Desarrollo de un sistema de gestión, comunicación y planificación para estudiantes universitarios

TRABAJO DE FIN DE GRADO EN INGENIERÍA INFORMÁTICA

CURSO 2017-2018

Daniel Antonio Coldeira Pérez

Director

Antonio Sarasa Cabezuelo

Facultad de Informática Universidad Complutense de Madrid

Mayo 2018

Resumen

En este trabajo se ha creado una aplicación para gestionar material formativo, comunicar a los alumnos de una misma facultad y planificar el desarrollo del curso académico.

El objetivo del trabajo es ayudar al alumno universitario a completar cualquier actividad relacionada con el ámbito universitario.

La aplicación realizada incluye dos tipos de actores: los usuarios, alumnos de una facultad, y los administradores.

Los usuarios disponen de funcionalidades tales como la búsqueda y visualización de material formativo relacionado con los estudios impartidos en la universidad en la que estudian, almacenamiento de materiales formativos, realización de comentarios y valoraciones de los materiales almacenados en el sistema.

Incluye la personalización de un horario y un panel de notas para llevar un seguimiento de su año académico, también tiene la opción de compartir experiencias y comunicarse con otros usuarios de la misma facultad a través de su red social. Dispone de una sección con enlaces de interés para el usuario, personalizados según su facultad.

Con respecto a los administradores, estos pueden gestionar todo el contenido, gestionar los usuarios o recibir avisos por material inadecuado.

.

Palabras clave: Aplicación web, red social, repositorio material universitario, personalización anotaciones y horario.

Abstract

In this work, an application has been created to manage training material, communicate students of the same faculty and planning the development of the academic year.

The objective of the work is to help the university student to complete any activity related to the university environment.

The application includes two types of actors: users, students of a faculty, and administrators.

Users have functionalities such as the search and visualization of training material related to the studies taught at the university where they study, storage of training materials, comments and assessments of materials stored in the system.

It includes the personalization of a schedule and a panel of notes to keep track of your academic year, you also have the option of sharing experiences and communicating with other users of the same faculty through your social network. It has a section with links of interest for the user, personalized according to their faculty.

With regard to administrators, they can manage all content, manage users or receive notifications for inappropriate material.

Keywords: Web application, social network, university material repository, personalization annotations and schedule.

Agradecimientos

"No dejes apagar el entusiasmo, virtud tan valiosa como necesaria; trabaja, aspira, tiende siempre hacia la altura".

Rubén Darío

Quiero dar las gracias a mi director de proyecto, Antonio Sarasa Cabezuelo, por ayudarme en la consecución de este proyecto.

Agradecer a mi familia y amigos por estar junto a mí durante estos duros meses de trabajo, mención especial a mi mejor amiga y compañera, María, por permanecer a mi lado en los momentos más difíciles, por confiar siempre en mi y darme todo su apoyo.

Gracias a todos ellos por creer en mí y por ayudarme a culminar este gran reto.

Índice General

Tabla de contenido

Índice de figuras

1.Introducción

1.1 Motivación

Durante la vida de un estudiante universitario, se tiene la necesidad de utilizar ciertas herramientas o recursos para complementar la experiencia en la universidad.

Dichos recursos se pueden obtener de diferentes fuentes, bien herramientas proporcionadas directamente por las propias universidades o bien en aplicaciones independientes que intentan proporcionar ese servicio. Para conocer eventos o noticias de actualidad referentes a una universidad se debe buscar en la web de la facultad o personarse en la propia facultad para recopilar información.

Otros recursos como apuntes o cierta información específica hay que buscarlos en internet sin tener la seguridad de encontrar exactamente lo que se está buscando.

Este contexto ha servido de punto de partida de este trabajo fin de grado: crear una aplicación web que cubriera todas las necesidades de materiales formativos que tiene un alumno.

1.2 Objetivos

El objetivo principal de este proyecto es crear un sistema de gestión, comunicación y planificación para estudiantes universitarios. Este objetivo principal se concreta en los siguientes objetivos:

- Facilitar a los estudiantes un sistema donde poder almacenar materiales formativos para compartirlos con otros estudiantes.
- Aportar un sistema de búsqueda de material formativo eficaz.
- Proporcionar un sistema de comentarios para comentar y evaluar los diferentes materiales formativos.
- Facilitar una red social para estudiantes que comparten universidad y titulación, permitiendo la comunicación.
- Mostrar un ranking con los estudiantes más activos en diferentes áreas para incentivar el uso de la aplicación.

- Proporcionar una sección de anotaciones para que cada estudiante guarde notas relacionadas con el transcurso de la carrera.
- Aportar un horario editable para que cada alumno edite sus asignaturas.
- Mostrar información personal y estadísticas de cada estudiante a través de su perfil.
- Facilitar un buzón a cada estudiante para recibir y enviar mensajes al resto de estudiantes.
- Proporcionar enlaces de interés para el estudiante relacionados con su facultad y su carrera.
- Aportar un sistema para administrar toda esta información y a los propios usuarios.

1.3 Estado del Arte

A continuación, se van a revisar otras aplicaciones y herramientas que disponen de funcionalidades parecidas a las desarrolladas en este trabajo de fin de grado.

Wuolah

Se trata de una aplicación web[35] que permite el almacenamiento y compartición de materiales entre los alumnos de una misma facultad y titulación (Figura 1). Para motivar a los usuarios para que compartan recursos se les ofrece pequeñas compensaciones económicas que son depositadas en una cuenta vinculada al usuario. La cantidad de esos pagos depende del número de descargas que se realicen sobre los recursos añadidos y del número total de archivos compartidos. Normalmente el contenido que puede encontrarse son apuntes, exámenes o prácticas. Dispone de una sección donde los alumnos pueden

colgar noticias y eventos. Sin embargo, la aplicación está muy enfocada a la gestión de apuntes.

Figura 1. Captura del sitio web de Wuolah

RincónDelVago

Se trata de un sitio web[29] que actúa como un gran repositorio de materiales orientados a la formación (Figura 2). Aquí se puede encontrar cualquier trabajo o apunte para todo tipo de enseñanzas.

Figura 2. Captura del sitio web de RincónDelVago

PatataBrava

Se trata de un sitio web[28] que permite compartir apuntes y otros materiales como trabajos o prácticas (Figura 3). Dispone de un buscador en el que se pueden buscar apuntes de titulaciones y asignaturas específicas, también dispone de información tanto de grados universitarios como de másteres.

Figura 3. Captura del sitio web de PatataBrava

Unybook

Se trata de un sitio web[34] especializado en encontrar apuntes específicos sobre alguna materia (Figura 4). El usuario debe realizar la solicitud a otros usuarios sobre el material que está interesado, cuando dicho material este disponible en el repositorio de la web se envía una notificación al usuario interesado.

Figura 4. Captura del sitio web de Unybook

UCM Móvil

Aplicación oficial [33] de la UCM (Universidad Complutense de Madrid) para dispositivos móviles (Figura 5). Dispone de diferentes secciones en las que el usuario puede obtener información útil, también puede recopilar información de los diferentes estudios ofertados, consultar la localización de una facultad específica y obtener información de contacto. Dispone de una sección de noticias de actualidad.

Figura 5. Captura de la app UCM móvil

1.4 Estructura de la memoria

A continuación, se describe brevemente la estructura de la memoria:

- En el primer capítulo se expone la motivación del proyecto, los objetivos planteados, se describe un breve estado del arte y la estructura de la memoria.
- En el segundo capítulo se explica con detalle cada una de las tecnologías que se utilizan en la consecución de este proyecto.
- En el tercer capítulo se detallan los casos de uso presentes en la aplicación. Para cada uno de ellos se explica su nombre, flujo principal y flujo alternativo.
- En el cuarto capítulo se hace un repaso a la arquitectura utilizada para la consecución del proyecto.
- En el quinto capítulo se describen el modelo de datos utilizado en la aplicación.
- En el sexto capítulo se describe el diseño utilizado en la realización del proyecto.
- En el séptimo capítulo se detalla cada una de las funcionalidades de la aplicación a través de capturas de la propia aplicación y partes del código especialmente importantes.
- En el octavo capítulo se exponen las conclusiones obtenidas a partir de la elaboración del proyecto y las líneas de trabajo futuro que podría realizarse a posteriori para mejorar.
- A continuación, en la bibliografía se encuentran las referencias utilizadas en el trabajo.
- Para finalizar, se incluyen como anexos una guía de instalación y otra de uso de la aplicación.

1. Introduction

1.1 Motivation

During the life of a university student, there is a need to use certain tools or resources to complement the experience in the university.

These resources can be obtained from different sources, either tools provided directly by the universities themselves or in independent applications that try to provide this service. To know events or news about a university, you must search the faculty's website or visit the faculty to collect information.

Other resources such as notes or certain specific information must be searched on the internet without having the security of finding exactly what you are looking for.

This context has served as the starting point for this final degree project: create a web application that covers all the training materials needs of a student.

1.2 Objetives

The main objective of this project is to create a management, communication and planning system for university students. This main objective is specified in the following objectives:

- Provide students with a system where they can store training materials to share with other students.
- Provide a search system for effective training material.
- Provide a system of comments to comment and evaluate the different training materials.
- Facilitate a social network for students who share university and qualifications, allowing communication.

.

- Show a ranking with the most active students in different areas to encourage the use of the application.
- Provide a section of notes for each student to keep notes related to the course of the race.
- Provide an editable schedule for each student to edit their subjects.
- Show personal information and statistics of each student through their profile.
- Provide a mailbox for each student to receive and send messages to the rest of the students.
- Provide links of interest to the student related to their faculty and career.
- Provide a system to manage all this information and the users themselves.

1.3 State of the Art

Then, we will review other applications and tools that have functionalities similar to those developed in this end-of-degree project.

Wuolah

It is a web application [35] that allows the storage and sharing of materials among students of the same faculty and degree (Figure 1). To motivate users to share resources, they are offered small financial compensations that are deposited in an account linked to the user. The amount of these payments depends on the number of downloads that are made on the added resources and the total number of shared files. Normally the content that can be found are notes, exams or practices. It has a section where students can post news and events. However, the application is very focused on the management of notes.

Figure 1. Wuolah website capture

RincónDelVago

It is a website [29] that acts as a large repository of training-oriented materials (Figure 2). Here you can find any work or point for all kinds of teachings.

Figure 2. Capture of the RincónDelVago website

PatataBrava

It is a website [28] that allows you to share notes and other materials such as jobs or practices (Figure 3). It has a search engine where you can find notes of specific degrees and subjects, also has information on both university degrees and master's degrees.

Figure 3. Capture of the PatataBrava website

Unybook

It is a website [34] that specializes in sharing notes (Figure 4). This allows each user to make requests to other users about the materials in which they are interested, so that when they are in the repository, they are sent a notification.

Figure 4. Capture of the Unybook website

UCM Móvil

Official application [33] of the UCM (Complutense University of Madrid) for mobile devices (Figure 5). It has different sections in which the user can obtain useful information, can also collect information from the different studies offered, consult the location of a specific faculty and obtain contact information. It has a current news section.

Figure 5. Capture of the mobile UCM app

1.4 Memory structure

Next, the structure of the memory is briefly described:

- In the first chapter, the motivation of the project, the stated objectives, a brief state of art and the structure of memory are described. • In the second chapter, each of the technologies used in the achievement of this project is explained in detail.
- In the third chapter the use cases present in the application are detailed, for each of them the name, main flow and alternative flow will be explained.
- In the fourth chapter there is a review of the architecture used to achieve the project.
- In the fifth chapter, the data models used in the application will be described. All collections and each of their fields will be explained.
- The sixth chapter describes the design used in the realization of the project. • In the seventh chapter, each of the application's features is detailed through captures of the application itself and especially important parts of the code.
- In the eighth chapter, the conclusions drawn during the elaboration and completion of the project and the work that could be done a posteriori to improve are exposed.
- After the bibliography where all the material used is found, naming all the content in a detailed and standardized way.
- Finally, the guide and installation annexes to use the application

2. Especificación de requisitos

En este capítulo se describe la especificación de requisitos del sistema que se ha desarrollado en este trabajo de fin de grado.

Los casos de uso se van a presentar de acuerdo con los dos tipos de actores que intervienen en el sistema.

a) Administrador

En la Figura 6 se muestran los casos de uso del actor Administrador. El papel que desempeña este actor es registrar y modificar facultades, validar recursos y gestionar la admisión y el bloqueo de usuarios.

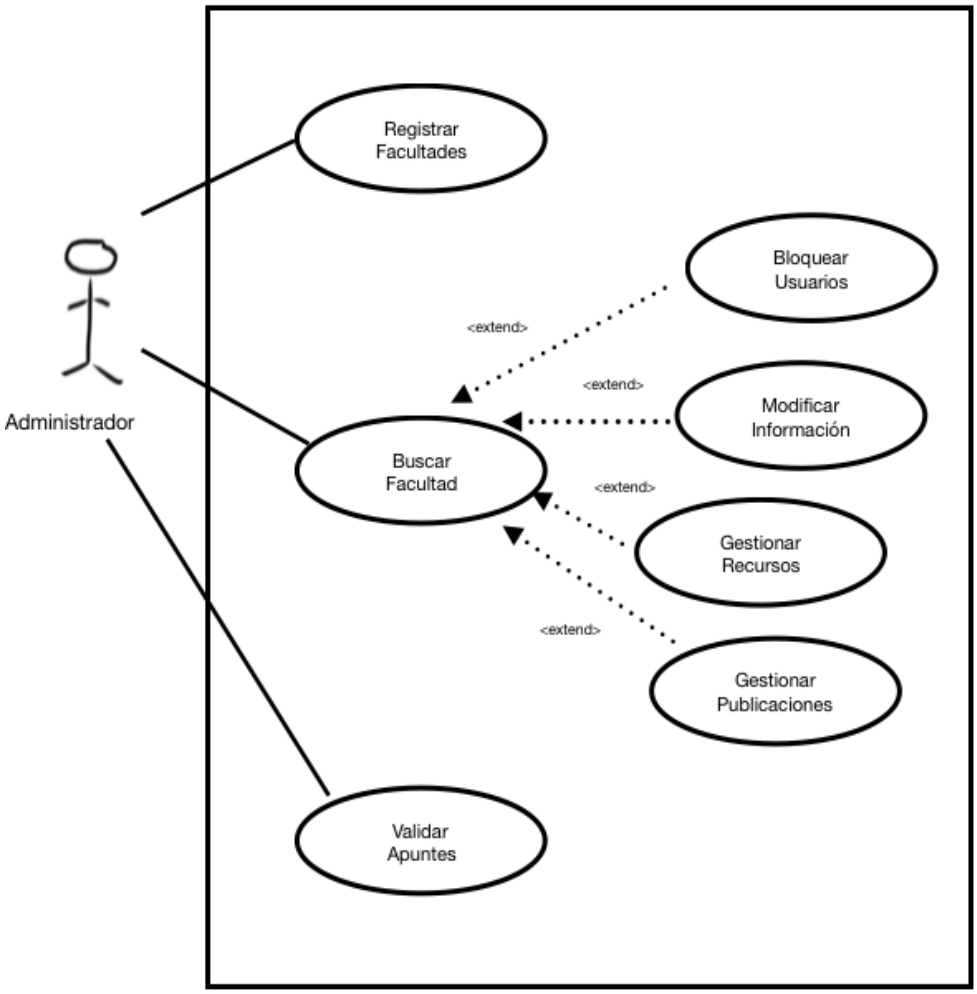


Figura 6: Diagrama de casos de uso para Actor administrador

A continuación, se describen los diferentes casos de uso.

Identificador del caso de uso: CU1.1

Nombre: Registrar Facultades

Actores: Administrador.

Descripción: El administrador puede registrar nuevas facultades. Para cada facultad se tendrán que añadir las titulaciones y las asignaturas asociadas a cada titulación.

Precondición: El administrador debe haberse registrado anteriormente y autenticado.

Postcondición: Ninguna.

Flujo Principal:

Paso 1– Administrador: Se sitúa en la pantalla de inicio. Allí podrá pinchar sobre un botón para registrar una nueva facultad.

Paso 2– Sistema: El sistema desplegará una nueva página para que el administrador rellene el nombre de la facultad, las titulaciones que oferta y las asignaturas asociadas a cada titulación.

Paso 3– Administrador: El administrador rellena todos los campos y envía la información al sistema.

Paso 4– Sistema: El sistema recibirá toda la información y añadirá la facultad a la lista de facultades registradas en la aplicación.

Flujo alternativo: Falta de rellenar campos

Paso 4 – Sistema: El sistema devuelve un mensaje de error indicando qué campos faltan por rellenar.

Identificador del caso de uso: CU1.2

Nombre: Validar Apuntes

Actores: Administrador.

Descripción: El administrador tiene un buzón donde llegan todos los recursos pendientes, que todavía no han sido revisados, subidos por los usuarios. El administrador tiene la opción de validar el recurso o descartarlo.

Precondición: El administrador debe haberse registrado anteriormente y autenticado.

Postcondición: Ninguna.

Flujo Principal:

Paso 1– Administrador: Se sitúa en la pantalla de inicio. Allí se dirige a la opción buzón de recursos pendientes.

Paso 2– Sistema: El sistema desplegará todos los recursos de todas las facultades que todavía no están revisados.

Paso 3– Administrador: El administrador podrá acceder a cada recurso y tiene la opción de validarlo o descartarlo.

Paso 4– Sistema: El sistema eliminará del buzón el recurso en cualquiera de las dos opciones posibles.

Identificador del caso de uso: CU1.3

Nombre: Buscar Facultad

Actores: Administrador

Descripción: Buscar la facultad de la que quiere modificar la información.

Precondición: El administrador debe haberse registrado anteriormente y autenticado.

Postcondición: Ninguna.

Flujo Principal:

Paso 1– Administrador: Se sitúa en la pantalla de inicio. Allí elige opción buscar facultad.

Paso 2– Sistema: El sistema devolverá una lista con todas las facultades registradas.

Paso 3– Administrador: El administrador elegirá alguna de las facultades de la lista.

Identificador del caso de uso: CU1.4

Nombre: Modificar Información

Actores: Administrador

Descripción: El administrador puede modificar cualquier información relacionada con una facultad, como asignaturas, cursos o titulaciones y enlaces.

Precondición: El administrador debe haberse registrado anteriormente y autenticado. Después debe buscar facultad a la que pertenece esa información.

Postcondición: Ninguna.

Flujo Principal:

Paso 1– Administrador: Una vez ha buscado la facultad el administrador elige la opción de modificar la información de dicha facultad.

Paso 2– Sistema: El sistema devolverá el contenido de la sección de información.

Paso 3– Administrador: El administrador puede borrar o modificar cualquier parte de esta sección.

Identificador del caso de uso: CU1.5

Nombre: Gestionar recursos

Actores: Administrador.

Descripción: El administrador puede borrar cualquier clase de recurso asociado a cualquier asignatura de cualquier facultad.

Precondición: El administrador debe haberse registrado anteriormente y autenticado. Buscar facultad.

Postcondición: Ninguna.

Flujo Principal:

Paso 1– Administrador: Una vez ha buscado la facultad el administrador elige la titulación de la que quiere modificar las asignaturas.

Paso 2– Sistema: El sistema devolverá las asignaturas pertenecientes a esa titulación.

Paso 3– Administrador: El administrador elegirá la asignatura de la que quiere modificar o añadir un recurso.

Paso 4– Sistema: El sistema devolverá todos los recursos asociados a la asignatura escogida.

Paso 5– Administrador: El administrador puede borrar dicho recurso.

Identificador del caso de uso: CU1.6

Nombre: Gestionar Publicaciones

Actores: Administrador

Descripción: El administrador puede borrar cualquier publicación asociado a cualquier facultad.

Precondición: El administrador debe haberse registrado anteriormente y autenticado. Buscar facultad.

Postcondición: Ninguna.

Flujo Principal:

Paso 1– Administrador: Una vez ha buscado la facultad el administrador elige la opción gestionar publicaciones.

Paso 2– Sistema: El sistema devolverá todas las publicaciones de esa facultad ordenadas por fecha de creación.

Paso 3– Administrador: El administrador tendrá la opción de borrar cualquiera de las publicaciones.

Identificador del caso de uso: CU1.7
Nombre: Bloquear Usuarios
Actores: Administrador
Descripción: El administrador puede bloquear a cualquier usuario registrado.
Precondición: El administrador debe haberse registrado anteriormente y autenticado. Buscar facultad.
Postcondición: Ninguna.
Flujo Principal:
Paso 1– Administrador: Una vez ha buscado la facultad, el administrador elige la opción bloquear usuario.
Paso 2– Sistema: El sistema devolverá todos los usuarios registrados de la facultad buscada.
Paso 3– Administrador: El administrador tendrá la opción de elegir un usuario y de bloquear a dicho usuario durante el tiempo que elija.

Usuario

En la Figura 7 se muestran los casos de uso del actor Usuario. El papel que desempeña este actor es interactuar con la aplicación.

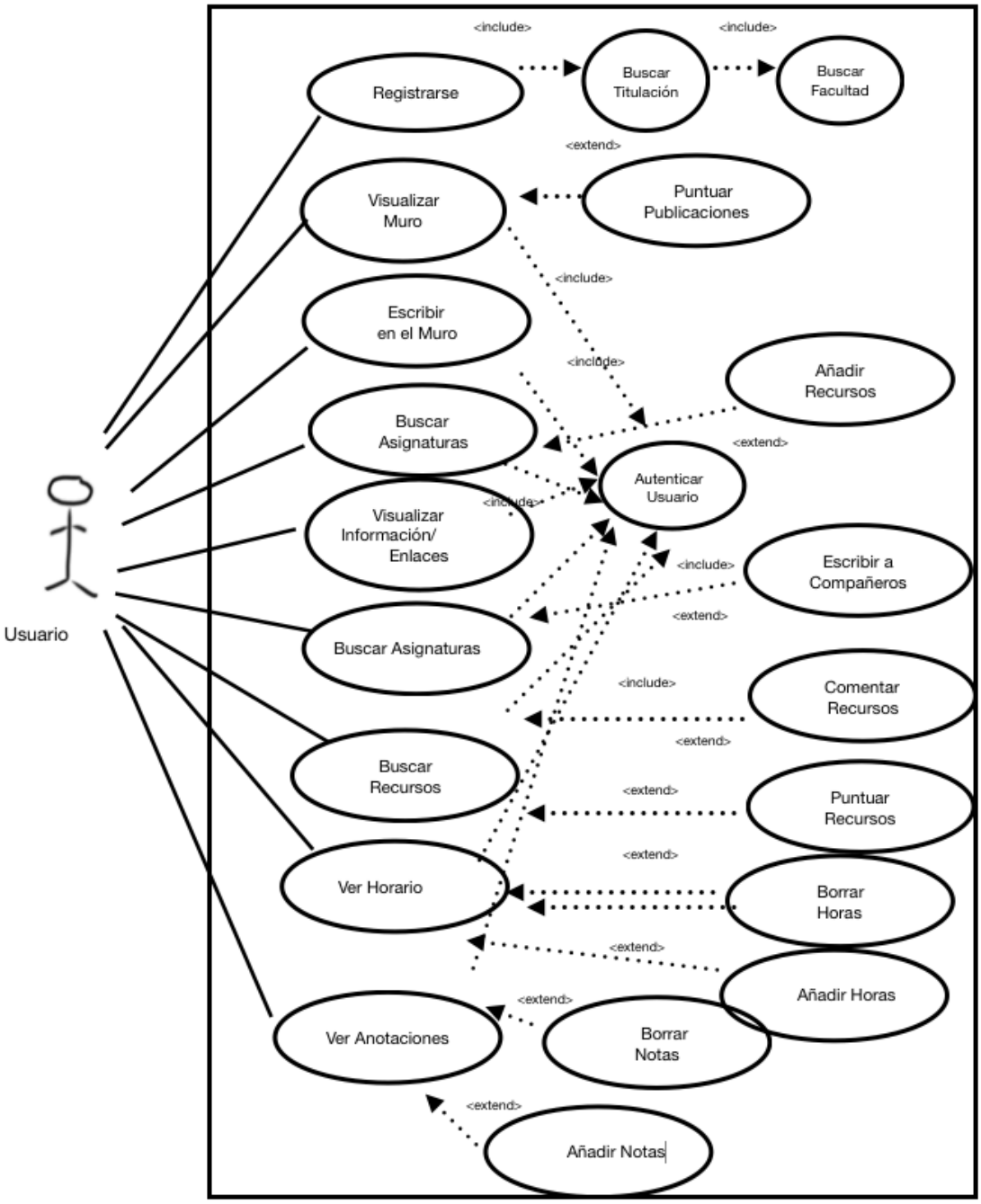


Figura 7: Diagrama de casos de uso para actor usuario

A continuación, se describen los diferentes casos de uso.

Identificador del caso de uso: CU2.1

Nombre: Registrarse

Actores: Usuario.

Descripción: En la página principal si el usuario no dispone de una cuenta creada

previamente tendrá que registrarse en el sistema aportando información y eligiendo un nombre y una contraseña.

Precondición: Ninguna.

Postcondición: Ninguna.

Flujo Principal:

Paso 1– Usuario: Accede a la aplicación.

Paso 2– Usuario: El accede a la sección de registro.

Paso 3– Sistema: El sistema despliega un formulario con datos que el usuario debe completar.

Paso 4– Usuario: El usuario tendrá que rellenar la información que se solicite.

Paso 5– Usuario: El usuario deberá elegir un nombre y una contraseña según las especificaciones que proporcione la aplicación.

Paso 6– Sistema: El sistema registrará al usuario asignado el nombre y la contraseña elegidos y dará acceso a la aplicación.

Paso 7– Usuario: El usuario verá la página de inicio de la aplicación.

Flujo alternativo: Usuario existente/Contraseña invalida

Paso 6 – Sistema: El sistema ya contiene un usuario con ese nombre o la contraseña no es válida .El sistema envía un mensaje al usuario informando de que se debe elegir un nombre de usuario que no esté registrado o cambiar la contraseña por una valida.

Identificador del caso de uso: CU2.2

Nombre: Autenticar usuario / Login

Actores: Usuario.

Descripción: El usuario entra en la aplicación, introduce el usuario y la contraseña en el login.

Precondición: Ninguna.

Postcondición: El usuario queda logueado, tiene acceso a la aplicación.

Paso 1 – Usuario: Accede a la web.

Paso 2 – Sistema: Pide al usuario que se identifique con un nombre de usuario y una contraseña.

Paso 3 – Usuario: El usuario introduce el nombre y la contraseña.

Paso 4 – Sistema: Realiza la autenticación de la cuenta. Después da acceso a la aplicación.

Paso 5 – Usuario: Se sitúa en la pantalla de inicio. Allí podrá visualizar el muro con las últimas publicaciones.

Flujo alternativo: Login incorrecto

Paso 4 – Sistema: Fallo a la hora de la autenticación. El sistema lanza un mensaje de error, especificando la causa de este.

Identificador del caso de uso: CU2.3

Nombre: Visualizar Muro

Actores: Usuario.

Descripción: En la página principal podrá visualizar el muro donde se ven las últimas publicaciones de los usuarios de la web. Dicho muro tendrá la opción de visualizar contenido según titulación.

Precondición: El usuario debe haberse registrado anteriormente y autenticado.

Postcondición: Ninguna.

Flujo Principal:

Paso 1– Usuario: Se sitúa en la pantalla de inicio. Allí podrá visualizar el muro con las últimas publicaciones.

Paso 2– Usuario: El usuario podrá filtrar dichas publicaciones según la titulación.

Paso 3– Sistema: El sistema devolverá las publicaciones según el criterio de búsqueda y ordenados según fecha de publicación.

Flujo alternativo: Búsqueda incorrecta

Paso 3 – Sistema: El sistema devuelve una búsqueda vacía.

Identificador del caso de uso: CU2.4

Nombre: Escribir en el Muro

Actores: Usuario.

Descripción: En la página principal tendrá la opción de generar una publicación, después tendrá la opción de enviar dicha publicación al sistema para posteriormente publicarla en el muro.

Precondición: El usuario debe haberse registrado anteriormente y autenticado.

Postcondición: Ninguna.

Flujo Principal:

Paso 1– Usuario: Se sitúa en la pantalla de inicio. Allí podrá generar contenido.

Paso 2– Usuario: Después lo enviará al sistema.

Paso 3– Sistema: El sistema recibirá el contenido enviado por el usuario y lo publicará en el muro.

Paso 4– Usuario: El usuario verá su publicación reflejada en el muro.

Identificador del caso de uso: CU2.5

Nombre: Búsqueda de asignaturas

Actores: Usuario.

Descripción: En la pestaña referente a asignaturas podrá visualizar las asignaturas asociadas a la titulación del usuario en su universidad. El usuario podrá pinchar en la asignatura directamente en la lista o encontrarla a través de un buscador.

Precondición: El usuario debe haberse registrado anteriormente y autenticado.

Postcondición: Ninguna.

Flujo Principal:

Paso 1– Usuario: Se sitúa en la pestaña asignaturas. Allí podrá todas las asignaturas de su titulación ordenadas por cada curso.

Paso 2– Usuario: El usuario podrá pinchar en alguna asignatura directamente de la lista o buscarla a través de un buscador.

Paso 3– Sistema: El sistema devolverá la información relativa a dicha asignatura y sus recursos asociados.

Paso 4– Usuario: El usuario verá la información de la asignatura elegida y podrá acceder a los recursos asociados a dicha asignatura.

Flujo alternativo: Búsqueda incorrecta

Paso 3 – Sistema: El sistema devuelve una búsqueda vacía.

Identificador del caso de uso: CU2.6

Nombre: Visualizar Información

Actores: Usuario.

Descripción: El usuario podrá visualizar información relativa a la universidad como pueden ser información de contacto de la universidad, como llegar a la universidad, información de la universidad en otros servicios como Facebook, twitter o YouTube, enlaces a la calculadora de la matricula.

Precondición: El usuario debe haberse registrado anteriormente y autenticado.

Postcondición: Ninguna.

Flujo Principal:

Paso 1– Usuario: Se sitúa en la pantalla de información. Allí podrá visualizar la información disponible sobre la facultad.

Paso 2– Sistema: El sistema devuelve toda la información y enlaces de interés.

Paso 3– Usuario: El usuario podrá pinchar sobre diferentes enlaces que darán información sobre la universidad asociada.

Paso 4– Sistema: El sistema re direccionara al usuario a las diferentes redes sociales realizando una búsqueda en estas de la facultad y titulación asociada.

Paso 5– Usuario: El usuario podrá visualizar directamente información de la universidad en las redes sociales sin necesidad de realizar la búsqueda.

Identificador del caso de uso: CU2.7

Nombre: Buscar compañeros

Actores: Usuario.

Descripción: En la página principal existirá una sección con una lista de todos los usuarios

Precondición: El usuario debe haberse registrado anteriormente y autenticado.

Postcondición: Ninguna.

Flujo Principal:

Paso 1– Usuario: Se sitúa en la pantalla de inicio. Allí podrá visualizar todos los usuarios asociados a esa facultad en forma de lista.

Paso 2– Usuario: El usuario podrá buscar usuario por palabras clave.

Paso 3– Sistema: El sistema devolverá información del usuario elegido.

Identificador del caso de uso: CU2.8

Nombre: Buscar recursos

Actores: Usuario.

Descripción: En la pestaña referente a asignaturas podrá visualizar las asignaturas asociadas a la titulación del usuario en su universidad. El usuario podrá pinchar en la asignatura directamente en la lista o encontrarla a través de un buscador. Después podrá

acceder a los recursos asociados a la asignatura. También puede utilizar el buscador para encontrar apuntes.

Precondición: El usuario debe haberse registrado anteriormente y autenticado. Si se quiere acceder a través de la lista de asignaturas se deberá buscar primero la asignatura correspondiente al recurso que se busca, Buscar Asignatura.

Postcondición: Ninguna.

Flujo Principal:

Paso 1– Usuario: El usuario se sitúa en la pestaña de las asignaturas.

Paso 2– Usuario: El usuario escribe en el buscador una palabra clave para encontrar el recurso que busca o lo encuentra buscando primero la asignatura y buscarlo en la lista de recursos desplegada.

Paso 3– Sistema: El sistema devolverá los recursos que coincidan con la palabra utilizada para la búsqueda.

Paso 4– Usuario: El usuario podrá pinchar en alguno de los resultados obtenidos para acceder al recurso (Examen, apuntes, prácticas o ejercicios)

Paso 5– Sistema: El sistema devolverá el recurso requerido por el usuario.

Flujo alternativo: Búsqueda incorrecta

Paso 3 – Sistema: El sistema devuelve una búsqueda vacía.

Identificador del caso de uso: CU2.9

Nombre: Añadir recursos

Actores: Usuario.

Descripción: El usuario podrá subir recursos a la aplicación web, tanto exámenes como apuntes como ejercicios asociándolos a una asignatura. Dichos recursos deben ser aprobados por el administrador antes de añadirlos definitivamente.

Precondición: El usuario debe haberse registrado anteriormente y autenticado. También debe haber buscado la asignatura de la que se quiere añadir el recurso.

Postcondición: Ninguna.

Flujo Principal:

Paso 1– Usuario: Pincha sobre el botón añadir recurso.

Paso 2– Sistema: El sistema pedirá al usuario que se especifique que clase de recurso es y que se suba dicho recurso al sistema

Paso 3– Usuario: El alumno elegirá el tipo de recurso y asociará el fichero adecuado.

Paso 4– Sistema: El sistema enviará dicho recurso al administrador para que autorice su subida al sistema.

Paso 5– Sistema: Una vez se ha validado el recurso el sistema lo asocia a la asignatura elegida por el usuario.

Paso 6 -Usuario: El alumno podrá acceder a dicho recurso.

Flujo alternativo: Archivo con formato incorrecto

Paso 4 – Sistema: El sistema devuelve un mensaje de error.

Identificador del caso de uso: CU2.10

Nombre: Escribir a compañeros

Actores: Usuario.

Descripción: El usuario puede escribir a algún compañero de facultad que haya buscado previamente.

Precondición: El usuario debe haberse registrado anteriormente y autenticado. También debe haber buscado al compañero con el que quiera comunicarse, Buscar compañero.

Postcondición: Ninguna.

Flujo Principal:

Paso 1– Usuario: El usuario pincha sobre el botón escribir mensaje asociado a otro usuario.

Paso 2– Sistema: El sistema devuelve una caja de texto donde escribir el mensaje.

Paso 3– Usuario: El usuario escribirá el mensaje en la zona desplegada por el sistema y lo enviará.

Paso 4– Sistema: El sistema recibirá el mensaje y lo enviará al usuario elegido.

Paso 5– Usuario: El usuario ve el mensaje en mensajes enviados.

Identificador del caso de uso: CU2.11

Nombre: Puntuar recursos

Actores: Usuario.

Descripción: En la pagina asociada a un recurso el alumno tiene la opción de puntuar dicho recurso a través de una calificación con estrellas.

Precondición: El usuario debe haberse registrado anteriormente y autenticado. Debe haber buscado el recurso que quiere puntuar.

Postcondición: Ninguna.

Flujo Principal:

Paso 1– Usuario: El usuario tendrá la opción de puntuar el recurso a través de una calificación de estrellas.

Paso 2– Sistema: El sistema hará la media de la última calificación con el resto de las calificaciones dadas a ese recurso y enviará dicha información.

Paso 3– Usuario: El usuario podrá visualizar la calificación actualizada.

Identificador del caso de uso: CU2.12

Nombre: Comentar recursos

Actores: Usuario.

Descripción: En la pagina asociada a un recurso el alumno tiene la opción de comentar dicho recurso.

Precondición: El usuario debe haberse registrado anteriormente y autenticado. Debe haber buscado el recurso que quiere comentar.

Postcondición: Ninguna.

Flujo Principal:

Paso 1– Usuario: El usuario pincha sobre la opción añadir comentario.

Paso 2– Sistema: El sistema devolverá una caja para escribir el comentario.

Paso 3– Usuario: El usuario escribirá el comentario y lo enviará al sistema.

Paso 4– Sistema: El sistema asociará al recurso el comentario del usuario.

Paso 5– Usuario: El usuario visualiza el comentario asociado al recurso.

Identificador del caso de uso: CU2.13

Nombre: Ver Horario

Actores: Usuario.

Descripción: En la pestaña referente horario el usuario podrá visualizar un horario en el cuál se detallan todas las horas de cada día de la semana laboral, de lunes a viernes, de 09:00 h. a 21:00.

Precondición: El usuario debe haberse registrado anteriormente y autenticado.

Postcondición: Ninguna.

Flujo Principal:

Paso 1– Usuario: El usuario se sitúa en la pestaña de horario.

Paso 2– Sistema: El sistema devolverá el horario correspondiente con el usuario que se ha logueado anteriormente.

Identificador del caso de uso: CU2.14

Nombre: Eliminar Horas

Actores: Usuario.

Descripción: El usuario podrá subir recursos a la aplicación web, tanto exámenes como apuntes como ejercicios asociándolos a una asignatura. Dichos recursos deben ser aprobados por el administrador antes de añadirlos definitivamente.

Precondición: El usuario debe haberse registrado anteriormente y autenticado. También debe haber buscado la asignatura de la que se quiere añadir el recurso.

Postcondición: Ninguna.

Flujo Principal:

Paso 1– Usuario: Pincha sobre el botón añadir recurso.

Paso 2– Sistema: El sistema pedirá al usuario que se especifique que clase de recurso es y que se suba dicho recurso al sistema

Paso 3– Usuario: El alumno elegirá el tipo de recurso y asociará el fichero adecuado.

Paso 4– Sistema: El sistema enviará dicho recurso al administrador para que autorice su subida al sistema.

Paso 5– Sistema: Una vez se ha validado el recurso el sistema lo asocia a la asignatura elegida por el usuario.

Paso 6 -Usuario: El alumno podrá acceder a dicho recurso.

Flujo alternativo: Archivo con formato incorrecto

Paso 4 – Sistema: El sistema devuelve un mensaje de error.

Identificador del caso de uso: CU2.15

Nombre: Añadir horas

Actores: Usuario.

Descripción: El usuario puede escribir a algún compañero de facultad que haya buscado previamente.

Precondición: El usuario debe haberse registrado anteriormente y autenticado. También debe haber buscado al compañero con el que quiera comunicarse, Buscar compañero.

Postcondición: Ninguna.

Flujo Principal:

Paso 1– Usuario: El usuario pincha sobre el botón escribir mensaje asociado a otro usuario.

Paso 2– Sistema: El sistema devuelve una caja de texto donde escribir el mensaje.

Paso 3– Usuario: El usuario escribirá el mensaje en la zona desplegada por el sistema y lo enviará.

Paso 4– Sistema: El sistema recibirá el mensaje y lo enviará al usuario elegido.

Paso 5– Usuario: El usuario ve el mensaje en mensajes enviados.

Identificador del caso de uso: CU2.16

Nombre: Ver Anotaciones

Actores: Usuario.

Descripción: En la pestaña referente horario el usuario podrá visualizar un horario en el cuál se detallan todas las horas de cada día de la semana laboral, de lunes a viernes, de 09:00 h. a 21:00.

Precondición: El usuario debe haberse registrado anteriormente y autenticado.

Postcondición: Ninguna.

Flujo Principal:

Paso 1– Usuario: El usuario se sitúa en la pestaña de horario.

Paso 2– Sistema: El sistema devolverá el horario correspondiente con el usuario que se ha logueado anteriormente.

Identificador del caso de uso: CU2.17

Nombre: Eliminar Notas

Actores: Usuario.

Descripción: El usuario podrá subir recursos a la aplicación web, tanto exámenes

como apuntes como ejercicios asociándolos a una asignatura. Dichos recursos deben ser aprobados por el administrador antes de añadirlos definitivamente.

Precondición: El usuario debe haberse registrado anteriormente y autenticado. También debe haber buscado la asignatura de la que se quiere añadir el recurso.

Postcondición: Ninguna.

Flujo Principal:

Paso 1– Usuario: Pincha sobre el botón añadir recurso.

Paso 2– Sistema: El sistema pedirá al usuario que se especifique que clase de recurso es y que se suba dicho recurso al sistema

Paso 3– Usuario: El alumno elegirá el tipo de recurso y asociará el fichero adecuado.

Paso 4– Sistema: El sistema enviará dicho recurso al administrador para que autorice su subida al sistema.

Paso 5– Sistema: Una vez se ha validado el recurso el sistema lo asocia a la asignatura elegida por el usuario.

Paso 6 -Usuario: El alumno podrá acceder a dicho recurso.

Flujo alternativo: Archivo con formato incorrecto

Paso 4 – Sistema: El sistema devuelve un mensaje de error.

Identificador del caso de uso: CU2.18

Nombre: Añadir Notas

Actores: Usuario.

Descripción: El usuario puede escribir a algún compañero de facultad que haya buscado previamente.

Precondición: El usuario debe haberse registrado anteriormente y autenticado. También debe haber buscado al compañero con el que quiera comunicarse, Buscar compañero.

Postcondición: Ninguna.

Flujo Principal:

Paso 1– Usuario: El usuario pincha sobre el botón escribir mensaje asociado a otro usuario.

Paso 2– Sistema: El sistema devuelve una caja de texto donde escribir el mensaje.

Paso 3– Usuario: El usuario escribirá el mensaje en la zona desplegada por el sistema y lo enviará.

Paso 4– Sistema: El sistema recibirá el mensaje y lo enviará al usuario elegido.

Paso 5– Usuario: El usuario ve el mensaje en mensajes enviados.

3. Tecnologías empleadas

En este capítulo se muestran las tecnologías que se han usado para desarrollar el proyecto:

Python

Python [22] es un lenguaje de propósito general que se ha utilizado como lenguaje del lado del servidor para implementar el acceso a las bases de datos o las conexiones de red. Se trata de un lenguaje utilizado para scripting y desarrollo ágil de aplicaciones que funciona sobre la mayoría de las plataformas.

Algunas de las características más destacables de Python es que es un lenguaje interpretado, usa tipado dinámico, multiplataforma, y multiparadigma. Se usa para manejar grandes cantidades de datos (Big Data) y en la realización de algoritmos matemáticos de gran complejidad. Admite diferentes paradigmas de programación.

El mantenimiento y desarrollo de este lenguaje es administrado por la Python Software Fundation.

Flask

Flask[20] es un microframenwork para Python, se utiliza en el desarrollo de aplicaciones web. Para crear contenidos dinámicos en HTML se utiliza el motor de templates Jinja2[18]. Como interfaz entre los servidores web y las aplicaciones web se utiliza WSGI de Werkzeug[21]. Posee una licencia BSD, una licencia de software libre permisiva.

Flask se basa en un diseño inicial con una funcionalidad básica, aunque existen un gran número de módulos para aumentar dicha funcionalidad y alcanzar un nivel de desarrollo mucho mayor.

MongoDB

MongoDB [13] pertenece a las bases de datos NoSQL, esta orientada a documentos en lugar de registros. Al contrario de las bases de datos relacionales no sigue un esquema predefinido. La información es almacenada de forma interna como archivos de tipo BSON. Este tipo de archivos es una representación binaria de JSON con información adicional. El valor de un campo puede ser de cualquier tipo de datos BSON, incluyendo

otros documentos, matrices y matrices de documentos. Todos los documentos se guardan en colecciones. Una colección es un conjunto de documentos relacionados, son análogas a las tablas en las bases de datos relacionales. Las características más destacadas de MongoDB son su velocidad y un sistema de consulta flexible.

HTML + CSS3 + JAVASCRIPT

Se trata de tres tecnologías que se utilizan de manera conjunta para crear sitios web, y que cubren los aspectos del maquetado, estilos y funcionalidad.

HTML (Hypertext Markup Language) [9] es un lenguaje de marcado, permite estructurar la información que será interpretada posteriormente por un navegador web. Puede usarse dentro del código otros lenguajes como por ejemplo php o como en el caso de este proyecto Jinja2 que esta incluido dentro del paquete de Flask.

CSS3(Cascading Style Sheets) [4] se utiliza para dar estilo a las páginas web. Con las hojas de estilo es posible especificar diferentes características como los colores, las fuentes y la posición de los elementos. Permiten la separación entre la estructura y el estilo, lo que otorga la posibilidad de que distintos documentos puedan compartir estilos declarados en un archivo css separado del documento HTML. Esto elimina la repetición de código y reduce la complejidad de los documentos.

JavaScript

JavaScript [11] es un lenguaje de programación usado para aportar características interactivas a una página web. Es un lenguaje cuya utilización se orienta principalmente del lado del cliente, utilizando framework como Angular [27] o librerías como React [32], aunque también se puede utilizar del lado de servidor a través de entornos como node.js [31]. Se utiliza prácticamente para construir cualquier herramienta web.

Infinite scroll

Infinite Scroll [10] es una herramienta de JavaScript que agrega automáticamente la página siguiente sin necesidad de volver a cargar la página completa realizando una llamada al servidor.

Es un sistema de paginación que se utiliza para desplegar recursos de forma que se vayan cargando según el usuario desplaza el cursor hacía abajo, de ahí el nombre infinite scroll. Seguirá desplegando páginas hasta llegar al final.
Es una herramienta muy utilizada, por ejemplo se ha utilizado en sitios web como YouTube o Facebook. Se puede obtener bajo licencia de código abierto para proyectos personales.

Google Fonts

Google Fonts [16] es una librería que incluye una gran cantidad de fuentes de estilo diferentes, dispone de más de 850 fuentes de estilos diferentes, liberados bajo licencias de software libres. La librería esta mantenida por Google a través de un repositorio Git [19].

FontAwesone

FontAwesone [8] es un recurso gratuito para añadir iconos estándar a una página web. Es utilizado para mostrar iconos sin necesidad de utilizar otras herramientas adicionales como puede ser JavaScript. Dispone de una lista con un gran número de iconos, los cuáles son fácilmente escalables y se pueden editar mediante CSS. Se disminuye mucho el tiempo de carga en comparación con el uso de imágenes.

4. Arquitectura.

Para diseñar la aplicación se ha utilizado el patrón MVC(Model View Controller)[30]. Este patrón de diseño divide una aplicación en tres partes diferentes que son el modelo, la vista y el controlador. De esta forma se separa los modelos de datos y la lógica de la aplicación de la interfaz de usuario (Figura 8). En este sentido, en la aplicación desarrollada, el modelo está implementado en el archivo models.py. Allí se encuentran todos los modelos que se utilizan en la aplicación como también todos los métodos relacionados con la creación y manipulación de dichos modelos.

Las vistas se lanzan desde Flask renderizando los HTML con sus respectivos css asociados para darles estilo. También cuentan con funciones JavaScript para darle más funcionalidad y dinamismo.

El controlador esta formado por user.py y admin.py, uno para la parte del usuario y otro para la parte de administración.

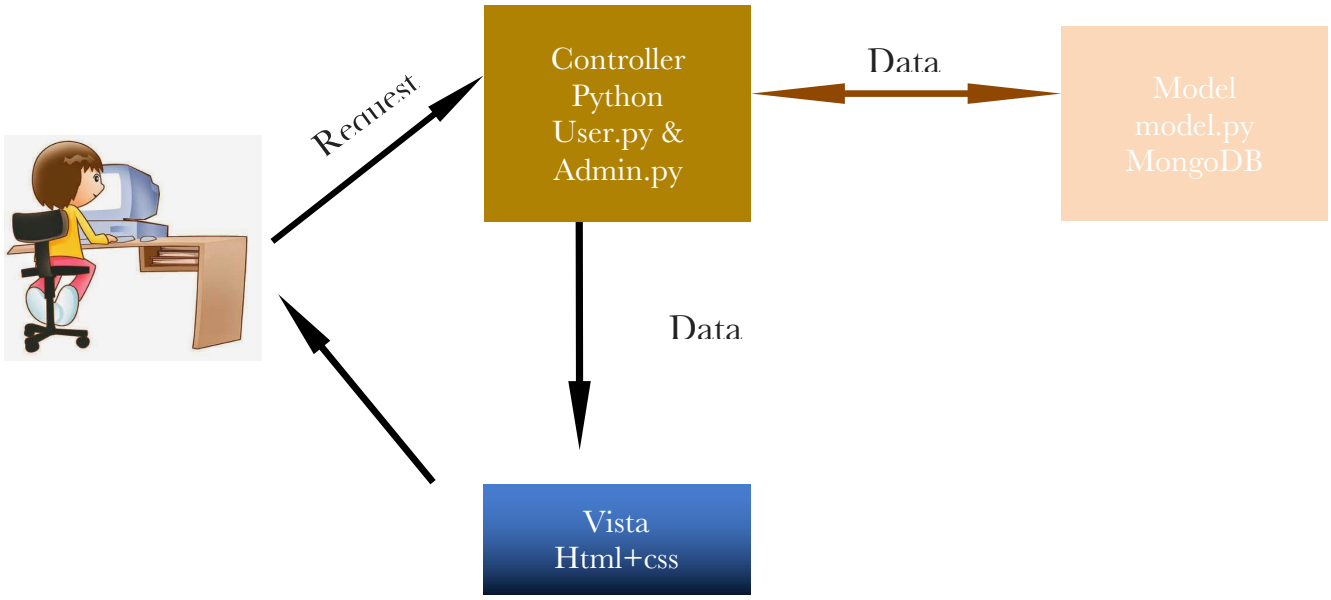


Figura 8. Imagen MVC adaptado al proyecto

Observar que para dividir el proyecto en distintas partes se ha utilizado el módulo de flask blueprints[2] que permite organizar la aplicación en distintos paquetes dependiendo su

funcionalidad. Por lo tanto, el modelo como los templetes están organizados según pertenezca a la parte de administración o a la parte del usuario (Figura 9).

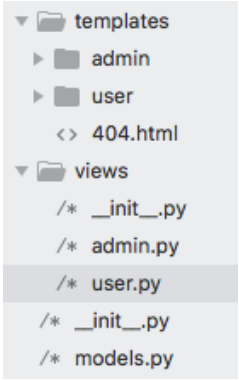


Figura 9: Organización del proyecto en carpetas

5. Modelo de Datos

La persistencia de la información se ha diseñado usando un modelo documental y se ha implementado mediante la base de datos NoSQL MongoDB.

Se han definido las siguientes colecciones (Figura 10): Usuarios, Buzón, Horario, Post, Notas, Facultad, titulación, curso, asignatura, archivo, comentario.

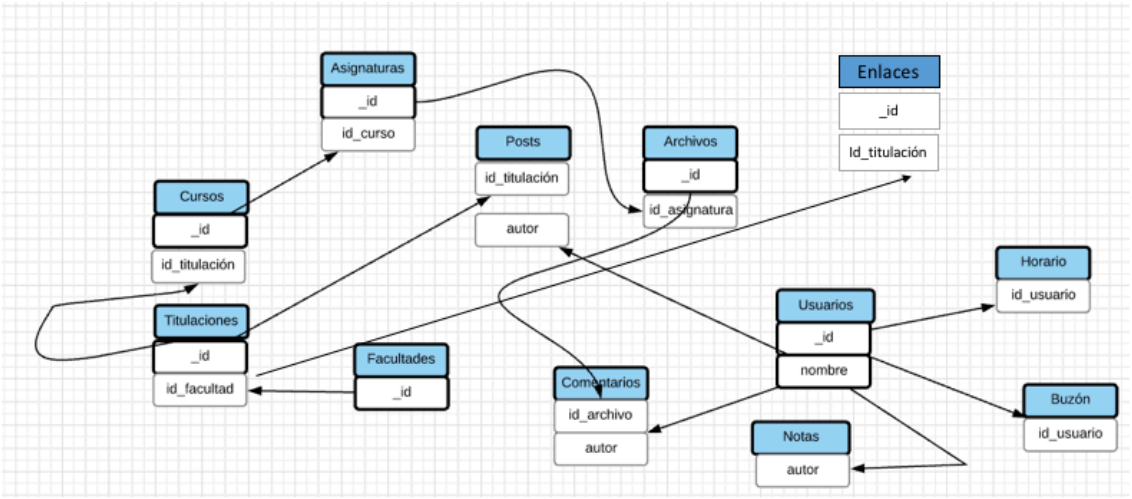


Figura 10: Colecciones definidas.

A continuación, se explicarán con detalle cada una de estas clases, comentando cada uno de sus campos.

Usuarios

Contiene todos los usuarios registrados en la aplicación, tanto usuarios normales como administradores. Los documentos contienen los siguientes campos de información:

nombre => Nombre del usuario
password => Password asociado a la cuenta
role => Puede ser Usuario o Administrador
email => Correo electrónico asociado a la cuenta por parte del usuario
facultad => Facultad a la que pertenece el usuario.
titulación => Titulación dentro de la facultad a la que pertenece el usuario.

archivos_subidos => Nº de archivos subidos a la web app por el usuario.

descargas => Nº de descargas realizadas por el usuario.

activo => Indica si el usuario se encuentra en un estado activo, puede estar bloqueado por parte del administrador si ha realizado alguna actividad no permitida por la web.

foto => Foto o avatar del alumno.

activo_desde => Fecha desde la que el usuario está activo en la web. Si hubiera sufrido alguna sanción por parte del administrador este campo indicaría hasta cuando el usuario se encuentra bloqueado.

comentarios => Nº de comentarios realizados por el usuario.

posts => Nº de post realizados por el alumno.

no_leídos => Si tiene mensajes no leídos.

Ejemplo de un usuario de la aplicación (Figura 11).

```
{u'_id': ObjectId('5a2b3e7c2b808529bebee09b'),
 u'activo': True,
 u'activo_desde': u'2018-01-04',
 u'archivos_subidos': 10,
 u'comentarios': 10,
 u'descargas': 9,
 u'email': u'pepe@ucm.es',
 u'facultad': u'Informatica',
 u'foto': u'ricknillo.jpg',
 u'horario': {u'AC': {u'10': [u'Jueves'], u'11': [u'Viernes']},
  u'DSI': {u'12': [u'Lunes', u'Martes']},
  u'RS': {u'18': [u'Mi\xe9rcoles', u'Jueves']},
  u'TOC': {u'10': [u'Lunes', u'Martes'], u'9': [u'Mi\xe9rcoles']}},
 u'no_leidos': 0,
 u'nombre': u'Pepe',
 u'password': u'Lukas',
 u'posts': 5,
 u'role': u'Usuario',
 u'titulacion': u'Ingenieria Informatica'}
```

Figura 11: Colección usuarios.

Buzón

Almacena todos los mensajes que se mandan unos usuarios a otros dentro de la aplicación. Los documentos contienen los siguientes campos de información:

asunto => 'asunto'

id_usuario => Identidad del usuario autor

autor => Nombre del usuario autor.

autor_avatar => Avatar del autor.

body => Cuerpo del mensaje

fecha => Fecha de creación del mensaje

estado => Leído o no leído

denuncia => True o False dependiendo si existe una denuncia por parte de otro usuario.

Ejemplo de un buzón de la aplicación (Figura 12).

```
{u'_id': ObjectId('5a98f8232b808511f2808cef'),
 u'asunto': u'HolaMundo',
 u'autor': u'sdsd',
 u'autor_avatar': u'404.jpg',
 u'body': u'hola caracola.',
 u'denuncia': False,
 u'estado': u'Le\xeddos',
 u'fecha': u'02/03/18',
 u'id_usuario': ObjectId('5a2b3e7c2b808529bebee09b')}
```

Figura 12: Colección buzón

Horario

Almacena la información sobre una hora concreta durante todos los días de la semana Representa el horario de un usuario. Los documentos contienen los siguientes campos de información:

hora => Hora del día

l => Lunes

m => Martes

x => Miércoles

j=> Jueves

v => Viernes

id_usuario => Identidad del usuario al cual pertenece esta franja horaria.

Ejemplo de un hora de la aplicación (Figura 13).

```
{u'_id': ObjectId('5aac8f0b2b80854dd092c5e3'),
 u'hora': 9,
 u'id_usuario': ObjectId('5a2b3e7c2b808529bebee09b'),
 u'j': u'',
 u'l': u'',
 u'm': u'',
 u'v': u'',
 u'x': u'TOC'}
```

Figura 13: Colección horario

Posts

Almacena la información sobre cada post que se publica dentro de la web. Los documentos contienen los siguientes campos de información:

tema => Tema del post

titulo => Título del post.

id_titulacion => Identidad de la titulación a la que pertenece el post.

likes => Nº de likes recibidos.

unlikes => Nº de dislikes recibidos.

autor => Autor del post

autor_avatar = > Avatar del autor. Se guarda aquí para no tener que realizar la búsqueda adicional.

body => Cuerpo del post.

fecha => Fecha de creación.

denuncia => True/ False dependiendo de si ha sido denunciado por otro usuario.

Ejemplo de un post de la aplicación (Figura 14).

```
{u'_id': ObjectId('5a94f7252b80850cb9b5566a'),
 u'autor': u'sdsd',
 u'autor_avatar': u'404.jpg',
 u'body': u'< Descripcion de la tarea >',
 u'denuncia': False,
 u'fecha': u'27/02/18',
 u'id_titulacion': ObjectId('5a2b11362b8085280c24c23c'),
 u'likes': 1,
 u'tema': u'Examen',
 u'titulo': u'Hola',
 u'unlikes': 0}
```

Figura 14: Colección posts

Notas

Almacena información relativa a la parte de anotaciones dentro de las secciones de cada usuario. Los documentos contienen los siguientes campos de información:

titulo => Título de la nota.

fecha => Fecha de creación de la nota por parte del usuario.

descripción => Descripción sobre el contenido de la nota.

asunto => Asunto de la nota.

autor => Nombre del usuario que creo la nota.

formato => Fecha de la creación en un formato para su ordenación dentro de las notas.

Ejemplo de una nota de la aplicación (Figura 15).


```
{u'_id': ObjectId('5a4e16f52b8085550dd5b7e9'),
 u'asunto': u'Examen',
 u'autor': u'Pepe',
 u'descripcion': u'< Descripcion de la tarea >',
 u'fecha': u'22/01/2018',
 u'formato': u'2018/01/22',
 u'titulo': u'ss'}
```


Figura 15: Colección notas

Facultades

Almacena información sobre las facultades que gestiona la aplicación. Los documentos contienen los siguientes campos de información:

nombre => Nombre de la facultad.

titulaciones => Array con todas las titulaciones que se ofertan dentro de esa universidad.

num_titul => Numero de las titulaciones.

cursos_x_titul => Array con todos los cursos por titulación.

Ejemplo de una facultad de la aplicación (Figura 16).


```
{u'_id': ObjectId('5a2b107e2b8085280c24c23b'),
 u'cursos_x_titul': [4, 4],
 u'nombre': u'Informatica',
 u'num_titulaciones': 2,
 u'titulaciones': [u'Ingenieria Informatica', u'Ingenieria del Software']}
```


Figura 16: Colección facultades

Titulaciones

Almacena información de todas las titulaciones gestionadas por la aplicación. Los documentos contienen los siguientes campos de información:

nombre => Nombre de las titulaciones

id_facultad => Identidad de la facultad a la que pertenece.

num_cursos => Número de cursos de la titulación.

asignaturas => Array con todas las asignaturas distribuidas por cursos.

Ejemplo de una titulación de la aplicación (Figura 17).

```
{u'_id': ObjectId('5a2b11362b8085280c24c23c'),
 u'asignaturas': [[u'Gestion Empresarial',
   u'Fundamentos de electricidad',
   u'Metodos Matematicos Ingenieria',
   u'Matematica Discreta',
   u'Fundamentos de Programacion',
   u'Fundamentos de Computadores'],
  [u'Estructura de Computadores',
   u'Estructura de Datos y Algoritmos',
   u'Tecnologia de Programacion',
   u'Ingenieria del Software',
   u'Bases de datos',
   u'TOC',
   u'Ampliacion de Matematicas',
   u'Probabilidad y Estadistica'],
  [u'Sistemas Operativos', u'Redes'],
  [u'ELP', u'Arquitectura de Computadores', u'ASOR']],
 u'cursos': 4,
 u'id_facultad': ObjectId('5a2b107e2b8085280c24c23b'),
 u'nombre': u'Ingenieria Informatica'}
```

Figura 17: Colección titulaciones

Cursos

Almacena información de cada uno de los cursos de las titulaciones gestionadas por la aplicación. Los documentos contienen los siguientes campos de información:

numero => Número/nombre del curso dentro de la titulación.

id_titulacion => Identidad de la titulación a la que pertenece.

num_asignaturas => Número de asignaturas del curso.

asignaturas => Nombre de cada una de las asignaturas.

Ejemplo de un curso de la aplicación (Figura 18).

```
{u'_id': ObjectId('5a2b11362b8085280c24c23d'),
 u'asignaturas': [u'Gestion Empresarial',
  u'Fundamentos de electricidad',
  u'Metodos Matematicos Ingenieria',
  u'Matematica Discreta',
  u'Fundamentos de Programacion',
  u'Fundamentos de Computadores'],
 u'id_titulacion': ObjectId('5a2b11362b8085280c24c23c'),
 u'num_asignaturas': 6,
 u'numero': 1}
```

Figura 18: Colección cursos

Asignaturas

Esta colección almacena información sobre las asignaturas gestionadas por la aplicación. Los documentos contienen los siguientes campos de información:

nombre => nombre de la asignatura.

id_curso => Identidad del curso al que pertenece.

num_ficheros => Número de ficheros asociados a esa asignatura.

nombre_ficheros => Array con el nombre de cada uno de los archivos asociados a la asignatura.

Ejemplo de una asignatura de la aplicación (Figura 19).

```
{u'_id': ObjectId('5a2b11362b8085280c24c23e'),
 u'id_curso': ObjectId('5a2b11362b8085280c24c23d'),
 u'nombre': u'Gestion Empresarial',
 u'nombre_ficheros': [],
 u'num_ficheros': 0}
```

Figura 19: Colección asignaturas

Archivos

Almacena información sobre todos los archivos gestionados por la aplicación. Los documentos contienen los siguientes campos de información:

nombre => nombre del archivo

id_asignatura => Identidad de la asignatura a la que pertenece el archivo.

asignatura => Nombre de la asignatura a la que pertenece.

valoracion => Valoración del archivo por parte de los usuarios.

valor_dec => Valor decimal de la puntuación.

valor_entero =>Valor entero de la valoración.

sum_valor => Suma de todas las valoraciones de los usuarios.

autor => Autor de la subida del archivo.

num_descargas => Número de descargas de este fichero.

num_votos =>Número de votos realizados en el archivo por parte de los usuarios.

autorizado => True o False si está autorizado por parte del administrador.

tipo => Tipo de archivo.

fichero => Archivo en binario guardado en la bb.dd.

num_comen => Número de comentarios que se han realizado en el archivo por parte de los usuarios.

id_titulacion => Identidad de la titulación a la que pertenece el archivo.

titulo => Título de archivo.

fecha => Fecha de creación del archivo.

formato => Tipo de archivo pdf, png,doc etc…

descripcion => Descripción del fichero

Ejemplo de un archivo de la aplicación (Figura 20).


```
{u'_id': ObjectId('5a4716212b808542f8164e13'),
 u'asignatura': u'Arquitectura de Computadores',
 u'autor': u'Pepe',
 u'autorizado': True,
 u'fecha': u'10/12/17',
 u'formato': u'17/12/10',
 u'id_asignatura': ObjectId('5a2b11362b8085280c24c252'),
 u'id_titulacion': ObjectId('5a2b11362b8085280c24c23c'),
 u'nombre': u'AC_tema1_2017-18.pdf',
 u'num_comen': 0,
 u'num_descargas': 12,
 u'numero_votos': 4,
 u'sum_valor': 14,
 u'tipo': u'Apuntes',
 u'valor_dec': 0.5,
 u'valor_entero': 3,
 u'valoracion': 3.5}
```


Figura 20: Colección archivos

Comentarios

Almacena información sobre los comentarios realizados por los usuarios de la aplicación.

Los documentos contienen los siguientes campos de información:

titulo => Título del comentario.

id_archivo => Identidad del archivo al que pertenece el comentario.

likes => Número de likes otorgados por los usuarios.

unlike => Número de dislikes dados por parte de los usuarios.

autor => Autor del comentario

autor_avatar => Avatar del autor

body => Cuerpo del comentario.

fecha => Fecha de creación del comentario.

denuncia => True o False si algún usuario ha denunciado el comentario.

Ejemplo de un comentario de la aplicación (Figura 21).

```
{u'_id': ObjectId('5a94f13b2b80850caa02a358'),
 u'autor': u'Pepe',
 u'autor_avatar': u'ricknillo.jpg',
 u'body': u'Hola',
 u'denuncia': False,
 u'fecha': u'Tue Feb 27 06:48:42 2018',
 u'id_archivo': ObjectId('5a4ad7102b80854b2dc40998'),
 u'likes': 0,
 u'titulo': u'HolaMundo',
 u'unlike': 0}
```


Figura 21: Colección comentarios

Enlaces

Almacena información sobre los enlaces de la aplicación. Los documentos contienen los siguientes campos de información:

nombre => Nombre del enlace.

id_titulación => Identidad de la titulación a la que pertenece el enlace.

enlace => Dirección web del sitio al cuál redirecciona el enlace.

Ejemplo de un enlace de la aplicación (Figura 22).


```
{u'_id': ObjectId('5b8c59f21ae6bb357a37fa94'),
 u'enlace': u'https://www.youtube.com/user/finformaticaucm',
 u'id_titulacion': ObjectId('5a2b11362b8085280c24c23c'),
 u'nombre': u'Fdi en Youtube'}
```


Figura 22: Colección enlaces

6. Diseño

En este capítulo se van a describir todos los aspectos referidos al diseño de la aplicación. En primer lugar, para desarrollar la aplicación se ha tenido en cuenta la usabilidad de la misma. En este sentido:

- La aplicación es rápida, las páginas tienen un tiempo de carga reducido y las operaciones que interactúan con el servidor consumen una cantidad de tiempo pequeña. De esta forma, el usuario no debe esperar para utilizar la aplicación, lo que incrementa la experiencia satisfactoria del usuario.
- Se ha buscado la simpleza en el diseño, lo que provoca que el tiempo de aprendizaje para utilizar la aplicación sea reducido. La aplicación tiene una interfaz fácil de entender, con elementos muy diferenciados para que el usuario mantenga una experiencia ágil y constante, sin tener que detenerse a entender algo.
- El código implementado está basado en HTML+CSS y utiliza una versión de JavaScript estable, por lo que no existen problemas en interpretar la aplicación para ningún explorador web, lo que la hace más accesible al ser compatible con todos los navegadores aunque no estén actualizados.

Se han aplicado los principios de diseño de Nielsen[36]:

- Visibilidad del estado del sistema. El sistema debe mantener al usuario siempre informado del estado actual y proporcionar información acerca de las acciones que realice dentro de la aplicación y los cambios que provoca.
- Relación entre el sistema y el mundo real (Metáforas y lenguaje familiares). Se ha utilizado un lenguaje sencillo y términos cercanos para facilitar el manejo al usuario que va a utilizar la aplicación, en este caso alumnos universitarios.
- Control y libertad del usuario. Cualquier acción que el usuario lleve a cabo debe ser reversible, dando la opción al usuario de recuperar su estado anterior. Esto aporta libertad al usuario a la hora de manejar la aplicación, sin temor a perder la información actual o realizar cambios permanentes.
- Reconocimiento mejor que recuerdo. Intentar que el usuario disponga de toda la información que necesita en cada momento, sin tener la obligación de desplazarse a otras páginas de la aplicación para recordar algo que ha sucedido anteriormente o el estado actual del usuario.

- Estética y diseño minimalista. Los textos e iconos que aparezcan en la aplicación deben ser familiares y sencillos, para que el usuario no se pierda en determinadas partes o no sea capaz de entender de forma correcta el funcionamiento de la aplicación.
- Ayudar al usuario a reconocer, diagnosticar y recuperarse de errores. Los errores deben arrojar mensajes informativos, los cuáles deben ser explícitos y utilizar un lenguaje sencillo, no deben incluir códigos o expresiones técnicas que el usuario no pueda comprender.

Para el diseño de la interface de usuario se ha utilizado la librería FontAwesone como ya se ha descrito en la sección de Tecnología empleada. Todos los iconos que aparecen en las vistas han sido importados de esta librería. Estos son algunos ejemplos (Figura 23).

Figura 23. Iconos FontAwesone utilizados en la aplicación

La fuente principal utilizada para esta parte de la aplicación ha sido Oswald, importada de GoogleFonts. También se han utilizado otras fuentes como ReenieBeanie o Montserrat Alternates para la sección del horario.

El diseño de la interface de usuario tiene un header con un menú en la parte derecha para cerrar sesión, acceder al perfil y volver a la página de inicio. Después la mayoría de las páginas poseen un menú para navegar entre las distintas opciones de la aplicación.

7. Implementación

En este capítulo se va a describir la implementación realizada. Debido a que la aplicación está orientada a recopilar recursos e información tendrá mucha relevancia las operaciones de cargar y guardar datos en la base datos. Proliferan los métodos implementados a través de funciones AJAX que suministran información al usuario de forma asíncrona, aumentando la velocidad de la aplicación. Por otra parte, los procedimientos de autenticación y registro también son importantes ya que todas las funcionalidades de la aplicación requieren que el usuario esté registrado en la aplicación con anterioridad y que se haya identificado correctamente en la misma.

Primero se muestran los diagramas de clases para comprender mejor el funcionamiento de la aplicación. Cada diagrama esta dividido en tres secciones: modelo, vista y controlador. La parte del modelo representa los diferentes modelos de datos presentes en la aplicación, la parte del controlador son los métodos que se utilizan para manejar dichos modelos de datos, los cuáles realizan cálculos, conectan con la base de datos y sirven información a las diferentes vistas. Las vistas son las plantillas html implementadas, permiten al usuario interactuar con la información y realizar acciones para manejar y utilizar los datos proporcionados.

Se dividen en dos partes, la primera la parte del usuario (Figura 24, Figura 25, Figura 26) y la segunda la parte del administrador (Figura 27, Figura 28, Figura 29).

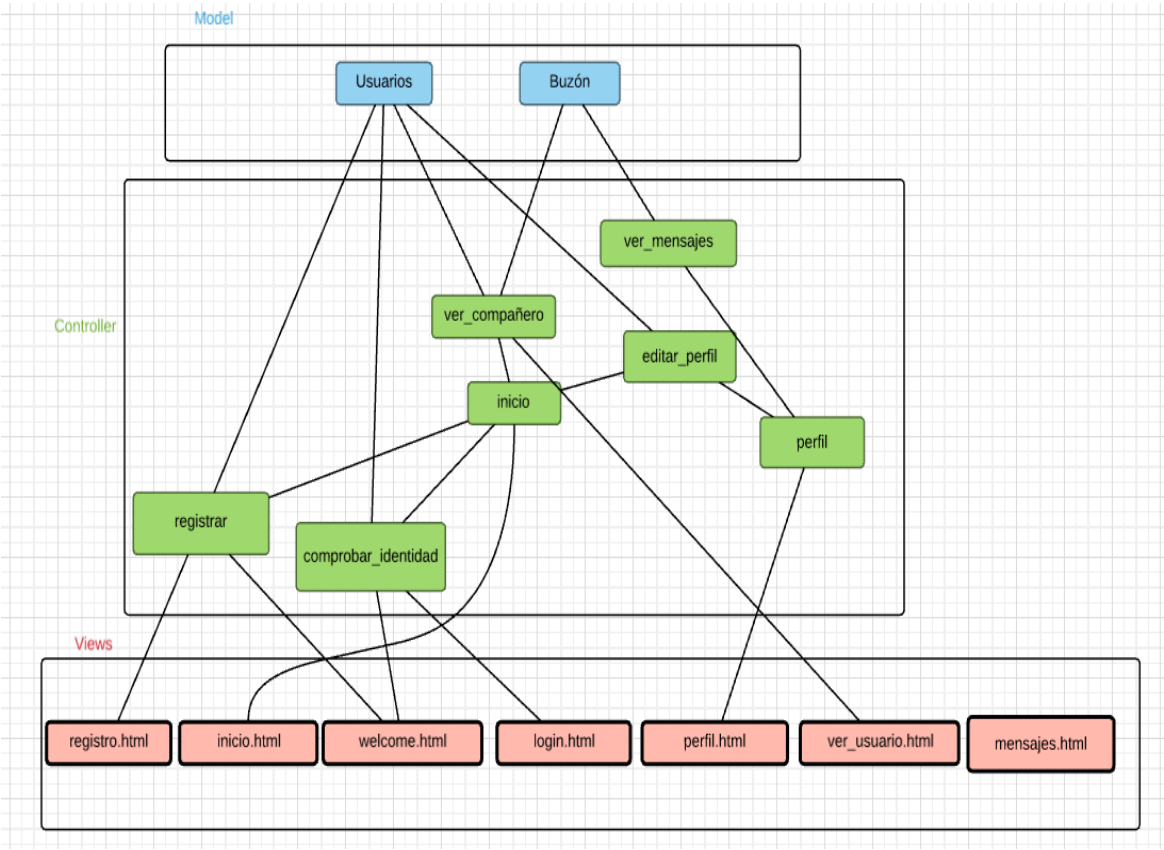


Figura 24. Diagrama de clases Usuario_1

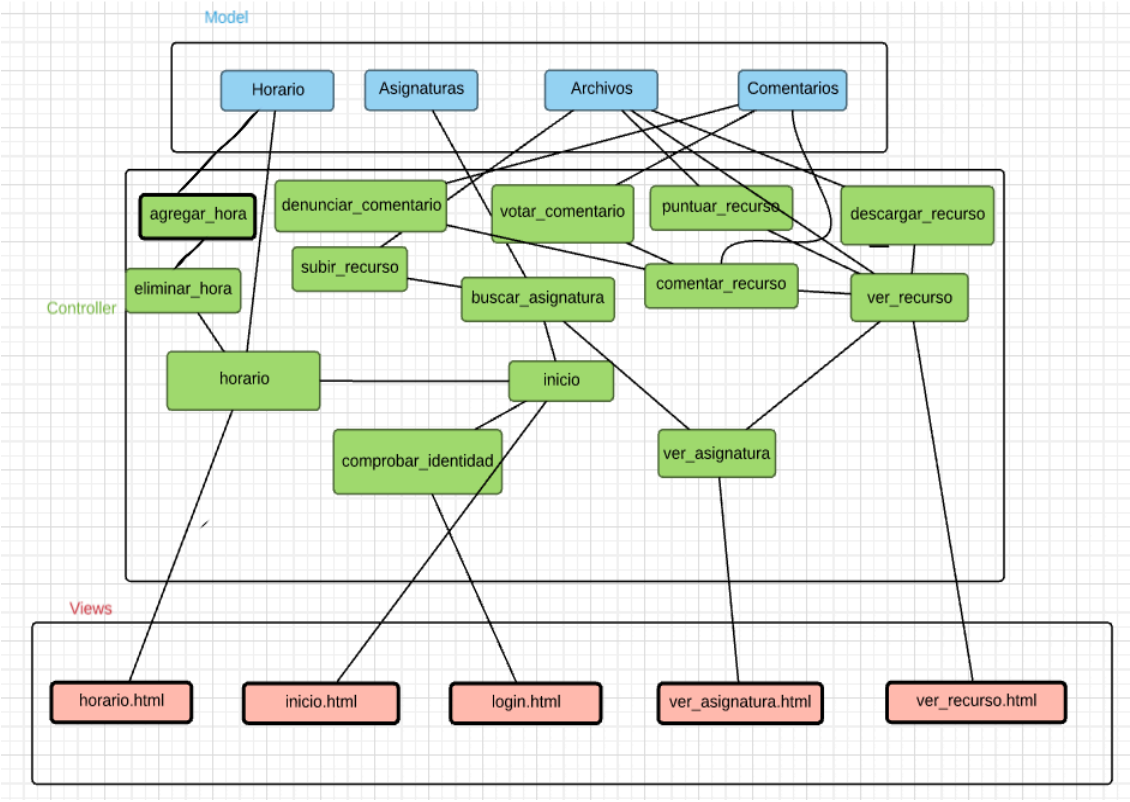


Figura 25. Diagrama de clases Usuario_2

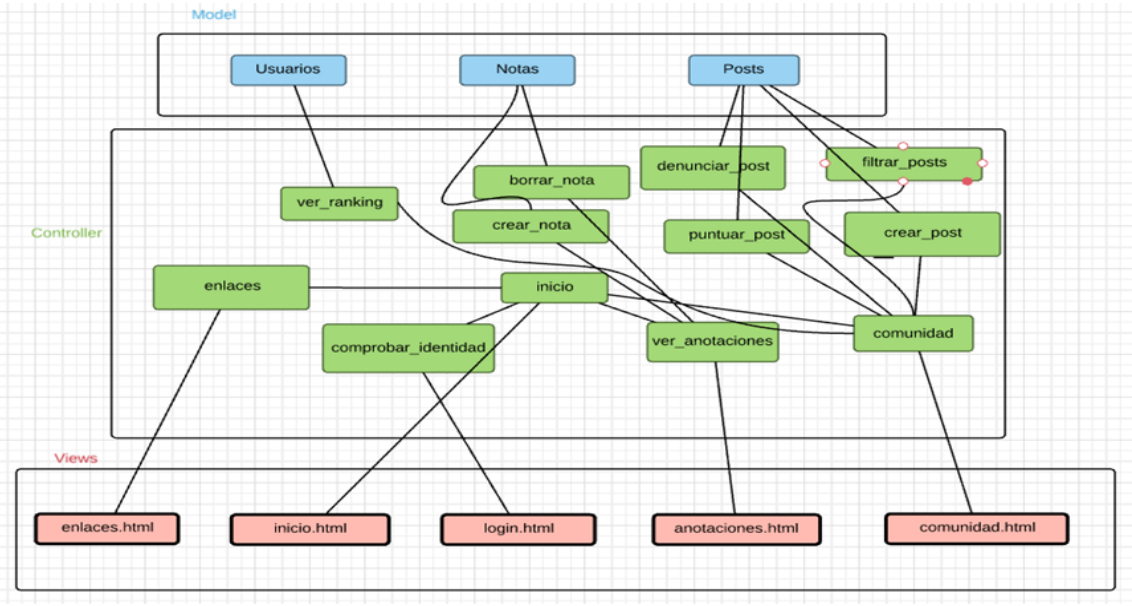


Figura 26. Diagrama de clases Usuario_3

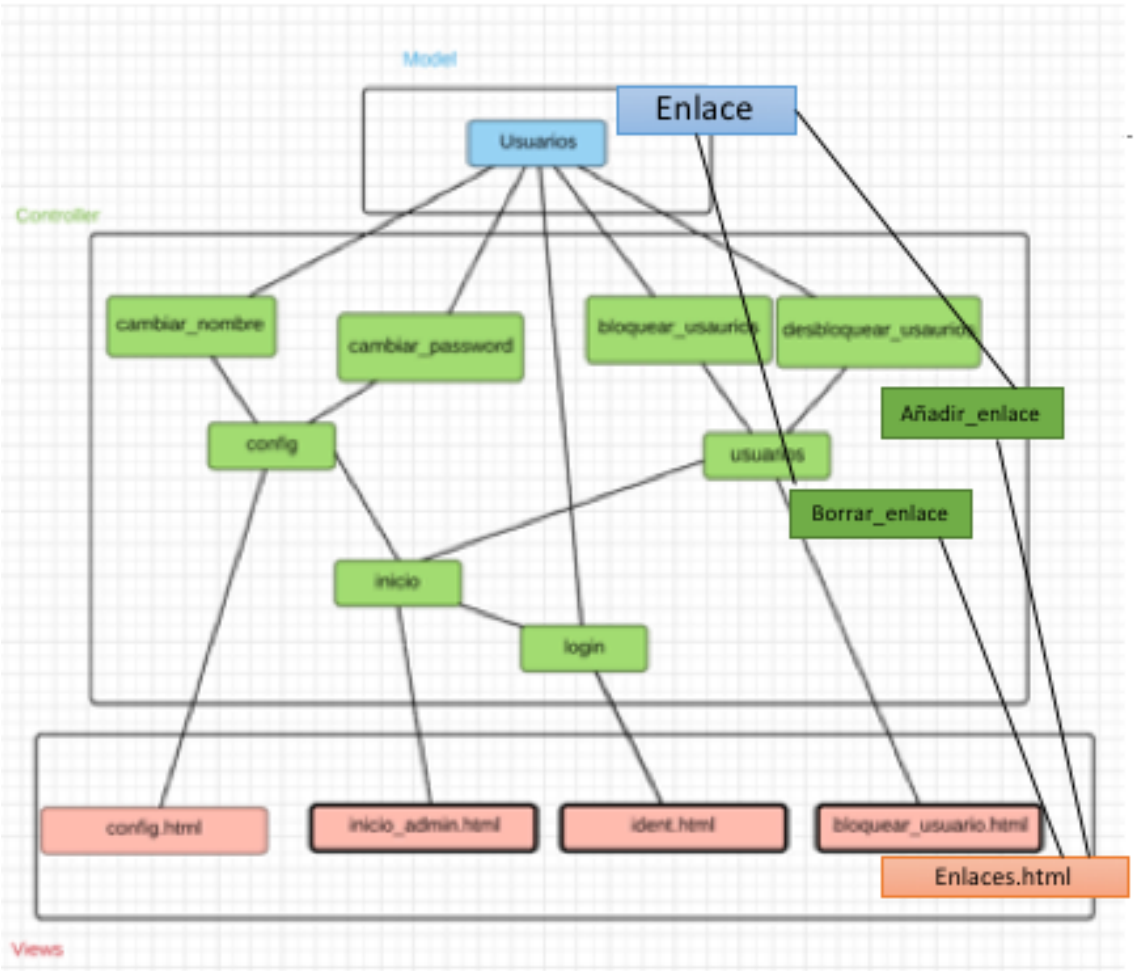


Figura 27. Diagrama de clases Admnistrador_1

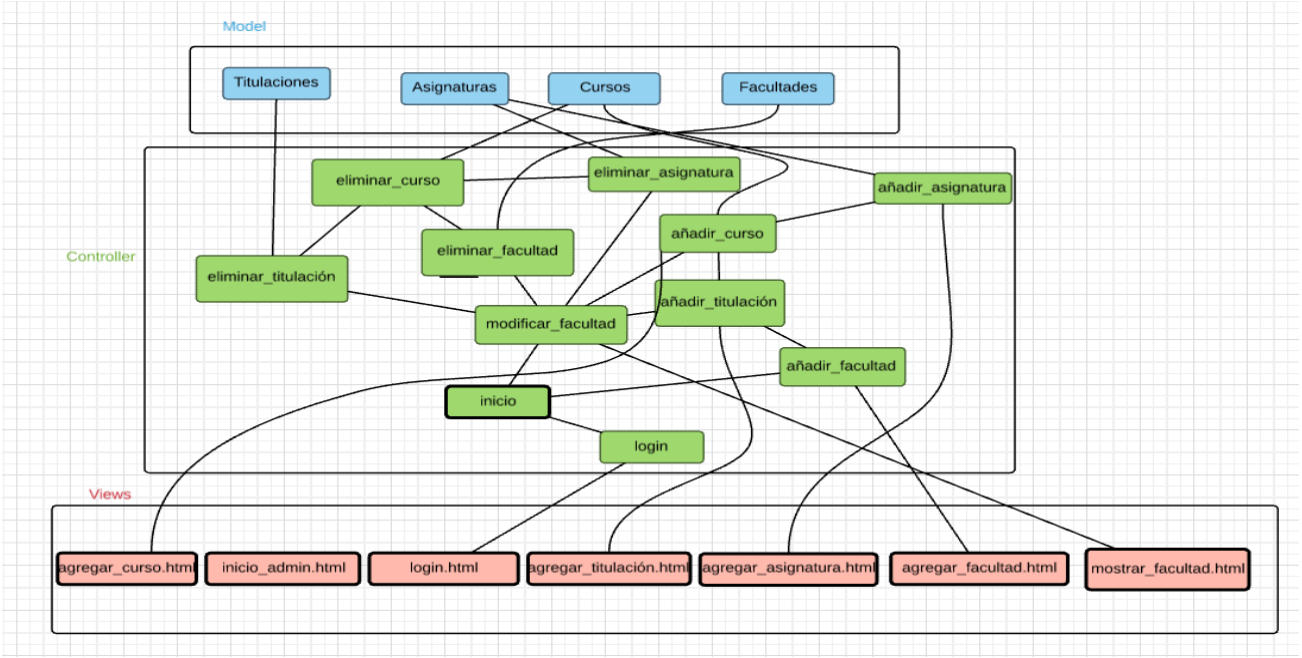


Figura 28. Diagrama de clases Admnistrador_2

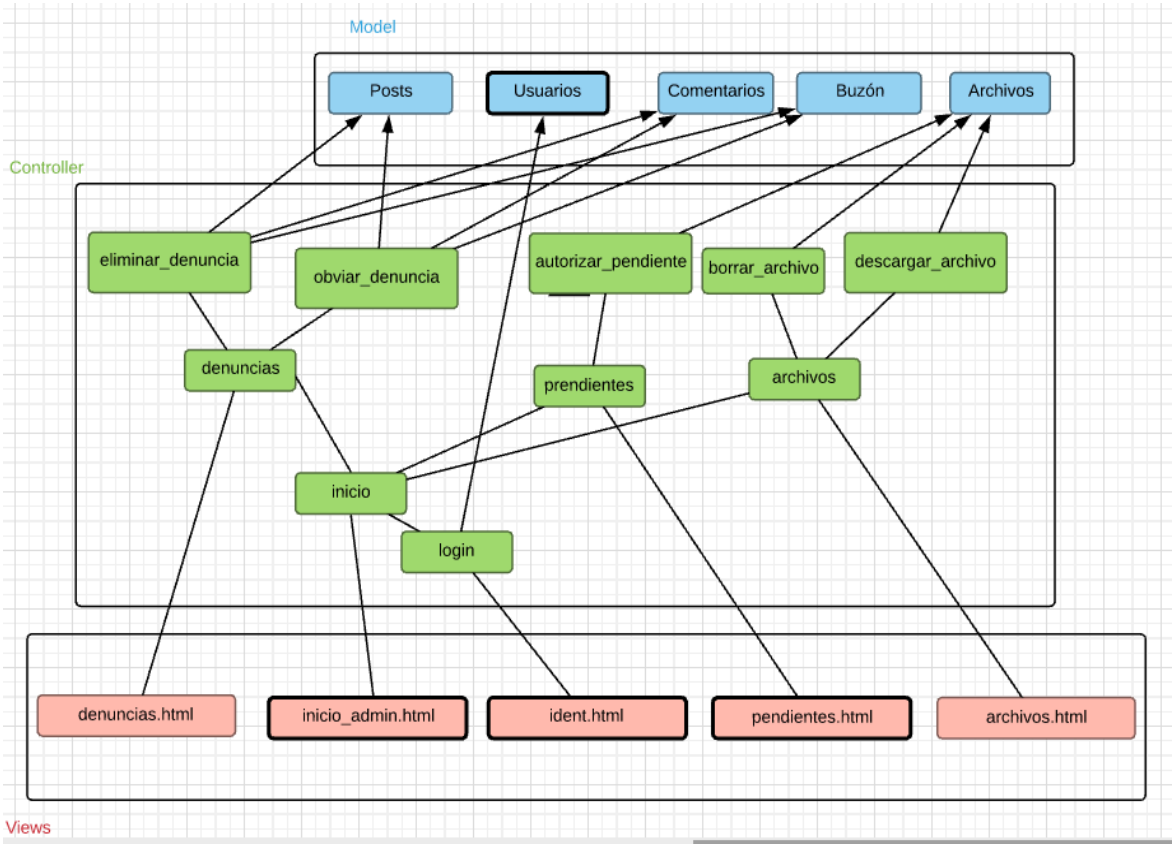


Figura 29. Diagrama de clases Administrador_3

Landing Page

La página de bienvenida está dividida en dos partes(Figura 30),en la parte izquierda se dispone el nombre de la aplicación junto a los diferentes reclamos para utilizar la aplicación, después en la parte derecha se muestran dos botones dispuestos en una columna central: “Identifícate”, para realizar la autenticación por parte de un usuario ya registrado y otro para realizar un registro por parte de un usuario que todavía no se ha registrado en la aplicación que dice “Regístrate”.

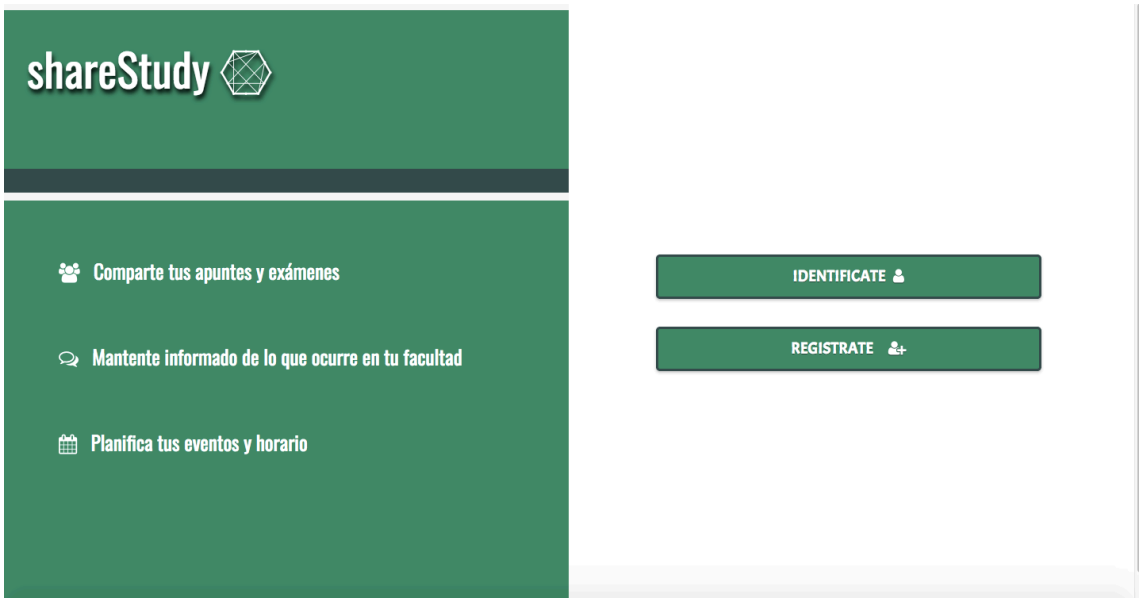


Figura 30. Página de lanzamiento

Header

El Header (Figura 31) está presente en la zona superior de todas las páginas excepto de la landing page. Permite el acceso a la pantalla inicial, cerrar sesión y acceder al perfil del usuario activo. En la zona izquierda está el nombre de la aplicación que lleva a la landing page. Si el usuario tiene algún mensaje sin leer en su buzón aparecerá un círculo rojo con un número de color blanco, que indica el número de mensajes sin leer.

Figura 31. Header

Menú de la aplicación

El menú de la aplicación es una lista de enlaces que permiten acceder a todas las opciones de la aplicación. Es un menú horizontal como se muestra en la figura 32.

Figura 32. Menú principal

Login

La página de login está formada por un formulario, el cuál se puede observar en la figura 33. En la cabecera del formulario se expone una pequeña descripción de la acción a realizar para completar el login. Tiene dos campos uno para el nombre y otro para el password. Ambos disponen de una ayuda visual para saber que se debe escribir.

Una vez estén completos ambos campos el formulario presenta un botón en la parte inferior que completará la autenticación. Si falla la validación de alguno de los campos de la aplicación devolverá un mensaje detallando cuál ha sido el problema.

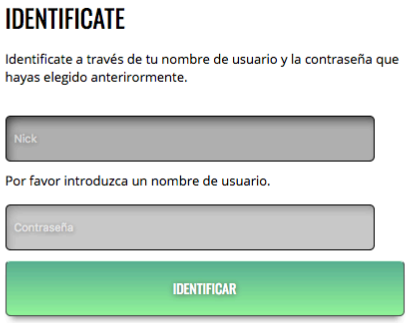


Figura 33. Login

El login recoge el nombre y el password introducidos en el formulario y los envía a la aplicación, donde se comprueba que dichos datos están almacenados en la colección Usuario.

Si los datos son validados, se redirecciona al usuario dependiendo del rol que tenga: si es **admin** se le envía a la parte de administración y si es **user** a la parte para usuarios.

Como se aprecia en la figura 34, también se crea un *login_user* con la información del usuario creado para iniciar sesión y dar acceso al usuario a las páginas que le correspondan según su rol.

```
probar_identidad():
request.method == 'POST':
 logout_user()
 values = {}
 values["nombre"] = request.form['nombre']
 values["password"] = request.form['password']
 d = datetime.date.today()
 date = d.isoformat()
 with Mongo:
     #usuario = Usuario.find_one({"nombre" : "Daniel"})
     usuario = Usuario.find_one({Usuario.nombre : values["nombre"]})
     if usuario is None:
         return render_template("user/ident_fallida.html" , motivo = "Nombre de Usuario")
     else:
         if values["password"] == usuario.password and usuario.role == "Admin":
             user = User(str(usuario._id),values["nombre"],usuario.role)
             login_user(user)
             return redirect(url_for('admin.admin_inicio',nombre = usuario.nombre))
         elif values["password"] == usuario.password and usuario.role == "Usuario" and usuario.activo_desde<=
             Usuario.update({Usuario._id : usuario._id },{'$set': {Usuario.activo: True}})
             t = Titulacion.find_one({Titulacion.nombre : usuario.titulacion})
             cursos = Curso.find({Curso.id_titulacion:t._id})
             user = User( str(usuario._id),values["nombre"],usuario.role)
             login_user(user)
             resultados = Archivo.find({Archivo.id_titulacion: t._id}).sort("fecha",-1).limit(3)
             return render_template("user/inicio.html", usuario = usuario, cursos = cursos , busqueda = False
         else:
             return render_template("user/ident_fallida.html", motivo="Password")
```

Figura 34. Comprobar identidad

A través de un decorador(Figura 35) se discrimina por el rol al usuario y le permite ver unas páginas u otras. En la figura se puede observar la implementación de la función.

```
def required_roles(*roles):
    def wrapper(f):
        @wraps(f)
        def wrapped(*args, **kwargs):
            if get_current_user_role() not in roles:
                flash('Authentication error, please check your details and try again','error')
                return redirect(url_for('identificacion'))
            return f(*args, **kwargs)
        return wrapped
    return wrapper
```

Figura 35. Comprobar identidad

A continuación, en la figura 36 se muestra como se utiliza el decorador para acceder a urls concretas dentro de la aplicación.

```
@user.route('/usuario/inicio/volver/<string:nombre>', methods = ["GET","POST"])
@login_required
@required_roles('Usuario')
def usuario_volver_inicio(nombre):
```

Figura 36. Comprobar identidad

Registro

Sirve para introducir nuevos usuarios en la bb.dd. Esta formada por dos pantallas. En la primera de ellas se debe elegir la universidad de la que se forma parte a través de un campo de tipo select, después se debe elegir la titulación, el nombre, password, email, y una foto o avatar. Una vez introducidos los datos, el usuario debe pulsar el botón verde de la parte inferior del formulario que dice "Registrar" (Figura 37).

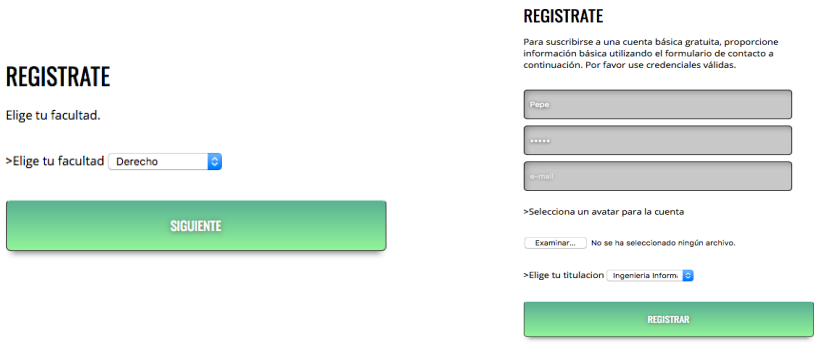


Figura 37. Registro de Usuario

El registro utiliza dos formularios para recoger los datos y enviarlo a la parte del servidor en forma de petición post. El primero solicita el nombre de la facultad, una vez seleccionada se envía al servidor que carga todas las titulaciones que tiene esa facultad y las devuelve en otro formulario, en el cuál se deben completar todos los datos restantes para completar el registro. Una vez enviados son guardados según el modelo de datos y almacenados en la bb.dd dentro de la colección Usuario.

La parte más complicada es decidir como guardar la imagen del avatar dentro de la base de datos. Puesto que es necesario desplegar la imagen de nuevo en páginas HTML, se ha optado por guardar el nombre de la foto en un campo de la collección Usuario, y después guardar la imagen propiamente dicha en la carpeta static para luego tener acceso a ella. Dicha carpeta está almacenada en al constante UPLOAD_FOLDER (Figura 38).

```
if file and allowed_file(file.filename):
    filename = secure_filename(file.filename)
usuario = Usuario()
usuario.crearUsuario(filename,request.form['nombre'],request.form['password'],request.form['email'],request.f
file.save(os.path.join(current_app.config['UPLOAD_FOLDER'], filename))
```

Figura 38. Guardar foto usuario

Búsqueda de asignaturas

Es la página predeterminada que aparece al finalizar la autenticación. En esta página se pueden buscar asignaturas de la titulación del usuario activo (Figura 39).

Figura 39. Vista inicio

Se pueden filtrar asignaturas tanto por curso como por el nombre. Ambas funcionalidades están implementadas con funciones AJAX, que devuelven la información al panel inferior de forma asíncrona. Dichas funciones recogen la información y la devuelven una vez se ha realizado la búsqueda en la base de datos según la información recogida. En las figuras 40 y 41 se puede observar la implementación de la función AJAX y como devuelve la información por parte del servidor.

```
$(document).ready(function(){

        var $SCRIPT_ROOT = {{ request.script_root|tojson|safe }};

        $("#cursos").change(function(){
            $.getJSON('/usuario/inicio/curso/_search',{
                    busqueda: $('#cursos').val() , name: $('#user_name').val(),
                },function(data){

        $(".busqueda").html(data.result);


            });

        });
        $("#busqueda_nombre").keyup(function(){
            $.getJSON('/usuario/inicio/nombre/_search',{
                    busqueda: $('#busqueda_nombre').val() , name: $('#user_name').val(),
                },function(data){

        $(".busqueda").html(data.result);


            });
```

Figura 40. Método AJAX

```
@user.route('/usuario/inicio/curso/_search', methods = ["GET"])
@login_required
@required_roles('Usuario')
def usuario_busqueda_curso():
    if request.method == 'GET':
        resultados = []
        curso = request.args.get('busqueda')
        nombre = request.args.get('name')
        with Mongo:
            asignaturas = Asignatura.find({Asignatura.id_curso: ObjectId(curso)}).sort("nombre")
        textosalida = ""
        for a in asignaturas:
            textosalida += '''<a href= /usuario/ver_asignatura/''' +str(a._id)+'''/''' + (nombre).encode('utf-8

        return jsonify(result=textosalida)
```

Figura 41. Buscar asignaturas

Mostrar últimos recursos

En la sección de recursos, en la parte derecha se muestran los tres últimos recursos subidos en esa titulación. Para mostrar esa información se realiza una búsqueda ordenada por la fecha de creación, se ordena de mayor a menor y se limita la búsqueda a tres (Figura 42).

```
resultados = Archivo.find({Archivo.id_titulacion: t._id}).sort("formato",-1).limit(3)
```

Figura 42. Buscar últimos recursos subidos

Subir recurso

La parte más difícil de subir un archivo a la aplicación es guardarlo posteriormente en la base de datos. Para completar esta acción se transforma la imagen a bytes para poder guardarlo en un campo de la colección, en este caso se llama *filebody* (Figura 43).

```
if file and allowed_file(file.filename):
    filename = secure_filename(file.filename)
    content = StringIO(file.read())
    filebody = Binary(content.getvalue())
```

Figura 43. Guardar fichero

La vista de esta página está dividida en tres zonas: la primera un pequeño resumen de estadísticas de la asignatura, después las tablas con los diferentes tipos de recursos y por último la zona de subir recursos (Figura 44).

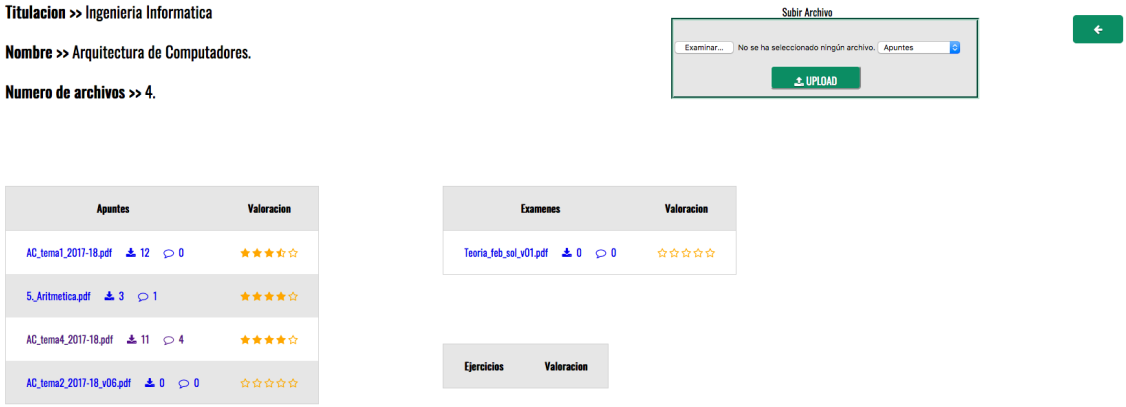


Figura 44. Página de visualización de asignatura.

Ver recurso

La vista de recurso está dividida en dos partes. Dentro del panel se muestra toda la información de forma clara y organizada (Figura 45).

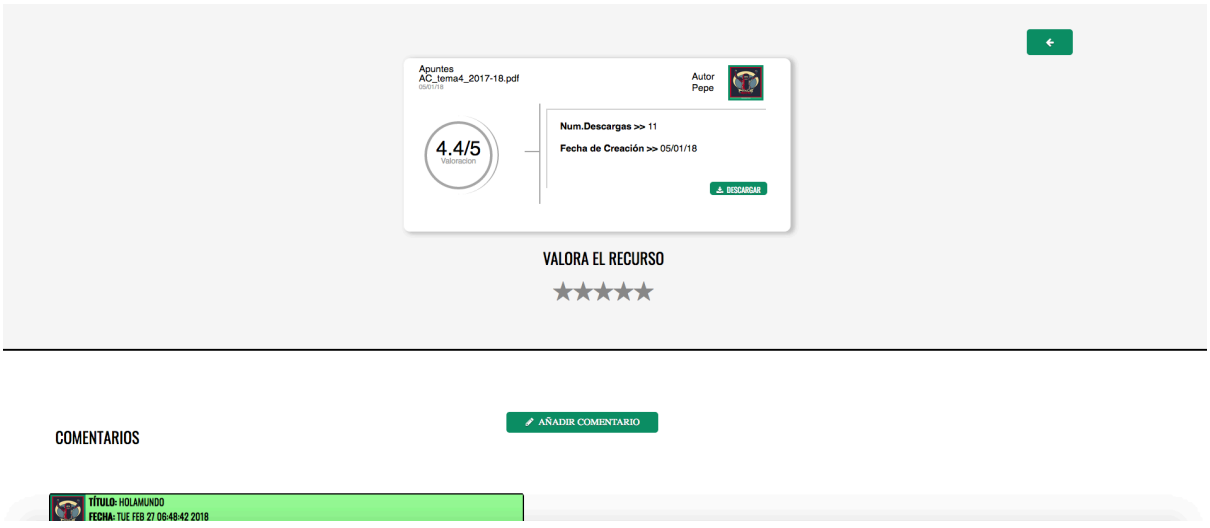


Figura 45. Ver recurso

Además, está página cuenta con una barra de estrellas para dar la valoración, la cuál se resalta al pasar por encima. Tiene un formulario para añadir comentarios ocultos que aparece cuando se hace click sobre el botón "Añadir comentario".

Para ver un recurso de una asignatura concreta se debe realizar una búsqueda en la base de datos sobre la colección **Archivo** y buscar según su _id. Una vez realizada la búsqueda se devuelve la información relativa a ese fichero. Lo más difícil es paginar los comentarios (ver figura 46) de manera que aparezcan gradualmente desplegados a través de la herramienta **infinite scroll.** Para paginar los comentarios se utiliza una clase especial de datos proporcionados por humbledb, de tipo array, que es capaz de almacenar los datos según unas especificaciones . En este caso distribuye las páginas de tres en tres elementos y con un tamaño máximo de 1000bytes (Ver figura 47).

```
first_page = []
pagina = 0
n = 0
if comments.length() > 0:
    n = comments.pages() - 1 # Return the current number of pages
    print(comments.length()) - 1  # Return the current number of entries
    first_page = comments[0]
    pagina = 0

comments.clear()  # Remove all entries
```

Figura 46. Paginación

```
class Comments(Array):
    config_database = 'trabajofingrado'
    config_collection = 'comentarios'
    config_padding = 1000 # Number of bytes to pad with
    config_max_size = 3  # Number of entries per page
    titulo = 'titulo'
    id_archivo = 'id_archivo'
    likes = 'likes'
    unlike = "unlike"
    autor = 'autor'
    body = "body"
    fecha = "fecha"
```

Figura 47. Modelo Array

Puntuar recurso

Para puntuar los recursos se ha utilizado una lista de estrellas que se resaltan cuando se pasa por encima y que permite puntuar el recurso según el número de estrellas elegidas.

Una vez realizada la elección se hace click y la aplicación envía la puntuación al servidor que la almacena junto al resto de valoraciones (Figura 48).

```
<div class="ec-stars-wrapper" style="text-align: center; position: absolute; top: 60%; left: 45%;
">
    <h1>Valora el recurso </h1>
    <a href={{url_for('user.puntuar_archivo', voto=1, archivo=archivo.nombre , usuario =
    usuario.nombre)}}  title="Votar con 1 estrellas">&#9733;</a>
    <a href={{url_for('user.puntuar_archivo', voto=2, archivo=archivo.nombre , usuario =
    usuario.nombre)}}  title="Votar con 2 estrellas">&#9733;</a>
    <a href={{url_for('user.puntuar_archivo', voto=3, archivo=archivo.nombre , usuario =
    usuario.nombre)}}  title="Votar con 3 estrellas">&#9733;</a>
    <a href={{url_for('user.puntuar_archivo', voto=4, archivo=archivo.nombre , usuario =
    usuario.nombre)}} title="Votar con 4 estrellas">&#9733;</a>
    <a href={{url_for('user.puntuar_archivo', voto=5, archivo=archivo.nombre , usuario =
    usuario.nombre)}} title="Votar con 5 estrellas">&#9733;</a>
</div>
```

Figura 48. Modelo de puntuación con estrellas

Descargar recurso

Para descargar un recurso se tiene que devolver la información almacenada en un campo de la colección **Archivo** representado en bytes. Es por ello que se debe transformar los bytes para recuperar el recurso original y poder devolverlo de forma correcta (Figura 49).El usuario tendrá la opción de descargar o abrir el archivo.

```
@user.route('/usuario/upload/<filename>/<string:usuario>',methods=['GET', 'POST'])
@login_required
@required_roles('Usuario')
def uploaded_file(filename, usuario):
    with Mongo:
        usuario = Usuario.find_one({Usuario.nombre : usuario})
        descargas = usuario.descargas + 1
        Usuario.update({Usuario._id : usuario._id },{'$set': {Usuario.descargas: descargas}})
        archivo = Archivo.find_one({Archivo.nombre : filename})
        num = int(archivo.num_descargas)+1
        Archivo.update({Archivo.nombre : filename },{'$set': {Archivo.num_descargas: num}})
        return send_file(io.BytesIO(archivo.fichero),
                         attachment_filename=filename,as_attachment=True)
```

Figura 49. Descargar archivo

Comentar recurso

Para realizar un comentario de un recurso se debe pulsar el botón añadir comentario que se encuentra justo después de la información referente al recurso elegido. A continuación se mostrará un formulario, el cuál se debe cumplimentar con la información necesaria, como el título del comentario o el cuerpo de este.

Cada comentario presente en la aplicación sigue el estilo y la estructura que se presenta a continuación (Figura 50).

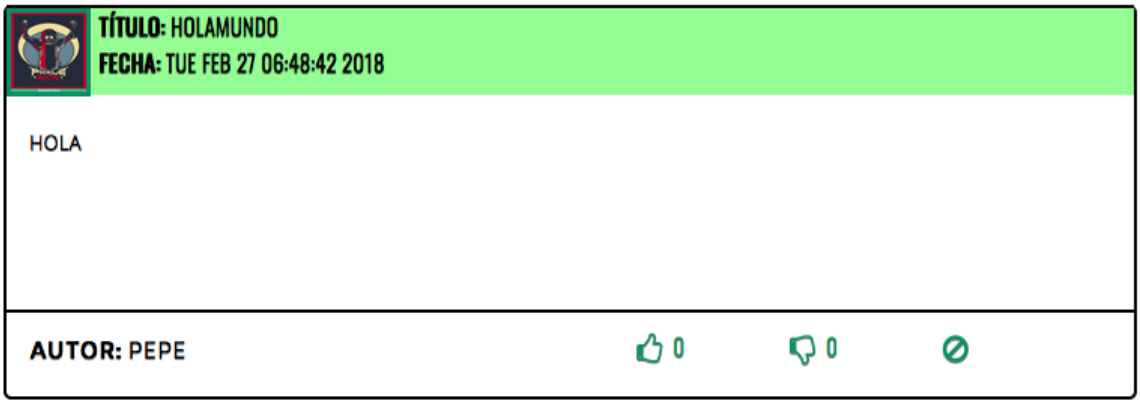


Figura 50. Comentario

Esta sección tiene incorporado la herramienta infinite scroll para desplegar todos los comentarios relacionados con el archivo. Como se ha comentado en la sección *Ver recurso,* se utiliza la paginación para servir los comentarios. En la figura 51 que se muestra a continuación se puede observar la implementación en jQuery de infinite scroll.

```
$(document).ready(function(){

    var ias = $.ias({
    container:  '.comments',
    item:       '.comentario',
    pagination: '.pagination',
    next:       '.pagination a.next'
    });

    ias.extension(new IASSpinnerExtension());
    ias.extension(new IASTriggerExtension({offset: 3}));
    ias.extension(new IASNoneLeftExtension({text: "No existen más comentarios"}));
```

Figura 51. Infinite scroll

Ver Horario

Está dividido en horas y representa el horario laborable del usuario.

Las filas del horario tienen colores alternos para poder diferenciarlas con claridad. Las horas del día se disponen en la parte izquierda en color rojo para resaltar. La cabecera del horario con los días de la semana permanece fija una vez ha alcanzado el tope superior con el menú de la aplicación para poder ver siempre los días de la semana.

Se dispone dos botones para poder alternar entre horario de mañana o de tarde. También hay una sección en la parte superior con el título, una pequeña explicación y dos botones que dan la posibilidad de añadir o eliminar una asignatura" (Figura 52).

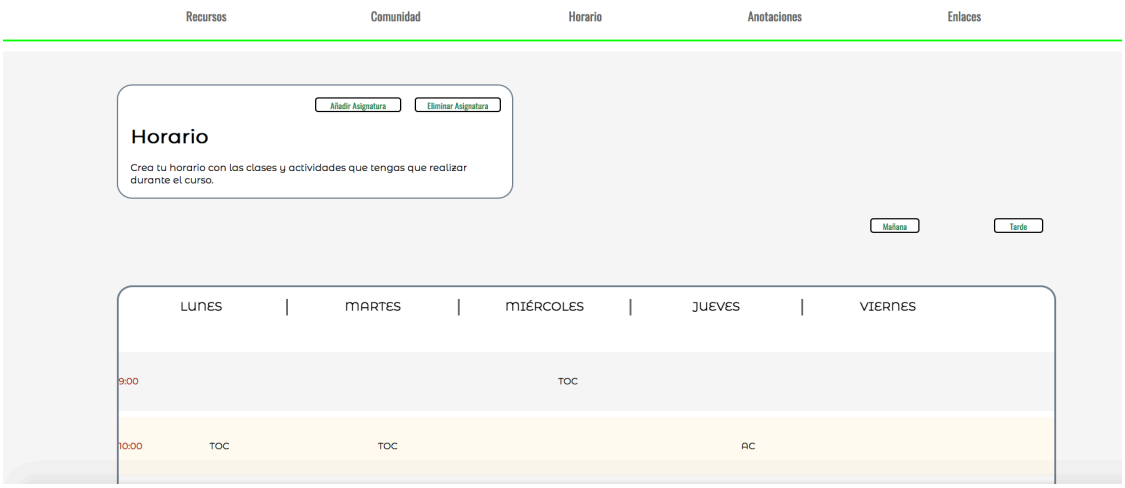


Figura 52. Horario

Para ver el horario de cada usuario hay que realizar una búsqueda en la colección **Horario** y devolver todas las horas que coincidan con el nombre del usuario. Una vez hecho esto se representa cada hora como se muestra en la figura 53.

```
<button id="ma" class="button" >Mañana </button>
<div class="resul_busqueda" id="ma">
    <div class="fila" style="position: sticky; top:80px; z-index: 3;background: white
        border-radius: 25px;
    -moz-border-radius: 25px; -webkit-border-radius: 25px;">
        <div class="celda_dias">Lunes</div>
        <div class="separador"></div>
        <div class="celda_dias">Martes</div>
        <div class="separador"></div>
        <div class="celda_dias">Miércoles</div>
        <div class="separador"></div>
        <div class="celda_dias">Jueves</div>
        <div class="separador"></div>
        <div class="celda_dias">Viernes</div>
    </div>
    {% for h in horas_ma %}
        <div class="fila"
        {% if (h.hora%2 != 0) %}
            style="background: #F5F5F5;"
        {% else %}
            style="background: #FFFAF0;"
        {% endif %}
        >
            <div class="celda">
                <div class="hora">{{h.hora}}:00</div>
                {{h.l}}
            </div>
            <div class="celda">{{h.m}}</div>
            <div class="celda">{{h.x}}</div>
            <div class="celda">{{h.j}}</div>
            <div class="celda">{{h.v}}</div>
        </div>
    {% endfor %}
```

Figura 53. Ver Horario

Añadir asignatura al horario

Para añadir una asignatura al horario primero hay que seleccionar en el formulario el número de horas que tiene esa asignatura a la semana, después se debe elegir para cada una de las horas el día y la hora concreta. También hay que proporcionar el nombre de la

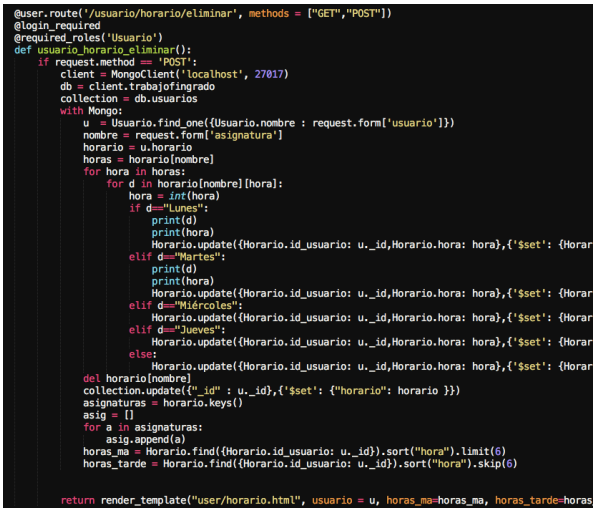
asignatura o tarea que se quiera apuntar en el calendario. Se muestra la implementación a continuación (Figura 54).

```
function genList(num){
clean();
impar = 0;
par = 0;
for(var i = 0; i < num; i++){

    var n = i + 1;
    var myDiv = document.getElementById("input_list");


        var array = ["Lunes","Martes","Miércoles","Jueves","Viernes"];
        var selectList = document.createElement("select");
        myDiv.appendChild(selectList);
        selectList.name = 'dia' + n;

    for (var j = 0; j < array.length; j++) {
        var option = document.createElement("option");
        option.value = array[j];
        option.text = array[j];
        selectList.appendChild(option);
        }

        var selectList = document.createElement("select");
        myDiv.appendChild(selectList);
        selectList.name = 'hora' + n;

    for (var j = 9; j < 21; j++) {
        var option = document.createElement("option");
        option.value = j;
        option.text = j;
        selectList.appendChild(option);
        }

    }
```

Figura 54. Añadir asignatura horario

Eliminar asignatura del horario

Para eliminar una asignatura del horario hay que seleccionar en una lista alguna de las asignaturas o tareas que coincidan con el id_usuario. Una vez seleccionada se manda al servidor para que sea borrada de la bb.dd (Figura 55).



```
@user.route('/usuario/horario/eliminar', methods = ["GET","POST"])
@login_required
@required_roles('Usuario')
def usuario_horario_eliminar():
    if request.method == 'POST':
        client = MongoClient('localhost', 27017)
        db = client.trabajofingrado
        collection = db.usuarios
        with Mongo:
            u  = Usuario.find_one({Usuario.nombre : request.form['usuario']})
            nombre = request.form['asignatura']
            horario = u.horario
            horas = horario[nombre]
            for hora in horas:
                for d in horario[nombre][hora]:
                    hora = int(hora)
                    if d=="Lunes":
                        print(d)
                        print(hora)
                        Horario.update({Horario.id_usuario: u._id,Horario.hora: hora},{'$set': {Horar
                    elif d=="Martes":
                        print(d)
                        print(hora)
                        Horario.update({Horario.id_usuario: u._id,Horario.hora: hora},{'$set': {Horar
                    elif d=="Miércoles":
                        Horario.update({Horario.id_usuario: u._id,Horario.hora: hora},{'$set': {Horar
                    elif d=="Jueves":
                        Horario.update({Horario.id_usuario: u._id,Horario.hora: hora},{'$set': {Horar
                    else:
                        Horario.update({Horario.id_usuario: u._id,Horario.hora: hora},{'$set': {Horar
            del horario[nombre]
            collection.update({"_id" : u._id},{'$set': {"horario": horario }})
            asignaturas = horario.keys()
            asig = []
            for a in asignaturas:
                asig.append(a)
            horas_ma = Horario.find({Horario.id_usuario: u._id}).sort("hora").limit(6)
            horas_tarde = Horario.find({Horario.id_usuario: u._id}).sort("hora").skip(6)


        return render_template("user/horario.html", usuario = u, horas_ma=horas_ma, horas_tarde=horas
```

Figura 55. Eliminar asignatura horario

Ver comunidad

En esta vista se observa en la parte izquierda todos los posts publicados para esta titulación de forma descendente respecto a la fecha de publicación. Se puede filtrar dichos posts por su asunto utilizando la lista de filtros situado en la parte superior.

En la parte derecha de la sección de comunidad se puede ver un ranking con el top de las diferentes secciones de la aplicación. Existe un Top de Descargas, otro Top para los archivos subidos y por último una parte con todos los usuarios de la aplicación (Figura 56).

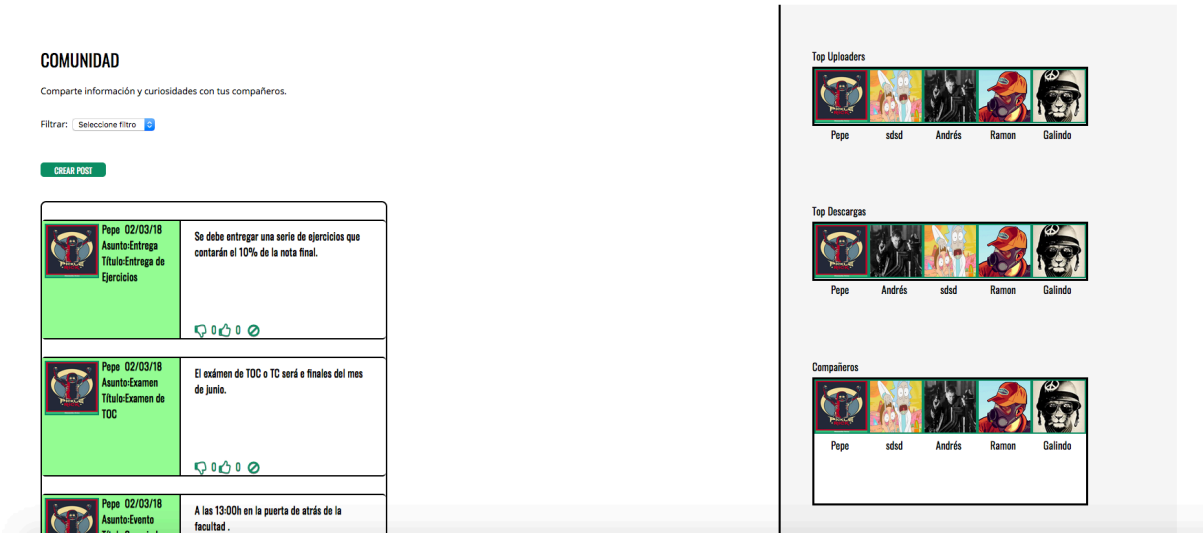


Figura 56. Comunidad

Para acceder a la página de la comunidad se cargan todos los posts que coincidan con el _id de la titulación del usuario activo. También se buscan todos los usuarios que están registrados en esa titulación, filtrando según distintos parámetros y limitando la búsqueda a 5 para mostrarlos en orden en la zona del ranking. (Ver figura 57).

```
@user.route('/usuario/comunidad/<string:nombre>', methods = ["GET","POST"])
@login_required
@required_roles('Usuario')
def usuario_comunidad(nombre):
    with Mongo:
        u  = Usuario.find_one({Usuario.nombre : nombre})
        f = Facultad.find_one({Facultad.nombre : u.facultad})
        top_subidas = Usuario.find({Usuario.titulacion: u.titulacion}).sort("archivos_subidos",-1).li
        top_descargas = Usuario.find({Usuario.titulacion: u.titulacion}).sort("descargas",-1).limit(5
        compa = Usuario.find({Usuario.titulacion: u.titulacion})
        for titulacion in f.titulaciones:
            if titulacion == u.titulacion:
                t = Titulacion.find_one({Titulacion.nombre : titulacion})
        posts = Post.find({Post.id_titulacion: t._id}).sort("_id",-1)

    return render_template("user/comunidad.html", usuario = u,  posts = posts, titulacion = t._id, to
```

Figura 57. Ver comunidad

Crear Post

Se deben rellenar todos los campos del formulario que se muestra en la figura 58. Después se envían dichos datos al servidor el cuál genera una colección de tipo **Post,** para después guardarla en la bb.dd (Figura 59)**.**

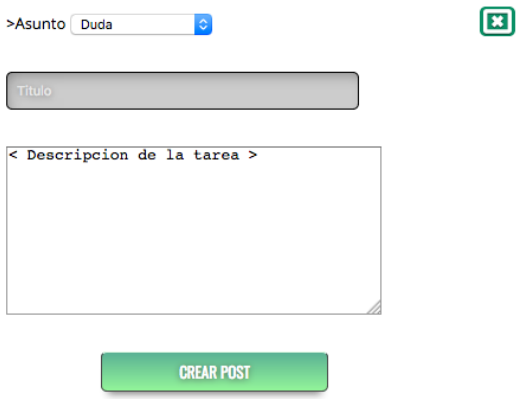


Figura 58. Crear post

```
<form class="form" method="post" id = "postform" action="/usuario/comunidad/
finalizar_post">
    <p><input id="user_name" type = "hidden" name = "nombre" value = "
    {{usuario.nombre}}" /></p>
    <p><input type = "hidden" name = "titulacion" value = "{{titulacion}}" /></p>
    <div>
        <p>>Asunto
            <select name="asunto"  id="asunto" style="width:120px">
                <option value= "Duda" >Duda</option>
                <option value= "Evento" >Evento</option>
                <option value= "Examen" >Examen</option>
                <option value= "Práctica" >Práctica</option>
                <option value= "Entrega" >Entrega</option>
            </select>
        </p>
    </div>
    <p><input type="text" name="titulo" class="pass" placeholder="Titulo"></p>

    <p><textarea rows="10" cols="40" name="body"  class="name" form="postform"><
    Descripcion de la tarea ></textarea></p>


 <input type="submit" class="submit" value="Crear Post">
form>
```

Figura 59. Crear post

Ver Ranking

Como se ha explicado en la sección Ver Comunidad, se cargan los usuarios y se filtran por distintos parámetros como el número de archivos subidos y el número de descargas realizadas. Se puede observar la implementación en la figura 60.

Figura 60. Ver Ranking

Ver compañero

Esta vista tiene un diseño similar al de la vista del perfil, pero con una distribución diferente. El avatar del usuario se dispone en la parte izquierda, al lado se puede ver la información del usuario y en la parte inferior sus estadísticas respecto a la aplicación. También cuenta con un botón en forma de buzón para dar la posibilidad de escribir un mensaje al usuario seleccionado (Figura 61).

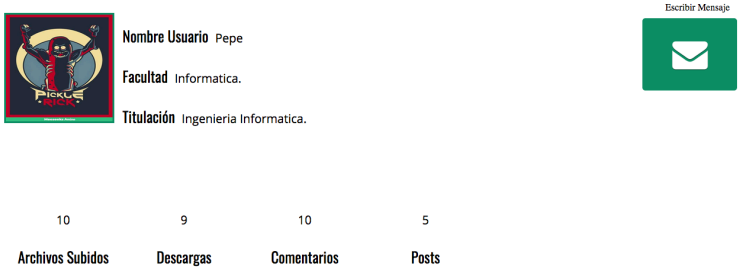


Figura 61. Perfil de compañero

Cuando se quiera acceder a la información de un usuario en particular solo se necesita el nombre de dicho usuario, realizar la búsqueda en la base de datos en la colección **Usuario** y servir dicha información en una página como se observa en la figura 62.

```
@user.route('/usuario/ver_comp/<string:usuario>/<string:nombre>' , methods = ["GET","POST"])
@login_required
def ver_comp(usuario,nombre):


    if request.method == 'GET':
        with Mongo:

            compa  = Usuario.find_one({Usuario.nombre : nombre})
            u = Usuario.find_one({Usuario.nombre : usuario})
        if nombre == usuario:
            return redirect(url_for('.perfil', nombre=usuario))
        else:
            return render_template("user/ver_usuario.html", usuario = u, compa= compa)
```

Figura 62. Ver Compañero

Ver perfil

La página de perfil es accesible a través del enlace "Perfil" del header. Una vez allí (figura 63) se puede observar que la información personal del usuario se expone en forma de listado, dando la opción a editar cualquiera de los campos, excepto el nombre que queda vinculado a la cuenta.

En la parte derecha se muestra el avatar de cada usuario también con la opción de cambiar de avatar cuando se quiera.

El parte inferior se muestran las estadísticas de cada usuario respecto a la aplicación como pueden ser número de descargas o números y comentarios realizados.

Por último, se encuentra el icono del buzón que da acceso al buzón personal de cada usuario. Éste cambia de abierto a cerrado según estemos dentro o fuera del buzón.

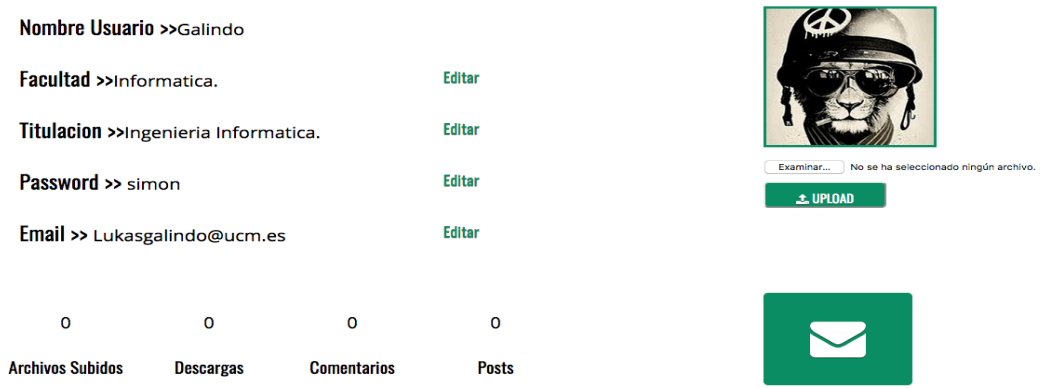


Figura 63. Perfil de usuario

Similar a la operación realizada en *Ver compañero,* se diferencian en que para cargar el perfil es necesario cargar más datos como las facultades y las titulaciones presentes en la base de datos para dar la oportunidad al usuario de editar dicha información y para eso debe cargar toda la información con anterioridad (Figura 64). Dicha información se puede cambiar a través de métodos asíncronos AJAX que solicitan información al servidor según las elecciones del usuario.

```
@user.route('/usuario/perfil/<string:nombre>',methods=['GET', 'POST'])
@login_required
@required_roles('Usuario')
def perfil(nombre):
    with Mongo:
        u  = Usuario.find_one({Usuario.nombre : nombre})
        facultad = Facultad.find_one({Facultad.nombre : u.facultad})
        titul = Titulacion.find({Titulacion.id_facultad : facultad._id}).sort("nombre")
        facul = Facultad.find()

    return render_template("user/perfil.html", usuario = u, titul = titul, facul = facul )
```

Figura 64. Ver Perfil

Desconectar

Para desconectarse la aplicación se debe cerrar sesión en login_user. Para ello se utiliza el método que se muestra en la figura 65, que llama al método logout_user, el cuál elimina al usuario de la sesión y lo envía a la página de identificación.

```python
@user.route("/logout")
@login_required
def logout():
    logout_user()
    return redirect(url_for('user.identificacion'))
```

Figura 65. Desconectar

Ir a pantalla de inicio

Redirecciona al usuario a la url inicio, desde la cuál se puede acceder a la búsqueda de recursos y a las demás opciones a través de su menú superior (Figura 66). Se debe recuperar toda la información necesaria de la base de datos para cargar dicha página.

```python
@user.route('/usuario/inicio/volver/<string:nombre>', methods = ["GET","POST"])
@login_required
@required_roles('Usuario')
def usuario_volver_inicio(nombre):
    with Mongo:
        u  = Usuario.find_one({Usuario.nombre : nombre})
        f = Facultad.find_one({Facultad.nombre : u.facultad})
        t = Titulacion.find_one({Titulacion.nombre : u.titulacion},{Titulacion.id_facultad: f._id})
        cursos = Curso.find({Curso.id_titulacion: t._id})
        resultados = Archivo.find({Archivo.id_titulacion: t._id}).sort("formato",-1).limit(3)

    return render_template("user/inicio.html", usuario = u, cursos = cursos ,resultados= resultados)
```

Figura 66. Ir pantalla de inicio

Ver anotaciones

Para ver las anotaciones de un usuario se deben cargar todas la notas que tiene como id_usuario el id del usuario activo. Después se devuelve toda la información en una lista ordenada a través del campo formato, que representa la fecha del cumplimiento de cada tarea, dependiendo del filtro activo, que de forma predeterminada está inactivo (Figura 67).

```python
@user.route('/usuario/anotaciones/<string:nombre>', methods = ["GET","POST"])
@login_required
@required_roles('Usuario')
def usuario_anotaciones(nombre):
    with Mongo:
        u  = Usuario.find_one({Usuario.nombre : nombre})
        notas = Notas.find({Notas.autor : nombre}).sort("formato")


    busqueda = False
    filtro = ""


    return render_template("user/anotaciones.html", usuario = u,  notas = notas, busqueda = busqueda, filtro =
```

Figura 67. Ver anotaciones

Filtrar Nota

Cuando se renderiza la página *anotaciones.html* existe un campo llamado *filtro* (Figura 68). En la primera llamada a las anotaciones ese campo está vacío, pero cuando se pulsa uno de los filtros disponibles se asigna el tipo de filtro a ese campo, se vuelve a cargar la página y solo se muestran las notas que coincidan con los parámetros del filtro seleccionado.

```
<div class="menu_notas">
    <a href={{url_for('user.crear_nota', nombre = usuario.nombre) }}   class="button" >Crear Nota </a>

    <p>Filtros:</p>
    <ul class="filtro"  >
        {% if  filtro == "Examen"  %}
            <li class="filtro" ><a class="filtro_activo" href= {{url_for('user.usuario_anotaciones_filtro
            ', nombre=usuario.nombre, filtro = "Examen") }}>Exámenes</a></li>
        {% else %}
            <li class="filtro" ><a class="filtro" href= {{url_for('user.usuario_anotaciones_filtro',
            nombre=usuario.nombre, filtro = "Examen") }}>Exámenes</a></li>
        {% endif %}
        {% if  filtro == "Práctica"  %}
            <li class="filtro"  ><a class="filtro_activo" href= {{url_for('
            user.usuario_anotaciones_filtro', nombre=usuario.nombre, filtro = "Práctica") }}>Prácticas</a
            ></li>
        {% else %}
            <li class="filtro"  ><a class="filtro" href= {{url_for('user.usuario_anotaciones_filtro',
            nombre=usuario.nombre, filtro = "Práctica") }}>Prácticas</a></li>
        {% endif %}
        {% if  filtro == "Entrega"  %}
            <li class="filtro" ><a class="filtro_activo" href= {{url_for('user.usuario_anotaciones_filtro
            ', nombre=usuario.nombre, filtro = "Entrega") }} >Entregas</a></li>
        {% else %}
            <li class="filtro" ><a class="filtro" href= {{url_for('user.usuario_anotaciones_filtro',
            nombre=usuario.nombre, filtro = "Entrega") }} >Entregas</a></li>
        {% endif %}
        {% if  busqueda  %}
            <li class="filtro" ><a class="filtro" href= {{url_for('user.usuario_anotaciones', nombre=
            usuario.nombre) }} >Sin filtrar</a></li>
        {% else %}
            <li class="filtro" ><a class="filtro_activo" href= {{url_for('user.usuario_anotaciones',
            nombre=usuario.nombre) }} >Sin filtrar</a></li>
        {% endif %}
    </ul>
```

Figura 68. Filtrar nota

Ver nota

Se elige la nota que se quiere ver y se envía su _id al servidor para que cargue toda la información referente a ésta. Como se puede ver en la figura 69, contiene los campos asunto, título, fecha y descripción.

```
<div class="wrapper">
    <div class="nota_ampliada">
        <h1 style="text-align: center;">Nota</h1>
        <p><b>Asunto: </b>    {{nota.asunto}}</p>
        <p><b>Título: </b>    {{nota.titulo}}</p>
        <p><b>Fecha: </b>    {{nota.fecha}}</p>
        <p><b>Descripción: </b></p>
        <p>{{nota.descripcion}}</p>
        <a  class = "icono" href={{url_for('user.borrar_nota', nombre = usuario.nombre, nota =
        nota._id) }}   class="button" ><i class="fa fa-trash fa-2x" aria-hidden="true"></i>
        Borrar Nota </a>
        <a class = "icono_salir" href={{url_for('user.usuario_anotaciones', nombre=
        usuario.nombre) }}   class="button" ><i class="fa fa-window-close-o fa-2x" aria-hidden=
        "true"></i></a>
    </div>
```

Figura 69. Ver nota

Crear nota

Para crear una nota hay que proporcionar la información a la aplicación a través de un formulario post, para luego generar un modelo de **Nota** y guardarlo en la base de datos (Figura 70).

```
<form class="form" method="post" id = "notaform" action="/usuario/anotaciones/
finalizar_nota">
    <p><input type = "hidden" name = "nombre" value = "{{usuario.nombre}}" /></p>
    <div>
        <p>>Asunto
            <select name="asunto"  id="asunto" style="width:120px">
                <option value= "Examen" >Examen</option>
                <option value= "Práctica" >Práctica</option>
                <option value= "Entrega" >Entrega</option>
            </select>
        </p>
    </div>
    <p><input type="text" name="titulo" class="pass" placeholder="Titulo"></p>
    <p><input type="text" name="datepicker" id="datepicker" placeholder="Fecha"
    readonly="readonly" size="12" /></p>
    <p><textarea rows="10" cols="40" name="descripcion"  class="name" form="
    notaform"><  Descripcion de la tarea ></textarea></p>
    <input type="submit" class="submit" value="Crear Nota">
    <a class = "icono_salir" href={{url_for('user.usuario_anotaciones', nombre=
    usuario.nombre) }}   class="button" ><i class="fa fa-window-close-o fa-2x"
    aria-hidden="true"></i></a>
</form>
```

Figura 70. Crear nota

Eliminar nota

Se envía el _id de la nota a la aplicación para borrarlo de la base de datos a través del método *borrar_nota* (Figura 71).

```
@user.route('/usuario/anotaciones/borrar_nota/<nombre>/<nota>',methods=
@login_required
@required_roles('Usuario')
def borrar_nota(nombre, nota):
    with Mongo:
        Notas.delete_one({Notas._id: ObjectId(nota)})
        u  = Usuario.find_one({Usuario.nombre : nombre})
        notas = Notas.find({Notas.autor : nombre}).sort("formato")
```

Figura 71. Eliminar nota

Ver enlaces

La sección de enlaces es accesible a través del menú de la aplicación haciendo click sobre la opción “Enlaces”. Se muestra una lista de enlaces de interés para el usuario relacionado con la facultad y la titulación en la que está registrado.

En la parte derecha aparece una foto de la facultad a la que pertenece el usuario (Figura 72).

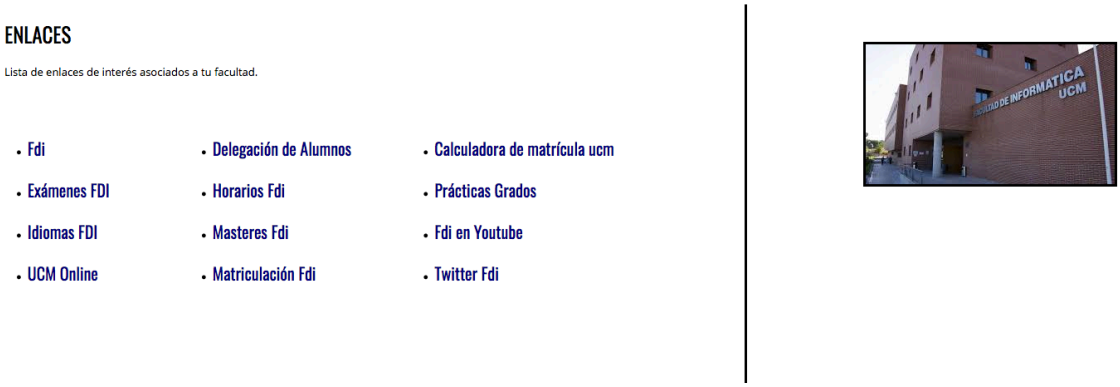


Figura 72. Enlaces

En la página enlace.html se encuentran todos los enlaces como se muestra a continuación (Figura 73).

```
@user.route('/usuario/enlaces/<string:nombre>', methods = ["GET","POST"])
@login_required
@required_roles('Usuario')
def usuario_enlaces(nombre):
    with Mongo:
        u  = Usuario.find_one({Usuario.nombre : nombre})
        f = Facultad.find_one({Facultad.nombre : u.facultad})
        for titulacion in f.titulaciones:
            if titulacion == u.titulacion:
                t = Titulacion.find_one({Titulacion.nombre : titulacion})
        enlaces = Enlace.find({Enlace.id_titulacion: t._id})

    return render_template("user/enlaces.html", usuario = u, titulacion = t._id, enlaces=enlaces)
```

Figura 73. Ver enlaces

Ver buzón

Cada usuario puede entrar en su propio buzón para ver sus mensajes. Si existe algún mensaje sin leer, la aplicación añadirá un aviso en forma de círculo rojo con un número en su interior que indica el número de mensajes sin leer.

El buzón está divido en secciones, no leídos, leídos y enviados. En cada una de las secciones se podrá ver una lista con los mensajes incluidos. También se puede enviar un mensaje al usuario que ha mandado un mensaje o denunciar el mensaje si su contenido es inapropiado. Se puede salir y entrar en la parte del buzón pulsando el sobre de la parte inferior derecha.

Dicho icono cambiará de abierto a cerrado dependiendo si está o no en el buzón (Figura 74).

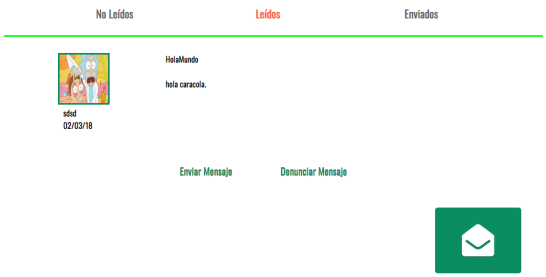


Figura 74. Buzón usuario

Para ver los mensajes de cada usuario se debe enviar el nombre identificativo de éste, a partir de este parámetro se buscan todos los mensajes que coincidan tanto como autor como destinatario y los devuelva a la página de buzón (Figura 75).

```
@user.route('/usuario/perfil/mensajes/leer/<string:nombre>/<string:mensaje>',methods=['GET', 'POST'])
@login_required
@required_roles('Usuario')
def usuario_leer_mensajes(nombre, mensaje):
    destino = {}
    with Mongo:
        u  = Usuario.find_one({Usuario.nombre : nombre})
        m = Mensaje.find_one({Mensaje._id: ObjectId(mensaje)})
        if m.estado.encode('utf-8') == "No Leídos":
            enlace = "#no_leidos"
            Mensaje.update({Mensaje._id:ObjectId(mensaje)},{'$set': { Mensaje.estado: "Leídos"}})
            Usuario.update({Usuario._id:u._id},{'$set': { Usuario.no_leidos: u.no_leidos - 1 }})
        elif m.estado.encode('utf-8') == "Leídos":
            enlace = "#leidos"
        else:
            enlace = "#enviados"
            destino = Usuario.find_one({Usuario._id : ObjectId(m.id_usuario)})
    return render_template("user/ver_mensaje.html", usuario = u, mensaje=m, enlace=enlace, destino = destino)
```

Figura 75. Ver buzón

Escribir a compañero

Se necesita el nombre del usuario al que se le quiere escribir el mensajee Después se debe introducir otra información como el asunto o el cuerpo del mensaje (Figura 76). A

continuación, se genera el modelo de datos de **Mensaje** y se guarda en la base de datos. El compañero al que se quiere escribir se guarda en el campo id_usuario (Figura 77).

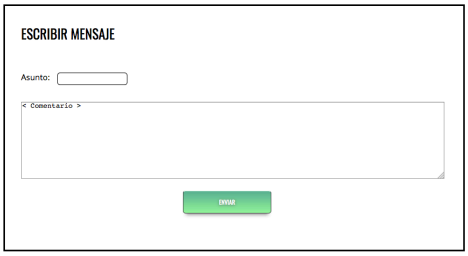


Figura 76. Escribir mensaje

```
@user.route('/usuario/perfil/finalizar_mensaje')
def escribir_mensaje_perfil():

    autor = request.args.get('autor')
    destinatario = request.args.get('destino')
    asunto = request.args.get('asunto')
    body = request.args.get('body')
    with Mongo:
        mensaje = Mensaje()
        mensaje2 = Mensaje()
        compa  = Usuario.find_one({Usuario.nombre : destinatario})
        Usuario.update({Usuario._id: compa._id},{'$set': {Usuario.no_leidos: compa.no_leidos + 1 }})
        autor = Usuario.find_one({Usuario.nombre : autor})
```

Figura 77. Escribir compañero

Denunciar mensaje/comentario/post

En cualquiera de los recursos mencionados en el título existe la posibilidad de realizar una denuncia que será archivada para que un administrador pueda revisarla, basta con hacer click sobre el icono de prohibido y se enviará al servidor.

Para registrar una denuncia se pone el campo ***denuncia*** de cualquiera de los modelos de datos anteriormente mencionados a true y el administrador podrá verlo en el buzón de denuncias (Figura 78).

```
@user.route('/usuario/comentarios/denuncias/<comentario>/<string:archivo>/<string:usuario>', methods = ["GET"])
def usuario_comentarios_denuncias(comentario,archivo,usuario):
    with Mongo:
        Comentario.update({Comentario._id: ObjectId(comentario)},{'$set': {Comentario.denuncia:  True }})

    return redirect(url_for('.ver_archivo',  archivo=archivo , usuario = usuario))
```

Figura 78. Denunciar recurso

Dar positivo/negativo

Los comentarios y posts tienen la oportunidad de ser valorados por los usuarios, para ello cuentan con dos botones, un pulgar hacia arriba y otro hacia abajo que representan el voto positivo y negativo (Figura 79).

Una vez se hace click en alguno de ellos se contabiliza el voto para el recurso seleccionado. La aplicación restringe los votos de cada usuario a uno para cada comentario o post, actualizando el voto si el usuario cambia de opinión. Se realiza este procedimiento gracias al campo voto que se encuentra tanto en los comentarios como en los posts, que guardan una lista de cada usuario que ha realizado el voto como el valor de este, el cuál puede cambiar a gusto del usuario.

```
@user.route('/usuario/comunidad/valorar_post/<int:valor>/<post>/<string:usuario>' ,methods=['GET', 'POST'])
@login_required
@required_roles('Usuario')
def value_post(valor,post,usuario):

    if valor==1:
        with Mongo:
            c = Post.find_one({Post._id : ObjectId(post)})
            votos = c.votos
            if votos.has_key(usuario):
                if votos[usuario] == 0:
                    unlikes = int(c.unlikes) - 1
                    likes = int(c.likes) + 1
                    votos[usuario] = 1
                    Post.update({Post._id: ObjectId(post)},{'$set': {Post.likes:  likes, Post.unlikes:  unlikes, Post.votos
            else:
                likes = int(c.likes) + 1
                votos[usuario] = 1
                Post.update({Post._id: ObjectId(post)},{'$set': {Post.likes:  likes, Post.votos:  votos }})

    else:
        with Mongo:
            c = Post.find_one({Post._id : ObjectId(post)})
            votos = c.votos
            if votos.has_key(usuario):
                if votos[usuario] == 1:
                    unlikes = int(c.unlikes) + 1
                    likes = int(c.likes) - 1
                    votos[usuario] = 0
                    Post.update({Post._id: ObjectId(post)},{'$set': {Post.likes:  likes, Post.unlikes:  unlikes, Post.votos
            else:
                unlikes = int(c.unlikes) + 1
                votos[usuario] = 0
                Post.update({Post._id: ObjectId(post)},{'$set': {Post.unlikes:  unlikes, Post.votos:  votos }})

    return redirect(url_for('.usuario_comunidad', nombre = usuario))
```

Figura 79. Denunciar recurso

Ver menú inicio administrador

El menú de administración se muestra como una lista de botones con las diferentes opciones de uso del administrador. Desde aquí se tiene acceso a todas las opciones disponibles para el administrador, agregar o modificar una facultad, revisar los buzones o gestionar archivos y usuarios.

Ambos buzones, tanto el de archivos pendientes como el de denuncias, darán información en forma de círculo blanco con números rojos del número de recursos pendientes de confirmación o revisión (Figura 80).

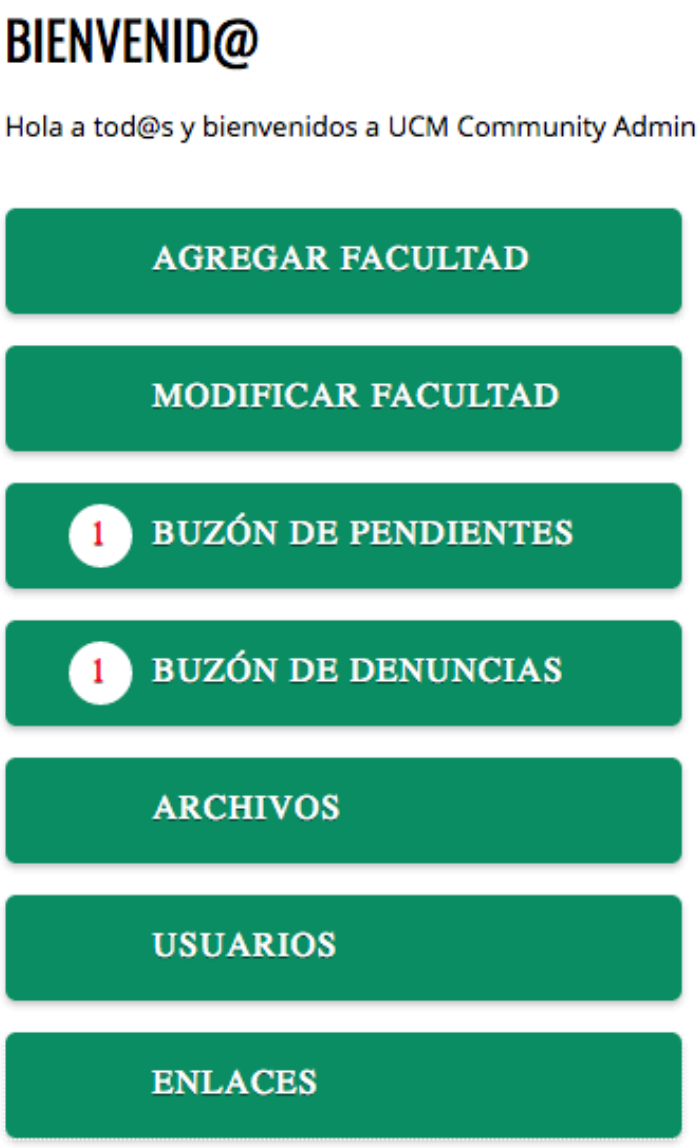


Figura 80. Menú administración

El menú del administrador está formado por una lista esta de enlaces que redireccionan a las distintas páginas que forman la aplicación para el administrador. Se muestra la implementación a continuación (Figura 81).

```
<body>

  {% include 'admin/header_admin.html' %}


  <div class="wrapper">
      <h1>Bienvenid@</h1>
      <p>Hola a tod@s y bienvenidos a UCM Community Admin</p>
      <a href={{url_for('admin.agregar_facult', nombre=usuario._id) }} class="button" > Agregar Facultad</a>
      <a href={{url_for('admin.admin_modificar', nombre=usuario._id) }}  class="button" >Modificar Facultad</a>
      <a href={{url_for('admin.admin_pendientes', nombre=usuario._id) }}  class="button" >Buzón de pendientes
        {% if pendientes>0 %}
          <div class="num">{{pendientes}}</div>
        {% endif %}
      </a>
      <a href={{url_for('admin.admin_denuncias', nombre=usuario._id) }}  class="button" > Buzón de denuncias
        {% if denuncias>0 %}
          <div class="num">{{denuncias}}</div>
        {% endif %}
      </a>
      <a href={{url_for('admin.admin_archivos', nombre=usuario._id) }}  class="button" >Archivos</a>
      <a href={{url_for('admin.admin_usuarios', nombre=usuario._id) }}  class="button" >Usuarios</a>
      <a href={{url_for('admin.admin_enlaces', nombre=usuario._id) }}  class="button" >Enlaces</a>

  </div>
</body>
```

Figura 81. Ver menú administrador

Configuración

Se puede cambiar el nombre del usuario administrador, el password o añadir un nuevo usuario administrador a través de la opción configuración del menú superior(Figura 82). Se podrán cambiar las diferentes opciones a través de formularios como queda reflejada en la figura 83.

Figura 82. Menú Superior Administrador

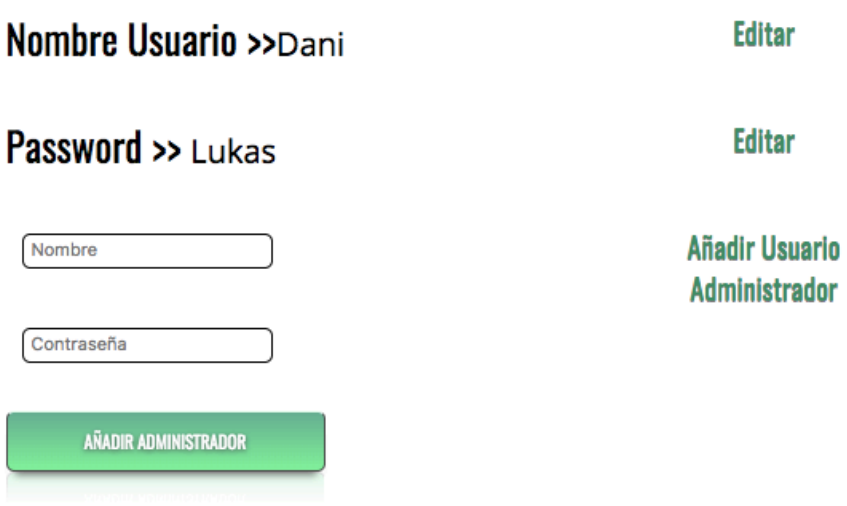


Figura 83. Configuración Administrador

Añadir Facultad

Desde el menú inicial el administrador puede registrar una nueva facultad en la aplicación. Para ello primero debe indicar a través de un formulario el nombre de la facultad y el número de titulaciones ofertadas por dicha facultad (Figura 84).

Figura 84. Registrar Universidad

Una vez se haya pulsado el botón siguiente se desplegará un nuevo formulario en la que el administrador tiene que introducir el nombre de cada titulación y el número de cursos que tendrá.

A continuación, se pasará a otro formulario en el que el administrador indicará el número de asignaturas de cada curso y el nombre de cada una de estas asignaturas.

Después se mostrará al administrador un listado con el resumen de la facultad, titulaciones, cursos y asignaturas que se han registrado en la aplicación.

En la figura 85 que se muestra a continuación se puede ver la implementación del método utilizado para agregar una facultad.

```
#Creamos la titulacion con los datos aportados por el form
t = Titulacion()
t.nombre = facultad.titulaciones[num]
t.id_facultad = facultad._id
t.num_cursos = facultad.cursos_x_titul[num]
t.asignaturas = []
asign_titul = []

#Insertamos la titulacion creada anteriormente en Titulaciones
with Mongo:
    titulacion_id = Titulacion.insert(t)

for i in range(1,lista_cursos[num] + 1):
    lista_asig = []
    indice = str(i)
    numero_asignaturas = int(request.form['asignaturas_'+ indice])
    for j in range(1 ,numero_asignaturas + 1):
        n = str(j)
        lista_asig.append(request.form['text'+n+indice])
#Creamos los cursos correspondientes a la titulacion creada y los insertamos en Cursos
    curso = Curso()
    curso.numero = i
    curso.id_titulacion = titulacion_id
    curso.num_asignaturas = j
    curso.asignaturas = lista_asig
    with Mongo:
        curso_id = Curso.insert(curso)
#Insertamos todas las asignaturas en la bb.dd  con la _id del curso correspondiente
    for j in range(1 ,numero_asignaturas + 1):
        n = str(j)
        asignatura = Asignatura()
        asignatura.nombre = request.form['text'+n+indice]
        asignatura.id_curso = curso_id
        asignatura.num_ficheros = 0
        asignatura.nombre_ficheros = []
        with Mongo:
            Asignatura.insert(asignatura)

    asign_titul.append(lista_asig)
#Adaptamos la titulacion con la lista de asignaturas enviadas a traves del form
with Mongo:
    Titulacion.update({Titulacion._id: titulacion_id },{'$set': {Titulacion.asignaturas: asign_titul}})
```

Figura 85. Añadir facultad

Modificar Facultad

Para modificar una facultad el administrador deberá desde el menú inicial de administración pinchar sobre la opción “Modificar Facultad”. A continuación, debe elegir que facultad es la que quiere modificar a través de una lista en un formulario (Figura 86).

Figura 86. Elegir Facultad

Una vez hecho aparecerán distintas opciones: borrar la facultad seleccionada, acceder a alguna de las titulaciones ya registradas para modificar algo o agregar una nueva titulación a la facultad (Figura 87).

Figura 87. Ver Facultad

Desde el menú anterior al pinchar en la opción "Agregar Titulación" el administrador accederá a un formulario similar al presentado durante el proceso de registrar una facultad.

Se debe introducir el nombre de la nueva titulación, el número de cursos y el número y nombre de las asignaturas de cada curso.

Añadir titulación

La aplicación ofrece la opción de añadir una titulación a una facultad concreta a través del _id de la facultad elegida, que se envía en un formulario a admin.py.

A continuación, se puede introducir toda la información sobre la titulación: nombre, número de cursos y número y nombre de las asignaturas de cada curso, en los formularios que vayan apareciendo.

Para introducir una nueva titulación hay que actualizar campos de la colección ***Facultad*** que pertenezcan a la facultad elegida para añadir una titulación, como son *num_titul,lista_titul y cursos_x_titul*

Por último, hay que crear todos los cursos que pertenecen al modelo de datos ***Curso*** con la información de los formularios, todas las asignaturas de cada curso y para finalizar la titulación (Figura 88).

```
@admin.route('/admin/modificar_titulacion_fin', methods = ["GET","POST"])
def modificar_titulacion_fin():
    #Encontrar la facultad asociada al nombre pasado por el form
    facul_nombre = request.form['facultad']
    with Mongo:
        facultad = Facultad.find_one({Facultad.nombre : facul_nombre })

    #Creamos la titulacion con los datos aportados por el form

    t = Titulacion()
    t.nombre = request.form['nombre']
    t.id_facultad = facultad._id
    t.num_cursos = int(request.form['cursos'])
    t.asignaturas = []
    asign_titul = []

    #Insertamos la titulacion creada anteriormente en Titulaciones
    with Mongo:
        titulacion_id = Titulacion.insert(t)



    for i in range(1,t.num_cursos + 1):
        lista_asig = []
        indice = str(i)
        numero_asignaturas = int(request.form['asignaturas_'+ indice])
        for j in range(1 ,numero_asignaturas + 1):
            n = str(j)
            lista_asig.append(request.form['text'+n+indice])
    #Creamos los cursos correspondientes a la titulacion creada y los insertamos en Cursos
        curso = Curso()
        curso.numero = i
        curso.id_titulacion = titulacion_id
        curso.num_asignaturas = j
        curso.asignaturas = lista_asig
        with Mongo:
            curso_id = Curso.insert(curso)
    #Insertamos todas las asignaturas en la bb.dd  con la _id del curso correspondiente
        for j in range(1 ,numero_asignaturas + 1):
            n = str(j)
            asignatura = Asignatura()
            asignatura.nombre = request.form['text'+n+indice]
            asignatura.id_curso = curso_id
            asignatura.num_ficheros = 0
            asignatura.nombre_ficheros = []
            with Mongo:
                Asignatura.insert(asignatura)

        asign_titul.append(lista_asig)
    #Adaptamos la titulacion con la lista de asignaturas enviadas a traves del form
    with Mongo:
        Titulacion.update({Titulacion.nombre: t.nombre },{'$set': {Titulacion.asignaturas: asign_titul}})
```

Figura 88. Añadir titulación

Eliminar titulación

Para eliminar la titulación es necesario primero acceder a la titulación y después eliminar la titulación desde el menú a través de la opción borrar facultad de la parte superior (Figura 89).

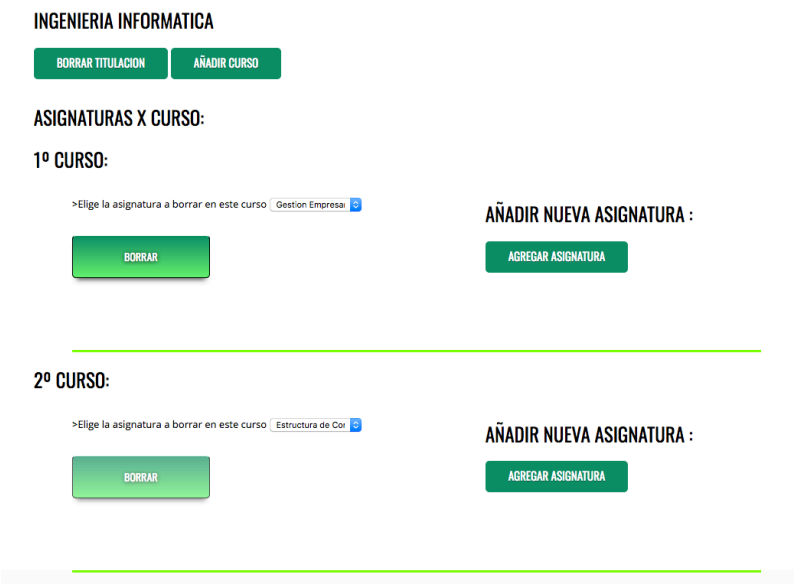


Figura 89. Ver Titulación

El administrador puede borrar una titulación de una facultad determinada, debe elegir la facultad a través de su _id.

Cuando se elimina una titulación hay que actualizar ciertos campos del documento de la facultad a la que pertenece la titulación como ***num_titul,lista_titul y cursos_x_titul*** como se ve en la figura. Después debe eliminar todos los archivos relacionados con cada asignatura de la titulación, las propias asignaturas y los cursos a los que pertenecen.

Para borrar los documentos mencionados se utiliza los métodos *delete_one()* y *delete_many()* (Figura 90).

```
@admin.route('/admin/borrar_facultad_titul/<string:titulacion>/<string:usuario>', methods = ["GET","POST"])
def admin_borrar_titul(titulacion, usuario):
    with Mongo:
        usuario = Usuario.find_one({Usuario._id : ObjectId(usuario)})
        lista_titul= []
        lista_cursos = []
        t = Titulacion.find_one({Titulacion.nombre: ObjectId(titulacion)})
        id_facultad = t.id_facultad
        facultad = Facultad.find_one({Facultad._id : id_facultad })
        numero = facultad.num_titul
        numero -= 1
        titulaciones = facultad.titulaciones
        for i in range(len(titulaciones)):
            if titulaciones[i]== nombre:
                num = i
            else:
                lista_titul.append(titulaciones[i])
        cursos_x_titul = facultad.cursos_x_titul
        for i in range(len(cursos_x_titul)):
            if i != num:
                lista_cursos.append(cursos_x_titul[i])
        Facultad.update({Facultad._id : id_facultad },{'$set': {Facultad.num_titul: numero}})
        Facultad.update({Facultad._id : id_facultad },{'$set': {Facultad.titulaciones: lista_titul}})
        Facultad.update({Facultad._id : id_facultad },{'$set': {Facultad.cursos_x_titul: lista_cursos}})
        cursor_titulacion = Curso.find({Curso.id_titulacion : t._id})
        for c in cursor_titulacion:
            id_curso = c._id
            #Se han agregado las 3 lineas siguientes
            cursor_asignaturas = Asignatura.find({Asignatura.id_curso : id_curso})
            for asignatura in cursor_asignaturas:
                Archivo.delete_many({Archivo.id_asignatura: asignatura._id})
            Asignatura.delete_many({Asignatura.id_curso : id_curso})
            Curso.delete_one({Curso._id : id_curso})
        Titulacion.delete_one({Titulacion._id : t._id})
    return redirect(url_for('admin/admin_inicio' , nombre = usuario.nombre))
```

Figura 90. Borrar Titulación

Añadir curso

Para añadir un curso a una titulación se pulsa sobre la opción “Añadir Curso” del menú de titulación (Figura 89). Después se debe introducir el número o nombre del curso y el número de asignaturas (Figura 91). Aparecerá un nuevo formulario donde el administrador introducirá todos los nombres de las asignaturas.

Figura 91. Añadir Curso

Para añadir un curso a una titulación específica hay que recoger toda la información necesaria utilizando formularios post que se envían a admin.py para generar los documentos basados en los modelos de datos de Curso, Asignatura y Titulación.

Hay que actualizar ciertos campos de la titulación y de la facultad a la que pertenece el curso añadido. Algunos de estos campos son *num_cursos, asignaturas o cursos_x_titul* (Figura 92*)*.

Una vez generado los documentos se guardan en sus respectivas colecciones.

```
u = request.form['usuario']
asi = []
for i in range(1,n +1):
    asignatura = request.form['asignatura_'+ str(i)]
    asi.append(asignatura)
with Mongo:
    usuario = Usuario.find_one({Usuario._id : ObjectId(u)})
    curso = Curso()
    curso.numero = numero
    curso.id_titulacion = ObjectId(nombre)
    curso.num_asignaturas = n
    curso.asignaturas = asi
    id_curso = Curso.insert(curso)
    for j in range(1,n +1):
        a = Asignatura()
        a.nombre = request.form['asignatura_'+ str(j)]
        a.id_curso = id_curso
        a.nombre_ficheros =[]
        a.num_ficheros = 0
        Asignatura.insert(a)
    titulacion = Titulacion.find_one({Titulacion._id: ObjectId(nombre)})
    facultad = Facultad.find_one({Facultad._id: titulacion.id_facultad})
    n = -1
    c=[]
    for t in facultad.titulaciones:
        n+=1
        if t == titulacion.nombre:
            d = facultad.cursos_x_titul[n] + 1
            for l in range(0,len(facultad.cursos_x_titul)):
                if l!=n:
                    c.append(facultad.cursos_x_titul[l])
                else:
                    c.append(d)
            Facultad.update({Facultad._id: titulacion.id_facultad },{'$set': {Facultad.cursos_x_titul: c  }})

    asig_titulacion = []
    for j in range(0,len(titulacion.asignaturas)):
        s=[]
        for v in range(0,len(titulacion.asignaturas[j])):
            s.append(titulacion.asignaturas[j][v])
        asig_titulacion.append(s)
    asig_titulacion.append(asi)
    cursos = Curso.find({Curso.id_titulacion : titulacion._id })
    Titulacion.update({Titulacion._id : ObjectId(nombre) },{'$set': {Titulacion.asignaturas: asig_titulacion }})
    Titulacion.update({Titulacion._id : ObjectId(nombre) },{'$set': {Titulacion.num_cursos: titulacion.num_cursos +


return render_template("admin/mostrar_titulacion.html", titulacion = titulacion, asignaturas = a, cursos = cursos,
```

Figura 92. Añadir curso

Añadir asignatura

Al acceder a una de las titulaciones, una vez se haya seleccionado una de las facultades registradas en la aplicación, aparecerá un listado de todos los cursos (Figura 88) con sus respectivas asignaturas y la opción en la parte superior de añadir un curso o eliminar la titulación entera.

Si se selecciona la opción añadir asignatura aparecerá un formulario (Figura 93) donde el administrador debe introducir el nombre de la nueva asignatura, después se hará click

sobre el botón “Crear Asignatura” y dicha asignatura aparecerá en el listado de asignaturas del curso donde se haya introducido.

Figura 93: Nueva asignatura

Para ello se recogen los datos a través del formulario y se guardan en el curso seleccionado de la titulación correspondiente, y se asigna a la asignatura el id del curso que a su vez incluye el id de la titulación (Figura 94).

```
@admin.route('/admin/agregar_asignatura/fin', methods = ["GET","POST"])
def agregar_asignatura_fin():

    if request.method == 'POST':
        titulacion = request.form['titulacion']
        curso = int(request.form['curso'])
        asignatura = request.form['asignatura']
        usuario = request.form['usuario']
        with Mongo:
            titul = Titulacion.find_one({Titulacion.nombre : titulacion})
            id_titul = titul._id
            asig_titulacion = []

            for j in range(len(titul.asignaturas)):
                asig_curso = []
                for i in range(len(titul.asignaturas[j])):

                    asig_curso.append(titul.asignaturas[j][i])
                if j+1 == curso:
                    asig_curso.append(asignatura)
                asig_titulacion.append(asig_curso)
            Titulacion.update({Titulacion._id : id_titul },{'$set': {Titulacion.asignaturas: asig_titu
            cursos = Curso.find({Curso.id_titulacion: id_titul})
            for c in cursos:
                if c.numero == curso:
                    numero = c.num_asignaturas + 1
                    lista_asig= []
                    id_curso = c._id
                    for i in range(len(c.asignaturas)):
                        lista_asig.append(c.asignaturas[i])
                    lista_asig.append(asignatura)
                    Curso.update({Curso._id : id_curso },{'$set': {Curso.num_asignaturas: numero}})
                    Curso.update({Curso._id : id_curso },{'$set': {Curso.asignaturas: lista_asig}})
```

Figura 94. Añadir asignatura

Esta acción también requiere que se actualicen el curso y la titulación a la que pertenece la asignatura ya que tienen campos que se guardan en un array con la información. Por ejemplo, el campo ***asignaturas*** guarda todas las asignaturas de la titulación para no tener que realizar una búsqueda. En el caso de querer saber todas las asignaturas de la titulación solo se necesita acceder a ese campo.

Borrar Facultad

Al borrar una facultad hay que eliminar todas las titulaciones de esa facultad, todos los cursos de las titulaciones y cada asignatura de cada curso (Figura 95).

```
@admin.route('/admin/borrar_facultad/<string:facultad>/<string:usuario>', methods = ["GET","POST"])
def admin_borrar(facultad,usuario):
    with Mongo:
        usuario = Usuario.find_one({Usuario._id: ObjectId(usuario)})
        facultad = Facultad.find_one({Facultad.nombre : facultad})
        Facultad.delete_one({Facultad.nombre : facultad.nombre})
        cursor_facultad = Titulacion.find({Titulacion.id_facultad: facultad._id})
        for t in cursor_facultad:
            id_titulacion = t._id
            cursor_titulacion = Curso.find({Curso.id_titulacion : id_titulacion})
            for c in cursor_titulacion:
                id_curso = c._id
                cursor_asignaturas = Asignatura.find({Asignatura.id_curso : id_curso})
                for asignatura in cursor_asignaturas:
                    Archivo.delete_many({Archivo.id_asignatura: asignatura._id})
                Asignatura.delete_many({Asignatura.id_curso : id_curso})
                Curso.delete_one({Curso._id : id_curso})
                Titulacion.delete_one({Titulacion._id : id_titulacion})

    return redirect(url_for('admin/admin_inicio' , nombre = usuario.nombre))
```

Figura 95. Borrar facultad

También todos los archivos que pertenecen a dichas asignaturas, pues si no se eliminan se quedarían sin relación con ninguna asignatura y no se tendría acceso a ellos.

Para borrar los documentos mencionados se utiliza el méritos *delete_one()* y *delete_many()*.

Borrar asignatura

Para borrar una asignatura primero hay que eliminar la asignatura de la lista de asignaturas del curso correspondiente y restar uno al número de asignaturas del propio curso. A continuación, se borra la propia asignatura de la colección Asignatura. Así mismo, habrá que actualizar el campo ***asignaturas*** de la titulación a la que pertenece, eliminando la asignatura del array.

También se deben borrar todos lo archivos que estuvieran guardados relacionados con esa asignatura como se ve en el método ***Archivo.delete_many({Archivo.id_asignatura: identidad})***(Figura 96).

```
@admin.route('/admin/borrar_asignatura', methods = ["POST"])
def borrar_asig():
    if request.method == 'POST':
        titulacion = str(request.form['titulacion'])
        curso = int(request.form['curso'])
        n = str(curso)
        asignatura = request.form['asignatura_'+ n]
        usuario = request.form['usuario']
        with Mongo:
            titul = Titulacion.find_one({Titulacion.nombre : titulacion})
            id_titul = titul._id
            asig_titulacion = []

            for a_x_c  in titul.asignaturas:
                asig_curso = []
                for i in range(len(a_x_c)):
                    if a_x_c[i]!= asignatura:
                        asig_curso.append(a_x_c[i])
                asig_titulacion.append(asig_curso)

            Titulacion.update({Titulacion._id : id_titul },{'$set': {Titulacion.asignaturas: asig_titulacion }})
            cursos = Curso.find({Curso.id_titulacion: id_titul})
            for c in cursos:
                if c.numero == curso:
                    numero = c.num_asignaturas - 1
                    lista_asig= []
                    id_curso = c._id
                    for i in range(len(c.asignaturas)):
                        if c.asignaturas[i]!= asignatura:
                            lista_asig.append(c.asignaturas[i])

                    Curso.update({Curso._id : id_curso },{'$set': {Curso.num_asignaturas: numero}})
                    Curso.update({Curso._id : id_curso },{'$set': {Curso.asignaturas: lista_asig}})
            a = Asignatura.find({Asignatura.nombre : asignatura})

            for a_s in a:
                if a_s.id_curso == id_curso:
                    Asignatura.delete_one({Asignatura._id : a_s._id})
                    identidad = a_s._id
            Archivo.delete_many({Archivo.id_asignatura: identidad })
```

Figura 96. Borrar asignatura

Ver pendientes

En esta vista se muestra el título en la parte superior de color negro, el nombre del archivo pendiente de revisión, un botón que permite la descarga del archivo, un select en el que se puede elegir entre todos los archivos pendientes de revisión y otro botón que autoriza al archivo a ser visible en la aplicación (Figura 97).

Figura 97. Archivos pendientes

El administrador puede ver todos los archivos pendientes de revisión. Para ello debe acceder a la sección desde el menú de administración llamado "Buzón de Pendientes". Una vez hecho esto la aplicación hace la búsqueda en la colección **Archivo** como se observa en la siguiente figura, a través del campo *autorizado*, y devuelve los resultados dentro de los parámetros de *pendientes.html* (Figura 98).

```
@admin.route('/admin/pendientes/<string:nombre>', methods = ["GET"])
def admin_pendientes(nombre):
    pendientes = []
    with Mongo:
        u = Usuario.find_one({Usuario._id : ObjectId(nombre)})
        archivos = Archivo.find({Archivo.autorizado: False})
        pendientes = Archivo.find({Archivo.autorizado: False})

    return render_template("admin/pendientes.html", pendientes=pendientes,archivos=archivos, usuario=u)
```

Figura 98. Ver pendientes

Autorizar pendientes

Para autorizar un archivo el administrador elige el archivo que desea autorizar a través de un formulario, se envía el **_id** del archivo para identificarlo, se busca en la colección **Archivo** y se actualiza el campo autorizado a **True**.

A partir de aquí el archivo será visible para el resto de los usuarios de la aplicación.

Después se redirecciona de nuevo a la página admin_pendientes.hmtl, cuya información será actualizada de manera que no aparecerá el archivo autorizado (Figura 99).

```
@admin.route('/admin/autorizar', methods = ["GET","POST"])
def admin_autorizar():
    if request.method == 'POST':
        nombre = request.form['nombre']
        archivo = request.form['revisar']
        with Mongo:
            u = Usuario.find_one({Usuario._id : ObjectId(nombre)})
            Archivo.update({Archivo.nombre : archivo },{'$set': {Archivo.autorizado: True}})

        return redirect(url_for('admin/admin_pendientes' , nombre = u))
```

Figura 99. Autorizar pendientes

Descargar pendientes

Si lo desea el administrador puede descargar el fichero pendiente de revisión para poder analizarlo y determinar si es válido para ser admitido en la aplicación. Alguna de las características que determina esta decisión pueden ser la corrección de la información o si está bien categorizado en esa asignatura. Para realizar la descarga se envía el nombre del archivo a través de un formulario Post, que busca en la colección ***Archivo,*** actualiza el número de descargas y envía el fichero a través del método ***send_file*** con la opción ***as_attachment*** = ***True*** (Figura 100).

```
@admin.route('/admin/upload/<filename>',methods=['GET', 'POST'])
@login_required
@required_roles('Admin')
def uploaded_file_admin(filename):
    with Mongo:
        archivo = Archivo.find_one({Archivo.nombre : filename})
        num = int(archivo.num_descargas)+1
        print(num)
        Archivo.update({Archivo.nombre : filename },{'$set': {Archivo.num_descargas: num}})
        return send_file(io.BytesIO(archivo.fichero),
                         attachment_filename=filename,as_attachment=True)
```

Figura 100. Descargar pendientes

Borrar denuncias

En esta sección el administrador puede ver el buzón de denuncias, donde aparecen todas las denuncias que se han hecho por parte de los usuarios. Está dividido en posts, comentarios y mensajes. Si se encuentra alguna denuncia todavía no revisada el

administrador tiene la opción de ver dicho recurso, para después poder eliminarlo o archivar la denuncia sin hacer nada.

Como se puede observar en la figura 101, las denuncias se muestran en una lista en la parte izquierda de la pantalla, al pasar por encima de alguna de las denuncias en enlace cambia de color. Al hacer click se despliega un panel con los bordes verdes y redondeados que permanecia oculto, el cuál muestra toda la información sobre el recurso denunciado.

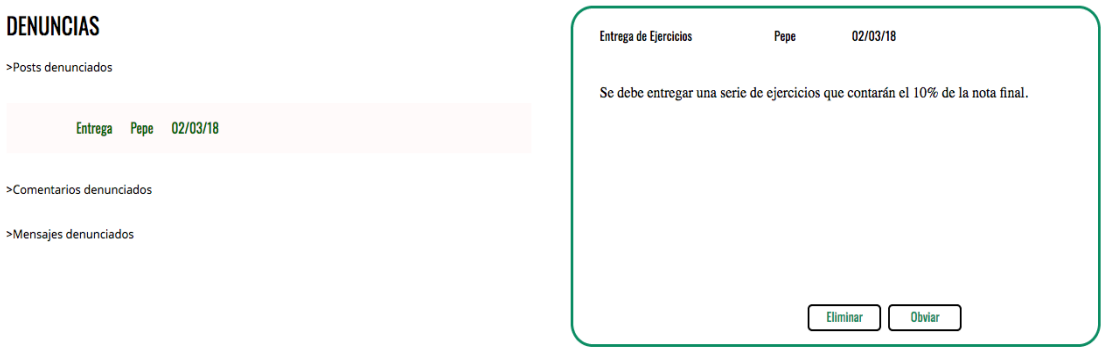


Figura 101. Denuncias

El administrador puede ver todos los recursos denunciados, elegir uno y ver toda su información de forma detallada desde el menú de administración. Si quiere borrar el recurso objeto de la denuncia se envía el id del recurso a través del método ***admin_denuncias_eliminar,*** dentro del archivo admin.py, el cuál se encarga de buscar en las tres colecciones Mensaje, Post y Comentario para encontrar y eliminar el recurso seleccionado. Para esto se utiliza la función remove de PyMongo, que elimina el recurso denunciado de la base de datos (Figura 102).

```
@admin.route('/admin/denuncias/eliminar/<string:nombre>/<string:identidad>', methods = ["GET"])
def admin_denuncias_eliminar(nombre,identidad):

    with Mongo:
        Mensaje.remove({Mensaje._id : ObjectId(identidad) })
        Post.remove({Post._id : ObjectId(identidad) })
        Comentario.remove({Comentario._id : ObjectId(identidad) })


    return redirect(url_for('admin.admin_denuncias', nombre = nombre))
```

Figura 102. Borrar denuncias

Obviar denuncias

Si el administrador obvia la denuncia, la aplicación envía a través de un formulario el id del mensaje en el parámetro <string:identidad>. Esta notación indica que es de tipo string y que el nombre del parámetro es identidad.

Después busca en las colecciones ***Mensaje, Post y Comentario*** y actualiza el campo *denuncias* a ***False***. Esto se hace a través el método update y la opción 'set' que modifica el valor de un campo dentro de una llamada al método update de PyMongo (Figura 103).

```
@admin.route('/admin/denuncias/obviar/<string:nombre>/<string:identidad>', methods = ["GET"])
def admin_denuncias_obviar(nombre,identidad):

    with Mongo:
        Mensaje.update({Mensaje._id : ObjectId(identidad) },{'$set': {Mensaje.denuncia: False}})
        Post.update({Post._id : ObjectId(identidad) },{'$set': {Post.denuncia: False}})
        Comentario.update({Comentario._id : ObjectId(identidad) },{'$set': {Comentario.denuncia: False}})


    return redirect(url_for('admin.admin_denuncias', nombre = nombre))
```

Figura 103. Obviar denuncia

Borrar archivos

Se elige el archivo por el nombre y se puede borrar pulsando sobre el botón borrar. A través de los tres input select, dispuestos en la parte superior de la página, el usuario filtra archivos por facultad, titulación y asignatura. Una vez haya elegido el archivo puede eliminarlo haciendo clic sobre el botón de color verde "Borrar" (Figura 104).

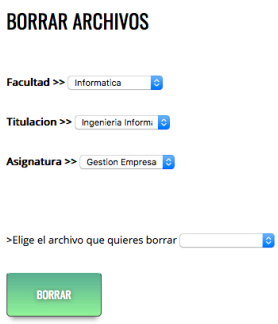


Figura 104. Borrar Archivos

El administrador puede borrar archivos filtrándolos por facultad, titulación y asignatura.

Dichos filtros se aplican a través de funciones AJAX. Después elige el archivo que quiera borrar a través de un formulario de tipo post, que envía la información a admin.py que elimina dicho archivo de la colección Archivo a través de su id que lo identifica de forma única usando el método delete_one de PyMongo (penúltima línea de la Figura 105).

```
@admin.route('/admin/archivo/borrar', methods = ["GET","POST"])
def admin_archivo_borrar():
    if request.method == 'POST':
        nombre = request.form['nombre']
        archivos = []
        borrar = request.form['borrar']
        print(borrar)
        with Mongo:
            u = Usuario.find_one({Usuario._id : ObjectId(nombre)})
            archivo = Archivo.find_one({Archivo._id : ObjectId(borrar) })
            asig = Asignatura.find_one({Asignatura._id: archivo.id_asignatura})
            numero = asig.num_ficheros - 1;
            for i in range(len(asig.nombre_ficheros)):
                if asig.nombre_ficheros[i] != nombre:
                    archivos.append(asig.nombre_ficheros[i])
            Asignatura.update({Asignatura._id: archivo.id_asignatura},{'$set': {Asignatura.num_fichero
            Asignatura.update({Asignatura._id: archivo.id_asignatura},{'$set': {Asignatura.nombre_fich
            Archivo.delete_one({Archivo._id: archivo._id})

        return redirect(url_for('admin.admin_archivos' , nombre = u._id))
```

Figura 105. Borrar archivo

Bloquear usuario

A través de peticiones AJAX se filtran los usuarios como se muestra en la Figura 105. Después se elige el usuario que se identifica a través del nombre, y una fecha que es elegida a través de un datepicker (Figura 106).

La herramienta datepicker está dentro del paquete de Jquery. Es una forma rápida y sencilla de implementar un calendario en el que el usuario pueda elegir una fecha concreta y esta se transforma en un string que se aloja en el input del formulario. Una vez hecho esto se envían ambos parámetros a través de un formulario post y se actualiza el campo "activo_desde" a la fecha que se haya dado como parámetro.

Esto implica que cuando el usuario bloqueado intente acceder a la aplicación través de la comprobación que se muestra en la Figura 107, si el usuario está bloqueado hasta una fecha posterior a la actual la aplicación deniega el acceso hasta que finalice el bloqueo (Figura 108).

```
<script type="text/javascript">
jQuery(function($){
    $.datepicker.regional['es'] = {
    closeText: 'Cerrar',
    prevText: '&#x3c;Ant',
    nextText: 'Sig&#x3e;',
    currentText: 'Hoy',
    monthNames: ['Enero','Febrero','Marzo','Abril','Mayo','Junio',
    'Julio','Agosto','Septiembre','Octubre','Noviembre','Diciembre'],
    monthNamesShort: ['Ene','Feb','Mar','Abr','May','Jun',
    'Jul','Ago','Sep','Oct','Nov','Dic'],
    dayNames: ['Domingo','Lunes','Martes','Mi&eacute;rcoles','Jueves','Viernes','S&aacute;bado'],
    dayNamesShort: ['Dom','Lun','Mar','Mi&eacute;','Juv','Vie','S&aacute;b'],
    dayNamesMin: ['Do','Lu','Ma','Mi','Ju','Vi','S&aacute;'],
    weekHeader: 'Sm',
    dateFormat: 'yy-mm-dd',
    firstDay: 1,
    isRTL: false,
    showMonthAfterYear: false,
    yearSuffix: ''};
$.datepicker.setDefaults($.datepicker.regional['es']);
});

$(document).ready(function() {
$("#datepicker").datepicker();
});
```

Figura 106. Datepicker Jquery

```
d = datetime.date.today()
date = d.isoformat()
```

```
elif values["password"] == usuario.password and usuario.role == "Usuario" and usuario.activo_desde<=date:
```

Figura 107. Datetime

```
@admin.route('/admin/bloquear_usuario/titulacion/_search')
def search_usuarios_t():
    activos = []
    bloqueados = []
    with Mongo:
        titul = Titulacion.find({Titulacion.nombre: request.args.get('busqueda')})
        t = Titulacion.find_one({Titulacion.nombre : request.args.get('busqueda')})
        usuarios = Usuario.find({Usuario.titulacion: t.nombre})
        for usuario in usuarios:
            if usuario.activo == True:
                activos.append(usuario.nombre)
            else:
                bloqueados.append(usuario.nombre)
    a=""
    b=""
    for t in activos:
        a += "<option value= "+ t +" >"+t +"</option>"
    for t in bloqueados:
        b += "<option value= "+ t +" >"+t+"</option>"
    return jsonify(activar=b,bloquear=a)


@admin.route('/admin/desbloquear_usuario', methods = ["GET","POST"])
def admin_desbloquear():
    if request.method == 'POST':
        nombre = request.form['nombre']
        usuario = request.form['activar']
        d = datetime.date.today()
        date = d.isoformat()
        with Mongo:
            u = Usuario.find_one({Usuario._id : ObjectId(nombre)})
            Usuario.update({Usuario.nombre : usuario },{'$set': {Usuario.activo: True}})
            Usuario.update({Usuario.nombre : usuario },{'$set': {Usuario.activo_desde: date }})
        return redirect(url_for('admin/admin_usuarios', nombre = u._id))
```

Figura 108. Bloquear usuario

Desbloquear usuario

En esta sección, el administrador tiene la opción de bloquear y desbloquear usuarios. La página consta de dos select en la parte superior para elegir la facultad y la titulación y dos formularios, el de la derecha es para bloquear usuarios, se debe elegir el nombre del usuario de la lista del input select y al hacer click sobre el input de la parte inferior se despliega un calendario donde se puede elegir la fecha hasta la cuál se quiere bloquear al usuario.

Para desbloquear usuarios hay que utilizar el formulario de la derecha eligiendo del select entre las posibles opciones (Figura 109).

Se da la oportunidad de bloquear a un usuario en el caso de que este haya infringido alguna de las normas de uso de la aplicación, escribiendo comentarios ofensivos o escribiendo comentarios inapropiados a otro usuario de la web o haya subido archivos que no tengan relación con la asignatura vinculada, etc. …

Figura 109. Administrar usuarios

En este caso simplemente se cambia el valor del campo “activo_desde” a la fecha actual, a través de la función de python ***datetime.date.today()*** y después dando formato con ***isofromat()*** (Figura 110).

Al hacer este cambio, cuando el usuario intente acceder a la aplicación de nuevo se comprobará la fecha y se permitirá el acceso.

```python
@admin.route('/admin/desbloquear_usuario', methods = ["GET","POST"])
def admin_desbloquear():
    if request.method == 'POST':
        nombre = request.form['nombre']
        usuario = request.form['activar']
        d = datetime.date.today()
        date = d.isoformat()
        with Mongo:
            u = Usuario.find_one({Usuario._id : ObjectId(nombre)})
            Usuario.update({Usuario.nombre : usuario },{'$set': {Usuario.activo: True}})
            Usuario.update({Usuario.nombre : usuario },{'$set': {Usuario.activo_desde: date }})
        return redirect(url_for('admin/admin_usuarios', nombre = u._id))
```

Figura 110. Desbloquear usuario

Borrar enlaces

Elige el enlace por el nombre y puede borrarlo pinchando el botón borrar. A través de los dos input select, dispuestos en la parte superior de la página, el usuario filtra enlaces por facultad y titulación. Una vez haya elegido el enlace puede eliminarlo haciendo clic sobre el botón de color verde "Borrar" (Figura 111).

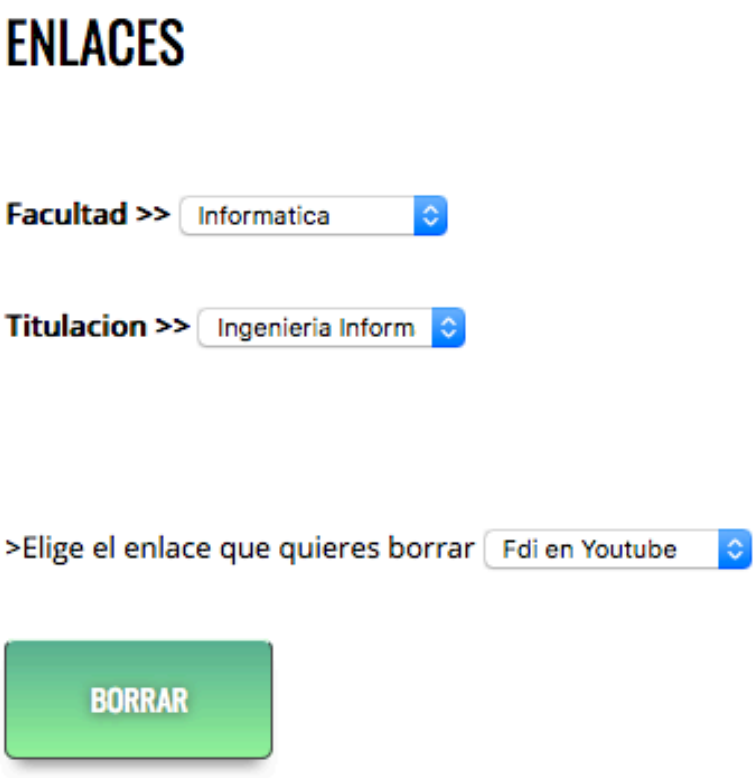


Figura 111. Borrar Enlaces

Dichos filtros se implementan a través de funciones AJAX. Después se elige el enlace que se quiere borrar a través de un formulario de tipo post. Se envía la información a admin.py el cual elimina dicho enlace de la colección Enlaces a través de su id (que lo identifica de forma única) usando el método delete_one de PyMongo (Figura 112).

```
@admin.route('/admin/enlaces/borrar', methods = ["GET","POST"])
@login_required
@required_roles('Admin')
def admin_enlaces_borrar():
    if request.method == 'POST':
        nombre = request.form['nombre']
        borrar = request.form['borrar']
        with Mongo:
            u = Usuario.find_one({Usuario._id : ObjectId(nombre)})
            Enlace.delete_one({Enlace._id: ObjectId(borrar)})

        return redirect(url_for('admin.admin_enlaces' , nombre = u._id))
```

Figura 112. Borrar enlace

Añadir enlace

Se rellena el formulario situado en la parte derecha bajo el título “Añadir enlace”, en el cuál se deben rellenar el nombre, la dirección url y seleccionar la titulación a la que va a pertenecer el nuevo enlace (Figura 113).

Figura 113. Añadir Enlace

Una vez completado el formulario se debe pulsar el botón finalizar para enviar el formulario al servidor el cuál guardará el nuevo enlace en la colección Enlaces (Figura 114).

```
@admin.route('/admin/enlaces/borrar', methods = ["GET","POST"])
@login_required
@required_roles('Admin')
def admin_enlaces_borrar():
    if request.method == 'POST':
        nombre = request.form['nombre']
        borrar = request.form['borrar']
        with Mongo:
            u = Usuario.find_one({Usuario._id : ObjectId(nombre)})
            Enlace.delete_one({Enlace._id: ObjectId(borrar)})

        return redirect(url_for('admin.admin_enlaces' , nombre = u._id))
```

Figura 114. Añadir enlace

8. Conclusiones y trabajo futuro

Las principales conclusiones de este trabajo fin de grado son:

- Se ha desarrollado una aplicación web sencilla e intuitiva que permite la gestión de recursos formativos como son exámenes, apuntes y prácticas. Además, dispone de otras secciones que permiten al alumno configurar opciones de personalización tales como crear notas personales o editar un horario de las clases que tiene durante el curso, lo que hace de la experiencia del usuario más satisfactoria.
- La aplicación facilita al usuario el acceso a los recursos almacenados en el sistema de una forma fácil, estando los recursos categorizados por asignatura. Además, permite la descarga de los recursos para su uso personal, o la inserción de comentarios y valoraciones sobre un recurso.
- Se ha implementado una red social que contribuye a la comunicación entre los usuarios incentivando las relaciones personales. Permite al usuario comunicarse con otros usuarios que pertenecen a su misma facultad a través de la publicación de posts o el intercambio de mensajes de forma privada.
- La aplicación dispone de una interfaz para administrar todo el contenido. Admite añadir cualquier facultad, con toda la información relacionada, como titulaciones, asignaturas, recursos o enlaces. También permite la gestión de usuarios y la moderación de comentarios y mensajes.

Se puede acceder a todo el código de la aplicación web implementada a través del repositorio de git: https://www.github.com/DColdei/TFG.

Algunas mejoras que se podrían realizar en la aplicación son:

- Utilizar el módulo Flask_Mail [7] de Flask para validar que el correo que el usuario ha enviado para registrarse es realmente válido y está en funcionamiento.
- Utilizar alguna plantilla de diseño de Bootstrap[24] de manera que la aplicación sea web_responsive y pueda verse correctamente en todo tipo de dispositivos.
- Facilitar la interface de usuario en otros idiomas diferentes al castellano para facilitar la internacionalización de la aplicación.

- Mejorar la búsqueda de asignaturas utilizando alguna herramienta como elasticsearch[26] que permite realizar búsquedas flexibles.
- Añadir un buscador de recursos por tags para encontrar los archivos de una manera más simple.
- Extender la aplicación con otras funcionalidades tales como la inserción de videos o la posibilidad de visualizar y utilizar el twitter de la facultad en tiempo real utilizando la API de Twitter[25].

Por último, las principales contribuciones personales al proyecto han sido:

- El proyecto se ha elaborado de manera individual.
- Durante el desarrollo del proyecto se ha experimentado lo que es el proceso de elaborar una aplicación desde el principio, empezando por pensar que tipo de aplicación se quería, planificar el diseño y después implementar la funcionalidad de la aplicación.
- Se ha aprendido la utilización de algunas herramientas no vistas en la carrera tales como JQuery, AJAX ,CSS, MongoDB o Python.

8. Conclusions and future work

The main conclusions of this degree final project are:

- A simple and intuitive web application has been developed that allows the management of training resources such as exams, notes and practices. In addition, it has other sections that allow students to configure personalization options such as creating personal notes or editing a schedule of the classes they have during the course, which makes the user experience more satisfactory.
- The application facilitates the user access to resources stored in the system in an easy way, with the resources categorized by subject. In addition, it allows the downloading of resources for personal use, or the insertion of comments and evaluations on a resource.
- A social network has been implemented that contributes to communication among users by encouraging personal relationships. The user has the option of communicating with other users who are in the same faculty through the publication of messages or the exchange of messages privately.
- The application has an interface to manage all the content. It admits adding any faculty, with all the related information, such as degrees, subjects, resources or links. It also allows user management and moderation of comments and messages.

It is possible to accede to all the code of the web application implemented through the git repository: https://www.github.com/DColdei/TFG.

Some improvements that could be made in the application are:

- Use the Flask module, Flask_Mail [7] to validate that the mail that the user has sent to register is really valid and is working.
- Use some Bootstrap design template [24] so that the application is web_responsive and can be seen correctly on all types of devices.
- Facilitate the user interface in languages other than Spanish to facilitate the internationalization of the application.

- Improve the search of subjects using a tool such as elasticsearch [26] that allows flexible searches.
- Add a resource search by tags to find the files in a simpler way.
- Extend the application with other functionalities such as the insertion of videos or the possibility of viewing and using the faculty twitter in real time using the Twitter API [25].

Finally, the main personal contributions to the project have been:

- The project has been drawn up individually.
- During the development of the project, we have experienced what is the process of developing an application from the beginning, starting with thinking about what type of application was wanted, planning the design and then implementing the functionality of the application.
- We have learned the use of some tools not seen in the race such as JQuery, AJAX, CSS, MongoDB or Python.

Bibliografía

1. AJAX Introduction [Internet]. [citado 19 de mayo de 2018]. Disponible en: https://www.w3schools.com/xml/ajax_intro.asp

2. Blueprints — Explore Flask 1.0 documentation [Internet]. [citado 1 de diciembre de 2017]. Disponible en: http://exploreflask.com/en/latest/blueprints.html

3. Changelog — HumbleDB 5.6.1 documentation [Internet]. [citado 2 de diciembre de 2017]. Disponible en: https://humbledb.readthedocs.io/en/latest/changes.html

4. CSS Tutorial [Internet]. [citado 19 de mayo de 2018]. Disponible en: https://www.w3schools.com/css/

5. Estándares - W3C España [Internet]. [citado 16 de mayo de 2018]. Disponible en: https://www.w3c.es/estandares/

6. Flask-Login — Flask-Login 0.4.1 documentation [Internet]. [citado 16 de mayo de 2018]. Disponible en: https://flask-login.readthedocs.io/en/latest/

7. flask-mail — Flask-Mail 0.9.1 documentation [Internet]. [citado 2 de diciembre de 2017]. Disponible en: https://pythonhosted.org/Flask-Mail/

8. Font Awesome [Internet]. [citado 12 de mayo de 2018]. Disponible en: https://fontawesome.com/

9. HTML Tutorial [Internet]. [citado 19 de mayo de 2018]. Disponible en: https://www.w3schools.com/html/

10. Infinite Scroll [Internet]. [citado 12 de mayo de 2018]. Disponible en: https://infinite-scroll.com/

11. JavaScript [Internet]. [citado 19 de mayo de 2018]. Disponible en: https://www.javascript.com/

12. jQuery.ajax() | jQuery API Documentation [Internet]. [citado 19 de mayo de 2018]. Disponible en: http://api.jquery.com/jquery.ajax/

13. MongoDB Documentation [Internet]. [citado 16 de mayo de 2018]. Disponible en: https://docs.mongodb.com/

14. PyMongo 3.6.1 Documentation — PyMongo 3.6.1 documentation [Internet]. [citado 16 de mayo de 2018]. Disponible en: https://api.mongodb.com/python/current/

15. Quickstart — Flask Documentation (0.12) [Internet]. [citado 2 de diciembre de 2017]. Disponible en: http://flask.pocoo.org/docs/0.12/quickstart/

16. Something went wrong - Google Fonts [Internet]. [citado 12 de mayo de 2018]. Disponible en: https://fonts.google.com/

17. Stack Overflow en español [Internet]. [citado 12 de mayo de 2018]. Disponible en: https://es.stackoverflow.com/

18. Template Designer Documentation — Jinja2 Documentation (2.9) [Internet]. [citado 3 de diciembre de 2017]. Disponible en: http://jinja.pocoo.org/docs/2.9/templates/

19. The world's leading software development platform · GitHub [Internet]. [citado 12 de mayo de 2018]. Disponible en: https://github.com/

20. Welcome | Flask (A Python Microframework) [Internet]. [citado 16 de mayo de 2018]. Disponible en: http://flask.pocoo.org/

21. Welcome | Werkzeug (The Python WSGI Utility Library) [Internet]. [citado 16 de mayo de 2018]. Disponible en: http://werkzeug.pocoo.org/

22. Welcome to Python.org [Internet]. [citado 19 de mayo de 2018]. Disponible en: https://www.python.org/

23. Wikipedia, la enciclopedia libre [Internet]. [citado 12 de mayo de 2018]. Disponible en: https://es.wikipedia.org/wiki/Wikipedia:Portada

24. Bootstrap · The most popular HTML, CSS, and JS library in the world. [Internet]. [citado 23 de mayo de 2018]. Disponible en: https://getbootstrap.com/

25. Docs — Twitter Developers [Internet]. [citado 23 de mayo de 2018]. Disponible en: https://developer.twitter.com/en/docs.html

26. Elasticsearch: RESTful, Distributed Search & Analytics | Elastic [Internet]. [citado 23 de mayo de 2018]. Disponible en: https://www.elastic.co/products/elasticsearch

27. Angular [Internet]. [citado 26 de mayo de 2018]. Disponible en: https://angular.io/

28. Apuntes, profesores y trucos para superar la universidad | Patatabrava.com [Internet]. [citado 26 de mayo de 2018]. Disponible en: http://www.patatabrava.com/

29. Encuentra aquí Apuntes, Tareas, Exámenes para tu escuela | Rincón del Vago [Internet]. [citado 26 de mayo de 2018]. Disponible en: https://www.rincondelvago.com/

30. Modelo–vista–controlador - Wikipedia, la enciclopedia libre [Internet]. [citado 26 de mayo de 2018]. Disponible en: https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador

31. Node.js [Internet]. [citado 26 de mayo de 2018]. Disponible en: https://nodejs.org/es/

32. React - A JavaScript library for building user interfaces [Internet]. [citado 26 de mayo de 2018]. Disponible en: https://reactjs.org/

33. UCMovil - Aplicaciones en Google Play [Internet]. [citado 26 de mayo de 2018]. Disponible en: https://play.google.com/store/apps/details?id=es.ucm.ucmovil

34. Unybook - Apuntes universitarios de calidad 100% verificados. [Internet]. [citado 26 de mayo de 2018]. Disponible en: https://unybook.com/

35. Wuolah | Estudiante, gana dinero con tus apuntes . [Internet]. [citado 26 de mayo de 2018]. Disponible en: https://www.wuolah.com/

36. Principios de usabilidad web de Jakob Nielsen: diseño UX | SEMrush community [Internet]. [citado 27 de mayo de 2018]. Disponible en: https://es.semrush.com/blog/usabilidad-web-principios-jakob-nielsen/

Anexo I: Guía de instalación.

Instalación

En primer lugar se debe tener la base de datos MongoDB operativa. Se puede descargar del siguiente enlace: https://www.mongodb.com/, a través de la pestaña de la parte superior "Download. En la pestaña "Community server" se elige el sistema operativo.

Después se debe arrancar MongoDB a través del comando mongod. En el siguiente enlace se dispone de instucciones detalladas para arrancar MongoDB en cualquier sistema operativo. https://docs.mongodb.com/manual/administration/install-community/.

Se adjuntará con el proyecto un volcado de la base de datos para poder utilizar la aplicación la primera vez. La colección esta dentro de la carpeta dump, con el nombre trabajofingrado.

Windows:

mongostore -d nombre_coleccion ruta_dump

Mac/Linux:

mongostore dump/

Instrucciones más detalladas se pueden encontrar en el siguiente enlace:

https://docs.mongodb.com/manual/tutorial/backup-and-restore-tools/

Instalar Python/Flask:

En Mac

Mac tiene una versión de Python por defecto en el sistema.

En Linux

- Instalar Python 2.7 en Linux:

$ sudo apt-get install python2.7

- Instalar pip:

$ sudo easy_install pip

Si no funciona el comando anterior utilizar este:

curl https://bootstrap.pypa.io/get-pip.py > get-pip.py

python get-pip.py --user

$ sudo pip install virtualenv

- Descargar el proyecto través de GoogleDrive o creando un repositorio Git.

- Posicionarse dentro de la carpeta trabajo y ejecutar los siguientes comandos:

$ source bin/activate

$ sudo pip install flask

- Instalar las librerías flask_login y humbledb, así como otras que falten:

$ sudo pip install flask_login

$ sudo pip install humbledb

$ sudo pip install cualquierlibreriaquefalte

- Para arrancar la aplicación:

$ python run.py

Para información más detallada sobre la instalación de Flask:
http://flask.pocoo.org/docs/0.12/installation/

En Windows:

- Instalar Python2.7. Se puede obtener el descargable a través de su página oficial https://www.python.org/downloads/.
- Se añade la ruta al PATH del sistema.
- Instalar pip a través del comando easy_install pip

Si no funciona utilizar este comando:

curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py

python get-pip.py

- Ejecutar el comando pip install virtualenv
- Descargar el proyecto través de GoogleDrive o creando un repositorio Git.
- Crear un entorno virtual con el siguiente comando: $virtualenv trabajo
- Copiar la carpeta tfg y el archivo run.py, que se encuentran dentro de la carpeta trabajo que contiene el proyecto, dentro de la carpeta de trabajo que se acaba de crear.
- A continuación ejecutar el siguiente comando para activar el entorno virtual:

> $ trabajo/Scripts/activate

- Después $ cd trabajo
- Luego se debe instalar Flask a través de pip: pip install flask.
- Instalar las librerías flask_login y humbledb.

$ pip install flask_login

$ pip install humbledb

- Después se ejecuta el comando python run.py
- Después se debe escribir http://127.0.0.1:5000/ en un navegador web y se tendrá acceso a la landing page de la aplicación.
- Una vez arrancada la aplicación se puede acceder a la aplicación como administrador utilizando el usuario : Admin, password : pass1234.
- Para acceder como usuario se puede utilizar el usuario : User, password: pass1234 para ver un usuario con información ya existente. Si no se puede registrar un nuevo usuario.

Más información sobre el proceso de instalación en:

- http://timmyreilly.azurewebsites.net/python-flask-windows-development-environment-setup/
- https://www.solvetic.com/tutoriales/article/1417-instalar-y-configurar-flask/

Anexo II: Guía de Uso

La landing page (Figura 115) da acceso a la aplicación. Para entrar se debe autenticar el usuario o registrar un nuevo usuario para tener acceso.

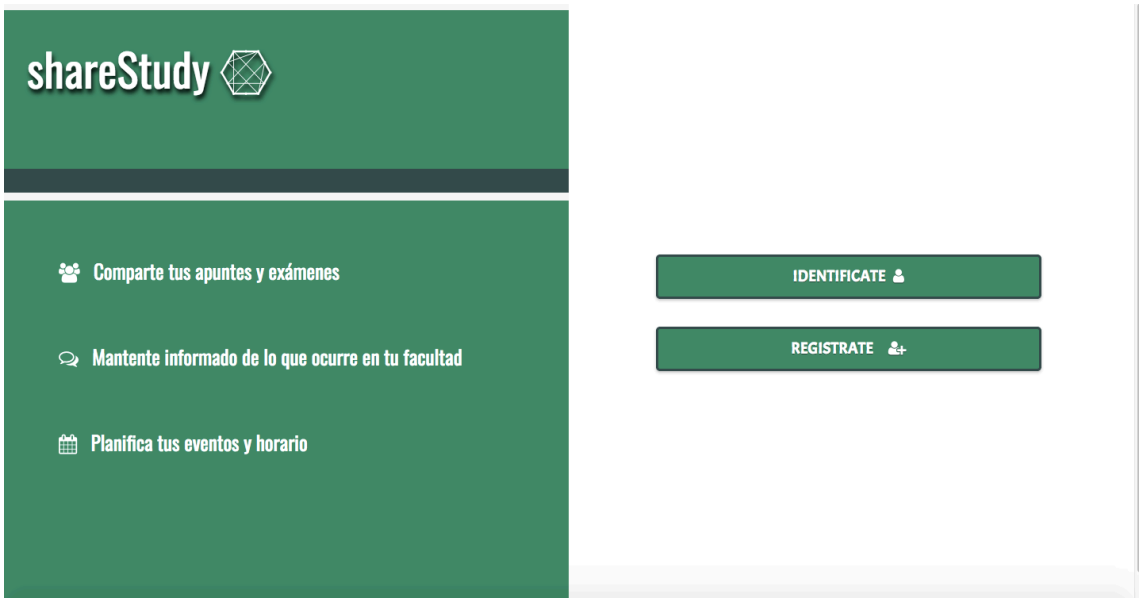


Figura 115. Landing Page

Se utiliza el login para reconocer a los usuarios que ya están registrados dentro del sistema. Consta de dos cajas de texto donde introducir un nombre de usuario y una contraseña. Dispone de una ayuda visual implementado en jquery, indicando qué hay que introducir en cada campo (Figura 116).

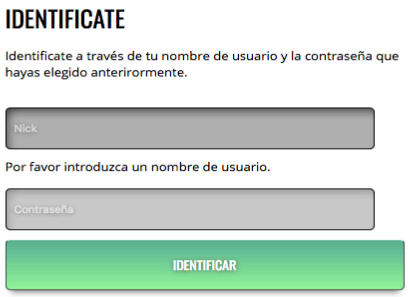


Figura 116. Identificación

Perfil

Se puede acceder al perfil a través del menú superior, pulsando en “Perfil”. Desde aquí se pueden ver todas las estadísticas del usuario, acceder al buzón personal a través del icono con forma de sobre o cambiar parte de la información suministrada por el usuario (Figura 117).

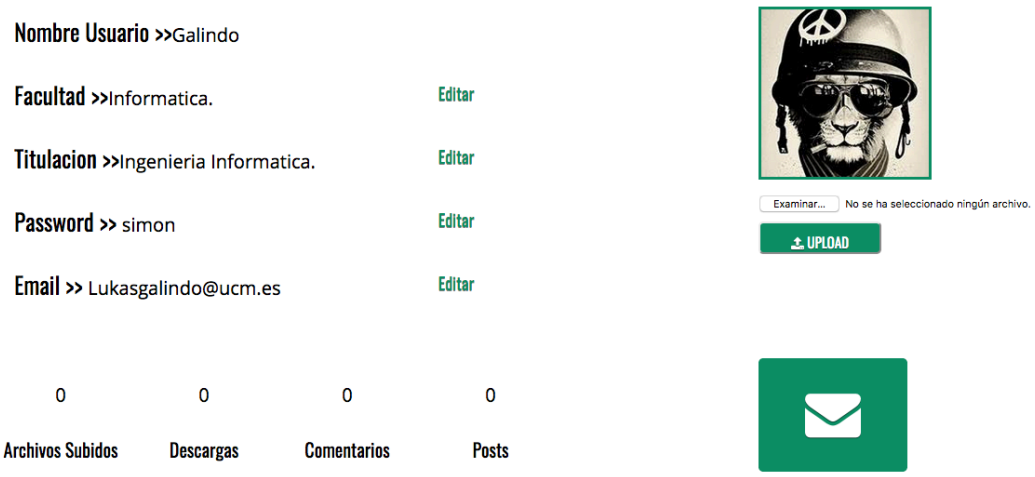


Figura 117. Perfil

Búsqueda de asignaturas

Una vez está realizado el login se accede a la página principal desde la cuál se puede acceder al buscador de asignaturas a través del menú de la aplicación. Tiene dos opciones: buscar la asignatura por curso o por nombre. Para realizar la búsqueda por curso existe un campo select con todos los cursos disponibles para la titulación del usuario, para buscar mediante el nombre una asignatura se debe escribir dentro del campo de la zona central de la página.

Las asignaturas que son resultado de la búsqueda se reflejan en la parte inferior de la página en forma de cajas de color verde con el nombre de las asignaturas en color blanco como se ve en la figura 118, y el número de archivos asociados a cada asignatura.

También el buscador dispone de una sección para resaltar los últimos archivos subidos para cualquier asignatura relacionada con esa titulación.

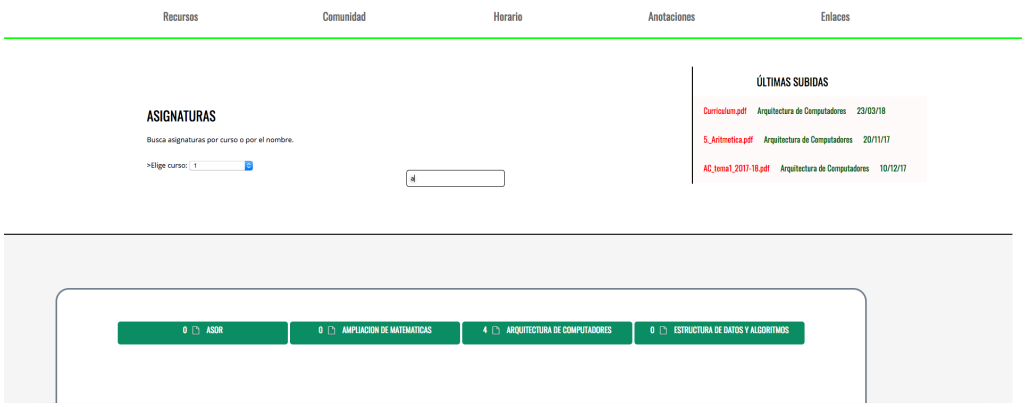


Figura 118. Búsqueda de asignaturas

Ver Asignatura

Una vez se ha elegido la asignatura en el paso anterior se puede pinchar sobre una de las asignaturas y acceder a una pantalla donde se pueden ver todos los archivos vinculados con esa asignatura, distribuidos por categorías, apuntes, exámenes y ejercicios. Para acceder a cada uno de los recursos solo hay que pinchar sobre el archivo o recurso que se quiera (Figura 119).

El usuario puede subir archivos a la aplicación eligiéndolos de su sistema pulsando al botón examinar, eligiendo una categoría y pulsando el botón upload.

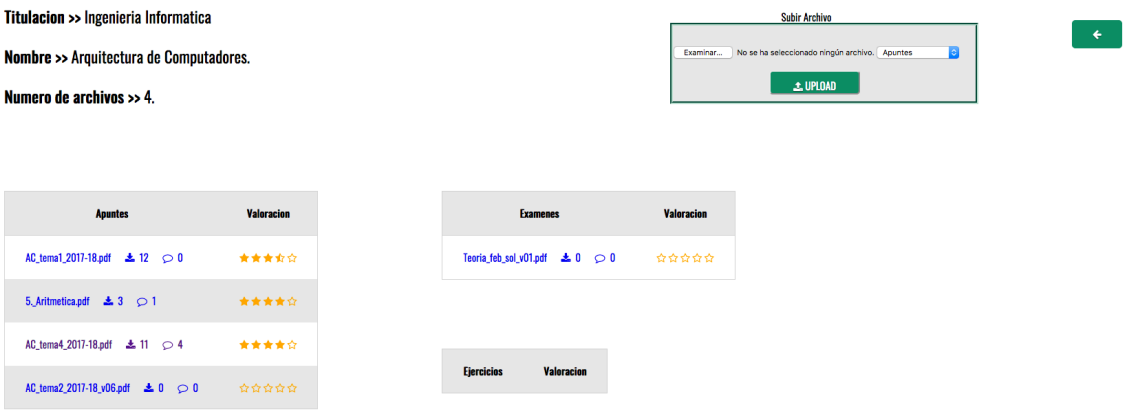


Figura 119. Página de Visualización de asignatura

Recursos

Una vez se haga click sobre el archivo se despliega una nueva página, en ella se da información detallada: nombre del archivo, número de descargas, fecha de creación, puntuación por parte de los usuarios, nombre y avatar del usuario responsable. También existe la opción, a través del botón que pone "Descargar", de ver o descargar el archivo.
Para puntuar el archivo basta con elegir el número de estrellas pasando por encima del grupo de estrellas de la parte inferior (Figura 120).
Cada recurso dispone de una sección de comentarios(Figura 122) en la parte inferior de la página. Allí se encuentran todos los comentarios que se han realizado por parte de otros usuarios ordenados según la fecha de creación. Cada comentario tendrá un cuerpo, y una cabecera donde se incluirá toda la información relevante acerca del comentario y del autor de este.

Figura 120. Página de Visualización de recurso

Al hacer click en el botón de la zona de comentarios que dice "Añadir comentario" aparecerá un formulario con dos campos: un input tipo text para introducir el titulo y una caja de texto para el cuerpo del comentario. El usuario finaliza su comentario al pulsar

sobre el botón "Añadir comentario"(Figura 121). Después se recarga la página y se añade el comentario al inicio de la lista de comentarios (Figura 122).

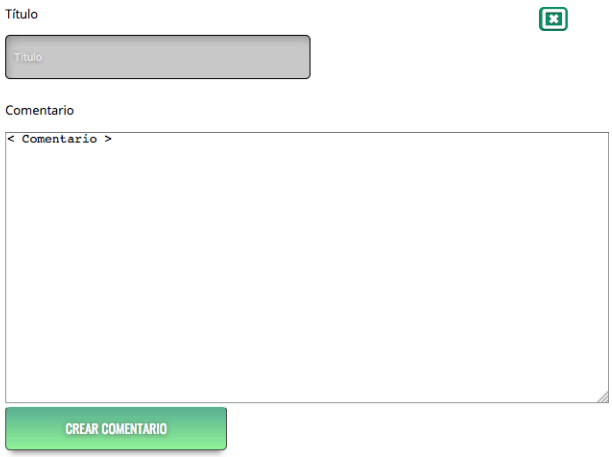


Figura 121. Crear comentario de un recurso

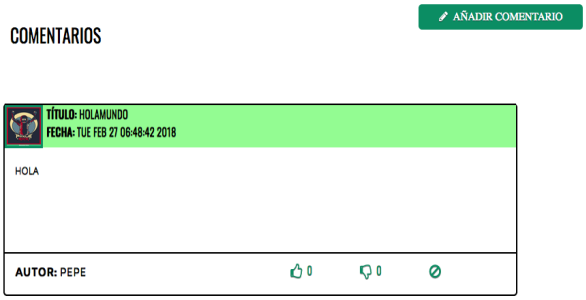


Figura 122. Zona de comentarios

Se puede acceder a la información de otro usuario en cualquiera de los recursos que haya generado como mensajes, posts, subida de archivos o mensajes. Pinchando sobre el avatar se accederá a la sección donde se detalla la información de dicho usuario y sus estadísticas en la aplicación. También es posible acceder a esta sección a través de los ranking de la zona comunidad.

Horario

El usuario puede acceder a la sección de horario pulsando sobre la opción horario del menú superior. Una vez allí (Figura 123) podrá ver todas las asignaturas que tiene programadas en su horario.

El horario se divide en mañana y tarde y se puede pasar de una a otra pulsando los botones que están sobre el mismo horario y que indican mañana y tarde.
El horario está dividido en horas y representan el horario laborable del usuario.
Las filas del horario tienen colores alternos para poder diferenciarlas con claridad. La cabecera del horario permanece fija una vez ha alcanzado el tope superior con el menú de la aplicación para poder ver siempre los días de la semana.

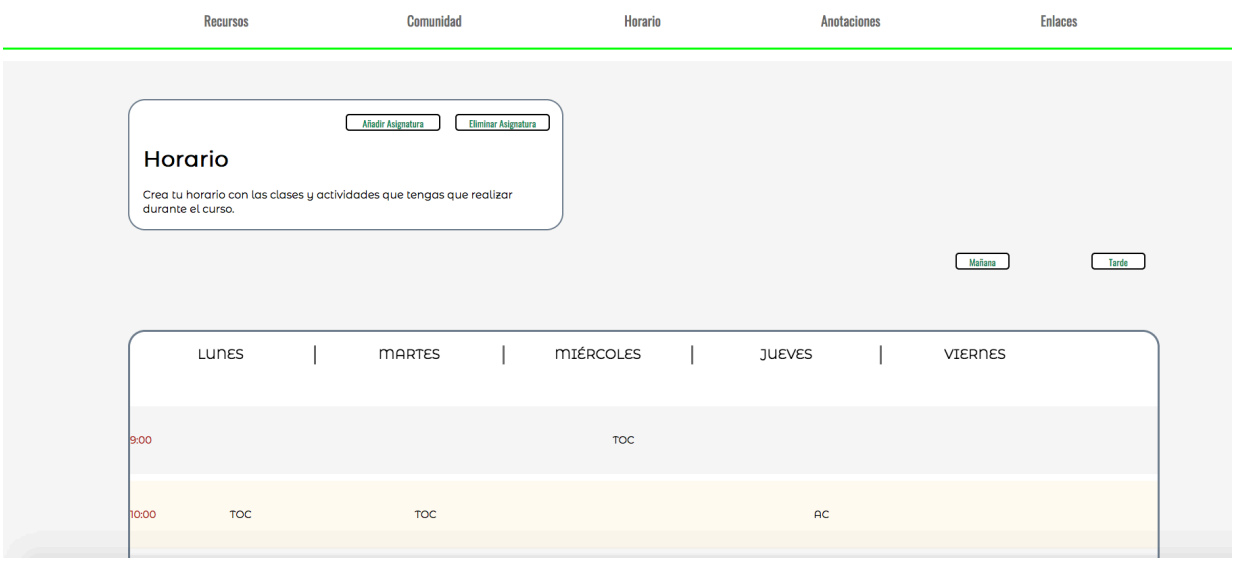


Figura 123. Horario

El usuario puede añadir una asignatura indicando el número de horas a la semana, después indica a qué día y hora pertenece cada una de las horas de esa asignatura como se muestra en la figura 124. También existe la posibilidad de eliminar una asignatura del horario (Figura 125).

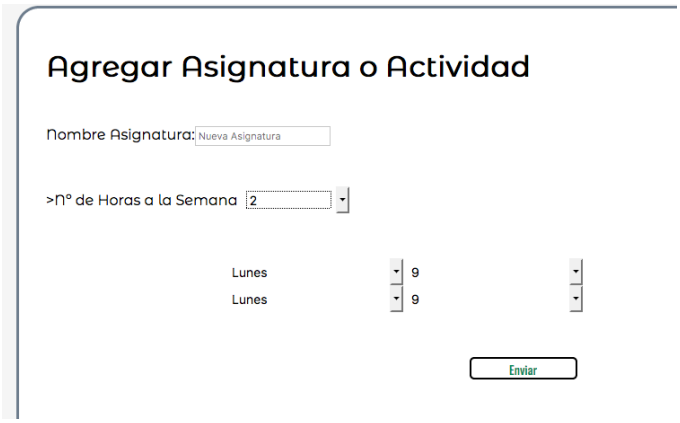


Figura 124. Añadir Asignatura

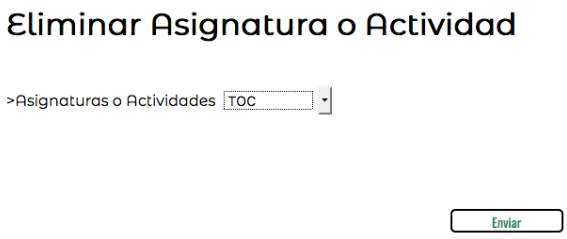


Figura 125. Eliminar Asignatura

Notas

El usuario accede a las notas a través del menú superior. Después puede ver en un muro todas las anotaciones(Figura 128) que ha creado alternadas por distintos colores, y en las que se refleja la razón de la nota, la fecha en la que se producirá el evento y una pequeña descripción.

El usuario puede pinchar sobre alguna de las notas que están en el muro y acceder a ella. Se desplegará la nota sobre el muro de anotaciones con todos los campos de los que dispone. En la parte inferior hay un botón que pone “Borrar Nota” con el dibujo de una papelera al lado que sirve para eliminar la nota (Figura 126).

Figura 126. Ver nota

Si el usuario pulsa “Crear Nota” saldrá un formulario (figura 127) que el usuario tendrá que rellenar. Tendrá que elegir el asunto de la nota y la fecha, la cual desplegará un calendario en el que poder elegir la fecha de realización del evento, y la descripción detallada de la creación de la nota.

Figura 127. Crear Nota

Se pueden filtrar las notas dependiendo del motivo de su creación: exámenes, prácticas o entregas. El filtro se pone de color verde al pasar por encima y de color rojo en el filtro actual (Figura 128).

Figura 128. Página de anotaciones

Comunidad

Ver muro

El usuario tiene acceso a esta sección a través del menú superior pulsando “Comunidad”. Allí podrá observar en la parte izquierda todos los posts publicados para esta titulación de forma descendente. Podrá filtrar dichos posts por su asunto utilizando la lista de filtrar situado en la parte superior. (Figura 129)

Añadir post

El usuario puede añadir un nuevo post al muro haciendo click en el botón crear post de la parte superior de la anterior vista. Una vez pulsado el botón se desplegará el formulario reflejado en la figura 130.

En la parte derecha de la sección de comunidad se puede ver un ranking con el top de las diferentes secciones de la aplicación. Existe un Top de Descargas, otro Top para los archivos subidos y por último una parte con todos lo usuarios de la aplicación.

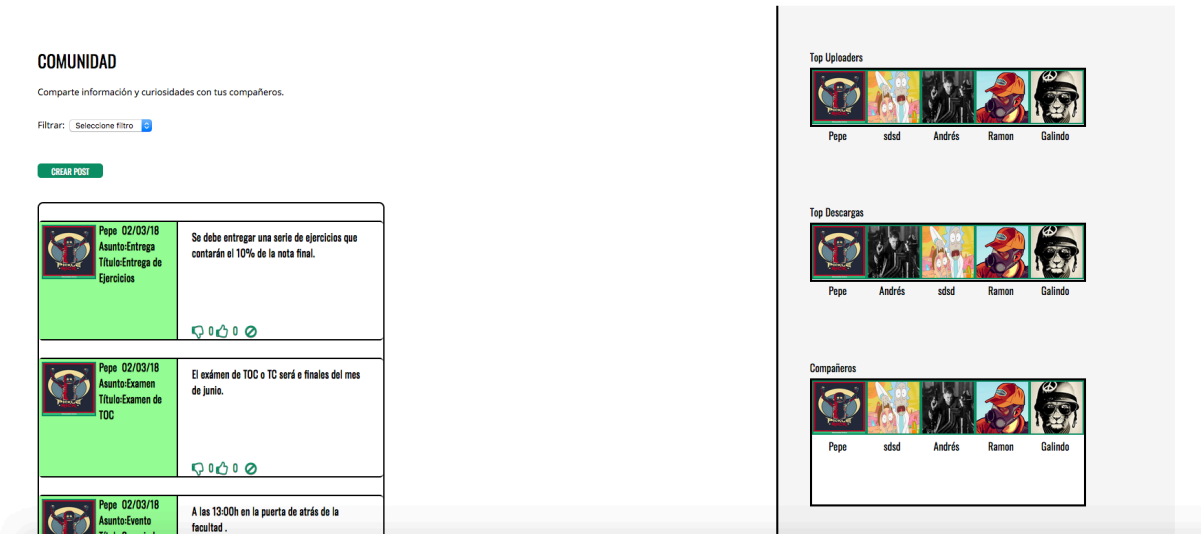
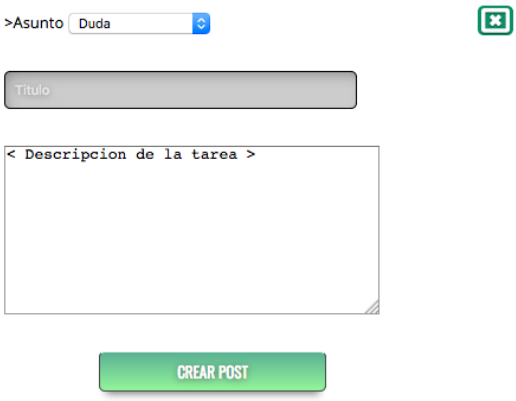


Figura 129. Comunidad

Figura 130. Crear post



Administrador

El administrador debe hacer el login como si fuera un usuario normal desde el mismo formulario de la landing page, la aplicación será la que redirija en función de su rol dentro de la aplicación.

Una vez hecho esto tendrá acceso al menú de la figura 131, y desde aquí ejecutar cualquiera de las opciones disponibles:

- Agregar Facultad
- Modificar Facultad
- Buzón de Pendientes
- Buzón de denuncias
- Archivos
- Usuarios

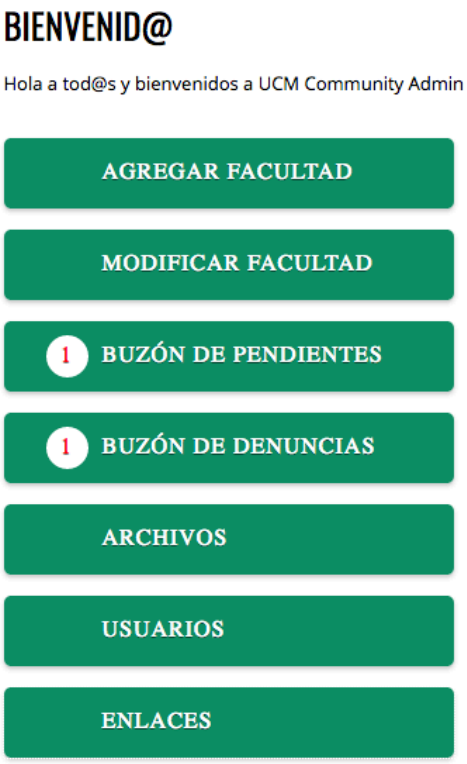


Figura 131. Menú administrador

Agregar facultad

Desde el menú inicial el administrador puede registrar una nueva facultad en la aplicación. Para ello primero debe indicar a través de un formulario el nombre de la facultad y el número de titulaciones ofertadas por dicha facultad (Figura 132).

Una vez se haya pulsado el botón siguiente se desplegará un nuevo formulario en la que el administrador tiene que introducir el nombre de cada titulación y el número de cursos que tendrá.

A continuación, se pasará a otro formulario en el que el administrador indicará el número de asignaturas de cada curso y el nombre de cada una de estas asignaturas.

Figura 132. Registrar Universidad

Modificar Facultad

Para modificar una facultad el administrador deberá desde el menú inicial de administración pulsar sobre la opción "Modificar Facultad". A continuación, debe elegir qué facultad es la que quiere modificar a través de una lista en un formulario (Figura 133).

Figura 133. Elegir Facultad

Una vez hecho aparecerán distintas opciones: borrar la facultad seleccionada, acceder a alguna de las titulaciones ya registradas para modificar algo o agregar una nueva titulación a la facultad (Figura 136).

Desde el menú anterior al pulsar en la opción "Agregar Titulación" el administrador accederá a un formulario similar al presentado durante el proceso de registrar una facultad. Se debe introducir el nombre de la nueva titulación, el número de cursos y el número y nombre de las asignaturas de cada curso.

Para eliminar la titulación es necesario primero acceder a la titulación y después eliminar la titulación desde el menú de la titulación(Figura 134).

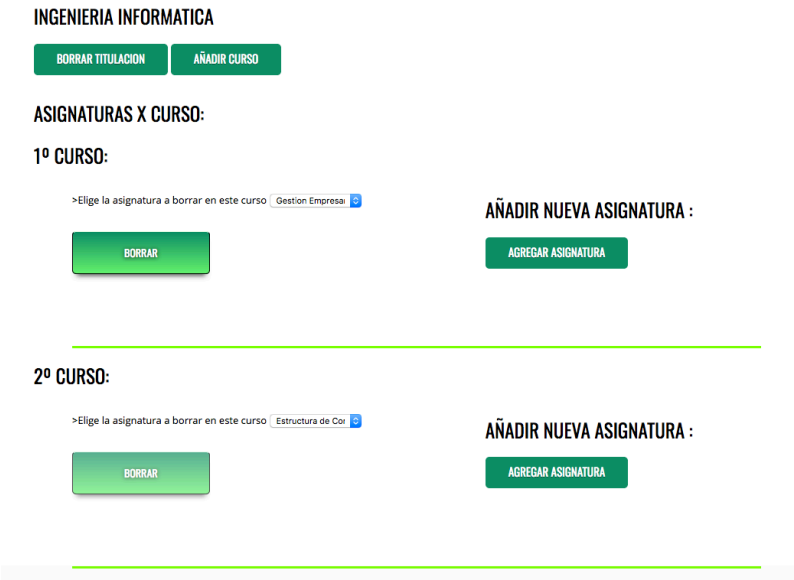


Figura 134. Ver Titulación

Figura 135. Ver Facultad

Para Añadir un Curso a una titulación se pincha sobre la opción "Añadir Curso" (Figura 136) del menú de titulación (Figura 134). Después se debe introducir el número o nombre del curso y el número de asignaturas. Aparecerá un nuevo formulario donde el administrador introducirá todos los nombres de las asignaturas.

Al acceder a una de las titulaciones, una vez se haya seleccionado una de las facultades registradas en la aplicación, aparecerá un listado de todos los cursos (Figura 134) con sus

respectivas asignaturas y la opción en la parte superior de añadir un curso o eliminar la titulación entera.

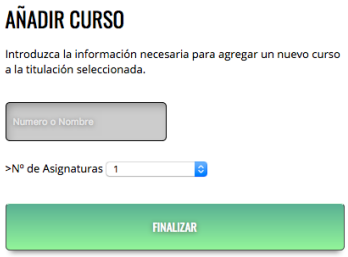


Figura 136. Curso

Si se selecciona la opción añadir asignatura aparecerá un formulario (Figura 137) donde el administrador debe introducir el nombre de la nueva asignatura, después se hará click sobre el botón "Crear Asignatura" y dicha asignatura aparecerá en el listado de asignaturas del curso donde se haya introducido.

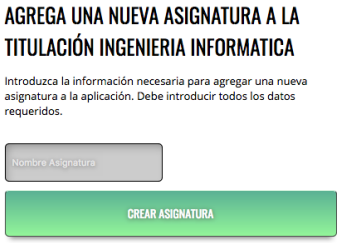


Figura 137: Nueva asignatura

Pendientes

Desde aquí el administrador puede autorizar la subida de archivos a la web por parte de los usuarios. Estos archivos se caracterizan porque aún no se han revisado y autorizado para su difusión dentro de la web (Figura 138).

Figura 138. Archivos pendientes

Buzón de Denuncias

En esta sección el administrador puede ver el buzón de denuncias, donde aparecen todas las denuncias que se han hecho por parte de los usuarios. Está dividido en posts, comentarios y mensajes (Figura 139). Si se encuentra alguna denuncia aún no revisada el administrador tiene la opción de ver dicho recurso, para después poder eliminarlo o archivar la denuncia sin hacer nada.

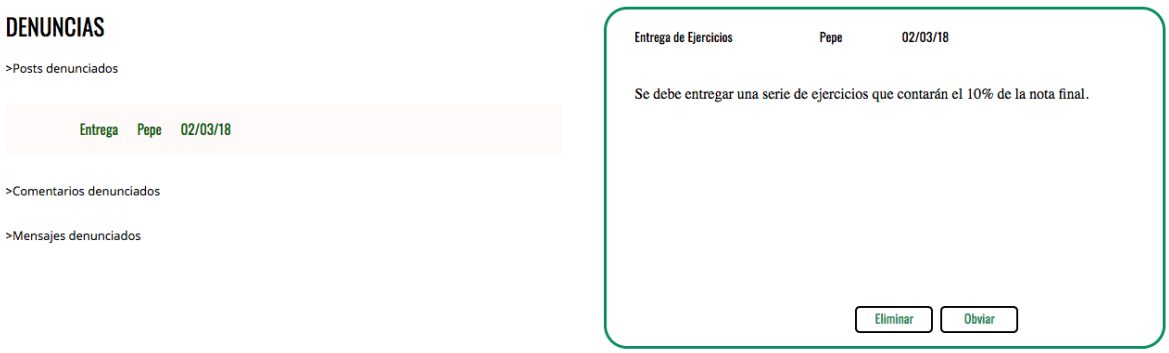


Figura 139. Denuncias

Archivos

Elige el archivo por el nombre y puede borrarlo pinchando el botón borrar. A través de los tres input select dispuestos en la parte superior de la página el usuario filtra archivos por facultad, titulación y asignatura. Una vez haya elegido el archivo puede eliminarlo haciendo click sobre el botón de color verde "Borrar" (Figura 140).

Figura 140. Borrar Archivos

Usuarios

En esta sección el administrador tiene la opción de bloquear y desbloquear usuarios (Figura 141). La página consta de dos select en la parte superior para elegir la facultad y la titulación y dos formularios, el de la derecha es para bloquear usuarios, se debe elegir el nombre del usuario de la lista del input select y al hacer click sobre el input de la parte inferior se despliega un calendario donde poder elegir la fecha hasta la cuál se quiere bloquear al usuario.

Para desbloquear usuarios hay que utilizar el formulario de la derecha eligiendo del select entre las posibles opciones.

Se da la oportunidad de bloquear a un usuario en el caso de que haya infringido alguna de las normas de uso de la aplicación, escribiendo comentarios ofensivos o escribiendo

comentarios inapropiados a otro usuario de la web o haya subido archivos que no tengan relación con la asignatura vinculada, etc. …

Figura 141. Administrar usuarios

Enlaces

En esta sección el administrador tiene la opción de añadir y eliminar enlaces de una titulación determinada (Figura 142). La página consta de dos select en la izquierda de la pantalla que sirven para escoger un enlace de una facultad y una titulación concretas para eliminar dicho enlace de la bb.dd.

Se da la oportunidad de añadir un nuevo enlace rellenando un formulario en el cuál se debe elegir la titulación y rellenar el nombre y la dirección url del enlace.

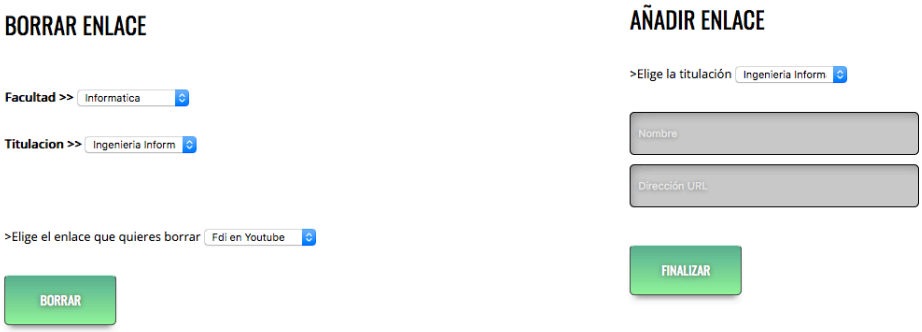


Figura 142. Administrar enlaces