UNIVERSIDAD DE EXTREMADURA

CENTRO UNIVERSITARIO DE MÉRIDA

Grado en Ingeniería Informática en Tecnologías de la Información

Trabajo Fin de Grado

CUMDoc: Gestor Documental del Centro Universitario de Mérida integrado con Campus Virtual

Autor: Jesús Salguero Serrat

Mérida, Septiembre de 2017

UNIVERSIDAD DE EXTREMADURA

CENTRO UNIVERSITARIO DE MÉRIDA

Grado en Ingeniería Informática en Tecnologías de la Información

Trabajo Fin de Grado

CUMDoc: Gestor Documental del Centro Universitario de Mérida integrado con Campus Virtual

Autor: Jesús Salguero Serrat
Fdo.:

Director: Luis J. Arévalo Rosado
Fdo.:

Resumen

En la actualidad, el sistema de gestión documental utilizado en el Centro Universitario de Mérida está fragmentado. En el Campus Virtual, la documentación proporcionada por el equipo docente está vinculada a una asignatura y a un año académico, siendo visible únicamente para los estudiantes matriculados en ese curso. La documentación administrativa suele encontrarse en distintos rincones de la web de la Universidad de Extremadura, siendo en ocasiones difíciles de localizar para el público interesado. Tampoco existe un espacio donde los estudiantes puedan realizar sus propias aportaciones para que estas puedan servir de ayuda a otros.

Este proyecto tiene como objetivo el análisis, diseño e implementación de un sistema formado por una aplicación web para la gestión documental que permita la colaboración de todos los miembros de la comunidad universitaria, una plataforma ownCloud para el almacenamiento de archivos, una plataforma Moodle para simular el Campus Virtual y una base de datos LDAP para que todas las plataformas anteriores cuenten con un sistema de autentificación único. Como resultado, se obtiene un gestor documental integrado con el Campus Virtual, consiguiendo una organización más real, integral, efectiva y eficiente de la información documental del Centro Universitario de Mérida, y un repositorio que cuente con apuntes oficiales, aportaciones de estudiantes y documentación administrativa en el que todos los miembros de la comunidad universitaria tienen importancia.

En este proyecto, se han evaluado diferentes soluciones para el desarrollo del sistema, siendo las elegidas LDAP, ownCloud y Moodle. Tras ello, se investigó la manera de integrar ownCloud en Moodle y en la aplicación web, siendo WebDAV la solución obtenida. Para el desarrollo de la aplicación web se utilizó una arquitectura de tres capas, empleando Hibernate para la capa de persistencia, Spring para la lógica de negocio y Spring MVC para la capa de presentación. Además, la aplicación web es responsive, siendo accesible desde cualquier dispositivo gracias a Bootstrap.

Palabras clave: Gestor documental, Moodle, LDAP, ownCloud, Aplicación web

Abstract

Nowadays, the document management system used in the University Center of Mérida is fragmented. In the Virtual Campus, the documentation provided by the teaching team is linked to a concrete subject and academic year, being visible only to students enrolled in that course. The administrative documentation is often found in different corners of the University of Extremadura web site, being sometimes difficult to locate for the interested individuals. Furthermore, there is no space where students can make their own contributions so that they can help others.

This project aims to analyze, design and implement a system consisting of a web application for document management that allows the collaboration of all members of the university community, a ownCloud platform for file storage, a Moodle platform to simulate The Virtual Campus and an LDAP database so that all the previous platforms have a unique authentication system. As a result, a documentary manager integrated with the Virtual Campus is obtained, obtaining a more real, integral, effective and efficient organization of the documentary information of the University Center of Mérida, and a repository that counts on official notes, students contributions and administrative documentation In which all members of the university community are important.

In this project, different solutions for the development of the system have been evaluated, being LDAP, ownCloud and Moodle the chosen ones. After that, we investigated the way to integrate ownCloud in Moodle and in the web application, being WebDAV the solution obtained. For the development of the web application we used a three-layer architecture, using Hibernate for the persistence layer, Spring for business logic and Spring MVC for the presentation layer. In addition, the web application is responsive, being accessible from any device thanks to Bootstrap.

Keywords: Document manager, Moodle, LDAP, ownCloud, Web Application

Agradecimientos

En primer lugar, quiero dar las gracias a Luis Jesús Arévalo Rosado, director de este Trabajo Fin de Grado, por su apoyo, correcciones y consejos, pero sobre todo por sacar tiempo de donde no lo había. También quisiera agradecerle la ayuda ofrecida durante todos estos años, que desde el primer curso resolvía nuestras dudas y problemas.

Me gustaría dedicar este Trabajo Fin de Grado a mis padres, Ramón y Dolores, y a mi hermano Manuel por todo el esfuerzo que han realizado estos años que ha hecho posible el estudio de las titulaciones que estoy a punto de finalizar, y por todo el apoyo y ánimo que me han dado cada año.

También, me gustaría dedicárselo a Noelia, por las palabras de apoyo que me ha regalado cada día y por darme la fuerza y motivación necesaria para lograr superar cada reto que aparecía en el camino. Gracias por confiar en mí.

Me gustaría agradecer también a mis compañeros con los que he compartido estos años, en especial a Adrián, por acompañarme estos cinco años y por la ayuda que siempre nos hemos ofrecido.

También quisiera darle las gracias a mis compañeros del Campus Virtual por ofrecerme su ayuda e intentar resolverme las cuestiones que fui planteando respecto a este proyecto.

Por último, me gustaría agradecer al Centro Universitario de Mérida, tanto el equipo docente como al personal que allí trabaja, toda la ayuda brindada durante estos años, y toda la formación que me han impartido con gran profesionalidad y cercanía.

Muchas gracias a todos.

Índice general

Índice de figuras

Índice de tablas

Algoritmos

Capítulo 1

Introducción

Contenidos

Este primer capítulo servirá para introducir este Trabajo Fin de Grado. Para ello, se abordarán los siguientes aspectos: introducción, antecedentes y objetivos que se pretende alcanzar.

En la sección 1.1 se explica la importancia de la gestión documental en el marco académico y, especialmente, en el universitario.

En la sección 1.2 se describe como los distintos miembros de la comunidad universitaria gestionan actualmente sus documentos.

En la sección 1.3 se detallan los objetivos de este trabajo fin de grado.

1.1. Introducción

Actualmente, los centros educativos cuentan con un gran número de documentos. Podemos encontrarnos desde apuntes hasta formularios administrativos. Una de las dimensiones más importantes y con mayor potencial que las nuevas tecnologías abren en la comunidad educativa es la gestión eficiente de sus documentos.

Los estudiantes utilizan principalmente los documentos proporcionados por los docentes de las distintas asignaturas para el estudio y el aprendizaje. Además, generan su propia documentación, como pueden ser resúmenes, apuntes o ejercicios resueltos.

Los profesores necesitan poder gestionar de forma eficaz la documentación de las asignaturas que imparten. No es suficiente subirla simplemente a un servidor, debe hacerse disponible selectivamente y de la forma adecuada a los estudiantes inscritos, y necesitan contar con mecanismos que permitan editarla y actualizarla.

Los centros cuentan además, con distintos documentos administrativos que deben ser correctamente gestionados para facilitar que miembros de la comunidad educativa puedan acceder a ellos con facilidad.

Una gestión documental eficaz es, en definitiva, fundamental para el correcto funcionamiento de la actividad académica.

1.2. Antecedentes

En la actualidad, el sistema de gestión documental utilizado en el Centro Universitario de Mérida, así como en la Universidad de Extremadura, está fragmentado y en él están ausentes funcionalidades que podrían mejorar su efectividad.

Por un lado, para la mayoría de los documentos académicos se utiliza el Campus Virtual, basado en la plataforma Moodle. Los apuntes, asociados a una única asignatura, son subidos por el equipo docente y los estudiantes pueden disponer de ellos accediendo al aula virtual destinado para dicha asignatura. Esto significa que unos apuntes que pueden servir para varias asignaturas a la vez no pueden estar disponibles en todas ellas a menos que se suban manualmente a cada una, ocupando innecesariamente un espacio adicional. Además, al finalizar el año académico, los estudiantes pueden perder el acceso a las distintas aulas virtuales, perdiendo así los distintos documentos que no hubiese guardado anteriormente.

Los documentos no académicos suelen encontrarse en distintos rincones de la web de la Universidad, siendo en ocasiones difíciles de localizar para el público interesado.

Tampoco existe un espacio donde los estudiantes puedan compartir sus apuntes para que estos puedan servir de ayuda a otros. El único lugar donde los estudiantes pueden hacer contribuciones es el Repositorio Institucional (Dehesa) en el cual pueden publicar sus Trabajos Fin de Estudios. Aún así, es necesario que dichos trabajos obtengan la calificación mínima exigida y sean propuestos a la inclusión en el repositorio por parte del presidente del tribunal de evaluación.

La fragmentación existente dificulta la exploración y la consulta de la documentación disponible. Por ello, Manuel Botellero Gutiérrez creó en su Trabajo Fin de Grado [1] un acercamiento al desarrollo de un portal único para la consulta y gestión de los documentos disponibles. Desarrolló una aplicación web basada en la plataforma Alfresco para conseguir una organización integral, efectiva y eficiente de la información documental del Centro Universitario de Mérida. Esta solución fue elaborada de manera independiente a las plataformas ya existentes en la Universidad de Extremadura, por lo que no sacaba partido a los recursos ya disponibles, ni trabajaba en un entorno real. Además, el sistema de administración de contenidos Alfresco únicamente ofrece una versión gratuita para desarrolladores, siendo más limitada que las distintas versiones de pago existentes. Este será el punto de partida de este Trabajo Fin de Grado.

1.3. Objetivos

El objetivo principal de este Trabajo Fin de Grado consiste en mejorar el sistema de gestión documental del Centro Universitario de Mérida, realizando una modificación y extensión de la solución aportada por Manuel Botellero Gutiérrez. Para ello, se desarrollará una aplicación web para la gestión documental, basada en un software libre que almacene los distintos archivos de los usuarios, y que pueda abarcar desde el ámbito académico hasta el administrativo. Esta aplicación permitirá compartir documentos con la comunidad académica y acceder a ellos de forma organizada.

Por otra parte, se buscará una sincronización entre los documentos existentes en esta plataforma y el Campus Virtual de la Universidad de Extremadura, para que así el profesorado pueda enlazar sus documentos a esta plataforma docente.

Finalmente, ambas plataformas necesitarán un sistema de autentificación único para per-

mitir el acceso y diferenciar los distintos roles que pueden adquirir los usuarios (profesorado, estudiante...). Para ello, se integrará una misma base de datos en las distintas plataformas, que simule a la utilizada actualmente por el Campus Virtual.

El objetivo general de este trabajo se alcanzará a través de la consecución de los siguientes objetivos específicos:

- Analizar y estudiar las distintas opciones disponibles para simular la base de datos del Campus Virtual.
- Analizar y estudiar las distintas plataformas que permitan almacenar los documentos de los usuarios.
- Analizar y estudiar la plataforma educativa Moodle.
- Realizar la base de datos elegida para almacenar a los usuarios del sistema.
- Realizar una plataforma que aloje los documentos de los usuarios con el software elegido.
- Realizar una plataforma educativa Moodle, simulando el Campus Virtual de la Universidad de Extremadura.
- Integrar ambas plataformas, permitiendo que el equipo docente pueda enlazar los documentos existentes en sus aulas virtuales. Además, ambas plataformas deben estar integradas con la base de datos elegida para que exista un sistema de autentificación único.
- Realizar una aplicación web para la gestión documental basada en el software escogido anteriormente.

Tras esto, obtendremos un gestor documental integrado con la base de datos y las plataformas ya existentes en la Universidad de Extremadura, consiguiendo una organización más real integral, efectiva y eficiente de la información documental del Centro Universitario de Mérida.

Capítulo 2

Análisis Previo

Contenidos

En este capítulo se realizará una introducción a una serie de conceptos previos necesarios para el desarrollo de este Trabajo Fin de Grado. Además, se realizará un análisis previo del sistema a desarrollar, detallando cada una de las partes necesarias para su implementación.

2.1. Conceptos previos

2.1.1. Base de datos

Una base de datos [2] es un conjunto de datos estructurados y relacionados que se encuentran almacenados con el objetivo de facilitar su posterior utilización.

Las bases de datos pueden ser locales, siendo utilizadas en un solo equipo por un usuario, o pueden ser distribuidas, almacenando la información en equipos remotos, accediendo a ella a través de una red.

La administración de las bases de datos se realiza con un Sistema de Gestión de Bases de Datos (SGBD), también llamado DBMS por sus siglas en inglés (Database Management System). El SGBD es un conjunto de servicios que permite a los usuarios un fácil acceso a la información y proporciona las herramientas necesarias para su manipulación. Un SGBD puede dividirse en tres subsistemas:

- **Sistema de administración de archivos:** Su función es almacenar la información en un medio físico.
- **SGBD interno:** Se encarga de ubicar la información en orden.
- **SGBD externo:** Representa la interfaz de usuario.

2.1.1.1. Ventajas e inconvenientes

La utilización de bases de datos ofrecen una serie de ventajas:

- **Independencia:** Los datos son independientes de las aplicaciones que los usan.
- **Disponibilidad:** Se facilita el acceso a los datos desde contextos, aplicaciones y medios distintos. Además, permite compartir la información de forma simultánea con otros usuarios o con otras bases de datos.
- **Seguridad:** Es más fácil replicar una base de datos para una copia de seguridad que hacerlo con un conjunto de ficheros almacenados de forma no estructurada. Además, se puede restringir el acceso a los usuarios, impidiendo que accedan a determinados elementos.
- **Menor redundancia:** No es necesario repetir los datos, solamente se debe indicar la manera en la que se relacionan.
- **Eficiencia:** Se facilita el acceso a los datos, haciendo más sencilla su explotación.
- **Mayor facilidad y sencillez de acceso:** El usuario dispone de herramientas y de una estructura solida para el acceso y uso de los datos.
- **Mayor valor informativo:** Resulta más sencillo extraer la información que los datos contienen.

No obstante, también presenta una serie de inconvenientes:

- **Espacio:** Una base de datos suele requerir mucho espacio en disco y a medida que se van introduciendo nuevos datos se volverán más pesadas.
- **Mantenimiento:** Su mantenimiento y manipulación es compleja, necesitando una formación especializada por parte de la persona encargada.
- **Lentitud:** Cuando una base de datos crece mucho puede volverse más lenta.

2.1.2. Gestor documental

Un gestor documental [3] es una herramienta software que permite almacenar, administrar y controlar el flujo de documentos dentro de una organización. Por lo tanto, se trata de una forma de organizar los distintos documentos en una localización centralizada a la que los distintos usuarios de la organización puedan acceder de forma fácil y sencilla.

Generalmente, los sistemas de gestión documental tienen la capacidad de mantener un registro de versiones de los documentos, pudiendo consultar las modificaciones que ha realizado cada usuario y si es oportuno restablecer el documento. Además, permiten digitalizar documentos que se encuentren en papel, con ayuda de un escáner, almacenando la copia en el gestor como un documento más.

Según el estudio realizado por Aberdeen Group[1], las organizaciones que han implantado un sistema de gestión documental han logrado reducir costes en un 10 %, los errores en un 30 % y han mejorado la velocidad de creación de contenidos en un 49 %.

[1]Estudio obtenido en https://www.ticportal.es/temas/sistema-gestion-documental/que-es-sistema-gestion-documental

2.1.2.1. Beneficios

Los beneficios obtenidos a la hora de implantar un gestor documental son los siguientes:

- **Mayor productividad:** Se mejora la eficacia y la eficiencia en la búsqueda y recuperación de documentos.
- **Ahorro de costes:** Normalmente se reducen los espacios donde se almacena la información, ya que se encuentra centralizada en el gestor. Además, se ahorra en las copias y almacenaje de los documentos en papel.
- **Ahorro de tiempo:** Al tener la documentación en un solo repositorio, con las mismas reglas a la hora de gestionar los documentos, es más fácil la creación, el mantenimiento y la recuperación de documentos.
- **Normativas de seguridad:** Trabajar con un gestor documental ya trae implícito el cumplimiento de las normativas de seguridad, como por ejemplo la LOPD a través del acceso controlado al repositorio. Además, muchas asociaciones han publicado listas de normas de control de documentos que se pueden aplicar en el gestor documental.
- **Reducción del riesgo:** Un buen sistema de gestión documental presenta medidas para evitar el riesgo de perder documentos.

2.1.3. Almacenamiento en la nube

El almacenamiento en la nube [4] consiste en almacenar, administrar y respaldar de forma remota los archivos de un sistema. Estos archivos normalmente se encuentran en servidores que están en la nube y que son administrador por un proveedor del servicio. Para que los usuarios accedan a estos archivos es necesario que se conecten a la red donde se encuentren los servidores, como pueden ser Internet.

2.1.3.1. Tipos de almacenamiento en la nube

Al hablar de almacenamiento en la nube, se suelen distinguir tres tipos de servicios:

- **Público:** Se requiere poco control administrativo y puede acceder cualquier persona que esté autorizada mediante Internet. El almacenamiento en la nube pública utiliza un mismo conjunto de hardware para almacenar la información de varios usuarios, con medidas de seguridad y espacios virtuales para que cada usuario pueda ver únicamente su información. Desde el punto de vista del usuario, no se desperdician recursos ya que se paga por lo que se utiliza, y si es necesario aumentar el espacio de almacenamiento únicamente deberá pagar más.
- **Privado:** Una nube privada pertenece a una sola organización, ofreciéndose un autoservicio con una arquitectura propietaria. Esta opción permite controlar los recursos informáticos necesarios e implementar mejores medidas de seguridad, ya que el alojamiento está limitado a un número de usuarios. Aunque si se emplea esta solución, la organización es la responsable de administrar la nube privada.

- **Híbrido:** Los sistemas de almacenamiento en nubes híbridas ofrecen una combinación entre las nubes públicas y privadas. Esta opción permite que la carga de trabajo se muevan entre las nubes públicas y privadas a medida que cambian las necesidades. Un ejemplo podría ser una organización que almacene los datos más importantes en una nube privada, mientras que los datos menos importantes se puedan almacenar en una nube pública con mejor acceso disponible.

2.1.3.2. Ventajas e inconvenientes

El almacenamiento en la nube presenta una serie de ventajas:

- Disponibilidad inmediata para poder ver, revisar y corregir información desde cualquier lugar y dispositivo con acceso.
- Independencia a la hora de no tener que preocuparse en guardar los datos a medida que se van actualizando.
- Se ahorra bastante dinero si se utilizan este servicio.
- Se permite compartir archivos fácilmente.
- Normalmente los servidores cuentan con varias copias de los datos para evitar pérdidas.

No obstante, también presenta una serie de inconvenientes:

- Los datos pueden estar en manos de terceros en cualquier parte del mundo, ya que se confía el almacenamiento a un proveedor de este servicio del que no se dispone información de donde se localizan sus servidores en el caso de las nubes públicas. Además, las leyes de cada país son distintas respecto al tratamiento de los datos.
- No se dispone de control de acceso total a los datos para su eliminación, ya que las nubes públicas suelen realizar copias y no hay total garantía de su completa eliminación.

2.1.4. Plataforma educativa virtual

Una plataforma educativa virtual [5] se compone de diferentes herramientas con fines docentes cuya principal función es facilitar la creación de entornos virtuales para impartir formación a través de Internet. Este tipo de plataformas contribuyen a la evolución de los procesos de enseñanza y aprendizaje, tanto como sustituto o como complemento a la docencia presencial.

Las herramientas por las que debe estar compuesta una plataforma educativa son las siguientes:

- LMS (Learning Management System): Punto de contacto entre los usuarios de la plataforma. Se encarga de presentar los cursos a los usuarios y del seguimiento de la actividad y progresos del estudiante.
- LCMS (Learning Content Management System): Esta herramienta permite la gestión y la publicación de contenidos en el curso.

- Herramientas de comunicación: Estas herramientas permiten la comunicación entre los distintos usuarios. Normalmente se realizan a través de chats, foros, correos electrónicos, intercambio de ficheros, entre otros.
- Herramientas de administración: Permite gestionar las inscripciones en los cursos y se encarga de la gestión de los roles y permisos dentro de la plataforma.

Actualmente, la mayoría de las universidades cuenta con plataformas educativas para sus usuarios. En la Universidad de Extremadura se encuentra el Campus Virtual de la UEx (CVUEx), una plataforma educativa que permite a los profesores impartir asignaturas a distancia de manera virtual o complementar las clases presenciales que imparten en la Universidad.

2.1.5. Aplicación web

Una aplicación web [6] es una herramienta que los usuarios pueden utilizar accediendo a un servidor web a través de una red mediante un navegador. Se trata de una aplicación software que se codifica en un lenguaje soportado por los navegadores web.

Este tipo de aplicaciones no necesitan ser instaladas en un dispositivo para ser utilizadas, son independientes del sistema operativo y son más fáciles de mantener y actualizar.

Una página web puede contener elementos que permitan la comunicación activa entre el usuario y la información. Esto permite que el usuario acceda a la información de manera interactiva, según las acciones que realice, como pueden ser rellenar y enviar formularios, acceder a una información específica que se encuentre en una base de datos, entre otras.

2.1.5.1. Ventajas e inconvenientes

El uso de aplicaciones web cuenta con una serie de ventajas:

- No es necesario descargar ni instalar ningún programa para su uso, basta con un navegador web.
- Como la aplicación web la gestiona el desarrollador el usuario no tendrá que actualizar la aplicación, sino que al acceder siempre contará con la última versión publicada.
- Se puede usar desde cualquier sistema operativo y si la aplicación web es responsive también se podrá usar desde cualquier dispositivo.
- La disponibilidad de la aplicación web suele ser alta ya que el servicio se suele ofrecer desde múltiples localizaciones para asegurar la continuidad si algún nodo cae.
- Permite la colaboración entre usuarios, ya que lo que publica un usuario llega a los demás, pudiendo compartir datos de manera sencilla.
- Los navegadores ofrecen cada vez más y mejores funcionalidades para crear aplicaciones web.

No obstante, también presenta una serie de inconvenientes:

- Normalmente ofrecen menos funcionalidades que las aplicaciones de escritorio debido a que las funciones que se pueden realizar desde un navegador son más limitadas.
- La disponibilidad depende de un tercero, tanto del servidor donde se aloja la aplicación como del proveedor de la conexión a Internet.

2.2. Sistema a desarrollar

El objetivo de este Trabajo Fin de Grado consiste en desarrollar un gestor documental para el Centro Universitario de Mérida. Este gestor documental deberá convivir con otra serie de plataformas ya existentes en la Universidad de Extremadura como es el Campus Virtual. Por lo tanto, para este proyecto se necesitará realizar:

- Una base de datos para almacenar la información referente a los usuarios que utilizarán la aplicación. Se almacenará información personal del usuario, como podrá ser su nombre, apellidos o correo electrónico, entre otros. Además, deberá almacenar el perfil que tiene cada usuario, identificar los grupos a los que pertenece y guardar su usuario y contraseña. Esta base de datos deberá permitir que el resto de plataformas y aplicaciones del sistema consulten estos datos, para ofrecer un sistema de autentificación único y permitir así que los usuarios se conecten a todas las plataformas y aplicaciones con las mismas credenciales. También podrán consultar los grupos a los que pertenece el usuario, para limitar sus funciones según las características de cada grupo.

- Un gestor documental o un servidor de almacenamiento en la nube que permita guardar los archivos de los distintos usuarios. Esta plataforma deberá poder ofrecer a los usuarios un modo de compartir los documentos existentes con los otros usuarios o con los distintos grupos del sistema. Además, deberá permitir las funciones básicas de gestión (crear, consultar, modificar y eliminar) y limitarlas según el perfil del usuario.

- Una plataforma educativa similar al Campus Virtual de la Universidad de Extremadura. Esta plataforma educativa permitirá a los profesores gestionar sus aulas virtuales donde impartan sus asignaturas. En sus aulas podrán interaccionar con los estudiantes a través del material educativo que suban, de tareas, cuestionarios y todas las herramientas que incluya la plataforma educativa. Respecto al gestor documental deberá permitir la conexión con él para obtener los distintos archivos que tenga almacenado el profesor en su cuenta y enlazarlos en sus aulas virtuales.

- Una aplicación web que facilite a los usuarios el acceso al gestor documental, mostrando los archivos de forma ordenada y simple. Además del acceso deberá ofrecer las acciones básicas de creación, consulta, modificación y eliminación. Esta aplicación web deberá ser responsive, es decir, adaptable al dispositivo con el que se acceda. También deberá ofrecer acceso sin autentificación para invitados. Así podrán consultar los documentos que sean públicos para todos los usuarios, actuando como un repositorio.

En la figura 2.1, se puede observar un esquema del sistema a desarrollar en este Trabajo Fin de Grado. Es importante destacar que los usuarios podrán acceder con cualquier tipo de dispositivo y a cualquiera de las plataformas y aplicaciones del sistema.

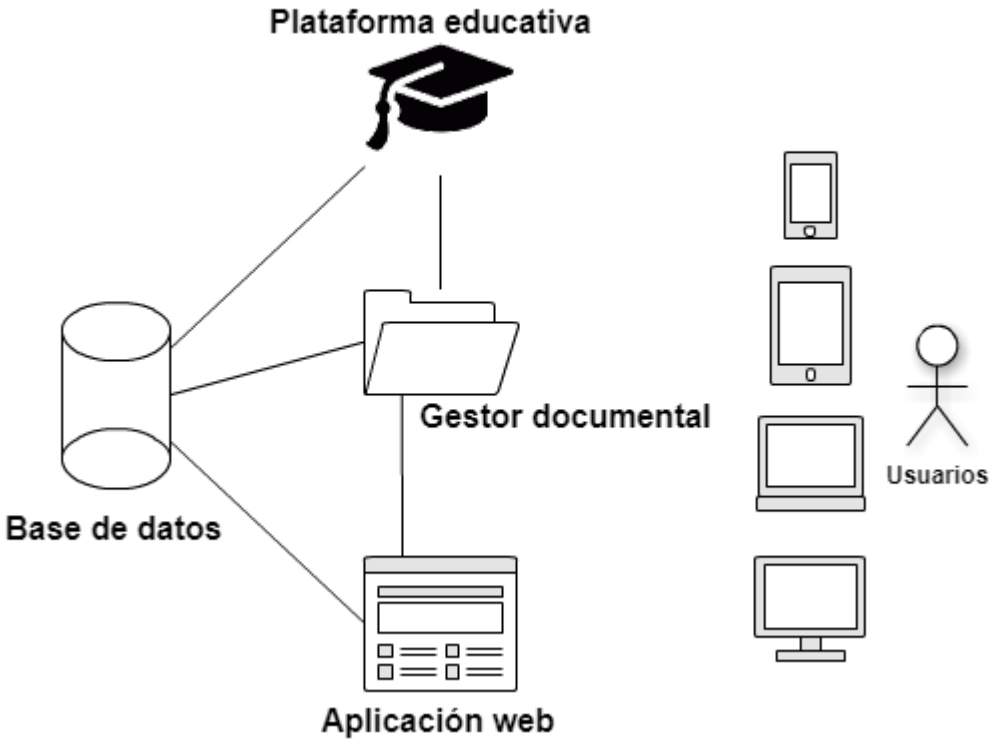

Figura 2.1: Esquema del sistema a desarrollar.

Capítulo 3

Metodología

Contenidos

En este capítulo se describe la metodología utilizada en este proyecto. Para la estructuración, planificación, y control del proyecto se aplicará un proceso de desarrollo software iterativo e incremental, y para el desarrollo del proyecto se utilizará una metodología orientada a objetos.

En la sección 3.1 se introducirá el concepto de desarrollo software iterativo e incremental.

En la sección 3.2 se realizará una breve explicación de la metodología orientada a objetos.

3.1. Desarrollo software iterativo e incremental

El desarrollo software iterativo e incremental permite controlar y gestionar los riesgos de forma eficiente y periódica, minimizando el número de errores y mejorando la calidad del proyecto.

3.1.1. Introducción

El desarrollo software iterativo e incremental [7] divide el proyecto en diferentes bloques temporales llamados iteraciones. En todas las iteraciones se repite un proceso de trabajo similar para proporcionar un resultado completo, obteniendo una versión funcional de la aplicación. De esta forma, el sistema se desarrolla poco a poco y obtiene una retroalimentación continua por parte del cliente, en este caso el director del Trabajo Fin de Grado.

En esta forma de desarrollo software, cada requisito se debe completar en una única iteración, debiendo realizar el equipo de trabajo todas las tareas necesarias para completarlo, incluyendo pruebas y documentación. En la figura 3.1[1], se puede observa el modelo de ciclo de vida iterativo e incremental.

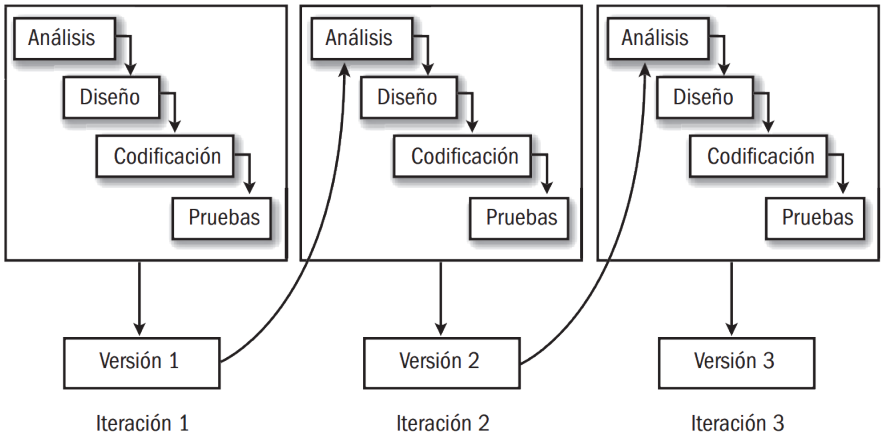

Figura 3.1: Modelo de ciclo de vida iterativo e incremental.

En cada iteración el equipo de trabajo va evolucionando el producto a partir de los resultados completados en las iteraciones anteriores, añadiendo nuevos objetivos o mejorando los que ya fueron completados.

3.1.2. Etapas del desarrollo software iterativo e incremental

El desarrollo software iterativo e incremental, aunque debe adaptarse a las características del proyecto, existen una serie de fases que se deben tener en cuenta a la hora de implementarla [8]:

1. **Requisitos:** En esta fase se identifica los objetivos centrales y específicos que persigue el proyecto.

2. **Definición de las tareas y las iteraciones:** Teniendo en cuenta el objetivo, el siguiente paso es hacer una lista de tareas y agruparlas en iteraciones que tendrá el proyecto.

3. **Diseño de los incrementos:** Una vez establecidas las iteraciones, es necesario definir cuál será la evolución del producto en cada una de ellas. Cada iteración debe generar un producto más evolucionado respecto a la anterior.

4. **Desarrollo del incremento:** En esta fase se realizan las tareas previstas y se desarrollan los incrementos establecidos en la etapa anterior. Para su desarrollo se utilizará una metodología orientada a objetos.

5. **Validación de los incrementos:** Al finalizar cada iteración, se debe validar el incremento realizado. Si los resultados no son los esperados es necesario volver al principio del incremento y buscar las causas de esto.

[1] Imagen obtenida de http://sings-ufps.blogspot.com.es/search/label/Ingenier%C3%ADa%20de%20Software

6. **Integración de incrementos:** Una vez que son validados, los incrementos dan forma a la denominada línea incremental o evolución del proyecto en conjunto. Cada incremento ha contribuido al resultado final.
7. **Entrega del producto:** Una vez que el producto en su conjunto ha sido validado, se procede a su entrega final al cliente.

3.1.3. Ventajas y desventajas

Utilizar el desarrollo software iterativo e incremental en el proyecto tiene una serie de beneficios[7]:

- Se puede gestionar las expectativas del cliente de manera regular, pudiendo tomar decisiones en cada iteración.
- No es necesario conocer al principio del proyecto todos los requisitos, únicamente será necesario conocer con detalle los requisitos de las primeras iteraciones y los requisitos de alto nivel que se irán refinando en las diferentes iteraciones.
- El cliente puede obtener resultados importantes y usables desde las primeras iteraciones.
- Se pueden gestionar de manera natural los cambios que van apareciendo durante el proyecto.
- Se pueden identificar riesgos desde el inicio del proyecto, debiendo gestionar los problemas desde la primera iteración.
- En este modelo se minimiza el número de errores que se producen durante el desarrollo y aumenta la calidad del proyecto.

Aunque este desarrollo software presenta también una serie de restricciones[7]:

- El cliente debe estar disponible durante el proyecto participando de manera continua.
- Se necesita una relación más cercana al cliente, basándose en los principios de colaboración y ganar/ganar.
- Cada resultado de una iteración debe ser funcional.
- Es necesario tener conocimientos, herramientas y experiencia en este modelo para poder realizar cambios fácilmente.

3.2. Metodología orientada a objetos

El desarrollo de este proyecto se realizará con una metodología orientada a objetos[9]. A continuación, se describe en que consiste y cómo se van a realizar cada una de las fases.

3.2.1. Introducción

La metodología orientada a objetos pretende ayudar a los desarrolladores software a explotar el poder de los lenguajes de programación orientados a objetos, utilizando las clases y objetos como bloques de construcción básicos. En la figura 3.2, se puede observar un esquema que plasma cómo esta organizada esta metodología.

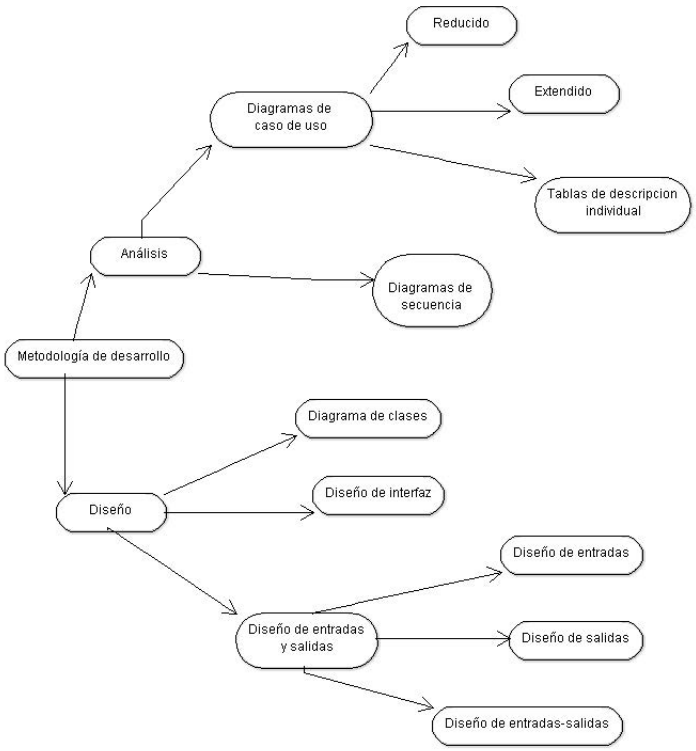

Figura 3.2: Esquema de la metodología orientada a objetos.

3.2.2. Análisis

El objetivo de este proceso es analizar el problema, llegando hasta una especificación completa y minuciosa del comportamiento del mismo, y consistencia, en la que se especifican las características funcionales y operacionales del mismo [10].

Este proceso debe dar información de las necesidades del sistema, cuáles van a ser las funciones del sistema sin introducir información sobre su implementación, y mostrar una visión completa del sistema y de lo que este debe hacer.

El fin del análisis orientado a objetos es desarrollar una serie de modelos que describan el software para satisfacer un conjunto de requerimientos definidos por el usuario.

3.2.2.1. Diagrama de casos de uso

La función del diagrama de casos de uso [10, 11] es documentar el comportamiento de un sistema desde el punto de vista del usuario. Por lo tanto, los casos de uso determinan los requisitos funcionales del sistema, representando las funciones que el sistema puede realizar.

Su principal ventaja es la facilidad que tienen para ser interpretados, logrando que la comunicación con el cliente sea más fácil.

En el próximo capítulo, en este apartado, se mostrará los casos de uso del sistema de forma reducida, después de forma extendida, y tras ello se describirán individualmente mediante tablas.

Los elementos básicos de los casos de uso de forma reducida son:

- **Actores:** Representan un tipo de usuario del sistema. Un usuario es cualquier persona, animal o cosa externa que interactúa con el sistema. Se representa mediante la figura de un muñeco.

- **Casos de uso:** Son las acciones o tareas que se realizan tras una orden de uno o varios actores en el sistema con un objetivo determinado.

- **Asociaciones:** Son las relaciones entre los actores y los casos de uso cuando el usuario interactúa con el sistema.

La tabla 3.1 representa el formato que siguen las tablas que describen los casos de uso específicos.

Código del caso de uso	Código al que corresponde el caso de uso de la lista
Nombre del caso de uso	Nombre completo del caso de uso
Descripción	Descripción del caso de uso
Actores	Actores que realizan la acción
Precondiciones	Condiciones que deben cumplirse para poder realizar la acción.
Flujo normal	Etapas que se realizan para el desarrollo de la función.
Flujo alternativo	Camino alternativo para realizar la función
Postcondiciones	Resultado que debe cumplirse tras la ejecución de la función.

Tabla 3.1: Formato de las tablas de los casos de uso específicos.

3.2.3. Diseño de la interfaz de usuario

La interfaz de usuario es lo que une al sistema informático con el usuario. Define como interactúan usuario y ordenador, jugando un papel importante en la aceptación del sistema.

En esta sección, se describirán los distintos diseños de cada pantalla que forman la interfaz de usuario. Estos diseños deberán seguir un formato que las haga sencillas, accesibles, atractivas y cómodas para el usuario.

3.2.4. Diseño de entradas y salidas

En este apartado se describen las entradas, salidas y entradas-salidas al sistema.

3.2.4.1. Diseño de entradas

En este apartado se describen las distintas pantallas que el usuario se encontrará cuando va a introducir datos en el sistema. Deben ser claras, sencillas y fáciles de entender.

3.2.4.2. Diseño de salidas

En esta sección se describirán las distintas pantallas que visualizará el usuario cuando se realizan confirmaciones.

3.2.4.3. Diseño de entradas-salidas

Algunas partes de la aplicación contienen pantallas que aportan datos del sistema al usuario y al mismo tiempo permiten introducir nuevos datos al mismo.

Capítulo 4

Análisis del sistema

Contenidos

En este capítulo se detalla el análisis del sistema propuesto, exponiendo de forma pormenorizada su comportamiento, consistencia y viabilidad, así como sus características funcionales y operacionales.

En la sección 4.1 se puede observar la planificación del sistema, analizando la viabilidad del proyecto y mostrando la planificación temporal estimada.

En la sección 4.2 se presenta el análisis funcional del sistema analizando en profundidad cada una de las operaciones que ofrece.

En la sección 4.3 se describen los diagramas de casos de uso.

4.1. Planificación del sistema

Los objetivos principales de este Trabajo Fin de Grado se plasmaron de forma general en el tercer apartado del capítulo de introducción.

En este apartado, se analizará la viabilidad del sistema propuesto así como las operaciones que se desean desarrollar. Para ello, se identificará las actividades y operaciones que el sistema deberá ofrecer, los diferentes actores que las realizarán, las condiciones que deben darse y los parámetros de entrada y salida, entre otros.

Como se vio anteriormente, el sistema que se desarrollará es el siguiente:

- **CUMDoc:** Gestor documental integrado con la base de datos y plataformas ya existentes en la Universidad de Extremadura. Contará con una aplicación web para la gestión documental, que facilite a los distintos miembros de la comunidad universitaria el acceso a la información, destacando la posible colaboración entre estudiantes, de forma que puedan compartir sus apuntes y otros documentos generados por ellos mismos.

El objetivo del estudio de viabilidad es examinar y definir de forma clara las características del sistema, de forma que la solución propuesta resuelva las necesidades identificadas, además de ajustarse a las restricciones económicas, operativas y de tiempo.

4.1.1. Viabilidad técnica

La viabilidad técnica [12] determina si es posible realizar el proyecto con la tecnología que existe actualmente, y si es conveniente llevarlo a cabo.

En este apartado, se estudiará la viabilidad técnica del sistema de gestión documental CUMDoc. Este se compondrá de servidores que alojarán todos los archivos existentes en el sistema mediante un software de almacenamiento de archivos, la base de datos con los distintos usuarios y grupos del centro, la plataforma Moodle del Campus Virtual de la Universidad de Extremadura y la aplicación web diseñada para el acceso al contenido de manera estructurada. Los usuarios podrán acceder al sistema desde cualquier dispositivo con capacidad de navegación web y hacer uso de las distintas funcionalidades presentes en el sistema.

Tras su estudio, se puede concluir que el sistema es viable, ya que las diferentes tecnologías que permiten la realización de lo previamente descrito existen y están siendo utilizadas desde hace años.

4.1.2. Viabilidad operacional

La viabilidad operacional [12] determina si el sistema funcionará y será utilizado una vez que se finalice su desarrollo, es decir, si la solución proporcionada satisfará a los usuarios y a sus necesidades. En este apartado, es importante escuchar con atención lo que realmente demandan los usuarios.

Los usuarios a los que está destinado el sistema son los distintos miembros de la comunidad universitaria, así como las personas interesadas en obtener documentación sobre el centro o sus titulaciones. Actualmente, no existe un gestor documental similar al que se desarrollará en este proyecto, la información se encuentra dividida y es complicado acceder a ella en algunas ocasiones. Por lo tanto, el sistema propuesto daría respuesta a las necesidades de los usuarios de encontrar de manera estructurada los distintos documentos del centro, así como la posibilidad de compartir los suyos propios.

Para obtener la información necesaria sobre las necesidades de los usuarios y la estructura que debe tener el sistema se tiene como objetivo trabajar con representantes de los distintos grupos del centro, para que vayan describiendo cuales serán los requerimientos del sistema.

Una vez finalizado el desarrollo del sistema, cualquier usuario podrá acceder a él desde cualquier dispositivo que tenga acceso a Internet.

4.1.3. Viabilidad económica

La viabilidad económica [12] determina si la solución aportada en este proyecto es factible económicamente. Para ello, se debe realizar un presupuesto de coste de los recursos técnicos,

humanos y materiales, tanto para el desarrollo como para la implementación del sistema.

Los recursos de software empleados para la realización de este proyecto son libres y no tienen coste económico, por lo que este se limitará al servidor dedicado para alojar el sistema y a la mano de obra empleada.

El coste de un servidor físico dedicado se estima en la tabla 4.1, y el de desarrollo del proyecto en la tabla 4.2. El coste del desarrollo indicado se corresponde con el desarrollo completo del sistema para su puesta en marcha, no con el desarrollo llevado a cabo dentro de las limitaciones de un Trabajo Fin de Grado.

PRESUPUESTO SERVIDOR	
Concepto	**Precio**
Adquisición	1.000€
Coste de mantenimiento durante un año	500€
TOTAL	**1.500€**

Tabla 4.1: Análisis del coste del servidor.

	Horas	**Salario/Hora**	**Precio**
Análisis del proyecto	100	50€	5.000€
Diseño del proyecto	200	47€	9.400€
Desarrollo del sistema	350	45€	15.750€
Pruebas	80	47€	3.760€
Documentación	100	47€	4.700€
TOTAL	**830**		**38.610€**

Tabla 4.2: Análisis del coste de desarrollo del proyecto.

4.1.4. Viabilidad de la solución

A partir de los análisis anteriores, se puede concluir que la realización del proyecto es viable técnicamente, operacionalmente y económicamente. Los requerimientos de los usuarios se cumplen, económicamente es asumible y técnicamente hace uso de una tecnología asentada, con bastante popularidad y documentación.

4.1.5. Planificación estimada

En la figura 4.1, se podrá observar la planificación estimada del proyecto. Para ello se ha dividido el proyecto en una serie de iteraciones y se ha estimado el periodo de tiempo en el que se espera completarlas.

Como se comentó en el capítulo anterior, para la estructuración, planificación y control del proyecto se aplicará un proceso software iterativo e incremental. Las iteraciones planificadas para el proyecto son las siguientes:

- **Iteración 1:** Esta compuesta por la planificación inicial del proyecto.
- **Iteración 2:** En esta iteración se realizará la base de datos que almacenará los usuarios y grupos del sistema.

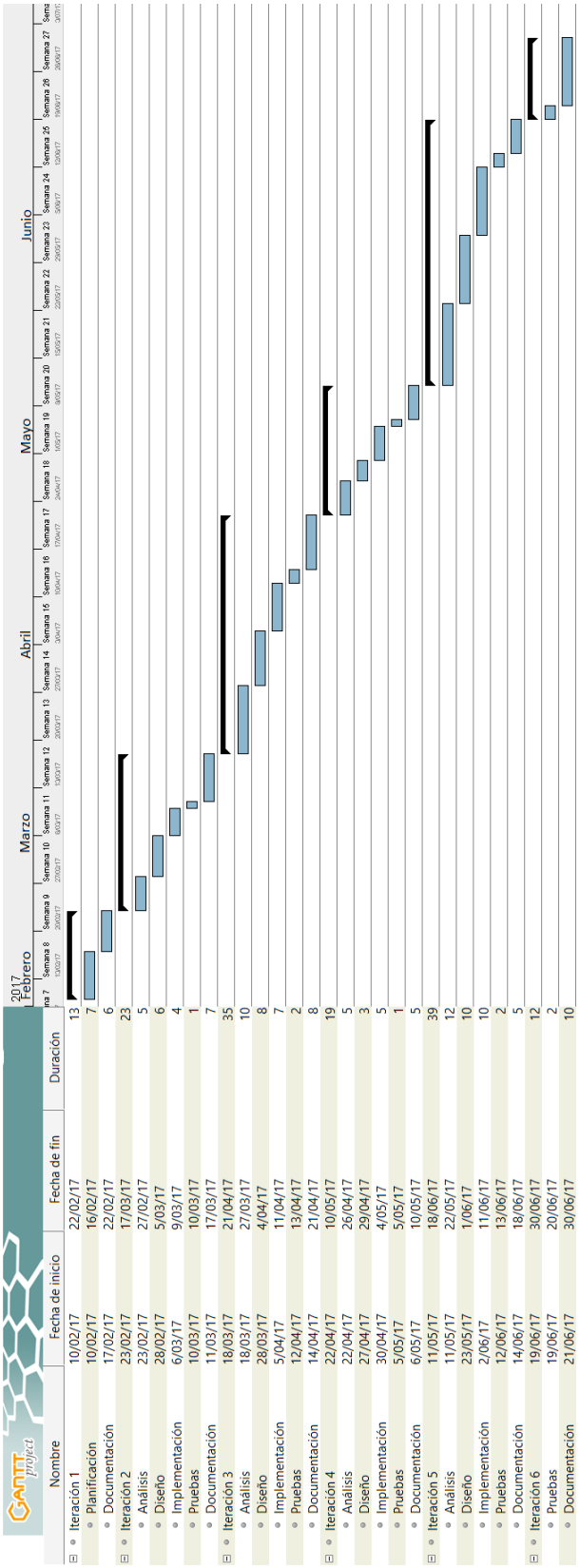

Figura 4.1: Diagrama de Gantt. Planificación estimada.

- **Iteración 3:** En esta iteración se llevará a cabo la parte del proyecto referente al software de alojamiento de archivos.
- **Iteración 4:** En esta iteración se realizará la parte referente a la plataforma educativa Moodle.
- **Iteración 5:** En esta iteración se desarrollará la aplicación web del proyecto.
- **Iteración 6:** En esta última iteración se realizará una serie de comprobaciones finales para asegurar el correcto funcionamiento del sistema.

En todas las iteraciones se realizará un proceso de documentación, en el cual se recogerá los distintos avances del proyecto.

4.2. Análisis funcional

En esta sección se realiza un estudio de las funcionalidades del sistema, utilizando las distintas metodologías expuestas en los puntos del capítulo anterior. Se observa la necesidad de cinco actores:

- **Administrador:** Encargado de la gestión y mantenimiento del sistema. Posee todos los permisos.
- **Personal Docente y de Investigación (PDI):** Encargado de la gestión de una asignatura o un área de conocimiento.
- **Personal de Administración y Servicios (PAS):** Encargado de la gestión administrativa.
- **Estudiante:** Puede gestionar sus propios documentos, así como consultar los demás.
- **Visitante:** Tiene permisos básicos, pudiendo consultar los distintos documentos.

4.2.1. Casos de uso globales

En la figura 4.2 se muestran los casos de uso del actor Administrador. Las operaciones que puede realizar son las siguientes:

- Crear usuario
- Modificar usuario
- Borrar usuario
- Crear grupo
- Borrar grupo
- Modificar grupo
- Añadir usuario a un grupo
- Eliminar usuario de un grupo

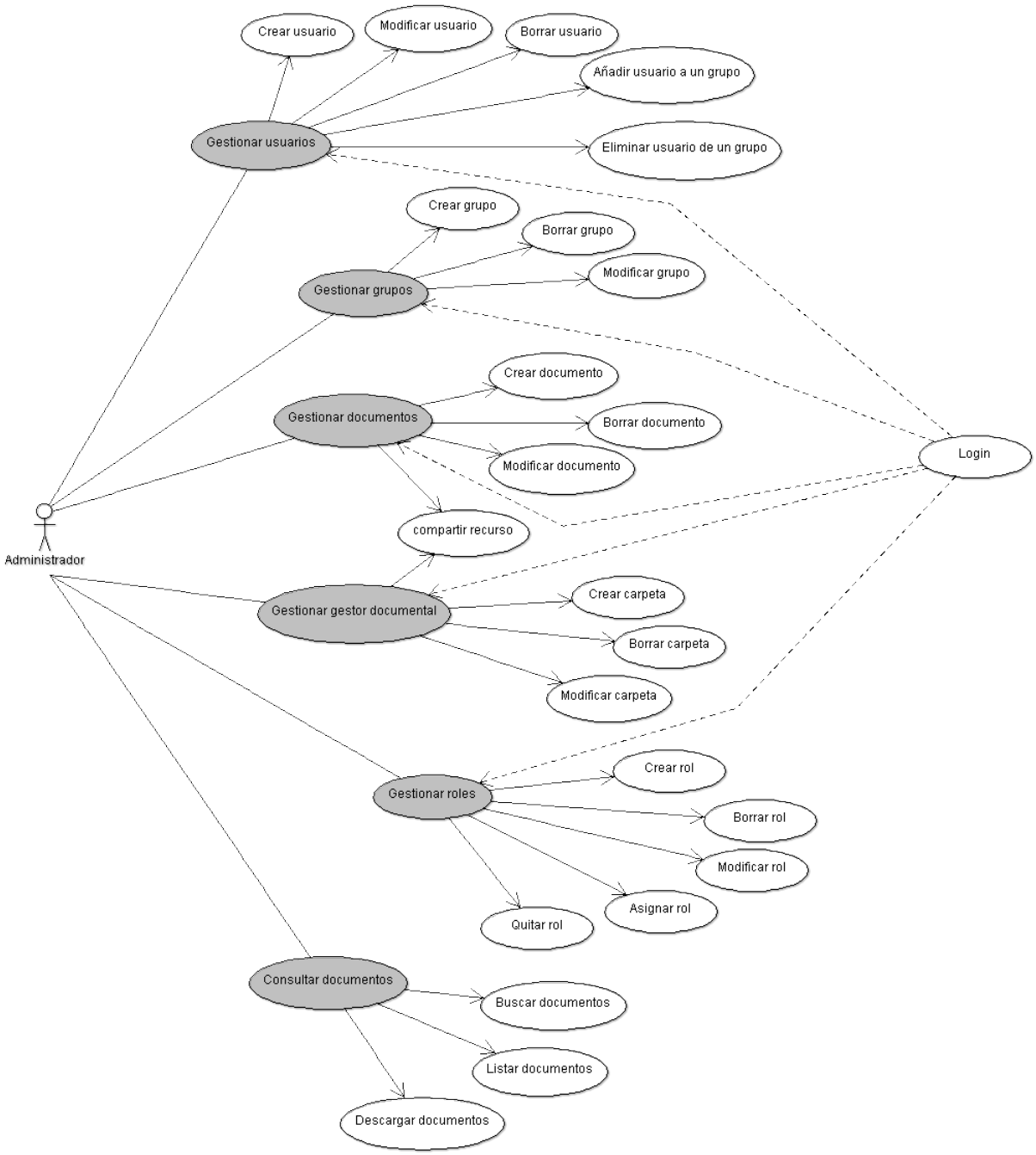

Figura 4.2: Casos de uso del actor Administrador

- Crear documento
- Borrar documento
- Modificar documento
- Crear carpeta
- Borrar carpeta
- Modificar carpeta
- Compartir carpeta o documento
- Asignar rol
- Modificar rol
- Quitar rol
- Buscar documentos
- Listar documentos
- Descargar documentos

Todas las operaciones, salvo las relacionadas con la consulta de documentos, necesita iniciar sesión con la cuenta correspondiente.

En la figura 4.3 se muestran los casos de uso del actor PDI. Las operaciones que puede realizar son las siguientes:

- Crear documento
- Borrar documento
- Modificar documento
- Enlazar documentos a su aula/espacio virtual
- Buscar documentos
- Listar documentos
- Descargar documentos
- Crear carpeta
- Borrar carpeta
- Modificar carpeta
- Compartir carpeta o documento

Todas las operaciones, salvo las relacionadas con la consulta de documentos, necesita iniciar sesión con la cuenta correspondiente.

En la figura 4.4 se muestran los casos de uso del actor PAS. Las operaciones que puede realizar son las siguientes:

Figura 4.3: Casos de uso del actor PDI

- Crear documento
- Borrar documento
- Modificar documento
- Enlazar documentos a su espacio virtual
- Buscar documentos
- Listar documentos
- Descargar documentos
- Crear carpeta
- Borrar carpeta
- Modificar carpeta
- Compartir carpeta o documento

Figura 4.4: Casos de uso del actor PAS

Todas las operaciones, salvo las relacionadas con la consulta de documentos, necesita iniciar sesión con la cuenta correspondiente.

En la figura 4.5 se muestran los casos de uso del actor Estudiante. Las operaciones que puede realizar son las siguientes:

- Crear documento
- Borrar documento
- Modificar documento
- Listar documentos
- Descargar documentos

Todas las operaciones, salvo las relacionadas con la consulta de documentos, necesita iniciar sesión con la cuenta correspondiente.

En la figura 4.6 se muestran los casos de uso del actor Visitante. Las operaciones que puede realizar son las siguientes:

- Listar documentos

- Descargar documentos

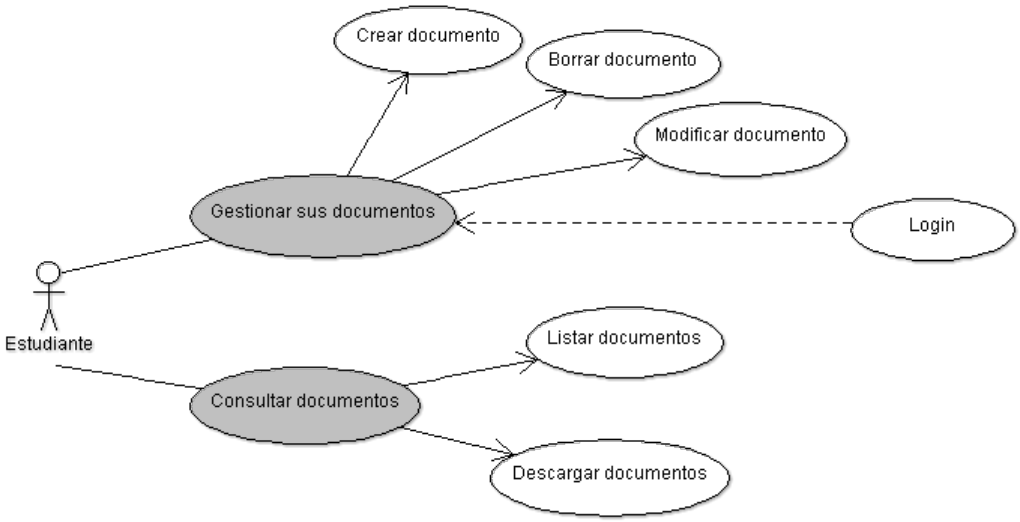

Figura 4.5: Casos de uso del actor Estudiante

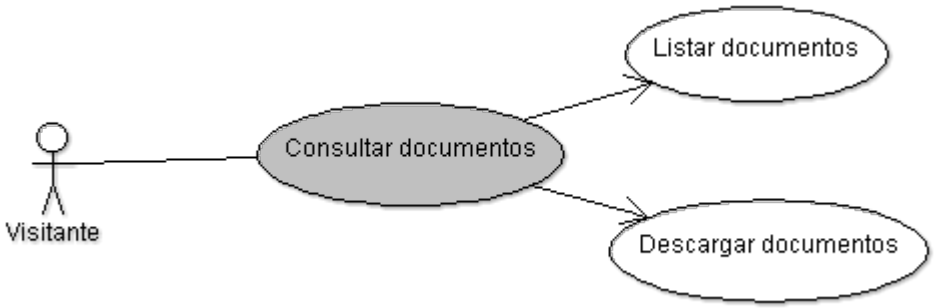

Figura 4.6: Casos de uso del actor Visitante

4.2.2. Listado de casos de uso

En el cuadro 4.4 podemos ver una tabla con todos los casos de uso, actores y el código que los identificará en los casos de uso detallados.

4.3. Diagramas de caso de uso detallados

Una vez definido el diagrama de caso de uso del sistema, en este apartado se realizará un análisis de los requisitos de cada uno de ellos.

Actores	Caso de uso	Código
Administrador	Crear usuario	CU
Administrador	Modificar usuario	MU
Administrador	Borrar usuario	BU
Administrador	Añadir usuario a un grupo	AUG
Administrador	Eliminar usuario de un grupo	EUG
Administrador	Crear grupo	CG
Administrador	Borrar grupo	BG
Administrador	Modificar grupo	MG
Administrador, PDI, PAS, Estudiante	Crear documento	CD
Administrador,PDI, PAS, Estudiante	Borrar documento	BD
Administrador, PDI, PAS, Estudiante	Modificar documento	MD
Administrador, PDI, PAS	Crear carpeta	CC
Administrador, PDI, PAS	Borrar carpeta	BC
Administrador, PDI, PAS	Modificar carpeta	MC
Administrador, PDI, PAS	Compartir carpeta o documento	CCD
Administrador	Asignar rol	AR
Administrador	Modificar rol	MR
Administrador	Quitar rol	QR
Administrador, PDI, PAS	Buscar documentos	BsD
Administrador, PDI, PAS, Estudiante, Visitante	Listar documentos	LD
Administrador, PDI, PAS, Estudiante, Visitante	Descargar documentos	DD
PDI,PAS	Enlazar documentos a su aula/espacio virtual	EDAV
Administrador, PDI, PAS, Estudiante	Login (Iniciar sesión)	L

Tabla 4.4: Listado de casos de uso

Código del caso de uso	CU
Nombre del caso de uso	Crear usuario
Descripción	Permite al administrador añadir un usuario al sistema.
Actores	Administrador.
Precondiciones	El administrador debe haber iniciado sesión previamente.
Flujo normal	1. Acceder a la gestión de usuarios. 2. Pulsar la opción *Crear usuario*. 3. Rellenar el formulario con los datos del usuario. 4. Pulsar en *Guardar*.
Flujo alternativo	1. No rellenar correctamente el formulario, por lo que al pulsar en *Guardar* se avisará del error. 2. Pulsar la opción *Cancelar* en vez de *Guardar*, por lo que no se creará el usuario del sistema.
Postcondiciones	Se mostrará un aviso al administrador del sistema de que el usuario se ha creado correctamente.

Tabla 4.5: Crear usuario

Código del caso de uso	MU
Nombre del caso de uso	Modificar usuario
Descripción	Permite al administrador modificar un usuario del sistema.
Actores	Administrador.
Precondiciones	El administrador debe haber iniciado sesión previamente.
Flujo normal	1. Acceder a la gestión de usuarios. 2. Localizar el usuario a modificar y pulsar la opción *Modificar usuario*. 3. Cambiar los datos que se deseen del formulario. 4. Pulsar en *Guardar*.
Flujo alternativo	1. No rellenar correctamente el formulario, por lo que al pulsar en *Guardar* se avisará del error. 2. Pulsar la opción *Cancelar* en vez de *Guardar*, por lo que no se modificará el usuario.
Postcondiciones	Se mostrará un aviso al administrador del sistema de que el usuario se ha modificado correctamente.

Tabla 4.6: Modificar usuario

Código del caso de uso	BU
Nombre del caso de uso	Borrar usuario
Descripción	Permite al administrador borrar un usuario del sistema.
Actores	Administrador.
Precondiciones	El administrador debe haber iniciado sesión previamente.
Flujo normal	1. Acceder a la gestión de usuarios. 2. Localizar el usuario a borrar y pulsar la opción *Eliminar usuario*. 3. Confirmar que se desea borrar el usuario del sistema.
Flujo alternativo	1. Pulsar *Cancelar* en la confirmación, por lo que no se borrará el usuario del sistema.
Postcondiciones	Se mostrará un aviso al administrador del sistema de que el usuario se ha borrado correctamente.

Tabla 4.7: Borrar usuario

Código del caso de uso	AUG
Nombre del caso de uso	Añadir usuario a un grupo
Descripción	Permite al administrador asignar un grupo a un usuario.
Actores	Administrador.
Precondiciones	El administrador debe haber iniciado sesión previamente. El usuario debe estar creado correctamente.
Flujo normal	1. Acceder a la gestión de grupos del sistema. 2. Localizar el grupo y pulsar en la opción *Añadir usuarios al grupo*. 3. Seleccionar los usuarios que se desean añadir al grupo. 4. Pulsar en *Guardar*.
Flujo alternativo	1. Pulsar *Cancelar*, por lo que no se añadrirán los usuarios al grupo. 2. No seleccionar ningún usuario, por lo que al pulsar en *Guardar* se avisará del error.
Postcondiciones	Se mostrará un aviso al administrador del sistema de que los usuarios se han añadido correctamente.

Tabla 4.8: Añadir usuario a un grupo

Código del caso de uso	EUG
Nombre del caso de uso	Eliminar usuario de un grupo
Descripción	Permite al administrador quitar a un usuario de un grupo.
Actores	Administrador.
Precondiciones	El administrador debe haber iniciado sesión previamente.
Flujo normal	1. Acceder a la gestión de grupos del sistema. 2. Localizar el grupo y pulsar en la opción *Eliminar usuarios del grupo*. 3. Seleccionar los usuarios que se desean quitar del grupo. 4. Pulsar en *Guardar*.
Flujo alternativo	1. Pulsar *Cancelar*, por lo que no se eliminarán los usuarios seleccionados. 2. No seleccionar ningún usuario, por lo que al pulsar en *Guardar* se avisará del error.
Postcondiciones	Se mostrará un aviso al administrador del sistema de que los usuarios se han eliminado correctamente.

Tabla 4.9: Eliminar usuario de un grupo

Código del caso de uso	CG
Nombre del caso de uso	Crear grupo
Descripción	Permite al administrador crear un grupo.
Actores	Administrador.
Precondiciones	El administrador debe haber iniciado sesión previamente.
Flujo normal	1. Acceder a la gestión de grupos del sistema. 2. Pulsar la opción *Crear grupo*. 3. Rellenar el formulario con los datos del grupo. 4. Pulsar en *Guardar*.
Flujo alternativo	1. No rellenar correctamente el formulario, por lo que al pulsar en *Guardar* se avisará del error.
Postcondiciones	Se mostrará un aviso al administrador del sistema de que el grupo se ha añadido correctamente.

Tabla 4.10: Crear grupo

Código del caso de uso	BG
Nombre del caso de uso	Borrar grupo
Descripción	Permite al administrador borrar un grupo del sistema.
Actores	Administrador.
Precondiciones	El administrador debe haber iniciado sesión previamente.
Flujo normal	1. Acceder a la gestión de grupos del sistema. 2. Localizar el grupo a borrar y pulsar la opción *Eliminar grupo*. 3. Confirmar que se desea borrar el grupo del sistema.
Flujo alternativo	1. Pulsar *Cancelar* en la confirmación, por lo que no se borrará el grupo del sistema.
Postcondiciones	Se mostrará un aviso al administrador del sistema de que el grupo se ha borrado correctamente.

Tabla 4.11: Borrar grupo

Código del caso de uso	MG
Nombre del caso de uso	Modificar grupo
Descripción	Permite al administrador modificar un grupo del sistema.
Actores	Administrador.
Precondiciones	El administrador debe haber iniciado sesión previamente.
Flujo normal	1. Acceder a la gestión de grupos del sistema. 2. Localizar el grupo a modificar y pulsar la opción *Modificar grupo*. 3. Cambiar los datos que se deseen del formulario. 4. Pulsar en *Guardar*.
Flujo alternativo	1. No rellenar correctamente el formulario, por lo que al pulsar en *Guardar* se avisará del error.
Postcondiciones	Se mostrará un aviso al administrador del sistema de que el grupo se ha modificado correctamente.

Tabla 4.12: Modificar grupo

Código del caso de uso	CD
Nombre del caso de uso	Crear documento
Descripción	Permite almacenar en el sistema un documento alojado en el dispositivo del usuario.
Actores	Administrador, PDI, PAS, Estudiante.
Precondiciones	El usuario debe haber iniciado sesión previamente.
Flujo normal	1. Dirigirse a la carpeta donde se desea subir el documento. 2. Seleccionar en el menú la opción *Subir*. 3. Rellenar los datos del documento. 4. Pulsar el botón *Subir documento.*
Flujo alternativo	1. El usuario puede no tener permiso para subir documentos en la carpeta seleccionada, por lo que se le informará de ello. 2. Si los datos necesarios para la subida del documento no son correctos, no se permitirá y se informará de ello al usuario.
Postcondiciones	Se mostrará un mensaje confirmando la subida correcta del documento.

Tabla 4.13: Crear documento

Código del caso de uso	BD
Nombre del caso de uso	Borrar documento
Descripción	Permite borrar un documento almacenado en el sistema.
Actores	Administrador, PDI, PAS, Estudiante.
Precondiciones	El usuario debe haber iniciado sesión previamente.
Flujo normal	1. Seleccionar un documento disponible para ser borrado por parte del usuario. 2. Seleccionar la opción de borrado. 3. Confirmar borrado.
Flujo alternativo	1. Si el usuario es distinto al administrador e intenta borrar un documento que no ha sido subido por él, no se permitirá y se informará de ello al usuario.
Postcondiciones	Se mostrará un mensaje confirmando el borrado.

Tabla 4.14: Borrar un documento

Código del caso de uso	MD
Nombre del caso de uso	Modificar documento
Descripción	Permite modificar un documento almacenado en el sistema.
Actores	Administrador, PDI, PAS, Estudiante.
Precondiciones	El usuario debe haber iniciado sesión previamente.
Flujo normal	1. Seleccionar un documento disponible para ser modificado por parte del usuario. 2. Seleccionar la opción de actualizar. 3. Modificar los datos necesarios del documento. 4. Confirmar modificación.
Flujo alternativo	1. Si el usuario es distinto al administrador e intenta modificar un documento que no ha sido subido por él, no se permitirá y se informará de ello al usuario. 2. Si los datos necesarios para la subida del documento no son correctos, no se permitirá y se informará de ello al usuario.
Postcondiciones	Se mostrará un mensaje confirmando la actualización correcta del documento.

Tabla 4.15: Modificar documento

Código del caso de uso	CC
Nombre del caso de uso	Crear carpeta
Descripción	Permite al administrador, PDI y/o PAS crear una carpeta en el sistema.
Actores	Administrador, PDI, PAS.
Precondiciones	El usuario debe haber iniciado sesión previamente.
Flujo normal	1. Navegar hacia el directorio donde se quiera crear la carpeta. 2. Seleccionar la opción de crear carpeta. 3. Escribir el nombre de la carpeta. 4. Seleccionar la opción de guardar.
Flujo alternativo	1. Si los datos necesarios no son correctos, no se creará la carpeta y se informará de ello al usuario.
Postcondiciones	Se mostrará un mensaje confirmando la correcta creación de la carpeta.

Tabla 4.16: Crear carpeta

Código del caso de uso	BC
Nombre del caso de uso	Borrar carpeta
Descripción	Permite al administrador, PDI y/o PAS borrar una carpeta del sistema.
Actores	Administrador, PDI, PAS.
Precondiciones	El usuario debe haber iniciado sesión previamente.
Flujo normal	1. Navegar hacia la carpeta que se quiera borrar. 2. Seleccionar la opción de borrar carpeta. 3. Confirmar el borrado.
Flujo alternativo	
Postcondiciones	Se mostrará un mensaje confirmando que se ha borrado la carpeta correctamente.

Tabla 4.17: Borrar carpeta

Código del caso de uso	MC
Nombre del caso de uso	Modificar carpeta
Descripción	Permite al administrador, PDI y/o PAS modificar una carpeta existente en el sistema.
Actores	Administrador, PDI, PAS.
Precondiciones	El usuario debe haber iniciado sesión previamente.
Flujo normal	1. Navegar hacia la carpeta que se quiera modificar. 2. Seleccionar la opción de modificar carpeta. 3. Cambiar los datos necesarios. 4. Seleccionar la opción de guardar.
Flujo alternativo	1. Si los datos necesarios no son correctos, no se creará la carpeta y se informará de ello al usuario.
Postcondiciones	Se mostrará un mensaje confirmando la correcta modificación de la carpeta.

Tabla 4.18: Modificar carpeta

Código del caso de uso	CCD
Nombre del caso de uso	Compartir carpeta o documento
Descripción	Permite al administrador, PDI y/o PAS compartir una carpeta y/o documento existente en el sistema.
Actores	Administrador, PDI, PAS.
Precondiciones	El usuario debe haber iniciado sesión previamente.
Flujo normal	1. Seleccionar la carpeta o documento a compartir. 2. Seleccionar la opción *Compartir*. 3. Seleccionar los grupos y/o usuarios con lo que se quiera compartir. 4. Confirmar la compartición del recuso.
Flujo alternativo	1. Si no se selecciona ningún usuario o grupo al que compartir se mostrará un mensaje de error avisando al usuario.
Postcondiciones	Se mostrará un mensaje informando que se ha compartido el recurso correctamente.

Tabla 4.19: Compartir carpeta o documento

Código del caso de uso	AR
Nombre del caso de uso	Asignar rol
Descripción	Permite al administrador añadir un rol (conjunto de permisos) a un grupo en una jerarquía de carpetas determinada.
Actores	Administrador.
Precondiciones	El administrador debe haber iniciado sesión previamente.
Flujo normal	1. Navegar a la carpeta raíz para la que desea asignar el rol. 2. Seleccionar la opción de gestionar permisos. 3. Seleccionar el grupo a añadir. 4. Seleccionar el rol que se le desea asignar. 5. Seleccionar la opción de guardar.
Flujo alternativo	
Postcondiciones	Se mostrará un mensaje confirmando que el rol se ha asignado correctamente.

Tabla 4.20: Asignar rol

Código del caso de uso	MR
Nombre del caso de uso	Modificar rol
Descripción	Permite al administrador modificar el rol (conjunto de permisos) de un grupo en una jerarquía de carpetas determinada.
Actores	Administrador.
Precondiciones	El administrador debe haber iniciado sesión previamente.
Flujo normal	1. Navegar a la carpeta raíz para la que desea modificar el rol. 2. Seleccionar la opción de gestionar permisos. 3. Seleccionar el grupo a modificar. 4. Seleccionar el nuevo rol que se le desea asignar. 5. Seleccionar la opción de guardar.
Flujo alternativo	
Postcondiciones	Se mostrará un mensaje confirmando que el rol se ha modificado correctamente.

Tabla 4.21: Modificar rol

Código del caso de uso	QR
Nombre del caso de uso	Quitar rol
Descripción	Permite al administrador quitar un rol (conjunto de permisos) a un grupo en una jerarquía de carpetas determinada.
Actores	Administrador.
Precondiciones	El administrador debe haber iniciado sesión previamente.
Flujo normal	1. Navegar a la carpeta raíz para la que desea quitar el rol. 2. Seleccionar la opción de gestionar permisos. 3. Borrar el grupo de la lista de grupos con roles en la carpeta. 4. Seleccionar la opción de guardar.
Flujo alternativo	
Postcondiciones	Se mostrará un mensaje confirmando que el rol se ha borrado correctamente.

Tabla 4.22: Quitar rol

Código del caso de uso	BsD
Nombre del caso de uso	Buscar documentos
Descripción	Busca los documentos que cumplen los criterios especificados por el usuario en el sistema.
Actores	Administrador, PDI, PAS.
Precondiciones	
Flujo normal	1. Seleccionar la opción *Buscar* en el menú. 2. Introducir los criterios de búsqueda deseados. 3. El sistema muestra los documentos encontrados.
Flujo alternativo	1. Si no se encuentran documentos que cumplan los criterios, se informará de ello al usuario.
Postcondiciones	

Tabla 4.23: Buscar documentos

Código del caso de uso	LD
Nombre del caso de uso	Listar documentos
Descripción	Lista el contenido de un directorio del sistema.
Actores	Administrador, PDI, PAS, Estudiante, Visitante.
Precondiciones	
Flujo normal	1. Seleccionar el directorio que se desea listar.
Flujo alternativo	
Postcondiciones	Se mostrará un listado del contenido que existe en un directorio.

Tabla 4.24: Listar documentos

Código del caso de uso	DD
Nombre del caso de uso	Descargar documentos
Descripción	Descarga un documento almacenado en el sistema.
Actores	Administrador, PDI, PAS, Estudiante, Visitante.
Precondiciones	
Flujo normal	1. Seleccionar documento. 2. Pulsar la opción de descargar.
Flujo alternativo	
Postcondiciones	El documento seleccionado se descargará en el equipo del usuario.

Tabla 4.25: Descargar documentos

Código del caso de uso	EDAV
Nombre del caso de uso	Enlazar documentos a su aula/espacio virtual
Descripción	Inserta un documento almacenado en el sistema en un aula o espacio del Campus Virtual.
Actores	PDI, PAS.
Precondiciones	El usuario debe haber iniciado sesión previamente. El usuario debe tener algún aula o espacio virtual en el Campus Virtual de la Universidad de Extremadura.
Flujo normal	1. Seleccionar el aula o espacio virtual donde desea introducir el archivo. 2. Pulsar el botón de *activar edición.* 3. Localizar el tema donde se quiere añadir el documento. 3. Pulsar la opción *Añade una actividad o recurso.* 4. Pulsar en agregar un archivo. 5. Abrir el selector de archivos pulsando en la opción *seleccionar archivos.* 6. Seleccionar el repositorio *CUMDoc.* 7. Localizar el documento que desea subir a su aula virtual. 8. Seleccionar la opción de *guardar cambios y regresar al curso.*
Flujo alternativo	1. Si no se selecciona un archivo se avisará al usuario del error. 2. Si pulsa en *Cancelar* no se subirá el archivo al aula virtual.
Postcondiciones	El documento seleccionado aparecerá en el aula o espacio virtual del usuario, en el tema donde se seleccionó la opción de añadir una actividad o recurso.

Tabla 4.26: Enlazar documentos a su aula/espacio virtual

Código del caso de uso	L
Nombre del caso de uso	Login (Iniciar sesión)
Descripción	Permite a un usuario registrado identificarse en el sistema.
Actores	Administrador, PDI, PAS, Estudiante.
Precondiciones	El usuario debe disponer de una cuenta en el sistema.
Flujo normal	1. Seleccionar la opción de identificarse. 2. Introducir su usuario y contraseña.
Flujo alternativo	1. Si las credenciales de identificación no son correctas, no se iniciará sesión y se informará al usuario del error.
Postcondiciones	Se inicia la sesión del usuario.

Tabla 4.27: Login (Iniciar sesión)

Capítulo 5

Diseño

Contenidos

Tras establecer las distintas funcionalidades del sistema, en este capítulo se realizará su diseño. Para ello, se analizarán las distintas partes que forman el proyecto, viendo las alternativas más destacables entre las existentes.

En la sección 5.1 se analizan distintos métodos de autentificación para las plataformas que se realizarán en este Trabajo Fin de Grado.

En la sección 5.2 se describe diferentes gestores documentales, analizando cada uno de ellos.

En la sección 5.3 se explica la plataforma educativa Moodle y su implementación en la Universidad de Extremadura.

En la sección 5.4 se explica la necesidad de crear una aplicación web para los usuarios del gestor documental, analizando las causas que motivan su creación y las soluciones que aporta.

En la sección 5.5 se explica cómo estará formada la aplicación web a realizar en este Trabajo Fin de Grado.

En la sección 5.6 se realizará una conclusión final de la fase de diseño.

5.1. Autentificación

Para la realización de la aplicación web se necesitará un sistema de autentificación único, que sirva para todas las herramientas utilizadas en este Trabajo Fin de Grado, como Moodle y ownCloud. Este sistema de autentificación deberá permitir el acceso a los documentos y deberá diferenciar los distintos roles que pueden adquirir los usuarios.

5.1.1. Base de datos

Una primera opción sería utilizar una base de datos relacional, pudiendo ser MySQL, PostgreSQL, MariaDB, entre otras. Se almacenaría en tablas los distintos usuarios con sus contraseñas, además de los datos necesarios para las aplicaciones, como pueden ser el nombre y apellidos del usuario.

Esta solución plantea una serie de problemas:

- Dificultad de definir una jerarquía y asignar roles entre los distintos usuarios.
- Dificultad de importar esta base de datos en ownCloud. Además de agregar los usuarios a ownCloud, se debería agregar los distintos grupos existentes en la comunidad universitaria y de manera automática asignarle los usuarios que los componen.

Estos problemas hacen que esta solución no sea la adecuada para nuestra aplicación.

5.1.2. LDAP

LDAP [48] es un protocolo de nivel de aplicación encargado de permitir el acceso a un servicio de directorio ordenado y distribuido. Además, se considera una base de datos a la que pueden realizarse consultas.

LDAP presenta la información en una estructura jerárquica de árbol denominada DIT [49] (Árbol de información de directorio). En la figura 5.1, se puede observar un ejemplo de la estructura creada por LDAP.

Cada entrada de LDAP está formada por un conjunto de pares clave/valor denominados atributos, que permiten caracterizar el objeto. Existen dos tipos de atributos:

- **Atributos normales:** Son los atributos comunes que distinguen al objeto, como son el nombre, el apellido, entre otros.
- **Atributos operativos:** Son los atributos que sólo el servidor puede acceder y manipular, como son las fechas de modificación.

Una entrada se indexa mediante un nombre completo (DN) que permite identificar al elemento de la estructura de árbol. Un DN se forma tomando el nombre de la ruta hasta llegar al elemento, agregándole el nombre de la entrada.

Algunas de las claves que se pueden utilizar en LDAP son las siguientes:

- **uid:** Identificación única obligatoria, siendo el id del usuario.

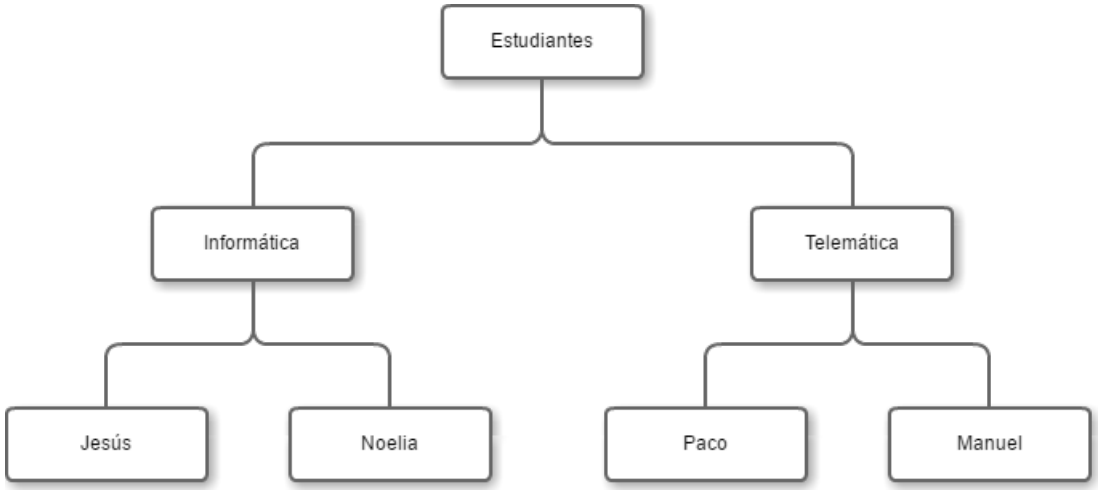

Figura 5.1: Ejemplo de estructura LDAP. .

- **cn:** Nombre completo de la persona.
- **givenname:** Nombre de pila de la persona.
- **sn:** Apellidos de la persona.
- **o:** Organización de la que forma parte la persona.
- **u:** Departamento de la organización de la que forma parte la persona.
- **mail:** Dirección de correo electrónico de la persona.

Para importar y exportar datos en LDAP es necesario crear archivos de texto simple en formato *'.ldif'*. Estos archivos deben cumplir una serie de normas:

- Cada entrada nueva deberá separarse de la entrada anterior mediante una línea en blanco.
- Es posible definir un atributo a través de diversas líneas, comenzando las líneas siguientes a la primera con un espacio en blanco o una tabulación.
- Es posible definir distintos valores para un mismo atributo, repitiendo la asignación *'name:value'* en otra línea.

En la figura 5.2, se puede observa un ejemplo de fichero *'.ldif'* para la creación de un usuario.

Actualmente, existen diversas herramientas encargadas de simplificar el proceso de creación de estos archivos *'.ldif'*. Además, se encargan de introducir los datos en LDAP, así como de borrarlos y editarlos.

Una de las herramientas más conocidas es *phpLDAPadmin* [50], un cliente LDAP basado en una sencilla interfaz web, que permite administrar un servidor LDAP a través del navegador. Esta herramienta se encuentra escrita en PHP, y fue lanzada bajo una licencia GPL. Además, ha sido traducida a diferentes idiomas, entre ellos el español.

phpLDAPadmin cuenta con las siguientes características:

```
 GNU nano 2.2.6          Fichero: usuario.ldif                Modificado

dn: uid=jesalguer,ou=usuarios,dc=cum,dc=es
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
uid: jesalguer
sn: Salguero
givenName: Jesus
cn: Jesus Salguero
displayName: Jesus Salguero
uidNumber: 2100
gidNumber: 2000
userPassword: 123456
gecos: Jesus Salguero
loginShell: /bin/bash
homeDirectory: /home/jesalguer
```

Figura 5.2: Ejemplo de archivo *.ldif*.

- Dispone de una vista con forma de árbol jerárquico que permite recorrer toda la estructura LDAP.
- Dispone de diferentes plantillas para introducir datos en LDAP, facilitando así su incorporación.
- Puede borrar desde entradas individuales hasta árboles completos.
- Se puede utilizar en diferentes plataformas, utilizando únicamente un navegador web.

En la figura 5.3, se puede observar la interfaz web de *phpLDAPadmin*, en la cual se está agregando un nuevo usuario al sistema. Además, en el lateral izquierdo, se encuentra la estructura jerárquica LDAP.

5.1.2.1. Conclusión

LDAP nos permite crear una estructura jerarquizada, donde podemos crear distintos grupos y agregar a esos grupos los usuarios deseados. Además, nos permite subdividir los grupos en otros más pequeños, permitiendo crear distinciones más precisas.

Hemos decidido utilizar LDAP para nuestra aplicación por los siguientes motivos:

- Permite crear fácilmente grupos y usuarios, obteniendo como resultado una estructura fácil de entender y útil para nuestra aplicación.
- ownCloud y Moodle permiten utilizar LDAP como base de datos para autentificación. Además, ownCloud importa a los grupos existentes en LDAP con sus usuarios, por lo que directamente en ownCloud se puede saber el perfil y los roles que tiene cada usuario sin necesidad de configurarlos de manera manual.
- Actualmente, el Campus Virtual de la Universidad de Extremadura utiliza LDAP, por lo que nuestra aplicación se podría integrar sin ningún problema con los recursos disponibles en la universidad.

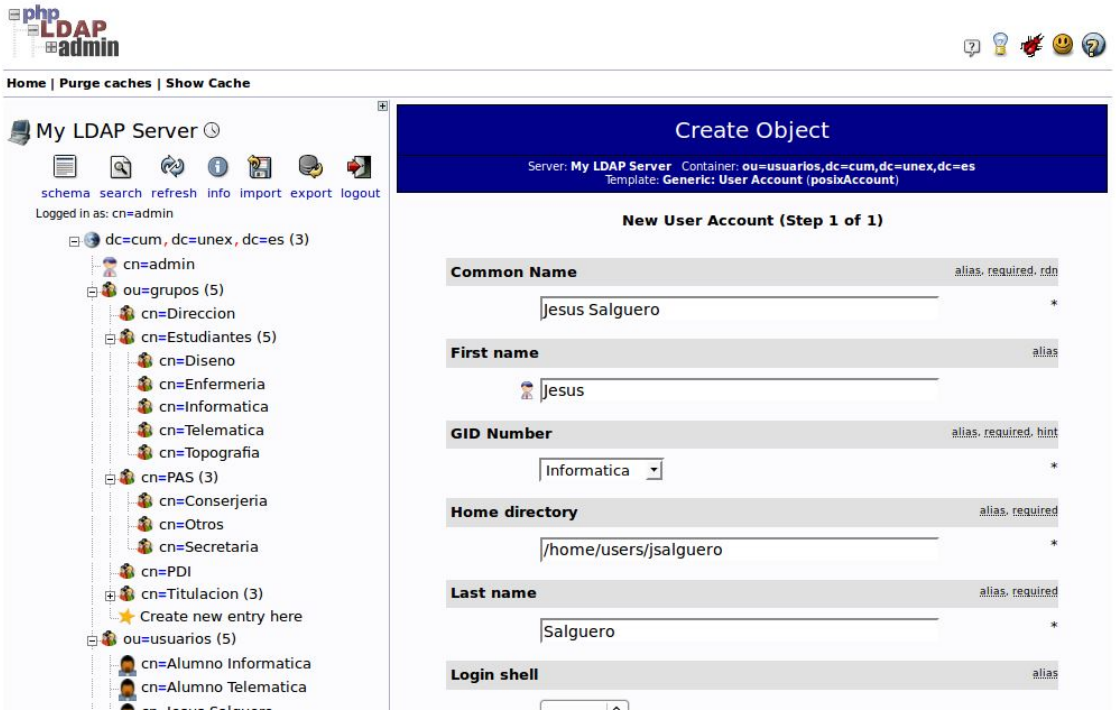

Figura 5.3: Interfaz web de *phpLDAPadmin*.

5.2. Gestor documental

Nuestra aplicación necesita almacenar los documentos que compartan los distintos miembros de la comunidad universitaria. Para ello debe existir una plataforma encargada del almacenamiento de los documentos y además, debe facilitar a la aplicación el acceso a ellos.

5.2.1. Base de datos

Se puede almacenar en una base de datos cualquier tipo de documento. Para ello, es necesario transformarlos en elementos Binary Large Object (BLOB). Un BLOB [33] es un archivo de gran tamaño, como puede ser una imagen o un archivo de audio, que se debe manipular de una manera especial en una base de datos.

Las ventajas [34] de utilizar este tipo de objetos son:

- Permite el acceso concurrente a los archivos.
- Los archivos quedan almacenados coherentemente en tablas de la base de datos.
- Al tener los datos separados del código, se facilita el mantenimiento y actualización de la aplicación.

Las desventajas de utilizar este tipo de objetos son:

- Una de las principales desventajas es el tiempo de acceso a los archivos. Se debe consultar a la base de datos sobre el archivo, traerlo y transformarlo, lo que requiere bastante tiempo. Además, a medida que aumenta el número de archivos en la base de datos

la consulta se ralentiza bastante, haciendo que la obtención del documento se demore bastante tiempo.

- La base de datos incrementará su tamaño bastante si se persisten una gran cantidad de documentos, haciendo que la administración sea bastante complicada.
- Se requiere un conocimiento bastante más complejo sobre la persistencia de documentos.

Estos inconvenientes hacen que no sea una buena opción almacenar los documentos en una base de datos para el uso de nuestra aplicación.

5.2.2. Alfresco

Alfresco [35] es un sistema de gestión de contenido empresarial desarrollado en Java, basado en estándares abiertos. Permite su uso en cualquier sistema operativo y arquitectura que soporte Java Enterprise Edition.

Alfresco ofrece tres versiones distintas:

- **Alfresco Community Edition**: Versión libre y gratuita de Alfresco para desarrolladores, ofreciendo el código bajo la Licencia Pública General Reducida de GNU (LGPL v3). Existe una comunidad en la cual se realizan distintas aportaciones no oficiales y se resuelven las cuestiones de los usuarios. Esta versión presenta limitaciones respecto a las versiones de pago.
- **Alfresco Content Services**: Versión de pago de Alfresco. Ofrece una administración más simplificada, el cifrado de documentos, mejoras en la seguridad, entre otras. Ofrecida para organizaciones y empresas que necesiten una gran escalabilidad, rendimiento y una asistencia de 24 horas todos los días de la semana.
- **Alfresco Cloud**: Versión de pago orientada a empresas más pequeñas o que presenten menos necesidades. No es necesario instalarlo a nivel local, ya que te ofrecen una versión de Alfresco en la nube, lo que permite el acceso desde cualquier lugar.

Las principales características de Alfresco son las siguientes:

- Permite manejar y compartir documentos.
- Almacena información sobre los accesos a los documentos y proporciona seguridad especializada.
- Permite incorporar metadatos en los documentos.
- Permite compartir directorios y archivos entre los distintos usuarios.
- Permite organizar a los usuarios en grupos, otorgándole roles y privilegios. Se puede limitar el acceso a un directorio a un grupo determinado.

Existen diferentes formas en la que una aplicación cliente interactúa con el repositorio Alfresco, ya que este facilita distintas APIs para ello, según el tipo de cliente o las funcionalidades que se requieran.

5.2.2.1. API CMIS

CMIS [1] es un estándar abierto que permite la interoperabilidad entre sistemas de gestión de contenidos y documentos. Ofrece a la aplicación cliente la posibilidad de interactuar con el repositorio, generando respuestas como XML, JSON o directamente el fichero requerido.

CMIS presenta varios problemas:

- La API proporcionada por Alfresco es una implementación genérica y no cubre toda la funcionalidad de Alfresco.
- La aplicación cliente debe ceñirse a la API existente, sin poder añadir funcionalidades necesarias.
- El rendimiento es bastante bajo.

5.2.2.2. Web scripts

Los web scripts [1] ofrecen una API REST de servicios web para el contenido del repositorio, basada en peticiones y respuestas HTTP. Son una implementación ligera, que no requiere modificaciones en el núcleo de Alfresco y cuentan con la funcionalidad de autentificación además de los servicios proporcionados por Alfresco.

Los web scripts soportan distintos formatos de documento, por lo que pueden funcionar mostrando una página HTML o sirviendo un fichero JSON.

Existen dos tipos de web scripts:

- De repositorio: Interaccionan con el repositorio. Permiten extender la API del repositorio Alfresco, definiendo nuevos métodos cuya implementación se basa en la API JavaScript embebida en Alfresco. Además, proveen al repositorio de un servicio de contenido web basado en el patrón MVC.
- De presentación: No tienen acceso al repositorio. Principalmente se usan en arquitecturas distribuidas y sobre la plataforma Spring Surf.

Los web scripts suelen ser la opción más habitual para interactuar con el repositorio Alfresco, ya que presentan una funcionalidad muy extensa y personalizable, son reusables y multiplataforma, no presenta dependencias externas, son escalables y no necesitan mantener sesiones.

5.2.2.3. Conclusión

En los primeros meses del proyecto se eligió la opción de continuar con el Trabajo Fin de Grado de Manuel Botellero Gutiérrez [1], utilizando como gestor documental la plataforma Alfresco. Se instaló la plataforma en una máquina y se realizaron pruebas de integración con LDAP y Moodle. Aunque Alfresco sería una buena propuesta para nuestro gestor documental, tras las distintas pruebas se comprobó que existen diferentes inconvenientes que no lo hacen óptimo para nuestra aplicación:

- Es una aplicación de pago, aunque presente una versión gratuita para desarrolladores, esta se encuentra limitada.
- Proporcionan distintas APIs para interactuar con la aplicación cliente, pero son más complicadas de programar que otras alternativas existentes.

- Desde la versión 5.0.c, Alfresco ha dejado de soportar el plugin que tenía para servir de repositorio en Moodle, por lo que Moodle ha borrado este repositorio de sus estándares desde su versión 3.2 [36].

5.2.3. Almacenamiento en la nube

El almacenamiento en la nube permite a los distintos usuarios guardar sus documentos en servidores que pueden ser accedidos desde Internet. Esto permite poder acceder a la información almacenada desde cualquier dispositivo y poder recuperarla en caso de pérdida. Existen numerosos servicios que ofrecen este tipo de solución, destacando los siguientes:

- **Dropbox[37]:** Es uno de los servicios más populares de almacenamiento en la nube. Ofrece un plan gratuito de 2GB llamado Basic, y un plan de pago de 1TB, con una suscripción mensual de 9,99€ o una suscripción anual de 99,00€. Existe además diferentes planes para empresas, siendo el más económico el plan 'Estándar', en cual tiene un coste de 10€ al mes por usuario, concediendo 2TB de almacenamiento. Ofrece aplicaciones gratuitas para acceder a Dropbox mediante el ordenador o dispositivo móvil, y permite compartir archivos con otras personas, almacenándolos en su cuenta o permitiendo la descarga de los archivos en su equipo personal.

- **OneDrive[38]:** Este servicio viene preinstalado en Windows 10 y funciona en otras plataformas como Mac OS o Android. Ofrece un plan gratuito de 5GB y uno de pago de 1TB con acceso para 5 usuarios distintos por 99,00€ al año. Existen diferentes planes para empresas, siendo el más económico el plan 'Empresa Essentials', donde cada usuario y mes cuesta 4,20€. OneDrive permite el uso de las herramientas incluidas en Microsoft Office, permitiendo la edición de documentos de manera online.

- **Google Drive[39]:** Este servicio de Google permite el almacenamiento de ficheros y su edición a través de Google Docs. El plan gratuito ofrece 15GB de almacenamiento en la nube, existiendo otros 3 planes de pago: 100GB por 1,99€ al mes, 1TB por 9,99€ al mes y 10TB por 99,99€ al mes. También ofrece un plan para empresas, siendo el más económico por 8€ por usuario al mes.

- **ownCloud:** Es un software libre multiplataforma que permite el almacenamiento de documentos en la nube. Es una alternativa libre y gratuita a los anteriores, pero se tiene que disponer de los servidores donde se alojarán los distintos documentos. Además del almacenamiento de archivos, permite su edición en línea utilizando la herramienta OpenOffice y permite compartir documentos fácilmente con usuarios y grupos de ownCloud y con usuarios externos. También permite el acceso de aplicaciones externas mediante WebDAV. Existe una variante de ownCloud llamada NextCloud, que presenta pequeñas modificaciones y dota de más importancia a la comunidad de usuarios y desarrolladores que hay detrás.

De las distintas opciones vistas, nos centraremos en ownCloud/NextCloud debido a que la Universidad posee sus propios servidores, el software es gratuito y libre, y proporciona una API llamada WebDAV para el acceso a los documentos por parte de nuestra aplicación web.

5.2.3.1. ownCloud/NextCloud

ownCloud [40] es un software libre multiplataforma que permite el almacenamiento de archivos en línea. Para ser instalado es necesario disponer de PHP v5.6 o superior, y una base de datos SQLite, MySQL o PostgreSQL.

ownCloud posee una serie de funcionalidades[41]:

- Sincronización de archivos entre diversos equipos informáticos.
- Almacenamiento seguro de archivos con cifrado.
- Compartición de archivos entre usuarios de ownCloud o de manera pública.
- Lector de música en línea.
- Servidor de archivos WebDAV.
- Calendario y sincronización de calendarios entre usuarios.
- Administración de contactos.
- Editor de texto en línea.
- Visor de documentos en línea.
- Galería de imágenes, permitiendo la visualización y la clasificación en álbumes.
- Administración de archivos favoritos.

Nextcloud [42] es una variante de ownCloud, creado por el mismo autor de ownCloud, con una tendencia más abierta y dotando de mayor importancia a la comunidad que existe detrás. Las principales diferencias son las siguientes:

- ownCloud, aun siendo un software libre con código abierto, ofrecen servicios de soporte y funcionalidades extras de pago. Estas funcionalidades extras son de código cerrado y no están presentes en la versión gratuita. En cambio, los desarrolladores de Nextcloud se comprometen a mantener todas las funcionalidades como software de código abierto.
- Nextcloud ofrece funciones de videoconferencia, llamadas de audio y la posibilidad de compartir la pantalla que no están presentes en ownCloud.
- Nextcloud cambia MySQL por MariaDB y OpenOffice por LibreOffice

En definitiva, es el mismo proyecto con pequeñas modificaciones, por lo que instalar ownCloud o Nextcloud no será una cuestión importante, a no ser que sea una empresa que necesita un soporte específico con alguna funcionalidad extra[43].

5.2.3.2. WebDAV

Para poder interaccionar con ownCloud desde nuestra aplicación web, este nos permite usar el protocolo WebDAV. Este protocolo fue desarrollado por Internet Engineering Task Force (IETF) y proporciona funcionalidades para crear, cambiar y mover documentos en un servidor remoto [44].

La mayoría de los sistemas operativos actuales tienen soporte para utilizar el protocolo WebDAV, permitiendo trabajar con archivos remotos como si fuese un disco duro local conectado al ordenador.

El proyecto WebDAV tiene varios subproyectos relacionados con el protocolo para estandarizarlo lo máximo posible. Entre ellos cabe destacar el proyecto 'Goliath' que permite la edición y gestión de páginas web con el protocolo WebDAV [45].

WebDAV está definido por una serie de características [46]:

- WebDAV proporciona un conjunto de nuevos métodos y cabeceras para usar en HTTP:
  - **PROPFIND:** Usado para recuperar propiedades, almacenadas como XML, desde un recurso. También permite recuperar la jerarquía de directorios de un sistema remoto.
  - **PROPPATCH:** Usado para cambiar y borrar múltiples propiedades de un recurso.
  - **MKCOL:** Usado para crear colecciones (directorios).
  - **COPY:** Usado para copiar un recurso desde un URI a otro.
  - **MOVE:** Usado para mover un recurso desde un URI a otro.
  - **LOCK:** Usado para bloquear un recurso.
  - **UNLOCK:** Usado para desbloquear un recurso.
- WebDAV utiliza colecciones. Este término se utiliza para definir un agrupamiento de recursos (URIs), siendo similar al concepto de directorios. Las colecciones terminan en '/' y pueden ser creadas con el método MKCOL. Las colecciones indican, en definitiva, una ruta dentro de la jerarquía de directorios, como por ejemplo "*http://192.168.1.126/owncloud/remote.php/webdav/Titulaciones/Informatica/PrimerCurso/*"
- WebDAV utiliza metadatos asociados a los archivos y colecciones, como por ejemplo la fecha de última modificación de un archivo, que pueden ser consultadas desde clientes WebDAV.
- WebDAV permite bloquear el acceso a un determinado recurso desde clientes WebDAV, así como bloquear la escritura únicamente.
- Este protocolo utiliza una conexión TCP para la transferencia de datos y control, no abriendo una nueva conexión por cada archivo a transferir, reduciendo la sobrecarga y mejorando la velocidad de transferencia.
- Usa el mismo puerto que HTTP, el puerto 80, lo cual reduce los problemas ocasionados con los firewall[47].
- WebDAV no tiene ninguna restricción sobre el tipo de documento, pudiendo trabajar con formatos JPEG, GIF, HTML, entre otros.

5.2.3.3. Conclusión

Tras analizar las distintas opciones para el almacenamiento de documentos de nuestra aplicación, se ha decidido utilizar ownCloud debido a las siguientes ventajas:

- Es gratuito y libre. No necesitamos ninguna funcionalidad extra especial, por lo que se podrá utilizar ownCloud o Nextcloud, ya que para nuestra aplicación no habrá ninguna diferencia.
- Permite el uso de WebDAV, facilitando la interacción entre nuestra aplicación y ownCloud, pudiendo obtener los documentos sin excesiva dificultad.
- Existe actualmente en la Universidad de Extremadura un servidor de Nextcloud para el PDI, por lo que nuestra aplicación aprovecharía este recurso ya existente.

5.3. Moodle

Moodle [51] es una plataforma de aprendizaje diseñada para ayudar a educadores a crear cursos en línea y entornos de aprendizaje virtuales. Es open source y posee una gran comunidad donde se realizan numerosas aportaciones y se resuelven los problemas y cuestiones de los usuarios.

Las principales características de Moodle son[52]:

- Sistema en constante evolución y actualización.
- Posibilidad de personalizar la plataforma.
- Creación de distintos perfiles de usuario, como puede ser estudiante, profesor, administrador, entre otros.
- Importación y exportación de datos en formato SCROM.
- Interfaz liviana, siguiendo las normas W3C.
- Sistema escalable en cuanto a la cantidad de estudiantes.
- Posibilidad de diversos métodos de evaluación y calificación.
- Accesibilidad desde cualquier navegador web.

5.3.1. Campus Virtual de la Universidad de Extremadura

La Universidad de Extremadura cuenta con su propia plataforma Moodle, llamada Campus Virtual. En ella, los distintos docentes pueden disponer de diversas aulas para sus asignaturas y los estudiantes pueden acceder al material proporcionado por el docente. Además, sirven como complemento a sus clases presenciales, pudiendo compartir videoconferencias, realizar exámenes, pedir la entrega de tareas, entre otras tantas funciones.

En la actualidad, el Campus Virtual de la Universidad de Extremadura se encuentra dividido en:

- **AVUEx:** En esta plataforma se encuentran las aulas utilizadas por los docentes para la impartición de las asignaturas oficiales, como complemento a sus clases presenciales, o como medio para realizar las asignaturas de manera virtual.
- **EVUEx:** En esta plataforma están los distintos espacios virtuales proporcionados a los docentes para todo lo demás. Se pueden encontrar espacios para departamentos, para las prácticas en empresa, para la impartición de asignaturas no oficiales de la universidad e incluso espacios para usuarios ajenos a la universidad, pudiendo acceder con su correo electrónico personal.

5.4. Aplicación web

En este Trabajo Fin de grado se desarrollará una aplicación web para la gestión documental del centro. A continuación se detalla la importancia y la necesidad de esta aplicación.

5.4.1. Necesidad de la aplicación web

La plataforma ownCloud proporciona a los usuarios una interfaz web desde la cual acceder a sus documentos. Desde esta interfaz se pueden realizar distintas acciones, como agregar nuevos archivos, crear nuevas carpetas o compartir documentos propios con otros usuarios y/o grupos, entre otras. Desde este punto de vista, se puede ver innecesario la creación de una nueva aplicación web para los usuarios, pero la plataforma ownCloud presenta una serie de problemas importantes por lo que no sería viable su uso para este proyecto, problemas que desde su interfaz web no se corrigen, pero si desde la aplicación web a desarrollar en este proyecto. Estos problemas son los siguientes:

- Las carpetas y archivos compartidos aparecen directamente en la ruta inicial de la cuenta del usuario con la que ha sido compartida. Si el número de carpetas y archivos compartidos es alto se vuelve bastante difícil localizar la información necesaria sin ningún tipo de orden ni estructura. Además, cada vez que se comporta algún archivo nuevo aparecerá en la ruta inicial, teniendo el usuario que organizar constantemente sus archivos.
- Si se comparte una estructura de carpetas existe un problema con los permisos de los usuarios, ya que los usuarios con los que ha sido compartida tendrían todos los permisos o permisos de solo lectura, pero no podría especificarse de forma concreta en que carpetas debe o no tener permisos ni cuales puede o no visualizar dentro de esa estructura.
- En carpetas compartidas, no se puede especificar de ninguna manera que únicamente tenga permisos de modificación el usuario que subió el documento, por lo que cualquier participante podría o modificar todos o ninguno de los archivos que se encuentren en estas carpetas, según el tipo de compartición elegido.
- No existe ningún tipo de cuenta *"invitada"* la cual permita el acceso a los archivos compartidos sin necesidad de introducir credenciales.
- No se puede limitar el uso de ownCloud como servidor de alojamiento de archivos privados al conjunto de usuarios que nos interese. Es decir, el grupo de estudiantes podría usar ownCloud para almacenar sus propios archivos privados, cargando demasiado los servidores. OwnCloud permite limitar la subida de archivos, pero si se prohíbe subir archivos

personales tampoco podrían subir archivos relacionados con las asignaturas impartidas del centro.

5.4.2. Soluciones

La aplicación web a desarrollar en este proyecto se encargará de solucionar los problemas anteriormente mencionados. Para ello se han ido analizando cada uno de los problemas y se ha pensado una posible solución.

Respecto a la cuenta *"invitada"* es posible crear un grupo en LDAP de cuentas de invitados, que serán utilizadas por ownCloud como si fuesen cuentas de usuarios. Estas cuentas tendrán un usuario y contraseña, únicamente conocidos por el administrador. Se habilitará una opción en la ventana de inicio de la aplicación web que permita acceder sin necesidad de introducir ningún tipo de credencial, pudiendo ver el contenido que está compartido con el grupo de invitados. El sistema, al detectar que el usuario ha pulsado este botón, conectará mediante WebDAV a ownCloud, introduciendo como parámetros de inicio de sesión una de estas cuentas de invitado. Aunque en definitiva se acceda con una cuenta con usuario y contraseña, este proceso será invisible para el usuario.

Los demás problemas están relacionados con la estructura de carpetas existente en ownCloud. Para ello se ha pensado realizar una estructura virtual previa a las carpetas localizadas en ownCloud. Esta estructura virtual será una jerarquía de directorios que estarán conectados a las distintas carpetas reales de ownCloud.

En la figura 5.4, se puede observar un ejemplo de estructura virtual. Existen diversos directorios virtuales que en realidad no existen en ownCloud (Titulaciones, Informática, MDP, MDP Profesor y MDP Estudiantes). Estos directorios virtuales permitirán al usuario navegar de manera organizada dentro del gestor, hasta llegar a un nodo hoja, el cual conectará con una carpeta real de ownCloud. En este ejemplo, al acceder a la carpeta *'MDP Profesor'*, se realizará una petición WebDAV a la carpeta real que tenga el profesor en ownCloud y se mostrará los archivos que se encuentren en ella.

Esta estructura virtual proporcionará las siguientes soluciones:

- Una estructura virtual de carpetas ofrecerá organización dentro del gestor documental, ordenando las distintas carpetas según una jerarquía de directorios virtuales. Esto permitirá al usuario encontrar los documentos deseados de manera fácil y sencilla.
- La estructura virtual estará conectada a carpetas reales en ownCloud que serán asignadas al usuario encargado de su gestión. Este usuario podrá elegir que usuarios o grupos podrán visualizar esa carpeta, impidiendo el acceso a los demás.
- Dentro de la estructura virtual existirá una parte dónde los estudiantes podrán subir el contenido relacionado con las asignaturas del centro. Este contenido solo podrá ser modificado por el usuario que lo subió aunque se encuentren en la misma carpeta.
- La estructura virtual solo habilita la posibilidad de subir archivos a las carpetas reales de ownCloud, por lo que no se podrá subir contenido personal ya que este contenido será compartido y visible para los demás usuarios.

5.4.3. Acciones permitidas en la aplicación web

En la aplicación web a desarrollar se podrá realizar las siguientes acciones:

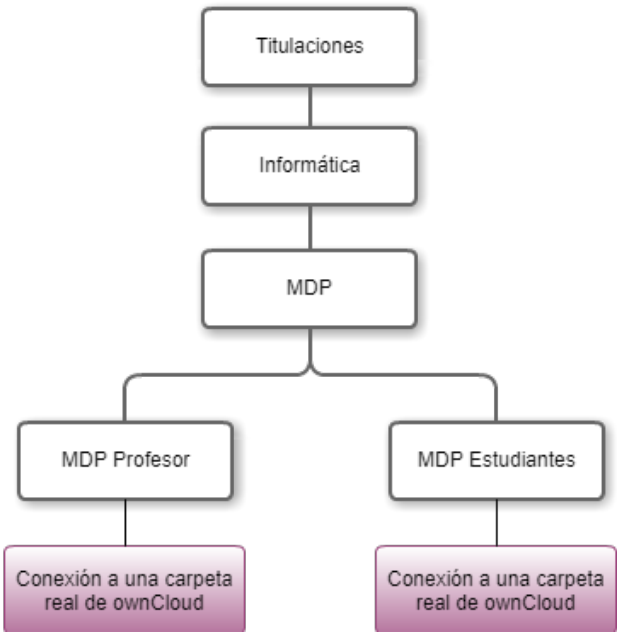

Figura 5.4: Ejemplo de estructura virtual de carpetas.

- Los estudiantes podrán realizar sus propias aportaciones en las carpetas de cada asignatura destinadas a almacenar sus archivos. Los propietarios de esos archivos podrán cambiar sus archivos y eliminarlos. Los documentos que suban los estudiantes serán públicos, pudiendo acceder a ellos todos los usuarios del sistema, incluidos los invitados.
- Los profesores podrán acceder a las carpetas de las asignaturas que imparte, y en ellas podrán subir, modificar y borrar archivos, y crear y borrar carpetas. Estas carpetas estarán disponibles para los grupos con los que las haya compartido, por lo que podrá no ser accesible a todos los usuarios.
- Los miembros del PAS encargados de la gestión de documentos disponibles para la comunidad universitaria, como pueden ser las normativas o la documentación administrativa, dispondrán de sus propias carpetas, en las cuales podrán subir, modificar y borrar archivos, y crear y borrar carpetas. Estas carpetas serán públicas, por lo que cualquier usuario podrá acceder a la documentación existente en su interior, incluido los invitados.

5.5. Desarrollo de la aplicación web

En este Trabajo Fin de Grado se desarrollará una aplicación web para la gestión documental del centro. Para realizar dicha aplicación nos basaremos en el patrón Modelo-Vista-Controlador (MVC).

5.5.1. Patrón MVC

MVC [13] es un patrón de diseño que permite separar la representación interna de la información con la forma en que dicha información se muestra a los usuarios. Para ello MVC

propone la construcción de tres componentes distintos llamados modelo, vista y controlador.

- El **modelo** es el responsable de acceder a la capa de almacenamiento de los datos, definir las reglas de negocio, que es la funcionalidad del sistema, y de proporcionar a la vista la información solicitada.
- La **vista** se encarga de presentar el modelo en un formato adecuado para interactuar, normalmente como interfaz de usuario.
- El **controlador** responde a eventos, normalmente provocados por acciones del usuario, e invoca peticiones al modelo cuando es necesario obtener alguna información. Se podría decir que el controlador hace de intermediario entre la vista y el modelo.

En la figura 5.5[1], se puede observar la colaboración entre los componentes de MVC.

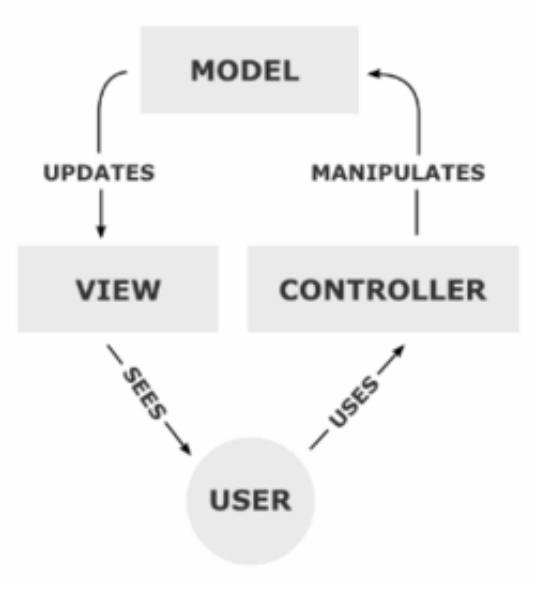

Figura 5.5: Colaboración entre los componentes de MVC.

Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo y mantenimiento de aplicaciones.

5.5.2. Programación por capas

La programación por capas [14] es un modelo de desarrollo software en el que el objetivo principal es la separación de las partes que componen un sistema software o una arquitectura cliente-servidor: la capa de presentación, la lógica de negocio y la capa de persistencia o de datos. Gracias a esta separación, un cambio en cualquiera de las capas no se propaga, por lo que no será necesario revisar las otras capas. En la figura 5.6[2], se puede observar la estructura de este modelo.

[1] Imagen obtenida de https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador
[2] Imagen obtenida de https://es.wikipedia.org/wiki/Programaci%C3%B3n_por_capas

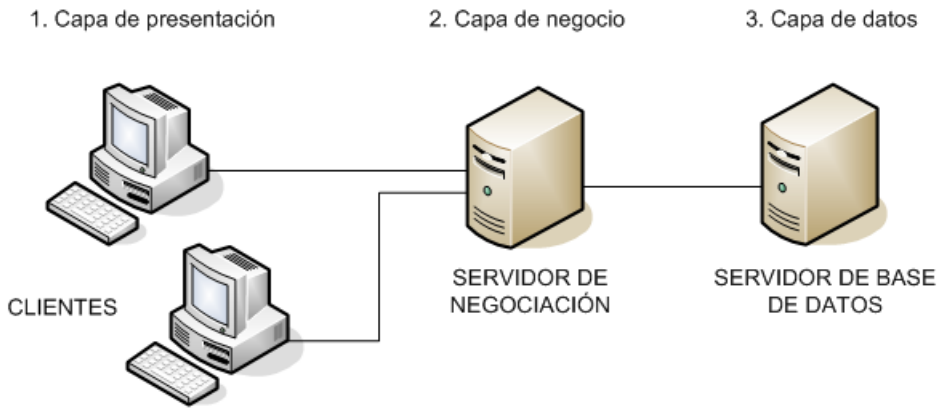

Figura 5.6: Estructura de programación por capas.

5.5.2.1. Capa de persistencia o de datos

La capa de persistencia es la encargada de la gestión de los datos de la aplicación. La capa de persistencia recibe solicitudes de almacenamiento o recuperación de la lógica de negocio e interactúa según estas solicitudes con los distintos gestores de base de datos, como puede ser MySQL.

Existen diferentes tecnologías para la capa de persistencia, destacando las siguientes:

JDBC[15]: Es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo que se use o de la base de datos a la que se accede., utilizando el lenguaje SQL. Es necesario gestionar los valores de los campos y su proyección en tablas de una base de datos relacional, por lo que hay que tratar con dos modelos de datos, lenguajes y paradigmas de acceso a los datos muy diferentes.

La utilización de JDBC presenta una serie de ventajas:

- Es la forma más directa de mandar instrucciones a la base de datos.
- Permite explotar al máximo las funcionalidades de la base de datos.

Pero JDBC presenta una serie de inconvenientes importantes:

- Se necesita un esfuerzo bastante grande para implementar un mapeo entre el modelo relacional utilizado en la base de datos y el modelo de objetos.
- El mantenimiento de la aplicación es mucho más costoso.
- Introduce muchos errores en tiempo de ejecución.
- El desarrollo es mucho más lento.

El algoritmo 5.1 presenta un ejemplo de código con JDBC, en el cual se realiza una consulta a una base de datos para obtener información sobre los alumnos almacenados en ella.

iBATIS[16]: Es un framework de código abierto desarrollado por Apache Software Foundation que se ocupa de la capa de persistencia. Puede ser implementado en Java, .NET y

Algoritmo 5.1 Ejemplo JDBC.

```
try {
    Class.forName("com.mysql.jdbc.Driver");
} catch (ClassNotFoundException ex) {
    log.error("No se encontro el Driver MySQL para JDBC.");
}
Connection cn = DriverManager.getConnection("jdbc:mysql://localhost:3306/cuentas", "root", "contrasena");

// Creamos el Statement para poder hacer consultas
Statement st = cn.createStatement();

// La consulta es un String con código SQL
String sql1 = "SELECT * FROM alumnos";

// Ejecuta una consulta que devuelve resultados
ResultSet rs = st.executeQuery(sql1);
while (rs.next())
{
    System.out.println (rs.getString ("nombre") + " " + rs.getString (apellido)+ " " + rs.getInt(numExpediente));
}
```

Ruby. iBATIS asocia los objetos de modelo (JavaBeans) con sentencias SQL o con procedimientos almacenados mediante ficheros descriptores XML, simplificando la utilización de base de datos.

iBATIS presenta una serie de ventajas [17]:

- Facilita el manejo de la capa de persistencia mediante la utilización de interfaces.
- Es fácil de mantener, realizando una separación entre SQL y el lenguaje de programación de la aplicación.
- Es open source.

No obstante, también existen una serie de inconvenientes:

- Las bases de datos que gestiona iBATIS deben ser exclusivamente relacionales.
- No es totalmente transparente, teniendo que programar en SQL.
- Pierde funcionalidad si casi todas las sentencias SQL son construidas dinámicamente.

El algoritmo 5.2 presenta un ejemplo de código con iBATIS, en el cual se realiza una consulta a una base de datos para obtener la información referente a un alumno a partir de su ID.

Algoritmo 5.2 Ejemplo iBATIS.

```
//Descriptor XML
<select id="getAlumno" resultClass="com.beans.Alumno">
      SELECT
          ID_ALUMNO as id,
          NOMBRE as nombre,
          APELLIDOS as apellidos,
          FECHA_NACIMIENTO as fechaNacimiento,
       FROM ALUMNO
       WHERE ID_ALUMNO = #valor#
</select>

//Código JAVA
Integer claveAlumno = new Integer(1);
Alumno alumno1 = (Alumno) sqlMap.queryForObject("getAlumno", claveAlumno);
```

JPA[18]: Es una API de persistencia desarrollada para la plataforma Java EE. No es un framework, sino una especificación de los principios básicos de gestión de la capa de persistencia[20]. Su misión es no perder las ventajas de la programación orientada a objetos al interactuar con una base de datos y permitir usar objetos regulares. Es necesario la presencia del fichero "persistence.xml", el cual se encarga de conectar la base de datos y definir el conjunto de entidades que se va a gestionar.

JPA presenta una serie de ventajas [19]:

- Utilizar un framework de mapeo objeto-relacional (ORM) simplifica enormemente la programación de la lógica de persistencia.
- El código suele ser mucho más sencillo y mantenible.
- Proporciona independencia de la base de datos, bajo acoplamiento entre la capa de persistencia y lógica de negocio, y un desarrollo rápido.

No obstante, también presenta una serie de inconvenientes:

- No ofrece toda la funcionalidad que ofrecería realizar las consultas nativas.
- Puede presentar una curva de aprendizaje más grande y costosa.

Se puede observar un ejemplo de código con JPA en el algoritmo 5.3, en el cual se intenta almacenar un alumno en la base de datos.

Algoritmo 5.3 Ejemplo JPA.

```
Alumno alumno1 = new Alumno("Jesús","Salguero Serrat","18/08/1994");
EntityManagerFactory emf =Persistence.createEntityManagerFactory("Alumnos");
EntityManager em = emf.createEntityManager();
try {
    em.getTransaction().begin();
    em.persist(alumno1);
    em.getTransaction().commit();
} catch (Exception e) {
    e.printStackTrace();
}finally {
    em.close();
}
```

Hibernate[21]: Es una herramienta de mapeo objeto-relacional (ORM) para la plataforma Java. Se encarga de facilitar el mapeo de atributos entre una base de datos relacional y el modelo de objetos de una aplicación. Este framework gestiona la capa de persistencia a través de ficheros xml o anotaciones.

Hibernate presenta una serie de ventajas [22]:

- Permite diseñar objetos persistentes que podrán incluir relaciones, colecciones y un gran número de tipos de datos.
- Presenta una gran escalabilidad, siendo muy eficiente.
- Es software libre, presentando una licencia LGPL.
- Presenta un mapeo mucho más flexible.
- Facilita las consultas debido a que se realizan con un lenguaje orientado a objetos.

- Permite el mapeo y la validación mediante anotaciones.

Los inconvenientes presentes en Hibernate son los mismos que se encuentran en JPA:

- No ofrece toda la funcionalidad que ofrecería realizar las consultas nativas.
- Puede presentar una curva de aprendizaje más grande y costosa.

En el algoritmo 5.4, se puede observar un ejemplo de código usando Hibernate. Este código es la clase *Alumnos* con las anotaciones de Hibernate.

Algoritmo 5.4 Ejemplo Hibernate.

```
@Entity
@Table(name = "ALUMNOS")
public class Alumnos {

    private Long id;
    private String nombre;

    @Id
    @GeneratedValue
    public Long getId() {
       return id;
    }

    private void setId(Long id) {
       this.id = id;
    }

    @Basic
    @Column(name = "NOMBRE")
    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre= nombre;
    }
}
```

Cada tecnología tiene sus ventajas y limitaciones. Se ha decidido usar Hibernate en el desarrollo de la aplicación debido a que proporciona una solución ORM completa, es muy popular y tiene una amplia comunidad que proporciona soporte para los usuarios.

5.5.2.2. Lógica de negocio

Es la capa que contiene la lógica principal de la aplicación. En ella se establecen las distintas funciones que se pueden ejecutar. Recibe las peticiones de la capa de presentación, procesa la información y le envía las respuestas tras ese proceso.

Esta capa se comunica con la capa de presentación para recibir las solicitudes y presentar los resultados, y con la capa de persistencia para solicitar al gestor de base de datos almacenar o recuperar datos de él.

Existen diferentes tecnologías para la capa de negocio, destacando las siguientes:

Seam[23]: JBoss Seam es un framework desarrollado por JBoss, una división de Red Hat. Seam introduce el concepto de contextos. Cada componente existe dentro de un contexto, por ejemplo, el contexto conversacional captura todas las acciones del usuario hasta que sale del sistema.

Seam presenta las siguientes ventajas [24]:

- Mejora el modelo de persistencia ya que fue diseñado por las mismas personas que crearon Hibernate.
- Usa anotaciones Java.
- Perte crear pruebas unitarias a través de anotaciones, que logran recrear el entorno de forma sencilla.

No obstante, también presenta una serie de inconvenientes:

- Existe buena documentación, pero no es bastante completa y exhaustiva.
- No se centra únicamente en la capa de negocio, sino que engloba desde la capa de presentación a la capa de datos.
- Obliga a usar JSP o Facelets para las vistas.

Enterprise JavaBeans 3[25]: Se trata de una arquitectura para el desarrollo y despliegue de aplicaciones de negocio basadas en componentes. Este proyecto estuvo parado durante unos años, debido a que los desarrolladores decidieron abandonalo debido a la complejidad y a las restricciones de uso. Sin embargo, con decidieron retomarlo, sacarlo la nueva versión EJB 3, simplificando su uso.

EJB presenta las siguientes ventajas:

- Funciona sobre Java, adquiriendo sus ventajas, como puede ser la portabilidad o la orientación a objetos.
- Ahorra trabajo tanto en la fase de diseño como en la de implementación ya que presenta una serie de esquemas que definen los EJBs.
- Fácil integración en aplicaciones web básadas en servlets o JSPs.

No obstante, el uso de EJBs presenta los siguientes inconvenientes:

- Dependencia de la tecnología para el mantenimiento y reutilización de la aplicación. Es muy costoso pasar el trabajo a otro sistema que no incluya esta tecnología.
- La curva de aprendizaje es bastante costosa, ya que el uso de EJBs no es sencillo.

Spring[26]: Es un framework open source que proporciona un marco de trabajo para el desarrollo de aplicaciones J2EE. Este framework está basado en el uso de ficheros planos JavaBeans para la lógica de la aplicación y archivos XML para su configuración.

Spring presenta las siguientes ventajas [24]:

- Spring soporta múltiples tecnologías en la capa de persistencia como son JDBC o Hibernate.
- Spring recomienda el uso de interfaces, por lo que al pasar el trabajo a otro sistema que no utilice Spring basta con reimplementar las interfaces.
- Permite la inyección de dependencias en tiempo de ejecución.

No obstante, Spring presenta algunos inconvenientes:

- No es posible evaluar si un objeto ha sido bien inyectado en tiempo de ejecución.
- El contenedor de Spring no es ligero, por lo que no es aconsejable su uso en aplicaciones de tiempo real o aplicaciones móviles.

Se ha decidido usar Spring para esta aplicación debido a la simplicidad que presenta su uso en Java, únicamente presenta anotaciones en clases Java corrientes, presentando un código más estructurado, limpio y reutilizable. Además, proporciona inyección de dependencias, recibiendo los objetos sus dependencias en tiempo de ejecución, de manera que ellos no son los responsables de la instanciación e inicialización de esas dependencias.

5.5.2.3. Capa de presentación

La capa de presentación es la encargada de mostrar a los usuarios la información y aceptar del usuario distintas entradas. Provee a la aplicación de una interfaz de usuario, que no realiza ningún procesamiento de los datos, relegando esta función a la capa de lógica de negocio. Esta interfaz debe ser entendible y fácil de usar, siendo amigable para el usuario.

Existen diferentes soluciones para la capa de presentación, destacando los siguientes:

Servlets+JSPs[54]: Los servlets son módulos escritos en Java que se utilizan en un servidor y son utilizados para generar contenidos dinámicos en páginas web. Un servlet recibe una petición de un usuario y devuelve un resultado a esa petición, como puede ser el resultado de una búsqueda en una base de datos.

El cliente del servlet realiza las peticiones utilizando una cabecera de petición HTTP. El servlet procesa la petición y realiza las operaciones pertinentes, devolviendo una respuesta en formato HTML al cliente.

JSP es una tecnología que permite mezclar HTML estático con HTML generado dinámicamente. Permite incorporar de forma rápida elementos dinámicos en páginas web, utilizando código Java y una serie de etiquetas determinadas. Este tipo de página pueden ser difíciles de mantener si hay una gran cantidad de código Java, por ello es importante emplear un diseño que separe la generación del contenido dinámico y estático, surgiendo la combinación Servlet+JSP. Así, se usará los servlets para la generación del contenido dinámico y las páginas JSP para la presentación del contenido en formato HTML.

Esta solución es de bajo nivel respecto a las que se describen a continuación, las cuales utilizan internamente servlets de Java, intentando facilitar el manejo de solicitudes y respuestas HTTP.

Apache Struts[27]: Es una herramienta de soporte para el desarrollo de aplicaciones web bajo el patrón MVC en la plataforma Java EE. Su segunda versión ha corregido todas las deficiencias en cuanto a flexibilidad y simplicidad de uso presentes en su primera versión. Aun así, no se trata de un framework bastante conocido debido a la presencia de JSF o Spring MVC. El controlador de Struts funciona mediante acciones, heredando de la clase 'Action', otorgando un nombre simbólico a cada acción.

Apache Struts presenta una serie de ventajas [28]:

- Diseño simplificado.
- Soporte para AJAX.
- Facilidad para la realización de test unitarios.

- Casi todos los elementos de configuración tienen definidos valores por defectos, que pueden ser modificados según la elección del programador.
- Posee arranque rápido, por lo que se pueden realizar cambios sin necesidad de reiniciar el contenedor web.

No obstante, presenta distintos inconvenientes:

- Puede presentar problemas de compatibilidad.
- La documentación sobre Apache Struts es deficiente.

JFS[29]: JavaServer Faces es un framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE. Utiliza JSP como la tecnología que permite hacer el despliegue de las páginas, aunque se puede usar otras como XUL, basado en XML. JSF incluye un conjunto de APIs para representar componentes de una interfaz de usuario y administrar su estado, manejar eventos, validar entradas, definir un esquema de navegación de las páginas y dar soporte para la accesibilidad. Además, añade bibliotecas de etiquetas personalizadas para expresar interfaces JSF en páginas JSP y un modelo de eventos en el lado del servidor.

JSF presenta una serie de ventajas:

- El código JSF es muy parecido al HTML estándar, facilitando su aprendizaje.
- JSF ofrece una separación clara entre el comportamiento y la presentación.
- JSF se nutre de la experiencia de Apache Struts y ha mejorado algunas de sus deficiencias.
- JSF ofrece mayor flexibilidad del controlador y del manejador de eventos que Struts.

No obstante, presenta una serie de inconvenientes [30]:

- Su evolución es más lenta que otras tecnologías al tratarse de un estándar.
- En los últimos años ha descendido su uso y su interés dentro de la comunidad[3].
- JSF gestiona su ciclo de vida a través del servidor, realizando todos los controles en él. Actualmente, otras tecnologías han optado por dejar una gran parte de responsabilidad al cliente, haciendo que JSF pierda parte de interés.

Spring MVC[31]: Es un framework basado en el patrón MVC que ofrece una división clara entre controladores, modelos y vistas. Esta diseñado en torno a un DispatcherServlet que maneja todas las peticiones y respuestas HTTP.

Spring MVC presenta una serie de ventajas:

- Spring MVC ofrece una división clara entre controladores, modelos y vistas.
- Spring MVC no obliga a usar JSP, pudiendo usar diferentes tecnologías de vistas.
- Spring MVC permite adaptar su comportamiento según el tipo de petición realizada.

[3]Estudio sacado de *¿Tiene futuro JSF?* http://www.arquitecturajava.com/tiene-futuro-jsf/

- Spring MVC ofrece una integración con las nuevas arquitecturas móviles y JavaScript MVC más directa que JSF[30].
- Desde su aparición ha tenido muy buena acogida, aumentando continuamente su utilización, documentación y comunidad.

No obstante, también presenta una serie de inconvenientes:

- Posee una curva de aprendizaje mayor.
- No es posible evaluar si un objeto ha sido bien inyectado en tiempo de ejecución.

En el algoritmo 5.5, se puede observar un ejemplo de un controlador realizado con Spring MVC.

Algoritmo 5.5 Ejemplo Spring MVC.

```
@Controller
public class UsuarioController {

    private final UsuariosService usuarioService;

    ...

    @RequestMapping(value= "/Registrarse", method = RequestMethod.GET)
    public String iniciarRegistro(ModelMap model){
        UsuariosVO usuario= new UsuariosVO();
        model.put("user",usuario);
        return "Registro";
    }
    ...
}
```

Se ha decidido utilizar para la realización de la aplicación web el framework Spring MVC, debido a que es uno de lo más populares actualmente, con una gran documentación y comunidad que permite resolver las cuestiones de los usuarios con facilidad. Además, la asignación del método que debe ejecutarse en el controlador es bastante sencilla, cargándose en la URL y anotando dicho método con la URL esperada.

5.6. Conclusión de la fase de diseño

En este Trabajo Fin de Grado, se realizará una plataforma Moodle simulando la existente en la Universidad de Extremadura, una plataforma ownCloud para alojar los archivos de los usuarios, una aplicación web para acceder a esos archivos y un sistema de autentificación único para todas las plataformas utilizando LDAP como base de datos. Además, el acceso a ownCloud, tanto desde la plataforma Moodle como desde la aplicación web a desarrollar, se realizará mediante WebDAV.

En el gestor documental a desarrollar en este Trabajo Fin de Grado existirán diferentes aplicaciones que permitirán al usuario interactuar con él:

- **Aplicación web de Moodle:** Permitirá al usuario introducir en sus aulas/espacios virtuales los archivos existentes en ownCloud.
- **Aplicación web de ownCloud:** La interfaz web proporcionada por ownCloud solo estará disponible para los grupos PDI y PAS. En esta interfaz se pueden subir, modificar

y borrar archivos y carpetas, así como compartirlos con los usuarios y grupos que se desee. Las acciones disponibles en esta interfaz que no se encontrarán en la aplicación web a desarrollar serán la compartición de archivos y el almacenamiento de archivos privados.

- Se ha considerado oportuno que los miembros de estos grupos puedan utilizar ownCloud para almacenar de manera privada sus archivos personales.
- Mediante WebDAV no se puede compartir archivos con otros usuarios o grupos, por lo que esta acción se deberá realizar en esta aplicación.

- **Aplicación web CUMDoc:** Esta será la aplicación web a desarrollar. En ella se creará la estructura virtual de carpetas, que ofrecerá al usuario el acceso al contenido de manera simple y ordenada. Además contará con *"una parte real"* que serán las carpetas que de verdad existen en ownCloud. El acceso a estas carpetas se realizará mediante WebDAV. Las acciones disponibles en esta aplicación dependerá del tipo de usuario que acceda a ella, según se describió en el apartado 5.4.3.

En definitiva, el esquema del sistema a desarrollar en este Trabajo Fin de Grado se puede observar en la figura 5.7.

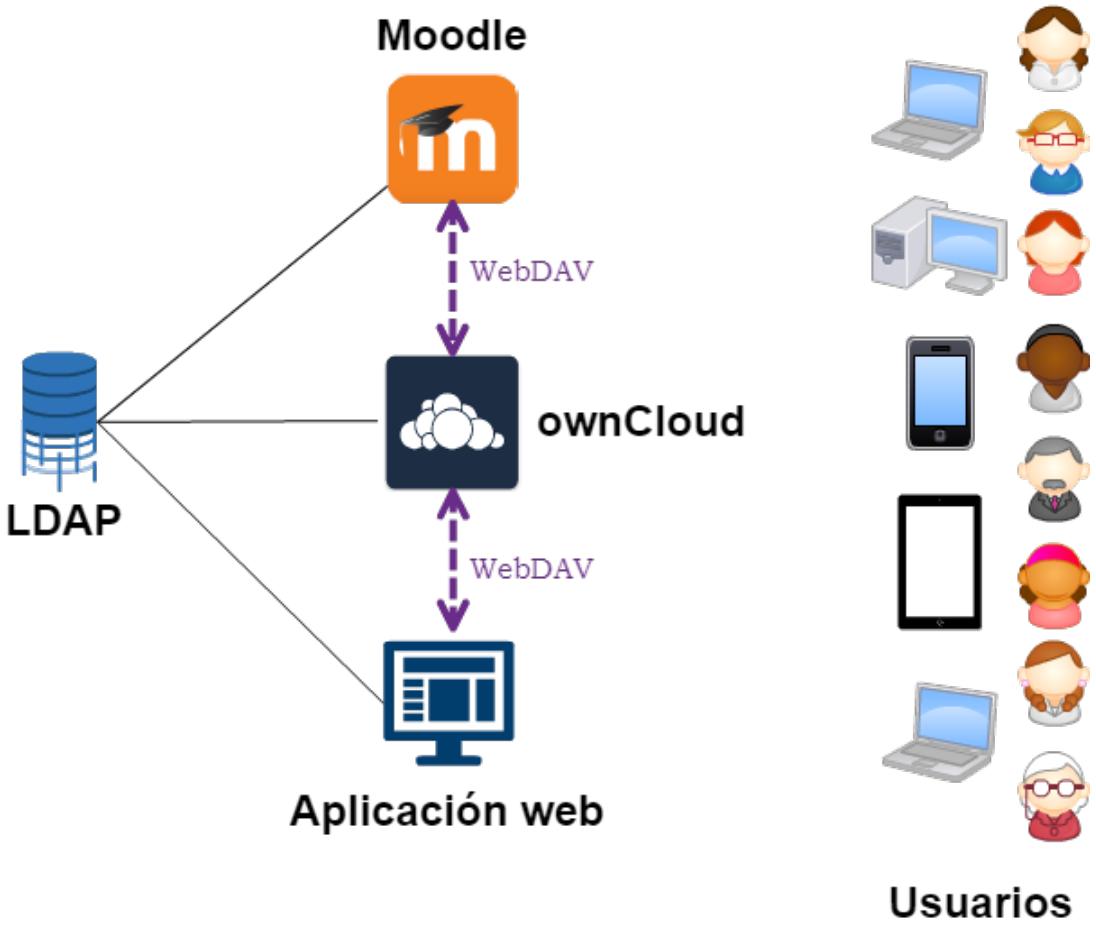

Figura 5.7: Esquema del sistema a desarrollar en este Trabajo Fin de Grado.

Capítulo 6

Implementación

Contenidos

Tras haber realizado el análisis y diseño del sistema a desarrollar, en esta apartado se describirá en detalle su implementación.

En la sección 6.1 se explica la configuración LDAP realizada en el proyecto, analizando los distintos grupos y usuarios creados.

En la sección 6.2 se describe la configuración ownCloud necesaria para el correcto funcionamiento de este proyecto.

En la sección 6.3 se analiza la configuración de Moodle realizada en el proyecto, describiendo los pasos necesarios para integrar LDAP y ownCloud en esta plataforma.

En la sección 6.4 se describe la aplicación web realizada en este proyecto, analizando cada una de las distintas capas que la forman, así como estructura de carpetas empleada.

6.1. Configuración LDAP

En primer lugar, es necesario instalar LDAP. Para ello se debe seguir los pasos que se encuentran en el *Anexo IV. Instalación de LDAP*, en la parte final de este documento. Después

de su instalación, es necesario configurar LDAP, añadiendo los distintos usuarios y grupos que utilizarán el sistema.

6.1.1. Creación de grupos

El primer paso será crear los distintos grupos que existen en el proyecto. Estos grupos son similares a los utilizados por el Campus Virtual de la Universidad de Extremadura para el control y acceso a su plataforma.

Los grupos en LDAP siguen una estructura jerárquica, donde cada nivel superior engloba a los inferiores, por lo que un usuario que pertenezca a un último nivel, pertenecerá también a los superiores a este. En la figura 6.1, se puede observar la jerarquía de grupos definida en este proyecto.

Los grupos creados son los siguientes:

- **Titulación:** Para cada titulación existente en el Centro Universitario de Mérida existirá un grupo diferente, que contendrá a los distintos usuarios de esa titulación, dividiéndose entre coordinadores (grupo Coordinación), miembros (grupo Miembro) o profesores (Grupo profesor). Este grupo a nivel interno del centro, se encargará de la documentación referente a las distintas titulaciones, así como a la generada por las distintas comisiones de calidad y personal perteneciente a la titulación.

- **Dirección:** En este grupo estarán los distintos miembros del equipo directivo del centro, los cuales se encargarán de la documentación propia de este grupo.

- **PDI:** En este grupo estará el equipo docente del centro, los cuales pueden ser divididos según su departamento. Se encargarán de los distintos documentos a nivel interno que utilicen, además de los documentos generados para las asignaturas que imparten en el centro.

- **PAS:** Este grupo está formado por el personal de administración y servicios del centro, dividiéndose en los grupos de Conserjería, Secretaría y otros. Este grupo se encargará de la documentación interna que utilicen, así como la documentación administrativa del centro, como pueden ser la relacionada con prácticas externas o TFGs.

- **Estudiantes:** Este grupo cuenta con los estudiantes matriculados en el centro, y se divide en las distintas titulaciones impartidas en él. No existe un grupo específico para los estudiantes del doble grado, ya que un usuario puede pertenecer a diferentes grupos, por lo que contarán como alumnos de informática y telemática. Los miembros de este grupo podrán subir su propia documentación, como pueden ser apuntes propios, y podrán gestionarla.

- **Administrador:** En este grupo se encontrarán los administradores del sistema. Estos serán los encargados de la creación y gestión de los usuarios, grupos y espacios de ownCloud.

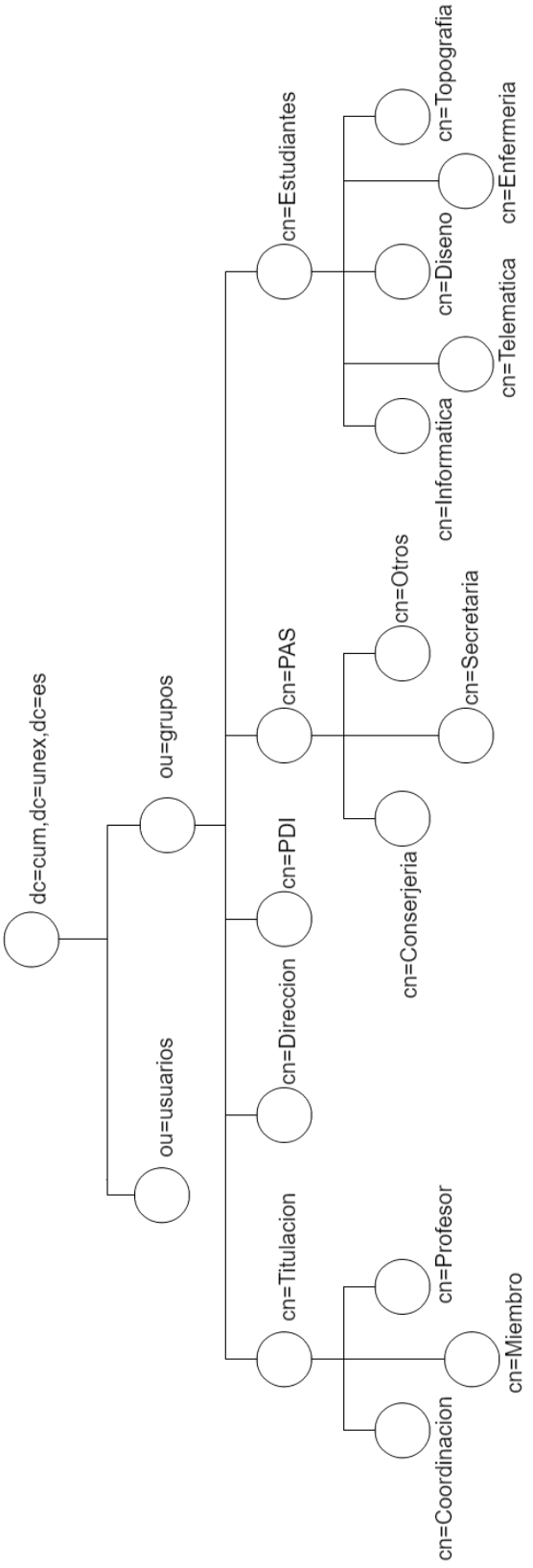

Figura 6.1: Jerarquía de grupos LDAP.

6.1.2. Creación de usuarios

Tal y como se estableció en en análisis, el sistema contará con los siguientes actores: Administrador, Personal Docente y de Investigación (PDI), Personal de Administración y Servicios (PAS), Estudiante y Visitante.

Los usuarios del sistema podrán adquirir estos roles, cada uno con los permisos plasmados en el capítulo de análisis. Además, salvo el perfil de Visitante, los distintos usuarios pertenecerán a los grupos comentados anteriormente, pudiendo pertenecer a uno o varios de ellos, adquiriendo los distintos roles, y por tanto los distintos privilegios con los que cuenta cada perfil.

En la figura 6.2, se puede observar un ejemplo de usuario, llamado *"profesor"*, el cual pertenece al grupo de coordinación de una titulación, es miembro del equipo de dirección y además es profesor del centro, perteneciendo al grupo PDI. Por lo tanto, este usuario, al pertenecer a estos grupos, adquirirá todos estos roles diferentes, teniendo los privilegios de cada uno de ellos. Además, al pertenecer al grupo de coordinación de una titulación, también pertenecerá al grupo de dicha titulación.

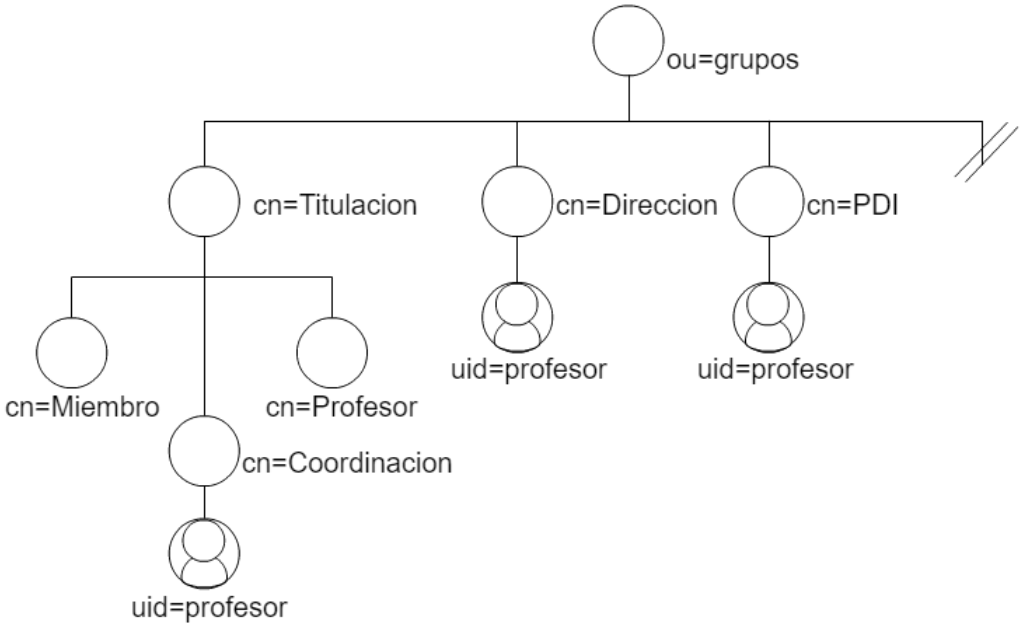

Figura 6.2: Ejemplo de usuario LDAP perteneciente a distintos grupos.

El usuario Visitante tendrá una cuenta dentro del sistema, perteneciendo al grupo *Visitantes*, pero no puede pertenecer a ningún otro grupo. Esta cuenta será a nivel interno, siendo controlada por el administrador, que habilitará la opción de acceder como invitado en la aplicación web a desarrollar. La persona que acceda a la aplicación web y solicite entrar como invitado, utilizará la cuenta Visitante de forma invisible para él, no sabiendo ni el usuario ni la contraseña de este perfil.

6.2. Configuración ownCloud

Para el correcto funcionamiento de ownCloud es necesario instalar XAMPP, teniendo toda la información referente a XAMPP y su instalación en el *Anexo I. Instalación de XAMPP*. Una vez instalado XAMPP es necesario instalar ownCloud siguiendo los pasos que se encuetran al final de este documento, en el *Anexo III. Instalación de ownCloud*.

Después de las instalaciones anteriores, es necesario importar en ownCloud los grupos y usuarios creados en LDAP, para que estos puedan iniciar sesión con las cuentas ya creadas y adquieran los roles ya definidos en LDAP, sin necesidad de configurarlos de nuevo en ownCloud.

También es necesario que el administrador cree las distintas carpetas necesarias para el correcto funcionamiento del sistema, las asigne a los usuarios responsables de ellas y estos elijan los grupos que podrán ver el contenido de ellas.

6.2.1. Configurar LDAP en ownCloud

Para configurar LDAP en ownCloud es necesario instalar en primer lugar la aplicación *LDAP user and group backend*. Los pasos a seguir son los siguientes:

- El administrador del sistema deberá iniciar sesión con su cuenta en ownCloud.
- Deberá desplegar el menú que se encuentra en el lateral superior izquierdo, pulsando en el símbolo con forma de triángulo invertido. Saldrán distintas opciones, siendo una de ellas *Aplicaciones* con el símbolo '+'. Deberá pulsar en esta opción y le redirigirá a una nueva ventana.
- En esta nueva ventana aparecerán las aplicaciones ya instaladas por defecto y una serie de aplicaciones que se pueden habilitar en ownCloud. Se deberá buscar entre las aplicaciones no habilitadas por defecto, la aplicación llamada *LDAP user and group backend*, subida por Dominik Schmidt y Arthur Schiwon con licencia AGPL y declarada oficial por parte de ownCloud.
- Se deberá pulsar en *'Activar'* y esperar a que finalice su instalación. LDAP. Si todo ha finalizado correctamente, se deberá observar la figura 6.3.

Figura 6.3: Activación de la aplicación *LDAP user and group backend*.

El siguiente paso es configurar el servidor LDAP en ownCloud. Para ello, en la esquina superior derecha, se deberá desplegar el menú de opciones y se deberá elegir la opción *Administración*. En la nueva ventana aparecerá los distintos bloques de administración disponibles, apareciendo tras la instalación de la aplicación anterior un nuevo bloque llamado *LDAP*. La

configuración LDAP se compone de seis bloques: *Servidor, Usuarios, Atributos de inicio de sesión, Grupos, Avanzado* y *Experto*. A continuación se describirá la configuración necesaria para la configuración LDAP en ownCloud:

- **Servidor:** En este bloque se deberá introducir toda la información referente al servidor LDAP. Se deberá incluir la IP del servidor, el puerto utilizado (por defecto el 389), el usuario administrador de LDAP, la contraseña de este usuario y la ruta básica del servidor LDAP instalado. Además, se podrá elegir la opción de añadir más de un servidor LDAP a ownCloud. En la figura 6.4, se puede observar la configuración realizada de este bloque para la aplicación a desarrollar en este Trabajo Fin de Grado.

- **Usuarios:** En este bloque se especifica las clases de objetos que son los usuarios de LDAP, es decir, la clase con la que se ha configurado al usuario al darle de alta en LDAP. En esta aplicación las clases son '*inetOrgPerson*' y '*posixAccount*'.

- **Atributos de inicio de sesión:** En este bloque se indica como iniciarán sesión los usuarios LDAP. Hay que señalar la casilla '*Nombre de usuario LDAP*' para que inicien sesión con su nombre de usuario. Si se desea, se podría señalar cualquier otro atributo LDAP para iniciar sesión.

- **Grupos:** En este bloque se especifica las clases de objetos que son los grupos LDAP. En esta aplicación se usará la clase '*posixGroup*'.

- **Avanzado:** En este bloque es necesario configurar la casilla '*Asociación Grupo-Miembro*' para que añada a los usuarios a los distintos grupos LDAP que pertenecen. Para ello hay que seleccionar la opción *'memberUid'*.

Tras todo esto, deberá aparecer '*Configuración correcta*' si consigue conectarse a LDAP.

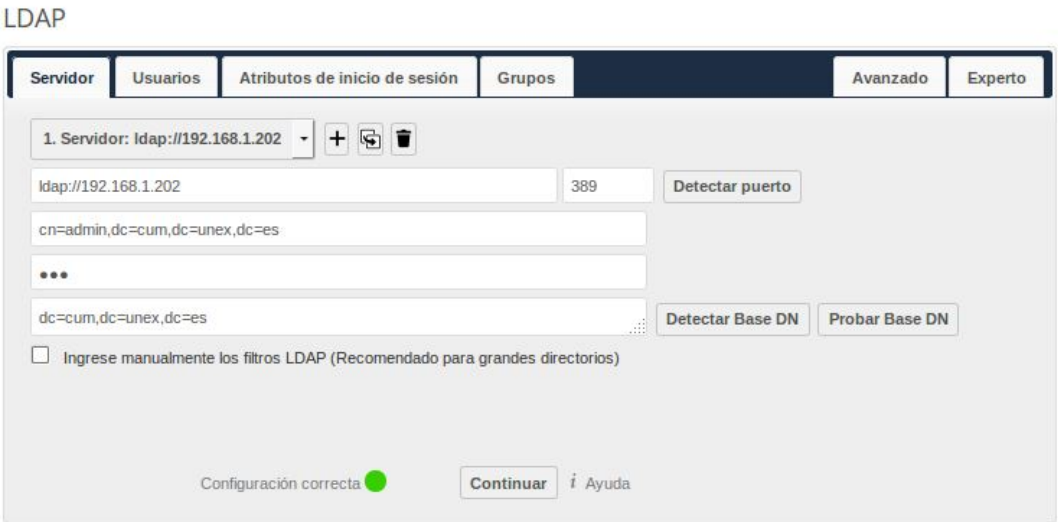

Figura 6.4: Configuración LDAP en ownCloud.

6.2.2. Creación de las carpetas de ownCloud

El administrador del sistema es el responsable de crear la estructura de directorios de ownCloud. Para facilitar la labor del administrador y ofrecer una solución organizada y simple, se ha decidido organizar las distintas carpetas pertenecientes a las asignaturas impartidas en el centro según su titulación, curso y semestre, como se puede observar en la figura 6.5, que se corresponde con una parte de dicha estructura.

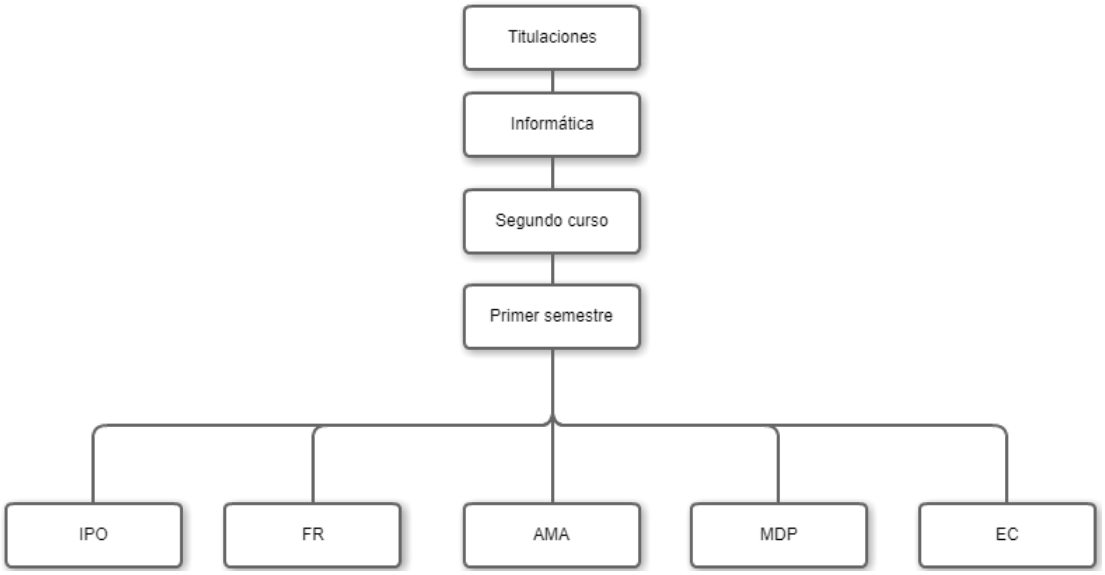

Figura 6.5: Ejemplo de estructura de carpetas ownCloud.

Los nodos hoja de la estructura de carpetas creada en ownCloud pertenecerán a los usuarios encargados de su gestión, que serán miembros de los grupos PDI para las carpetas de las asignaturas que tengan asignadas en su POD, y PAS para las carpetas relacionadas con la parte administrativa del centro. Para esta asignación el administrador compartirá la carpeta en cuestión con privilegios de edición al usuario correspondiente. Una vez que a este usuario le aparezca la carpeta en su cuenta, podrá realizar todas las acciones que quiera en ella y compartirlas con los usuarios y grupos que desee.

En la figura 6.6, se puede observar el proceso de asignación y compartición de carpetas. Para realizar esta acción basta con pulsar en el icono resaltado en la figura en un recuadro negro (icono de compartir que acompaña a la palabra *"Compartiendo"*). Tras pulsar en él, aparecerá en el lateral derecho información sobre los usuarios y grupos con lo que está compartiendo esa carpeta. Si introducimos un nombre en el recuadro de búsqueda de usuario o grupo aparecerán nombres relacionados con lo que está escribiendo. En la figura 6.6, se puede observar que se ha introducido la palabra *"infor"* y aparece debajo *"Informática (Grupo)"*. Si se selecciona el grupo automáticamente se compartirá con dicho grupo.

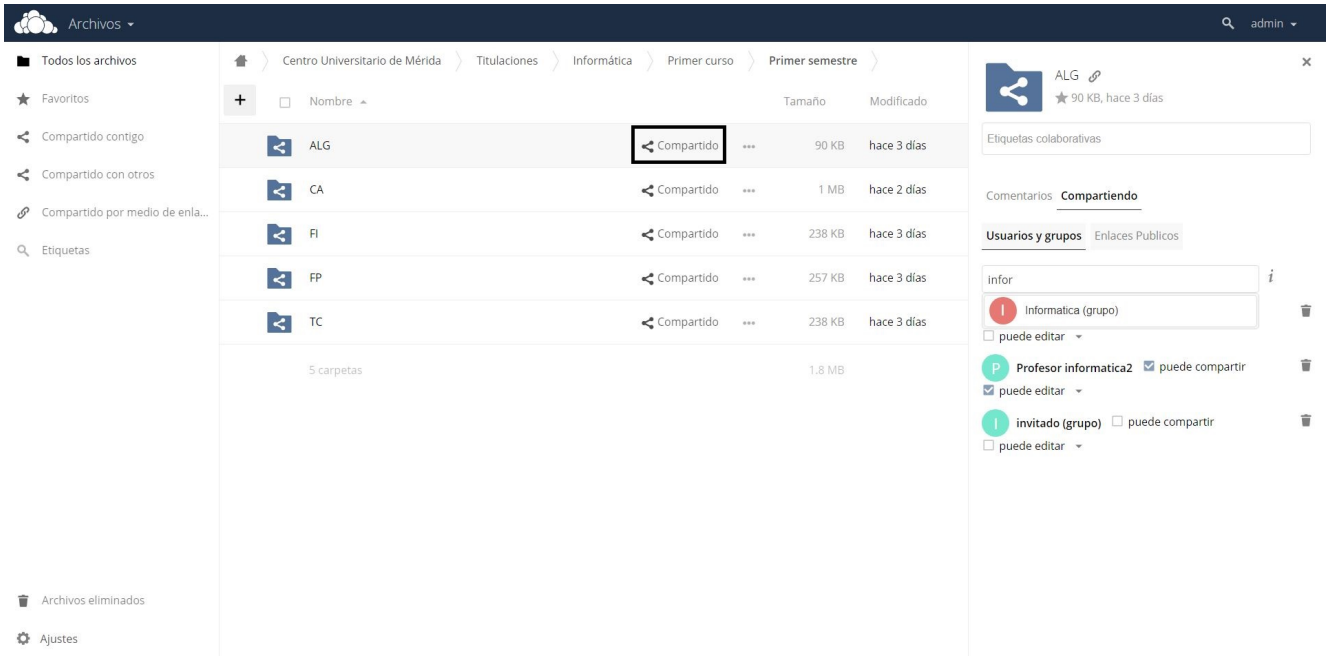

Figura 6.6: Compartir una carpeta en ownCloud.

Además de las carpetas de las asignaturas y las relacionadas con la administración del centro, se ha creado una carpeta en la cual los estudiantes del centro subirán sus aportaciones relacionados con las asignaturas. Esta carpeta estará compartida con todos los grupos del sistema, teniendo únicamente los estudiantes derecho de edición. Las carpetas relacionadas con la administración del centro también serán públicas, teniendo el derecho de edición los usuarios encargados de su gestión.

Las carpetas de las asignaturas impartidas en el centro son las únicas que aparecerán de manera privada en la configuración inicial. Solamente los profesores encargados de esas carpetas podrán ver su contenido y acceder a ella. Estos profesores deberán elegir con qué grupos de los creados en el sistema (Figura 6.1) deciden compartir su contenido, teniendo distintas opciones:

- Se podrá compartir con el grupo entero de estudiantes, por lo que todos los estudiantes del centro podrán ver el contenido.
- Se podrá compartir con una titulación específica, por lo que solo los estudiantes pertenecientes dicha titulación podrán ver el contenido. Por ejemplo, si se decide compartir únicamente con el grupo *'Informática'*, solo los estudiantes de la titulación de informática podrán ver la carpeta y su contenido.
- Se podrá compartir con el grupo de invitados, permitiendo que el contenido sea libre y pueda acceder a él cualquier usuario del sistema, sea miembro o no del centro.

6.3. Configuración Moodle

En el *Anexo II. Instalación de Moodle* se encuentra toda la información necesaria para la instalación de la plataforma Moodle. Después de su instalación, es necesario configurar LDAP

en Moodle para que los usuarios puedan acceder a todas las plataformas con los mismos datos, consiguiendo un sistema de autentificación único. Además, es necesario vincular ownCloud como repositorio en Moodle, para que los distintos miembros del equipo docente puedan vincular los archivos existentes en ownCloud en sus aulas virtuales.

6.3.1. Configuración LDAP en Moodle

Para configurar LDAP en Moodle es necesario acceder a la administración de la plataforma pulsando en el botón '*Administración del sitio*', que se encuentra situado en el lateral izquierdo del menú del administrador. Una vez cargada la nueva ventana, se deberá pulsar en el bloque '*Extensiones*', y dentro seleccionar la opción '*Gestionar identificación*' dentro del apartado '*Identificación*'.

En la nueva ventana aparecerán las distintas opciones de identificación presentes en Moodle. Se debe buscar la opción '*Usar un servidor LDAP*' y pulsar el icono del ojo tachado para habilitar esta función. Se actualizará la página y saldrá ya como una de las opciones activas. El siguiente paso es pulsar en el botón '*Configuración*' y ajustar los parámetros necesarios del servidor LDAP:

- **URL del host:** Dirección del servidor LDAP. En este proyecto es *ldap://192.168.1.105*
- **Nombre distinguido:** Nombre del administrador LDAP. En este proyecto es *cn=admin,dc=cum,dc=unex,dc=es*
- **Contraseña:** Contraseña del administrador LDAP.
- **Tipo de usuario:** Tipo de usuario creado en LDAP. En este proyecto es *posixAccount (rfc2307)*
- **Contextos:** Lugar donde se encuentran los usuarios en LDAP. En este proyecto es *ou=usuarios,dc=cum,dc=unex,dc=es*
- **Mapeado de datos:** Consiste en asociar campos de datos de Moodle con campos de datos de los usuarios LDAP. Los mínimos obligatorios son Nombre (*givenName*), Apellidos (*sn*) y Dirección de correo (*mail*).

El resto de parámetros deben tener los valores por defecto. Tras esto los usuarios podrán acceder a Moodle con los mismos datos que en el resto de plataformas.

6.3.2. Configuración ownCloud en Moodle

En Moodle aparecerá por defecto como repositorio la plataforma ownCloud montada en este proyecto, con acceso como invitado. Por lo tanto, se podrán vincular todos aquellos archivos que hayan sido compartidos con la cuenta '*Invitado*' directamente a las distintas aulas virtuales sin que el profesor tenga que configurar nada por su parte. Para ello el administrador del sistema ha configurado dicho repositorio siguiendo los pasos siguientes:

- Primero se debe acceder al bloque de administración de Moodle y pulsar la pestaña de extensiones. Una vez dentro se debe pulsar el botón '*Gestionar repositorios*' en el apartado '*Repositorios*'.

- En la nueva ventana se debe buscar la opción *'Repositorio WebDAV'* y pulsar la opción de '*Activado y visible*'.
- Aparecerá una nueva ventana en la cual preguntará si se permite a los usuarios agregar repositorios dentro de un curso y dentro del contexto de usuario. Se debe seleccionar la primera opción, únicamente dentro de un curso.
- Tras realizar el paso anterior, aparecerá de nuevo la página de gestión de repositorios, con la opción de WebDAV activada. El siguiente paso es crear una instancia de repositorio por defecto, para ello se debe pulsar el botón '*Configuración*'.
- En la nueva ventana se debe seleccionar la opción '*Crear una instancia de repositorio*'. Tras esto aparecerá un formulario con los datos de Configuración WebDAV como se puede observar en la figura 6.7.

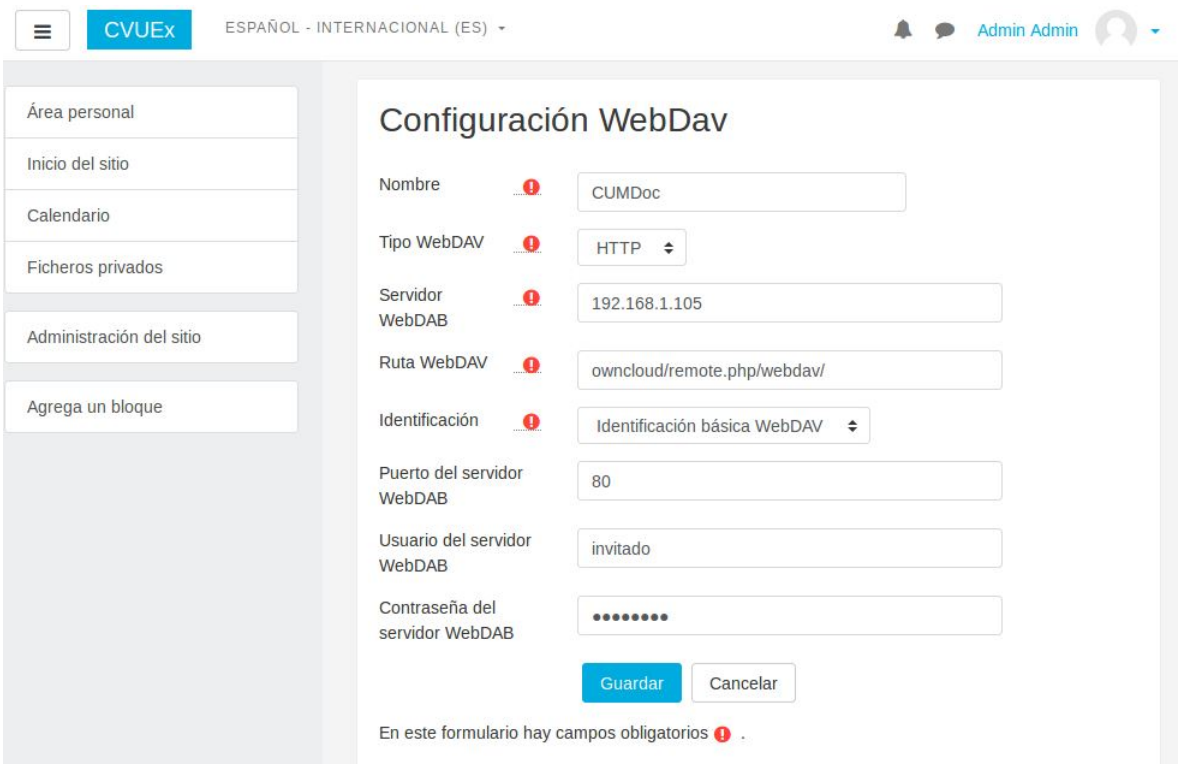

Figura 6.7: Configuración de un repositorio WebDAV en Moodle.

Al acabar esta configuración, los usuarios podrán enlazar los archivos de ownCloud desde su propias aulas virtuales. Para ello, a la hora de agregar un nuevo archivo al aula, en el selector de archivos aparecerá como opción CUMDoc y los archivos alojados en ownCloud, como se puede observar en la figura 6.8.

Además, los usuarios podrán agregar su propio repositorio ownCloud para poder enlazar sus propios documentos no compartidos. Para ello deberán pulsar en la opción '*Repositorios*', disponible en el bloque de administración de su aula virtual. La configuración deberá ser la misma que la anterior, cambiando el usuario y la contraseña por los suyos personales.

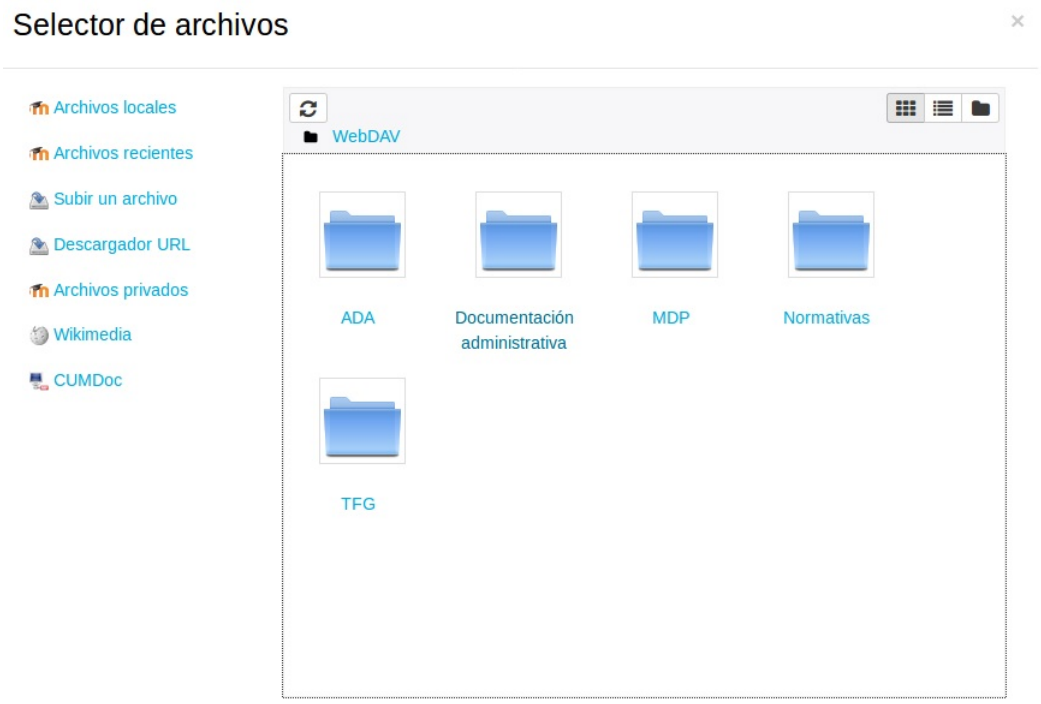

Figura 6.8: Selector de archivos en Moodle con la opción CUMDoc habilitada.

6.4. Implementación de la aplicación web

La aplicación web realizada en este proyecto está dividida en las diferentes capas introducidas en el capítulo anterior. A continuación, se dividirá la implementación de la aplicación en las distintas capas, analizando cada una de ellas en profundidad.

6.4.1. Capa de persistencia

Para el correcto funcionamiento de la aplicación web se necesita almacenar la información de las distintas asignaturas impartidas en el centro, así como recuperarla cuando sea necesaria. Para ello se decidió usar Hibernate como tecnología de la capa de persistencia.

Se decidió crear una clase llamada '*AsignaturasVO*', la cual definiese los atributos que son necesarios almacenar en la base de datos. Esta clase incorporaría además las distintas anotaciones de Hibernate para gestionar la capa de persistencia correctamente.

En el algoritmo 6.1, se muestra parte de la clase '*AsignaturasVO*', destacando lo siguiente:

- La anotación *@Entity* sirve para identificar la clase como una entidad, por lo que se mapeará directamente con una tabla de la base de datos.
- La anotación *@Table* indica el nombre que recibirá la tabla de la base de datos, siendo '*ASIGNATURAS*' la elegida para este proyecto.
- Se han elegido los siguientes atributos para definir las asignaturas del centro:
  - **ID:** ID de cada asignatura.

Algoritmo 6.1 Clase *'AsignaturasVO'* con anotaciones de Hibernate.

```
@Entity
@Table(name = "ASIGNATURAS")
public class AsignaturasVO {

    private Long id;
    private String siglas;
    private String nombre;
    private String titulacion;
    private String semestre;
    private String curso;

    @Id
    @GeneratedValue(generator = "increment")
    @GenericGenerator(name = "increment", strategy = "increment")
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    @Column(name = "SIGLAS")
    public String getSiglas() {
        return siglas;
    }

    public void setSiglas(String siglas) {
        this.siglas = siglas;
    }

    ...
}
```

  - **Siglas:** Indica las siglas con la que se identifica una asignatura. Por ejemplo, la asignatura *Metodología y Desarrollo de Programas* se identifica con las siglas *MDP.*
  - **Nombre:** Nombre completo de la asignatura.
  - **Titulación:** Titulación a la que pertenece la asignatura.
  - **Curso:** Curso académico en el que se imparte la asignatura dentro de la titulación.
  - **Semestre:** Semestre en el que se imparte la asignatura.

- La anotaciones *@Id, @GeneratedValue* y *@GenericGenerator* harán que se genere automáticamente un ID a cada asignatura de manera incremental y que este se utilice como clave primaria de la tabla.
- La anotación *@Column* indica el nombre que recibirá esa columna en la tabla de la base de datos.

Para la realización del proyecto se ha utilizado el patrón de diseño DAO, que promueve la encapsulación del código de acceso a datos en un objeto independiente. Por lo tanto, se utiliza este patrón para encapsular los métodos de acceso a los datos de la aplicación, realizando una nueva clase llamada *'AsignaturasDAO'* y su interfaz correspondiente.

Algoritmo 6.2 Clase *'AsignaturasDAO'*.

```
@Component
@Transactional
public class AsignaturasDAOImpl implements AsignaturasDAO {

    @PersistenceContext
    protected EntityManager entityManager;

    public AsignaturasVO create(AsignaturasVO asignatura) {
        this.entityManager.persist(asignatura);
        return asignatura;
    }

    public AsignaturasVO read(long id) {
        return this.entityManager.find(AsignaturasVO.class, id);
    }

    public AsignaturasVO update(AsignaturasVO asignatura) {
        return this.entityManager.merge(asignatura);
    }

    public void delete(AsignaturasVO asignatura) {
        asignatura = this.entityManager.merge(asignatura);
        this.entityManager.remove(asignatura);
    }

    public AsignaturasVO leerPorSigla(String sigla) {
        return this.entityManager.createQuery("from AsignaturasVO where sigla =
        '"+sigla+"'", AsignaturasVO.class)
                .getSingleResult();
    }

    public List<AsignaturasVO> listaDeAsignaturas(String titulacion, String curso,
    String semestre){
        return entityManager.createQuery("from AsignaturasVO where
        titulacion='"+titulacion+"' and curso='"+curso+"' and
        semestre='"+semestre+"'", AsignaturasVO.class).getResultList();
    }
}
```

En el algoritmo 6.2, se puede observar parte de la clase *'AsignaturasDAO'*. En ella se encuentran los diferentes métodos que interaccionan con la base de datos gracias a un objeto *EntityManager*, encargado de gestionar un conjunto de entidades y hacer transacciones con la base de datos. Estos métodos permiten crear, obtener, modificar y borrar asignaturas. De entre los métodos existentes cabe destacar *'listaDeAsignaturas'*, que permite obtener una lista de asignaturas a partir de la titulación, del curso y del semestre.

6.4.2. Lógica de negocio

Esta capa contiene la lógica principal de la aplicación, siendo la capa intermedia entre la capa de persistencia y la capa de presentación.

En esta aplicación, su única función es enlazar las otras dos capas, ya que no se realiza ningún tratamiento especial de los datos. Conviene aún así seguir dividiendo y separando los distintos niveles, para obtener independencia una capa de otra y por si en el futuro se debe realizar algún tratamiento en los datos.

El algoritmo 6.3, es una parte de la clase *'AsignaturaServiceImpl'*, en el cual aparece un constructor que recibe por parámetros *AsignaturasDAO* a través de inyección de dependencias, y el método usado para obtener la lista de asignaturas a partir de la titulación, del curso y del semestre.

Algoritmo 6.3 Clase *'AsignaturasServiceImpl'*.

```
@Component
public class AsignaturasServiceImpl implements AsignaturasService {

    final private AsignaturasDAO asignaturasDAO;

    @Autowired
    public AsignaturasServiceImpl(AsignaturasDAO asignaturasDAO){
        this.asignaturasDAO=asignaturasDAO;
    }

    ...

    public List<AsignaturasVO> listaDeAsignaturas(String titulacion, String curso,
    String semestre){
        return this.asignaturasDAO.listaDeAsignaturas(titulacion, curso, semestre);
    }
}
```

6.4.3. Capa de presentación

La capa de presentación es la encargada de proporcionar al usuario la información necesario y aceptar del usuario distintas entradas. Para el desarrollo de esta aplicación se decidió usar como framework Spring MVC.

6.4.3.1. Controladores

Los controladores son los encargados de preparar el modelo, que son los datos manejados por la aplicación, y de seleccionar la vista que mostrará el modelo al usuario.

SpringMVC utiliza anotaciones como las tecnologías vistas en los puntos anteriores. Para declarar una clase como controlador basta con anotarla con *@Controller*. Cada método del controlador atiende una serie de peticiones. Para ello hay que indicarle la URL y el tipo de operación que debe atender.

El algoritmo 6.4 muestra los métodos del controlador encargados del inicio de sesión de los usuarios en la aplicación. Es importante destacar:

- La anotación *@RequestMapping* contiene la URL y el tipo de operación que el método debe atender. En este caso ambos métodos actuarán cuando la URL sea '*<Dominio o IP>/iniciarSesion*', pero la primera se ejecutará cuando se trate de una operación GET y la segunda cuando se trate de una operación POST.
- Los dos métodos finalizan de manera diferente:
    - El primer método carga la vista *'IniciarSesion.jsp'*, que mostrará al usuario un formulario donde debe ingresar su nombre de usuario y contraseña para iniciar sesión.

Algoritmo 6.4 Métodos del controlador para iniciar sesión en la aplicación.

```
@RequestMapping(value = "/iniciarSesion", method = RequestMethod.GET)
public String iniciarSesion(ModelMap model) {
    model.put("error", false); //Indica que no se ha producido ningun error de autentificacion
    return "IniciarSesion";
}

@RequestMapping(value = "/iniciarSesion", method = RequestMethod.POST)
public String comprobarInicioSesion(@RequestParam(value = "user", required = true) String usuario,
        @RequestParam(value = "contrasena", required = true) String contrasena, ModelMap model,
        HttpSession session) {
    sardine = SardineFactory.begin(usuario, contrasena); //Inicio de la conexion WebDAV
    List<DavResource> res=new LinkedList<DavResource>();
    try {
        res = sardine.list(PATH);  //Se intenta establecer la conexion realizando una peticion
    } catch (IOException e) {
        return "redirect:/iniciarSesionError"; //Si la peticion causa error se redirigira
                                               //al metodo de inicio de sesion erroneo
    }
    /*Como por WebDAV no podemos averiguar el grupo al que pertenece
    el usuario realizamos la consulta a LDAP*/
    String server="ldap://192.168.56.101"; // servidor de LDAP
    String dn="cn="+usuario+",ou=usuarios,dc=cum,dc=unex,dc=es"; // Ruta del Arbol LDAP
    String tipoAth="simple";//tipo de autentificacion simple o por SSL
    ldapAuth ldapAuth=new ldapAuth(server,dn,tipoAth,usuario,contrasena);
    if(ldapAuth.isAutenticado()){ //Conseguimos conectarnos a LDAP
        Attribute atr=ldapAuth.cargarPropiedadConexion("gidNumber"); //Obtenemos el grupo
        try {
            session.setAttribute("grupo", atr.get().toString()); //Guardamos el grupo en la sesion
        } catch (NamingException e) {
            e.printStackTrace();
            model.put("error","Problema con la conexion a LDAP.");
            return "Error";
        }
    }
    /*Si se pudo establecer la conexion WebDAV y obtuvimos el grupo guardamos el usuario y
    la conexion WebDAV en la sesion*/
    session.setAttribute("user", usuario);
    session.setAttribute("sardine", sardine);
    return "redirect:/contenidoVirtual/Inicio/Titulaciones";
}
```

  - El segundo método, en cambio, redirecciona a otro método del controlador, por ello lleva previamente la palabra '*redirect*'.

- El segundo método, intenta establecer la conexión WebDAV con la plataforma ownCloud y si se produce alguna excepción redireccionará al método que espere la URL '*iniciarSesionError*' para mostrar al usuario que el nombre de usuario y/o contraseña introducidos no son correctos. Tras esto,intenta conectar a LDAP para averiguar los grupos a los que pertenece el usuario. Si se logra la comunicación por WebDAV y la petición a LDAP, se almacena el nombre de usuario, el grupo y la conexión WebDAV en la sesión y se redirecciona al método encargado de mostrar los contenidos de la aplicación.

6.4.3.2. WebDAV

En algunos métodos del controlador se deben realizar distintas operaciones WebDAV para interaccionar con ownCloud y obtener y/o mandar archivos a la plataforma. Existe una librería open source llamada '*Sardine*' [53], disponible en GitHub y desarrollada en Java, que permite

interactuar con un servidor WebDAV. Esta librería permite realizar las operaciones básicas como PUT, GET, DELETE, entre otras.

Para poder utilizar esta librería en un proyecto Maven es necesario añadir su dependencia en el *'pom.xml'* del proyecto. En el algoritmo 6.5, se puede observar la dependencia necesaria para utilizar *Sardine*.

Algoritmo 6.5 Dependencia Maven *Sardine*.

```
<dependency>
    <groupId>com.github.lookfirst</groupId>
    <artifactId>sardine</artifactId>
    <version>5.7</version>
</dependency>
```

Para utilizar *Sardine* en el controlador es necesario declarar un objeto de esta clase y establecer los parámetros necesarios para la comunicación con el servidor WebDAB. Estos parámetros serán el usuario y la contraseña con la que se accede a ownCloud, como se puede observar en el algoritmo 6.6.

Algoritmo 6.6 Declaración de un objeto *Sardine*.

```
Sardine sardine = SardineFactory.begin(usuario, contrasena);
```

Tras esto ya se puede realizar las distintas operaciones disponibles. *Sardine* utiliza para todas las operaciones el *PATH* o ruta donde debe realizar la operación. En el algoritmo 6.7, se puede observar las distintas operaciones realizadas en la aplicación.

Algoritmo 6.7 Operaciones de *Sardine* realizadas en la aplicación.

```
String PATH = "http://192.168.1.126/owncloud/remote.php/webdav/";
List<DavResource> res=new LinkedList<DavResource>();
res = sardine.list(PATH);
...
if(sardine.exists(PATH)){
...
}
...
DavResource archivo=sardine.get(PATH);
...
sardine.put(PATH+nombreArchivo,archivo);
...
sardine.delete(PATH+nombreArchivo);
...
```

Es importante destacar:

- *Sardine* devuelve objetos tipo *DavResource*, los cuales contienen información sobre el archivo buscado en ownCloud, como su nombre, su ruta, la fecha de última modificación, el tipo de archivo, entre otros.
- *sardine.list(PATH)* devuelve los archivos existentes en la ruta como una lista de *DavResources*.
- *sardine.exists(PATH)* devuelve si existe o no el archivo indicado en la ruta.

- *sardine.get(PATH)* recupera un único archivo solicitado como un objeto *DavResource*.
- *sardine.put(PATH,archivo)* sube a la plataforma ownCloud el archivo en la ruta indicada.
- *sardine.delete(PATH)* borra el archivo indicado en la ruta.

6.4.3.3. Vistas

Las vistas son las encargadas de mostrar el modelo en un formato adecuado al usuario, por lo que suele denominarse interfaz de usuario.

CUMDoc utiliza JavaServer Pages (JSP) para sus vistas. Esta tecnología [54] simplifica el desarrollo de páginas web dinámicas, incorporando de forma rápida elementos dinámicos utilizando Java. Las páginas JSP se componen de código HTML junto con código Java incrustado en la página y una serie de etiquetas especiales, que se ejecutarán en el servidor y darán como resultado de la ejecución una página HTML. Gracias a esta tecnología se mostrará al usuario el contenido de ownCloud obtenido por WebDAV.

En el algoritmo 6.8, se puede observar parte de la interfaz creada, en la cual el código Java se encuentra entre las etiquetas '<%%>'. Este código se encarga de redireccionar al usuario a la página de inicio de sesión en caso de que no se encuentre logueado e intente acceder a una página que lo necesite.

Algoritmo 6.8 Comprobación de sesión iniciada.

```
<%
    if (session.getAttribute("user") == null) {
%>
<jsp:forward page="/iniciarSesion" />
<%
    } else {
%>
<%@ include file="menu.jsp"%>
...
<%
    }
%>
```

Además, en el algoritmo 6.8, se puede observar el código '<%@ *include file="menu.jsp"* %>' cuya función es incorporar la cabecera con el menú en todas las páginas de la aplicación. Como es código repetido, se ha guardado en un fichero aparte, simplificando el mantenimiento de la aplicación y reduciendo la duplicidad de código.

Uno de los aspectos más destacables de la interfaz de CUMDoc es la capacidad de adaptarse a cualquier dispositivo, siendo una aplicación web responsive. El diseño responsive [55] de páginas web comprende una serie de técnicas y pautas que permiten adaptar las distintas páginas web al entorno de navegación del usuario. Para el desarrollo de la aplicación se ha utilizado el framework Bootstrap, uno de los más populares y con mayor documentación.

Bootstrap [56] es un framework web de código abierto que proporciona plantillas de HTML, CSS y JavaScript para construir componentes responsives, como columnas, botones, formularios entre otros. Existen dos formas de utilizar Bootstrap en las aplicaciones web. La primera es descargarse los ficheros de Bootstrap e importarlos dentro del proyecto. La segunda es obtenerlo vía CDN, que ofrece soporte para Bootstrap y será la utilizada para este proyecto.

CDN [58] es una red de servidores diseñados para proporcionar contenido web, en este caso los distintos ficheros de Bootstrap. Para usarlo es necesario adjuntar las líneas del algoritmo 6.9, en las páginas de la aplicación, exactamente en el *head* [57].

Algoritmo 6.9 Configuración de Bootstrap.

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
crossorigin="anonymous">

<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css"
integrity="sha384-rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/Sp"
crossorigin="anonymous">

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa"
crossorigin="anonymous"></script>
```

Una vez incluido en las páginas, para usarlo solamente es necesario aplicar sus clases a los distintos elementos HTML. Por ejemplo, para que un botón sea responsive es necesario asignar la clase '*btn*' de Bootstrap.

En CUMDoc se ha utilizado el framework Bootstrap así como las tecnologías HTML5 y CSS3 para que la interfaz web funcione correctamente en todas las plataformas. A continuación se verán unos ejemplos de su uso en la aplicación web desarrollada.

El algoritmo 6.10, es parte de la página de inicio de sesión de los usuarios que se corresponde con la figura 6.9.

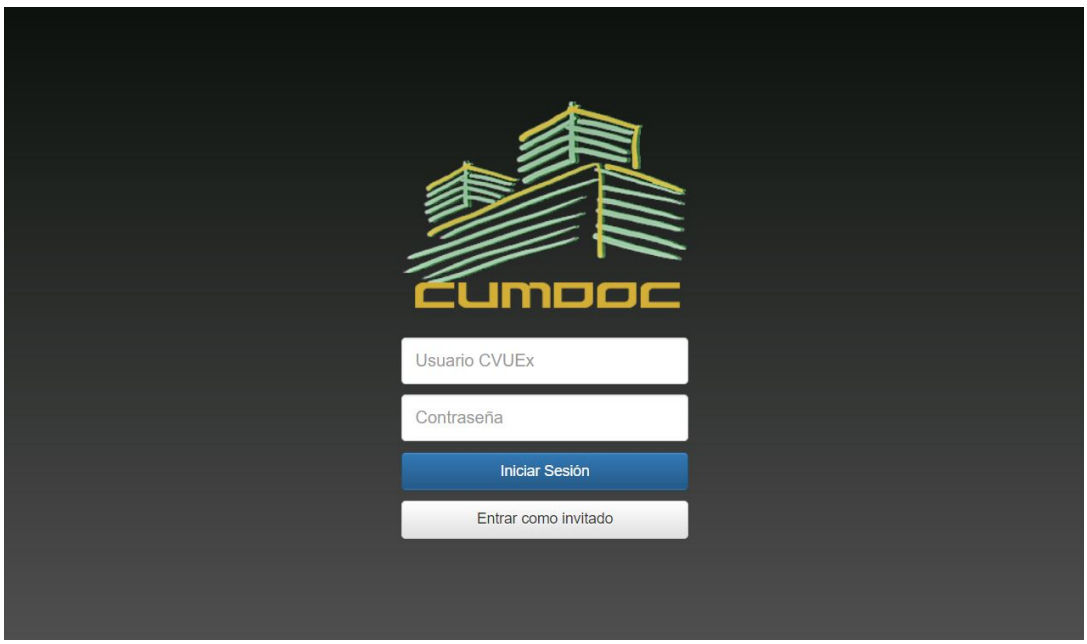

Figura 6.9: Interfaz de CUMDoc. Formulario de inicio de sesión.

Algoritmo 6.10 Formulario de inicio de sesión.

```
<div class="container">
    <div class="card card-container">
        <img id="profile-img" class="img-responsive"
            src="/TFG_jesalguer/resources/images/logo3.png" /> </br>
        <form action="iniciarSesion" method="post" class="form-signin">
            <c:if test="${error==true}">
                <div class="alert alert-danger" role="alert">
                    <span class="glyphicon glyphicon-exclamation-sign"
                        aria-hidden="true"></span> <span class="sr-only">Error:</span>
                    Usuario y/o contraseña no validos.
                </div>
            </c:if>
            <input type="text" name="user" id="inputEmail" class="form-control"
                placeholder="Usuario CVUEx" required autofocus> <input
                type="password" name="contrasena" id="inputPassword"
                class="form-control" placeholder="Contraseña" required>

            <button class="btn btn-primary btn-block" type="submit">Iniciar
                Sesión</button>
            <a class="btn btn-default btn-block" href="iniciarSesionInvitado"
                role="button">Entrar como invitado</a>
        </form>
    </div>
</div>
```

Es importante destacar:

- Todos los elementos Bootstrap deben ir recogidos dentro de un *div* con la clase *'container'*.
- La imagen que se encuentra en la parte superior del formulario (el logotipo del Centro Universitario de Mérida) tiene asignada la clase *'img-responsive'* para que su posición varíe dependiendo del dispositivo con el que se acceda.
- La sección de código comprendido entre las etiquetas *<c:if>* sirve para mostrar un mensaje al usuario que ha introducido erróneamente su usuario y contraseña. Se activará cuando el controlador indique que se ha producido un error.
- Los *input* tienen asignada la clase *'form-control'* de Bootstrap que permite que se adapten según el dispositivo con el que acceda el usuario. Tienen además el atributo *required* de HTML5 que obliga a que los campos del formulario no se queden vacíos.
- Tanto el elemento *button* como el elemento *a* tienen asignada la clase *'btn'* de Bootstrap para que se adapten según el dispositivo. Además, cuentan con diferentes opciones de la clase *'btn'* que modifican la apariencia del elemento.

El algoritmo 6.11, es parte de la página donde se muestran los distintos archivos de ownCloud que se corresponde con la figura 6.10.

Algoritmo 6.11 Página para mostrar archivos de ownCloud.

```
<c:forEach var="contenidos" items="${contenidos}">
    <hr class="stylehr">
    <div class="row">
        <c:choose>
            <c:when
                test="${contenidos.getContentType().split('/')[1].equals('unix-directory')}">
                <div class="col-md-10">
                    <span class="glyphicon glyphicon-folder-open folders"></span> <a
                        class="btn btn-link cortar"
                        href='/TFG_jesalguer/calculoRutaReal/<c:out value="${rutaVirtual}"/>
                        /<c:out value="${ruta}"/>/<c:out value="${contenidos.getName()}"/>/
                        <c:out value="${contenidos.getContentType().split('/')[1]}"/>
                        /<c:out value="${tipoUsuario}"/>'> <c:out value="${contenidos.getName()}" />
                    </a>
                </div>
            </c:when>
            <c:otherwise>
                <c:choose>
                    <c:when
                        test="${contenidos.getContentType().split('/')[1].equals('pdf')}">
                        <div class="col-md-8">
                            <span class="glyphicon glyphicon-file folders"></span> <a
                                class="btn btn-link cortar" target="_blank"
                                href='/TFG_jesalguer/calculoRutaReal/<c:out value="${rutaVirtual}"/>
                                /<c:out value="${ruta}"/>/<c:out value="${contenidos.getName()}"/>
                                /<c:out value="${contenidos.getContentType().split('/')[1]}"/>
                                /<c:out value="${tipoUsuario}"/>'><c:out value="${contenidos.getName()}"/>
                                </a>
                        </div>
                    </c:when>
                    <c:otherwise>
                        <div class="col-md-8">
                            ...
                            <a class="btn btn-link cortar"
                                href='/TFG_jesalguer/calculoRutaReal/<c:out value="${rutaVirtual}"/>
                                /<c:out value="${ruta}"/>/<c:out value="${contenidos.getName()}"/>
                                /<c:out value="${contenidos.getContentType().split('/')[1]}"/>
                                /<c:out value="${tipoUsuario}"/>'> <c:out value="${contenidos.getName()}"/>
                            </a>
                        </div>
                    </c:otherwise>
                </c:choose>
                <div class="col-xs-4 col-md-2">
                    <a class="btn btn-link foldersR"
                        href='/TFG_jesalguer/calculoRutaReal/<c:out value="${rutaVirtual}"/>
                        /<c:out value="${ruta}"/>/<c:out value="${contenidos.getName()}"/>
                        /other/<c:out value="${tipoUsuario}"/>'>
                        <span class="glyphicon glyphicon-download-alt"></span> Descargar</a>
                </div>
            </c:otherwise>
        </c:choose>
        <c:if test="${editar==true}">
            <div class="col-xs-4 col-md-2">
                <a class="btn btn-link foldersR"
                    href='/TFG_jesalguer/contenidoPropio/<c:out value="${rutaVirtual}"/>
                    /<c:out value="${ruta}"/>/<c:out value="${contenidos.getName()}" />
                    /<c:out value="${tipoUsuario}"/>/borrarFicheroPDIoPAS'>
                    <span class="glyphicon glyphicon-trash"></span> Eliminar
                </a>
            </div>
        </c:if>
    </div>
</c:forEach>
```

Es importante destacar:

- Se trata de un bucle, donde cada iteración se tratará un contenido de ownCloud. Con la

Figura 6.10: Interfaz de CUMDoc. Página donde se visualizan los documentos de ownCloud.

clase *'row'* se indica que cada archivo será una fila en la página web.

- Dentro de * * se realiza una elección dependiendo si se trata de una carpeta (unix-directory), un PDF (pdf) u otro tipo de archivo mediante las etiquetas *<c:when>*. Dependiendo del tipo de archivo, se obtendrá un icono u otro (por ejemplo, si se trata de una carpeta se utilizará el icono de una carpeta).

- Si se trata de una carpeta, al pulsar en ella, se llamará al controlador encargado de realizar una petición WebDAV para obtener los archivos que contiene esa carpeta. Además, si es el usuario encargado de gestionar la carpeta aparecerá la opción de eliminar la carpeta.

- Si se trata de un archivo, su nombre ocupará 8 columnas, y la opción de descargar 4 columnas en pantallas pequeñas y 2 en pantallas superiores. Si es el encargado de la carpeta aparecerá la opción de *'Eliminar'*, que ocupar´las mismas columnas que la opción de descargar. Entre todas las opciones se sumarían el total de columnas con las que trabaja Bootstrap. Si se pulsa el nombre del documento y se trata de un archivo PDF, se abrirá en una nueva ventana el documento previsualizado sin descargar, en cualquier otro caso comenzaría la descarga.

6.4.4. Estructura virtual de carpetas

Como se describió en el sección 5.4 del capítulo de Diseño, es necesario crear una estructura virtual de carpetas en la aplicación web. La estructura implementada se puede observar en la figura 6.11. Esta estructura de carpetas se compone, por lo tanto, de una parte virtual y una parte real existente en ownCloud. Es importante destacar:

- La estructura virtual comienza en una raíz de *Inicio*, la cual se divide en las ramas de *Titulaciones*, *Documentación administrativa*, *Normativas* y *TFG*. Salvo la rama de *Titulaciones*, las otras están directamente conectadas a carpetas ownCloud con su mismo nombre, que son administradas por los usuarios pertinentes. Al acceder a una de ellas

se realizará una petición mediante WebDAV y se recuperará el contenido de dichas carpetas.

- La rama de *Titulaciones* se divide en las distintas carreras impartidas en el centro. Cada una de estas carreras estarán formadas los los cuatro años correspondientes, así como los distintos semestres. Una vez accedido a una titulación, a un curso y semestre aparecerán las distintas asignaturas que se imparten según esos criterios. Al acceder a una de estas asignaturas podrán aparecer dos carpetas. La primera carpeta se llamará como la asignatura más la palabra "Alumnos" , y contendrá todas las aportaciones que han hecho los distintos alumnos a la asignatura. La segunda carpeta contendrá el contenido aportado por el profesor de la asignatura y puede ser que dependiendo de los criterios del profesor aparezca o no. El profesor puede decidir si compartir la carpeta solo con sus alumnos, con todos los estudiantes del centro y/o con los invitados. Una vez accedido a una de estas dos carpetas se realizará una petición mediante WebDAV y se obtendrá el contenido de la carpeta.

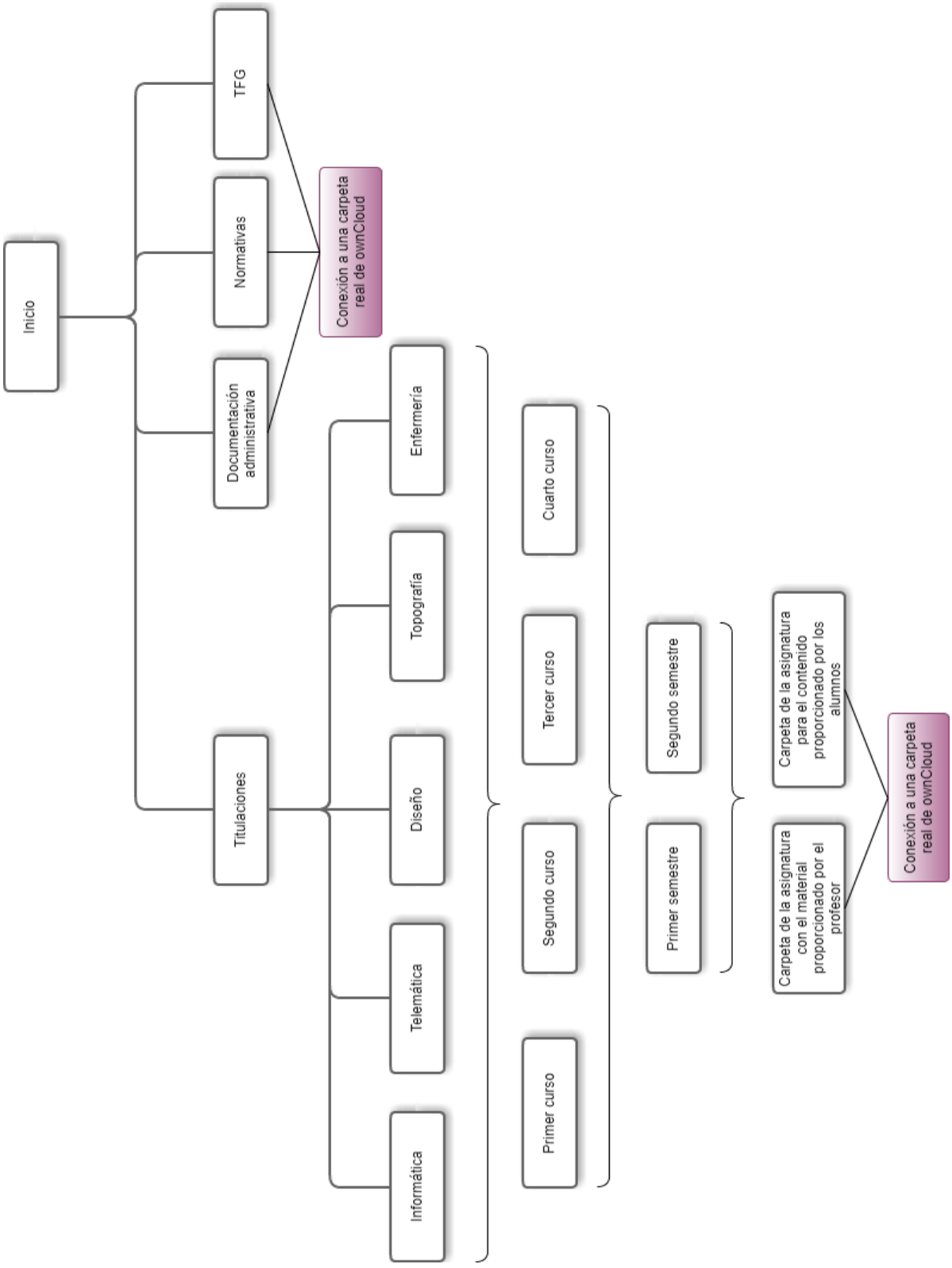

Figura 6.11: Estructura de carpetas de CUMDoc.

Capítulo 7

Manual de usuario

Contenidos

En este capítulo se explica cómo se usa el sistema de gestión documental CUMDoc desde los diferentes puntos de vistas de cada uno de los usuarios de la aplicación.

En la sección 7.1 se encuentra el manual de usuario de la aplicación web de ownCloud.

En la sección 7.2 se encuentra el manual de usuario de la aplicación web de Moodle.

En la sección 7.3 se encuentra el manual de usuario de la aplicación web CUMDoc.

7.1. Aplicación web de ownCloud

OwnCloud proporciona a los usuarios una aplicación web desde la cual pueden gestionar sus archivos privados y archivos compartidos. Esta aplicación está pensada para los usuarios *Administrador, PDI* y *PAS* de nuestro sistema. Todos los usuarios pueden realizar las mismas acciones que a continuación se describen, salvo el perfil de *Administrador*, que cuenta con una serie de añadidos.

7.1.1. Acceso a la aplicación

Para acceder a la aplicación web proporcionada por ownCloud es necesario utilizar la URL *http://<IPservidorOwnCloud>/owncloud* desde un navegador web. Al acceder aparecerá la pantalla de inicio de sesión, en la cual el usuario deberá introducir su usuario y su contraseña como se puede observar en la figura 7.1.

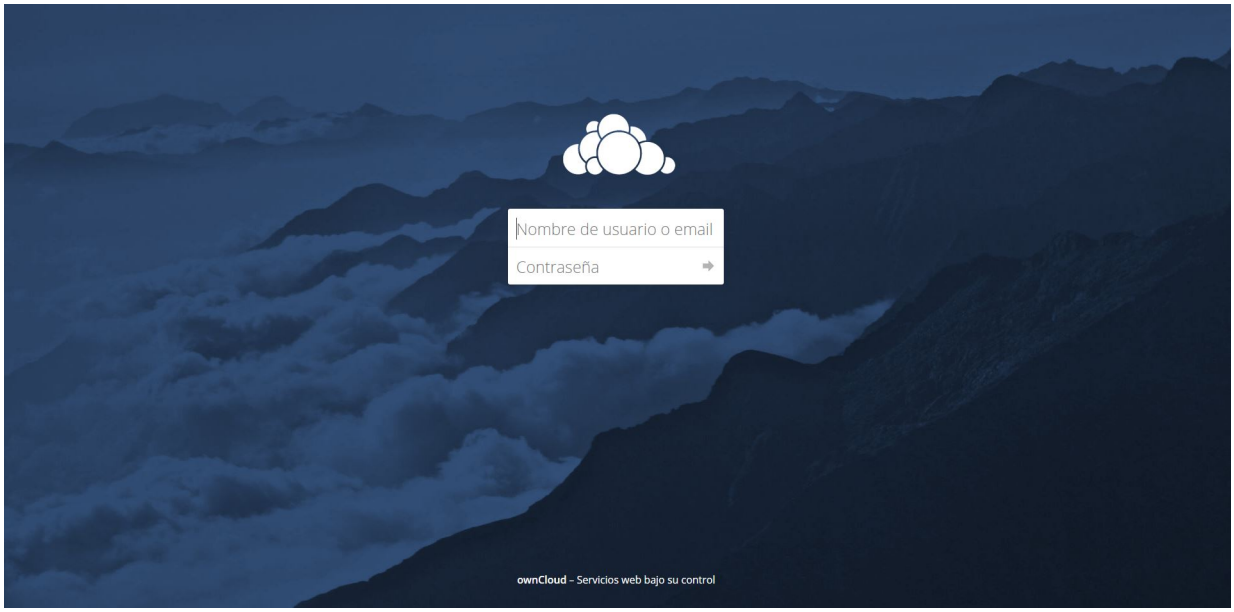

Figura 7.1: Inicio de sesión en la aplicación de ownCloud.

Al acceder aparecerá la pantalla principal de la aplicación.

7.1.2. Pantalla principal de la aplicación

En la figura 7.2 se puede observar la pantalla principal de la aplicación de ownCloud. Esta figura contiene una enumeración destacando las diferentes partes por la que está compuesta:

1. En esta parte se encuentra la barra de **navegación de la aplicación**. Según se seleccione se podrá ver los archivos que se encuentran en cada una de las secciones. Por ejemplo, si se selecciona *'Todos los archivos'* se podrán ver todos los archivos que se encuentran alojados en el sistema, si se selecciona *'Compartido contigo'* se verán únicamente los archivos que han compartido contigo otros usuarios del sistema.

2. Aquí se encontrará los **archivos propios de cada usuario**. Según el tipo de archivo se podrán visualizar, descargar o acceder a ellos. En esta sección se podrá compartir los archivos con otros usuarios, ver los detalles del archivo, eliminarlos y cambiar su nombre.

3. Si se pulsa en el **botón** '+' se podrá:

   a) Crear una carpeta nueva. Para ello se debe seleccionar la opción *'Carpeta'* y aparecerá un recuadro para insertar el nombre de la nueva carpeta. Tras escribir el nombre y pulsar la tecla Enter aparecerá la nueva carpeta creada.

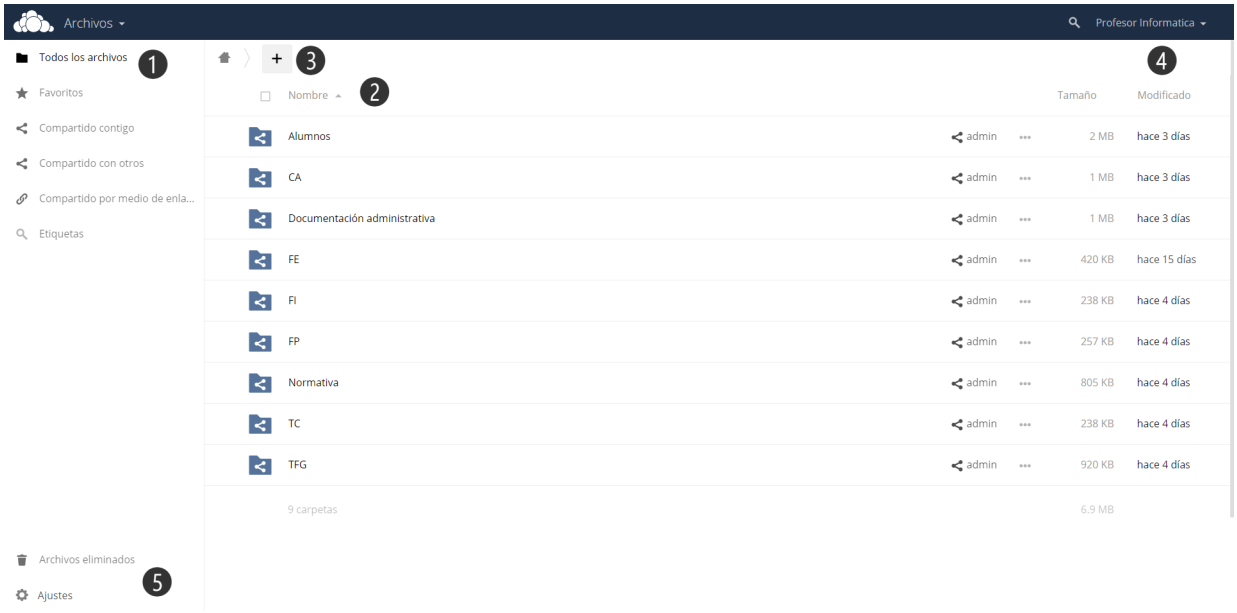

Figura 7.2: Pantalla principal de la aplicación de ownCloud.

b) Subir un archivo nuevo. Para ello se debe seleccionar la opción *'Subir'* y aparecerá una nueva ventana en la cual hay que seleccionar el archivo del equipo del usuario que se desea subir. Tras elegir el archivo y pulsar la opción de subir, aparecerá el nuevo archivo del usuario.

4. En el **menú superior** se podrá:

   a) Buscar archivos dentro del sistema, pulsando el botón con forma de lupa y escribiendo el nombre del archivo.

   b) Ver los ajustes de la aplicación, donde aparecerán los datos personales del usuario, grupos a los que pertenece y podrá cambiar el idioma de la interfaz web, como se puede observar en la figura 7.3.

   c) Ver la ayuda que ofrece ownCloud sobre su interfaz web.

   d) Salir de la aplicación.

5. En esta parte de la aplicación se puede:

   a) Consultar los archivos eliminados.

   b) Si se pulsa en el botón '*Ajustes*' aparecerá el enlace que se debe poner en los clientes WebDAV, como puede ser Moodle a la hora de configurar un repositorio.

7.1.3. Compartir archivos

Los usuarios de ownCloud pueden compartir archivos y carpetas con otros usuarios de la plataforma. Para realizar esta compartición deberán pulsar el icono de compartir (3 puntos conectados). Una vez pulsado dicho icono, se abrirá una nueva pestaña en el lateral derecho

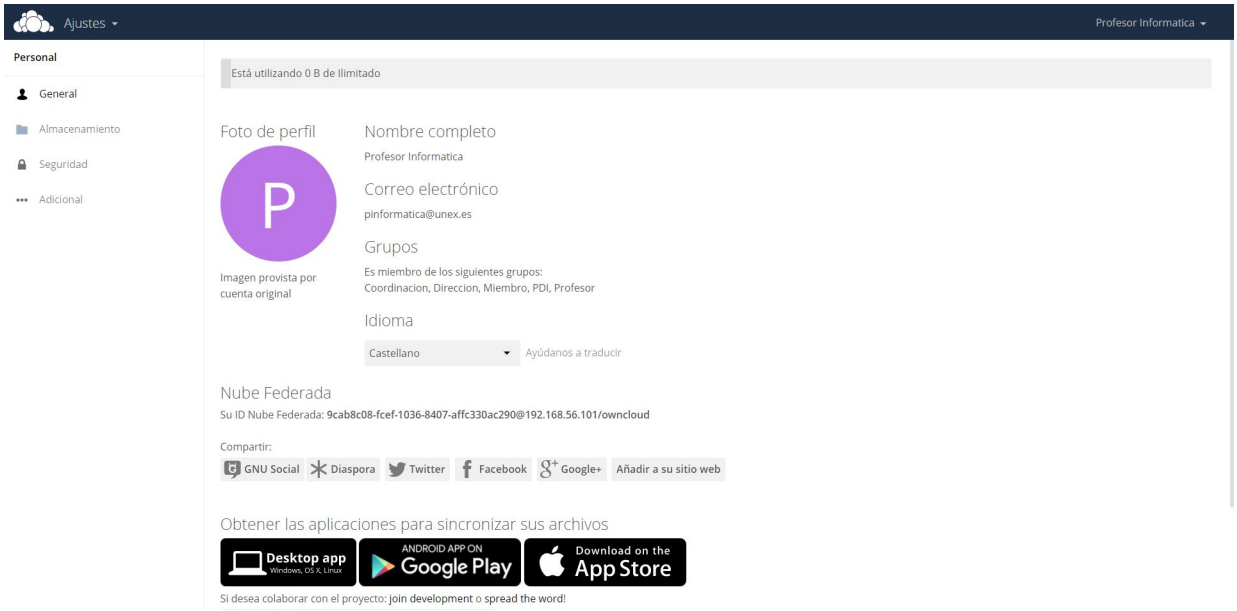

Figura 7.3: Pantalla de ajustes de la aplicación de ownCloud.

con la información sobre el archivo o carpeta a compartir y los usuarios y/o grupos con los que ya se ha compartido si los hubiese. Para compartir el archivo o carpeta basta con escribir el nombre del usuario o grupo en el formulario que se encuentra en la parte inferior de esta nueva pestaña. Una vez se empieza a escribir el nombre del usuario o grupo irán apareciendo distintas posibilidades relacionadas con lo que se está escribiendo. Una vez aparezca el usuario o grupo con el que queremos compartir el archivo lo seleccionamos y automáticamente se añadirá a la lista de usuarios y grupos con los que esta compartido dicho archivo. Esta lista se encuentra en la parte inferior de la nueva pestaña, apareciendo además las distintas posibilidades de compartición:

- **Puede compartir:** Permite al usuario o grupo compartir el archivo o carpeta con otros usuarios de la aplicación.
- **Puede editar:** Esta opción está formada por:
  - **Crear:** Permite al usuario o grupo crear archivos dentro de la carpeta si lo que se ha compartido es un directorio.
  - **Cambiar:** Permite al usuario o grupo actualizar el documento.
  - **Eliminar:** Permite al usuario o grupo poder eliminar el archivo o la carpeta y su contenido.

En la figura 7.4, se puede observar como la carpeta *'Centro Universitario de Mérida'* ha sido compartida con el grupo *'Estudiantes'* permitiendo que compartan la carpeta y que puedan crear contenido dentro de ella, pero no modificarlo ni eliminarlo.

7.1.4. Acciones especiales del administrador

A parte de las acciones anteriormente descritas, el administrador cuenta con una serie de acciones disponibles solo para su perfil.

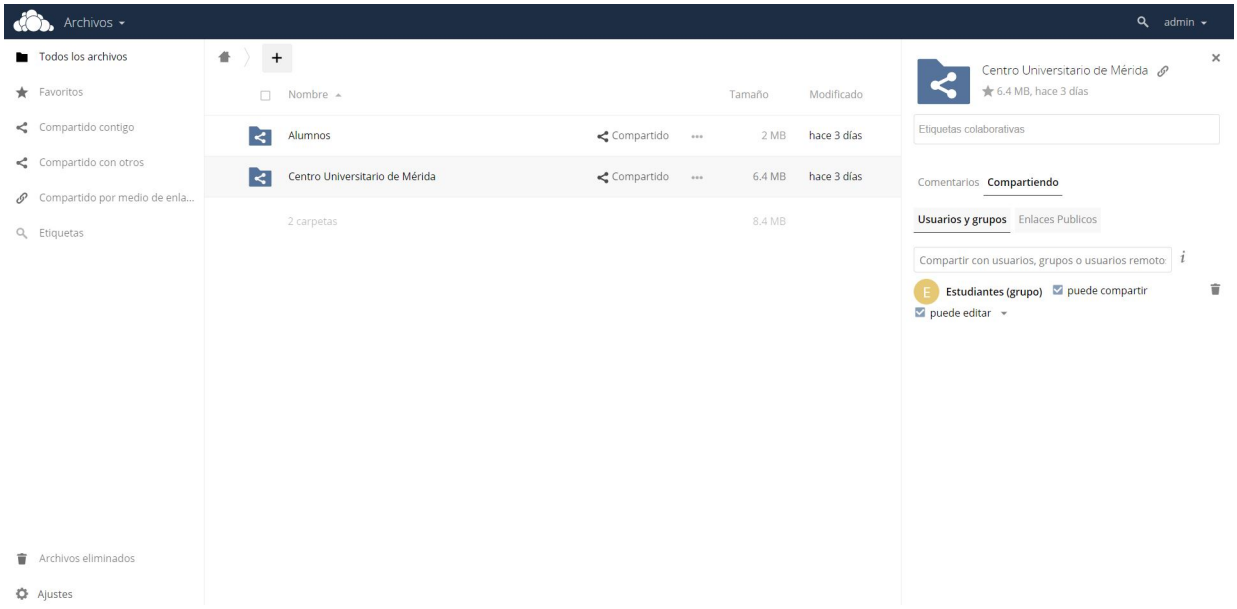

Figura 7.4: Compartir una carpeta de ownCloud con un grupo y limitar sus permisos.

7.1.4.1. Instalar aplicaciones

El administrador tiene la opción de acceder al *'Market'*, un lugar donde la comunidad realiza sus aportaciones para mejorar y añadir funcionalidades a ownCloud. Para ello deberá pulsar el menú despegable que se encuentra en el lateral superior izquierdo de la aplicación y seleccionar *'Market'*. Una vez realizado los pasos anteriores, aparecerá una nueva pantalla con las diferentes aplicaciones adicionales, como se puede observar en la figura 7.5.

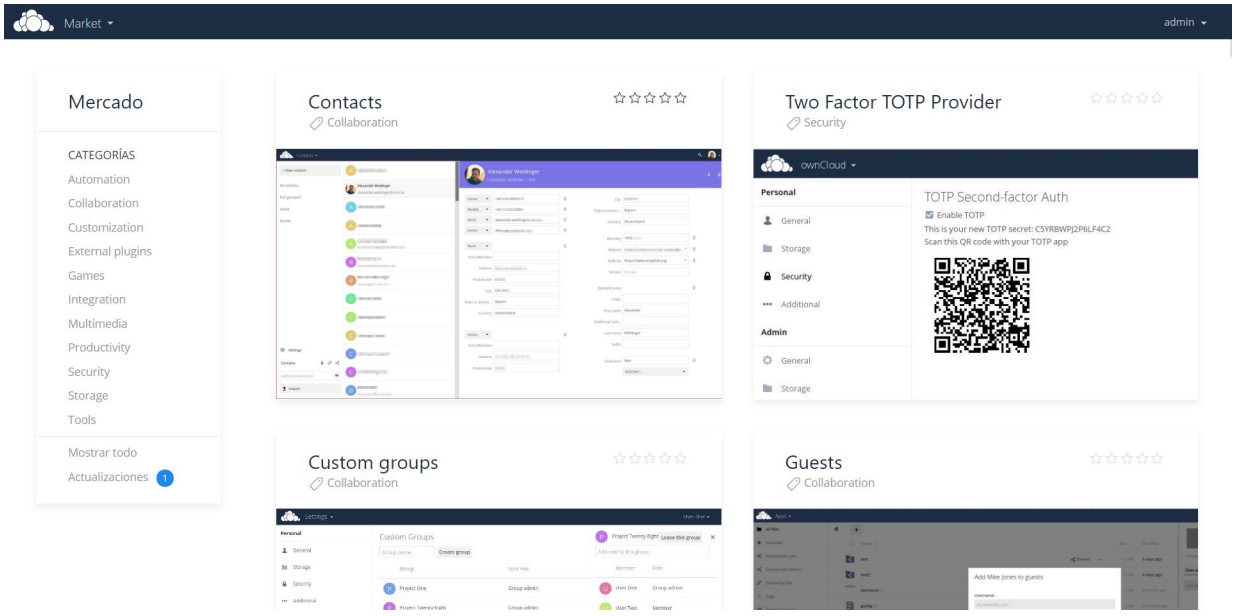

Figura 7.5: Market de ownCloud.

En esta página se encuentra un gran número de aplicaciones, como la aplicación *'LDAP user and group backend'* que permite importar los usuarios y grupos de LDAP, *'Contacts'* que permite almacenar información sobre contactos como si se tratase de una agenda personal, o *'Collabora Online'*, que permite editar simultáneamente documentos de texto entre distintos usuarios.

Si se pulsa sobre cualquier aplicación, se abrirá una nueva ventana con la información de la aplicación y un botón *'Install'* que permite al administrador instalar la aplicación en el sistema.

7.1.4.2. Usuarios

El administrador puede visualizar la lista de usuarios del sistema, ver su información y modificarla en caso de que sea necesario. Para ello deberá pulsar en el menú superior derecho la opción *'Usuarios'*, apareciendo una nueva ventana similar a la de la figura 7.6.

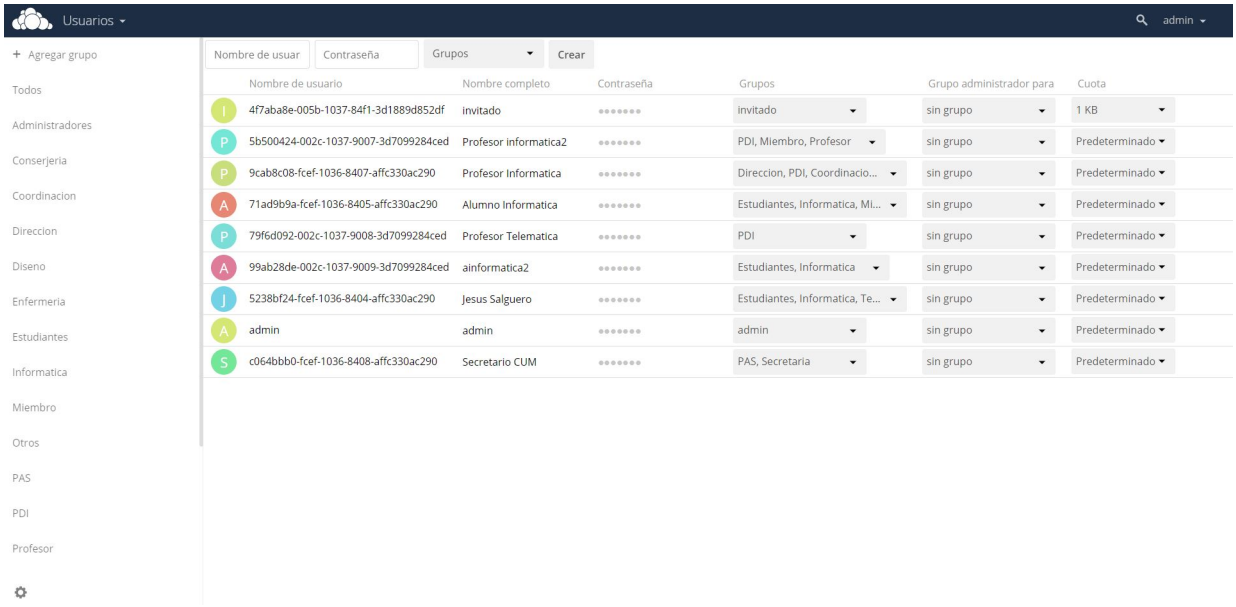

Figura 7.6: Gestión de los usuarios de ownCloud.

Además de las opciones descritas anteriormente, el administrador podrá:

- **Crear nuevos usuarios**: Para crear nuevos usuarios deberá rellenar el formulario que se encuentra en la parte superior, añadiendo un nombre de usuario, contraseña y los grupos a los que pertenece, y pulsar la opción *'Crear'*.
- **Crear nuevos grupos**: Para crear un nuevo grupo es necesario pulsar en la opción *'+Agregar grupo'* que se encuentra en el lateral superior izquierdo, rellenar el formulario con el nombre del nuevo grupo y pulsar la tecla *Enter*.

7.1.4.3. Ajustes

El administrador podrá editar los ajustes de ownCloud desde el menú superior derecho pulsando en la opción *'Ajustes'*. En esta sección podrá:

- Modificar sus datos personales y cambiar su contraseña.
- Ajustar el almacenamiento y la seguridad de la aplicación.
- Gestionar la autentificación de los usuarios, configurando LDAP.

En la figura 7.7, se puede observar dentro de la edición de ajustes de ownCloud, parte de la configuración LDAP, así como las distintas opciones que se encuentran en el menú lateral izquierdo.

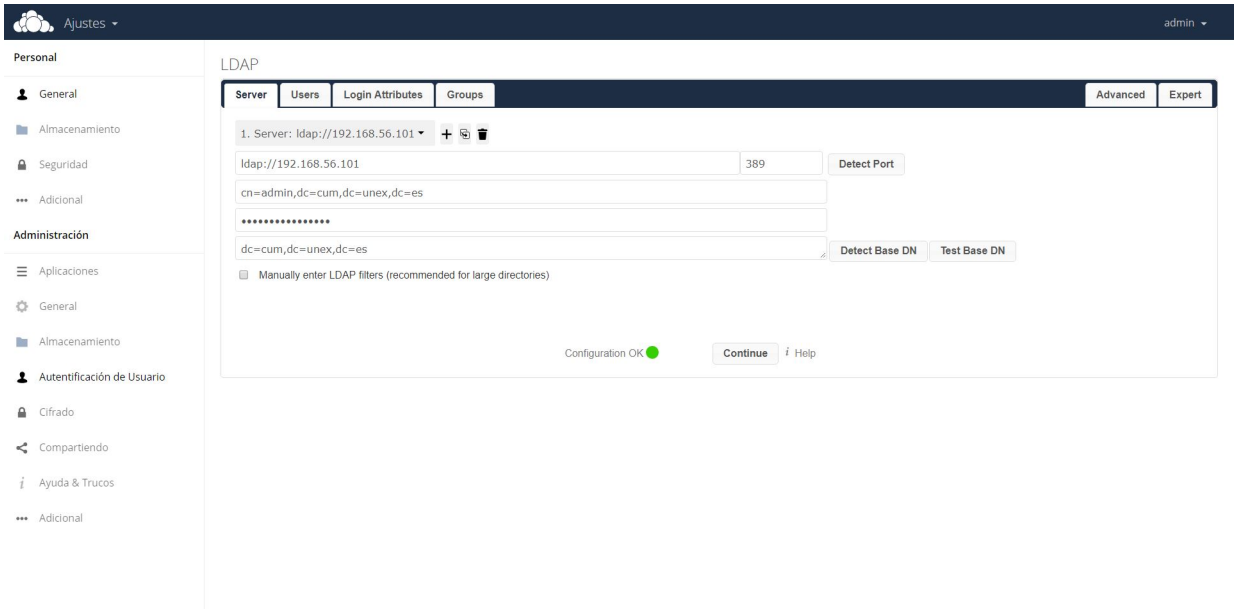

Figura 7.7: Gestión de los usuarios de ownCloud.

7.2. Aplicación web de Moodle

Moodle permite a los profesores y estudiantes matriculados en distintas asignaturas el acceso al contenido vinculado al aula virtual de cada una de ellas. Los profesores pueden gestionar sus aulas virtuales, pudiendo subir archivos, realizar cuestionarios, encuestas, consultas, talleres, tareas, entre muchas otras actividades.

En esta sección nos centraremos en la configuración de un repositorio por parte de un profesor y en cómo puede subir los archivos localizados en ownCloud a un aula virtual.

7.2.1. Agregar repositorio

Los profesores pueden agregar sus propios repositorios a sus aulas virtuales para poder enlazar los archivos que tienen en sus cuentas de ownCloud sin necesidad de tenerlos almacenados en sus equipos. Los pasos a seguir para añadir un repositorio son los siguientes:

1. Entrar en la plataforma Moodle (Campus Virtual de la UEx) e iniciar sesión con su usuario y contraseña.
2. Seleccionar un aula virtual y acceder a ella.

3. En el lateral izquierdo existe un bloque denominado *'Administración'*, donde se encuentra todo lo referente sobre la administración del curso. Dentro de ese bloque se deberá seleccionar la opción *'Repositorios'* haciendo clic sobre ella.

4. Aparecerá una nueva ventana con los repositorios creados previamente. Aparecerá también la opción *'Crear una instancia de repositorio "Repositorio WebDAV"'*, como se puede observar en la figura 7.8.

5. Si se pulsa sobre esa opción, aparecerá una nueva ventana de configuración WebDAV, como se puede observar en la figura 7.9. Los datos necesarios son:

   a) **Nombre:** Nombre que le quiera dar al repositorio para identificarlo dentro de Moodle.

   b) **Tipo WebDAV:** Puede ser HTTP o HTTPS según la configuración de ownCloud.

   c) **Servidor WebDAV:** Dirección IP o página web del servidor ownCloud.

   d) **Ruta WebDAV:** Resto de la ruta para conectarse al servidor ownCloud.

   e) **Identificación:** Tipo de identificación utilizada en WebDAV.

   f) **Puerto del servidor WebDAV:** Puerto del servidor WebDAV. Normalmente 80 si es HTTP y 443 si es HTTPS.

   g) **Usuario del servidor WebDAV:** Usuario con el que iniciar sesión en ownCloud.

   h) **Contraseña del servidor WebDAV:** Contraseña del usuario ownCloud.

6. Tras configurar todos los parámetros necesarios, se debe pulsar en el botón *'Guardar'*. Tras esto ya estará configurado el repositorio en Moodle.

Figura 7.8: Crear repositorio en Moodle.

Figura 7.9: Configuración de un repositorio WebDAV en Moodle.

7.2.2. Agregar un archivo del repositorio en un aula de Moodle

Una vez configurado el repositorio, los profesores pueden añadir los archivos que se encuentran en ownCloud en el aula virtual. Para realizar esta acción los pasos a seguir son los siguientes:

1. Dentro del aula virtual donde se ha configurado el repositorio se deberá activar la edición, pulsando el botón ***'Activar edición'*** localizado en la esquina superior derecha.

2. Después de activar la edición, se deberá situar en el tema que quiera subir el archivo del repositorio y pulsar en el botón ***'+ Añade una actividad o recurso'***.

3. Se abrirá una nueva ventana con las distintas actividades o recursos que pueden utilizar los profesores en sus aulas virtuales. Se deberá seleccionar ***'Archivo'*** y pulsar en ***'Agregar'***.

4. En la nueva ventana se deberá rellenar el formulario con los datos solicitados, como pueden ser el nombre que se le quiera dar al archivo en el aula o la descripción del archivo, entre otros.

5. En la opción ***'Seleccionar archivos'***, se deberá pulsar en el icono de una página con un símbolo '+' o dentro del recuadro con una flecha azul. Ambas opciones están destacadas en la figura 7.10.

6. Al realizar el paso anterior, se abrirá una nueva ventana llamada ***'Selector de archivos'*** que ofrecerá distintas opciones para subir un archivo. En el lateral izquierdo se encontrarán estas opciones, debiendo buscar el nombre que se le dio al repositorio a la hora de configurarlo, como se puede observar en la figura 7.11. También se podrá seleccionar la opción *'CUMDoc'* que aparece por defecto, siendo el repositorio configurado por el administrador. Este repositorio cuenta con los archivos que han sido compartidos con el grupo de invitados.

7. Una vez seleccionado el repositorio aparecerán las distintas carpetas y archivos disponibles en ownCloud. Basta con pulsar sobre el archivo que se desea subir.

8. Aparecerá una nueva ventana con los datos del archivo, pudiendo modificar su nombre, autor y licencia como se puede observar en la figura 7.12. Una vez configurado correctamente se deberá pulsar en ***'Seleccionar este archivo'***.

9. Se volverá a la pantalla de configuración inicial. Para finalizar se deberá pulsar el botón ***'Guardar cambios y regresar al curso'***.

10. Una vez realizado los pasos anteriores aparecerá en el aula virtual el archivo elegido.

Es importante destacar que los repositorios que pueden crear los profesores son en un curso, por lo que si el profesor cambia a otra aula virtual que tenga asignada no le aparecerá el repositorio propio que había configurado previamente en el otro aula. Por lo tanto, si se desea utilizar el mismo repositorio en otro aula virtual es necesario repetir los pasos previos de configuración en el nuevo aula virtual (sección 7.2.1) para poder subir archivos de ownCloud en ella.

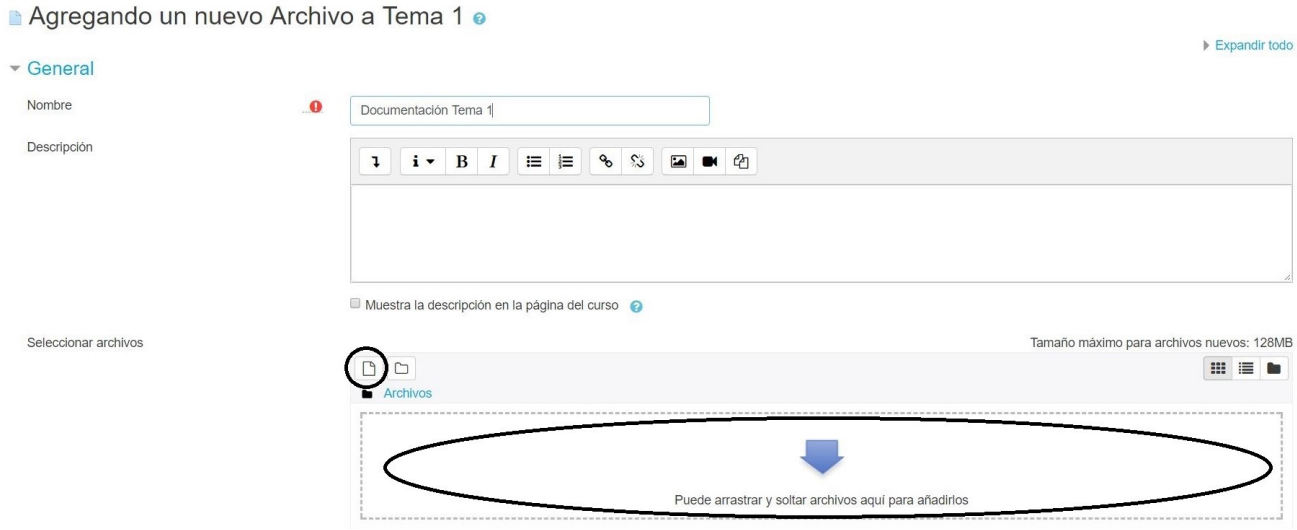

Figura 7.10: Configuración de un nuevo archivo en Moodle.

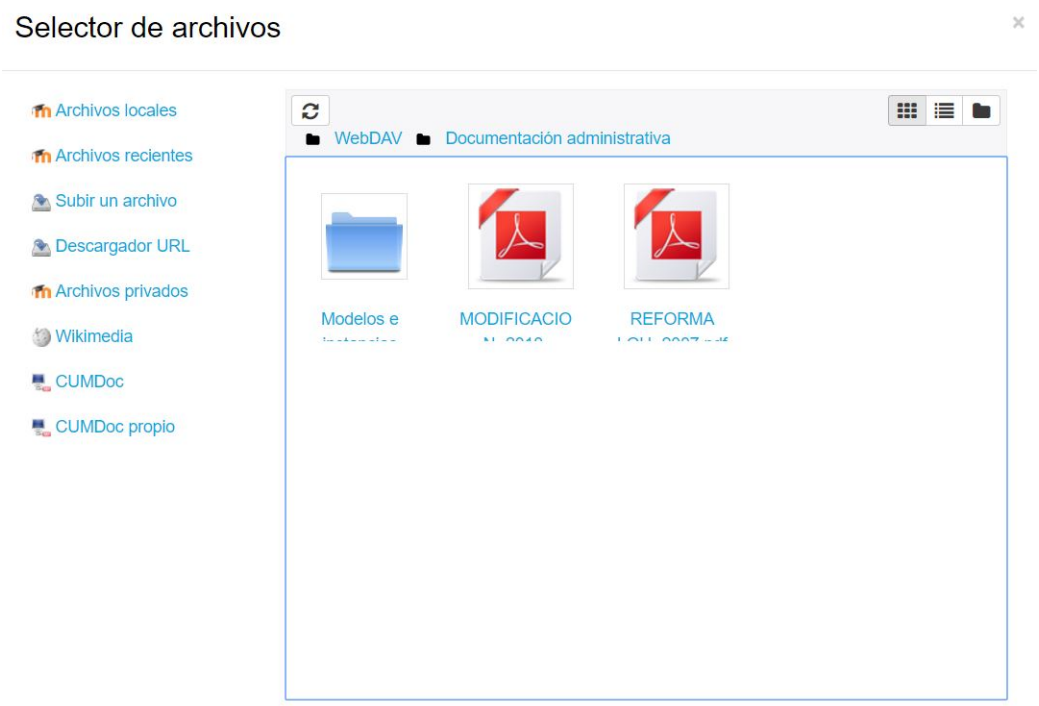

Figura 7.11: Selector de archivos en Moodle. Seleccionar un archivo.

Figura 7.12: Configuración de un archivo del repositorio WebDAV en Moodle.

7.3. Aplicación web CUMDoc

La aplicación web CUMDoc permite a los usuarios del sistema interaccionar con los archivos de ownCloud de manera ordenada y simple. Esta aplicación está disponible para todos los actores del sistema (PDI, PAS, Estudiantes, Invitados y Administrador). La gran parte del uso de la aplicación es común a todos los tipos de usuario, salvo la parte de la gestión de sus propios archivos y carpetas, que dependerá del perfil del usuario.

7.3.1. Acceso a la aplicación

Para acceder a la aplicación web CUMDoc es necesario utilizar la URL *http://<host>:<PuertoTomcat>/TFG_jesalguer/iniciarSesion* desde un navegador web. En caso de que se aloje en la misma máquina y se haya utilizado la configuración por defecto, la URL será *http://localhost:8080/TFG_jesalguer/iniciarSesion.* Al acceder aparecerá la pantalla de inicio de sesión, en la cual el usuario tendrá dos posibilidades:

- Acceder a la aplicación mediante su usuario y contraseña común al resto de aplicaciones. Al acceder con esta opción el usuario podrá ver las carpetas que hayan sido compartidas con su grupo o con él y si son propias gestionarlas.
- Si el usuario no quiere acceder a la aplicación con su usuario y contraseña o no dispone de él por no pertenecer a la comunidad universitaria, tendrá la opción de entrar como un invitado pulsando la opción *'Entrar como invitado'*, pudiendo consultar los documentos que hayan sido compartidos para todos los usuarios.

En la figura 7.13, se puede observar la página de inicio de sesión de la aplicación web vista en un PC, y en la figura 7.14 vista desde un smartphone[1].

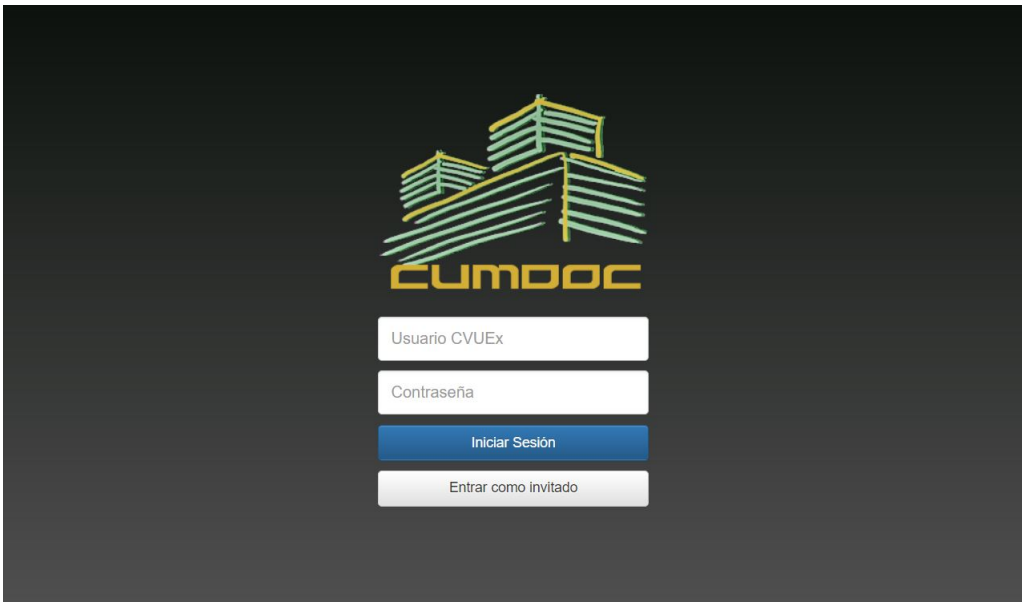

Figura 7.13: Página de inicio de sesión en la aplicación web CUMDoc desde un PC.

[1]Para la visualización desde un dispositivo móvil se ha utilizado la herramienta *Screenfly* que permite la visualización de páginas web pudiendo elegir distintos dispositivos, como pueden ser smartphones, tablets entre otros. Esta herramienta se encuentra disponible en http://quirktools.com/screenfly/

Figura 7.14: Página de inicio de sesión en la aplicación web CUMDoc desde un dispositivo móvil.

7.3.2. Pantalla principal de la aplicación

Al entrar en la aplicación aparecerá la pantalla principal. Según el grupo al que pertenezca el usuario y según las carpetas que le hayan compartido podrá visualizar diferente contenido. Hay contenido que está accesible para todos los usuarios:

- **Documentación administrativa:** En esta carpeta se encuentra todos los documentos referentes a trámites administrativos, como los modelos e instancias para los distintos grupos de la comunidad universitaria.

- **Normativas:** En esta carpeta se encuentra la documentación referente a la normativa general, de la Universidad de Extremadura y boletines oficiales.

- **TFG:** En esta carpeta se encuentra la documentación referente a los Trabajos Fin de Estudios, como la normativa y anexos, ofertas de temas, fechas y plazos de interés, y próximas lecturas.

- **Titulaciones:** Esta carpeta contiene todos los documentos relacionados con las titulaciones impartidas en el centro, divididas según la titulación, curso y semestre. Dentro de esta estructura serán accesibles para todos los usuarios las carpetas pertenecientes a los estudiantes, donde suben su propio material, y las carpetas de las asignaturas cuyo profesor decida hacerla pública a todos los usuarios.

En la figura 7.15, se puede observar la pantalla principal de la aplicación web vista desde un PC y en la figura 7.16 vista desde un smartphone.

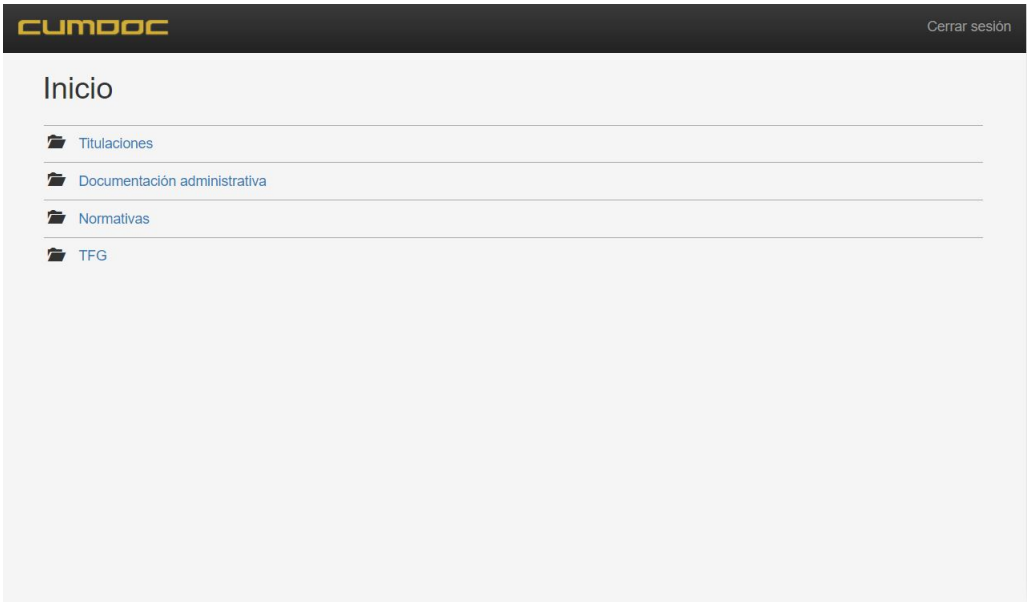

Figura 7.15: Página principal de la aplicación web CUMDoc desde un PC.

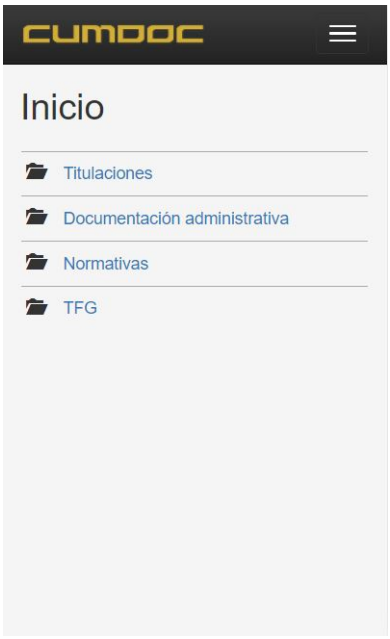

Figura 7.16: Página principal de la aplicación web CUMDoc desde un dispositivo móvil.

7.3.3. Gestión de las carpetas

Según el grupo al que pertenezca el usuario y si tiene asignada la gestión de una carpeta podrá ocurrir cuatro situaciones:

- Que no tenga asignada la gestión de la carpeta ni tenga derecho de acceso al contenido. En esta situación no verá la carpeta en la aplicación, por lo que no podrá acceder al

contenido.

- Que no tenga asignada la gestión pero si tenga derecho de acceso al contenido. En este caso podrá acceder al contenido y descargarlo, pero no podrá realizar ninguna modificación en él.

- Que tenga asignada la gestión y sea PDI o PAS. En este caso podrán acceder al contenido, descargarlo, añadir, modificar y borrar archivos, y crear y borrar carpetas.

- Que sea un estudiante, por lo que tiene permiso de edición en las carpetas de las asignaturas vinculadas a los estudiantes, es decir, las carpetas que sirven para que los estudiantes puedan compartir sus documentos. En esta carpeta podrán subir, modificar y eliminar sus propios archivos, pero no los de los demás.

7.3.3.1. Acceso a una carpeta compartida con el usuario sin derecho de edición

Este caso es el segundo punto del apartado anterior. El usuario no tiene asignada la gestión de la carpeta pero si puede acceder al contenido. En esta situación el usuario solo podrá:

- Visualizar el contenido pulsando su nombre si se trata de un PDF.

- Descargar el contenido pulsando en la opción *'Descargar'* o pulsando sobre si nombre si el archivo no es un PDF.

En la figura 7.17, se puede observar la carpeta *'EDI_ Alumnos'*, accedida por un usuario PDI, por lo que no puede añadir contenido en ella. En la figura 7.18, se puede observar la misma página vista desde un smartphone.

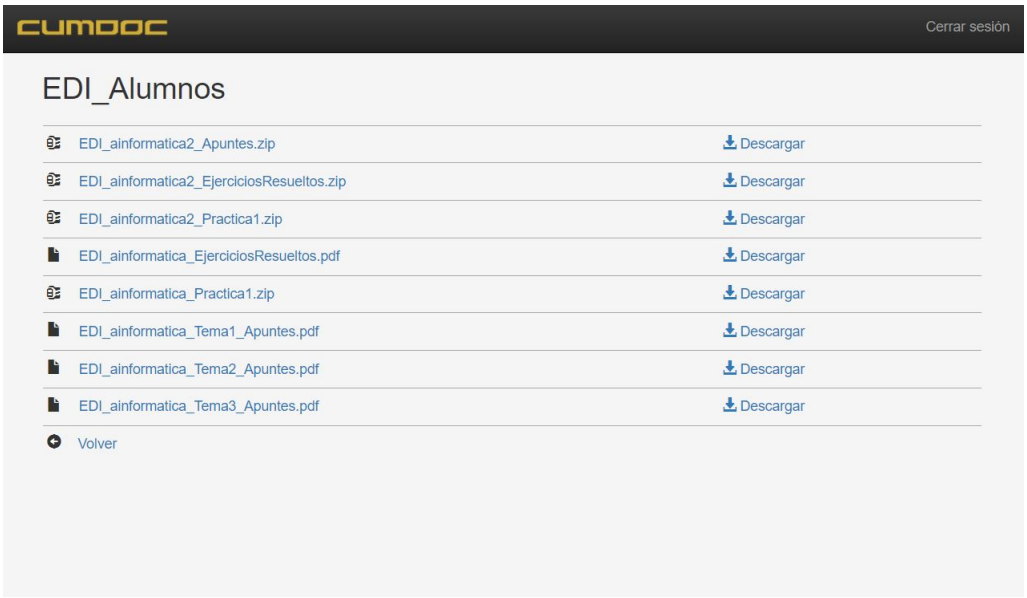

Figura 7.17: Carpeta *EDI_ Alumnos* consultada por un usuario PDI desde un PC.

Figura 7.18: Carpeta *EDI_ Alumnos* consultada por un usuario PDI desde un dispositivo móvil.

7.3.3.2. Acceso a una carpeta con derecho de edición por parte de un usuario PDI o PAS

En este caso, el usuario podrá no solo descargar el contenido, sino que podrá realizar las siguientes acciones:

- **Subir un archivo:** Para ello deberá pulsar el botón *'Subir archivo'*. Tras pulsarlo aparecerá una nueva ventana en la pantalla que permite seleccionar el archivo del equipo del usuario que desea subir. Tras seleccionar el archivo se deberá pulsar el botón *'Subir'*, para añadir el archivo elegido en la carpeta.

- **Crear una carpeta:** Para crear una carpeta se deberá pulsar el botón *'Crear carpeta'*. Aparecerá una nueva ventana en la pantalla con un formulario que pide el nombre de la carpeta a crear. Tras rellenarlo se deberá pulsar el botón *'Crear'* y aparecerá la carpeta creada dentro de la carpeta inicial.

- **Eliminar archivo o carpeta:** Para eliminar un archivo o carpeta únicamente se deberá pulsar la opción *'Eliminar'* que se encuentra al lado del nombre del archivo o carpeta a eliminar.

En la figura 7.19, se puede observar una carpeta con derecho de edición en la aplicación CUMDoc, en concreto la carpeta *'Normativas'* accedida por el secretario del CUM. Si se pulsa el botón *'Subir archivo'* aparecerá la figura 7.20. En cambio, si se pulsa el botón *'Crear carpeta'* aparecerá la figura 7.21. Si esta última acción se realizase en un dispositivo móvil se vería de manera similar a la figura 7.22.

Nota importante: Para poder gestionar las carpetas desde la aplicación web de CUMDoc y no desde la aplicación web de ownCloud se deberá dejar estas carpetas en la ruta inicial de ownCloud, es decir en la primera página tras iniciar sesión en la aplicación de ownCloud.

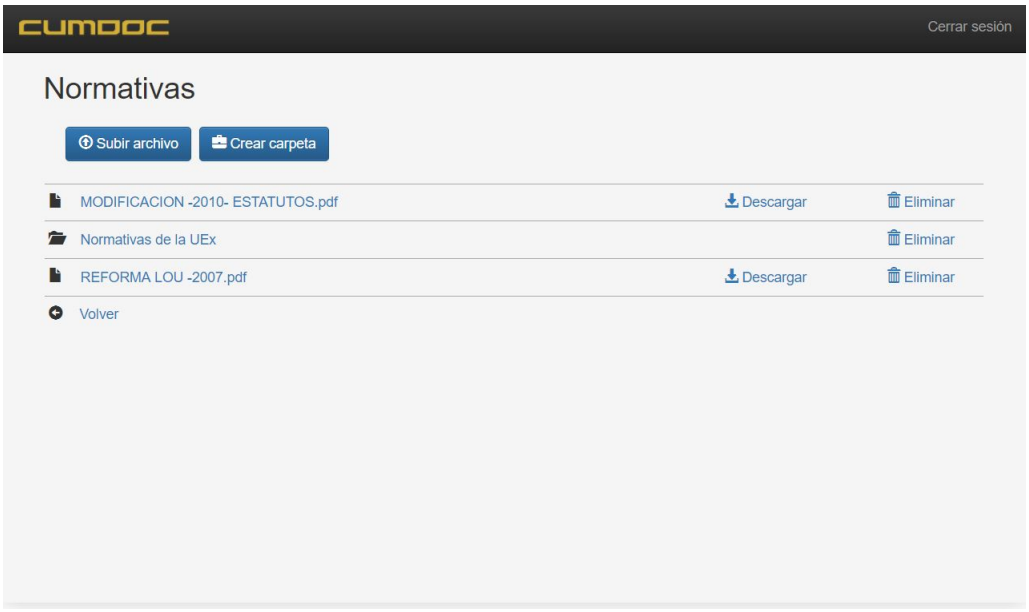

Figura 7.19: Carpeta *Normativas* consultada por el usuario Secretario desde un PC.

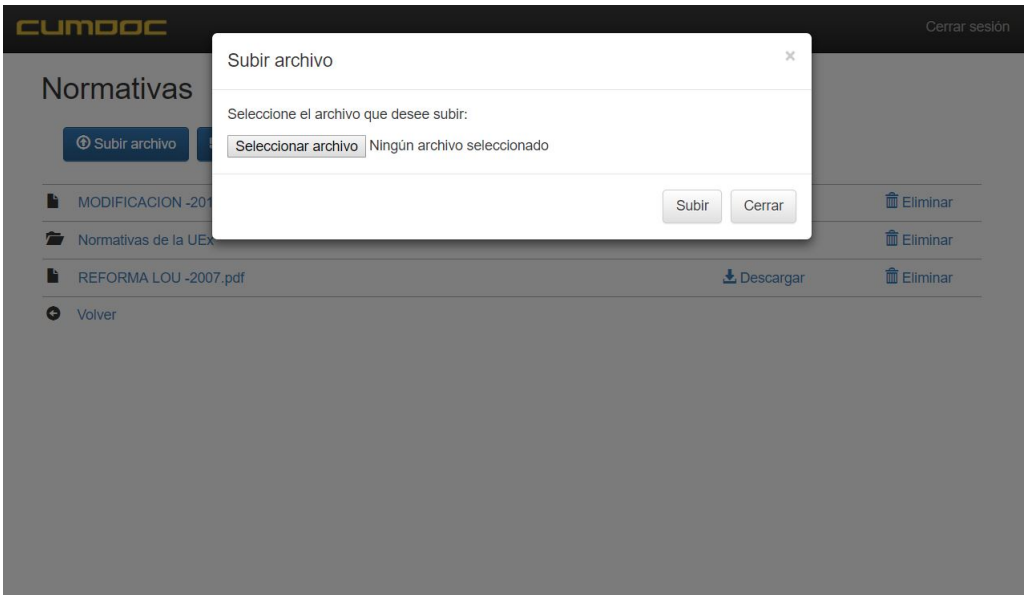

Figura 7.20: Carpeta *Normativas* consultada por el usuario Secretario desde un PC. Subiendo un archivo.

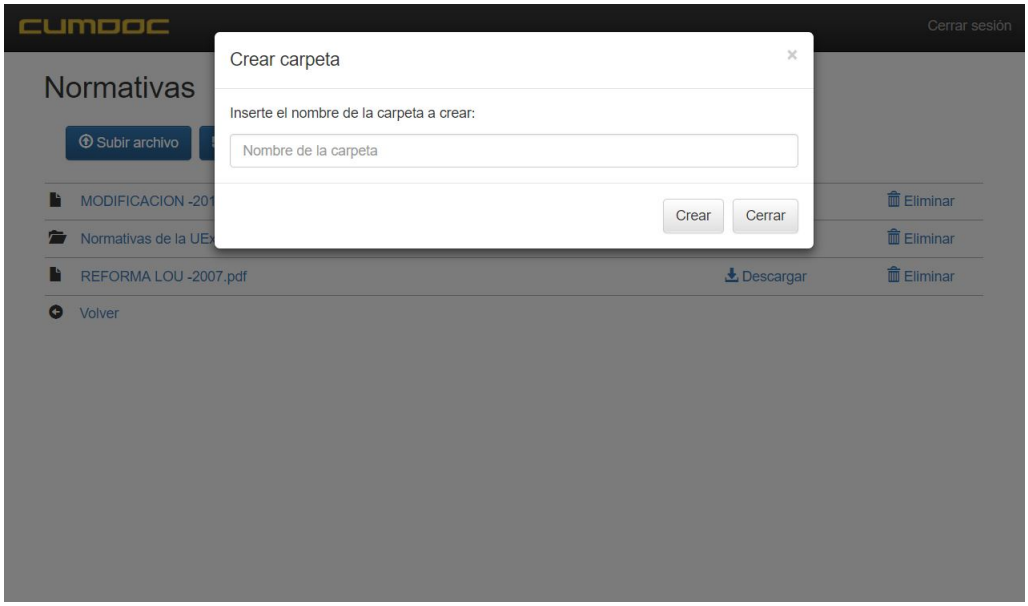

Figura 7.21: Carpeta *Normativas* consultada por el usuario Secretario desde un PC. Creando una carpeta.

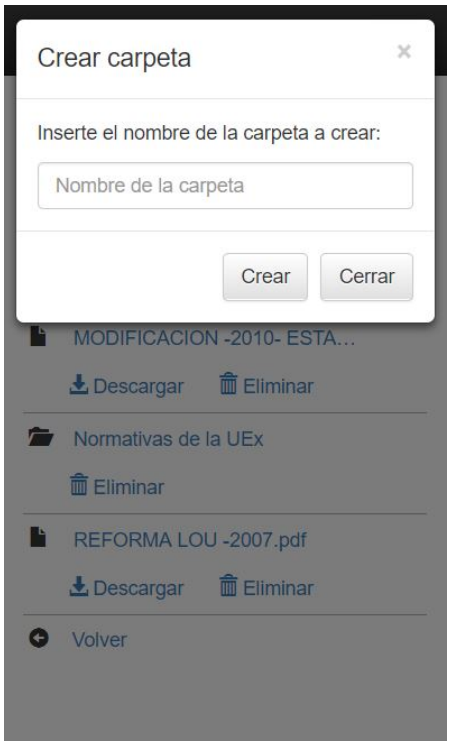

Figura 7.22: Carpeta *Normativas* consultada por el usuario Secretario desde un dispositivo móvil. Creando una carpeta.

7.3.3.3. Acceso a una carpeta de estudiantes por un estudiante

En este caso, el usuario podrá no solo descargar el contenido, sino que podrá realizar las siguientes acciones:

- **Subir un archivo:** Para ello deberá pulsar el botón *'Subir archivo'*. Tras pulsarlo aparecerá una nueva ventana en la pantalla que permite seleccionar el archivo del equipo del usuario que desea subir. Tras seleccionar el archivo se deberá pulsar el botón *'Subir'*, para añadir el archivo elegido en la carpeta. Una vez subido el archivo este tendrá delante del nombre que tenía el nombre de la asignatura y el nombre del usuario que lo ha subido siguiendo el siguiente formato: *<nombreAsignatura>_ <nombreUsuario>_ <nombreArchivo>*.
- **Eliminar archivo o carpeta:** Para eliminar un archivo o carpeta únicamente se deberá pulsar la opción *'Eliminar'* que se encuentra al lado del nombre del archivo o carpeta a eliminar. Solo aparecerá esta opción al lado de los archivos que han sido subidos por el usuario que accede a la aplicación, impidiendo que se borren archivos aportados por otros estudiantes.

En la figura 7.23, se puede observar la carpeta de la asignatura *'EDI'* para las aportaciones realizadas por los estudiantes accedida por un estudiante. Es importante destacar que solo aparece la opción de *'Eliminar'* en los archivos que ha subido este estudiante.

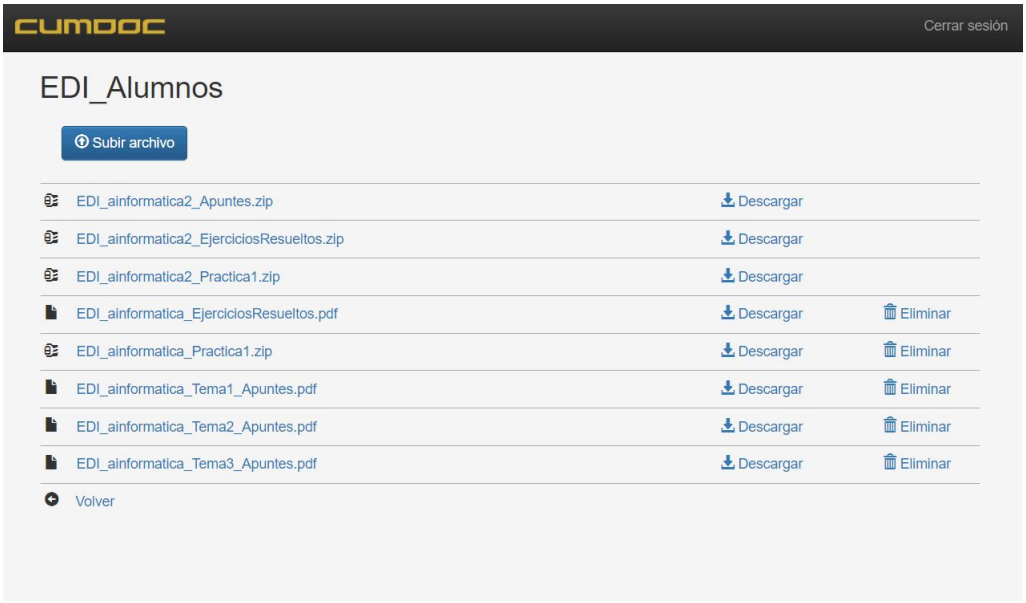

Figura 7.23: Carpeta *EDI_ Alumnos* consultada por un estudiante desde un PC.

Capítulo 8

Conclusiones y Trabajos Futuros

Contenidos

En este capítulo se resumirán las principales conclusiones obtenidas tras la realización de este Trabajo Fin de Grado, así como los trabajos futuros que pueden plantearse y la planificación real del proyecto.

En la sección 8.1 se encuentra las principales conclusiones del proyecto.

En la sección 8.2 se comenta algunos trabajos futuros que pueden plantearse teniendo como punto de partida este proyecto.

En la sección 8.3 se muestra la planificación real del proyecto argumentando las variaciones sufridas respecto a la planificación estimada.

8.1. Conclusiones

8.1.1. Conclusiones del proyecto

La idea de un gestor documental en el Centro Universitario de Mérida surgió de la necesidad de tener organizada la documentación existente para facilitar su acceso y de la imposibilidad de realizar aportaciones por parte de los estudiantes del centro. Tras analizar este problema, Manuel Botellero Gutiérrez en su Trabajo Fin de Grado [1] decidió realizar un acercamiento al desarrollo de un gestor documental.

Partiendo de este proyecto, se decidió continuar con la idea de realizar un gestor documental, surgiendo así este Trabajo Fin de Grado. Gracias a este proyecto se ha desarrollado un gestor documental integrado con las distintas plataformas existentes en la Universidad de Extremadura.

Este gestor documental sirve como repositorio para el centro, permitiendo localizar los documentos de una manera rápida y sencilla tanto para los miembros del centro como para usuarios externos. Además, hace participe al estudiante, permitiendo que elabore y comparta

su propia documentación, siendo un miembro de importancia dentro del repositorio. Todo esto transforma al Centro Universitario de Mérida en un centro donde todos sus miembros cooperan y se relacionan, mejorando la participación, la elaboración de material didáctico y el aprendizaje, al disponer de más documentación con mayor facilidad. También mejora el entorno administrativo, agilizando el acceso a los trámites existentes, llegando con más facilidad a los usuarios.

Este Trabajo Fin de Grado supone un gran paso para que el gestor documental se vuelva una realidad y se configure e implante en el Centro Universitario de Mérida.

8.1.2. Conclusiones personales

Este Trabajo Fin de Grado es el último paso para finalizar una etapa que comenzó hace cinco años, cuando pisé por primera vez el Centro Universitario de Mérida. La finalización de este proyecto no significa únicamente la finalización de una etapa, sino el comienzo de una nueva más orientada al mundo laboral.

No solo he aprendido estos años contenido relacionado con la Informática, como puede ser la programación, el análisis y diseño de sistemas, entre otros, sino que he adquirido una serie de competencias transversales. Algunas de estas, se han visto reforzadas en la realización de este proyecto, queriendo destacar las siguientes:

- **Búsqueda de información:** Ha sido fundamental para poder empezar y avanzar a lo largo del proyecto.
- **Comunicación escrita:** Sin duda la más destacable. A lo largo de estos años he realizado documentos, pero ninguno con este volumen, tiempo dedicado y una estructura tan correcta. Esta memoria es uno de los resultados de este Trabajo Fin de Grado de los que me siento más orgulloso.
- **Planificación y organización del trabajo:** Para el correcto desarrollo de este proyecto se realizó una planificación previa y se organizó el trabajo en función de ella.
- **Comunicación en lengua inglesa:** La mayor parte de la documentación existente se encuentra en ingles.
- **Toma de decisiones:** A lo largo del desarrollo del proyecto se han tenido que ir tomando una serie de decisiones, algunas acertadas y otras no tan acertadas.
- **Comunicación oral:** Con la presentación que realizaré de este trabajo.

8.2. Trabajos futuros

Desde la amplitud del proyecto,existen múltiples enfoques de trabajo futuro sobre él:

Implementación en un entorno real

En este proyecto se ha simulado las diferentes plataformas existentes en la Universidad de Extremadura: se ha creado un Campus Virtual a través de la implementación de una plataforma Moodle, se ha creado y configurado LDAP siendo la base de datos utilizada por

la Universidad y se ha implementado una plataforma ownCloud simulando la existente en la Universidad.

Para un despliegue real del gestor documental sería conveniente la utilización de las plataformas existentes y su adaptación a las necesidades del Centro Universitario de Mérida.

Mejorar la seguridad del sistema

Cada una de las plataformas que han sido implementadas en el proyecto permiten añadir mayor seguridad. Se puede configurar en LDAP el uso de la Capa de Conexión Segura (SSL) y la Seguridad de la Capa de Transporte (TLS), para proteger así los datos confidenciales que almacena [59]. OwnCloud por su parte, permite el uso del protocolo HTTPS, HSTS para evitar los ataques *man in the middle* y certificados SSL [60]. Moodle también permite el uso de SSL, HTTPS y la utilización de firewalls y antivirus en el servidor donde se encuentre alojado la plataforma.

Nuevas funcionalidades

Sería interesante añadir nuevas funcionalidades y mejoras a la aplicación CUMDoc, siendo algunas de ellas las siguientes:

- Habilitar la visualización de documentos con formatos distintos a PDF y mostrar el contenido de archivos comprimidos sin necesidad de descargar.
- Permitir la edición de documentos de texto desde el propio navegador. Esta mejora podría añadir la elección de los usuarios que contarán con permisos de edición del documento y un registro de los cambios que han realizado cada uno de ellos.
- Habilitar la búsqueda de documentos dentro de una carpeta o una jerarquía de carpetas.
- Incluir etiquetas y categorías en los documentos para poder buscar todos los documentos que pertenezcan a esa categoría o posean esas etiquetas.

8.3. Planificación estimada vs Planificación real

.En la figura 8.1, se puede observar un Diagrama de Gantt con la planificación real de este proyecto. En comparación con la planificación estimada que se realizó en el capítulo 4 *Análisis del sistema* de este documento, se puede apreciar que es ligeramente superior. Esto se debe a varios motivos:

- Otras actividades académicas coincidieron en el mismo periodo de tiempo que este proyecto, no pudiéndole dedicar el tiempo esperado.
- En un principio, se decidió usar la plataforma Alfresco para la gestión de los documentos, se instaló y se hicieron numerosas pruebas antes de decidir cambiar la plataforma por ownCloud. Esto llevó un tiempo y afectó a la iteración del gestor documental, ampliando su duración.
- Durante el desarrollo de este Trabajo Fin de Grado estuve disfrutando de una beca en el Campus Virtual de la Universidad de Extremadura, limitando parte del tiempo disponible que tenía.

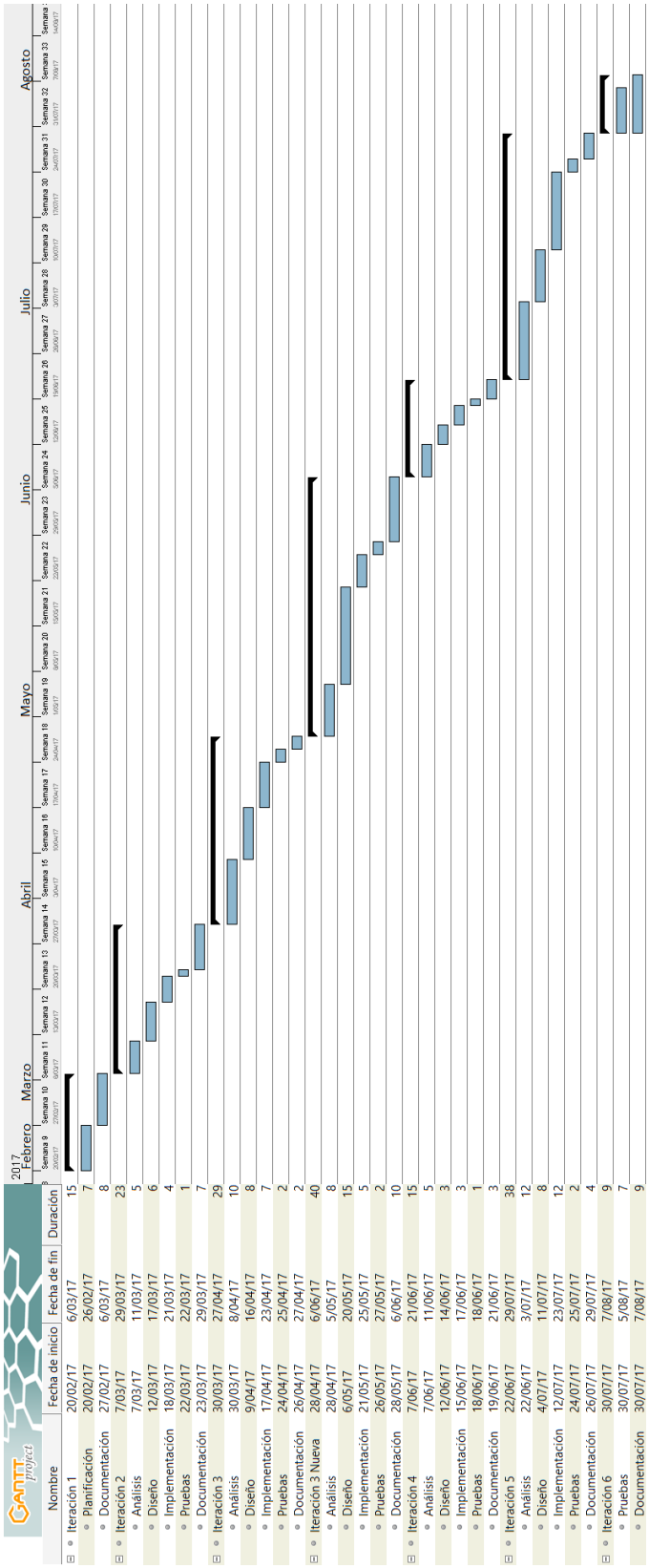

Figura 8.1: Diagrama de Gantt. Planificación real.

Bibliografía

[1] Botellero Gutiérrez, Manuel. Trabajo Fin de Grado: *CUMDoc: Gestor documental del Centro Universitario de Mérida.*

[2] *Bases de datos.* Accedido el 3 de marzo de 2017. Accesible en http://volaya.github.io/libro-sig/chapters/Bases_datos.html

[3] *Los 10 beneficios de la gestión documenal en las organizaciones.* Accedido el 3 de marzo de 2017. Accesible en http://www.comunidadbaratz.com/blog/los-10-beneficios-de-la-gestion-documental-en-las-organizaciones/

[4] Castro, Luis. *¿Qué es el almacenamiento en la nube?.* Accedido el 3 de marzo de 2017. Accesible en https://www.aboutespanol.com/que-es-almacenamiento-en-la-nube-157946

[5] *Plataformas educativas, ¿qué son y para qué sirven?.* Accedido el 4 de marzo de 2017. Accesible en http://www.aula1.com/plataformas-educativas/

[6] Wikipedia: *Aplicación web.* Accedido el 5 de marzo de 2017. Accesible en https://es.wikipedia.org/wiki/Aplicaci%C3%B3n_web

[7] *Desarrollo iterativo e incremental.* Accedido el 13 de junio de 2017. Accesible en https://proyectosagiles.org/desarrollo-iterativo-incremental/

[8] Project Management: *Características y fases del modelo incremental.* Accedido el 13 de junio de 2017. Accesible en http://www.obs-edu.com/es/blog-project-management/metodologias-agiles/caracteristicas-y-fases-del-modelo-incremental

[9] *Conceptos de la metodología orientada a objetos.* Accedido el 25 de junio de 2017. Accesible en http://profesores.fi-b.unam.mx/carlos/aydoo/conceptos_oo.html

[10] Pressman, Roger S. *Ingeniería del Software: Un enfoque práctico.* McGraw-Hill (2002)

[11] Alarcón, Raúl. *Diseño orientado a objetos con UML.* Grupo EIDOS (2000)

[12] E. Kendall, Kenneth y E. Kendall, Julie. *Análisis y diseño de sistemas. Sexta edición.* Pearson Educación (2005).

[13] Wikipedia: *Modelo-Vista-Controlador.* Accedido el 7 de junio de 2017. Accesible en https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador

[14] Wikipedia: *Programación por capas.* Accedido el 7 de junio de 2017. Accesible en https://es.wikipedia.org/wiki/Programaci%C3%B3n_por_capas

[15] Molina Atiencia, Tatiana E. y Mantilla Rivera, Cesar E. *Análisis comparativo de la productividad en el desarrollo de aplicaciones web utilizando las tecnologías JDBC y JPA.* Accedido el 28 de julio de 2017. Accesible en http://dspace.unach.edu.ec/bitstream/51000/1525/1/UNACH-EC-ISC-2016-0013.pdf

[16] Wikipedia: *iBATIS.* Accedido el 8 de junio de 2017. Accesible en https://es.wikipedia.org/wiki/IBATIS

[17] Marco de Desarrollo de la Junta de Andalucía: *iBATIS.* Accedido el 28 de julio de 2017. Accesible en http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/98

[18] Wikipedia: *Java Persistence API.* Accedido el 8 de junio de 2017. Accesible en https://es.wikipedia.org/wiki/Java_Persistence_API

[19] Marco de Desarrollo de la Junta de Andalucía: *JPA.* Accedido el 28 de julio de 2017. Accesible en http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/96

[20] *JPA vs Hibernate ¿Cuál es la diferencia?.* Accedido el 8 de junio de 2017. Accesible en https://www.facilcloud.com/noticias/jpa-vs-hibernate/

[21] Wikipedia: *Hibernate.* Accedido el 8 de junio de 2017. Accesible en https://es.wikipedia.org/wiki/Hibernate

[22] Marco de Desarrollo de la Junta de Andalucía: *Hibernate.* Accedido el 28 de julio de 2017. Accesible en http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/97

[23] Marco de Desarrollo de la Junta de Andalucía: *Seam.* Accedido el 8 de junio de 2017. Accesible en http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/94

[24] Marco de Desarrollo de la Junta de Andalucía: *Estudio comparativo entre JBoss Seam y Spring.* Accedido el 28 de julio de 2017. Accesible en http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/169

[25] Marco de Desarrollo de la Junta de Andalucía: *Enterprise JavaBeans 3.* Accedido el 8 de junio de 2017. Accesible en http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/95

[26] Marco de Desarrollo de la Junta de Andalucía: *Spring.* Accedido el 8 de junio de 2017. Accesible en http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/93

[27] *Introducción a Struts: El controlador y las acciones.* Accedido el 8 de junio de 2017. Accesible en http://www.jtech.ua.es/j2ee/publico/struts-2010-11/sesion01-struts-apuntes.html

[28] *STRUTS 2 vs SPRINGMVC: Know the Difference & Choose the Best One Based On Your Requirements.* Accedido el 29 de julio de 2017. Accesible en http://www.cygnet-infotech.com/blog/struts-2-vs-springmvc

[29] Wikipedia: *JavaServer Faces.* Accedido el 8 de junio de 2017. Accesible en https://es.wikipedia.org/wiki/JavaServer_Faces

[30] *¿Tiene futuro JSF?* Accedido el 29 de julio de 2017. Accesible en http://www.arquitecturajava.com/tiene-futuro-jsf/

[31] *Spring - MVC Framework.* Accedido el 8 de junio de 2017. Accesible en https://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm

[32] *Introducción a Spring MVC.* Accedido el 29 de julio de 2017. Accesible en http://jaimecarmonaloeches.blogspot.com.es/2012/01/introduccion-spring-mvc.html

[33] Wikipedia: *Binary large object.* Accedido el 8 de junio de 2017. Accesible en https://es.wikipedia.org/wiki/Binary_large_object

[34] *Ventajas y desventajas de guardar imágenes en la base de datos.* Accedido el 8 de junio de 2017. Accesible en http://www.phpcentral.com/pregunta/252/ventajas-y-desventajas-de-guardar-imagenes-en-la-base-de-datos

[35] Página oficial de Alfresco. Accedido el 8 de junio de 2017. Accesible en https://www.alfresco.com/es/alfresco-software-para-ecm-y-bpm

[36] Documentación oficial de Moodle: *Alfresco repository.* Accedido el 8 de junio de 2017. Accesible en https://docs.moodle.org/33/en/Alfresco_repository

[37] Página oficial de Dropbox. Accedido el 19 de junio de 2017. Accesible en https://www.dropbox.com/help

[38] Página oficial de OneDrive. Accedido el 19 de junio de 2017. Accesible en https://products.office.com/es-es/compare-all-microsoft-office-products?tab=1

[39] Página oficial de Google Drive. Accedido el 19 de junio de 2017. Accesible en https://www.google.com/drive/

[40] Página oficial de ownCloud. Accedido el 10 de junio de 2017. Accesible en https://owncloud.org/

[41] Wikipedia: *ownCloud.* Accedido el 10 de junio de 2017. Accesible en https://es.wikipedia.org/wiki/OwnCloud

[42] *NextCloud es una bifurcación de ownCloud.* Accedido el 10 de junio de 2017. Accesible en http://www.muylinux.com/2016/06/02/nextcloud-bifurcacion-owncloud

[43] *Diferencias entre ownCloud y NextCloud.* Accedido el 10 de junio de 2017. Accesible en http://www.vozidea.com/diferencias-entre-owncloud-y-nextcloud

[44] Wikipedia: *WebDAV.* Accedido el 10 de junio de 2017. Accesible en https://es.wikipedia.org/wiki/WebDAV

[45] *WebDAV: Qué es y cómo funciona. Manual para usarlo con Otixo.* Accedido el 19 de junio de 2017. Accesible en https://www.redeszone.net/2013/03/12/webdav-que-es-y-como-funciona-manual-para-usarlo-con-otixo/

[46] *Conceptos básicos de WebDAV.* Accedido el 19 de junio de 2017. Accesible en http://svnbook.red-bean.com/es/1.0/svn-ap-c-sect-1.html

[47] *Utilizando WebDAV.* Accedido el 19 de junio de 2017. Accesible en http://www.ilustrados.com/documentos/utilizando-webDAV.pdf

[48] Wikipedia: *Protocolo Ligero de Acceso a Directorios.* Accedido el 10 de junio de 2017. Accesible en https://es.wikipedia.org/wiki/Protocolo_Ligero_de_Acceso_a_Directorios

[49] *Protocolo LDAP.* Accedido el 10 de junio de 2017. Accesible en http://es.ccm.net/contents/269-protocolo-ldap

[50] *Usar una interfaz web para gestionar usuarios y grupos en el servidor OpenLDAP.* Accedido el 12 de julio de 2017. Accesible en http://somebooks.es/12-8-usar-una-interfaz-web-para-gestionar-usuarios-y-grupos-en-el-servidor-openldap/

[51] Página oficial de Moodle. Accedido el 10 de Junio de 2017. Accesible en https://moodle.org/

[52] *¿Qué es Moodle?.* Accedido el 10 de Junio de 2017. Accesible en http://www.entornos.com.ar/moodle

[53] GitHub: *lookfirst/sardine.* Accedido el 18 de Julio de 2017. Accesible en https://github.com/lookfirst/sardine

[54] Esteban, Ángel. *Tecnologías de servidor con Java: Servlets, JavaBeans y JSP.* Grupo Eidos (2000).

[55] Gónzalez, Daniel y Marcos,María del Carmen. *Responsive web design: Diseño multidispositivo para mejorar la experencia de usuario*. Bid, textos universitaris de biblioteconomia i documentació (2013). Accedido el 18 de Julio de 2017. Accesible en http://bid.ub.edu/pdf/31/es/gonzalez2.pdf

[56] Mark Otto, Jacob Thornton. *Bootstrap 3.* LibrosWeb. Accedido el 18 de julio de 2017. Accesible en http://librosweb.es/libro/bootstrap_3/

[57] Página oficial de Bootstrap. Accedido el 18 de julio de 2017. Accesible en http://getbootstrap.com/

[58] *¿Qué es un CDN y cuándo usarlo?.* Accedido el 18 de julio de 2017. Accesible en http://blog.hostdime.com.co/que-es-un-cdn-y-cuando-usarlo/

[59] *Protocolo ligero de acceso a directorios (LDAP).* Accedido el 7 de agosto de 2017. Accesible en http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-es-4/ch-ldap.html

[60] *Configurar y asegurar un servidor ownCloud.* Accedido el 7 de agosto de 2017. Accesible enhttps://www.ochobitshacenunbyte.com/2016/05/05/configurar-servidor-asegurar-owncloud/

Capítulo 9

Anexo I. Instalación de XAMPP

XAMPP es un paquete de instalación indepediente de la plataforma, de software libre, que contiene el servidor web Apache, el sistema de gestión de bases de datos MariaDB, el cual es un derivado de MySQL, y los intérpretes para lenguajes PHP y Perl.

En este Trabajo Fin de Grado se ha utilizado XAMPP para cubrir los requisitos de instalación y funcionamiento de Moodle y ownCloud.

En este anexo se describirá el proceso de instalación de XAMPP v5.6.30 en el sistema operativo utilizado para el desarrollo del Trabajo Fin de Grado, Debian 8.0. Todos los comandos se deberán realizar siendo superusuario o con la palabra "sudo" delante del comando.

- En primer lugar, es necesario descargar el instalador de XAMPP de su web oficial https://www.apachefriends.org/es/index.html. En la página web, se deberá seleccionar la opción XAMPP para Linux.

- Una vez finalizada la descarga, es necesario abrir un terminal y situarnos en la carpeta donde se encuentra el archivo descargado mediante el comando '*cd <rutaAlArchivo>*'. Después, se deberá cambiar los permisos de lectura, escritura y ejecución del fichero descargado mediante el comando '*chmod 755 <nombreDelArchivoDescargado.run>*'.

- Se comenzará la instalación mediante el comando '*./<nombreDelArchivoDescargado.run>*', el cual abrirá instalador de XAMPP.

- En la primera ventana se deberá pulsar "Next" para pasar a la opción de seleccionar los componentes deseados.

- En esta nueva ventana, se deberá marcar las dos opciones disponibles, "XAMPP Core Files" y "XAMPP Developer Files", y se hará clic en "Next".

- En la siguiente ventana nos mostrará la ruta en la que será instalado XAMPP. Por defecto se instalará en "opt/lampp". Tras pulsar "Next" comenzará la instalación.

- Una vez instalado XAMPP es necesario arrancarlo, para ello se deberá ejecutar el comando '*/opt/lampp/lampp start*'. Para visualizar su estado se podrá ejecutar el comando '*/opt/lampp/lampp status*'.

Figura 9.1: Web oficial de XAMPP. Descarga de XAMPP.

Figura 9.2: Instalador de XAMPP. Inicio.

Figura 9.3: Instalador de XAMPP. Opciones de instalación.

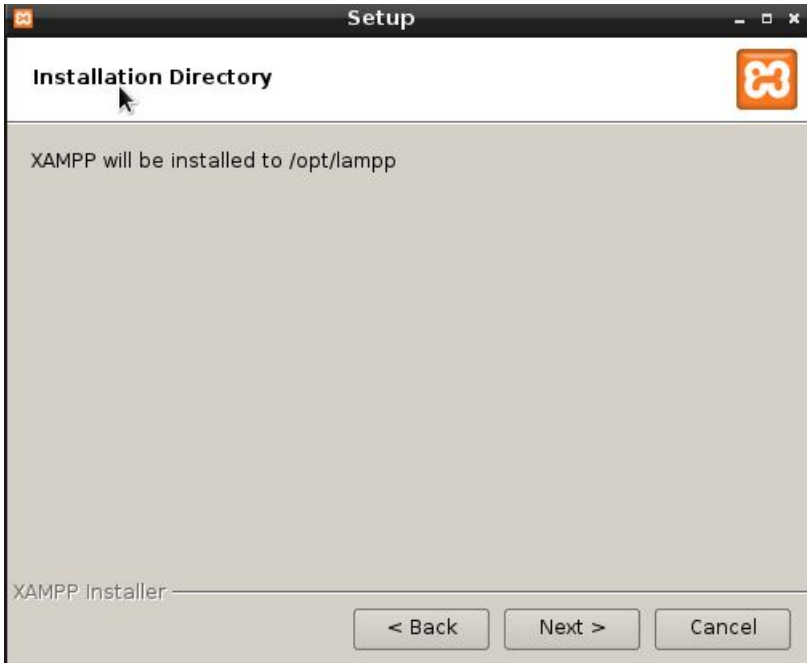

Figura 9.4: Instalador de XAMPP. Ruta de la instalación.

Capítulo 10

Anexo II. Instalación de Moodle

Moodle es una plataforma de aprendizaje diseñada para proporcionar a profesores, administradores y estudiantes un sistema robusto y seguro para crear ambientes de aprendizaje personalizados. Actualmente, la Universidad de Extremadura tiene su propia plataforma Moodle, llamada CVUEx, que facilita a los distintos participantes de la comunidad universitaria espacios virtuales propios.

En este anexo se describirá el proceso de instalación de Moodle 3.3+ en el sistema operativo utilizado para el desarrollo del Trabajo Fin de Grado, Debian 8.0. Es necesario tener instalado previamente XAMPP, y tener creada una base de datos para Moodle. Todos los comandos se deberán realizar siendo superusuario o con la palabra "sudo" delante del comando.

- En primer lugar, es necesario descargar el paquete de instalación de Moodle de su web oficial https://download.moodle.org/releases/latest/. En la página web, se deberá buscar la versión 3.3+, en este momento la versión más actual, y pulsar en "Download tgz".

- Una vez finalizada la descarga, es necesario abrir un terminal y situarnos en la carpeta donde se encuentra el archivo descargado mediante el comando '*cd <rutaAlArchivo>*'. Después, se deberá crear una carpeta mediante el comando '*mkdir <nombreDeLaCarperta>*' . Se deberá descomprimir el archivo descargado, por lo que primero debemos situarnos en la nueva carpeta creada y, seguidamente, se deberá ejecutar el comando '*cp -R <nombreDelArchivoDescargado> .*' Este comando copiará el archivo descargado en la carpeta creada. Ahora, se deberá descomprimir el archivo mediante el comando '*tar xzvf <nombreDelArchivoDescargado>*', que creará una carpeta llamada "moodle".

- Se deberá crear una carpeta con el nombre que se quiere dar a la aplicación Moodle, en la ruta /opt/lampp/htdocs/ mediante el comando '*mkdir /opt/lampp/htdocs/<nombreMoodle>*'. Se copiará todo el contenido de la carpeta "moodle" en el nuevo directorio creado, mediante el comando '*cp -R * /opt/lampp/htdocs/<nombreMoodle>*'.

- Nos deberemos situar en la carpeta creada en el paso anterior y otorgarle los permisos de lectura, escritura y modificación mediante el comando '*chmod 777 **'.

- Por último, antes de comenzar con la instalación, se deberá crear un directorio de datos para Moodle. Para ello se ejecutará el comando '*mkdir /opt/lampp/moodledata*'. Ade-

más, deberá tener los permisos de lectura, escritura y modificación, por lo que ejecutamos el comando '*chmod 777 /opt/lampp/moodledata*'.

- Ya se podrá comenzar con la instalación de Moodle. Para ello, se deberá abrir el navegador y acceder a la dirección "IPDeLaMaquinaMoodle/<nombreMoodle>".

- Al acceder a esta dirección, aparecerá el proceso de instalación. El primer paso es seleccionar el idioma de la instalación, así como el idoma por defecto en el que aparecerá la plataforma Moodle. Se deberá seleccionar "Español-Internacional(es)" y, una vez elegido, el botón "Siguiente".

- En la siguiente ventana aparecerá la dirección web para Moodle, el directorio donde se encuentra instalado y el directorio de datos de Moodle. Por defecto aparecerá los valores configurados anteriormente, por lo que se podrá continuar sin necesidad de cambiarlos pulsando en "Siguiente".

- En esta nueva ventana, se deberá elegir la base de datos que Moodle utilizará, seleccionando la opción "MariaDB(nativo/mariadb)".

- Una vez pulsado el botón "Siguiente", aparecerá una nueva ventana en la cual hay que configurar la base de datos donde se almacenarán la mayoría de los datos de Moodle. Los datos solicitados son la IP del servidor de la base de datos, el nomre de la base de datos creada para Moodle, un usuario de la base de datos con todos los permisos y la contraseña de este usuario. Una vez rellenado correctamente, se deberá pulsar el botón "Siguiente".

- En la siguiente ventana aparecerá el fichero de configuración creado por Moodle. Es necesario copiar el código que aparece en la ventana en un archivo llamado "config.php" en el directorio raíz de Moodle. Para eso, se deberá volver al terminal y ejecutar el comando '*nano /opt/lampp/htdocs/<nombreMoodle>/config.php*'. Aparecerá en el terminal una nueva ventana en la cual se deberá pegar el código anterior. Una vez pegado, se deberá pulsar "Control+X". Nos preguntará si se desea guardar los cambios, a lo que se tendrá que contenstar con "s" y pulsar "Intro". Una vez finalizado este proceso, se deberá ejecutar el comando '*chmod 777 /opt/lampp/htdocs/<nombreMoodle>/config.php*', se deberá volver al navegador y pulsar en "Siguiente".

- Aparecerá una nueva ventana con los términos y condiciones de Moodle. Se deberán aceptar pulsando en "Continuar".

- En la nueva ventana, aparecerá una lista con las distintas comprobaciones del servidor, y si todo está correcto, aparecerá el botón "Continuar" en la parte inferior.

- Comenzará ahora el proceso de instalación de Moodle. Una vez haya finalizado, aparecerá el botón "Continuar".

- En esta nueva ventana, aparecerá un formulario para rellenar con los datos solicitados del administrador (nombre de usuario, contraseña, nombre del administrador, apellidos del administrador, ciudad, entre otros). Una vez completado el formulario se deberá pulsar el botón "Actualizar información personal".

- Aparecerá una nueva ventana, en la cual se pide el nombre completo del sitio web, el nombre corto de una sola palabra y la descripción. Para finalizar, se deberá pulsar en "Guardar cambios".
- Ya estará instalada y configurada la plataforma Moodle, en la cual se podrán crear los diferentes espacios para los miembros de la comunidad universitaria, y se podrá administrar la plataforma.

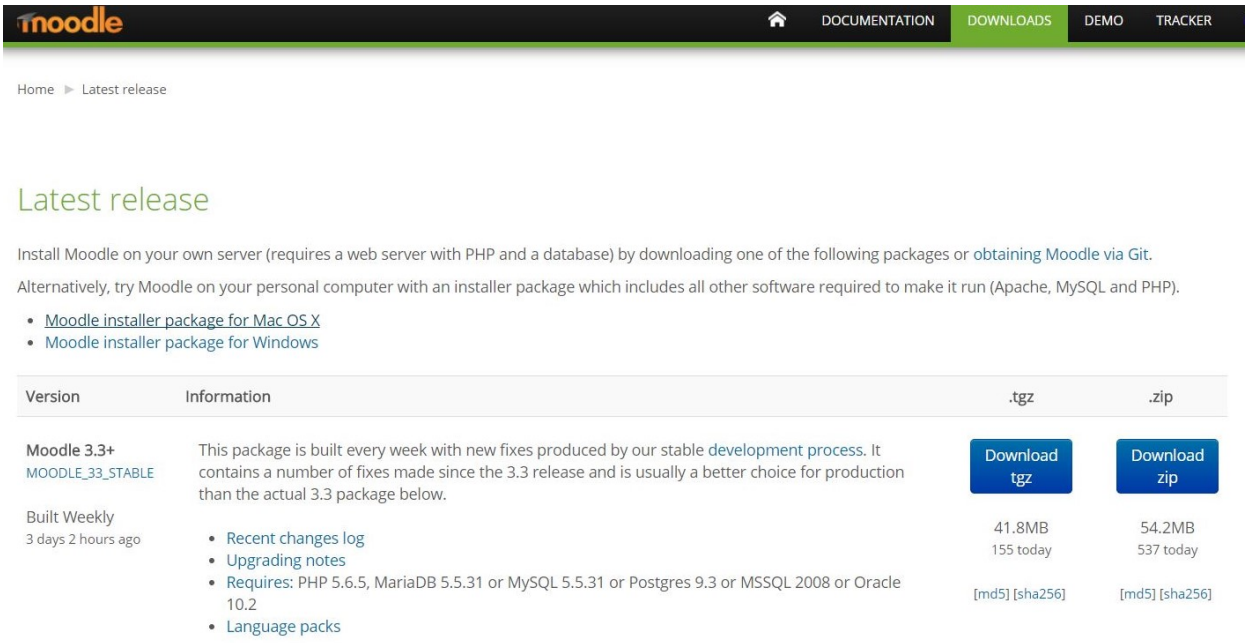

Figura 10.1: Web oficial. Descarga de Moodle.

Figura 10.2: Instalación de Moodle. Selección del idioma.

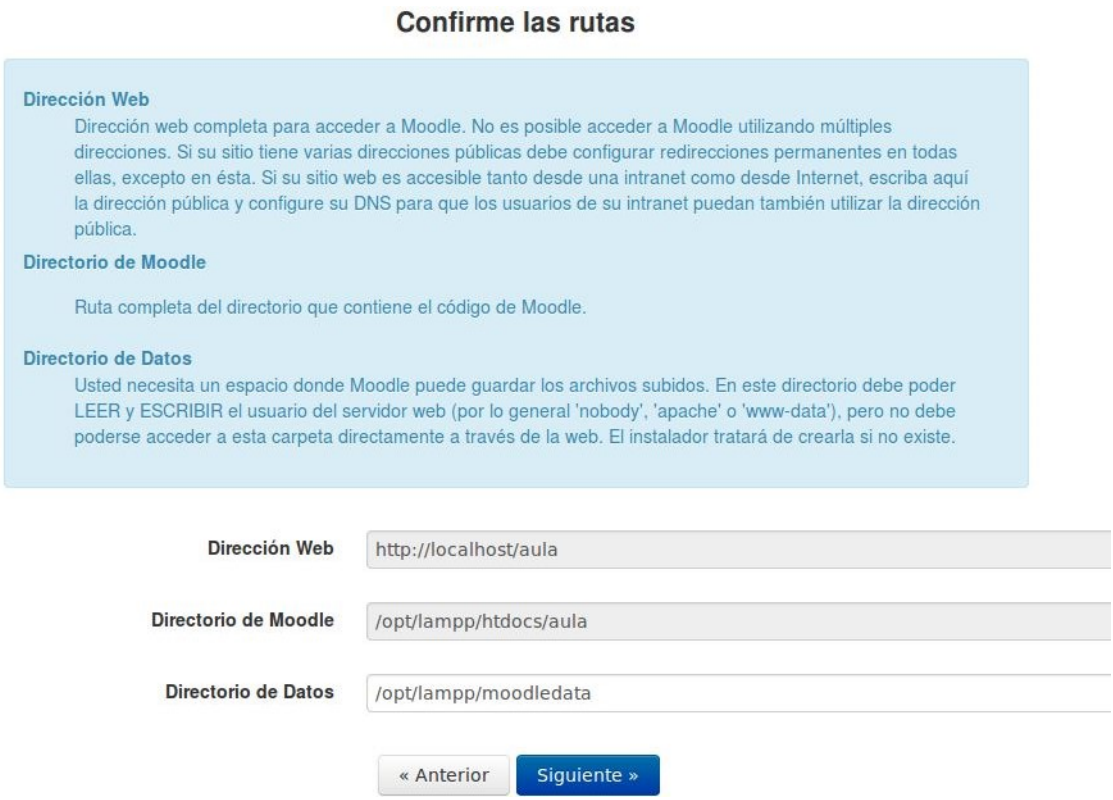

Figura 10.3: Instalación de Moodle. Confirmación de las rutas.

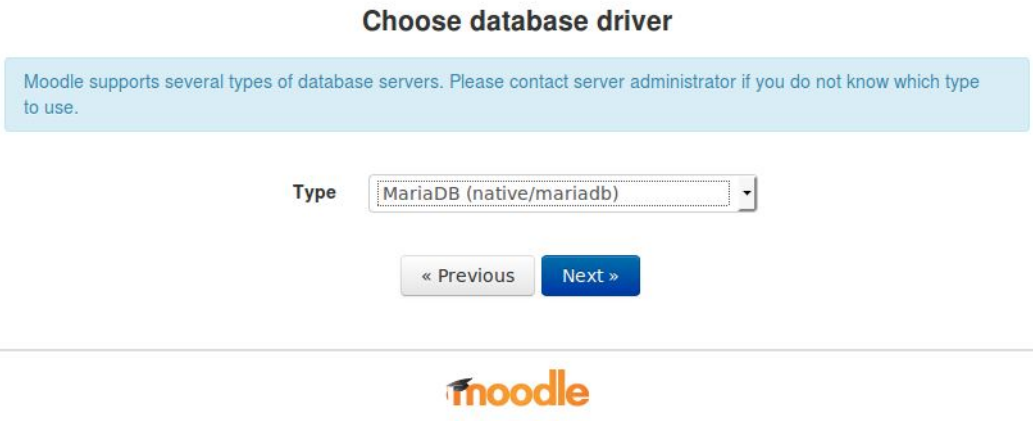

Figura 10.4: Instalación de Moodle. Selección de la base de datos.

Figura 10.5: Instalación de Moodle. Ajuste de la base de datos.

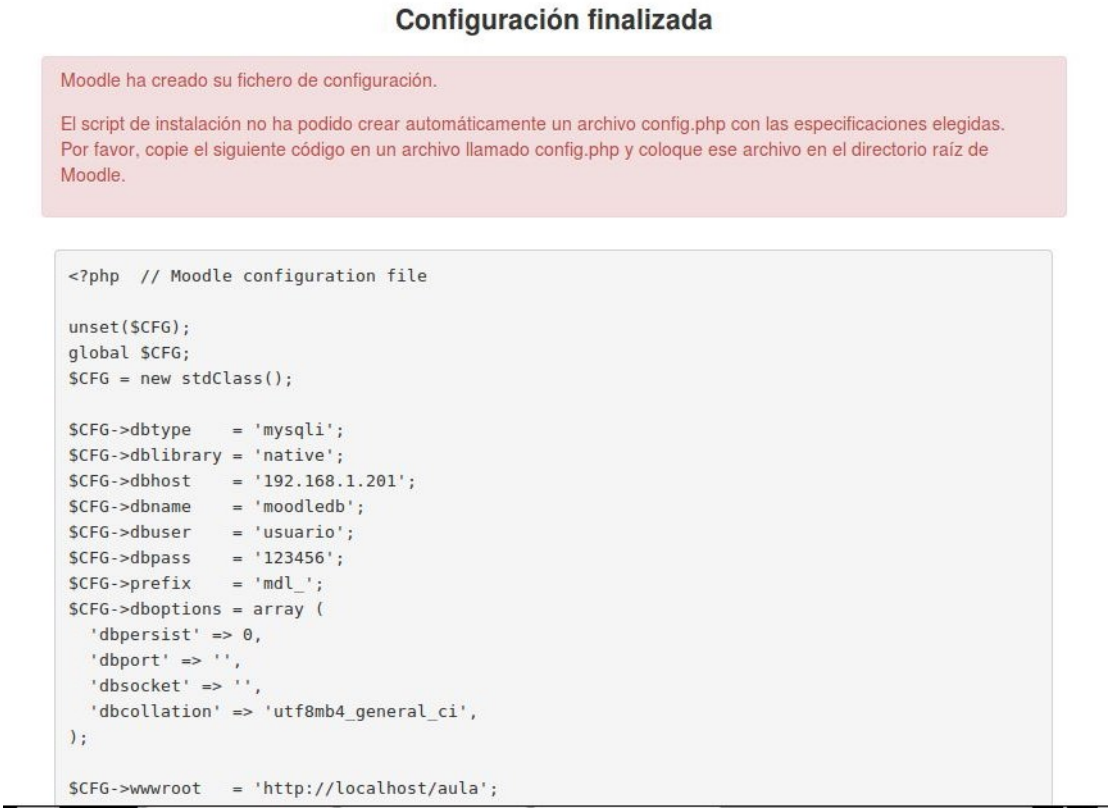

Figura 10.6: Instalación de Moodle. Fichero de configuración.

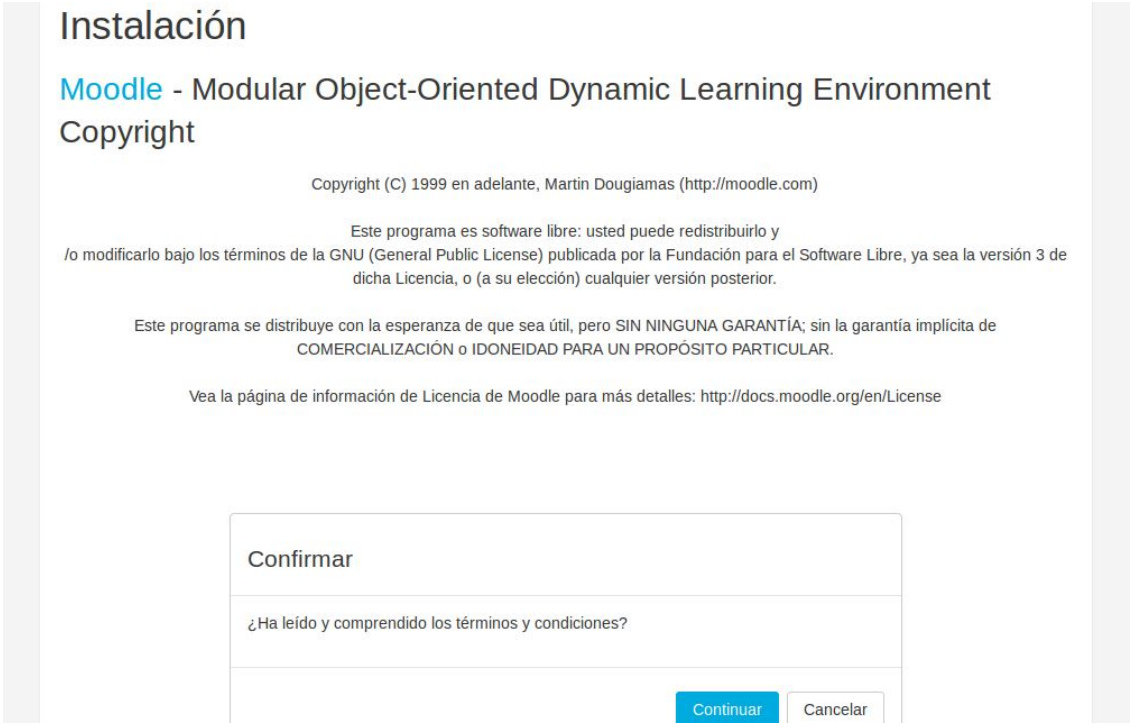

Figura 10.7: Instalación de Moodle. Términos y condiciones.

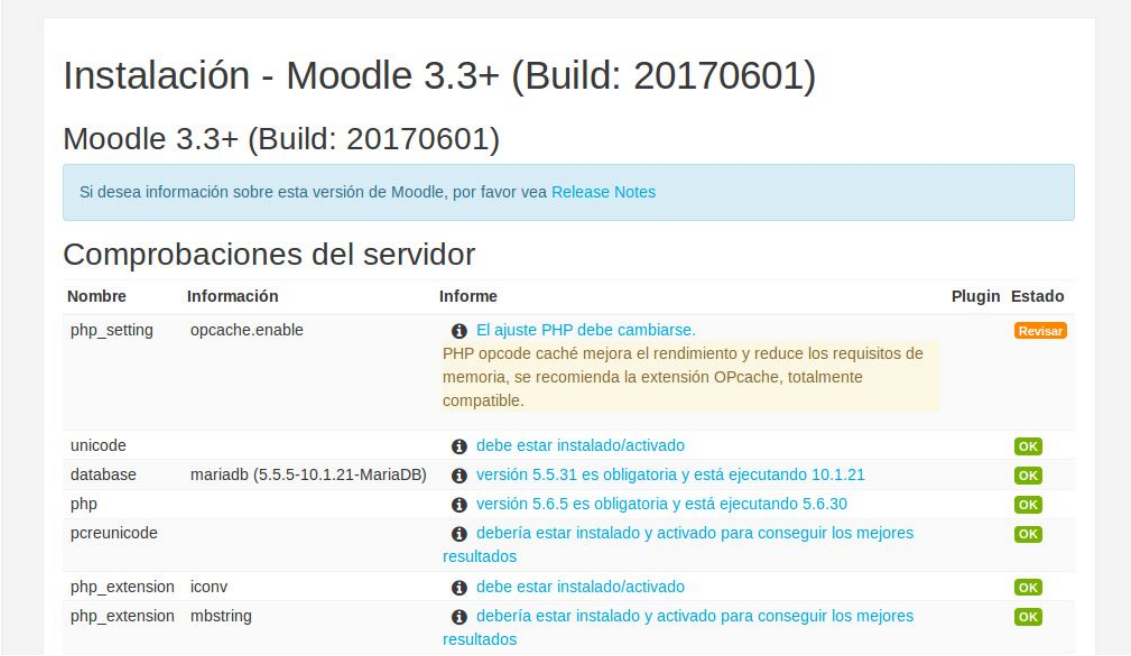

Figura 10.8: Instalación de Moodle. Comprobaciones del servidor.

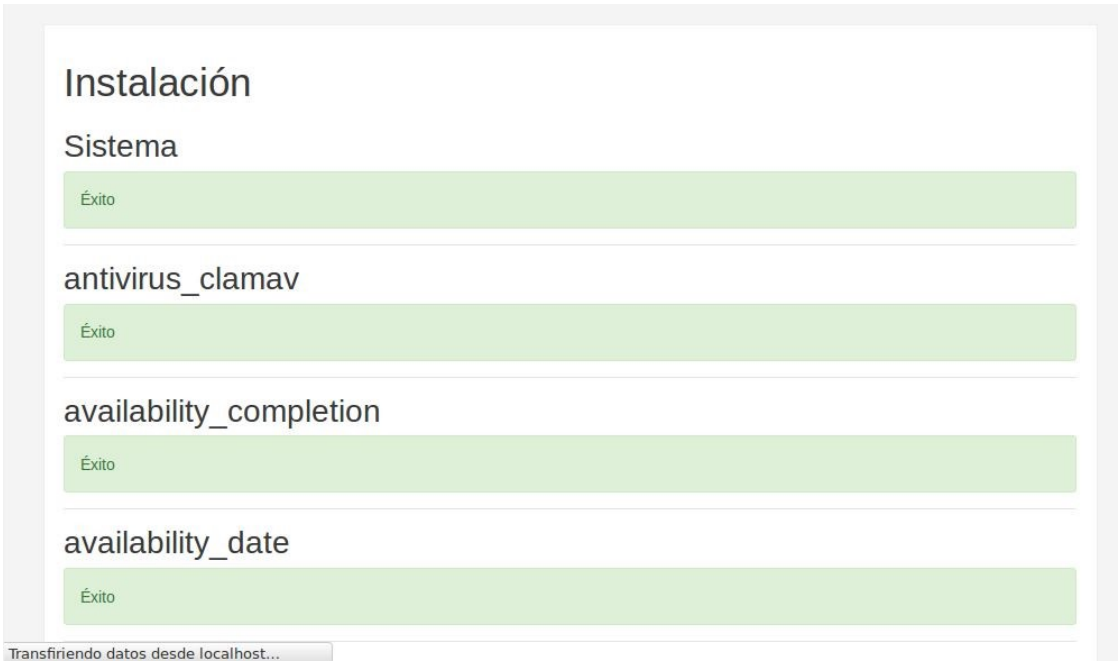

Figura 10.9: Instalación de Moodle.

Figura 10.10: Instalación de Moodle. Configuración del administrador.

Figura 10.11: Instalación de Moodle. Configuración del sitio.

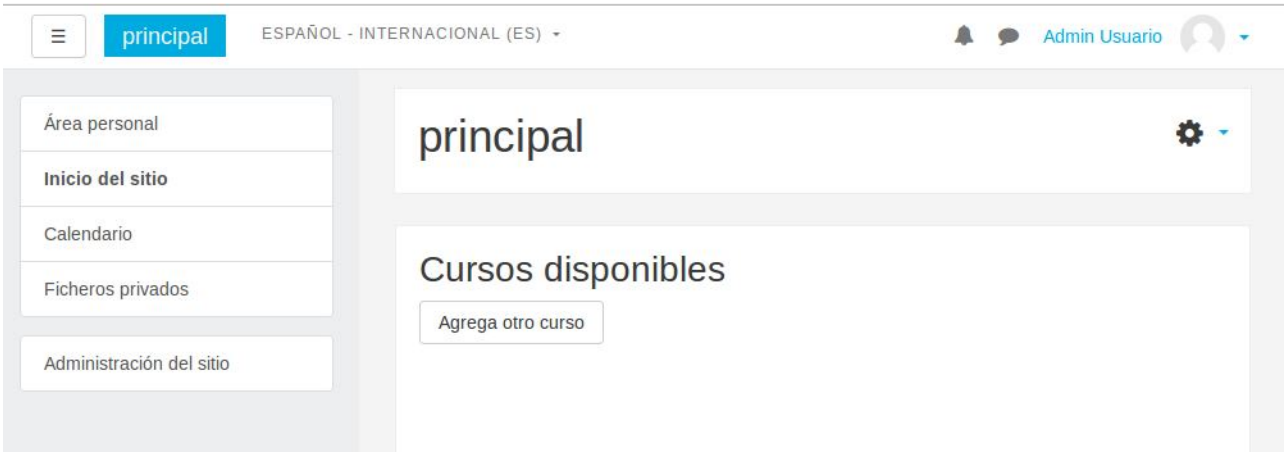

Figura 10.12: Fin de la instalación de Moodle.

Capítulo 11

Anexo III. Instalación de ownCloud

ownCloud es un software libre que permite crear un servidor donde alojar archivos. En él, los distintos miembros de la comunidad universitaria, podrán subir sus propios archivos, así como acceder a los archivos de los demás si tienen los permisos necesarios.

En este anexo se describirá el proceso de instalación de ownCloud v10.0.2 en el sistema operativo utilizado para el desarrollo del Trabajo Fin de Grado, Debian 8.0. Es necesario tener instalado previamente XAMPP, y tener creada una base de datos para ownCloud. Todos los comandos se deberán realizar siendo superusuario o con la palabra "sudo" delante del comando.

- En primer lugar, es necesario descargar el paquete de instalación de ownCloud de su web oficial https://owncloud.org/install/#edition. En la página web, se deberá buscar la opción de descargar el archivo ".tar.bz2" y seleccionarla. Una vez hecho comenzará la descarga.

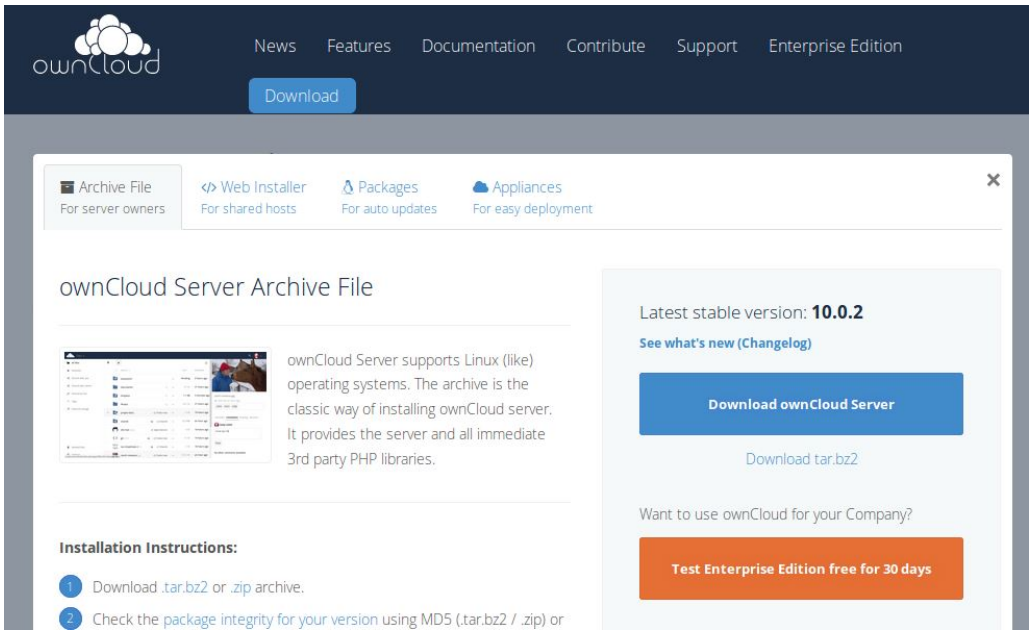

Figura 11.1: Página oficial de ownCloud. Descarga.

- Una vez finalizada la descarga, es necesario abrir un terminal y situarnos en la carpeta donde se encuentra el archivo descargado mediante el comando *'cd <rutaAlArchivo>'*. Después, se deberá descomprimir el archivo descargado, mediante el comando *'tar -xvf <archivoDescargado>.tar.bz2 /opt/lampp/htdcocs'*. Después, se identificará al usuario que se esté utilizado en Debian como propietario de la carpeta owncloud generada mediante el comando *'chown -R <usuarioDebian>:www-data /opt/lampp/htdocs/owncloud'*.

- Se deberá modificar los permisos sobre el directorio de ownCloud mediante el comando *'chmod 777 -R /opt/lampp/htdocs/owncloud'*.

- El siguiente paso será abrir una ventana en el navegador con la ruta "http://IpServidorOwnCloud/owncloud". Se mostrará una ventana en la que se debe rellenar con los datos solicitados. Se deberá introducir un usuario administrador para ownCloud, su contraseña, el nombre de la base de datos para ownCloud, el usuario con todos los permisos de la base de datos, su contraseña y la dirección del servidor de la base de datos.

- Una vez completado el formulario anterior, se deberá pulsar en el botón "Completar la instalación". Una vez hecho ya estará instalado ownCloud.

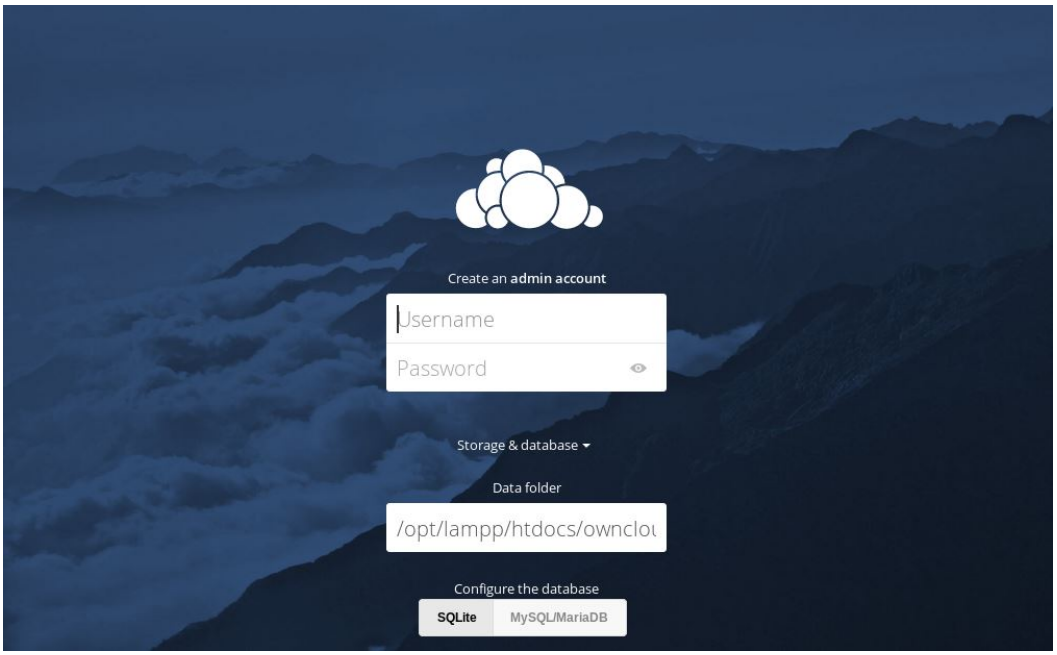

Figura 11.2: Instalación de ownCloud. Datos del administrador.

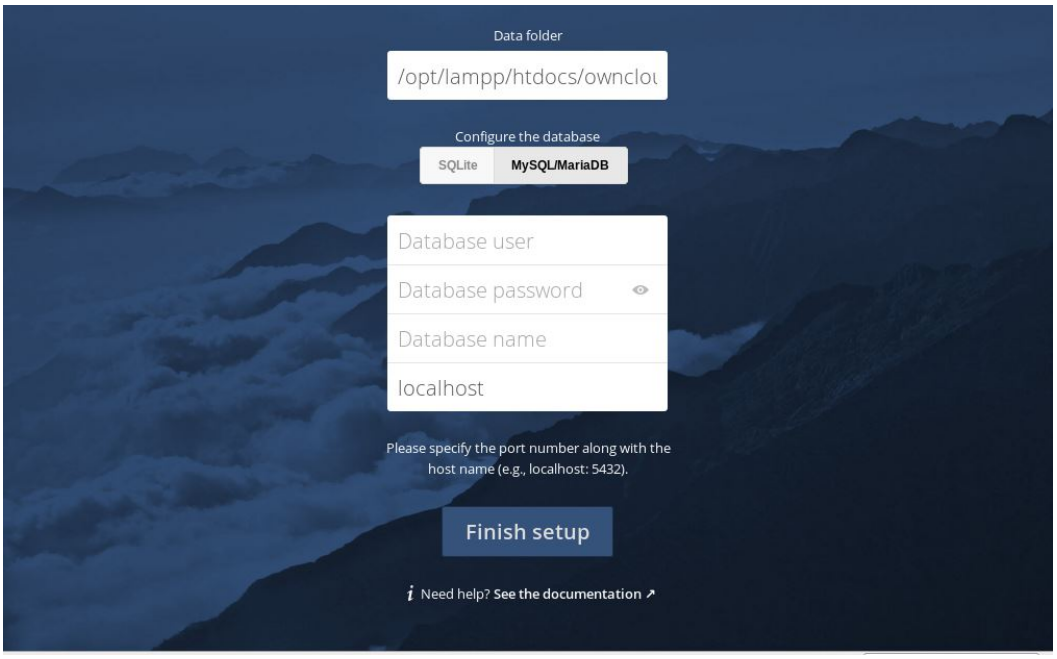

Figura 11.3: Instalación de ownCloud. Información de la base de datos de ownCloud.

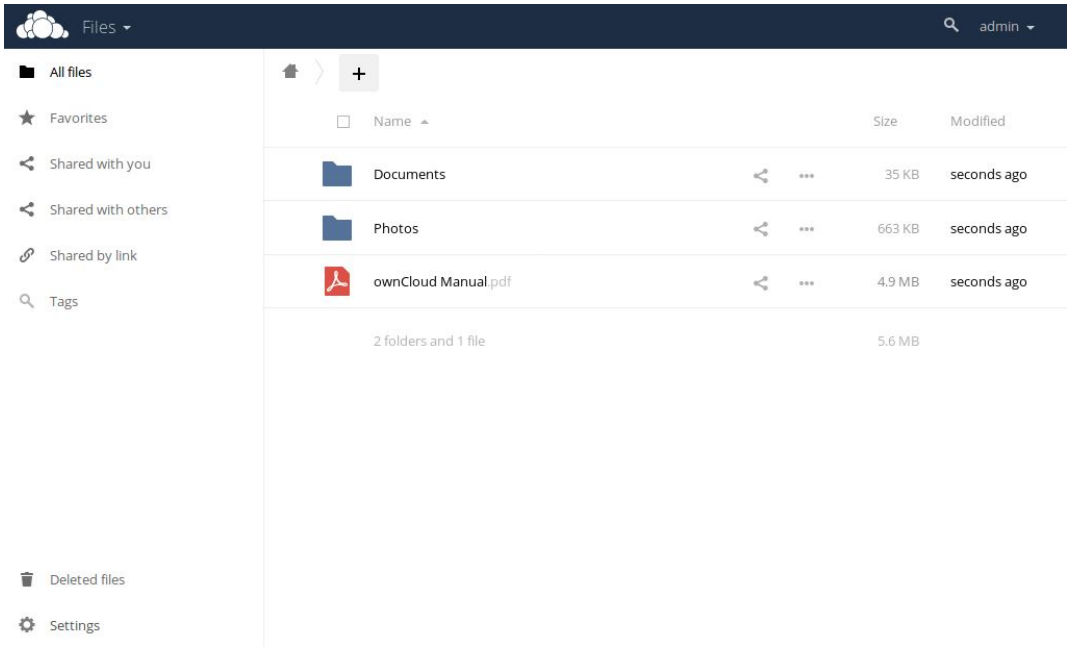

Figura 11.4: Instalación de ownCloud. Fin de la instalación.

Capítulo 12

Anexo IV. Instalación de LDAP

LDAP es un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado de manera jerárquica para buscar información en un entorno de red.

En este Trabajo Fin de Grado se ha utilizado LDAP para la autentificación de los distintos usuarios en las aplicaciones realizadas, ya que es la base de datos que utiliza la Universidad de Extremadura para iniciar sesión en Campus Virtual.

En este anexo se describirá el proceso de instalación de OpenLDAP en el sistema operativo utilizado para el desarrollo del Trabajo Fin de Grado, Debian 8.0. Todos los comandos se deberán realizar siendo superusuario o con la palabra "sudo" delante del comando.

- En primer lugar, es necesario modificar el archivo "hosts" para agregar la IP del servidor LDAP, así como el dominio LDAP. Para ello, se deberá ejecutar el comando '*nano /etc/hosts*' y se abrirá en el terminal el archivo. Ahora es necesario modificar el archivo, borrando si existe la línea "127.0.1.1 <nombreDelEquipo>" y poniendo en su lugar "<IPdelServidorLDAP> <nombreDelEquipo>.<nombreDelDominio> <nombreDelEquipo>". Una vez realizada esta modificación, pulsamos "Control+X" y después "s" para que se guarden los cambios. En la figura 12.1, se puede observar un ejemplo del archivo "host" sin modificar, y en la figura 12.2, este archivo ya modificado.

- El siguiente paso es instalar los paquetes del servidor OpenLDAP ejecutando el comando '*apt-get install slapd ldap-utils -y*'.

- En el proceso de la instalación, como se puede observar en la figura 12.3, se pedirá una contraseña para el administrador de LDAP. Se introducirá la contraseña que se desee y se pulsará "Intro". Ahora se pedirá que se confirme la contraseña, por lo que se volverá a introducir la contraseña introducida anteriormente.

- Tras finalizar la instalación, se podrá ejecutar el comando '*slapcat*' para visualizar si se ha instalado correctamente.

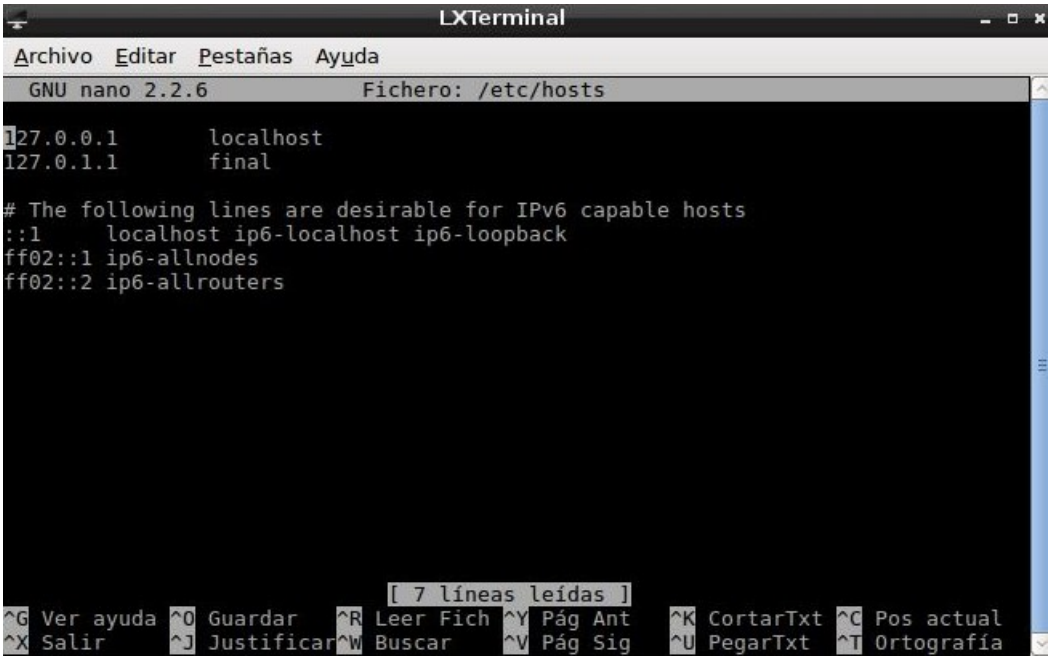

Figura 12.1: Ejemplo de /etc/hosts original.

Figura 12.2: Ejemplo de /etc/hosts modificado.

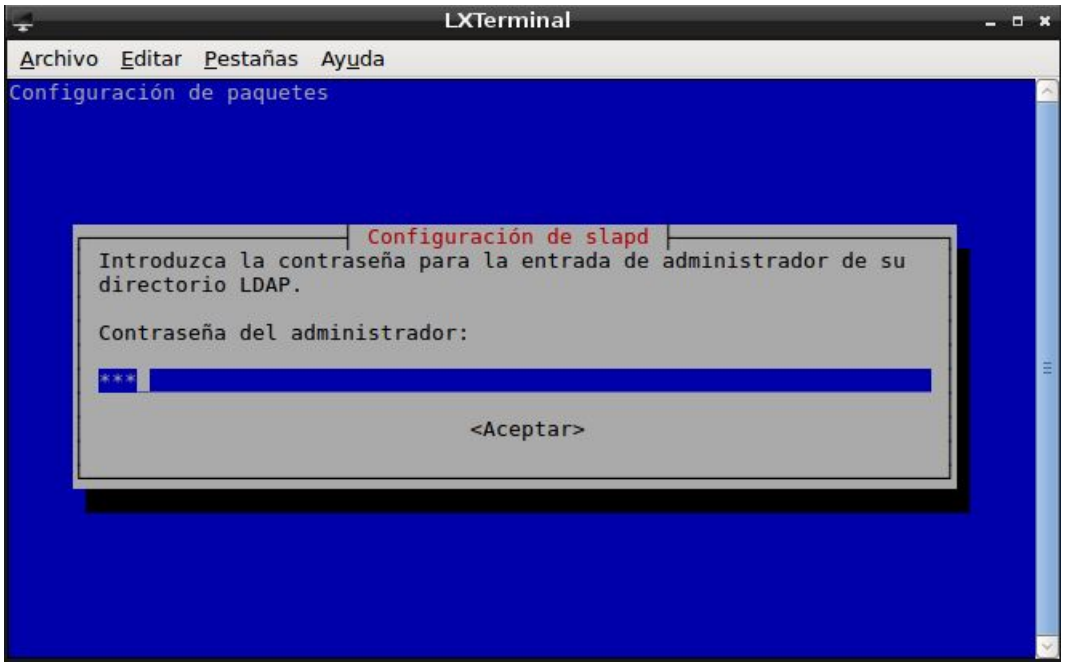

Figura 12.3: Instalación LDAP. Insertar contraseña para el administrador.