UNIVERSIDAD EUROPEA
MIGUEL DE CERVANTES

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

PROYECTO FIN DE CARRERA

DESARROLLO DE UN SISTEMA DE DISCO DURO VIRTUAL PARA LA DOCENCIA

AUTORES

DAVID MEDINA ORTEGA
FERNANDO PINTADO MIRA

DIRECTOR

FRANCISCO PALACIOS BURGOS

VALLADOLID, MARZO DE 2009

AGRADECIMIENTOS

Nos gustaría dar las gracias a nuestra familia en primer lugar, pues han sido los que nos han animado en todo momento desde el comienzo de la carrera y nos han aguantado todas las ideas que nos surgían, así como de qué iba a constar la aplicación, sin saber siquiera de qué les hablábamos. También agradecer a todos los amigos el apoyo y las ayudas que nos han proporcionado, sabiendo que el esfuerzo requerido en un PFC es muy grande. Gracias a todos los profesores que nos han aportado su experiencia, ánimos e interés en mejorar tanto el fondo como las formas en el proyecto, en momentos en los que todavía no se ve el final del proyecto. Y por último queremos agradecernos mutuamente las críticas que nos hacíamos, las cuales nos han hecho avanzar, mejorar y encontrar a un amigo.

ÍNDICE

Índice de contenido

Índice de figuras

1. Introducción

En este momento internet se ha convertido en una de las herramientas más utilizadas en todo el mundo para el intercambio de información por parte de los usuarios. Los canales de comunicación están en continua mejora con el fin de aumentar la velocidad de envío y recepción de datos para su posterior tratamiento y conversión en información. Siguiendo esta línea, los servicios web son los que más han crecido debido a la cantidad de programas y estándares que se han creado para hacer el desarrollo web más sencillo y universal.

Viendo la tendencia mundial, surge la necesidad de crear un sistema de archivos que sea capaz de trabajar en grupo, esto es, que los usuarios puedan interactuar entre ellos por medio de una aplicación que les sirva de enlace y haga su utilización más sencilla.

Este proyecto surge de esa necesidad de guardar archivos y compartirlos con otros usuarios conocidos a los que les damos permiso para modificar nuestros archivos y descargarlos, y a su vez permitirnos acceder a las carpetas de esos usuarios.

1.1. Descripción del proyecto y objetivos a alcanzar

La interacción entre diferentes actores de un sistema esta a la orden del día en un campo tan amplio como es Internet. Es el futuro y no tiene una parada fija.

Sabemos que el proyecto va a estar encaminado a un uso de intranet para la Universidad. Partiendo de esta base y de nuestra experiencia como alumnos de la misma, intentamos buscar puntos vacíos que necesitaran de esta interacción.

Un disco duro virtual en que cada alumno o profesor tenga su espacio para mantener sus documentos y poder acceder a ellos desde cualquier parte es una de las necesidades encontradas y la cual elegimos como proyecto.

Puesto que una aplicación de gestión de usuarios con autentificación en la que cada uno disponga de su propio disco virtual carecía totalmente de interacción, se decide implantar un sistema de discos compartidos de manera que un usuario no

solo tenga acceso a sus propios archivos sino a los que otros usuarios decidieran compartirle y viceversa.

Todo este sistema requeriría de una capa de seguridad y por ende hemos decidido añadirle una capa de permisos de manera que esos documentos compartidos puedan tener ciertos límites.

En la universidad encontramos claramente dos actores completamente definidos y diferenciados. Aunque ambos son gestionados como usuarios del disco duro virtual estos podrían tener diferentes funcionalidades. Son los alumnos y los profesores.

Como hemos dicho anteriormente, en base a nuestra experiencia hemos localizamos **una de las mayores necesidades**. Cuando un profesor necesita proporcionarnos un artículo o el temario de manera electrónica recurre a servidores web o ftp gratuitos, o simplemente reúne una lista de direcciones de correo electrónico de sus alumnos (en el peor de los casos, sólo a uno) para enviarlo.

A su vez cuando un alumno necesita entregar una práctica u otro tipo de documento necesario para el profesor, éste proporciona su correo electrónico para que dichos alumnos se lo envíen. Esto trae consecuencias que si bien no son fatales pueden ser molestas: llenar el buzón de correo personal del profesor, archivos demasiado grandes o con formatos comprometidos para poderlos enviar, necesidad de identificación del alumno por el correo, evasión de las fechas de entrega…

Es por ello que se implementaron dos procesos:

1. El profesor podrá colocar los enunciados de las asignaturas siendo estos visibles con unos permisos restrictivos a los alumnos de dicha asignatura.

2. Un alumno podrá enviar prácticas o ejercicios al profesor para que éste disponga de ellos.

En este punto es donde decidimos realizar la base de la aplicación, de manera que pueda expandirse gradualmente con la ingente cantidad de módulos que se podrán añadir tales como:

1. Envío de correos
2. Sistema de mensajes de intranet
3. Cuotas de disco
4. Sincronización por FTP
5. Espacio web

1.2. Estado del arte

BSCW

BSCW (Basic Support for Cooperative Working) es un espacio de trabajo compartido, una aplicación general que permite usar este espacio de trabajo para compartir documentos a través de distintas plataformas (Windows, Macintosh o Unix). Se puede acceder a un espacio de trabajo, navegar a través de las carpetas, y obtener objetos de igual manera que en las páginas web ordinarias. Mantiene alerta de todos los sucesos acaecidos (creación, lectura o modificación de objetos), sin necesidad de instalar ningún tipo de software adicional, sólo un navegador de Internet ordinario.

Para poder utilizar BSCW, los requerimientos mínimos son: disponer de una dirección de correo electrónico POP3 (Post Office Protocol, versión 3) para registrarse como usuario del servidor público y disponer de un navegador de Internet que soporte formularios y autenticaciones básicas.

El objetivo que se plantea es el desarrollo de un sistema de trabajo compartido BSCW (BSCW shared workspace system) que proporcione facilidades para la cooperación de grupos cuyos miembros se encuentren muy alejados físicamente,

incluso en diferentes países y empleando diversas plataformas de trabajo, el soporte para la comunicación es Internet. Se pretende transformar la Web de un simple depósito pasivo de información a una herramienta activa de colaboración.

Microsoft Office Groove 2007

Es un software de escritorio diseñado para la colaboración y la comunicación de los miembros de pequeños grupos para la plataforma Microsoft Windows, que no es gratuito.

Groove es un workspace (espacio de trabajo) compartido. Un usuario Groove invita a otros miembros Groove después de crear un workspace. Al responder a una invitación de la persona se convierte en miembro activo de ese trabajo y se le envía una copia del Workspace que se instalará en el disco duro.

Todos los datos están cifrados tanto en el disco, como a través de la red, cada Workspace tiene un juego único de claves criptográficas. Los miembros interactúan y colaboran en el Workspace, que es un lugar privado virtual. Todos los cambios que se realizan son seguidos por Groove y todas las copias se sincronizan a través de la red de forma P2P y de forma casi instantánea.

Se envían a todos los miembros los cambios introducidos en el Workspace por cualquiera de los usuarios y los documentos se actualizan automáticamente. En caso de que un miembro esté off-line en el momento en que se haga el cambio, los cambios se ponen en cola y se sincronizan para otros miembros del Workspace, pero la copia de este miembro inactivo se actualiza cuando el miembro está de nuevo en línea.

Esnips

Disco duro virtual que requiere registrarse, pero que es gratuito. Permite 5GB de memoria con la posibilidad de compartir las carpetas y los archivos con otros usuarios a los que se invita a través del correo. De esta manera damos permisos sobre las carpetas a otros usuarios. Se puede compartir con un grupo de usuarios, o permitir que toda la comunidad tenga acceso a esos datos. A su vez

puede no permitirse que nadie tenga acceso a esos datos de manera que la carpeta sería privada.

También permite crear carpetas en las que se inserten fotos y presentarlas de manera continua, pasando de una a otra. También es posible crear una lista de reproducción de archivos de música.

Toda esta gestión se realiza a través del navegador web.

Wua.la

Esta página actúa como un servidor. Para poder utilizar este sistema, debemos bajarnos un cliente que actúa sobre Java para que no haya problemas de portabilidad entre diferentes sistemas operativos. Pero lo más importante de esta aplicación, es que para compartir los archivos utiliza la red P2P.

Hay que instalar la aplicación cliente en nuestro ordenador, ésta se integra perfectamente como una nueva carpeta que contiene los archivos privados, los archivos compartidos nuestros amigos y los archivos públicos compartidos con todos los usuarios de este sistema. No constituye un disco duro propiamente dicho, ya que toda la información la almacenamos en nuestro disco duro.

Cuando añadimos alguna carpeta o archivo y lo compartimos, la manera de sincronizarlo con el resto de usuarios se vale de la red P2P.

Arkiva

Disco duro virtual capaz de contener hasta 1 GB de memoria gratuita. Tiene una serie de carpetas reservadas para los videos, para las fotos, para la música y para los documentos. Ofrece una cuenta de correo especialmente diseñada para compartir el contenido que se suba en Arkiva y permite editar documentos office online.

Una buena opción a tener en cuenta a la hora de hacer backup de nuestra información.

Hordit

En Hordit se puede guardar de todo, desde archivos hasta links, clasificándolos en directorios y enviando una descripción de cada elemento incluido.

Hordit permite almacenar cualquier cosa y para ello ofrece:

Un espacio web para que centraliza todos los archivos dispersos por la web: fotografías, documentos, rss, marcadores.

- Un espacio ilimitado.
- Los archivos pueden ser privados o públicos.
- Una comunidad virtual.
- Un buscador de cosas y gentes con intereses comunes.

Lo curioso es que ofrecen espacio ilimitado gratis, con seguridad garantizada y sin miedo de perder los derechos del contenido enviado.

Hay que destacar que Hordit está en fase beta y sólo se puede acceder mediante un sistema algo complicado en el que debes registrar tu correo y ellos te envían un código para acceder.

- **Otros**

Otros discos duros que podemos nombrar por su relevancia son: Box, Dropbox, StreamLoad y Adrive. Cada uno tiene sus peculiaridades, ya sean de tamaño de espacio, de funcionalidad, gratuidad o la manera en que propagan los ficheros, pero todos siguen la misma pauta que los mencionados anteriormente.

2. Tecnologías y software empleado

2.1. MySql

Descripción

MySql es un sistema de gestión de bases de datos (SGBD) relacional, multihilo y multiusuario de código abierto licenciado bajo la licencia GPL (Licencia General Pública) de GNU. MySql fue creada por la empresa sueca MySql AB, que mantiene los derechos sobre el código fuente del servidor SQL, así como también de la marca. MySql AB es desde Enero de 2008 una empresa subsidiaria de Sun Microsystems. Aunque MySql es software libre, MySql AB distribuye una versión comercial de MySql, que únicamente se diferencia de la versión libre en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de no ser así, se vulneraría la licencia GPL. Está desarrollado en su mayor parte en C y C++.

Este gestor de bases de datos es, probablemente, el gestor más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración.

Justificación

- Escalabilidad: es posible manipular bases de datos enormes, del orden de seis mil tablas y alrededor de cincuenta millones de registros, y hasta 32 índices por tabla.

- MySql está escrito en C y C++ y probado con multitud de compiladores y dispone de APIs para muchas plataformas diferentes.

- Conectividad: es decir, permite conexiones entre diferentes máquinas con distintos sistemas operativos. Es corriente que servidores Linux o Unix, usando MySql, sirvan datos para ordenadores con Windows, Linux, Solaris, etc. Para ello se usa TCP/IP, tuberías, o sockets Unix.

- Es multihilo, con lo que puede beneficiarse de sistemas multiprocesador.
- Permite manejar multitud de tipos para columnas.
- Permite manejar registros de longitud fija o variable.
- Potencia: SQL es un lenguaje muy potente para consulta de bases de datos, usar un motor nos ahorra una enorme cantidad de trabajo.
- Portabilidad: SQL es también un lenguaje estandarizado, de modo que las consultas hechas usando SQL son fácilmente portables a otros sistemas y plataformas. Esto, unido al uso de C/C++ proporciona una portabilidad enorme.

2.2. Apache 2.2

Descripción

Apache es un servidor HTTP de código abierto (software libre) que puede trabajar sobre plataformas Unix, Windows, Macintosh y otras. Su desarrollo empezó en 1995 e inicialmente se basó en código de NCSA HTTPd 1.3, pero más tarde fue reescrito. El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation, que es una fundación no lucrativa creada para dar soporte a proyectos bajo la denominación Apache.

Apache tiene una alta aceptación en la red y es el servidor HTTP más usado en el mundo

Es más fácil usar un motor o servidor que hace las funciones de intérprete entre las aplicaciones y los usuarios con las bases de datos.

Esta utilidad se traduce en ventajas, entre las que podemos mencionar las siguientes:

- Acceso a las bases de datos de forma simultánea por varios usuarios y/o

aplicaciones.

- Seguridad, en forma de permisos y privilegios, determinados usuarios tendrán permiso para consulta o modificación de determinadas tablas. Esto permite compartir datos sin que peligre la integridad de la base de datos o protegiendo determinados contenidos.

Justificación

Siendo el servidor más usado en todo el mundo con clara superioridad (ref *http://news.netcraft.com/archives/2009/02/18/february_2009_web_server_survey.html*), junto con el hecho de ser libre no es de extrañar que decidiéramos usar este servidor web.

Claro que no solo esta razón nos impulso a usarla.

Teníamos cierta experiencia con este servidor web tanto en instalación y uso como en configuración, que junto con la inmensa cantidad de documentación disponible oficialmente identificamos claramente los puntos que íbamos a necesitar de un servidor web y este los cumplía a la perfección.

Se procede a detallarlos:

1. Soporte para gran cantidad de peticiones

2. Fácil ampliación de funcionalidades mediante módulos, entre ellos PHP, SSL…

3. Soporte unicode (UTF-8)

4. Reglas de seguridad para diferentes directorios.

5. Soporte de virtualhosts, algo imprescindible si se quería montar espacio web para cada profesor.

2.3. XHTML

Descripción

XHTML, acrónimo inglés de eXtensible Hypertext Markup Language (lenguaje extensible de marcado de hipertexto), es el lenguaje de marcado pensado para sustituir a HTML como estándar para las páginas web. En su versión 1.0, XHTML es solamente la versión XML de HTML, por lo que tiene, básicamente, las mismas funcionalidades, pero cumple las especificaciones, más estrictas, de XML. Su objetivo lograr una web semántica, donde la información, y la forma de presentarla estén claramente separadas. La versión 1.1 es similar, pero parte a la especificación en módulos. En sucesivas versiones la W3C planea romper con los tags clásicos traídos de HTML.

XHTML es una "reformulación de los tres tipos de documento definidos por HTML 4, pero como aplicaciones de XML". Al mismo tiempo el W3C continúa recomendando el uso de HTML 4 y al mismo tiempo avanza en la especificación de HTML 5 (y XHTML5).

Se sintió que era necesaria una versión más estricta de HTML principalmente porque el contenido de la World Wide Web ahora puede visualizarse desde numerosos dispositivos aparte de los ordenadores tradicionales, donde no se contaría con los recursos necesarios para afrontar la complejidad de la sintaxis del HTML. Sin embargo, en la práctica, fueron apareciendo navegadores para dispositivos móviles que pueden manejar documentos HTML comunes, antes que XHTML sea adoptado por los navegadores principales.

En octubre de 2005 aproximadamente el 10% de los internautas utilizaban un navegador compatible con el estándar XHTML. El navegador Internet Explorer de Microsoft es incompatible con XHTML, a pesar de que esta empresa sea miembro de la W3C. Por tanto, gran parte de los autores de sitios web se ven forzados a elegir entre la escritura de documentos válidos, respetuosos con los estándares u ofrecer contenido que se visualice correctamente en la mayor parte de los navegadores, del mismo modo, la corroboración de los específicos es vital para

las fluctuaciones pertinentes.

Las diferencias entre HTML y la primera generación de XHTML (es decir, XHTML 1.x) son menores ya que, principalmente, están destinados a conseguir la conformidad con XML. El cambio más importante es el requisito de que el documento esté bien formado y que todas las etiquetas estén explícitamente cerradas, como se requiere en XML.

Justificación

Como ya se ha indicado apostamos por las nuevas tecnologías. No hay una razón concreta por la que no usamos HTML 4.0 puesto que sigue siendo un estándar válido y recomendado.

Posiblemente el concepto de *la web 2.0* nos impulso en gran medida.

Por ello nos centramos en xHtml de manera que respetáramos la semántica independiente del contenido y puesto que también íbamos a usar otras tecnologías para definir claramente esta separación de conceptos no lo dudamos y a parte de usarla elegimos la rama STRICT que no soporta etiquetas ya obsoletas como *font frames marques*…

2.4. UML

Descripción

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, *Unified Modeling Language*) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y

componentes reutilizables.

Es importante resaltar que UML es un "lenguaje" para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo.

En UML 2.0 hay 13 tipos diferentes de diagramas.

Los Diagramas de Estructura enfatizan en los elementos que deben existir en el sistema modelado:

- Diagrama de clases
- Diagrama de componentes
- Diagrama de objetos
- Diagrama de estructura compuesta (UML 2.0)
- Diagrama de despliegue
- Diagrama de paquetes

Los Diagramas de Comportamiento enfatizan en lo que debe suceder en el sistema modelado:

- Diagrama de actividades
- Diagrama de casos de uso
- Diagrama de estado

Los Diagramas de Interacción son un subtipo de diagramas de comportamiento, que enfatiza sobre el flujo de control y de datos entre los elementos del sistema

modelado:

- Diagrama de secuencia
- Diagrama de comunicación
- Diagrama de tiempos (UML 2.0)
- Diagrama de vista de interacción (UMLL 2.0)

Justificación

Se ha utilizado este sistema porque es una buena manera de explicar de manera gráfica los puntos más importantes de la aplicación así como los usuarios que intervienen.

Se analizan las acciones que pertenecen al sistema y las que realizan los usuarios, de manera que se puede tener una visión más amplia de la aplicación, sin tener muchos conocimientos de informática. Es bastante intuitiva para el espectador, pero no es fácil de usar, ya que requiere un conocimiento avanzado de uso y abstracción sobre la aplicación para plasmar los procesos que se utilizan al nivel adecuado.

2.5. GNU/Linux Debian 5.0

Descripción

Partiremos en el momento de la historia en que el proyecto GNU iniciado por Richard Stallman tenía como objetivos desarrollar un sistema UNIX totalmente *libre* (con libre se refiere a la disponibilidad total del código fuente) bajo la licencia GPL.

A finales de los 80, el proyecto GNU ya había desarrollado muchos de los

componentes necesarios para el sistema operativo tales como un *intérprete de comandos*, *biblioteca C y compilador (gnu C compiler)* pero sin embargo necesitaba del núcleo del sistema.

Por esas fechas, un estudiante de la universidad de Helsinki empezó a desarrollar un sistema operativo desde cero como proyecto personal pues no le gustaba el sistema que utilizaba en su universidad (Minix) dando lugar a un núcleo capaz de mostrar una shell (intérprete de comandos) y ejecutar el *gnu C compiler* el cual fue liberado en 1991 como *opensource*.

Es en este punto cuando el proyecto GNU dispone del núcleo que le faltaba (pues aunque existía un proyecto denominado *Hurd,* éste se había estancado hace tiempo por discrepancias entre sus creadores) y surge a lo que actualmente se denomina como GNU/Linux. El sistema operativo funcional tal y como lo conocemos hoy en día.

Este sistema operativo crecerá exponencialmente a partir de aquí debido al *opensource* pues múltiples desarrolladores se unen al proyecto así como diferentes empresas formando lo que se conoce actualmente como distribuciones.

Justificación

El sistema operativo Linux está presente en un gran número en el ámbito de servidores, no solo por pertenecer al grupo denominado *opensource* que abarata los costes de despliegue y mantenimiento sino por su gran capacidad de configuración, estabilidad y seguridad.

Aunque podríamos decir que Linux es la base del sistema operativo, más conocido por el *kernel*, GNU/Linux consiste en la agrupación de dicho kernel y las diferentes herramientas básicas y adicionales que necesita una distribución.

Llamaremos **distribución** a la mezcla final de todo el núcleo básico junto con una gran compilación de paquetes software que pueden ser tanto *opensource* como privativo.

Existen un elevado número de distribuciones Linux en el mercado, destacando diferentes características u orientándola a la funcionalidad (servidor, escritorio, media system…)

Nosotros hemos escogido **Debian** en su versión 5.0 con el alias *Lenny* que ha sido marcada como estable el 14 de febrero del 2009.

Debian ha mantenido siempre una **política de seguridad ante la innovación** y puesto que en nuestro proyecto necesitamos esta misma política se ha considerado la opción más correcta.

Sin embargo, esto no quiere decir que vayamos a usar paquetes obsoletos.

Debian diferencia la inmensa cantidad de paquetes software de la que dispone en tres ramas diferentes:

1. stable
2. testing
3. unstable

No vamos a entrar a explicar detalladamente cada rama y la gestión de versiones que realiza. Nos basta con saber que se ha escogido la rama *stable* la cual contiene las últimas ramas de versión de los paquetes software que necesitaremos para este proyecto, las cuales a su vez están marcadas como estables en sus determinados repositorios.

Comentar también la facilidad de instalación de paquetes software gracias a la herramienta *apt* característica de esta distribución que está siendo utilizada en un gran número de distribuciones.

En resumen, se ha escogido GNU/Linux Debian 5.0 por las siguientes razones:

1. Seguridad

2. Estabilidad

3. Amplia línea de configuración y mantenimiento

4. Facilidad de instalación tanto de la distribución elegida como de las herramientas software a utilizar.

Como último detalle comentar que puesto que todo el software utilizado es *opensource* no podríamos (moralmente) ejecutarlo sobre otro sistema operativo como: Windows o Mac OS.

2.6. PHP 5

Descripción

PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor pero actualmente puede ser utilizado desde una interfaz de línea de comando.

PHP significa P*HP Hypertext Pre-processor*. Fue creado por Rasmus Lerdorf en 1994; sin embargo la implementación principal de PHP es producida ahora por The PHP Group y sirve como el estándar para PHP al no haber una especificación formal. Publicado bajo la PHP License, la Free Software Foundation considera esta licencia como software libre.

PHP puede ser incrustado dentro de código HTML. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. Es también el módulo Apache más popular entre las computadoras que utilizan Apache como servidor web.

Ventajas

- Es un lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de base de datos que se utilizan en la actualidad.
- Gran potencial utilizando la enorme cantidad de módulos.
- Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de Programación Orientada a Objetos.
- Biblioteca nativa de funciones incluida.
- No requiere definición de tipos de variables.
- Tiene manejo de excepciones (desde PHP5).,

Desventajas

Si bien PHP no obliga a quien lo usa a seguir una determinada metodología a la hora de programar, el programador puede aplicar en su trabajo cualquier técnica de programación y/o desarrollo que le permita escribir código ordenado, estructurado y manejable. Un ejemplo de esto son los desarrollos que en PHP se han hecho del patrón de diseño Modelo Vista Controlador (o MVC), que permiten separar el tratamiento y acceso a los datos, la lógica de control y la interfaz de usuario.

Dentro de PHP podemos encontrar PHP Data Objects (o PDO) que es una

extensión que provee una capa de abstracción de acceso a datos para PHP 5, con lo cual se consigue hacer uso de las mismas funciones para hacer consultas y obtener datos de distintos manejadores de bases de datos.

PHP 6

Está previsto el lanzamiento en breve de la rama 6 de PHP. Cuando se lance esta nueva versión quedarán solo dos ramas activas en desarrollo (PHP 5 y 6)

Las diferencias que encontraremos frente a PHP 5.* son:

- soportará Unicode.
- limpieza de funcionalidades obsoletas como *register_globals*, *safe_mode*, etc.
- PECL.
- mejoras en orientación a objetos.

PHP Data Objects (o PDO) es una extensión que provee una capa de abstracción de acceso a datos para PHP 5, con lo cual se consigue hacer uso de las mismas funciones para hacer consultas y obtener datos de distintos manejadores de bases de datos.

Justificación

Ya desde sus primeras versiones se veía que este lenguaje de programación tendría un futuro y que iba a calar fuerte en la comunidad, y no ha defraudado en absoluto.

La simplicidad en la programación con características como la falta de tipos de datos hace que PHP sea fácil de aprender aunque como ya se ha dicho trae otro tipo de desventajas tales como la falta de diseño lógico con la consecuente falta de seguridad y consistencia.

No sólo su simplicidad si no; la posibilidad del código embebido con (x)html, ampliación mediante módulos, conexión con todo tipo de base de datos han hecho de PHP el lenguaje de scripting para servidores Web; una de las soluciones más factibles tanto a nivel autónomo como empresa. Es una de las soluciones preferidas por las Pymes que encuentran en PHP la relación calidad/coste ideal.

Con la llegada de la versión PHP5, se amplia el reducido concepto de *programación orientada a objetos* de la que disponía su versión PHP4.

Polimorfismo, herencia, encapsulación, constructores y destructores, métodos mágicos, clonado, excepciones son solo algunos de los nuevos conceptos de ingeniería de los que disponía la nueva versión.

Con ella llegaron toda una batería de librerías a su alrededor tales como SPL, PDO (dos de las que usamos) incluidas dentro de PHP o activadas mediante módulos, así como una infinidad de nuevas clases diseñadas por la comunidad *opensource* para la gestión de todo tipo de tareas. (phpclases.org, PEAR…)

Tres de estas *librerías* son usadas con frecuencia en nuestra aplicación.

La primera de ellas, **SPL** (Standard PHP library) fue centro de un esfuerzo de documentación bastante extenso debido a los patrones de programación usados en dicha API que se describirán más adelante en la sección: *Análisis y diseño*.

La segunda fue **PDO**, una librería de abstracción de base de datos con la cual solventamos dos problemas graves que nos encontramos durante la fase de diseño: *inyección SQL* y el posible uso futuro de otra base de datos diferente a MYSQL. PDO además estaba intensamente relacionada con uno de los patrones de diseño de SPL obteniendo por añadido otro gran número de ventajas.

La última fue **SMARTY**, un sistema de plantillas programado en PHP de manera que la separación entre programación y diseño nos alejo del código embebido pudiendo diseñar una aplicación limpia donde los futuros cambios en uno u otro campo se hicieran de manera independiente obteniendo unos flujos de trabajo

más estables y reduciendo costes.

El MVC y el uso de estas tres tecnologías fueron el centro de nuestra amplia formación en la fase de diseño que desembocó en la fortaleza del núcleo de nuestra aplicación.

Todas estas tecnologías y su uso serán descritas más adelante en la sección de *Análisis y diseño* de una manera más amplia.

2.7. Smarty

Descripción

Smarty es un motor de plantillas para PHP. Smarty separa el código PHP, desarrollo, del código HTML, presentación, y genera contenidos web mediante la colocación de *etiquetas Smarty* en un documento. Se encuentra bajo la licencia LGPL por lo que puede ser usado libremente.

Smarty permite programar plantillas con un gran número de funcionalidades, como:

- expresiones regulares
- bucles *foreach*, *while*
- *if*, *elseif*, *else*
- modificadores de variables
- funciones creadas por el usuario
- evaluación de expresiones matemáticas en la plantilla

Justificación

Es uno de los retos que nos hemos propuesto y creemos que uno de los puntos fuertes de nuestra aplicación, ya que es capaz de separar el código de programación, de la parte de presentación, con lo que se generan archivos de código más pequeños y sencillos de interpretar por otro programador.

2.8. PhpDocumentor

Descripción

Es un sistema para crear documentación de aplicaciones creadas con PHP. PhpDocumentor está escrito en PHP. Puede usarse directamente por línea de comandos o a través de una interfaz web. Crea una documentación profesional a partir del código PHP. Soporta links entre documentación, herencia automática de clases de programación orientada a objetos. Generación de código fuente resaltado con referencias a la documentación. Funciona por plantillas, que se pueden extender, pero que ya cuenta con maneras diferentes de mostrar la documentación. También soporta formato PDF para generar las documentaciones, así como otros formatos como el sistema de ayuda de Windows (formato CHM), formato XML DocBook...

Justificación

Bien conocida es la necesidad de un manual del programador para el futuro de la aplicación.

Es por ello que si bien creamos un pequeño *framework* adaptado a nuestras necesidades era necesaria la documentación del mismo. Phpdocumentor se encarga de formar la API con formato html de manera que el futuro programador que necesite ampliar la aplicación pueda formarse fácil y rápidamente gracias a la descripción de cada elemento usado y su interacción por enlaces.

2.9. CSS

Descripción

Las hojas de estilo en cascada (*Cascading Style Sheets*, CSS) son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C (World Wide Web Consortium) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores.

La idea que se encuentra detrás del desarrollo de CSS es separar la *estructura* de un documento de su *presentación*.

Por ejemplo, el elemento de HTML <H1> indica que un bloque de texto es un encabezamiento y que es más importante que un bloque etiquetado como <H2>. Versiones más antiguas de HTML permitían atributos extra dentro de la etiqueta abierta para darle formato (como el color o el tamaño de fuente). No obstante, cada etiqueta <H1> debía disponer de la información si se deseaba un diseño consistente para una página y, además, una persona que lea esa página con un navegador pierde totalmente el control sobre la visualización del texto.

Cuando se utiliza CSS, la etiqueta <H1> no debería proporcionar información sobre como va a ser visualizado, solamente marca la estructura del documento. La información de estilo separada en una hoja de estilo, especifica cómo se ha de mostrar <H1>: color, fuente, alineación del texto, tamaño y otras características no visuales.

La información de estilo puede ser adjuntada tanto como un documento separado o en el mismo documento HTML. En este último caso podrían definirse estilos generales en la cabecera del documento o en cada etiqueta particular mediante el atributo "*style*".

Justificación

Separamos completamente el contenido semántico del diseño dando lugar a un código xhtml limpio fácilmente editable.

Bien conocido es el gran problema de los diseñadores web respecto al respeto del estándar CSS por los diferentes navegadores y sus versiones. Debido a esto centramos nuestro esfuerzo en la creación de una interfaz sencilla, usable e intuitiva que respetara el estándar CSS2.

En última instancia se añadieron diferentes *cheats* para el polémico navegador 'Internet Explorer' y sus anteriores versiones, y alguna propiedad CSS3 soportada por los navegadores más modernos que si bien no afectaban en gran medida a la aplicación supuso un sacrificio de validación CSS2.

2.10.PhpMyAdmin

Descripción

Es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web, utilizando internet. Actualmente puede crear y eliminar Bases de Datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar claves en campos, administrar privilegios, exportar datos en varios formatos. Se encuentra disponible bajo la licencia GPL.

Como esta herramienta corre en máquinas con Servidores Web y Soporte de PHP y MySQL, la tecnología utilizada ha ido variando durante su desarrollo.

Justificación

Buen gestor gráfico de base de datos capaz de lanzar comandos SQL desde muchas partes del programa. Bastante sencillo de utilizar, es una buena opción a la hora de manejar tablas y campos de una base de datos, ya que ahorra mucho tiempo y facilita la administración.

3. Especificación de Requisitos

3.1. Descripción del problema

Ya a la hora de enfrentarnos con la decisión de qué proyecto escoger, nos dimos cuenta que existía un vacío en la Universidad, por lo menos a nivel de alumnos, a la hora de gestionar las prácticas. Viendo esta necesidad que existía, no sólo se centraba en las prácticas, si no que se extendía al uso de lapiceros USB para manejar toda la información de los usuarios entre la Universidad y su casa; prácticas que no llegan porque las cuentas de correo están llenas, olvidos… El proyecto se extendió de un simple gestor de prácticas, a un disco duro virtual en toda regla. Este disco duro virtual debería poder funcionar para los alumnos con los profesores y entre iguales, es decir alumnos con alumnos, y profesores con profesores

3.2. Especificación de Requisitos Software (ERS)

a) Descripción de la aplicación

Esta aplicación permite la interacción por parte del usuario mediante la red para poder acceder a la página web, donde podrá subir sus archivos y guardarlos en carpetas jerárquicamente organizadas. La mayor potencia de la aplicación consiste en que cada usuario introduce unos amigos a los que comparte archivos con unos permisos determinados para poder trabajar en grupo y de manera sencilla, sin tener que generar cantidades de información duplicada por la red. Otra parte del potencial de la aplicación es la posibilidad, por parte de los profesores, de colgar enunciados y que automáticamente todos los usuarios matriculados en esa asignatura puedan descargar dicho enunciado, a parte de poder subir las prácticas correspondientes únicamente a ese profesor.

b) Requisitos software

OBJETIVOS DEL SISTEMA

OBJ - 01	**Gestionar los usuarios**

OBJ - 01	**Gestionar los usuarios**
Descripción	El sistema deberá gestionar de forma eficiente a los usuarios, agrupándolos por profesores, alumnos y el administrador.

OBJ - 02	**Gestionar las asignaturas**
Descripción	El sistema deberá ser capaz de dar de alta, baja y modificaciones las asignaturas de los diferentes cursos de cada carrera

OBJ - 03	**Gestionar los ficheros**
Descripción	El sistema debe ser capaz de gestionar los ficheros subidos y almacenados por cada usuario

OBJ - 04	**Gestionar los compartidos**
Descripción	El sistema debe ser capaz de gestionar los archivos que se comparten con otros usuarios

REQUISITOS DE ALMACENAMIENTO DE INFORMACION

RI - 01	**Información sobre los ficheros**
Objetivos asociados	● OBJ – 3 Gestionar los ficheros ● OBJ – 4 Gestionar los compartidos
Requisitos asociados	● RF – 01 Crear carpeta ● RF – 02 Eliminar fichero ● RF – 03 Eliminar carpeta

RI - 01	**Información sobre los ficheros**
	● RF – 04 Subir fichero ● RF – 05 Bajar fichero ● RF – 06 Renombrar fichero ● RF – 07 Renombrar carpeta ● RF – 08 Compartir carpeta ● RF – 12 Compartir fichero ● RF – 13 Descompartir fichero ● RF – 14 Consultar usuarios que me comparten ● RF – 15 Consultar usuarios a los que comparto ● RF – 16 Consultar archivos locales ● RF – 17 Consultar archivos compartidos ● RF – 18 Consultar archivos docentes ● RF – 21 Permisos
Descripción	El sistema debe ser capaz de gestionar los ficheros: subidas, bajadas, etc.
Datos específicos	• Número de fichero, que deberá ser único • Padre • Ruta

RI 02	**Información sobre los usuarios**
Objetivos asociados	● OBJ – 1 Gestionar los usuarios ● OBJ – 3 Gestionar los ficheros ● OBJ – 4 Gestionar los compartidos
Requisitos asociados	● RF – 09 Consultar asignaturas de usuario ● RF – 10 Activar docencia ● RF – 11 Identificación usuario ● RF – 12 Compartir fichero ● RF – 13 Descompartir fichero ● RF – 14 Consultar usuarios que me comparten

RI 02	**Información sobre los usuarios**
	● RF – 15 Consultar usuarios a los que comparto ● RF – 16 Consultar archivos locales ● RF – 17 Consultar archivos compartidos ● RF – 18 Consultar archivos docentes ● RF – 19 Consultar alumnos ● RF – 21 Consultar profesores ● RF – 21 Permisos
Descripción	El sistema debe ser capaz de gestionar a los usuarios: altas, bajas, modificaciones de datos, cuentas, compartir carpetas, etc.
Datos específicos	• Número de usuario, que deberá ser único • Número de Alumno o Profesor • Nombre y apellidos • Correo • Usuario y password • Curso y Carrera • Asignaturas

RI 03	**Información sobre las cuentas de usuario**
Objetivos asociados	● OBJ – 1 Gestionar los usuarios ● OBJ – 2 Gestionar las asignaturas ● OBJ – 3 Gestionar los ficheros ● OBJ – 4 Gestionar los compartidos
Requisitos asociados	● RF – 08 Compartir carpeta ● RF – 09 Consultar asignaturas de usuario ● RF – 10 Activar docencia ● RF – 11 Identificación usuario ● RF – 12 Compartir fichero ● RF – 13 Descompartir fichero ● RF – 14 Consultar usuarios que me comparten

RI 03	**Información sobre las cuentas de usuario**
	● RF – 15 Consultar usuarios a los que comparto ● RF – 16 Consultar archivos locales ● RF – 17 Consultar archivos compartidos ● RF – 18 Consultar archivos docentes ● RF – 19 Consultar alumnos ● RF – 20 Consultar profesores ● RF – 21 Permisos
Descripción	El sistema deberá almacenar la información correspondiente a las cuentas de usuario y sus compartidos
Datos específicos	• Espacio en disco • Ver archivos locales • Ver archivos compartidos • Ver archivos docentes • Ver usuarios que me comparten • Permisos de los ficheros • Usuarios a los que se comparte

DEFINICIÓN DE ACTORES

ACTOR-01	**Administrador**
Descripción	Este actor representa al administrador del sistema

ACTOR-02	**Profesor**
Descripción	Este actor representa a los profesores del sistema

ACTOR-03	**Alumno**
Descripción	Este actor representa a los alumnos del sistema

REQUISITOS FUNCIONALES

Requisitos Funcionales - 01	**Crear carpeta**
Objetivos asociados	OBJ – 03 Gestionar los ficheros
Requisitos asociados	RI – 01 Información sobre los ficheros
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario solicite la creación de carpetas
Precondición	Deber ser un usuario del sistema y tener permisos para tal uso
Secuencia normal	El usuario crea una carpeta y le da un nombre

Requisitos Funcionales - 02	**Eliminar fichero**
Objetivos asociados	OBJ – 03 Gestionar los ficheros
Requisitos asociados	RI – 01 Información sobre los ficheros
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario solicite la eliminación de un fichero
Precondición	Deber ser un usuario del sistema y tener permisos para tal uso
Secuencia normal	El usuario pincha en el enlace para eliminar un fichero y este se elimina.

Requisitos Funcionales - 03	**Eliminar carpeta**
Objetivos asociados	OBJ – 03 Gestionar los ficheros
Requisitos asociados	RI – 01 Información sobre los ficheros
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario solicite la eliminación de una carpeta
Precondición	Deber ser un usuario del sistema y tener permisos para tal uso
Secuencia normal	El usuario pincha en el enlace para eliminar una carpeta y esta se elimina con todo su contenido

Requisitos Funcionales - 04	**Subir archivo**
Objetivos asociados	OBJ – 03 Gestionar los ficheros
Requisitos asociados	RI – 01 Información sobre los ficheros
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario solicite la subida de un fichero
Precondición	Deber ser un usuario del sistema y tener permisos para tal uso
Secuencia normal	El sistema pide qué archivo se quiere subir y se acepta el archivo

Requisitos Funcionales - 05	**Bajar archivo**

Requisitos Funcionales - 05	**Bajar archivo**
Objetivos asociados	OBJ – 03 Gestionar los ficheros
Requisitos asociados	RI – 01 Información sobre los ficheros
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario solicite la descarga de un archivo
Precondición	Deber ser un usuario del sistema y tener permisos para tal uso
Secuencia normal	Se pincha en el archivo y se selecciona el lugar a descargar

Requisitos Funcionales - 06	**Renombrar archivo**
Objetivos asociados	OBJ – 03 Gestionar los ficheros
Requisitos asociados	RI – 01 Información sobre los ficheros
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario solicite la modificación de un fichero
Precondición	Deber ser un usuario del sistema y tener permisos para tal uso
Secuencia normal	Se pide un nombre nuevo para el fichero

Requisitos Funcionales - 07	**Renombrar carpeta**
Objetivos asociados	OBJ – 03 Gestionar los ficheros
Requisitos asociados	RI – 01 Información sobre los ficheros
Descripción	El sistema deberá comportarse tal como se

Requisitos Funcionales - 07	**Renombrar carpeta**
	describe en el siguiente caso de uso cuando un usuario solicite la modificación de una carpeta
Precondición	Deber ser un usuario del sistema y tener permisos para tal uso
Secuencia normal	Se pide un nombre nuevo para la carpeta

Requisitos Funcionales - 08	**Compartir carpeta**
Objetivos asociados	OBJ – 03 Gestionar los archivos
Requisitos asociados	RI – 01 Información sobre los ficheros RI – 02 Información sobre los usuarios
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario solicite la modificación de una carpeta
Precondición	Deber ser un usuario del sistema y tener permisos para tal uso
Secuencia normal	Aparece un nuevo formulario en el que se piden los permisos sobre la carpeta y los usuarios a los que compartir

Requisitos Funcionales - 09	**Consultar asignaturas de usuario**
Objetivos asociados	OBJ – 02 Gestionar las asignaturas
Requisitos asociados	RI – 02 Información sobre los usuarios
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se consulten las asignaturas para

Requisitos Funcionales - 09	**Consultar asignaturas de usuario**
	un usuario
Precondición	Deber ser un usuario del sistema y tener permisos para tal uso
Secuencia normal	En el disco docente aparecen las asignaturas en las que el usuario está matriculado

Requisitos Funcionales - 10	**Activar docencia**
Objetivos asociados	OBJ – 01 Gestionar los usuarios OBJ – 02 Gestionar asignaturas OBJ – 03 Gestionar los ficheros OBJ – 04 Gestionar compartidos
Requisitos asociados	RI – 01 Información sobre los ficheros RI – 02 Información sobre los usuarios RI – 03 Información sobre las cuentas de usuario
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso durante la realización de los casos de uso:
Precondición	Deber ser un usuario profesor del sistema, tener permisos, nombre de usuario y clave
Secuencia normal	El profesor tiene la opción de activar la docencia

Requisitos Funcionales - 11	**Identificación usuario**
Objetivos asociados	OBJ – 01 Gestionar los usuarios
Requisitos asociados	RI – 02 Información sobre los usuarios

Requisitos Funcionales - 11	**Identificación usuario**
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso durante la realización de los casos de uso:
Precondición	Deber ser un usuario del sistema, tener permisos, nombre de usuario y clave
Secuencia normal	El usuario puede ver una serie de datos que se tienen almacenados respecto a su perfil, como pueden ser el nombre y apellidos o el correo electrónico.

Requisitos Funcionales - 12	**Compartir fichero**
Objetivos asociados	OBJ – 03 Gestionar los ficheros OBJ – 04 Gestionar los compartidos
Requisitos asociados	RI – 01 Información de ficheros RI – 02 Información sobre los usuarios RI – 03 Información sobre las cuentas de usuario
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se comparta un fichero
Precondición	Deber ser un usuario del sistema y propietario del fichero, ya que todos los usuarios pueden compartir con otros.
Secuencia normal	El usuario elige los amigos a los que puede compartir el fichero

Requisitos Funcionales - 13	**Descompartir fichero**
Objetivos asociados	OBJ – 03 Gestionar los ficheros OBJ – 04 Gestionar los compartidos
Requisitos asociados	RI – 01 Información de ficheros RI – 02 Información sobre los usuarios RI – 03 Información sobre las cuentas de usuario
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se deje de compartir un fichero
Precondición	Deber ser un usuario del sistema y ser propietario del fichero, ya que todos los usuarios pueden dejar de compartir con otros.
Secuencia normal	El usuario elige los usuarios a los que quiere dejar de compartir o deja de compartir el fichero totalmente.

Requisitos Funcionales - 14	**Consultar usuarios que me comparten**
Objetivos asociados	OBJ – 01 Gestionar los usuarios OBJ – 04 Gestionar los compartidos
Requisitos asociados	RI – 01 Información de ficheros RI – 02 Información sobre los usuarios RI – 03 Información sobre las cuentas de usuario
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se consulten los usuarios que me comparten algún fichero

Requisitos Funcionales - 14	**Consultar usuarios que me comparten**
Precondición	Deber ser un usuario del sistema y que algún amigo le haya compartido un fichero o una carpeta
Secuencia normal	En disco compartido aparecen los nombres de los usuario que han compartido algún archivo con el usuario

Requisitos Funcionales - 15	**Consultar los usuarios a los que comparto**
Objetivos asociados	OBJ – 01 Gestionar los usuarios OBJ – 04 Gestionar los compartidos
Requisitos asociados	RI – 01 Información de ficheros RI – 02 Información sobre los usuarios RI – 03 Información sobre las cuentas de usuario
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando comparta un fichero
Precondición	Deber ser un usuario del sistema y tener algún archivo compartido
Secuencia normal	El usuario puede ver los archivos y consultar con quién los ha compartido

Requisitos Funcionales - 16	**Consultar archivos locales**
Objetivos asociados	OBJ – 03 Gestionar los ficheros
Requisitos asociados	RI – 01 Información de ficheros RI – 02 Información sobre los usuarios

Requisitos Funcionales - 16	**Consultar archivos locales**
	RI – 03 Información sobre las cuentas de usuario
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando consulten los archivos del propio usuario
Precondición	Deber ser un usuario del sistema
Secuencia normal	En el disco local aparecen todas las carpetas y ficheros que se han subido con su orden jerárquico correspondiente

Requisitos Funcionales - 17	**Consultar archivos compartidos**
Objetivos asociados	OBJ – 04 Gestionar los compartidos
Requisitos asociados	RI – 01 Información de ficheros RI – 02 Información sobre los usuarios RI – 03 Información sobre las cuentas de usuario
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando consulten los archivos que le comparten
Precondición	Deber ser un usuario del sistema y que hayan compartido algún archivo con él
Secuencia normal	En el disco compartidos aparecen todas las carpetas y ficheros que le han compartido con su orden jerárquico correspondiente

Requisitos Funcionales - 18	**Consultar archivos docentes**
Objetivos asociados	OBJ – 03 Gestionar los ficheros OBJ – 04 Gestionar los compartidos
Requisitos asociados	RI – 01 Información de ficheros RI – 02 Información sobre los usuarios RI – 03 Información sobre las cuentas de usuario
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando consulten los enunciados que el profesor suba
Precondición	Deber ser un usuario del sistema y estar matriculado en la asignatura a consultar
Secuencia normal	En el disco docente, dentro de la asignatura elegida, le aparecen todos los ficheros que el profesor ha subido

Requisitos Funcionales – 19	**Consultar alumnos**
Objetivos asociados	OBJ – 01 Gestionar los usuarios
Requisitos asociados	RI – 02 Información sobre los usuarios
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario solicite ver los alumnos
Precondición	Deber ser un usuario profesor del sistema y tener activada la docencia
Secuencia normal	El profesor consulta el disco docente y en cada asignatura aparecen los alumnos que

Requisitos Funcionales – 19	**Consultar alumnos**
	están matriculados en ella

Requisitos Funcionales - 20	**Consultar profesores**
Objetivos asociados	OBJ – 01 Gestionar los usuarios
Requisitos asociados	RI – 02 Información sobre los usuarios
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario solicite ver los profesores que le imparten clase
Precondición	Deber ser un usuario alumno del sistema y estar matriculado en las asignaturas
Secuencia normal	El alumno consulta el disco decente y en cada asignatura aparece el profesor que la imparte

Requisitos Funcionales - 21	**Permisos**
Objetivos asociados	OBJ – 01 Gestionar los usuarios OBJ – 03 Gestionar los archivos OBJ – 04 Gestionar los compartidos
Requisitos asociados	RI – 01 Información sobre los archivos RI – 02 Información sobre los usuarios
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario solicite dar permisos
Precondición	Deber ser un usuario del sistema y tener archivos a compartir
Secuencia normal	En el momento que un usuario decide

Requisitos Funcionales - 21	**Permisos**
	compartir una carpeta o un fichero, deberá elegir el tipo de permisos que aplicará y que serán válidos para los usuarios a los que se comparte

4. Análisis y Diseño

4.1. Diseño del diagrama de tareas

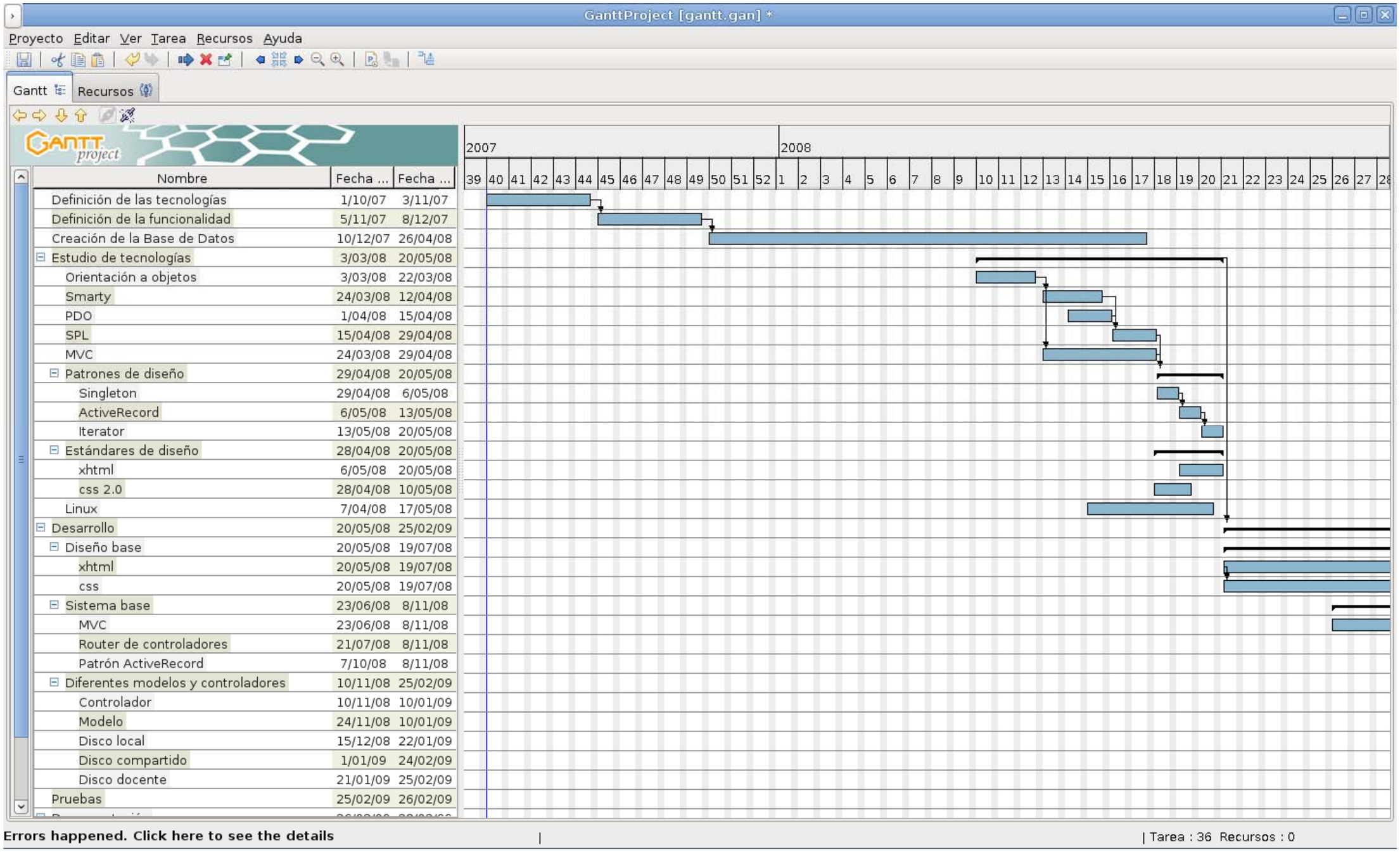


Figura 1. Diagrama de Gantt

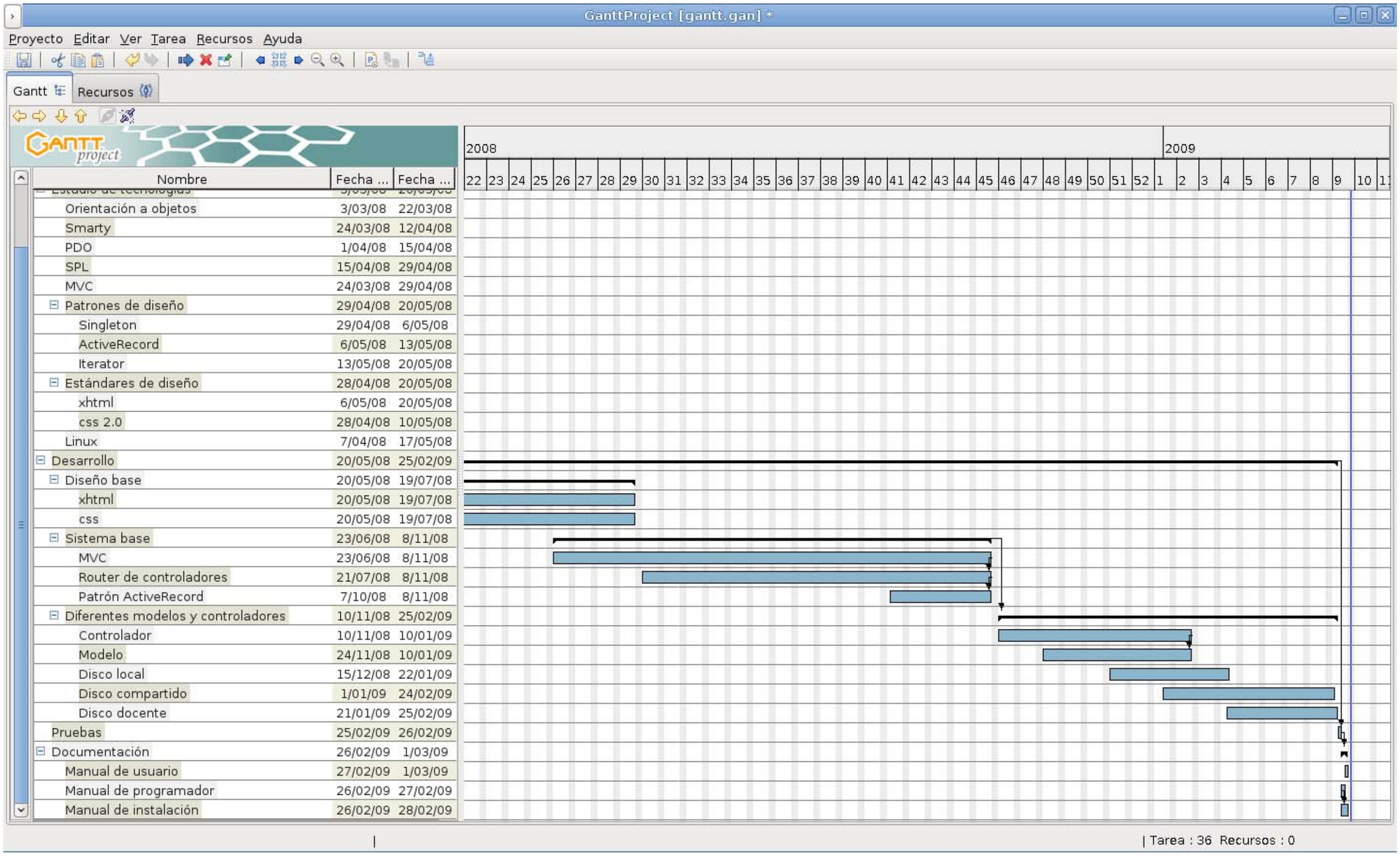


Figura 2. Diagrama de Gantt (continuación)

4.2. Análisis de la estructura de la base de datos

En un inicio el proyecto iba a constar de 3 partes bien diferenciadas: un sistema de *wikis*, un disco virtual y espacio web.

Cuando comenzamos a investigar las tecnologías a emplear, empezamos a dar vueltas a la funcionalidad sobre todo del disco duro. Lo que en principio iba a ser un simple sistema de almacenamiento de información con respecto a un usuario, se convirtió en un sistema de disco virtual con entidad propia.

Lo primero que hicimos fue diferenciar a los alumnos de los profesores, y esto nos hizo darnos cuenta de que ambos actores del sistema podían interactuar entre ellos con sus respectivas particularidades. Estaba claro que los usuarios iban a tener sus archivos y carpetas ordenados como ellos quisieran, pero ¿y si se creara una carpeta con el nombre de una asignatura y todos los usuarios que la tuvieran pudieran acceder a ella?

Éste fue el punto de inflexión que lo cambió todo. Nos dimos cuenta que ambos usuarios iban a tener asignaturas en común, ya fuera porque las impartieran o porque las cursaran. Empezamos a pensar en que esos usuarios iban a estar juntos de alguna manera, y ahí fue donde con los conocimientos que teníamos, más los que nos aportó nuestro tutor sobre cómo gestiona Linux su sistema de archivos, se empezó a dar forma a la base de datos.

Asignaturas, usuarios y grupos tenían ya una tabla propia, a lo que había que añadir el curso y la carrera a la que pertenecía esa asignatura. Llegados a este punto, se eliminaron del proyecto el sistema de wikis y el espacio web, ya que el trabajo a realizar era lo bastante grande para considerarlo como proyecto y los costes en tiempo se excedían del limite.

Para compartir una carpeta o un fichero, éste debería tener un creador y algún usuario al que compartir y guardar esa información en la base de datos para poder tener sus relaciones. Si se tienen archivos, deberemos tener permisos sobre esos archivos. Con lo que se crea una nueva tabla con todos los permisos que se pueden tener.

Con todas estas tablas y las que surgen de las relaciones *n-arias* se conforma

nuestra base de datos.

La base de datos está pensada para que se pueda modificar o añadir tablas sin que cambiase su configuración. En posibles mejoras de nuestra aplicación y para las cuales se necesiten crear tablas que intervengan en el sistema, es sencillo adaptar la base de datos sin tener que modificar la estructura ya existente.

A simple vista la base de datos es muy intuitiva y no parece que conlleve mucha complejidad, pero hasta llegar a un modelo que contuviera todas las opciones pensadas y que fuera fácilmente ampliable para el desarrollador, fueron necesarias muchas horas haciendo posibles pruebas sobre si las relaciones aguantarían este modelo.

Resaltar también que la base de datos no solo se comporta como sistema de almacenamiento de datos sino que tiene una base semántica para relación entre estos datos. Usamos un amplio potencial de MYSQL.

Se hicieron cambios bastante sustanciales que han dado como resultado una base de datos con 13 tablas que más adelante procederemos a explicar.

4.3. Diseño de la Base de Datos

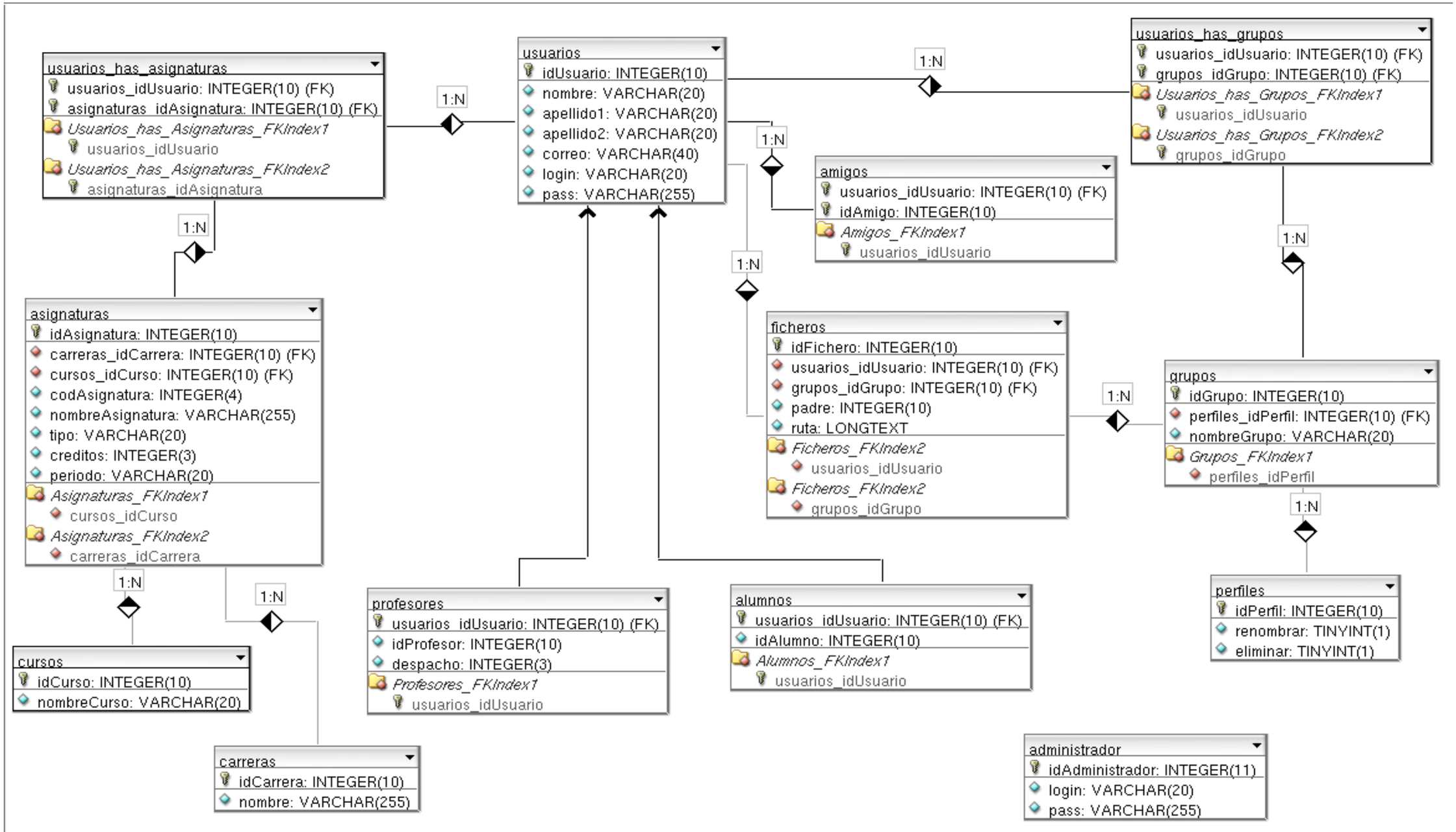


Figura 3. Diseño de la Base de Datos

4.4. Descripción de las tablas de la BD

administrador

Campo	Tipo	
IdAdministrador	Integer (11)	Número autoincrementable que empieza en 1 y no puede tomar valores negativos
login	Varchar (20)	Cadena de texto que contiene el nombre de usuario
pass	Varchar (255)	Cadena de texto que contiene la clave encriptada

La tabla administrador contiene un campo idAdministrador que es la clave principal de la tabla y dos campos más correspondientes al login y pass del administrador. En caso de que haya más de un administrador es la tabla que lo verifica.

```
-- Estructura de tabla para la tabla `administrador`
--


CREATE TABLE `administrador` (
  `idAdministrador` int(11) NOT NULL auto_increment,
  `login` varchar(20) collate utf8_unicode_ci NOT NULL,
  `pass` varchar(255) collate utf8_unicode_ci NOT NULL,
  PRIMARY KEY  (`idAdministrador`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
AUTO_INCREMENT=1 ;
```

alumnos

Campo	Tipo	Significado
IdAlumno	Integer (10)	Número autoincrementable que empieza en 1 y no puede tomar valores negativos
usuarios_idUsuario	Integer (10) FK	Número que no puede tomar valores negativos y que es clave foránea proveniente de la tabla usuarios.

La tabla alumnos contiene un campo usuarios_idUsuario que es una clave foránea de la tabla usuarios, que se verá más adelante, y que es clave principal de la tabla. El campo idAlumno es el que identifica que un usuario es un alumno. También se crea un índice con el campo usuarios_idUsuario para una búsqueda más rápida.

```
-- Estructura de tabla para la tabla `alumnos`
--

CREATE TABLE `alumnos` (
  `idAlumno` int(10) NOT NULL auto_increment,
  `usuarios_idUsuario` int(10) NOT NULL,
  PRIMARY KEY  (`idAlumno`,`usuarios_idUsuario`),
  KEY `Alumnos_FKIndex1` (`usuarios_idUsuario`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci AUTO_INCREMENT=1 ;
```

amigos

Campo	Tipo	Significado
idAmigo	Integer (10)	Número autoincrementable que empieza en 1 y no puede tomar valores negativos. Sirve para identificar un idUsuario amigo
usuarios_idUsuario	Integer (10) FK	Número que no puede tomar valores negativos y que es clave foránea proveniente de la tabla usuarios.

La tabla amigos contiene un campo usuarios_idUsuarios que es clave foránea de la tabla usuarios, que se verá más adelante, y que es clave principal de la tabla junto con el campo idAmigo. El campo idAmigo es el que almacena el idusuario del amigo. También se crea un índice con el campo usuarios_idUsuario para una búsqueda más rápida. Los dos campos deben ser clave principal para poder almacenar varios amigos para un mismo usuario.

```
-- Estructura de tabla para la tabla `amigos`
--

CREATE TABLE `amigos` (
  `usuarios_idUsuario` int(10) NOT NULL,
  `idAmigo` int(10) NOT NULL,
  PRIMARY KEY  (`usuarios_idUsuario`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

asignaturas

Campo	Tipo	Significado
IdAsignatura	Integer (10)	Número autoincrementable que identifica a cada asignatura y no puede tomar valores negativos
carreras_idCarrera	Integer (10) FK	Número que no puede tomar valores negativos y que es clave foránea proveniente de la tabla carreras
cursos_idCurso	Integer(10) FK	Número que no puede tomar valores negativos y que es clave foránea proveniente de la tabla cursos
codAsignatura	Integer (4)	Número de 4 cifras que no puede ser negativo y que identifica a cada asignatura
nombreAsignatura	Varchar (255)	Texto con el nombre de cada asignatura
tipo	Varchar (20)	Texto que proporciona si una asignatura es obligatoria, troncal, optativa…
creditos	Integer (3)	Número que no puede tomar valores negativos
periodo	Varchar (20)	Texto que identifica si la asignatura es anual o cuatrimestral

La tabla asignaturas contiene un campo idAsignatura que es la clave principal. Contiene también dos campos foráneos, carreras_idCarrera y cursos_idCurso,

que identifican completamente a cada asignatura. En esta tabla tenemos también un codAsignatura que es único (en principio) para cada asignatura, nombreAsignatura, tipo que se refiere a si es troncal, obligatoria... créditos, y periodo que se refiere a si es anual o cuatrimestral. También tenemos como índices las dos claves foráneas.

```
-- Estructura de tabla para la tabla `asignaturas`
--

CREATE TABLE `asignaturas` (
  `idAsignatura` int(10) unsigned NOT NULL auto_increment,
  `idCarrera` int(10) unsigned NOT NULL,
  `cursos_idCurso` int(10) unsigned NOT NULL,
  `codAsignatura` int(4) NOT NULL,
  `nombreAsignatura` varchar(255) collate utf8_unicode_ci NOT NULL,
  `tipo` varchar(20) collate utf8_unicode_ci NOT NULL,
  `creditos` int(3) unsigned NOT NULL,
  `periodo` varchar(20) collate utf8_unicode_ci NOT NULL,
  PRIMARY KEY  (`idAsignatura`,`cursos_idCurso`),
  KEY `Asignaturas_FKIndex1` (`cursos_idCurso`),
  KEY `idcarrera` (`idCarrera`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci AUTO_INCREMENT=1 ;
```

carreras

Campo	Tipo	Significado
idCarrera	Integer (10)	Número autoincrementable que comienza con 1 e identifica a las carreras que haya en el sistema
nombre	Varchar (255)	Texto que identifica el nombre de la carrera

La tabla carreras contiene como calve principal un campo idCarrera que la identifica de manera única. Contiene un campo nombre de la asignatura.

```
-- Estructura de tabla para la tabla `carreras`
--

CREATE TABLE `carreras` (
  `idCarrera` int(10) unsigned NOT NULL auto_increment,
  `nombre` varchar(255) collate utf8_unicode_ci NOT NULL,
  PRIMARY KEY  (`idCarrera`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci AUTO_INCREMENT=1 ;
```

cursos

Campo	Tipo	Significado
IdCurso	Integer (10)	Número que identifica a cada curso empezando por el 1 y no puede tomar valores negativos

nombreCurso	Varchar (20)	Texto que identifica el nombre del curso

La tabla cursos contiene el campo idCurso que es su clave principal y el campo nombre Curso.

```
-- Estructura de tabla para la tabla `cursos`
--

CREATE TABLE `cursos` (
  `idCurso` int(10) unsigned NOT NULL,
  `nombreCurso` varchar(20) collate utf8_unicode_ci NOT NULL,
  PRIMARY KEY  (`idCurso`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

ficheros

Campo	**Tipo**	**Significado**
idFichero	Integer (10)	Número autoincrementable que identifica a cada fichero que se comparte empezando por el 1 y no puede tomar valores negativos
usuarios_idUsuarios	Integer (10) FK	Número que no puede tomar valores negativos y que es clave foránea proveniente de la tabla usuarios
grupos_idGrupo	Integer (10) FK	Número que no puede tomar valores negativos y que es

		clave foránea proveniente de la tabla grupos
padre	Integer (10)	Número que identifica quién es el padre jerárquicamente habando, y que proviene del idFichero
ruta	Longtext	Texto que contiene la ruta completa hasta llegar al fichero, pasando por los directorios padre

La tabla ficheros contiene los ficheros que se comparten con otros usuarios mediante el disco compartido, cuya clave principal es el idFichero. Contiene como claves foráneas usuarios_idUsuario y grupos_idGrupo que se explican más adelante. El campo padre contiene el idFichero de su padre, jerárquicamente hablando. El campo ruta contiene la ruta completa del archivo o fichero.

```
-- Estructura de tabla para la tabla `ficheros`
--

CREATE TABLE `ficheros` (
  `idFichero` int(10) NOT NULL auto_increment,
  `usuarios_idUsuario` int(10) NOT NULL,
  `grupos_idGrupo` int(10) NOT NULL,
  `padre` int(10) NOT NULL,
  `ruta` longtext collate utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`idFichero`,`usuarios_idUsuario`,`grupos_idGrupo`),
  KEY `Ficheros_FKIndex1` (`usuarios_idUsuario`),---
```

```
  KEY `grupos_idGrupo` (`grupos_idGrupo`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci AUTO_INCREMENT=1 ;
```

grupos

Campo	Tipo	Significado
idGrupo	Integer (10)	Número autoincrementable que identifica a cada grupo cuando se comparte. Empieza en el 1 y no puede tomar valores negativos
perfiles_idPerfil	Integer (10)	Número que no puede tomar valores negativos y que es clave foránea proveniente de la tabla perfiles
nombreGrupo	Varchar (20)	Texto que contiene el nombre del grupo que se crea al compartir

La tabla grupos tiene como clave principal el idGrupo, el cual identifica los grupos que se crean cuando se comparten archivos, y como clave foránea perfil_idPerfil que se explica más adelante. Se crea un índice con el campo perfiles_idPerfil para acelerar las búsquedas.

```
-- Estructura de tabla para la tabla `grupos`
--

CREATE TABLE `grupos` (
  `idGrupo` int(10) NOT NULL auto_increment,
  `perfiles_idPerfil` int(10) NOT NULL,
```

```
  `nombreGrupo` varchar(20) collate utf8_unicode_ci NOT NULL,
  PRIMARY KEY  (`idGrupo`,`perfiles_idPerfil`),
  KEY `perfiles_idPerfil` (`perfiles_idPerfil`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci AUTO_INCREMENT=1 ;
```

perfiles

Campo	**Tipo**	**Significado**
IdPerfil	Integer (10)	Número que identifica a cada perfil, que es quien contiene los permisos. No puede tomar valores negativos
renombrar	Boolean (1)	Booleano que toma el valor 0 y 1 para denotar que si puede renombrar o no
eliminar	Boolean (1)	Booleano que toma el valor 0 y 1 para denotar que si puede eliminar o no

La tabla perfiles contiene los permisos que se pueden aplicar a los ficheros. Como clave principal está el campo idPerfil y también tenemos dos campos más, renombrar y eliminar, que pueden tomar el valor 0 si no tiene permiso, o 1 si tiene permiso.

```
-- Estructura de tabla para la tabla `perfiles`
--

CREATE TABLE `perfiles` (
```

```
  `idPerfil` int(10) NOT NULL,
  `eliminar` tinyint(1) NOT NULL,
  `renombrar` tinyint(1) NOT NULL,
  PRIMARY KEY  (`idPerfil`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

profesores

Campo	Tipo	Significado
idProfesor	Integer (10)	Número autoincrementable que empieza en 1 y no puede tomar valores negativos
usuarios_idusuario	Integer (10) FK	Número que no puede tomar valores negativos y que es clave foránea proveniente de la tabla usuarios
despacho	Integer (10)	Número que identifica el despacho en el que está el profesor

La tabla profesores contiene un campo usuarios_idProfesor que es una clave foránea de la tabla usuarios, que se verá más adelante, y que es clave principal de la tabla. El campo idProfesor es el que identifica que un usuario es un profesor. También se crea un índice con el campo usuarios_idUsuario para una búsqueda más rápida.

```
CREATE TABLE `profesores` (
  `idProfesor` int(10) NOT NULL auto_increment,
  `usuarios_idUsuario` int(10) NOT NULL,
```

```
  `despacho` int(3) default NULL,
  PRIMARY KEY  (`idProfesor`,`usuarios_idUsuario`),
  KEY `Profesores_FKIndex1` (`usuarios_idUsuario`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci AUTO_INCREMENT=1 ;
```

usuarios

Campo	**Tipo**	**Significado**
idUsuario	Integer (10)	Número autoincrementable que empieza en 1 y no puede tomar valores negativos
nombre	Varchar (20)	Texto con el nombre de cada usuario
apellido1	Varchar (20)	Texto con el primer apellido de cada usuario
apellido2	Varchar (20)	Texto con el segundo apellido de cada usuario
correo	Varchar (20)	Texto que contiene el correo electrónico de cada usuario
login	Varchar (20)	Texto que contiene el nombre para la autentificación de cada usuario
pass	Varchar (255)	Texto que contiene la contraseña de cada usuario

La tabla usuarios contiene como clave principal el campo idUsuario, que es el que identifica a todos los usuarios de manera unívoca. Los campos nombre, apellido1, apellido2, correo, login y pass, corresponden a los datos personales de

los usuarios.

```
-- Estructura de tabla para la tabla `usuarios`
--

CREATE TABLE `usuarios` (
  `idUsuario` int(10) NOT NULL auto_increment,
  `nombre` varchar(20) collate utf8_unicode_ci NOT NULL,
  `apellido1` varchar(20) collate utf8_unicode_ci NOT NULL,
  `apellido2` varchar(20) collate utf8_unicode_ci NOT NULL,
  `correo` varchar(40) collate utf8_unicode_ci NOT NULL,
  `login` varchar(20) collate utf8_unicode_ci NOT NULL,
  `pass` varchar(255) collate utf8_unicode_ci NOT NULL,
  PRIMARY KEY  (`idUsuario`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci AUTO_INCREMENT=1 ;
```

usuarios_has_asignaturas

Campo	**Tipo**	**Significado**
usuarios_idUsuario	Integer (10) FK	Número que no puede tomar valores negativos y que es clave foránea proveniente de la tabla usuarios
asignaturas_idAsignatura	Integer (10) FK	Número que no puede tomar valores negativos y que es clave foránea proveniente de la tabla asignaturas

La tabla usuarios_has_asignaturas se crea al tener una relación N:N entre las tablas usuarios y asignaturas, con lo que se tiene que generar una tabla intermedia que contenga como campos las claves principales de ambas tablas. La clave principal se compone de usuarios_idUsuario y asignaturas_idAsignatura que son claves foráneas. Los índices se crean con ambos campos.

```
-- Estructura de tabla para la tabla `usuarios_has_asignaturas`
--

CREATE TABLE `usuarios_has_asignaturas` (
  `usuarios_idUsuario` int(10) NOT NULL,
  `asignaturas_idAsignatura` int(10) unsigned NOT NULL,
  PRIMARY KEY (`usuarios_idUsuario`,`asignaturas_idAsignatura`),
  KEY `Usuarios_has_Asignaturas_FKIndex1` (`usuarios_idUsuario`),
  KEY `Usuarios_has_Asignaturas_FKIndex2` (`asignaturas_idAsignatura`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

usuarios_has_grupos

Campo	**Tipo**	**Significado**
usuarios_idusuario	Integer (10) FK	Número que no puede tomar valores negativos y que es clave foránea proveniente de la tabla usuarios
grupos_idGrupo	Integer (10) FK	Número que no puede tomar valores negativos y que es

		clave foránea proveniente de la tabla grupos

La tabla usuarios_has_grupos se crea por la relación N:N entre las tablas usuarios y grupos, de esta manera se crean relaciones 1:N con esta tabla. La clave principal se compone de los campos usuarios_idUsuario y grupos_idGrupo que son claves foráneas y claves principales en sus respectivas tablas. El índice que se crea es con ambos campos de la tabla.

```
-- Estructura de tabla para la tabla `usuarios_has_grupos`
--

CREATE TABLE `usuarios_has_grupos` (
  `usuarios_idUsuario` int(10) NOT NULL,
  `grupos_idGrupo` int(10) NOT NULL,
  PRIMARY KEY  (`usuarios_idUsuario`,`grupos_idGrupo`),
  KEY `Usuarios_has_Grupos_FKIndex1` (`usuarios_idUsuario`),
  KEY `Usuarios_has_Grupos_FKIndex2` (`grupos_idGrupo`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

4.5. Diseño lógico

Concepto

Se procede a explicar toda la ingeniería software desarrollada; patrones de diseño, librerías, modelos etc.

Comentar que se describirá a nivel lógico el sistema, pues está a disposición un manual del programador perfectamente detallado.

Lógica

Debido a ser un proyecto que partiera desde cero y como reto personal decidimos adaptar nuestro proyecto totalmente a las nuevas versiones de las tecnologías olvidando por completo la compatibilidad hacia atrás para evitar que la posible expansión de la aplicación volviera al antiguo cauce.

Olvidamos totalmente la programación estructurada excepto en los límites que el ámbito en el que se desarrollaba la aplicación no permitía saltárnoslos (mínimos e insignificantes).

La nueva lógica de diseño orientada a objetos, que en la versión de *PHP* 5.0 estaba totalmente madura, dirigió nuestra motivación y esfuerzos durante todo el desarrollo pudiendo obtener una gran mayoría de sus ventajas de polimorfismo, herencia, encapsulamiento…

Apostamos totalmente no solo por las nuevas versiones, sino por tecnologías recién liberadas pero totalmente estables así como modelos de software que permitieran un rápido desarrollo y que procederemos a detallar a continuación.

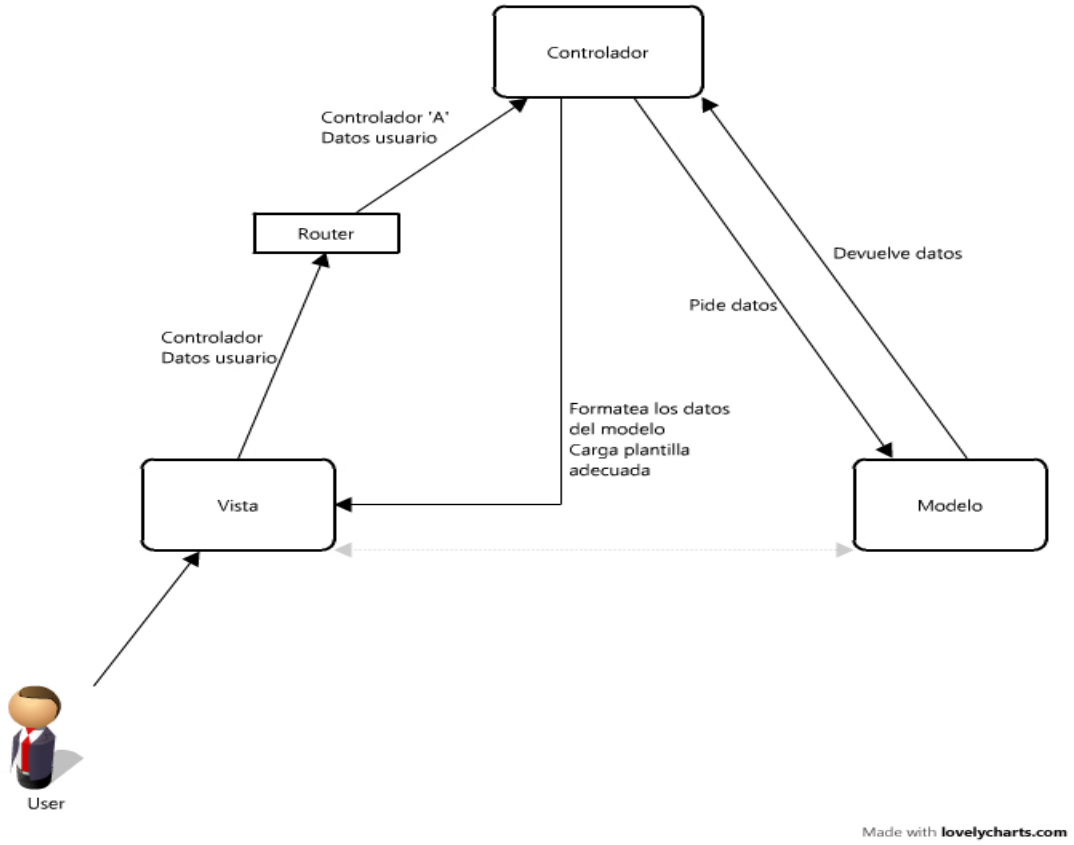
MVC

Patrón software consistente en la programación en 3 capas:

- Interfaz de usuario (Vista)

- Lógica de control (Controlador)

- Acceso a datos (Modelo)



La **vista** es la parte que ve el usuario final, que en nuestro caso será la interfaz web. Es a través de ella desde donde el usuario introduce los datos y donde comienza el dinamismo de la aplicación.

Puesto que en una aplicación web como la nuestra la vista consiste únicamente en xHtml y CSS, es fácil intuir que el concepto de orientación a objetos se escapa en este punto por lo que se busco una solución siempre teniendo como máxima la separación de lógica y diseño.

Es cuando dimos con las plantillas, que serían cargadas por los controladores. Pero el concepto primario de plantillas con nuestros conocimientos requeriría en última instancia de código embebido (obligatorio al fin y al cabo), por lo que tras ardua documentación dimos con la solución: **smarty**.

Smarty no es más que una capa programada en PHP la cual transforma plantillas en, que a parte de poder incluir código xhtml (la vista) se podría añadir un formato de código característico de esta capa de manera que se pudieran acceder a variables, bucles, ifs y cientos de opciones más.

Este código es muy simple y ni ensuciaba en extremo el código html ni producía la sensación de embeber nada.

Pero para asignar el valor a las variables que se cargaran a las plantillas así como las propias plantillas necesitaríamos del uso de los controladores.

Antes de proceder a detallar los controladores debemos indicar que mientras se investigaba sobre el patrón de diseño encontramos diferentes *features* para el mismo que añadían más funcionalidad; una de ellas era el **Router**.

Esta pequeña capa se encargaría de cargar el controlador correspondiente de manera que todo se centralizaba en un solo archivo (index.php) lo que simplificaba no solo la estructura de directorios sino la recogida de excepciones que se tratarán mas adelante.

Los **controladores** son las clases encargadas de la lógica de control. Son ellos los que definen que información se debe mostrar así como que acceso se da a otros controladores en la interfaz. Si la vista es la capa de conexión con el usuario y la aplicación, los controladores son la capa de control entre esta vista y el modelo.

Como ya se ha indicado, son los controladores los encargados de gestionar las plantillas y sus variables tipo, por lo que todo controlador específico de aplicación heredará de uno base encargado de la gestión de la clase **smarty** entre otros detalles como semáforos de control para la clase Router.

Ya en base a nuestra aplicación, sabíamos que los alumnos y profesores iban a tener funcionalidad en común y otra independiente, por lo que un nivel más de herencia era evidente. El controlador de usuario (*controller_user)* tendrá dos hijos: controlador de alumno y de profesor que se encargarán de las tareas específicas

de estos dos actores.

Puesto que los errores lanzados por el modelo vendrán con excepciones es en estos controladores donde las captaremos y mostraremos en función de la excepción.

Es en el **modelo** donde se implementa la mayoría de la lógica a nivel de aplicación. Es en esta capa donde se usan los diferentes patrones usados:

- Iterator (SPL library)
- Singleton
- ActiveRecord

Así como otras tecnologías:

- Directory Iterator (SPL library)
- PDO (abstraction libreray)

SPL

Quizás descubrimos esta librería demasiado tarde pues su funcionalidad y potencial es enorme.

Esta librería esta implantada en la base de PHP5, de manera que aunque consista en una API programada en PHP las capas mas superiores están implantadas en el núcleo de PHP (ZEND) y por lo tanto programadas sobre el lenguaje C.

Una de estas interfaces superiores es la llamada “Iterator”, cuyo concepto radica en la posibilidad de implementar dicha *interface* en cualquier clase de manera que dicha clase pueda ser recorrida como si de un array se tratara con el bucle *foreach* (este tipo de bucle es característico de PHP y no requiere contadores).

Si bien esta interfaz dispone de métodos los cuales pueden ser usados directamente o sobrescritos para modificar su funcionalidad en base a los deseos del programador, nuestra aplicación usa su funcionalidad básica que es más que suficiente.

Otro tipo de *interfaces* daban la posibilidad de acceder a las propiedades de un objeto como si de un array en el nivel de sintaxis se tratara, pudiendo simplificar el MVC en gran medida. Decidimos no implementar esta opción en nuestro *mini-framework* por dos razones:

1. Como ya indiqué el descubrimiento fue demasiado tarde y tendríamos que modificar una base ya sólida y funcional

2. Aunque el código quedara más sencillo complicaba los conceptos en nuestro nivel de programación.

Este patrón no solo es usado en la clase **Directory Iterator** (se encarga de obtener los archivos de un directorio), la cual fue el catalizador del descubrimiento de esta librería, sino que ya estaba implementado en otras librerías como **PDO.**

PDO

Una librería de abstracción de base de datos que nos solucionaba grandes problemas definidos en la fase de diseño: Inyección SQL, uso de otras bases de datos, implementación del patrón *active record*, tratar errores de conexión, tratar datos recogidos por consultas de selección, entre otras.

- Inyección SQL: Dos conceptos como los *prepared statments* y los *placeholders* evitaban tener que chequear los datos de entrada del usuario para evitar este error común en la programación. Resumiremos estos dos conceptos diciendo que con ellos era completamente imposible insertar código SQL en las variables usadas en la consulta SQL original pues dichas variables se sustituyen con *placeholders*, se “prepara” la sentencia SQL y después se ejecuta generando los errores correspondientes.

- Uso de otras bases de datos: Aquí esta el concepto de *abstraction library,* puesto que PDO dispone de drivers para diferentes bases de datos (sqlitle, orable, postgremysql, mysql…) de manera que instalando dichos drivers a nivel de servidor e indicando cual usar a nivel de programación el código de uso de la base de datos se simplifica usando siempre los mismos métodos.

- **Patrón Active Record**: Consiste en mapear la base de datos en el modelo de la aplicación de manera que tengamos por cada tabla de la base de datos una clase de tipo *model* que herederá de un modelo base. Es en este modelo base donde se definen dos conceptos:

    - Inicio de conexión con base de datos usando el patrón *Singleton* y la librería PDO.

    - Declaración de los métodos del patrón *active record* para la modificación de los objetos del modelo paralelamente a la base de datos.

  En esencia consiste en que todo modelo podrá ser insertado, eliminado o modificado tanto a nivel de lógica de programación como en base de datos centralizando aún más en el paradigma de la orientación a objetos: *"Programa una vez"*.

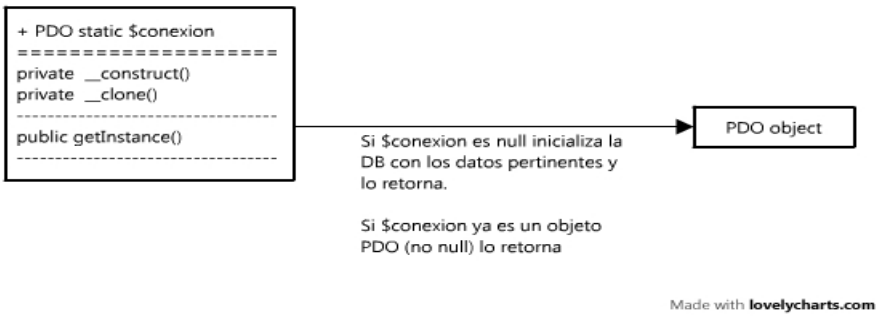
  Por ejemplo: *Model_user* cuando se instancia, si se le pasa por argumento el ID de usuario automáticamente el objeto se rellenará con los datos de la tupla de la base de datos correspondiente así como la llamada a métodos heredados de Model_base (active record): _dbInsert(), _dbDelete(), _dbUpdate(array $valores) efectuarán los cambios necesarios en los dos niveles ya descritos.

- Tratar errores de conexión: PDO implementa un tipo de excepción específico (PDOException) que podrá ser capturado fácilmente pudiendo separar errores de base de datos, de la lógica de la aplicación

- Tratar datos recogidos por consultas de selección: Como ya hemos descrito la librería PDO implementa la interfaz *iterator* de manera que la recogida de datos con consultas tipo SELECT se realiza de manera limpia y sencilla pudiendo tratar a lo que se llama en PDO *preparedstatment,* que consiste en un tipo de objeto con los datos de la consulta y sus métodos, pudiendo formatear el resultado de múltiples maneras y recorrerlo fácilmente con un *foreach*.

Aunque se ha comentado no se ha detallado el **patrón singleton**, implementado en el Modelo base del que heredan todos los demás y donde se inicia la conexión. Es aquí donde actúa este patrón de diseño.

El concepto radica en tener solo una instanciación de una clase usando el encapsulamiento y propiedades *static*. Justo esta funcionalidad la queremos en la instanciación del objeto PDO resultante al iniciar la conexión, de manera que obliguemos a todos los modelos a usar el mismo objeto PDO en todas sus fases.



El diseño MVC junto con el paradigma de la orientación a objetos es posiblemente el más usado y el mejor en cuanto a aplicaciones web se refiere.

Decidimos implementar este diseño desde cero implementando nosotros un *mini-framework* de manera que no solo pudiéramos aprender sino que no tuviéramos que formarnos en un *framework* sin entender el concepto.

4.4. Diagrama de clases

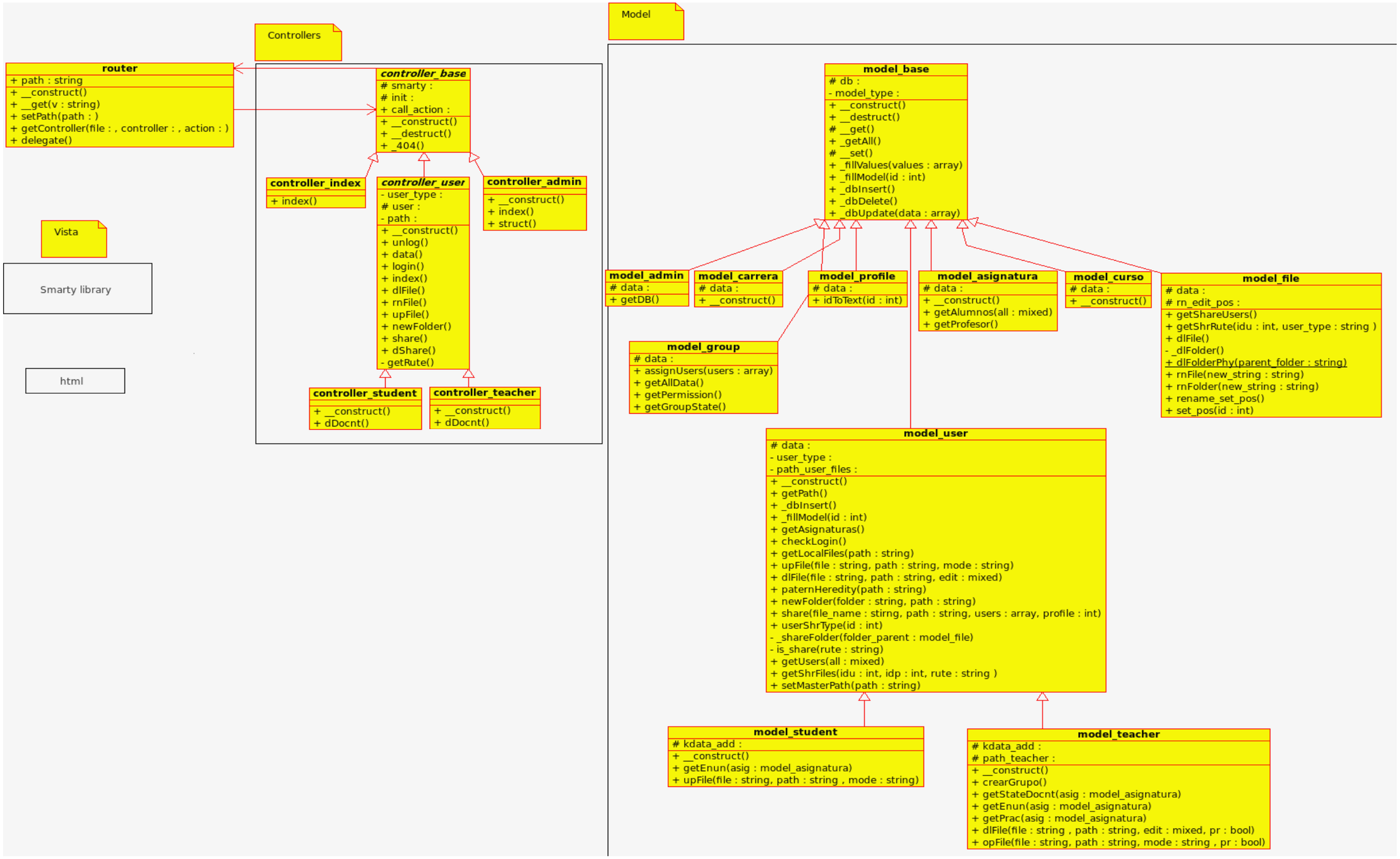


Figura 4. Diagrama de clases

4.5. Diagrama de casos de uso

Diagrama de casos de uso del administrador

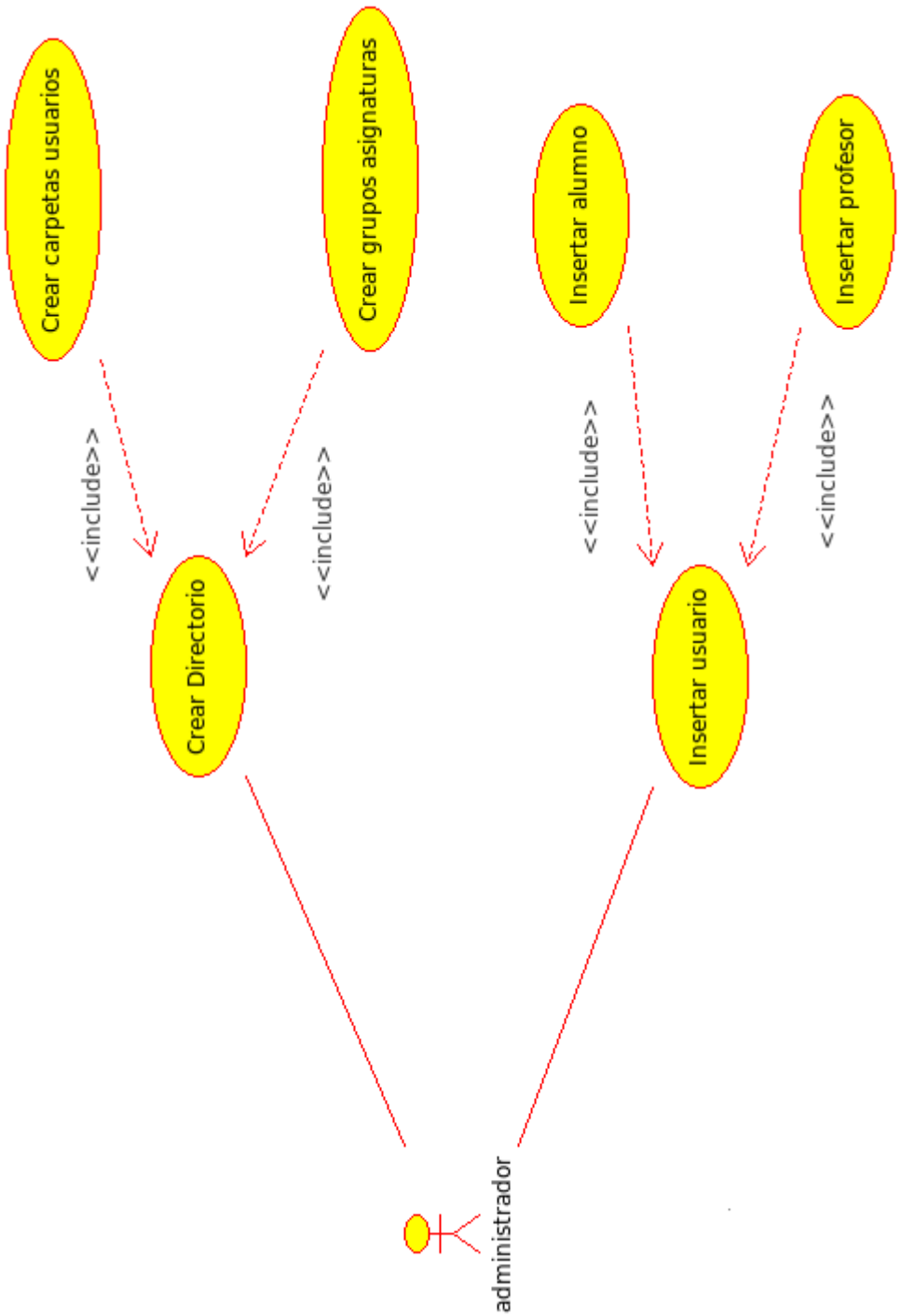


Figura 5. Casos de uso del administrador

Diagrama de casos de uso de los usuarios

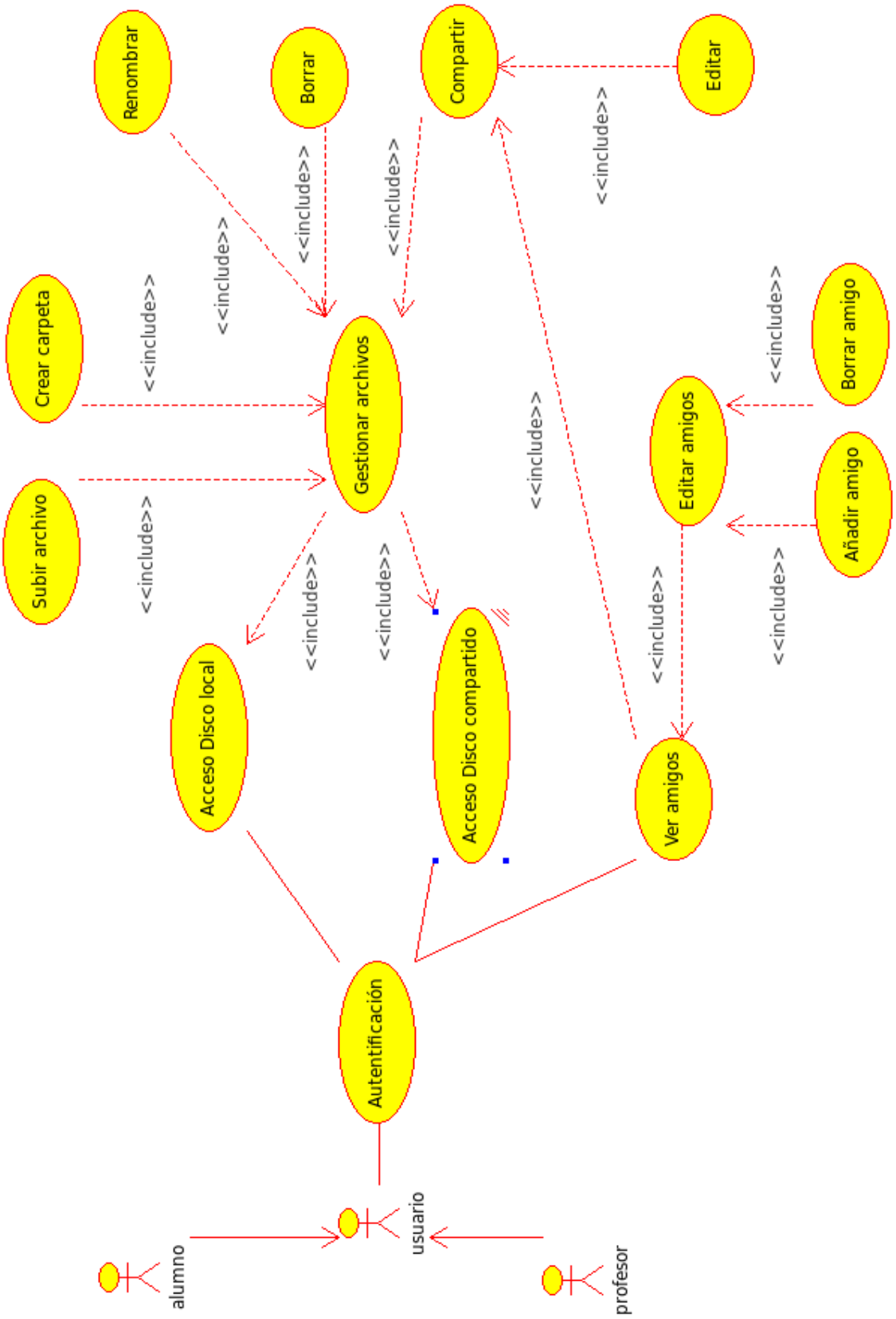


Figura 6. Casos de uso de los usuarios

Diagrama de casos de uso particular del profesor

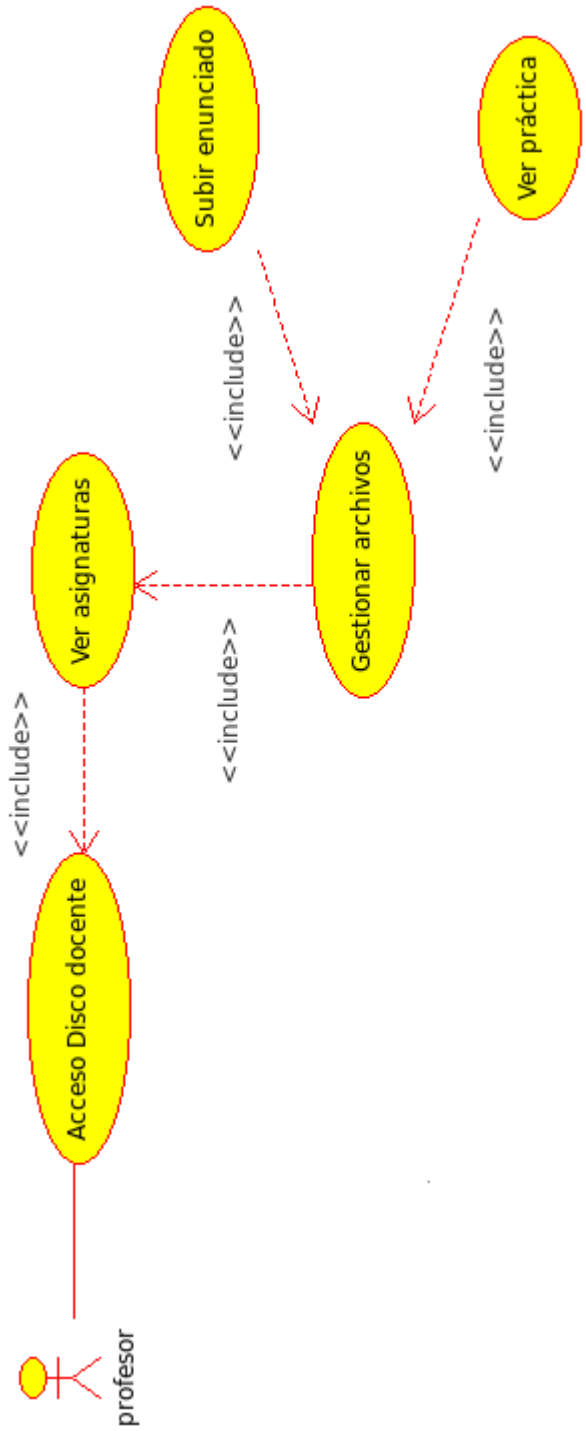


Figura 7. Casos de uso del profesor

Diagrama de casos de uso particular del alumno

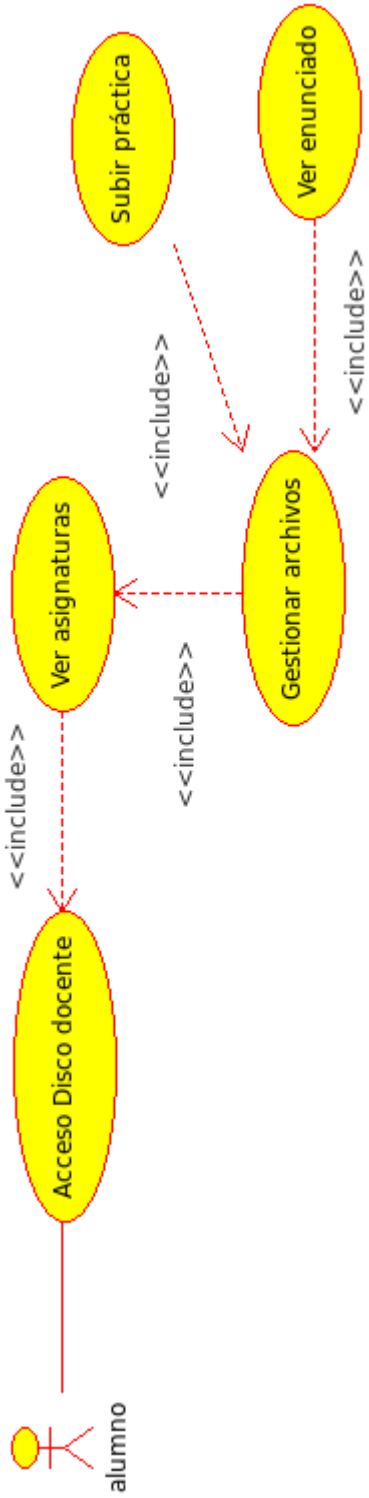


Figura 8. Casos de uso del alumno

5. Pruebas

Concepto

Estas pruebas consisten en que para todo proceso que requiera datos de entrada, que en nuestra aplicación serán los datos de entrada del usuario, forzar al mismo a todos los tipos de datos de entrada imaginables.

Realización

Hemos de detallar que se seguirá un orden lógico en la utilización de la aplicación.

Identificación de usuario

Posiblemente el formulario donde más seguridad está implementada debido a su funcionalidad, puesto que una vez identificado serás un usuario válido y dispondrás de sesión.

Únicamente si se insertan los datos correctos se procederá al login. La inyección SQL esta protegida tanto por la librería PDO y sus *placeholders* como por procedimientos almacenados. La contraseña irá encriptada a nivel de servidor.

Realizamos 2 tipos de pruebas:

1. Inyección SQL

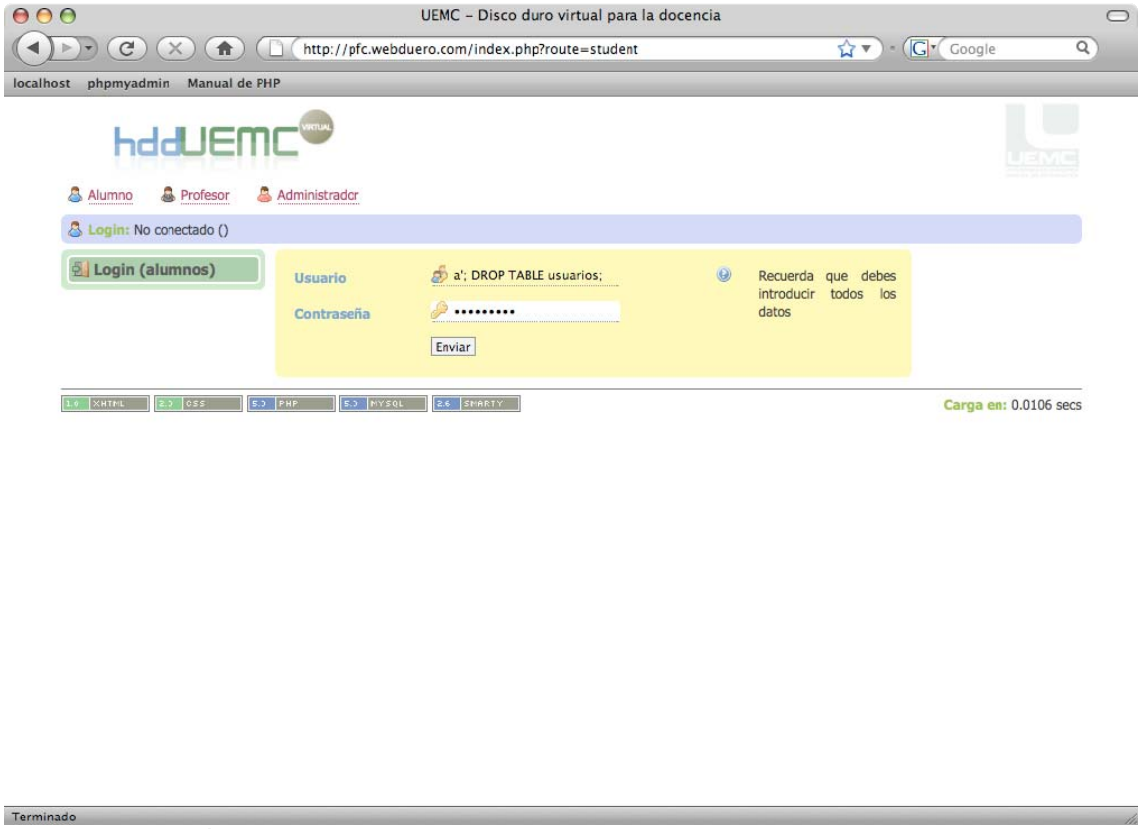


Figura 9. Inyección SQL

2. Caracteres comprometidos tales como ‘ “ / \ ·$@*^Ç¨ etc

Figura 10. Caracteres comprometidos

En ambos casos el resultado es:

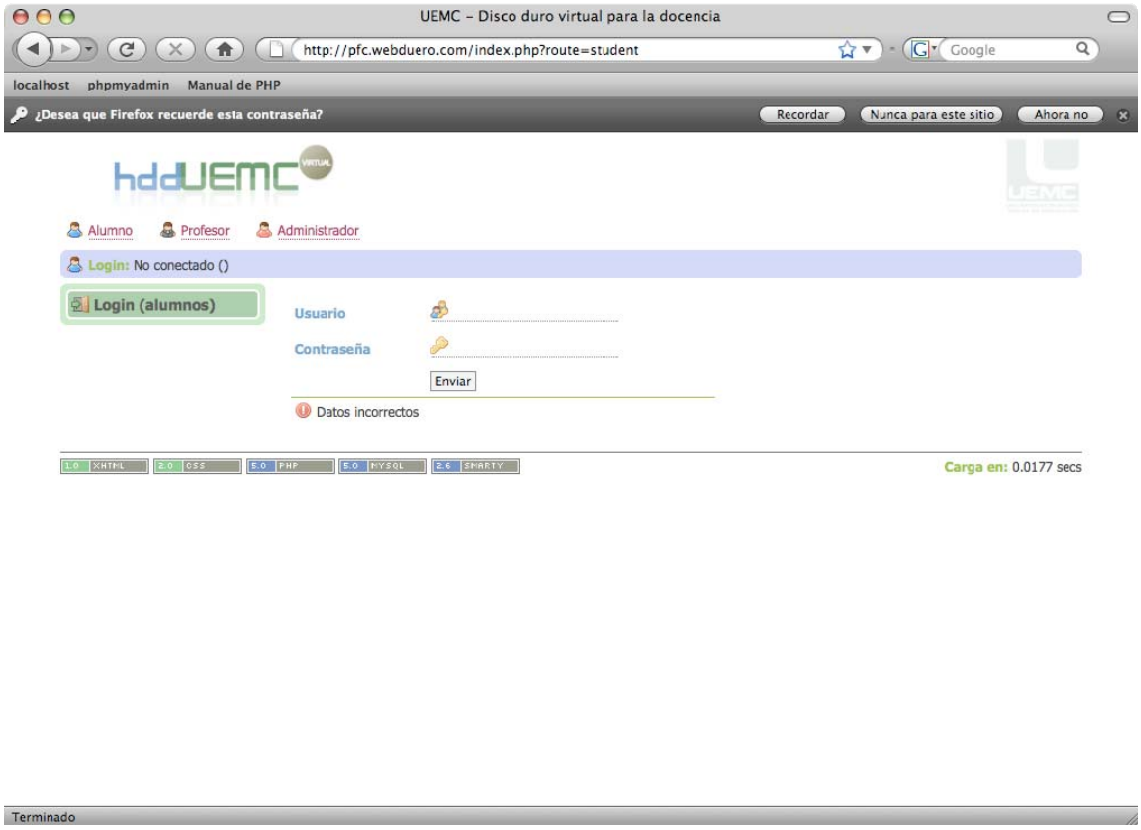


Figura 11. Datos incorrectos

Detallar también que aunque la pantalla de identificación sea idéntica, un alumno no podrá identificarse en la sección del profesor y viceversa, obteniendo el mismo error.

Una vez dentro del disco local del usuario las opciones donde se requieren datos de entrada del usuario serán: crear carpeta, subir archivo, renombrar archivo, eliminar archivo, compartir y editar (y la gestión de amigos).

Crear carpeta

Para crear carpeta se necesita introducir el nombre de ésta y por ello se deberán tratar los diferentes errores tales como:

Nombre vacío

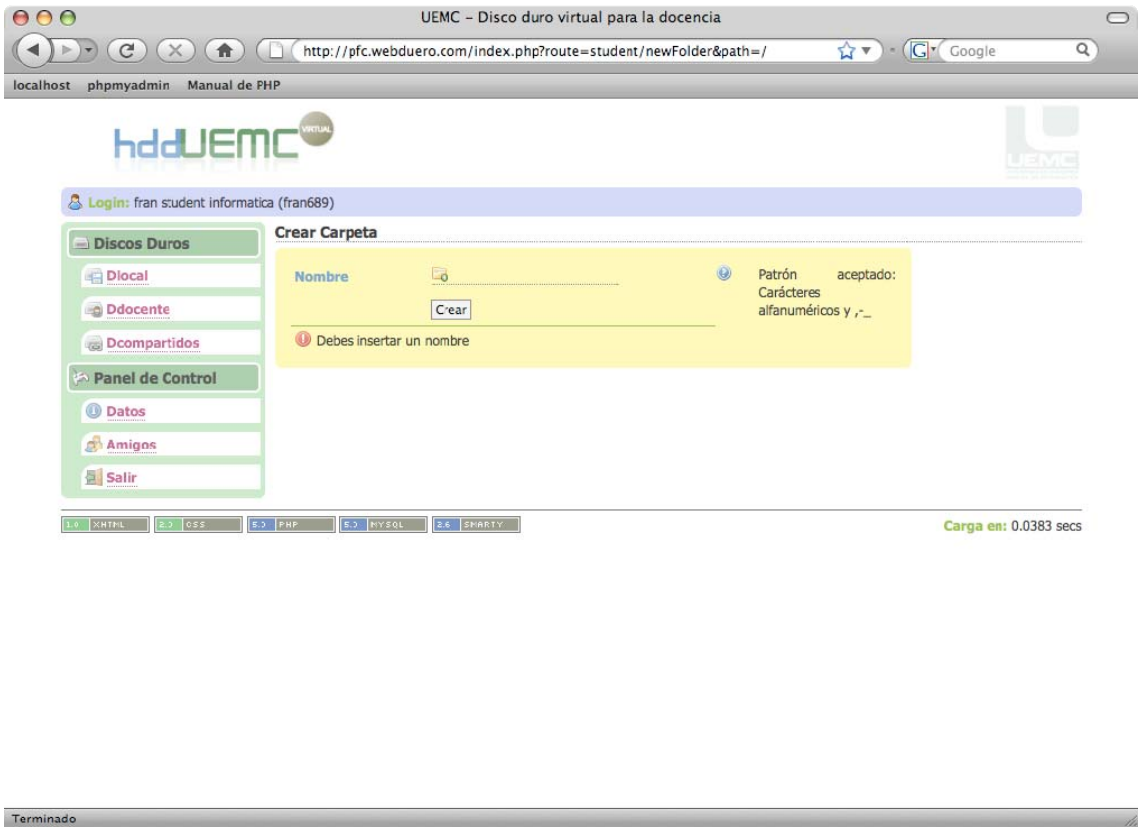


Figura 12. Nombre vacío

Carpeta ya existente

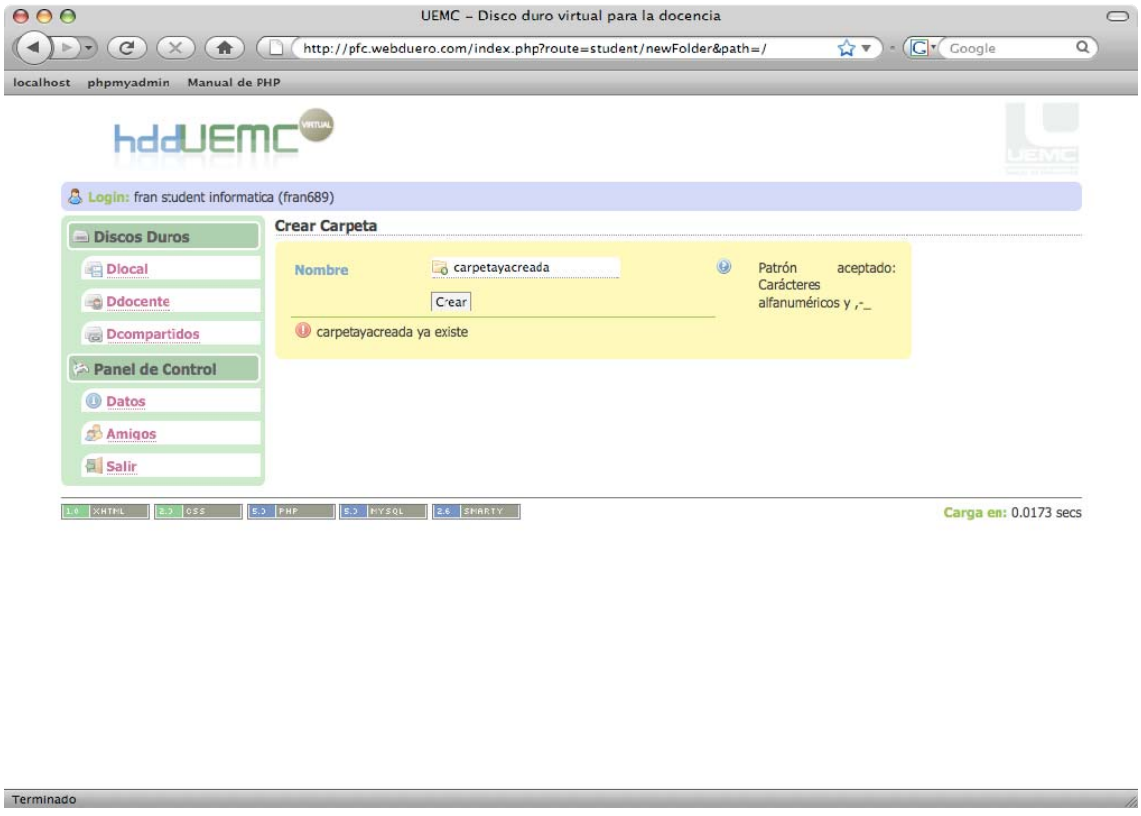


Figura 13. Carpeta existente

o en caso de no haber asignado los permisos adecuados a nivel de servidor. (Debería salir un error de PHP si la directiva *'display_errors'* estuviera activada. Puesto que suponemos estar en un entorno de producción esta directiva se encuentra desactivada).

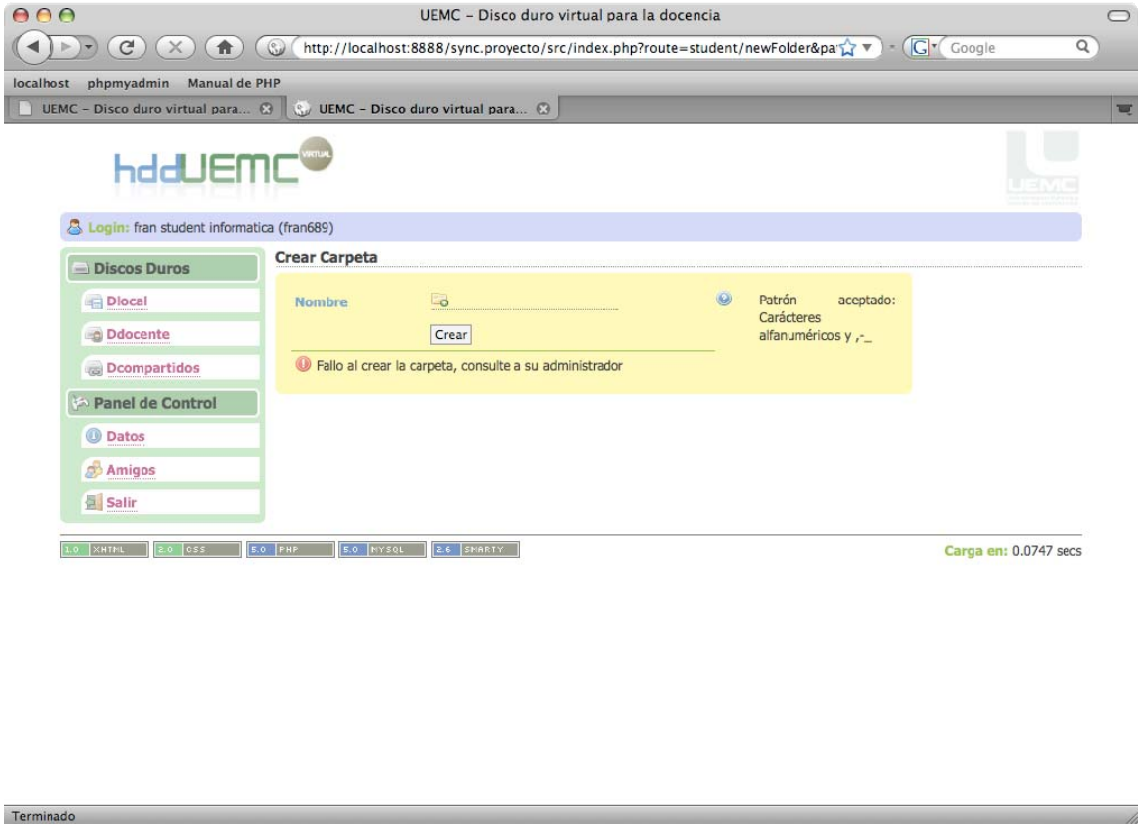


Figura 14. Error interno

Puesto que se avisa de los caracteres admitidos para la creación de la carpeta, comentar que si se incluyen caracteres prohibidos estos se omitirán pero la carpeta será creada.

Suponiendo que se ha introducido una carpeta con el nombre "Carpeta 'a for@#matear'""", la carpeta creada será:

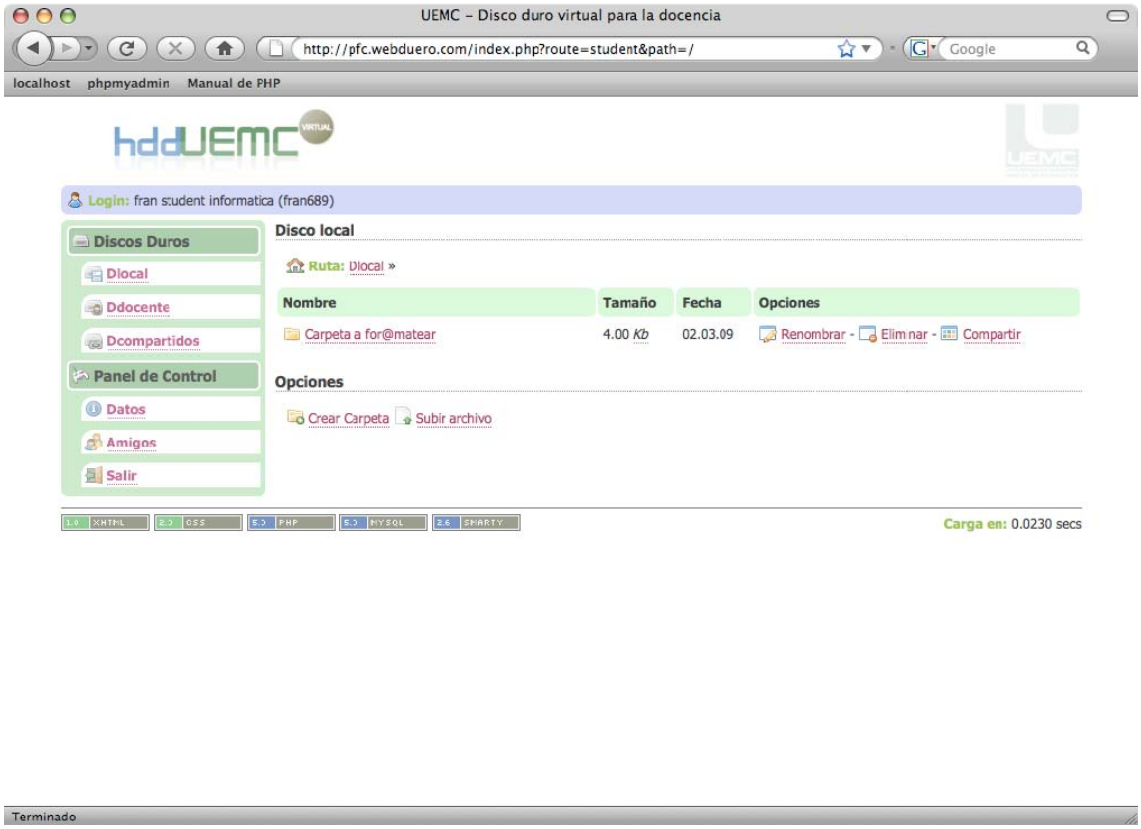


Figura 15. Carpeta formateada

El límite de la cadena a introducir esta limitado a 32 caracteres con xhtml.

Subir archivo

En caso de que no se elija ningún archivo.

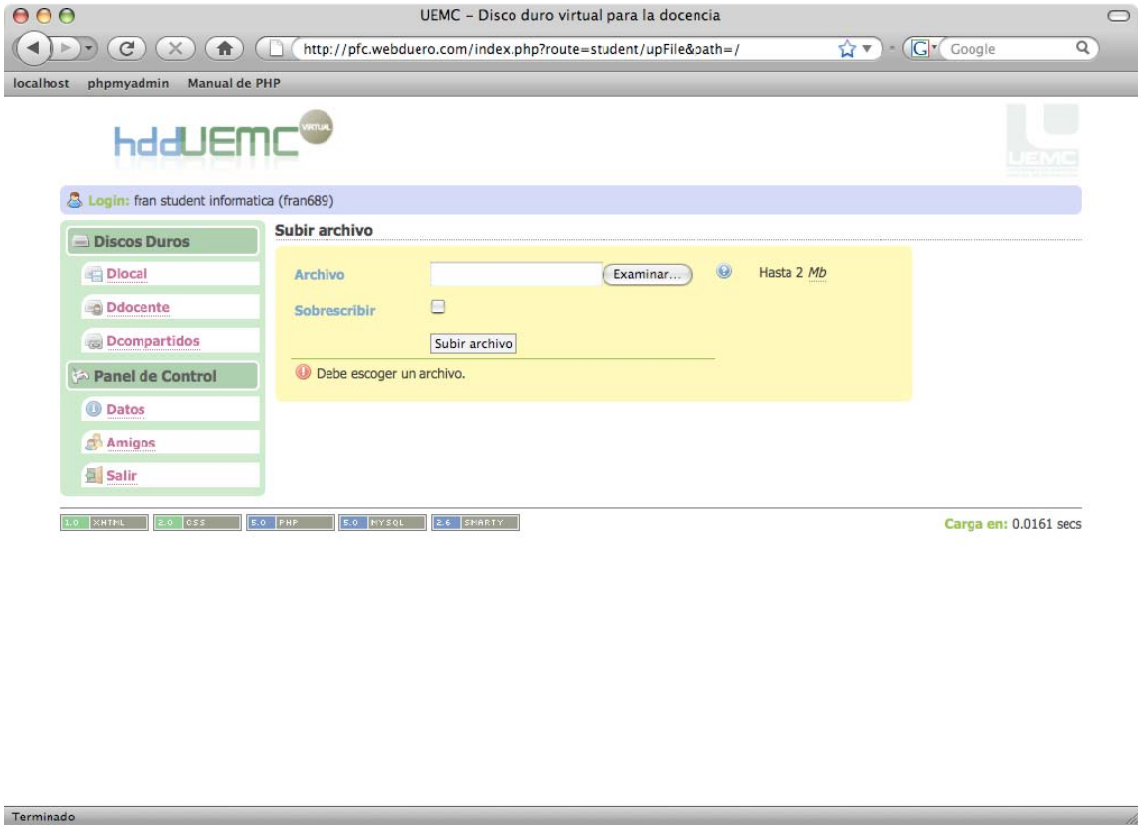


Figura 16. Archivo vacío

Si el archivo ya existe y no se ha marcado la opción sobrescribir

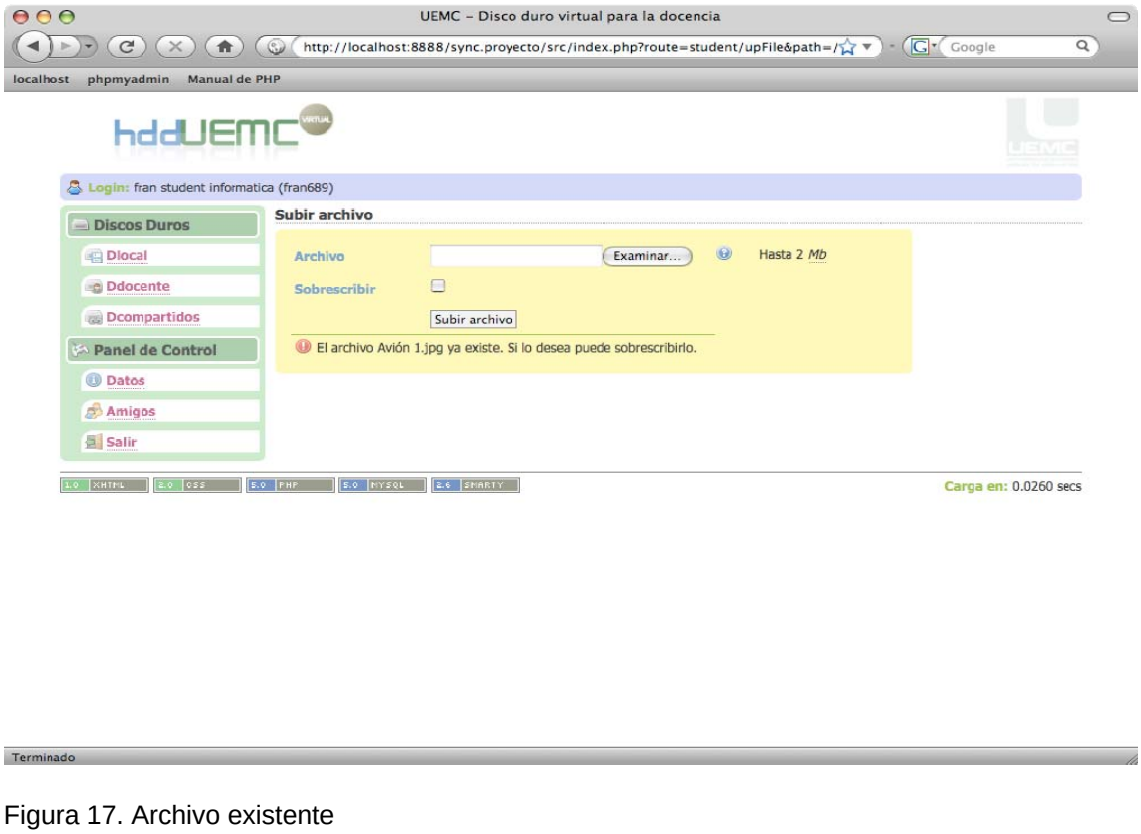


Figura 17. Archivo existente

Partiendo de la base de que la comprobación del tamaño de archivo únicamente con PHP es muy irregular no por fallas de programación sino por las limitaciones de PHP (debido a que sólo podrá comprobar su tamaño una vez en el servidor, pues las cabeceras http tampoco son fiables) se estima el error si el archivo supera el limite en una ligera diferencia.

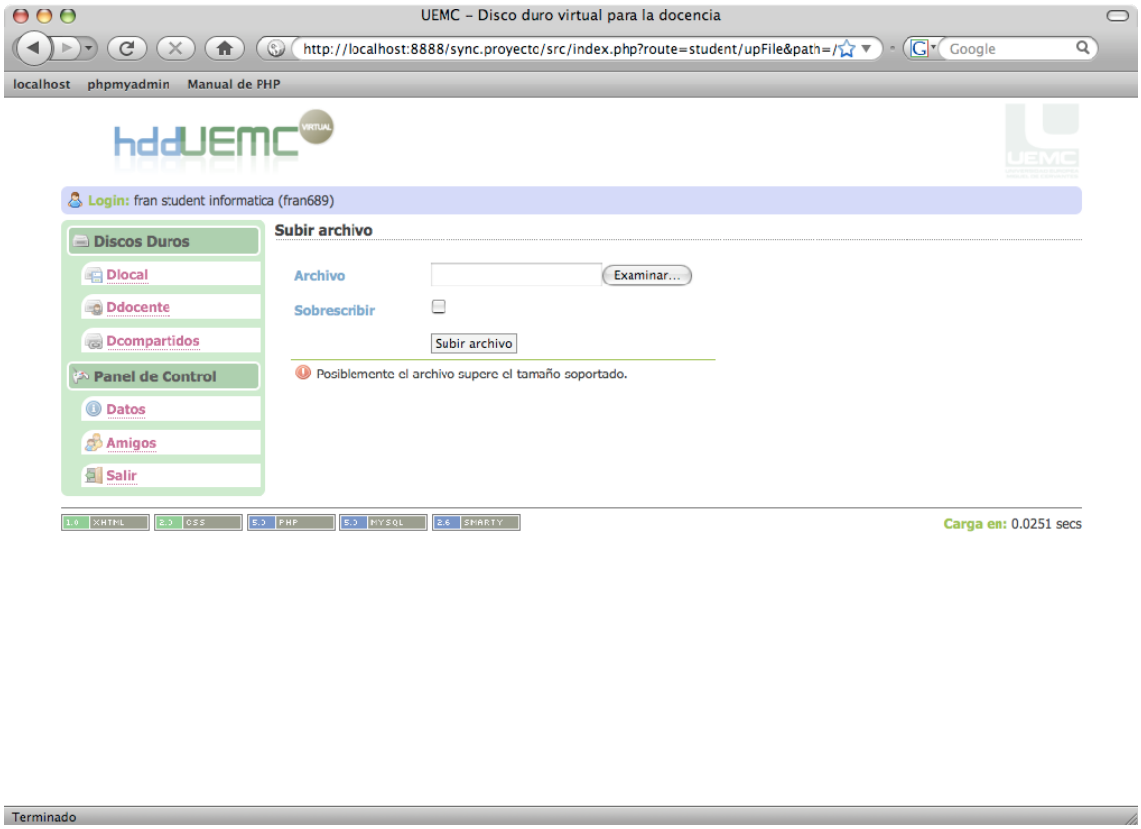


Figura 18. Tamaño excesivo

Si ocurre un error de PHP (como ya se ha descrito antes, posiblemente debido a permisos a nivel de servidor), este también se captura.

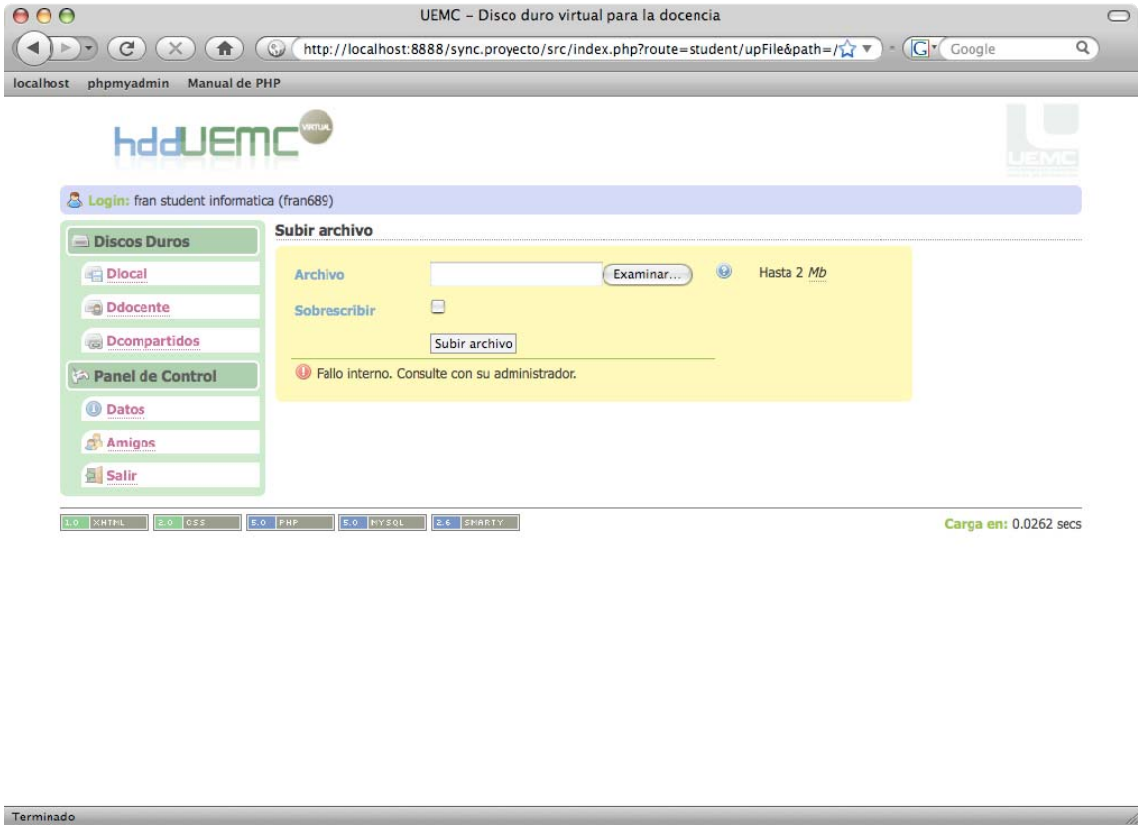


Figura 19. Fallo interno

Renombrar

En caso de no insertar un nuevo nombre

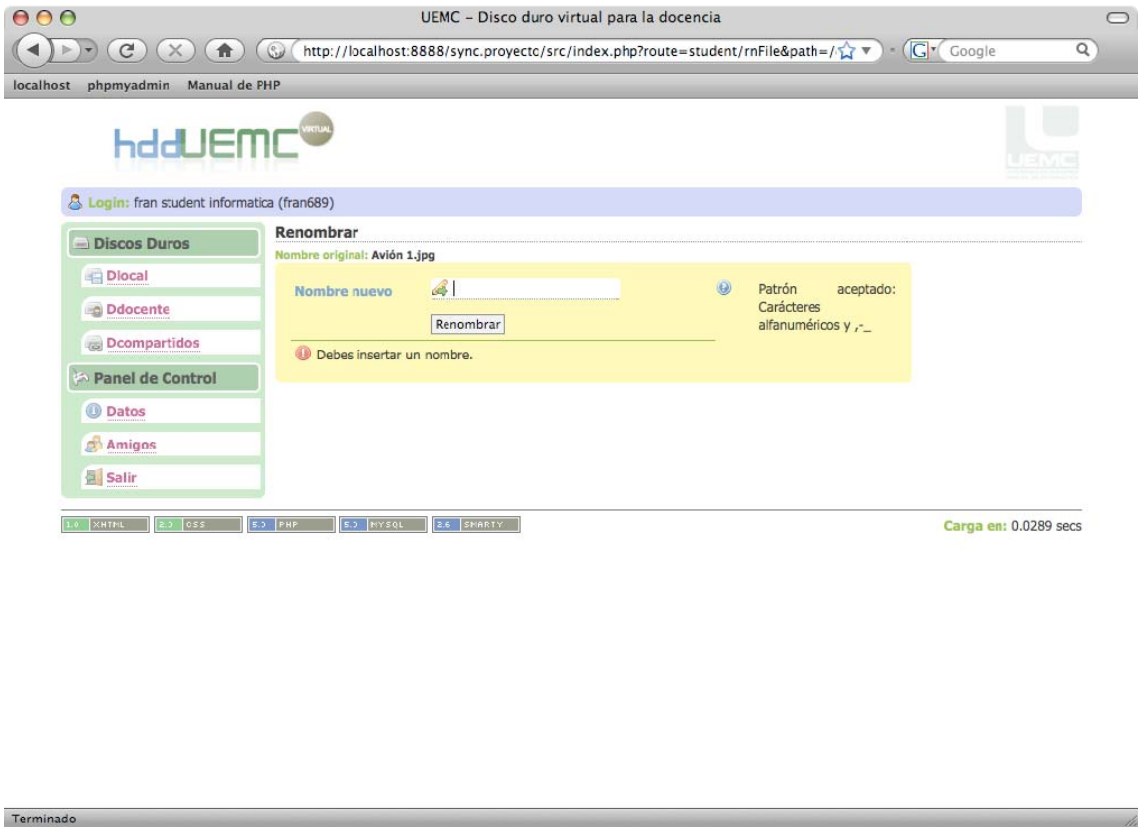


Figura 20. Nombre vacío

Si editas el archivo o carpeta y ya existe otro con el mismo nombre

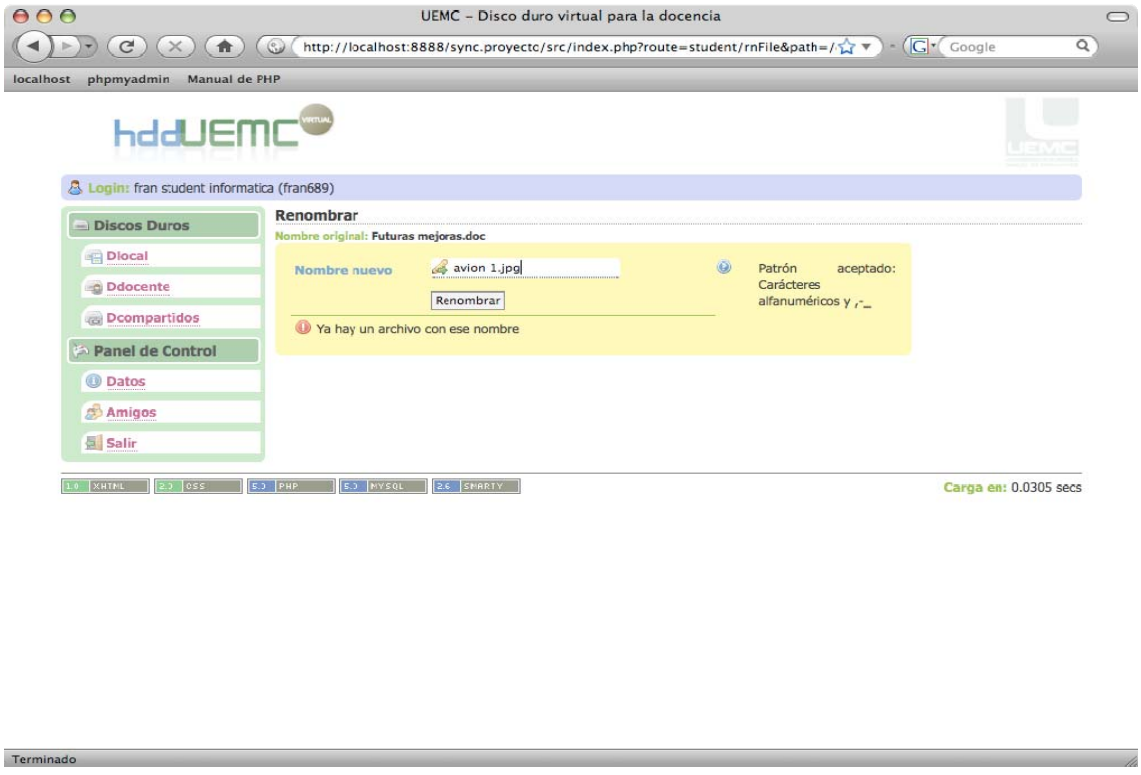


Figura 21. Ya existe otro archivo con el mismo nombre

Eliminar

Puesto que si elegimos esta opción eliminará directamente hay que proteger el controlador destinado a ello (URL) por lo que si se edita obtendremos un error.

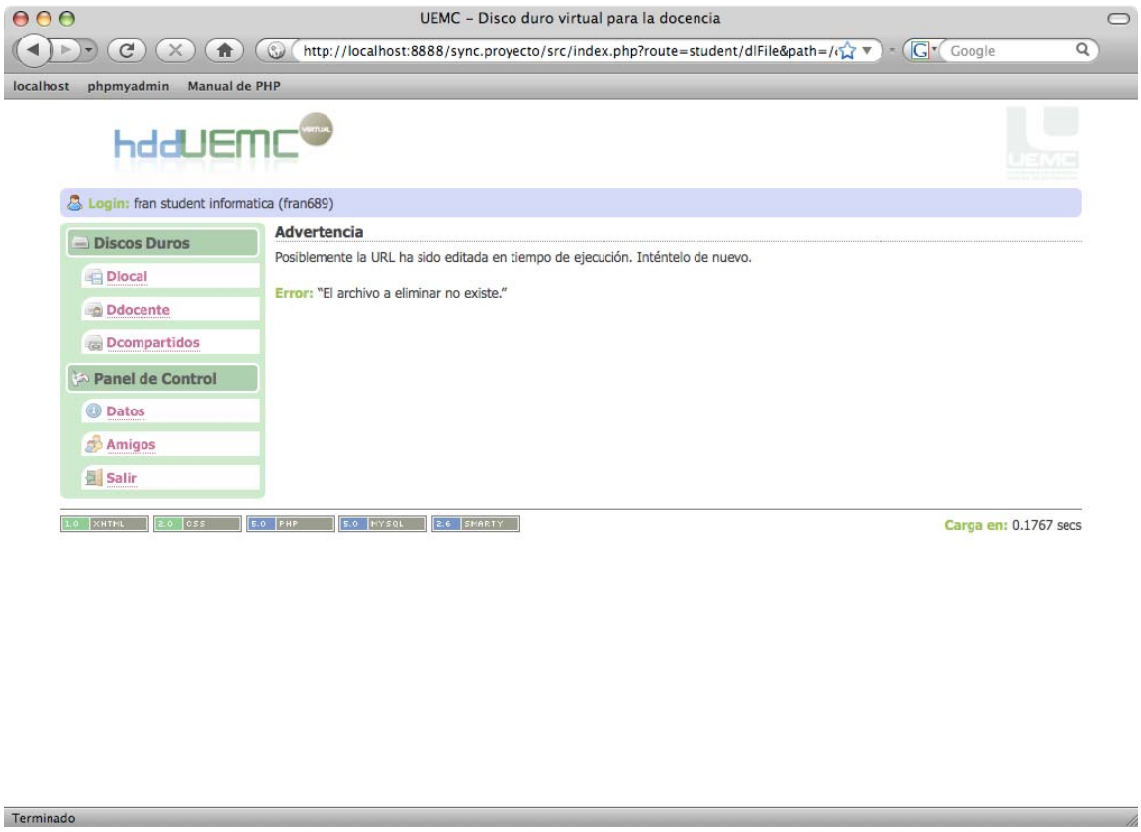


Figura 22. El archivo a eliminar no existe

Compartir

Deberemos elegir algún amigo de la lista obligatoriamente, si no salta un error que nos advierte de ello.

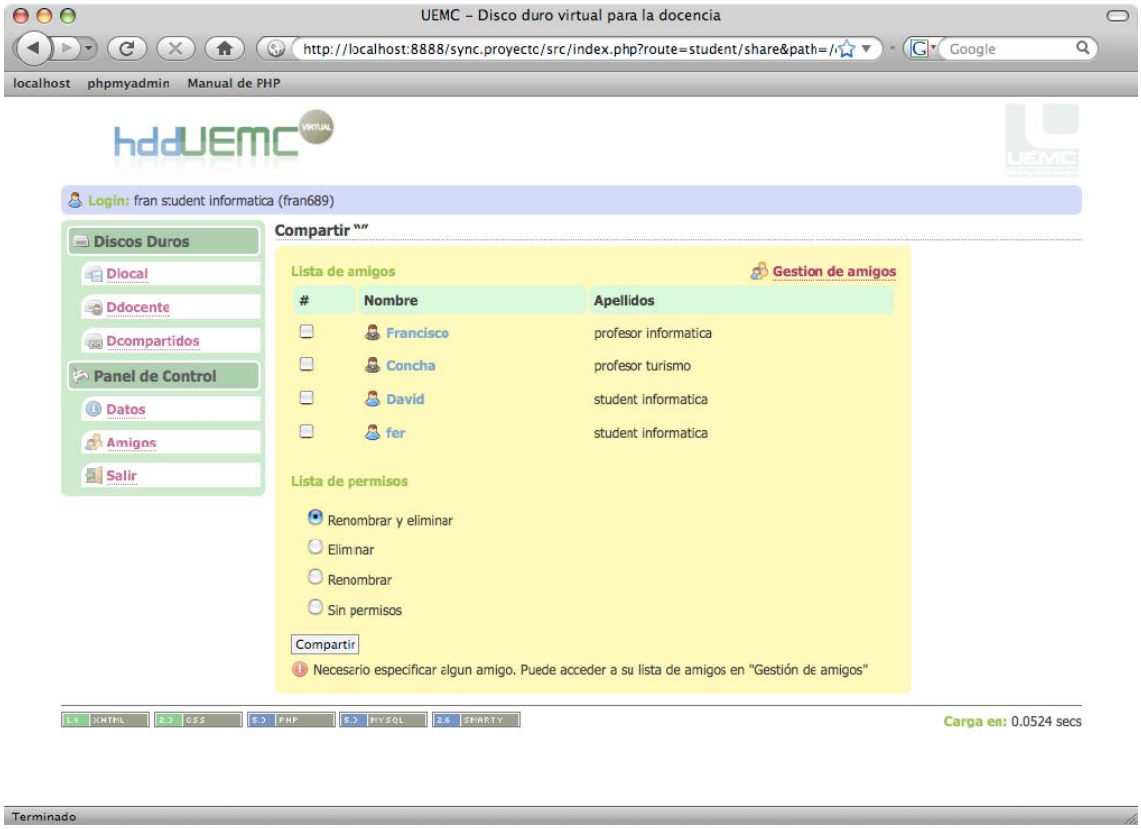


Figura 23. No hay ningún amigo elegido

Si se edita el controlador

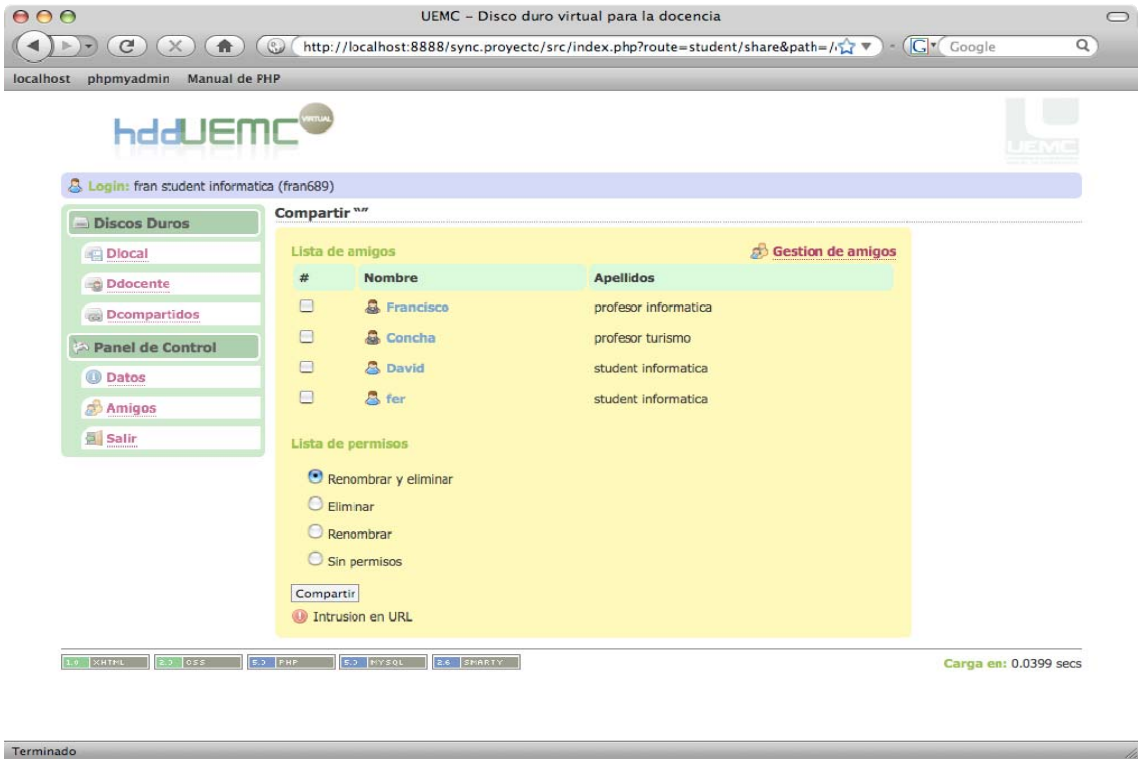


Figura 24. Controlador modificado

Gestión de amigos

Eliminar

Si no se selecciona ningún amigo

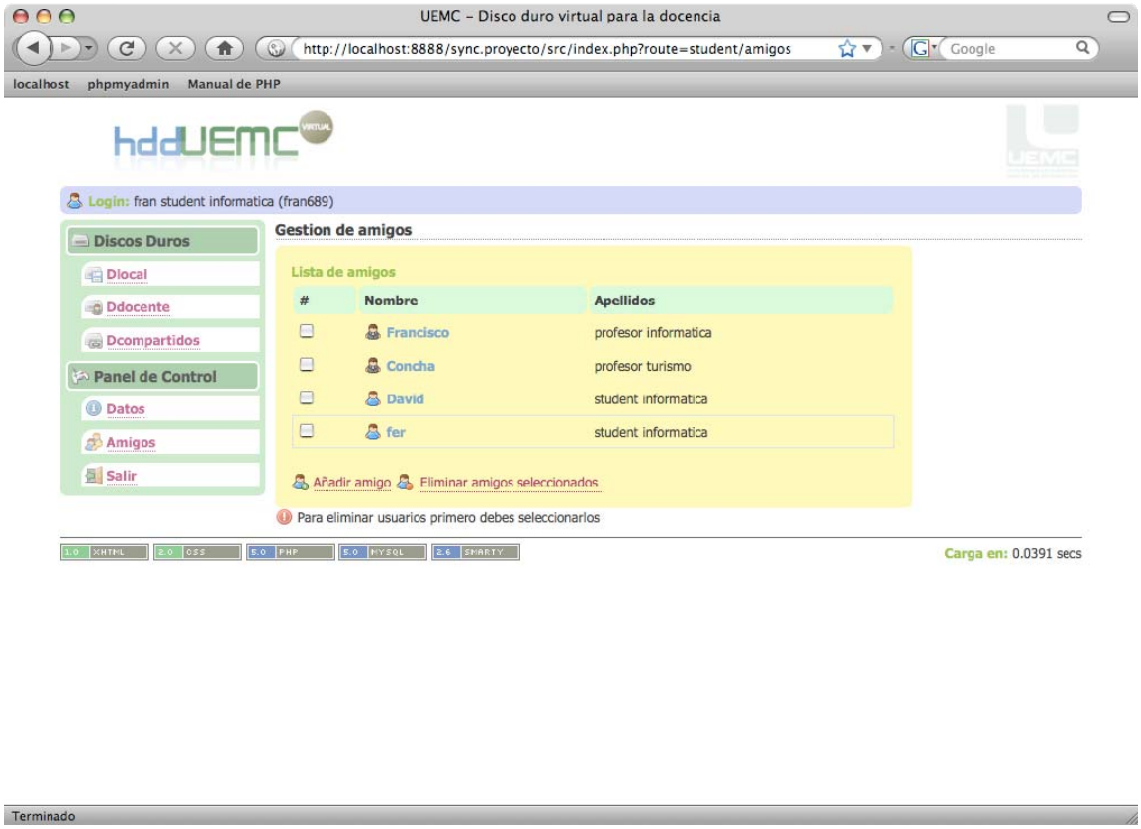


Figura 25. Error al eliminar usuario

Únicamente si escogemos alguno obtendremos mensaje de confirmación

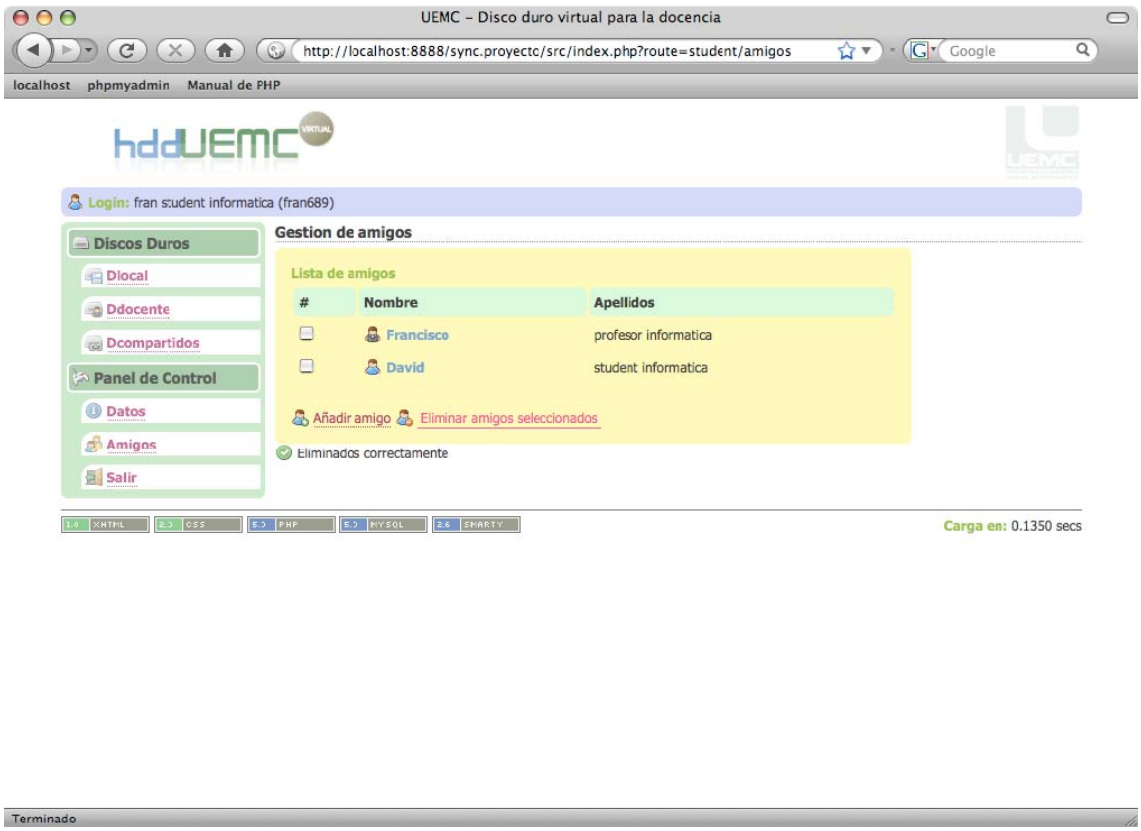


Figura 26. Usuario eliminado correctamente

Añadir amigo

Se nos indica que deberemos introducir los id's de usuario separados por espacios. Es un campo completamente formateado y únicamente introducirá usuarios válidos que no sean ya amigos y que tampoco puede ser propio usuario.

Por lo que suponiendo que hemos introducido en el formulario "”·$!$__,a asdasd fran689 ivan689 W·$%F asd$ eva689" este es el mensaje que obtendremos.

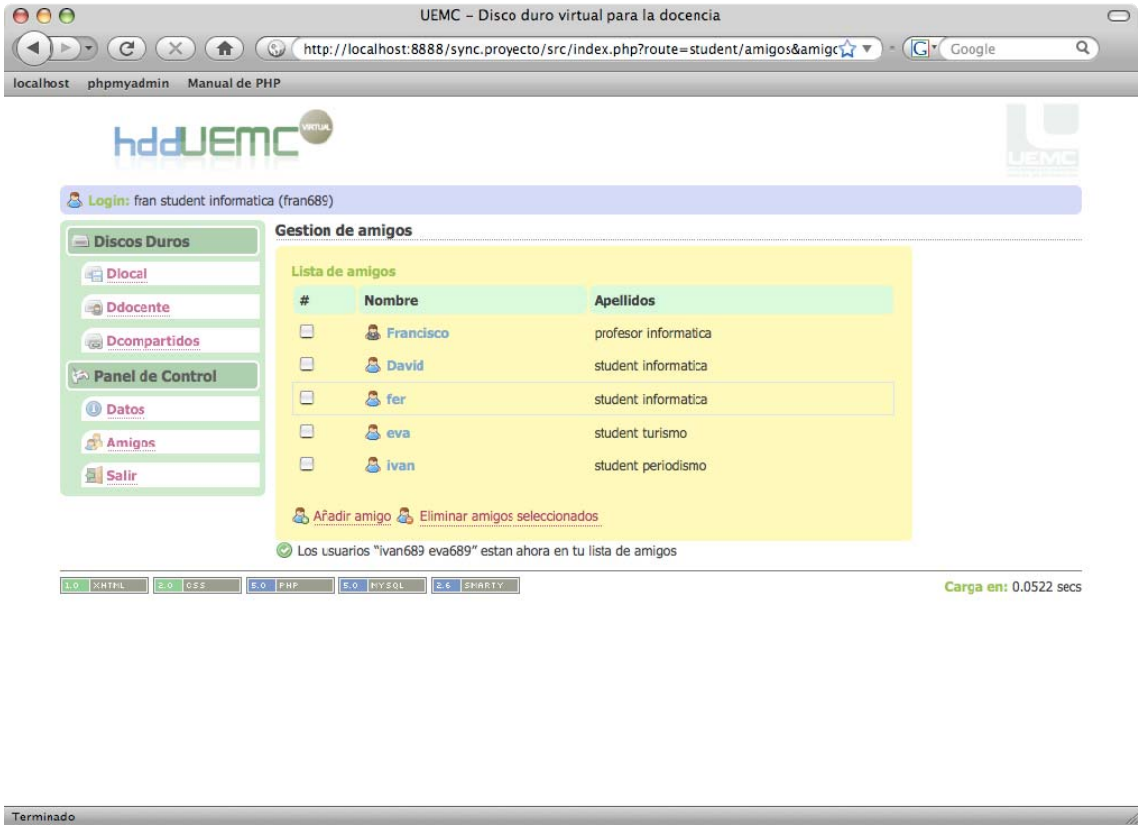


Figura 27. Añadidos amigos

Disco compartido

Se aplica como normal general que hay diferentes pruebas que son idénticas que en el disco local, como la inserción de caracteres comprometidos, los cuales son meros filtrados de texto que muestran el mismo error para las mismas pantallas.

Por lo tanto se mostraran otras pruebas.

Editando controladores

Suponiendo que el usuario a llegado a conocer el ID de un fichero que no le pertenece ni propietariamente ni por compartidos, si intentara efectuar cualquier operación con él, se generaría un error.

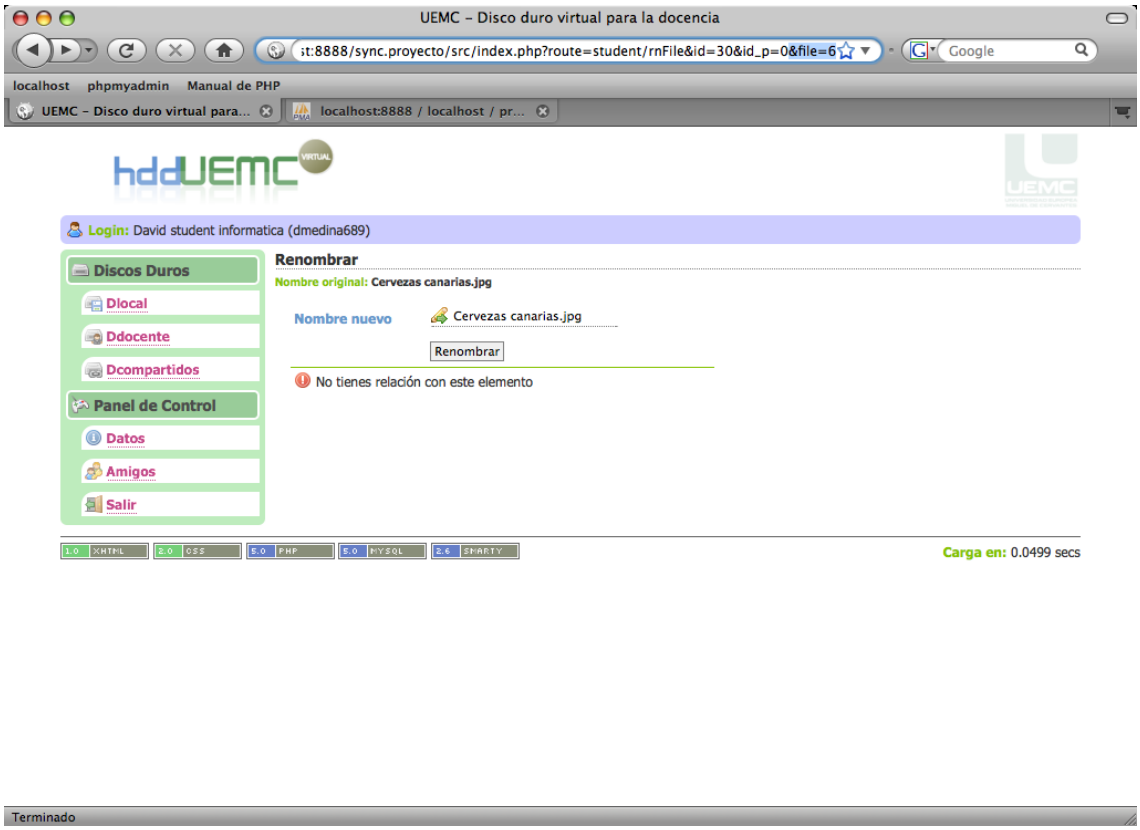


Figura 28. Modificar la URL

O si no se tuvieran permisos sobre el archivo, aunque estuviera compartido

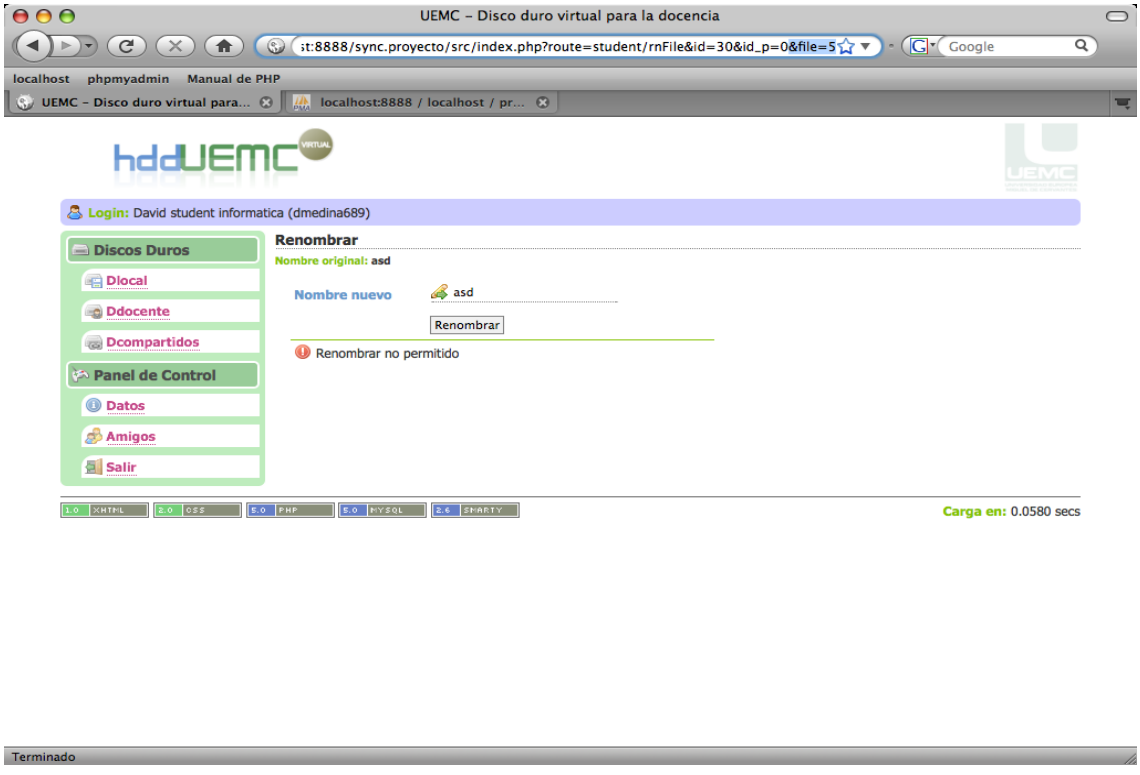


Figura 29. Error al renombrar sin permisos por URL

Figura 30. Error al eliminar sin permisos por URL

Disco docente

Respecto al alumno

Si se editaran los controladores con una ID de asignatura que no te perteneciera.

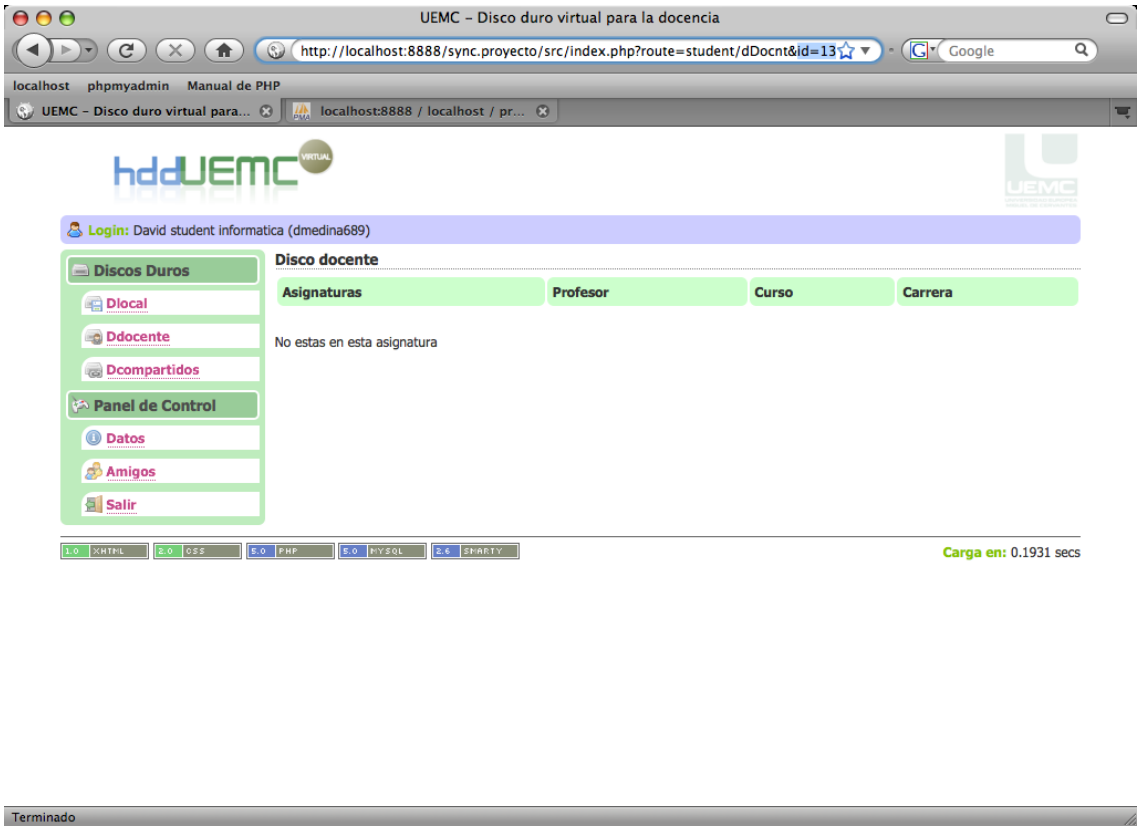


Figura 31. Ver asignatura no matriculada

Figura 32. Subir archivo a asignatura no matriculada

Si el archivo es correcto respecto a tamaño, caracteres y demás pruebas ya

realizadas con anterioridad.

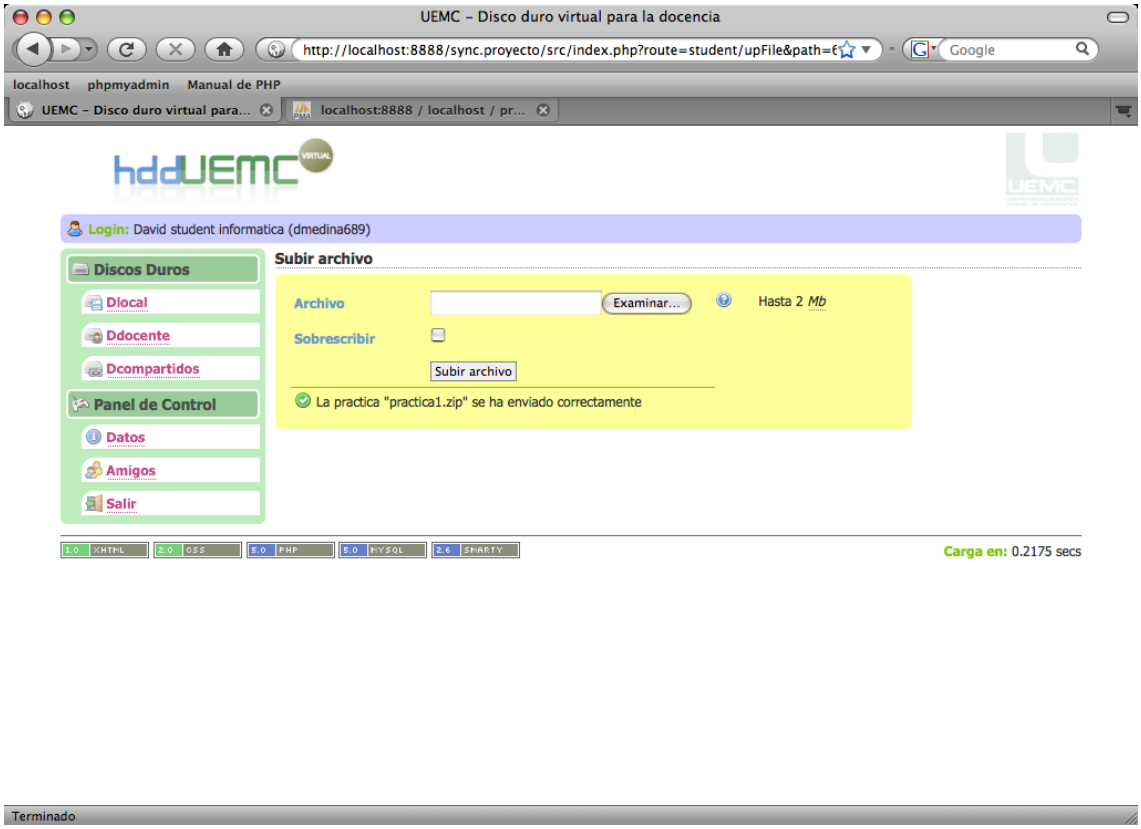


Figura 33. Subir archivo correctamente

Aunque en caso de que ya existiera y si no se ha marcado la opción sobrescribir.

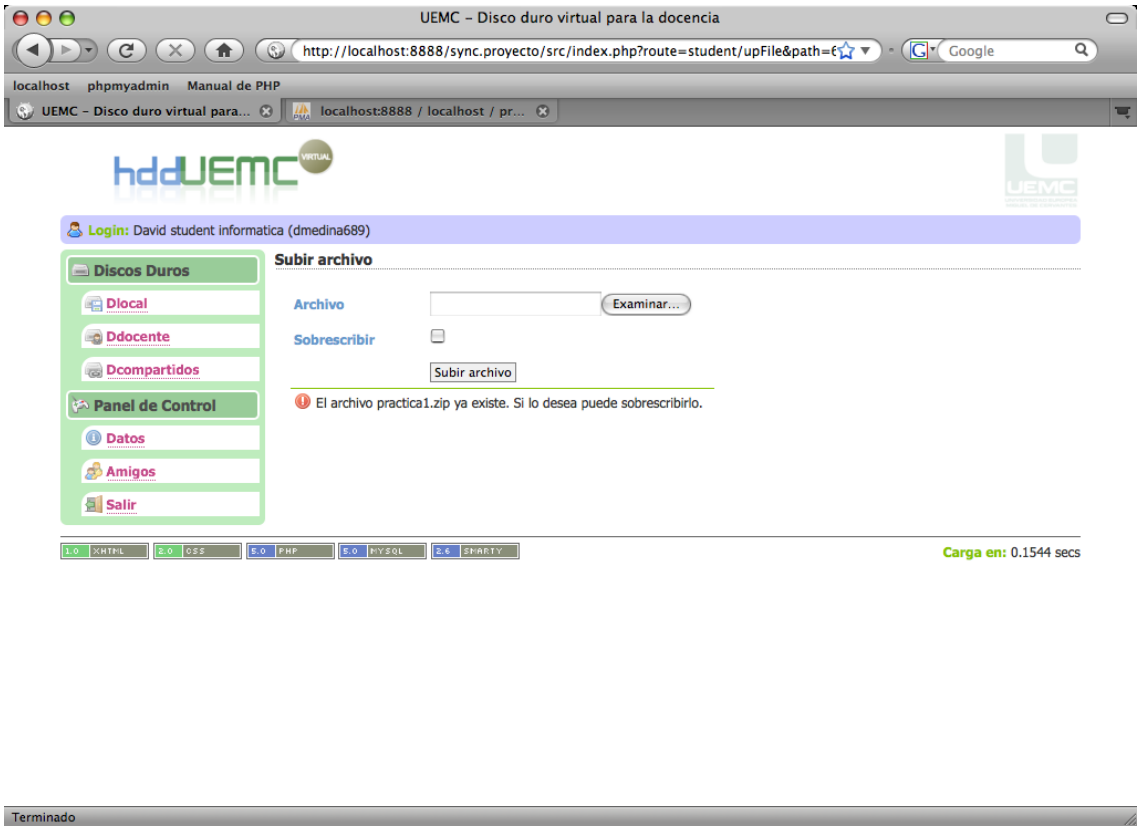


Figura 34. El archivo ya existe

Para el profesor comentar que se trata similar al disco local por lo que las pruebas serán idénticas con las mismas respuestas.

Conclusiones

Hemos podido comprobar que se intentan tratar todas las posibilidades desde todos los datos de entrada que puede introducir el usuario.

Comentar que el uso de PDO junto con sus características como *prepared statments* y *placeholders* añadido a la directiva *magic_quotes* de PHP activada por defecto nos ha evitado muchos problemas y reducido ampliamente costes de tiempo y producción.

6. Instalación y configuración de la aplicación

A fecha de 14 de Febrero sale la versión 5.0 de Debian. Basamos el entorno de producción en Debian 5.0, con lo que todos los comandos y rutas de configuración estarán basados en esta distro.

Como información adicional la versión del kernel donde instalamos está en la rama 2.6.26 y el sistema de ficheros está formado en *ext3*.

A partir de ahora estaremos en entorno de *root*

Procedemos a instalar el servidor web Apache, la rama 2.2

```
# apt-get install apache 2
```

Una vez instalado el servicio se arrancará automáticamente y la ruta de configuración se encontrará en `/etc/apache2` .

Usamos la configuración por defecto basado en virtualhosts corriendo sobre el usuario *www-data*, a excepción de ciertas directivas que modificamos personalmente como reglas de directorios o errores por defecto.

Edición del virtualhost por defecto situado en:

`/etc/apache2/sites-enabled/000-default`

Una vez configurado el servidor instalamos php5 versión 5.2.6

```
# apt-get install php5
```

Instalamos un servidor de base de datos

```
# apt-get install mysql-server
```

Pide una nueva contraseña para mysql, insertamos la contraseña que queramos.

Instalamos la extensión para que php conecte con mysql

```
# apt-get install php5-mysql
```

Con este comando se instala también la librería de Abstracción PDO (capa intermedia que realiza conexiones con diferentes bases de datos) y el driver *pdo-mysql*.

La ruta hacia el virtualhost por defecto se encuentra en */var/www* por lo que en este directorio es donde copiamos los archivos de la aplicación.
Suponiendo que los archivos de la aplicación están en */home/proyecto*

```
# cp -rv /home/proyecto /var/www
```

Modificamos los permisos con el comando

```
# chown www-data:www-data /var/www -R
```

Exportamos el archivo proyecto.sql que contiene la base de datos a nuestro servidor.

Pasos

1- Conectarnos a mysql:

```
# mysql -u root -p
```

Nos pide la contraseña que hayamos introducido durante la instalación

2- Una vez en la línea de comandos de mysql, introducir:

```
mysql> source /home/proyecto/proyecto.sql
```

3- Salir de mysql

```
mysql> quit
```

Después debemos configurar los privilegios de usuario sobre la base de datos creada para mejorar su seguridad. Detallamos la consulta SQL pudiéndola introducir con PhpMyAdmin

```
CREATE USER nombreUsuario@localhost IDENTIFIED BY ‘password’;

GRANT SELECT ON *.*
TO ‘nombreUsuario’@‘localhost’ IDENTIFIED BY ‘password’
WITH MAX_QUERIES_PER_HOUR 0
MAZ_CONNECTIONS_PER_HOUR 0
MAX_UPDATES_PER_HOUR 0
MAX_USER_CONNECTIONS 0;

GRANT SELECT, INSERT, UPDATE, DELETE ON ‘nombreBaseDatos’,
‘ficheros’ TO ‘nombreUsuario’@‘localhost’;

GRANT SELECT, INSERT, UPDATE, DELETE ON ‘nombreBaseDatos’,
‘ficheros_has_grupos’ TO ‘nombreUsuario’@‘localhost’;

GRANT SELECT, INSERT, UPDATE, DELETE ON ‘nombreBaseDatos’,
‘grupos’ TO ‘nombreUsuario’@‘localhost’;

GRANT SELECT, INSERT, UPDATE, DELETE ON ‘nombreBaseDatos’,
‘usuarios_has_grupos’ TO ‘nombreUsuario’@‘localhost’;
```

Una vez importada la base de datos y credo el usuario, editamos el archivo de configuración de nuestra aplicación situado en */var/www/config.php* modificando los datos de conexión de la base de datos.

```
# vim /var/www/config.php
```

Editar las variables
$db_user
$db_pass
$db_db
con los datos que correspondan

Una vez instalado accedemos a la aplicación desde el navegador. En el usuario administrador pinchamos en crear directorio. Esto creará toda la estructura de directorios para los usuarios insertados en la base de datos, así como el sistema docente.

7. Futuras mejoras

Cuando se definió el proyecto pudimos observar la inmensa funcionalidad que se podía desarrollar. Sin embargo, debido a límites de tiempo tuvimos que establecer una serie de objetivos puesto que ante todo queríamos la fortaleza de la aplicación así como su posterior expansión de manera que no tuviéramos que modificar la base.

Sin embargo se plantearon una serie de funcionalidades extras durante la fase de análisis y durante el desarrollo mismo de la programación.

- **Espacio web:** Crear un espacio en disco donde el usuario pueda colocar sus scripts de la página web estáticos o un sistema de plantillas. También se podría diseñar un espacio para colocar scripts dinámicos siempre con unos permisos restrictivos y una capa de seguridad bien construida.

- **Sistemas de logs:** Desarrollo funcional para el administrador de manera que dentro de cada espacio en disco del usuario se creara un log con todos los eventos relacionados con dicho usuario creando así una capa para el administrador de manera que pudiera acceder a esos logs.

- **Sistema de mensajes:** Mensajes automáticos para los usuarios de manera que cuando actuara un evento dinámico de aplicación del estilo de compartir una carpeta al usuario al que se comparta la aplicación le envíe un mensaje avisándole de tal suceso.

- **Ampliación del sistema de mensajes:** Podríamos añadir funcionalidad a dicho sistema con las *invitaciones* de manera que una carpeta no sea compartida a menos que el usuario afectado no acepte la invitación.

- **Bloqueo de prácticas a determinadas fechas:** Crear una opción para el profesor de manera que se impida a los alumnos de la asignatura elegida por el profesor subir prácticas en determinadas fechas. Más conocido como límites de entrega

- **Envío de correo:** Crear un sistema de formularios para cada usuario de manera que puedan enviar un correo al usuario elegido.

- **Cuotas de disco:** No todos los usuarios necesitaran del mismo uso del disco. Como ejemplo pondremos alumnos de la carrera de informática, los cuales tienen cierta predisposición a subir muchos archivos y prácticas.

- **Opciones de usuario:** Crear un sistema de administración de la aplicación para cada usuario, de manera que el mismo pueda elegir cuantos archivos mostrar, que estilo desea usar en la aplicación, paginación de archivos…

- **Compresión de archivos:** Dar una opción a los usuarios que eligiendo una lista de archivos o carpetas puedas formar un archivo comprimido con ellos y dar la opción de descargarlo.

- **Implantación de *ajax*:** Debido a la expansión de esta tecnología podría diseñarse una capa intermedia entre el modelo de diseño establecido y el propio diseño de la aplicación.

- **Gestor de estilos:** Gracias a la separación de contenido y diseño que se ha realizado la creación de nuevos estilos se hace una tarea fácil.

- **Sistemas de noticias:** Implantación de un sistema por el cual un profesor pudiera subir noticias cada cierto tiempo actualizándose dinámicamente para sus alumnos.

- **Tablas ordenadas dinámicamente:** Ordenar los archivos por nombre, fecha en tiempo de ejecución. Íntimamente relacionado con la implantación de *ajax*.

Intentaremos describir brevemente el proceso de desarrollo para una de estas mejoras que iba a ser implantada antes de realizar el último de los diagramas de *gantt* donde definimos que no iba a ser viable: sistema de mensajes

Primero definiremos la funcionalidad de dicho sistema:

Queremos que al compartir un elemento con un usuario, a dicho usuario se le notifique dicho suceso. Algo que es común a todos los usuarios podría ser mas específico para los alumnos y profesores de manera que a los primeros se les avise de nuevos enunciados o temario subidos por el profesor y a estos se les notificara el envío de nuevas prácticas.

Una vez definida la funcionalidad está claro que habrá que añadir una tabla a la base de datos la cual podría ser:

Nombre	Tipo	Descripción
idMensaje	Integer	Id identificativo y clave primaria
idUsuario	Integer	Id del usuario al que le llego el mensaje
texto	varchar	Texto del mensaje
fecha	datetime	Fecha de envío
importancia	smallint	Un posible añadido a los sucesos.

Formaría una relación 1:N con la tabla usuarios con las consecuentes relaciones a nivel de base de datos para edición y eliminación dinámica a dicho nivel (en cascada).

Una vez clara la funcionalidad y editada la base de datos está claro que deberemos crear no solo los diferentes controladores sino un nuevo modelo encargado de esta tabla con el formato típico de este tipo de modelos y su sistema de herencias.

Model_mensaje extends Model_base

Se podrían crear diferentes métodos como: crearMensaje(), asignarImportancia()…

Este sería la clase a instanciar en los diferentes modelos donde se generen los sucesos tales como compartir o subir practica de manera que usando sus métodos se actualizara la base de datos en consecuencia para el uso del controlador que se describe a continuación.

Puesto que no sería usable que la aplicación mostrara durante todo su tiempo de vida la lista de mensajes para un usuario, se procedería a crear un controlador (*Controller_user::filterMensajes())* para usuario de manera que filtrara la lista de mensajes por un intervalo de fechas que podría ser definido a nivel de aplicación o a nivel de usuario.

Una vez desarrollado el controlador haríamos uso de él junto con *smarty* para mostrar únicamente lo que este controlador nos indicara.

Se ha podido demostrar que implantar un nuevo módulo no ha requerido modificar la base de nuestra aplicación más allá de lo necesario funcionalmente y su desarrollo ha sido rápido y limpio.

8. Conclusiones

La valoración que hemos sacado a la hora de terminar la aplicación, ha sido bastante positiva en todos los aspectos. Desde las tecnologías aprendidas específicamente para el desarrollo del proyecto, así como tecnologías que ya se conocían y que se han afianzado con nuevos conocimientos, hasta tecnologías totalmente novedosas y que se han usado para separar la programación del diseño gráfico y simplificar la posible modificación de la aplicación en futuras mejoras.

Los objetivos que nos propusimos realizar sobre la aplicación, se han cumplido y esto ha proporcionado que se pudieran implementar nuevas funcionalidades secundarias que también habíamos pensado, pero que dependiendo de cómo se cumplieran los tiempos estimados para realizar las tareas principales, se implementarían o se dejarían como posibles mejoras. La planificación de los tiempos para las tareas nos ha sido muy útil desde el punto de vista en nos hemos obligado a seguir unos tiempos razonables de desarrollo y estudio, que de otra manera no se hubieran podido realizar por el excesivo retraso.

Según nuestro criterio, es bueno marcarse una fecha límite de finalización del proyecto, ya que hemos sufrido las consecuencias de no hacerlo y ver que el tiempo corre en nuestra contra. Al no marcar unos tiempos para cada desarrollo, nos vimos con un estudio de tecnologías muy amplio, muy profundo y que nos retardaba la entrega del proyecto en exceso. Crea también una situación de desánimo el ver que todo lo que se estudia no se refleja en el proyecto en un primer momento, pero que a la larga puede diferenciar un proyecto mediocre de uno muy bueno.

Comentar también que a medida que se iba desarrollando también nos documentamos puesto que al enfrentarnos a diferentes problemas las soluciones que se nos ocurrían o no utilizaban la metodología orientada a objetos o nos parecía obsoleta (Como ejemplo podemos indicar recorrer los archivos de un directorio que en un principio iba a hacerse con las funciones base de PHP, cuando nos documentamos sobre la librería SPL descubrimos los patrones iterator).

Esto nos conducía a una situación en que siempre estábamos documentándonos

y no avanzábamos pues nos introducíamos en una espiral difícil de solventar. En este punto decidimos marcarnos unas metas y cumplirlas sin investigar más de lo necesario. Con ello aumentamos los flujos de trabajo rápidamente sin perder en ningún momento el enfoque de utilizar el paradigma de la orientación a objetos y las nuevas tecnologías.

Al estar enfocada esta aplicación a usuarios que pueden no tener ningún conocimiento de informática, creímos que la realización de un diseño intuitivo para el usuario debía ser una prioridad. En este aspecto nos hemos volcado creando un sistema fácil de comprender y manejar por cualquier usuario, ayudado con pequeñas ayudas en momentos que pudieran ser necesarias, pero sin perder la funcionalidad que debía tener un sistema con este potencial.

Creemos que la implantación de la aplicación, podría crear un punto de inflexión en la UEMC, ya que crea un sistema de gestión de archivos para los usuarios, que revierte en el propio beneficio de la Universidad, ya sea para la entrega de trabajos o incluso para poder hacer exámenes on-line desde la cuenta del propio usuario.

Se puede implantar fácilmente a otras opciones que actualmente existen, como son las cuentas de correo para cada usuario.

Tras haber terminado el proyecto, nos damos cuenta que las tecnologías elegidas han sido un gran acierto. Obviando las tecnologías obligatorias que se han tenido que utilizar y que tenían que ser libre, creemos que hemos acertado en el uso de otras tecnologías que siguen la premisa *opensource*.

Una de estas tecnologías y que más podemos destacar es **Smarty**, que permite separar el código, de la presentación evitando el engorroso código embebido. Esto que igual a simple vista pueda parecer que no tiene relevancia, ha marcado gran parte del proyecto, ya que podemos observar plantillas limpias fáciles de ver cuando se quieran diseñar, *escalabilidad*, permite un mantenimiento más sencillo y óptimo del código, al tener ficheros más pequeños únicamente con código PHP, así como funciones que facilitan el tratamiento de variables.

Esta tecnología debemos destacarla por encima del resto que se hemos usado,

ya que para nosotros era totalmente desconocida, y tuvimos que partir de cero, ver la potencia que tenía e implementarla en nuestro proyecto de la manera en que lo hemos hecho, dando como resultado el resultado mostrado a lo largo de esta memoria.

También debemos valorar muy positivamente la Orientación a Objetos que hemos podido desarrollar con PHP, ya que está todavía en proceso de desarrollo y que se ampliará próximamente con PHP 6.

Con las funciones que tiene PHP 5 para manejar objetos y que hemos usado, animamos a programar a todos aquellos que se enfrenten a este lenguaje de programación, ya que aunque puede resultar tedioso coger el concepto de los objetos, una vez se entienden, la programación se hace bastante rápida. Esta rapidez se debe a la implementación una única vez de un módulo, y todas las llamadas que se quieran desde cualquier lugar del código sin tener que volver a escribir el mismo código. Es una tarea lenta la que lleva pensar cómo se va a organizar el código, pero una vez se tienen los conceptos claros, todas las modificaciones que se hacen sobre el código, se hacen una vez y sirven para todos. No requiere copiar y pegar código y cambiar lo mismo en todos los sitios.

El mayor problema que nos hemos encontrado en este aspecto, ha sido pasar de pensar y programar código estructurado, a escribir código con orientación a objetos, pero una vez se salva ese obstáculo, todo se hace más sencillo.

Gracias a la metodología de programación extrema o extreme programming, en la que cada uno programa sus módulos, pero son testeados por el otro, hemos podido desarrollar una aplicación tan sólida internamente, ya que uno mismo no se da cuenta de los fallos que puede cometer, mientras que de esta manera el control por parte de otra persona y las discusiones con los diferentes enfoques que se generan, logran un código más depurado y con mayor estabilidad.

Con todo ello podemos concluir que este proyecto nos ha ayudado notablemente a incrementar nuestros conocimientos en todas las tecnologías utilizadas, así como diferentes métodos y técnicas a la hora de generar código.

No podemos terminar esta sección sin mencionar que la utilización de diverso

material de aprendizaje en forma de libros o páginas web, ha mejorado en nosotros la habilidad para buscar información y discriminar la información válida de aquella que no lo es.

9. Anexos

9.1. Manual de usuario

Una vez introducida la dirección en el navegador, lo primero que nos encontraremos es una pantalla de bienvenida.

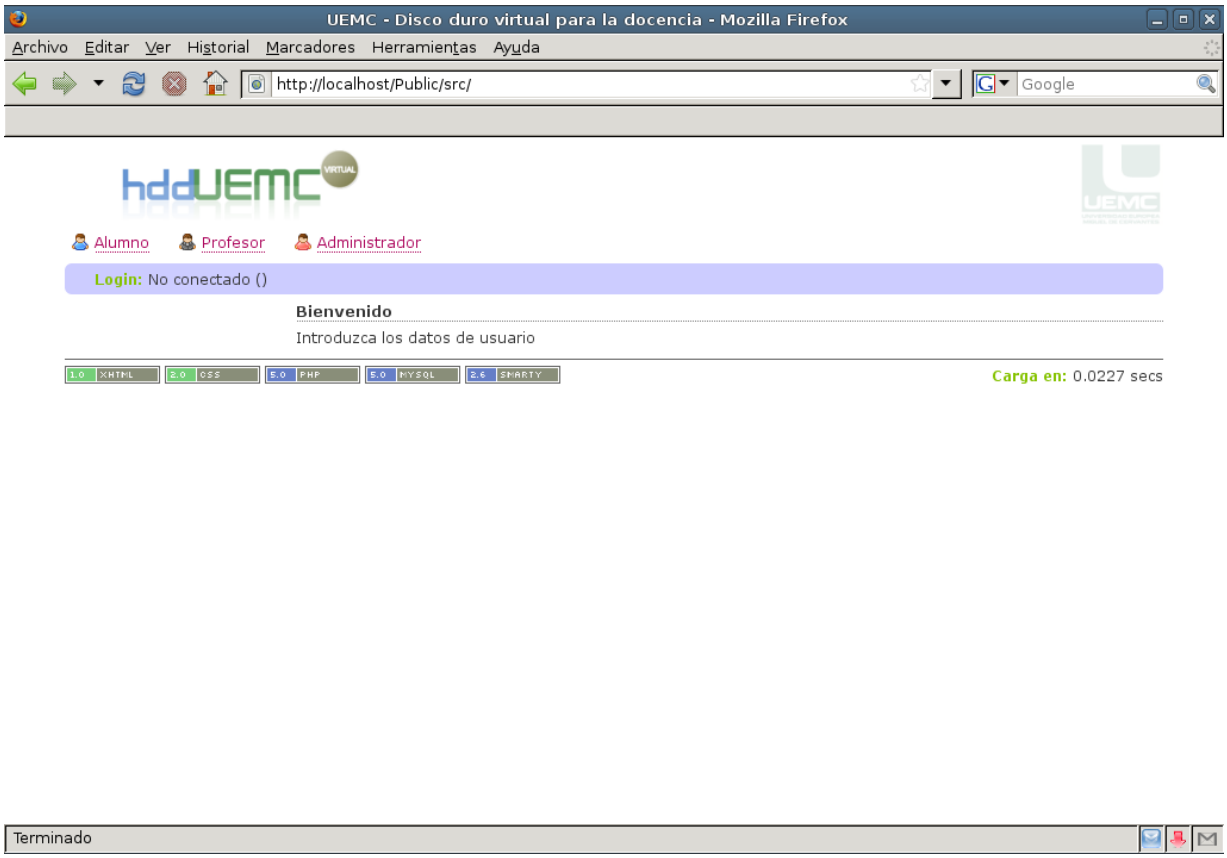


Figura 35. Pantalla de bienvenida

En la parte superior veremos un menú de navegación con 3 opciones: alumno, profesor y administrador.

Dependiendo del usuario que se te haya proporcionado deberás escoger una de estas tres opciones para pasar a la pantalla de acceso.

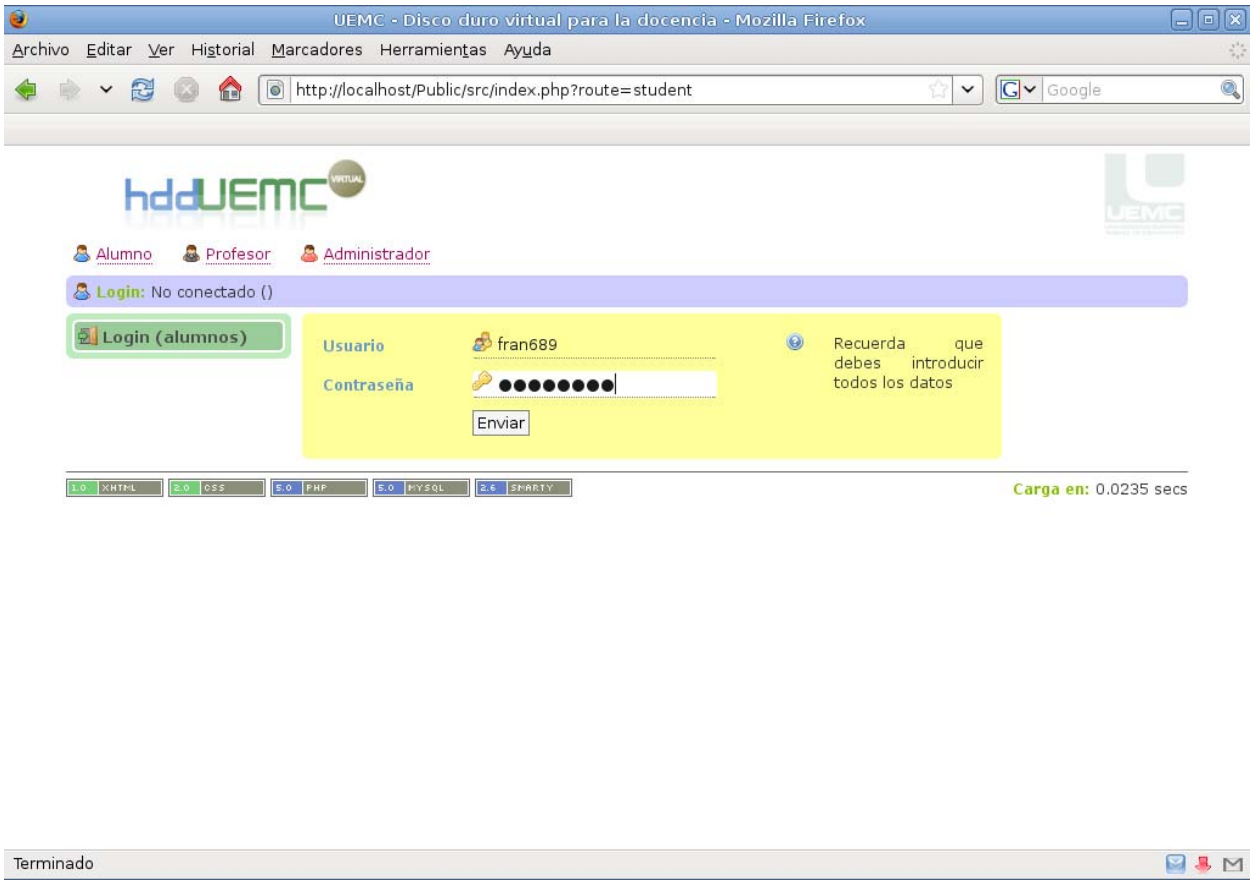


Figura 36. Pantalla de acceso

En esta pantalla podemos ver dos campos de texto claramente identificados: el nombre de usuario y su contraseña. Para poder acceder, deberemos introducir los datos que previamente nos ha proporcionado la entidad encargada de la gestión de la aplicación, en este caso la Universidad.

Cabe destacar que al situar el ratón encima de los campos de texto, se nos advierte en la parte derecha del formulario con un icono identificativo de información, que deberemos introducir ambos datos correctamente.

Una vez introducidos estos datos, y si los mismos son correctos accederemos a la pantalla principal de cualquier usuario; el disco local.

En la pantalla que se nos muestra podemos diferenciar dos partes claramente identificadas

- Menú lateral izquierdo: donde están situados los tres discos que posee cada usuario, a parte de las opciones propias del usuario.

- Parte central: donde encontramos el contenido de las opciones ejecutadas

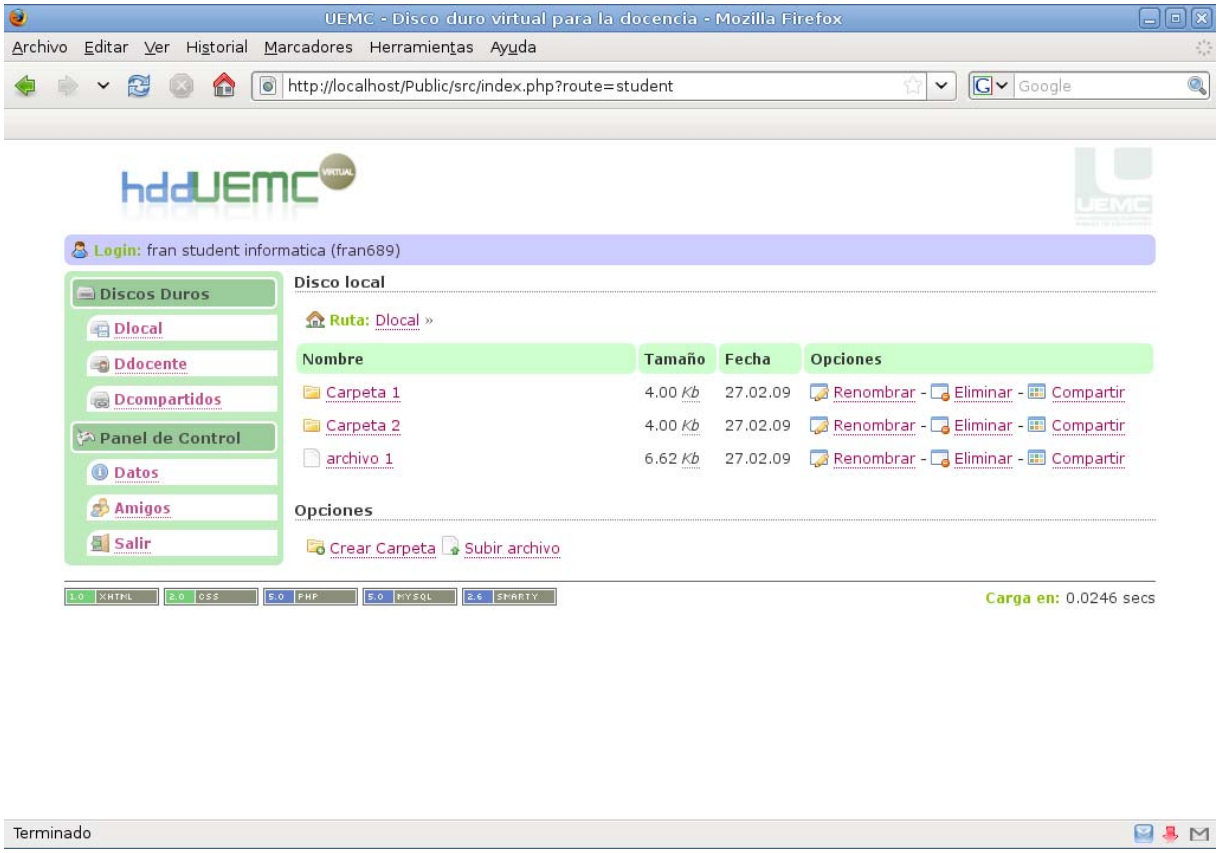


Figura 37. Pantalla principal de disco local

Puesto que una vez identificados en el sistema accedemos directamente al disco local, procedemos a describir su funcionalidad.

En la pantalla central se visualiza en **primer lugar** un menú de navegación en el se nos permite interactuar entre las diferentes carpetas del directorio en el que estemos situados. Para diferenciar este menú se muestra un icono de una casa que simboliza el directorio principal.

En **segundo lugar** se visualizan las carpetas y ficheros ordenados jerárquica y alfabéticamente que al igual que el menú de navegación superior cambiará dependiendo de donde estemos situados.

Como detalles de cada elemento, podemos ver tanto su nombre, como su tamaño en kilobytes, la fecha de modificación y la opción *Renombrar*, *Eliminar* y *Compartir*

(se detallan más adelante).

En **tercer lugar** veremos una serie de opciones propias del directorio local:

1. Crear carpeta

   Cuando pinchemos en ella, nos redirigirá a otra pantalla donde aparece un formulario con un campo de texto en el que introduciremos el nombre de la nueva carpeta a crear. Una vez hayamos rellenado este campo, pinchamos en el botón que pone *Crear.*

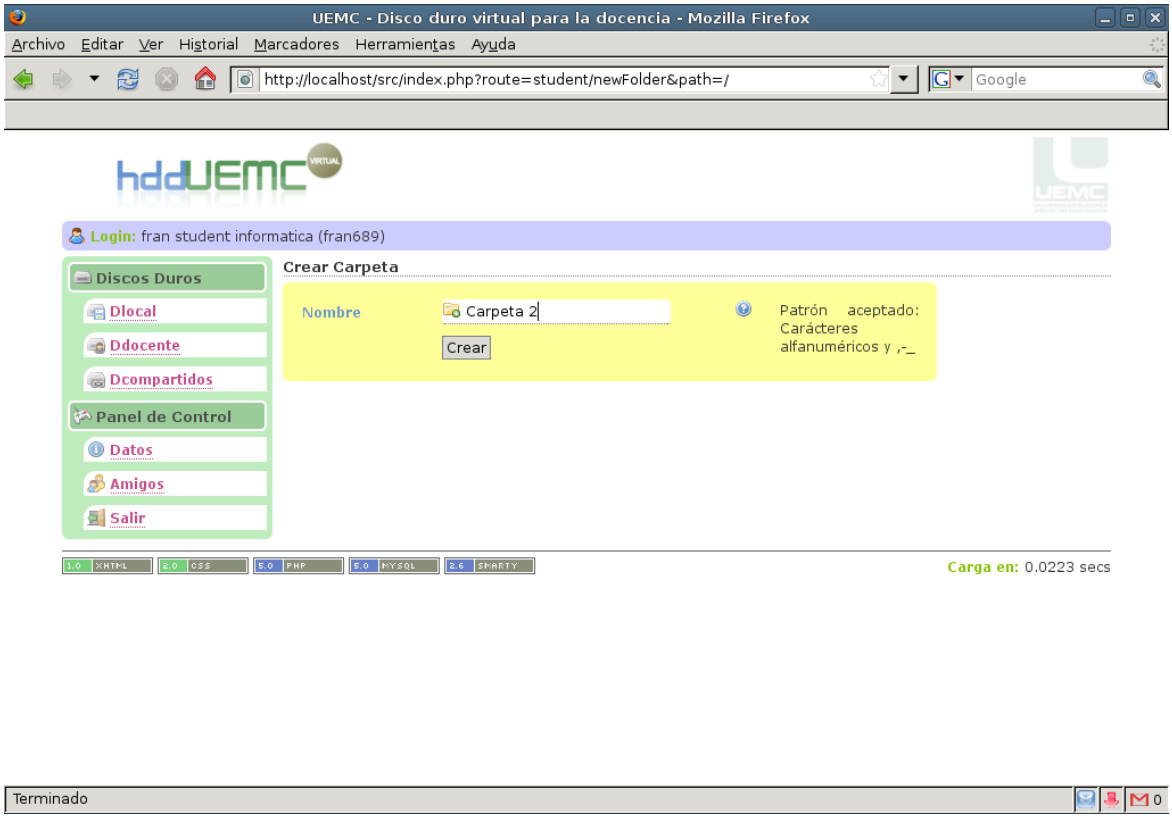


Figura 38. Crear carpeta

   En esta pantalla también se muestra un aviso con los caracteres aceptados para rellenar el campo de texto.

   El campo de texto no puede quedar vacío y tampoco puede tener un nombre de carpeta de otra que ya exista y que esté a su mismo nivel jerárquico.

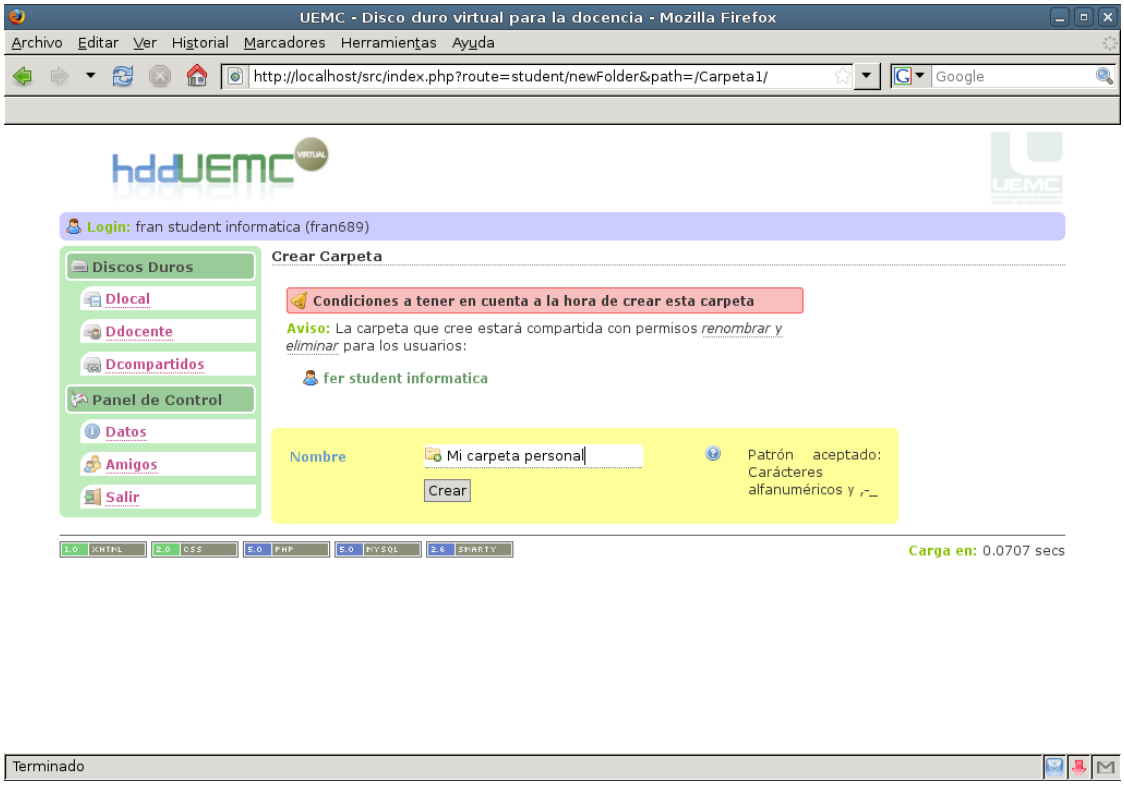


Figura 39. Crear carpeta en otra compartida

En el supuesto caso en que la carpeta padre estuviera compartida, se mostrará un aviso que indica los permisos con los que se crea y los usuarios con los que se ha compartido.

2. Subir archivo

Cuando pinchemos en esta opción, nos redirigirá a otra pantalla donde aparece un formulario con un campo *Examinar,* en el que seleccionaremos el archivo a subir y una vez elegido, pincharemos en el botón *Subir archivo*.

En el caso de que haya un archivo con el mismo nombre en el directorio donde nos encontremos, deberemos seleccionar la casilla *Sobrescribir.* De esta manera el archivo original será reemplazado por el seleccionado.

En esta pantalla también se muestra un aviso con el tamaño máximo que puede tener el archivo.

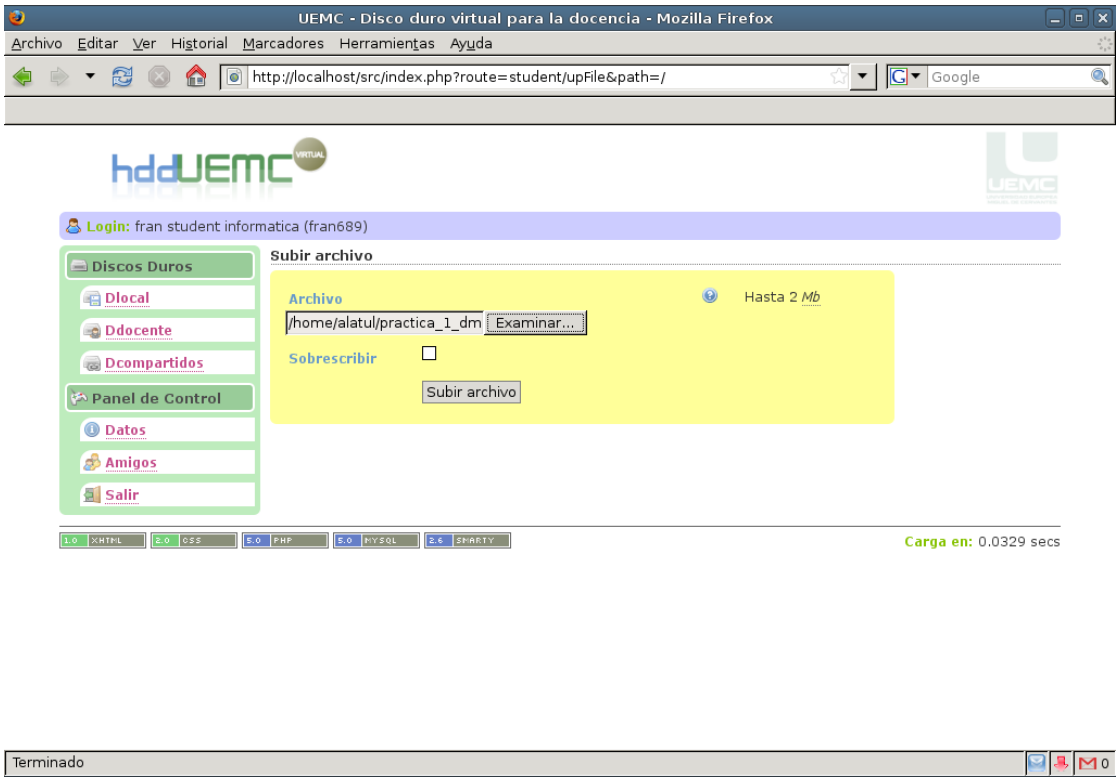


Figura 40. Subir archivo

En el supuesto caso en que la carpeta padre estuviera compartida, se mostrará un aviso que indica los permisos con los que se sube y los usuarios con los que se ha compartido.

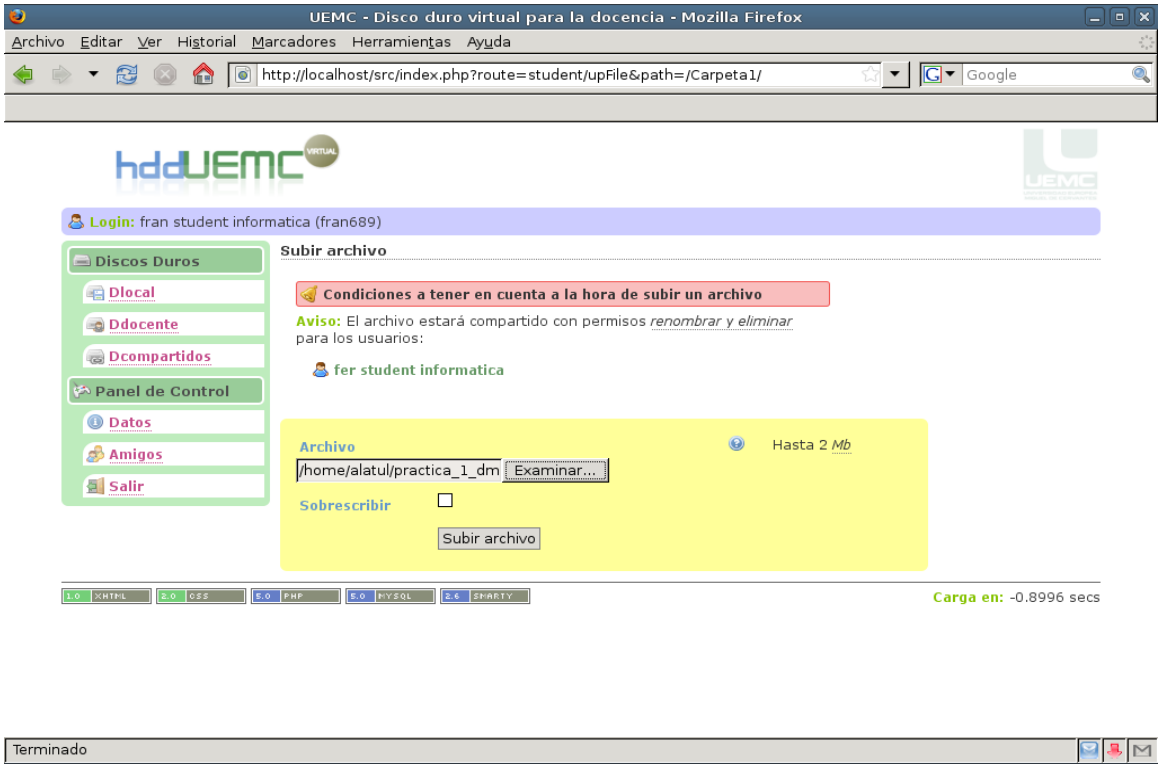


Figura 41. Subir archivo en carpeta compartida

En este punto se procede a describir las opciones para cada elemento que se había dejado pendiente. Como recordatorio indicamos que éstas son: *Renombrar, Eliminar* y *Compartir.*

1. *Renombrar*

   Cuando pinchamos en esta opción, nos redirigirá a otra pantalla en la que se nos mostrará el nombre original del elemento, así como un campo de texto con el nombre actual, en el que podremos modificar a nuestro antojo siguiendo las indicaciones informativas que nos aparecen para insertar correctamente el nombre. Una vez introducidos los cambios se pulsará sobre el botón *Renombrar*, que nos enviará de vuelta donde estuviéramos con los cambios aplicados.

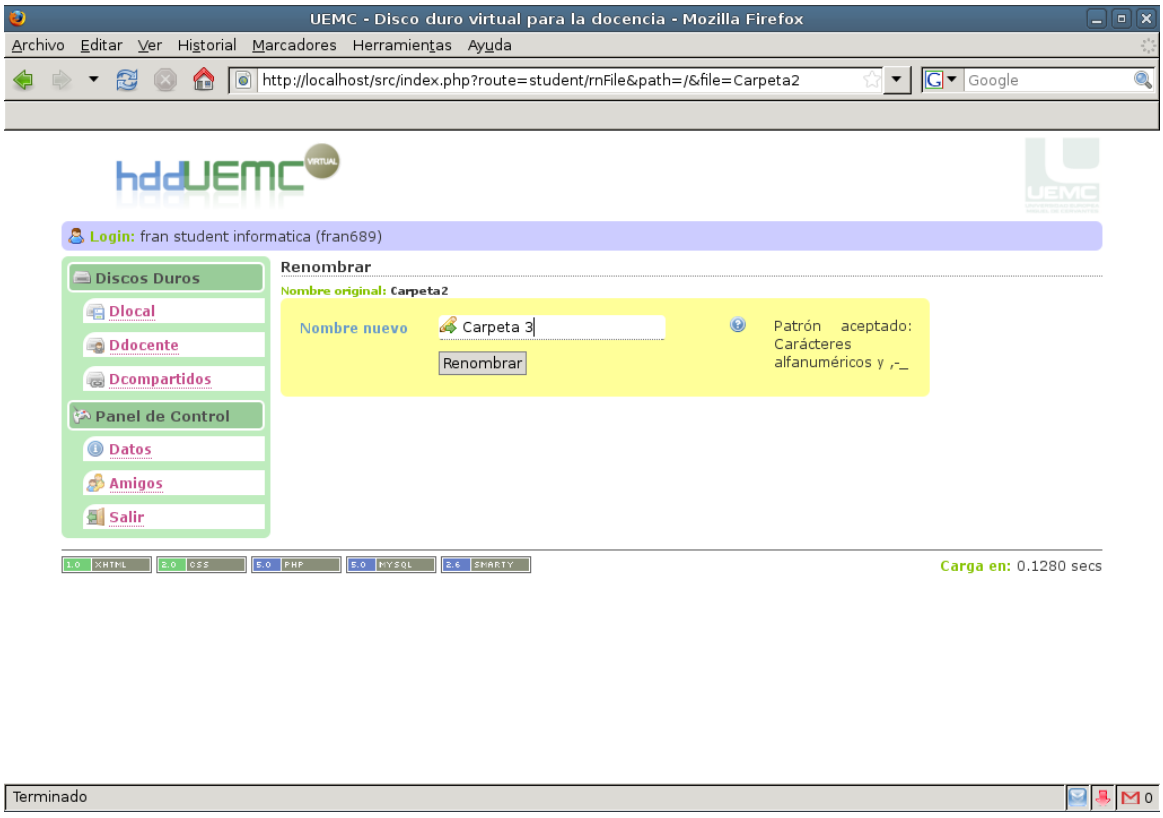


Figura 42. Renombrar carpeta

2. *Eliminar*

   Al pinchar sobre esta opción, se eliminará el elemento y en caso de ser una carpeta, se perderá toda la información que ésta contenga. Tenga cuidado.

3. *Compartir*

   Si elegimos esta opción, nos aparece otra pantalla que nos muestra tanto los amigos que tenemos, con su nombre y apellidos, como una lista de permisos a aplicar sobre el elemento a compartir.

   Seleccionamos los amigos a los que queremos compartir el elemento pulsando sobre las casillas de verificación, que se pondrán con una marca, para diferenciarlos de los que no se han elegido.

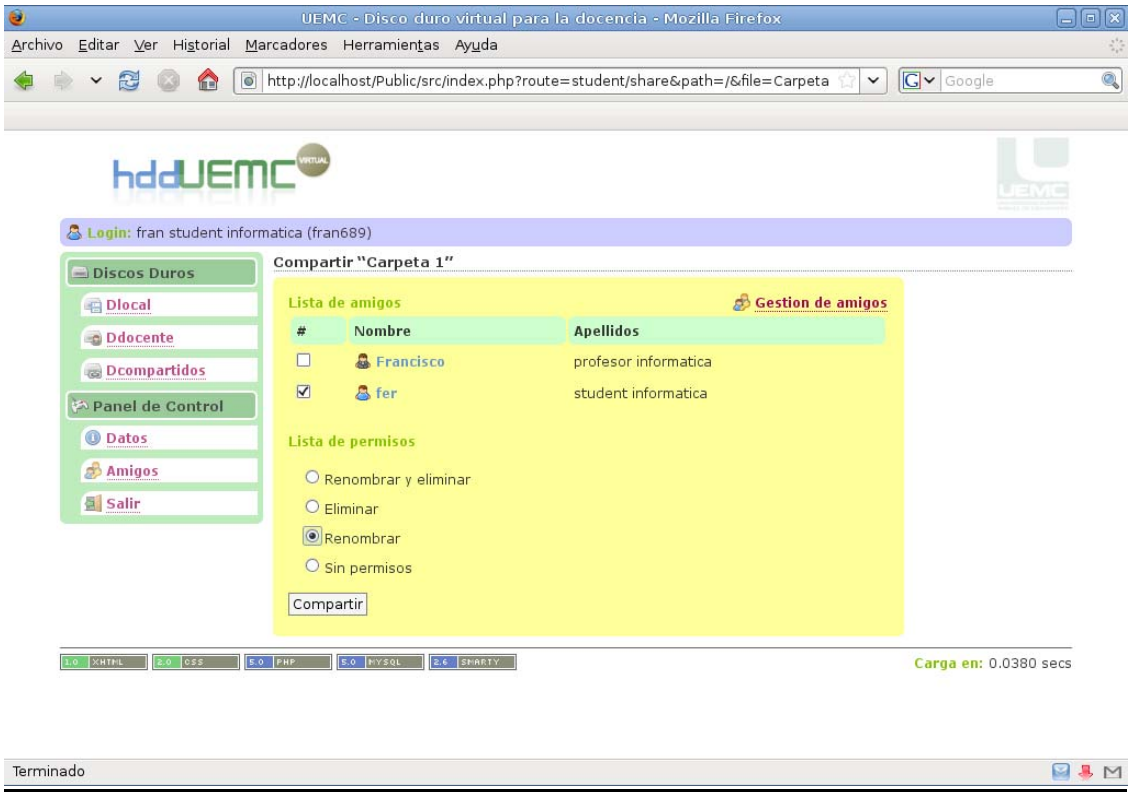


Figura 43. Compartir carpeta

Se disponen de cuatro permisos específicos, pudiendo elegir solo uno, cuyo significado queda implícito en su nombre.

1. *Renombrar y eliminar*
2. *Eliminar*

3. *Renombrar*
4. *Sin permisos*

Ya elegidos los amigos y los permisos aplicables, podremos compartir el elemento pulsando en el botón *Compartir* que nos redirigirá a la pantalla anterior.

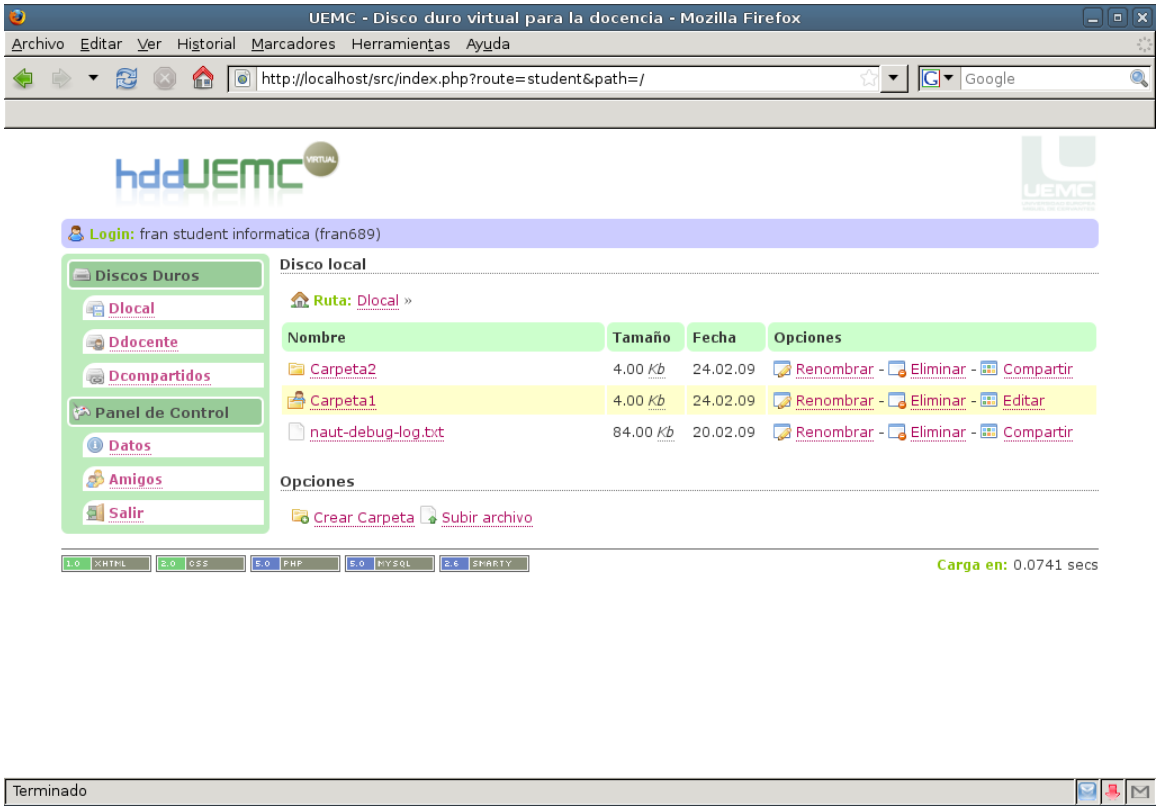


Figura 44. Disco local con compartidos

Observaremos que el elemento compartido ha modificado su aspecto en nuestro disco local, diferenciándolo con un ligero fondo marrón claro así como un icono diferente.

En caso de que quisiéramos compartir y no tuviéramos ningún amigo o quisiéramos añadir algún otro a los que ya teníamos, se deberá pinchar sobre el enlace Gestión de amigos, que desarrollaremos más tarde.

Una vez se ha compartido con algún usuario y estamos situados en la pantalla del disco local, cambiará la opción *Compartir* y en su lugar pondrá *Editar*. Si

pinchamos sobre esta última opción nos redirigirá a una nueva pantalla.

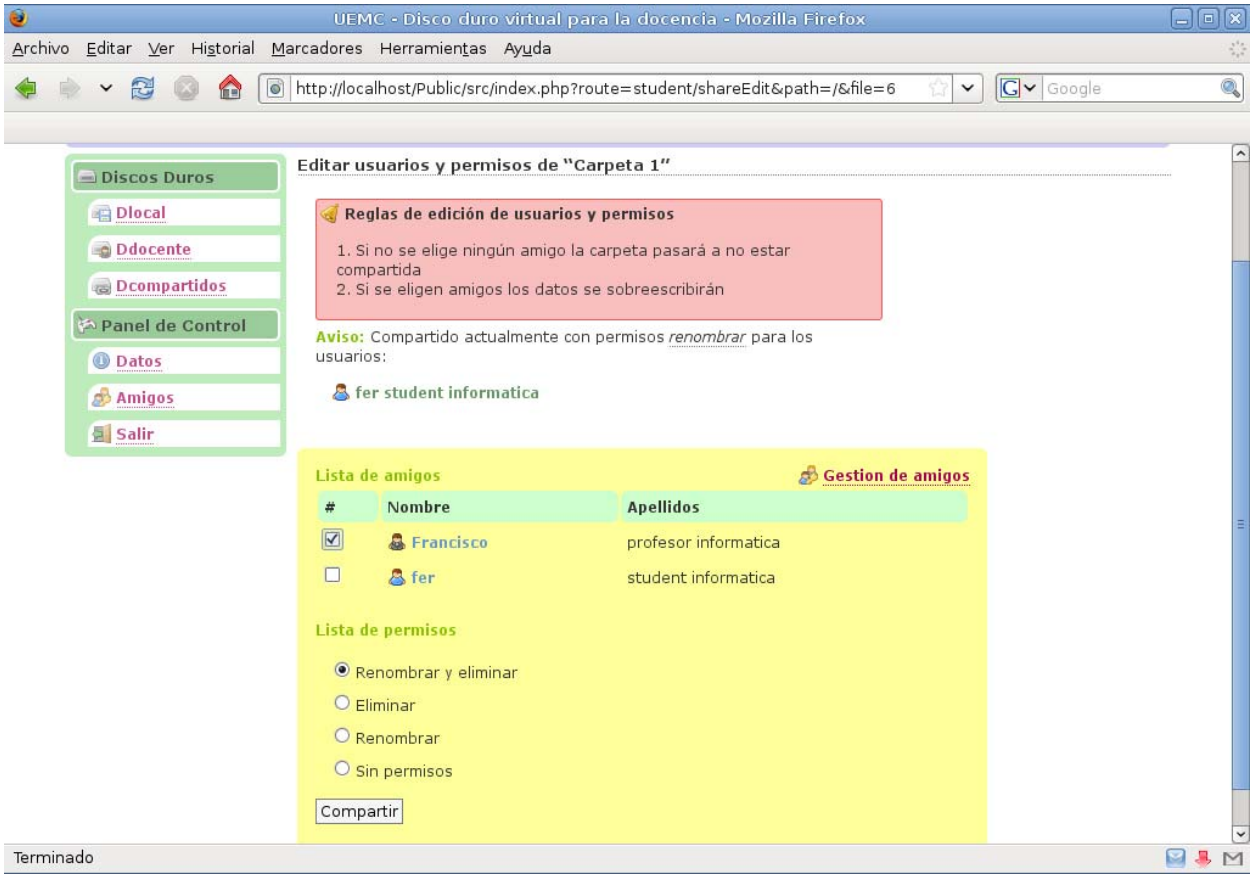


Figura 45. Editar permisos

Esta nueva pantalla se usará para editar los usuarios a los que se compartirá el elemento seleccionado.

En primer lugar aparecen unas reglas de uso para que el usuario pueda editar por sí mismo los permisos y los usuarios en caso de querer cambiarlos. Las reglas son:

- *Si no se elige ningún amigo, la carpeta pasará a no estar compartida*.

  Esto quiere decir que si no se marca ninguno de los amigos que aparecen o se añade alguno nuevo, significará que no se quiere compartir con nadie y el resto de usuarios a los que antes se compartía, dejarán de ver el elemento seleccionado y los que contenga en caso de ser una capeta.

- *Si se eligen amigos los datos se sobrescribirán*.

  En el supuesto en que se seleccionen otros usuarios distintos a los que ya

tenían permisos, no se mantendrán los anteriores y se añadirán los nuevos, si no que se sobrescribirán los usuarios. Se dejará de compartir a los que no se hayan marcado, aunque antes se les compartiera ese elemento. En definitiva que los usuarios que se marquen en esta pantalla serán los únicos que tengan permisos.

En segundo lugar se muestran los permisos así como los amigos con los que está compartido el elemento seleccionado en ese momento y que se pretende cambiar.

En tercer lugar aparecen los amigos que se han añadido con su nombre y apellidos y su casilla de verificación correspondiente para poder seleccionarle.

En cuarto lugar tenemos los permisos que se van a aplicar sobre el elemento elegido y que sobrescribirán los permisos que tuviera anteriormente.

Estos dos últimos ya han sido tratados anteriormente.

Una vez tengamos todas las opciones marcadas, pulsaremos en el botón *Compartir* que aplicará los nuevos permisos al elemento y nos redirigirá a la pantalla de disco local.

Todos los elementos que se suban o se creen sobre una carpeta compartida, adquieren los permisos que su contenedor posea.

Al pinchar en el apartado disco compartido que aparece como una de las opciones del menú lateral izquierdo, nos mostrará en la pantalla central los usuarios que tienen algún elemento compartido con nosotros. En este punto elegiremos el usuario del cual queremos ver qué nos comparte.

La pantalla que nos aparece a continuación tiene un aspecto muy parecido a la pantalla de disco local, pero únicamente veremos los elementos que el usuario elegido nos comparte. En la parte superior aparece el nombre del usuario elegido, mientras que en la parte inferior aparece un menú de navegación para que en todo momento sepamos la ruta jerárquica en la que estamos situados.

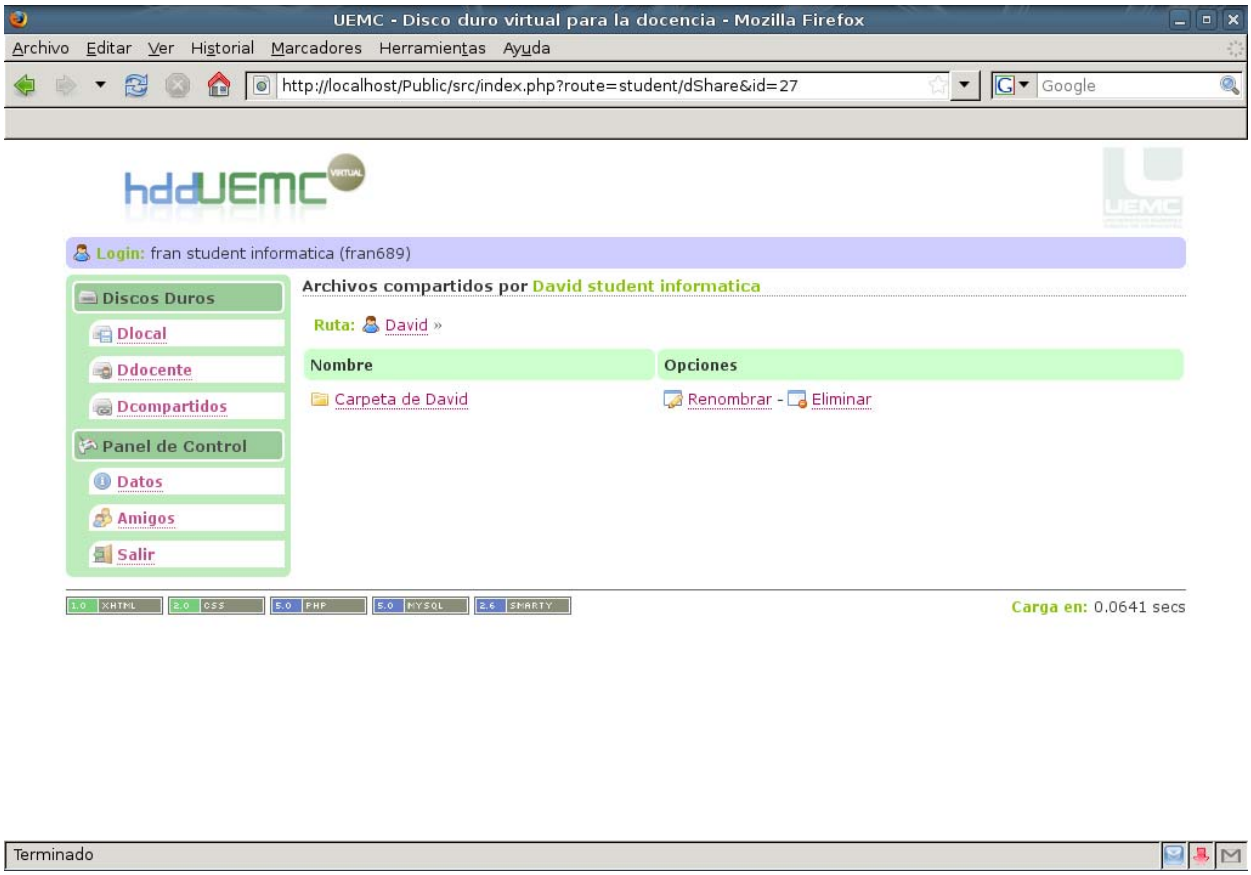


Figura 46. Disco compartido

En la parte central se mostrarán los elementos que se comparten ordenados alfabética y jerárquicamente junto con las opciones *Renombrar* y *Eliminar* las cuales estarán desactivadas o activadas respecto a los permisos que dicho usuario nos ha asignado.

Procedemos a mostrar el disco docente, que separaremos en dos partes dependiendo del usuario que sea. Las dos opciones son: **alumno** o un **profesor**. Esta distinción es importante porque de ella van a depender una serie de opciones para cada usuario.

1. Profesor

Cuando un profesor selecciona la opción de disco docente, en la pantalla central, se mostrarán las asignaturas que imparte ese profesor ordenadas por el curso al

que pertenecen. En esta pantalla se mostrarán los nombres de las asignaturas, así como una lista desplegable con todos los usuarios que están matriculados en la asignatura. También se muestra el curso y el nombre de la carrera al que pertenece cada asignatura.

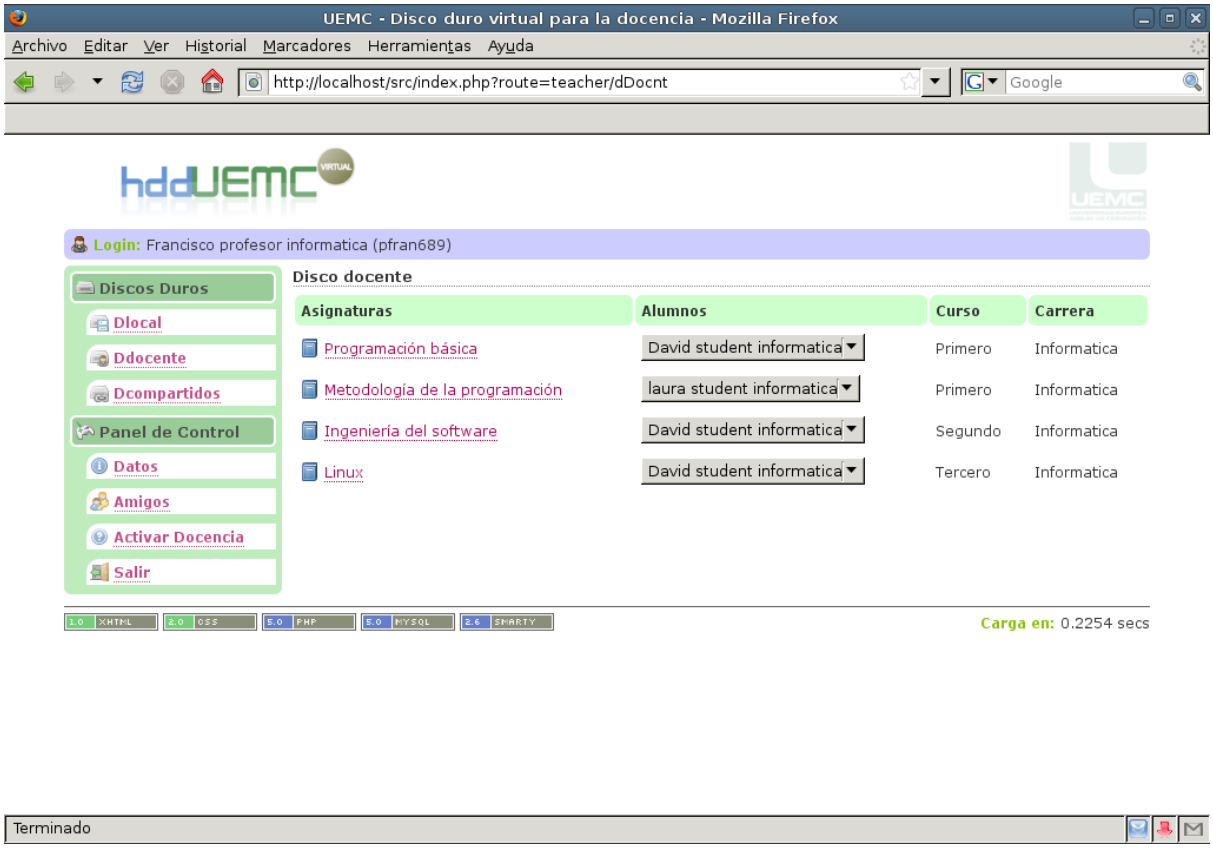


Figura 47. Disco docente del profesor

Si pinchamos en una signatura, la pantalla a la que nos redirige se divide en 2 partes bien diferenciables.

- La primera pertenece a los archivos que el profesor sube y que se muestran junto con su nombre, tamaño, fecha y las opciones *Renombrar* y *Eliminar.* Aquí el profesor colgará los enunciados de las prácticas u otros archivos para que los usuarios matriculados en esa asignatura puedan descargarse y utilizar. También se puede subir un archivo pinchando en el enlace *Subir enunciado*.

  Al pulsar sobre esta última opción, se mostrará la pantalla para poder subir

un archivo y que ya se ha comentado anteriormente.

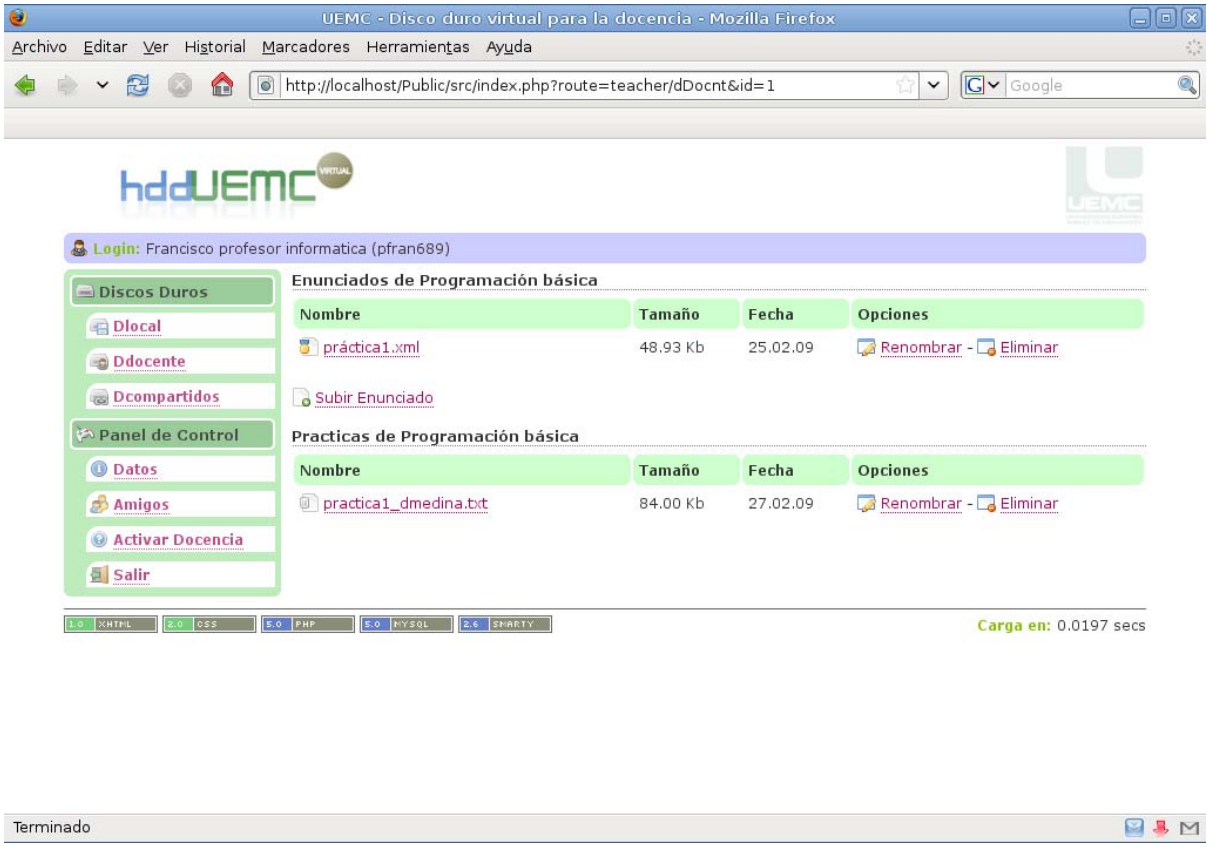


Figura 48. Disco docente del profesor de una asignatura

- En la segunda parte, que es la de abajo, se pueden observar las prácticas que suben los usuarios y que deberán llevar un formato que determinará el profesor porque no se muestra quién es el usuario que sube cada práctica, si no que deberá ir el nombre del usuario en el formato que determine el profesor a la hora de dar nombre al archivo a subir. A parte del nombre del archivo se muestra el tamaño, la fecha y las opciones *Renombrar* y *Eliminar.* En el momento en que el profesor pulse sobre cualquier práctica, ésta se descargará en el lugar asignado por el usuario.

2. Alumno

Al pulsar sobre la opción disco docente, en la pantalla central se mostrarán las asignaturas en las que el usuario está matriculado, ordenadas por el curso al que pertenecen. Se muestran las asignaturas junto con el nombre del profesor que la

imparte, el curso y la carrera a la que pertenece.

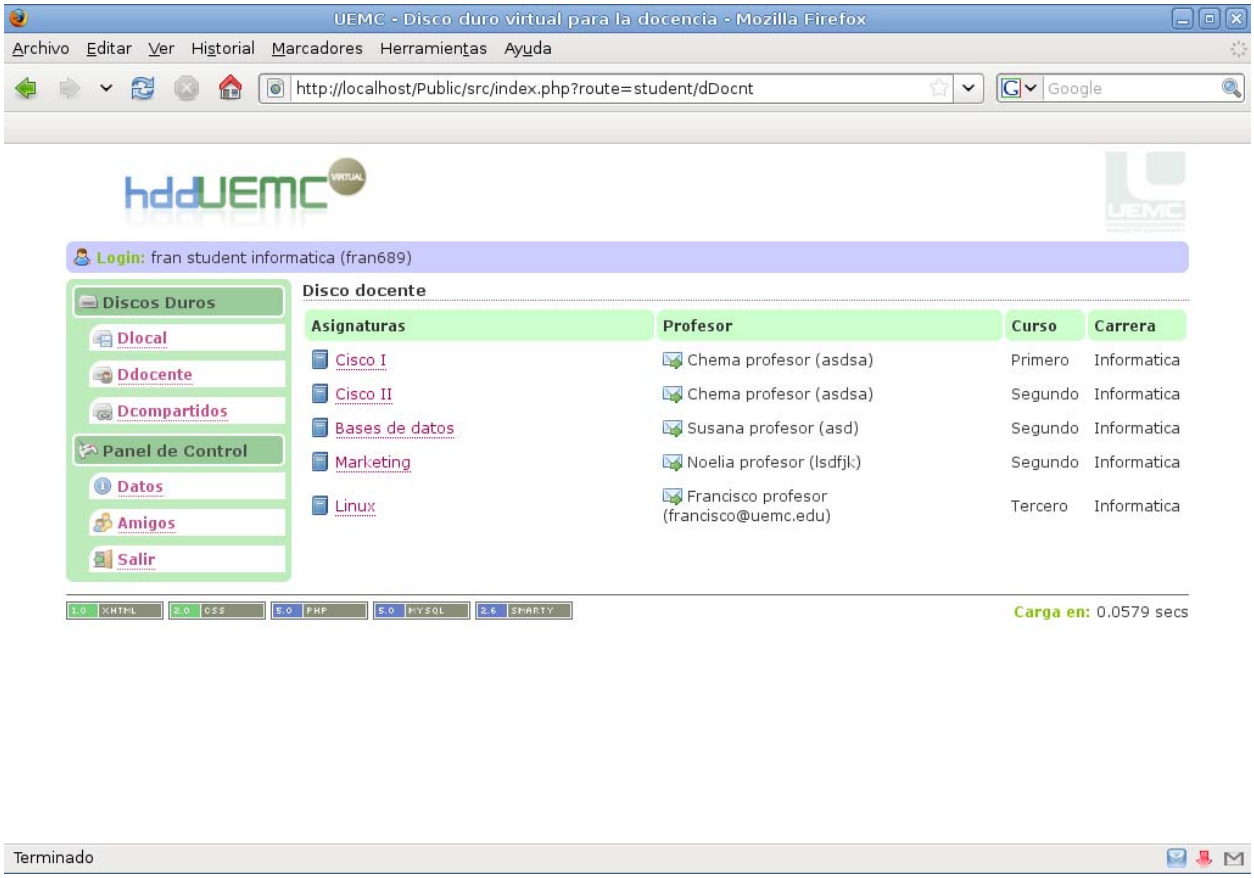


Figura 49. Disco docente del alumno

Si pinchamos en una asignatura, se mostrarán los enunciados y archivos que haya subido el profesor y que podremos descargarnos, pero en ningún caso tendremos permisos para borrarles o cambiarles el nombre.

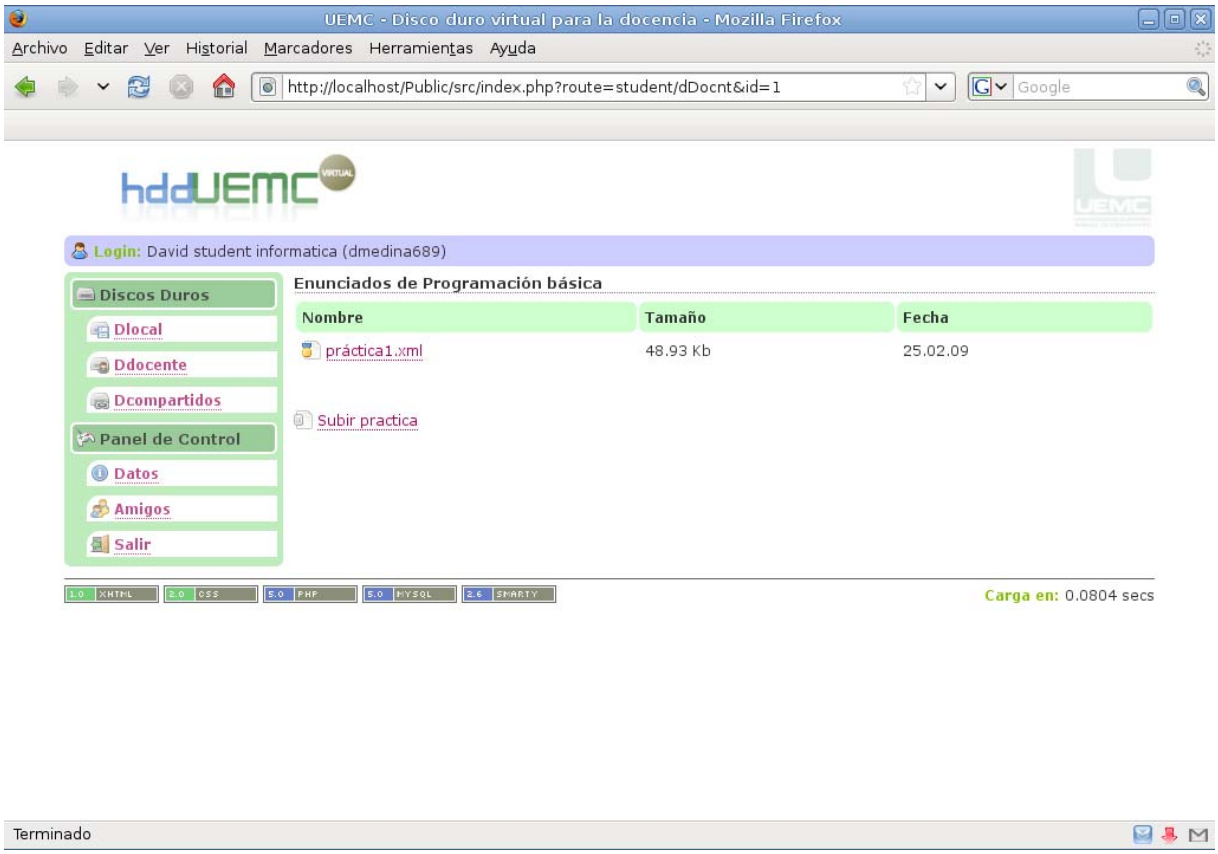


Figura 50. Disco docente del alumno de una asignatura

Una opción que sólo tiene el alumno, es la de subir práctica. Al pulsar sobre el enlace, se mostrará la pantalla para subir un archivo y que ya hemos comentado anteriormente. A la hora de subir una práctica se nos muestra una frase diciendo si se ha subido correctamente o por el contrario no se ha podido subir la práctica. Una vez subida el usuario ya no tendrá acceso a ese archivo, si no que ahora pertenece al profesor.

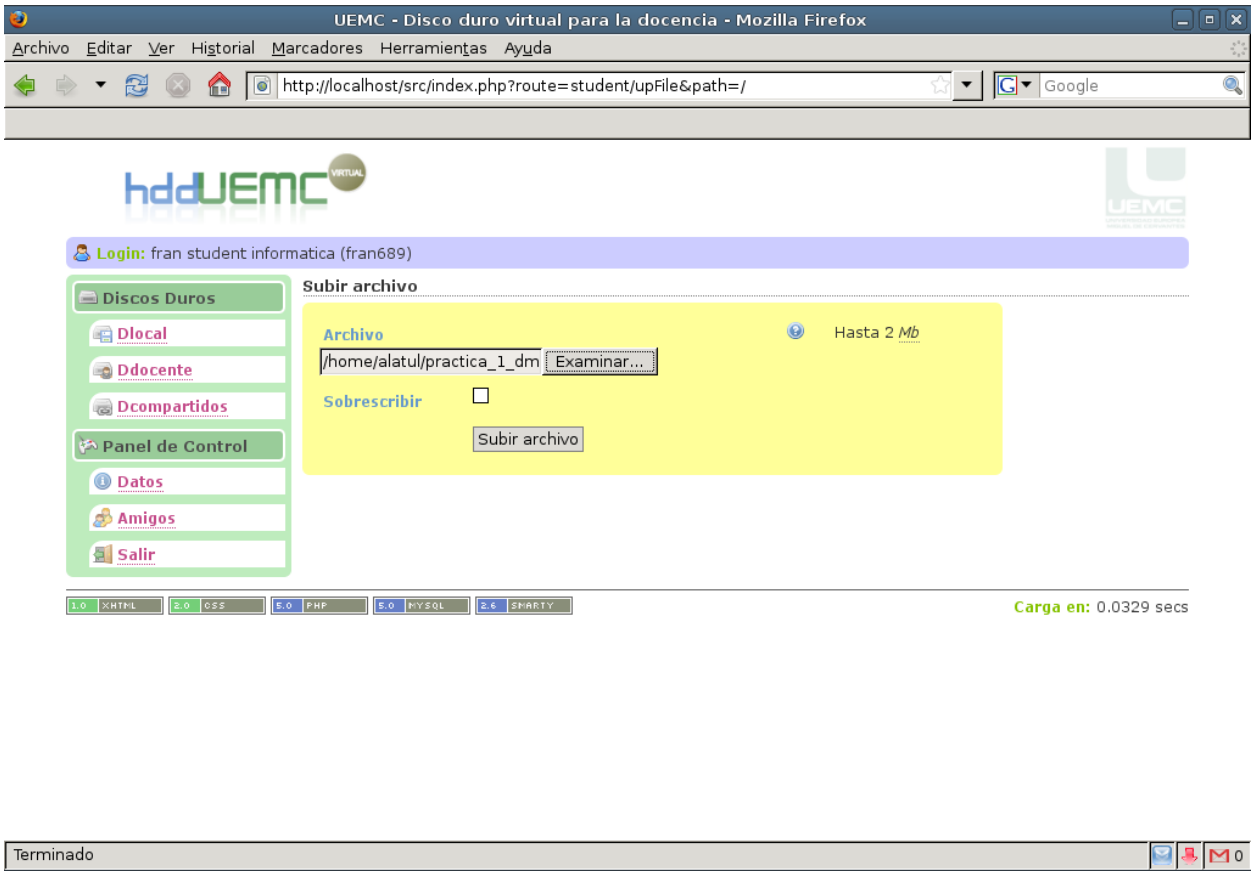


Figura 51. Subir práctica

Ahora se procederá a hablar de las opciones dentro del panel de control que se encuentran en el menú lateral izquierdo.

En primer lugar tenemos la opción ***Datos***, que si la pulsamos nos mostrará en el panel central los datos relativos al usuario que está conectado. Estos datos son:

Nombre, *Apellido1*, *Apellido2*, *Correo*, *Login* (nombre de usuario), *Despacho* (en caso de ser un profesor, si es un alumno esto no se mostraría) y la *Ruta* donde se almacenan las archivos del usuario.

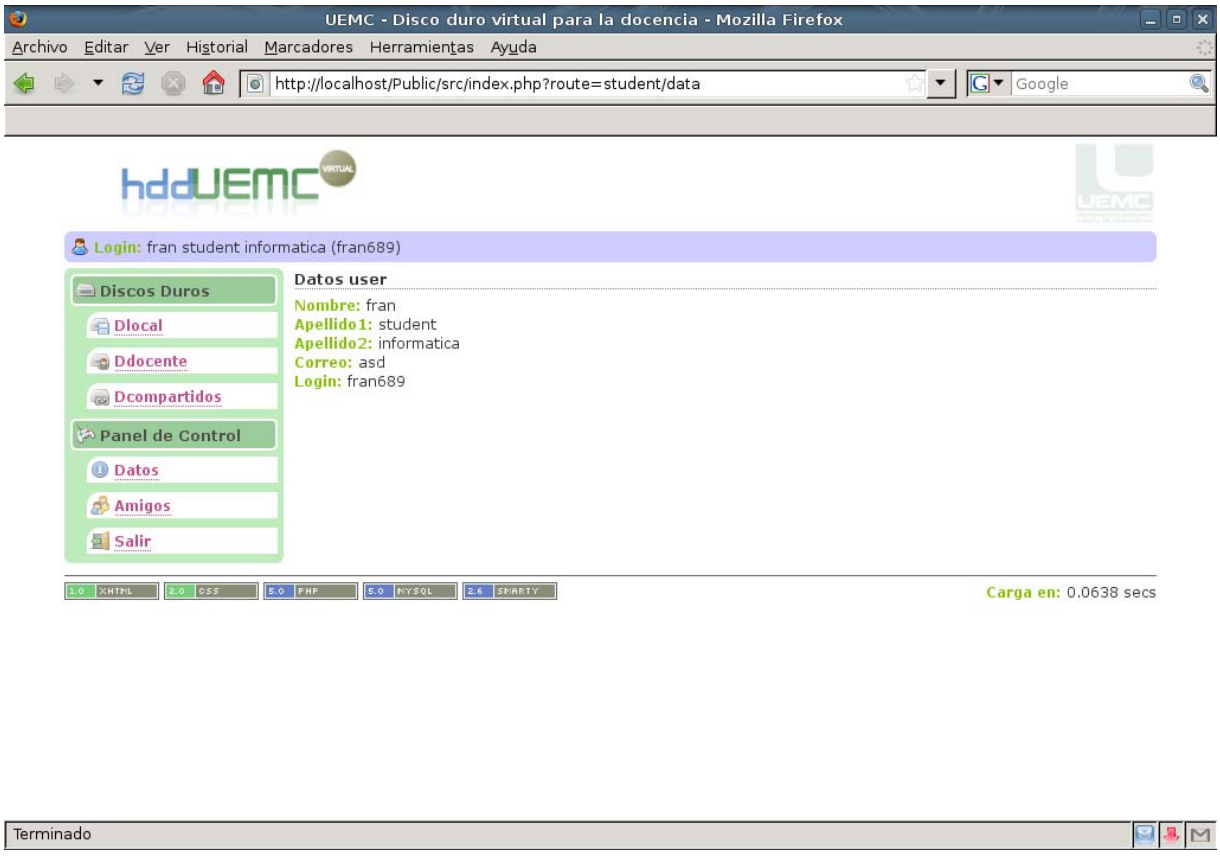


Figura 52. Datos

En segundo lugar vemos la opción **Amigos**, en la cual pulsamos y nos redirige a otra pantalla en la que en la parte central se muestran los amigos que tenemos como usuarios. Se muestra una casilla de verificación para cada usuario junto con el nombre del usuario y sus apellidos. En la parte de abajo tenemos dos opciones más *Añadir amigo* y *Eliminar amigos seleccionados*.

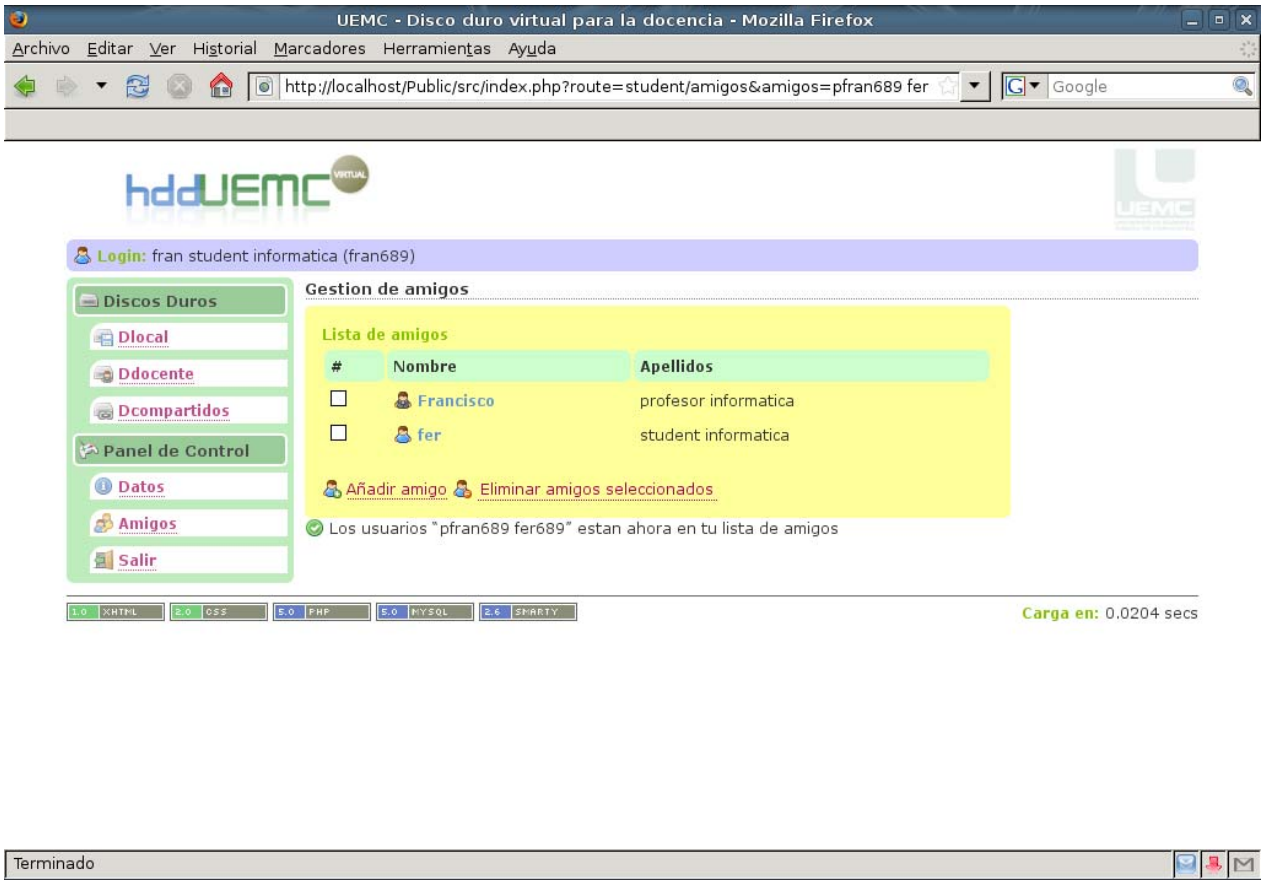


Figura 53. Pantalla amigos

Para esta última opción, se seleccionarán los usuarios que se quieren eliminar mediante las casillas de verificación y se pulsa sobre el enlace para eliminar.

Si por el contrario queremos añadir un amigo, pulsamos sobre la opción que nos redirige a otra página. En ella se muestra una nota para poder introducir a uno o varios amigos. Siguiendo estas instrucciones escribimos en el campo de texto el *login* (nombre de usuario) de todos los amigos que queramos añadir. Pulsamos en el botón *Añadir amigos*, que nos retornará a la pantalla anterior en la que se podrán visualizar todos los amigos junto con una nota en la que se muestran los usuarios que se acaban de añadir.

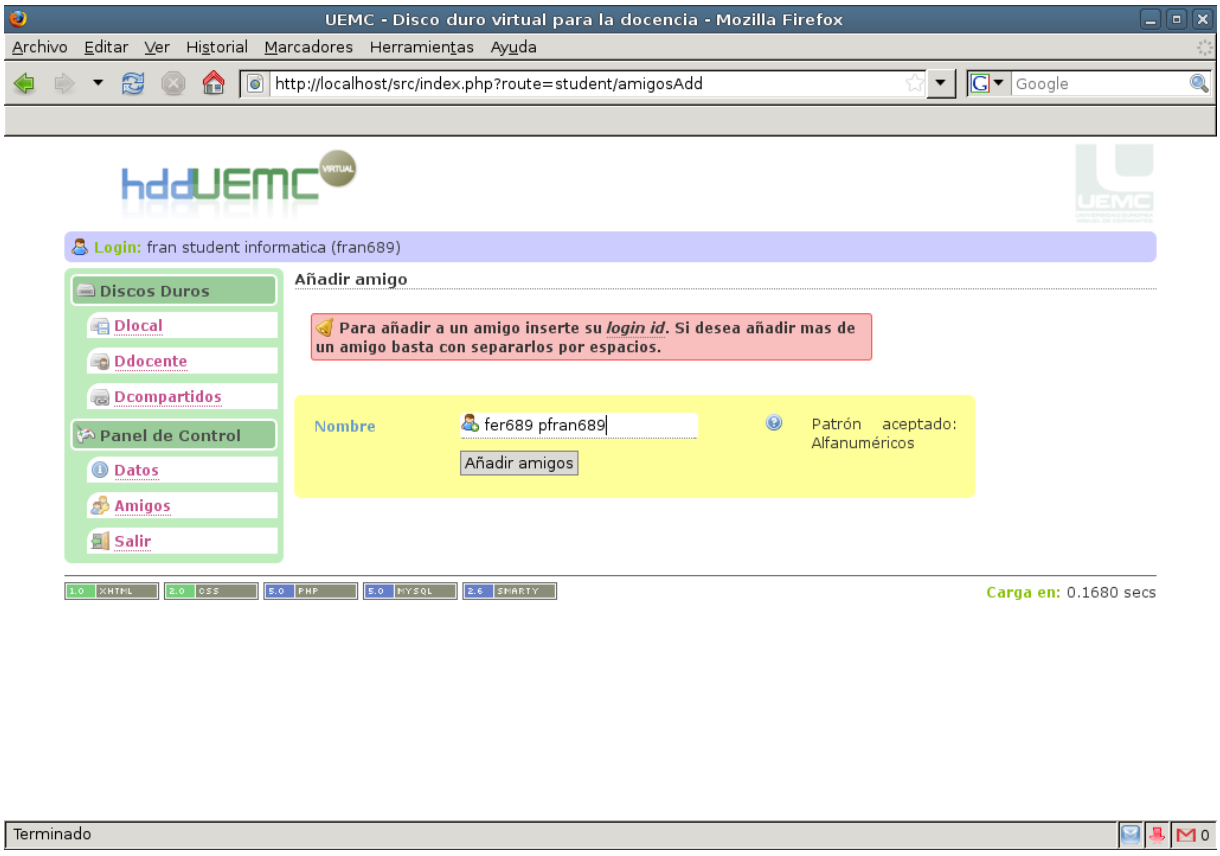


Figura 54. Añadir amigo

Como última opción tenemos **Salir**, que cierra la sesión del usuario y vuelve a la pantalla de bienvenida.

9.2. Bibliografía

Tim Converse and Joyce Park with Clark Morgan, 2004: *"PHP5 and MySQL Bible"*

Clare Churcher, 2008: *"Beginning SQL Queries: From Novice to Professional"*

Matt Zandstra, 2008: *"PHP Objects, Patterns, and Practice, Second Edition"*

Quentin Zervaas, 2008: *"Practical Web 2.0 Applications with PHP"*

W. Jason Gilmore, 2008: *"Beginning PHP and MySQL: From Novice to Professional, Third Edition"*

Perdita Stevens and Rob Pooley, 2002: *"Utilización de UML en Ingeniería de Software con Objetos y Componentes"*

Craig Larman, 2003: *"UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado, 2ª Edición"*

Dave W. Mercer, Allan Kent, Steven D. Nowicki, David Mercer, Dan Squier, Wankyu Choi, 2004: *"Fundamentos PHP5"*

Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, 2003: *"Patrones de diseño. Elementos de software orientado a objetos reutilizable"*

Michele E. Davis y Jon A. Phillips, 2007: *"Php y MySql. Creación de sitios Web dinámicos y con bases de datos"*

Juan Diego Gutiérrez Gallardo, 2004: *"Desarrollo Web con Php 5 y MySql"*

Adoración de Miguel y Mario Piattini, 1999: *"Fundamentos y modelos de Bases de Datos, 2ª Edición"*

Alan Beaulieu, 2005: *"Aprende SQL. Una traducción A SQL para desarrolladores y administradores de Bases de Datos"*

Gary W. Hansen and James V. Hansen, 1997: *"Diseño y administración de Bases de Datos, 2ª Edición"*

http://www.librosweb.es/css

http://www.librosweb.es/css_avanzado

http://www.librosweb.es/xhtml

9.3. Glosario de términos

MVC – "*Modelo Vista Controlador*"

XHTML – "*e**X**tensible **H**yper**t**ext **M**arkup **L**anguage*"

CSS – "*Cascading Style Sheets***"**

SMARTY – "Gestor de plantillas**"**

FEATURE – "Nueva funcionalidad software**"**

SPL – "*Standard PHP Library*"

PHP – "***P**HP **H**ypertext **P**re-processor*"

PDO – "*PHP Data Objects*"

ZEND – "Núcleo de PHP"

C – "Lenguaje de programación de bajo nivel"

C++ – "Lenguaje de programación de nivel medio"

API – "***A**pplication **P**rogramming **I**nterface*"

INTERFACE – "Capa de programación**"**

FOREACH – "Bucle específico en PHP**"**

FRAMEWORK – "Estructura software para desarrollar otro software**"**

SQL – "*Structured **Q**uery **L**anguage***"**

UML – "*Unified Modeling Language***"**

FTP – "*File Transfer Protocol***"**

POP – "*Post Office Protocol***"**

P2P – *"Peer to Peer***"**

GNU – "*GNU is Not Unix***"**

GPL – "*General Public License***"**

UNIX – "Sistema operativo multiusuario y multiproceso**"**

TCP / IP – *"Transmission Control Protocol" & "Internet Protocol"*

SOLARIS – "Sistema operativo de SUN**"**

HTTP – "*HyperText Transfer Protocol***"**

UTF – "*Unicode Transformation Format***"**

SSL – "*Secure Sockets Layer***"**

XML – "*Extensible Markup Language***"**

W3C – "*World Wide Web Consortium***"**

MINIX – "Sistema operativo 'predecesor' de Linux**"**

HURD – "Núcleo de GNU**"**

OPENSOURCE – "Comunidad de usuarios entorno al software libre**"**

KERNEL – "Núcleo de Linux**"**

PECL – "*PHP Extension Community Library***"**

PEAR – *"PHP Extension and Application Repository"*

PYME – "*Pequeña y Mediana Empresa***"**

CHM – "*Microsoft Compile Html***"**

PDF – "*Portable Document Software***"**