UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

YourStudies: Plataforma de Aprendizaje en línea

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Mónica Romero Mayordomo

Tutor: Sergio Sáez Barona

Curso 2020-2021

Resum

A causa de la creixent necessitat de realitzar classes telemàtiques per part dels diversos equips docents, aquest projecte pretén abordar el desenvolupament d'una aplicació perquè, tant professorat com alumnat, puga compartir recursos entre si i, d'aquesta manera, proporcionar una alternativa en línia per a l'avaluació de coneixements.

Per a aconseguir aquest propòsit, s'han analitzat diverses aplicacions similars del mercat actual i s'han definit els objectius i les tecnologies a utilitzar. Les més destacables van ser Django i PostgreSQL. A més, s'han desenvolupat diversos diagrames de cas d'ús per a especificar les funcionalitats de l'aplicació, com són l'administració d'usuaris i assignatures, o la creació, edició i correcció de tasques per part del professorat. Així mateix, l'aplicació s'ha plantejat perquè puga ser accessible des de qualsevol lloc i dispositiu, ja siga mòbil o ordinador de sobretaula.

Després de realitzar diverses proves, el principal resultat d'aquest projecte és una aplicació adaptativa a multitud de dispositius que compta amb les funcions bàsiques per a una correcta educació i avaluació de l'alumnat.

Paraules clau: aprenentatge, recursos, aplicació, Django

Resumen

Debido a la creciente necesidad de realizar clases telemáticas por parte de los diversos equipos docentes, este proyecto pretende abordar el desarrollo de una aplicación para que, tanto profesorado como alumnado, pueda compartir recursos entre sí y, de este modo, proporcionar una alternativa on-line para la evaluación de conocimientos.

Para conseguir este propósito, se han analizado varias aplicaciones similares del mercado actual y se han definido los objetivos y las tecnologías a utilizar. Las más destacables fueron Django y PostgreSQL. Además, se han desarrollado diversos diagramas de caso de uso para especificar las funcionalidades de la aplicación, como son la administración de usuarios y asignaturas, o la creación, edición y corrección de tareas por parte del profesorado. Asimismo, la aplicación se ha planteado para que pueda ser accesible desde cualquier lugar y dispositivo, ya sea móvil u ordenador de sobremesa.

Después de realizar diversas pruebas, el principal resultado de este proyecto es una aplicación adaptativa a multitud de dispositivos que cuenta con las funciones básicas para una correcta educación y evaluación del alumnado.

Palabras clave: aprendizaje, recursos, aplicación, Django

Abstract

Due to the increasing need for telematic classes by the various teaching teams, this project aims to address the development of an application so that both teachers and students can share resources with each other and thus provide an on-line alternative for the evaluation of knowledge.

To achieve this purpose, several similar applications in the current market have been analyzed and the objectives and technologies to be used have been defined. The most important ones were Django and PostgreSQL. In addition, several use case diagrams have been developed to specify the functionalities of the application, such as the administration of users and subjects, or the creation, edition and correction of assignments by the teaching staff. The application has also been designed to be accessible from any location and device, whether mobile or desktop.

After several tests, the main result of this project is an application adaptable to a multitude of devices that has the basic functions for a correct education and evaluation of students.

Key words: learning, resources, application, Django

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

A lo largo de estos últimos años ha habido un auge en el uso de tecnologías en diferentes apartados de nuestras vidas. Entre ellos se encuentra la pedagogía, en la cual los docentes intentan proporcionar lecciones lo más entretenidas y dinámicas posibles para captar fácilmente la atención de sus estudiantes mediante el uso de diapositivas o cuestionarios a tiempo real en el aula. Por otra parte, cada vez se hace más necesaria la entrega telemática de recursos debido a la enseñanza de las nuevas tecnologías digitales, como son ofimática o diseño gráfico, y para poder organizar y planificar de forma sencilla las clases que se impartirán en un futuro.

Dado el interés en esta cuestión, en este trabajo se pretende diseñar e implementar una plataforma web para facilitar la distribución y administración de recursos utilizados en diferentes asignaturas, de forma que ayude al profesorado a impartirlas en el aula o telemáticamente. Además, la plataforma se podrá consultar desde cualquier dispositivo, como un móvil o un ordenador de sobremesa, para que los estudiantes no tengan problemas para consultar documentación en cualquier momento.

1.1 Motivación

El principal motivo por el cual se ha desarrollado este trabajo es la dificultad que existía para entregar tareas o recursos al profesorado en el instituto en el que realicé mis estudios secundarios y de bachillerato. Se producían muchas ocasiones en las que si un alumno olvidaba el trabajo realizado en su hogar, implicaba que no lo había entregado con sus pertinentes consecuencias. Por otro lado, si un alumno faltaba a una clase en la que el profesor había entregado algún recurso (como un folio con actividades a realizar o temario complementario) necesario para su posterior aprendizaje, podía suponer una pérdida del tiempo de estudio del alumno, o incluso un fallo grave en un examen.

Otro de los motivos por los que decidí realizar este proyecto fue la reciente llegada del virus *SARS-CoV-2*, el cual supuso una paralización total de la educación en todo el territorio español durante los primeros meses, lo que más adelante se convirtió en una necesidad de tener plataformas para poder realizar un aprendizaje online. Esta situación supuso una oportunidad ideal para la creación de una plataforma en la que los centros educativos tuvieran una plataforma propia para poder manejar la pedagogía de forma óptima en esta difícil situación, y que permitiese al profesorado impartir correctamente y sin inconvenientes sus asignaturas.

Por último, con la creación de este proyecto, se pretende adquirir nuevos conocimientos relativos a tecnologías sobre desarrollo de aplicaciones web que sean adaptables a

todo tipo de dispositivos y que sean accesibles desde cualquier lugar, lo que me ayudará a desarrollarme profesionalmente en este ámbito.

1.2 Objetivos

Como se ha comentado anteriormente, el objetivo principal es desarrollar una aplicación web para facilitar la distribución y administración de recursos utilizados en las diferentes materias de los centros de enseñanza, de forma que facilite la enseñanza de las asignaturas al cuerpo docente y al alumnado.

Para poder llevarlo a la práctica, definimos los siguientes objetivos específicos:

- Implementar un sistema de registro de asignaturas, profesorado y alumnado de fácil aprendizaje y uso.
- Diseñar e implementar una base de datos relacional proporcionando persistencia al modelo de datos de forma que almacene correctamente todos los datos necesarios sobre los usuarios de la aplicación.
- Evitar pérdida de documentación por parte de los alumnos o profesorado manteniendo todos los recursos en la nube.
- Publicar la aplicación web de forma que sea accesible desde cualquier sitio y momento siempre y cuando se disponga de una conexión a Internet estable.

1.3 Impacto esperado

El mayor impacto esperado de este proyecto es mejorar la distribución de recursos entre el cuerpo docente y los estudiantes para evitar problemas como pérdida de material educativo, ahorrar tiempos de estudio u optimizar la organización.

Para mostrar un ejemplo, tenemos dos tipos de usuarios, profesores y alumnos. El profesorado, en el sistema educativo clásico, solamente puede distribuir material en papel, como un trabajo corregido o ejercicios a realizar, presencialmente a sus alumnos. Si un alumno faltase a esa clase, perdería un tiempo de estudio valioso, en el caso que sea material educativo nuevo, o no podría examinar a tiempo la corrección dada por el profesor. Mediante esta aplicación se desea solucionar este problema, de forma que los recursos que quiera añadir el profesor como materia adicional a la dada en los libros de texto, puede subirla a la asignatura o trabajo correspondiente, por lo que los alumnos podrán verlo en cualquier momento.

Otra ventaja es la mejora en la planificación para realizar tareas escolares. El alumnado dispondrá de más tiempo para entregar una tarea ya que el profesor puede elegir la hora y el día exactos en el que quiere que se cierre, facilitando así una correcta planificación para el estudio. Además, cómo cualquier recurso está en la nube, no hay posibilidad de pérdida de estos por factores externos como el transporte.

1.4 Estructura de la memoria

El contenido de esta memoria se organiza en 9 capítulos, en los que se abordarán los siguientes temas:

- **Capítulo 1 - Introducción.** En este capítulo se presenta la problemática que aborda el proyecto, en el que se definen los conceptos principales, así como los objetivos que se pretenden conseguir y las mejoras que supondrá el trabajo resultante.
- **Capítulo 2 - Estudio de mercado.** Esta sección documenta las diferentes aplicaciones que existen en el mercado actual relacionadas con nuestro proyecto. Se hablará sobre las más utilizadas por los centros docentes. Por otro lado, se analizarán los trabajos presentados anteriormente y se realizará un estudio crítico resaltando las diferencias y similitudes entre estos, así como un análisis DAFO para evaluar el riesgo de nuestra aplicación en el mercado actual.
- **Capítulo 3 - Análisis del problema.** Se plantearán distintas soluciones con las cuales desarrollar nuestro trabajo. También se presentarán los diferentes casos de uso que describen las funcionalidades de nuestro proyecto, así como un modelo entidad-relación que definirá el diseño de la base de datos.
- **Capítulo 4 - Diseño de la solución.** En este apartado se detallarán los diferentes elementos que componen la arquitectura del sistema de nuestra aplicación, del mismo modo que se explicarán las diferentes tecnologías utilizadas. Además se incluirán los prototipos de la aplicación y los diagramas UML de clases y de la base de datos.
- **Capítulo 5 - Desarrollo de la solución propuesta: YourStudies.** Se detallarán los problemas y dificultades surgidas en la realización de la aplicación, incidiendo en las soluciones planteadas para cada problemática.
- **Capítulo 6 - Implantación.** Este capítulo aborda la instalación de los frameworks con los que se desarrollará el proyecto, así como la puesta en producción para las posteriores pruebas de este.
- **Capítulo 7 - Pruebas.** En este apartado se presentarán las pruebas que se hayan realizado para corroborar que la solución implementada funciona correctamente. También se mostrarán los resultados finales del proyecto.
- **Capítulo 8 - Conclusiones.** Se abordarán los resultados finales de nuestra aplicación, de forma que quede claro que se ha aprendido y las dificultades de todo el desarrollo. Igualmente se valorará la relación de este trabajo respecto a los estudios cursados.
- **Capítulo 9 - Trabajos futuros.** Por último, se propondrán diversas mejoras o nuevas funcionales para futuras versiones de la aplicación, que se podrán implementar posteriormente a la finalización de este trabajo.

CAPÍTULO 2
Estudio de mercado

En el mercado actual podemos encontrar un amplio abanico de aplicaciones destinadas a la formación online y al aprendizaje. En este capítulo se pretende dar una visión general sobre ellas, describiendo sus funcionalidades principales con las que recopilaremos las más usadas y, seguidamente, realizar una priorización de tareas para nuestra aplicación mediante el método MoSCoW. Adenás de esto, se realizará un estudio de riesgo del mercado actual mediante un análisis DAFO. Por último, tomando como referencia los resultados obtenidos del análisis con esta metodología, realizaremos una propuesta final con las funcionalidades más importantes que se incorporarán a nuestra aplicación.

2.1 Concepto de LMS o Sistema de gestión de aprendizaje

Un LMS (*Learning Management System)*[1] es un sistema de gestión de aprendizaje online, el cual permite administrar, distribuir, monitorizar y evaluar las diversas actividades previamente diseñadas para proporcionar una formación totalmente virtual o semipresencial.

Están diseñados para que sean fácilmente accesibles e intuitivos de forma que lo pueda utilizar cualquier tipo de usuario, ya sean administradores, profesorado o alumnado, además de permitir utilizarlos en cualquier momento y lugar, siempre que se tenga una conexión a Internet. Por otro lado, impulsan la interacción online entre los usuarios de un mismo proceso de aprendizaje. Existen dos tipos de plataformas LMS[2]:

2.1.1. Bajo licencia (SaaS)

Este tipo de plataformas requieren una suscripción para su uso. Una vez adquieres la licencia se deberá crear el curso de formación que se desee. En la gran mayoría de casos, no se necesitará ningún tipo de instalación pues todo el software se encuentra en los servidores de la empresa desarrolladora del producto.

Dependiendo del plan contratado, tendrá unas características u otras según la oferta de la empresa, lo que proporciona flexibilidad a la hora de escoger lo necesario para personalizar el curso.

[1] https://www.avanzo.com/lms-que-es-como-funciona/
[2] https://rockcontent.com/es/blog/plataforma-lms/

2.1.2. Recurso educativo abierto (Open Source)

Son aquellas plataformas cuyo código es libre para su uso, por lo que cualquiera puede instalar, modificar y utilizar sin tener que pagar una licencia. De forma general, necesitarán una instalación en un servidor propio, lo cual generará gastos al requerir un equipo técnico propio que gestione la plataforma personalizada.

Estos proyectos plantean mayor libertad que los anteriores, pues no poseen las restricciones que tiene un producto comercial.

2.2 Aplicaciones similares

En este apartado se mostrarán las diferentes herramientas o aplicaciones de la misma temática que nuestro proyecto que existen en el mercado. Para ello, se realizará una pequeña descripción de cada una de ellas y se nombrarán sus funcionalidades principales, de forma que podamos analizarlas y extraer ideas para las funcionalidades de nuestra futura aplicación.

2.2.1. Moodle

Es una aplicación web de código abierto diseñada para crear y gestionar espacios de aprendizaje online adaptados a las necesidades de profesores, estudiantes y administradores. Está basada en la tecnología *PHP* y utilizada la base de datos *MySQL*.

Su primera versión fue desarrollada por el pedagogo Martin Dougiamas en el año 2002, y su nombre proviene del acrónimo *Module Object-Oriented Dynamic Learning Environment*, que en español significa "Entorno Modular de Aprendizaje Dinámico Orientado a Objetos".

Según sus propias estadísticas, a día de hoy es de las más utilizadas del mundo, con más de 192000 sitios creados. Concretamente, en España existen 15400 aplicaciones Moodle creadas aproximadamente.

Top 10 from 247 countries by registrations

Country	Registered sites
Spain	15,436
United States	14,087
Mexico	13,377
Germany	9,873
Brazil	9,412
India	6,643
France	6,556
Indonesia	6,398
Colombia	6,107
Russian Federation	5,975

Figura 2.1: Estadísticas de los diez países que más utilizan Moodle

2.2.1.1. Funcionalidades principales

En el apartado de administración, esta aplicación nos permite gestionar diferentes accesos y roles para cada uno de los tipos de usuario de la aplicación, como profesores y alumnos.

Asimismo, en la pantalla principal podemos ver un listado de las asignaturas o curso que imparte un profesor. En ellos se puede ver los participantes de este, como también las notas de cada estudiante, el temario y si el curso ha sido premiado en algún momento. También tenemos un repositorio de ficheros propio en el que podemos subir todo tipo de ficheros.

Adicionalmente, disponemos de un apartado de noticias, en el que se pueden ver las noticias más recientes de la administración. Así mismo, contiene un calendario en el que se pueden añadir, editar o eliminar eventos, para que posteriormente aparezcan en la *timeline* en la pantalla de inicio. Además de esto, contiene una sección de mensajería, en la que podemos escribir a diferentes usuarios por medio de un chat. Por último, se puede visualizar el perfil, así como editarlo en cualquier momento.

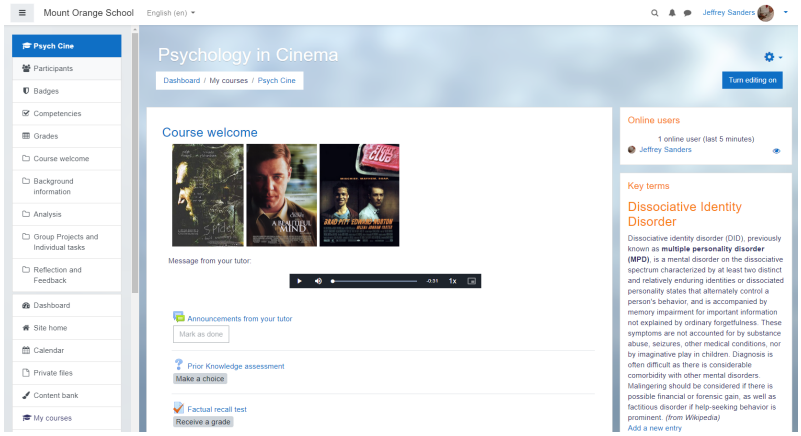


Figura 2.2: Pantalla principal de un curso en Moodle

2.2.2. Sakai

Es un software educativo de código abierto cuyo objetivo es crear un entorno de colaboración y aprendizaje para la educación superior, que mejore otras iniciativas como la anterior nombrada Moodle. Está basada en el lenguaje Java.

El proyecto tiene como origen la Universidad de Michigan, la Universidad de Indiana, el Instituto Tecnológico de Massachussetts, la Universidad Stanford, junto a la Iniciativa de Conocimiento Abierto (OKI) y el consorcio uPortal. La primera versión fue lanzada al mercado en 2005. Muchas universidades alrededor del mundo han adoptado este software como método adicional para enseñar y evaluar a sus estudiantes, entre ellas nuestra universidad, pues PoliformaT está basado en esta tecnología.

Figura 2.3: Logo de la aplicación Sakai

2.2.2.1. Funcionalidades principales

Al igual que Moodle, en la página principal disponemos de un listado de las asignaturas de las que somos partícipes, así como un calendario mensual en el que se visualizan los eventos programados. También podemos observar un tablón de anuncios de los últi-

mos 10 días publicados por los profesores de las asignaturas y las calificaciones de cada una de ellas, que podremos ordenar según diferentes filtros.

Una característica interesante es la posibilidad de crear exámenes tipo test o de respuesta abierta, los cuáles pueden ser publicados en el momento que sea necesario, sin ser obligatorio en la creación. Adicionalmente, la estructura de los cursos o asignaturas se organiza mediante la subida de ficheros o la creación de carpetas.

Por último, del mismo modo que la anterior aplicación, disponemos de un repositorio personal en el que subir cualquier fichero que se desee, además de poder editar diferentes aspectos de nuestro perfil.

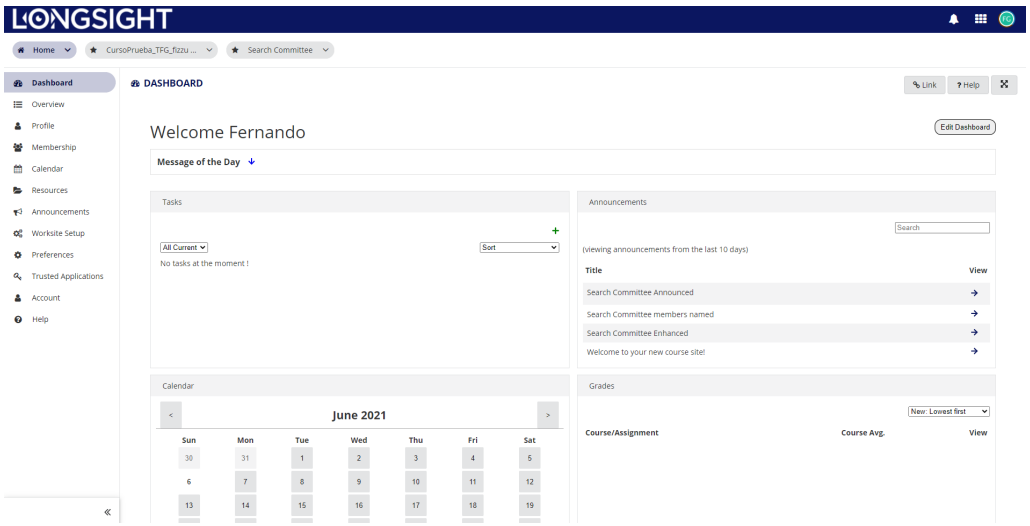


Figura 2.4: Pantalla principal de un curso en Moodle

2.2.3. Edmodo

Figura 2.5: Logo de la aplicación Edmodo

Es una plataforma pedagógica que permite la comunicación entre educadores y estudiantes de forma sencilla ya que su diseño está inspirado en las redes sociales. Edmodo es privativo, pero puede ser utilizado gratis por cualquier persona que pretenda usarlo en su centro docente.

Fue fundado en el año 2008 por Nick Borg, Jeff O'Hara, y Crystal Hutter, los cuales querían buscar una solución para que los estudiantes pudieran compaginar sus vidas privadas con sus estudios de una forma simple. Desde su lanzamiento, Edmodo tiene más de 100 millones de usuarios registrados por todo el mundo, y por encima de 700 millones de recursos compartidos.

2.2.3.1. Funcionalidades principales

Como hemos comentado anteriormente, su diseño está inspirado en las redes sociales, en concreto en el *microblogging*, por lo que la mayoría de usuarios lo encuentran sencillo de utilizar pues están acostumbrados a utilizar este tipo de aplicaciones diariamente. Esta aplicación permite crear un aula virtual a cualquier docente que esté registrado en la aplicación, como también crear tantos grupos como alumnos tengan. En estos grupos podrán acceder otros profesores que sean invitados, como los responsables legales de los alumnos, dependiendo de la configuración dada por el educador.

Edmodo proporciona la posibilidad de compartir recursos, así como archivos y mensajes entre usuarios. Se pueden proponer tareas y actividades, realizar pruebas, corregirlas y calificarlas. Además, el docente puede evaluar el progreso de un alumno mediante diversos aspectos, entre ellos, tener en cuenta el resultado de las tareas asignadas, como también los comentarios, aportaciones y críticas que hayan realizado en el muro de la asignatura, valorando la capacidad crítica, la creatividad y el trabajo en equipo. Estos aspectos son importantes hoy en día pues es tomado en cuenta por muchas empresas en el momento de la contratación.

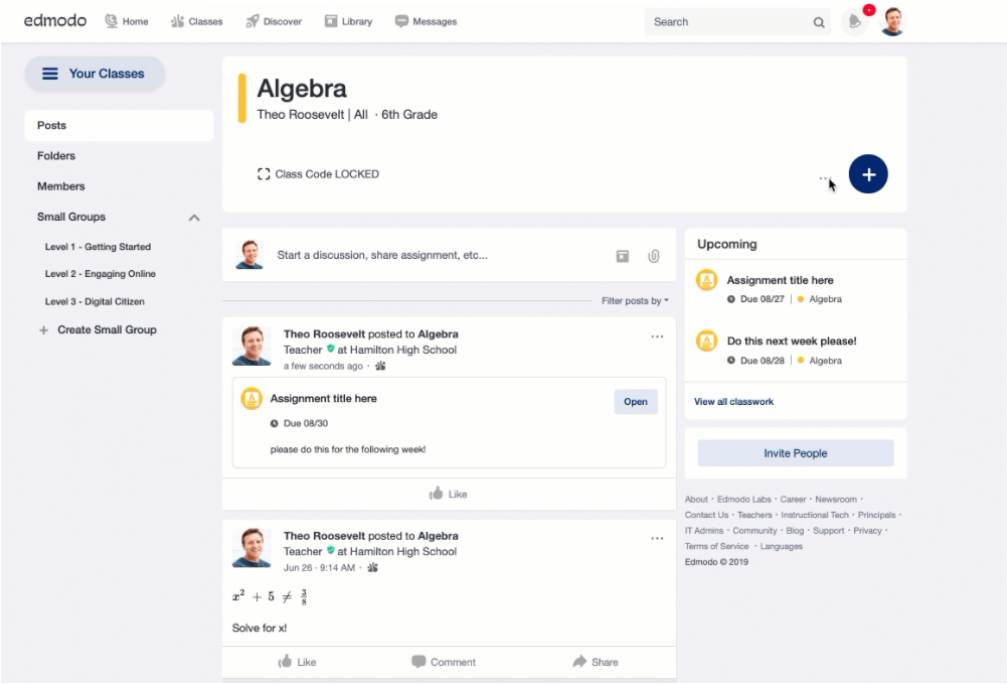


Figura 2.6: Pantalla principal de una asignatura en Edmodo

2.3 Análisis de la situación actual

En la tabla que podemos ver a continuación se muestran las funcionales generales que tienen las aplicaciones evaluadas en la sección anterior. Estas funciones han sido recolectadas haciendo pruebas con cada una de las aplicaciones.

Funcionalidad	Moodle	Sakai	Edmodo
Crear, editar y borrar cursos/asignaturas	✓	✓	✓
Crear, editar y borrar categorías	✓	✓	✓
Crear, editar y borrar usuarios	✓	✓	✓
Definir roles para los usuarios	✓	✓	
Definir diferentes estilos para la aplicación(letra, tamaño, etc...)	✓	✓	
Repositorio de ficheros personales	✓	✓	
Calendario para visualizar eventos	✓	✓	✓
Visualizar calificaciones de los alumnos de una asignatura	✓	✓	✓
Corregir tareas y exámenes de los alumnos	✓	✓	✓
Crear, editar y borrar exámenes y tareas	✓	✓	✓
Editar características de perfil	✓	✓	✓
Ver perfiles de otros usuarios	✓	✓	✓
Añadir y borrar anuncios (en el tablón de anuncios)	✓	✓	
Chat entre usuarios	✓	✓	✓
Tutorial de uso al iniciar la aplicación		✓	✓
Comentar en un foro	✓	✓	✓
Estadísticas de un/a curso/asignatura	✓	✓	✓
Ver otros cursos creados por otros profesores			✓
Crear grupos		✓	✓
Añadir a otros profesores al curso siendo profesor			✓
Cambiar de idioma	✓	✓	✓
Crear grupos de usuario		✓	✓
Buscar comentarios en el foro vía *hastag* de otros usuarios			✓
Crear *posts* para interactuar con otros usuarios	✓	✓	✓
Unirse a comunidades o grupos para visualizar su contenido			✓
Importar y exportar contenido del curso	✓		

Tabla 2.1: Comparativa entre funcionalidades recogidas de diferentes aplicaciones del mercado actual.

Tras analizar la comparativa anterior, se concluye que nuestra aplicación debería considerar las funcionalidades que se encuentran en los tres programas anteriores, además de alguna funcionalidad que pueda diferenciar nuestra aplicación de las demás:

- Crear, editar y borrar asignaturas.
- Crear, editar y borrar categorías.
- Crear, editar y borrar usuarios.
- Crear, editar y borrar exámenes y tareas.
- Calendario para visualizar eventos.

- Visualizar calificaciones de los alumnos de una asignatura
- Corregir tareas y exámenes de los alumnos.
- Editar características de perfil.
- Chat entre usuarios.
- Comentar en un foro dudas o pensamientos.
- Estadísticas de un/a curso/asignatura.
- Cambiar de idioma.
- Crear posts en el foro para interactuar con otros usuarios.

En la sección siguiente, se realizará un análisis de prioridades mediante el método MoSCoW para definir cuáles de estas funcionales se realizarán finalmente en la aplicación según el orden de prioridad de esta.

2.3.1. Método MoSCoW

El método MoSCoW [3] es una técnica inventada por Dai Clegg que sirve para determinar la priorización de proyectos con limitaciones de tiempo. Esta técnica asigna más valor y prioridad a las características de un proyecto que más valor comercial tienen, por lo que son las que más posibilidad tienen de ser implementadas, mientras que el desarrollo será pospuesto o desechado en las que menos valor tienen.

El modelo MoSCoW se define en cuatro grupos de grado de prioridad:

- ***Must Have* (Debe tener).** Se tratan de requisitos totalmente imprescindibles que tienen que estar incluidos ya que si no se llevan a cabo el proyecto no puede salir adelante.
- ***Should Have* (Debería tener).** Requisitos de alta prioridad que en la medida de lo posible deberían ser incluidos en la solución final, pero que llegado el momento y si fuera necesario, podrían ser prescindibles si hubiera alguna causa que lo justificara.
- ***Could Have* (Podría tener).** Requisito deseable pero no necesario, se implementaría si hubiera posibilidades presupuestarias y temporales.
- ***Won't Have* (No se van a implementar).** Hace referencia a requisitos que están descartados de momento pero que en un futuro podrían ser tenidos en cuenta y ser reclasificados en una de las categorías anteriores.

[3] https://proagilist.es/blog/agilidad-y-gestion-agil/priorizar-requisitos-tecnica-priorizacion-moscow/

Must Have	Should Have	Could Have	Won't Have
▪ Crear, editar y borrar asignaturas ▪ Crear, editar y borrar categorías ▪ Crear, editar y borrar usuarios ▪ Crear, editar y borrar tareas ▪ Visualizar calificaciones de los alumnos de una asignatura ▪ Corregir tareas realizadas por los alumnos. ▪ Repositorio de ficheros personales	▪ Editar características de perfil ▪ Crear, editar y borrar exámenes ▪ Corregir exámenes realizados por los alumnos ▪ Estadísticas de un/a curso/asignatura	▪ Calendario para visualizar eventos ▪ Chat entre usuarios ▪ Crear posts en un foro para interactuar con otros usuarios ▪ Comentar dudas o pensamientos en un foro ▪ Cambiar de idioma	▪ Versión de pago ▪ Importar y exportar contenido del curso

Tabla 2.2: Análisis de priorización de requisitos utilizando el método MoSCoW.

Cómo se puede observar en el análisis del método MoSCoW, se han priorizado las tareas relacionadas con la creación, edición y borrado de elementos, así como la corrección de tareas del alumnado por parte del profesor debido a que es una característica importante de nuestro proyecto. También se ha priorizado el guardado de ficheros personales en nuestra aplicación pues, al tener que entregar tareas o crear documentación para las respectivas asignaturas, gran parte del alumnado y profesorado deseará poder guardar ficheros u otro tipo de recursos en la nube.

Por otro lado, otras características menos urgentes, como editar las características de los perfiles de los usuarios, pero que siguen siendo necesarias, se intentarán implementar en la solución final en la medida de lo posible.

Por último, el resto de funcionalidades del apartado *Could Have*, se dejarán para futuras versiones de la aplicación fuera del marco de este documento. Además, los requisitos que se han rechazado, se volverán a analizar en un futuro para adaptarse a las nuevas demandas de los usuarios.

2.3.2. Análisis DAFO

Para poder analizar el riesgo de nuestra aplicación en el mercado actual, utilizaremos el análisis DAFO.[4] Es una técnica que, mediante un análisis del entorno externo y las características internas de nuestro proyecto, permite obtener una representación gráfica de los siguientes puntos:

[4] https://www.infoautonomos.com/plan-de-negocio/analisis-dafo/

- **Debilidades.** Son aquellas propiedades de las que carece la empresa (en nuestro caso este proyecto), que son inferiores al resto de empresas o proyectos del mercado o en los que se debe mejorar.

- **Amenazas.** Se definen como los factores externos que pueden poner en peligro la supervivencia de la empresa. Se deben identificar con suficiente antelación para poder evitarlas o, en el caso en que sea posible, convertirlas en posibilidades.

- **Oportunidades.** Son factores externos al proyecto que favorecen su desarrollo o permite la posibilidad de implantar mejoras que impulsen al proyecto.

- **Fortalezas.** Son aquellos factores y recursos internos a explotar que representan una ventaja competitiva y oportunidades de sobresalir en el mercado actual.

Debilidades	Amenazas
1. Existen muchos productos compitiendo en el mercado, algunos de los cuales utilizados por universidades e institutos públicos. 2. Baja capacidad de financiación. 3. Necesidad de conocer cómo funciona la organización de diversos centros educativos.	1. Posible entrada de nuevos competidores debido a la situación actual. 2. Constante evolución y cambio de la pedagogía. 3. Barreras administrativas en el sector ya que algunas aplicaciones son desarrolladas por universidades importantes.
Fortalezas	**Oportunidades**
1. Bajo coste de desarrollo del proyecto. 2. Facilidad de testing de la aplicación. 3. Posibilidad de utilizarlo en cualquier dispositivo que tenga acceso a Internet. 4. La aplicación brinda buena experiencia de usuario.	1. Gran aumento de la digitalización en el sector de la pedagogía y enseñanza. 2. Creciente necesidad de utilizar un sistema que ayude a realizar clases online. 3. Existe una gran cantidad de usuarios potenciales que utilicen la aplicación.

Figura 2.7: Análisis DAFO de la futura aplicación.

Como se observa en el análisis DAFO realizado, nuestra futura aplicación tiene más fortalezas que amenazas, mientras que tiene el mismo número de oportunidades que debilidades. Si bien en un futuro cercano esto podría cambiar pues nuestro proyecto puede reportar beneficios rápidamente debido a la alta demanda de clases online en la actualidad.

Igualmente se deben prestar atención a las debilidades y amenazas, pues existen muchas aplicaciones en el mercado que pueden eclipsar a la nuestra, haciendo que nuestro proyecto no despegue y no genere rentabilidad. Por último, se deberá reforzar la experiencia de usuario para atraer público nuevo que busque una herramienta para proporcionar clases online y no se haya decidido por ninguna hasta el momento.

2.4 Conclusiones y propuesta

A pesar de que existen un gran número de aplicaciones en el mercado y que todas proporcionan funcionalidades muy interesantes como hemos visto en las secciones anteriores, se pretende unificar todas las características más utilizadas por todas ellas de una forma consistente y original que atraiga al usuario a probar nuestra aplicación. Para ello, se hará uso de una base de datos relacional que almacene toda la información y un backend que tendrá toda la lógica de la aplicación.

Por otra parte, en nuestro análisis DAFO hemos observado que existe un gran número de potenciales usuarios a los que va dirigida nuestra aplicación, en concreto a la multitud de centros educativos que existen en España. Debemos tener en cuenta la constante evolución que sufre la pedagogía, ya sea debido a la necesidad de los profesores de renovar la forma de enseñar su temario o los cambios constantes que sufren los temarios de los cursos, por lo que nuestra aplicación deberá estar en constante desarrollo.

Finalmente, el objetivo principal de este proyecto es ofrecer una plataforma para que, de una forma sencilla, los diferentes centros educativos puedan gestionar el aprendizaje y educación de su alumnado a través de los distintos recursos y materiales que proporcione el profesorado, por lo que se priorizará los requisitos seleccionados en el análisis realizado con el método MoSCoW.

CAPÍTULO 3
Análisis del problema

En este capítulo se realizará un análisis detallado de la idea propuesta, de forma que plantearemos los casos de uso que debería tener según los requisitos vistos anteriormente. Por otro lado, se tendrá en cuenta la persistencia de la aplicación, analizando las distintas tecnologías para ello y decidiendo cuál se adapta más a nuestras necesidades, ya que tendrá que guardar distintos tipos de objetos lógicos, como usuarios, asignaturas, tareas, etc. Por último, para manejar la lógica de la aplicación, se estudiarán diferentes *frameworks* para determinar cuál se ajusta correctamente a nuestros casos de uso.

3.1 Especificación

Para el planteamiento inicial tendremos que estudiar los diferentes actores que harán uso de la aplicación, así como definir los requisitos que tendrá que tener para cumplir el objetivo principal visto anteriormente.

3.1.1. Propósito inicial

Como hemos dicho anteriormente, el objetivo principal de nuestro proyecto es facilitar la distribución de recursos educativos por parte de los profesores hacia el alumnado, de forma que organizar la forma de dar la materia o asignatura sea una tarea más sencilla.

Nuestra aplicación web ofrecerá a los centros educativos la opción de añadir los profesores y las asignaturas de los cursos, de forma que sea sencillo gestionarlos y editarlos cuando sea necesario. Por otra parte, el profesorado podrá ver un listado de las asignaturas que imparte en el curso académico actual. También podrá editar las características de las asignaturas, como también añadir unidades y tareas a ella.

El alumnado tendrá la opción de visualizar las asignaturas en las que está matriculado, además de su contenido, que les permitirá ampliar conocimientos sobre la materia. La aplicación también les ofrecerá la posibilidad de rellenar tareas, escribiendo un comentario y añadiendo ficheros, que posteriormente corregirá el profesor de la asignatura.

Por último, tanto los profesores como los alumnos dispondrán de un repositorio propio de ficheros, que podrán utilizar para subir los ficheros que deseen en cualquier momento.

Así pues los actores y requisitos serán los siguientes:

3.1.2. Actores

Existen tres tipos diferentes de actores en nuestra aplicación:

- **Administrador.** Aquellas personas que se encargan del registro de los distintos usuarios de la aplicación, así como de las asignaturas que son impartidas por los profesores, además del mantenimiento de estas.
- **Profesor.** Son los usuarios que configuran y personalizan las asignaturas. Son los responsables de subir temario y corregir tareas.
- **Alumno.** Aquellos usuarios que pueden visualizar los recursos de las asignaturas a las que han sido matriculados, así como contestar una tarea añadida por el profesor de la materia.

3.1.3. Requisitos

A continuación se definirán los requisitos funcionales y no funcionales de nuestro proyecto. Mientras los requisitos funcionales describen lo que debe de hacer un sistema, es decir, su funcionamiento, los requisitos no funcionales imponen restricciones sobre cómo deberá realizar algunas funcionalidades el sistema.

3.1.3.1. Requisitos funcionales

1. Los usuarios deberán ser capaces de iniciar sesión en la aplicación con un nombre de usuario y contraseña personales.
2. Tanto profesores como alumnos serán capaces de editar su perfil, pudiendo modificar datos personales como el correo electrónico o cambiar su contraseña, o añadir una foto de perfil.
3. Los administradores podrán dar de alta profesores, alumnos, asignaturas y categorías de las materias, así como visualizarlas, editarlas y borrarlas.
4. Tanto profesores como alumnos deberán poder subir y borrar ficheros en un repositorio propio.
5. Los profesores y alumnos deberán ser capaces de ver un listado de asignaturas, así como filtrarlas por nombre y ver su contenido.
6. Los profesores deberán poder cambiar la configuración de las asignaturas (descripción, métodos de evaluación, etc.).
7. El profesorado podrá ver un listado de los alumnos que están matriculados de una asignatura.
8. La aplicación debe ofrecer la posibilidad de guardar información sobre las tareas que cree un profesor, así como la información rellenada por los alumnos para poder completarla.
9. El sistema debe permitir corregir las tareas anteriormente mencionadas, pudiendo asignarles una calificación a cada una.
10. Los alumnos deben ser capaces de ver las puntuaciones y valoraciones de una tarea dadas por el profesor.
11. Los usuarios deberán poder recuperar su contraseña en caso de pérdida.

3.1.3.2. Requisitos no funcionales

1. El guardado de datos debe ser transparente para cualquiera de los usuarios de la aplicación.

2. El sistema debe visualizarse correctamente en cualquier navegador y en cualquier tipo de pantalla, ya sea de móvil o de ordenador.

3.2 Modelado conceptual: Casos de uso

En esta sección se mostrarán los casos de uso más relevantes de nuestra aplicación, relacionándolos con los requisitos vistos anteriormente:

3.2.1. Iniciar sesión

Caso de uso 1	Iniciar sesión en la aplicación
Objetivo	**Requisito funcional 1.** Iniciar sesión
Descripción	El usuario inicia sesión introduciendo su nombre de usuario y contraseña
Precondición	El usuario debe estar registrado en la aplicación.
Secuencia	▪ Introducir nombre de usuario y contraseña. ▪ Pulsar el botón de inicio de sesión.
Postcondición	El usuario ha iniciado sesión y: 1. Si es administrador: aparecerá en la página principal de administración. 2. Si es profesor o alumno: aparecerá en la página principal de la aplicación.

Tabla 3.1: Caso de uso 1: Inicio de sesión.

3.2.2. Ver perfil

Caso de uso 2	Ver perfil.
Objetivo	**Requisito funcional 2**. Ver perfil en la aplicación.
Descripción	Los profesores y alumnos podrán visualizar los datos relacionados con su perfil.
Precondición	El usuario debe estar registrado en la aplicación y ser propietario del perfil.
Secuencia	▪ Pulsar el botón de ver perfil.
Postcondición	—-

Tabla 3.2: Caso de uso 2: Ver perfil

3.2.3. Editar perfil

Caso de uso 3	Editar perfil.
Objetivo	**Requisito funcional 2**. Editar perfil en la aplicación.
Descripción	Los profesores y alumnos podrán editar algunos datos almacenados en su perfil.
Precondición	El usuario debe estar registrado en la aplicación y ser propietario del perfil.
Secuencia	▪ Pulsar el botón de editar perfil. ▪ Modificar los datos correspondientes. ▪ Presionar el botón guardar.
Postcondición	El perfil aparecerá con las información correspondiente modificada.

Tabla 3.3: Caso de uso 3: Editar perfil

*__Nota__: En los casos de uso presentados a continuación se ha englobado en *objetos* los siguientes objetos lógicos: Alumnos, Profesores, Categorías, Asignaturas.

3.2.4. Listar objetos*

Caso de uso 4	Visualizar listado de objetos registrados.
Objetivo	**Requisito funcional 3.** Visualizar el listado de objetos registrados en la aplicación.
Descripción	Los administradores podrán visualizar el listado de todos los objetos que estén registrados en la aplicación.
Precondición	El usuario debe estar registrado en la aplicación y ser administrador.
Secuencia	▪ Pulsar en la sección correspondiente.
Postcondición	—-

Tabla 3.4: Caso de uso 4: Listar objetos

3.2.5. Añadir objetos*

Caso de uso 5	Añadir objetos.
Objetivo	**Requisito funcional 3.** Añadir un objeto en la aplicación.
Descripción	Los administradores podrán agregar objetos en la aplicación.
Precondición	El usuario debe estar registrado en la aplicación y ser administrador.
Secuencia	▪ Pulsar en la sección correspondiente para visualizar la lista de objetos. ▪ Pulsar el botón de añadir objeto. ▪ Rellenar la información pertinente. ▪ Pulsar el botón de guardar.
Postcondición	Se ha agregado un nuevo objeto en la aplicación y se ha añadido un registro en la base de datos.

Tabla 3.5: Caso de uso 5: Añadir objetos

3.2.6. Editar objetos*

Caso de uso 6	Editar objetos.
Objetivo	**Requisito funcional 3.** Editar un objeto en la aplicación.
Descripción	Los administradores podrán editar objetos en la aplicación.
Precondición	Requisito funcional 3. El usuario debe estar registrado en la aplicación y ser administrador.
Secuencia	▪ Pulsar en la sección correspondiente para visualizar la lista de objetos. ▪ Pulsar el botón de editar en objeto que se desee modificar. ▪ Rellenar la información pertinente. ▪ Pulsar el botón de guardar.
Postcondición	Se ha editado un objeto existente en la aplicación y se ha modificado un registro en la base de datos.

Tabla 3.6: Caso de uso 6: Editar objetos

3.2.7. Borrar objetos*

Caso de uso 7	Borrar objetos.
Objetivo	**Requisito funcional 3.** Borrar un objeto en la aplicación.
Descripción	Los administradores podrán borrar objetos en la aplicación.
Precondición	El usuario debe estar registrado en la aplicación y ser administrador.
Secuencia	▪ Pulsar en la sección correspondiente para visualizar la lista de objetos. ▪ Pulsar el botón de borrar objeto que se desee eliminar. ▪ Aceptar la confirmación de borrado.
Postcondición	Se ha borrado un objeto existente en la aplicación y se ha eliminado un registro en la base de datos.

Tabla 3.7: Caso de uso 7: Editar objetos

3.2.8. Visualizar asignaturas asignadas

Caso de uso 8	Visualizar asignaturas asignadas.
Objetivo	**Requisito funcional 5.** Visualizar las asignaturas que tiene asignadas un profesor o un alumno.
Descripción	Los profesores y alumnos podrán visualizar las asignaturas de las que son partícipes.
Precondición	El usuario debe estar registrado en la aplicación, y ser profesor o alumno en una asignatura como mínimo.
Secuencia	▪ Iniciar sesión en la aplicación.
Postcondición	—-

Tabla 3.8: Caso de uso 8: Visualizar asignaturas asignadas

3.2.9. Buscar asignaturas asignadas

Caso de uso 9	**Requisito funcional 5.** Buscar asignaturas asignadas.
Objetivo	Buscar las asignaturas que tiene asignadas un profesor o un alumno.
Descripción	Los profesores y alumnos podrán realizar una búsqueda por nombre de las asignaturas de las que son partícipes.
Precondición	El usuario debe estar registrado en la aplicación, y ser profesor o alumno en una asignatura como mínimo.
Secuencia	▪ Iniciar sesión en la aplicación. ▪ Escribir parte del nombre o el nombre completo de la asignatura a buscar.
Postcondición	Visualización de las asignaturas obtenidas de la búsqueda por nombre.

Tabla 3.9: Caso de uso 9: Buscar asignaturas asignadas

3.2.10. Subir ficheros a un repositorio personal

Caso de uso 10	Subir ficheros a un repositorio personal.
Objetivo	**Requisito funcional 4.** Subir ficheros a un repositorio personal en la aplicación.
Descripción	Los profesores y alumnos podrán subir ficheros en un espacio personal propio.
Precondición	El usuario debe estar registrado en la aplicación como profesor o alumno.
Secuencia	▪ Iniciar sesión en la aplicación. ▪ Acceder a la sección “Repositorio personal” ▪ Arrastrar un fichero al apartado subida de ficheros o seleccionar uno desde el dispositivo.
Postcondición	Se ha añadido un fichero en la aplicación y un registro en la base de datos.

Tabla 3.10: Caso de uso 10: Subir ficheros a un repositorio personal

3.2.11. Visualizar contenido de una asignatura

Caso de uso 11	Visualizar contenido de una asignatura.
Objetivo	**Requisito funcional 5.** Visualizar contenido de una asignatura.
Descripción	Visualizar contenido de la asignatura seleccionada en la aplicación.
Precondición	El usuario debe estar registrado como profesor o alumno partícipe de la asignatura seleccionada.
Secuencia	▪ Iniciar sesión en la aplicación. ▪ Acceder a la asignatura deseada. ▪ Se mostrará el contenido (unidades y temario) de la asignatura.
Postcondición	—-

Tabla 3.11: Caso de uso 11: Visualizar contenido de una asignatura.

3.2.12. Editar configuración de la asignatura

Caso de uso 12	Editar configuración de la asignatura.
Objetivo	**Requisito funcional 6.** Editar configuración de una asignatura.
Descripción	Editar configuración de la asignatura elegida de la aplicación.
Precondición	El usuario debe estar registrado como profesor y ser partícipe en la asignatura
Secuencia	▪ Iniciar sesión en la aplicación. ▪ Acceder a la asignatura deseada. ▪ Acceder a la configuración de la asignatura. ▪ Rellenar los campos deseados. ▪ Darle al botón guardar.
Postcondición	Se han guardado los cambios introducidos del contenido de la asignatura.

Tabla 3.12: Caso de uso 12: Editar configuración de la asignatura.

3.2.13. Listar alumnado de una asignatura

Caso de uso 13	Listar alumnado de una asignatura
Objetivo	**Requisito funcional 7.** Listar alumnado de una asignatura.
Descripción	El profesor puede visualizar un listado de los alumnos que son partícipes en la asignatura.
Precondición	El usuario debe estar registrado como profesor y ser partícipe en la asignatura.
Secuencia	▪ Iniciar sesión en la aplicación. ▪ Acceder a la asignatura deseada. ▪ Acceder al apartado de alumnos de la asignatura.
Postcondición	—-

Tabla 3.13: Caso de uso 13: Listar alumnado de una asignatura.

3.2.14. Corregir tarea

Caso de uso 14	Corregir tarea.
Objetivo	**Requisito funcional 9.** Corregir tarea de una asignatura.
Descripción	El profesor puede corregir la tarea realizada por un alumno en una asignatura.
Precondición	▪ El usuario debe estar registrado como profesor de la asignatura. ▪ La asignatura debe existir en la aplicación. La tarea debe existir en la aplicación y debe haberla entregado el alumno seleccionado en algún momento.
Secuencia	▪ Iniciar sesión en la aplicación. ▪ Acceder a la asignatura deseada. ▪ Seleccionar la tarea que se quiere corregir. ▪ Seleccionar el alumno a evaluar. ▪ Escribir puntuación y si se desea algún comentario. ▪ Pulsar el botón guardar.
Postcondición	Se ha guardado la corrección de la tarea seleccionada en la base de datos.

Tabla 3.14: Caso de uso 14: Corregir tarea.

3.2.15. Entregar tarea

Caso de uso 15	Entregar tarea.
Objetivo	**Requisito funcional 8.** Entregar tarea de una asignatura.
Descripción	El alumno puede rellenar una tarea con información y ficheros para poder entregarla y que el profesor correspondiente la evalúe posteriormente.
Precondición	El usuario debe estar registrado como alumno y ser partícipe en la asignatura.
Secuencia	▪ Iniciar sesión en la aplicación. ▪ Acceder a la asignatura deseada. ▪ Seleccionar la tarea a entregar. ▪ Rellenar la información y subir los ficheros correspondientes. ▪ Pulsar el botón guardar.
Postcondición	Se ha actualizado la información de la tarea entregada en el sistema. También aparecerá el alumno a corregir en la tarea para el profesor.

Tabla 3.15: Caso de uso 15: Entregar tarea.

3.2.16. Visualizar calificaciones

Caso de uso 15	Visualizar calificaciones.
Objetivo	**Requisito funcional 10.** Visualizar calificaciones de las tareas de una asignatura.
Descripción	El alumno puede visualizar las calificaciones de las tareas, sean entregadas o no.
Precondición	El usuario debe estar registrado como alumno y ser partícipe en la asignatura.
Secuencia	▪ Iniciar sesión en la aplicación. ▪ Acceder a la asignatura deseada. ▪ Seleccionar el apartado "Notas". ▪ Visualizar las tablas con las calificaciones de cada tarea.
Postcondición	—-

Tabla 3.16: Caso de uso 16: Visualizar calificaciones.

3.2.17. Recuperar contraseña

Caso de uso 15	Recuperar contraseña.
Objetivo	**Requisito funcional 11.** Recuperar contraseña de un usuario.
Descripción	Si un usuario pierde la contraseña de su perfil registrado, puede pedir cambiarla mediante la aplicación.
Precondición	El usuario debe estar registrado en la aplicación.
Secuencia	▪ En el inicio de sesión, seleccionar la opción: "¿Has olvidado la contraseña?". ▪ Escribir el correo electrónico y darle a Recuperar. ▪ En el correo electrónico recibido, pulsar el enlace correspondiente. ▪ Rellenar los nuevos datos de la contraseña. ▪ Pulsar el botón Guardar.
Postcondición	Se ha cambiado la contraseña del usuario correspondiente.

Tabla 3.17: Caso de uso 17: Recuperar contraseña.

3.3 Análisis de soluciones posibles

A continuación analizaremos las posibles soluciones que disponemos para realizar nuestro proyecto. Se deberá tener en cuenta las tecnologías existentes de bases de datos para la persistencia de datos, como también los *frameworks de backend* que podemos encontrar en el mercado para que recoja la lógica de la futura aplicación.

3.3.1. Diseño interno

Actualmente existen diferentes herramientas que pueden adecuarse a nuestras necesidades. Podemos encontrar multitud de *frameworks de backend* en el mercado, pero se

van a tener en consideración los más utilizados, pues estos tienen una comunidad grande resolviendo problemas y actualizándolos constantemente, por lo que el desarrollo con alguno de ellos será más sencillo. Seguidamente procedemos a listar los *frameworks de backend* más utilizados en 2021:

- **Laravel.** Es un *framework* basado en PHP que utiliza el patrón de arquitectura MVC (Modelo-Vista-Controlador). Distribuido bajo licencia MIT, mantiene todo su código disponible en Github. Entre sus funcionalidades destacan la integración directa con servicios de e-mail, controladores para la integración de SMTP, Amazon u otros servicios de mensajería, por lo que la generación de emails es muy rápida. Además también tiene un potente motor de plantillas, protección ante vulnerabilidades y ataques, y es un excelente programador de tareas y crones.

- **Django.** Está basado en Python y utiliza el patrón de arquitectura MTV (Modelo-Plantilla-Vista). Algunas webs destacadas que utilizan Django son National Geografic, Instagram o Pinterest. Una de sus características más interesantes es su rapidez para el desarrollo, ya que permite desarrollar una aplicación en poco tiempo ahorrando costes. También dispone de una alta seguridad al igual que Laravel, para que no haya fallos de seguridad como puede ser una *SQL Injection* o robo de información mediante ataques CSRF. Por último, es muy escalable y versátil.

- **Flask.** Es un microweb *framework* escrito en Python, al igual que Django. No tiene una curva de aprendizaje tan elevada como otras tecnologías y permite desarrollar aplicaciones en las que no se necesiten muchas extensiones ni sean muy complejas. Además, soporta de manera nativa el uso de cookies seguras, y a pesar de que no tiene ORMs inicialmente como otros *frameworks*, se puede usar una extensión para ello.

- **ExpressJS.** Está diseñado como *framework de backend* para Node.js y es distribuido bajo licencia MIT, por lo que es *open-source* (código abierto). Está planeado para desarrollar aplicaciones web o APIs. Al estar basado en JavaScript, su desarrollo se facilita ya que para construir una aplicación solo necesitamos utilizarlo junto a HTML. Otra ventaja es su facilidad de trabajar con otros *frameworks* y librerías para añadir más funcionalidades a la aplicación. Por último, destaca en su simplicidad para conectarse con gran cantidad de tipos de bases de datos, como MongoDB, Redis o MySQL.

- **Ruby on Rails.** Está basado en Ruby y utiliza el patrón de arquitectura MVC, al igual que Laravel. Una de sus grandes ventajas es la existencia de *gems* (gemas), los cuales son paquetes de código realizados por otras personas que aportan funcionalidades muy comunes en aplicaciones web que es posible de utilizar en nuestro proyecto. Por otro lado, la velocidad de desarrollo con esta herramienta es muy rápida, y permite desplegar de forma separada contenido estático y dinámico. Además dispone de una librería Ajax interna que permite cargar páginas rápidamente, como también un sistema de testing integrado.

3.3.2. Persistencia de los datos

Existen diferentes formas para mantener la persistencia de la información tales como ficheros de texto plano, bases de datos relacionales o no relacionales, etc. Cada una de ellas tiene diferentes funciones y usos según el tipo de datos que se quieran almacenar en ellas. Dado que la información que queremos guardar debe ser privada y mantener

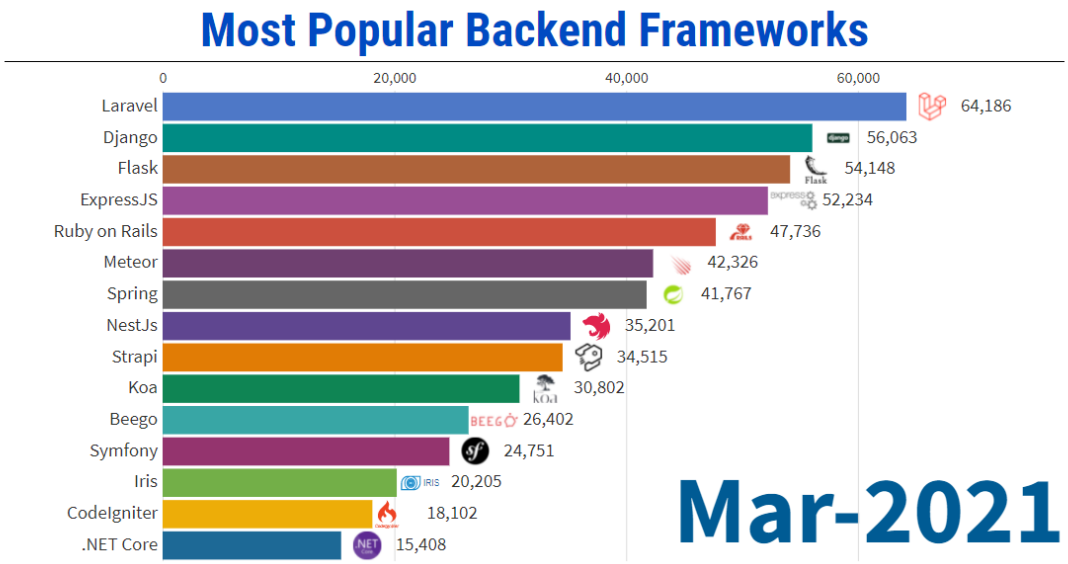


Figura 3.1: Gráfica de frameworks de backend más utilizados en 2021. Fuente: Statistics&Data https://cutt.ly/AnMORxE

unos estándares de seguridad mínimos, deberemos elegir un gestor de bases de datos, ya que nos permiten perservar la información de forma segura. Por otro lado, como en el grado hemos visto las bases de datos relacionales, nos decantaremos por estas, pues nos resultará más sencillo evaluarlas y analizarlas.

A continuación, se expondrán diferentes sistemas de gestión de bases de datos relacionales que existen en el mercado:

- **Oracle.** Oracle es el gestor de bases de datos más utilizado en todo el mundo actualmente. Una de sus ventajas es su facilidad para agregar o reasignar recursos según sea necesario, además de poder dividir el procesamiento entre el servidor de la base de datos y el programa del cliente. También tiene una gran capacidad de evitar accesos no autorizados en un entorno multiusuario gracias a su sistema de base de datos de contenedores multiusuario (CDB). Por último, tiene una versión gratuita para desarrolladores novatos que no recibe parches y otra versión de pago con soporte para entornos de producción.

- **MySQL.** Es una base de datos gratuita y de código abierto, por lo que su uso no tiene ningún tipo de coste. Es de los más utilizados por sistemas de gestión de contenidos como WordPress o Drupal. Entre sus ventajas destaca su velocidad. Es un gestor muy rápido sin tener que añadirle ninguna funcionalidad avanzada. Tiene una gran flexibilidad y utiliza varias capas de seguridad para encriptar contraseñas, derechos de acceso y privilegios para los usuarios.

- **PostgreSQL.** Al igual que MySQL, PostgreSQL es un gestor de base de datos gratuito y de código abierto. Cuenta con diversos foros oficiales donde los usuarios pueden exponer sus dudas, como también un soporte empresarial. Una de sus ventajas es la gran variedad de extensiones distribuidas por el grupo de desarrolladores de la herramienta. Además cualquier usuario puede crear sus propias extensiones en distintos lenguajes de programación como Java, Python o C++. Por último, PostgreSQL cumple con la característica *ACID*, acrónimo de "Atomicidad, Consistencia, Aislamiento y Durabilidad", lo que permite que las transacciones no interfieran unas con otras.

- **MariaDB.** Es un sistema de gestión de base de datos nacido a partir de MySQL. Tiene una gran compatibilidad con este último pues posee las mismas órdenes, interfaces, API y bibliotecas, pues tiene el objetivo de poder intercambiar un servidor por otro directamente. Existen bastantes paquetes y extensiones que, aunque son diseñados para MySQL, también se encuentran disponibles para MariaDB.

3.4 Solución propuesta

En el apartado anterior hemos visto diferentes tecnologías que podemos utilizar para desarrollar nuestro proyecto, tanto para la parte lógica como para la persistencia de los datos. Para la parte lógica concretamente hemos visto cinco *frameworks de backend*, cada uno con sus características propias y funcionalidades que se adaptan mejor o peor a los requisitos de nuestro proyecto. Para empezar, nuestro sistema debe ser escalable en un futuro, por lo que necesitamos que tenga una alta flexibilidad en este sentido. Flask no cumple este requisito, ya que a pesar de ser muy potente debido a las extensiones que le podemos añadir, está enfocado a aplicaciones pequeñas y poco complejas. Por otra parte, sería útil que nos proporcionase un fácil aprendizaje y un rápido desarrollo para nuestro proyecto, por lo que, a pesar de que todos tienen características interesantes como alta seguridad y sencillez para instalar multitud de extensiones, nos decantamos por Django, pues está enfocado a realizar aplicaciones de tamaño pequeño-medio rápidamente, y además destaca por ser fácilmente escalable.

En cuanto a las tecnologías para la persistencia de datos, hemos visto cuatro tipos de gestores de bases de datos. Para nuestro proyecto, la base de datos de datos a utilizar debería ser gratuita para ahorrar costes, por lo que a pesar de que Oracle tiene una versión gratuita, esta no dispone de actualización ni parches de seguridad, por lo que no sería nada recomendable para un entorno de producción. Por otro lado, MySQL a pesar de ser open-source, tiene licencia dual con Oracle desde su compra, por lo que Oracle podría cambiar los términos de uso en cualquier momento afectándonos directamente. Además, nuestro proyecto requiere un fuerte control sobre la concurrencia, pues múltiples usuarios accederán al mismo tiempo a la aplicación. A pesar de que MariaDB tiene un buen control de la concurrencia, PostgreSQL destaca por su gran control sobre la concurrencia a nivel de transacción, por lo que es la mejor opción para nosotros. Además, PostgreSQL maneja el estándar SQL de forma más estricta que MariaDB, por lo que nos asegura mejor compatibilidad y portabilidad hacia otros sistemas de gestión de bases de datos que este último.

Adicionalmente, se ha decidido instalar un subsistema WSL[1], que nos permite ejecutar un entorno de Ubuntu con la versión 20.04 en Windows, para facilitar la instalación del gestor de base de datos y de las diferentes extensiones que se utilizarán en Django.

Por último, vamos a plantear el almacenamiento de datos que va a manejar nuestra aplicación. Para ello, utilizaremos el modelado conceptual mediante UML de la base de datos, y en las siguientes secciones, mostraremos el diagrama de clases y el diagrama de bases de datos relacional.

Como definición, UML [2] es un lenguaje gráfico para representar partes de un sistema de software, como pueden ser el diseño, comportamiento o la arquitectura. Nos permite representar las ideas de cómo estructurar los programas de forma visual y estándar para que cualquier persona que conozca UML comprenda nuestras intenciones. A continua-

[1] https://docs.microsoft.com/es-es/windows/wsl/about
[2] https://openwebinars.net/blog/que-es-uml-unified-modeling-language/

ción, se mostrará el modelo entidad-relación con la finalidad de representar las entidades de la base de datos y las posibles relaciones entre estas:

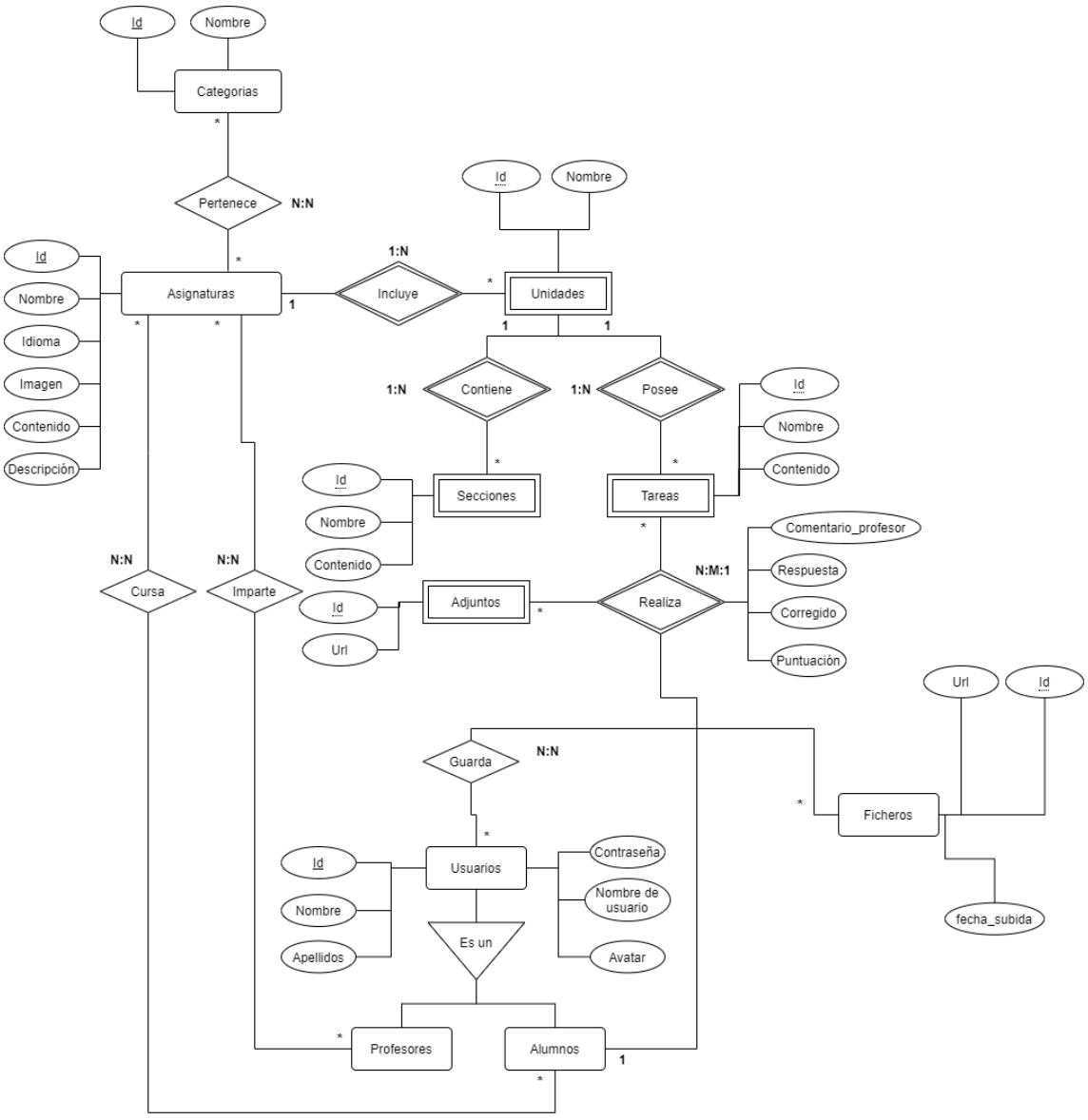


Figura 3.2: Diagrama Entidad-Relación de la base de datos.

Como se puede observar en la figura 3.18, la base de datos está compuesta por nueve entidades:

- **Categorías.** Contiene los tipos de categorías en las que se pueden agrupar las asignaturas.
- **Asignaturas.** Representa las asignaturas que imparte un centro educativo. Se relaciona con alumnos y profesores que son partícipes de la materia, y con unidades, que serán las secciones en las que se divide una asignatura.
- **Unidades.** Simboliza las diferentes subsecciones dentro de una asignatura. Contienen secciones y tareas las cuales puede añadir un profesor.
- **Secciones.** Apartado explicativo dentro de una unidad. Pueden haber tantos como se requiera.

- **Tareas.** Representa las tareas que debe rellenar un alumno. Un alumno puede escribir una respuesta y añadir ficheros (entidad adjuntos) para que, posteriormente, el profesor la evalúe y le asigne una puntuación.
- **Adjuntos.** Como se ha dicho anteriormente, son los ficheros que adjunta el alumno al contestar una tarea.
- **Ficheros.** Son los ficheros que un usuario, sea alumno o profesor, puede subir a su repositorio personal.
- **Profesores y alumnos.** Representan al profesorado y alumnado de un centro educativo. Como se puede ver, la entidad Usuarios se especializa en Profesores y Alumnos. Un usuario solo puede ser o profesor o alumno, por lo que es una especialización disjunta. Por otra parte, es una especialización parcial, pues los administradores son usuarios pero no pertenecen ni a profesores ni a alumnos.

CAPÍTULO 4
Diseño de la solución

4.1 Arquitectura del sistema

En esta sección se va a detallar la arquitectura que tendrá nuestra aplicación, de forma que se divida en diversos componentes para un correcto funcionamiento de esta. Cómo anteriormente hemos elegido Django como nuestro *framework de backend*, seguiremos el patrón de diseño Modelo-Vista-Plantilla [1]:

- **Modelo.** Este componente representa la capa de acceso a la base de datos. Contiene toda la información sobre los datos, cómo acceder a estos, cómo validarlos, su comportamiento y las relaciones entre entidades.
- **Vista.** Es la capa de la lógica de negocios. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada.
- **Plantilla**. Representa la capa de presentación. Contiene todas las decisiones relacionadas con cómo se presentará la aplicación al usuario final, qué datos mostrar y cómo mostrarlos.

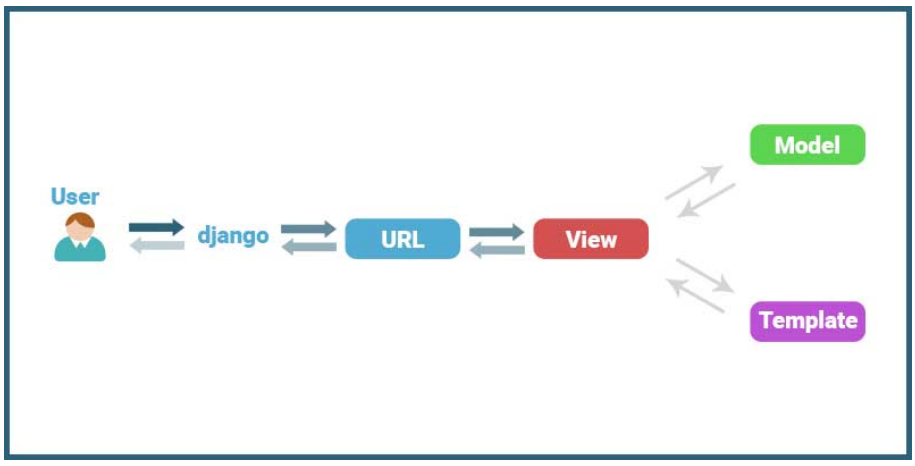


Figura 4.1: Explicación del Modelo-Vista-Plantilla utilizado por Django.

Como ejemplo para explicar el flujo en las interacciones entre componentes en esta arquitectura, cuando se hace click en un enlace o se escribe una dirección en la barra de direcciones del navegador, se accederá al archivo de configuración de URLs del proyecto,

[1] https://uniwebsidad.com/libros/django-1-0/capitulo-5/el-patron-de-diseno-mtv

en el que tenemos un mapeado de URLs en el que cada ruta tiene asociada una vista. Si la vista necesita algún dato, se accederá al modelo, el cual generará la consulta a la base de datos. Una vez que los datos han sido extraídos con éxito, se enviarán a la plantilla, la cuál mostrará las vistas al usuario en forma de página web, enviándosela como respuesta a su petición.

Esta arquitectura nos presenta diferentes ventajas muy útiles al momento de desarrollar una aplicación:

- **Acoplamiento débil.** La arquitectura MVT sigue el principio de acoplamiento débil, por lo que al realizar cambios sobre una de las piezas o partes de la aplicación tendrán poco o ningún efecto en las demás partes, lo que favorece en gran medida a la modularización.

- **Fácilmente modificable.** Al tener componentes separados, es fácil añadir nuevas funcionalidades a la aplicación sin que produzca efectos negativos en otras partes del proyecto.

4.2 Diseño detallado

4.2.1. Diseño de la base de datos

Como hemos dicho anteriormente, nuestra aplicación necesita almacenar distintos tipos de información, sobretodo la relacionada con los usuarios registrados en el sistema y los diferentes contenidos de cada asignatura. Dadas las entidades y sus relaciones del diagrama entidad-relación propuesto anteriormente, se ha realizado un diseño de la base de datos especificando los atributos de cada tabla, las claves ajenas y las nuevas tablas que surgen de las relaciones entre las entidades:

Como podemos observar, existen nuevas tablas que no aparecían en el diagrama entidad-relación:

- **Asignaturas-Profesores**. Esta tabla representa la docencia de las asignaturas por parte de los profesores. Por ello, contiene como claves ajenas tanto la id de las asignaturas como la de los profesores.

- **Asignaturas-Alumnos**. Representa la matrícula de los alumnos en cada asignatura. Por ello, contiene como claves ajenas tanto la id de las asignaturas como la de los alumnos.

- **Asignaturas-Categorías**. Muestra qué categorías tiene cada asignatura.

- **Entrega**. Es una tabla que surge de la relación entre Tarea y Alumno. Representa cuando un alumno ha entregado una tarea, es decir, si ha rellenado el contenido y/o ha subido algún fichero y lo ha guardado en el sistema. Almacena tanto la respuesta del alumno, cómo si ha sido evaluada o no y qué puntuación ha sido dada por el profesor.

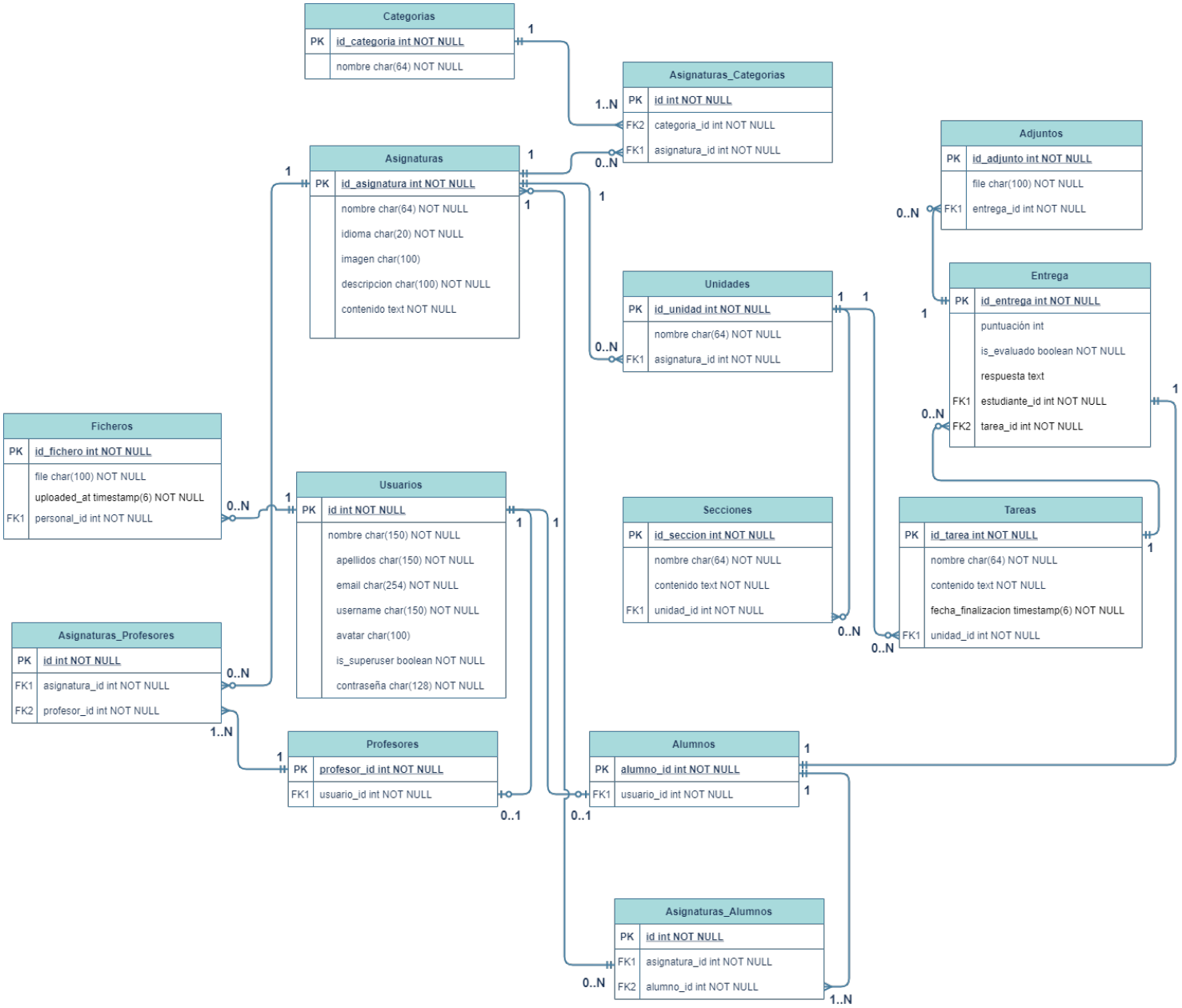


Figura 4.2: Diagrama dela base de datos de la aplicación.

4.2.2. Diseño de la lógica

La lógica de nuestra aplicación se encargará de diferentes tareas. Como se ha comentado anteriormente, nuestra aplicación debe permitir realizar funciones de administración tales como añadir alumnos y profesores de un centro docente, como asignaturas y categorías de estas. Por otra parte, debe permitir subir ficheros en un repositorio a los usuarios de la aplicación, así como ver las asignaturas de las que son partícipes. Además, como visión general, en cada asignatura el profesor debe poder crear unidades, secciones y tareas. También debe poder corregir estas últimas pudiendo dar una puntuación y un comentario.

Como alumno, el sistema debe ofrecer la posibilidad de ver el contenido de una asignatura, rellenar y entregar una tarea o ver las puntuaciones y comentarios de cada ejercicio propuesto por el profesor. La aplicación deberá enviar correos electrónicos a los

usuarios pertinentes cuando una actividad sea creada y cuando el profesor corrija una tarea. Por ello, la figura 4.3 representa la estructura de clases tendrá nuestro sistema.

Como se puede ver, se ha decidido que los atributos de las clases sean públicos ya que su acceso es controlado por el framework Django, y en raras ocasiones se utilizan estos atributos como privados cuando se trabaja con él. A continuación, se explicarán las diferentes vistas que contienen la mayor parte de la lógica de la aplicación.

- **AutenticacionView**. Esta clase se encarga de funciones de autenticación tales como iniciar sesión, cerrar sesión y registro de alumnos y profesores.
- **AlumnoView y ProfesorView**. Contendrán las funcionalidades básicas relacionadas con las clases Profesor y Alumno. Principalmente reúne funcionalidades como listar, edición, borrado y búsqueda, así como su visualización.
- **AdminView**. Su funcionalidad es mostrar la página principal de la administración y de enviar un correo electrónico cuando un profesor o alumno es creado para que pueda introducir una contraseña para su cuenta nueva.
- **IndexView**. Se encarga de mostrar las página de bienvenida y la página principal una vez un profesor o alumno inicia sesión. Además contiene diferentes métodos que son usados para mostrar datos que son comunes a la mayoría de páginas de la aplicación.
- **RepositorioView**. Se encarga de toda la lógica del repositorio de ficheros propios de cada usuario, como son la subida y borrado de archivos.
- **CategoriaView, UnidadView y SeccionView**. El objetivo de estas clases es la visualización, creación, edición y borrado de categorías, unidades y secciones respectivamente.
- **AsignaturaView**. Contiene toda la lógica relativa a las asignaturas, como es su creación, edición, visualización y borrado de estas. También contendrá la edición por parte del profesorado de su descripción y parte de su contenido, y un listado de los estudiantes que están matriculados en ella. Por último, se encargará de que cada alumno pueda visualizar correctamente las puntuaciones de sus tareas entregadas.
- **TareaView**. Al igual que las anteriores, también contiene todas las funcionalidades consideradas básicas para la creación y borrado de tareas. Además, se encarga de que los profesores puedan corregir las tareas, como de que los alumnos puedan entregarlas.

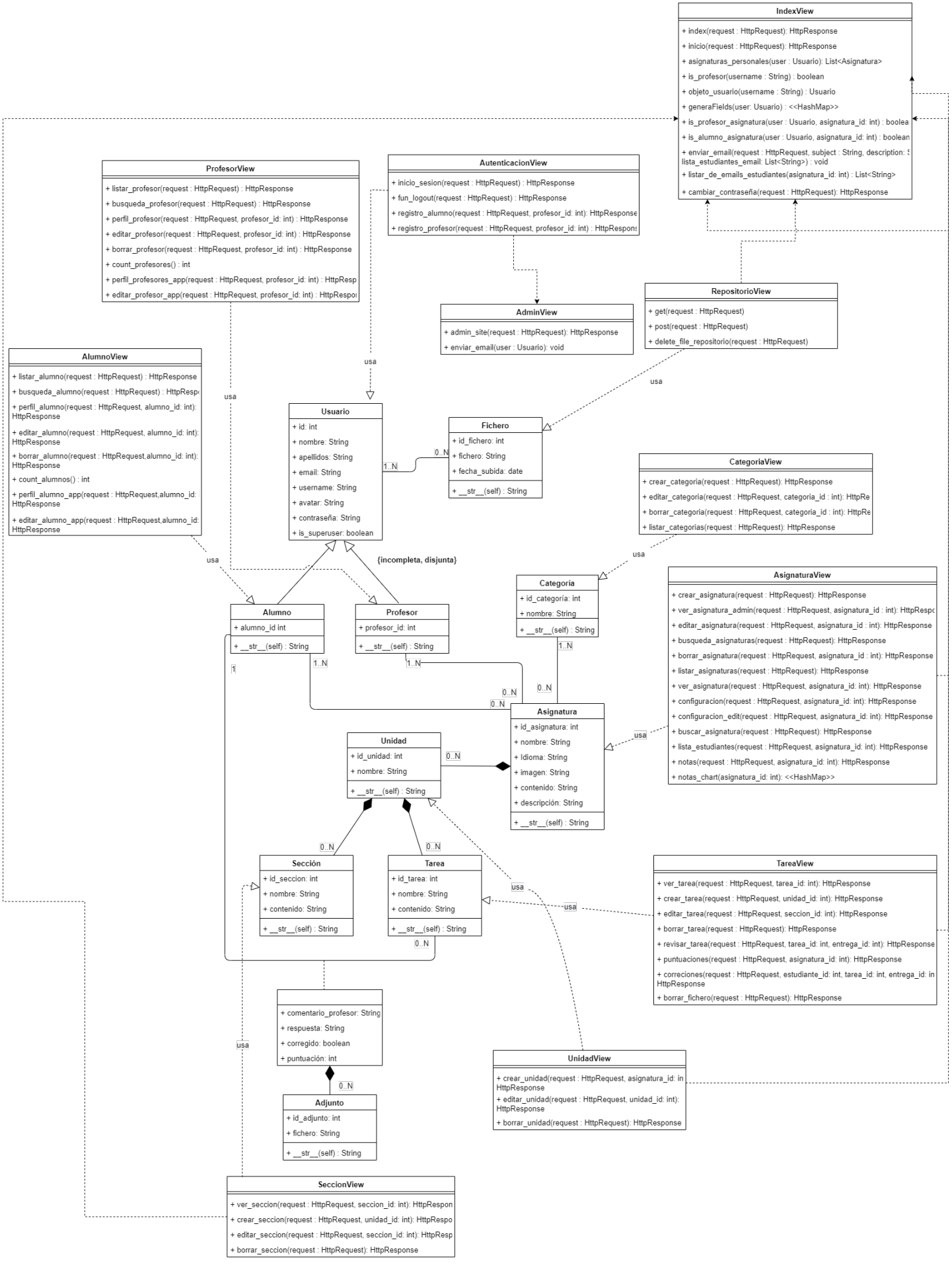


Figura 4.3: Diagrama de clases de la aplicación.

4.2.3. Capa de presentación: Mockups

En esta sección se presentarán los *mockups* basados en las funcionalidades más importantes descritas en el apartado 3.2. De esta forma el usuario es capaz de observar el estado de cada caso de uso de la aplicación.

4.2.3.1. Iniciar sesión

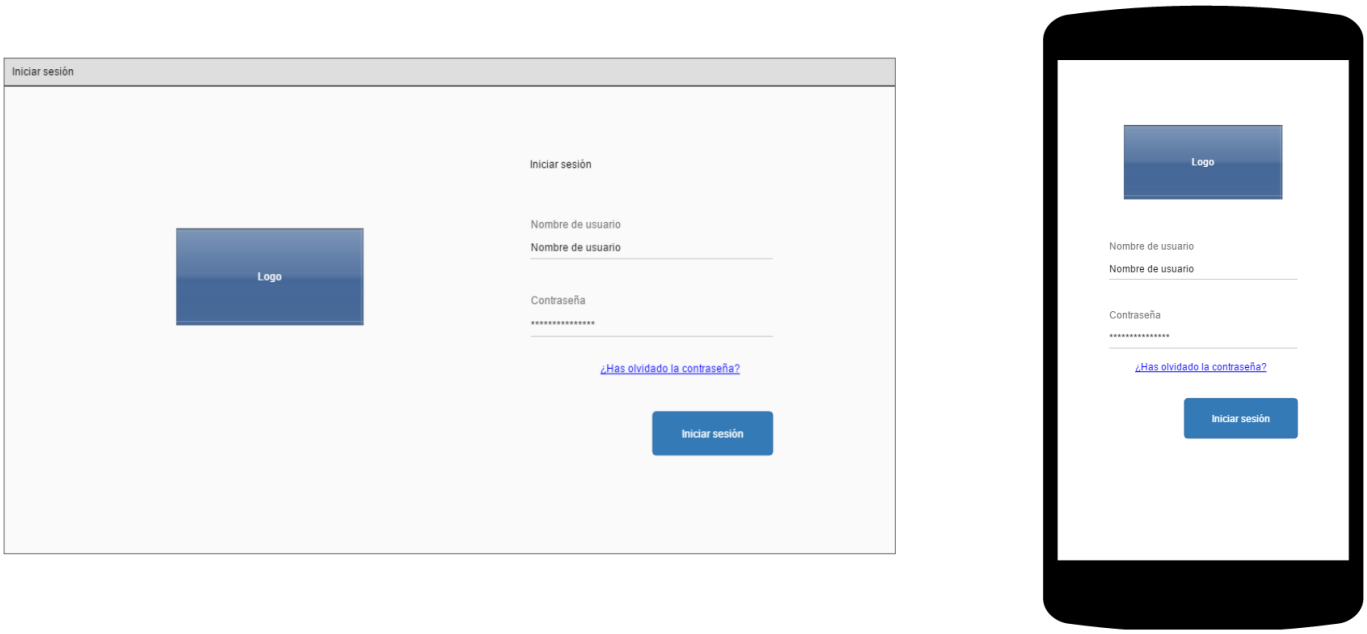


Figura 4.4: Mockup para el caso de uso 1 - Iniciar sesión.

4.2.3.2. Ver perfil

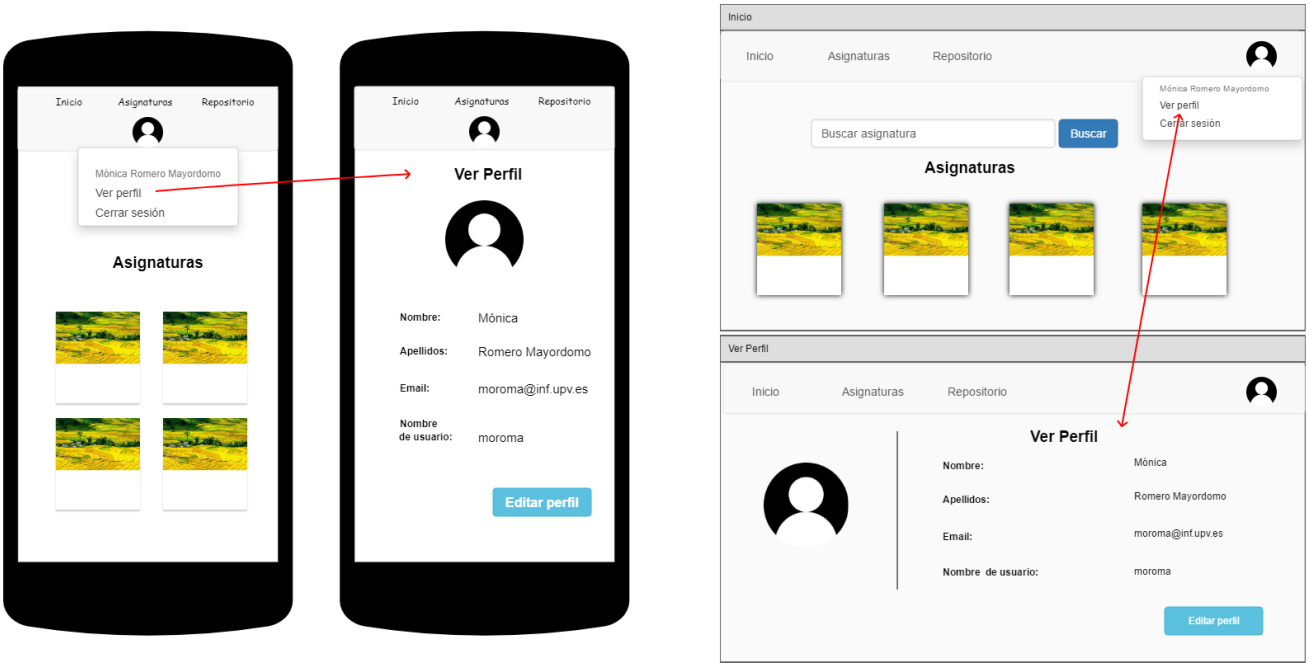


Figura 4.5: Mockup para el caso de uso 2 - Ver perfil.

4.2.3.3. Editar perfil

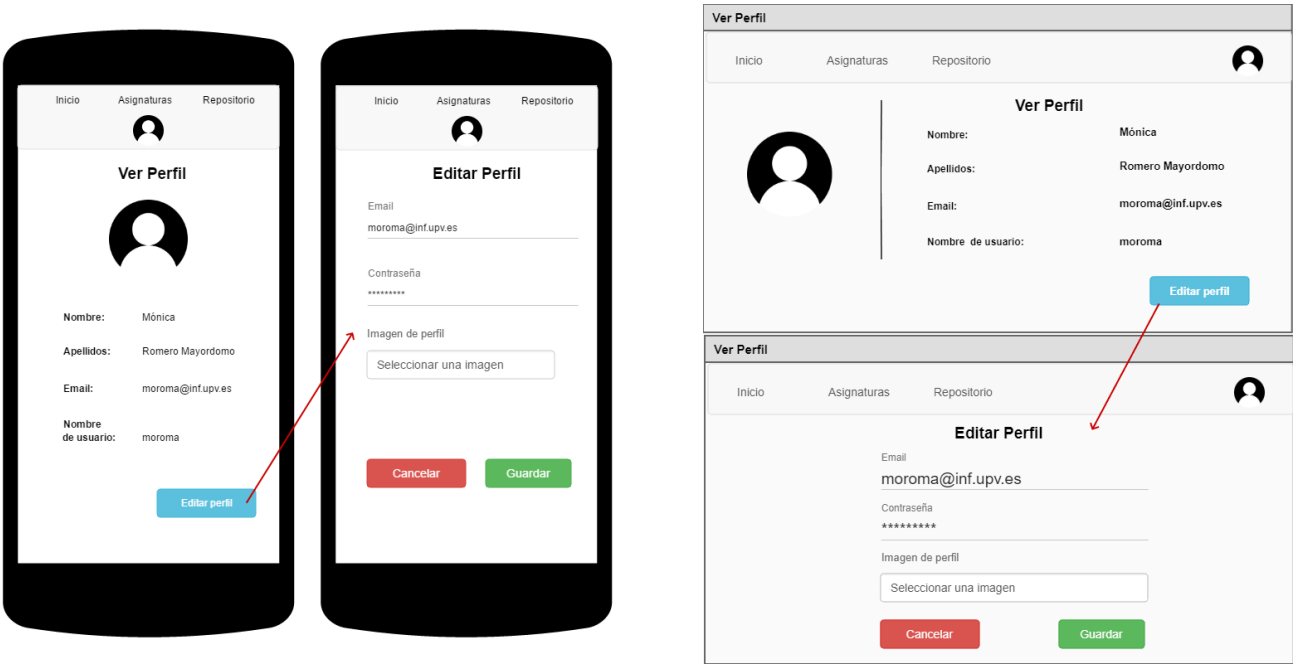


Figura 4.6: Mockup para el caso de uso 3 - Editar perfil.

4.2.3.4. Listar objetos

En este *mockup* se observa el listado de alumnos en representación al resto de objetos lógicos de la aplicación.

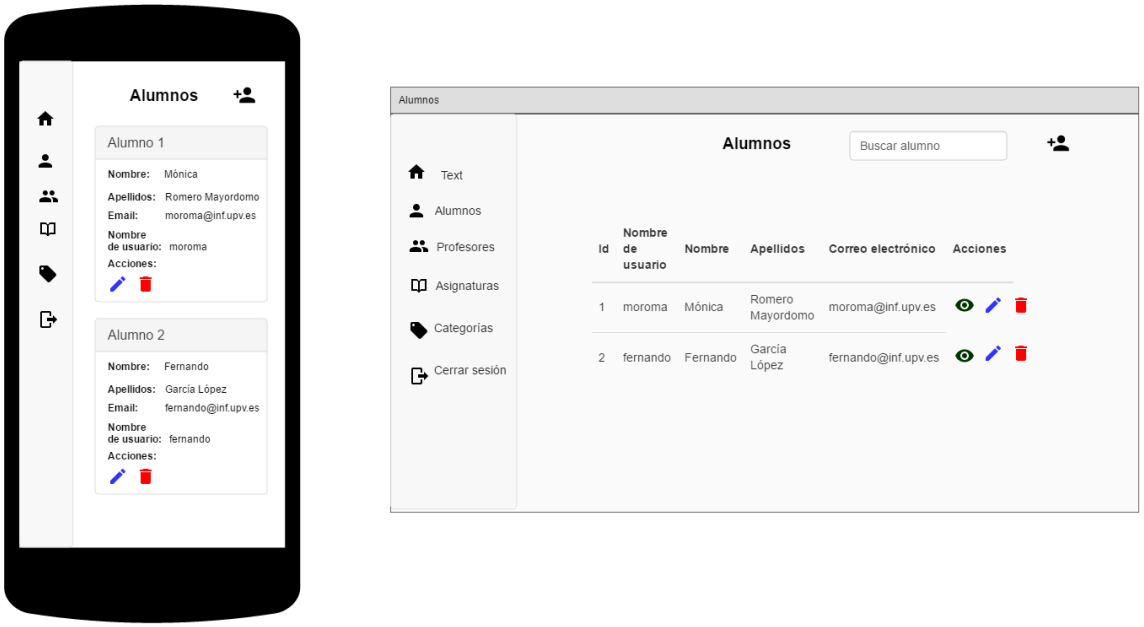


Figura 4.7: Mockup para el caso de uso 4 - Listar objetos.

4.2.3.5. Añadir y editar objetos

Al igual que en el anterior, se observa cómo añadir un alumno en representación al resto de objetos lógicos de la aplicación. El caso de uso 6 - Editar objetos es muy similar a excepción del título "Añadir Alumno" que se transformaría en "Editar Alumno", por lo que se obvia su muestra.

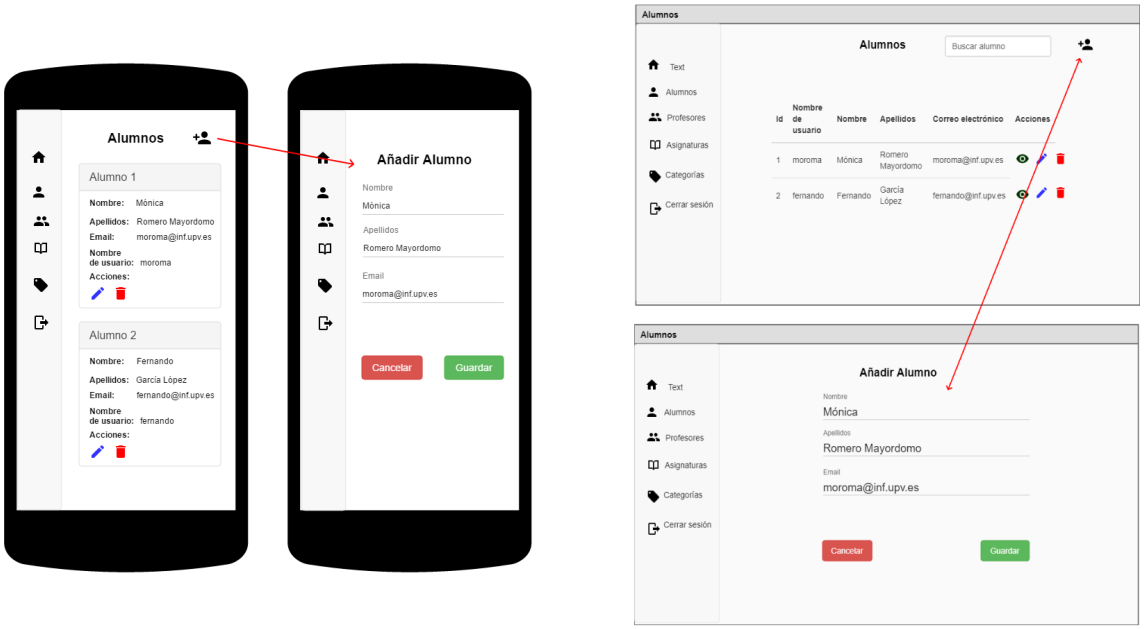


Figura 4.8: Mockup para los casos de uso 5 y 6.

4.2.3.6. Borrar objetos

La alerta que se puede observar en la figura 3.6 aparecerá cuando se pulse el botón con forma de basura de un objeto (figura 3.5). Al aceptar, el objeto se borrará tanto de la aplicación como de la base de datos.

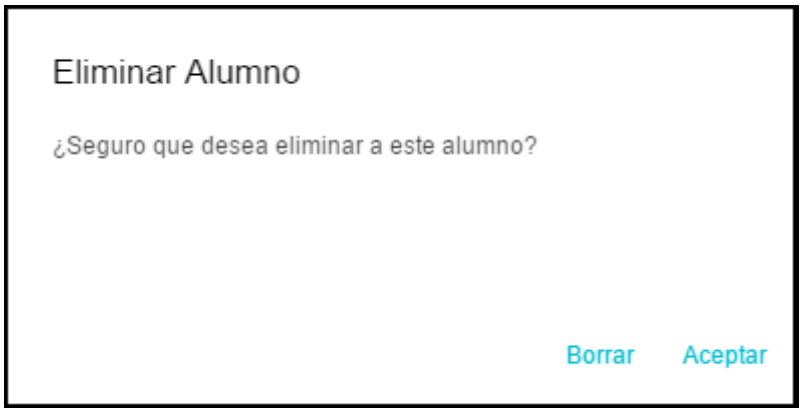


Figura 4.9: Mockup para el caso de uso 7 - Borrar objetos.

4.2.3.7. Visualizar y buscar asignaturas asignadas

En el siguiente *mockup*, la zona roja simboliza el caso de uso 8, en el que aparecen las distintas asignaturas en las que el usuario es partícipe, tanto en el desplegable como en la parte inferior. Por otro lado, la flecha azul indica el caso de uso 9, en el que tenemos una barra de búsqueda con la que podremos buscar materias asignadas introduciendo su nombre.

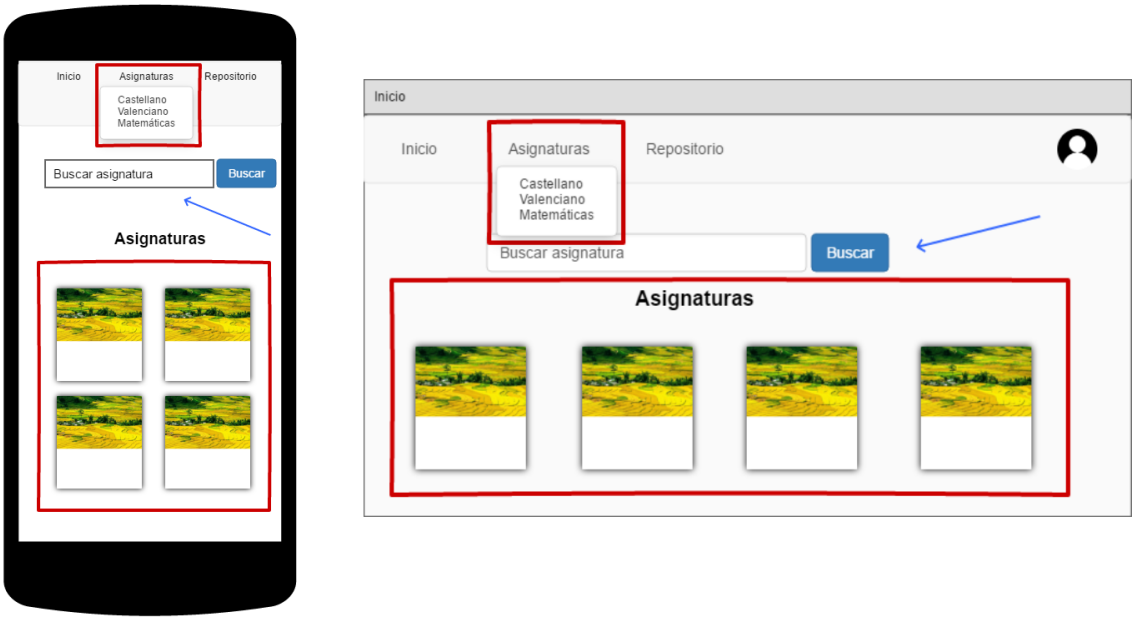


Figura 4.10: Mockup para los casos de uso 8 y 9 - Visualizar y buscar asignaturas asignadas.

4.2.3.8. Subir ficheros a un repositorio personal

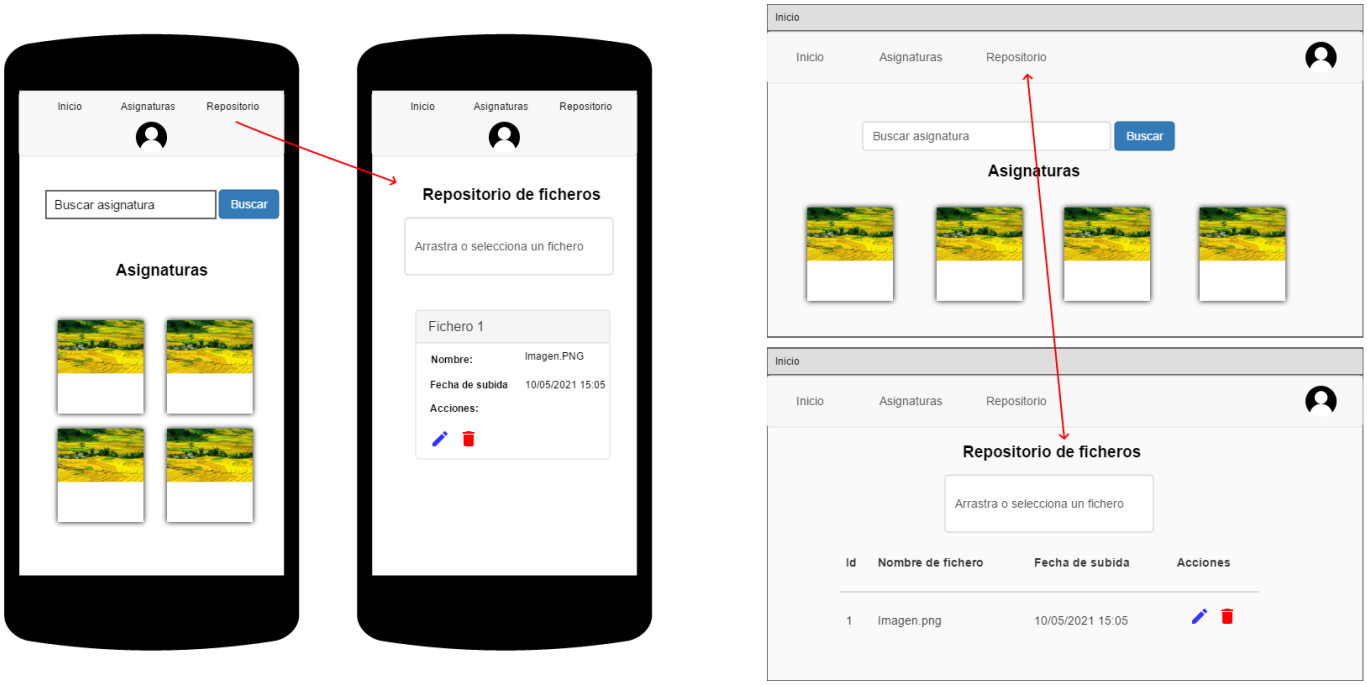


Figura 4.11: Mockup para el caso de uso 10 - Subir ficheros a un repositorio personal.

4.2.4. Visualizar contenido de una asignatura

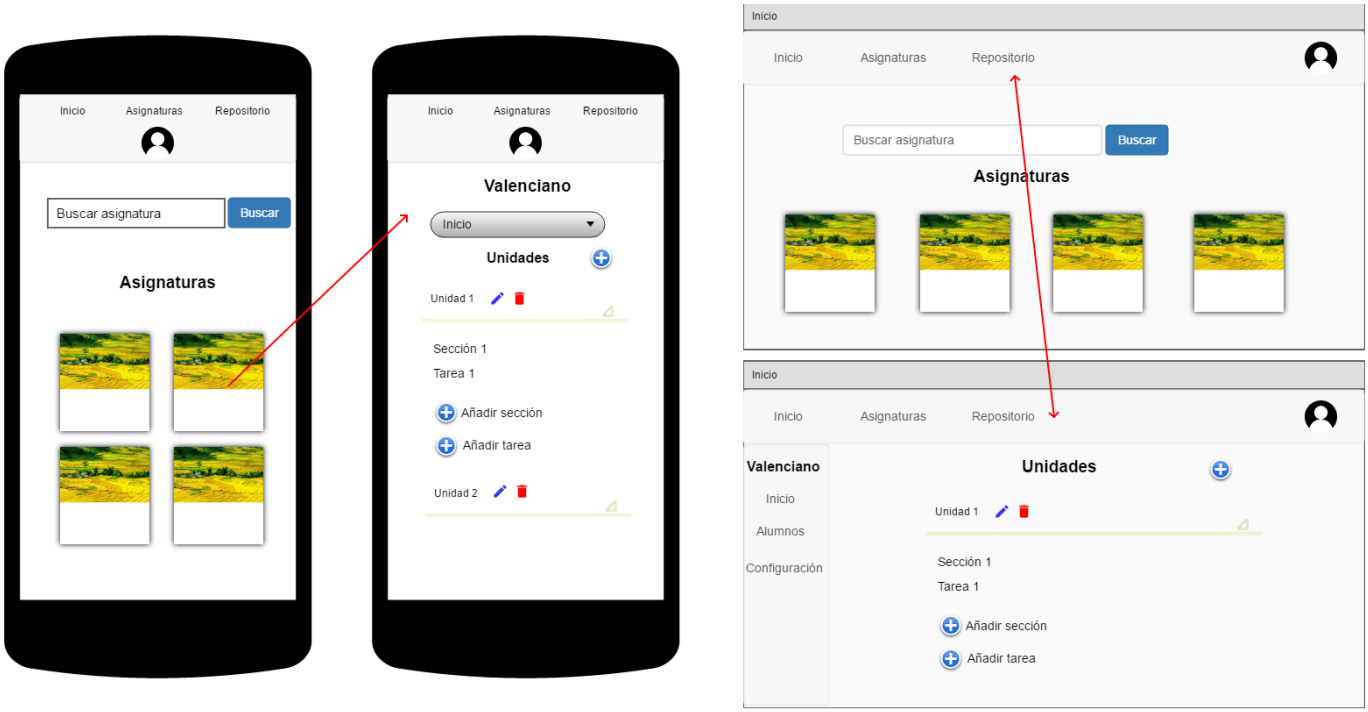


Figura 4.12: Mockup para el Caso de uso 11 - Visualizar contenido de una asignatura.

4.2.4.1. Editar configuración de la asignatura

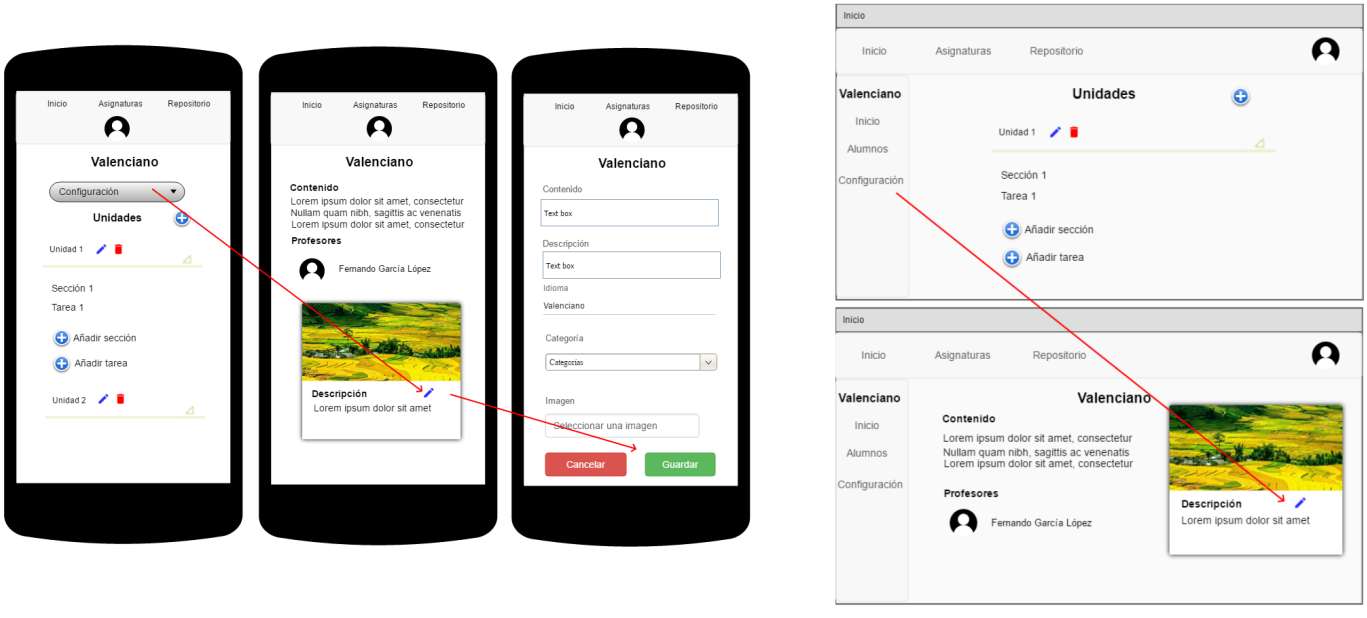


Figura 4.13: Mockup para el caso de uso 12 - Editar configuración de la asignatura.

4.2.4.2. Listar alumnado de una asignatura

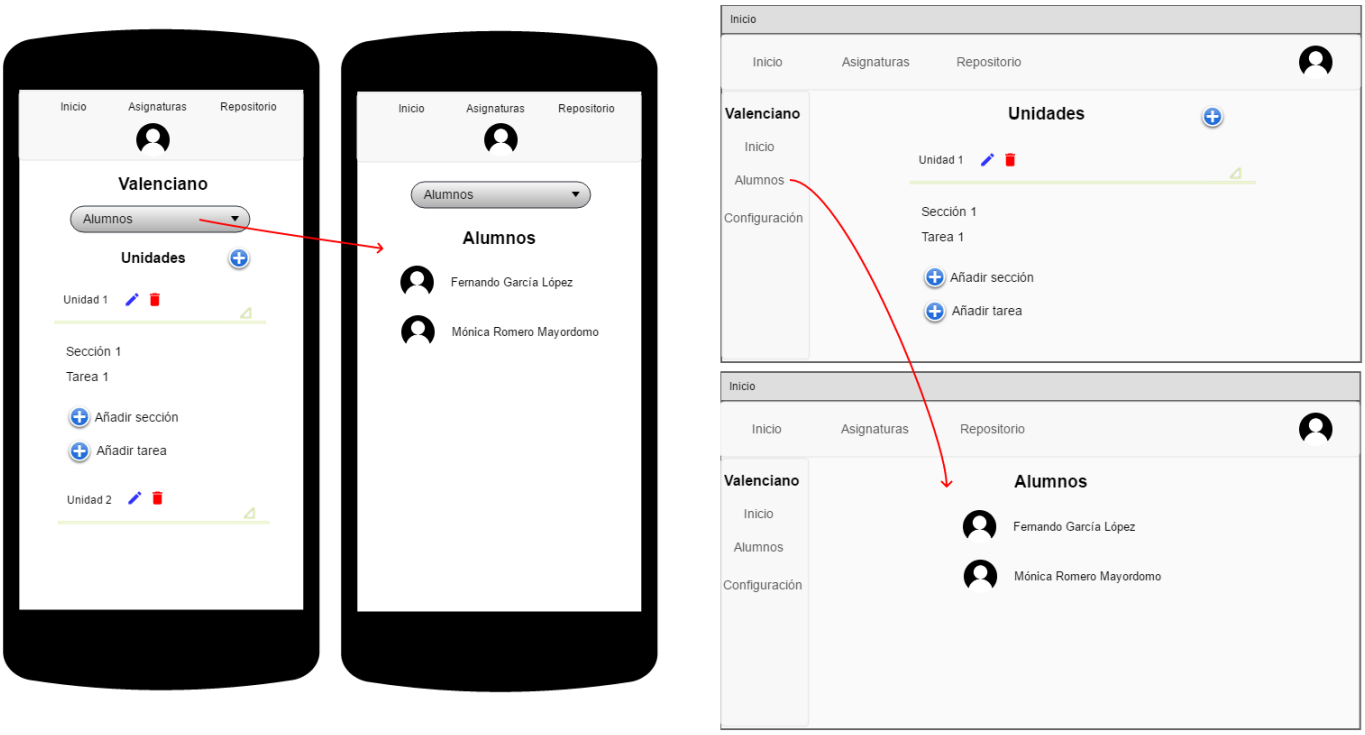


Figura 4.14: Mockup para el caso de uso 13 - Listar alumnado de una asignatura.

4.2.4.3. Corregir tarea

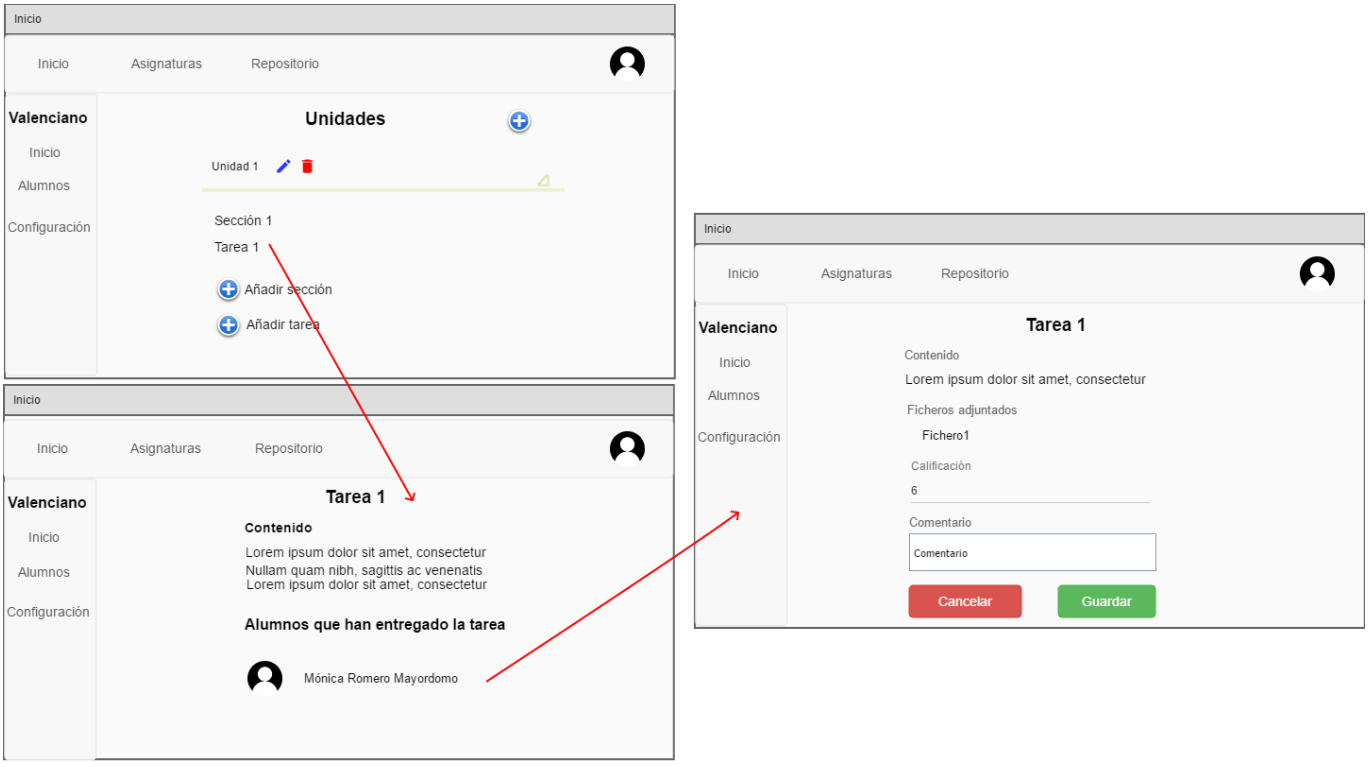


Figura 4.15: Mockup para el aso de uso 14 - Corregir tarea.

4.2.5. Entregar tarea

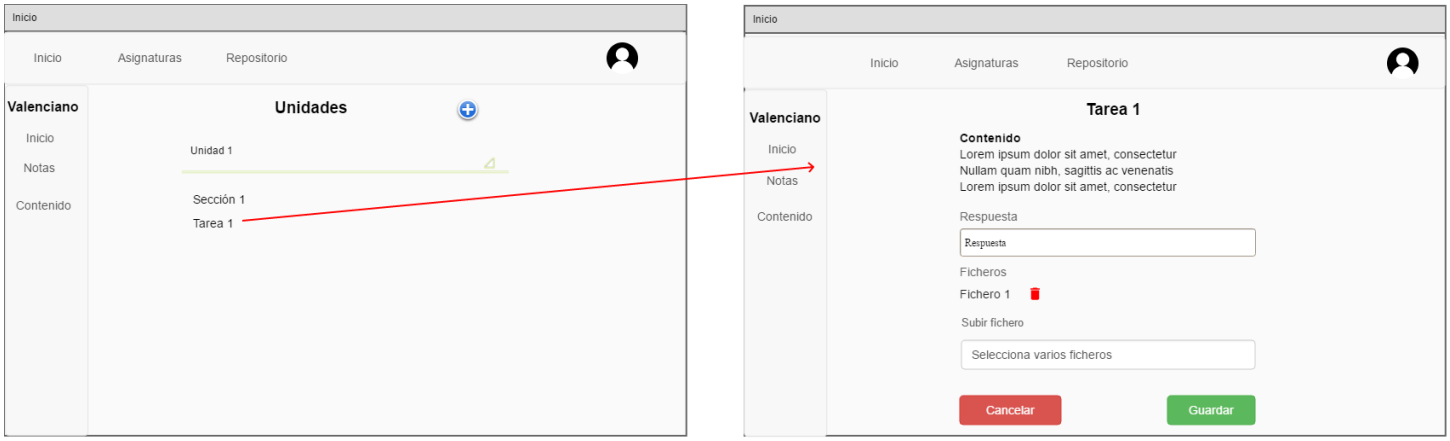
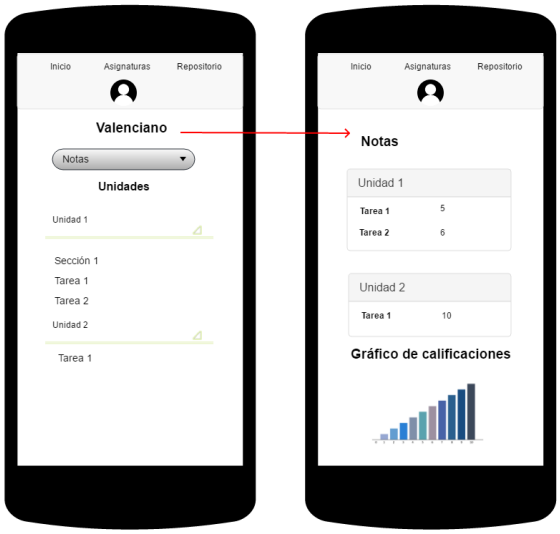


Figura 4.16: Mockup para el caso de uso 15 - Entregar tarea.

4.2.5.1. Visualizar calificaciones



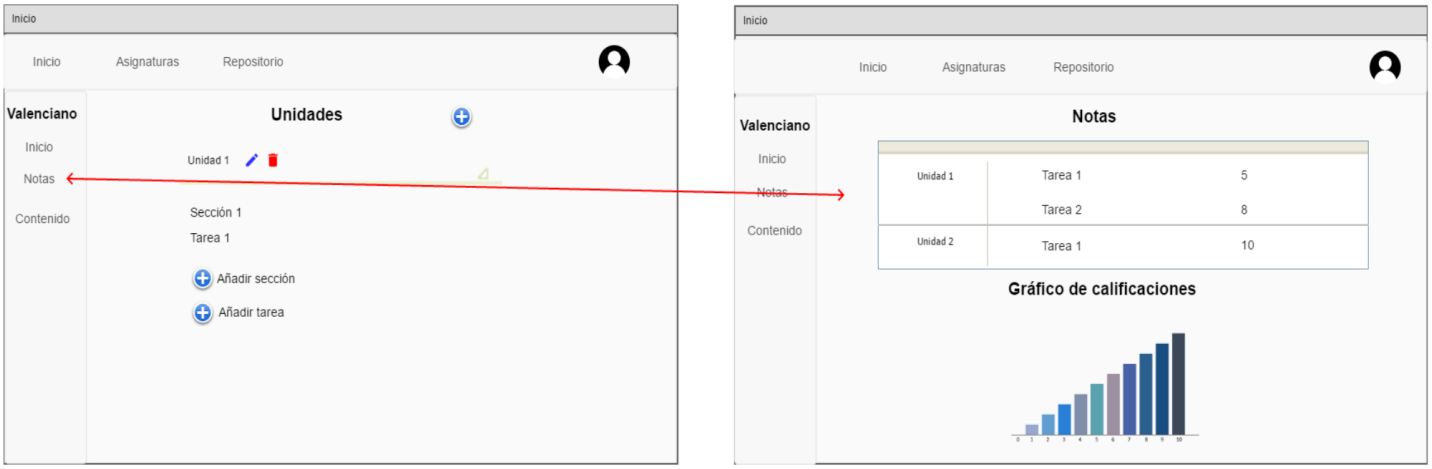


Figura 4.17: Mockup para el caso de uso 16 - Visualizar calificaciones.

4.2.5.2. Recuperar contraseña

En el siguiente *mockup*, una vez que el usuario escriba su correo electrónico para recuperar su contraseña, se le enviará un e-mail con instrucciones para recuperarla.

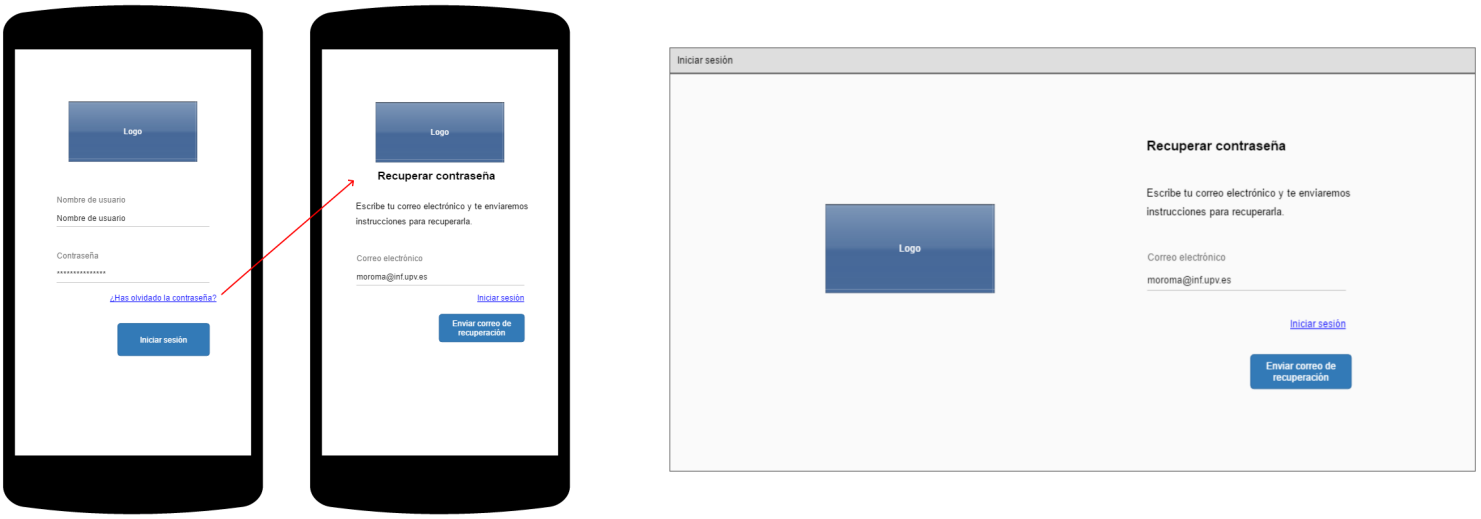


Figura 4.18: Primer mockup para el caso de uso 17 - Recuperar contraseña.

Después de seguir las instrucciones del correo electrónico, se le redirigirá a esta pantalla para que introduzca su nueva contraseña, la cual usará para futuros accesos en la aplicación.

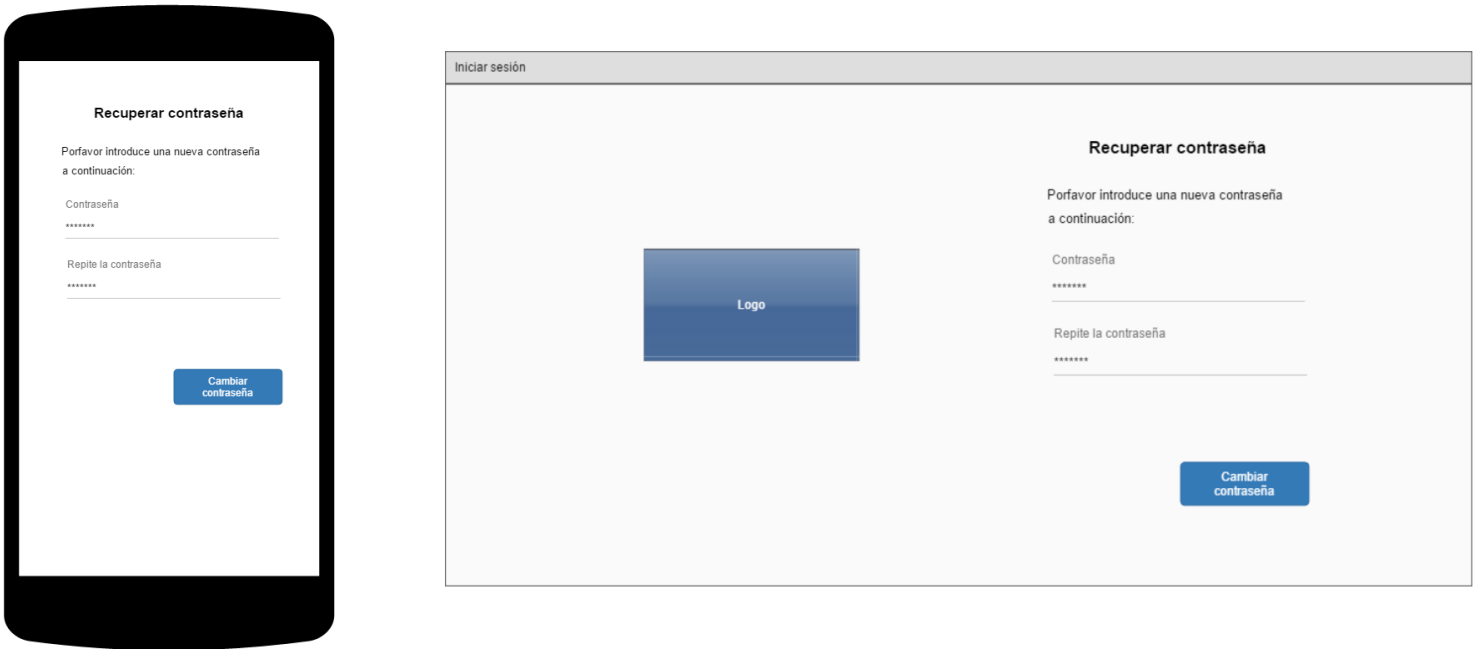


Figura 4.19: Segundo mockup para el caso de uso 17 - Recuperar contraseña.

4.3 Tecnología Utilizada

A continuación, se presentará las distintas tecnologías y aplicaciones que han sido utilizadas para el desarrollo de este proyecto:

4.3.1. Tecnologías

4.3.1.1. Django

Como se ha dicho anteriormente, Django es un framework de alto nivel basado en Python que permite el desarrollo rápido de sitios web tanto seguros como mantenibles.[2] Es de código abierto y, por tanto, gratuito. También tiene una gran comunidad activa, por lo que dispone soporte y de una gran documentación. Contiene muchos módulos incorporados de base que proporcionan funciones como iniciar sesión y ORM, el cual permite mapear las estructuras de una base de datos relacional a una estructura lógica como objetos.

Figura 4.20: Logo de Django.

Por último, este framework tiene una sintaxis práctica, lo que lo hace ideal para detección de errores, y tiene un soporte de base de datos incorporado, lo que proporciona una sencilla utilización de la mayoría de base de datos populares. En nuestro proyecto se ha utilizado la versión 3.2.

4.3.1.2. PostgreSQL

Figura 4.21: Logo de PostgreSQL.

Es un gestor de base de datos de tipo relacional de código abierto, en el cual es posible ejecutar tanto consultas relacionales como no relacionales. Las consultas relacionales son basadas en SQL, mientras que las no relacionales utilizan JSON. Por otro lado, permite manejar un gran volumen de datos y tiene un soporte total de ACID (atomicidad, consistencia, aislamiento y durabilidad), lo que lo hace un sistema seguro y con bajo porcentaje de pérdida de datos. Para nuestro proyecto se ha utilizado la versión 12.7.

4.3.1.3. HTML 5

HTML es el lenguaje de hipertexto con el que se define el contenido de las páginas web. Se trata de un conjunto de etiquetas que sirven para definir el texto y otros elementos que componen una página web, como pueden ser imágenes, vídeos, listas, etc.

Figura 4.22: Logo de HTML 5.

[2] https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Introduction

4.3.1.4. JavaScript y jQuery

Figura 4.23: Logo de jQuery.

Javascript es un lenguaje de programación que se encarga de dar interactividad y dinamismo a las páginas web. Cuando se ejecuta en un servidor, no requiere de compilación, lo que lo hace un lenguaje interpretado por el navegador. Gracias a este lenguaje, podemos crear, en el lado del cliente, efectos y animaciones sin ninguna interacción o como respuesta a eventos causados por el usuario tales como la pulsación de botones.

Por otro lado tenemos Jquery. Es una librería basada en JavaScript, que permite que la manipulación, el manejo de eventos, las animaciones y las peticiones Ajax sean mucho más simples de utilizar que con JavaScript nativo. Hemos utilizado la versión 3.6.0.

4.3.1.5. Bootstrap

Es un framework CSS que fue desarrollado por Twitter en 2010 para estandarizar las herramientas utilizadas en la compañía. Un año después se transformó en código abierto, y a día de hoy ha sido actualizado varias veces hasta la versión 5.1

Figura 4.24: Logo de Bootstrap.

Bootstrap [3] combina CSS y JavaScript para estilizar los elementos de una página HTML, lo que proporciona muchas funcionalidades útiles a la hora de desarrollar una página web. Además, proporciona una serie de componentes que facilitan la comunicación con el usuario, como pueden ser menús de navegación, controles de página, barras de progreso, etc. Una de sus mejores características es la construcción de sitios web adaptables a dispositivos móviles y tablets, por lo que hemos decidido utilizarlo. Se ha utilizado la versión 5 de Bootstrap.

4.3.2. Herramientas

4.3.2.1. Visual Studio Code

Figura 4.25: Logo de Visual Studio Code.

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Permite trabajar de forma cómoda con diferentes lenguajes de programación, gestionar atajos del teclado y refactorizar código. Es de código abierto y gratuito, y proporciona una utilidad para gestionar y descargar extensiones con la que poder personalizar y añadir funcionalidad al editor.

4.3.2.2. Github

Github [4] es un portal creado para alojar código en la nube escrito por cualquier desarrollador. La web utiliza un sistema de gestión de versiones con el que los desarrolladores

[3] https://getbootstrap.com/
[4] https://www.xataka.com/basics/que-github-que-que-le-ofrece-a-desarrolladores

pueden administrar su proyecto, por lo que no se pierden estados anteriores del proyecto cuando se va a actualizar por si en algún momento se necesita recuperar alguna parte.

En nuestro proyecto se ha instalado las herramientas de Git que ofrecen algunas distribuciones Linux y se ha configurado de forma que contacte con el proyecto creado en la plataforma de Github.

Figura 4.26: Logo de Github.

4.3.2.3. Pencil Project

Es una herramienta que permite realizar prototipos de GUI de código abierto[5]. Contiene diferentes plantillas para diagramas y prototipos, edición de texto y las operaciones de dibujo estándar. En este proyecto se ha utilizado para crear los prototipos vistos anteriormente a razón de enseñar un primer diseño de la aplicación.

4.3.2.4. Drawio

Drawio [6] permite la creación y edición de diagramas. Consiste en una aplicación web realizada en JavaScript y licenciada con Apache v2 y cuenta con diversos modelos para diversos tipos de UML, como pueden ser diseño de base de datos, diagramas de clases o diagramas entidad-relación como los vistos anteriormente.

Figura 4.27: Logo de Pencil Project.

Figura 4.28: Logo de Drawio.

4.3.2.5. Windows Subsystem for Linux: WSL

WSL [7] es una característica que permite a los usuarios de Windows 10 usar la consola de la distribución Linux que se elija entre las opciones disponibles en su tienda, por lo que permite a los desarrolladores crear aplicaciones en estos sistemas operativos sin necesidad de instalarlos. Ofrece la mayoría de herramientas de línea de órdenes, utilidades y aplicaciones de estos sistemas operativos Linux sin la sobrecarga que conlleva la ejecución de una máquina virtual.

[5] https://www.lafactoriacreativa.com/blog/diseno-web/herramientas-de-diseno-web-para-ahorrar-tiempo-ii-pencil-project/
[6] https://www.mancomun.gal/es/solucion-tic/draw-io/
[7] https://docs.microsoft.com/es-es/windows/wsl/about

4.3.3. Plataformas

4.3.3.1. Heroku

Es una plataforma de servicios que permite desplegar una aplicación en la nube sin necesidad de preocuparse por aspectos como el escalamiento o la administración. A Heroku se le debe indicar qué lenguaje de programación se utiliza en el proyecto y qué base de datos se utilizará para poder desplegar, además de las diferentes dependencias que use la aplicación desarrollada.

Heroku [8] tiene dos versiones, una gratuita y una de pago. La versión gratuita entra en modo de suspensión o *"sleep"* cada 30 minutos sin recibir tráfico, mientras que la de pago son 7 USD pero agrega ventajas como la administración de servidores. En este proyecto se utilizará la versión gratuita para reducir los costos de la aplicación. En el momento que se requiera una administración más compleja, se deberá cambiar a la versión de pago.

Figura 4.29: Logo de Heroku.

4.3.3.2. Amazon S3

Amazon S3 [9] es un servicio de almacenamiento de objetos en la nube que ofrece escalabilidad, disponibilidad de datos, seguridad y un buen rendimiento. Proporciona características de administración fáciles de utilizar que permiten configurar los accesos a los objetos y ficheros almacenados en el servicio.

En esta aplicación, se utilizará para guardar los archivos estáticos de la aplicación, como pueden ser imágenes decorativas en algunas pantallas, y dinámicos, como las imágenes de perfil que suban los usuarios y ficheros personales.

Figura 4.30: Logo de Amazon S3.

[8] https://platzi.com/blog/que-es-heroku/
[9] https://aws.amazon.com/es/s3/

CAPÍTULO 5
Desarrollo de la solución propuesta: YourStudies

Concluido el análisis del proyecto se comienza el desarrollo y la implementación de los casos de uso presentados anteriormente. En este apartado se explicará de forma general los diferentes pasos que se han seguido en el desarrollo, estructurando el contenido en cuatro apartados y adjuntando imágenes del código en los casos en que sea necesario.

5.1 Autenticación

En nuestra aplicación era imprescindible un sistema de autenticación para que los usuarios pudieran iniciar y cerrar sesión cuando fuera necesario. Para ello se ha hecho uso de la librería que incorpora Django la cuál añade ORM [1]. Como se ha dicho anteriormente, el ORM es un lenguaje de programación que permite mapear las estructuras de una base de datos relacionales, en nuestro caso la base de datos PostgreSQL, en una estructura lógica de entidades, lo que simplifica en gran parte el desarrollo pues las acciones de creación, lectura, actualización y borrado se hacen de forma indirecta, sin tener que hacer las consultas SQL a mano.

Estos objetos lógicos Django los llama "modelos", como hemos podido ver en la estructura de la figura 4.1. Además, Django nos ofrece un módulo de sistema de autenticación incorporado por defecto llamado *"django.contrib.auth"*, que nos proporciona dos métodos muy interesantes: *login y logout*. Estos dos métodos permiten que un usuario del sistema, registrados mediante el modelo *User* proporcionado también por este *framework*, inicie sesión o cierre sesión respectivamente, guardando su id en la sesión del navegador o eliminándola.

El modelo User que nos proporciona Django tiene bastantes atributos, por lo que se listarán los que se han utilizado en este proyecto:

- ***username***. Nombre de usuario.
- ***first_name***. Nombre del usuario registrado.
- ***last_name***. Apellidos del usuario registrado.
- ***email***. Correo electrónico.
- ***password***. Contraseña.

[1] https://www2.deloitte.com/es/es/pages/technology/articles/que-es-orm.html

- ***is_superuser***. Si el usuario es administrador y puede acceder al sitio de administración.

Como en nuestra aplicación tenemos tres tipos de usuario como son los administradores, profesores y alumnos, se ha decidido crear dos modelos: Alumno y Profesor, que contengan una relación uno a uno con el modelo User y la imagen de perfil de cada uno de ellos, con lo que así podremos utilizar el sistema de autenticación que nos proporciona el *framework*. Una consideración a tener en cuenta es que los administradores no son llevados al misma vista que los profesores y los alumnos, por lo que la aplicación redirigirá a los primeros al sitio de administración que se enseñará más adelante, y al resto a la página principal de YourStudies. A continuación se muestra el código de los modelos y de la función de inicio de sesión.

```python
# Create your models here.
class Alumno(models.Model):
    alumno_id = models.AutoField(primary_key=True)
    usuario = models.OneToOneField(User, on_delete=models.CASCADE, related_name="user_alumno")
    avatar = models.ImageField(upload_to='perfil_images/', null=True)

    def __str__(self):
        return f"Alumno {self.usuario.username}"

class Profesor(models.Model):
    profesor_id = models.AutoField(primary_key=True)
    usuario = models.OneToOneField(User, on_delete=models.CASCADE, related_name="user_profesor")
    avatar = models.ImageField(upload_to='perfil_images/', null=True)
    class Meta:
        verbose_name_plural = 'Profesores'

    def __str__(self):
        return f"Profesor {self.usuario.username}"
```

Figura 5.1: Clases que representan a los modelos de alumno y profesor.

```python
def inicio_sesion(request):
    if request.user.is_authenticated:
        if(request.user.is_superuser):
            return redirect('admin-site')
        return redirect('inicio')
    form = LoginForm()
    if request.method == "POST":
        form = LoginForm(data=request.POST)
        if form.is_valid():
            username = form.cleaned_data['username']
            password = form.cleaned_data['password']
            recuerdame = form.cleaned_data['recuerdame']
            user = authenticate(username=username, password=password)
            # Si existe el usuario con ese nombre y contraseña
            if user is not None:
                login(request, user)
                if not recuerdame:
                    request.session.set_expiry(0)
                if(user.is_superuser):
                    return redirect('admin-site')
                return redirect('inicio')
            else:
                return render(request, "autenticacion/login.html", {'form': form, 'error': "Nombre de usuario o contraseña incorrectos"})
    return render(request, "autenticacion/login.html", {'form': form})
```

Figura 5.2: Código que muestra el inicio de sesión.

5.2 Administración

En este apartado del desarrollo se incluyen todo lo relacionado con las tareas de administración, como son la creación y edición de los profesores y alumnos del centro educativo, las asignaturas que impartirán los profesores y las categorías que tendrán estas asignaturas. También incluye un listado y una búsqueda de cada uno de estos modelos, su eliminación del sistema y de la base de datos y la consulta de su información.

Para el registro y edición de los modelos se ha trabajado con *forms*. Django ofrece una clase llamada *Form* que crea automáticamente un formulario HTML con los atributos del modelo indicado por el desarrollador convirtiéndolos en etiquetas HTML como <input>o <select>. Además, controla los datos que han sido introducidos en el formulario y permite validarlos sin código adicional. Para ello, se debe indicar que método HTTP se usará, ya sea GET o POST. En nuestro caso, en estos *forms* se ha utilizado el método POST, pues vamos a hacer modificaciones en el servidor. Por último, se debe indicar a qué *url* debe enviar toda la información recogida, que se representará en el formulario HTML como el atributo *action*.

Además, para darle una apariencia más agradable a los formularios generados, se ha utilizado un módulo llamado *"django-crispy-forms"*, que indicándole que *framework CSS* se desea utilizar, personaliza los formularios utilizándolo al renderizar el HTML dinámicamente.

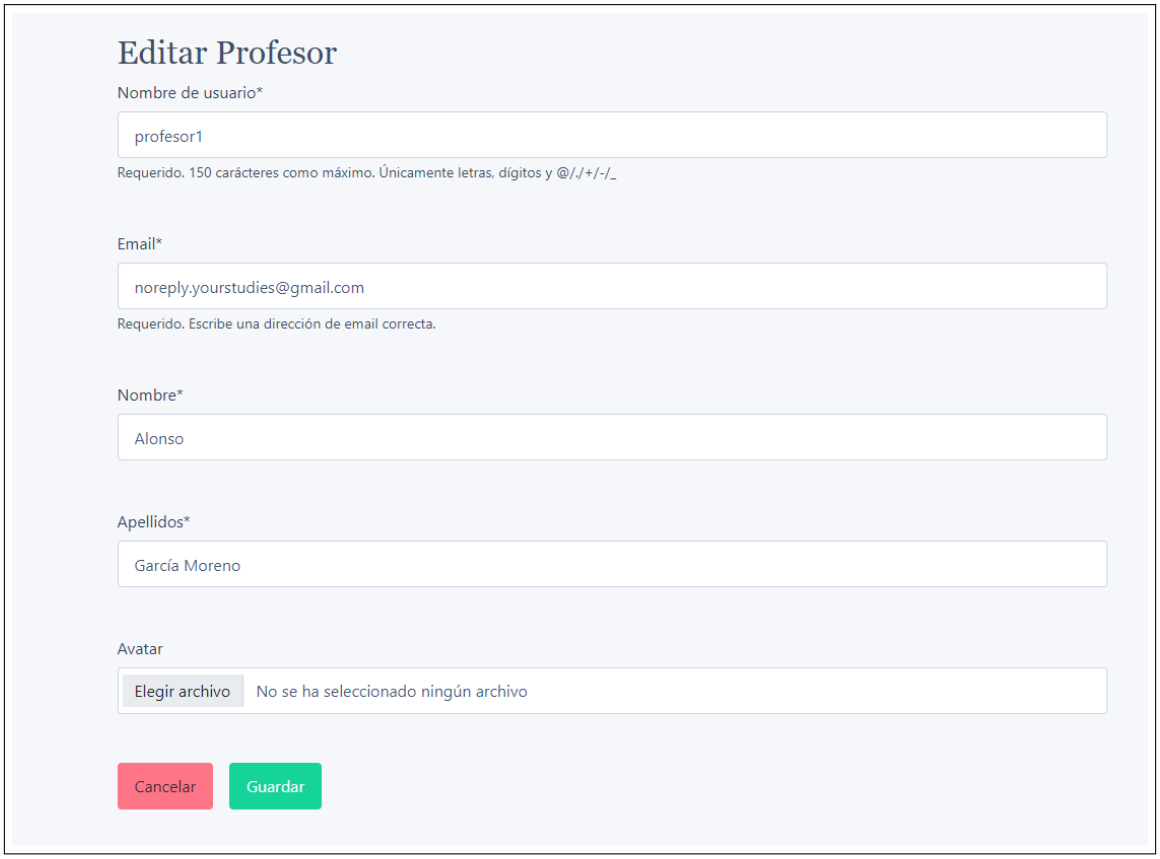


Figura 5.3: Ejemplo de visualización de un formulario.

Para el listado de las diferentes clases, se ha utilizado paginación para limitar el número de objetos que aparecen en pantalla y aligerar la carga de la página. El concepto de paginación se define como separar las páginas numéricamente de forma que muestre un

número limitado de objetos en pantalla, además de poder pasar de página en cualquier momento mediante un sistema de navegación integrado.

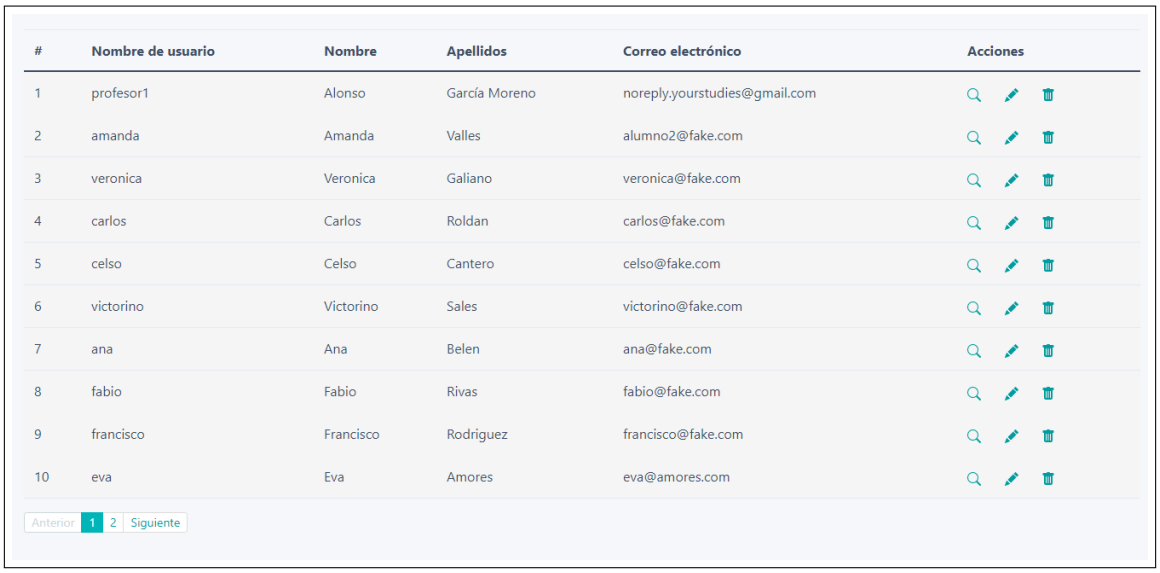


Figura 5.4: Ejemplo de visualización de paginación de una lista.

Por último, se ha implementado un sistema de envío de correos electrónicos a los alumnos y profesores cuando estos son añadidos al sistema. Cuando un usuario se añade al sistema, se le genera una contraseña aleatoria que es cifrada automáticamente por Django cuando es añadida a la base de datos. Como esta contraseña no se puede recuperar, al usuario correspondiente se le envía un e-mail con su nombre de usuario y un enlace para introducir una nueva contraseña. Una vez complete el formulario en el que introducirá una nueva contraseña, podrá iniciar sesión con ella. Este sistema también se ha implementado de una forma similar para la recuperación de contraseña a petición de un usuario introduciendo un correo electrónico.

```
1 EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
2 EMAIL_USE_LOCALTIME = True
3 EMAIL_HOST = 'smtp.gmail.com'
4 EMAIL_USE_TLS = False
5 EMAIL_USE_SSL = True
6 EMAIL_PORT = 465
7 EMAIL_HOST_USER = os.getenv('EMAIL_HOST_USER')
8 EMAIL_HOST_PASSWORD = os.getenv('PASSWORD')
```

Algoritmo 5.1: Configuración del correo electrónico que manda los correos electrónicos

5.3 Repositorio

Los usuarios tienen un apartado en el que pueden guardar ficheros personales y poder descargarlo en cualquier momento. Para ello, se ha desarrollado un *drag and drop* para que sea más intuitivo para el usuario adjuntar un fichero o varios simplemente arrastrándolos a la pantalla del navegador. Además, se ha añadido una tabla paginada que muestra todos los ficheros que tiene el usuario subidos, que se actualiza dinámicamente mediante AJAX cuando un usuario sube un nuevo fichero borrando el mismo número de filas de la tabla que el número de ficheros que se añaden nuevos. Por último, esos ficheros se pueden borrar apretando un botón con forma de basura, con el que aparece-

rá un modal en el que el usuario deberá confirmar si realmente desea borrar el fichero seleccionado o no.

Figura 5.5: Drag and Drop implementado en el repositorio.

5.4 Asignaturas

Dentro de una asignatura podemos encontrar unidades, que a su vez contienen secciones y tareas. Para ello se ha implementado un sistema de desplegables, en concreto acordeones, que despliegan el contenido de cada unidad cuando el usuario pulsa encima de ellos. Además, se ha implementado una barra de navegación a la izquierda en la que según el tipo de usuario tendremos distintas opciones. Si entramos como profesores, se nos habilitarán las opciones Estudiantes y Configuración, mientras que si accedemos como alumno, la opción de estudiantes se cambiará por Notas. Esta barra de navegación aparecerá en forma de desplegable en dispositivos móviles.

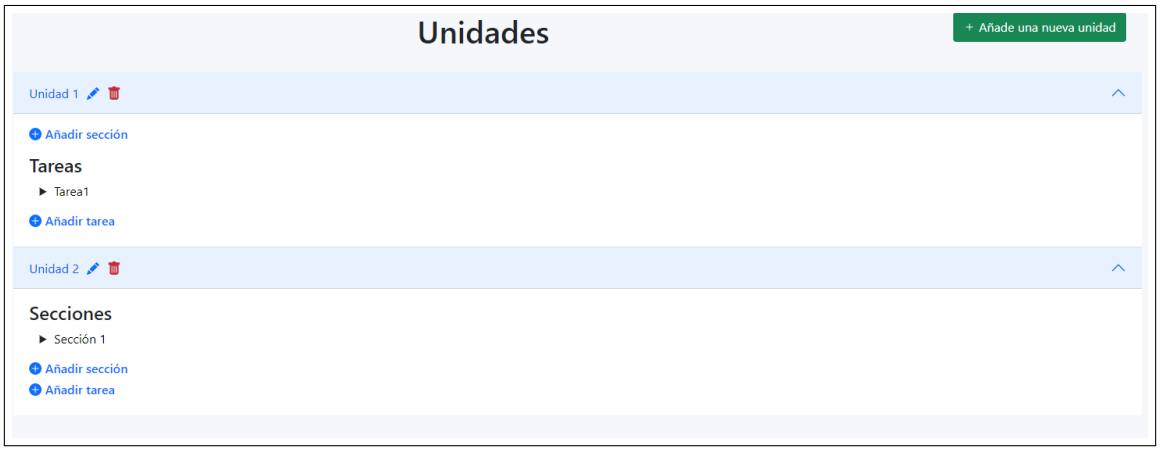


Figura 5.6: Desplegables de los contenidos de las unidades.

Para crear las unidades, secciones y tareas se han utilizado *forms* similares a los usados en la figura 5.3, pero con un añadido. Para los contenidos de las secciones y las tareas se ha utilidado CKEditor, un editor de texto HTML de código abierto que permite escribir listas y tablas entre otros, pues los *textarea* de HTML solo permiten texto plano. Para ello se ha utilizado un módulo llamado *Django CKEditor* [2] y se ha configurado en la aplicación según indica la API. Además también se han añadido varias extensiones que permiten funcionalidades como añadir vídeos de youtube y verlos desde la aplicación entre otras.

[2] https://django-ckeditor.readthedocs.io/en/latest/optional-for-file-upload

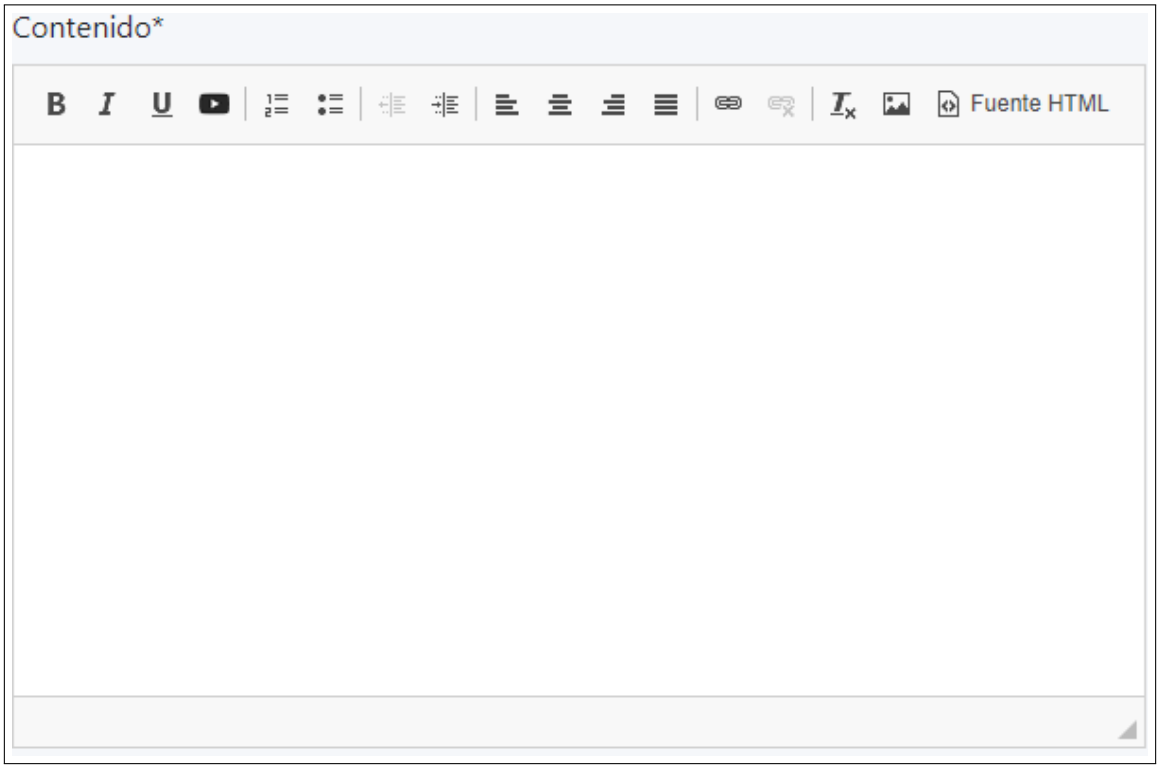


Figura 5.7: Visualización del editor CKEditor.

Por otra parte, en la creación de las tareas, la fecha de finalización tiene como restricción que no puede ser anterior a la fecha actual, pues si no los alumnos no tendrían tiempo de entregarla. Además se envía un correo a todos los alumnos matriculados en la asignatura avisándoles de que una tarea se ha abierto, concretando el nombre de la tarea y la asignatura en el correo electrónico. Una vez creada, el profesor podrá ver un listado de alumnos que ha entregado la tarea, y si pincha encima de uno de ellos, podrá ver la respuesta y los ficheros subidos por ese alumno, pero no podrá dar una puntuación ni un comentario hasta que el tiempo de la tarea haya finalizado ya que el alumno podrá cambiar su respuesta hasta que se cierre. Una vez que la haya evaluado, se enviará un e-mail al alumno indicándole su nota de la tarea.

Como se acaba de comentar, en el lado del alumno se pueden subir tantos ficheros como se quieran, y además se podrán eliminar los que ya hay subidos en el caso de un error o reemplazo. Esto último se realiza mediante AJAX al igual que los ficheros alojados del repositorio. En el apartado de Notas, podemos ver una tabla de las notas que ha recibido el alumno en las tareas de la asignatura correspondiente. Si una tarea no se ha entregado a tiempo, se pondrá un 0 automáticamente en este listado, si no ha acabado todavía o el profesor no la ha corregido, aparecerá en blanco; si el profesor la ha corregido, aparecerá la puntuación proporcionada por el profesor. Además se puede ver una gráfica de las tareas con sus respectivas puntuaciones a modo de resumen. Esta gráfica se ha implementado mediante la librería *chart.js*[3].

[3]https://www.chartjs.org/

```python
def notas_chart(asignatura_id, context):
    labels = []
    data = []

    lista_tareas = []
    unidades = Unidad.objects.filter(asignatura__id_asignatura=asignatura_id)
    for unidad in unidades:
        tareas = Tarea.objects.filter(unidad=unidad)
        lista_tareas.extend(tareas)
    for tarea in lista_tareas:
        entrega = Entrega.objects.filter(tarea__id_tarea=tarea.id_tarea)
        labels.append(tarea.nombre)
        if entrega:
            data.append(entrega[0].puntuación)
        else:
            data.append(0)

    context['labels'] = labels
    context['data'] = data
```

Figura 5.8: Código que recupera las notas de un alumno para que la gráfica las utilice.

5.5 Estructura de los directorios

Por último, se muestra la organización de los directorios que componen la aplicación:

5.5.1. Carpeta YourStudies

Esta es la carpeta que contiene los ficheros de configuración de toda la aplicación. Se crea automáticamente en la creación del proyecto de Django junto al archivo manage.py, que contiene el código para ejecutar el servidor de la aplicación. El fichero urls.py actúa de router, es decir, en él se incluyen todas las urls que contendrá el proyecto y que la aplicación utilizará para redirigir las peticiones a la función indicada. Por otro lado, el fichero settings.py contiene todas las configuraciones que utilizará el servidor para distintas funciones, como son los módulos instalados, la configuración de la base de datos, en cuál carpeta se guardan los ficheros por defecto, etc. Por último, también contiene la carpeta *migrations*, donde almacena parte de las migraciones generadas automáticamente al crear las tablas a partir de los modelos escritos.

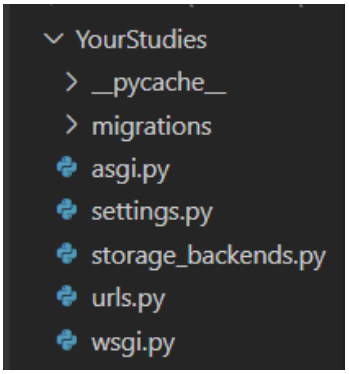


Figura 5.9: Organización de la carpeta principal YourStudies.

5.5.2. Carpetas *autenticacion* y *cursos_app*

La carpeta *autenticacion* representa toda la parte del proyecto relacionada con la autenticación de los usuarios y la parte administrativa, que es manejada por el supervisor de la aplicación. Por otro lado, la carpeta *cursos_app* contiene la mayoría de la lógica del proyecto: la página principal, el repositorio, la edición de perfil y toda la visualización y edición de los contenidos de las asignaturas. Estas dos carpetas comparten varios ficheros en común:

- **Carpeta migrations**. Al igual que la carpeta YourStudies, contiene las migraciones de la base de datos.
- **Carpeta static**. Esta carpeta contiene los ficheros estáticos de la app correspondiente. Se divide en tres subcarpetas:
  - **Carpeta css**. Contiene los ficheros css necesarios para la personalizar las páginas web.
  - **Carpeta js**. Incluye los ficheros con código JavaScript utilizado para dotar de dinamismo la aplicación.
  - **Carpeta images.** Contiene las imágenes estáticas de la aplicación, como el logo.
- **Carpeta templates**. Representa la parte visual de la aplicación. Incluye los ficheros HTML que representan las páginas web.
- **Carpeta templatetags**. Contiene pequeños trozos de código HTML usados en los *forms*.
- **Carpeta views**. Incluye toda la lógica de las respectivas aplicaciones. Cada fichero se llama "«nombre»View", siendo nombre el modelo o clase que hace referencia. Ej: Alumno - AlumnoView.
- **__init__.py** Fichero que contiene código autogenerado de inicialización de la aplicación.
- **admin.py** Este fichero contiene el código para que los modelos se registren en el administrador por defecto de Django.
- **apps.py** Fichero autogenerado que representa el nombre de la aplicación para poder incluirlo en settings.py y que el sistema de Django lo detecte.
- **forms.py** Incluye todos los forms que se utilizan para generar los formularios de creación y edición en la aplicación.
- **models.py** Representa los modelos que se utilizarán para generar las tablas en la base de datos y que utilizará el ORM.
- **test.py** Incluye el código para las pruebas técnicas de la aplicación.
- **urls.py** Contiene todas las urls que redirigen a la función de la lógica necesaria para el correcto funcionamiento de la aplicación.
- **widgets.py** Extensiones que utilizan los formularios para cambiar un componente concreto de este. Por ejemplo un *datepicker* personalizado.

Por último, podemos ver una carpeta *media*, donde se guardarán todas las imágenes dinámicas en diferentes carpetas que se auto generarán según los parámetros indicados. Como ejemplo, los avatares de los usuarios se guardan por fecha.

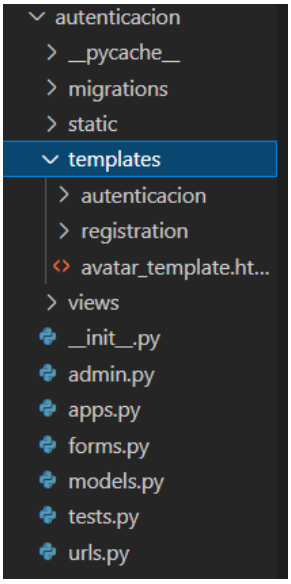


Figura 5.10: Estructura de la carpeta *autenticacion*.

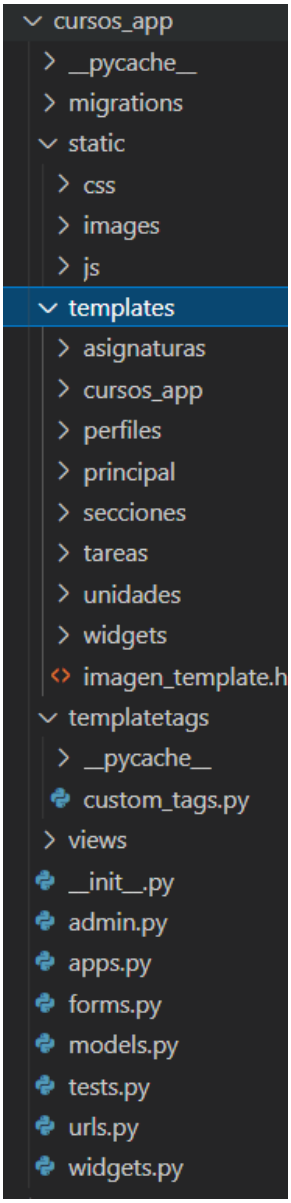


Figura 5.11: Estructura de la carpeta *cursos_app*.

CAPÍTULO 6
Implantación

Para poder implementar las funcionalidades anteriormente expuestas, ha sido necesaria la instalación de software, por lo que en este capítulo se mostrarán qué extensiones han sido necesarias para la puesta final en producción.

6.1 Instalación inicial

En este proyecto se ha decidido trabajar en un entorno Linux debido a que la instalación de componentes y extensiones del *framework* Django resultaba más sencilla. Para ello, se utilizará un subsistema WSL en Windows, que nos permite incluir la mayoría de utilidades y órdenes de Linux desde Windows, ya que así no es necesario configurar un arranque dual ni instalar un sistema operativo nuevo en el ordenador.

Para la instalación, es necesario habilitar el subsistema de Windows para Linux desde una terminal Powershell como administrador. Para ello, se deberá ejecutar la siguiente orden:

```
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux
    /all /norestart
```

Algoritmo 6.1: Orden para habilitar WSL en Windows

Una vez ejecutado, se deberá ir a la tienda de Microsoft y elegir la distribución que se quiera. En nuestro caso se ha elegido Ubuntu 20.04, pues es la versión más actualizada de este sistema operativo con una gran comunidad detrás apoyándolo. Una vez instalado, se podrá ejecutar como una terminal, en la que se tendrá que crear un usuario y una contraseña.

Una vez hecho esto, se actualizará el sistema e instalaremos Python para poder utilizar el framework Django. Para que todo lo que instalemos de Python no afecte a todo el sistema operativo, se ha decidido crear un entorno virtual por si en un futuro se desea realizar otro proyecto que necesite otras dependencias. Además, también se instalará pip [1], un programa que nos ayudará a instalar paquetes de Python. Cada vez que se requiera utilizar este entorno virtual, habrá que activarlo mediante una orden.

```
python3 -m venv yourStudies-tfg
sudo apt install python3-pip
```

Algoritmo 6.2: Creación de un entorno virtual e instalación del programa pip

Por último, instalaremos el framework Django y sus dependencias. Para crear un proyecto de Django deberemos ejecutar la siguiente orden:

[1] https://pypi.org/

```
1  django-admin.py startproject YourStudies
```

Algoritmo 6.3: Orden para crear el proyecto YourStudies de Django

Nos creará la carpeta YourStudies vista anteriormente en la figura 5.9. Para Django, la parte de la lógica está contenida dentro de lo que nombran como *apps* o aplicaciones. Se deberá crear las que se consideren necesarias. En nuestro caso hemos creado dos, "autenticacion" y "cursos_app", como se ha visto en el capítulo anterior. Deberemos editar la configuración principal de Django para que detecte estas aplicaciones:

```
1  python manage.py startapp autenticacion
2  python manage.py startapp cursos_app
3
4  #settings.py
5  INSTALLED_APPS = [
6  'autenticacion.apps.AutenticacionConfig',
7  'cursos_app.apps.CursosAppConfig',
8  ...
9  ]
```

Algoritmo 6.4: Órdenes para crear las aplicaciones "autenticacion" y "cursos_app" y su añadido al fichero de configuración del proyecto de Django

Una vez hecho esto, solo faltará instalar la base de datos PostgreSQL y modificar la configuración para que Django sepa dónde está el servidor. Por ello, instalamos PostgreSQL desde el gestor de paquetes de Ubuntu, y creamos una nueva tabla llamada yourstudies. Además deberemos cambiar la contraseña del usuario por defecto postgres.

Una vez instalado esto, se cambiará la configuración de Django, por lo que quedará así:

```
1  DATABASES = {
2
3    'default': {
4      'ENGINE': 'django.db.backends.postgresql_psycopg2',
5      'NAME': 'yourstudies',
6      'USER': 'postgres',
7      'PASSWORD': 'postgres',
8      'HOST': 'localhost',
9      'PORT': '5432',
10   }
11 }
```

Algoritmo 6.5: Configuración en el fichero settings.py para conectar el proyecto con la base de datos PostgreSQL

Como Django contiene un servidor propio integrado por defecto, no fue necesario instalar ninguna herramienta para poder desarrollar en él.

6.2 Configuración del correo electrónico con Gmail

Para los envíos de correo electrónicos que suceden en algunos eventos en la aplicación se ha utilizado el servicio de e-mail de Google: Gmail. Para ello, hemos creado ua nueva cuenta en Gmail llamada "noreply.yourstudies@gmail.com", que será la cuenta con la que se enviarán todos los correos automáticos.

Para ello, en el apartado de seguridad de la cuenta deberemos activar la verificación en 2 pasos, ya que es un requisito imprescindible para añadir aplicaciones. Una vez hecho esto, seguiremos unos pasos para añadir nuestra aplicación y generar una contraseña

única, que no podrá verse una vez se termine el proceso para evitar problemas de seguridad.

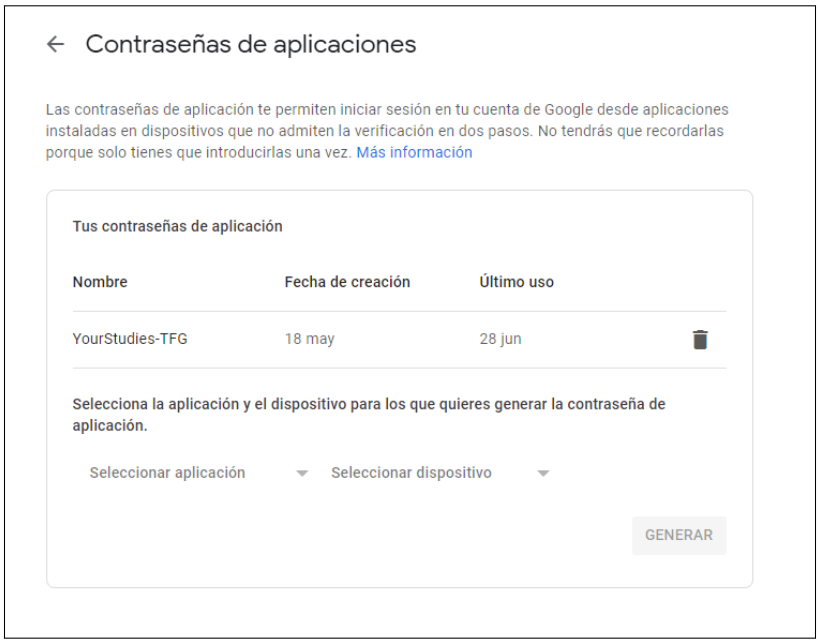


Figura 6.1: Contraseñas de aplicaciones guardadas en Gmail.

En el fichero *settings.py* de nuestro proyecto Django deberemos añadir la configuración siguiente:

```
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_USE_LOCALTIME = True
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_USE_TLS = False
EMAIL_USE_SSL = True
EMAIL_PORT = 465
EMAIL_HOST_USER = os.getenv('EMAIL_HOST_USER')
EMAIL_HOST_PASSWORD = os.getenv('PASSWORD')
```

Algoritmo 6.6: Configuración en el fichero settings.py para poder enviar correos electrónicos con Gmail

En nuestro caso, le hemos indicado que utilice SSL y el puerto 465 para enviar correos electrónicos, que es el predeterminado que nos indica Gmail. Además, deberemos indicarle el usuario y la contraseña con las que tenemos añadida la aplicación. Como utilizamos Github para el control de versiones, es aconsejable no subir información personal delicada, por lo que se recomienda utilizar la función *"os.getenv(«nombre»)"* y recopilar esta información en un archivo .env que el entorno virtual de Python usará para buscar el valor de ese nombre al hacer la llamada de esta función.

6.3 Amazon Storage

Como se quiere desplegar la aplicación en la nube, necesitamos un servicio que aloje todos los ficheros estáticos y no estáticos de la aplicación. Para ello se ha utilizado el servicio en la nube Amazon Storage o también llamado Amazon S3.

Para ello deberemos registrarnos en su página web en el enlace https://aws.amazon.com/es/. Este servicio nos ofrece una prueba gratuita de 12 meses. Pasado este periodo,

se deberá pagar pasadas unas limitaciones del servicio. Una vez registrados, deberemos crear un usuario en la sección llamada IAM, la cual se utiliza para gestionar los permisos sobre los recursos subidos. Siguiendo los pasos que nos indica Amazon, crearemos un usuario en el sistema y un grupo al que le daremos acceso total a los contenedores que crearemos a continuación y que, por último, añadiremos al usuario que acabamos de crear a él. En nuestro proyecto hemos llamado al usuario *django-YourStudies* y al grupo *django-s3-assets*.

Figura 6.2: Usuario creado en la sección IAM del servicio Amazon S3.

Una vez esto, Amazon nos proporcionará una contraseña, que deberemos guardar para añadirla posteriormente en el archivo *settings.py* de Django.

Por otro lado, crearemos un contenedor o *bucket* en el que se almacenarán todos los ficheros. Deberemos establecer un nombre para que Amazon proporcione un enlace único cada vez que se haga una petición a su sistema y en el apartado de administración le daremos acceso público con una política concreta para que nuestro proyecto Django pueda realizar peticiones sobre los ficheros o carpetas y no sean rechazadas. A continuación se muestran las políticas de uso compartido de recursos entre orígenes (CORS) que ha sido necesario definir en el contenedor:

Figura 6.3: Políticas de uso compartido de recursos entre orígenes (CORS) configuradas en el contenedor de nuestro proyecto.

En la parte del proyecto de Django, deberemos instalar en nuestro entorno virtual, mediante el programa *pip*, dos librerías llamadas *boto* y *django-storages*, las cuáles ofrecen utilizar un servicio de guardado de ficheros en la nube según la configuración que le proporciones. *Django-storages* es compatible con más servicios a parte del utilizado, como por ejemplo Azure.

Como antes se ha mencionado, para su configuración se ha modificado el fichero *settings.py* de la siguiente forma:

```
AWS_ACCESS_KEY_ID = os.getenv('AWS_ACCESS_KEY_ID')
AWS_SECRET_ACCESS_KEY = os.getenv('AWS_SECRET_ACCESS_KEY')
AWS_STORAGE_BUCKET_NAME = os.getenv('AWS_STORAGE_BUCKET_NAME')
AWS_S3_CUSTOM_DOMAIN = '%s.s3.amazonaws.com' % AWS_STORAGE_BUCKET_NAME

AWS_S3_OBJECT_PARAMETERS = {
    'CacheControl': 'max-age=86400',
}

AWS_LOCATION = 'static'
STATICFILES_STORAGE = 'storages.backends.s3boto3.S3Boto3Storage'
STATIC_URL = "https://%s/%s/" % (AWS_S3_CUSTOM_DOMAIN, AWS_LOCATION)

DEFAULT_FILE_STORAGE = 'YourStudies.storage_backends.MediaStorage'
```

Algoritmo 6.7: Configuración en el fichero settings.py para poder enviar utilizar los servicios de almacenamiento de Amazon Storage

Como se puede ver, se ha configurado la id y la contraseña de acceso del usuario que nos había proporcionado Amazon anteriormente. Además también ha sido necesario especificar el nombre del contenedor en el que se guardarán los recursos, y el enlace en el que Amazon recibe las peticiones del contenedor. También se ha definido la URL en la que se encuentran los archivos estáticos, que hemos llamado *static*.

Para los archivos no estáticos, que se suben a partir de ciertos eventos a la nube, como cuando un usuario quiere actualizar su foto de perfil, se debe definir una clase para ello, que hemos llamado *MediaStorage*, la cual contiene el nombre de la carpeta en la que se crearán estos ficheros, que en nuestro caso se llamará *media* y si se deben sobreescribir o no. Al igual que antes, se ha especificado en el fichero ".env" las variables necesarias para el correcto funcionamiento del contenedor.

6.4 Puesta en marcha: Heroku

Finalmente, para la puesta en marcha en producción de la aplicación se ha decidido utilizar la plataforma web Heroku, la cual nos permite desplegar una aplicación en la nube y que sea accesible por todo el mundo a través de su servicio. Al igual que en Amazon, deberemos registrarnos para tener acceso a sus servicios.

Una vez registrados, deberemos crear una aplicación en el servicio de Heroku siguiendo los pasos indicados por este. Como necesitamos que nuestra base de datos sea accesible, se ha añadido una extensión a la aplicación de Heroku de las muchas que ofrece llamada Heroku Postgres, la cual nos proporciona una base de datos PostgreSQL accesible mediante una URL.

Para preparar la aplicación en la parte de Django, deberemos crear un fichero llamado Procfile, que añadiremos a la raíz del proyecto. Deberemos instalar también una librería llamada *gunicorn*, que utilizará Heroku para ejecutar las ordenes que le indiquemos. Además, debemos indicarle a Heroku que librerías externas utiliza nuestro proyecto. Como estamos utilizando un entorno virtual de Python, podemos extraer las librerías instaladas mediante la orden *"pip freeze >requirements.txt"*, que nos creará un fichero llamado requirements con todas las librerías instaladas y sus versiones. Por último, deberemos especificar que versión de Python hemos utilizado en un fichero llamado *runtime.txt* si-

tuado en la raíz del proyecto.

```
web: gunicorn YourStudies.wsgi --log-file -
heroku run python manage.py migrate
```

Algoritmo 6.8: Visualización del fichero Procfile con las órdenes para desplegar la aplicación en Heroku

Por otra parte, deberemos hacer algunas modificaciones en el fichero settings.py. En concreto, deberemos cambiar los siguientes parámetros:

- **SECRET_KEY**. Como se subirá este código a producción, por seguridad Django nos indica que debemos ocultar la contraseña por lo que la cambiaremos por *"os.environ.get('SECRET_KEY')"*.
- **DEBUG**. Este atributo indica si la aplicación se está ejecutando en un entorno de desarrollo o no. Lo cambiaremos a False indicando que está en producción.
- **ALLOWED_HOSTS**. Se deberá añadir este atributo para que Django permita peticiones desde servidores y aplicaciones externas. Lo inicializaremos a `https://yourstudies.herokuapp.com/`, pues esta es la URL que nos ha provisto Heroku para que nuestra aplicación sea accesible a todo el mundo.
- **DATABASES**. Como nuestra base de datos a partir de ahora estará alojada con Heroku, se pondrá el siguiente código:

```
DATABASES = {
    'default': dj_database_url.config(
    default=config('DATABASE_URL')
    )
}
```

Algoritmo 6.9: Configuración de la base de datos en el fichero settings.py

- **LANGUAGE_CODE**. Deberemos cambiar el código del idioma para que Django sepa en que lenguaje se ha desarrollado el proyecto. Se especificará como 'es-ES'.
- **TIME_ZONE**. En algunos eventos utilizamos fechas y horas para limitar a los usuarios de realizar una acción. Debemos indicarle que utilizaremos la zona horaria de Madrid (España). Por ello, definiremos este atributo como: 'Europe/Madrid'.
- **DEFAULT_DOMAIN**. Aunque Django no contiene este atributo por defecto, se ha añadido ya que era necesario para los envíos de recuperación de contraseña desarrollados en este proyecto, ya que el usuario debía recibir un mensaje con el enlace de recuperación en su correo electrónico. En la fase de desarrollo se especificó como 'http://127.0.0.1:8000', pues el servidor funcionaba en local. Como ahora debe funcionar en la URL dada por Heroku, lo cambiaremos por `https://yourstudies.herokuapp.com/`.

Como durante el proceso hemos visto que usábamos variables de entorno locales obteniéndolas mediante *"os.getenv(«nombre»)"*, estas variables se deberán especificar a Heroku en el apartado *Config Vars* de nuestra aplicación Heroku:

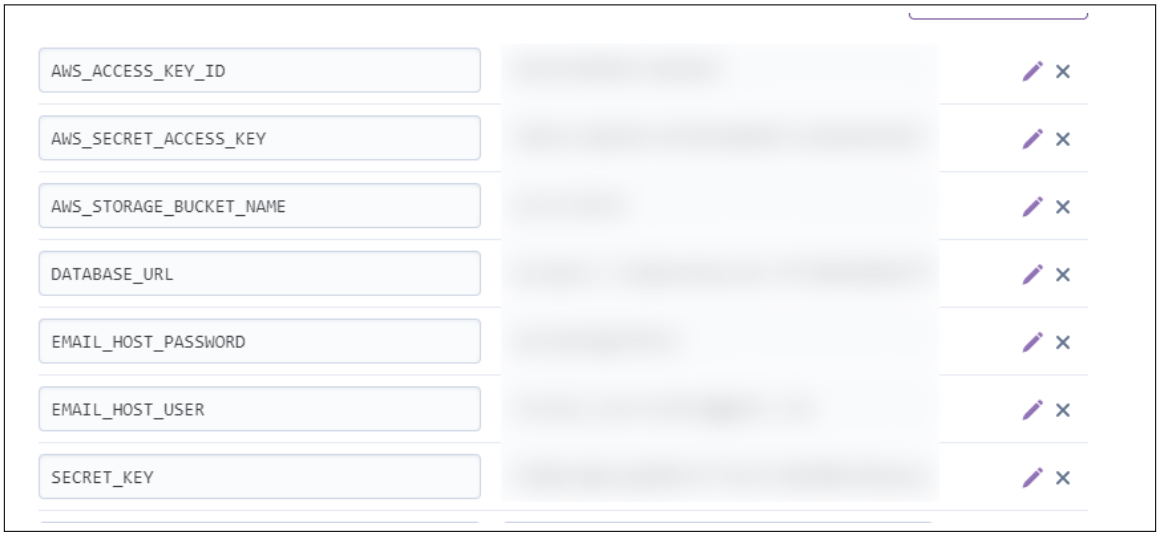


Figura 6.4: Variables de configuración creadas en Heroku.

Por último, una vez habiendo realizado todas estas configuraciones, solo queda el despliegue de nuestro sistema y código en el servidor de Heroku. Heroku permite desplegar una aplicación mediante una conexión de Github, por lo que solo deberemos aceptar los permisos que nos solicita sobre Github para que comience a desplegarla. Una vez desplegada, será accesible desde la URL `https://yourstudies.herokuapp.com/`

CAPÍTULO 7
Pruebas

Para comprobar que se cumplen las especificaciones y los objetivos establecidos al inicio de esta memoria una vez acabada la fase de desarrollo, se han creado diversos casos de prueba para verificar el correcto funcionamiento de la aplicación. Por otro lado también se mostrarán los resultados de los casos de uso presentados en el capítulo 3 para ilustrar el estado final de la aplicación.

7.1 Casos de prueba

Se han definido los casos de prueba de las funcionalidades que más validaciones tienen de la aplicación. Por ello, realizaremos unas acciones y especificaremos las entradas y las salidas esperadas de cada una de ellas. Estas entradas se probarán en la aplicación obteniendo una salida, que deberá coincidir con la salida esperada.

Nombre: Pruebas de inicio de sesión				**Identificador:** CP-01
Propósito: Verificar que el sistema comprueba los datos introducidos por el usuario y, de ser correctos, inicia sesión en el sistema.				
Acciones y salidas				
Id	**Acciones**	**Entradas**	**Salida esperada**	**Salida obtenida**
01	Iniciar sesión sin tener una cuenta registrada.	Usuario: monica Contraseña: contraseña	Mensaje indicando que las credenciales no son correctas y no inicia sesión.	Aparece el mensaje "Nombre de usuario o contraseña incorrectos" y no inicia sesión.
02	Iniciar sesión con credenciales correctas siendo administrador.	Usuario: admin Contraseña: admin	Inicia sesión y nos envía a la página de inicio del administrador.	Inicia sesión y nos envía correctamente a la página de inicio del administrador.
03	Iniciar sesión con credenciales correctas siendo profesor o alumno.	Usuario: morom Contraseña: contraseña	Inicia sesión y nos envía a la página de inicio de la aplicación.	Inicia sesión y nos envía correctamente a la página de inicio de la aplicación.
04	Cerrar sesión.	—	Cierra sesión y nos envía a la página inicial de la aplicación.	Cierra sesión y nos envía a la página inicial de la aplicación.

Tabla 7.1: Caso de prueba 1. Inicio de sesión.

Nombre: Pruebas de administración				**Identificador:** CP-02
Propósito: Verificar que el sistema crea correctamente los objetos lógicos que existen en la aplicación. Nota: Se han realizado pruebas similares con los objetos lógicos Profesores, Asignaturas, Categorías, Unidades y Secciones, pero que no se muestran por su similitud.				
Acciones y salidas				
Id	**Acciones**	**Entradas**	**Salida esperada**	**Salida obtenida**
01	Crear un alumno con datos correctos.	Usuario: eva E-mail: eva@alumno.com Nombre: Eva Apellidos: Sanchez Calatayud	Envía al usuario al listado de alumnos donde se visualiza al nuevo alumno.	Envía al usuario al listado de alumnos y se puede ver al nuevo alumno en la tabla.
02	Crear un alumno con el nombre de usuario y/o el correo electrónico repetido.	Usuario: eva E-mail: eva@alumno.com Nombre: Eva María Apellidos: Gutierrez Sánchez	Aparece un error informando que ese correo electrónico ya está registrado y/o que el nombre de usuario está registrado.	Aparece un error informando que ese correo electrónico ya está registrado y/o que el nombre de usuario está registrado.
03	Editar un alumno con datos correctos.	Usuario: eva E-mail: evamaria@alumno.com Nombre: Eva María Apellidos: Gutierrez Sánchez	Envía al usuario al listado de alumnos donde se visualiza al nuevo alumno.	Envía al usuario al listado de alumnos y se puede ver al nuevo alumno en la tabla.
03	Editar un alumno con el nombre de usuario y/o el correo electrónico repetido.	Usuario: eva E-mail: eva@alumno.com Nombre: Eva María Apellidos: Gutierrez Sánchez	Aparece un error informando que ese correo electrónico ya está registrado y/o que el nombre de usuario está registrado.	Aparece un error informando que ese correo electrónico ya está registrado y/o que el nombre de usuario está registrado.
04	Borrar un alumno	—	Debe aparecer una alerta validando si realmente se desea borrar al alumno. Si se selecciona el botón “Borrar”, el alumno debe ser eliminado del sistema. Si es que no, no debe realizarse nada.	Aparece una alerta con la pregunta "¿Seguro que deseas borrar este alumno?". Si se cierra la alerta, no realiza ninguna acción. Si se selecciona el botón “Borrar”, se borra el registro en la tabla de alumnos.
05	Ver un alumno	—	El usuario debe ser redirigido al perfil del alumno seleccionado	Se redirige al perfil del alumno seleccionado en el que se pueden ver sus características.
06	Buscar un alumno con resultados	Búsqueda: ana	Debe aparecer en la tabla todos los alumnos que su nombre o su nombre de su usuario comience por ana.	Aparece en el listado los alumnos que comienzan con ana.
07	Buscar un alumno sin resultados	Búsqueda: anamaría	Debe aparecer un mensaje informando que no se ha encontrado ningún alumno.	Aparece en pantalla el mensaje "No se ha encontrado ningún alumno en la búsqueda."

Tabla 7.2: Caso de prueba 2. Creación de un usuario.

Nombre: Pruebas del repositorio de ficheros				**Identificador:** CP-03
Propósito: Verificar que el sistema crea y elimina correctamente ficheros del repositorio personal.				
Acciones y salidas				
Id	**Acciones**	**Entradas**	**Salida esperada**	**Salida obtenida**
01	Subir fichero a la aplicación.	—	Debe aparecer el fichero subido en la lista de ficheros, junto a la fecha de subida y el botón de borrado.	Al subir el fichero se crea una nueva fila en la tabla de ficheros con la fecha de subida y el botón de borrado.
02	Borrar fichero.	—	Debe aparecer una alerta que verifique si se quiere eliminar el fichero o no. Si se pincha en el botón "Borrar", eliminará la fila del fichero correspondiente. Si se cancela, el sistema no deberá realizar ninguna acción.	Aparece una alerta con el mensaje "¿Estás seguro que quieres borrar este fichero?". Si se elige el botón "Cancelar", el sistema no realiza ninguna acción. Si se pincha en el botón "Borrar", se borra la fila del fichero y se elimina del sistema.

Tabla 7.3: Caso de prueba 3. Repositorio de ficheros.

Nombre: Entrega de tareas por parte del alumnado.				**Identificador:** CP-04
Propósito: Verificar que el sistema crea una tarea entregada y se puede editar sobre ella.				
Acciones y salidas				
Id	**Acciones**	**Entradas**	**Salida esperada**	**Salida obtenida**
01	Entregar una tarea con datos correctos dentro del tiempo dado.	Respuesta: La respuesta a la tarea. Ficheros adjuntos: Fichero .jpg	El sistema debe permitir guardar los datos y redireccionar a la vista del contenido de la asignatura.	El sistema debe permitir guardar los datos y redireccionar a la vista del contenido de la asignatura.
02	Entregar una tarea con datos correctos pasado el tiempo de entrega.	Respuesta: La respuesta a la tarea. Ficheros adjuntos: Fichero .jpg	El sistema debe indicar que no se puede entregar una tarea porque ha finalizado el tiempo.	El sistema avisa al alumno con un mensaje indicando que no se puede entregar porque ha finalizado el tiempo dado.
03	Acceder a una tarea entregada pasado el tiempo de entrega.	—	El sistema debe visualizar los datos entregados por el alumno, sin que aparezca el botón de guardar.	Se visualiza la tarea con los datos guardados por el alumno, sin posibilidad de guardar.
04	Acceder a una tarea sin entregar pasado el tiempo de entrega.	—	En la tarea debe aparecer un 0 como puntuación y un mensaje avisando que esta tarea ya no se puede entregar pues el tiempo de entrega ha acabado.	En la tarea aparece un 0 como puntuación y una alerta avisando que esta tarea ya no se puede entregar porque el tiempo de entrega ha acabado.

Tabla 7.4: Caso de prueba 4. Entregar tarea.

Nombre: Pruebas de creación y corrección de tareas por parte del profesorado.				**Identificador:** CP-04
Propósito: Verificar que el sistema crea y permite corregir correctamente las tareas al profesorado.				
Acciones y salidas				
Id	**Acciones**	**Entradas**	**Salida esperada**	**Salida obtenida**
01	Crear una tarea con datos correctos.	Nombre: Tarea 1 Contenido: Para probar las tareas Fecha finalización: 31/07/2021 21:58	El sistema debe redirigir al profesor a la página principal de la asignatura donde podrá ver la tarea creada.	El sistema permite guardar y redirige a la página principal de la asignatura.
02	Crear tarea con fecha incorrecta.	Nombre: Tarea 2 Contenido: Para probar las tareas Fecha finalización: 31/06/2020 21:58	El sistema debe indicar que no se puede crear una tarea con fecha anterior a hoy.	Aparece un error en pantalla indicando que no se puede crear una tarea con fecha anterior a hoy.
03	Corrección de una tarea con datos correctos y después de que haya finalizado el tiempo.	Puntuación: 5 Comentario: Se debe mejorar la caligrafía.	El sistema debe guardar correctamente y redirigir a la visualización de la tarea.	El sistema almacena la puntuación y el comentario dado y redirige a la vista de la tarea.
04	Corrección de una tarea antes de que haya finalizado el tiempo.	Puntuación: 5 Comentario: Se debe mejorar la caligrafía.	El sistema debe alertar al usuario de que no se puede corregir una tarea hasta que la fecha de finalización sea posterior a la de hoy.	Aparece un mensaje avisando de que no se puede corregir una tarea porque la fecha de finalización es posterior a la actual.

Tabla 7.5: Caso de prueba 5. Creación y corrección de tareas.

Como se puede observar, todas las salidas obtenidas coinciden con las salidas esperadas, por lo que podemos concluir que las funcionalidades de nuestra aplicación funcionan correctamente.

7.2 Resultados

Por último, se mostrarán los resultados finales que hemos obtenido, enseñando las pantallas de los casos de usos principales de nuestra aplicación:

7.2.1. Caso de uso 1: Iniciar sesión

Como se puede observar en la figura 7.1, el usuario iniciará sesión mediante su nombre de usuario y su contraseña. Si el usuario es un administrador, será redirigido a la pantalla de administración, mientras que si es un profesor o un alumno, será redirigido a la página principal de la aplicación, donde podrá ver sus asignaturas. También desde esta

pantalla cualquier usuario puede restaurar la contraseña desde el enlace "¿Has olvidado la contraseña?", que le redirigirá a un formulario donde escribir su correo electrónico como se verá más adelante.

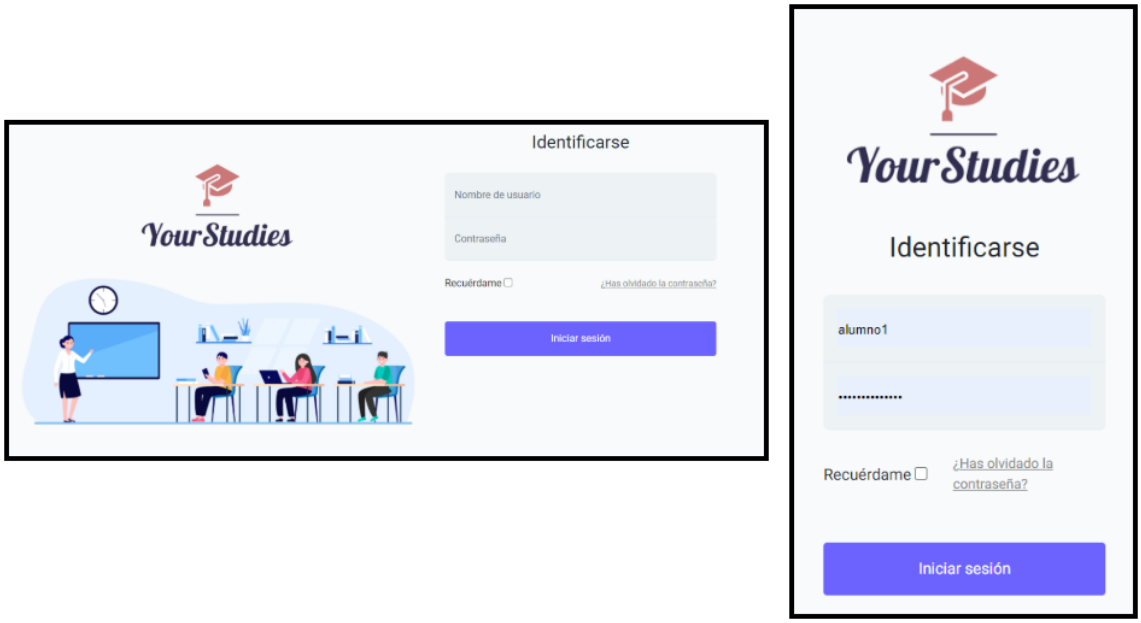


Figura 7.1: Caso de uso 1 - Iniciar sesión.

7.2.2. Caso de uso 2: Ver perfil

Tanto los profesores como los alumnos pueden ver su perfil accediendo a él desde la página principal. Desde esa página también pueden editar su contraseña y editar su perfil. En esta figura concretamente aparece el título "Alumno" seguido del nombre del alumno. Cuando sea un profesor, esta palabra se sustituirá por "Profesor".

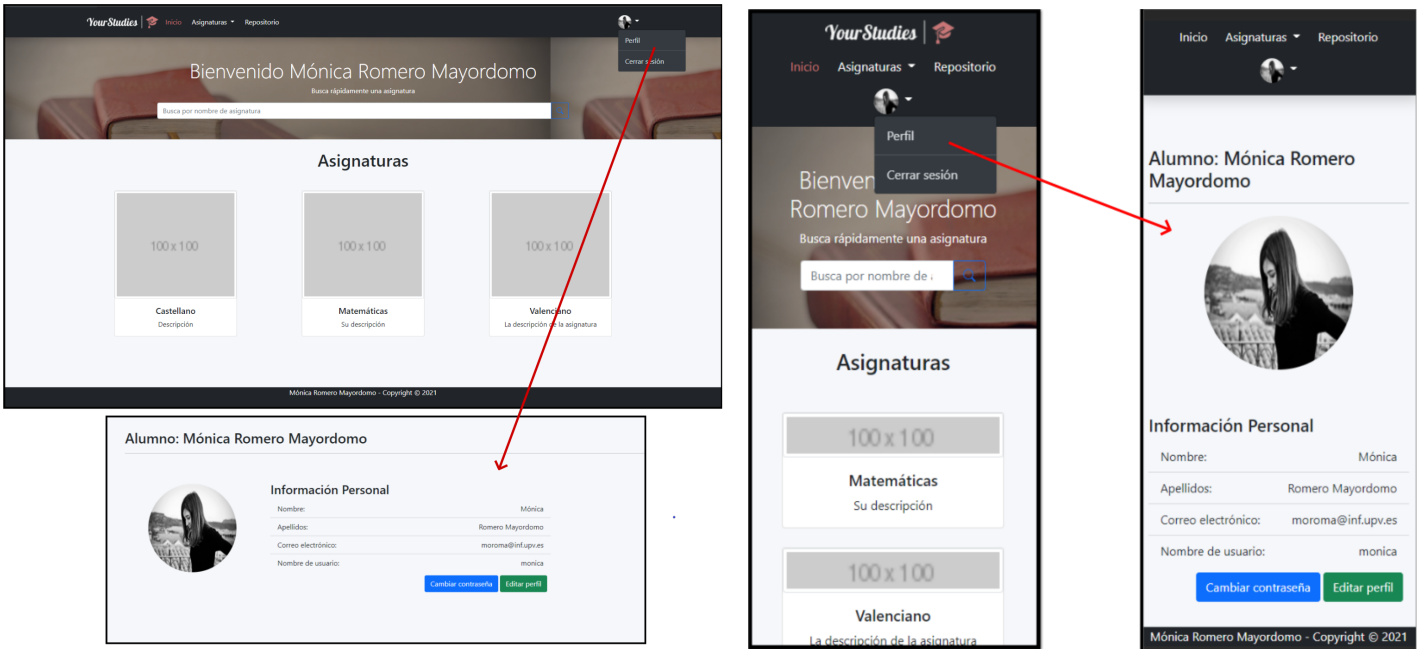


Figura 7.2: Caso de uso 2 - Ver perfil.

7.2.3. Caso de uso 3: Editar perfil

En la edición del perfil, el alumnado y profesorado solo podrán modificar su nombre de correo electrónico y su avatar, ya que el resto de campos son rellenados en su creación, y no se deben modificar ya que son parte de la administración del sistema. Después de haber modificado lo necesario, deberá pincharse el botón "Guardar", y se le redirigirá a la página de "Ver perfil"mostrada en la figura 7.2.

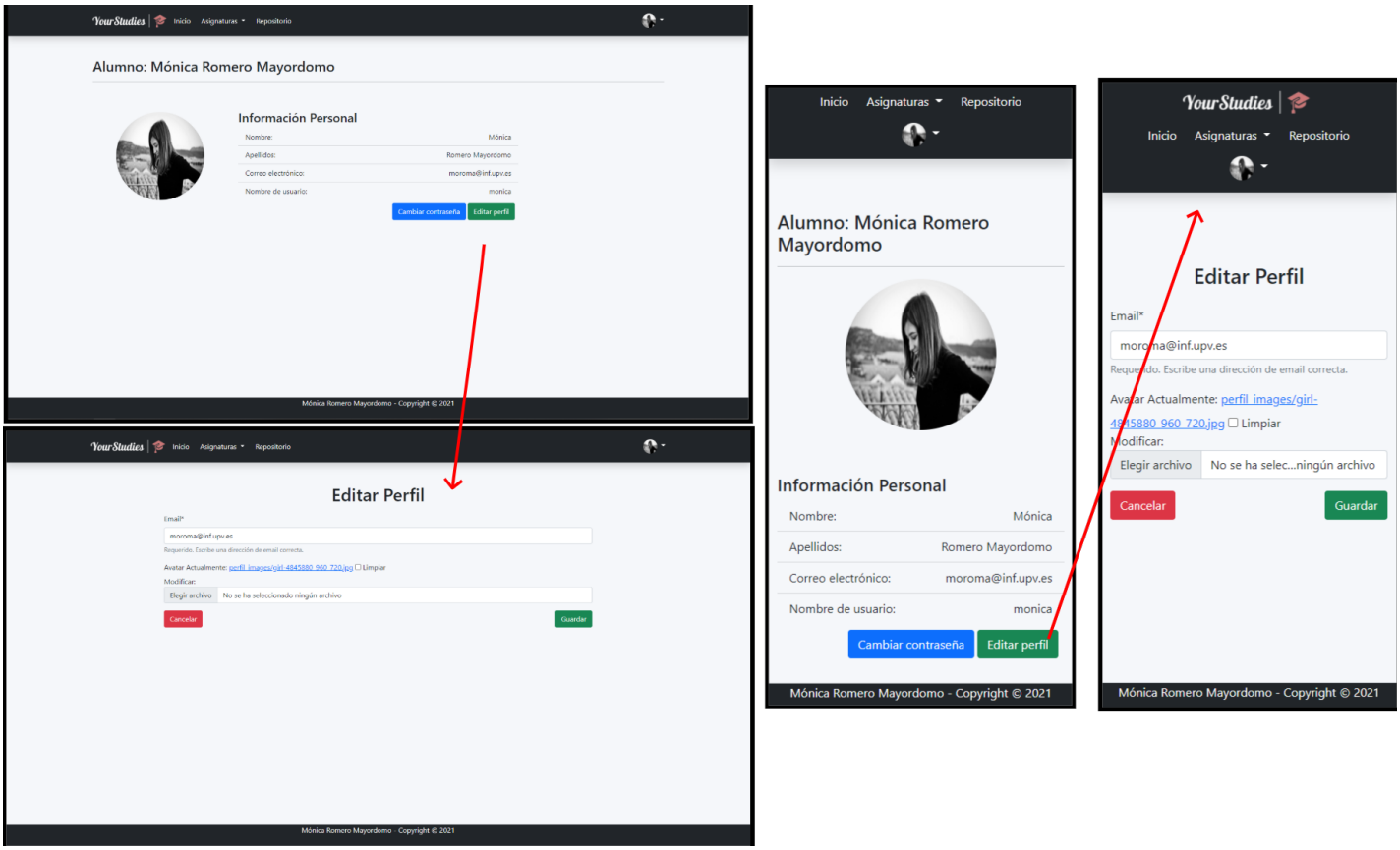


Figura 7.3: Caso de uso 3 - Editar perfil.

7.2.4. Caso de uso 4: Listar objetos

Una vez iniciada sesión, el administrador podrá ver los listados de los diversos objetos lógicos que están registrados en la aplicación. Estos objetos son alumnos, profesores, asignaturas y categorías. Para acceder a los listados deberá seleccionar el apartado desde el menú de navegación que aparece a la izquierda, o desde la página principal de administración pinchando en una de las opciones que aparecen en los recuadros. Además, desde los listados también se puede realizar una búsqueda de los objetos desde la barra que aparece arriba de la página, para que le resulte más sencillo al usuario encontrar un objeto en particular.

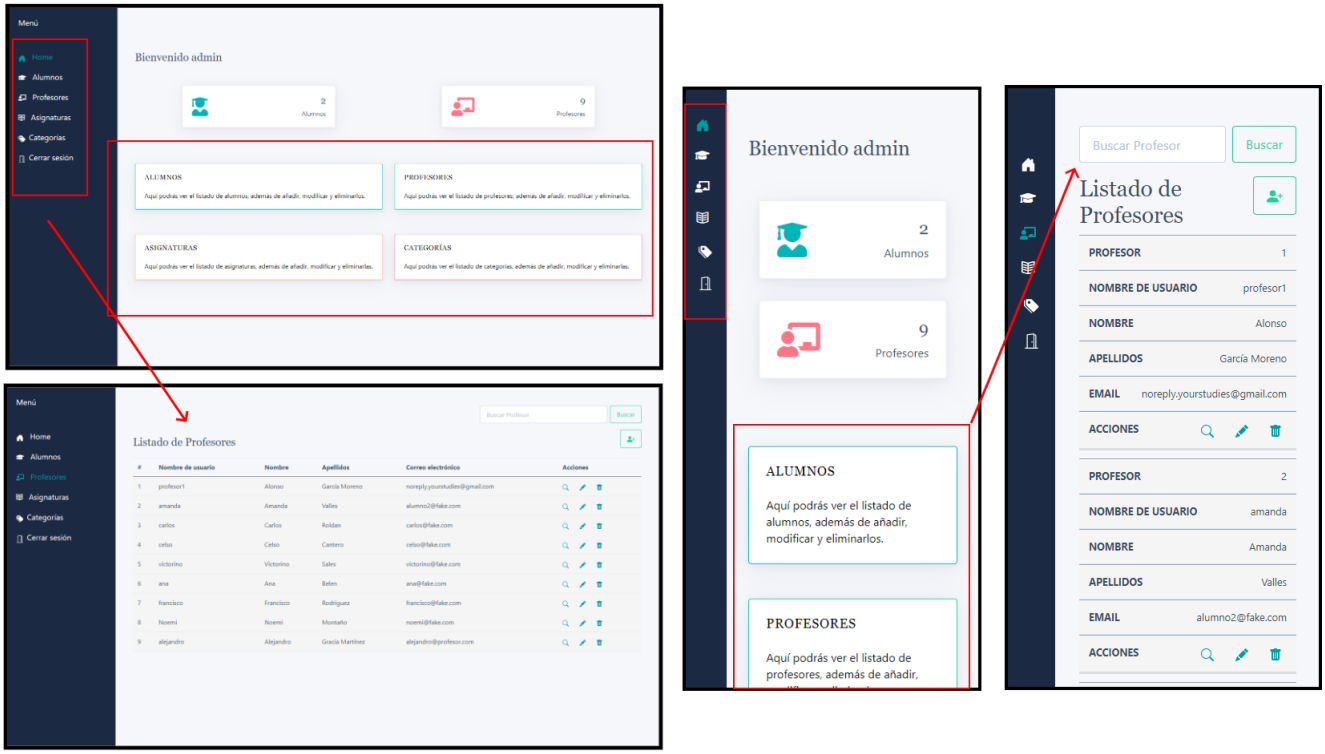


Figura 7.4: Caso de uso 4 - Listar objetos.

7.2.5. Casos de uso 5 y 6: Añadir y editar objetos

Para añadir un objeto, se deberá pinchar en el botón "+" que aparece en la figura 7.4 justo abajo de la barra de búsqueda. El sistema redirigirá al usuario a un formulario, en el que se deberán rellenar los campos y darle al botón "Guardar". Si el usuario cancela, la aplicación le redirigirá al listado de ese objeto. El mismo procedimiento se aplica para la edición de un objeto, con la diferencia de que se deberá pinchar al botón que se asemeja a un lápiz.

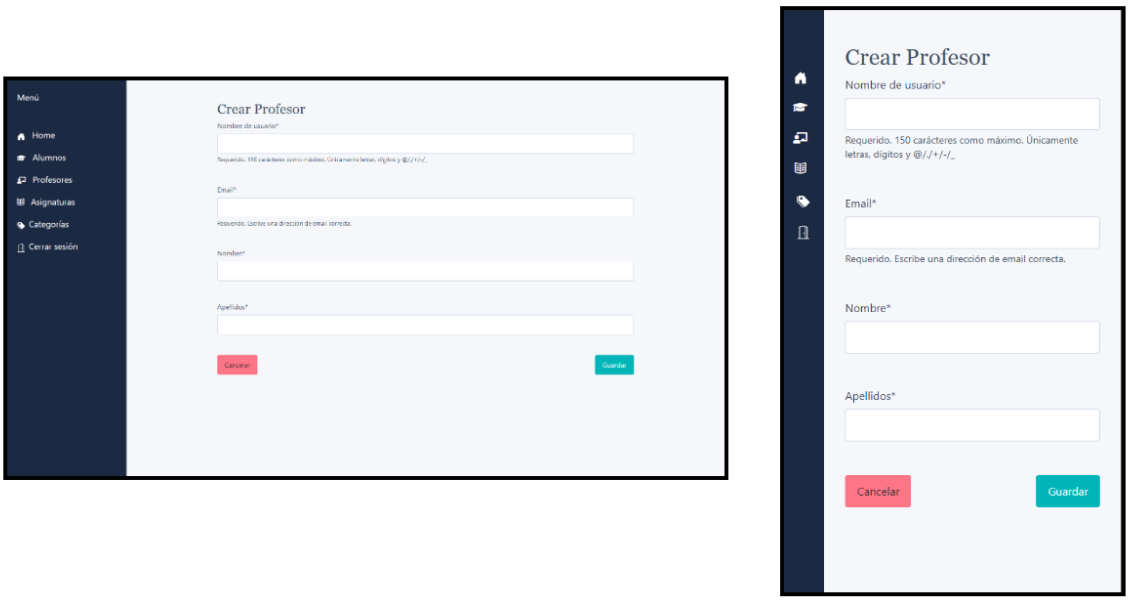


Figura 7.5: Casos de uso 5 y 6 - Añadir y editar objetos.

7.2.6. Caso de uso 7: Borrar un objeto

Para borrar un objeto en particular, se deberá pinchar en el botón que se asemeja a una papelera. El sistema mostrará una alerta como la que se ve en la figura 7.6, alertándolo de si desea borrar totalmente ese objeto. Si el usuario cancela, no se realizará ninguna opción, mientras que si acepta el borrado, se recargará la página para actualizar el listado correspondiente.

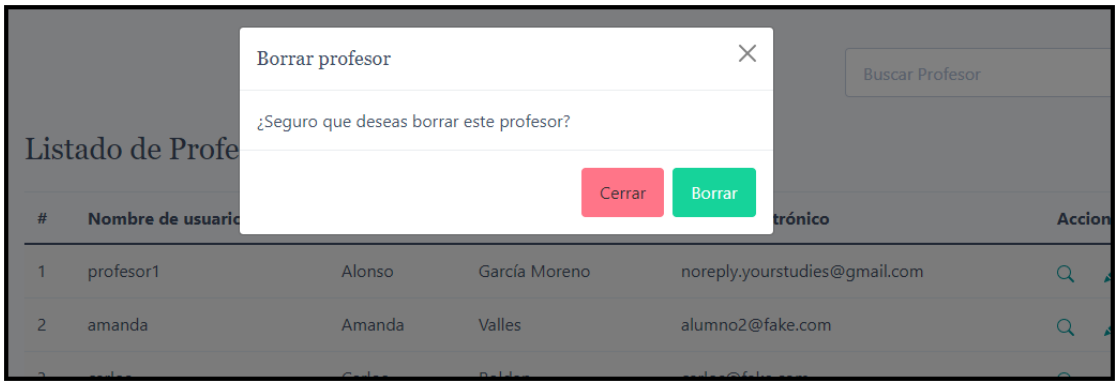


Figura 7.6: Caso de uso 7 - Borrar objetos.

7.2.7. Casos de uso 8 y 9: Visualizar y buscar asignaturas asignadas

Cuando los profesores y los alumnos inicien sesión, serán redirigidos a la página principal, en la que podrán visualizar las asignaturas, junto a su descripción e imagen, en las que imparten clases o están matriculados. También podrán verlas desde la barra de navegación principal en cualquier momento, ya que también se listan en forma de desplegable en la opción "Asignaturas". Además de esto, se pueden buscar por nombre mediante la barra de búsqueda central, para poder encontrar rápidamente cualquier asignatura.

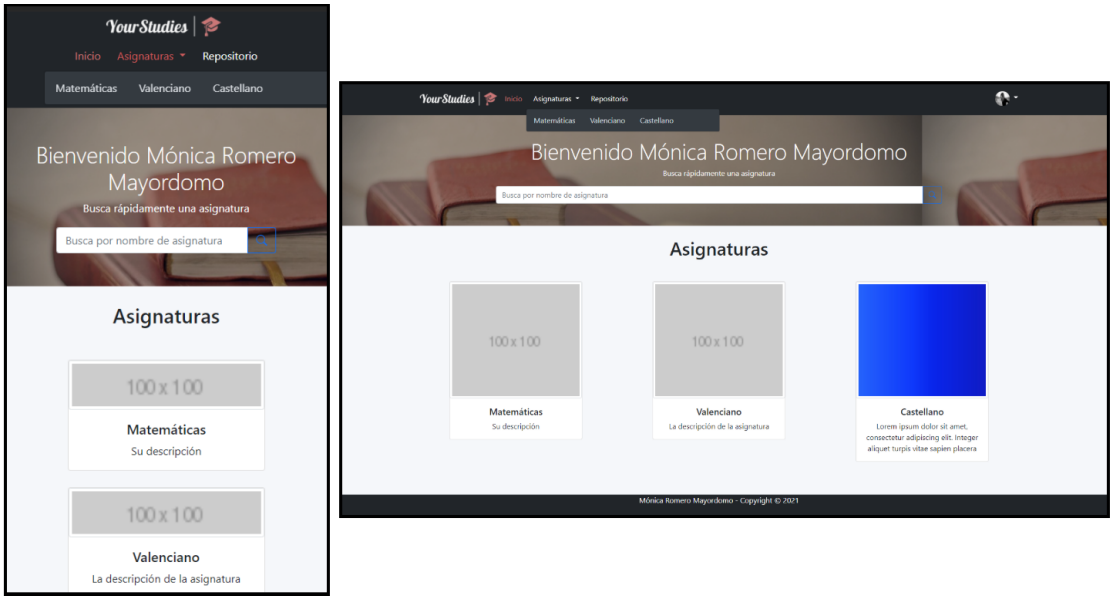


Figura 7.7: Casos de uso 8 y 9 - Visualizar y buscar asignaturas.

7.2.8. Caso de uso 10: Subir ficheros a un repositorio personal

El alumnado y el profesorado puede acceder a la sección del repositorio de ficheros desde el menú de navegación principal, desde el apartado "Repositorio". Aquí podrá ver sus ficheros subidos anteriormente en una tabla, junto a la fecha de subida y el botón de borrado en forma de papelera. Para subir uno o varios ficheros nuevos, podrá hacerlo arrastrando al apartado "Subir fichero" o pinchando en el desplegable que ofrece el sistema operativo para elegir fichero.

Para que el usuario sepa cuando la aplicación ha acabado de subir los archivos, aparecerá una barra de carga que se rellenará automáticamente hasta el 100 %. Por otra parte, si el usuario desea borrar un fichero subido y pincha en el botón de borrado, le aparecerá un aviso como la figura 7.6, preguntándole si desea borrar el fichero definitivamente.

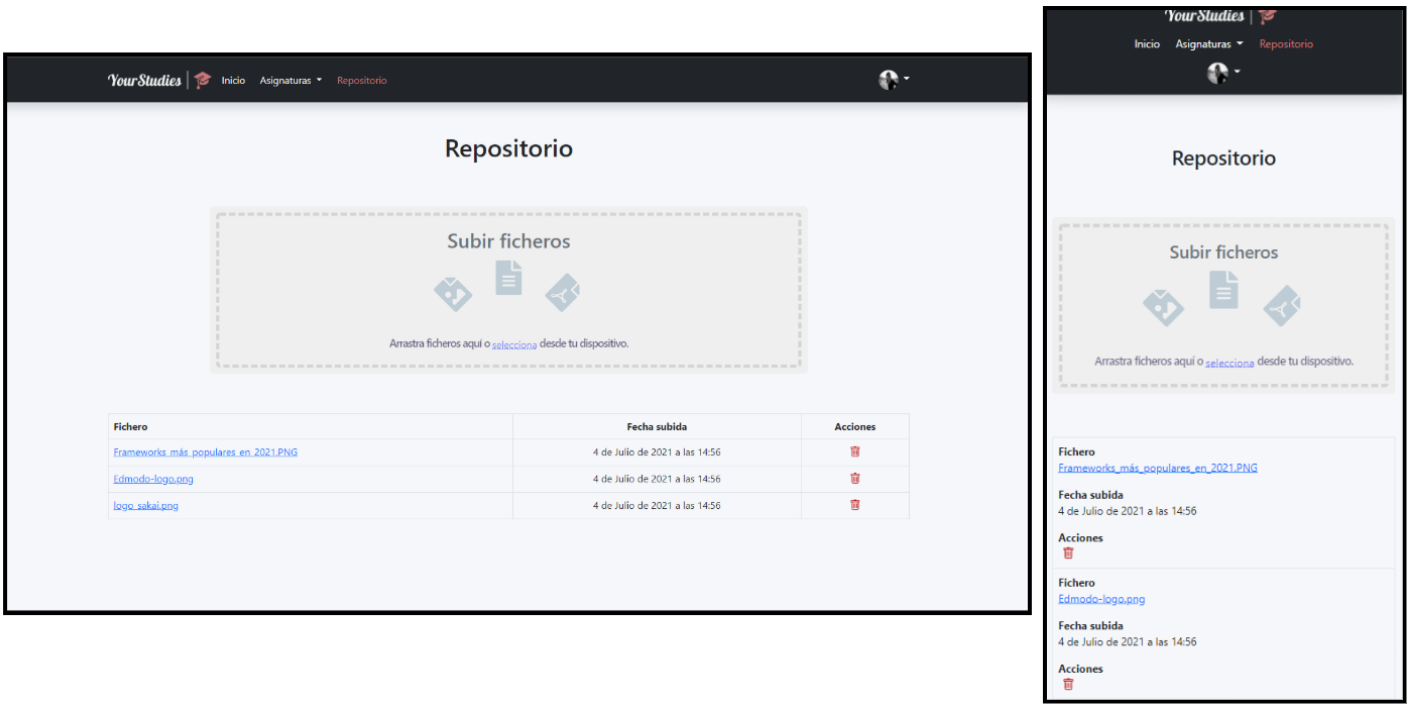


Figura 7.8: Caso de uso 10 - Repositorio de ficheros.

7.2.9. Caso de uso 11: Visualizar contenido de una asignatura

En el menú de las asignaturas, podemos ver las unidades que la contienen en forma de acordeones donde, además, se muestran las secciones y las tareas de cada unidad. También podremos añadir nuevas unidades, secciones y tareas como navegar por los diferentes apartados de la asignatura con el menú de navegación izquierdo, en pantallas grandes, o con el desplegable de navegación en pantallas pequeñas.

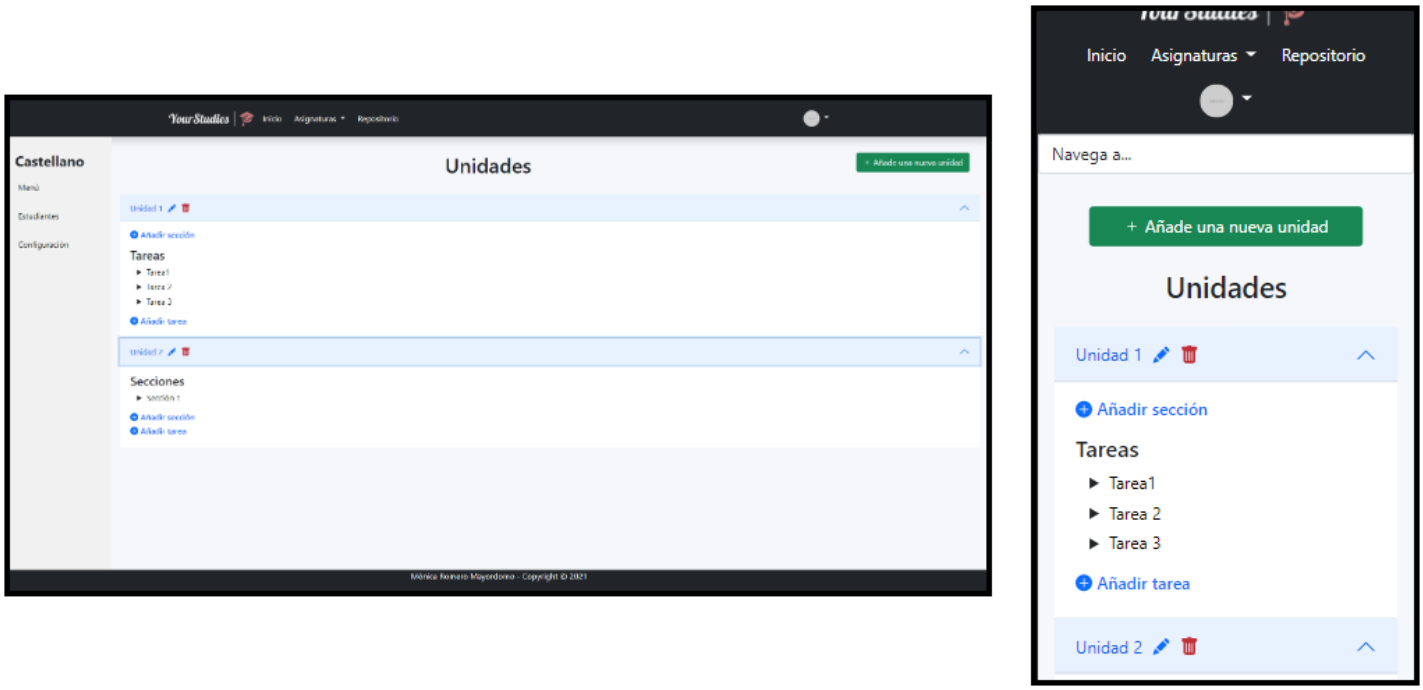


Figura 7.9: Caso de uso 11 - Visualizar contenido de una asignatura.

7.2.10. Caso de uso 12: Editar configuración de la asignatura

Para acceder al apartado de configuración de la asignatura, los profesores deberán seleccionar la opción "Configuración" en el menú de navegación izquierdo en la versión de ordenador, mientras que en la versión de móvil deberán elegirlo desde el desplegable debajo del menú de navegación principal.

En esta sección podemos ver el contenido de la asignatura, los profesores que la imparten, su descripción, el idioma en el que se imparte y la imagen relacionada a él. Solamente podrán editar los profesores, por lo que a los alumnos no les aparecerá el botón de edición.

Para editar, los profesores deberán pinchar en el lápiz azul que aparece al lado del título "Características". La aplicación les redirigirá a un formulario en el que podrán editar la descripción, el contenido y la imagen relacionada a la asignatura.

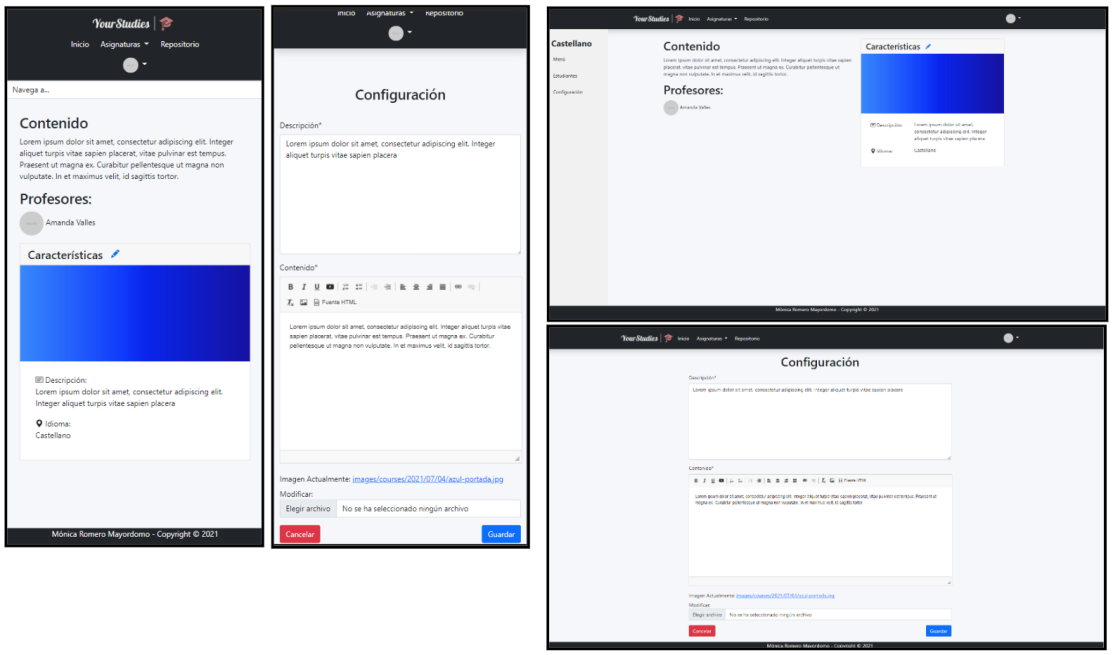


Figura 7.10: Caso de uso 12 - Editar configuración de la asignatura.

7.2.11. Caso de uso 13: Listar alumnado de una asignatura

A esta sección solamente podrán acceder el profesorado de cada asignatura. Al igual que el caso de uso anterior, deberán acceder mediante la barra o el desplegable de navegación, según el dispositivo, pinchando en la sección "Alumnos". Podrán visualizar un listado completo de todos los alumnos matriculados en la asignatura, así como sus fotografías.

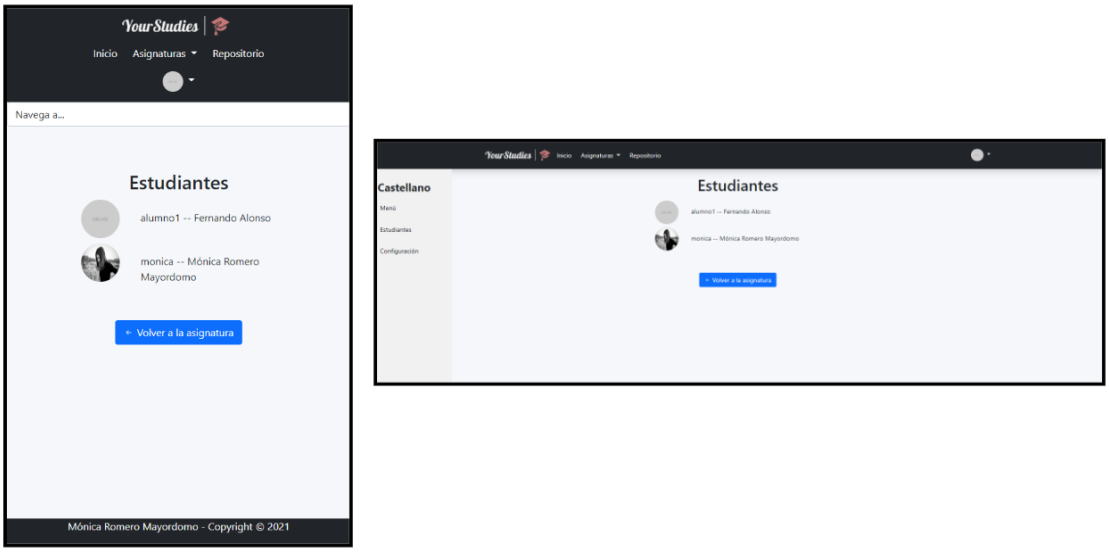


Figura 7.11: Caso de uso 13 - Listar alumnado de una asignatura.

7.2.12. Caso de uso 14: Corregir tarea

El profesorado tiene la posibilidad de corregir las tareas que le entregan sus alumnos telemáticamente. Para ello, en la página principal del contenido de la asignatura, deberá escoger que tarea desea corregir pinchando en ella. Una vez allí, deberá pinchar en el estudiante al que desea puntuar y la aplicación le redirigirá a un formulario en el que aparecerán la respuesta del alumno y los ficheros subidos por él. El profesor podrá dar una puntuación y un comentario adicional a la tarea, pero no podrá guardar hasta que la tarea haya concluido.

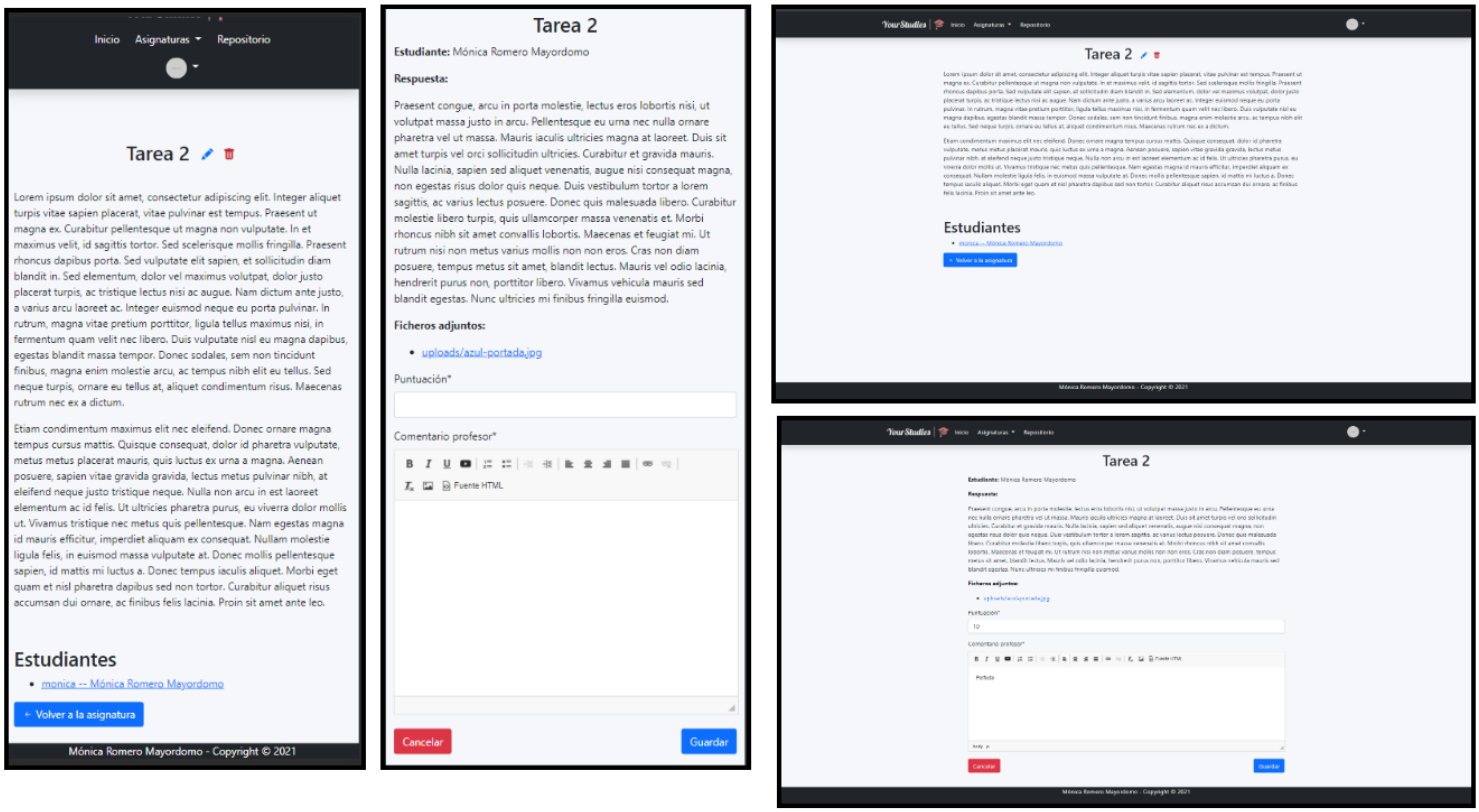


Figura 7.12: Caso de uso 14 - Corregir tarea.

7.2.13. Caso de uso 15: Entregar tarea

Para poder entregarle ficheros o una redacción al profesor, los alumnos deberán rellenar un formulario, en el que deberán escribir la respuesta e incluir ficheros en el caso en que sea necesario. Además pueden revisar la tarea las veces que sea necesario hasta que finalice el tiempo estipulado para completar la tarea. Una vez haya concluido el tiempo, deberán esperar a la puntuación y al comentario del profesor.

Por otra parte, si un alumno no entrega la tarea a tiempo, no le aparecerá este formulario cuando acceda a la tarea, si no solamente la descripción de la tarea y la puntuación con un 0.

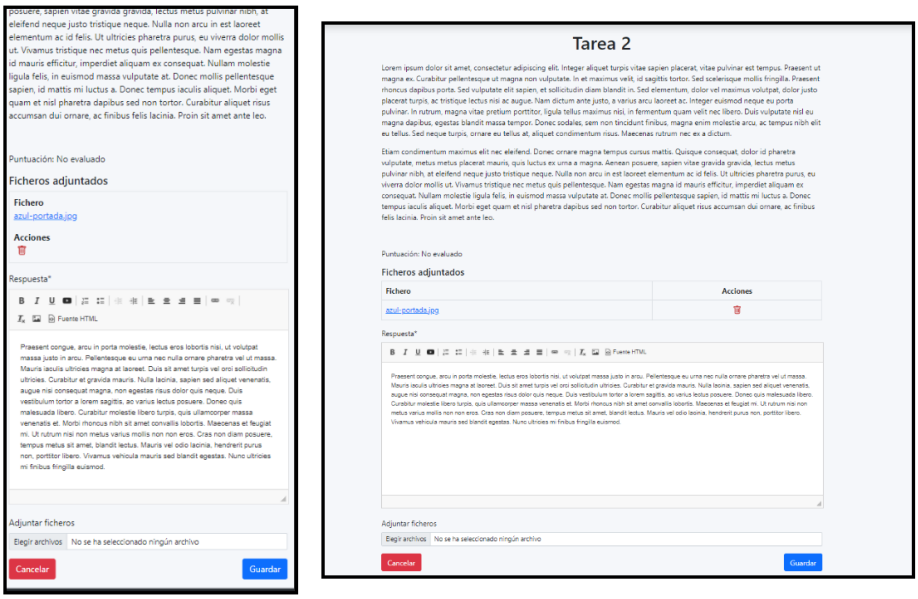


Figura 7.13: Caso de uso 15 - Entregar tarea.

7.2.14. Caso de uso 16: Visualizar calificaciones

Dentro de una asignatura, los alumnos pueden acceder a una sección en la que podrán ver sus calificaciones de cada tarea, además de una gráfica resumen con la relación tarea-calificación.

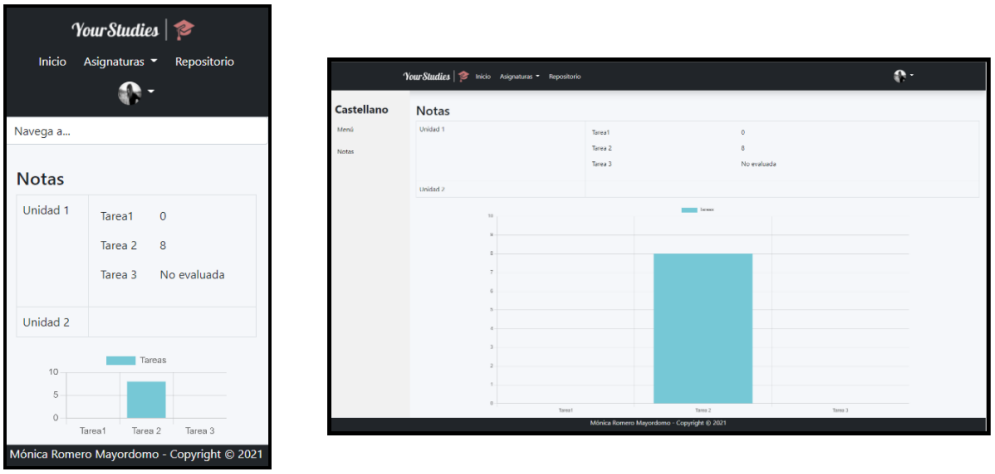


Figura 7.14: Caso de uso 16 - Visualizar calificaciones.

7.2.15. Caso de uso 17: Recuperar contraseña

Los usuarios de la aplicación deben ser capaces de recuperar su contraseña en cualquier momento por si la pierden. Escribiendo el correo electrónico que está asociado a sus cuentas, la aplicación verificará si ese usuario se encuentra registrado o no. En el caso en el que lo esté, enviará un correo con instrucciones para recuperar la contraseña. Contendrá un enlace, que llevará al usuario a un formulario para que introduzca su nueva contraseña.

Figura 7.15: Caso de uso 17 - Recuperar contraseña. Primera parte.

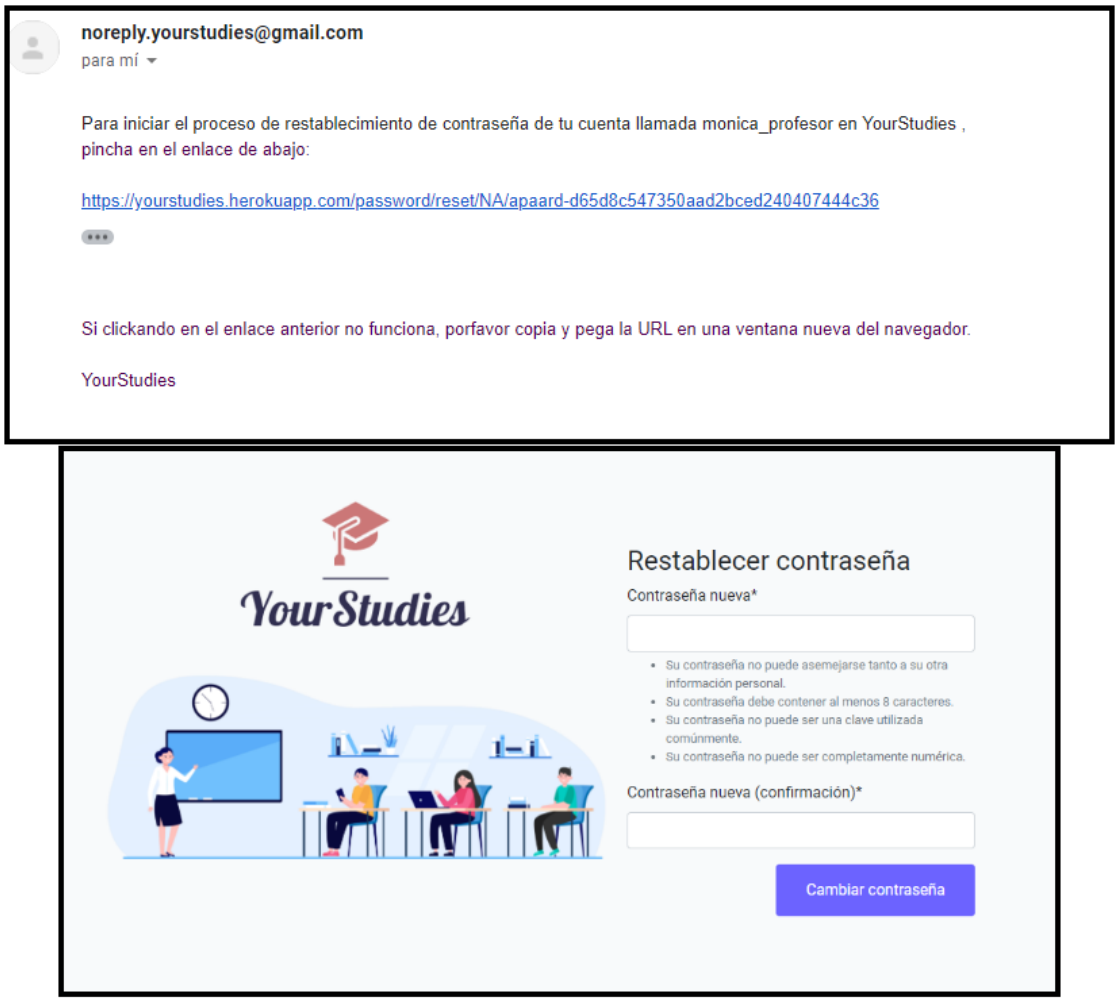


Figura 7.16: Caso de uso 17 - Recuperar contraseña. Segunda parte.

CAPÍTULO 8
Conclusiones

A lo largo de este trabajo hemos visto cómo diseñar e implementar una aplicación web con el objetivo de facilitar la distribución de recursos entre profesores y alumnos para una correcta organización de tiempo y evitar la pérdida de ficheros. Para llevarlo a cabo se han afrontado distintos retos durante el desarrollo con el fin de conocer y aprender nuevas tecnologías y reforzar los conocimientos enseñados en el grado.

En primer lugar, fue un gran reto elegir un lenguaje y un patrón de diseño que no habían sido vistos en la carrera, pues entender los controles de flujo del patrón MVT es complicado por primera vez, lo cuál hizo que el tiempo para iniciar el desarrollo del proyecto se hiciese bastante largo. Además, la falta de experiencia en la organización de un proyecto en Django y la multitud de ficheros autogenerados al crear el proyecto generó confusión al comienzo. A pesar de todos estos problemas, este trabajo ha aportado grandes conocimientos, pues fue muy interesante el uso de ORM para realizar inserciones y borrado de filas en las tablas en la base de datos como si fueran objetos en Python en vez de tener que hacer consultas SQL.

Por otra parte, el uso de CKEditor supuso bastantes problemas pues, a pesar de que se instaló mediante una librería de Django, la configuración e instalación de extensiones de este editor fue confusa, y no generaba ningún tipo de error cuando fallaba, por lo que el proceso de depuración fue costoso.

El apartado de despliegue fue realmente enriquecedor ya que el uso de un servicio externo en el que almacenar ficheros estáticos y dinámicos hizo que se aprendiese a manejar diversas políticas de permisos sobre los ficheros de diferentes usuarios. Además, fue realmente satisfactorio poder acceder desde cualquier dispositivo a la aplicación desarrollada mediante la URL proporcionada por Heroku una vez había acabado el despliegue.

Consecuentemente, se han cumplido los objetivos propuestos en el apartado 3.1.3. Se ha recorrido la situación actual del mercado observando otras aplicaciones similares a la nuestra que más se utilizan actualmente, además de realizar una investigación sobre las tecnologías más utilizadas y un proceso de elección de estas para determinar cuál se adaptaba más a nuestras necesidades. Como resultado final, se ha implementado una aplicación en Django desplegada en la nube que ayudará a la entrega de tareas por parte del alumnado y a la distribución de recursos por parte del profesorado.

Para concluir, este proyecto ha sido muy gratificante a nivel técnico pues se han descubierto y aprendido muchas tecnologías alternativas a las vistas en el grado, lo que nos ha aportado gran cantidad de conocimientos nuevos.

8.1 Relación del trabajo con los estudios cursados

A lo largo del grado cursado en la Universitat Politècnica de València se han visto temas que han influido en gran medida en el desarrollo de este trabajo, por lo que cabe destacar las diferentes asignaturas que más han influido en este proyecto.

En primer lugar, para la especificación de los casos de uso y del diagrama de clase se han utilizado los conocimientos aprendidos en la asignatura Ingeniería de Software, además de las bases de la estructura que debe tener un proyecto internamente para que esté correctamente modularizado.

Asimismo, para el desarrollo de las páginas webs que se utilizan como parte visual de la aplicación se ha hecho uso de los conocimientos de HTML, CSS, Bootstrap y JavaScript aprendidos en la asignatura Desarrollo Web de la rama Tecnologías de la información. Además, para que el diseño de la aplicación fuese sencilla de usar para los posibles usuarios de la aplicación se utilizaron los principios de Nielsen aprendidos en la asignatura Desarrollo centrado en el usuario.

En relación al diseño del diagrama de la base de datos, se llevó a cabo utilizando los conocimientos aprendidos en las asignaturas Base de datos y Tecnología de base de datos. Esta última, al igual que Desarrollo Web, pertenece al bloque Tecnologías de La Información. A pesar de que la base de datos no era demasiado compleja, para diseñar algunas relaciones hubo que repasar algunos conceptos dados en estas asignaturas.

Por último, cabe destacar algunas asignaturas del primer y segundo curso, como son Introducción a la informática y a la programación, Programación y Estructuras de datos y algoritmo, que sentaron las bases de la programación y la capacidad de buscar soluciones por uno mismo, lo que ha sido clave para el desarrollo de este proyecto.

CAPÍTULO 9
Trabajos futuros

La primera versión de YourStudies cumple con los objetivos principales que fueron planteados al principio, sin embargo algunos objetivos secundarios quedaron sin desarrollar por no ser tan importantes para esta primera versión. Por ello, a continuación se nombran algunas de las futuras funcionalidades u objetivos para futuras versiones:

- **Creación, edición y borrado de exámenes.** En nuestra aplicación solo es posible evaluar al alumno mediante tareas dándoles una puntuación. En una futura actualización sería recomendable añadir la opción de poder crear exámenes, ya sean tipo test o de respuesta abierta, para que el profesor sea capaz de tener más formas de evaluación o poder dar clases a distancia.
- **Calendario de eventos futuros en cada asignatura.** Una función muy útil y que fue utilizada por los profesores de la asignatura del grado Administración de sistemas, fue la realización de un calendario en el que se añadían la programación de las futuras clases, entregas y exámenes que se iban a realizar en la asignatura, lo que favorecía en gran medida la organización del tiempo del alumnado y evitaba posibles olvidos de los días en que había una evaluación.
- **Chat entre usuarios y un foro**. Para poder llevar a cabo una correcta docencia a distancia, sería bastante interesante una opción en el que pudiera haber un chat entre usuarios para resolver dudas rápidamente, pudiendo definir el profesor un horario de consulta y que el alumnado no tenga la necesidad de ir presencialmente a resolver algunas dudas que por correo tal vez no llegan a explicarse claramente. Por otro lado, la funcionalidad de un foro serviría para que los alumnos puedan debatir la respuesta a una pregunta y así poder evaluarles algunas competencias transversales.
- **Internalización.** En futuras versiones nos gustaría que cualquier persona que no sepa español pueda utilizar la aplicación sin que exista una barrera con el idioma. Para ello, sería conveniente implementar diferentes idiomas y traducciones en la aplicación.
- **Pruebas con usuarios reales.** En nuestro trabajo solo realizamos pruebas de funcionamiento, pero no se han realizado pruebas con usuarios reales para poder reconocer las dificultades que tienen estos con el uso de este tipo de aplicaciones. Por otra parte, los usuarios podrían dar nuevas ideas para futuras implementaciones sobre aspectos que las aplicaciones líderes del mercado no recogen, por lo que sería interesante realizar pruebas con profesorado y alumnado de distintos institutos o universidades.

Bibliografía

[1] Dr. Charles Russell Severance.
Python para todos: Explorando la información con Python 3.
Publicado independientemente, primera edición, 2020.

[2] Mark Lutz.
Learning Python.
O'Reilly Media, Inc., quinta edición, 2013.

[3] Julia Elman, Mark Lavin.
Lightweight Django.
O'Reilly Media, Inc., primera edición, 2013.

[4] Ben Shaw, Saurabh Badhwar, Andrew Bird, Bharath Chandra K S, Chris Guest.
Web Development with Django: Learn to build modern web applications with a Python-based framework.
Packt Publishing, primera edición, 2021.

[5] Estadísticas de uso de Moodle en todo el mundo.
Consultado en https://stats.moodle.org/.

[6] Página principal de la aplicación Sakai
Consultado en https://www.sakailms.org/.

[7] Características de la plataforma Edmodo
Consultado en https://emtic.educarex.es/PildoTIC/pildoras_experiencia/edmodo/la_plataforma_edmodo.html.

[8] Explicación del método MoSCoW
Consultado en https://proagilist.es/blog/agilidad-y-gestion-agil/priorizar-requisitos-tecnica-priorizacion-moscow/.

[9] Explicación del análisis DAFO
Consultado en https://www.infoautonomos.com/plan-de-negocio/analisis-dafo/.

[10] Características y ventajas de Laravel
Consultado en https://openwebinars.net/blog/que-es-laravel-caracteristicas-y-ventajas/.

[11] Qué es Django y por qué utilizarlo
Consultado en https://openwebinars.net/blog/que-es-django-y-por-que-usarlo/.

[12] Qué es Flask
Consultado en https://openwebinars.net/blog/que-es-flask/.

[13] Que és Express.js y ventajas sobre su uso
Consultado en https://www.tutorialsteacher.com/nodejs/expressjs.

[14] Ventajas sobre la programación con Ruby on Rails
Consultado en https://coditramuntana.com/es/blog/ventajas-programacion-ruby-on-rails.

[15] Estadística y explicación sobre los frameworks de backend más utilizados desde 2012 a 2021
Consultado en https://statisticsanddata.org/data/most-popular-backend-frameworks-2012-2021/.

[16] Estadística y explicación sobre los frameworks de backend más utilizados desde 2012 a 2021
Consultado en https://statisticsanddata.org/data/most-popular-backend-frameworks-2012-2021/.

[17] Acoplamiento débil en Django
Consultado en https://uniwebsidad.com/libros/django-1-0/capitulo-3/urlconfs-y-el-acoplamiento-debil.

[18] Instalación de PostgreSQL y configuración de la base de datos en Django.
Consultado en https://www.enterprisedb.com/postgres-tutorials/how-use-postgresql-django.

[19] Que és JavaScript y para qué sirve.
Consultado en https://soyrafaramos.com/que-es-javascript-para-que-sirve/.

[20] Cómo configurar Amazon S3 en Django desde cero.
Consultado en https://simpleisbetterthancomplex.com/tutorial/2017/08/01/how-to-setup-amazon-s3-in-a-django-project.html/.

[21] Cómo desplegar una aplicación Django en Heroku.
Consultado en https://simpleisbetterthancomplex.com/tutorial/2016/08/09/how-to-deploy-django-applications-on-heroku.html.