Sistemas Informáticos
Curso 2007-2008

iChasqui: Repositorio de Objetos Virtuales Independientes del Dominio

Jesús Enrique Arnaiz Barrero

Dirigido por:
José Luis Sierra Rodríguez

Facultad de Informática
Universidad Complutense de Madrid

Resumen

En los últimos años, el e-learning o aprendizaje apoyado por recursos informáticos se ha extendido de forma bastante notoria en casi todos los campos del conocimiento humano. Incluso, se han creado centros cuyo método de enseñanza principal es éste. La independencia de la distancia física y la eliminación de la necesidad de que alumno y profesor sigan un horario preestablecido parecen haber contribuido a ello.

Este trabajo pretende ser un paso más en lo que es una línea de investigación del Departamento de Ingeniería del Software e Inteligencia Artificial de la Facultad de informática de la Universidad Complutense de Madrid, continuando los esfuerzos iniciados en otros proyectos llevados a cabo en el mismo departamento en otros cursos. Basándonos en las ideas que éstos aportan, se ha intentado definir un estándar para objeto virtuales sencillo e intuitivo, que haga fácil el manejo y comprensión del funcionamiento del sistema al usuario.

Al mismo tiempo, y a diferencia de estos trabajos, éste es el primer proyecto que no está diseñado pensando en un dominio específico, lo que le proporciona una gran versatilidad.

Abstract

During last years, e-learning has extended its presence in almost every area of human knowledge. Also, several educational centres whose main teaching method is this has been created.

This project tries to be the a new step in what has become a research area of the Software Engineering and Artificial Intelligence Department of the Computer Science School of the Universidad Complutense de Madrid. According to the results of other projects developed in the same department, we have tried to define a simple standard for virtual objects, which is going to make the utilization and comprehension of the way the system works easier.

At the same time, it is necessary to consider that this is the first project that has not been designed for a specific domain, which makes it so versatile.

Palabras clave

E-learning, objeto virtual, objeto de aprendizaje, parser, lenguaje, gramática

Índice

Capítulo 1. Introducción

1.1. Introducción al sistema

Este documento ofrece una perspectiva sobre el sistema iChasqui, realizado en el marco de la asignatura Sistemas Informáticos de la titulación Ingeniería en Informática de la Universidad Complutense de Madrid, durante el curso 2007 - 2008.

Está basado en el trabajo llevado a cabo en esta asignatura durante cursos anteriores, así como en otras aplicaciones desarrolladas por el Departamento de Ingeniería del Software e Inteligencia Artificial de la Facultad de Informática de la misma universidad, que a su vez sigue una de las líneas de investigación de este Departamento: los sistemas de e-learning.

En este sentido, contamos con cuatro antecedentes claros, como son los sistemas: Chasqui, Chasqui 2, MIGS 0.0 y MIGS 1.0, orientados a la digitalización de museos universitarios.

Sin embargo, al igual que estas aplicaciones desarrolladas en el Departamento y a diferencia de los sistemas de e-learning clásicos, no está centrado en proporcionar al usuario una forma de elaborar unidades didácticas, sino en posibilitar que el usuario digitalice objetos o ideas, de modo que puedan ser tratados con recursos informáticos.

Además, aunque guarda cierto parecido con estos sistemas, su desarrollo ha sido llevado a cabo de forma completamente independiente al de éstos. No se ha partido de ninguna aplicación previa.

Uno de los principales objetivos perseguidos con este proyecto es el de que el sistema sea independiente del dominio en el que trabaje el usuario. En otras palabras, se trata de que un usuario dedicado a la informática pueda usar la aplicación del mismo modo que lo hace otro dedicado a la construcción.

Esto marca una diferencia importante, puesto que otras aplicaciones se centraban en un museo determinado, mientras que el sistema iChasqui es más flexible.

1.2. Estructura de la memoria

En primer lugar se explica el contexto del cual surge este proyecto, formado por los sistemas e-learning.

A continuación, se explica con mayor profundidad los objetos virtuales y de aprendizaje, que son la estructura fundamental del sistema.

Después, se describen los antecedentes de esta aplicación: los primeros miembros de esta familia, que son los sistemas Chasqui, Chasqui 2, MIGS 0.0 y MIGS 1.0.

Más tarde, se comentan todos los detalles relativos al sistema iChasqui, que es el que nos ocupa: tipos de datos, cómo los trata, ...

El siguiente apartado es el de la arquitectura, diseño e implementación de la aplicación: tecnologías involucradas, capas de las que consta, casos de uso, estructura de la base de datos, ...

Finalmente, se incluyen tanto las conclusiones y las futuras líneas de trabajo como los apéndices: instalación, ...

Capítulo 2. Contexto del proyecto

2.1. E-learning

El e-learning es el término que se usa para referirse al aprendizaje que es distribuido o posibilitado haciendo uso de tecnologías electrónicas. De esta definición se puede inferir que el e-learning incluye un rango bastante amplio de tecnologías: televisión, video, Internet, aprendizaje basado en ordenador, ...

Normalmente, el instructor y el alumno están separados geográfica y/o temporalmente, y las tecnologías de la información son las encargadas de subsanar esta distancia. De esta manera, se consigue hacer accesible el conocimiento y dar la posibilidad de estudiar a personas que de otro modo se verían imposibilitadas para hacerlo por razones de diversa índole.

En la actualidad, este tipo de aprendizaje se ha extendido de forma bastante notoria: universidades a distancia o presenciales que refuerzan las clases con recursos on-line, empresas que dan formación a sus empleados sin necesidad de que éstos se desplacen. En gran parte, ésto es debido a la creciente expansión de Internet y al desarrollo de aplicaciones para este medio.

Algunos ejemplos de organizaciones que emplean estas prácticas son la Open University (Reino Unido) o la Penn State World Campus (Estados Unidos). En España podemos destacar la Universitat Oberta de Catalunya (UOC).

A continuación, pasamos a estudiar diferentes conceptos relacionados con el e-learning y sus tendencias más actuales.

2.1.1. Aprendizaje basado en ordenador

Se refiere al uso de ordenadores como componentes principales dentro del entorno formativo. De un modo más formal, se podría definir como el proceso de educación llevado a cabo usando un ordenador para enseñar y manejar el proceso de aprendizaje **[2]**.

2.1.2. Entrenamiento basado en ordenador

Se refiere a los servicios mediante los cuales el estudiante aprende trabajando con programas especialmente diseñados para el entrenamiento en las tareas propias de la ocupación a desarrollar **[2]**.

Un ejemplo de este tipo de servicios estaría constituido por los tutoriales que se incluyen con muchas aplicaciones, y cuya finalidad es instruir al usuario en el manejo de la misma.

En algunas ocasiones, el programa informático es distribuido por el profesor en clase, mientras que en otros se trata de un servicio ofrecido a través de la Internet.

Un caso particular de entrenamiento basado en ordenador es el entrenamiento basado en web (*web based training* o WBT), en el que es muy frecuente usar métodos interactivos, que incluyen chats, mensajería instantánea, boletines, videoconferencias, ...

2.1.3. Análisis asistido por ordenador del aprendizaje

Comprende desde tests de varias opciones hasta sistemas más complejos. En algunos casos, el programa puede generar realimentación para el alumno, basándose en los aciertos y errores de éste. Además de mostrarle los resultados de sus tests, la aplicación será la encargada de proporcionarle recursos encaminados a reforzar las áreas que todavía no domina **[2]**.

2.1.4. Elementos pedagógicos

Desde el punto de vista de las aplicaciones e-learning, se podría decir que son intentos de definir estructuras o unidades de material educativo. Pueden ser tests, lecciones, casos de estudio, grupos de discusión, ... Sin embargo, la definición debe ser independiente del formato: libro de texto, CD, página web, ...

2.1.5. Aprendizaje combinado

Es un término que aparece cada vez con más frecuencia, especialmente en entornos empresariales. Procede del inglés: *blended learning*.

De una forma simple, se puede definir como el aprendizaje basado en varios medios, y suele consistir en una combinación de aprendizaje dirigido por un instructor y herramientas basadas en web **[3]**.

Como caso paradigmático de este tipo de enseñanza podríamos citar el de la Open University británica, que tras realizar diversos estudios, descubrió que sus estudiantes preferían los libros en papel a la tecnología online a la hora de estudiar grandes temarios, pero que usaban mejor esos libros si estaban enlazados mediante un sitio web.

El aprendizaje combinado es una muestra de que el e-learning no desplaza o elimina otras formas de aprendizaje, sino que las complementa.

2.1.6. Configuraciones de productos e-learning

2.1.6.1. Portales de aprendizaje

Unifican las aplicaciones e-learning, el contenido y el entorno de distribución, y lo organizan según el acceso individual de cada persona al portal **[3]**.

También se suelen usar como soporte y lugar de encuentro para comunidades de aprendizaje, que son grupos de personas con unos intereses comunes en un área o tema

determinado. De hecho, algunas instituciones de enseñanza superior los usan como parte integral de la comunidad.

2.1.6.2. Sistemas de gestión de contenidos pedagógicos

El nombre procede de la traducción de la expresión en inglés Learning Content Management System, a la que corresponden las siglas *CMS* o *LCMS* **[3]**.

Una definición posible sería entorno multiusuario que permite a los desarrolladores del aprendizaje crear, almacenar, reutilizar, gestionar y distribuir contenidos pedagógicos digitales desde un repositorio de objetos central.

Los subsistemas principales de un LMCS son:

- Herramientas de ensamblaje de contenidos
- Herramientas de creación de contenidos
- Herramienta de integración que da soporte al registro, almacenamiento y recuperación de objetos
- Gestor de perfiles de alumnos
- Sistema de distribución de contenidos pedagógicos

2.1.6.3. Sistemas de gestión de cursos

Este tipo de sistema es especialmente relevante en el mercado de la educación, donde uno de los focos principales se encuentra en combinar distintos modos de distribución de los contenidos pedagógicos en un entorno guiado por un instructor **[3]**.

Estos sistemas se diferencian de los del apartado anterior en que están diseñados para integrar cursos enteros y combinarse con información de estudiantes o sistemas de registro.

Es importante destacar que deben ofrecer:

- Integración de componentes de los cursos con navegación o secuenciación
- Distribución de contenido al estudiante
- Posibilidad de creación de tests y encuestas
- Integración con otras herramientas como chats, hilos de discusión, videoconferencia etc.
- Herramientas de administración de alumnos

Entre los sistemas más conocidos de tipo comercial, podríamos citar: eCollege, Blackboard, y WebCT, siendo éste último en el que está basado el Campus Virtual de la Universidad Complutense de Madrid.

2.1.6.4. Herramientas de creación de contenidos

Permiten a los expertos en las áreas de conocimiento con las que está relacionado el proceso de aprendizaje crear y modificar objetos de aprendizaje **[3]**.

Cada herramienta suele estar centrada en un tipo de contenido: texto, audiovisual, gráfico, … Además, deben proporcionar facilidad de uso, para que sea fácil el aprendizaje de su manejo por parte de estas personas.

2.1.6.5. Herramientas de integración de contenidos

La integración de contenidos se centra en el enlace de objetos de aprendizaje y unidades pedagógicas entre sí, para conseguir módulos cohesionados. De esta forma, se permitiría una navegación adecuada y claramente definida entre los distintos objetos **[3]**.

También las herramientas de creación de contenidos ofrecen en algunos casos este tipo de posibilidades, aunque normalmente se suele utilizar una herramienta diferente.

2.2. Objetos virtuales y de aprendizaje

2.2.1. Objetos de aprendizaje

La orientación a objetos es un paradigma dentro de la informática que da una enorme importancia al diseño y uso de componentes reutilizables (Dahl & Nygaard, 1966) **[1]**.

Esta misma filosofía es en la que están basados los objetos de aprendizaje: componentes formativos pequeños (al menos en comparación con el tamaño de un curso, asignatura, seminario, museo, ...) que pueden ser reutilizados varias veces en distintos contextos educativos.

De hecho, la definición que da Wiley para un objeto de aprendizaje es "*un recurso digital que puede ser reusado para ayudar en el aprendizaje*". El Learning Technology Standards Committee del IEEE, por su parte, lo define como "*una entidad, digital o no digital, que puede ser usada para aprendizaje, educación o entrenamiento*" (*IEEE Standard for Learning Object Metadata)*.

Cabe destacar la existencia de dicho comité, creado en 1996, como muestra de la creciente importancia de este tipo de aplicaciones. O la de la *Alliance of Remote Instructional Authoring and Distribution Networks for Europe* (ARIADNE), institución creada en 2000, y que trabaja a nivel europeo, y el proyecto *Instructional Management Systems* (IMS) desarrollado en Estados Unidos **[1]**.

Una de sus principales ventajas reside en que, al ser entidades digitales, pueden ser distribuidos por Internet (entre otros medios) de forma prácticamente ilimitada; no tienen limitaciones físicas para su movimiento como ocurriría, por ejemplo, con una cinta de vídeo.

Precisamente ahí radica otra de sus principales características: pueden mejorarse siendo actualizados cada vez que haya una nueva versión de los mismos, lo que supone otra diferencia significativa con respecto a los medios de enseñanza tradicionales.

Una de las principales peculiaridades de los objetos de aprendizaje es la existencia de metadatos, llamados en ocasiones *Learning Object Metadata* (LOM). Se trata, por regla general, de un modelo de datos codificado en XML, y cuya finalidad es describir el contenido del objeto. El *IEEE 1484.12.1 – 2002 Standard for Learning Object Metadata* se encarga de dar un estándar abierto e internacionalmente reconocido.

Entre los datos interesantes que deben estar contenidos en el conjunto de metadatos de un objeto de aprendizaje estarían: el tipo de objeto, el autor, el propietario, los términos relativos a la distribución, el formato y los atributos pedagógicos, como los estilos de enseñanza y de interacción.

2.2.2. Objetos virtuales

Podemos definir un objeto virtual como un "*objeto digital que sirve para agrupar con fines educativos toda la información relacionada con un objeto real*". Su estructura suele seguir este patrón (Figura 2.1):

- **Datos:** permiten organizar y describir de forma extensible las características de un objeto.
- **Recursos:** constituyen un conjunto de elementos informativos asociados al objeto. Pueden ser clasificados en tres clases:
    - **Propios:** archivos digitales asociados de forma directa al objeto virtual.
    - **Ajenos:** archivos digitales asociados de forma directa a otros objetos del sistema pero que guardan una cierta relación con el objeto actual.
    - **Otros objetos virtuales:** considerados de forma unitaria, se asocian al objeto a través de una relación.
- **Metadatos:** describen el contenido y otras características de los datos, permitiendo a una persona entender y ubicar los datos.

En cuanto a los datos, hay que tener en cuenta que no deben sólo hacer referencia a los atributos físicos del objeto, sino a cualquier otro tipo de característica del mismo.

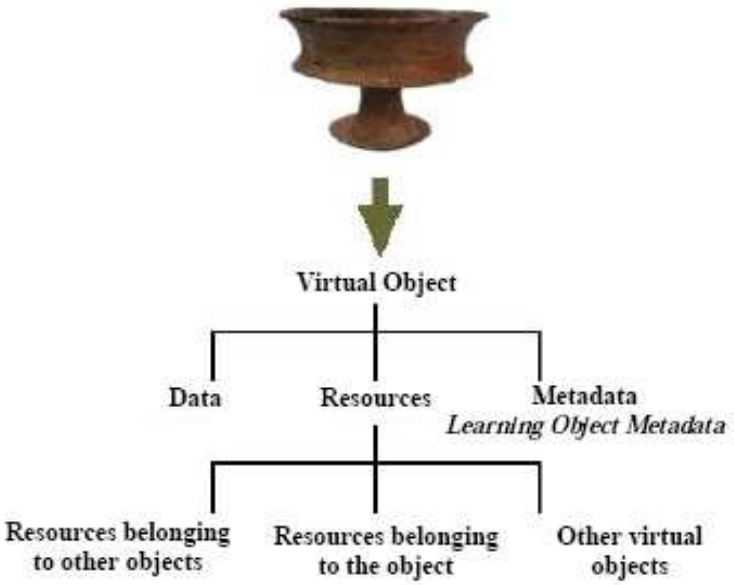


Figura 2.1. Objeto virtual visto desde el sistema Chasqui

Para los metadatos, existen modelos definidos que permiten disponer de una forma estándar de acceder a los objetos. Uno de ellos es el *Learning Object Metadata* (LOM), del que hemos hablado anteriormente.

El concepto de objeto virtual está íntimamente relacionado con el de objeto de aprendizaje, tal y como es usado en los entornos e-learning, y particularmente como es definido en el *Sharable Content Object Reference Model* (SCORM).

De esta forma, los objetos virtuales más simples, que únicamente tienen como recursos archivos multimedia, serían equivalentes a objetos atómicos en SCORM., mientras que los más complejos serían equivalentes a los Shareable Content Objects (SCOs). Éstos son objetos de aprendizaje compuestos por otros objetos.

Normalmente, la información que compone un objeto virtual se almacena en una base de datos, y los recursos se guardan en el sistema de archivos del servidor. La base de datos, por lo general, únicamente almacena la dirección en la que se encuentra dicho recurso.

2.3. Sistemas Chasqui

El sistema Chasqui es una aplicación informática destinada al manejo de objetos virtuales, un tipo de entidades basadas en los objetos de aprendizaje. Ha sido desarrollada en el Departamento de Ingeniería del Software e Inteligencia Artificial de la Facultad de Informática de la Universidad Complutense de Madrid.

Se han desarrollado varias versiones del mismo: la primera fue el resultado de la virtualización del Museo de Arqueología Antonio Ballesteros, de la Facultad de Geografía e Historia, mientras que la segunda fue relativa al Museo José García

Santesmases de la Facultad de Informática. Así mismo, cada una de estas dos ha sido revisada y actualizada posteriormente.

En realidad, y dado el número de versiones y las distintas características de cada una, tenemos que hablar de una familia de sistemas, cuyo mérito principal consiste en hacer posibles experiencias pedagógicas, basadas en la utilización de objetos virtuales como instrumentos para promover estrategias de aprendizaje constructivista y colaborativo.

2.3.1. Sistema Chasqui

El Departamento de Historia de América II de la Facultad de Geografía e Historia de la Universidad Complutense de Madrid posee una gran cantidad de recursos relacionados con su área de conocimiento: desde objetos con gran valor arqueológico hasta diarios de excavaciones, pasando por informes con resultados sobre análisis y perfiles de piezas arqueológicas). Se calcula que, en total, eran más de 2.000 piezas procedentes tanto de donaciones como de los diversos proyectos de investigación llevados a cabo por el departamento, encargado de estudiar las culturas americanas.

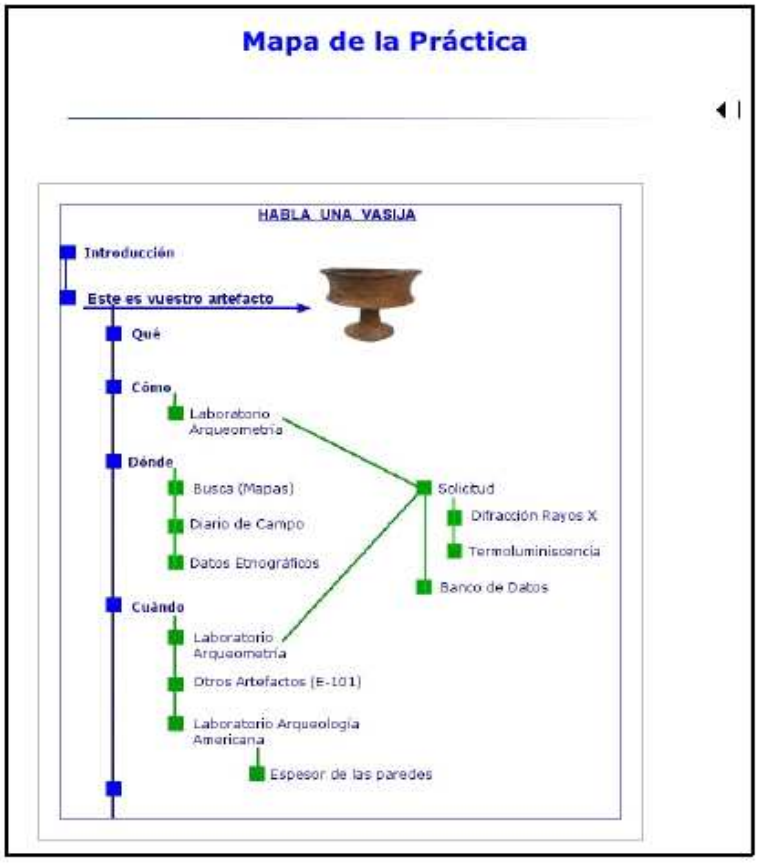


Figura 2.2. Ejemplo de objeto virtual visto desde el sistema Chasqui

Suelen usarse como material comparativo y de apoyo en las clases prácticas, ya que también permite especificar tareas para dirigir el aprendizaje de los alumnos.

Entre los objetos arqueológicos abundan los cerámicos, aunque otros son de piedra, concha o metal. Entre los etnográficos, en cambio, también los hay de cuerda, cuero, madera, plumas, e incluso caparazones de animales ... En definitiva, están fabricados con materiales relativamente frágiles, que hacen desaconsejable su manipulación constante (Figura 2.2).

Además, aunque lo más conveniente sería el estudio presencial de estos objetos, es necesario que lo más importante desde el punto de vista pedagógico es conocer sus características.

Este departamento cuenta también en sus instalaciones con un laboratorio y el Museo Arqueológico Antonio Ballesteros, en el que están expuestos muchos de los objetos arqueológicos mencionados anteriormente.

Como es lógico, muchos de estos objetos tienen un elevado valor tanto económico como arqueológico, por lo que el acceso a los mismos es bastante limitado. De hecho, en muchas ocasiones se requiere la presencia simultánea de los profesores y los estudiantes en el laboratorio en el que se están desarrollando los estudios. Obviamente, esto supone un inconveniente a la hora de dar una formación de calidad a los estudiantes.

Para solucionar este problema, surge el sistema Chasqui, cuyo fin principal era permitir a los profesores, investigadores y estudiantes del Departamento de Historia de América II el acceso *online* al material del museo.

Sin embargo, con el desarrollo del proyecto, el sistema se ha encargado de dar también acceso a otros objetos de los que dispone el departamento: los existentes en el laboratorio, además de documentos, informes técnicos y gráficos producidos por profesores, investigadores y alumnos del departamento.

En un principio, la aplicación iba a ser un museo virtual anexo al real, pero el desarrollo del proyecto la ha transformado en una herramienta útil no sólo para visualizar y manejar la información sobre los objetos del mismo, sino para establecer relaciones entre ellos. De este modo, podemos decir que Chasqui no es un simple contenedor de datos ni el resultado de virtualizar el museo, sino que va más allá, dando lugar, mediante la posibilidad de crear estas relaciones, a generar conocimiento.

De hecho, a través de las relaciones entre varios objetos determinados, se puede llegar a construir un modelo o descripción objetiva de la cultura en la cual se creó dicho objeto.

Una de las características de la investigación arqueológica de un objeto es que ésta no se suele dar nunca por cerrada. Por ello, el sistema Chasqui debía dar posibilidad de acceder a los datos y modificarlos de forma constante.

Además, dado que el usuario (en este caso el estudiante) no puede acceder al objeto real de forma física, se hizo necesario que Chasqui contase con un acceso hipermedia lo más intuitivo posible.

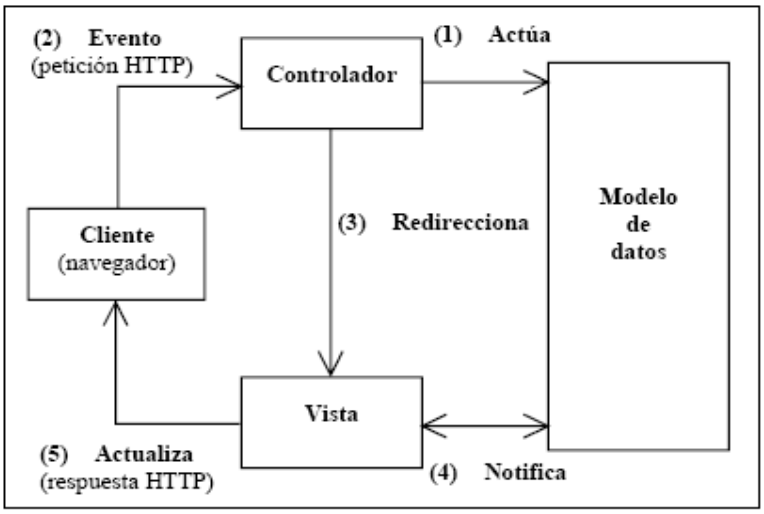


Figura 2.3. *Uso del patrón MVC en el sistema Chasqui*

Entre los diversos sistemas de representación hipermedia considerados, se eligió Pipe por ser la opción más equilibrada de acuerdo a las necesidades de esta versión de Chasqui. Se trata de un modelo que tiene ideas en común con el modelo Dexter. Este modelo está basado en la separación de los contenidos y relaciones navegacionales (*grafo de contenidos*) de la interfaz de usuario e interpretación de las relaciones que se les va a proporcionar (*esquema navegacional*), estando estos dos aspectos, así como las relaciones existentes entre ellos, soportados por una notación visual. También es conveniente tener en cuenta que el modelo Pipe cuenta con una representación en formato XML, y que las caracterizaciones de aplicaciones hipermedia en este formato son interpretadas por una aplicación Java (*GAP*) que se encarga de la generación rápida de prototipos.

Otro requisito importante, también desde el punto de vista del usuario alumno, es el de que el sistema sea accesible desde cualquier máquina con conexión a Internet. Por ello, se optó por realizar un desarrollo basado en un entorno web. La arquitectura de la aplicación es del tipo cliente-servidor y está basada en un navegador estándar y en servidores http (*HyperText Transfer Protocol*).

Así, en el lado del cliente tenemos un navegador estándar que interpreta un documento e interacciona con el usuario utilizando HTML y JavaScript. En el lado del servidor es donde se gestionan y procesan los datos mostrados al usuario utilizando diferentes lenguajes y/o SGBDs (sistemas de gestión de bases de datos) como son, en este caso, PHP y MySql.

Además, para evitar el acoplamiento, se optó por usar el Modelo Vista Controlador en su versión 2 (MVC 2), especialmente definida para desarrollos web (Figura 2.3).

Chasqui se basa en el concepto de objeto virtual, es decir, un objeto digital formado por:

- un conjunto de datos que describen las características físicas y/o conceptuales del objeto que se está modelando;
- un conjunto de metadatos (datos de los datos) que describen y clasifican el objeto desde el punto de vista educativo.
- un conjunto de recursos, consistentes en representaciones virtuales del objeto.

2.3.2. Sistema MIGS

El Museo de Informática José García Santesmases está situado en la tercera planta de la Facultad de Informática de la Universidad Complutense de Madrid. Recibe su nombre en honor al catedrático de la Facultad de Ciencias de la Universidad de Madrid, antecesora de esta universidad.

Esta institución alberga un gran número de máquinas, algunas de las cuales fueron desarrolladas en las instalaciones de la propia universidad. Otras, en cambio, son ordenadores comerciales que desarrollaron su vida útil en el Centro de Cálculo de esta universidad. Además, también hay otros objetos relacionados con la historia de esta ciencia, como manuales de programación y de uso de estos computadores.

Se muestran de forma cronológica, siendo explicadas sus características y modo de funcionamiento mediante paneles explicativos.

En cuanto al sistema informático, su objetivo principal es a grandes rasgos el mismo que el del que se usó para virtualizar el Museo de Arqueología Antonio Ballesteros: digitalizar el Museo y convertirse en una herramienta de ayuda en la enseñanza e investigación en informática (Figura 2.4).

También hay que destacar que, por diversas razones como la falta de espacio, existe una cantidad importante de objetos que no se pueden mostrar al público. En este aspecto, la existencia de un sistema que permita su virtualización y puesta a disposición del público puede resultar muy interesante, puesto da más valor y más posibilidades a la visita virtual al museo que a la real, al ofrecer más piezas.

Sin embargo, la filosofía que mueve este proyecto es la misma que la del proyecto Chasqui.

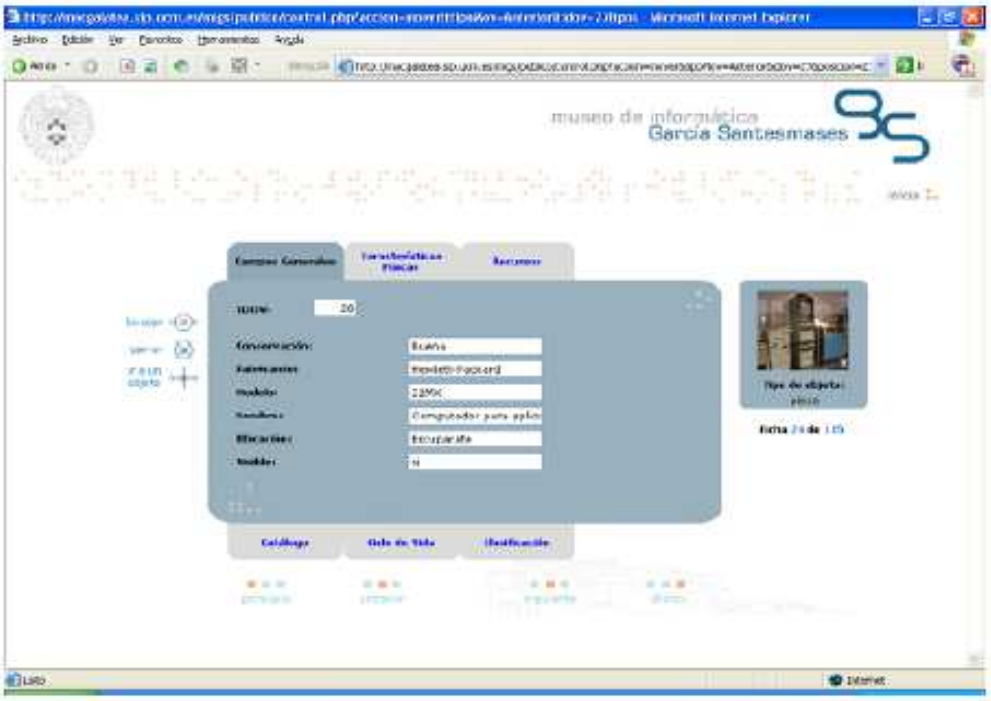


Figura 2.4. Interfaz del sistema MIGS 0.0

Una de las novedades de este sistema es la opción que ofrece de exportar contenido del sistema a otros entornos empaquetando los objetos, para lo cual utiliza el modelo de agregación de objetos propuesto por IMS-CP (IMS Content Packaging Information Model), que fue definido inicialmente para objetos de aprendizaje y ha sido adaptado para objetos virtuales. Un paquete IMS integra todos los recursos que componen el objeto.

Al mismo tiempo, un paquete IMS posee un documento XML denominado *manifiesto*, que incluye los metadatos que describen el objeto y las dependencias entre los elementos que lo componen. Este manifiesto puede usarse como base del desarrollo de los mecanismos de validación encargados de mantener la coherencia entre los objetos y de la información contenida en el museo.

Además, el empaquetado de objetos posibilita la importación y exportación de los objetos virtuales, bien para ser consultados como objetos individuales o bien para ser integrados en otros sistemas de aprendizaje.

Entre las funciones relacionadas con el empaquetado y desempaquetado de objetos virtuales, se han definido las siguientes, inspiradas, como ya hemos dicho, en la especificación IMS:

- Selección de los contenidos que componen un objeto virtual, y análisis de las dependencias entre los recursos de un objeto y los de diferentes objetos.

- Selección de los metadatos necesarios para etiquetar los contenidos elegidos que serán incluidos en el objeto virtual.
- Empaquetado del objeto virtual.

2.3.3. Sistema Chasqui II

Con las lecciones aprendidas en los dos proyectos de virtualización anteriormente mencionados, y con la colaboración de Telefónica I+D, se llevó a cabo un proyecto llamado Chasqui II, cuyo fin era extender los desarrollos anteriores a escenarios más amplios **[7]**.

Es necesario tener en cuenta que las aplicaciones de las que ya hemos hablado fueron diseñadas e implementadas como soluciones a medida, aunque guarden cierta relación entre sí. Tienen en cuenta los problemas que surgen en el museo para el que fueron diseñadas, pero su adaptabilidad para otros casos es bastante pequeña.

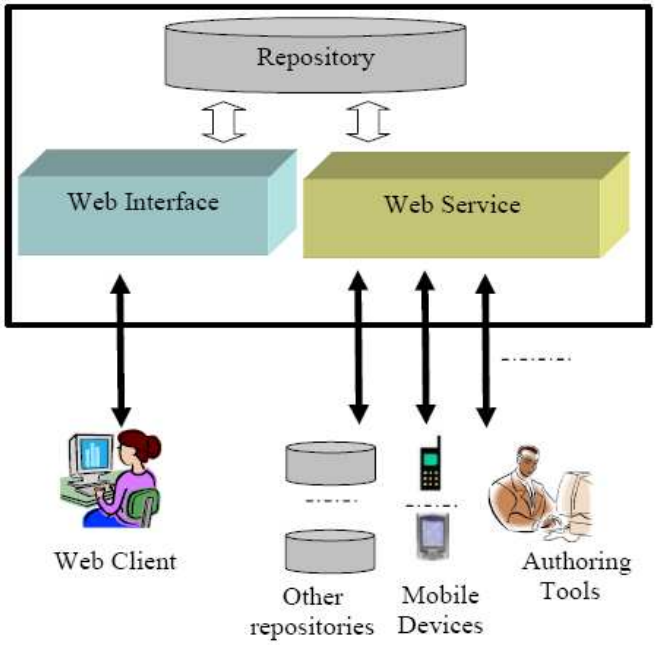


Figura 2.5. Mecanismos de acceso al repositorio de objetos virtuales

Una de las novedades que ofrece Chasqui II es la de emplear servicios web, usando interfaces que integran aplicaciones y servicios de forma neutral, gracias al uso de estándares XML como SOAP (*Simple Object Access Protocol*). De este modo, los servicios web dan la posibilidad de subir objetos al sistema, actualizarlos, eliminarlos o buscarlos de acuerdo a unas características determinadas de sus datos o de sus metadatos.

Es destacable que las interfaces facilitan la utilización de mecanismos de acceso alternativo, entre los cuales llaman la atención, por el potencial que dan, las herramientas de autoría o los basados en dispositivos móviles.

Además, hay que tener en cuenta que el interfaz web puede ser ampliado para que cuente con nuevos servicios y funcionalidades, de modo que se pueda tener un nivel de integración mayor entre Chasqui II y otros repositorios o aplicaciones finales.

Del mismo modo que en el caso del Museo de la Informática García Santesmases, los objetos se empaquetan siguiendo el modelo propuesto por IMS.

Precisamente, una de las herramientas de autoría es IMSCP_UCM (Figura 2.6), que extiende esta estructura para permitir:

- **Creación de objetos, y su inclusión en repositorios.** Esto implica la definición de los datos, los metadatos y los recursos, ya sean internos o externos.
- **Recuperación y eliminación de los objetos.** Se direccionan mediante sus identificadores, y para el borrado hay que tener en cuenta las dependencias entre ellos.
- **Visualización de los objetos.**
- **Empaquetamiento de los objetos.** Los paquetes son ficheros de tipo *zip*, que contiene como ficheros tanto sus recursos como su *manifiesto*, así como otros objetos virtuales relacionados de forma directa o indirecta con el objeto que está siendo empaquetado.

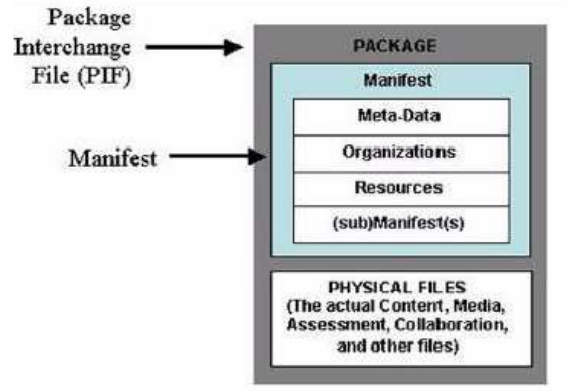


Figura 2.6. Estructura de IMSCP_UCM

Otra de las ventajas que ofrece IMSCP_UCM es la posibilidad de estar conectado simultáneamente a varios repositorios a través de la interfaz basada en servicios web.

2.3.4. MIGS 1.0

Desde septiembre de 2004 y hasta junio de 2005, se trabaja en una nueva versión del sistema MIGS, que se denomina MIGS 1.0. Su objetivo principal era adaptar el sistema existente a los nuevos requisitos, así como analizar y diseñar nuevas funcionalidades.

Todo este proyecto se llevó a cabo en el marco de la asignatura "Sistemas Informáticos", de 5º curso de Ingeniería Informática de la Universidad Complutense de Madrid, y contó con las etapas de Iniciación, Reingeniería e Ingeniería.

En la primera de ellas, las labores principales fueron el estudio de la aplicación existente anteriormente (MIGS 0.0) y la detección de errores en la misma. La segunda se centró en la corrección de los errores detectados en la fase anterior, y se desarrolló por capas del patrón "*Modelo-Vista-Controlador*". La tercera es la etapa de Ingeniería, durante la que se aplicaron nuevas mejoras encaminadas a cambiar o extender la herramienta.

En cuanto a la tecnología, el servidor web elegido fue Apache 1.3.31, el sistema gestor de bases de datos que almacenaría la información relativa a los objetos fue MySQL 4.0.17, y el lenguaje de desarrollo, PHP 4.3.3. Se trata de una configuración algo complicada, y que puede provocar problemas durante su instalación.

Capítulo 3. El sistema iChasqui

3.1. Modelo conceptual

En el marco de los sistemas Chasqui y MIGS que se han comentado anteriormente, y siguiendo una de las líneas de investigación del Departamento de Ingeniería del Software e Inteligencia Artificial de la Facultad de Informática (que anteriormente formaba parte del extinto Departamento de Sistemas Informáticos y Programación), se comienza en octubre de 2007 a desarrollar la presente aplicación.

El objetivo principal que se persigue con esta aplicación es que el usuario pueda virtualizar objetos, dejándolos disponibles en el sistema, de forma totalmente independiente del contexto.

El programa da la posibilidad de especificar, para cada objeto, una serie de atributos y recursos. Los primeros son características del objeto, mientras que los segundos son archivos relacionados con el mismo (libros electrónicos, fotografías, archivos de audio y video, ...). En este sentido, los recursos son simplemente almacenados en el sistema, por lo que se acepta cualquier tipo de fichero.

3.1.1. Atributos

Los atributos se pueden organizar en estructuras arborescentes: un atributo puede tener ninguno, uno o varios atributos hijos (Figura 3.1). Así, por ejemplo, podríamos tener un atributo *fecha*, que a su vez tuviese como hijos a los atributos *día*, *mes* y *año*.

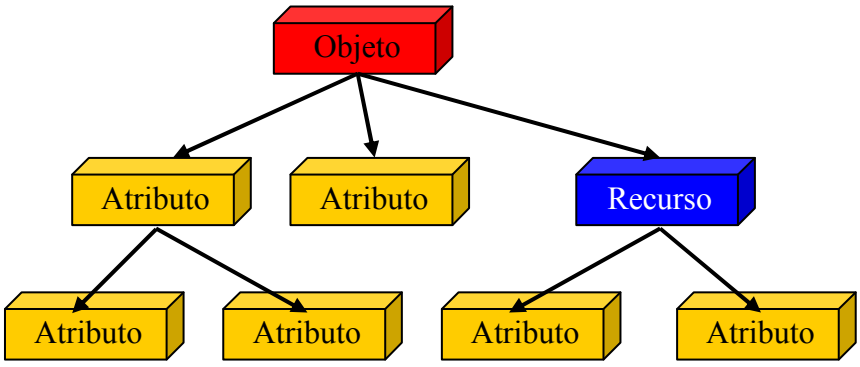


Figura 3.1. Estructura en árbol de un objeto virtual

Además, hay que tener en cuenta que un recurso puede tener atributos que lo describan, que también pueden ser libremente diseñados por el usuario. Por ejemplo, pueden

informar de quién es el autor, de la fecha en la que se generó el archivo, ... También estos recursos pueden organizarse en forma de árbol.

Como se puede deducir, la aplicación puede ser utilizada para virtualizar un museo, como los ejemplos desarrollados anteriormente por el Departamento, o para servir de apoyo a un curso impartido de forma virtual o presencial. Los objetos no tienen por qué hacer referencia a elementos físicos, sino que pueden corresponder a elementos abstractos: conceptos, ideas, temas, ...

3.1.2. Tipos de recursos

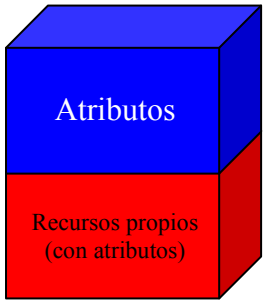


Figura 3.2. Estructura de un objeto virtual

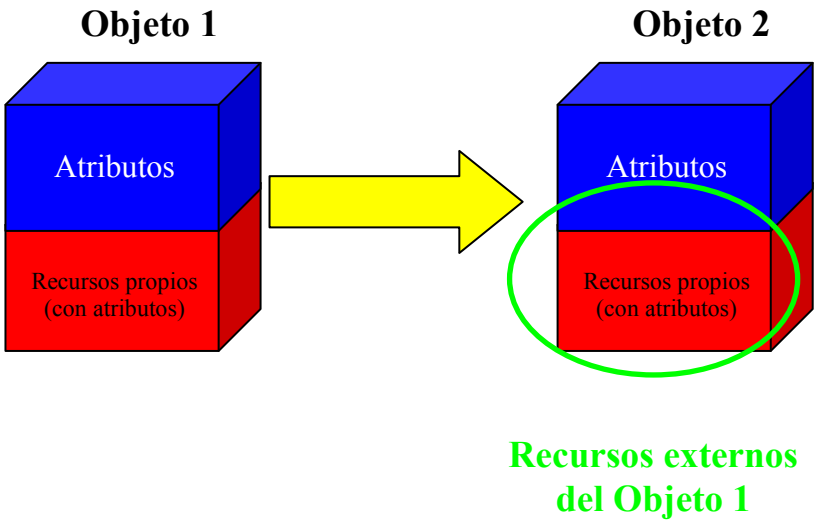


Figura 3.3. Recurso externo

Como ya se ha mencionado anteriormente, un recurso puede ser un archivo de cualquier tipo. Sin embargo, si atendemos a la clasificación con respecto al objeto al cual pertenece, encontramos tres tipos diferentes.

En primer lugar, tenemos el recurso propio, que es el que pertenece de forma directa a un objeto (Figura 3.2).

En segundo lugar, tendríamos el recurso externo, que es el que es recurso propio de otro objeto (Figura 3.3).

Por último, está el recurso objeto, que es un objeto virtual que es recurso de otro objeto virtual (Figura 3.4).

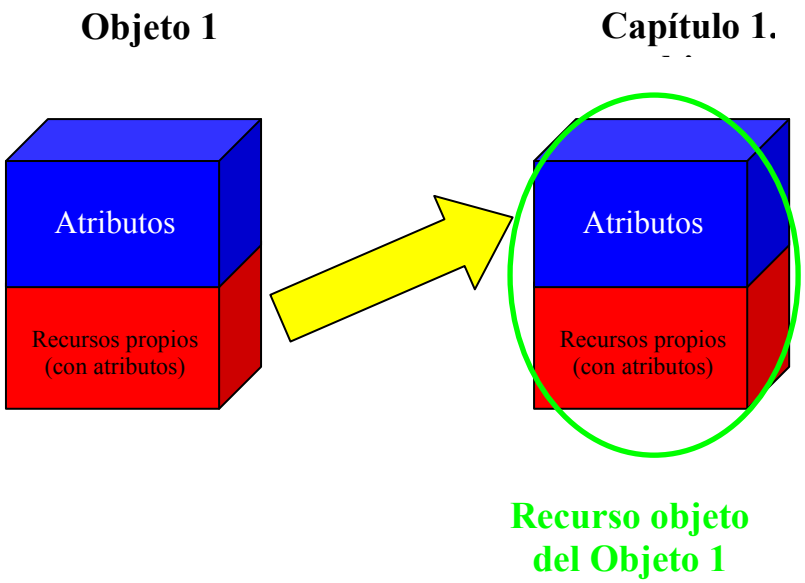


Figura 3.4. Recurso objeto

3.2. Arquitectura

La arquitectura de la aplicación está organizada en tres capas, cada una de las cuales cumple una función y posee unas características determinadas (Figura 3.5).

- **Capa de datos:** es la que tiene como misión proveer de datos a las demás capas, y dar persistencia a los que recibe de éstas. Además, debe guardarlos de forma estructurada, para que sea posible recuperarlos.
- **Capa de proceso:** como su nombre da a entender, es la que procesará los datos que reciba de una de las otras capas, de modo que sea posible dárselos a una tercera capa. Por ejemplo, los que recibe de la capa de datos para ser presentados al usuario a través de la capa de interacción con el usuario, o los que vienen de ésta para ser almacenados en la de datos, o para recuperar otros de la misma.
- **Capa de interacción con el usuario:** es la encargada de dar al sistema la oportunidad de manejar el sistema; en una palabra, es su interfaz. En ella, el usuario introducirá los datos con respecto a los que quiere hacer una consulta, o los que desea almacenar.

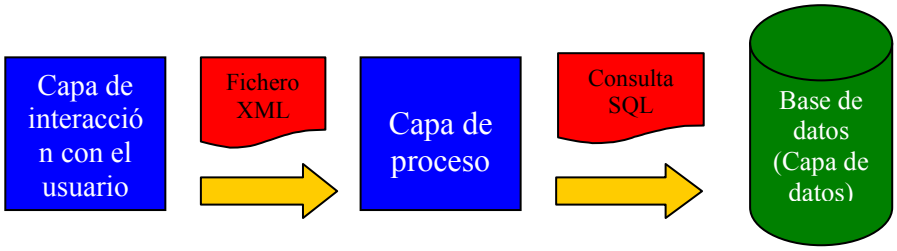


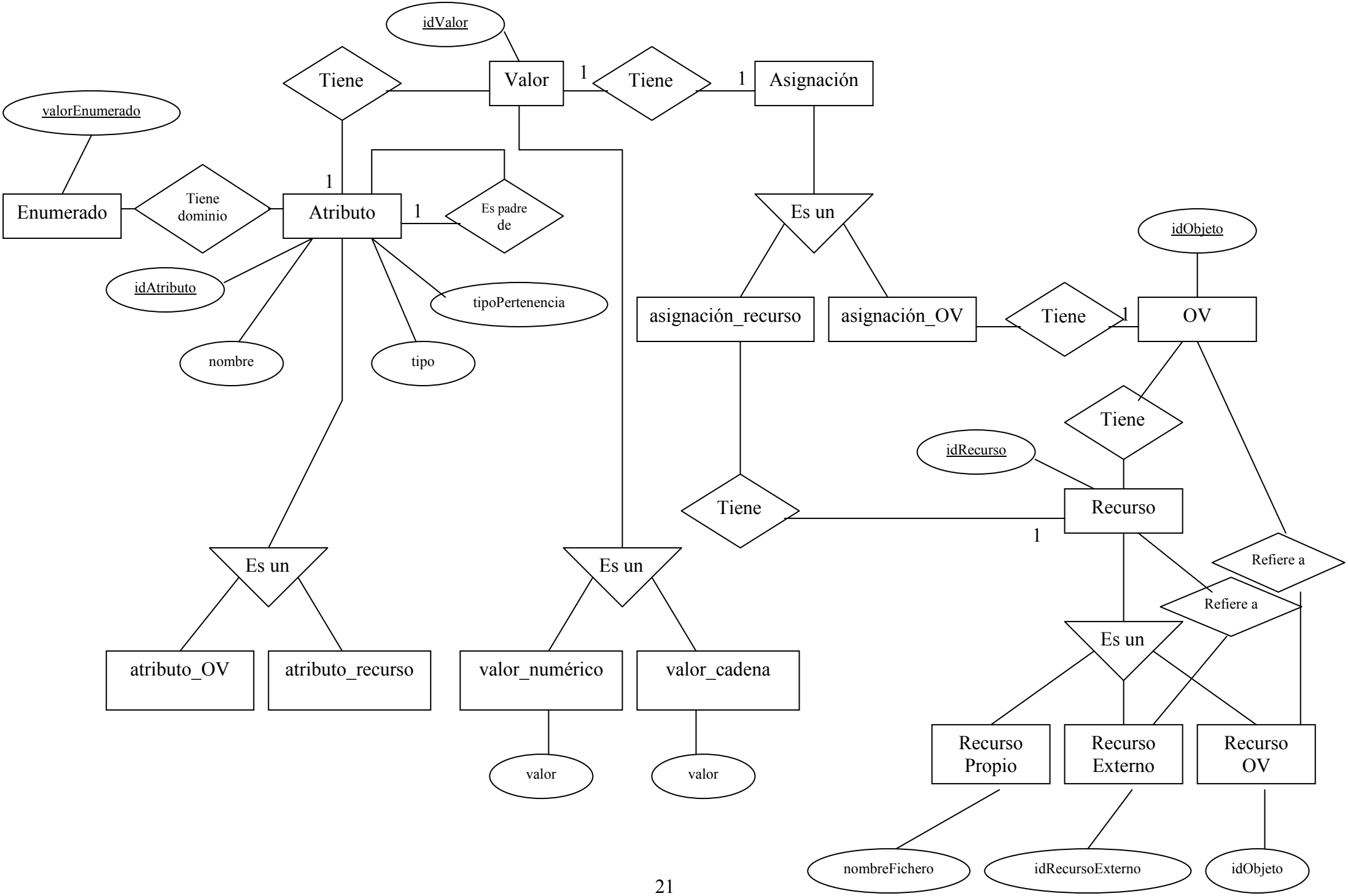
Figura 3.5. Capas del sistema

3.2.1. Arquitectura de la capa de datos

Consiste en una base de datos cuya estructura viene determinada por el diagrama entidad – relación adjunto en este punto.

Observando dicho diagrama, podemos extraer diversas conclusiones:

- un atributo pertenece a un recurso o a un objeto;
- un atributo puede ser padre de otro atributo;
- un valor es de tipo numérico o de tipo cadena de caracteres (“cadena”, para simplificar);
- un valor está asignado una única vez, y lo estará a un recurso o a un objeto;
- un objeto puede tener ninguno, uno o varios recursos;
- un recurso puede ser de tipo propio, externo u objeto virtual;
- un recurso propio es un archivo que pertenece a un objeto virtual, sin hacer referencia a otras entidades; por ello, la base de datos guardará únicamente este dato;
- un recurso objeto es un objeto referenciado por otro objeto como recurso suyo, lo que se indica mediante una relación en el diagrama entidad-relación;
- un recurso externo es un recurso de un objeto que es referenciado por otro objeto como recurso suyo, lo que se indica – al igual que en el caso anterior - mediante una relación en el diagrama entidad-relación.


idValor
Tiene
Valor
1
Tiene
1
Asignación
valorEnumerado
1
Enumerado
Tiene dominio
Atributo
1
Es padre de
Es un
idObjeto
idAtributo
tipoPertenencia
asignación_recurso
asignación_OV
Tiene
1
OV
nombre
tipo
Tiene
idRecurso
Tiene
Recurso
1
Es un
Es un
Refiere a
Refiere a
Es un
atributo_OV
atributo_recurso
valor_numérico
valor_cadena
Recurso Propio
Recurso Externo
Recurso OV
valor
valor
nombreFichero
idRecursoExterno
idObjeto

3.2.2. Arquitectura de la capa de proceso

Esta capa es la encargada de procesar – como su propio nombre indica – la información que recibe de la capa de interacción e interaccionar, en base a ella, con la capa de datos. Es decir, efectuar las correspondientes consultas a la base de datos.

Tras ésto, deberá generar la información necesaria para que la capa de interacción pueda presentar al usuario el resultado de la operación.

3.2.3. Arquitectura de la capa de interacción

La capa de interacción tiene como misión interaccionar con el usuario, permitiéndole que introduzca los datos necesarios en función del caso de uso particular, y que pueda ver los resultados de la operación tras ser realizada.

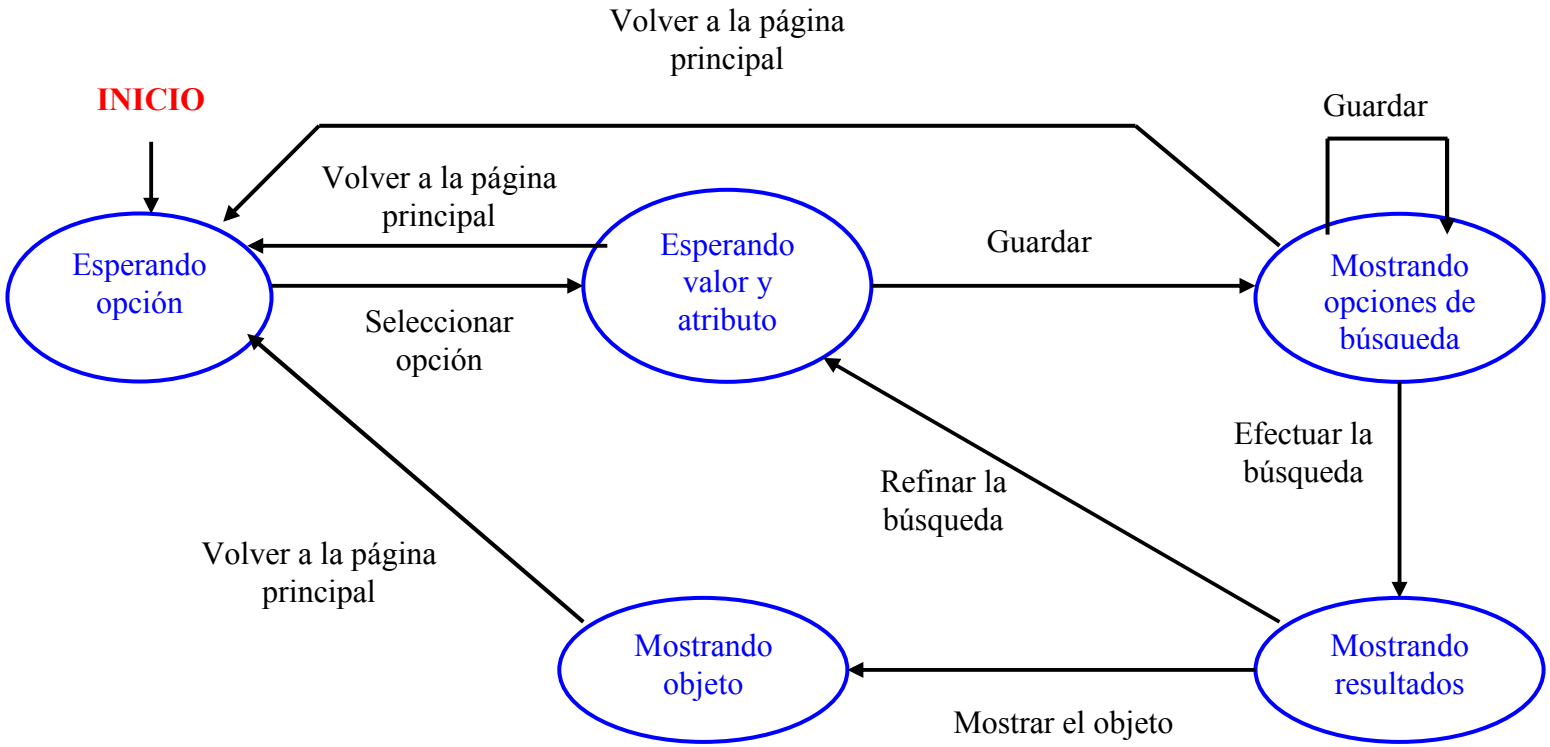
Puesto que esta parte es la destinada a recoger y mostrar datos, se usa bastante HTML en combinación con PHP, para la creación de formularios así como para la presentación de los datos.

A continuación se incluyen los diagramas de interacción que corresponden a los distintos casos de uso de la aplicación.

3.2.3.1. Buscar objetos por valores de sus atributos

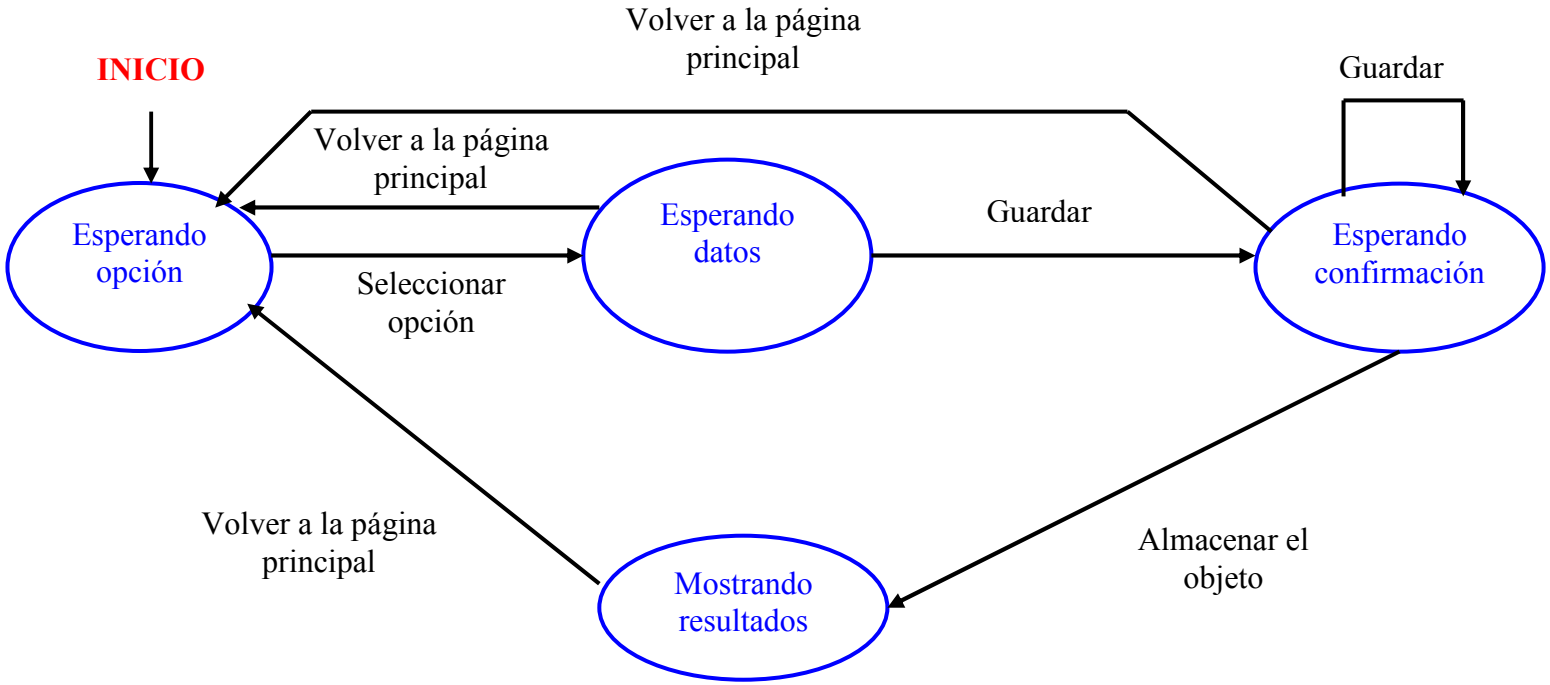
Si el usuario selecciona la opción necesaria en la pantalla principal, se le mostrará un interfaz en el que podrá seleccionar los atributos y valores para hacer la búsqueda. Una vez hecho esto, si guarda las opciones que ha seleccionado, se le mostrarán para que pueda modificarlas antes de hacer la búsqueda. Una vez que efectúa la búsqueda, la puede refinar o visualizar uno de los objetos obtenidos en la búsqueda.

Desde todas las pantallas, el usuario puede volver a la pantalla principal (donde le sistema espera la elección de una opción), con lo cual la operación quedará cancelada (esto ocurrirá también en el resto de los casos).



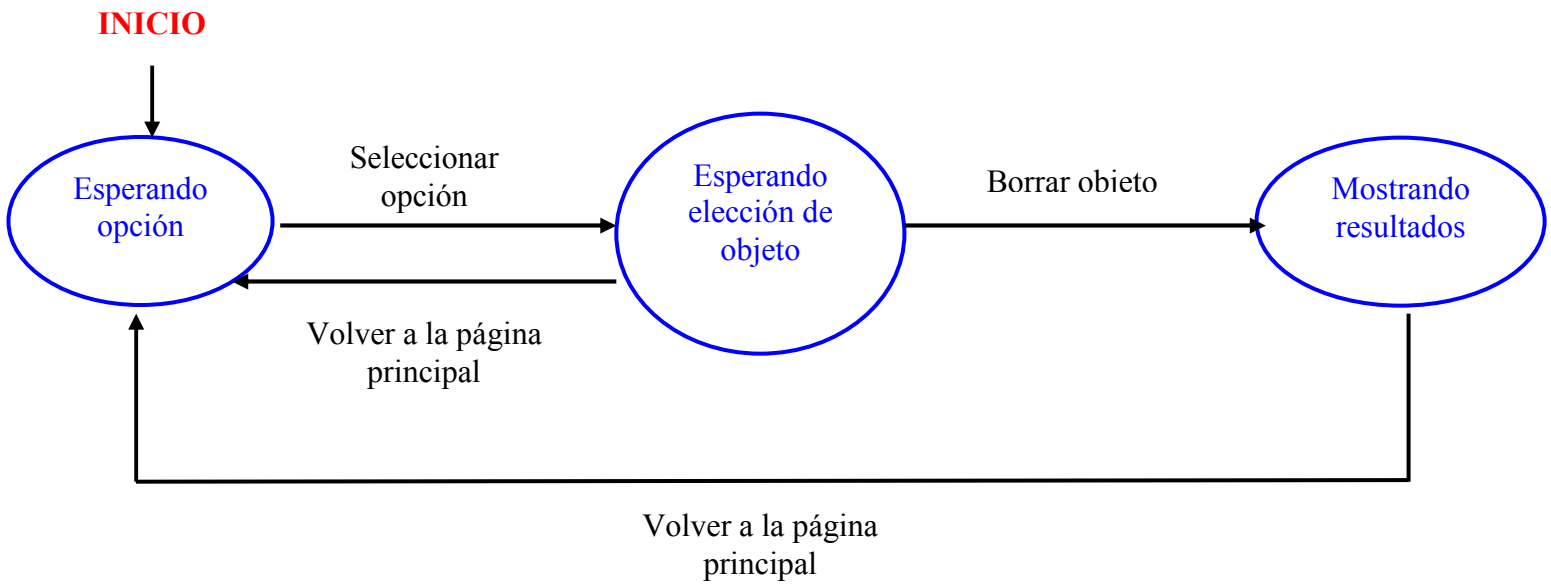
3.2.3.2. Insertar un objeto o un atributo

En la página principal, el usuario de la aplicación deberá solicitar la operación correspondiente. Después, se le mostrará un interfaz en el que podrá insertar los datos del nuevo objeto o atributo (esperando datos). El usuario pulsará el botón guardar, que hará que el sistema guarde las preferencias y las muestre por pantalla, esperando la confirmación para efectuar la inserción. Cuando la reciba, almacenará el objeto en la base de datos, y el usuario podrá ver el resultado de la operación por pantalla.



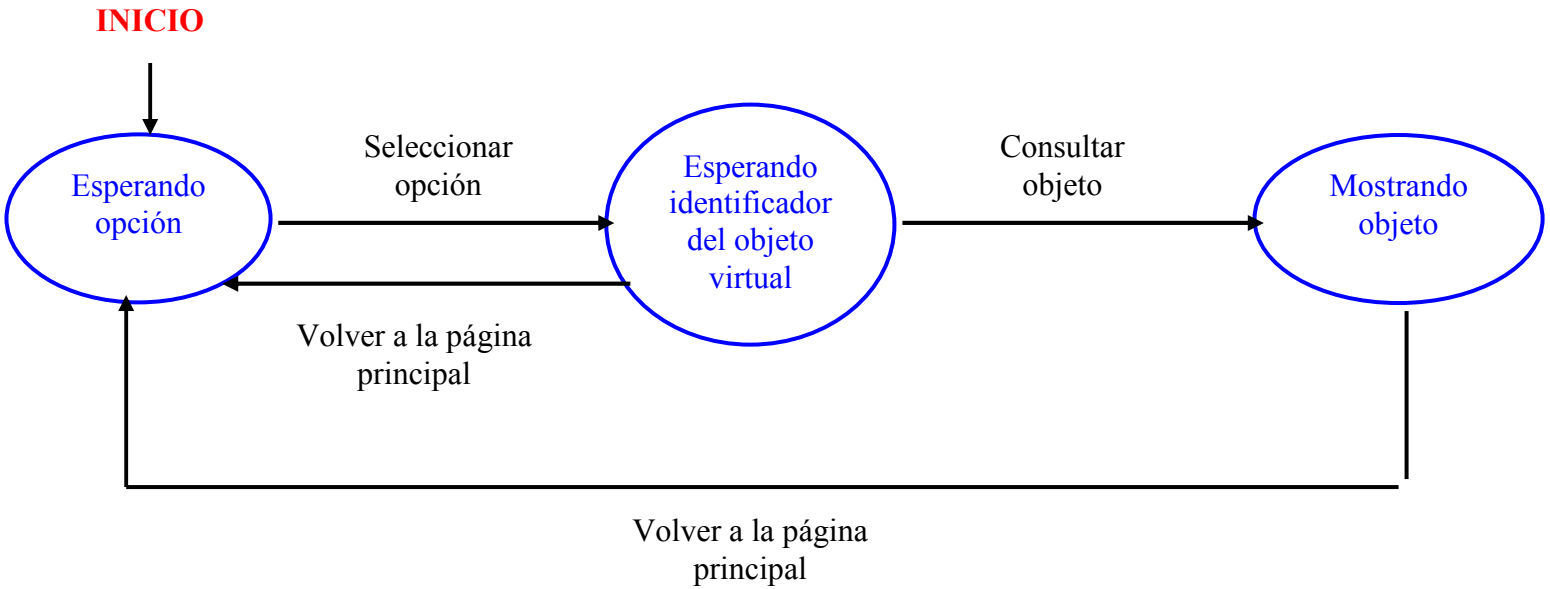
3.2.3.3. Borrar un objeto virtual

Si el usuario selecciona esta opción, se le muestra una pantalla desde la que podrá insertar el identificador del objeto que quiera eliminar. Una vez que lo introduzca y pulse el botón, el sistema tratará de eliminarlo de la capa de datos. Si lo consigue, le informará de que ha tenido éxito en la operación; y en caso contrario se le informará de ello, indicándole el motivo del error.



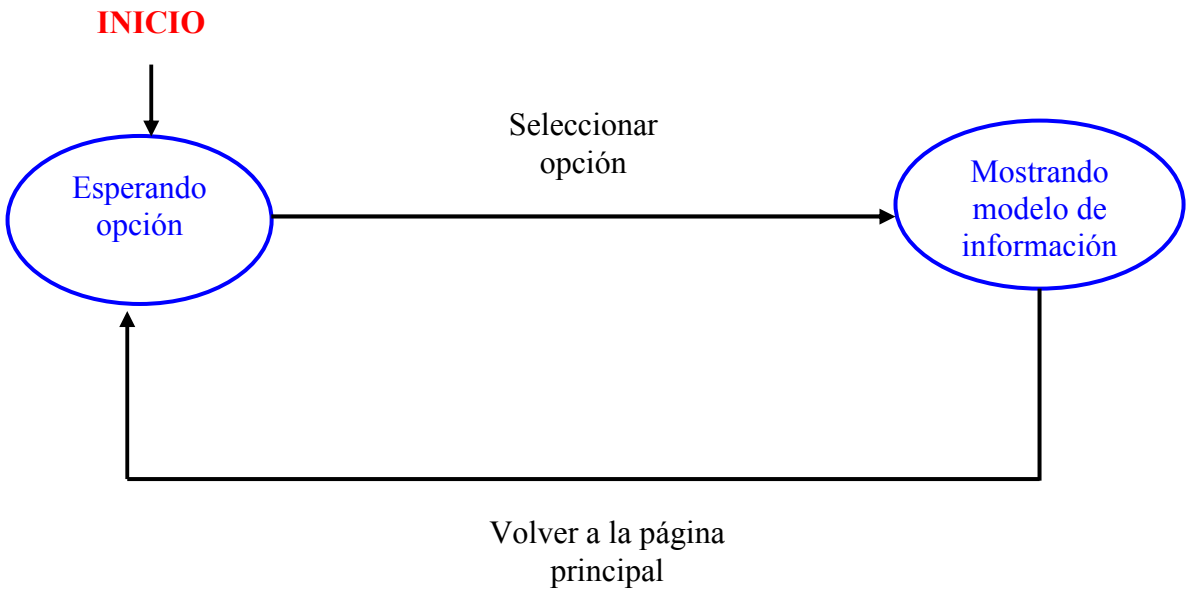
3.2.3.4. Consultar un objeto virtual

El usuario elige la opción de consultar un objeto virtual, con lo que se le redirige a un interfaz desde el cual puede introducir el identificador del objeto a consultar. Una vez introducido, se le muestra el objeto (los atributos con sus valores y enlaces a los recursos).



3.2.3.5. Mostrar el modelo de información de objetos virtuales y recursos

Si el usuario selecciona una de estas opciones, se le muestra directamente el modelo de información (atributos disponibles).



3.3. Implementación

3.3.1. Tecnologías utilizadas

3.3.1.1. PHP

PHP es un lenguaje de programación interpretado, que fue diseñado originalmente para la creación de páginas web dinámicas (Figura 3.6). En la actualidad, suele usarse para programar aplicaciones web desde el lado del servidor, siendo uno de los lenguajes para la web más extendidos **[10]**.

Inicialmente, PHP significaba *P*ersonal *H*ome *P*age Tools, pero con el tiempo, ésta definición se incluyó en el acrónimo ***P****HP* ***H****ypertext* ***P****re-processor*.

Fue creado en 1994 por Rasmus Lerdof, aunque la implementación principal es producida por The PHP Group. Esta implementación constituye un estándar de facto, pues por el momento este lenguaje no dispone de una especificación formal.

Una de las características principales de PHP es la posibilidad que ofrece de insertar código embebido de este tipo en páginas HTML. Para ello, el código se introduce dentro de unas etiquetas determinadas. De este modo, el servidor web en el que esté una aplicación en PHP tomará como entrada el código PHP, y generará una página web como salida.

Otra de las ventajas de PHP es que, al ser bastante parecido a lenguajes de programación estructurada muy parecidos como Perl o C, es bastante fácil de aprender para los programadores que ya conocían alguno de éstos.

Por otro lado, hay que destacar que PHP permite hacer aplicaciones orientadas a objetos.

Incluso, existe una versión llamada PHP CLI (*Command Line Interface*), que permite usarlo desde la línea de comandos. También hay extensiones que permiten la generación de archivos de distintos formatos.

En cuanto a su conexión a servidores de bases de datos, PHP ofrece la posibilidad de trabajar con varios, entre los que destacan MySQL, Postgres, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite. En general, destaca la combinación PHP – MySQL, muy utilizada en la actualidad en gran cantidad de sitios web.

Además, es posible ejecutar aplicaciones escritas en PHP en sistemas operativos tanto de tipo Windows como de tipo Unix. Es un lenguaje multiplataforma.

Entre los lenguajes que hacen la competencia a PHP, se encuentran Microsoft ASP y ASP.NET (que utiliza C#/VB.NET como lenguajes), ColdFusion de la compañía Adobe (antes Macromedia), aJSP/Java de Sun Microsystems, y CGI/Perl.

Se creó como sistema libre, aunque en este momento existen entornos de desarrollo integrado comerciales como Zend Studio, y CodeGear (la división de lenguajes de programación de Borland) ha empezado a comercializar recientemente **Delphi for PHP**

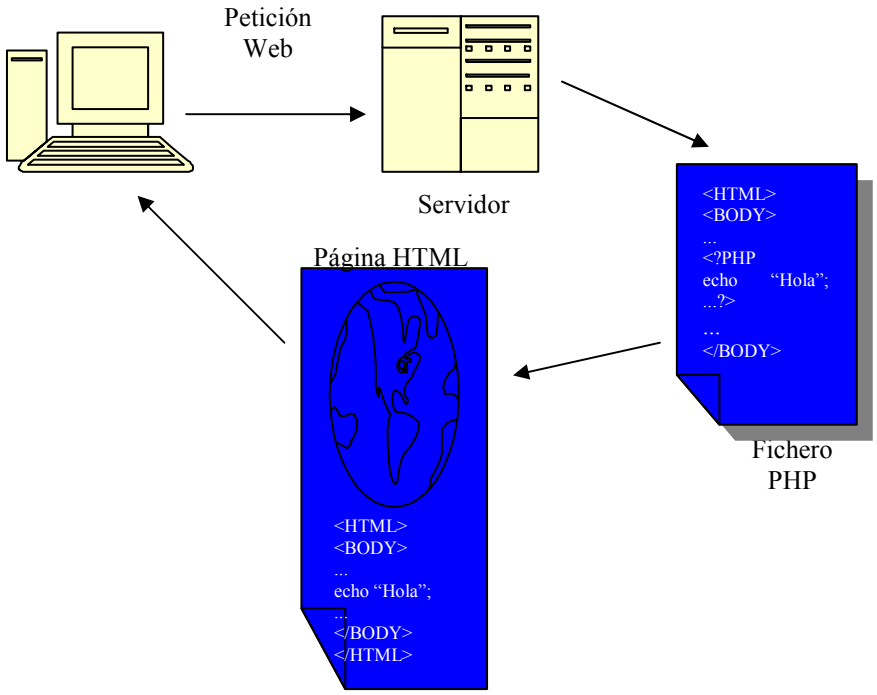


Figura 3.6. Esquema de petición de una web dinámica

3.3.1.2. MySQL

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario. Mayoritariamente, está desarrollado en ANSI C **[14]**.

Se ofrece bajo la licencia GNU GPL de software libre, aunque las empresas que lo quieran utilizar para sus productos comerciales pueden adquirir una licencia. Todo esto es debido a que el copyright de la mayor parte del código es propiedad de una empresa privada, MySQL AB. Creada por David Axmark, Allan Larsson, y Michael Widenius, desde enero de 2008 es una compañía subsidiaria de Sun Microsystems.

SQL ("*Structured Query Languaje*" o "*Lenguaje de Consulta Estructurado"*) es un lenguaje de acceso a bases de datos, que fue comercializado por primera vez en 1981 por IBM, el cual fue presentado a ANSI y desde ese entonces ha sido considerado como el estándar de facto para las bases de datos relacionales.

Desde 1986, han aparecido distintas versiones del estándar SQL , siendo las más conocidas: SQL:92, SQL:99, SQL:2003. MySQL es una idea originaria de la empresa

opensource MySQL AB establecida inicialmente en Suecia en 1995 y cuyos fundadores son David Axmark, Allan Larsson, y Michael "*Monty*" Widenius. El objetivo que persigue esta empresa consiste en que MySQL cumpla el estándar SQL, pero sin sacrificar velocidad, fiabilidad o usabilidad.

En la década de los 90, Michael Widenius intentó usar mSQL para conectar las tablas usando rutinas de bajo nivel ISAM, pero no resultó todo lo rápido ni flexible que era necesario. Por ello, decidió crear una API para SQL similar a mSQL, pero más portable. Precisamente, para la portabilidad usa GNU Automake, Autoconf, y Libtool.

En la actualidad, la gran mayoría de los lenguajes ofrece APIs para incluir en su código llamadas a MySQL: C, C++, C#, Pascal, Delphi (via dbExpress), Eiffel, Smalltalk, Java (con una implementación nativa del driver de Java), Lisp, Perl, PHP, Python, Ruby,Gambas, REALbasic (Mac), FreeBASIC, Tcl, ...

Es un sistema gestor de bases de datos muy rápido en la lectura cuando utiliza el motor no transaccional MyISAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. Resulta muy apropiado para aplicaciones web, puesto que en este tipo de sistemas hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones.

Uno de los principales campos de aplicación de MySQL en la actualidad está en las aplicaciones web. De hecho, es bastante común encontrarlo en combinación con PHP, como ya se ha dicho anteriormente.

3.3.1.3. Apache

La labor de un servidor web es analizar cualquier archivo solicitado por un navegador y mostrar el resultado correcto en función del código del archivo.

Apache es uno de los servidores Web más extendidos actualmente. Es libre, es decir, se trata de código abierto. Al igual que PHP, es multiplataforma, pues puede funcionar tanto con Unix como con Windows y Macintosh **[15]**.

En 1995 empezó su desarrollo, que estuvo basado inicialmente en el código del servidor Web NCSA HTTPD 1.3, pero más tarde sería totalmente reescrito. Su nombre procede de la tribu Apache, última en rendirse al Gobierno de los Estados Unidos. Behelendorf lo eligió porque daba imagen de firmeza y energía pero no agresividad. También hay que tener en consideración que Apache consistía únicamente en un conjunto de parches para el servidor NCSA, por lo que se decía que era, en inglés, *a patchy server* (un servidor "*parcheado*").

El servidor Apache se desarrolla dentro del proyecto HTTP Server (HTTPD) de la Apache Software Foundation.

Entre las prestaciones de Apache destacan los mensajes de error altamente configurables y las bases de datos de autenticación y negociado de contenido, aunque en un primer momento fue criticado por la falta de una interfaz gráfica que ayudara en su configuración.

En cuanto a la seguridad, hay que tener en cuenta que la mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales y no remotamente. A pesar de ello, algunas se pueden accionar remotamente en ciertas situaciones, o aprovechar por los usuarios locales malévolos en las disposiciones de recibimiento compartidas que utilizan PHP como módulo de Apache.

3.3.1.4. XML

XML proviene de la expresión inglesa ***Ex**tensible **M**arkup **L**anguage* o "*lenguaje de marcas extensible*". Se trata de un metalenguaje extensible de etiquetas definido por el *World Wide Web Consortium* (*W3C*) **[18] [19]**.

Basado en SGML, lo simplifica y lo adapta, definiendo la gramática de lenguajes específicos. Por eso decimos que XML realmente no es un lenguaje, sino una manera de definir lenguajes; es decir, un lenguaje para lenguajes o metalenguaje. Como ejemplo, cabe destacar que XHTML, SVG, MathML fueron definidos en XML.

Propuesto como un estándar para el intercambio de información estructurada entre diferentes plataformas, alcanza por ello su mejor campo de utilización en Internet.

Encontramos su origen en el lenguaje inventado por IBM en los años setenta, llamado GML (*Generalized Markup Language*), cuya finalidad era ayudar a la empresa a almacenar de forma estructurada grandes cantidades de información. En 1986 la ISO comenzó a trabajar con el fin de normalizarlo, creándose SGML (*Standard Generalized Markup Language*). A partir de él se crearían muchos sistemas para almacenar información.

En el año 1989 Tim Berners Lee al crear la web, definió HTML a partir de SGML. Este lenguaje se definió en el marco de SGML y fue de lejos la aplicación más conocida de este estándar.

Sus ventajas principales son:

- Es extensible, lo que quiere decir que una vez diseñado un lenguaje y puesto en producción, se puede extender con nuevas etiquetas.
- El analizador es estándar, no es necesario crear uno específico para cada lenguaje. Así, el desarrollo de la aplicación es mucho más rápido.

- Si un tercero decide usar un documento creado en XML, le resultará relativamente sencillo entender su estructura y procesarlo.

Un documento XML contendrá:

- **Prólogo:** no es obligatorio. Describe, entre otras cosas, la versión XML y el tipo de documento.
    - **Declaración XML:** es la sentencia que declara al documento como un documento XML.Declaración de tipo de documento. Enlaza el documento con su DTD (definición de tipo de documento), o el DTD puede estar incluido en la propia declaración o ambas cosas al mismo tiempo.
    - **Comentarios e instrucciones de procesamiento.**
- **Cuerpo:** no es opcional en un documento XML. Debe contener un único elemento raíz, lo que es indispensable también para que el documento esté bien formado.

3.3.2. Organización de la implementación

La aplicación consiste en una serie de ficheros PHP, conectados a una base de datos MySQL, que será la encargada de dotar de persistencia a los objetos virtuales que los usuarios decidan almacenar en el sistema.

Figura 3.7. Comunicación entre capas a través de XML

Para la comunicación entre los distintos ficheros PHP, se utilizarán datos codificados en XML, que serán generados en el fichero de origen y parseados (analizados) en el de destino, donde se procesarán (Figura 3.7). Se elige esta tecnología por ser uno de los estándares más extendidos en el mundo de la Informática.

Para ello, se han utilizado dos *parsers* diferentes de los soportados por PHP: SAX y SimpleXML. Se ha desistido de emplear DOM, porque implica un coste computacional más elevado.

3.3.3. Implementación de la capa de datos

La base de datos consta de 16 tablas. En ellas se guarda de forma estructurada toda la información relativa a los objetos virtuales: atributos y dirección de los recursos (en el caso de ser propios) o se indica cuáles son (si son de tipo objeto o externos).

Entre las tablas que componen esta base de datos, algunas proceden del diagrama entidad-relación mientras que otras surgen como soporte, es decir, que se ha conocido su necesidad duran te el proceso de desarrollo.

De este modo surgen las tablas *idOVLibres*, *idValorLibres* e *idRecursoLibres*, que sirven respectivamente para almacenar los identificadores de objetos virtuales, valores y recursos cuando son eliminados, de forma que se puedan reutilizar para otros elementos de su clase. En caso de que la tabla correspondiente esté vacía, la aplicación generará un nuevo identificador.

En lugar de aparecer dos tablas para los atributos según el tipo de entidad a la que pertenecen sus valores, como correspondería según el diagrama entidad-relación, se ha optado por incluir una única tabla llamada atributo, en la que hay un atributo *tipoPertenencia*, que lo indicará.

En cambio, el resto de las tablas son consecuencia directa de la traducción del diagrama entidad-relación.

Las tablas de la base de datos son:

- **asignación_objeto:** relaciona cada valor con el objeto al que pertenece
    - **idObjeto**
    - **idValor**
- **asignación_recurso:** relaciona cada valor con el recurso al que pertenence
- **atributo:** almacena la información relativa a cada atributo que el usuario ha introducido en el sistema
    - **idAtributo**
    - **idValor**
- **dependencias_recursos:** guarda la dirección de los recursos que puede haber asociados a un recurso
    - **idRecurso**
    - **nombreFichero**
- **enumerado:** guarda los valores de tipo enumerado
    - **idAtributo**

    - **valorEnumerado**
- **idOvLibres:** guarda los identificadores de los objetos que han sido borrados por el usuario
    - **idObjeto**
- **idRecLibres:** guarda los identificadores de los recursos que han sido borrados por el usuario
    - **idRecurso**
- **idValorLibres:** guarda los identificadores de los valores que han sido borrados por el usuario
    - **idValor**
- **objeto:** almacena la información relativa a cada objeto
    - **idObjeto**
- **recurso:** guarda la información correspondiente a cada recurso
    - **idObjeto**
    - **idRecurso**
    - **tipo_recurso**
- **recurso_externo:** guarda la información correspondiente a cada recurso externo
    - **idRecurso**
    - **idRecursoExterno**
- **recurso_objeto:** guarda la información correspondiente a cada recurso objeto
    - **idRecurso**
    - **idObjeto**
- **recurso_propio:** guarda la información correspondiente a cada recurso propio
    - **idRecurso**
    - **nombreFichero**
- **valor:** relaciona cada valor con el atributo al que pertenece
    - **idValor**
    - **idAtributo**
- **valor_cadena:** almacena los valores de los atributos de tipo cadena
    - **idValor**

    - **valor**

- **valor_numérico:** almacena los valores de los atributos de tipo numérico
    - **idValor**
    - **valor**

En el sistema, cada objeto, atributo, valor y recurso está identificado de forma un identificador. Dependiendo del tipo de entidad de que se trate, el identificador empezará de una forma u otra:

- *OV** para objetos virtuales: *OV1*, *OV2*, ...
- *A** para atributos: *A1*, *A2*, ...
- *V** para valores: *V1*, *V2*, ...
- *R** para recursos: *R1*, *R2*, ...

3.3.4. Implementación de la capa de proceso

Para esta capa, se ha utilizado el lenguaje PHP, del que se ha hablado anteriormente. Esta capa recibe los datos que el usuario ha introducido en los formularios que forman parte de la capa de interacción.

Tras recibirlos, los procesa y en función de cuáles sean, lleva a cabo las consultas SQL necesarias a la base de datos. Y, una vez recibidos, muestra la información necesaria al usuario de nuevo a través de la capa de interacción.

Los datos se envían en formato XML, por lo que es necesario parsearlos para obtenerlos y poder usarlos. Al parsearlos, se guardan en variables del programa en PHP, para de este modo decidir qué consultas realiza e incluirlos en las mismas.

Por ejemplo, supongamos que tenemos un objeto virtual con las siguiente forma:

- **Identificador:** OV1
- **Atributos**
    - **Identificador:** A1 ; **Nombre:** Procedencia ; **Valor:** -
        - **Identificador**: A2 ; **Nombre:** Ciudad ; **Valor:** Elche
        - **Identificador:** A3 ; **Nombre:** Provincia; **Valor:** Alicante
    - **Identificador:** A4 ; **Nombre:** Pueblo ; **Valor:** Íberos
- **Recursos**
    - **Identificador:** R1 ; **Nombre del fichero:** elche.bmp

- **Atributos del recurso R1**
  - **Identificador:** A5 ; **Nombre:** Autor ; **Valor:** José López García

El usuario quiere consultar este objeto, por lo que introduce su identificador en el interfaz que le ofrece la capa de interacción, que llega a la de proceso de la siguiente forma:

```
<contenido>
  <objeto idObjeto="OV1" />
</contenido>
```

Esta información en formato XML viaja entre las capas en forma de variable de sesión, de la siguiente manera. En el fichero de salida:

```
$_SESSION["nombre de la variable"] = $informaciónEnXML;
```

y en el de destino:

```
$informaciónEnXML = $_SESSION["nombre de la variable"];
```

Al llegar a la capa de proceso, es analizada, obteniéndose el identificador del objeto virtual (en este caso, "*OV1*"). A partir de éste, se extrae de la base de datos todo el objeto que corresponde a ese identificador, que se codifica también en formato XML. Para el ejemplo anterior tendríamos:

```
<contenido>
<objeto idObjeto="OV1">
<listaAtributos>
<atributo idAtributo="A2">
  <nombre>Ciudad</nombre>
  <tipo>Cadena</tipo>
  <tipoPertenecia />
  <valor>Elche</valor>
  <hijos />
</atributo>
<atributo idAtributo="A3">
  <nombre>Provincia</nombre>
  <tipo>Cadena</tipo>
  <tipoPertenecia />
  <valor>Alicante</valor>
  <hijos />
</atributo>
<atributo idAtributo="A4">
  <nombre>Pueblo</nombre>
  <tipo>Cadena</tipo>
  <tipoPertenecia />
  <valor>Íberos</valor>
  <hijos />
</atributo>
</listaAtributos>
<listaRecursos>
<recurso idRecurso="R1" tipoRecurso="Propio">
<nombreFichero>elche.bmp</nombreFichero>
<listaAtributos>
<atributo idAtributo="A5">
  <nombre>Autor</nombre>
  <tipo>Cadena</tipo>
  <valor>José López García</valor>
  </atributo>
  </listaAtributos>
```

```
  </recurso>
  </listaRecursos>
 </objeto>
</contenido>
```

A su vez, en la capa de interfaz también deberá ser parseada esta variable, para obtener la información que será mostrada por pantalla.

Para hacer las consultas a la base de datos desde un fichero PHP, se utiliza un API formado por las siguientes funciones:

- **mysql_connect(*$nombreHost*,*$nombreUsuario*,*$contraseñaUsuario*):** sirve para que el usuario *$nombreUsuario* se conecte con el servidor de base de datos MYSQL en el host *$nombreHost*.
- **mysql_select_db(*$nombreBaseDeDatos*,*$descriptor*):** selecciona la base de datos cuyo nombre corresponde al valor de la variable *$nombreBaseDeDatos* en el servidor al que nos hemos conectado con la función anterior. Devuelve el parámetro *$descriptor*, que es el descriptor de la conexión al servidor de bases de datos.
- **mysql_query(*$consultaSQL*,*$descriptor*):** ejecuta la consulta SQL almacenada en la variable *$consultaSQL* en la base de datos seleccionada anteriormente.
- **mysql_close(*$descriptor*):** cierra la conexión con la base de datos.

La capa está formada por los siguientes archivos PHP:

- **buscarPorAtributos.php:** recibe una lista de atributos con los sus respectivos valores. Recupera de la capa de datos los objetos cuyos valores para esos atributos son los especificados.
- **insertarAtributo.php:** recibe los datos de un nuevo atributo y lo inserta en la capa de datos, de modo que queda disponible para que el usuario lo use en un futuro al insertar nuevos objetos.
- **insertarOV.php:** recibe los datos de un nuevo atributo y lo inserta en la capa de datos. En concreto, recibirá los atributos con los valores que el objeto tendrá para ellos.
- **consultarOV.php:** recibe el identificador del objeto que el usuario quiere visualizar, y saca de la capa de datos los valores y recursos, para enviárselos a la capa de interacción.
- **borrarOV.php:** comprueba si es posible eliminar el objeto virtual cuyo identificador ha recibido. Si lo es, lo elimina. En cualquier caso, envía un XML con el resultado a la capa de interacción con el usuario.

- **miOV.php**: recupera todos los atributos cuyo tipo de pertenencia es "*objeto*".
- **miRec.php:** recupera todos los atributos cuyo tipo de pertenencia es "*recurso*".
- **miOVSeleccionar.php:** recupera todos los atributos cuyo tipo de pertenencia es "*objeto*", de modo que queden visibles para el usuario.

3.3.5. Implementación de la capa de interacción

Se trata de la capa que resulta visible al usuario de la aplicación. A través de ella, el usuario introducirá los datos necesarios para la acción que quiera realizar, y se le mostrarán los resultados.

Como la capa de proceso, de la que se ha hablado antes, está implementada en PHP.

Los ficheros que forman parte de esta capa son básicamente de dos tipos: de toma de datos y de presentación de datos. La misión de los primeros es que el usuario pueda introducir los datos que sean necesarios, por lo que suelen ser de tipo formulario, mientras que los segundos están destinados a que el sistema muestre el resultado de la operación pedida por el usuario.

En este caso, la información suele viajar usando el método POST. Así, en el fichero PHP de la capa de interacción con el usuario, aparecería:

```
echo "<form action='ficheroDeDestino' method='POST'>
      ...
      </form>";
```

Por ejemplo, supongamos que queremos buscar el objeto virtual cuyo valor para el atributo "*A3*" sea "*Alicante*". La aplicación mostrará un interfaz para ello (Figura 3.8).

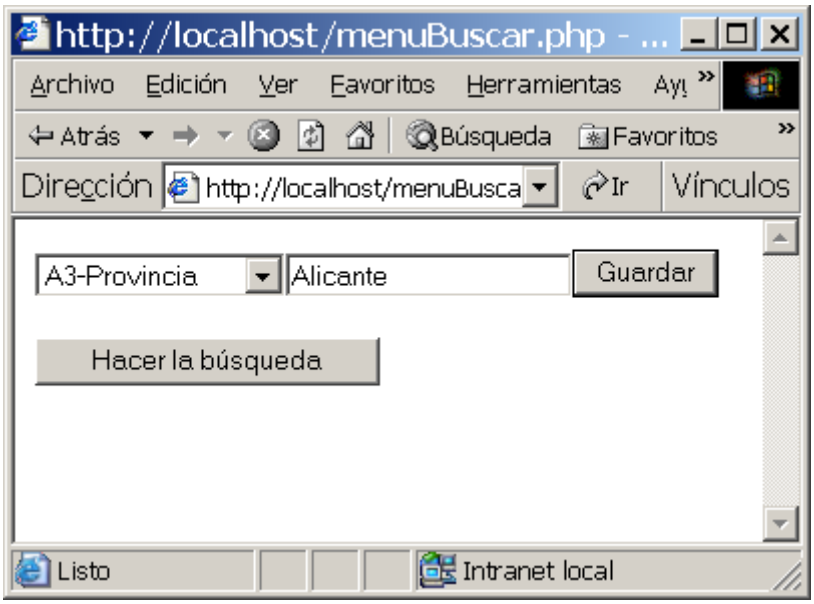


Figura 3.8. *Interfaz en la que el usuario introducirá los datos*

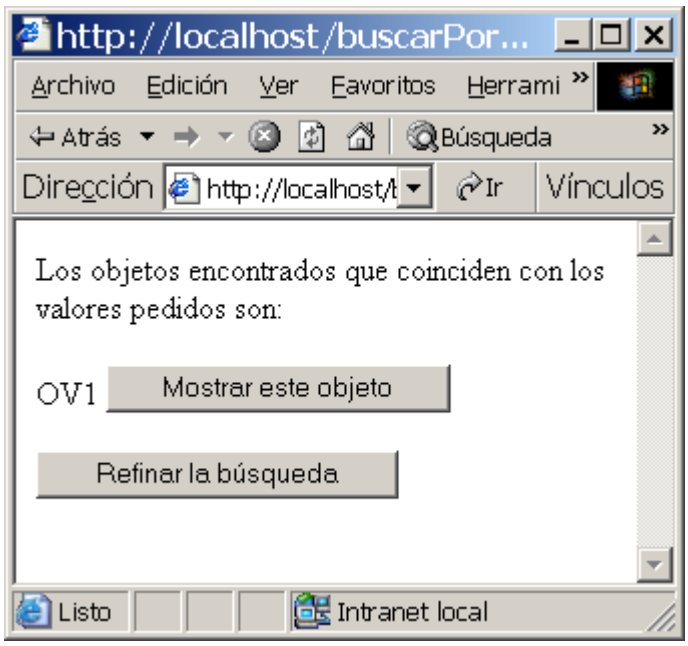


Figura 3.9. Interfaz en la que el usuario verá los datos

y el dato será enviado a la capa de proceso de esta manera:

```
<contenido>
  <listaAtributos>
    <atributo idAtributo="A3">
      <valor>Alicante</valor>
    </atributo>
  </listaAtributos>
</contenido>
```

Por su parte, ésta, una vez analizados y efectuadas las consultas necesarias a la capa de datos, devolverá a la capa de interacción un listado de los objetos que tienen ese valor codificado de la siguiente forma:

```
<contenido>
```

```
  <objeto idObjeto="OV1" />
</contenido>
```

La capa de interacción se encarga de transformar este XML, presentándolo de una forma más inteligible para el usuario (Figura 3.9).

La capa de interacción con el usuario está formada por estos ficheros:

- **indexIChasqui.php:** es el interfaz principal del sistema, que da acceso a todos las opciones del mismo.
- **generaBorrarOV.php**: encapsula el identificador de un objeto virtual en un XML que enviará a la capa de proceso (*borrarOV.php*).
- **generaConsultarOV.php:** encapsula el identificador de un objeto virtual en un XML que enviará a la capa de proceso (*consultarOV.php*).
- **generaInsertarOV.php:** genera un XML que enviará a la capa de proceso (*insertarOV.php*), en el que viajarán los datos que el usuario quiere insertar como un nuevo objeto.
- **generaInsertarAtributo.php:** genera un XML que enviará a la capa de proceso (*insertarAtributo.php*), en el que viajarán las características que el usuario desea que tenga el nuevo atributo.
- **miInterfaz.php:** es el interfaz a través del cual se muestran los atributos que puede tener un objeto virtual o un recurso (datos procedentes de *miOV.php* o *miRec.php*)
- **buscarPorAtributosInterfaz.php:** muestra el resultado del a busqueda de objetos por valores de uno o varios de sus atributos.

3.3.6. Lenguaje de comunicación entre ficheros PHP

En general, podemos decir que las gramáticas que definen el lenguaje por el cual se comunican los distintos ficheros PHP son DTDs relacionadas entre sí (Figura 3.10).

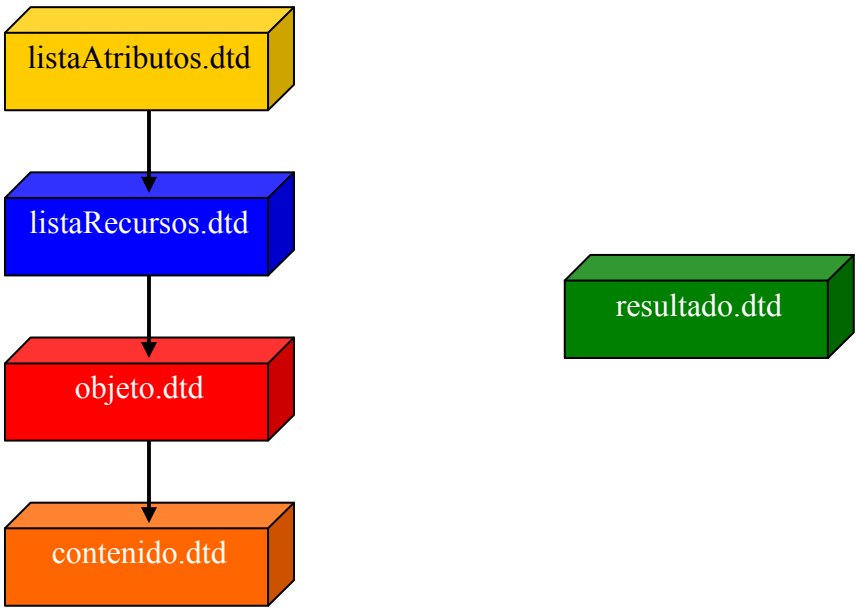


Figura 3.10. Estructura de las DTDs

Así, *listaRecursos.dtd* incluye a *listaAtributos.dtd*, y a su vez está incluida en *objeto.dtd*. *contenido.dtd* puede ser considerada como una lista de objetos. *resultado.dtd* es independiente, puesto que se usa para enviar un mensaje con el resultado de una operación (inserción, borrado, ...).

3.3.6.1. listaAtributos.dtd

Con esta DTD se especifica un lenguaje para crear listas de atributos pertenecientes a objetos virtuales.

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

<!ELEMENT listaAtributos (atributo*)>
<!—Una lista de atributos puede tener entre 0 y varios elementos de este tipo
-->

<!ELEMENT atributo (nombre,tipo,valor,hijos*)>
<!-- Un atributo puede no tener hijos -->
<!ATTLIST atributo
        idAtributo CDATA #REQUIRED>

<!ELEMENT nombre (#PCDATA)>
<!—- Nombre del atributo -->

<!ELEMENT tipo (#PCDATA)>
<!—- Tipo del atributo -->

<!ELEMENT hijos (atributo)>
<!—- Hijos del atributo, que también serán atributos -->
```

3.3.6.2. listaRecursos.dtd

Esta DTD pretende, reutilizando "*listaAtributos.dtd*", que el lenguaje obtenido permita crear listas de atributos. Hay que tener en cuenta que cada recurso de la

lista puede tener una lista de atributos asociada, lo que se puede especificar gracias a la reutilización de "*listaAtributos.dtd*".

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

<!ENTITY % lista_de_atributos SYSTEM "listaAtributos.dtd">
%lista_de_atributos;
<!—- Declaración del contenido de la DTD anterior -->

<!ELEMENT listaRecursos (recurso*)>
<!—- Lista de recursos -->

<!ELEMENT recurso ((dato,listaAtributos)>
<!-- De cada recurso, tendremos la ruta (si es propio),
     o el identificador (si es otro objeto o externo) -->

<!ELEMENT dato (#PCDATA)>
<!-- Dato del recurso: si es propio, será el nombre del fichero que lo
constituye; si es externo, será el identificador del recurso al que esté
referenciando; y si es objeto, el del objeto al que esté referenciando -->

<!ATTLIST recurso idRecurso CDATA #IMPLIED>
<!—- El atributo de un elemento de tipo "recurso" es el "idRecurso" (su
identificador) . Además, el atributo es opcional. De este modo, la DTD puede
servir tanto para introducir datos al sistema como para extraerlos: en el
primer caso no hay "idRecurso" mientras que en el segundo lo hay -->
```

3.3.6.3. objeto.dtd

Con este documento, especificamos un lenguaje que permite crear XMLs cuyo contenido estará compuesto por los datos de un objeto virtual: valores para sus atributos, recursos y valores para los atributos de éstos. Se podrá usar tanto desde la capa de proceso a la de interacción como en sentido inverso.

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

<!ENTITY % lista_de_recursos SYSTEM "listaRecursos.dtd">
%lista_de_recursos;
<!—- Declaración del contenido de la DTD anterior ("listaRecursos.dtd" ya
incluye a "listaAtributos.dtd") -->

<!ELEMENT objeto (listaAtributos,listaRecursos)>
<!-- Un objeto es una lista de atributos y de recursos -->

<!ATTLIST objeto
        idObjeto CDATA #IMPLIED>
<!—- El atributo de un elemento de tipo "objeto" es el "idObjeto" (su
identificador). Además, el atributo es opcional. De este modo, la DTD puede
servir tanto para introducir datos al sistema como para extraerlos: en el
primer caso no hay "idObjeto" mientras que en el segundo lo hay -->
```

3.3.6.4. contenido.dtd

Los documentos que se generen a partir de esta DTD serán listas de objetos (en ocasiones sólo de un elemento).

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

<!ENTITY % contenido_objeto SYSTEM "objeto.dtd">
%contenido_objeto;

<!ELEMENT contenido (objeto)*>
<!-- Un elemento de tipo "contenido" es una lista de elementos de tipo
"objeto" -->
```

3.3.6.5. resultado.dtd

Los XMLs basados en esta DTD se utilizan para que la capa de proceso informe a la de interacción del resultado de una operación efectuada a petición del usuario. En concreto, el mensaje será el atributo "*mensaje*".

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

<!ELEMENT resultado EMPTY>

<!ATTLIST resultado
        mensaje CDATA #REQUIRED>
<!-- El resultado de una operación se especificará mediante un mensaje, que
será un atributo del elemento de tipo "resultado" -->
```

Capítulo 4. Conclusiones y trabajo final

4.1. Conclusiones

El sistema elaborado en el marco de este trabajo ofrece una funcionalidad básica para la gestión de objetos de aprendizaje.

Hay que considerar que, aunque la idea está basada en otros sistemas que se han desarrollado anteriormente, se trata de un prototipo desarrollado desde cero, por lo que el resultado logrado en estos meses es un sistema elemental.

Proporciona una definición más básica de objeto virtual que la propuesta con otros sistemas, ya que en este caso no se habla de metadatos, sino sólo de atributos y recursos (que también tienen sus atributos). De este modo, se simplifica más el manejo del sistema para un usuario que no posea un perfil técnico, que posiblemente sea el caso más frecuente.

También teniendo en cuenta cuáles van a ser los posibles usuarios, se ha intentado que el manejo de la aplicación sea sencillo e intuitivo, de modo que la persona que empiece a utilizarla no tenga demasiados problemas a la hora de lograr un dominio de la misma.

4.2. Trabajo futuro

Tal y como se ha comentado en el apartado anterior, el sistema es básicamente un embrión de lo que puede ser un nuevo miembro de la familia Chasqui. Precisamente por ello, ofrece unas posibilidades bastante importantes para ser ampliado.

En primer lugar, sería posible mejorar el aspecto gráfico, de modo que el interfaz resulte más atractivo para el usuario. En este sentido, sería recomendable utilizar *CSSs* (*Cascade Style Sheets*).

Por otro lado, sería también interesante ofrecer un control de acceso para los distintos usuarios de la aplicación. Así, podría hacerse que los objetos sólo sean accesibles por su autor, o que sólo éste tenga determinados privilegios como la modificación y el borrado. Obviamente, ésto supondría una modificación de la base de datos, en la que sería necesario incluir nuevas tablas para gestionar los usuarios, así como para saber qué objetos pertenecen a cada usuario.

De esta forma, incluso se podrían definir usuarios con distintos roles, como el usuario básico, el administrador, ... y a cada rol le corresponderían distintos privilegios, del mismo modo que ocurre con las bases de datos.

Así, el sistema podría ser usado por varios usuarios para fines totalmente distintos. Por ejemplo, uno de ellos podría gestionar objetos relacionados con la historia de una región determinada, mientras que otro lo haría con motores de aviación.

De cara a la portabilidad, una buena idea podría ser tratar de crear una nueva funcionalidad que permita empaquetar los objetos en ficheros comprimidos (de tipo **.zip*, **.tar*, ... o similares). De este modo, el usuario podría bajar del sistema los datos relativos a cada objeto virtual y consultarlos *offline*. Esto ya se puede hacer en los sistemas anteriores a éste: *Chasqui*, *Chasqui 2*, *MIGS 0.0* y *MIGS 1.0*.

Referencias

[1] David A. Wiley, II, Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy

[2] *http://en.wikipedia.org/wiki/E-learning*

[3] Sun Microsystems, E-learning application infrastructure

[4] Alfredo Fernández-Valmayor, Mercedes Guinea, Mariano Jiménez, Antonio Navarro, Antonio Sarasa, Objetos virtuales: una aproximación a la construcción de objetos de aprendizaje en Arqueología

[5] José-Luis Sierra, Alfredo Fernández-Valmayor, Tagging Learning Objects with Evolving Metadata Schemas

[6] José Luis Sierra, Alfredo Fernández-Valmayor, Mercedes Guinea, Héctor Hernanz, From Research Resources to Learning Objects: Process Model and Virtualization Experiences

[7] José Luis Sierra, Alfredo Fernández-Valmayor, Mercedes Guinea, Héctor Hernanz. From Chasqui to Chasqui II: an Evolution in the Conceptualization of Virtual Objects

[8] Juan José Calvo Diaz De Neira, Alberto Espadero Guillermo, David Sánchez Llorente. Creación, Gestión y Uso de Objetos de Aprendizaje en la Web

[9] Jose Luis Rojo Rojo, Miguel Ángel Alonso Pajuelo. Creación, gestión y uso de "objetos de aprendizaje" en los Sistemas Chasqui

[10] *http://en.wikipedia.org/es/wiki/PHP*

[11] Cabezas Granado Luis Miguel. PHP 5, Manual Imprescindible

[12] Andi Gutmans, Stig Saether Bakken, Derick Rethans . PHP 5 Power Programming

[13] *http://www.php.net*

[14] *http://en.wikipedia.org/es/wiki/MySQL*

[15] *http://en.wikipedia.org/es/wiki/Apache*

[16] *http://en.wikipedia.org/es/wiki/XAMPP*

[17] *http://www.apachefriends.org/es/xampp.html*

[18] *http://en.wikipedia.org/es/wiki/XML*

[19] Óscar González. XML. Guía práctica para usuarios.

Apéndice 1. Manual de instalación

Para poner en producción la presente aplicación, se recomienda utilizar la misma configuración que se usó para su desarrollo: el servidor web Apache, el lenguaje de programación de páginas web dinámicas PHP y el sistema gestor de bases de datos MySQL. El ordenador utilizado funcionaba con Microsoft Windows 2000.

Para facilitar la instalación, se utilizó XAMPP, que es un servidor independiente de plataforma, software libre, formado por la base de datos MySQL, el servidor web Apache e intérpretes para lenguajes de script como PHP y Perl.

Precisamente, su nombre proviene del las primeras letras de **A**pache, **M**ySQL, **P**HP, **P**erl. La *X* indica que es posible utilizarlo para cualquier sistema operativo. Actualmente está disponible para Microsoft Windows, GNU/Linux, Solaris, y MacOS X.

El programa esta liberado bajo la licencia GNU, por lo que puede descargarse de forma totalmente libre y gratuita desde la dirección:
`http://www.apachefriends.org/es/xampp.html`

En la actualidad, está disponible con la siguiente configuración para Windows:

- *Apache HTTPD 2.2.8 + Openssl 0.9.8g*
- *MySQL 5.0.51a*
- *PHP 5.2.5*
- *PHP 4.4.8*
- *phpMyAdmin 2.11.4*
- *FileZilla FTP Server 0.9.25*
- *Mercury Mail Transport System 4.52*

En primer lugar se debe instalar XAMPP, para lo cual se deben seguir las instrucciones que se incluyen en la página web desde la que se descarga.

Como se trata de una aplicación web, es necesario arrancar los servidores Apache y MySQL, lo que se hace seleccionando Inicio ➔ Programas ➔ XAMPP (si estamos en una máquina con Windows). Una vez hecho esto, veremos la consola de control de XAMPP. En ella, basta con pulsar los botones con la etiqueta *Start* al lado del nombre del servidor correspondiente(Figuras A1.1 y Figura A1.2).

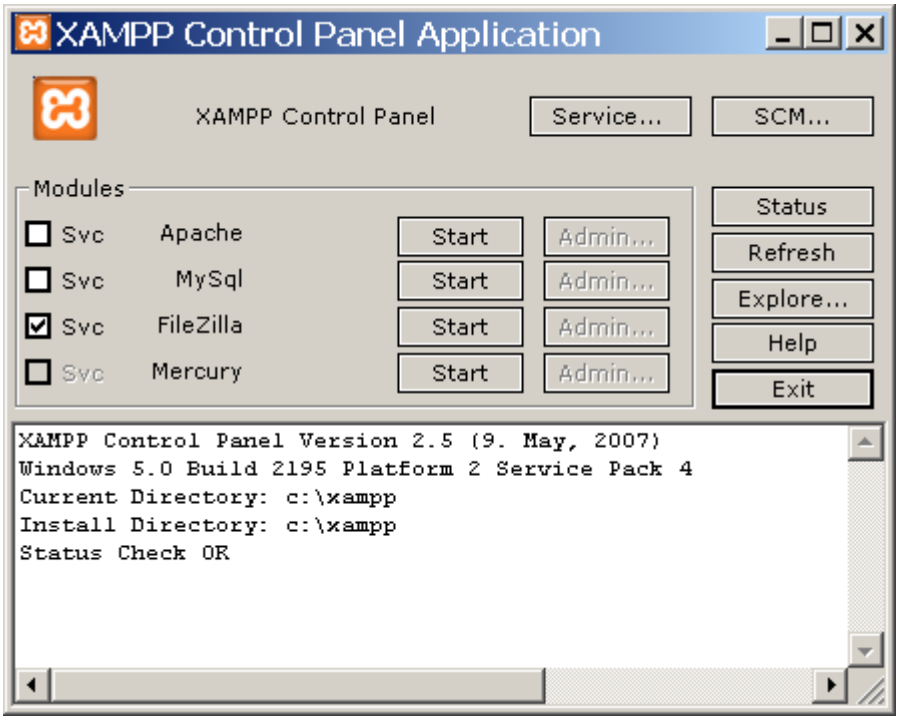


Figura A1.1. Panel de control con todas las aplicaciones inactivas

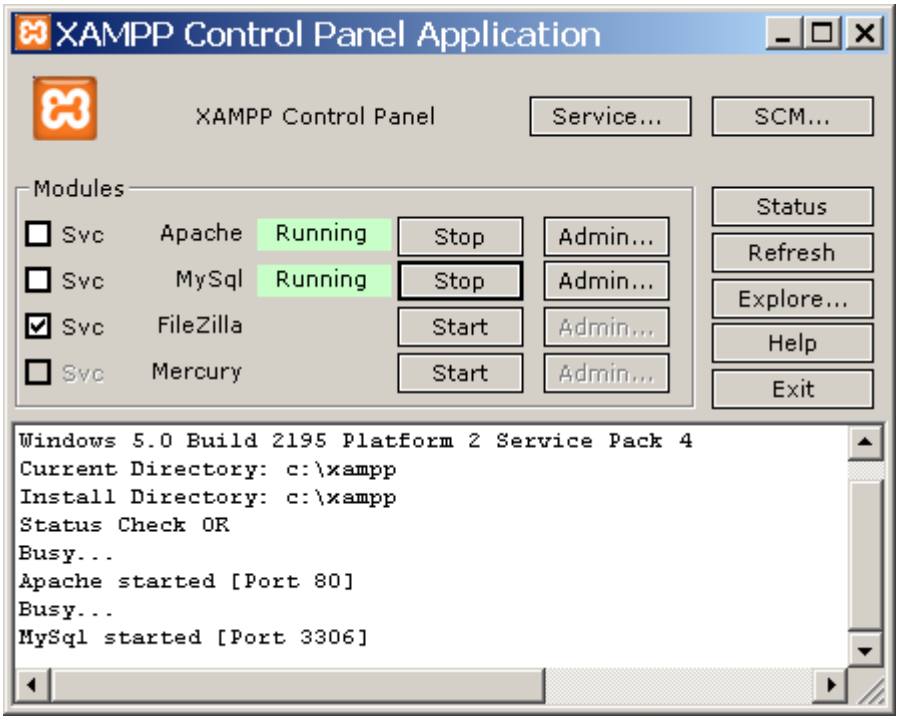


Figura A1.2. Panel de control con MySQL y Apache activos

Se recomienda, una vez efectuado ésto, probar si Apache funciona correctamente. Escribiremos `http://localhost/` y, si funciona, accederemos a una página de presentación (Figura A1.3).

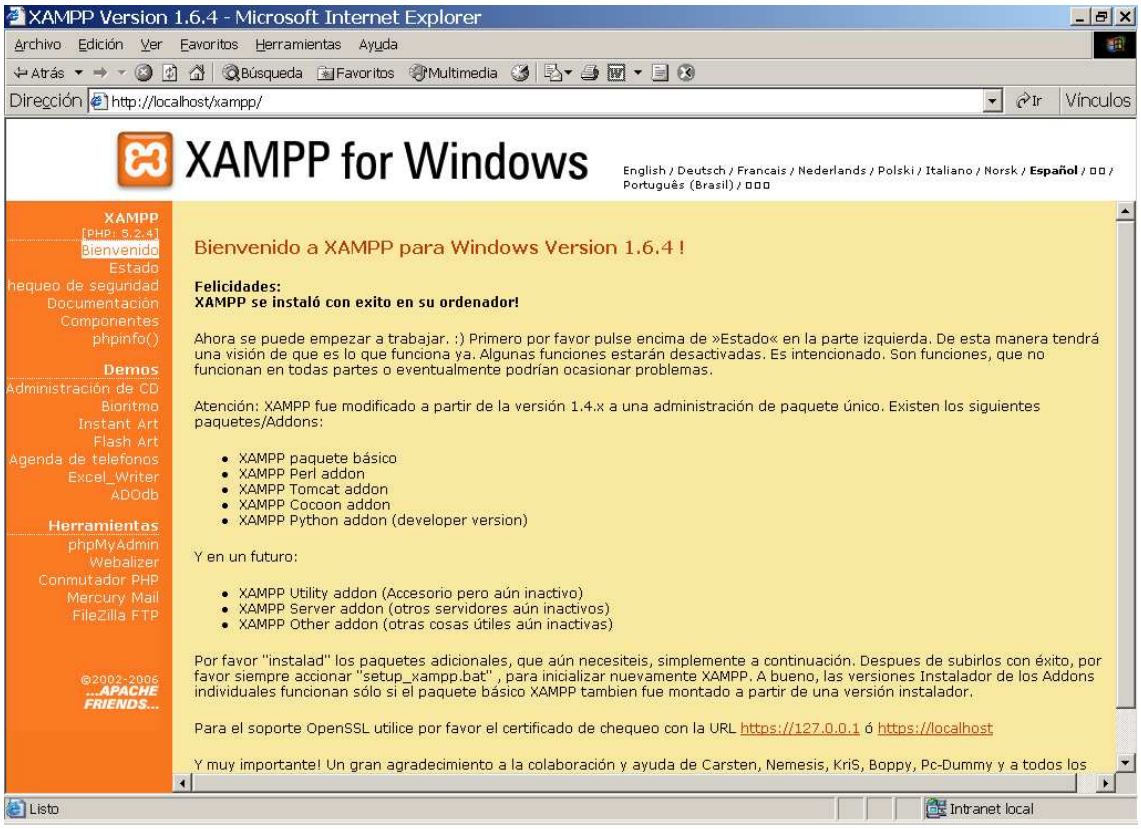


Figura A1.3. Página principal de XAMPP

En caso de que se desee parar alguno de estos servidores, se debe pulsar el botón de *Stop*.

Una vez en esta pantalla, se debe pulsar *Herramientas* ➔*phpMyAdmin*, lo que mostrará el interfaz desde el que se puede manejar la base de datos.

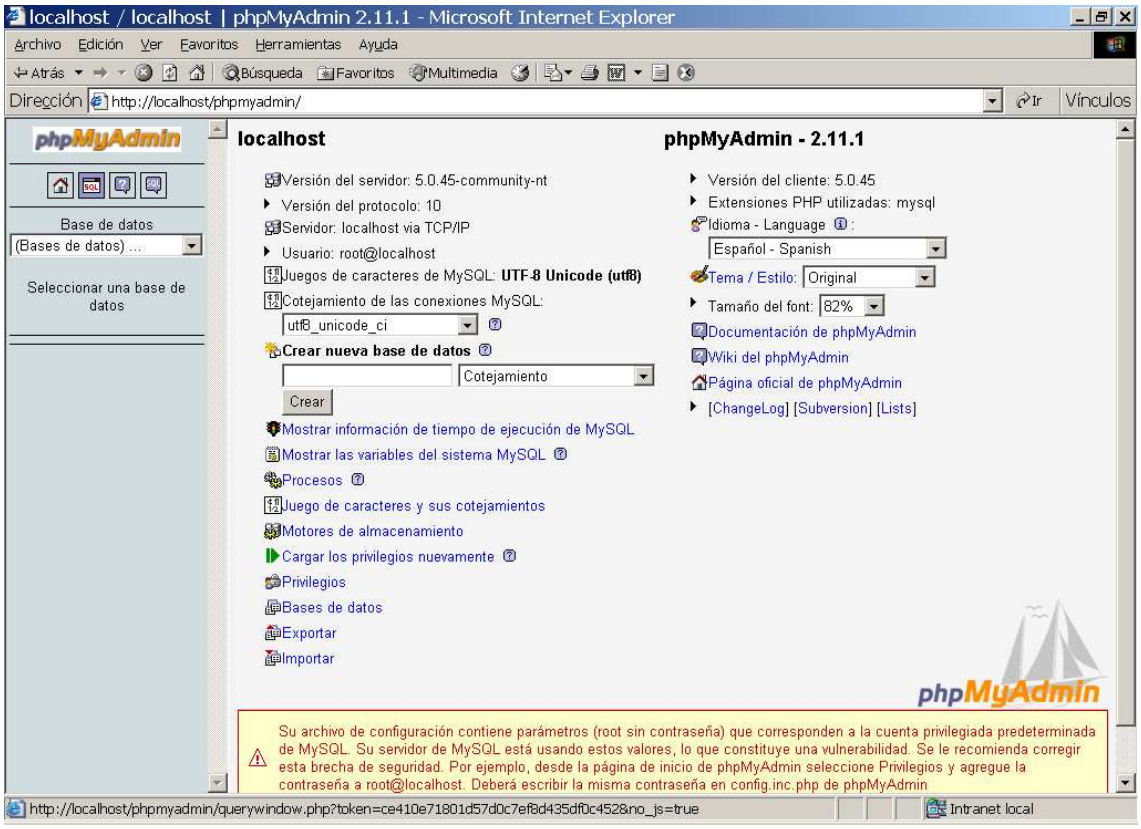


Figura A1.4. Interfaz de administración de la base de datos

En esta pantalla, es necesario hacer clic en el botón SQL, que abrirá una ventana de consulta como la siguiente:

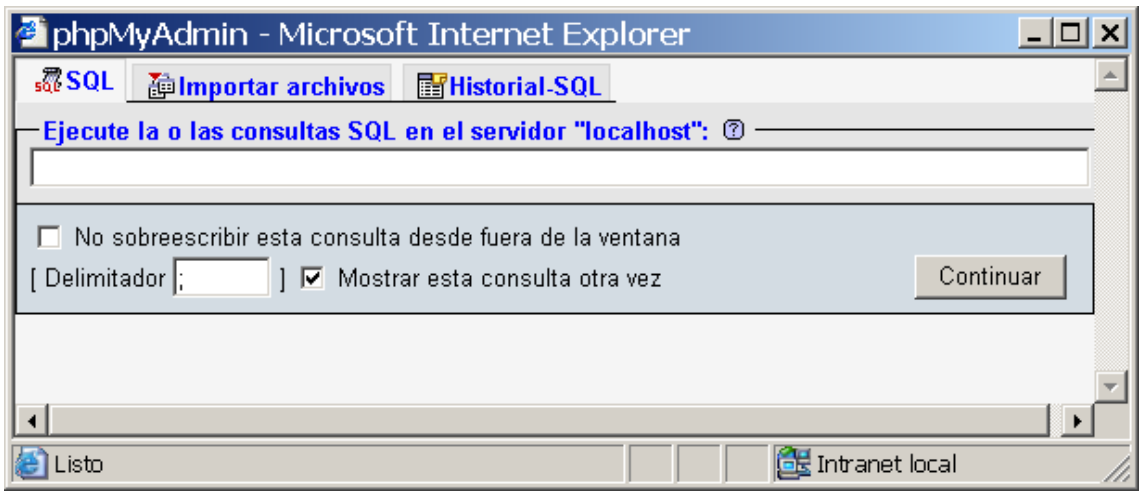


Figura A1.5. Consola para consultas SQL

En ella, se deben escribir las siguientes consultas, que se encargarán de crear las tablas que sean necesarias:

```
CREATE DATABASE oda_virtual;
USE oda_virtual;
CREATE TABLE atributo(
       idAtributo VARCHAR(10),
       nombre VARCHAR(20),
       tipoAtributo VARCHAR(10),
```

```
       tipoPertenencia VARCHAR(10),
       atributoPadre VARCHAR(10),
primary key (idAtributo));

CREATE TABLE enumerado(
       idAtributo VARCHAR(10),
       valorEnumerado VARCHAR(20),
primary key(idAtributo,valorEnumerado));

CREATE TABLE valor(
       idValor VARCHAR(10),
       idAtributo VARCHAR(10),
primary key (idValor));

CREATE TABLE valor_cadena(
       idValor VARCHAR(10),
       valor VARCHAR(50),
primary key (idValor));

CREATE TABLE valor_numerico(
       idValor VARCHAR(10),
       valor INTEGER,
primary key (idValor));

CREATE TABLE asignacion_recurso(
       idRecurso VARCHAR(10),
       idValor VARCHAR(10),
primary key (idRecurso,idValor));

CREATE TABLE asignacion_objeto(
       idObjeto VARCHAR(10),
       idValor VARCHAR(10),
primary key (idObjeto,idValor));

CREATE TABLE objeto(
       idObjeto VARCHAR(10),
primary key (idObjeto));

CREATE TABLE recurso(
       idRecurso VARCHAR(10),
       idObjeto VARCHAR(10),
       tipo_recurso VARCHAR(10),
primary key (idRecurso));

CREATE TABLE recurso_propio(
       idRecurso VARCHAR(10),
       nombreFichero VARCHAR(40),
primary key (idRecurso));

CREATE TABLE recurso_externo(
       idRecurso VARCHAR(10),
       idRecursoExterno VARCHAR(10),
primary key (idRecurso));

CREATE TABLE recurso_objeto(
       idRecurso VARCHAR(10),
       idObjeto VARCHAR(10),
primary key (idRecurso));

CREATE TABLE dependencias_recursos(
       idRecurso VARCHAR(10),
       nombreFichero VARCHAR(40),
primary key (idRecurso));

CREATE TABLE idAtLibres(
       idAtributo VARCHAR(10),
primary key(idAtributo));
```

```
CREATE TABLE idOVLibres(
       idObjeto VARCHAR(10),
primary key(idObjeto));

CREATE TABLE idRecLibres(
       idRecurso VARCHAR(10),
primary key(idRecurso));

CREATE TABLE idValorLibres(
       idValor VARCHAR(10),
primary key(idValor));
```

A continuación, para finalizar de instalar la aplicación es necesario copiar los archivos que forman parte de ella en el directorio *xampp\htdocs*.

Apéndice 2. Manual de usuario

En primer lugar, es necesario que el usuario se sitúe en la página principal de la aplicación. Para ello, debe teclear en el navegador:

`http://localhost/indexIChasqui.php`

Así, se le mostrarán las distintas opciones para el uso del programa.

A2.1. Buscar objetos por valores de sus atributos

Tras seleccionar la opción en la pantalla principal del programa, aparece otra pantalla. En ella, se debe elegir el atributo sobre el que se quiera hacer la búsqueda en el menú desplegable de la izquierda. A continuación, se debe introducir el valor que se quiera que tenga el objeto buscado (*Figura A2.1 y A2.2)*.

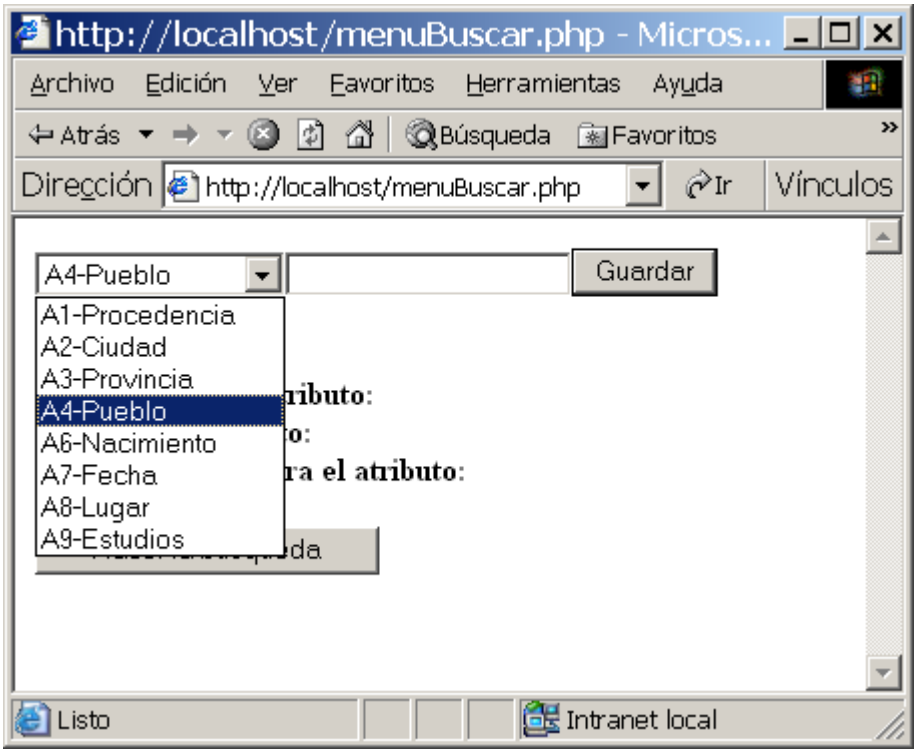


Figura A2.1. Interfaz para introducir una nueva condición

Una vez hecho esto, se pulsa el botón guardar, de modo que se muestren las condiciones seleccionadas para la búsqueda.

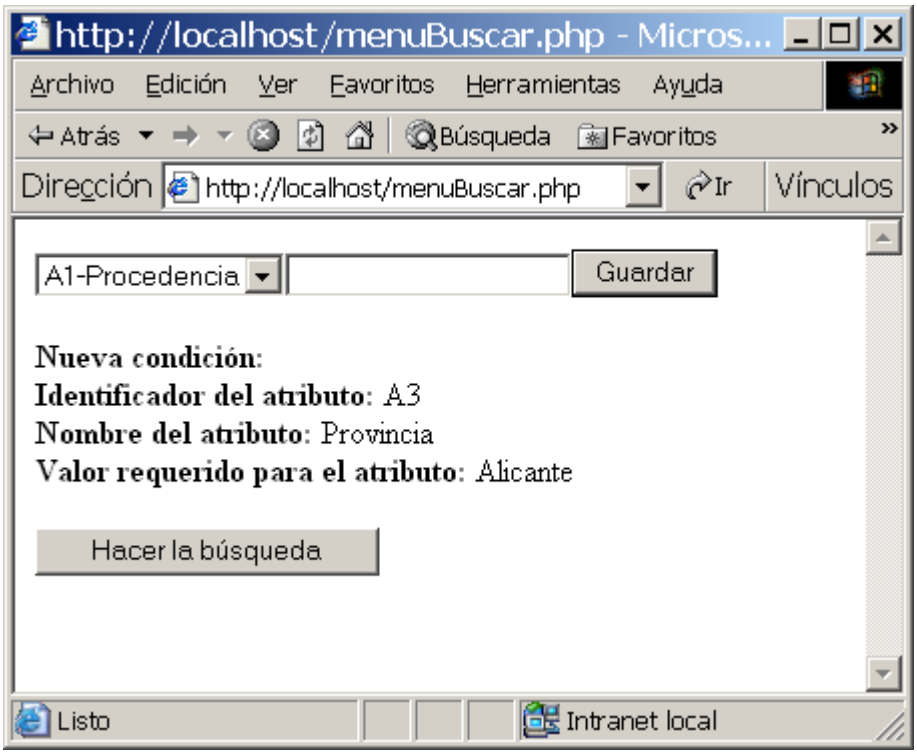


Figura A2.2. Interfaz para introducir una nueva condición, tras haberlo hecho

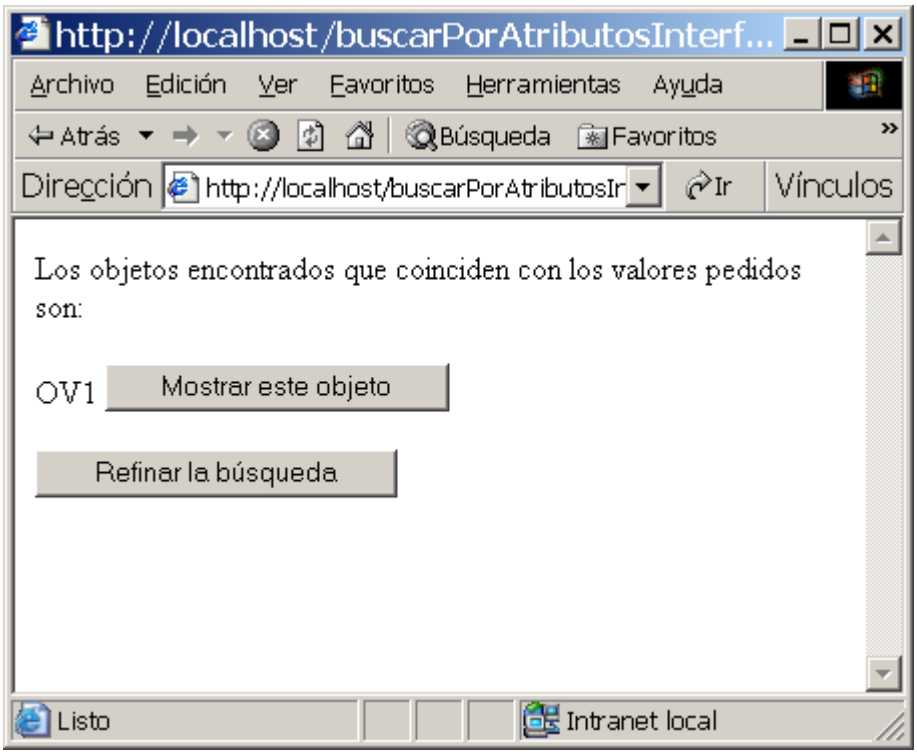


Figura A2.3. Interfaz para refinar una búsqueda o consultar uno de los objetos encontrados

Por último, se debe pulsar el botón "Hacer la búsqueda", tras lo que el sistema mostrará una página con los resultados, en la que se permite refinar la búsqueda (añadir nuevas condiciones a la misma), o mostrar los objetos obtenidos (*Figura A2.3)*.

A2.2. Consultar un objeto virtual

En este caso, el sistema redirecciona al usuario a una página en la que éste debe introducir el identificador del objeto que quiere borrar, en el campo correspondiente.

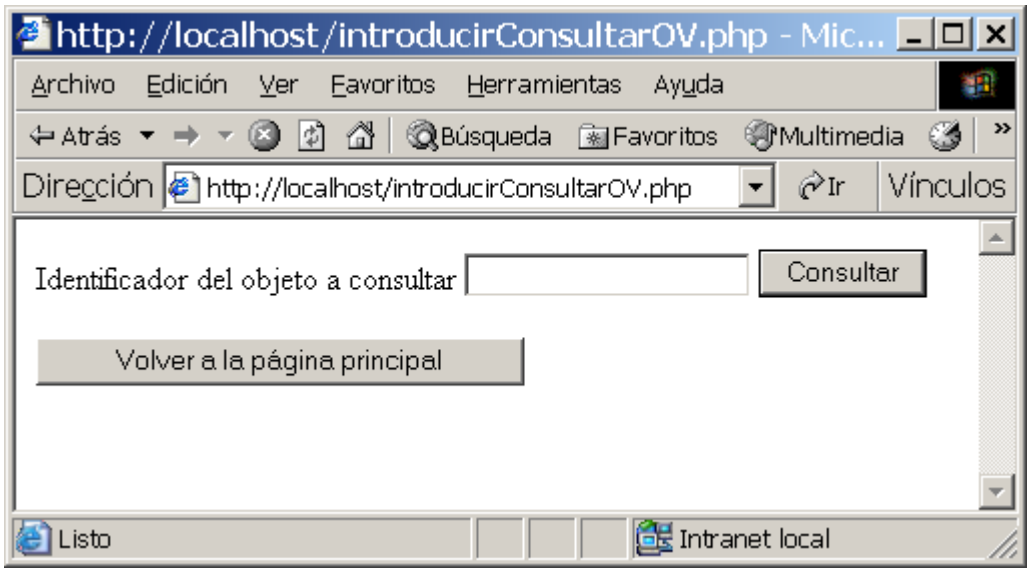


Figura A2.4. Interfaz para consultar un objeto

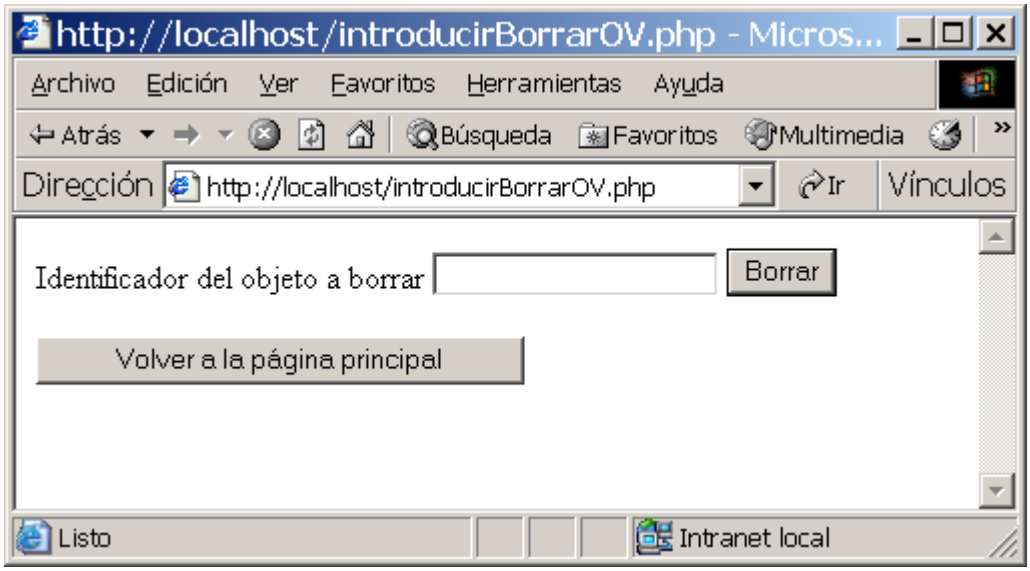


Figura A2.5. Interfaz para consultar un objeto

Después, se muestra el resultado. Es decir, si el objeto ha podido ser borrado correctamente o si, por el contrario, existen dependencias que lo impiden (Figura A2.6).

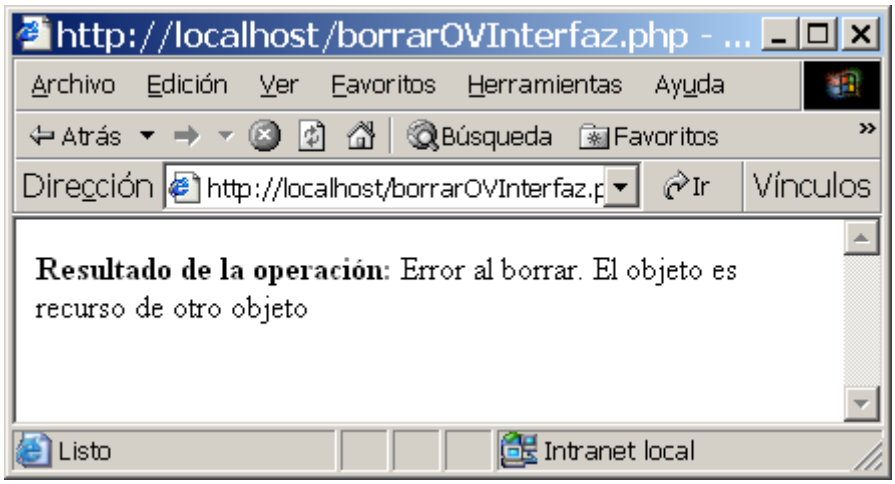


Figura A2.6. Interfaz que muestra el resultado de una operación

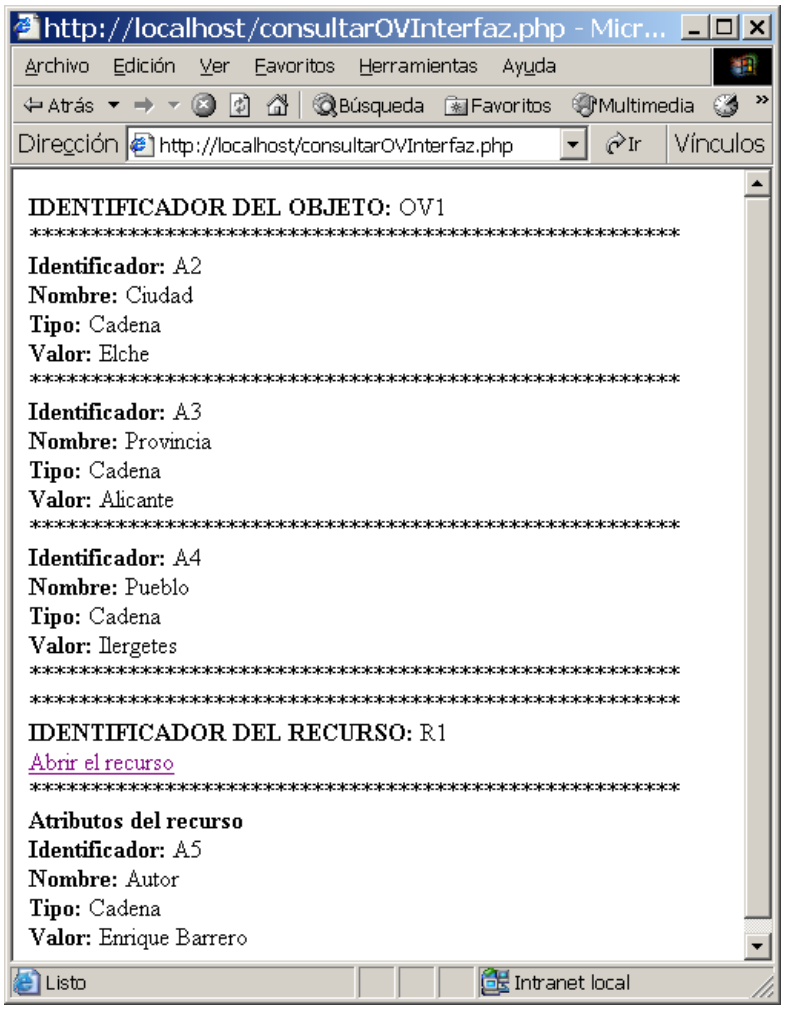


Figura A2.7. Interfaz para la consulta de objetos virtuales

A2.3. Borrar un objeto virtual

El caso de consultar funciona de un modo bastante parecido al anterior, pero lo que se muestra es la estructura del objeto. Además, se ofrece un link según el cual se puede ver el fichero en el que consiste el recurso (Figura A2.7).

A2.4. Introducir un nuevo objeto virtual

El mecanismo que se sigue para la introducción de los datos es básicamente el mismo que en el caso de buscar un objeto en función de los valores de sus atributos. Tras rellenarse los campos correspondientes, se guardan, de modo que se muestran por pantalla. En caso de que la elección haya sido errónea, el usuario puede modificarla y guardarla tantas veces como quiera (Figura A2.8).

Para efectuar la inserción, debe pulsar el botón de "*Almacenar el objeto*". Si el tipo de recurso es propio, el sistema asumirá que el contenido del campo "Dato del recurso" será la ruta de la cual debe coger el archivo en el que consistirá el recurso. Si es de tipo externo, se trata del identificador del recurso al que hace referencia, y si es de tipo objeto, del identificador del objeto.

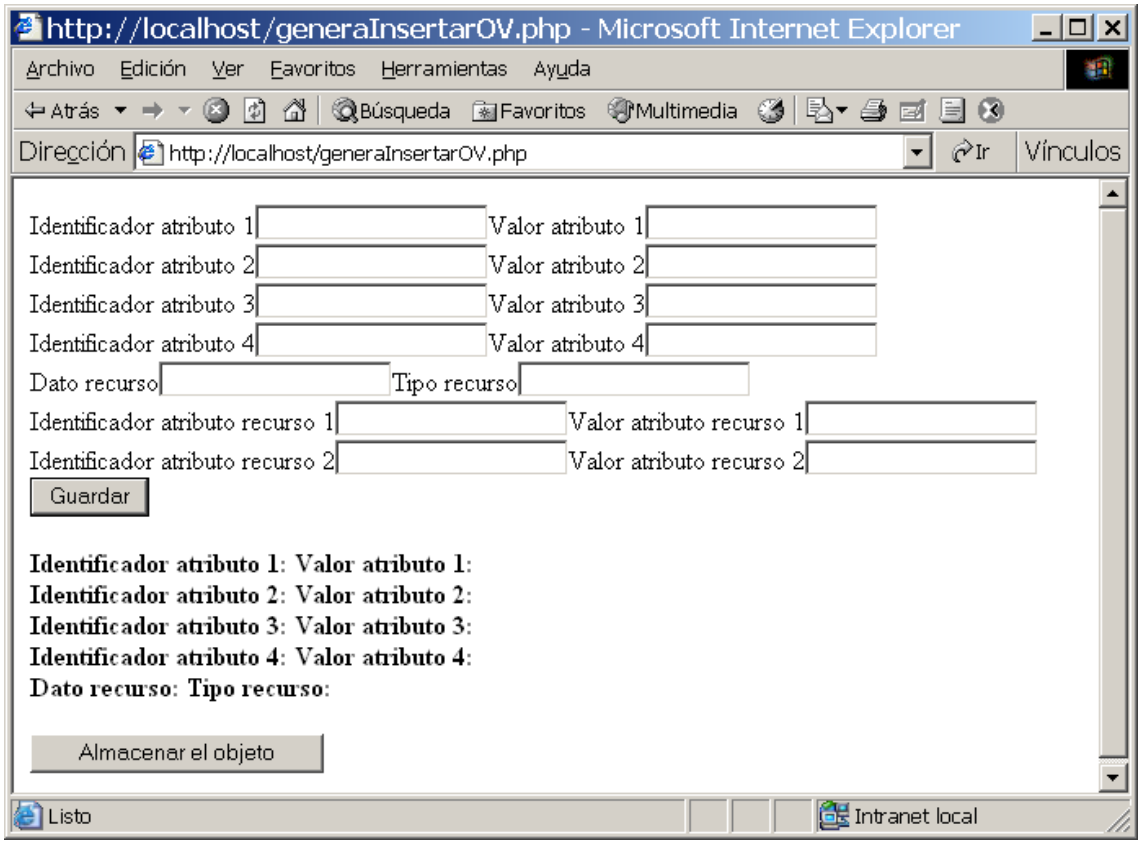


Figura A2.8. Interfaz para la inserción de objetos virtuales

Una vez hecho esto, se mostrará el resultado de la operación, del mismo modo que en el caso anterior.

A2.5. Introducir un nuevo atributo

El funcionamiento es básicamente el mismo que el del caso anterior: basta con rellenar las opciones necesarias en el formulario presentado por la aplicación (Figura A2.9).

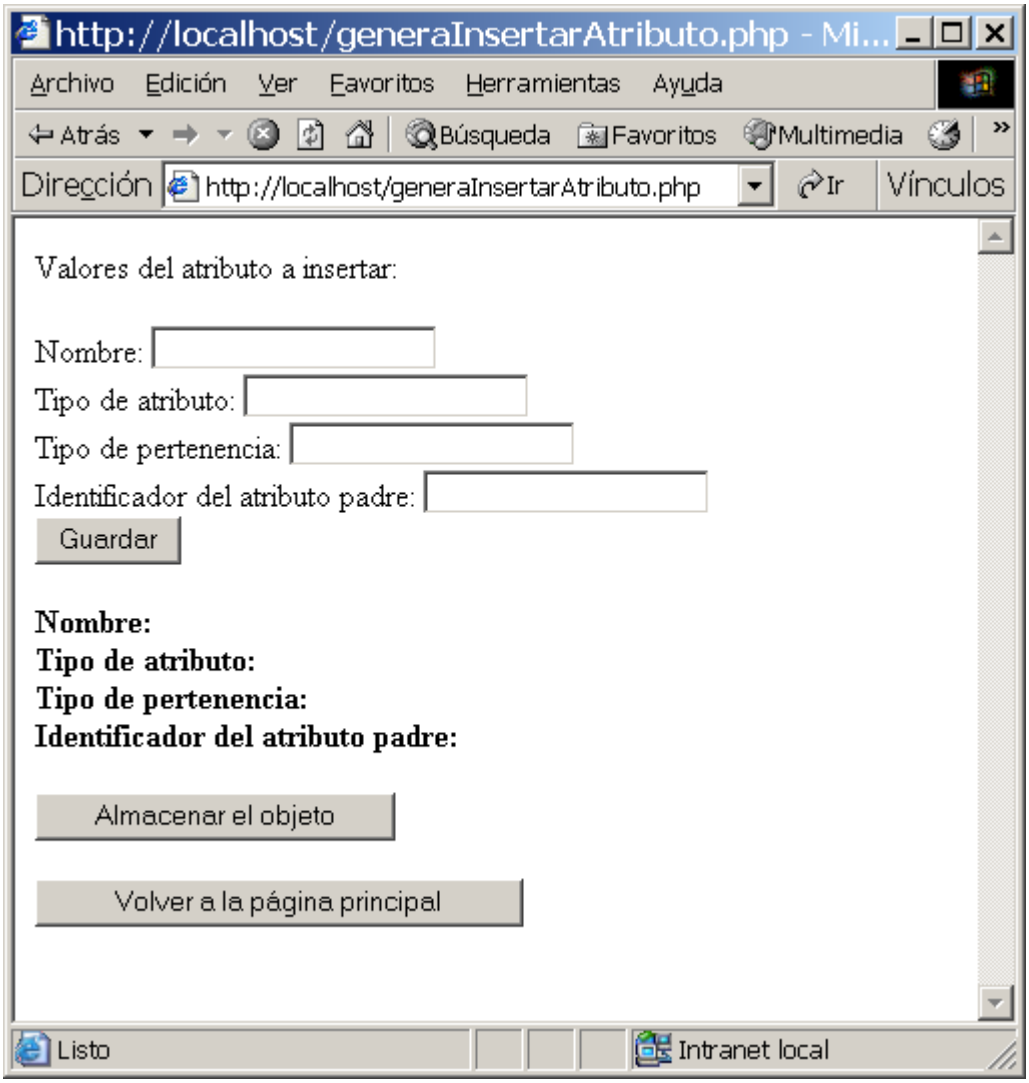
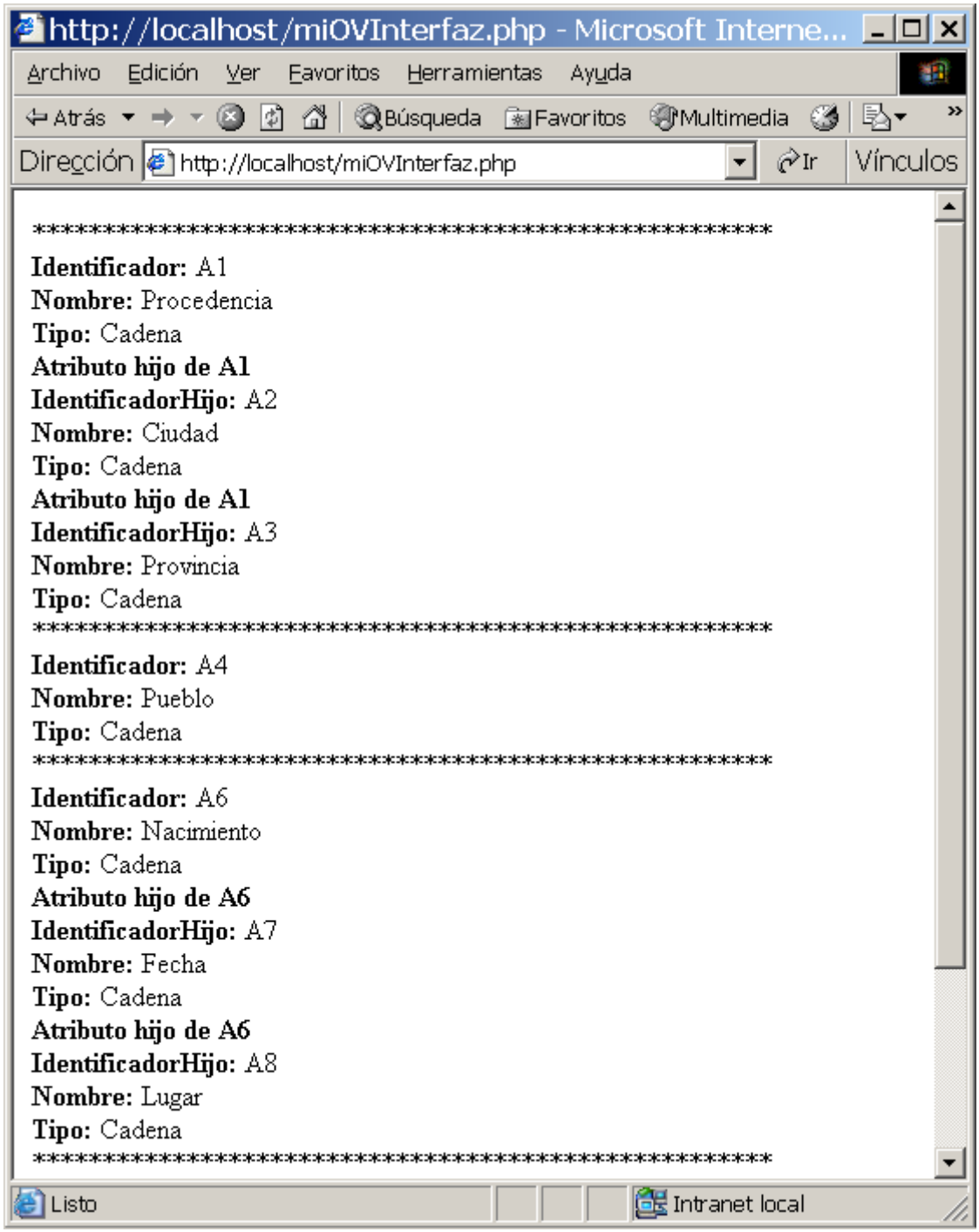


Figura A2.9. Interfaz para la inserción de atributos

A2.6. Mostrar el modelo de información

En este caso, sólo hay que seleccionar la opción en la página principal de la aplicación, y el sistema mostrará todos los atributos disponibles (modelo de información).


http://localhost/miOVInterfaz.php - Microsoft Interne...
Archivo Edición Ver Favoritos Herramientas Ayuda
Atrás Búsqueda Favoritos Multimedia
Dirección http://localhost/miOVInterfaz.php
Ir Vínculos
****
Identificador: A1
Nombre: Procedencia
Tipo: Cadena
Atributo hijo de A1
IdentificadorHijo: A2
Nombre: Ciudad
Tipo: Cadena
Atributo hijo de A1
IdentificadorHijo: A3
Nombre: Provincia
Tipo: Cadena
****
Identificador: A4
Nombre: Pueblo
Tipo: Cadena
****
Identificador: A6
Nombre: Nacimiento
Tipo: Cadena
Atributo hijo de A6
IdentificadorHijo: A7
Nombre: Fecha
Tipo: Cadena
Atributo hijo de A6
IdentificadorHijo: A8
Nombre: Lugar
Tipo: Cadena
****
Listo
Intranet local

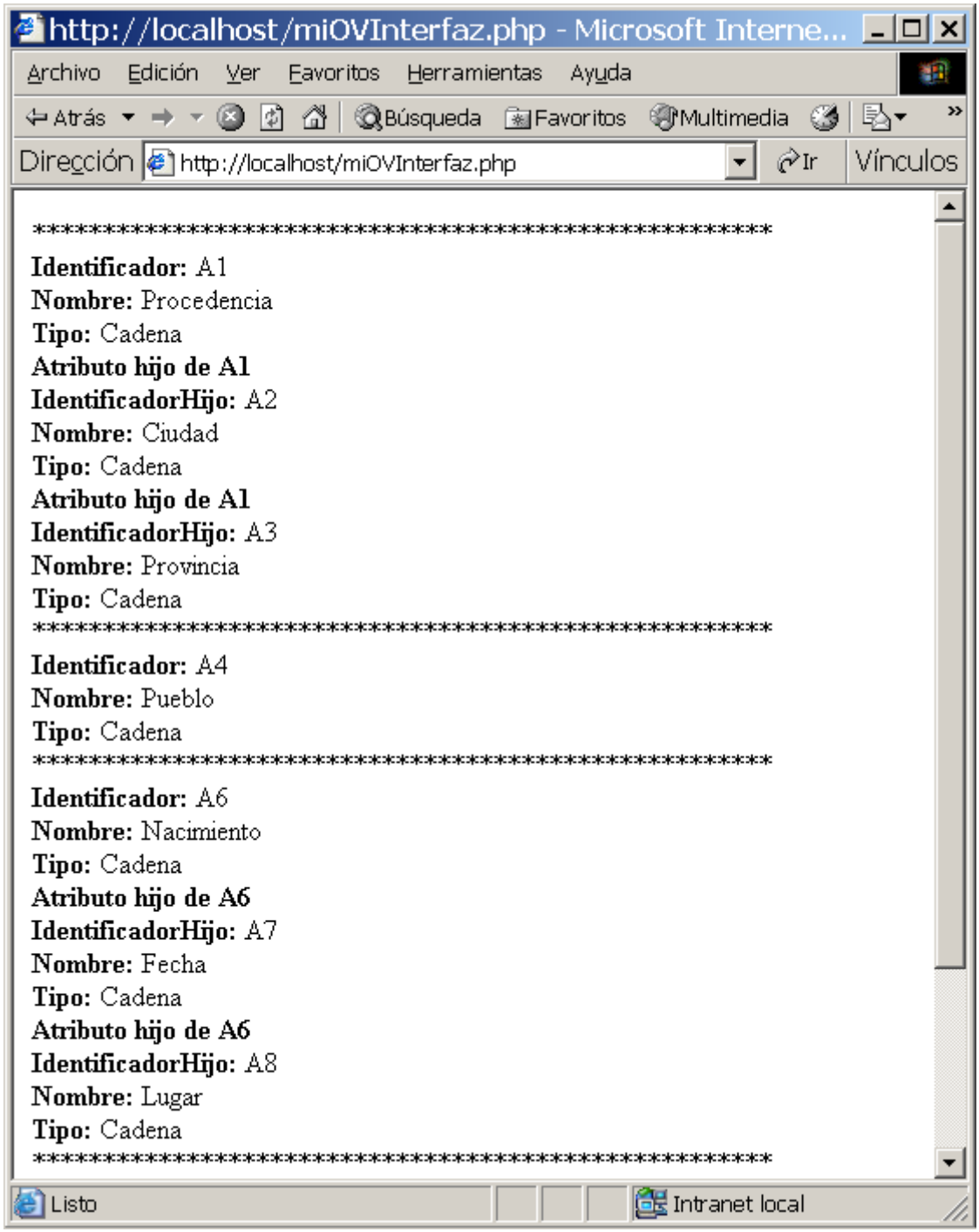


Figura A2.10. Interfaz que muestra el modelo de información de los objetos virtuales

Autorización

Se autoriza a la Universidad Complutense 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

Firma del autor:

Jesús Enrique Arnaiz Barrero