la arquitectura propuesta Modelo 1 está compuesta por un conjunto de páginas con las que el usuario interactúa. La lógica de la aplicación reside en estas páginas, normalmente implementadas mediante JSP, accediendo directamente a los datos. En esta arquitectura el cliente accede directamente a las páginas servidas por el servidor de aplicaciones, de manera que estas páginas son usadas para dar servicio tanto a la petición del usuario como a la generación de la respuesta para el mismo.

Este tipo de arquitectura permite implementar de manera simple y rápida una aplicación web, pero sin embargo puede acarrearnos numerosos problemas cuando la funcionalidad aumenta. Algunos de estos problemas son:

- **Mantenibilidad:** Si se realiza una modificación de la aplicación web tendremos que buscar por toda la aplicación y eliminar, o cambiar, los nombres de las páginas.
- **Extensibilidad:** El añadir / modificar la funcionalidad es complejo debido a que las páginas implementan diversa funcionalidad.
- **Seguridad:** Si nuestra aplicación está dividida en una parte pública y una parte restringida necesitaremos incluir el código que lo compruebe en cada página restringida para verificar que un usuario ha hecho login en nuestro sistema.

5.2.2.- Arquitectura Modelo 2

El Modelo 2 intenta paliar estos problemas. Trataremos de presentarlo de forma general para luego observar como podemos beneficiarnos de sus características

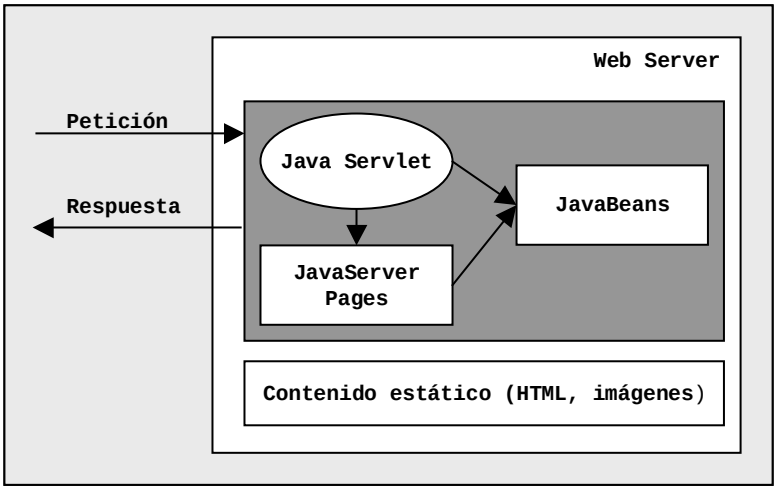


Fig. 5: Arquitectura Modelo 2 genérica

El Modelo 2 está basado en el patrón Modelo-Vista-Controlador. Al mencionar el Modelo 1 hemos visto que en esta arquitectura las páginas son accedidas directamente, sin embargo en el Modelo 2 todos los accesos se realizan a través de un Controlador (normalmente implementado por un Servlet).

Con esta arquitectura podemos usar el mismo **controlador** para todo el sitio web o usar múltiples controladores donde cada uno es responsable de las secciones en las que esta dividida la aplicación.

Cuando un controlador procesa una petición, éste realiza modificaciones sobre el modelo de datos subyacente. Una vez que la petición ha sido procesada, el controlador delega la tarea de generar la respuesta a la **vista**. La vista normalmente contiene una cantidad mínima de código cuyo fin es la generación de la información que se presentará.

Las ventajas que obtenemos con esta arquitectura son las siguientes:

- **Aumento de la mantenibilidad:** En esta arquitectura el componente que implementa el controlador es el responsable en determinar que página de la aplicación web debería ser la siguiente. De esta manera la estructura de la aplicación web está definida en un único lugar.
- **Promocionar la Extensibilidad:** La división entre la lógica de procesamiento de peticiones y la lógica necesaria para generar el contenido, permite la definición de componentes que pueden ser reutilizados.
- **Seguridad:** Ya que las peticiones pasan todas a través del controlador, podemos usar esta propiedad para centralizar la seguridad en un componente eliminando la dispersión de dicha funcionalidad además que evita el problema fallos de seguridad debido al olvido de la inclusión del código que comprueba la seguridad.

5.3.- Arquitectura por capas

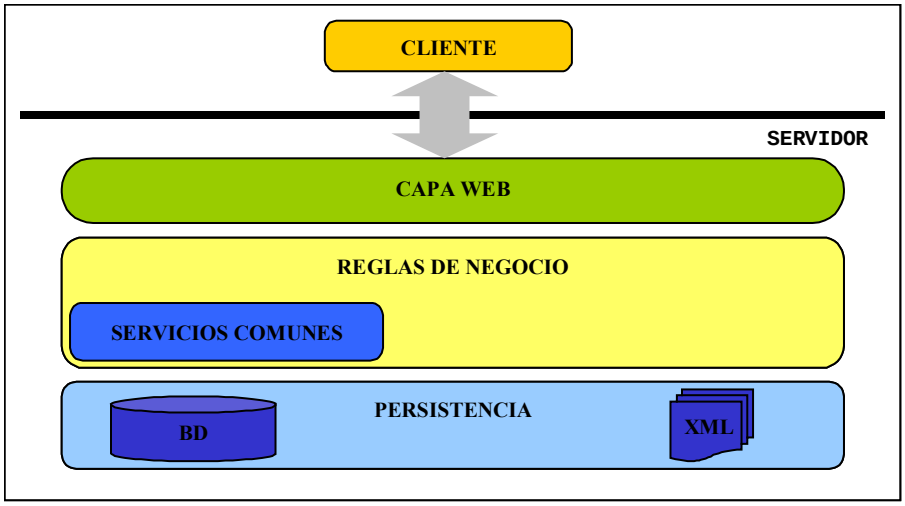


Fig. 6: Arquitectura por capas

En la figura que se encuentra sobre estas líneas se presenta la división por capas diferenciadas del sistema <e-Aula>, a continuación se da una descripción de las tareas que se realizan en dichas capas:

Cliente. En esta capa se encuentra el navegador del cliente mediante el cual se interactúa con el sistema. En esta capa se encuentra el contenido WEB con el que el usuario interactúa, actualmente páginas XHTML, junto a la definición de estilos correspondiente mediante el uso de CSS y código Javascript contenido en las páginas usado para enriquecer la interacción con el usuario.

Capa WEB. Esta capa es la encargada de 2 tareas:

1. Generar las páginas que visualizará el usuario. El contenido visualizado será generado mediante JSP y JSP TagLibraries además de por el uso de transformación XSLT cuando se procesa contenido en formato XML.

2. Procesar y validar los datos suministrados por el usuario para las capas inferiores. El proceso de validación de datos se realiza mediante el uso de ActionForm de Apache Struts..

Reglas de negocio. Esta capa se encarga de procesar las peticiones que el usuario realiza a la aplicación e implementa los servicios necesarios para dar soporte a la funcionalidad requerida para la aplicación. Las peticiones son gestionadas mediante los Action de Apache Struts, que se apoyan en el resto de servicios implementados en esta capa para dar servicio a las peticiones del usuario.

Servicios comunes. Subcapa de las reglas de negocio. En esta capa se implementan las utilerías necesarias para que la implementación de los servicios de las reglas de negocio sea más simple. Estas utilerías son proporcionadas por librerías desarrolladas dentro del proyecto y librerías auxiliares de proyectos de código abierto.

Persistencia: En esta capa se gestionan el almacenamiento de los datos de la aplicación. En nuestro caso tenemos 2 tipos de almacenamiento:

- Almacenamiento en una base de datos relacional para los datos que necesitamos consultar / actualizar rápidamente en el procesamiento de las peticiones.
- Almacenamiento en el sistema de ficheros para datos de gran tamaño como ficheros HTML, XML, imágenes, etc.

En las dos siguientes secciones se describe con mayor detalle pero de manera conceptual la capa de reglas de negocio y la subcapa de servicios comunes.

5.4.- Reglas de Negocio

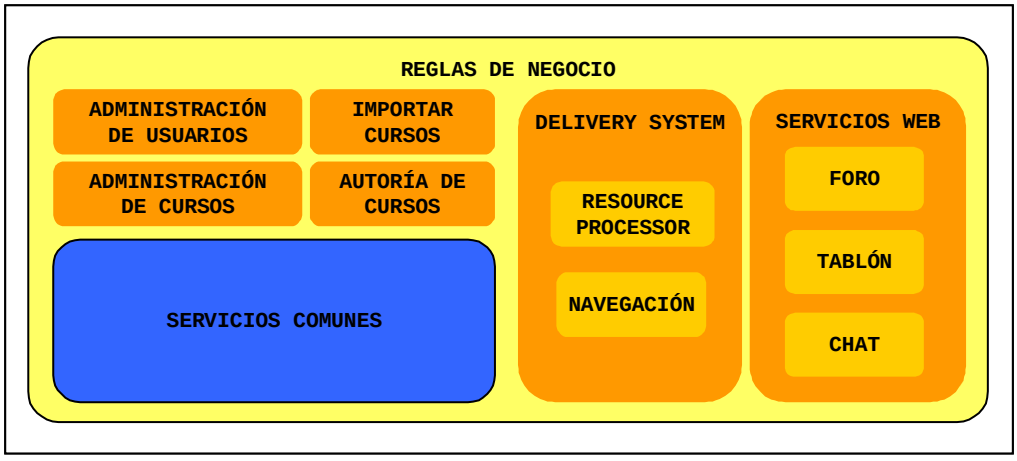


Fig. 7: Reglas de negocio

Como se ha mencionado en el apartado anterior, en esta capa se implementa la lógica propiamente dicha del sistema <e-Aula>. Para promover la modularidad del sistema, la aplicación está subdividida en distintos subsistemas que se han ido desarrollados durante el proyecto. A continuación se detalla la funcionalidad de los mismos.

5.4.1.- Administración de Usuarios

Este servicio es el encargado proporcionar la funcionalidad necesaria para gestionar la información de todos los usuarios, como puede ser su nombre real, nombre de usuario, dirección de correo electrónico, etc. Existe información que es común a todos los roles existentes en la aplicación e información específica de cada rol.

Por ejemplo, un usuario con rol tutor contiene información sobre los cursos que está impartiendo. Otro ejemplo es el caso de los usuarios con rol alumno, para los que se guarda información acerca de su interacción con el contenido de los cursos que visita (tiempo que ha estado visitando una página, número de visitas, última pagina visitada).

Además de almacenar la información también se encarga de verificar si un usuario tiene los permisos suficientes para realizar una acción, de manera que la esta capa es la encarga de llevar a cabo las tareas de autenticación y autorización.

5.4.2.- Administración de Cursos

Este servicio se encarga de proporcionar la funcionalidad necesaria para gestionar los cursos que existen en el sistema, tanto cursos que actualmente se están impartiendo, como cursos que se encuentran aun en desarrollo. Por ejemplo la modificación de la descripción del curso, su nombre, etc. Además también se encarga de ofrecer la funcionalidad necesaria para matricular un alumno en dichos cursos y asignar profesores para impartirlos.

5.4.3.- Importación de Cursos

Este servicio es el encargado de procesar un PIF (para más información consulte el apartado acerca del sistema de importación en la siguiente sección) para que el sistema sea capaz de impartir el contenido de dicho PIF. Este proceso de importación es una tarea compleja en la se procesa la información contenida en dicho PIF para averiguar, entre otras cosas, sobre que estándares de IMS contiene información

Este servicio está basado en un sistema de simulación de eventos discretos basado en agenda de manera que este sistema permite amplia adaptabilidad. Si desea más información a cerca de esta técnica puede consultar el libro *Structure and Interpretation of Computer Programs* (ver bibliografía).

5.4.4.- Autoría de Cursos

Este servicio proporciona a los tutores una herramienta web para editar los cursos que imparten. Las herramientas proporcionadas son:

- Herramienta de edición para modificar el título y la descripción de un curso.
- Editor gráfico para definir la estructura del curso. Sirve para editar las organizaciones del manifesto que define el curso o añadir nuevas organizaciones.
- Editor de las FAQ del curso. Esta herramienta permite editar de manera online el recurso <e-Aula> que es usado para almacenar las preguntas más frecuentes de un curso.

- Editor de la portada del curso. Esta herramienta permite editar el recurso <e-Aula> usado para definir la información sobre recursos auxiliares, libros de texto e información general sobre el curso que pueda resultar interesante para el curso.
- Editor del glosario del curso. Permite editar el glosario de términos que tiene un curso.

Como se ha indicado, este conjunto de herramientas está disponible para los profesores asociados a un curso, y ya que pueden existir varios, las herramientas están desarrolladas de manera que permiten el uso concurrente de cada una de ellas. Por uso concurrente se entiende que pueden existir distintos tutores que editen el curso a la vez pero cuando un tutor usa una herramienta, la usa en exclusiva hasta que termina con ella.

Además el sistema de autoría también hace uso del ResourceProcessor (véase más en el apartado siguiente), pero en vez de para generar la vista para cada tipo de recurso, es usado para delegar la edición de un recurso determinado a una herramienta que soporte la edición de ese tipo de recurso.

5.4.5.- Delivery System

Este servicio proporciona la maquinaria necesaria para permitir al usuario la navegación y visualización de un curso que está en nuestro sistema.

Como se ha mencionado en apartados anteriores los recursos pueden ser de distinto tipo. Como tipo, entendemos que dos recursos son de distinto tipo, si la lógica necesaria para presentarlo al usuario es distinto.

Por ejemplo, la lógica necesaria para procesar el tipo **webcontent** que es el asociado a contenido directamente visualizable en el navegador, simplemente devuelve el contenido de la página al usuario. Sin embargo el tipo de contenido **qti**, es usado para los exámenes que siguen la especificación IMS QTI y la lógica necesaria para soportar este tipo de recurso es la lógica necesaria para realizar un examen a través del navegador.

El sistema debe manejar estos tipos de contenido desde dos perspectivas. La primera es que el sistema de saber manejarlo de manera de manera abstracta como contenido educativo para poder secuenciar el mismo, es decir, para permitir al alumno que lo pueda seleccionar. También es necesario que el sistema sepa como presentar dicho contenido ya que la finalidad es que el contenido teniendo en cuenta las características del mismo

Navegación

Este servicio es el encargado de mostrar al alumno la estructura del curso que está visualizando, además de implementar la maquinaria necesaria para manejar una petición sobre uno de los ítems de dicho manifiesto.

ResourceProcessor

Este servicio es el encargado de lidiar con el problema de la presentación de los distintos tipos de contenido. El sistema de navegación relega la tarea de visualización a este servicio, el cual se vale de la información incluida en el manifiesto para delegar al componente específico que soporta ese tipo de contenido para que genere la página WEB que será mostrada al alumno.

5.4.6.- Servicios WEB

Este conjunto proporcionan canales de comunicación a la comunidad educativa. Cada curso tiene un foro donde los alumnos pueden plantear las dudas que tengan del mismo de manera que otros alumnos o los tutores puedan responder a dichas dudas.

El caso de del tablón de anuncios es un caso similar al foro, la única diferencia es que en él solo pueden escribir los tutores del curso. Este canal puede usarse para informar a los alumnos matriculados de noticias sobre el curso o para suministrar información relacionada con el mismo.

El Chat puede usarse como medio de comunicación privado, por ejemplo si se desea realizar una tutoría online.

5.5.- Servicios Comunes

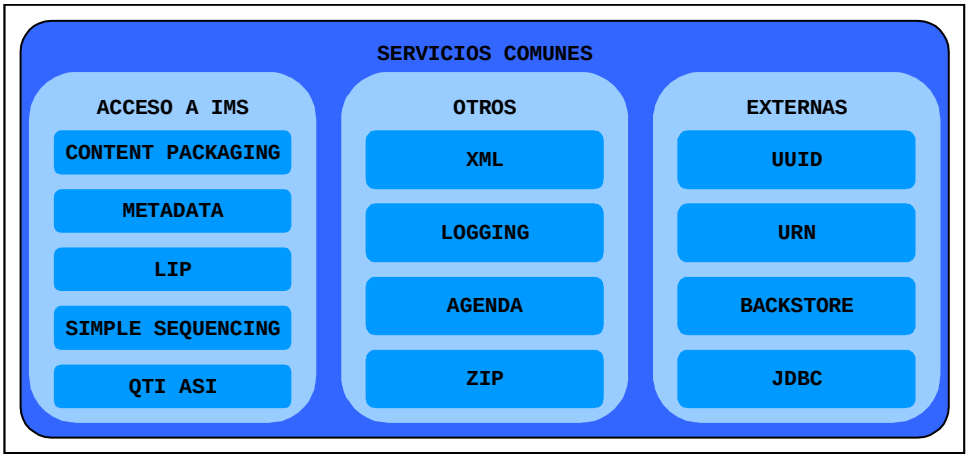


Fig. 8: Servicios comunes

Los servicios presentados en la esta figura son denominados servicios comunes debido a que son servicios independientes del sistema <e-Aula>. Estos servicios pueden considerarse como librerías de utilería para los servicios que implementan la funcionalidad específica del sistema, pero que pueden usarse en otros sistemas.

5.5.1.- Acceso a IMS

Como ya se ha mencionado, IMS contiene numerosas y complicadas especificaciones. Debido a este motivo durante el proyecto se ha desarrollado un conjunto de componentes para localizar toda la lógica necesaria para trabajar con los estándares de IMS.

Actualmente está implementada una librería que proporciona una API completa para la especificación IMS Content Packaging 1.1.3 al completo. Para más información véase el apartado sobre implementación de IMS CP en la siguiente sección.

5.5.2.- XML

En el sistema trabajamos con las API DOM y SAX para el tratamiento de XML, aunque principalmente con DOM debido a que necesitamos la estructura y la visión jerárquica

que nos proporciona esta API. DOM resulta incómodo para realizar algunas tareas de acceso a la información que contiene, de manera que hemos implementado una pequeña librería de utilerías para realizar un acceso y consulta más cómodo sobre DOM.

5.5.3.- Agenda

Como se mencionó en el apartado de importación de cursos este proceso se ha llevado a cabo mediante el uso de una agenda. Ya que quizás nos sería útil la agenda en otro parte del sistema consideramos necesario implementar dicha agenda como un componente reutilizable.

5.5.4.- UUID / URN

Las especificaciones propuestas por el consorcio IMS se basan en mayor o menor medida en la existencia de identificadores únicos para poder identificar de manera unívoca los recursos educativos. IMS propone que se usen como identificadores el estándar *Universal Resource Name* (URN), sin embargo no deja de ser una indicación ya que su implantación y uso es complejo.

En el sistema <e-Aula> tenemos soporte para identificadores basados en URN, sin embargo para la generación de identificadores únicos hemos decidido la utilización de *Universally Unique Identifier* (UUID) ya que nos aseguramos que el identificar es único (al menos en nuestra máquina), usando una librería que nos genera dichos identificadores.

Sección VI: Detalles de diseño e implementación del LMS

6.1.- Cambios en el framework Apache Struts

Como se ha mencionado la información almacenada en una parte pública y una parte privada. En este apartado describiremos el problema que se plantea debido a este motivo y los cambios necesarios sobre Apache Struts par remediarlos.

Como hemos indicado, en la parte privada de los datos, se encuentra el fichero principal de cada recurso del curso, digamos una página HTML. Además en la parte pública tenemos los archivos auxiliares del recurso, sean estos por ejemplo, archivos de imágenes.

También se ha visto que un paquete está estructurado de manera jerárquica y es auto contenido, es decir, si el recurso es interno al paquete, todos los archivos que lo definen tienen que estar en dicho paquete.

En el caso de una página HTML todas las etiquetas que hagan referencia a recursos, como la etiqueta `<IMG>`, la ruta a la imagen referenciada puede ser una URL absoluta (en el caso de no estar en el paquete) o una URL relativa.

En este punto es necesario mencionar el funcionamiento de un navegador web. Cuando el navegador procesa la página y encuentra una etiqueta que referencia un recurso externo si la ruta es relativa el navegador realizará una resolución de la URL con respecto a la URL de la página que está intentado mostrar.

Teniendo en cuenta este último apunte, si queremos que el servidor Apache sirva el contenido auxiliar, simplemente tenemos que hacer creer al navegador que está en el sitio adecuado, una vez servido el recurso principal, el resto de peticiones que se realicen y en la que intervenga una URL relativa, se realizarán de la manera correcta.

Veamos un ejemplo para aclarar conceptos. Supongamos que existe un curso cuyo nombre es "**CURSO1"** y se realiza una petición de un ítem cuyo identificador es "**ítem1**" y que tiene como recurso principal a "**pagina1.html**". Para simplificar el ejemplo, supongamos que la petición se realiza desde un frame para que la página se cargue en otro frame. Los pasos desde que se realiza la petición hasta se devuelve el archivo principal del recurso son los siguientes:

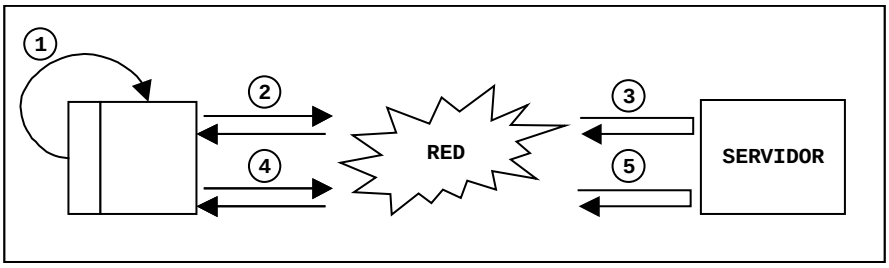


Fig. 9: Ejemplo de adaptación

1. El usuario pulsa sobre un Item en el sistema de navegación, se construye una URL para poder realizar la petición del ítem "**ítem1**". Esta URL se usa para recargar el frame donde se mostrará el contenido, siendo la correspondiente URL: http://www.servidor.com/aulav/muestraItem.do?idItem=ítem1
2. El navegador realiza la petición HTTP pertinente al servidor.
3. El servidor recibe la petición del "**ítem1**", este construye una ruta para el archivo principal del recurso asociado al ítem. El servidor responde con un **sendRedirect** indicando al navegador que debe realizar una nueva petición. La nueva URL sería:

   `http://www.servidor.com/aulav/repo/CURSO1/Mostrarpagina.do?idItem=ítem1`

4. El navegador realiza la nueva petición usando la URL que nos ha proporcionado el servidor.
5. El servidor devuelve la página requerida al navegador.
6. Cuando el navegador procesa el HTML de la página encuentra el tag `<IMG src="figura1.gif">` ya que se trata de una ruta relativa, el navegado resuelve esta ruta relativa, obteniendo:

   `http://www.servidor.com/aulav/repo/CURSO1/figura1.gif.`
7. Como esta URL corresponde a contenido que puede servir Apache HTTP Server directamente, se devuelve directamente el contenido de la imagen.

Como se ha indicado en la descripción de Struts, existe un archivo denominado `struts-config.xml` que es el lugar donde se indica a que URL responde un Action. Sin embargo, la sintaxis no nos permitía realizar la reescritura de URL como se ha indicado en el ejemplo. Para suplir esa falta, hemos extendido la clase `org.apache.struts.RequestProcessor`, implementándose la funcionalidad requerida mediante la clase:

`es.ucm.sip.eaula.runtimeSystem.controller.RequestProcessor`

(ver figura UML-8 en apéndice A.2).

6.2.- Implementación de servicios comunes

6.2.1.- Utilerías de XML para DOM

Como se indicó en la descripción del servicio de XML, DOM es una API para manejar documentos XML que mantiene una vista jerárquica en forma de árbol para el documento XML: Esta API ha sido propuesta por el World Wide Web Consortium siendo independiente del lenguaje sobre el que se implementa.

Debido a esa independencia del lenguaje, en ocasiones resulta incómodo el manejo de documentos XML mediante DOM, sin embargo seguimos necesitando usar DOM debido a que queremos tener una vista en forma de árbol del documento. Para llenar este hueco, hemos implementado un conjunto de utilerías en la clase `es.ucm.sip.eaula.util.xml.XMLUtils.`

6.2.2.- Logging

Este servicio no se ha descrito anteriormente ya que no proporciona funcionalidad explícita al sistema. El fin de este servicio es eliminar el uso de las llamadas `System.out.println(...)` en el código del sistema, sin embargo necesitamos una manera de "loggear" tanto mensajes de depuración y pruebas como mensajes de alertas de errores detectados para que posteriormente podamos analizar dichos mensajes para corregir los errores.

Con este objetivo apareció en el J2SE 1.4.X esta funcionalidad (para más información véase la descripción del paquete `java.util.logging`). Sin embargo existen otras API que proporcionan este servicio y que estaban orientadas a aplicaciones de alto rendimiento y en las que se proporciona soporte para multi-hilo. Debido a este hecho hemos considerado que en nuestro servicio de logging debía ser lo suficientemente flexible para que se pudieran configurar diversos sistemas de logging.

Por ello utilizamos la librería *Apache Commons Logging* que nos proporciona una capa de abstracción muy ligera sobre los distintos sistemas de logging, de manera que se puede elegir que sistema real se está usando debajo de la capa. Además esta librería permite que se use una "Abstract Factory" propia. En nuestro caso usamos esta funcionalidad implementando esta factoría en

```
es.ucm.sip.eaula.util.logging.LogFactory
```

de manera que conseguimos configurar las propiedades del sistema de logging usando el servicio de configuración del sistema <e-Aula>.

6.2.3.- Configuración y BackStore

La API de Java Servlet proporciona mínima funcionalidad para llevar a cabo la configuración de dichos Servlets mediante la inclusión de información que se necesita en el archivo "`web.xml`" que define la aplicación. Sin embargo este sistema es inapropiado cuando se pueden configurar numerosos parámetros además de que en nuestro caso, al usar Apache Struts, tenemos un solo Servlet.

A partir de J2SE 1.4.X existe una API implementada en el paquete `java.util.prefs` que proporciona un sistema de configuración para aplicaciones J2SE. Esta API proporciona un servicio más completo que el proporcionados por `java.util.Properties`, ya que permite tener información estructurada en forma de árbol, además de permitir configuración personalizada para cada clase.

Como hemos mencionado esta API está pensada para aplicaciones J2SE y no aplicaciones J2EE, para éstas se recomienda el uso de JNDI o quizás el uso de JMX, sin embargo estos servicios son complejos de usar o no es posible su integración dentro de la aplicación de manera sencilla. Hemos considerado por tanto el uso de la API `java.util.prefs` para configurar la aplicación. Esta API se basa en la existencia de un almacén de los datos, "BackStore". Este almacén de datos es proporcionado por el Java Runtime Environment y en el caso de sistemas operativos Windows usa el registro para almacenar los datos. Por ello, hemos usado una librería implementada en el paquete `ucar.util.prefs` que nos proporcionaba este "Backstore". Si desea más información puede consultar la página del autor:

```
http://www.unidata.ucar.edu/staff/caron/prefs/index.html
```

Para poder usar este almacén de datos y también poder seguir teniendo acceso a los datos de configuración para los Java Servlets, hemos definido la clase

```
es.ucm.sip.eaula.runtimeSystem.SystemConfig
```

que nos proporciona acceso a estos servicios de configuración (ver figura UML-9 en apéndice A.2).

6.2.4.- UUID y URN

El soporte de UUID está proporcionado por la paquete `com.stafney.UID` que es capaz de generar UUID (usando JNI para acceder a la tarjeta de red) o Locally Unique Identifier (LUID), si desea más información consulte la página del autor:

```
http://www.stafney.com/~tstafney/opensource/
```

Para manejar URN en el sistema usamos una librería externa implementada en el paquete `org.ietf.uri` en la que se implementan distintos servicios de resolución para URL / URN / URI.. Si desea más información consulte la página del autor:

```
http://www.vlc.com.au/urilib/
```

Ya que existen diversos tipos de identificadores y ya que no existe un consenso claro de cual usar, hemos optado por incluir una abstracción dentro de la API de acceso al

manifiesto para los identificadores, esta abstracción está implementada en la clase `es.ucm.sip.eaula.cp.Identifier` (ver figura UML-14 en apéndice A.2).

6.3.5.- Agenda

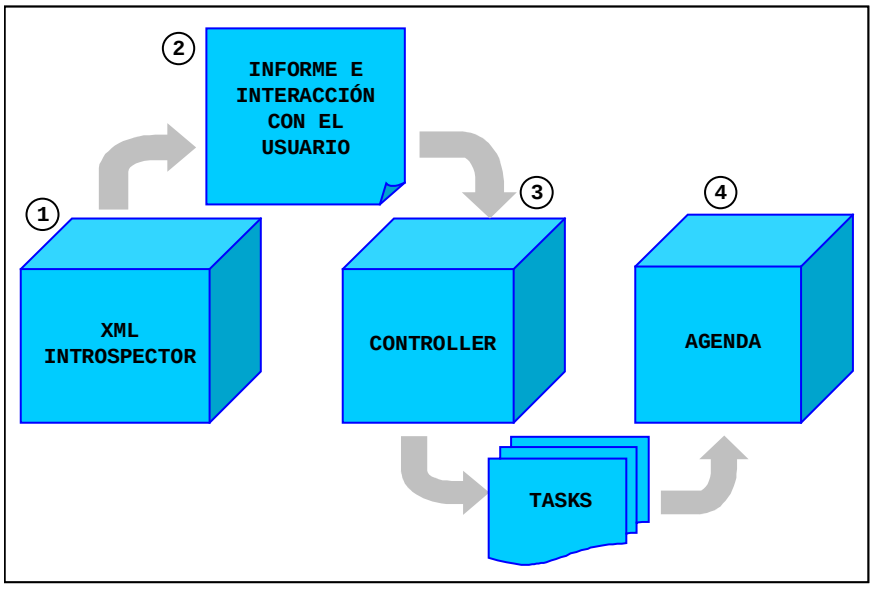
Este servicios está implementado en el paquete `es.ucm.sip.eaula.util.agenda` (ver figura UML-6 en apéndice A.2). Se ha definido una interfaz genérica para las tareas a ejecutar, los mensajes que se obtienen de los resultados de la ejecución de tareas y la agenda en sí misma.

Las tareas extienden la interfaz `java.lang.Comparable` ya que consideramos necesarias que, aunque actualmente no está contemplado, la necesidad de que algunas tareas deben ejecutarse antes que otras.

La implementación específica del sistema de importación del sistema <e-Aula> se implementa en el paquete `es.ucm.sip.eaula.runtimeSystem.validator`.

6.3.- Sistema de Importación

Como se ha indicado anteriormente, el proceso de importación es un proceso previo para que el sistema pueda impartir un curso que no ha sido desarrollado con nuestro sistema. A continuación daremos una explicación del funcionamiento del mismo y como se encuentra implementado.



10. Sistema de importación

El funcionamiento de la importación es el siguiente:

1. En esta etapa inicial se inspecciona el manifiesto del PIF que intentamos importar, la información que obtenemos en esta etapa es información referente a la estructura del manifiesto como archivo XML, como por ejemplo si declara una DTD, los espacios de nombres declarados dentro del documento, etc. Para llevar a cabo esta tarea usamos SAX y en particular un manejador de eventos SAX que se encuentra implementado la clase

`es.ucm.sip.eaula.util.xml.XMLIntrospector` (ver figura UML-7 en apéndice A.2).

3. Tras esta primera etapa actualmente se le presenta la información obtenida del XML al usuario. Esta etapa está pensada para que el usuario pudiera aportar información al sistema de que hacer con las especificaciones encontradas en el manifiesto, ya que los espacios de nombre declarados ayudan a identificar las especificaciones, y si son soportadas o no por el sistema.
4. Tras informar al usuario y obtener información del mismo si procede, en esta fase se incluyen en la agenda las tareas necesarias para llevar a cabo la importación del curso.
5. Se ejecutan una a una las tareas, parando la importación si una de las tareas falla. La importación termina cuando no se encuentran más tareas a ejecutar.

Actualmente las tareas que son añadidas para llevar a cabo la importación son:

- Adaptación del manifiesto para que en vez de usar DTD use XSD para realizar la validación.
- Incluye la declaración de las XSD para los espacios de nombres que hagan falta.
- Validación el manifiesto como archivo XML usando XSDs.
- Comprueba si todos los archivos principales y auxiliares de los `<resource>`declarados en el manifiesto se encuentran en el paquete.
- Realiza la copia de archivos principales y auxiliares de los `<resource>` a la carpeta del curso.

Si las tareas terminan, el curso importado se da de alta en el sistema para que los alumnos puedan matricularse en él o para que el administrador asocie a un tutor a dicho curso.

6.4.- Autenticación y Autorización

La autenticación y la autorización son dos características que hemos tenido que tener muy presentes en la implementación, ya que como es obvio en este tipo de sistemas tenemos que saber si un usuario que ha iniciado sesión en el sistema tiene privilegios suficientes para realizar una acción.

El fin de la tarea de autenticación consiste en identificar al usuario que está interactuando con el sistema. La autorización tiene como propósito el poder controlar que acciones puede realizar que usuario.

El servidor Apache Tomcat nos proporciona estas dos características configurando apropiadamente el fichero `web.xml` que define la aplicación ya que en la especificación JavaServlet 2.3 está contemplada la seguridad. Sin embargo los medios proporcionados para controlar autorización son insuficientes para nuestro sistema. En el caso de la autenticación los métodos son muy rígidos y poco adaptables a las necesidades del sitio web.

Debido a estas razones, hemos decidido integrar en el sistema nuestros propios servicios de autenticación y autorización.

Para simplificar la tarea y abstraer al resto del sistema de la existencia de este control, hemos implementado estos dos servicios mediante "filtros". Ya se ha mencionado que

los filtros se encuentran entre el cliente y la aplicación web correspondiente y son usados para examinar y modificar las peticiones y respuesta y el flujo entre ellas.

6.4.1.- Autenticación

En el caso de la autenticación el funcionamiento es similar e incluso, con el mismo código, que el proporcionado por Apache Tomcat. El servicio de autenticación esta implementado por la clase

`es.ucm.sip.eaula.runtimeSystem.SignOnFilter`.

Para mayor información consulte la figura UML-18 en el apéndice A.2 que representa la máquina de estados de `SignOnFilter`.

La mejora sustancial respecto a la seguridad proporcionada por el contenedor es la posibilidad de poder indicar reglas del estilo allow / deny permitiendo especificar el orden de aplicación de las mismas. A continuación presentamos un fragmento del archivo de configuración en el que se especifica una parte de la información de autenticación:

```
<node name="SignOnFilter">
  <map>
      <entry key="order" value="allow,deny" />
  </map>
  <node name="rules">
      <map />
      <node name="allow">
        <map>
           <entry key="js" value="*.js" />
           <entry key="css" value="/style/*" />
           <entry key="img" value="/img/*" />
           <entry key="index" value="*.xhtml" />
           <entry key="passwordRetrieval" value="/forgotPass.JSP" />
           <entry key="userInUse" value="/UserInUse.do" />
           <entry key="AuthenticationFailed" value="/AuthFailed.do" />
           <entry key="matriculacion1" value="/registerUser.JSP" />
           <entry key="matriculacion2" value="/RegisterUser.do" />
           <entry key="nuSuchUser" value="/NoSuchUser.do" />
           <entry key="recordar" value="/SendPassword.do" />
           <entry key="raiz" value="/" />
        </map>
      </node>
      <node name="deny">
        <map>
           <entry key="Actions" value="*.do" />
           entry key="default" value="/*" />
        </map>
      </node>
    </node>
</node>
```

6.4.2.- Autorización

Ya que en el sistema tenemos distintos roles, el sistema de autorización soporta asignar reglas de autorización para cada rol. El servicio de autorización está implementado por `es.ucm.sip.eaula.runtimeSytem.SecurityFilter`.

Al igual que en el caso anterior, es posible asignar reglas allow / deny y especificar el orden de aplicación de las reglas. A continuación se presenta un fragmento del archivo de configuración:

```
<node name="SecurityFilter">
  <map>
  <entry key="order" value="allow,deny" />
  </map>
  <!--
     Roles consultados en la clase es.ucm.sip.eaula.runtimeSystem.Rol
   -->
```

```
<node name="LEARNER">
<map />
<node name="allow">
  <map>
    <entry key="js" value="*.js" />
    <entry key="css" value="/style/*" />
    <entry key="img" value="/img/*" />
    <entry key="index" value="/index.xhtml" />
    <entry key="login" value="/login.xhtml" />
    <entry key="userInUse" value="/UserInUse.do" />
    <entry key="AuthenticationFailed" value="/AuthFailed.do" />
    <entry key="matriculacion1" value="/registerUser.JSP" />
    <entry key="matriculacion2" value="/RegisterUser.do" />
    <entry key="privileges" value="/LacksPrivileges.do" />
    <entry key="recordar" value="/recordarPassword.do" />
    <entry key="recordar" value="/Logout.do" />
    <entry key="raiz" value="/" />
  </map>
</node>
<node name="deny">
  <map />
</node>
</node>
<node name="TUTOR">
<map />
  <node name="allow">
  <map>
    <entry key="js" value="*.js" />
    <entry key="css" value="/style/*" />
    <entry key="img" value="/img/*" />
    <entry key="index" value="/index.xhtml" />
    <entry key="login" value="/login.xhtml" />
    <entry key="userInUse" value="/UserInUse.do" />
    <entry key="AuthenticationFailed" value="/AuthFailed.do" />
    <entry key="matriculacion1" value="/registerUser.JSP" />
    <entry key="matriculacion2" value="/RegisterUser.do" />
    <entry key="privileges" value="/LacksPrivileges.do" />
    <entry key="recordar" value="/recordarPassword.do" />
    <entry key="raiz" value="/" />
    <entry key="recordar" value="/Logout.do" />
    <entry key="recordar" value="/tutor/*" />
  </map>
  </node>
  <node name="deny">
  <map>
    <entry key="Actions" value="*.do" />
    <entry key="default" value="/*" />
  </map>
  </node>
</node>
```

6.5.- Delivery System

6.5.1.- Implementación de los ResourceHandler

La idea que existe detrás del `ResourceHandler` es que se trata de un componente del sistema que se encarga de generar la página WEB con la que visualiza el alumno el `<resource>` asociado al `<ítem>` que ha seleccionado el alumno.

Cada `<resource>` tiene un tipo asociado, de manera que el objetivo es que exista un `ResourceHandler` por cada tipo de `<resource>` que soportemos en el sistema.

6.5.2.- Implementación de los ResourceProcessor

Como se indicó en el apartado de descripción de servicios comunes, el `ResourceProcessor` es el encargado de gestionar la presentación del contenido asociado a los recursos.

Dentro de los tags permitidos dentro de un manifiesto se encuentran los tags `<schema>` y `<schemaversion>`. Estas dos marcas son usadas para identificar que especificación de IMS Content Packaging es la que se está usando, por ejemplo:

```
<schema>IMS Content</schem>
<schemaversion>1.1</schemaversion>
```

ó

```
<schema>ADL SCORM</schema>
<schemaversion>CAM 1.3</schemaversion>
```

Como observamos puede contener diversos valores, algunos para las especificaciones propuestas por el consorcio IMS y otras extensiones como ADL SCORM 2004.

Debido a este hecho y ya que queremos que el sistema sea flexible con respecto a este hecho, la implementación del `ResourceProcessor` es obtenida particularizada para un `<schema>` y `<schemaversion>`. De esta manera podemos dar soporte a distintos tipos de empaquetamiento.

El `RessourceProcessor` está implementado por la clase

```
es.ucm.sip.eaula.runtimeSystem.ResourceProcessor
```

(ver figura UML-1 en apéndice A.2) de manera que podemos obtener a través del método factoría `newInstance()` el `ResourceProcessor` específico.

Existe una implementación por defecto para el caso de que el `<schema>` y `<schemaversion>` no coincida con ninguna implementación más específica, esta implementación por defecto está implementada por

```
es.ucm.sip.eaula.runtimeSystem.DefaultResourceProcessor
```

6.5.3.- Tipos de contenido soportados

Actualmente en el sistema <e-Aula> se encuentran soportados los siguientes tipos:

1. **webcontent**: Este es el tipo de un `<resource>` IMS para las páginas HTML.
2. **eaula**: Este es el tipo para las páginas <e-Aula>
3. **portada**: Este es el tipo define las páginas <e-Aula> que sirven como portada de un curso.
4. **faq**: Este tipo de recurso definen los recursos que representan páginas FAQ
5. **qti**: Este tipo de recurso define un examen QTI.

6.5.4.- Integración del ResourceProcessor y el ResourceHandler con Apache Struts

Como se ha presentado en los dos apartados anteriores el `ResourceProcessor` se encarga de devolver un `ResourceHandler` para un tipo de recurso particular. Este comportamiento es similar a como funciona un `ActionMapping` de Struts, de hecho la idea del `ResourceProcessor` es la misma, sirve para asociar un tipo de recurso con su manejador.

En el caso de `ResourceHandler`, sirve para representar a la parte de la vista que se encarga de gestionar un tipo de recurso particular. Este comportamiento es el mismo que tiene un `ActionForward` de Struts al que se le añade la información necesaria para obtener el recurso.

En resumen el procesamiento de un `<ítem>` seleccionado por un alumno es el siguiente:

1. Se obtiene el recurso asociado al `<ítem>`.
2. Se pide al `ResourceProcessor` que nos de un `ResourceHandler` para el tipo del recurso a mostrar.
3. Se devuelve el `ResourceHandler` como `ActionForward` de salida.

6.5.6.- Extensibilidad y Modularidad

La asociación entre los tipos `ResourceProcessor` y los `ResourceHandler` se realiza a través del sistema de configuración de la aplicación.

Además como ha podido observarse en los puntos anteriores, este sistema de procesamiento de los recursos es modular ya que se puede ir dando soporte poco a poco a cada tipo de recurso que nos interese, basta con implementar el `ResourceHandler` apropiado y definirlo en el `RequestProcessor` específico y el sistema ya será capaz de dar soporte al nuevo tipo de recurso.

6.6.- Sistema de Autoría

La responsabilidad del sistema de autoría es permitir al usuario (en este caso al profesor) acceder a los contenidos de los cursos para modificarlos, añadiendo eliminando o editando los recursos existentes. El criterio general de diseño que se ha seguido para la implementación de este subsistema a sido el siguiente:

- Asignar un Action para cada una de las partes de que se compone un curso. Ese Action dispone cuatro métodos que son:
    - `Add`: Para añadir un nuevo elemento de ese tipo al curso.
    - `Edit`: Para acceder a un elemento ya existente en el curso y prepararlo para la edición.
    - `Delete`: Para eliminar un elemento del curso.
    - `Commit`: Para actualizar los cambios sobre ese elemento.
- Asignar un `ActionForm` a cada uno de los elementos editables del curso. Este `ActionForm` es el encargado de "llevar" los datos desde el modelo hasta la vista y viceversa. Este `ActionForm` contiene métodos para validar los datos que han sido introducidos por el usuario y para modificar el modelo si se recibe la orden commit del usuario.
- Asignar un JSP a cada uno de los elementos editables del sistema. Este JSP es capaz de mostrar un `ActionForm` y de editarlo.
- Si un elemento contiene a su vez subelementos editables (como una organización contiene ítems o un manifest resources) desde el JSP de edición del elemento padre se podrán añadir, borrar o editar (accediendo a sus propios JSP's de edición) cada uno de los elementos hijos.

En la figura UML-20 del apéndice A.2 se puede ver un ejemplo de esta arquitectura en la parte de edición del los ítems del manifiesto.

Aunque la arquitectura del sistema de autoría es bastante homogénea, debido a varios problemas que han surgido han aparecido varias clases que rompen con esta estructura uniforme:

6.6.1.- Control de concurrencia: CourseState

Debido a que es posible que varios profesores estén editando el mismo curso a la vez es necesario mantener el control de la concurrencia en el acceso a las diferentes partes de que consta un curso. Una posible política es restringir el acceso a sólo un profesor cada vez, pero eso empeoraría el rendimiento total del sistema, a la vez que impediría el desarrollo normal de la cooperación entre profesores.

Como solución de compromiso se ha dividido cada curso en 5 partes principales, las cuales pueden estar editándose a la vez, aunque cada una de ellas sólo puede ser editada por un profesor simultáneamente.

Las 5 partes son:

- Portada.
- Glosario.
- FAQ.
- `CourseInfo`: Contiene el título y la descripción del curso.
- Contenido: El Manifiesto y todo lo que ello conlleva.

Para controlar el acceso a estas partes hemos creado la clase `CourseState`, que contiene la información de estado del curso, así como los métodos para acceder a esta información y para modificar el estado del curso. La información que se encuentra almacenada en esta clase es:

- Información sobre si el curso el público o provisional.
- Lista de lectores de cada sección.
- Lista de escritores de cada sección.
- Estado de cada sección.

Esta clase implementa un cierre lectores/escritores sobre cada una de las partes del curso, de tal manera que todo `Action` que quiera acceder a modificar alguna de estas partes del curso deberá "pedir permiso" primero al objeto `CourseState` asociado al curso que está modificando. Una vez terminada la edición deberá "devolver el permiso" para así permitir a otro usuario editar esa sección. Debido a que no es recomendable hacer llamadas bloqueantes dentro de una aplicación web, aquellas peticiones de permiso que se realicen sobre secciones que ya estén boqueadas por otro usuario serán rechazadas, en vez de puestas a la espera.

Para permitir la comunicación entre diferentes usuarios que puedan estar esperando a que ocurran ciertos eventos en el curso (paso de público a provisional y viceversa) hemos implementado un sistema de eventos con una lista de *listeners* que pueden registrarse en el objeto `CourseState`. (ver figura UML-12 en apéndice A.2)

6.6.2.-Control de cambios: UndoableAction

El control de flujo en una aplicación web es bastante complejo, debido a que el usuario puede "saltar" inopinadamente hacia secciones del caso de uso en el que se mueve con tan sólo introducir una URL a mano en el navegador, o pulsando el botón de "atrás" que incluyen la mayoría de navegadores.

Si unimos esta dificultad en el control de flujo a la necesidad de guardar, de manera fiable, el estado intermedio de una sección de un curso que esta siendo editada en un momento dado por el usuario, hasta que decida guardarla, tenemos un problema de una cierta complejidad, pues es posible que un usuario "abra" varias veces una sección que no ha terminado de editar, o que salte a la edición de una sección sin terminar de editar otra.

Para resolver este problema hemos implementado la siguiente política de control de cambios:

- Si el usuario accede a una sección que a la que no ha entrado todavía, se "abre" la sección y se guardan los datos relativos en la sesión de usuario.

- Si el usuario accede a una sección a la que ya había accedido anteriormente, pero que no había cerrado se le ofrecen dos opciones:
    - Puede volver a editar los datos originales (cerrar y abrir de nuevo)
    - Puede continuar editando los datos en el punto donde lo dejó.

Como esta política es común a todas las secciones del sistema, la hemos encapsulado dentro de un `Action` al que hemos llamado `UndoableAction`. La idea es que los `Action` que acceden por primera vez a cada una de las secciones principales del curso deben extender a `UndoableAction` sobrescribiendo los métodos que sirven para averiguar si ya se había abierto la sección actual y cómo se reinician los datos cuando se quiere volver a los datos originales.

El diagrama UML que describe esta parte de la arquitectura se encuentra en la figura UML-19 del apéndice A.2.

6.6.3.- Control de la coherencia: Security Manager

Cuando un usuario está editando los datos de un curso es necesario que los datos del curso mantengan unas ciertas restricciones para que el curso siga siendo válido dentro del sistema. Estas restricciones son de 3 tipos:

- Restricciones impuestas por el estándar IMS.
- Restricciones impuestas por el *Application Profile* <e-Aula>: Estas restricciones están relacionadas principalmente con la gestión de los ítems especiales de <e-Aula> portada, FAQ y glosario.
- Restricciones debidas a la política de gestión de cursos públicos y provisionales. Estas políticas se resumen en estas dos reglas:
    - Si el curso el provisional se permite editar todo.
    - Si el curso ya está publicado se permite añadir ítems y modificar el contenido de los ítems ya existentes, pero no se pueden eliminar ítems ni modificar la estructura de lo que ya hubiera antes.

Del cumplimiento del primer tipo de restricciones, es decir, las relacionadas con el estándar, se encarga la librería de manejo del Content Packaging. Esta librería es parte del núcleo del sistema, así como el estándar IMS forma parte del núcleo del LMS.

El cumplimiento de los otros dos tipos de restricciones es tarea del `SecurityManager`. Esta interfaz contiene métodos para preguntar si es posible editar una cierta parte del curso (esta vez con una granularidad mucho más fina que en el `CourseState`) según la política actual del sistema, y para actualizar el estado actual de esas partes en caso de modificación.

Actualmente existen dos implementaciones para esta interfaz, una para la política en cursos provisionales y otra para la política en cursos publicados. Además para permitir el acceso al `SecurityManager` desde una XSL (para permitir dar un aspecto diferente al árbol de edición en función de la posibilidad de editar o no un cierto ítem) hemos implementado un *wrapper*.

El diagrama UML que describe esta parte del sistema se encuentra en la figura 11 del apéndice A.2.

6.7.- Librería de soporte del Content Packaging

La información proporcionada por la especificación IMS Content Packaging, proporcionada a través del manifiesto es usada en varios puntos de la aplicación. Además existen otras especificaciones de IMS que usan como contenedor al IMS Content Packging. Por tanto es necesario dar un soporte adecuado a esta especificación.

En nuestro caso hemos decidido desarrollar una API completa de acceso y edición de la información contenido en el manifiesto de manera que hemos concentrado la lógica y el conocimiento de la especificación. (ver figura UML-13 en apéndice A.2).

Esta API viene a suplir el hueco existente entre el manejo del manifiesto como un mero documento XML y el concepto abstracto de manifiesto propuesto en la especificación del Content Packaging.

Como sabemos, necesitamos tener una visión jerárquica, en forma de árbol, de las organizaciones del manifiesto de manera que es obvio pensar en DOM como tecnología a usar, sin embargo como implementación específica de esta API pueden usarse otras tecnologías como SAX o JAXB.

Otro motivo por el que hemos decidido desarrollar esta API ha sido para poder dar consistencia a la especificación IMS Content Packging, durante la etapa de análisis hemos usado la versión 1.1.3 de dicha especificación y hemos podido observar inconsistencias en la misma e información contradictoria, de manera que cuando se especifique de manera más clara los puntos en cuestión la modificación del sistema será más simple en comparación a tener la lógica distribuida por toda la aplicación.

En nuestro caso hemos hecho el uso de DOM como tecnología subyacente, de manera que hemos dotado a DOM de la semántica necesaria para dar soporte al IMS Content Packaging. Entre las características implementadas en esta librería se encuentran:

La API proporcionada por está librería se encuentra definida en el paquete `es.ucm.sip.eaula.cp`, si se desea una descripción detallada véase la figura UML-14 en el apéndice A.2.

6.7.1.- Control de cambios de identificadores

Ya que algunas etiquetas pueden hacer referencia a otras etiquetas a través del campo "`identifier`" proporcionamos la capacidad de que estas etiquetas que realizan la referencia, se actualicen automáticamente.

6.7.2.- Comprobación de validez de referencias

Las referencias que pueden existir dentro de un manifiesto siguen unas reglas a cerca de las etiquetas que pueden referenciarse y unas reglas de visibilidad de las mismas.

La etiqueta que realizan las referencias son la etiqueta `<ítem>` y la etiqueta `<resource>`. `El caso de la etiqueta <ítem>` es uno de los casos inconsistentes existentes en la especificación ya que en uno de los documentos se indica que puede referenciar a cualquier etiqueta que tenga identificador y que verifique las reglas de visibilidad, sin embargo en otro de los documentos de la especificación esto se restringe.

Las reglas de visibilidad son similares a los contexto de visibilidad de variables en los lenguajes de programación. Sin embargo en este caso una etiqueta puede hacer referencia a otra etiqueta que se encuentre en el mismo contexto o en un contexto más interno. La implementación se encarga de verificar que cuando se hace referencia a otra etiqueta, verifique estas reglas.

6.7.3.- Resolución de rutas

Las etiquetas `<resource>` y `<file>` contienen atributos que representan URL a recursos. Estas rutas pueden ser absolutas, de manera que hacen referencia a recursos externos al paquete, o rutas relativas, que hacen referencia a recursos internos del paquete.

La especificación permite incluir, mediante los atributos `xml:base`, URL que son usadas para resolverla URL especificada en las etiquetas mencionadas, de manera que una ruta relativa puede finalmente convertirse en una ruta absoluta debido a esta resolución. Mediante nuestra API abstraemos al cliente de la misma este proceso de resolución de rutas.

En el caso particular de recursos externos, si existe cierto atributo de la etiqueta `<ítem>` debe aplicarse un algoritmo para generar la URL que se usará para acceder a dicho recurso (Content Packaging Information Model 1.1.3 sección 4.2).

6.7.4.- Independencia del almacenamiento del paquete

En el desarrollo de la librería hemos intentado abstraer el medio físico donde se encontraba almacenado el manifiesto así como los recursos físicos definidos en el mismo. La definición de esta abstracción se encuentra en la interfaz `es.ucm.sip.eaula.cp.DataSource` que usada conjuntamente con las URL proporcionadas por el resto de la librería nos permite acceder a los recursos físicos incluidos dentro del paquete.

6.7.5.- Extensibilidad

La implementación específica de esta API se selecciona a través de un "método factoría" que se encuentra parametrizado con los valores de las etiquetas `<schema>` y `<schemaversion>` de esta manera se pueden proporcionar distintas implementaciones que estén adaptadas a una versión específica de IMS Content Packaging o que proporciona funcionalidad propietaria definida mediante una extensión del Content Packaging, como sería el caso de ADL SCORM.

Sección VII: Resultados y Conclusiones

7.1.- Estado final del sistema

Tras la finalización del proceso de trabajo correspondiente al presente curso señalaremos el estado de desarrollo en que se encuentra el LMS.

7.1.1.- Funcionalidad Implementada

Subsistema del alumno

La funcionalidad básica del alumno está implementada y probada. Esto incluye el correcto funcionamiento del sistema de matriculación de nuevos alumnos, el acceso al sistema y la posibilidad de matricularse en los cursos disponibles.

Tras la matriculación en un curso, los alumnos pueden acceder al mismo sin problemas. Esto implica la correcta implementación de los procesos de registro y administración del sistema

La funcionalidad de navegación por los contenidos de un curso está completamente implementada, lo cual indica un completo dominio de la problemática de dar una implementación válida a los estándares de organización de contenidos de IMS. El alumno puede seleccionar a voluntad la organización deseada. Una vez escogida, al alumno se le presenta un árbol de navegación (con funcionalidad para contraer/expandir los distintos nodos del mismo) con todos los ítems correspondientes a dicha organización. Se puede navegar por el curso mediante selección directa de los ítems del árbol (al seleccionar un ítem se procesa y muestra su contenido) o siguiendo un recorrido por defecto empleando una flechas ubicadas en la parte superior de la pantalla que le permiten pasar al ítem siguiente o anterior (considerando una ordenación del árbol en preorden).

Por otro lado, todos los posibles tipos de contenido pueden ser visualizados gracias a haber superado con éxito la problemática de transformar contenido en XML en código HTML visualizable mediante un navegador web. Los contenidos especificados por el estándar se visualizan sin necesidad de tratamiento previo, pero se han implementado los mecanismos necesarios para manejar correctamente los siguientes tipos:

- Los tipos secundarios de <e-Aula> (Portada, Glosario y FAQ). Estos recursos tienen su contenido almacenado en un fichero XML que debe ser transformado y adaptado por medio de las correspondientes transformaciones XSL.
- El tipo Página definido por <e-Aula>. Este tipo es el formato XML empleado para almacenar los contenidos puramente didácticos en el sistema. La transformación y adaptación de los contenidos es compleja puesto que incluye, entre otras cosas, la adaptación del nivel educativo según la elección del alumno.
- Exámenes QTI: Los tests se almacenan en formato XML y deben ser transformados antes de ser mostrados al alumno. Tras la ejecución del test, las respuestas deben ser procesadas y se asigna al alumno una calificación que se almacena en la base de datos que puede ser consultada por el tutor.

Subsistema del tutor

La funcionalidad básica del tutor también ha sido implementada. Esto incluye las capacidades de edición y gestión de los cursos de los que el tutor es responsable.

En el aspecto de gestión, se le ofrece al tutor la posibilidad de consultar los distintos datos estadísticos almacenados acerca la de interacción de los alumnos con el curso (recursos visitados, tiempo empleado dentro del sistema, calificaciones en los tests, etc).

En el aspecto de la edición de cursos, todo el sistema de edición de organizaciones, ítems y recursos ha sido completamente implementado. Al tutor se le dan mecanismos para añadir ficheros a un curso y posteriormente, para crear recursos y enlazarlos con dichos ficheros. Tras esto, se pueden crear las organizaciones dentro de las cuales se ofrece una interfaz para generar el árbol de ítems con los enlaces a los correspondientes recursos.

Dentro también del aspecto de edición de cursos también se ofrece a los tutores la posibilidad de editar los elementos secundarios (Portada, Glosario y FAQ) directamente desde la propia interfaz web. Dicho proceso de edición se realiza siempre desde una interfaz WYSIWYG (*what you see is what you get*), quedando así el tutor aislado de las complicaciones que pudiera suscitarle el formato XML en que se almacenan los contenidos de dichos elementos.

Subsistema del administrador

El subsistema del administrador también ha visto implementadas sus principales funcionalidades.

En el apartado de gestión del sistema ha sido implementada toda la funcionalidad relacionada con la administración de usuarios. Esto incluye dar de baja a tutores y alumnos y dar de alta nuevos tutores. Por otro lado, el apartado de configuración del sistema mediante interfaz web no ha sido completado a falta de una especificación de cuales eran los parámetros que debían ser personalizables.

En relación con la gestión de recursos, se dispone de mecanismos para asignar tutores a los cursos o revocar dichas asignaciones.

Finalmente, se ha implementado un módulo encargado de manejar el proceso de importación de cursos ajenos al LMS de <e-Aula> que sigan la especificación Content Packaging y de integrarlos en el sistema. Esto, unido a las capacidades de navegación y visualización de cursos y al proceso de edición de cursos (organizaciones, ítems, recursos), cierra la implementación completa del estándar Content Packaging de IMS. Esto implica la consecución con éxito de uno de los objetivos fundamentales del proyecto en el presente curso.

Mecanismos de comunicación

Sin estar incluidos explícitamente dentro de ninguno de los subsistemas mencionados, también se ha conseguido una implementación satisfactoria de los foros y los tablones de anuncios. Dichos subsistemas han sido construidos independientemente del sistema <e-Aula> par ser integrados posteriormente como módulos autónomos contenidos dentro del sistema gracias a la arquitectura modular del sistema.

7.1.2.- Funcionalidad planificada

A corto plazo y con pocas modificaciones, se podrían añadir al sistema determinados elementos que han sido dejados de lado por restricciones de tiempo tras haber sido identificados como poco relevantes dados los objetivos del proyecto.

Sistema de Chat

El sistema de Chat no ha sido incluido finalmente en el LMS. Una vez demostrada la flexibilidad de la arquitectura tras la inclusión de los sistemas de foros y tablones, el Chat se podría incluir de una forma similar.

Por tanto, el desarrollo del sistema de Chat se podría realizar independientemente del sistema global y, puesto que carece de interés en la línea de investigación del proyecto se ha optado por postergar su implementación.

Configuración web

Inicialmente se planificó ofrecer al administrador un panel de configuración donde poder determinar ciertos parámetros del sistema de manera remota a través de una interfaz web. Por el momento, la configuración del sistema se guarda en una serie de ficheros XML en lugar de estar codificada en el código fuente.

Por tanto, se podría elaborar una interfaz web para modificar el contenido de dichos ficheros permitiendo así al administrador cambiar cualquier parámetro de forma remota.

Perfeccionamiento de interfaces

Aunque se ha hecho un gran esfuerzo para obtener una arquitectura cómoda y manejable no se ha priorizado la obtención de unas interfaces de acabado profesional. En el futuro, con la colaboración de especialistas en la materia (diseñadores profesionales) se podría intentar buscar un acabado profesional de la interfaz como último paso antes de la comercialización del sistema.

7.2.- El futuro del proyecto <e-Aula>

7.2.1.- Evaluación de la arquitectura obtenida

El objetivo principal para el presenta curso era dar con una arquitectura base sólida sobre la que construir en el futuro distintos sistemas orientados a distintos exámenes con la finalidad de evaluar a estos últimos.

Por ello, hasta cierto punto, los próximos pasos del proyecto dependerían de que el presente trabajo consiga sus objetivos.

Así pues, queremos destacar los motivos por los que se puede considerar que la arquitectura propuesta cumple con los objetivos especificados mediante la ejemplificación de los resultados obtenidos a partir de la misma.

El subsistema QTI

Como ya se ha mencionado, la evaluación de los alumnos en el LMS se realiza a través de tests planteados según el estándar QTI propuesto por IMS.

La implementación de dicho estándar contempla los siguientes aspectos:

- Creación de tests
- Edición de ficheros de test
- Visualización de tests
- Corrección y evaluación de resultados

Todos estos elementos han sido tratados aparte del resto del sistema puesto que se consideró interesante publicarlos como una aplicación aparte (*standalone*). Tras el proceso

de implementación independiente la integración de dichas funcionalidades con el resto del sistema fue sencilla gracias a la modularidad de la aplicación (que permitió deducir con facilidad los mecanismos de unión entre ambos sistemas) y a su mantenibilidad (que permitió solventar los problemas que surgieron durante la integración con facilidad).

Foros y tablones

Los sistemas de foros y tablones también fueron implementados por separado. Dado que estos sistemas eran mucho más sencillos que el subsistema QTI, la integración fue prácticamente inmediata, lo cual verifica de nuevo el alto grado de modularidad de la arquitectura.

Comentarios Generales

La integración de los mecanismos de los foros y del subsistema QTI han sido dos ensayos importantes de lo que podrían ser futuras extensiones al sistema planteado.

Por un lado, la integración de los foros demostró la sencillez extrema de añadir sistemas "externos" al LMS, entendiendo por externos el hecho de que eran sistemas ajenos al proceso educativo en sí. Podemos afirmar que en el futuro, cualquier mecanismo de servicios complementarios al proceso educativo podrá ser integrado con relativa sencillez mejorando así la experiencia de los alumnos en el sistema <e-Aula>

Pero el objetivo principal del proyecto es la evaluación (como ayuda a la concreción) de estándares educativos. El ejemplo del subsistema QTI ha sido un ensayo exitoso de lo que podría ser el proceso de añadir soporte a un nuevo estándar, dando pie a pensar que la arquitectura obtenida tras este trabajo servirá en el futuro como base sobre la que realizar las evaluaciones de futuros estándares educativos.

7.2.2.- Posibles extensiones

Dentro del marco de los objetivos globales del proyecto <e-Aula>, el LMS ha sido elaborado teniendo en cuenta una serie de grande funcionalidades que podrían ser añadidas al proyecto en un futuro por personas ajenas al desarrollo original.

En el futuro, a medida que aparezcan nuevos estándares el sistema podrá ser fácilmente modificado para adaptarse a ellos. A corto plazo, existen una serie de líneas de investigación que podrían servir como futuros trabajos de los miembros del proyecto <e-Aula>, siempre construyendo sobre el trabajo realizado en el presente curso:

Implementación del estándar Simple Sequencing

IMS propone un estándar para la navegación en los cursos denominado Simple Sequencing. Dicho estándar especifica como definir esquemas de prerrequisitos entre los ítems de una organización (reglas como por ejemplo "el ítem 'x' no puede ser visitado si no se ha visitado antes el ítem 'y').

El sistema ha sido diseñado teniendo dicho estándar en mente. Dentro del marco del proyecto <e-Aula> y su objetivo global de evaluación de estándares educativos, la implementación del estándar Simple Sequencing podría ser un importante proyecto ya que es un campo en el que no se ha profundizado mucho aún.

Evaluación de perfiles de alumnos

Siguiendo la idea de que un LMS como el propuesto podría ser capaz de adaptarse automáticamente al alumno que lo utiliza, se podría crear un subsistema que, partiendo

de los datos recabados acerca de un alumno, adaptase el nivel de dificultad del contenido presentado.

En el LMS implementado se dispone de un mecanismo de adaptación del nivel educativo a través de los documentos de tipo Página. Así mismo, se recopila información sobre el rendimiento de cada alumno como por ejemplo los recursos visitados o las calificaciones en los tests. Se podría desarrollar un sistema que partiendo de la información recopilada seleccionase automáticamente el nivel educativo.

Adaptación al tipo de cliente

Por el momento se han establecido los mecanismos que permiten filtrar el contenido en XML mediante transformaciones XSL. Por el momento, sólo se han elaborado transformaciones genéricas que adaptan el contenido a un formato válido en la mayoría de los navegadores actuales.

Se podrían aprovechar dichos mecanismos para reconocer el tipo de navegador empleado por el cliente y modificar automáticamente la selección de la transformación a aplicar.

7.2.3.- Conclusiones

Se ha logrado implementar un sistema de e-learning bastante completo y con una serie de funcionalidades extra que lo hacen atractivo como Campus Virtual (aunque no necesariamente al nivel de los sistemas comerciales de pago).

Pero ese no era el principal objetivo. El objetivo primario del trabajo era conseguir un sistema base estable, robusto, modular y mantenible sobre el que poder basar el trabajo de evaluación de estándares del proyecto <e-Aula> en los próximos años. Las pruebas de extensibilidad y mantenibilidad anteriormente explicadas demuestran que este objetivo se ha visto cumplido, con lo que se puede considerar que el resultado obtenido durante el curso ha sido satisfactorio.

Los esfuerzos dedicados a la consecución del núcleo arquitectónico del sistema hubieran supuesto un retraso inaceptable si el objetivo hubiera sido obtener un LMS funcional cuanto antes, pero a largo plazo el gran porcentaje del esfuerzo dedicado al diseño de la arquitectura redundará en un beneficio a medio plazo del proyecto <e-Aula> que no se hubiera obtenido mediante el desarrollo de un LMS sin proyección de futuro.

7.3.- Trabajo de los alumnos y habilidades adquiridas

Finalizamos esta memoria con una reflexión sobre el trabajo realizado por los alumnos en el ámbito de este proyecto de Sistemas Informáticos así como de la metodología de trabajo y las habilidades que ha sido necesario adquirir y ejercer durante el transcurso del mismo.

7.3.1.- Proceso de desarrollo

El proceso de desarrollo del sistema se ha basado fundamentalmente en una comunicación fluida entre los integrantes del grupo de trabajo. Esto ha permitido aunar esfuerzos en los procesos delicados como podían ser las discusiones sobre el diseño de

las distintas partes de la arquitectura o las decisiones sobre la especificación y objetivos del trabajo a realizar.

Para cada módulo del sistema se comenzaba con una discusión en la que se perfilaba cual sería la funcionalidad esperada. Una vez tomadas las decisiones preliminares sobre la funcionalidad, se discutían con el director del proyecto el cual expresaba su opinión proponiendo modificaciones que volvían a ser discutidas, dando así lugar a un proceso iterativo de refinamiento.

Una vez tomada la decisión sobre la funcionalidad se procedía a discutir la organización de la arquitectura en reuniones entre los desarrolladores donde se estudiaban las ventajas e inconvenientes de cada solución propuesta.

Finalmente los trabajos de desarrollo se repartían sin consideraciones especiales o roles. Pese a ello, dadas las habilidades o experiencias de cada miembro del grupo, si se tendía a la especialización de cada uno en determinadas materias para agilizar el proceso de desarrollo.

7.3.2.- Metodología de trabajo

Dado el tamaño reducido del grupo, se ha optado por una metodología de trabajo con un alto nivel de concurrencia donde la actividad presencial no fuese imprescindible facilitando así un ritmo de trabajo alto dado que no era necesario ponerse de acuerdo de antemano para realizar todo el trabajo juntos (excepto en los procesos de toma de decisiones importantes).

Para lograr este tipo de rendimiento y mantener la coherencia del trabajo realizado se ha empleado un servidor CVS (Sistema de control de versiones) donde se ubican todos los ficheros del sistema. El servidor se encargaba de evitar que dos personas modificasen el mismo fichero por separado mediante un sistema de bloqueo selectivo y actualización posterior.

De este modo, el reparto de trabajo ha sido sencillo pues no había que estar pendiente de evitar modificar el trabajo de otro. Esto es importante debido a que en todo sistema existen una serie de ficheros comunes que se relacionan con partes distintas del sistema, lo cual en condiciones normales dificulta los procesos de trabajo en equipo.

Con este método de trabajo, las responsabilidades se repartían en series de tareas pequeñas a realizar a corto plazo (por ejemplo, 5 o 6 tareas a realizar por cada desarrollador en el transcurso de una semana).

Para realizar cada tarea, el desarrollador bloqueaba los ficheros necesarios para ejecutarla y comenzaba su trabajo sobre los mismos, probando los resultados en un entorno de pruebas propio distinto de la versión "maestra". Una vez que el resultado era satisfactorio el desarrollador actualizaba (o publicaba) los ficheros modificados en el servidor de modo que los cambios realizados quedasen a disposición de todos. Si en un momento dado un desarrollador necesitaba un fichero bloqueado por otro, podía proceder a realizar alguna otra de sus tareas asignadas en espera de la liberación de los ficheros.

Esta metodología unida a una fácil comunicación entre los integrantes del grupo ha dado lugar a un ritmo de trabajo elevado y, además, seguro. El sistema del servidor CVS permitía volver atrás si algún cambio resultaba fatal dado que se guardan todas las versiones intermedias de los ficheros junto con un registro de las modificaciones realizadas en cada versión junto con el autor de dichas modificaciones.

7.3.3.- Puntos de esfuerzo

El desarrollo del trabajo ha demandado de los alumnos el desarrollo y aplicación de determinadas habilidades que, tras demostrar su utilidad en el presente proyecto, se espera que también resulten útiles en futuros proyectos:

Aprendizaje de nuevas tecnologías

El desarrollo del presente trabajo ofrecía el reto de emplear tecnologías distintas a las que puedan aparecer en los temarios de las asignaturas de la titulación de Ingeniero en Informática. Para poder llevarlo a cabo los alumnos han tenido que localizar la información relativa a las tecnologías que serían usadas para después asimilarla y finalmente poner lo aprendido en práctica.

Aparte del evidente beneficio directo de haber aprendido a manejar las tecnologías implicadas, existe también el beneficio indirecto de desarrollar la habilidad de acercarse a nuevos conceptos tecnológicos sin que existan recursos docentes disponibles.

Lectura de estándares

Los estándares (y más aún cuando son en dominios poco maduros) distan mucho de tener una orientación didáctica sino que tienden a presuponer un amplio conocimiento del dominio y, sobre todo, a estar poco estructurados. La redacción de un estándar no pretende ser una referencia didáctica a la que se pueda o deba acudir para aprender sobre un dominio, sino una concreción por escrito de una serie de normas y directrices que deben ser cumplidas.

La consecuencia de estos hechos es que (salvo que el dominio sea estable y surjan textos didácticos acerca de lo estándares) los estándares en sí tienden a ser grandes volúmenes de documentación, en inglés técnico y poco accesibles para neófitos.

En el caso particular de los estándares del IMS Global Consortium se maneja un volumen de información superior a las 5.000 páginas de texto, repartidas entre varios documentos sin ningún tipo de guía externa sobre el contenido, objetivo o utilidad de cada uno de ellos.

Siendo la lectura de nuevos estándares una actividad frecuente en el mundo de la informática, es evidente que la experiencia obtenida en el estudio intensivo de los estándares durante el presente proyecto será de mucha utilidad a la hora de enfrentarse a futuros estándares del mismo o distinto dominio.

Concreción de estándares en implementaciones

Otra tendencia observada en los procesos de estandarización es el enorme hueco temporal que se produce entre la publicación del estándar o la aparición de las primeras implementaciones de referencia por parte de los autores del estándar o grupos ajenos.

La existencia de implementaciones de referencia es fundamental, pues la mera redacción de los estándares no es suficiente para servir de guía a la hora de resolver los problemas introducidos por los matices encontrados en el momento de la implementación real.

Sin ellas, la concreción de un estándar en una implementación es muy complicada puesto que durante el proceso surgen infinidad de dudas y conflictos sin que exista un punto de referencia con el cual contrastar los resultados. Suele ser común que ni tan siquiera los autores de los estándares puedan ayudar en estos casos si ellos mismos nunca han dado el paso de implementar los estándares viendo así los huecos presentes en el estándar.

Por tanto, el desarrollo de este tipo de concreciones requiere (y desarrolla) unas determinadas habilidades para moverse a medio camino entre la interpretación del texto de los estándares, la aplicación de la lógica para resolver las contradicciones y la creatividad a la hora de llenar los huecos.

Uso de la plataforma J2EE

La plataforma J2EE es uno de los paradigmas de desarrollo más importantes del mercado en la actualidad, en pugna con la iniciativa de Microsoft .NET (con la diferencia de que la plataforma J2EE es libre y .NET no).

El trabajo ha hecho uso intensivo de las características avanzadas de la plataforma J2EE en sus distintos aspectos (programación, tecnología, metodología, arquitectura…).

Esto ha aportado a los alumnos una amplia experiencia en el manejo de dicha plataforma que, unida a la popularidad de la plataforma en la actualidad, será muy provechosa en futuros desarrollos.

Desarrollo de un sistema Web

En el mundo de las aplicaciones de empresa, la mayoría de los desarrollos son en la línea de aplicaciones web corporativas que permiten a las empresas trasladar sus procesos hacia Internet para facilitar así el rendimiento de la empresa sin preocupación por la ubicación de sus trabajadores.

Estos sistemas web, que representan la casi totalidad del volumen de desarrollo moderno, son muy similares al desarrollo web realizado en el presente proyecto (de hecho, la plataforma J2EE está específicamente diseñada para el desarrollo de dichas aplicaciones corporativas).

El resultado positivo para los alumnos es la adquisición de una importante experiencia en un campo con un volumen de negocio muy elevado.

Integración del paradigma de capas y el patrón MVC

La plataforma J2EE defiende como paradigma de arquitectura el modelo de capas explicado en secciones anteriores. Por otro lado, la disciplina de la Ingeniería del Software defiende como buena práctica el empleo del patrón Modelo-Vista-Controlador (MVC) también explicado anteriormente.

Durante este proyecto se ha conseguido integrar ambos paradigmas en una única arquitectura, proceso del que se han obtenido ideas y conclusiones que pueden resultar muy útiles en el futuro.

Trabajo con tecnologías de marcado

La fuerza con que las tecnologías de marcado (e.g. XML) han entrado en el mercado son un indicativo de la potencia y utilidad de las mismas. Muchas empresas multinacionales apuestan claramente por ellas para todo tipo de transacciones de datos que son, en última instancia, el motor del campo de la informática en la actualidad.

El proyecto <e-Aula> ha presentado siempre una clara apuesta por estas tecnologías y la colaboración en el proyecto obliga a tener un contacto muy estrecho con ellas, aportando una experiencia en su uso muy superior al que se puede obtener a partir de los planes educativos en las universidades españolas.

Metodología de trabajo en grupo con servidor central

Las metodologías de trabajo descritas en el apartado anterior (sistemas de comunicación, utilización de un servidor CVS, etc.) representan un paso de gigante respecto a las metodologías tradicionales en los proyectos prácticos universitarios.

En realidad, representan un paso adelante en la adopción de las metodologías de trabajo empleadas en el mercado profesional actual. El haberse familiarizado con dichas metodologías será útil en futuros desarrollos en equipo.

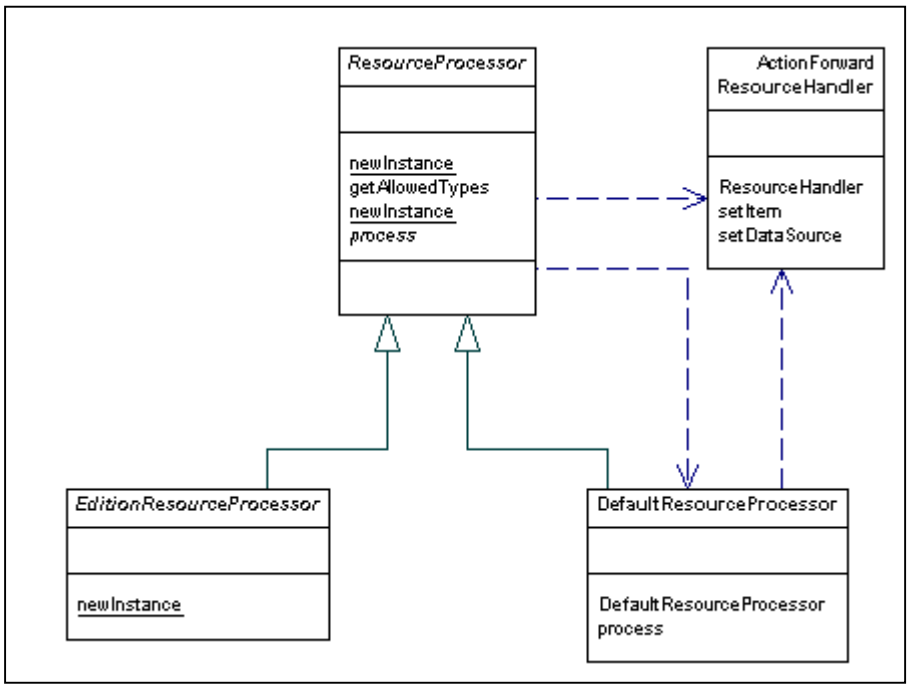
Apéndice A: Documentación adicional

A.1.- Planificación del proyecto

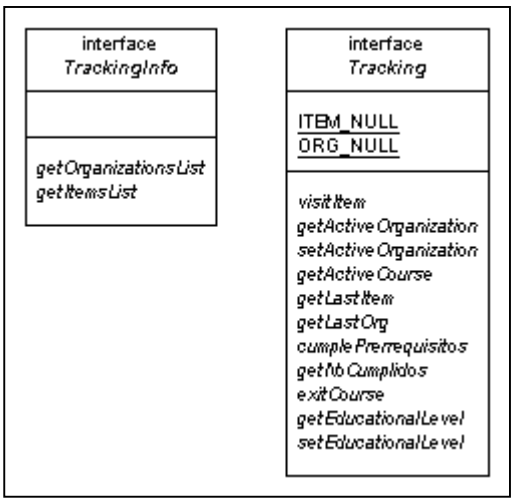
A continuación se presenta la planificación que se ha seguido a lo largo del desarrollo del proyecto. Como se puede ver en el diagrama, el proyecto ha seguido un proceso de desarrollo iterativo e incremental, dividido en dos iteraciones principales en las que se han desarrollado los principales casos de uso para el alumno y el tutor.

Esta planificación está representada mediante un diagrama de Gant generado con MSProject.

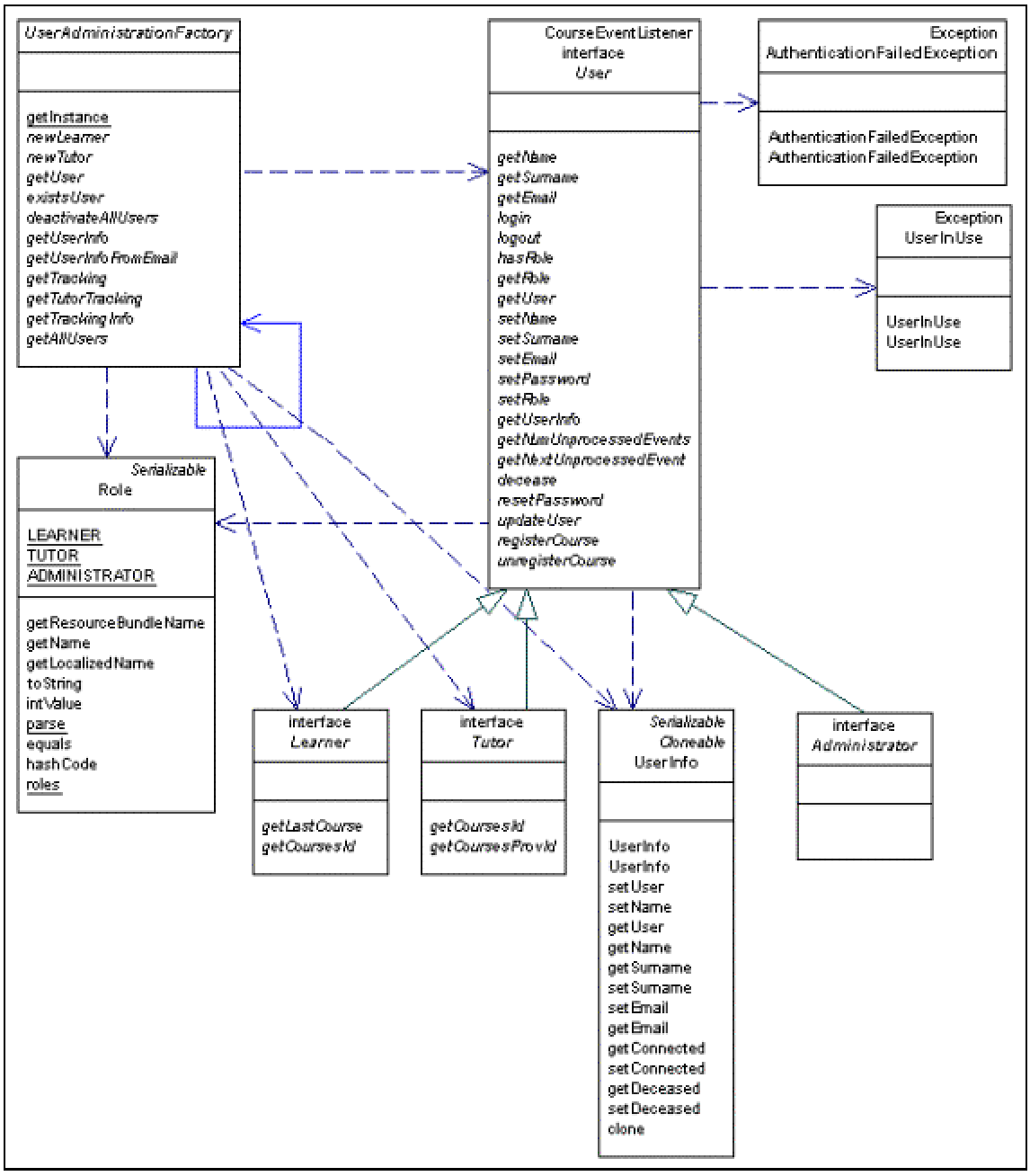
A.2.- Diagramas UML



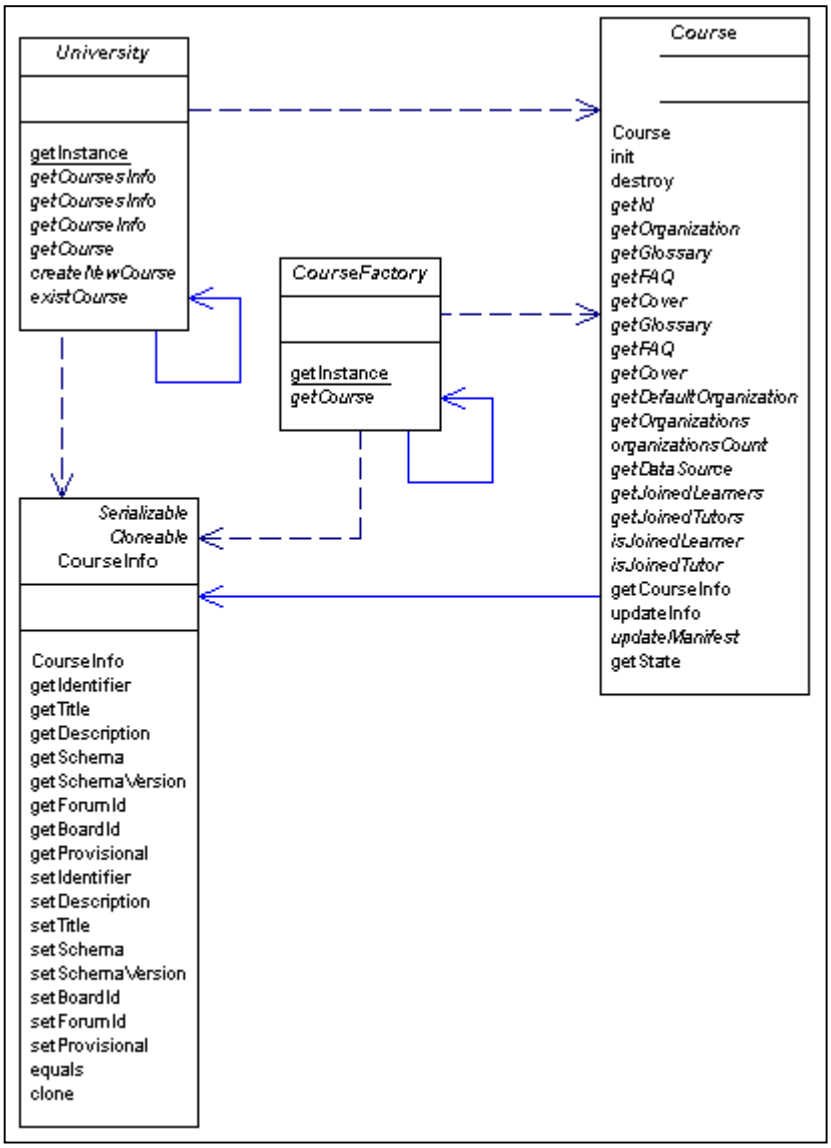
UML-1. Diagrama de clases del Resource Processor



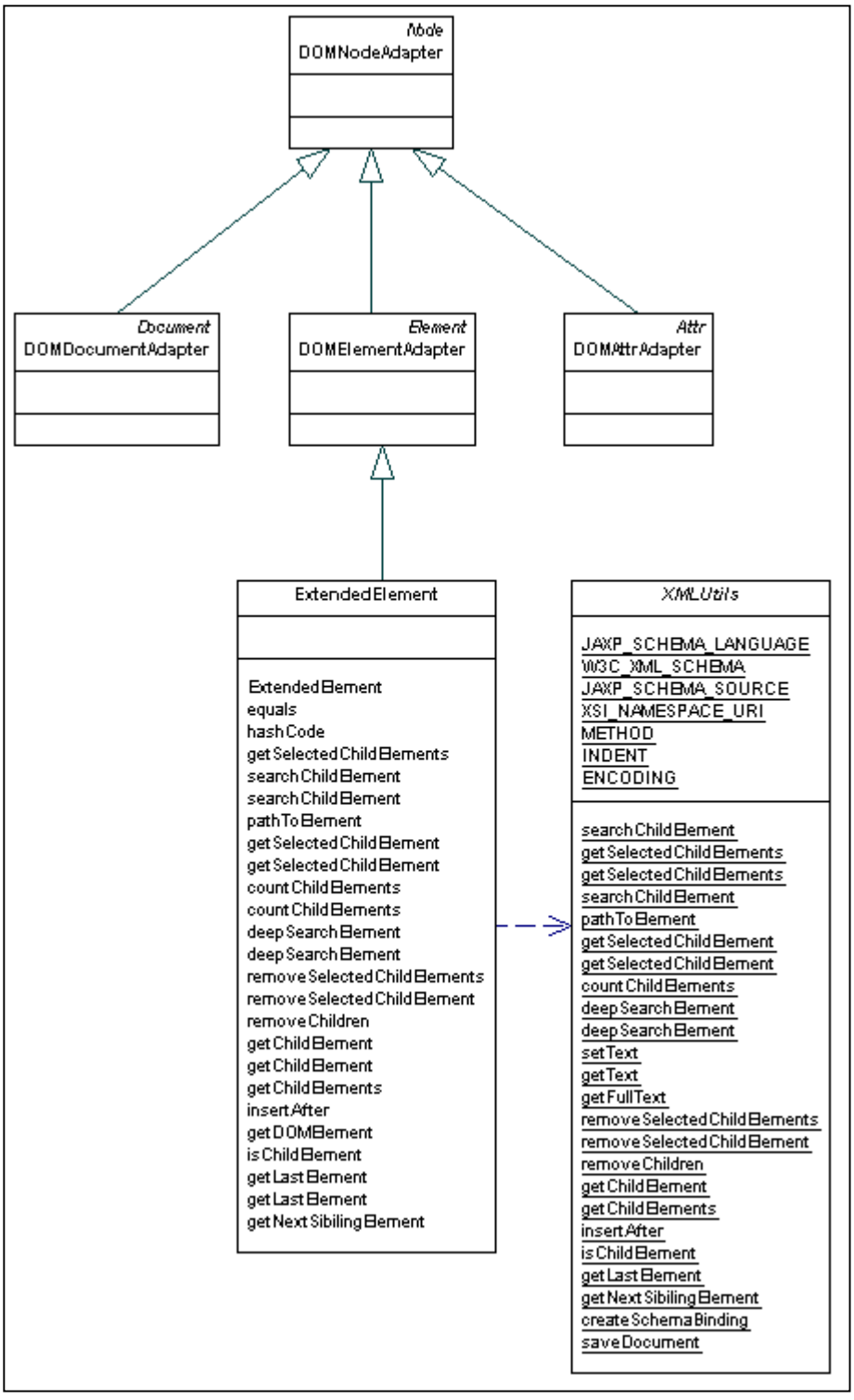
UML-2. Seguimiento de los Alumnos



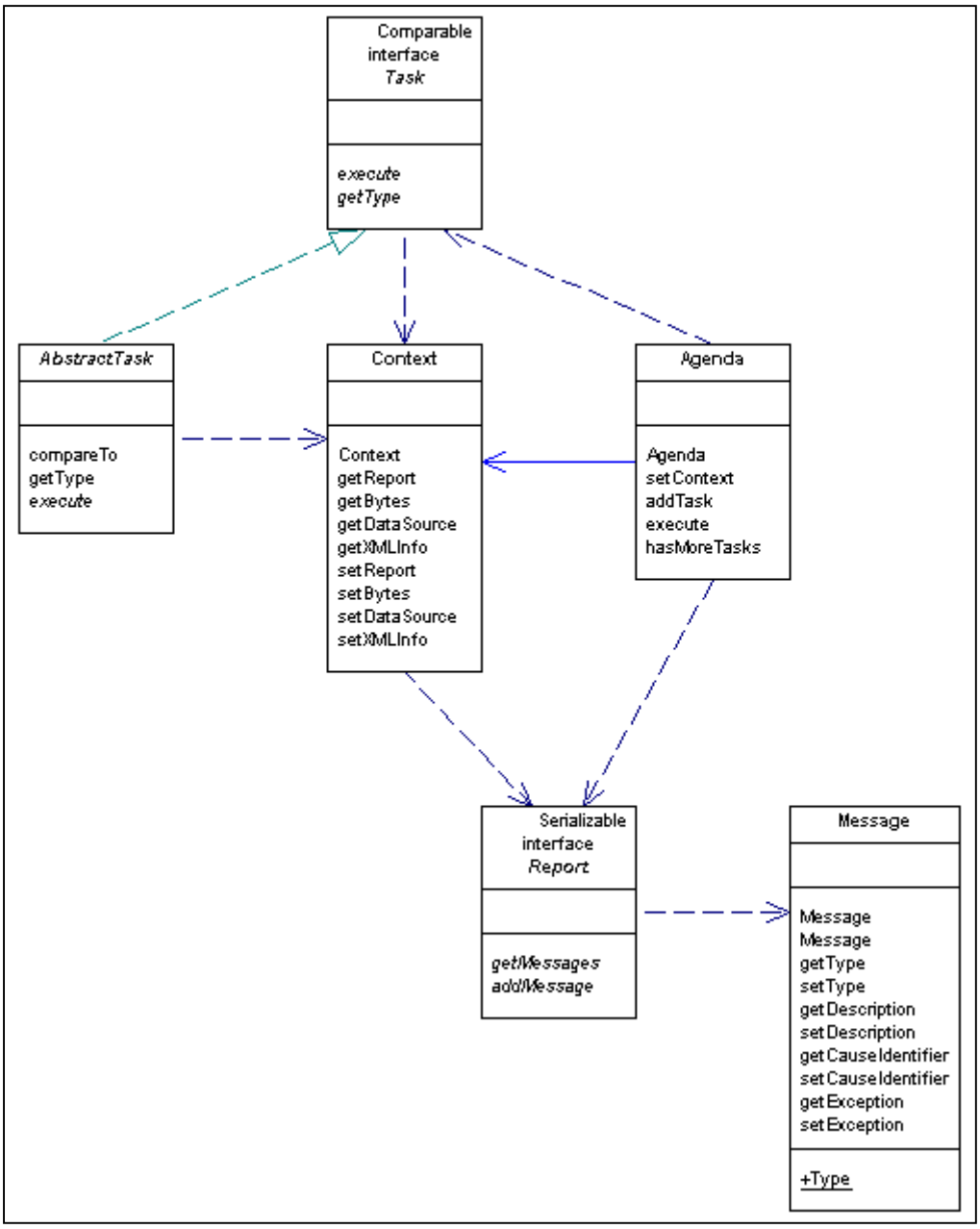
UML-3. Diagrama de clases para la administración de usuarios.



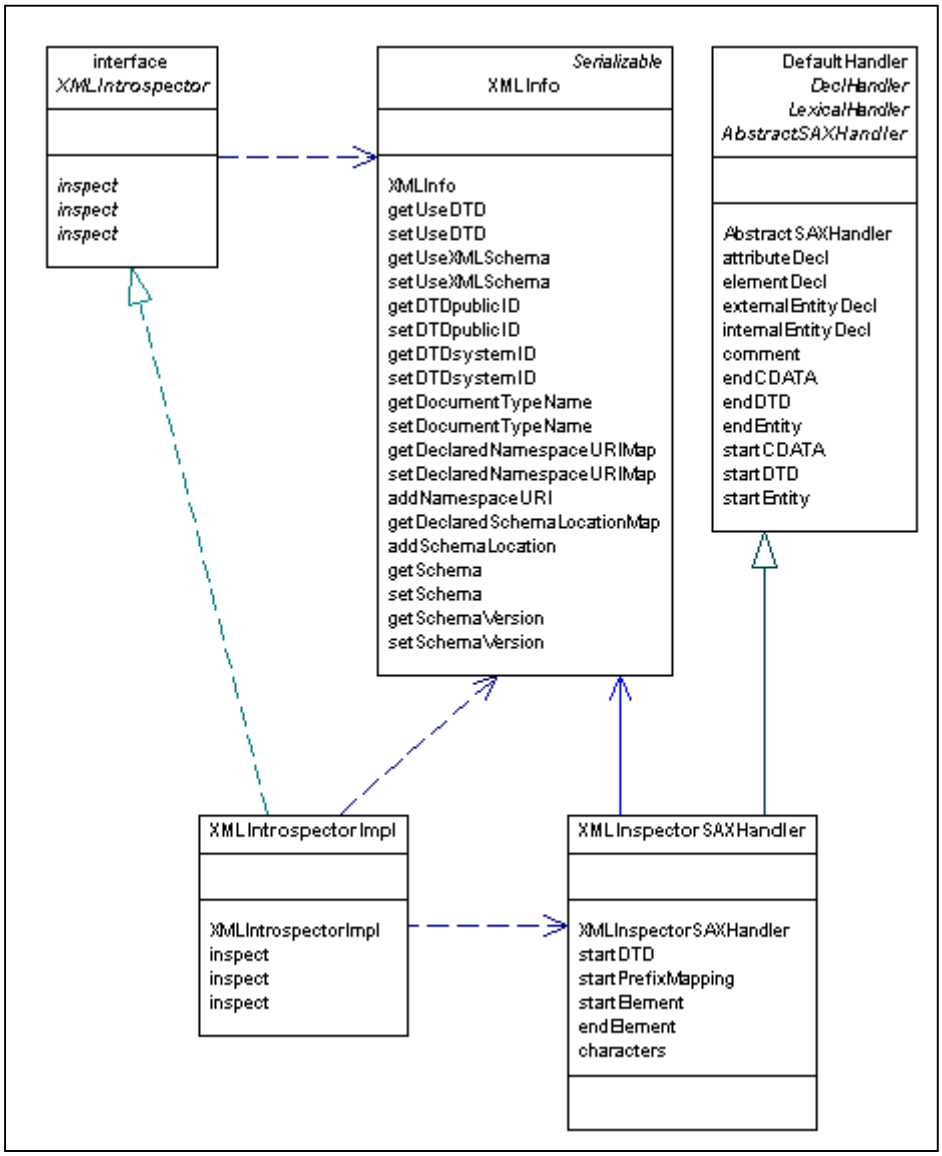
UML-4. Diagrama de clases para la administración de cursos



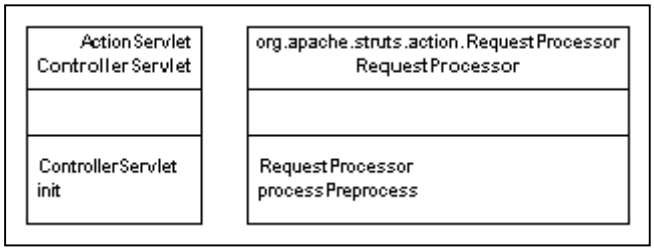
UML-5. Diagrama de clases de soporte para XML



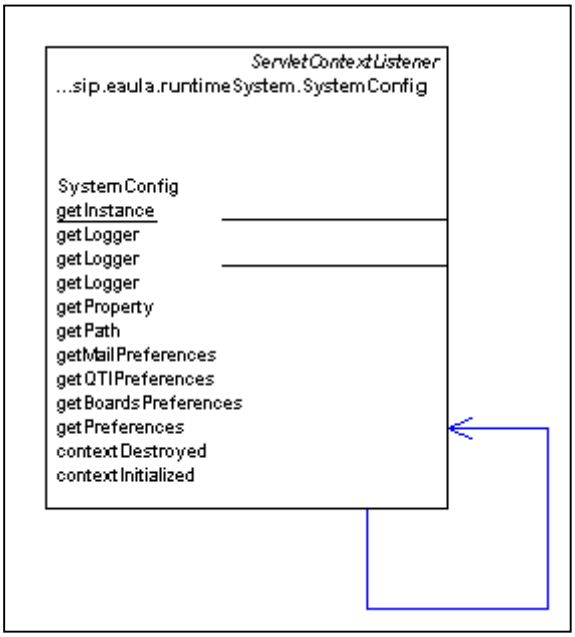
UML-6. Diagrama de clases para la Agenda



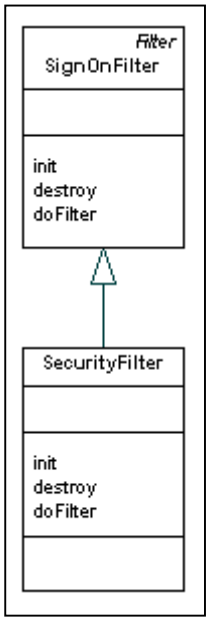
UML-7. Diagrama de clases para el sistema de Introspección



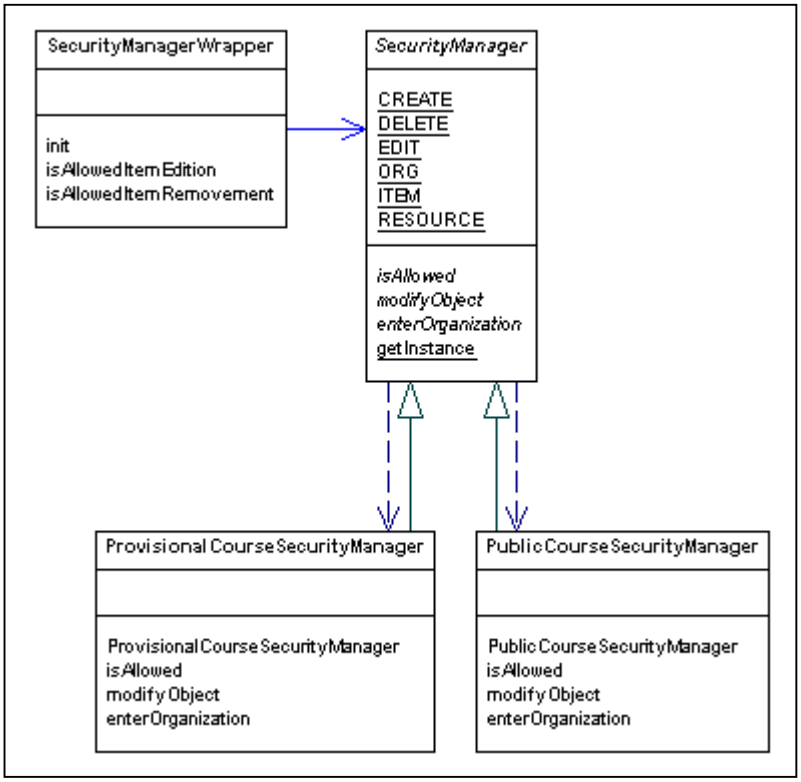
UML-8. Controlador y RequestProcessor



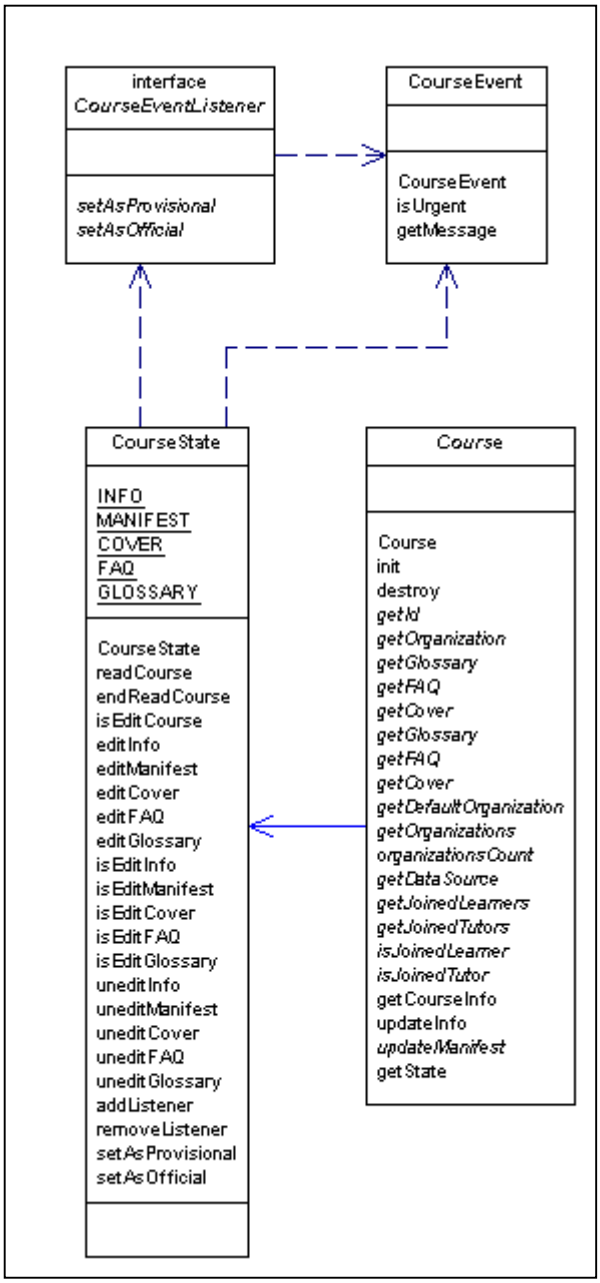
UML-9. Configuración



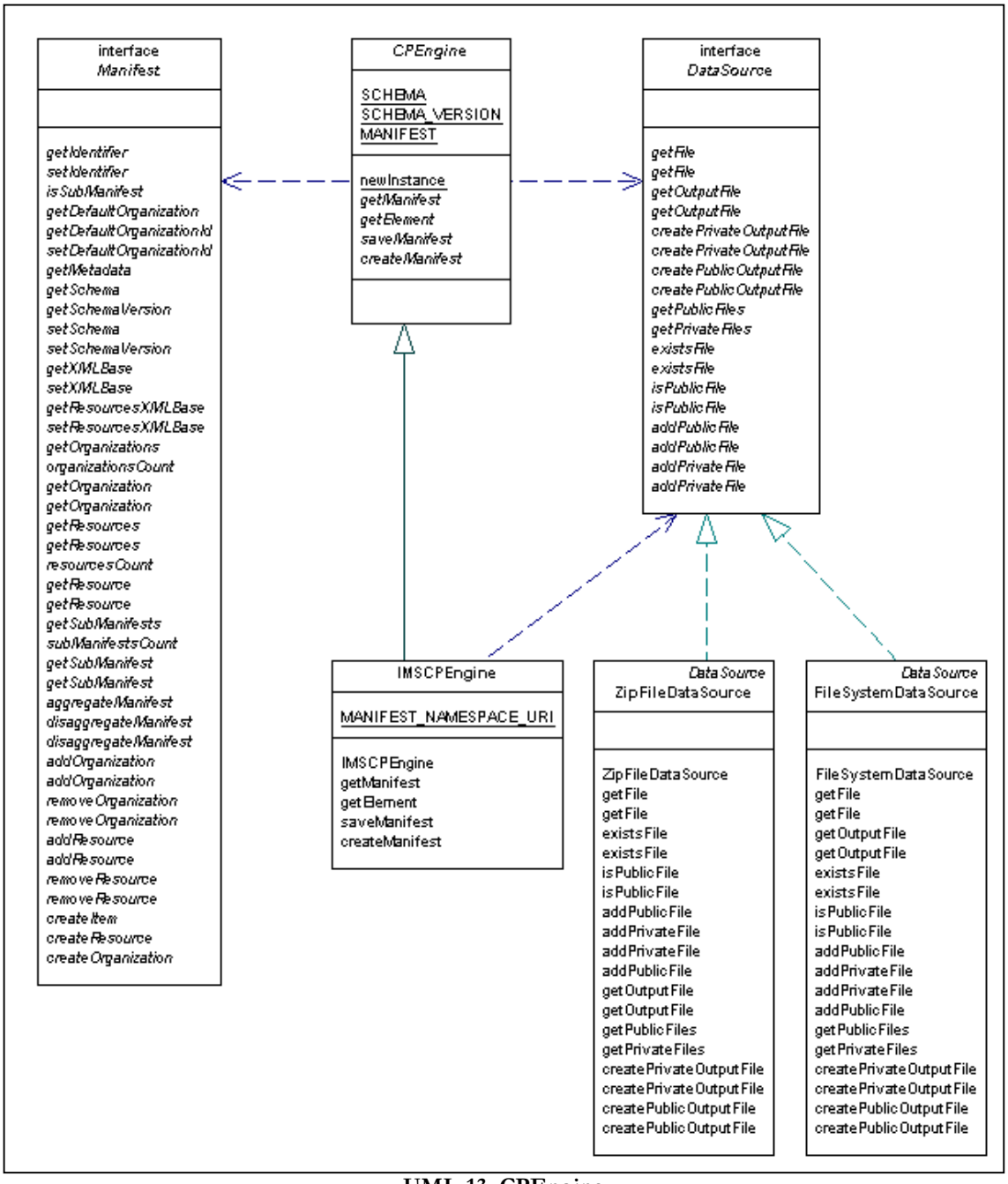
UML-10. Autenticación y Autorización



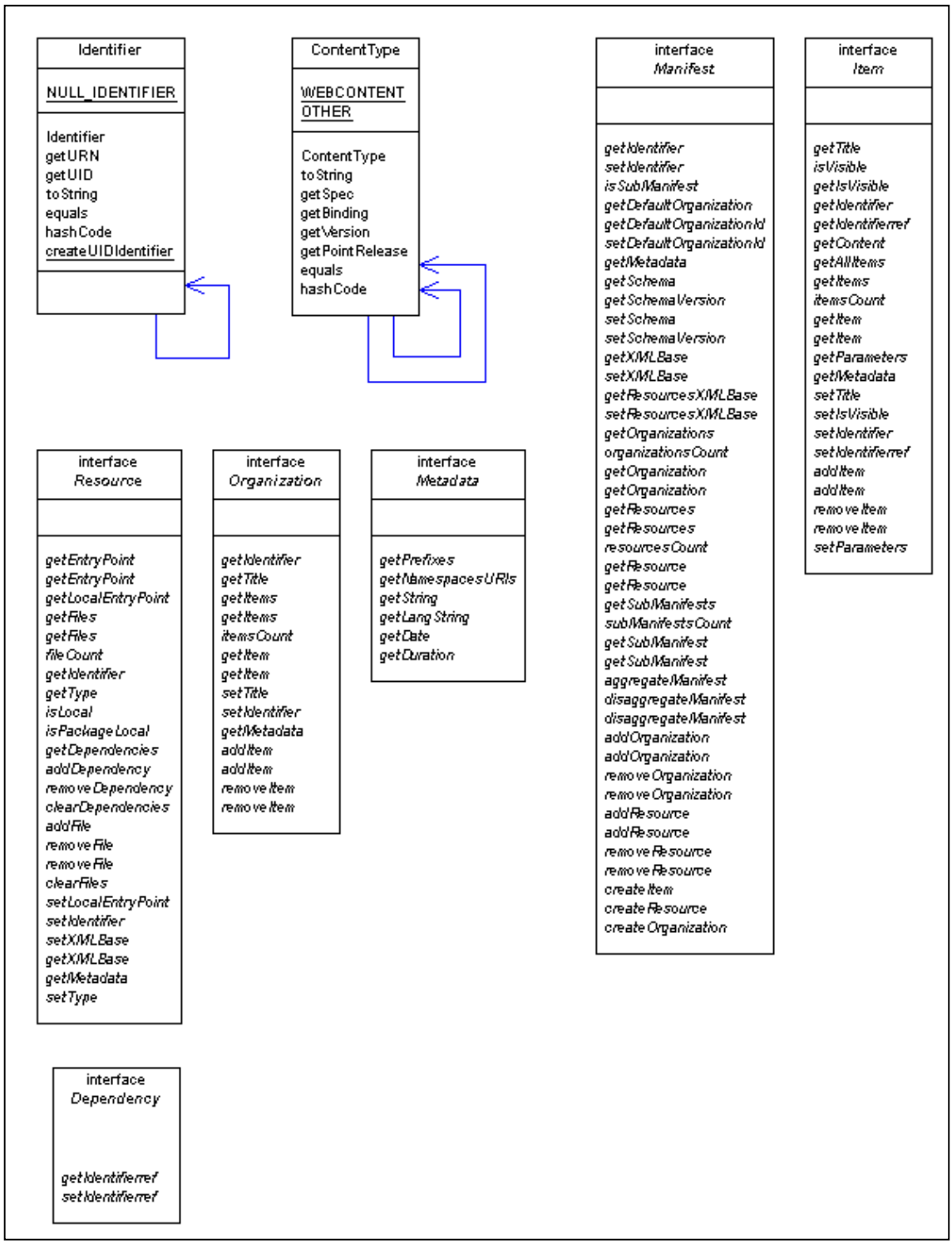
UML-11. Security Manager



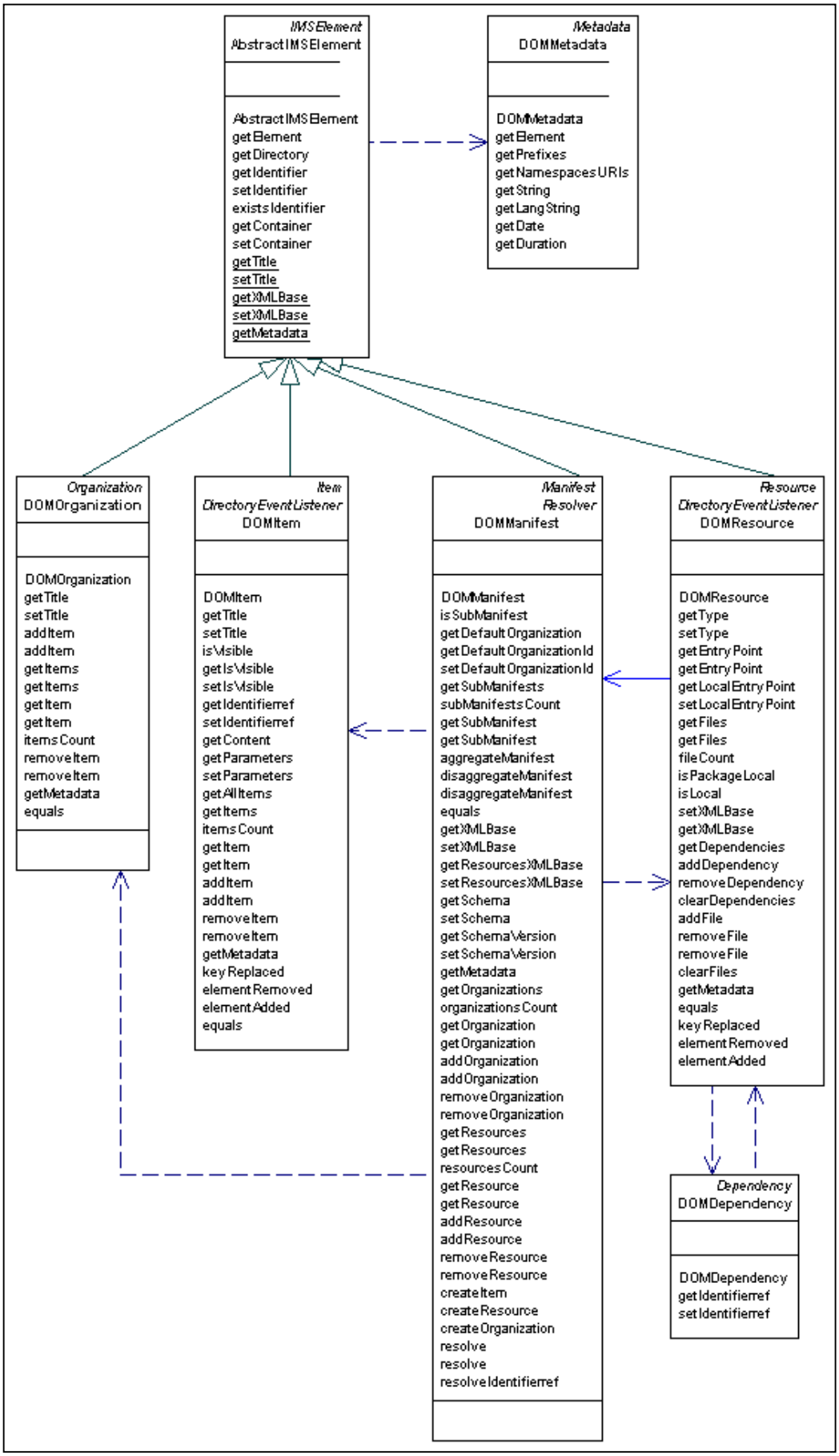
UML-12. Edición de cursos



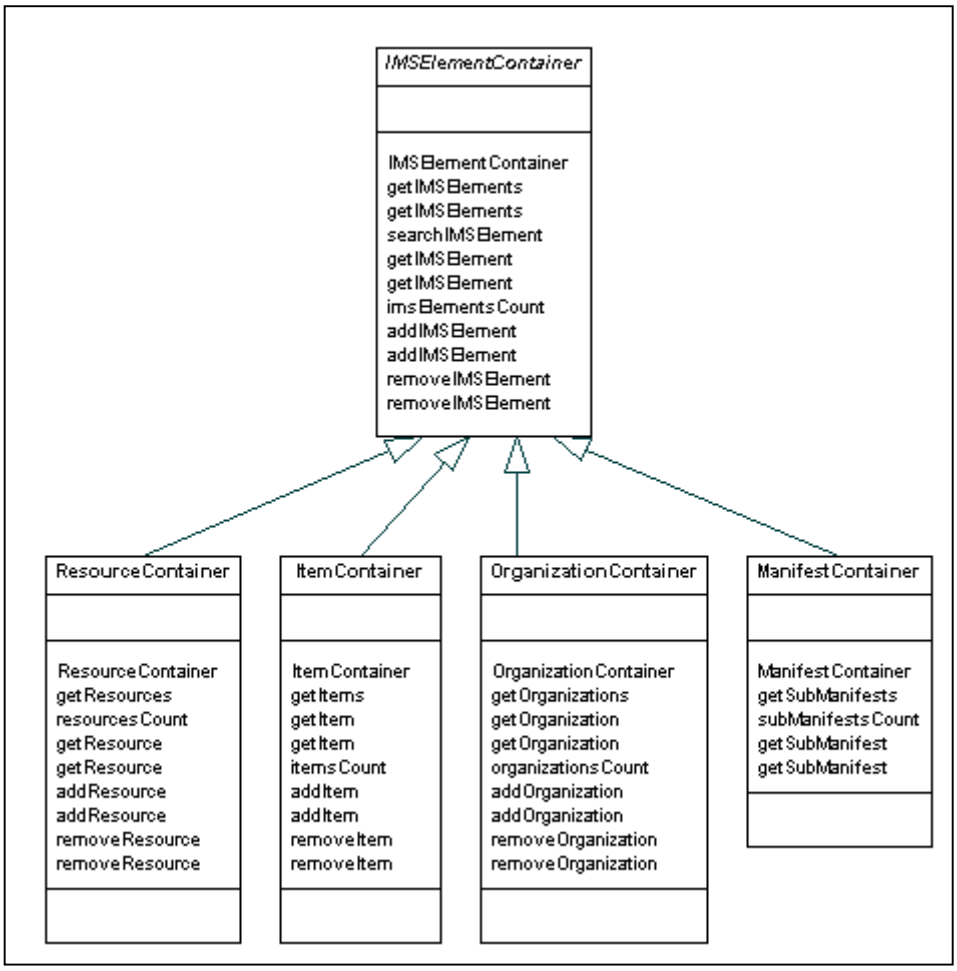
UML-13. CPEngine



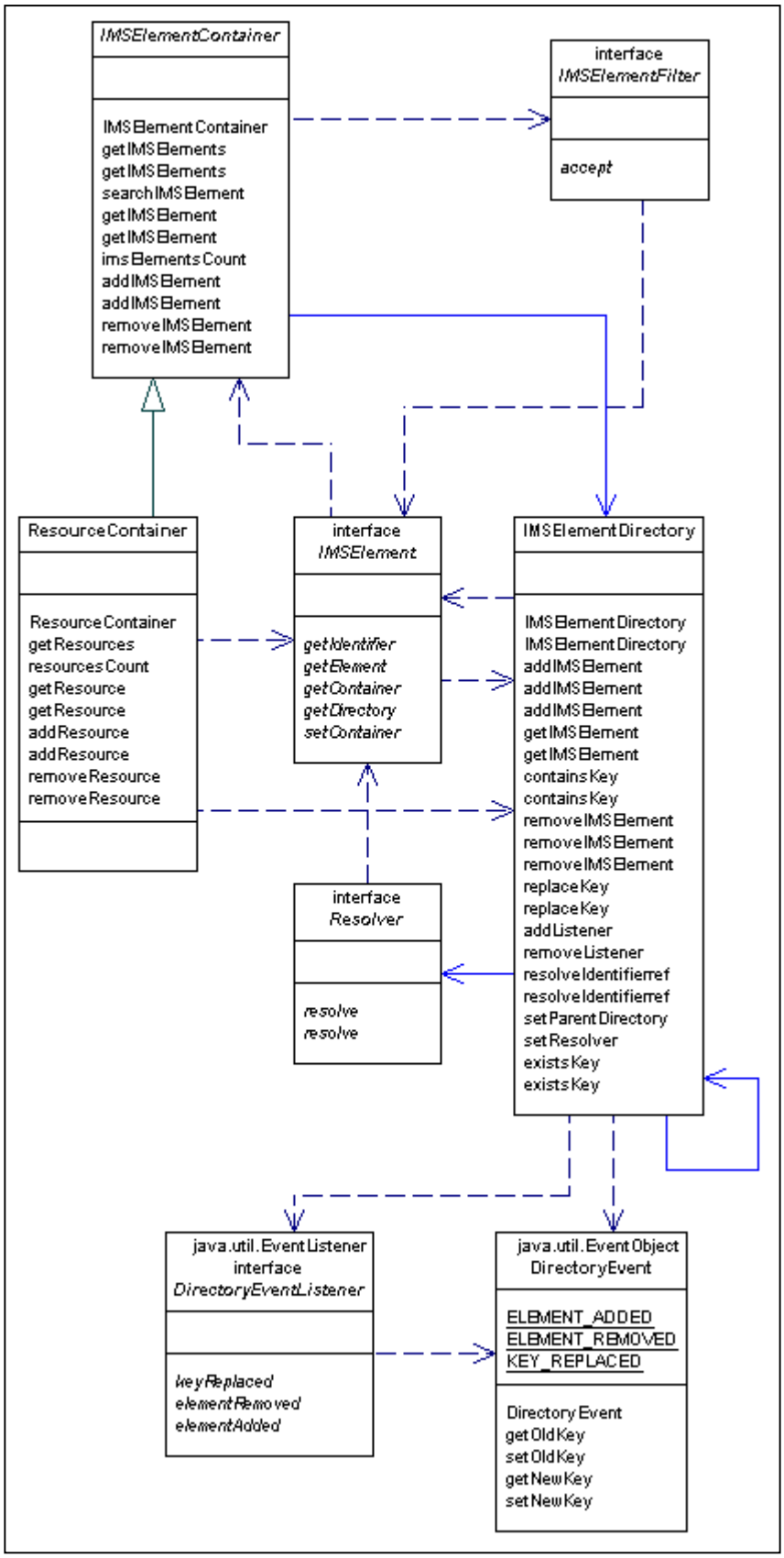
UML-14. API para el manejo del Content Packaging



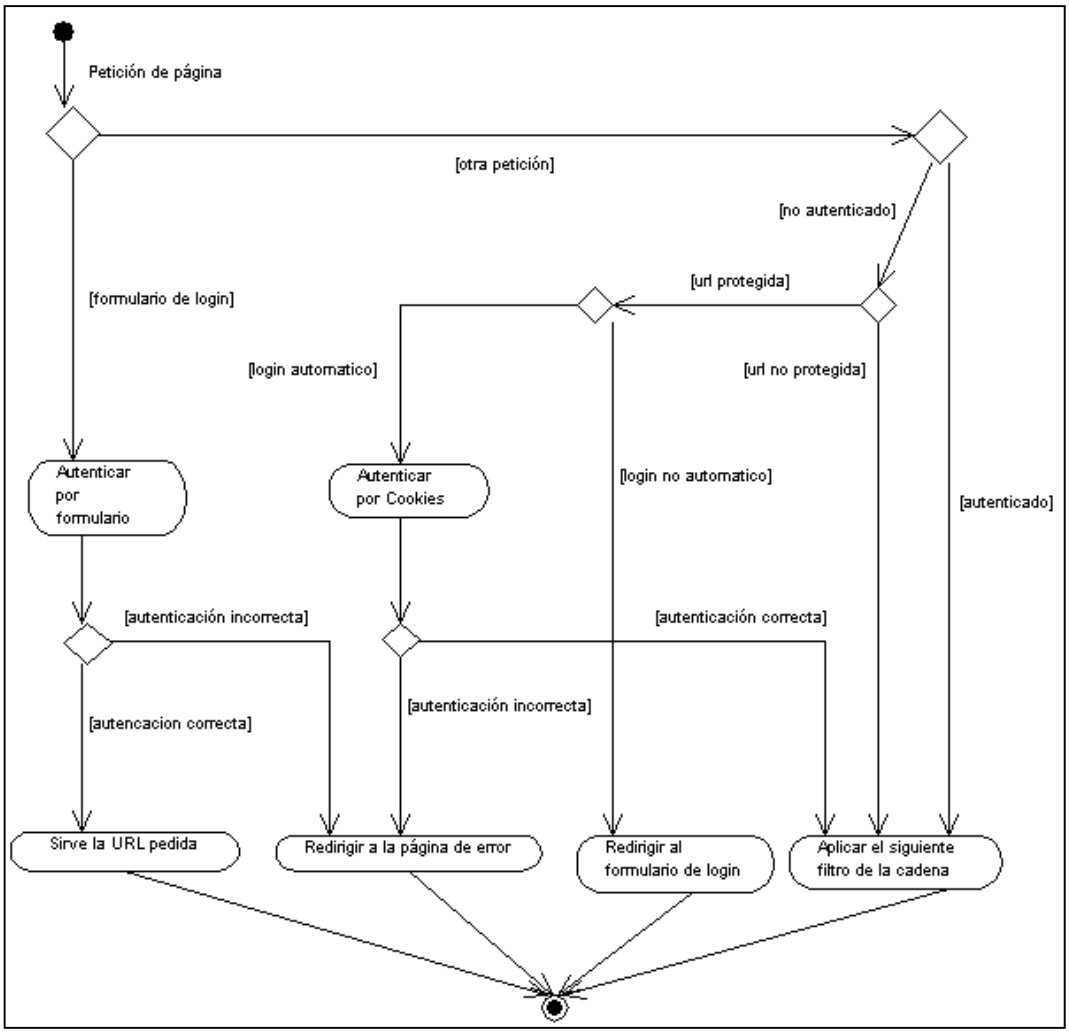
UML-15. IMS Implementation



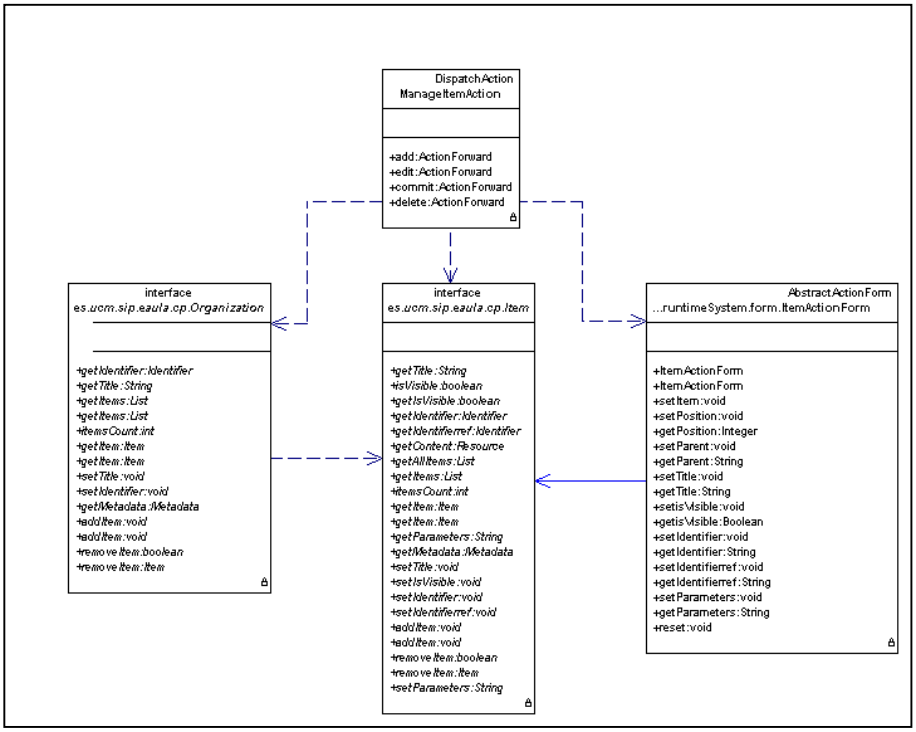
UML-16. IMS Element Container



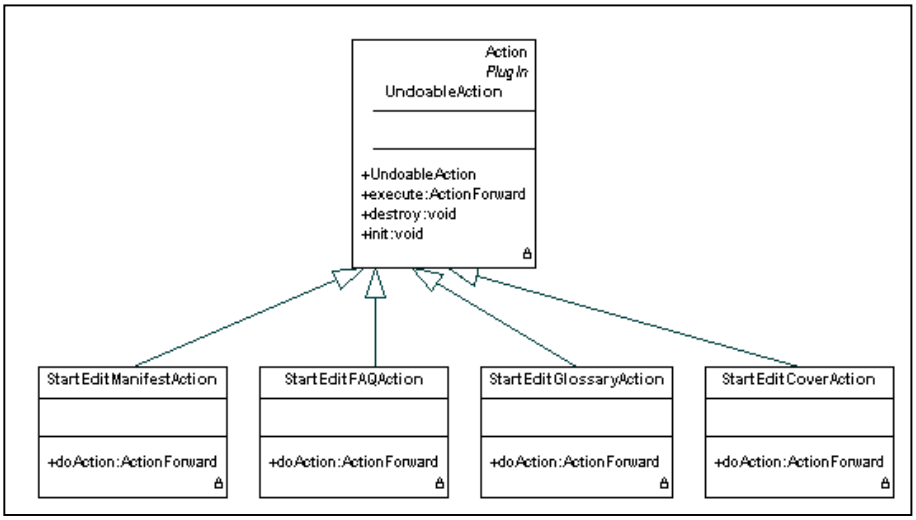
UML-17. IMS Elements



UML-18. Máquina de estados de SignOnFilter



UML-19. UndoableAction



UML-20. Ejemplo de edición

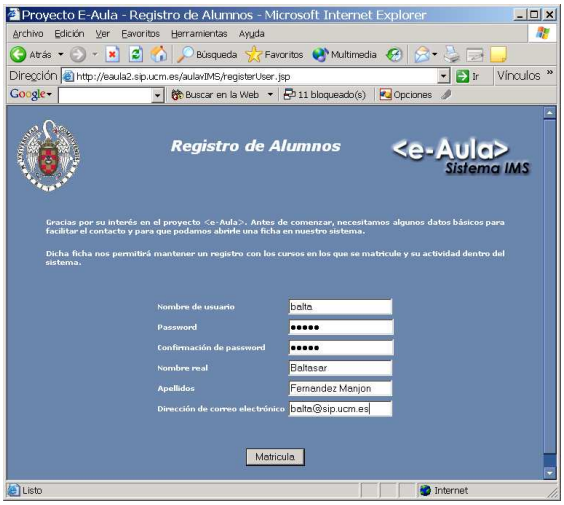
Apéndice B: Manuales de uso

B.1.- Manual para alumnos

A partir de la página general de <e-aula> en su apartado de prototipos se puede acceder a la plataforma en la cual se distribuye el curso de UML (`http://eaula.sip.ucm.es/#prototipos`). También es posible acceder al sistema desde la dirección `http://eaula.sip.ucm.es/aulavIMS`. Se accede a la siguiente pantalla:

B.1.1.- Matrícula

Para acceder al sistema <e-aula> es necesario haberse matriculado previamente. El proceso de matrícula se realiza pulsando en el botón matricula de la pantalla principal.



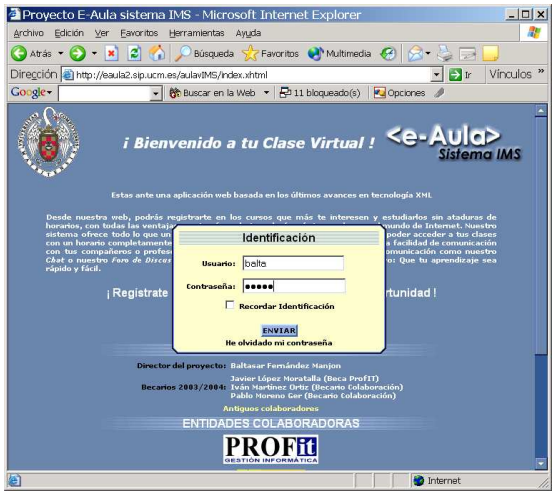
Una vez pulsado el botón aparecerá un formulario de registro en el que habrá que introducir los datos personales del usuario:

- Nombre de usuario: Nombre de acceso al sistema
- Nombre real.
- Apellidos.
- Dirección de correo electrónico.

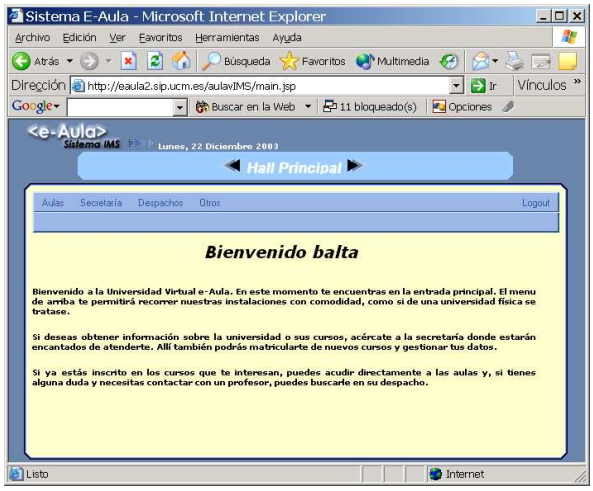
El password será generado aleatoriamente por el sistema y será enviado a la dirección de correo electrónico indicada anteriormente.

B.1.2.- Login

Si el alumno ya se ha matriculado previamente deberá pulsar el botón "entrar" en la pantalla inicial del sistema obteniendo la siguiente pantalla.



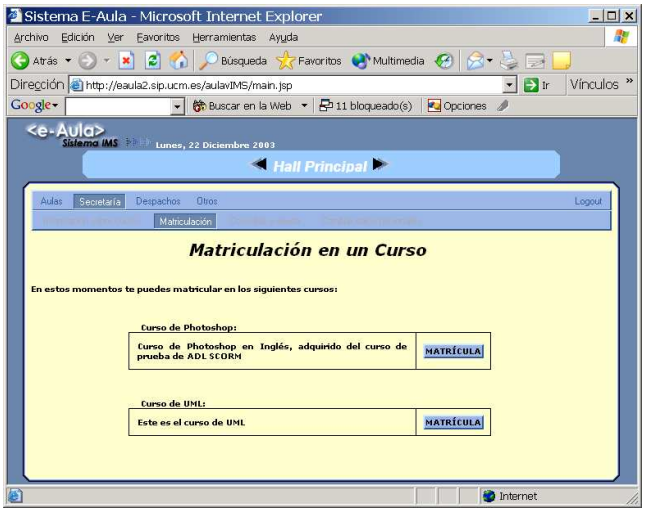
Pulsando el botón *enviar* entra en el sistema y si los datos han sido introducidos correctamente aparecerá la pantalla principal del alumno, desde la que se podrá acceder a todos los servicios de <e-Aula>



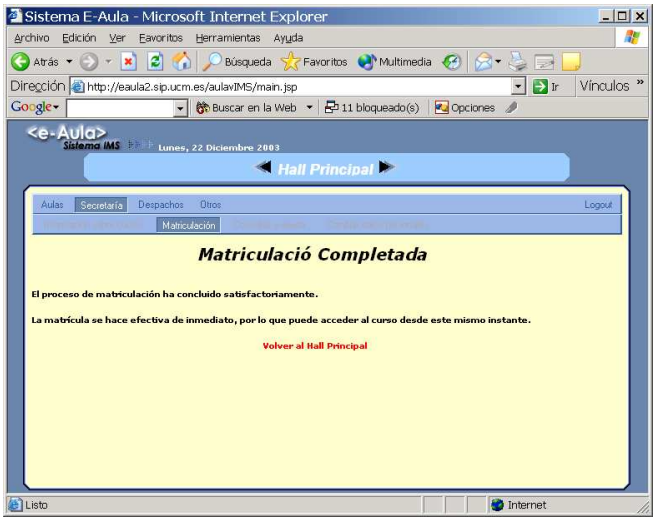
B.1.3.- Matriculación en un curso

Para acceder a los contenidos de un curso es preciso matricularse en ese curso previamente. El proceso de matriculación se realiza desde la pantalla principal del alumno.

Pulsando en el subapartado *Matriculación* del menu *Secretaría* aparecen los cursos de los que se puede matricular el alumno en ese momento:

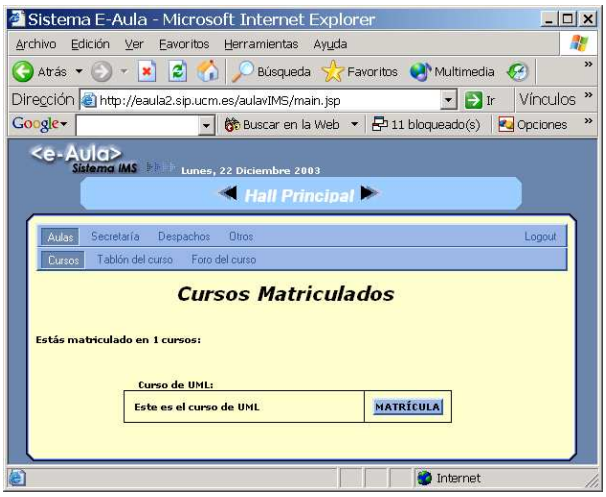


Pulsando sobre el botón *matrícula* del curso de UML el alumno quedaría matriculado en el curso de UML.

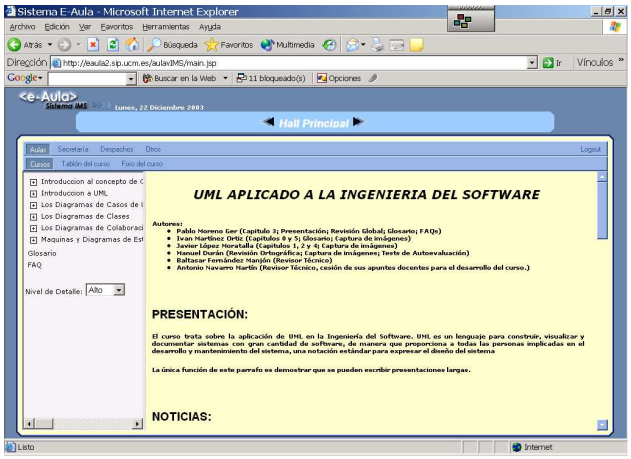


B.1.4.- Acceso a los cursos

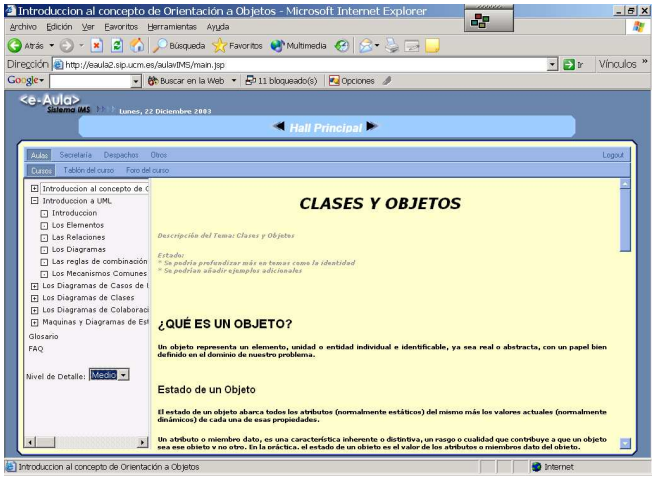
Una vez el alumno ya se haya matriculado en el curso puede acceder a sus contenidos a través del menú *Aulas* y la opción *Cursos*:



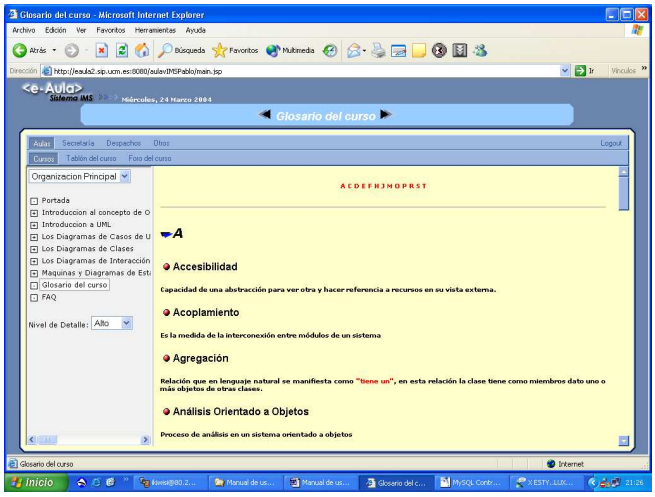
Pulsando en el botón "entrar" asociado al curso UML accederíamos al contenido del curso:



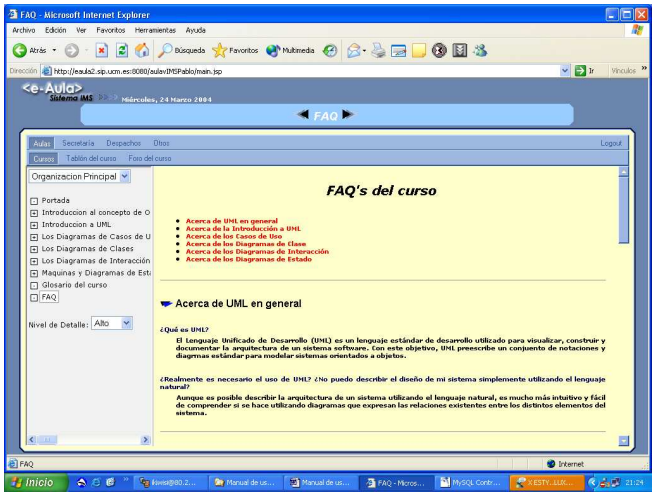
En la parte izquierda de la pantalla aparecerá el índice de contenidos del curso y el alumno podrá seleccionar el nivel de detalle con el que quiere visualizar dichos contenidos (alto, medio o bajo). El sistema adecuará la presentación de los contenidos a esta selección del usuario.



Además del contenido propiamente dicho del curso todos los cursos incorporan un glosario con la definición de los términos más importantes

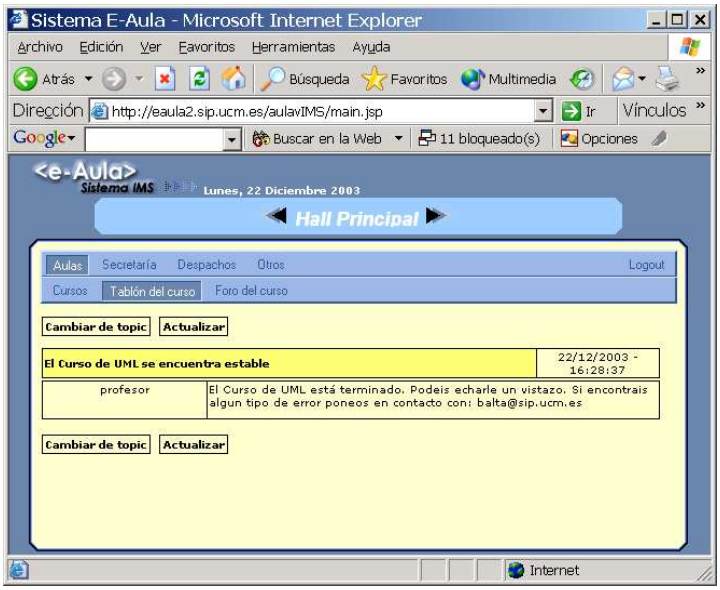


y un apartado con las preguntas mas frecuentes realizadas por los usuarios



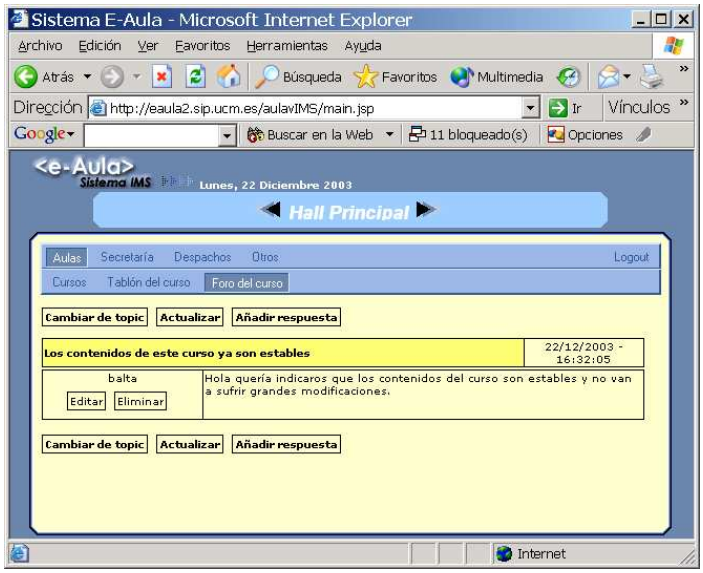
B.1.5.- Tablón

Por otro lado desde el menú *Aulas* se puede acceder también a un *Tablón del curso* donde el profesor pone los mensajes relativos a este curso concreto.

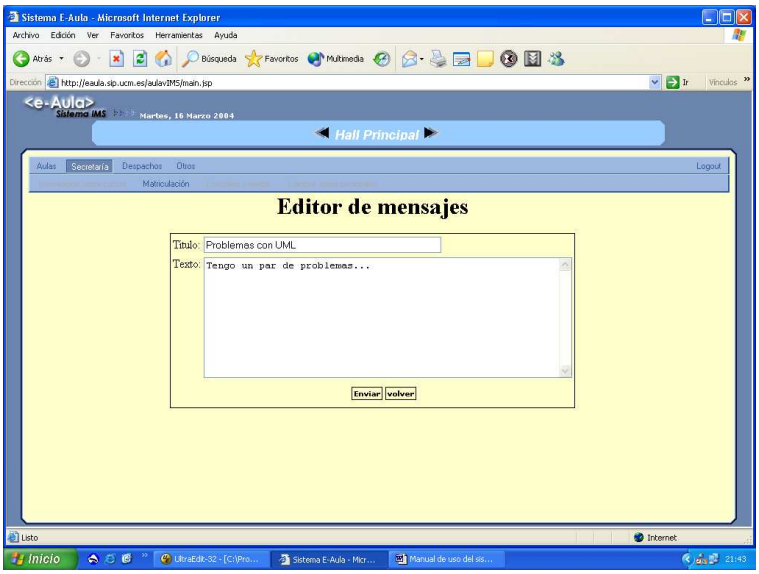


B.1.6.- Foro

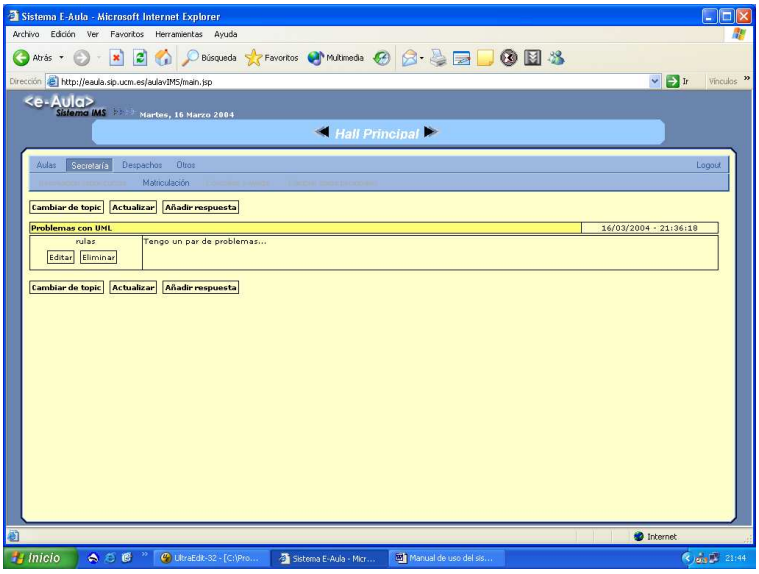
Asimismo desde *Aulas* el sistema proporciona a los alumnos una herramienta de comunicación asíncrona para que intercambien mensajes que es el *Foro del curso*.



Este foro de comunicación está abierto a la participación de cualquier alumno pudiendo responder a la contribución de otro alumno o abrir un nuevo tema. Esta edición se realiza de modo sencillo mediante una página en la que se le presenta el editor de la pregunta



Rellenando los campos *Título* y *Texto* y pinchando sobre el botón *enviar* publicamos un nuevo mensaje en el foro de comunicación entre alumnos. El aspecto sería el siguiente:



El usuario que escribió el mensaje puede volver a editarlo o eliminarlo. Además el resto de alumnos pueden contestar a las preguntas planteadas por otros alumnos.

El botón *Cambiar de topic* nos abre la ventana principal del foro para acceder a otro tema (topic).

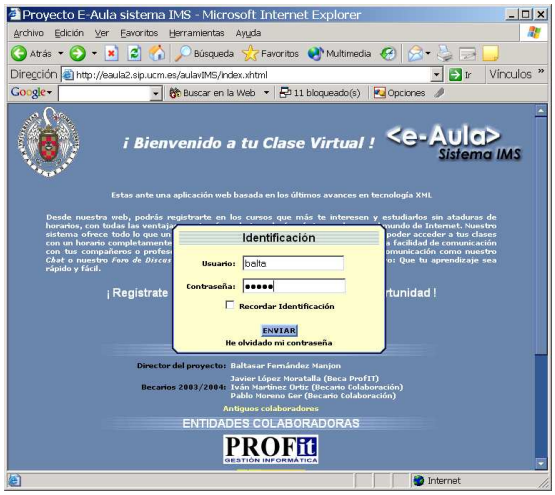
El botón *Actualizar*, refresca la página para que se muestren posibles mensajes escritos por otros alumnos mientras el usuario ha permanecido por un tiempo en la página del foro.

El botón *Añadir respuesta* muestra el editor de mensajes para que el usuario escriba un mensaje de respuesta a otro alumno.

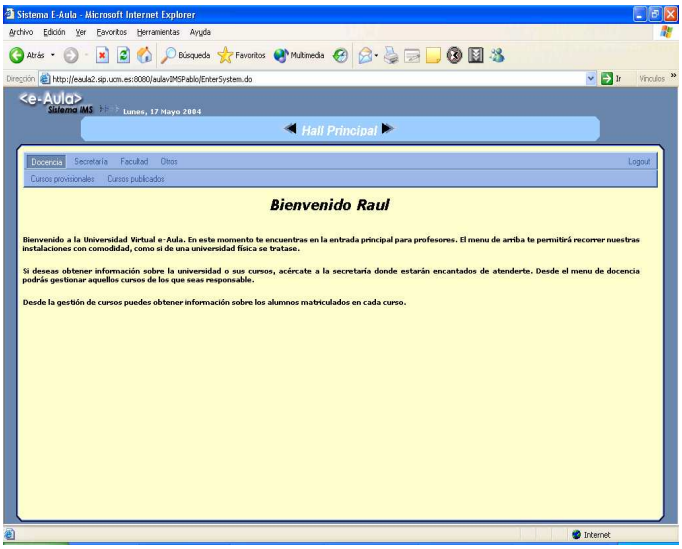
B.2.- Manual para Tutores

B.2.1.- Login

El tutor puede acceder a su área restringida pulsando en el botón "Entrar" de la página principal e introduciendo su nombre de usuario y contraseña.



Pulsando el botón *enviar* entra en el sistema

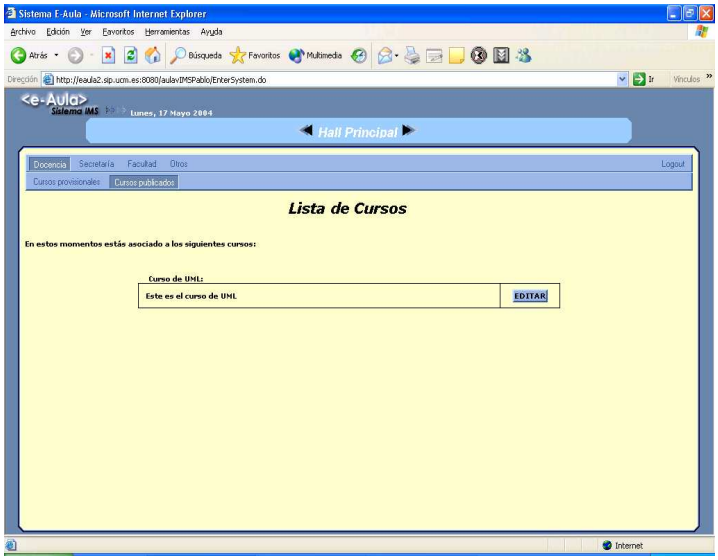


B.2.2.- Cursos publicados y provisionales

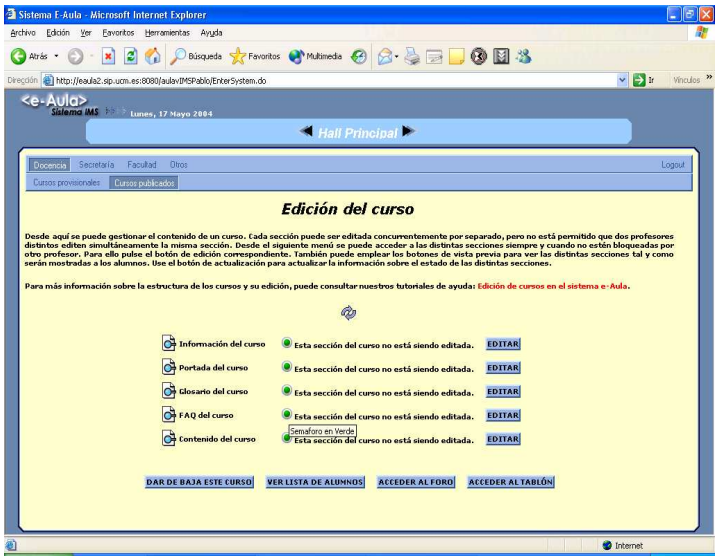
Los cursos a los que está asociado un profesor pueden ser "publicados" y "provisionales". Los cursos provisionales son aquellos que todavía están en proceso de edición y no se han dado a conocer a los alumnos. Los cursos publicados son visibles por los alumnos y están en pleno funcionamiento. Estos cursos pueden ser editados, pero sólo añadiendo nuevos ítems o recursos, nunca borrando alguno de los existentes. Para hacer una modificación drástica de un curso publicado lo mejor es darlo de baja, modificarlo y volver a darlo de alta.

B.2.3.- Gestión de cursos

En el subapartado *Cursos publicados/Cursos provisionales* del menú *Docencia*, se muestran al profesor los cursos con los que está asociado, es decir, los cursos que imparte.



Pinchando sobre el botón *Editar* al lado derecho de un curso se accede a la ventana principal de edición de cursos online proporcionada por el sistema e-Aula.



Desde la ventana anterior se puede acceder a la edición de cualquiera de las partes del curso: información, portada, glosario, FAQ y contenido del curso.

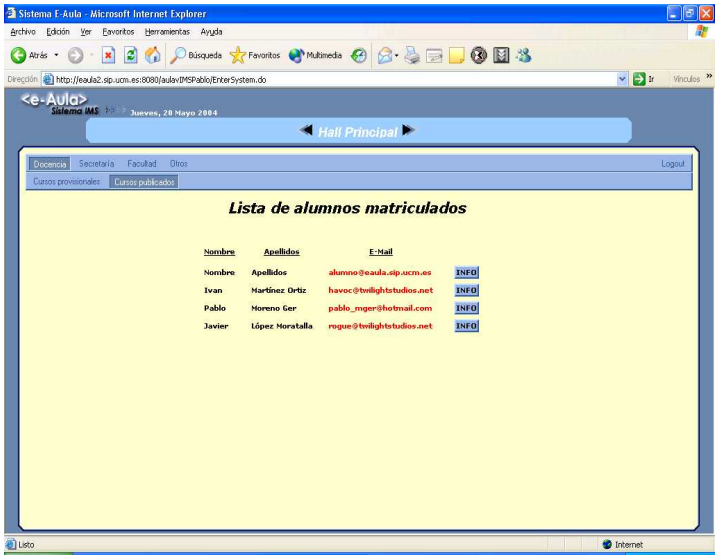
En todo momento, en esta página se muestra al profesor información visual sobre el estado de las diferentes secciones editables del curso: un semáforo verde si la sección se encuentra actualizada, un semáforo en ámbar si se han producido en la sección cambios y no se han guardado y un semáforo rojo si la sección está siendo editada en ese momento por otro profesor

Dar de alta/baja el curso

Además, pinchando en el botón *Dar de baja/alta el curso* se elimina el curso de la lista de cursos publicados y pasa a la lista de cursos provisionales o viceversa, dependiendo del estado actual del curso.

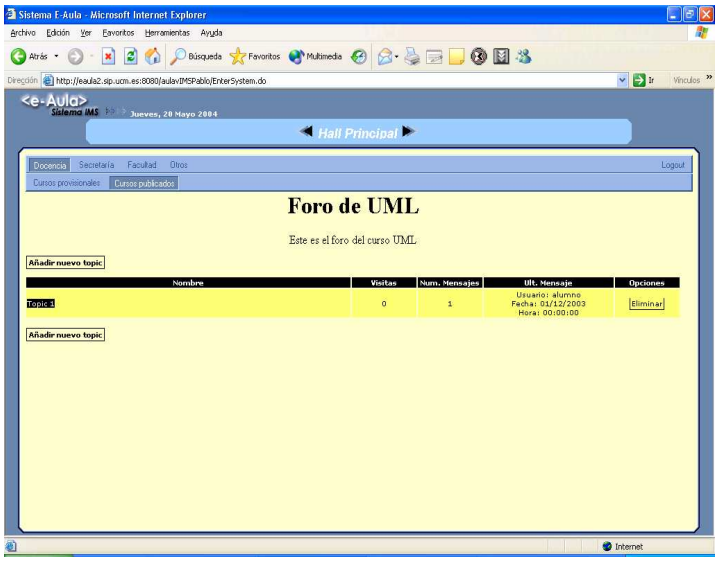
Listado de alumnos

Otra opción a realizar en esta página es la de solicitar el *listado de alumnos* matriculados en el curso, obteniendo un resultado de este tipo



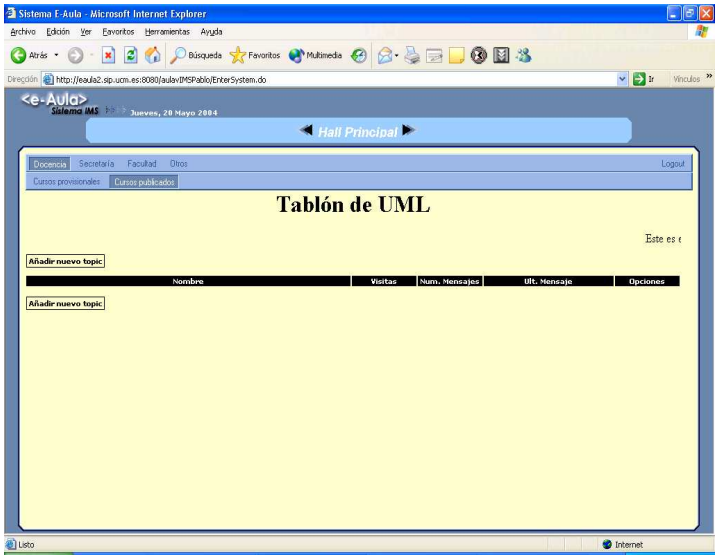
Foro

Pulsando sobre el botón *Acceder al foro* se accede al foro del curso pudiendo entrar en contacto con los alumnos del curso para responder a dudas u otra clase de información que éstos soliciten.



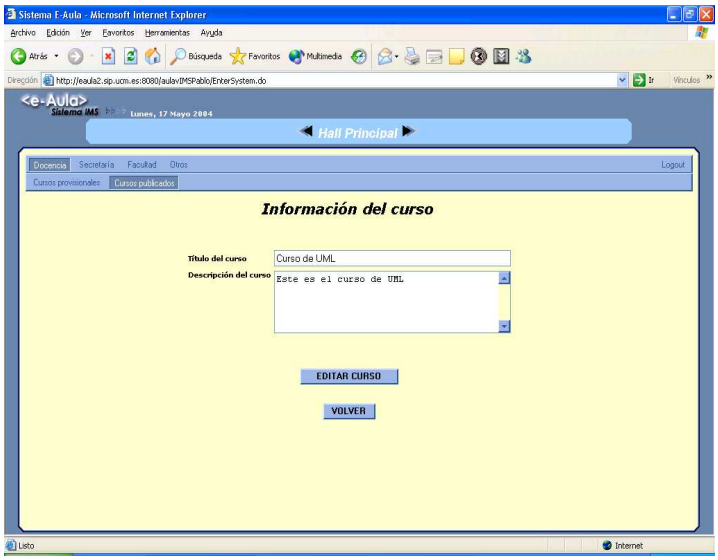
Tablón

El botón *Acceder el tablón* permite al profesor colgar noticias relacionadas con el curso en el tablón y así hacerlas visibles a los alumnos matriculados en el mismo. El aspecto que obtendríamos sería el siguiente



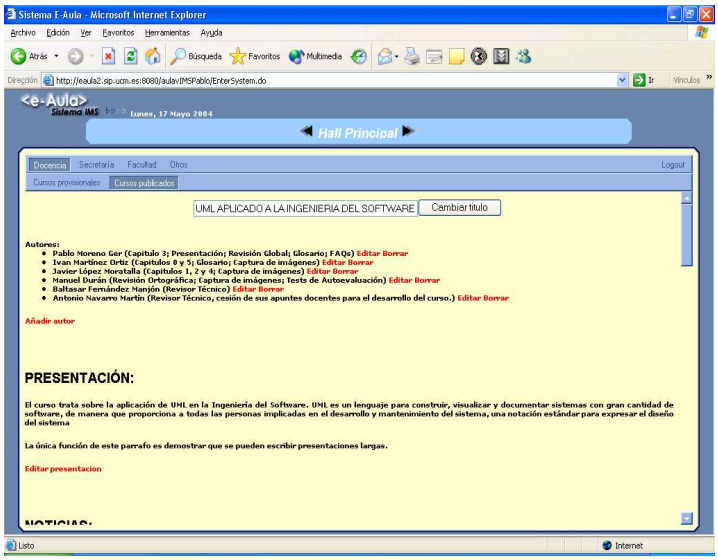
Edición de la información del curso

Pulsando el botón *Editar información* del curso aparecerá la siguiente pantalla en la que podremos editar una información general mínima asociada al curso.



Edición de la portada del curso

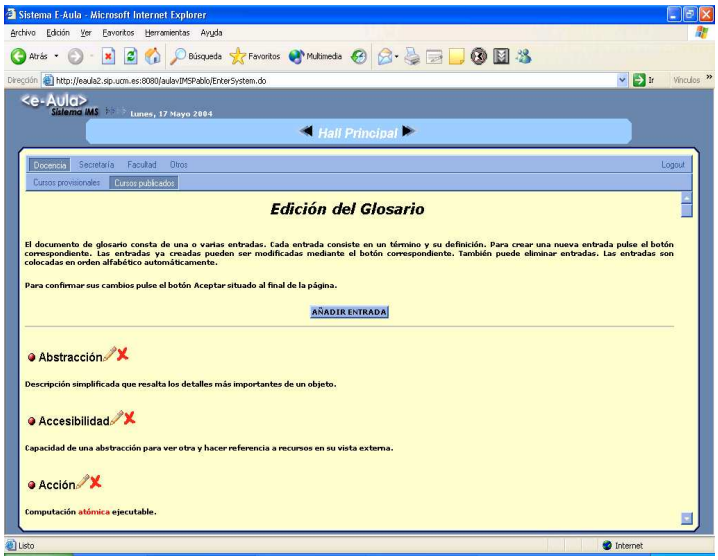
Al pulsar el botón de *Editar* asociado a la *portada* del curso se accede a la siguiente página



en la que podemos modificar el título de la portada del curso, así como añadir, eliminar o editar los autores, la presentación, las noticias, los recursos y/o los artículos asociados al curso, de manera muy sencilla.

Edición del glosario

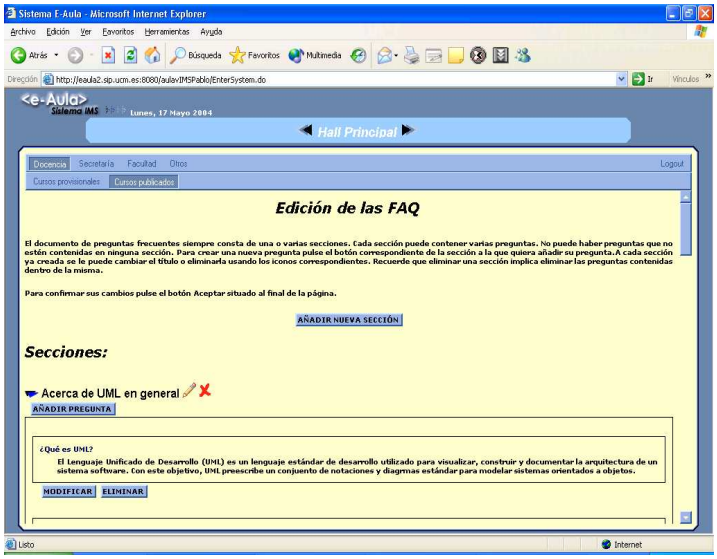
Desde la ventana principal de edición de cursos se puede acceder también a la *edición* del *glosario* del curso



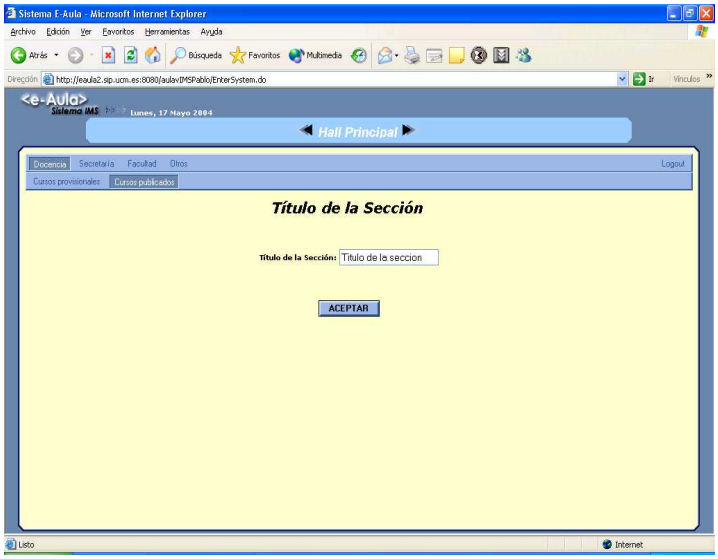
Desde la página anterior es muy sencillo editar o eliminar cualquiera de los términos del glosario. Además, es posible también añadir nuevos términos de manera sencilla.

Edición de las FAQ

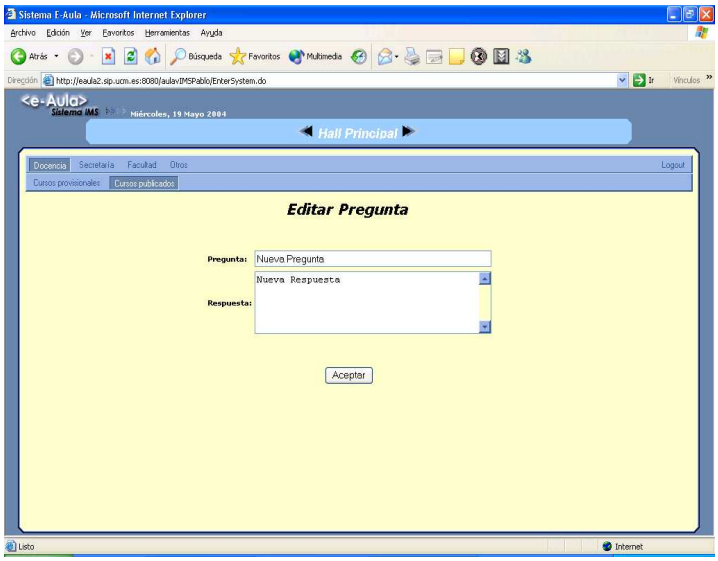
Pulsando sobre el botón de Edición de FAQ en la página principal de edición de cursos podemos modificar las preguntas más frecuentemente preguntadas por los alumnos, además de añadir nuevas o eliminar antiguas.



Todas las preguntas se encuentran incluidas en diferentes secciones, siendo éstas también editables de forma sencilla. El aspecto del editor de secciones sería el siguiente

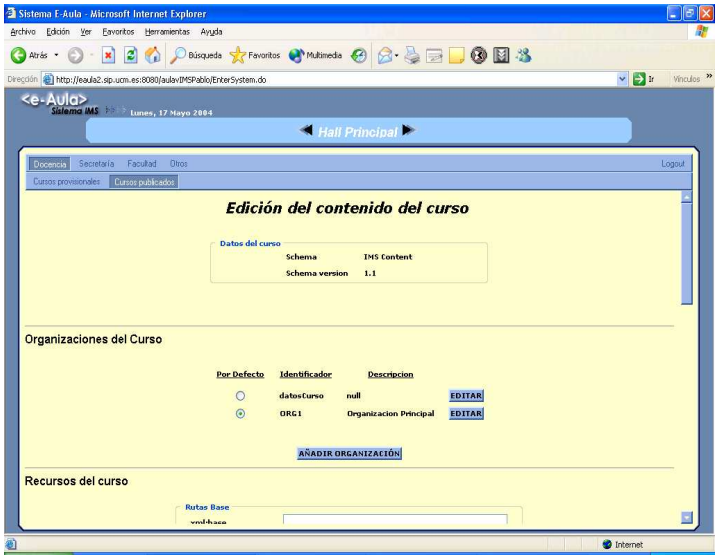


y el del editor de preguntas



Edición del contenido del curso

La *edición del contenido del curso* es un poco más compleja que los apartados anteriormente comentados. La página principal del editor de contenidos tiene el siguiente aspecto

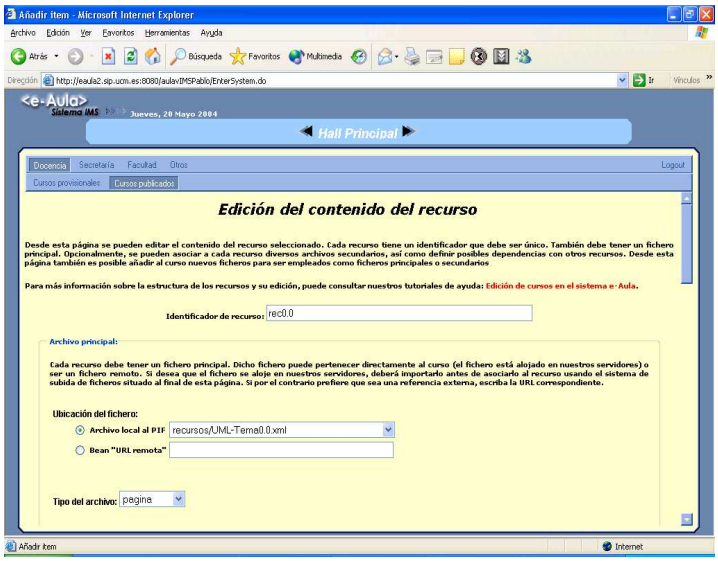


La edición del contenido del curso está basada en el *IMS Content Packaging Specification,* y está claramente dividida en dos aspectos fundamentales: edición de la organización del curso y edición de los recursos del curso.

Edición de los recursos

Al igual que ocurre con las organizaciones, en la página de edición de contenido de un curso se ofrece un listado de los recursos asociados a ese curso. Todos estos recursos son editables desde este punto, excepto tres de ellos que son obligatorios en los cursos del sistema e-Aula y cuyo punto de acceso a edición se encuentra, como ya hemos comentado anteriormente, en la página principal de edición de cursos: la portada, el glosario y las FAQ.

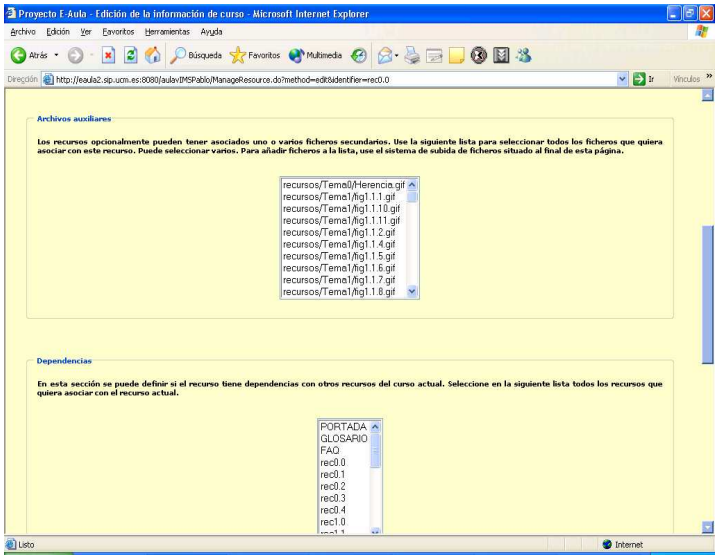
Pulsando en el botón *editar* a la derecha de un recurso de accede a la siguiente página



en la que se ofrece la opción al usuario de cambiar información asociada al recurso.

Los pasos necesarios en la edición de un recurso serían los siguientes:

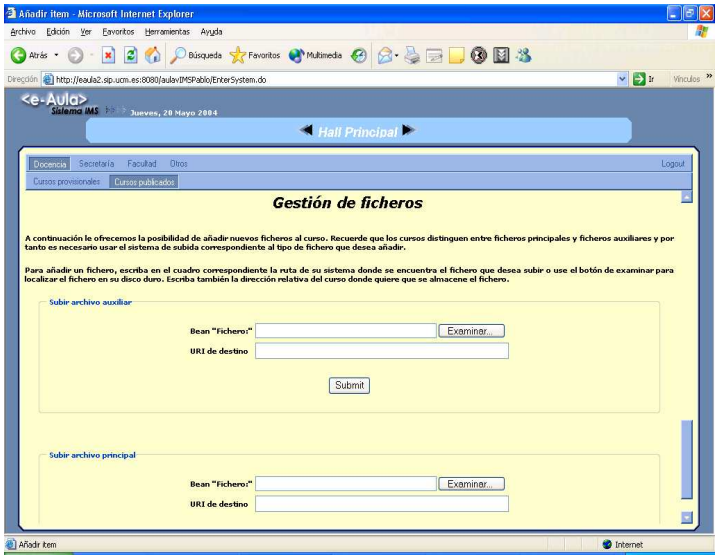
1. Seleccionar un identificador de recurso: Debe ser único dentro del manifiesto y servirá para referenciar el recurso desde las organizaciones del curso.
2. Seleccionar el archivo principal del recurso: Puede tratarse de un archivo local (que se encuentre en nuestros servidores) o una dirección URL remota. Además, en este punto se debe indicar el tipo del fichero principal a elegir entre los siguientes: Glosario, FAQ, Portada y Página (definidos en la especificación de cursos <e-Aula>) o `webcontent` (IMS Content Packaging Specification). Se puede indicar también la ruta base (`xml:base`). De no indicar ninguna ruta, la ruta base será la base del PIF.
3. Seleccionar de la lista los archivos auxiliares asociados al recurso: Archivos necesarios para la correcta visualización del archivo principal, como por ejemplo, archivos de imagen, de audio, etc. que utilice éste.
4. Seleccionar de la lista los recursos con los que tenga dependencias.



Realizando los pasos anteriores y pulsando sobre el botón de *guardar recurso* se salvan las modificaciones realizadas en el mismo. Si se pulsa sobre *descartar cambios* se vuelve a la página de edición de contenido del curso sin guardar los cambios realizados.

Los pasos a seguir cuando se desea añadir un nuevo recurso son similares a los comentados anteriormente para la edición de recursos.

En esta página, el sistema ofrece una herramienta para permitir al usuario subir ficheros a los servidores de e-Aula con el objetivo de asociar estos ficheros a recursos en la edición de un curso. El aspecto sería el siguiente:



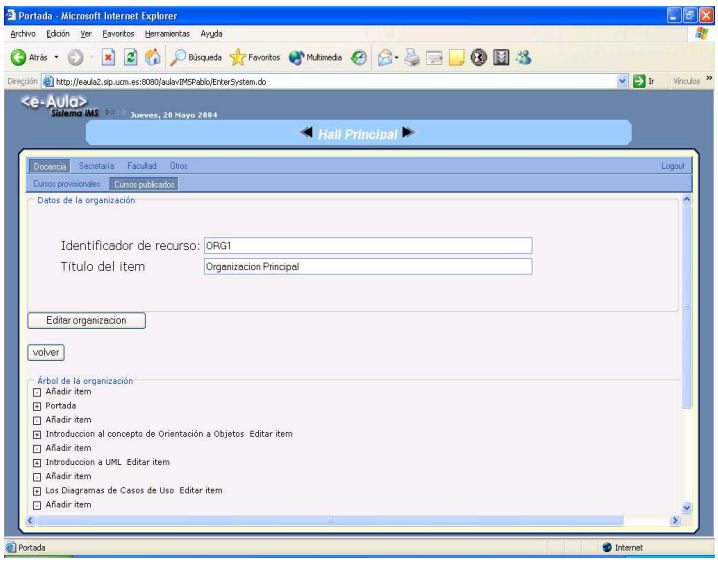
Esta herramienta permite tanto subir ficheros que serán asociados al curso como archivos principales del recurso, como subir archivos auxiliares.

Una vez añadido o editado un recurso, se mostrarán las modificaciones realizadas en la página de edición de contenido del curso. Cualquiera de los recursos que se muestran en este listado ya puede ser referenciado desde cualquier organización.

Edición de las organizaciones

En la página de edición de contenido del curso se muestran las distintas organizaciones de que consta el curso, pudiéndose éstas editar. También se ofrece la posibilidad de añadir una nueva organización. Desde esta misma página se puede seleccionar la organización por defecto que se quiere utilizar para el curso que estamos editando.

Al pulsar sobre el botón editar, al lado derecho de una organización se accede a la siguiente pantalla

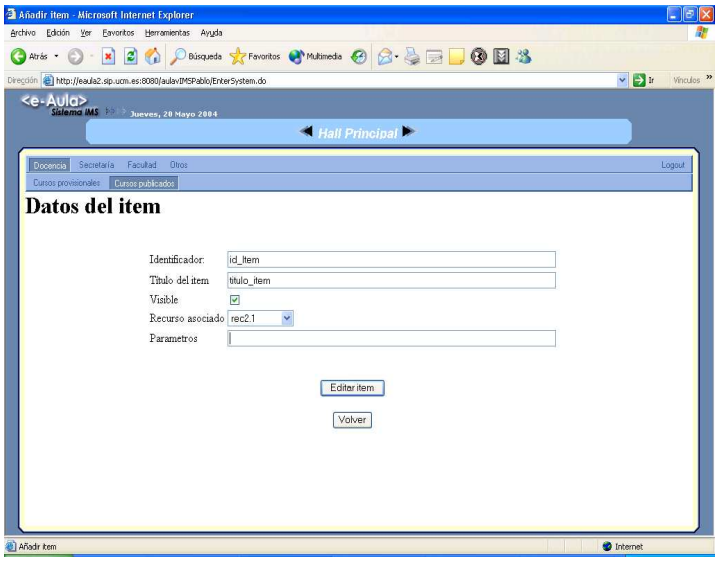


En la parte superior de la página anterior se puede observar tanto el identificador (único) de la organización como el título de la misma. Mientras, en la parte inferior, se puede observar el árbol de ítems de la organización.

La edición del árbol de la organización permite añadir ítems en cualquiera de los niveles del árbol, así como eliminar o modificar todos los ítems del árbol, excepto los relativos a la portada, las FAQ y el glosario del curso ya que son elementos imprescindibles según la definición de los cursos del sistema e-Aula.

Pulsando sobre el título de los ítems del árbol se despliega el subárbol de ítems dentro del ítem seleccionado.

Para añadir un ítem basta con pulsar sobre el botón de *Añadir ítem* en el nivel de la jerarquía donde desea que aparezca el nuevo ítem y se le mostrará la siguiente página.



en la que se solicitará al usuario que rellene los siguientes datos:

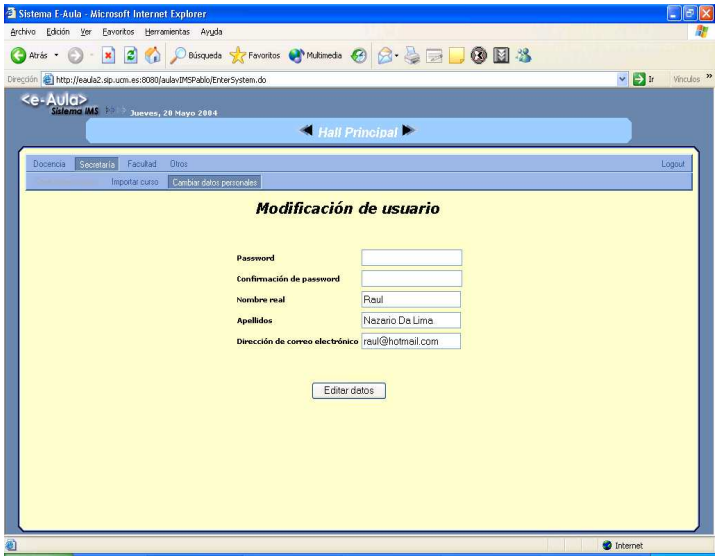
1. Identificador: Se refiere al identificador del nuevo ítem. Debe ser único dentro del manifiesto.
2. Título del ítem: Indica el título del nuevo ítem.
3. Visible: Indica si el ítem será visible.
4. Recurso asociado: Indica el recurso al que está asociado el ítem; es el punto de unión entre las organizaciones y los recursos del manifiesto. El sistema sólo muestra los recursos que ya han sido definidos para el curso actual mediante una lista desplegable para evitar problemas de inconsistencia.
5. Parámetros: Parámetros asociados al ítem (raramente se utilizan) definidos en la especificación IMS Content Packaging.

Al pulsar sobre el botón de *Editar ítem* se mostrará un formulario a rellenar similar al anterior mostrando los datos actuales del ítem y permitiendo al tutor modificarlos a su antojo.

Después de haber realizado los cambios pertinentes en la organización, basta con pulsar el botón editar organización para que éstos queden guardados en el sistema. Si no se desea que los cambios sean salvados se pulsará sobre el botón *volver*.

B.2.4.- Edición de datos personales

Desde el submenú *Cambiar datos personales* de *Secretaría* se pueden modificar los datos personales del profesor.



B.3.- Manual para Administradores

B.3.1.- Instalación y configuración

Requisitos (sistema en producción)

Para que pueda instalarse el sistema <e-Aula> es necesario que se tenga instalado previamente el siguiente software:

1. Java JDK 1.4.X
2. Apache Tomcat 4.1.X ~ 5.0.X
3. MySQL 4.1.16
4. Opcionalmente Apache HTTP web Server.

Si deseamos instalar el servidor web Apache, debemos integrar este servidor con el contenedor Apache Tomcat..

Requisitos (sistema demo)

Para que pueda instalarse el sistema <e-Aula> es necesario que se tenga instalado previamente el siguiente software:

1. Java JDK 1.4.X
2. Apache Tomcat 4.1.X ~ 5.0.X

Requisitos (sistema de desarrollo)

Para que pueda instalarse el sistema <e-Aula> es necesario que se tenga instalado previamente el siguiente software:

1. Java JDK 1.4.X
2. Apache Tomcat 4.1.X ~ 5.0.X
3. MySQL 4.1.16
4. Apache Ant

Instalación <e-Aula> en producción

Tras la instalación del software requerido por el sistema. Además necesitamos el archivo zip, que se encuentra en el CD del proyecto, que contiene la versión de producción del sistema. Los pasos a seguir para que la aplicación esté disponible son

1. Obtener el archivo zip en el que se encuentra la aplicación completa, los script para generar la BD y la definición del contexto de Apache Tomcat.
2. Descomprimir el archivo en donde se considere oportuno.
3. Generamos la base de datos a partir de los scripts. En estos se incluye tanto la creación de la base de datos, como la creación de las tablas, creación del usuario para la BD y carga de datos iniciales. Los archivos pueden ser ejecutados mediante el uso de la herramienta mysql.exe incluida con el servidor MySQL.
4. Editamos el archivo de definición de contexto, el archivo "`aulav.xml`" para adecuar las rutas y parámetros de la BD (consúltense los siguientes apartados).

5. Copiamos el archivo de definición de contexto dentro de la carpeta `webapps` de Apache Tomcat y la aplicación estará disponible para su uso (nótese que Apache Tomcat debe estar configurado para permitir el arranque en caliente de aplicaciones). En caso de que la aplicación no esté disponible en pocos momentos será necesario utilizar el Tomcat Manager para poder inicial la aplicación manualmente.

En el caso de que deseemos que la aplicación esté disponible cuando el servidor se reinicie, es necesario copiar el contenido del archivo de definición de contexto dentro del archivo "`server.xml`" que se encuentra dentro de la carpeta `conf` en el directorio donde se encuentra instalado Apache Tomcat.

Instalación <e-Aula> en versión demo

Para instalar una demo del sistema, podemos obtener desde la página del proyecto un instalador con todos los requisitos necesarios para poder utilizar el sistema.

En el caso de que tenga instalado todo el software que es requerido, puede obtener un instalador tamaño mucho menor en el que sólo se incluye el sistema.

Para instalar esta versión sólo es necesario ejecutar el instalador y seguir los pasos que allí se indican.

Esta versión del sistema es completamente funcional. El SGBD que usamos en esta versión es HSQL que es un SGBD desarrollado en Java y que no es necesario que funcione como servidor a parte para poder hacer uso del mismo.

Instalación <e-Aula> en versión desarrollo

Como puede observarse, la diferencia entre los requisitos para la versión de producción y la versión de desarrollo es que es necesario tener instalado Apache Ant para poder instalar esta versión.

Durante el desarrollo del proyecto hemos podido comprobar que Apache Ant es fundamental para la gestión de tareas repetitivas. Es recomendable que el desarrollador conozca de manera básica como se trabaja con un proyecto de Ant, al menos como se invoca la herramienta, y nociones básicas de los archivos de configuración "`build.properties`".

Esta versión se distribuye con los directorios que incluyen el código fuente y los proyectos de Apache Ant necesarios para generar la aplicación.

El proyecto que genera la aplicación WEB se encuentra en la carpeta `lms`. En esta carpeta podemos encontrar un archivo llamado "`build.properties`" que es necesario personalizar, ya que en esta versión del sistema los archivos están parametrizados para permitir instalar varias instancias de la aplicación usando incluso con distintas bases de datos para que sea posible el desarrollo concurrente de varios programadores.

Para mayor información a cerca del significado de los parámetros, consulte la documentación incluida dentro de los archivos de configuración y dentro de la definición del proyecto de Apache Ant

Archivo de definición de contexto de aplicación en Apache Tomcat

A continuación se explicará la información básica necesaria para configurar el archivo de contexto del sistema. El resto de parámetros de configuración se encuentran documentados dentro del propio archivo de definición del contexto.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!--
  Definición y configuración de la aplicación como contexto de Tomcat
-->
<Context  path="/@APPNAME@"
          docBase="@DIRBASE@"
          debug="0"
          reloadable="false"
          swallowOutput="true"
          useNaming="true">

  <Logger className="org.apache.catalina.logger.FileLogger"
          prefix="@APPNAME@."
          suffix=".txt"
          timestamp="true"
  />


  <!-- Definición del DataSource para conectarse a la BD E-Aula -->
  <Resource name="jdbc/@RECURSOBD@"
            auth="Container"
            type="javax.sql.DataSource" />

  <ResourceParams name="jdbc/@RECURSOBD@">

  <!--
  Factoría que crea el DataSource (la misma que trae Tomcat por
defecto)

  Esta factoría es la encargada crear el DataSource de JDBC.
  -->
  <parameter>
  <name>factory</name>
  <value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
  </parameter>


  <!-- ===== PARAMETROS DE CONEXION ===== -->


  <!-- Nombre de usuario y password para conectarse a la BD MySQL  -->
  <parameter>
    <name>username</name>
    <value>@DBUSER@</value>
  </parameter>
  <parameter>
    <name>password</name>
    <value>@DBPASS@</value>
  </parameter>

  <!-- Nombre de la clase del Driver JDBC de MySQL -->
  <parameter>
    <name>driverClassName</name>
    <value>@DBDRIVER@ </value>
  </parameter>
  <!-- URL de conexión de JDBC para conectarse a la BD MySQL.

  NOTA: el parámetro autoReconnect=true asegura que el Driver
reconectará automáticamente en el caso de que el servidor cierre la
conexión.   -->
  <parameter>
    <name>url</name>
    <value>jdbc:mysql://@DBHOST@/@DBNAME@?autoReconnect=true</value>
  </parameter>
...
...
```

@APPNAME@: Contexto en el que se definirá la aplicación, por ejemplo aulav, de manera que la aplicación podrá ser accedida por el navegador tecleando `http://localhost:8080/aulav`. Este nombre también será usado para generar el archivo de log que se almacenará en la carpeta `logs` del directorio de instalación de Apache Tomcat.

@DIRBASE@: Directorio en el que se encuentra la aplicación completa, HTML, JSP, WEB-INF\classes, etc.

@RECURSOBD@: Nombre del recurso que representa la BD dentro de la aplicación. Este nombre debe coincidir con el que se encuentra en el archivo `web.xml` que define la aplicación. El parámetro `jdbc/@RECURSOBD@` es usado dentro de la aplicación para acceder por JNDI a la base de datos.

@DBUSER@: Usuario del SGBD con privilegios suficientes para insertar, actualizar y eliminar sobre las tablas de la base de datos sobre la que trabaja la aplicación.

@DBPASS@: Password del usuario de base de datos.

@DBDRIVER@: Driver JDBC para acceder a la Base de datos. En caso de usar MySQL el driver es `com.mysql.jdbc.Driver`.

@DBHOST@: Nombre o IP de la máquina donde se encuentra el servidor de base de datos.

@DBNAME@: Nombre de la base de datos sobre la que trabajamos.

Configuración del archivo de preferencias

El archivo de preferencias (`preferencias.xml`) es el archivo que contiene la mayor parte de la configuración de la aplicación. Este archivo es un documento XML cuya dtd es `http://java.sun.com/dtd/preferences.dtd`, y sigue la estructura dictada por el paquete de preferencias (`java.util.prefs`) que viene con el JDK de Sun desde su versión 1.4. Al ser un archivo XML su estructura interna puede ser vista en forma de árbol en la que cada nodo contiene una serie de pares atributo-valor.

La lista de nodos utilizados y su significado dentro del archivo de configuración es la siguiente:

- `/paths`: Contiene los diferentes paths del sistema. Los atributos que se utilizan actualmente en la aplicación son:
    - `courses_private`: Directorio donde se ubicarán los recursos privados de un curso, este directorio debería estar protegido para que sólo se pueda acceder a él a través del sistema. Por defecto `/WEB-INF/repositorio`
    - `courses_public`: Directorio donde se ubicarán los recursos públicos de un sistema. Por defecto `/repo`. Para mejorar la eficiencia del sistema es recomendable que este directorio sea accesible desde el servidor HTTP.
    - `xsl`: Directorio donde se ubicarán las xsl. Por defecto `/WEB-INF/xsl`
    - `uploadDir`: Directorio base donde se subirán los archivos. También es el directorio donde se buscarán los PIF en el sistema de importación. Por defecto `/WEB-INF/upload`
    - `eaula_items`: Directorio donde se ubicarán los esqueletos para los ítems <e-Aula>: portada, faq y glosario. Por defecto `/WEB-INF/esqueleto`
    - `xsd`: Directorio donde se ubicarán las xsd con las que se validarán los manifiestos en el proceso de importación. Por defecto `/WEB-INF/xsd`.
- `/mail`: Configuración para el envío de correos electrónicos automáticos. Los atributos utilizados son:

  - `mail.smtp.host`: Servidor de correo saliente
  - `mail.user`: Nombre de usuario de la cuenta de correo.
  - `mail.password`: Password de la cuenta de correo
  - `mail.smtp.auth`: Indica si el servidor de correo saliente requiere autenticación. Por defecto "true".
  - `mail.smtp.from`: Dirección de correo que aparecerá en el campo `from` de los correos enviados por el sistema.
  - `mail.smtp.sendpartial`: Indica si se enviarán el resto de correos en una lista en la que falla uno de los envíos.

- `/es/ucm/sip/eaula/runtimeSystem/SignOnFilter`: Configuración del filtro que controla la autenticación en el sistema.
  - `order`: Indica el orden en el que se aplicarán las reglas del filtro. Por defecto `allow,deny`

- `/es/ucm/sip/eaula/runtimeSystem/SignOnFilter/rules/allow`: Reglas en las que se da permiso para entrar sin autentificación. Los pares atributo valor serán de la forma `key="nombreDeLaRegla"` (nombre para depuración de la regla) `value="patron"` donde patron es un patrón de URL que será aceptado por esta regla.

- `/es/ucm/sip/eaula/runtimeSystem/SignOnFilter/rules/deny`: Reglas en las que se deniega el permiso para entrar sin autentificación. Los pares atributo valor serán de la forma `key="nombreDeLaRegla"` (nombre para depuración de la regla) `value="patron"` donde patron es un patrón de URL que será denegado por esta regla.

- `/es/ucm/sip/eaula/runtimeSystem/SecurityFilter`: Configuración del filtro de control de acceso según los roles. El sistema es exactamente el mismo que para el `SignOnFilter`.

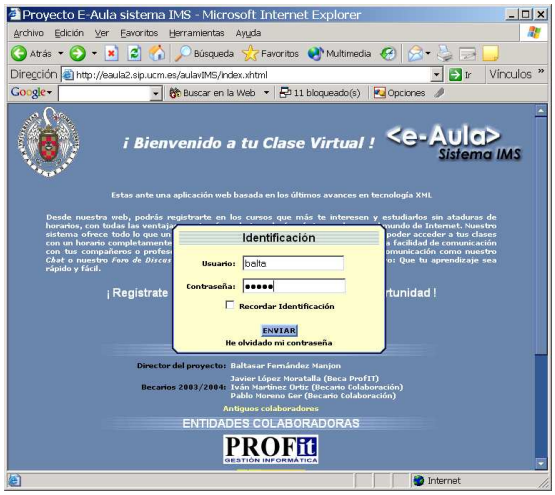
- `es/ucm/sip/eaula/runtimeSystem/validator/Controller`: Configuración de los schemas soportados directamente por el sistema de importación. En este nodo cada nombre de atributo se corresponderá con una URI de un `schema` XML y cada valor será el nombre del archivo xsd que contendrá la definición del `schema`. Si un Manifiesto utiliza algún `schema` que no esté dentro de esta lista deberá incluir el documento xsd dentro del propio PIF para que el sistema de importación pueda validarlo correctamente.

- `es/ucm/sip/eaula/runtimeSystem/ResourceProcessor/processors`: Contiene una lista de nodos en los que cada uno de ellos será el nodo de configuración para un resource processor en particular. En el caso del resource processor que viene por defecto los atributos válidos son:
  - `schema`: Indica la versión del schema IMS que soporta este resource processor
  - `version`: Versión del schema que soporta este processor
  - `class`: Clase que implementa este resource processor. Esta clase, además, definirá un path en el árbol del archivo de preferencias donde se encontrará la configuración relativa al propio resource processor. En esta configuración deberá figurar un atributo

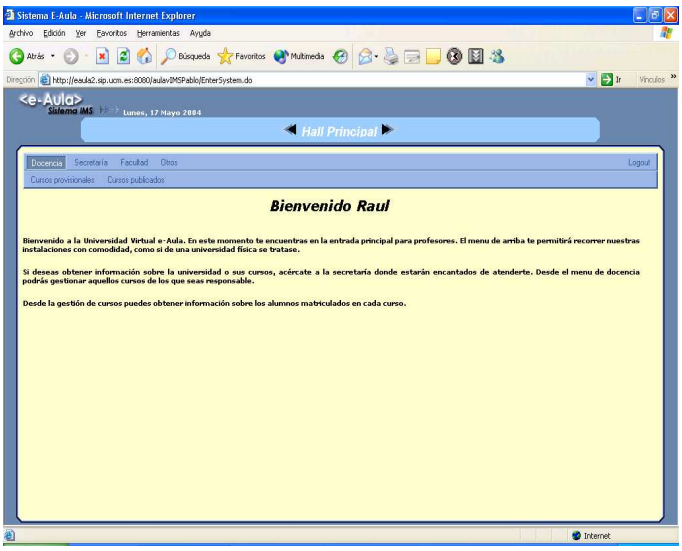
B.3.2.- Login

El acceso a la aplicación como administrador es similar al acceso como alumno desde la URL `http://eaula.sip.ucm.es/aulavIMS` o desde la página general de <e-aula> en el apartado prototipos (`http://eaula.sip.ucm.es/#prototipos`).

El administrador puede acceder a su área restringida pulsando en el botón "Entrar" de la página principal e introduciendo su nombre de usuario y contraseña.

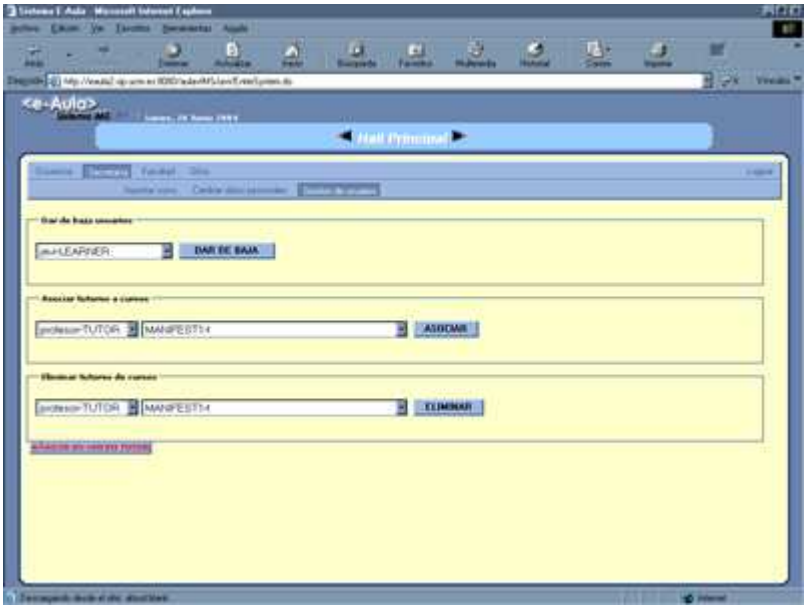


Pulsando el botón *enviar* entra en el sistema



B.3.4.- Gestión de usuarios y cursos

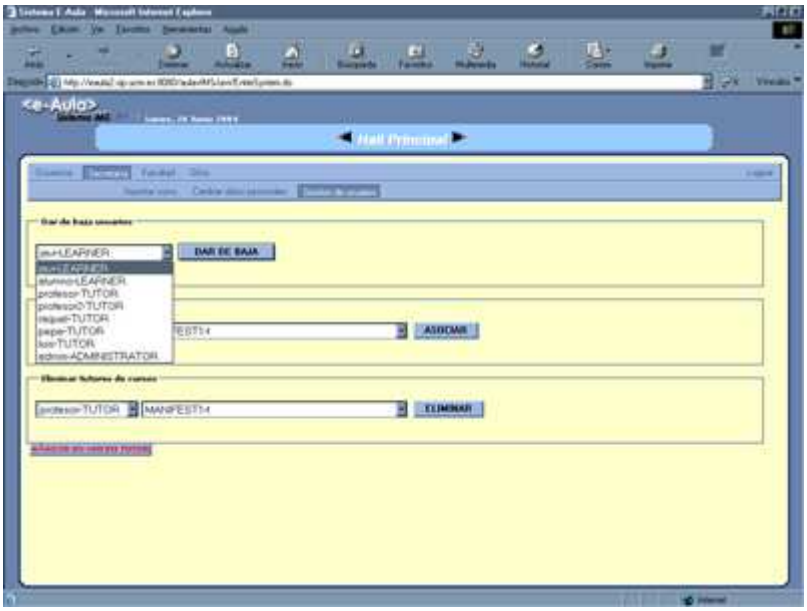
Desde el apartado Gestión de Usuarios del menú principal se llega a la pantalla principal de gestión de usuarios.



Desde esta pantalla se pueden realizar las siguientes acciones:

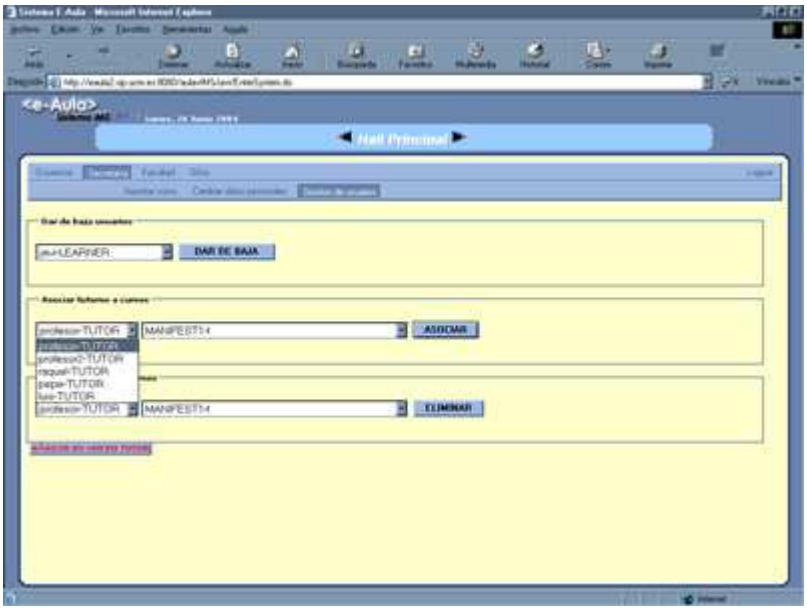
Dar de baja usuarios

Para dar de baja a un usuario (alumno o tutor) en el sistema no hay más que seleccionar el usuario en el panel "Dar de baja usuarios" y pulsar "Dar de baja".



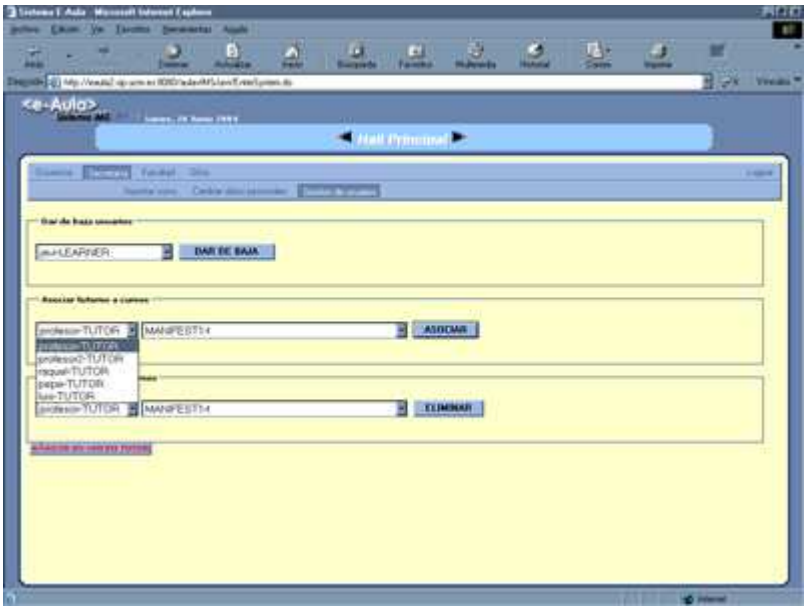
Asociar tutores a un curso

Para asociar un tutor a un curso (público o provisional) se deben seleccionar el tutor y el curso en el panel "Asociar tutores a cursos" y pulsar el botón "Asociar".



Eliminar tutores de un curso

Para eliminar un tutor de un curso (público o provisional) se deben seleccionar el tutor y el curso en el panel "Eliminar tutores de cursos" y pulsar el botón "Eliminar".

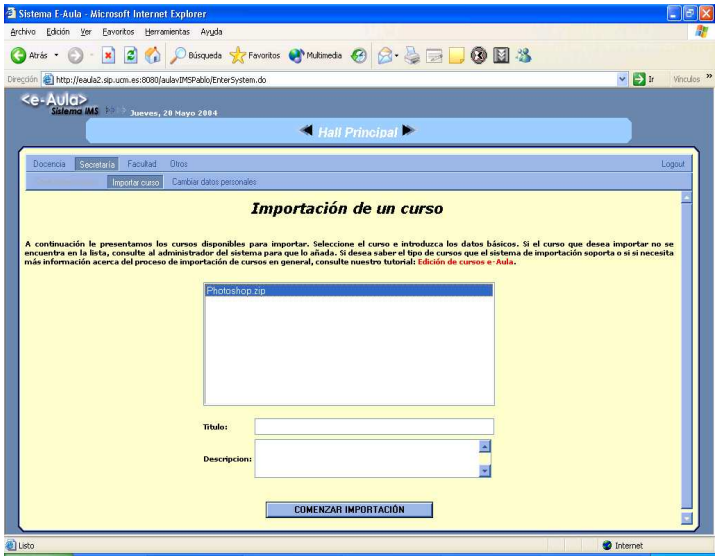


Añadir un tutor nuevo

Para añadir un tutor al sistema debemos pulsar sobre el enlace "Añadir un nuevo tutor", que nos llevará a la pantalla de matrícula de tutores. Allí deberán rellenarse los datos necesarios y pulsar enviar.

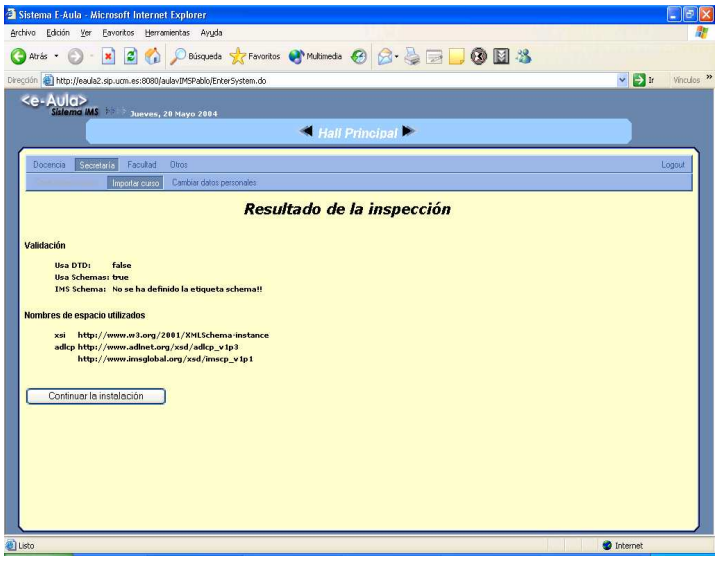
B.3.5.- Importación de cursos

En el apartado Importar, también en el menú de Secretaría, se muestra un listado de los posibles cursos a importar al sistema.

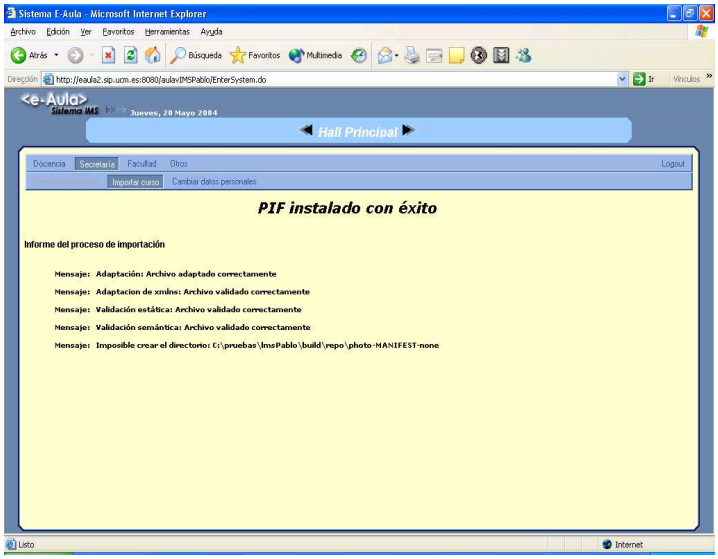


Seleccionando en esta página el curso que se desea importar al sistema e-Aula, rellenando la información complementaria (título del curso y descripción) y pulsando sobre *Comenzar importación* se inicia la importación del curso.

Si la importación se realiza de forma correcta se mostrará una pantalla con información de una primera inspección previa a la instalación del curso en el sistema



Y pulsando en *continuar instalación* llegaríamos a la siguiente página si no ocurre ningún problema durante la instalación del curso.



A partir de este momento el curso estaría apto para ser visualizado por los alumnos del sistema <e-Aula>.

B.3.6.- Edición de datos personales

Desde el submenú *Cambiar datos personales* de *Secretaría* se pueden modificar los datos personales del profesor.

Apéndice C: Ficheros de ejemplo

C.1.- Los tipos <e-Aula>

C.1.1.- Página

DTD

```
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by D (D) -->
<!ELEMENT pagina (meta, contenido)>
<!ATTLIST pagina
    id ID #REQUIRED
>
<!ELEMENT meta (titulo?, descripcion?, creador?, ultimaRevision?, palabrasClave?, comentario?)>
<!ELEMENT titulo (texto)>
<!ELEMENT creador (#PCDATA)>
<!ELEMENT ultimaRevision (#PCDATA)>
<!ELEMENT descripcion (texto)>
<!ELEMENT palabrasClave (#PCDATA)>
<!ELEMENT contenido (parrafo | lista | foto | fotosola | flash | texto | referencia | HTML | negrita |
comentario)*>
<!ELEMENT parrafo (meta?, contenido)>
<!ATTLIST parrafo
    nivel (alto | medio | bajo) #REQUIRED
>
<!ELEMENT lista (contenido+)>
<!ATTLIST lista
    ord (no | yes) #REQUIRED
>
<!ELEMENT foto (meta?)>
<!ATTLIST foto
    id CDATA #REQUIRED
    href CDATA #REQUIRED
    anchura CDATA #IMPLIED
    altura CDATA #IMPLIED
>
<!ELEMENT flash (meta?)>
<!ATTLIST flash
    id ID #REQUIRED
    href CDATA #REQUIRED
    anchura CDATA #IMPLIED
    altura CDATA #IMPLIED
>
<!ELEMENT fotosola EMPTY>
<!ATTLIST fotosola
    id CDATA #REQUIRED
    href CDATA #REQUIRED
    anchura CDATA #IMPLIED
    altura CDATA #IMPLIED
>
<!ELEMENT texto (#PCDATA | HTML | negrita | referencia)*>
<!ELEMENT HTML (#PCDATA)>
<!ELEMENT comentario (#PCDATA)>
<!ELEMENT refGlosario (#PCDATA)>
<!ATTLIST refGlosario
    id CDATA #REQUIRED
>
<!ELEMENT negrita (#PCDATA)>
```

Página <e-Aula> de ejemplo

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- <!DOCTYPE pagina SYSTEM "../dtd/pagina.dtd"> -->
<pagina id="UML-Tema0.0">
    <meta>
        <titulo>
            <texto>Introduccion</texto>
        </titulo>
        <creador>Iván Martínez Ortíz</creador>
        <ultimaRevision>Pablo Moreno Ger</ultimaRevision>
        <comentario><![CDATA[
Estado:
    <br>* Completo]]></comentario>
    </meta>
    <contenido>
        <parrafo nivel="bajo">
            <contenido>
                <texto>Antes de comenzar a estudiar UML es necesario conocer y entender los
conceptos relacionados con la Programación Orientada a Objetos, puesto que UML está
principalmente destinado a representar información sobre Modelos Orientados a Objetos. En este
capítulo introductorio trataremos los siguientes temas:</texto>
            </contenido>
        </parrafo>
        <parrafo nivel="bajo">
            <meta>
                <titulo>
                    <texto>Elementos de los Modelos Orientados a Objetos</texto>
                </titulo>
            </meta>
            <contenido>
                <texto>La mayor parte de los sistemas con Orientación a Objetos incluyen una
serie de conceptos que aportan gran parte de la potencia de dichos modelos. Estudiaremos varios
de estos elementos de los modelos Orientados a Objetos, algunos de los cuales son
fundamentales, puesto que ningún paradigma que no los incluya puede considerarse como un
paradigma Orientado a Objetos</texto>
            </contenido>
        </parrafo>
        <parrafo nivel="bajo">
            <meta>
                <titulo>
                    <texto>Clases y Objetos</texto>
                </titulo>
            </meta>
            <contenido>
                <texto>Las clases y los objetos son la base fundamental de un programa Orientado
a Objetos. Dedicaremos un tema completo al estudio de estos dos conceptos y la relación entre
ellos</texto>
            </contenido>
        </parrafo>
        <parrafo nivel="bajo">
            <meta>
                <titulo>
                    <texto>El Análisis Orientado a Objetos</texto>
                </titulo>
            </meta>
            <contenido>
                <texto>La mayoría de los procesos de desarrollo incluyen una fase de Análisis. En
los paradigmas Orientados a Obetos se puden aplicar una serie de técnicas de análisis específicas
para dichos paradigmas. Dedicaremos un tema al estudio de las técnicas de análisis orientadas a
Objetos</texto>
            </contenido>
        </parrafo>
        <parrafo nivel="bajo">
            <meta>
                <titulo>
                    <texto>El Diseño Orientado a Objetos</texto>
                </titulo>
            </meta>
            <contenido>
```

```
                <texto>Otro paso fundamental de los procesos de desarrollo suele ser el diseño, y, de nuevo, encontramos técnicas de diseño diractamente relacionadas con la Orientación a Objetos. Estudiaremos brevemente algunas de dichas técnicas.</texto>
            </contenido>
        </parrafo>
        <parrafo nivel="bajo">
            <meta>
                <titulo>
                    <texto>Problemas planteados</texto>
                </titulo>
            </meta>
            <contenido>
                <texto>Para aplicar con efectividad los elementos del modelo de objetos, es necesario entender varios conceptos:</texto>
                <lista ord="no">
                    <contenido>
                        <texto>¿Qué son las clases y los objetos? Trataremos de dar respuesta a esa pregunta en el primer tema.</texto>
                    </contenido>
                    <contenido>
                        <texto>¿Cómo se identifican correctamente las clases y objetos relevantes de una aplicación concreta? Trataremos de resolver esa pregunta estudiando técnicas de análsis y diseño orientadas a objetos</texto>
                    </contenido>
                    <contenido>
                        <texto>¿Cómo sería una notación adecuada para expresar el diseño de un sistema OO? Esta es la clave fundamental de este curso, puesto que lo que UML ofrece es una notación universal para representar sistemas Orientados a Objetos. Esta notación nos permite comprender cómo será  nuestro sistema tanto estática como dinámicamente, por lo que nos será útil tanto en la fase de diseño como en la fase de implementación.</texto>
                    </contenido>
                    <contenido>
                        <texto>¿Qué proceso puede conducir a un sistema OO bien estructurado? Existen varios procesos en los cuales no nos vamos a centrar. Uno de los más conocidos (y muy relacionado con UML) es el Proceso Unificado.</texto>
                    </contenido>
                </lista>
            </contenido>
        </parrafo>
    </contenido>

    </pagina>
```

C.1.2.- Glosario

DTD

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

    Definicion del glosario

-->
<!ELEMENT glosario (entrada*)>
<!ELEMENT entrada (termino, definicion)>
<!ATTLIST entrada
    id ID #REQUIRED
>
<!ELEMENT termino (#PCDATA)>
<!ELEMENT definicion (#PCDATA | ref)*>
<!ELEMENT ref (#PCDATA)>
<!ATTLIST ref
    target CDATA #REQUIRED

>
```

Glosario <e-Aula> de ejemplo

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--<!DOCTYPE glosario SYSTEM "glosario.dtd">-->
<?xml-stylesheet type="text/xsl" href="glosario.xsl"?>
<glosario>
    <entrada id="id32">
        <termino>Abstracción</termino>
        <definicion>Descripción simplificada que resalta los detalles más
        importantes de un objeto.</definicion>
    </entrada>
    <entrada id="id19">
        <termino>Accesibilidad</termino>
        <definicion>Capacidad de una abstracción para ver otra y hacer
        referencia a recursos en su vista externa.</definicion>
    </entrada>
    <entrada id="id37">
        <termino>Acción</termino>
        <definicion>Computación <ref target="id38">atómica</ref> ejecutable.</definicion>
    </entrada>
    <entrada id="id12">
        <termino>Acoplamiento</termino>
        <definicion>Es la medida de la interconexión entre módulos de un sistema.</definicion>
    </entrada>
    <entrada id="id113">
        <termino>Transición de terminación</termino>
        <definicion>
            <ref target="id112">Transición</ref> que no posee
        <ref target="id68">evento disparador</ref>.</definicion>
    </entrada>
    <entrada id="id114">
        <termino>Valores etiquetados</termino>
        <definicion>Se utlizan para añadir propiedades nuevas a los elementos ya
        existentes.</definicion>
    </entrada>
    <entrada id="id115">
        <termino>Visibilidad</termino>
        <definicion>Una clase es visible a otra si la segunda puede pasar mensajes
        a la primera.</definicion>
    </entrada>
</glosario>
```

C.1.3.- FAQ

DTD

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Definicion de una faq -->
<!ENTITY % EGeneral SYSTEM  "general.dtd">
<!ELEMENT faq (seccion*)>
<!ELEMENT seccion (item*)>
<!ATTLIST seccion
    titulo CDATA #REQUIRED
>
<!ELEMENT item (pregunta, respuesta)>
<!ELEMENT pregunta (#PCDATA)>
<!ELEMENT respuesta (#PCDATA)>
```

FAQ <e-Aula> de ejemplo

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<faq>
    <seccion titulo="Acerca de UML en general">
        <item>
            <pregunta>¿Qué es UML?</pregunta>
            <respuesta>El Lenguaje Unificado de Desarrollo (UML) es un lenguaje estándar de
desarrollo utilizado para visualizar,
            construir y documentar la arquitectura de un sistema software. Con este objetivo, UML
preescribe un conjuento de
            notaciones y diagrmas estándar para modelar sistemas orientados a
objetos.</respuesta>
        </item>
        <item>
            <pregunta>¿Realmente es necesario el uso de UML? ¿No puedo describir el diseño de
mi sistema simplemente
            utilizando el lenguaje natural?</pregunta>
            <respuesta>Aunque es posible describir la arquitectura de un sistema utilizando el
lenguaje natural, es mucho más
            intuitivo y fácil de comprender si se hace utilizando diagramas que expresan las
relaciones existentes entre
            los distintos elementos del sistema.</respuesta>
        </item>
    </seccion>
    <!-- ********************************************************************* -->
    <seccion titulo="Acerca de la Introducción a UML">
        <item>
            <pregunta>¿Hay alguna diferencia entre los términos clase y objeto?</pregunta>
            <respuesta>Un objeto es una entidad individual e identificable con un papel bien
definido en el dominio de nuestro
            problema, mientras que una clase es una descripción de un conjuento de objetos que
comparten el mismo
            tipo de atributos y operaciones.</respuesta>
        </item>
    </seccion>
</faq>
```

C.1.4.- Portada

DTD

```
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by D (D) -->
<!ELEMENT infoCurso (meta, contenido)>
<!ATTLIST infoCurso
    id ID #REQUIRED
>
<!ELEMENT meta (titulo, creadores?)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT creadores (autor)*>
<!ELEMENT contenido (presentacion, noticias,  recursos)>
<!ELEMENT presentacion (texto)*>
<!ELEMENT noticias (noticia)*>
<!ELEMENT noticia (texto)*>
<!ATTLIST noticia
    fecha CDATA #REQUIRED
>
<!ELEMENT recursos (libro | recursoWeb | articulo)*>
<!ELEMENT libro (titulo, autores, anio?, editorial?, imagen?, comentario?, enlace*)>
<!ELEMENT anio (#PCDATA)>
<!ELEMENT editorial (#PCDATA)>
<!ELEMENT autor (nombre, datos?)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT datos (#PCDATA)>
<!ELEMENT comentario (texto)*>
```

```
<!ELEMENT imagen (#PCDATA)>
<!ATTLIST imagen
    src CDATA #REQUIRED
>
<!ELEMENT enlace (#PCDATA)>
<!ATTLIST enlace
    URL CDATA #REQUIRED
>
<!ELEMENT recursoWeb (descripcion)>
<!ATTLIST recursoWeb
    URL CDATA #REQUIRED
    nombre CDATA #REQUIRED
>
<!ELEMENT articulo (titulo, autores, publicacion)>
<!ELEMENT publicacion (#PCDATA)>
<!ELEMENT descripcion (#PCDATA)>
<!ELEMENT autores (#PCDATA)>
<!ELEMENT texto (#PCDATA | HTML)*>
<!ELEMENT HTML (#PCDATA)>
```

Portada <e-Aula> de ejemplo

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<infoCurso id="infoCursoUML">
    <meta>
        <titulo>UML APLICADO A LA INGENIERIA DEL SOFTWARE</titulo>
        <creadores>
            <autor>
                <nombre>Pablo Moreno Ger</nombre>
                <datos>Capitulo 3; Presentación; Revisión Global; Glosario; FAQs</datos>
            </autor>
            <autor>
                <nombre>Ivan Martínez Ortiz</nombre>
                <datos>Capitulos 0 y 5; Glosario; Captura de imágenes</datos>
            </autor>
            <autor>
                <nombre>Javier López Moratalla</nombre>
                <datos>Capitulos 1, 2 y 4; Captura de imágenes</datos>
            </autor>
            <autor>
                <nombre>Manuel Durán</nombre>
                <datos>Revisión Ortográfica; Captura de imágenes; Tests de
Autoevaluación</datos>
            </autor>
            <autor>
                <nombre>Baltasar Fernández Manjón</nombre>
                <datos>Revisor Técnico</datos>
            </autor>
            <autor>
                <nombre>Antonio Navarro Martín</nombre>
                <datos>Revisor Técnico, cesión de sus apuntes docentes para el desarrollo del
curso.</datos>
            </autor>
        </creadores>
    </meta>
    <contenido>
        <presentacion>
            <texto>El curso trata sobre la aplicación de UML en la Ingeniería del Software. UML es
un lenguaje para construir, visualizar y documentar sistemas con gran cantidad de software, de
manera que proporciona a todas las personas implicadas en el desarrollo y mantenimiento del
sistema, una notación estándar para expresar el diseño del sistema</texto>
            <texto>La única función de este parrafo es demostrar que se pueden escribir
presentaciones largas.</texto>
        </presentacion>
        <noticias>
            <noticia fecha="10 de Julio de 2003">
                <texto>El curso está prácticamente completado a nivel de temario. En los próximos
días añadiremos los siguientes elementos:</texto>
                <texto>* Un glosario</texto>
```

```
                <texto>* Una tabla de FAQs</texto>
                <texto>* Cuestionarios de autoevaluación.</texto>
            </noticia>
            <noticia fecha="20 de Junio de 2003">
                <texto>Comenzamos una última revisión de contenido. La dedicación principal será
obtener capturas de pantalla para el curso.</texto>
            </noticia>
            <noticia fecha="Julio 2002">
                <texto>El curso comienza su andadura. Comenzamos con la recopilación de
información.</texto>
            </noticia>
        </noticias>
        <recursos>
            <libro>
                <titulo>Utilización de UML en Ingeniería del Software con Objetos y
Componentes</titulo>
                <autores>Perdita Stevens, Robert Pooley</autores>
                <anio>2002</anio>
                <editorial>Addison Wesley.</editorial>
                <imagen src="UsingUML.jpg"/>
                <comentario>
                    <texto>Un libro mu güeno.</texto>
                    <texto>En la web del libro pueden encontrarse numerosas referencias a
enlaces de interés tanto de UML como otros lenguajes de modelado.</texto>
                </comentario>
                <enlace URL="http://www.addisonwesley.com">Web de la editorial</enlace>
                <enlace URL="http://www.dcs.ed.ac.uk/home/pxs/Book">Web del libro</enlace>
                <enlace URL="http://www.amazon.com/exec/obidos/ASIN/0201648601/twilightstudi-
20">
Comprar en Amazon.com (version en inglés)</enlace>
            </libro>
            <libro>
                <titulo>El Lenguaje Unificado de Modelado</titulo>
                <autores>Grady Booch, Ivar Jacobson y James Rumbaugh</autores>
                <anio>2000</anio>
                <editorial>Addison Wesley</editorial>
            </libro>
            <libro>
                <titulo>Análisis y Diseño Orientado a Objetos</titulo>
                <autores>Grady Booch</autores>
                <anio>1996</anio>
                <editorial>Addison Wesley</editorial>
            </libro>
            <libro>
                <titulo>Design Patterns, Elements of Reusable Object-Oriented Software</titulo>
                <autores>Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides,
John</autores>
                <anio>1995</anio>
                <editorial>Addison Wesley</editorial>
            </libro>
            <recursoWeb URL="http://www.omg.org/uml/" nombre="Página Oficial de UML">
                <descripcion>En esta página, como cabía esperar, se pueden encontrar numerosa
documentación de UML, la última especificación formal y los nuevos proyectos.</descripcion>
            </recursoWeb>
            <recursoWeb URL="http://www.extremeprogramming.org/" nombre="Página Oficial de
XP">
                <descripcion>Esta página es una de las principales referencias en cuanto a
Extreme Programming se refiere.</descripcion>
            </recursoWeb>
            <articulo>
                <titulo>Articulo de UML</titulo>
                <autores>Pablo Moreno Ger</autores>
                <publicacion>Revista de Ejemplo, Nº173. (Mayo 2003)</publicacion>
            </articulo>
        </recursos>
    </contenido>

    </infoCurso>
```

C.2.- Código fuente de las interfaces de la Librería de Acceso a IMS

C.2.1.- Manifest.java

```
/*
  Copyright  © 2.003 Universidad Complutense de Madrid, Proyecto E-Aula
*/

package es.ucm.sip.eaula.cp;

import org.w3c.dom.Element;
import java.net.URI;
import java.net.URISyntaxException;
import java.util.List;

/**
 * Manifiesto que define el el paquete de intercambio en IMS.
 */
public interface Manifest {

       /**
     * Devuelve el identificador del manifiesto
     */
    public Identifier getIdentifier();


       /**
     * Establece el <code>Identifier</code> de la <code>Manifest</code>
     *
     * @param identifier Nuevo <code>Identifier</code> de la
<code>Manifest</code>.
     *
     * @throws NullPointerException Se lanza si <code>identifier</code>
es <code>null</code>.
     * @throws IllegalArgumentException Se lanza en el caso de que
<code>identifier</code> sea
     * un <code>Identifier</code> que ya existe en el
<code>Manifest</code> y es distinto al
     * <code>Identifier</code> actual del <code>Manifest</code>.
     */
    public void setIdentifier(Identifier identifier);

    /**
     * Verifica si el <code>Manifest</code> es un subManifiesto.
     *
     * @return Devuelve <code>true</code> en caso de que el
<code>Manifest</code>
     * sea un subManifiesto, o <code>false</code> en caso contrario.
     */
     public boolean isSubManifest();

       /**
     * Devuelve la organización por defecto o <code>null</code> si el
     * <code>Manifest</code> no contiene ninguna organización.
     *
     * @return Devuelve la organización por defecto si existe.
     */
    public Organization getDefaultOrganization();

       /**
     * Devuelve el <code>Identifier</code> de la organización por
defecto o
     * <code>null</code> si el <code>Manifest</code> no contiene
ninguna
     * organización.
     *
     * @return Devuelve el <code>Identifier</code> la organización por
defecto
     * si existe.
     */
```

```
    public Identifier getDefaultOrganizationId();

    /**
     * Establece la <code>Organization</code> con
<code>Identifier</code> <code>identifier</code>
     * como <code>Organization</code> por defecto.
     *
     * @param identifier <code>Identifier</code> de la nueva
<code>Organization</code> por defecto
     *
     * @throws NullPointerException Se lanza si <code>identifier</code>
es <code>null</code>.
     * @throws IllegalArgumentException Se lanza si
<code>identifier</code> no es el <code>Identifier</code>
     * del <code>Manifest</code> o no es el <code>Identifier</code> de
una <code>Organization</code>.
     */
    public void setDefaultOrganizationId(Identifier identifier );

        /**
         * Devuelve los metadatos del <code>Manifes</code>
         */
    public Metadata getMetadata();

        /**
         * Devuelve el &lt;schema&gt; del <code>Manifest</code>.
         * <p>Devuelve el &lt;schema&gt; del <code>Manifest</code> o
<code>null</code>
         * en caso de que no exista</p>
         */
    public String getSchema();

        /**
         * Devuelve el &lt;schemaversoin&gt; del <code>Manifest</code>.
         * <p>Devuelve el &lt;schemaversion&gt; del
<code>Manifest</code> o <code>null</code>
         * en caso de que no exista</p>
         */
    public String getSchemaVersion();

        /**
         * Establece el &lt;schema&gt; del <code>Manifiest</code>
         * <p>Establece el &lt;schema&gt; del <code>Manifiest</code>,
         * si <code>schema</code> es <code>null</code> o la cadena
         * vacía la etiqueta <code>schema</code> sera eliminada.</p>
         *
         * @param schema Nuevo &lt;schema&gt; del
<code>Manifiest</code>
         */
    public void setSchema(String schema);

        /**
         * Establece el &lt;schemaversion&gt; del
<code>Manifiest</code>
         * <p>Establece el &lt;schemaversion&gt; del
<code>Manifiest</code>,
         * si <code>schemaVersion</code> es <code>null</code> o la
cadena
         * vacía la etiqueta <code>schema</code> sera eliminada.</p>
         *
         * @param schemaVersion Nuevo &lt;schemaversion&gt; del
<code>Manifiest</code>
         */
    public void setSchemaVersion(String schemaVersion);


        /**
         * Devuelve una <code>URI</code> con el el atributo "xml:base"
de la etiqueta
     * &lt;manifest&gt;.
         * <p>Si el atributo "xml:base" es relativo, se intentará
resolver de la
     * siguiente manera:</p>
         * <ol>
```

```
     * <li>En el caso de que la etiqueta &lt;manifest&gt; esté dentro
de otra
     * etiqueta &lt;manifest&gt;, es decir, que sea un submanifiesto;
si su
     * atributo "xml:base" es relativo, será resuelto con el atributo
"xml:base"
     * del la etiqueta &lt;manifest&gt; que lo contiene.</li>
     * <li>En el caso de que se trate de un submanifiesto y no se haya
especificado
     * el atributo "xml:base" se heredará el valor de la etiqueta
&lt;manifest&gt;
     * que contiene este submanifiesto.</li>
     * <li>Si la etiqueta &lt;manifest&gt; no tiene el atributo
"xml:base" y no
     * es un submanifesto se devolverá <code>null</code>.</li>
     * <ol>
     */
    URI getXMLBase() throws URISyntaxException;

    /**
     * Establece el atributo "xml:base" del <code>Manifest</code>
     * <p>Si <code>xmlBase</code> es <code>null</code> se eliminará
     * el atributo "xml:base"</p>
     *
     * @param xmlBase Nuevo valor del atributo "xml:base"
     */
    void setXMLBase(URI xmlBase);


       /**
        * Devuelve una <code>URI</code> con el el atributo "xml:base"
de la etiqueta
     * &lt;resources&gt;.
       * <p>Si el atributo "xml:base" es relativo, se intentará
resolver de la
     * a través del atributo "xml:base" de la etiqueta &lt;manifest&gt;
que la
     * contiene.</p>
     * <p>Si la etiqueta &lt;resources&gt; se devolvera el valor del
del
     * atributo "xml:base" de la etiqueta &lt;manifest&gt; que la
     * contiene. </p>
     */
    public URI getResourcesXMLBase() throws URISyntaxException;

    /**
     * Establece el atributo "xml:base" de la etiqueta
&lt;resources&gt;.
     * <p>Si <code>xmlBase</code> es <code>null</code> se eliminará
     * el atributo "xml:base"</p>
     *
     * @param xmlBase Nuevo valor del atributo "xml:base"
     */
    void setResourcesXMLBase(URI xmlBase);

       /**
     * Devuelve las <code>Organization</code> que haya en el
<code>Manifest</code>
     */
    public List getOrganizations();

    /**
     * Devuelve el número de <code>Organization</code> que tiene.
     *
     * @return Devuelve un número >= 0.
     */
    public int organizationsCount();

    /**
     * Devuelve la <i>index-ésima</i> <code>Organization</code> del
<code>Manifest</code>.
     * <code>index</code> será el índice con el que accederemos, y su
valor debe moverse
     * entre 0 y {@link #organizationsCount()} -1
     *
```

```
     * @param index Índice que ocupa la <code>Organization</code> que
deseamos obtener.
     *
     * @return Devuelve la <i>index-ésima</i> <code>Organization</code>
del <code>Manifest</code>
     * o <code>null</code> en caso de que no exista.
     *
     * @throws IndexOutOfBoundsException Se lanza si index no es >= 0 y
<= {@link #organizationsCount()} -1
     */
    public Organization getOrganization(int index);

    /**
     * Devuelve la <code>Organization</code> del <code>Manifest</code>
cuyo
     * <code>Identifier</code> coincida con <code>identifier</code>.
     * <p>Devuelve la <code>Organization</code> del
<code>Manifest</code> cuyo
     * <code>Identifier</code> coincide con <code>identifier</code> o
     * <code>null</code> en el caso de que no exista.</p>
     *
     * @param identifier <code>Identifier</code> de la
<code>Organization</code>
     * que deseamos obtener.
     *
     * @return Devuelve la <code>Organization</code> del
<code>Manifest</code>
     * o <code>null</code> en caso de que no exista.
     *
     */
    public Organization getOrganization(Identifier identifier);

    /**
     * Devuelve los <code>Resource</code>s que se definene en el
<code>Manifest</code>.
     */
    public List getResources();

       /**
        *
        * @since 1.2
        */
       public List getResources(ContentType resType);

    /**
     * Devuelve el número de <code>Resource</code>s que se definen en
el <code>Manifest</code>
     *
     * @return Devuelve un número >= 0
     */
    public int resourcesCount();

    /**
     * Devuelve el <i>index-ésimo</i> <code>Resource</code> del
<code>Manifest</code>.
     * <code>index</code> será el índice con el que accederemos, y su
valor debe moverse
     * entre 0 y {@link #resourcesCount()} -1
     *
     * @param index Índice que ocupa el <code>Resource</code> que
deseamos obtener.
     *
     * @return Devuelve el <i>index-ésimo</i> <code>Resource</code> del
<code>Manifest</code>
     *
     * @throws IndexOutOfBoundsException Se lanza si index no es >= 0 y
<= {@link #resourcesCount()} -1
     */
    public Resource getResource(int index);

    /**
     *
     */
       public Resource getResource(Identifier identifier);
```

```
        /**
         * Devuelve todos los (sub)<code>Manifest</code> de este
<code>Manifest</code>.
         *
         */
    public List getSubManifests();

        /**
         * Devuelve el número de (sub)<code>Manifest</code> de este
<code>Manifest</code>.
         */
    public int subManifestsCount();

        /**
         * Devuelve el número de (sub)<code>Manifest</code> de este
<code>Manifest</code>.
         *
     * @param index Índice que ocupa el (sub)<code>Manifest</code> que
deseamos obtener.
     *
     * @return Devuelve el <i>index-ésimo</i>
(sub)<code>Manifest</code> de este <code>Manifest</code>
     *
     * @throws IndexOutOfBoundsException Se lanza si index no es >= 0 y
<= {@link #subManifestsCount()} -1
         */
    public Manifest getSubManifest(int index);

        /**
         * Devuelve el (sub)<code>Manifest</code> de este
<code>Manifest</code> cuyo
     * <code>Identifier</code> coincide con <code>identifier</code>.
     * <p>Devuelve el (sub)<code>Manifest</code> de este
<code>Manifest</code> cuyo
     * <code>Identifier</code> coincide con <code>identifier</code> o
     * <code>null</code> en el caso de que no exista.</p>
     *
     * @param identifier <code>Identifier</code> del
(sub)<code>Manifest</code>.
         */
    public Manifest getSubManifest(Identifier identifier);

        /**
         * Añade un (sub)<code>Manifest</code> a este
<code>Manifest</code>.
         *
         * @param manifest <code>Manifest</code> a añadir.
         */
    public void aggregateManifest(Manifest manifest);

        /**
         *
         */
    public Manifest disaggregateManifest(int index);

        /**
         *
         */
    public List disaggregateManifest();


    public void addOrganization(Organization organization);

    public void addOrganization(int index, Organization organization);

    public boolean removeOrganization(Organization organization);

    public Organization removeOrganization(int index);


    public void addResource(Resource resource);

    public void addResource(int index, Resource resource);
```

```
    public boolean removeResource(Resource resource);

    public Resource removeResource(int index);

        /**
         * Crea un nuevo <code>Item</code>
         */
    public Item createItem();

        /**
         * Crea un nuevo <code>Resource</code>
         */
    public Resource createResource();

    /**
     * Crea una nueva <code>Organization</code>
     */
    public Organization createOrganization();

}
```

C.2.2.- Organization.java

```
/*
  Copyright  © 2.003 Universidad Complutense de Madrid, Proyecto E-Aula
*/

package es.ucm.sip.eaula.cp;

import org.w3c.dom.Element;
import java.util.List;
import es.ucm.sip.eaula.cp.ContentType;

/**
 * Interfaz que representa una &lt;organization&gt; de un manifiesto.
 */
public interface Organization {

       /**
     * Devuelve el <code>Identfier</code> de la
<code>Organization</code>
     */
    public Identifier getIdentifier();

       /**
     * Devuelve el título de la <code>Organization</code>
     */
    public String getTitle();

    /**
     * Devuelve los <code>Item</code>s que son hijos directos de la
<code>Organization</code>.
     */
    public List getItems();

       /**
        * Devuelve los <code>Item</code> que referencian un
<code>Resource</code> de tipo
        * <code>type</code>.
        *
        * @param type <code>ContentType</code> del
<code>Resource</code> que referencia
        * los <code>Item</code> a devolver.
        *
        * @since 1.2
        */
       public List getItems(ContentType type);

       /**
     * Devuelve el número de <code>Item</code>s que son hijos directos
de la <code>Organization</code>.
```

```
     */
    public int itemsCount();

       /**
     * Devuelve el <code>index-esimo</code> <code>Item</code> de los
hijos directos de la
     * <code>Organization</code>.
     *
     * @param index Posición del <code>Item</code> a devolver.
     *
     * @throws IndexOutOfBoundsException Se lanza en el caso de que
index sea < 0 o
     * sea >= {@link #itemsCount() }
     */
    public Item getItem(int index);

       /**
     * Devuelve el <code>Item</code> con identificador
<code>identifier</code>
     * <p>Devuelve el <code>Item</code> con identificador
<code>identifier</code> y
     * que es descenciente (de cualquier nivel), de esta
<code>Organization</code>.
     * En otro caso se devolverá <code>null</code>.</p>
     *
     * @param identifier <code>Identifier</code>
     *
     * @throws NullPointerException Se lanza si <code>identifier</code>
es <code>null</code>
     */
    public Item getItem(Identifier identifier);

       /**
     * Establece el título de la <code>Organization</code>.
     * <p>Establece el valor de &lt;title&gt; que es usado como título
de la
     * <code>Organization</code>. Si el parámetro <code>title</code> es
la cadena
     * vacía o <code>null</code>, se eliminará la etiqueta
&lt;title&gt;.</p>
     *
     * @param title Nuevo titulo de la <code>Organization</code>
     */
    public void setTitle(String title);

       /**
     * Establece el <code>Identifier</code> de la
<code>Organization</code>
     *
     * @param identifier Nuevo <code>Identifier</code> de la
<code>Organization</code>.
     *
     * @throws NullPointerException Se lanza si <code>identifier</code>
es <code>null</code>.
     * @throws IllegalArgumentException Se lanza en el caso de que
<code>identifier</code> sea
     * un <code>Identifier</code> que ya existe en el
<code>Manifest</code> y es distinto al
     * <code>Identifier</code> actual de la <code>Organization</code>.
     */
    public void setIdentifier(Identifier identifier);


       /**
        * Devuelve los metadatos del <code>Organization</code>
        */
    public Metadata getMetadata();

       /**
     * Añade <code>item</code> al final de los <code>Item</code>'s de
la <code>Organization</code>.
     *
     * @param item <code>Item</code> que queremos añadir.
     *
```

```
        * @throw IllegalStateException Se lazan en el caso de que
<code>item</code>
        * no sea un <code>Item</code> nuevo; es decir, que no haya
sido añadido
        * al <code>Manifest</code> y no haya sido obtenido a través de
los métodos
        * getXXX(...);
        *
        * @throws IllegalArgumentException Se lanza en el caso de que
<code>item</code> no
        * se añada a un <code>Organization</code> que esté en un
<code>Manifest</code> y éste
        * sea distinto al que ha creado a <code>item</code>.
     */
    public void addItem(Item item);

       /**
     * Añade <code>item</code> en la <code>index-ésima</code> posicion.
     * <p>Añade <code>item</code> en la <code>index-ésima</code>
posicion de los
     * <code>Item</code> que son hijos directos de la
<code>Organization</code>.</p>
     * <p><code>index</code> tiene que ser >= 0  ( el 0 indica que se
quiere insertar
     * antes del primer hijo) y <= el valor devuelto por {@link
#itemsCount()} (un
     * valor igual al devuelto por {@link #itemsCount()} indica que se
quiere añadir
     * al final).</p>
     *
     * @param index Posición en la que queremos añadir el nuevo
<code>Item</code>
     * @param item <code>Item</code> que queremos añadir
     *
     * @throws IndexOutOfBoundsException Se lanza si <code>index</code>
es < 0 ó si
     *  es > {@link #itemsCount()}.
     *
        * @throw IllegalStateException Se lazan en el caso de que
<code>item</code>
        * no sea un <code>Item</code> nuevo; es decir, que no haya
sido añadido
        * al <code>Manifest</code> y no haya sido obtenido a través de
los métodos
        * getXXX(...);
        *
        * @throws IllegalArgumentException Se lanza en el caso de que
<code>item</code> no
        * se añada a un <code>Organization</code> que esté en un
<code>Manifest</code> y éste
        * sea distinto al que ha creado a <code>item</code>.
     */
    public void addItem(int index, Item item);

       /**
     * Elimina <code>item</code> de la <code>Organization</code>.
     * <p>Elimina <code>item</code> tiene que ser hijo directo de la
<code>Organization</code>.</p>
     *
     * @param item <code>Item</code> que queremos eliminar.
       *
     * @return Devuelve <code>true</code> en el caso de que se haya
elminado el <code>Item</code>
     * y <code>false</code> en otro caso.
     */
    public boolean removeItem(Item item);

       /**
     * Elimina el  <code>index-ésimo</code> <code>Item</code>.
        * <p>Elmina el <code>Item</code> que es hijo directo de la
<code>Organization</code>
     * y está en la <code>index-ésima</code> posición.</p>
     *
       *
     * @return Devuelve el <code>Item</code> que ha sido eliminado.
```

```
     *
     * @throws IndexOutOfBoundsException Se lanza en el caso de que
index sea < 0 o
     * sea >= que el valor devuelto por {@link #itemsCount()}.
     */
    public Item removeItem(int index);
}
```

C.2.3.- Item.java

```
/*
  Copyright  © 2.003 Universidad Complutense de Madrid, Proyecto E-Aula
*/

package es.ucm.sip.eaula.cp;

import org.w3c.dom.Element;
import java.util.List;

/**
 * Representa un <code>Item</code> de la especificación de IMS Content
Packaging.
 *
 * <p><a name="identifierref"><strong>Atributo
identifierref</strong></a></p>
 * <p>El <code>Identifier</code> que devuelve la funcion {@link
#getIdentifierref()}
 * puede referenciar alguno de los siguientes elementos:</p>
 * <ul>
 *             <li>{@link es.ucm.sip.eaula.cp.Resource}</li>
 *             <li>{@link es.ucm.sip.eaula.cp.Manifest}</li>
 *             <li>{@link es.ucm.sip.eaula.cp.Organization}</li>
 *             <li>{@link es.ucm.sip.eaula.cp.Item}</li>
 * </ul>
 * <table>
 * <caption>Tratamiento de los distintos valores de
"identifierref"</caption>
 * <thead>
 *  <tr>
 *    <th>Elemento referenciado</th>
 *    <th>Acción</th>
 *  </tr>
 * </thead>
 * <tbody>
 *  <tr>
 *             <td><code>Manifest</code></td>
 *             <td>
 *                     <p>Se añadiran los <code>Item</code> de su
<code>Organization</code>
 *                     por defecto.</p>
 *                     <p>Si no existe organización por defecto se
referenciará el primer
 *                     <code>Resource</code> que existe y si no existe
ninguno estaremos en el
 *                     caso de la referencia nula (<i> vea sección 4.1
del IMS Content Packaging
 *                     Information Model v1.1.3</i>)</p>
 *             </td>
 *  </tr>
 *  <tr>
 *             <td><code>Item</code></td>
 *             <td>
 *                     <p>Los <code>Item</code> que son hijos directos
del <code>Item</code>
 *                     referenciado, serán considerados como hijos
directos del <code>Item</code>
 *                     que referencia, estando estos nuevos hijos
"virtuales" tras los hijos
 *                     directos del <code>Item</code> que hace la
referencia.</p>
 *             </td>
```

```
 *   </tr>
 *  <tr>
 *             <td><code>Organization</code></td>
 *             <td>
 *             </td>
 *   </tr>
 *  <tr>
 *             <td><code>Resource</code></td>
 *             <td>
 *             </td>
 *   </tr>
 * </tbody>
 * </table>
 *
 * <p><strong><a name="itemsVirtuales">Items
"virtuales"</a></strong></p>
 */
public interface Item {

       /**
     * Devuelve el título del <code>Item</code>.
     * Devuelve el título del <code>Item</code> o <code>null</code> en
el caso
     * de que no se haya especificado.
     */
    public String getTitle();

       /**
        * Indica si el <code>Item</code> es visible.
     * <p>Devuelve <code>true</code> en el caso del que el atributo
"isvisible"
     * de la etiqueta &lt;item&gt; se haya establecido al valor "true"
y se devolverá
     * falso cuando dicho atributo haya sido establecido a "false".</p>
     * <p>En caso de no haberse establecido ningún valor explícitamente
el valor
     * devuelto será <code>true</code>.</p>
     * <p>Cuando se devuelve <code>true</code> se indica que el
<code>Item</code>
     * debe ser visible cuando se presente la estructura del PIF.</p>
     * <p>Cuando se devuelve <code>false</code> el <code>Item</code> no
debe
     * mostrarse, sin embargo sus <code>Item</code> hijos si pueden
mostrarse
     * dependiendo del valor de su atributo "isvisible".</p>
     *
     * @deprecated Reemplazado por {@link #getIsVisible()}
     *
     * @see <i>Sección 4.10 del IMS Content Packaging Best Practice
Guide v1.1.3</i>
     *
     */
    public boolean isVisible();

       /**
        * Indica si el <code>Item</code> es visible.
     * <p>Devuelve <code>true</code> en el caso del que el atributo
"isvisible"
     * de la etiqueta &lt;item&gt; se haya establecido al valor "true"
y se devolverá
     * falso cuando dicho atributo haya sido establecido a "false".</p>
     * <p>En caso de no haberse establecido ningún valor explícitamente
el valor
     * devuelto será <code>true</code>.</p>
     * <p>Cuando se devuelve <code>true</code> se indica que el
<code>Item</code>
     * debe ser visible cuando se presente la estructura del PIF.</p>
     * <p>Cuando se devuelve <code>false</code> el <code>Item</code> no
debe
     * mostrarse, sin embargo sus <code>Item</code> hijos si pueden
mostrarse
     * dependiendo del valor de su atributo "isvisible".</p>
     *
     * @see <i>Sección 4.10 del IMS Content Packaging Best Practice
Guide v1.1.3</i>
```

```
     */
    boolean getIsVisible();

    /**
     * Devuelve el <code>Identifier</code> del <code>Item</code>.
     */
    public Identifier getIdentifier();

    /**
     * Devuelve el atributo "identifierref" del <code>Item</code>.
     * <p> Este <code>Identifier</code> puede ser de un
<code>Resource</code>,
     * un <code>Item</code>, una <code>Organization</code> o un
<code>Manifest</code>
     * que estén en el ámbito de este <code>Item</code>.</p>
     *
     * @return Devuelve el atributo "identifierref" del
<code>Item</code> como
     * un <code>Identifier</code>.
     *
     * @see <a href="#identifierref">Atributo "identifierref"</a>
     */
    public Identifier getIdentifierref();

       /**
        * Devuelve el <code>Resource</code> asociado al
<code>Item</code>.
        * <p>Un <code>Item</code> tienen asociado un
<code>Resource</code> en el
        * caso de que el <code>Identifier</code> devuelto por {@link
#getIdentifierref()}
        * pertenezca a un <code>Resource</code>.</p>
        */
    public Resource getContent();

    /**
     * Devuelve todos los <code>Item</code> que son hijos directos de
este <code>Item</code>.
     * <p>Devuelve todos los <code>Item</code> que son hijos directos
de este
     * <code>Item</code> esto incluye a:</p>
     * <ol>
     * <li><strong>Hijos directos:</strong> Se tratan de etiquetas
&amp;lt;item&amp;gt;
     * dentro del propio <code>Item</code></li>
     * <li><strong>Hijos "virtuales":</strong> Se tratan de
<code>Item</code> que
     * han sido añadidos debido a que el atributo "identifierref" del
<code>Item</code>
     * referencia a alguna etiqueta que no es un
<code>Resource</code>.</li>
     * </ol>
     *
     * @see <a href="#identifierref">Atributo "identifierref"</a>
     * @see <a href="#itemsVirtuales">Items "virtuales"</a>
     */
    public List getAllItems();

    /**
     * Devuelve todos los <code>Item</code> que son hijos directos de
este <code>Item</code>.
     */
    public List getItems();


    /**
     * Devuelve el número de hijos del <code>Item</code>.
     */
    public int itemsCount();


       /**
        * Devuelve el <code>index-ésimo</code> <code>Item</code>.
        * <p>Devuelve el <code>index-ésimo</code> <code>Item</code>
que es hijo
```

```
         * directo del <code>Item</code></p>
         *
         * @param index Posición del <code>Item</code> hijo que
queremos obtener
         *
         * @throws IndexOutOfBoundsException Se lanza si
<code>index</code> < 0 ó
         * <code>index</code> >= {@link #itemsCount() }
         */
    public Item getItem(int index);

    /**
     * Devuelve el <code>Item</code> cuyo <code>Identifier</code> es
<code>identifier</code>.
     * <p>Devuelve el <code>Item</code> cuyo atributo "identifier" es
     * <code>identifier</code>.</p>
     * <p>El <code>Item</code> que se devuelve puede ser un
descenciente de cualquier
     * nivel.</p>
     */
    public Item getItem(Identifier identifier);

    /**
     * Devuelve el valor del atributo "parameters" del
<code>Item</code>.
     */
    public String getParameters();

        /**
         * Devuelve los metadatos del <code>Item</code>
         */
    public Metadata getMetadata();

        /**
     * Establece el título del <code>Item</code>.
     * <p>Establece el valor de &lt;title&gt; que es usado como título
del
     * <code>Item</code>. Si el parámetro <code>title</code> es la
cadena
     * vacía o <code>null</code>, se eliminará la etiqueta
&lt;title&gt;.</p>
     *
     * @param title Nuevo titulo del <code>Item</code>
     */
    public void setTitle(String title);

    /**
     * Establece si el <code>Item</code> será visible.
     *
     * @see #getIsVisible()
     *
     * @param isVisible Indica si el <code>Item</code> es visible.
     */
    public void setIsVisible(boolean isVisible);

        /**
     * Establece el <code>Identifier</code> del <code>Item</code>
     *
     * @param identifier Nuevo <code>Identifier</code> del
<code>Item</code>.
     *
     * @throws NullPointerException Se lanza si <code>identifier</code>
es <code>null</code>.
     * @throws IllegalArgumentException Se lanza en el caso de que
<code>identifier</code> sea
     * un <code>Identifier</code> que ya existe en el
<code>Manifest</code> y es distinto al
     * <code>Identifier</code> actual del <code>Item</code>.
     */
    public void setIdentifier(Identifier identifier);

        /**
     * Establece el <code>Identifier</code> del elemento que referencia
el <code>Item</code>
```

```
        * <p>Establece el <code>Identifier</code> del elemento que
referencia el <code>Item</code>,
        * si <code>newIdentifier</code> no está en el contexto del
<code>Item</code> o
        * no existe ningúna etiqueta con el atributo "identifier" con
dicho o valor,
        * el atributo "identifierref" del <code>Item</code> será
eliminado.</p>
        * <p>si <code>newIdentifierref</code> es <code>null</code> se
eliminara el
        * atributo "identifierref" si existía.</p>
        *
        * @param newIdentifierref <code>Identifier</code> del nuevo
elemento a referenciar
        *
        * @see <a href="#identifierref">Atributo "identifierref"</a>
        *
     * @throws IllegalArgumentException Se lanza en el caso de que se
quiera
     * referenciar a sí mismo.
     */
    public void setIdentifierref(Identifier newIdentifierref);

       /**
        * Añande un nuevo <code>Item</code>
        *
        * @param item <code>Item</code> a añadir.
        *
        * @throws IllegalStateException Se lazan en el caso de que
<code>item</code>
        * no sea un <code>Item</code> nuevo; es decir, que no haya
sido añadido
        * al <code>Manifest</code> y no haya sido obtenido a través de
los métodos
        * getXXX(...);
        *
        * @throws IllegalArgumentException Se lanza en el caso de que
<code>item</code> no
        * se añada a un <code>Item</code> que esté en un
<code>Manifest</code> y éste
        * sea distinto al que ha creado a <code>item</code>.
        */
    public void addItem(Item item);


       /**
     * Añade <code>item</code> en la <code>index-ésima</code> posicion.
     * <p>Añade <code>item</code> en la <code>index-ésima</code>
posicion de los
     * <code>Item</code> que son hijos directos del
<code>Item</code>.</p>
     * <p><code>index</code> tiene que ser >= 0  ( el 0 indica que se
quiere insertar
     * antes del primer hijo) y <= el valor devuelto por {@link
#itemsCount()} (un
     * valor igual al devuelto por {@link #itemsCount()} indica que se
quiere añadir
     * al final).</p>
     *
     * @param index Posición en la que queremos añadir el nuevo
<code>Item</code>
     * @param item <code>Item</code> que queremos añadir
     *
     * @throws IndexOutOfBoundsException Se lanza si <code>index</code>
es < 0 ó si
     *  es > {@link #itemsCount()}.
     *
        * @throws IllegalStateException Se lazan en el caso de que
<code>item</code>
        * no sea un <code>Item</code> nuevo; es decir, que no haya
sido añadido
        * al <code>Manifest</code> y no haya sido obtenido a través de
los métodos
        * getXXX(...);
        *
```

```
         * @throws IllegalArgumentException Se lanza en el caso de que
<code>item</code> no
         * se añada a un <code>Item</code> que esté en un
<code>Manifest</code> y éste
         * sea distinto al que ha creado a <code>item</code>.
      */
    public void addItem(int index, Item item);

       /**
     * Elimina <code>item</code> del <code>Item</code>.
     * <p>Elimina <code>item</code> tiene que ser hijo directo del
<code>Item</code>.</p>
     *
     * @param item <code>Item</code> que queremos eliminar.
        *
     * @return Devuelve <code>true</code> en el caso de que se haya
elminado el <code>Item</code>
     * y <code>false</code> en otro caso.
     */
    public boolean removeItem(Item item);

       /**
     * Elimina el  <code>index-ésimo</code> <code>Item</code>.
        * <p>Elmina el <code>Item</code> que es hijo directo del
<code>Item</code>
     * y está en la <code>index-ésima</code> posición.</p>
     *
        *
     * @return Devuelve el <code>Item</code> que ha sido eliminado.
     *
     * @throws IndexOutOfBoundsException Se lanza en el caso de que
index sea < 0 o
     * sea >= que el valor devuelto por {@link #itemsCount()}.
     */
    public Item removeItem(int index);

    /**
     * Establece el atributo "parameters" del <code>Item</code>.
     * <p>Establece el atributo "parameters" del <code>Item</code>, en
caso
     * de que <code>parameters</code> sea la cadena vacía o
<code>null</code>
     * el atributo será eliminado.</p>
     *
     * @param parameters Nuevos parámetros del <code>Item</code>
     *
     * @return Devuelve los antiguos parámetros o <code>null</code> en
el
     * caso de que no tuviera
     */
    public String setParameters(String parameters);
}
```

C.2.4.- Resource.java

```
/*
  Copyright  © 2.003 Universidad Complutense de Madrid, Proyecto E-Aula
*/

package es.ucm.sip.eaula.cp;

import java.util.List;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Interfaz que representa un &lt;resource&gt; de un manifiesto.
 * <p>Dentro de la etiqueta &lt;resource&gt; podemos distinguir dos
tipos de
 * recursos:</p>
 * <ol>
```

```
 * <li><strong>Recurso principal:</strong> Se trata del recurso que
será mostrado
 * al usuario</li>
 * <li><strong>Recursos auxiliares:</strong></li> Se tratan de recursos
que usa el
 * recurso principal.</li>
 * </ol>
 * <p>El <strong>recurso principal</strong> est&aacute; identificado en
la etiqueta
 * &lt;resource&gt; a trav&eacute;s de su atributo &quot;href&quot;.
Podemos tener
 * dos casos dependiendo de como sea la URI que contiene este
atributo:</p>
 * <ul>
 * <li><strong>La URI es absoluta:</strong> En este caso el recurso
principal
 * est&aacute; almacenado en el paquete. Normalmente cuando se da este
caso, la etiqueta
 * &lt;resource&gt; deber&iacute;a de ser vac&iacute;a.<br>
 * Consulte {@link #getEntryPoint(Item)} para m&aacute;s
informaci&oacute;n.</li>
 * <li><strong>La URI es relativa:</strong> En este caso el recurso
principal est&aacute;
 * en el paquete, adem&aacute;s el atributo &quot;href&quot; coincidira
con el
 * el mismo atributo de alguna de las etiquetas &lt;file&gt; que
contendr&aacute;
 * la etiqueta &lt;resource&gt;
 * </ul>
 */
public interface Resource {
    /** @link dependency */
    /*#Dependency lnkDependency;*/

    /**
     * Devuelve una <code>URI</code> que apunta al recurso físico que
define este
     * <code>Resource</code>.
        * <p>Devuelve una <code>URI</code> que apunta al recurso
físico o
     * <code>null</code> en el caso de que no esté especificado el
atributo
     * &quot;href&quot. Si la <code>URI</code> es relativa se intentará
resolver
     * de la siguiente manera:</p>
     * <ol>
     * <li>Si <code>hrefURI</code> es absoluta hemos terminado.</li>
     * <li>Se intenta resolver <code>hrefURI</code> con el valor del
atributo
     * "xml:base" del &lt;resource&gt; al que pertenece
<code>hrefURI</code>,
     * si no existe o sigue siendo relativa seguimos con el algoritmo.
Si por el
     * contrario es absoluta hemos terminado.</li>
     * <li>Se intenta resolver <code>hrefURI</code> con el valor del
atributo
     * "xml:base" de la etiqueta &lt;resources&gt; que contiene la
etiqueta
     * &lt;resource&gt; al que pertenece <code>hrefURI</code>, si no
existe o
     * sigue siendo relativa seguimos con el algoritmo. Si por el
contrario es
     * absoluta hemos terminado.</li>
     * <li>Se intenta resolver <code>hrefURI</code> con el valor del
atributo
     * "xml:base" de la etiqueta &lt;manifest&gt; que contiene la
etiqueta
     * &lt;resources&gt; que contiene la etiqueta &lt;resource&gt; al
que
     * pertenece <code>hrefURI</code>, en cualquier caso el algoritmo
termina
     * llegado este paso.</li>
     * </ol>
     * <p>En el caso de que la URI se trate de una URI absoluta, se
aplicará el
```

```
     * algoritmo que se define en la sección 4.2 del <i>IMS Content
Packaging
     * Information Model v1.1.3</i>.</p>
     * <p>Por lo que se ha podido deducir de dicho documento, los
parámetros son
     * usados solo en el caso de que se trate de una URL absoluta, es
decir,
     * que la URL apunte a un recurso externo al PIF.</p>
     *
     * @param refItem <code>Item</code> del que se tomarán los
parámetros.
     *
     */
    public URI getEntryPoint(Item refItem)throws URISyntaxException;

    /**
     * Devuelve una <code>URI</code> que apunta al recurso físico que
define este
     * <code>Resource</code>.
     * <p>En esta llamada no se añadirán parámetros.</p>
     *
     * @see #getEntryPoint(Item)
     *
     */
    public URI getEntryPoint()throws URISyntaxException;

       /**
     * Devuelve una <code>URI</code> con el valor del atributo
&amp;quot;href&amp;quot;
     * de la etiqueta &amp;lt;resource&amp;gt; que representa este
<code>Resource</code>.
     * Devuelve una <code>URI</code> con el valor del atributo
&amp;quot;href&amp;quot;
     * o <code>null</code> en el caso de que no esté especificado.
     *
     * @since 1.1
     */
    public URI getLocalEntryPoint()throws URISyntaxException;

    /**
     * Devuelve una lista de <code>URI</code> de los archivos del
recurso.
     * <p>Devuelve una lista de <code>URI</code> de los archivos del
recurso que están
     * especificados mediante los atributos &amp;quot;href&amp;quot; de las
etiquetas
     * &amp;lt;file&amp;gt;.</p>
     * <p>Estas <code>URI</code> han intentado resolverse de la
siguiente manera:</p>
     * <ol>
     * <li>Si <code>hrefURI</code> es absoluta hemos terminado.</li>
     * <li>Se intenta resolver <code>hrefURI</code> con el valor del
atributo
     * "xml:base" del &amp;lt;resource&amp;gt; al que pertenece
<code>hrefURI</code>,
     * si no existe o sigue siendo relativa seguimos con el algoritmo.
Si por el
     * contrario es absoluta hemos terminado.</li>
     * <li>Se intenta resolver <code>hrefURI</code> con el valor del
atributo
     * "xml:base" de la etiqueta &amp;lt;resources&amp;gt; que contiene la
etiqueta
     * &amp;lt;resource&amp;gt; al que pertenece <code>hrefURI</code>, si no
existe o
     * sigue siendo relativa seguimos con el algoritmo. Si por el
contrario es
     * absoluta hemos terminado.</li>
     * <li>Se intenta resolver <code>hrefURI</code> con el valor del
atributo
     * "xml:base" de la etiqueta &amp;lt;manifest&amp;gt; que contiene la
etiqueta
     * &amp;lt;resources&amp;gt; que contiene la etiqueta &amp;lt;resource&amp;gt; al
que
     * pertenece <code>hrefURI</code>, en cualquier caso el algoritmo
termina
```

```
     * llegado este paso.</li>
     * </ol>
     * <p>Las depencias especificadas mediante las etiquetas
&lt;dependency&gt;
     * solo se buscaran dentro de las etiquetas &lt;resource&gt; del
mismo manifiesto
     * que contiene la etiqueta &lt;resource&gt; que est&aacute;
representada por
     * este <code>Resource</code>.</p>
     *
     * @deprecated Ha sido reemplazada por {@link #getFiles()} ya que
en el
     * <i>IMS Content Packaging Information Model v 1.1.3</i> no se
indica que
     * haya que aplicar el algoritmo definido en la sección 4.2 del
citado documento a
     * los atributos &quot;href&quot; de las etiquetas &lt;file&gt;,
por lo que
     * no lo aplicamos.
     *
     * @param refItem <code>Item</code> del que se tomarán los
parámetros.
     */
    public List getFiles(Item refItem) throws URISyntaxException;

    /**
     * Devuelve una lista de <code>URI</code> de los archivos del
recurso.
     * <p>Devuelve una lista de <code>URI</code> de los archivos del
recurso que están
     * especificados mediante los atributos &quot;href&quot; de las
etiquetas
     * &lt;file&gt;.</p>
     * <p>Estas <code>URI</code> han intentado resolversede la
siguiente manera:</p>
     * <ol>
     * </ol>
     * <p>Las depencias especificadas mediante las etiquetas
&lt;dependency&gt;
     * solo se buscaran dentro de las etiquetas &lt;resource&gt; del
mismo manifiesto
     * que contiene la etiqueta &lt;resource&gt; que est&aacute;
representada por
     * este <code>Resource</code>.</p>
     * <p>En el <i>IMS Content Packaging Information Model v 1.1.3</i>
no se indica que
     * haya que aplicar el algoritmo definido en la sección 4.2 del
citado documento a
     * los atributos &quot;href&quot; de las etiquetas &lt;file&gt;,
por lo que
     * no lo aplicamos.</p>
     */
    public List getFiles() throws URISyntaxException;

       /**
     * Devuelve el número de &lt;file&gt; que tiene el
<code>Resource</code>
     */
       public int fileCount();

       /**
     * Devuelve el <code>Identifier</code> del <code>Resource</code>
     *
     * Devuelve el <code>Identifier</code> del <code>Resource</code> o
     * <code>null</code> si no se ha especificado.
     */
    public Identifier getIdentifier();

       /**
     * Devuelve el <code>ContentType</code> de este
<code>Resource</code> o
     * <code>null</code> si no se ha especificado.
     */
    public ContentType getType ();
```

```
       /**
     *
     * @param item <code>Item</code> que referencia el
<code>Resource</code>
     *
     * @see #getEntryPoint(Item )
     * @see #getLocalEntryPoint()
     *
     * @deprecated <code>isLocal(Item item)</code> ha sido sutituida
por {@link #isPackageLocal()}
     * ya que para saber si el recurso físico apuntado por
     * &lt;resource href=&quot;...&quot; &gt;es local al paquete que
contiene el
     * manifiesto en el que aparece esta etiqueta, no es necesario
disponer de un
     * <code>Item</code> que lo referencie.
     */
    public boolean isLocal(Item item);

       /**
     * Indica si el recurso principal es local al paquete.
      * <p>Devuelve <code>true</code> en el caso de que el recurso
principal del
     * <code>Resource</code> es local al paquete que contiene el
manifiesto
     * en el que aparece este <code>Resource</code>.</p>
     * <p>Un <code>Resource</code> es local al PIF si el atributo
"href"
     * es un <code>URI</code> relativa.</p>
     *
     * @return Devuelve <code>true</code> si "href" es una
<code>URI</code>
     * relativa y <code>false</code> en otro caso.
     */
    public boolean isPackageLocal();

       /**
     * Devuelve una lista de <code>Dependency</code>'s con todas las
     * dependencias que hay en el <code>Resource</code>
     *
     */
    public List getDependencies();


       /**
        * Añade una nueva &lt;dependency&gt; con otro
<code>Resource</code>.
     *
     * @param identifier <code>Identifier</code> del
<code>Resource</code>
     * del que queremos depender.
     *
     * @throws IllegalArgumentException Se lanza en el caso de que el
     * identificador no corresponda a un &lt;resource&gt; accesible.
También
     * se lanza en el caso de que se quiera añadir al
<code>Resource</code> una
     * dependencia consigo mismo.
     *
     * @throws NullPointerException Se lanza si <code>identifier</code>
es <code>null</code>.
     */
    public void addDependency(Identifier identifier);

    /**
     * Elimina la &lt;dependency&gt; con otro <code>Resource</code>
     *
     * @param identifier <code>Identifier</code> del
<code>Resource</code> dependiente.
     */
       public void removeDependency(Identifier identifier);

    /**
     * Elimina todas las &lt;dependency&gt; de este
<code>Resource</code>
```

```
     */
    public void clearDependencies();

       /**
     * Añade un nuevo archivo como recurso secundario del
<code>Resource</code>.
        * <p>Añade un nuevo archivo como recurso secundario,
<code>fileURI</code> tiene
     * que ser una <code>URI</code> relativa teniendo en cuenta que
cuando se
     * obtengan dichas <code>URI</code>'s mediante {@link #getFiles()}
se
     * aplicar&a el m&eacute;todo de resoluci&oacute;n all&iacute;
especificado.</p>
        *
     * @param fileURI <code>URI</code> que apunta al fichero.
     *
     * @throws IllegalArgumentException Se lanza en el caso de que
<code>fileURI</code>
     * sea una <code>URI</code> absoluta o <code>null</code>.
     */
    public void addFile(URI fileURI);

       /**
     * Elimina el &lt;file&gt; que se encuentra en la posicion
<code>index</code>.
     *
     * @param index Posición del &lt;file&gt; a eliminar.
     *
        * @return Devuelve la URI resuleta del &lt;file&gt; que ha
sido eliminado.
     *
     * @throws IndexOutOfBoundsException Se lanza en el caso de que
<code>index</code>
     * sea menor que 0, o que sea mayor o igual que el tamaño devuelto
por {@link #fileCount()}.
     *
     * @throws IllegalArgumentException Se lanza en el caso de que
fileURI corresponda
     * al &lt;file&gt; cuyo atributo href coincide con el valor
devuelto por {@link #getLocalEntryPoint() }
     */
    public URI removeFile(int index);

       /**
     * Elimina el &lt;file&gt; cuyo atributo "href" resuelto, coincida
con <code>fileURI</code>.
     *
     * @param fileURI <code>URI</code> del &lt;file&gt; a eliminar.
     *
     * @return Devuelve <code>true</code> en el caso de que se haya
eliminado el &lt;file&gt; o
     * <code>false</code en otro caso.
     *
     * @throws IllegalArgumentException Se lanza en el caso de que
fileURI corresponda
     * al &lt;file&gt; cuyo atributo href coincide con el valor
devuelto por {@link #getLocalEntryPoint() }
     */
    public boolean removeFile(URI fileURI);


    /**
     * Elimina todos los &lt;file&gt; del <code>Resource</code>
     */
    public void clearFiles();


       /**
     * Establece el atributo "href" del <code>Resource</code>.
     *
     * @param newURI Nuevo valor del artributo "href" del
<code>Resource</code>
     *
```

```
     * @throws NullPointerException Se lanza si <code>newURI</code> es
<code>null</code>
     */
    public void setLocalEntryPoint(URI newURI);

       /**
     * Establece el <code>Identifier</code> del <code>Resource</code>
     *
     * @param identifier Nuevo <code>Identifier</code> del
<code>Resource</code>.
     *
     * @throws NullPointerException Se lanza si <code>identifier</code>
es <code>null</code>.
     * @throws IllegalArgumentException Se lanza en el caso de que
<code>identifier</code> sea
     * un <code>Identifier</code> que ya existe en el
<code>Manifest</code> y es distinto al
     * <code>Identifier</code> actual del <code>Resource</code>.
     */
    public void setIdentifier(Identifier identifier);

    /**
     * Establece el atributo "xml:base" del <code>Resource</code>
     * <p>Si <code>baseURI</code> es <code>null</code> se eliminará
     * el atributo "xml:base".</p>
     *
     * @param baseURI Nuevo valor del atributo "xml:base"
     */
    public void setXMLBase(URI baseURI);

    /**
     * Devuelve el atributo "xml:base" del <code>Resource</code>.
     * <p>Devuelve el atributo "xml:base" del <code>Resource</code> sin
resolver
     * en el caso de que la <code>URI</code> sea relativa.</p>
     *
     * @return <code>URI</code> con el valor del atributo "xml:base"
     */
    public URI getXMLBase() throws URISyntaxException;

       /**
     * Devuelve los metadatos del <code>Resource</code>
     * <p>Devuelve los metadatos del <code>Resource</code>. En el caso
de tratar
     * los metadatos de alguna manera específica, se debería devolver
un objeto
     * de la clase que implemente la API para acceder a dichos
metatados. En su
     * defecto se devolverán los metadatos en bruto (dependiente de la
API que
     * usemos para acceder al XML) o <code>null</code> si no se tienen
metadatos.</p>
     */
    public Metadata getMetadata();

       /**
     * Establece el tipo del <code>Resource</code>
     *
     * @param contentType <code>ContentType</code> del
<code>Resource</code>
     *
     * @throws NullPointerException Se lanza si
<code>contentType</code> es
     * <code>null</code>
     */
    void setType(ContentType contentType);
}
```

C.2.5.- Identifier.java

```
/*
  Copyright  © 2.003 Universidad Complutense de Madrid, Proyecto E-Aula
*/

package es.ucm.sip.eaula.cp;

import org.ietf.uri.URN;
import com.stafney.UID;
import org.ietf.uri.URN;
import org.ietf.uri.MalformedURNException;
import com.stafney.UID;
import es.ucm.sip.eaula.util.logging.LogFactory;
import org.apache.commons.logging.Log;
public class Identifier {
    private static final Log log =
LogFactory.getLog(es.ucm.sip.eaula.cp.Identifier.class);
    /**
     * Es el identificador nulo
     */
    public  static final Identifier NULL_IDENTIFIER = new
NullIdentifier();
    private String ident;
    private URN urn;
    private UID uid;

    public Identifier(String ident) {
        this.ident=null;
        this.urn=null;
        this.uid=null;
        if( ident == null ){
            throw new NullPointerException("ident es null");
        }

        if( ident.equals("")){
               throw new IllegalArgumentException("ident no puede ser
\"\"");
        }

        try{
            this.urn= new URN(ident);
        }catch(MalformedURNException ex1){
                      try{
               this.uid = UID.valueOf(ident);
            }catch(IllegalArgumentException ex2){
                   this.ident=ident;
            }
        }
    }

    /**
     * Constructor protegido
     */
    protected Identifier(){
        this.uid=null;
        this.ident=null;
        this.urn=null;
    }

       /**
     * Devuelve la URN que representa el <code>Identifier</code>.
     * En caso de que el <code>Identifier</code> tenga la estructura de
una
     * URN, se devuelve un un objeto de la clase {@link
org.ietf.uri.URN}, en otro
     * caso se devuelve <code>null</code>
     */
    public URN getURN(){
        return this.urn;
    }

       /**
```

```
     * Devuelve el UID que representa el <code>Identifier</code>.
     * En caso de que el <code>Identifier</code> tenga la estructura de
un
     * UID, se devuelve un un objeto de la clase {@link
com.stafney.UID}, en otro
     * caso se devuelve <code>null</code>
     */
    public UID getUID(){
               return this.uid;
    }

       /**
     * Devuelve una cadena de texto que representa el
<code>Identifier</code>
     */
    public String toString(){
        String result = null;

        if ( this.urn != null ){
                        result = this.urn.toString();
        }else if ( this.uid != null ){
                        result = this.uid.toString();
        }else{
                        result = this.ident;
        }

        return result;
    }

       /**
     * Comparación de igualdad String
     */
    public boolean equals(Object object){
        boolean result=false;

        if(object instanceof Identifier){
            Identifier id = (Identifier)object;

            if ( ! NULL_IDENTIFIER.equals(id)  ){
                        result = this.toString().equals(id.toString());
            }
        }
        return result;
    }

       /**
     * Devuelve el hashcode del identificador
     */
    public int hashCode(){
        int result = 0;
        if( this.urn != null){
            result = this.urn.hashCode();
        }else if(this.uid != null){
            result = this.uid.hashCode();
        }else{
            result = this.ident.hashCode();
        }
        return result;
    }

       /**
     * Crea un <code>Identifier</code> basado en la generacion de
UUID's.
     *
     * <p><b>NOTA DE IMPLEMENTACION</b></p>
     * <p>Para que los identificadores sean GUID's es necesario usar
JNI para acceder
     * a la MAC de la targeta Ethernet de la máquina. Para ello se usa
una DLL
     * (en el caso de Windows) y para que esta funcione tiene que
encontrarse en un
     * directorio que esté en el PATH del sistema.</p>
     */
    static public Identifier createUIDIdentifier() {
               Identifier identifier = new Identifier();
```

```
            identifier.uid = UID.newInstance();
                   return identifier;
        }

        /**
         * Innerclass usada para representar el identificador nulo.
         */
        private static final class NullIdentifier extends Identifier{
            private static final String NULL_IDENTIFIER_STRING =
    "[NULL_IDENTIFIER]";
            protected NullIdentifier(){
                super();
            }

            public boolean equals(Object obj){
                return (obj instanceof NullIdentifier);
            }

            public int hasCode(){
                return 0;
            }

            public String toString(){
                return NULL_IDENTIFIER_STRING;
            }
        }
    }
```

Apéndice D: Bibliografía e Información legal

D.1.- Bibliografía

1. Allamaraju, S.: Professional Java Servlets 2.3. Wrox Press, ©2002
2. Goodwill, J: Professional Jakarta Struts. Wiley, © 2004
3. Gamma, E.: Design Patterns, Elements of Reusable Object Oriented Software. Addison Wesley, © 1995.
4. IMSRID. IMS Persistent, Location-Independent, Resource Identifier Implementation Handbook. Versión 1.0 Final Handbook. Disponible on-line: http://www.imsglobal.org.
5. Leach, Paul J. . UUIDs and GUIDs. INTERNET-DRAFT <draft-leach-uuids-guids-01.txt>. Network Working Group, 1998.
6. Berners-Lee, T.."Uniform Resource Identifiers (URI): Generic Syntax". Masinter, August 1998. Note that RFC 2396.
7. Abelson, H., Sussman, J., Sussman, J.. *Structure and Interpretation of Computer Programs*, MIT Press, 1984.
8. IMS CP(2003). Content Packaging Information Model. Version 1.1.3 Final Specification. Disponible on-line: http://www.imsglobal.org/content/packaging/index.cfm.
9. Birbeck, M. Professional XML. Wrox Press, © 2001.
10. IMS SS(2003). Simple Sequencing Specification. Version 1.0 Final Specification. Disponible on-line: http://www.imsglobal.org/simplesequencing/index.cfm
11. IMS LD(2003). Learning Design Specification. Version 1.0 Final Specification. Disponible on-line: http://www.imsglobal.org/learningdesign/index.cfm.
12. Manero Iglesias, B. Estudio de la propuesta IMS de estandarización de enseñanza asistida por computadora. Trabajo de Doctorado, disponible on-line: http://eaula.sip.ucm.es/bibestandares.pdf.
13. Sancho, P., Manero, B., Fernández-Manjón, B. (2004, in press). Learning objects definition and use in <e-Aula>: Towards a Personalized Learning Experience. EduTech'2004. IFIP World Conference.
14. Sancho P., Lenguajes de marcado y su aplicación en el dominio de las tecnologías de aprendizaje en la web, UCM, 2002.
15. Manero B., Estudio de la propuesta IMS de estandarización de enseñanza asistida por computadora, UCM, 2003.
16. Fernández-Manjón, B., Fernández-Valmayor, A., Navarro, A., Sierra, J.L. (2002). "Application of XML Mark-up Languages to Software Development". Upgrade (SIN: 1684-5285), also in Novática (ISSN: 0211-2124). III, 4. **15-20**
17. Pilar Sancho, Borja Manero Iglesias, Baltasar Fernández-Manjón.(2003). "Usos experimentales de estándares educativos en el sistema <e-aula>". CHALLENGES 2003, III Conferencia Internacional sobre Tecnologías de la Información y la Comunicación en la Educación, realizada conjuntamente con el 5º SIIE Simposio Internacional de Informática Educativa, Braga,. **297-304.**
18. Fernandez-Manjon, B., Fernandez-Valmayor, A., (1997). Improving World Wide Web educational uses promoting hypertext and standard general markup language content-based features. *Education and Information Technologies*, 2(3), 193-206.

19. Fernandez-Manjon, B., Sancho P. Creating Cost-effective Adaptative Educational Hypermedia Based on Markup Technologies and E-Learning Standards. *Interactive Educational Multimedia* (ISSN: 1576- 4990)
20. Tennison, J.: Beggining XSLT. Wrox Press, © 2002
21. Brown S.: Professional JSP Tag Libraries. Wrox Press, © 2002
22. Brown S.: Professional JSP. Wrox Press, © 2002
23. Ahmed K.: Professional Java XML Wrox Press, © 2001
24. Cavaness C.: Programming Jakarta Struts. O'Reilly, ©2003

D.2.- Palabras clave

- Estándares educativos
- IMS
- XML
- J2EE
- QTI
- Content Packaging
- Universidad virtual
- Educación a distancia
- e-learning
- e-Aula

D.3.- Autorización legal

Los autores de este proyecto autorizan a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado.

Javier López Moratalla Iván Martínez Ortiz Pablo Moreno Ger