ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
INFORMÁTICA
INGENIERÍA INFORMÁTICA

Desarrollo de un sistema de información para una asociación cultural.

Development of an information system for a cultural association.

Realizado por
José María Racero Sánchez

Tutorizado por
Carlos Manuel Rossi Jiménez

Departamento
Lenguajes y ciencias de la computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, (septiembre 2023)

Resumen

Todo arte necesita un medio para promocionarse al mundo. En estos días es imprescindible tener un medio en el cual dar voz a las diversas actividades que se pretendan realizar, en especial a las que tratan temas relacionados con el arte y la cultura. Por suerte, hoy en día existe el mejor altavoz: Internet.

En este TFG se tomará como caso de estudio las necesidades de una asociación cultural, cuya actividad consiste en el desarrollo de charlas, ciclos, conferencias, seminarios, exposiciones y actividades relacionadas por lo general con el arte y la cultura. Sobre todo, en el ámbito académico.

El mayor problema que presenta esta asociación es la ausencia de un medio donde promocionar, organizar y almacenar todas las actividades anteriormente mencionadas. Para resolver estos problemas se plantea desarrollar una aplicación web a medida que cumpla con los requisitos necesarios para dar soporte a dichas actividades.

El objetivo será mejorar la organización y la promoción de estas actividades para hacer más fácil que la gente participe y se informe de las diferentes actividades anteriormente mencionadas.

Este proyecto se ha desarrollado utilizando la metodología Scrum con sprints de 3 semanas usando principalmente Django, MongoDB, HTML, CSS y Bootstrap.

Palabras claves:

Sistemas de información, Análisis y diseño web, Python, Django, Aplicación web.

Abstract

All kind of art needs a medium to promote itself to the world. These days it is essential to have médium in which give voice to the various activities that are intended to be performed. Especialy those that are related to art and culture. Luckily, nowadays there is the best medium: the Internet.

In this End-of-degree Project, the needs of a cultural association Will be taken as a case study, whose main activities consists of the development of talks, conferences, seminars, exhibitions and activities generally related to the world of art and culture. Above all, in the academic field.

The biggest problem that this association presents is the absence of a way to promote, organize and store all the aforementioned activities.

To solve these problems, it is proposed to develop a customized web page that meets the necessary requirements to support these activities.

The objective Will be to improve the organization and promotion of these activities to make it easier for people to participate and learn about the different activities mentioned above.

This project has been developed using the Scrum methodology with 3-week sprints using mainly Django, MongoDB, HTML, CSS and Bootstrap.

Keywords:

Information systems, web's Analysis and design, Python, Django, Web application

Indice

1

Introducción

1.1. Motivación

La necesidad de tener una plataforma en la cual organizar la información y comunicar las diferentes actividades y eventos que realizas es muy importante para una asociación cultural. Por eso la mejor de abordar este problema es la creación de una web personalizada que dé una solución a esto.

Lo que se pretende en este TFG es crear una aplicación que solucione estos problemas y ayude a esta asociación a mejorar su forma de promocionar sus actividades.

1.2. Objetivos

El objetivo de este proyecto consiste en el análisis, el diseño y la implementación de una web que ayude a esta asociación cultural a promocionar y estructurar sus actividades y su material audiovisual.

Esta web permitirá realizar las siguientes acciones

- Envío automático de correos informativos.

- Subir archivos para que todos los usuarios puedan acceder a ellos.
- Creación de las actividades que realiza esta asociación.
- Capacidad de inscripción en dichas actividades para los usuarios inscritos.
- Capacidad de crear noticias para informar a los usuarios de las últimas novedades y eventos.

1.3. Metodología

Para la gestión del proyecto la metodología de trabajo será Scrum, adaptada a un proyecto desarrollado por una sola persona. De esta forma se abordará una estrategia de trabajo incremental, organizada en diferentes sprints, en los que se desarrollarán las historias de usuario que se identifiquen en las fases previas del proyecto. Se utilizará UML como lenguaje de modelado.

1.4. Entorno tecnológico

Para este proyecto se ha desarrollado utilizando como lenguaje principal Python, usando el framework de Django para el backend y para el frontend se ha utilizado HTML y CSS, utilizando el framework de Bootstrap.
Se ha utilizado RabbitMQ como broker de mensajería para el envío de mensajes asíncronos.
Para la gestión de las tareas del proyecto se ha utilizado Taiga.
El Modelado de la aplicación se ha llevado a cabo con Visual Paradigm.

1.5. Estructura de la memoria

La estructura de la memoria es la siguiente:

- Contexto: en él se detalla el proceso que sigue la asociación a la hora de realizar sus actividades y se evalúan diferentes aplicaciones similares.

- Análisis: se describen los requisitos de la aplicación y se detalla el sistema a implementar. Explicando los casos de usos.
- Diseño: se detalla el proceso de diseño de la aplicación. Explicando los diferentes diagramas relacionados con la aplicación.
- Implementación: se detalla la estructura del proyecto y la implementación de una de sus características principales.
- Conclusiones: pensamientos finales sobre el proyecto.

2

Contexto

En este apartado se va a proceder a describir la idea de negocio y a analizar diferentes aplicaciones que actúan de manera similar a la realizada.

2.1. Descripción de la asociación y de las actividades culturales.

Malas Lenguas es una comunidad de aprendizaje asociada al proyecto llamado las oficinas impulsado por el Área de Artes Visuales y el Área de música de la Madraza, Centro de Cultura Contemporánea de la Universidad de Granada y de la Casa de Porras del Vicerrectorado de Extensión Universitaria y de Patrimonio, con la colaboración de Casa del Estudiante del Vicerrectorado de Estudiantes y Empleabilidad.

Este proyecto tiene como objetivo principal fomentar la divulgación científica y académica tanto dentro como fuera de la comunidad universitaria. Se trata de crear un espacio en el que el estudiantado, ya sean alumnos de grado o de doctorado, puedan compartir sus descubrimientos y avances de forma más accesible y dinámica. A través

de la organización de conferencias, podcasts y otros eventos similares se busca promover el intercambio de conocimientos y el debate de ideas entre los miembros de la comunidad universitaria.

Esta iniciativa busca involucrar a los estudiantes de la universidad en la difusión de sus investigaciones. Se espera que esta actividad no solo contribuya al avance del conocimiento, sino que también permita a los participantes adquirir y mejorar habilidades de comunicación y divulgación que les serán útiles a lo largo de su futura carrera profesional.

En resumen, este proyecto es una oportunidad para que el alumnado de la comunidad universitaria tenga un espacio de encuentro, dialogo y exposición en torno a los diferentes temas que se planteen dentro del marco de la cultura y la investigación.

Malas lenguas provee un espacio inclusivo, abierto y accesible donde cualquier integrante de la comunidad universitaria puede aportar una propuesta.

La presentación de las propuestas es un aspecto fundamental en el funcionamiento de Malas Lenguas.

Las propuestas se dividen en dos grupos diferenciados:

- Propuestas del equipo de Malas Lenguas.
- Propuestas externas al equipo de Malas Lenguas.

Toda propuesta debe enviarse a los canales de comunicación abiertos, en este caso el correo electrónico propio de Malas Lenguas, adjuntando en copia al correo electrónico de Casa de Porras, incluyendo los datos personales requeridos y la titulación a la que se adscribe el interesado.

Toda propuesta recibida será evaluada conforme a una serie de criterios establecidos por la propia Casa de Porras, siempre intentando abarcar un amplio espectro cronológico y temático para dirigirse a la sociedad y al resto del mundo universitario interesado. En todo momento se respetarán las normas de protección de datos de la

propia Universidad de Granada. De esta manera se busca garantizar la privacidad y seguridad de los datos personales adquiridos de los propios usuarios.

Entre las propuestas que se han aceptado desde que Malas lenguas fue creada, se encuentran:

- Exposiciones
- Entrevistas
- Debates
- Seminarios
- Certámenes de poesía

Además, en la era digital en la que nos encontramos, se han aceptado propuestas que utilizan plataformas y redes sociales como herramientas de difusión. Por ejemplo, se han aceptado:

- Hilos de Twitter hablando sobre diferentes temas relacionados con artistas y diferentes figuras del arte.
- Posts de Instagram mostrando información mediante fotos de manera dinámica.

Todos estos posts han sido y serán publicados en las cuentas de Twitter e Instagram (@MalasLenguasDE) de Malas lenguas.

Una vez la propuesta ha sido analizada, se procede a la comunicación con el creador de dicha actividad para que pueda prepararse el desarrollo de la actividad propuesta. Tanto el formato como el contenido de esta.

Malas Lenguas no se interpone en el desarrollo ni en el contenido de las actividades propuestas por los usuarios pertenecientes a la Universidad de Granada para así no perjudicar a la calidad de esta. La actividad al haber sido analizada con anterioridad se sabe que cumple con los requisitos que la Casa de Porras establece y por ende es apta para su posterior realización.

Una vez la propuesta haya sido aceptada, la asociación se comunica con el creador de esta, estableciendo los parámetros de la actividad, siempre, atendiendo a la naturaleza de la propuesta. Por ejemplo, acordando la fecha de una presentación o una conferencia. Indiferentemente de que la actividad sea de manera presencial o no, esta fecha deberá ser elegida de manera que no coincida con ninguna otra actividad cultural programada por La Madraza, Centro de Cultura contemporánea, para así, poder evitar los solapamientos de actividades que afecten a la asistencia del público que esté interesado en las mismas.

Cuando se establece una fecha para la actividad a desarrollar, se procede a la difusión de la actividad mediante los diferentes canales de comunicación activos que haya en el momento. Generalmente se usan las redes sociales de Instagram, Twitter y Facebook para su difusión.

Las propuestas realizadas por el equipo de Malas Lenguas por lo general se basan en la generación de los diferentes posts informativos sobre diferentes temas relacionados con el mundo del arte y la cultura, como los anteriormente mencionados hilos de Twitter y posts de Instagram. Además, también son los organizadores de las entrevistas con diferentes personas o grupos dentro del mundo del arte y la cultura.

Las propuestas realizadas por el equipo de Malas Lenguas siguen el mismo protocolo que las propuestas realizadas por el alumnado de la universidad de Granada.

Para la difusión de las actividades, se elabora un cartel para dicha actividad, visualmente relacionado con la actividad a desarrollar. Siempre incluyendo los logos tanto de Malas Lenguas como de los diferentes centros asociados, en este caso, la Universidad de Granada y la Madraza, centro de cultura contemporánea. A continuación, se van a mostrar algunos de los carteles creados para actividades que en su momento fueron aceptadas y realizadas.

Figura 1: Facebook. Ejemplo de cartel [1]

Toda actividad presencial se realizará en la sede del Palacio del Almirante, siempre y cuando se le haya dado el visto bueno a la actividad y se le haya designado una fecha que así lo permita.

2.2. Modelos de evaluación de aplicaciones similares.

A continuación, se evaluarán una serie de aplicaciones web con similitudes respecto a las necesidades de Malas Lenguas.

El modelo de calificación consistirá en las siguientes partes:

Criterio	Ponderación (1-4)
A. FUNCIONALES	
1. Contenido	
1.1. Gestión de Noticias	3
1.2. repositorio de material	4
1.3. Gestión de Actividades	4
1.4. Gestión de asistencia	2
Subtotal Conenido	
2. Mensajería	
2.1. Gestión de Newsletter	3
2.2. Gestión de notificaciones	3
Subtotal Mensajería	
3. Seguridad	
3.1. Control de Acceso	2
3.2 Gestión de usuarios	3
Subtotal Seguridad	
B. NO FUNCIONALES	
Accesibilidad	4
Rendimiento	4
Adaptabilidad	3
Escalabilidad	2
Personalización	2
Subtotal no funcional	
TOTAL	

Figura 2: Modelo de calificación

2.2.1. CAC – Centro de Arte contemporáneo

Esta web está creada por el ayuntamiento de málaga y financiada por los fondos NextGenerationEU. Fue creada en 2003 y ha estado actualizándose de manera reiterada durante los años. Está creada mediante WordPress y WooCommerce, es decir, sus lenguajes principales son PHP y JavaScript [2]

La funcionalidad de la web es informar de los distintos eventos y exposiciones que se encuentran o se encontrarán disponibles dentro del CAC. Según su web su misión es la siguiente:

El Centro de Arte Contemporáneo de Málaga, tiene como propósito acercar el Arte Contemporáneo a la población, mediante exposiciones culturales para la ciudad de Málaga. [3]

La web sigue una arquitectura Modelo-Vista-Controlador.

Estas son algunas capturas de como se ve la web:

Figura 3: Home CAC

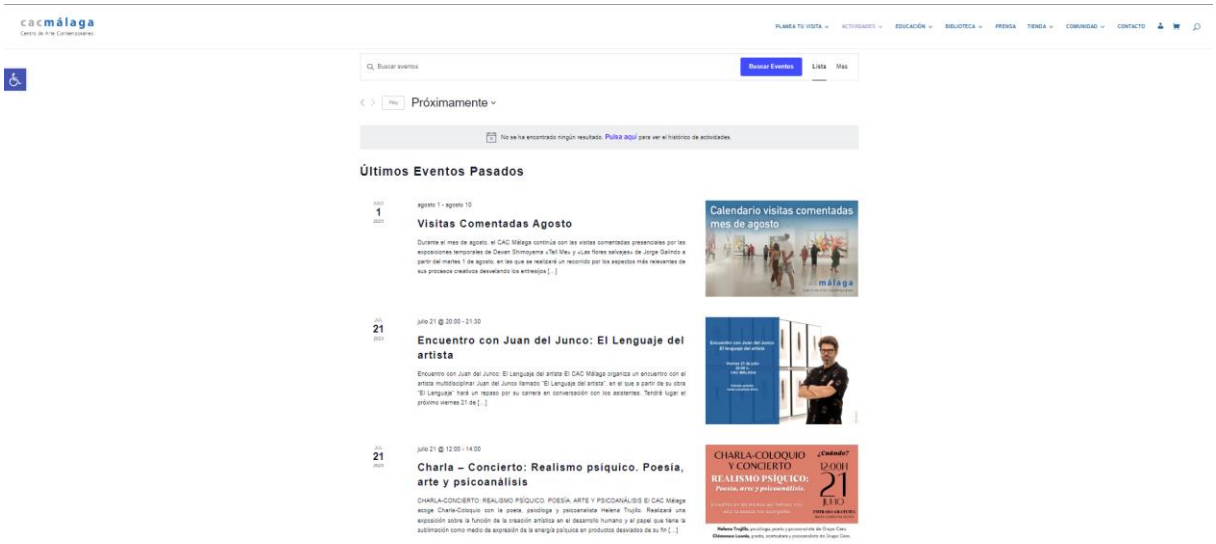

Figura 4: Sección de actividades CAC

El modelo de calificación es el siguiente:

Valoracion CAC (0-100)	Calificación CAC
76	228
54	216
40	160
20	40
	644
82	246
30	90
	336
64	128
57	171
	299
90	360
80	320
74	222
63	126
67	134
	1162
	2441

Figura 5: Modelo calificación CAC

2.2.2. AF - Alianza Francesa

Alianza Francesa ha sido creada por Onion st, una agencia de marketing digital malagueña especializada en WordPress.

Los primeros indicios visibles de la web son de 2023. La web está creada mediante WordPress, utilizando los lenguajes PHP en la parte de backend y JavaScript en la parte de frontend. [4]

La función de la web según su ellos es la siguiente:

Alianza francesa es una organización que se basa en la producción, colaboración y difusión de eventos franceses y francófonos. [5]

Estas son algunas capturas de la web.

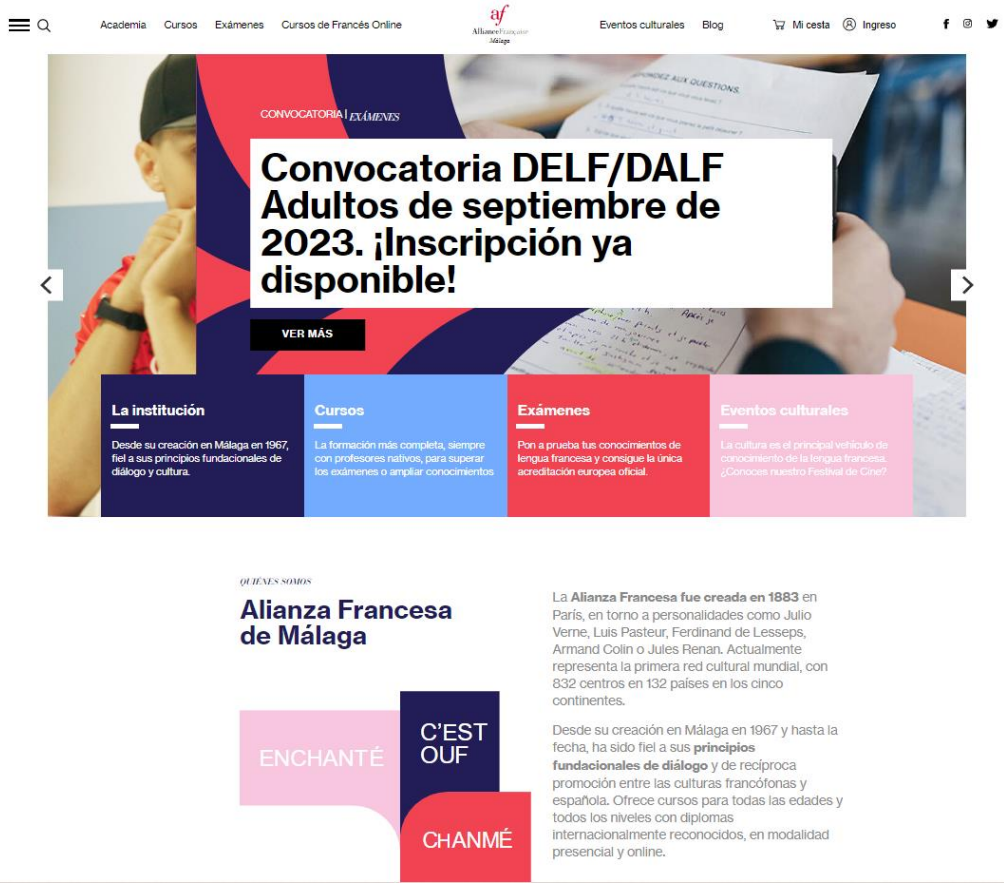

Figura 6:Home AF

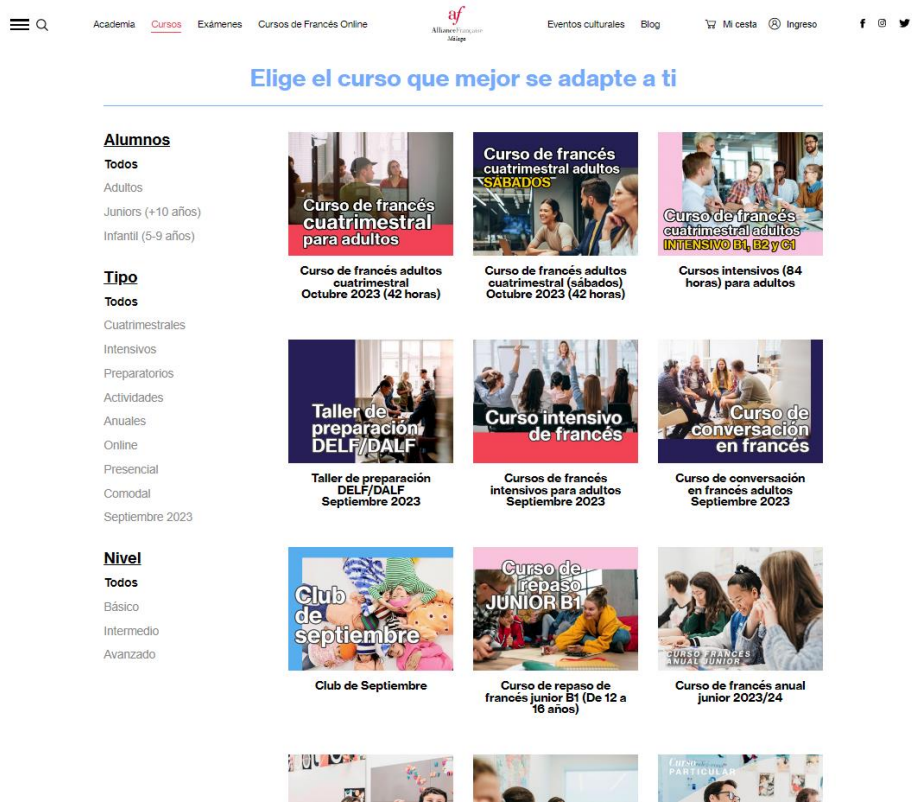

Figura 7:Cursos AF

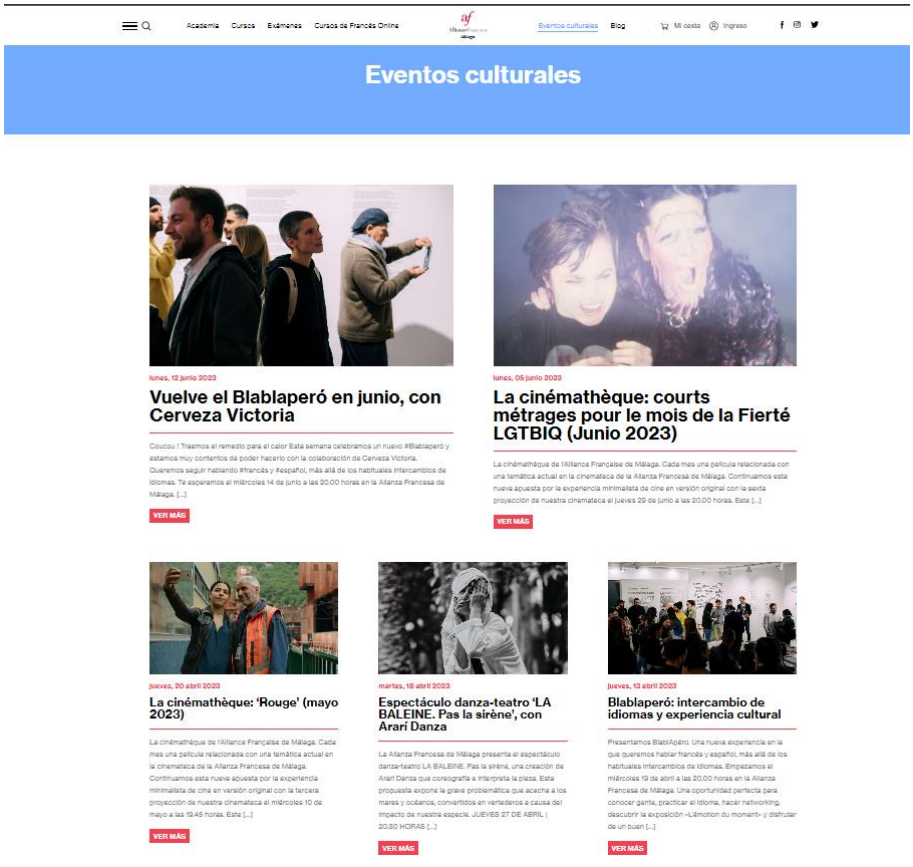

Figura 8: sección de eventos AF

El modelo de calificación es el siguiente:

Valoracion Alianza Francesa (0-100)	Calificación Alianza Francesa
79	237
36	144
86	344
78	156
	881
76	228
45	135
	363
65	130
84	252
	382
18	72
80	320
67	201
63	126
61	122
	841
	2467

Figura 9: Modelo de calificación AF

2.2.3. SS - Sevilla Secreta

Sevilla Secreta ha sido creada por la empresa Fever. Una empresa startup española nacida en 2014. [6]

La web está creada mediante WordPress, utilizando los lenguajes PHP en la parte de backend y JavaScript en la parte de frontend. [7]

La web sigue una arquitectura Modelo-Vista-Controlador.

Sevilla Secreta es una guía online en la cual se oferta y promociona los distintos bares, restaurantes, museos y eventos que ofrece, en este caso, Sevilla [8]

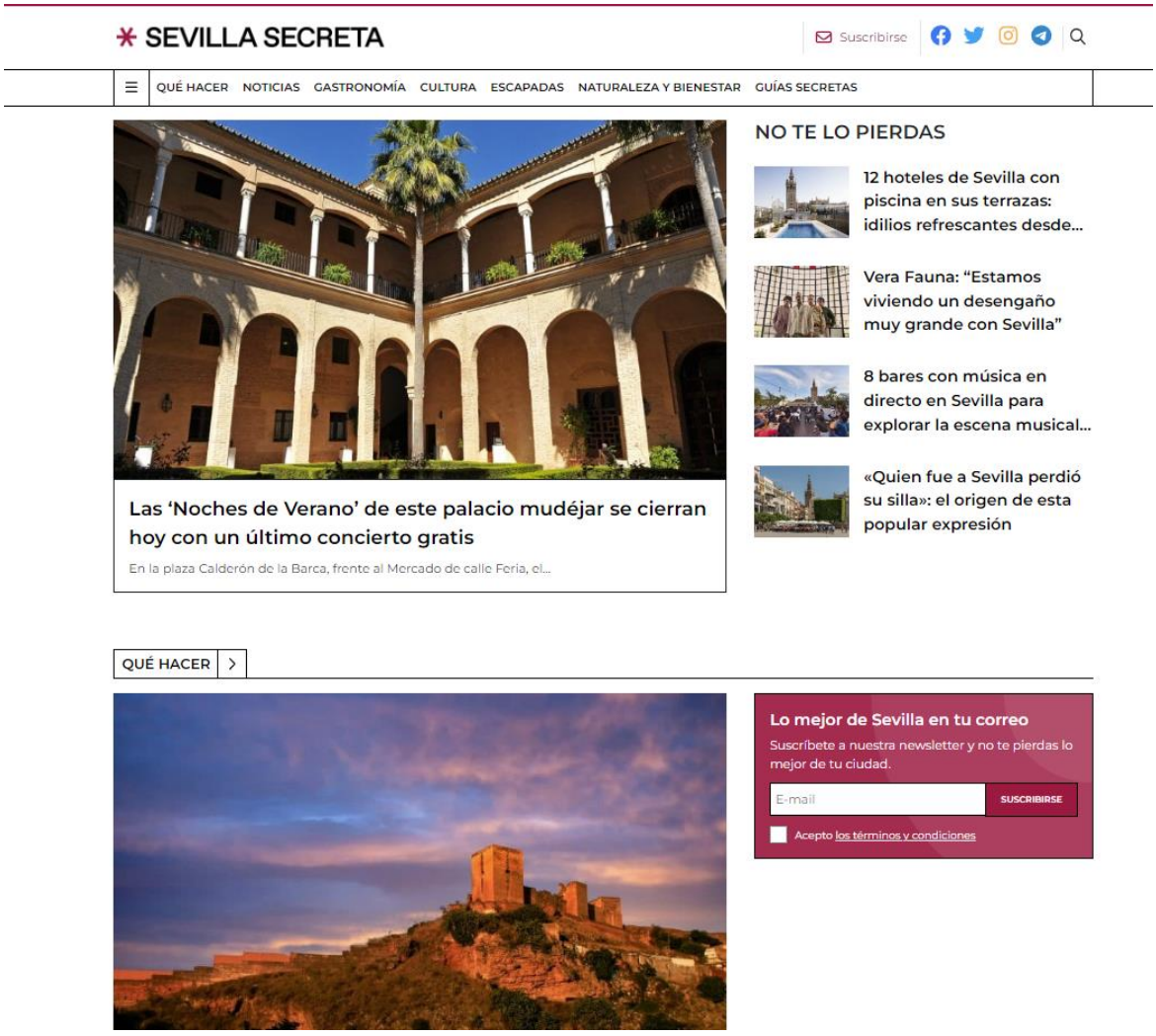

Figura 10: Home SS

Figura 11: Noticias SS

Figura 12: Dentro de una noticia SS

El modelo de calificación es el siguiente:

Valoración SevillaSecreta	Calificación SevillaSecreta
82	246
50	200
15	60
0	0
	506
79	237
68	204
	441
0	0
0	0
	0
20	80
74	296
33	99
45	90
62	124
	689
	1636

Figura 13: Modelo de calificación SS

2.2.4. Conclusión

Se puede observar un mismo patrón dentro de todas las páginas web analizadas. Todas tienen por lo general una puntuación bastante alta. Además, todas siguen el mismo

patrón de diseño vertical, todas siguen la arquitectura modelo-vista-controlador y están creadas con PHP y JavaScript debido a su escalabilidad y fácil creación gracias a WordPress.

3

Análisis

En este apartado se va a explicar el proceso de análisis de la aplicación. Realizando una descripción del sistema a implementar. Explicando los casos de uso y describiendo los diferentes requisitos de la aplicación.

3.1. Descripción general del sistema

El sistema consistirá en una aplicación web que contendrá, principalmente, las siguientes funcionalidades:

- Login
- Usuarios
- Newsletter
- Noticias
- Actividades

La disposición de la web cumplirá con el propósito de facilitar al usuario su uso. Para ello, la web contará con una pantalla de inicio donde aparecerán las ultimas noticias y actividades que se están llevando a cabo, con toda la información al alcance del usuario.

En esta pantalla de inicio se incluyen una serie de pestañas que permiten navegar por la web al clicar en ellas. Una de esas pestañas será la pantalla de login.

El objetivo del sistema de login es permitir al usuario acceder a la plataforma a través de sus credenciales personales, verificando que el usuario está autorizado para acceder a la información y funcionalidades protegidas de la plataforma.

Las credenciales necesarias para poder acceder a la web serán el correo electrónico y la contraseña que previamente habrán sido elegidos por el usuario en cuestión.

Si el usuario no está inscrito en la página, necesitará registrarse de alguna forma. Eso se realizará mediante la opción de Registrarse.

Esta opción da la posibilidad a cualquier usuario que ingrese en la web a poder crearse una cuenta con la cual podrá acceder a las diferentes funcionalidades que se detallarán a continuación.

Para poder registrarse serán necesarios diferentes datos. Estos datos serán:

- Nombre.
- Apellidos.
- Correo electrónico.
- Contraseña.

Una vez el usuario haya rellenado estos campos podrá acceder a la página web con sus credenciales y podrá acceder a las diferentes funcionalidades de esta.

Habrá principalmente tres. Estos serán:

- Usuarios administradores.
- Usuarios Avanzados
- Usuarios básicos.

Los usuarios con el rol de Usuario Administrador serán los que puedan controlar todo tipo de información relacionada con los usuarios y sus relaciones con las actividades. Esto incluye el poder agregar y eliminar usuarios de manera manual, asignar y quitar roles de seguridad, agregar y eliminar subscripciones a actividades y administrar el

sistema de newsletter. Además, podrán realizar todas las acciones de los usuarios con roles inferiores, es decir, las acciones de los roles Usuarios Avanzados y Usuarios Básicos.

Los usuarios con el rol de Usuario Avanzado serán todos aquellos que pertenezcan de una forma u otra al equipo de Malas Lenguas. Estos podrán realizar todo tipo de acciones que no alteren la seguridad de la página web. Con esto me refiero a que podrán publicar noticias y actividades además de poder controlar el registro de las personas que asisten a las actividades creadas por ellos mismos. No podrán realizar las acciones relacionadas con roles y seguridad del nivel de Usuario administrador. Podrán realizar todas las acciones que puedan realizar los usuarios con el rol Usuario Básico.

Los usuarios con el rol de Usuario Básico son los que de manera predeterminada se crean mediante el sistema de registro mencionado anteriormente. Es decir, se le asignará de manera automática a todo usuario que se registre desde la página web. Será el rol más común en la página web. Los usuarios que tengan este rol podrán subscribirse a las diferentes actividades que estén creadas. También podrán gestionar sus propias subscripciones, de manera que podrán ver y cancelar sus subscripciones mediante la pestaña habilitada para ello. Además, podrán contactar con el equipo de Malas lenguas para crear propuestas de actividades para su posterior publicación en la página web.

Si el usuario no está registrado solo tendrá acceso a la página principal, al login, al botón de registrarse en la web, a inscribirse en la newsletter y podrá ver las noticias, el material y las actividades que estén creadas, pero no podrá registrarse a las mismas ni interactuar con ellas de ninguna otra forma.

Uno de los elementos importantes de la aplicación será el sistema de newsletter. Esto consistirá en una caja de texto situado en la página principal en la cual el usuario introducirá su email y aceptará las condiciones. Esto incluirá al usuario en la lista de mailing en la cual cada cierto tiempo el usuario recibirá un email informativo

relacionado con las actividades y noticias que se vayan creando. Cualquier usuario inscrito a la newsletter podrá darse de baja de esta desde la página web. Una vez el usuario se dé de baja en la newsletter se le eliminará de la lista y no recibirá ningún correo relacionado con la misma. Cualquier usuario que se haya dado de baja podrá volver a darse de alta sin ningún tipo de problema. Para inscribirse en esta newsletter, el usuario debe estar registrado en la aplicación. En el caso en el que un usuario no registrado haya introducido su email, recibirá a través de esa dirección de correo electrónico el formulario de registro.

El sistema de noticias consistirá en una serie de posts informativos donde se tratarán los temas correspondientes creados por el equipo de Malas lenguas. Solo los usuarios con los roles de Usuario avanzado o Usuario administrador podrán crear estos posts. Los posts serán textos informativos con imágenes que cualquier usuario podrá leer.

Las noticias serán accesibles mediante la pestaña de Noticias que será visible para todos los usuarios que accedan a la web. Sean usuarios de la web o no.

Los usuarios con los roles anteriormente mencionados tendrán activos el botón de creación de actividad. Este botón permitirá mostrar el editor de texto en el cual redactarán la noticia y podrán subirla para que el resto de gente la vea.

Las noticias estarán vinculados al sistema de newsletter. Siempre que se cree una noticia, los usuarios subscritos a la newsletter recibirán un email de información de que la misma se ha creado cada cierto tiempo.

3.2. Catálogo de Usuarios

La aplicación estará diseñada para administrar tres tipos de usuarios diferentes. Estos usuarios son los siguientes:

3.2.1. Usuario no registrado

Este usuario será el más común de toda la aplicación. Estará formado por toda persona que no tenga una cuenta registrada en la aplicación y acceda a ella sin registrarse.

Este usuario tendrá acceso a las funciones mínimas de la aplicación. Estas son:

- Podrá acceder a las noticias y buscar en ellas.
- Podrá acceder a las actividades, pero no podrá registrarse en ellas. Solo ver la información relacionada a la misma.
- Podrá acceder al archivo de la página para ver las fotos y videos.

El objetivo de este usuario es acceder a la página para ver las posibles actividades en las cuales podría inscribirse si se registra, además de ver las noticias para saber si le interesa el contenido de estas.

3.2.2. Usuario Básico

Este usuario estará formado de manera predeterminada por todo usuario que inscriba en la página mediante el formulario de inscripción creado para ello.
Este usuario tendrá acceso a las funciones más básicas de la aplicación. Estas son:

- Acceso a noticias y búsqueda de ellas.
- Acceso a actividades. Incluyendo la capacidad de registrarse a las mismas
- Registro de la newsletter para recibir noticias relacionadas con la página.
- Acceso al archivo de la página para ver las fotos y los videos.
- Acceso a la pestaña *Mis inscripciones* donde el usuario podrá gestionar las actividades en las que está inscrito. Esto quiere decir que podrá acceder a las mismas y cancelar la inscripción de estas.

El objetivo de este usuario es poder utilizar la aplicación para inscribirse en las diferentes actividades que sean subidas en la página web y poder recibir notificaciones sobre diferentes temas de interés.

3.2.3. Usuario avanzado

Este usuario estará formado por los componentes de Malas lenguas.

Este usuario, además de lo anteriormente mencionado, tendrá acceso a funciones un poco más avanzadas. Estas funciones son:

- Acceso a noticias y búsqueda de ellas
- Capacidad de eliminar y añadir noticias
- Gestionar usuarios
- Gestionar todas las actividades
- Gestión de los asistentes de todas las actividades

3.2.4. Usuario administrador

Estos usuarios están compuestos por los administradores de la aplicación.
Este usuario tendrá completo acceso a todas las funcionalidades de la aplicación. Su principal rol será controlar la seguridad y arreglar los fallos que puedan surgir a futuro en la misma.
Además de todas las funciones anteriormente mencionadas este usuario podrá:

- Cambiar roles a los usuarios.
- Modificación de la plantilla de la newsletter

3.3. Catálogo de requisitos

En este apartado se procederá a enumerar y explicar cada uno de los requisitos de la aplicación. Estos están divididos en 3 tipos diferentes.

- Requisitos funcionales
- Requisitos no funcionales
- Requisitos de información

3.3.1. Requisitos funcionales

Los requisitos funcionales son aquellos que indican las funciones que debe incluir el sistema a desarrollar. La lista está compuesta por 3 elementos.

- Código
- Nombre

- Descripción

- PML-RF-001. Registro de Usuarios: La aplicación debe permitir a los usuarios poder registrarse en este.

- PML-RF-002. Verificación de credenciales: La aplicación debe poder verificar que el usuario ha introducidos sus credenciales de manera correcta.

- PML-RF-003. Reinicio de contraseñas: La aplicación debe permitir a los usuarios cambiar su contraseña en caso de que este la olvide.

- PML-RF-004. CRUD Usuarios: La aplicación debe permitir que los usuarios administradores puedan gestionar los usuarios de esta.

- PML-RF-005. CRUD Actividades: La aplicación debe permitir que los usuarios puedan gestionar actividades.

- PML-RF-006. CRUD Noticias: La aplicación debe permitir que los usuarios puedan gestionar las noticias dentro de la misma.

- PML-RF-007. Automatización de contenido: La aplicación debe permitir que se realicen automáticamente funciones de mensajería.

- PML-RF-008. repositorio de material: La aplicación debe de poder almacenar archivos de imágenes y videos en una pestaña dedicada a ello.

- PML-RF-009. CRUD Inscripciones: la aplicación debe de permitir a los usuarios gestionar las inscripciones a las actividades

- PML-RF-010. gestión de Avisos: el sistema debe Informar al usuario mediante avisos en pantalla de que las acciones se han realizado de manera correcta o incorrecta.

- PML-RF-011. RU Suscripciones a newsletter: El sistema debe permitir Leer y Editar las suscripciones de la Newsletter

- PML-RF-012. Mostrar feed de Twitter: El sistema debe poder enlazar el Twitter de malas lenguas con la web

- PML-RF-013. Eliminación de usuario: El sistema debe de permitir al usuario poder darse de baja de la aplicación en cualquier momento

- PML-RF-014. Subir Videos: El sistema debe de mostrar los videos que se suban

- PML-RF-015. Enviar Newsletter: El sistema deberá enviar periódicamente newsletters indicando lo último en la web

- PML-RF-016. RU Newsletter: El sistema deberá permitir ver y editar la newsletter

- PML-RF-017. Editar Perfil: El sistema deberá permitir a los usuarios modificar sus datos personales dentro de la aplicación

- PML-RF-018. Subir Imágenes: El sistema debe de mostrar las imágenes que se suban

- PML-RF-019. CRD Artículos: El sistema debe de permitir gestionar los artículos que se suben a la web.

3.3.2. Requisitos no funcionales

Los requisitos no funcionales son aquellos que indican aspectos menos técnicos sobre cómo debe de comportarse el sistema. Estos son los siguientes

- PML-RNF-001. Seguridad: El sistema debe garantizar la seguridad de la información que esté almacenada en la plataforma. Se hashearan contraseñas y se evitarán inyecciones de script

- PML-RNF-002. Usabilidad: Cualquier transacción en la aplicación se podrá realizar en menos de 5 clics.

- PML-RNF-003. Escalabilidad: La aplicación debe de ser escalable, es decir, debe permitir aumentar el número de funcionalidades sin afectar a la misma.

- PML-RNF-004. Eficiencia: El tiempo de carga de la aplicación no debe superar los 5 segundos.

- PML-RNF-005. Adaptabilidad: El sistema debe de ser legible tanto en pc como en móvil.

- PML-RNF-006. Accesibilidad: La aplicación debe de ser intuitiva. Se debe entender que hace cada elemento dentro de la misma.

- PML-RNF-007. Disponibilidad: La aplicación debe de encontrarse operativa en todo momento, a excepción de cuando esté en mantenimiento que se informará con anterioridad.

- PML-RNF-008. Tiempo de respuesta: El sistema debe de realizar las acciones de manera rápida, disminuyendo el tiempo de respuesta lo máximo posible.

- PML-RNF-009. Tolerancia a fallos: El sistema debe de ser tolerante a fallos.

- PML-RNF-010. Rendimiento: El sistema debe de funcionar de manera fluida sin parones.

- PML-RNF-011. Personalización: El sistema debe de ser personalizable, es decir, puede cambiar de aspecto visual de manera sencilla a petición del usuario.

- PML-RNF-012. Fiabilidad: El sistema debe de asegurarse de que las funciones se realizan de manera correcta gracias a las pruebas.

3.3.3. Requisitos de información

Los requisitos de información indican la información que debe de gestionar la aplicación. Estos son los siguientes:

- PML-RI-001. Usuario: Nombre, Primer apellido, Segundo apellido, Email, Contraseña, Rol, Acepta newsletter, teléfono, Fecha de nacimiento

- PML-RI-002. Actividad: Nombre, descripción, lugar, fecha de inicio, fecha de fin, fecha de creación, imagen, propietario, enlace, plazas

- PML-RI-003. Rol: Nombre.

- PML-RI-004. Archivo: Imagen, Video

- PML-RI-005. Noticia: Titulo, cuerpo, fecha de creación, imagen, propietario

- PML-RI-006. Artículos: Nombre, descripción, fecha de creación, imagen

3.4. Casos de uso

Los casos de uso se pueden definir como las descripciones de las acciones que lleva a cabo un sistema. Esto, además, incluye los actores o entidades que participan en las acciones. [9]

3.4.1. Diagrama de casos de uso

A continuación, se muestra el diagrama con todos los casos de uso de la aplicación desarrollada.

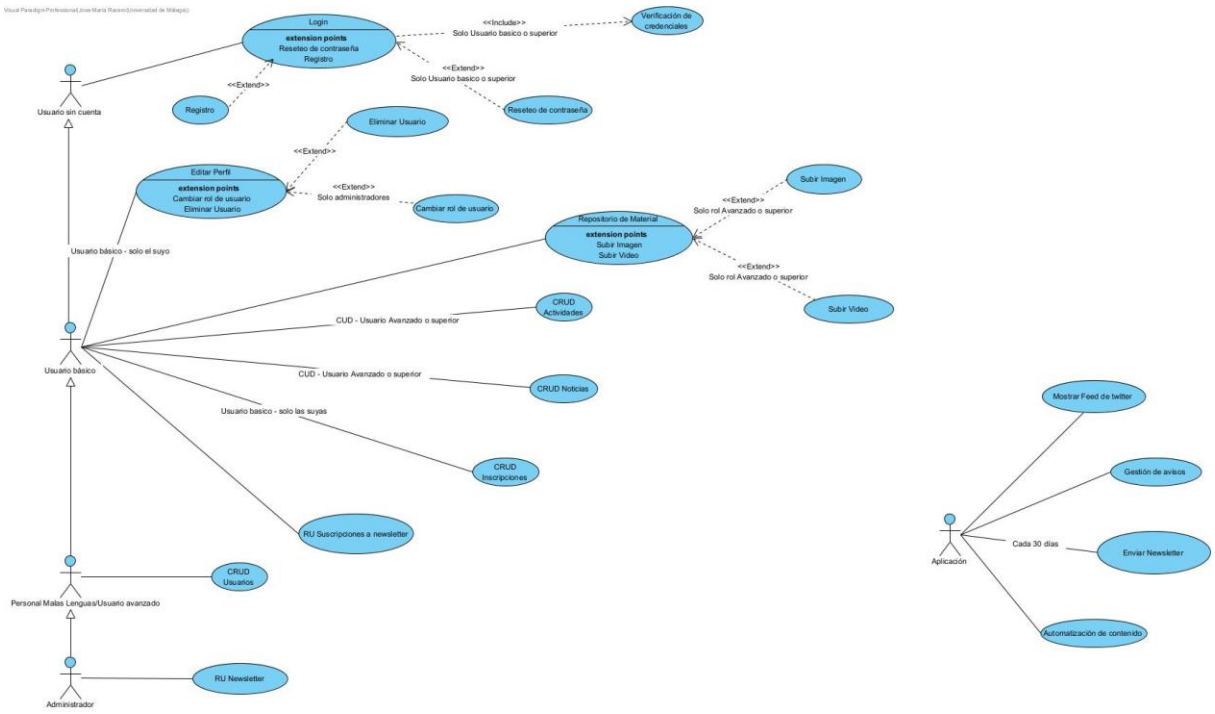

Figura 14: Diagrama de casos de uso

3.4.2. Especificación de casos de uso

A continuación, se detallarán todos los casos de uso de la aplicación.

3.4.2.1. Registro

Título	Registro
Descripción	Formulario de registro para guardar al usuario en la página web.
Pre-condición	No estar inscrito en la web.
Post-condición	Creación del perfil del usuario registrado.
Prioridad	Alta
Autor(es)	Usuario sin cuenta

Control de cambios	Creación del formulario con los siguientes datos: Nombre, primer apellido, segundo apellido, email, contraseña. Añadido fecha de nacimiento y teléfono Añadido campo de titulación
Escenario principal	
1. El usuario pulsa el botón de registro dentro del home de la web 2. El sistema le muestra el formulario de login 3. El usuario pulsa el botón de registrarse. 4. El sistema le muestra el formulario de registro 5. El usuario rellena los campos y pulsa el botón de registrarse 6. El sistema valida los datos y crea el usuario con el rol Usuario básico	
Escenario alternativo	
5.b. El usuario no rellena los datos correctamente. 6.b. El sistema muestra un mensaje de error o no lo deja pulsar el botón dependiendo del tipo de error. 5.c. El usuario pone un email que se encuentra en la web. 6.c El sistema muestra un mensaje indicando que ya existe la cuenta en la base de datos.	
Clases de análisis	
A. Clases de entidad	Usuario, Rol
B. Clases de control	RegistroView.py
C. Clases de interfaz	Registro.html, login.html

Tabla 1: Caso de uso. Registro

3.4.2.2. Login

Título	Login
Descripción	Formulario de inicio de sesión de la web.
Pre-condición	1. Haberse registrado previamente en la web
Post-condición	1. Se comprobarán las credenciales introducidas. 2. Se redireccionará al usuario a la página de inicio. 3. Se habilitarán las diferentes funcionalidades asociadas al rol del usuario.
Prioridad	Alta
Autor(es)	Usuario sin registrar
Control de cambios	
Escenario principal	
1. El usuario introducirá su email. 2. El usuario introducirá su contraseña. 3. El usuario le dará al botón de Login. 4. El sistema comprobará que el email existe. 5. El sistema comprobará que la contraseña coincide con la relacionada con el email 6. El sistema desviará a la página principal de la web con la sesión iniciada.	
Escenario alternativo	

1.b. el usuario introduce un email que no existe 4.b. El sistema comprobará que el email existe. 5.b. El sistema comprobará que la contraseña coincide con la relacionada con el email 6.b. El sistema muestra un mensaje de error indicando que el email o la contraseña son incorrectos 2.c. el usuario introduce una contraseña erronea. 4.c. El sistema comprobará que el email existe. 5.c. El sistema comprobará que la contraseña coincide con la relacionada con el email 6.c. El sistema muestra un mensaje de error indicando que el email o la contraseña son incorrectos	
Clases de análisis	
A. Clases de entidad	Usuario
B. Clases de control	LoginView.py
C. Clases de interfaz	Login.html

Tabla 2: Caso de uso. Login

3.4.2.3. Editar Perfil

Título	Editar Perfil
Descripción	Permite modificar los datos del usuario
Pre-condición	1. Estar registrado en la web 2. Haber iniciado sesión
Post-condición	1. Se verán los cambios realizados en el perfil del usuario.

Prioridad	media
Autor(es)	Usuario Normal
Control de cambios	1. Modificación de los campos: Nombre, Primer apellido, Segundo apellido, Email, Contraseña. 2. Añadidos campos: Fecha de nacimiento y Teléfono 3. Añadido campo Titulación 4. Añadido campo imagen
Escenario principal	
1. El usuario Entra en su perfil 2. El Sistema muestra la ficha con los datos del usuario 3. El usuario entra en editar perfil 4. El Sistema muestra el formulario de edición 5. El usuario ingresa los datos que desea modificar 6. El Sistema valida los datos y realiza las modificaciones 7. El Sistema redirecciona al perfil	
Escenario alternativo	
5.b. El usuario introduce algún dato erroneo 6.b. El Sistema muestra un mensaje de error por pantalla.	
Clases de análisis	
A. Clases de entidad	Usuario
B. Clases de control	AdministracionView.py

C. Clases de interfaz	Perfil.html

Tabla 3: Caso de uso. Editar perfil

3.4.2.4. Repositorio de material

Título	Repositorio de Material
Descripción	Clase de almacenaje de fotos y videos para mostrar en la sección Archivo.
Pre-condición	
Post-condición	
Prioridad	Baja
Autor(es)	Usuario
Control de cambios	
Escenario principal	
1. El usuario entra en la sección de Archivo. 2. El sistema le muestra por defecto la vista de las imagenes. 3. El usuario pulsa en la pestaña de videos. 4. El Sistema le muestra la vista con los videos.	
Escenario alternativo	
Clases de análisis	
A. Clases de entidad	Usuario, Archivo

B. Clases de control	ArchivoView.py
C. Clases de interfaz	Archivo.html

Tabla 4: Caso de uso. Repositorio de material

3.4.2.5. CRUD Actividades

Título	CRUD Actividades
Descripción	Creación, lectura, edición y borrado de las actividades.
Pre-condición	1. Haber iniciado sesión previamente 2. Tener un rol asignado
Post-condición	1. Los cambios de la actividad serán visibles para todos los usuarios de la web.
Prioridad	Alta
Autor(es)	Usuario básico, Usuario Avanzado, Administrador
Control de cambios	
Escenario principal	
Creación 1. El usuario con rol Avanzado o administrador entra en la sección de actividades 2. El Sistema muestra la sección con todas las actividades 3. El usuario pulsa el botón de Crear Actividad 4. El Sistema muestra el formulario de creación de actividad. 5. El usuario rellena los datos correspondientes 6. El usuario pulsa el botón de subir actividad	

7. El Sistema procesa la petición y muestra un mensaje indicando que la creación ha sido un éxito

Lectura

1. El usuario entra en la sección de actividades
2. El Sistema muestra la sección con todas las actividades
3. El usuario puede buscar y entrar en cualquier actividad previamente creada.

Edición

1. El usuario con rol Avanzado o administrador entra en la sección de actividades
2. El Sistema muestra la sección con todas las actividades
3. El usuario entra en la actividad que desea editar
4. El Sistema muestra la actividad
5. El usuario pulsa el botón de editar actividad
6. El sistema muestra el formulario de edición con los datos precargados
7. El usuario modifica los datos deseados
8. El Sistema muestra un mensaje indicando que los cambios se han realizado
9. El Sistema redirige a la noticia con los datos visibles

Borrado

1. El usuario con rol Avanzado o administrador entra en la sección de actividades
2. El Sistema muestra la sección con todas las actividades
3. El usuario entra en la actividad que desea eliminar
4. El sistema muestra la actividad
5. El usuario pulsa el botón de eliminar actividad
6. El Sistema elimina la actividad de la base de datos

7. El Sistema muestra un mensaje indicando que la operación se ha realizado con éxito 8. El Sistema redirige a la página principal de actividades	
Escenario alternativo	
Creación 5b. El usuario no rellena todos los campos necesarios 6b. El sistema no permite crear la actividad	
Clases de análisis	
A. Clases de entidad	Usuario, Actividad
B. Clases de control	ActividadesView.py
C. Clases de interfaz	abrirActividad.html, actividades.html, editarActividad.html, crearActividad.html

Tabla 5: Caso de uso. CRUD actividad

3.4.2.6. CRUD Noticias

Título	CRUD Noticias
Descripción	Creación, lectura, edición y borrado de las noticias
Pre-condición	1. Haber iniciado sesión previamente 2. Tener un rol asignado
Post-condición	1. Los cambios de la noticia serán visibles para todos los usuarios de la web.
Prioridad	Alta

Autor(es)	Usuario básico, Usuario Avanzado, administrador
Control de cambios	¿Modificado desde la última entrega? o Descripción de los cambios
Escenario principal	

Creación

1. El usuario con rol Avanzado o administrador entra en la sección de noticias
2. El Sistema muestra la sección con todas las noticias
3. El usuario pulsa el botón de Crear Noticia
4. El Sistema muestra el formulario de creación de noticias.
5. El usuario rellena los datos correspondientes
6. El usuario pulsa el botón de subir noticia
7. El Sistema procesa la petición y muestra un mensaje indicando que la creación ha sido un éxito

Lectura

1. El usuario entra en la sección de noticias
2. El Sistema muestra la sección con todas las noticias
3. El usuario puede buscar y entrar en cualquier noticia previamente creada.

Edición

1. El usuario con rol Avanzado o administrador entra en la sección de noticias
2. El Sistema muestra la sección con todas las noticias
3. El usuario entra en la noticia que desea editar
4. El Sistema muestra la noticia
5. El usuario pulsa el botón de editar noticia

6. El sistema muestra el formulario de edición con los datos precargados
7. El usuario modifica los datos deseados
8. El Sistema muestra un mensaje indicando que los cambios se han realizado
9. El Sistema redirige a la noticia con los datos visibles

Borrado

1. El usuario con rol Avanzado o administrador entra en la sección de noticias
2. El Sistema muestra la sección con todas las noticias
3. El usuario entra en la noticia que desea eliminar
4. El sistema muestra la noticia
5. El usuario pulsa el botón de eliminar noticia
6. El Sistema elimina la noticia de la base de datos
7. El Sistema muestra un mensaje indicando que la operación se ha realizado con éxito
8. El Sistema redirige a la página principal de noticias

Escenario alternativo	
Creación 5b. El usuario no rellena todos los campos necesarios 6b. El sistema no permite crear la actividad	
Clases de análisis	
A. Clases de entidad	Usuario, Noticia
B. Clases de control	NoticiasView.py

C. Clases de interfaz	Noticias.html, editarNoticia.html, crearNoticia.html, abrirNoticia.html

Tabla 6: Caso de uso. CRUD noticia

3.4.2.7. CRUD Inscripciones

Título	CRUD Inscripciones
Descripción	Creación, lectura, edición y borrado de inscripciones
Pre-condición	1. El usuario debe de haber iniciado sesión 2. Al menos debe existir una actividad en la web 3. La actividad debe de tener una fecha de inicio superior al día actual.
Post-condición	1. Se creará una relación entre el usuario inscrito y la actividad en cuestión 2. El botón de inscribirse cambiará a Cancelar inscripción
Prioridad	Alta
Autor(es)	Usuario básico, usuario Avanzado, administrador
Control de cambios	1. Las inscripciones solo podrán ser creadas por el usuario en cuestión.
Escenario principal	
Creación 1. El usuario iniciará sesión. 2. El usuario pulsará la pestaña de actividades 3. El sistema le mostrará la pestaña con todas las actividades	

4. El usuario elegirá la actividad en la que quiera inscribirse
5. El sistema le mostrará la actividad en cuestión
6. El usuario pulsará el botón de inscribirse
7. El sistema relacionará al usuario con la actividad con la tabla inscripciones.
8. El sistema mostrará un mensaje indicando que la operación se ha completado con éxito

Lectura

1. El usuario iniciará sesión
2. El usuario pulsará en la pestaña de Perfil
3. El sistema le mostrará su perfil
4. El usuario pulsará en la pestaña de inscripciones
5. El sistema mostrará una tabla con las actividades en las que está inscrito

Modificación

1. El usuario iniciará sesión con una cuenta con rol de usuario avanzado o administrador
2. El usuario pulsará la pestaña de actividades
3. El sistema le mostrará la pestaña con todas las actividades
4. El usuario elegirá la actividad que desee.
5. El sistema le mostrará el contenido de dicha actividad
6. El usuario pulsará el botón de ver inscripciones
7. El sistema le mostrará una tabla con todas las inscripciones existentes con dicha actividad
8. El usuario podrá eliminar cualquier inscripción de dicha actividad pulsando el botón eliminar inscripción

9. El sistema mostrará un mensaje indicando que se ha llevado a cabo la operación con éxito

Borrado

1. El usuario iniciará sesión
2. El usuario pulsará en la pestaña de Perfil
3. El sistema le mostrará su perfil
4. El usuario pulsará en la pestaña de inscripciones
5. El sistema mostrará una tabla con las actividades en las que está inscrito
6. El usuario podrá cancelar la inscripción de cualquiera de esas actividades pulsando el botón de cancelar inscripción
7. El sistema mostrará un mensaje indicando que la operación se ha llevado a cabo con éxito

Escenario alternativo

Creación 6b. El usuario no verá el botón de inscripción porque la fecha de inicio de la actividad es anterior a la actual Modificación 8b. el usuario no verá el botó de eliminar inscripción porque la fecha de inicio de la actividad es anterior a la actual Borrado 6b. El usuario no verá el botón debido a que la fecha de inicio de la actividad es anterior a la actual	
Clases de análisis	
A. Clases de entidad	Usuario, Inscripción, Actividad
B. Clases de control	ActividadesView.py
C. Clases de interfaz	Perfil.html, inscripcionesActividad.html, abrirNoticia.html

Tabla 7: Caso de uso. CRUD inscripciones

3.4.2.8. RU Suscripciones a Newsletter

Título	RU Suscripciones a newsletter
Descripción	Leer y editar Suscripciones a newsletter
Pre-condición	1. Debe de existir un usuario en la base de datos
Post-condición	1. Se mostrarán los cambios en el perfil del usuario

Prioridad	Alta
Autor(es)	Usuario básico, Usuario avanzado y administrador
Control de cambios	
Escenario principal	

Lectura Usuario básico

1. El usuario iniciará sesión
2. El usuario entrará en su perfil
3. El sistema le mostrará su perfil
4. El usuario ahora puede ver si acepta o no newsletter

Lectura Usuario avanzado/administrador

1. El usuario inicia sesión
2. El usuario entra en la pestaña de gestionar usuarios
3. El sistema le muestra una tabla con todos los usuarios
4. El usuario puede ver en esa tabla quienes tienen aceptado la newsletter y quien no

Editar Usuario básico

1. El usuario iniciará sesión
2. El usuario entrará en su perfil
3. El sistema le mostrará su perfil
4. El usuario pulsará en modificar perfil/configuración
5. El sistema le mostrará la ventana de configuración
6. El usuario se va a la pestaña de general donde verá un campo de newsletter.
7. El usuario elige la opción y le da a guardar.

8. El sistema modifica dicho campo dentro del usuario y muestra un mensaje por pantalla

Editar Usuario avanzado/administrador

1. El usuario inicia sesión
2. El usuario entra en la pestaña de gestionar usuarios
3. El sistema le muestra una tabla con todos los usuarios
4. El usuario busca a quien desea cambiarle el campo de aceptar newsletter
5. El usuario pulsa el botón del perfil del usuario en cuestión
6. El sistema le muestra el perfil de dicho usuario
7. El usuario pulsará en modificar perfil/configuración
8. El sistema le mostrará la ventana de configuración
9. El usuario se va a la pestaña de general donde verá un campo de newsletter.
10. El usuario elige la opción y le da a guardar.
11. El sistema modifica dicho campo dentro del usuario y muestra un mensaje por pantalla

Escenario alternativo	
Clases de análisis	
A. Clases de entidad	Usuario
B. Clases de control	AdministraciónView.py

C. Clases de interfaz	gestionUsuarios.html, administración.html, perfil.html

Tabla 8: Caso de uso. RU inscripciones a newsletter

3.4.2.9. CRUD Usuarios

Título	Nombre del caso de uso
Descripción	Creación, lectura, edición y borrado de usuarios.
Pre-condición	Creación 1. El email del usuario en cuestión no debe de existir previamente en la web Edición, lectura y borrado 1. El usuario debe de haber iniciado sesión
Post-condición	Creación 1. El usuario tendrá una cuenta personal vinculada a dicho email. Edición 1. El usuario tendrá los cambios aplicados Borrado 1. El usuario no tendrá acceso a la web
Prioridad	Alta
Autor(es)	Usuario
Control de cambios	
Escenario principal	

Creación

1. El usuario pulsa en la pestaña de gestionar usuarios
2. El sistema le muestra una tabla con todos los usuarios de la aplicación
3. El usuario pulsa el botón de añadir usuario
4. El sistema muestra un formulario de registro
5. El usuario rellena los datos
6. El Sistema comprueba los datos
7. El sistema muestra un mensaje indicando que la operación ha funcionado correctamente

Lectura

1. El usuario pulsa en la pestaña de gestionar usuarios
2. El sistema le muestra una tabla con todos los usuarios de la aplicación
3. El usuario pulsa el botón de ver perfil del usuario que desee

Edición

1. El usuario pulsa en la pestaña de gestionar usuarios
2. El sistema le muestra una tabla con todos los usuarios de la aplicación
3. El usuario pulsa el botón de ver perfil
4. El usuario dentro del perfil pulsa el botón editar perfil
5. El sistema le muestra la pestaña de edición
6. El usuario rellena los datos correspondientes y pulsa el botón de guardar
7. El sistema muestra un mensaje indicando que todo ha salido bien.

Borrado

1. El usuario pulsa en la pestaña de gestionar usuarios

2. El sistema le muestra una tabla con todos los usuarios de la aplicación
3. El usuario pulsa el botón de ver perfil
4. El usuario dentro del perfil pulsa el botón editar perfil/configuración
5. El sistema le muestra la pestaña de edición
6. El usuario pulsa la pestaña de general
7. El sistema le muestra la pestaña general
8. El usuario pulsa el botón de desactivar cuenta
9. El sistema borra el usuario de la base de datos y muestra un mensaje indicando que todo ha salido bien.

Escenario alternativo

Creación

5b. El usuario no rellena correctamente algún campo

6b. El sistema comprueba los campos e indica que hay campos mal rellenados.

Edición

6b. Si el usuario no rellena correctamente el campo a editar, no podrá guardar.

6c. Si el usuario le da al botón de guardar sin realizar ningún cambio, no hará nada el botón.

Borrado

9b. Si el usuario eliminado es el mismo que el que ha iniciado sesión también borra las cookies y cierra la sesión.

Clases de análisis

A. Clases de entidad	Usuario, Rol
B. Clases de control	RegistroView.py, AdministracionView.py
C. Clases de interfaz	Perfil.html, RegistroGestion.html, administración.html, gestionUsuarios.html

Tabla 9: Caso de uso. CRUD usuario

3.4.2.10. RU Newsletter

Título	RU Newsletter
Descripción	Leer y editar la plantilla de la newsletter
Pre-condición	1. Haber iniciado sesión con un usuario con el rol administrador
Post-condición	1. La plantilla de la newsletter modificada
Prioridad	media
Autor(es)	Administrador
Control de cambios	1. Añadido botón para visualizar la plantilla actual 2. Añadido enlace para descargar la plantilla base como copia de seguridad en caso de subir una plantilla errónea
Escenario principal	

<table>
<tr><td colspan="2">Lectura
1. El usuario inicia sesión con una cuenta con el rol de administrador
2. El usuario abre la pestaña de Cambiar newsleter
3. El sistema le muestra la vista correspondiente
4. El usuario pulsa el botón de mostrar newsletter
5. El sistema le redirecciona al html de la plantilla para que la visualice

Edición
1. El usuario inicia sesión con una cuenta con el rol de administrador
2. El usuario abre la pestaña de Cambiar newsleter
3. El sistema le muestra la vista correspondiente
4. El usuario pulsa en el botón de seleccionar archivo
5. El usuario introduce el archivo del tipo html que será la plantilla en cuestión
6. El usuario pulsa el botón modificar newsletter
7. El sistema valida dicha acción y muestra un mensaje indicando que la operación ha funcionado</td></tr>
<tr><td colspan="2">Escenario alternativo</td></tr>
<tr><td colspan="2">7b. El usuario pulsa el botón sin seleccionar un archivo
8b. El sistema desactiva el botón</td></tr>
<tr><td colspan="2">Clases de análisis</td></tr>
<tr><td>A. Clases de entidad</td><td>Usuario, Rol, NewsletterTemplate</td></tr>
<tr><td>B. Clases de control</td><td>AdministraciónView.py</td></tr>
</table>

C. Clases de interfaz	cambiarNewsletter.html

Tabla 10: Caso de uso. RU newsletter

3.4.2.11. Mostrar feed de Twitter

Título	Mostrar Feed de Twitter
Descripción	Mostrar Feed de Twitter
Pre-condición	1. El usuario que acceda a la web debe de tener la cuenta de Twitter abierta
Post-condición	1. La feed de malas lenguas estará visible en la página principal
Prioridad	Baja
Autor(es)	Aplicación
Control de cambios	
Escenario principal	
1. El usuario inicia sesión en Twitter 2. El usuario Entra en la web 3. La feed del twitter de Malas Lenguas está visible en la página de inicio	
Escenario alternativo	
2b. El usuario entra en la web sin iniciar sesión en Twitter 3b. La feed de twitter le pedirá iniciar sesión en Twitter	
Clases de análisis	

A. Clases de entidad	Usuario
B. Clases de control	HomeView.py
C. Clases de interfaz	Home.html

Tabla 11: Caso de uso. Mostrar feed de Twitter

3.4.2.12. Enviar Newsletter

Título	Enviar newsletter
Descripción	El sistema enviará periódicamente newsletters indicando lo último en la web
Pre-condición	1. Deben existir usuarios en la aplicación con el campo de newsletter a True
Post-condición	1. Se les enviará un email con la plantilla de la newsletter
Prioridad	Alta
Autor(es)	Aplicación
Control de cambios	
Escenario principal	

1. El sistema comprueba el tiempo desde la última newsletter 2. El sistema a los 30 días llamará a la función que envía las newsletters 3. El sistema recoge a todos los emails de usuarios que tengan el campo newsletter a True 4. El sistema envía los emails con los datos del usuario y las últimas novedades en la web.	
Escenario alternativo	
Clases de análisis	
A. Clases de entidad	Usuario, Actividades, Noticias, Artículos
B. Clases de control	AdministraciónView.py
C. Clases de interfaz	Newsletter_template.html

Tabla 12: Caso de uso. Enviar newsletter

3.4.2.13. Gestión de avisos

Título	Gestión de avisos
Descripción	El sistema mostrará por pantalla mensajes cada vez que una acción se realice.
Pre-condición	1. Debe realizarse cualquier acción que requiera un movimiento de datos
Post-condición	1. Se mostrará por pantalla un mensaje en la parte superior

Prioridad	Media
Autor(es)	Aplicación
Control de cambios	
Escenario principal	
1. El usuario realiza una acción dentro de la aplicación (por ejemplo, crear una noticia) 2. El sistema realiza la acción 3. Al final redirecciona junto con un mensaje predefinido.	
Escenario alternativo	
Clases de análisis	
A. Clases de entidad	Todas
B. Clases de control	Todas
C. Clases de interfaz	Base.html

Tabla 13: Caso de uso. Gestión de avisos

3.4.2.14. Automatización de contenido

Título	Automatización de contenido
Descripción	El sistema permitirá realizar tareas de manera automática. Por ejemplo, Borrar registros, enviar mensajes, etc…
Pre-condición	1. Debe de existir un disparador para la acción 2. Debe de especificarse que tarea se debe realizar

Post-condición	1. La tarea se realiza
Prioridad	Alta
Autor(es)	Aplicación
Control de cambios	
Escenario principal	
1. El sistema comprueba diariamente el disparador 2. Cuando el disparador se cumpla, realiza la acción	
Escenario alternativo	
Clases de análisis	
A. Clases de entidad	Todos
B. Clases de control	Tasks.py
C. Clases de interfaz	

Tabla 14; Caso de uso. Automatización de contenido

3.4.2.15. Validación de credenciales

Título	Verificación de credenciales
Descripción	El sistema comprobará que el email y la contraseña introducidas coincidan

Pre-condición	1. El campo de email debe de estar relleno 2. El campo de contraseña debe de estar relleno
Post-condición	1. Permitirá seguir la acción del login
Prioridad	Alta
Autor(es)	Usuario sin cuenta
Control de cambios	
Escenario principal	
1. El sistema comprobará que el email existe en la base de datos 2. El sistema comprobará que la contraseña es la correcta para el usuario con el email anterior 3. devolverá True para que el login prosiga	
Escenario alternativo	
3b. en caso de que uno de los dos puntos anteriores sea falso, el sistema devolverá False para que el login muestre un mensaje de error.	
Clases de análisis	
A. Clases de entidad	Usuario
B. Clases de control	LoginView.py
C. Clases de interfaz	Login.html

Tabla 15: Caso de uso. Validación de credenciales

3.4.2.16. Reinicio de contraseña

Título	Reinicio de contraseña
Descripción	Botón para permitir a un usuario que ya tenga una cuenta creada reinicie su contraseña recibiendo en su email una nueva contraseña
Pre-condición	1. El usuario debe de tener una cuenta en la aplicación 2. El usuario debe de poder entrar en dicho email
Post-condición	1. El email con la nueva contraseña estará en el buzón del email.
Prioridad	Media
Autor(es)	Usuario sin cuenta
Control de cambios	
Escenario principal	
1. El usuario pulsa el botón de Has olvidado la contraseña 2. El sistema le muestra un formulario con un campo de email 3. El usuario introduce el email 4. El sistema busca el usuario que tenga el email introducido 5. El sistema genera una contraseña aleatoria y envía el email	
Escenario alternativo	

3b. El email introducido no existe en la base de datos 4b. el sistema busca el usuario que tenga el email introducido 5b. el sistema muestra un mensaje indicando de que no existe un usuario con dicho email	
Clases de análisis	
A. Clases de entidad	Usuario
B. Clases de control	LoginView.py
C. Clases de interfaz	Login.html, reestablecerContraseña.html

Tabla 16: Caso de uso. Reinicio de contraseña

3.4.2.17. Subir imagen

Título	Subir Imagen
Descripción	Sube imágenes al repositorio de material
Pre-condición	1. El usuario debe de haber iniciado sesión 2. El usuario debe de tener el rol de usuario avanzado o administrador
Post-condición	1. La imagen será visible
Prioridad	Baja
Autor(es)	Usuario
Control de cambios	
Escenario principal	

1. El usuario con rol Avanzado o administrador entra en la sección de Archivo 2. El Sistema e muestra por defecto la vista de las imágenes 3. El usuario ve la sección para subir imágenes 4. El usuario inserta una imagen y le da a subir 5. El Sistema guarda la imagen en la base de datos y redirecciona a la página de Archivo	
Escenario alternativo	
Clases de análisis	
A. Clases de entidad	Usuario, Archivo
B. Clases de control	ArchivoView.py
C. Clases de interfaz	Archivo.html

Tabla 17: Caso de uso. Subir imagen

3.4.2.18. Subir videos

Título	Subir videos
Descripción	Sube videos al repositorio de material
Pre-condición	1. El usuario debe de haber iniciado sesión 2. El usuario debe de tener el rol de usuario avanzado o administrador
Post-condición	1. El video estará disponible

Prioridad	Baja
Autor(es)	Usuario
Control de cambios	
Escenario principal	
1. El usuario con rol Avanzado o administrador entra en la sección de Archivo 2. El Sistema e muestra por defecto la vista de las imágenes 3. El usuario pulsa en la pestaña de videos 4. El usuario ve la sección para subir videos 5. El usuario inserta un video y le da a subir 6. El Sistema guarda el video en la base de datos y redirecciona a la página de Archivo	
Escenario alternativo	
Clases de análisis	
A. Clases de entidad	Usuario, Archivo
B. Clases de control	ArchivoView.py
C. Clases de interfaz	Archivo.html

Tabla 18: Caso de uso. Subir video

3.4.2.19. Eliminar Usuario

Título	Eliminar usuario

Descripción	Opción para que un usuario pueda eliminar su cuenta de la web
Pre-condición	1. Haber iniciado sesión
Post-condición	1. El usuario quedará borrado de la base de datos
Prioridad	Baja
Autor(es)	Usuario
Control de cambios	
Escenario principal	
1. El usuario inicia sesión 2. El usuario pulsa en la pestaña de perfil 3. El sistema le muestra su perfil 4. El usuario pulsa el botón de editar perfil/configuración 5. El sistema le muestra la configuración 6. El usuario pulsa la pestaña general 7. El usuario pulsa el botón de eliminar cuenta 8. El sistema borra las cookies y cierra la sesión del usuario 9. El sistema muestra un mensaje indicando que la cuenta se ha borrado	
Escenario alternativo	
Clases de análisis	
A. Clases de entidad	Usuario

B. Clases de control	AdministracionView.py
C. Clases de interfaz	home.html, perfil.html, administración.html

Tabla 19: Caso de uso. Eliminar usuario

3.4.2.20. Cambiar rol del usuario

Título	Cambiar rol del usuario
Descripción	Permite cambiar el rol del usuario
Pre-condición	1. Haber iniciado sesión 2. Tener el rol de administrador
Post-condición	1. El usuario verá su rol cambiado
Prioridad	Alta
Autor(es)	Administrador
Control de cambios	
Escenario principal	

1. El usuario inicia sesión 2. El usuario pulsa en la pestaña de gestionar usuarios 3. El sistema le muestra la tabla con todos los usuarios 4. El usuario pulsa el botón de perfil 5. El sistema le muestra el perfil 6. El usuario pulsa el botón de editar perfil/configuración 7. El sistema le muestra la configuración 8. El usuario pulsa la pestaña editar usuario 9. El usuario cambia el rol y pulsa el botón de guardar cambios 10. El sistema muestra un mensaje indicando el rol se ha cambiado	
Escenario alternativo	
2b. El usuario no tiene rol administrador 9b. El usuario no tendrá visible dicho campo	
Clases de análisis	
A. Clases de entidad	Usuario, rol
B. Clases de control	AdministracionView.py
C. Clases de interfaz	home.html, gestionarUsuario.html, perfil.html, administración.html

Tabla 20: Caso de uso. Cambiar rol de usuario

3.5. Maquetado de la interfaz de usuario

A continuación, se van a mostrar una serie de imágenes de las maquetas de la aplicación realizadas mediante Balsamiq. Estas maquetas posteriormente se le enseñaron al cliente para validar la interfaz gráfica del sistema.

Figura 15: Maqueta Home

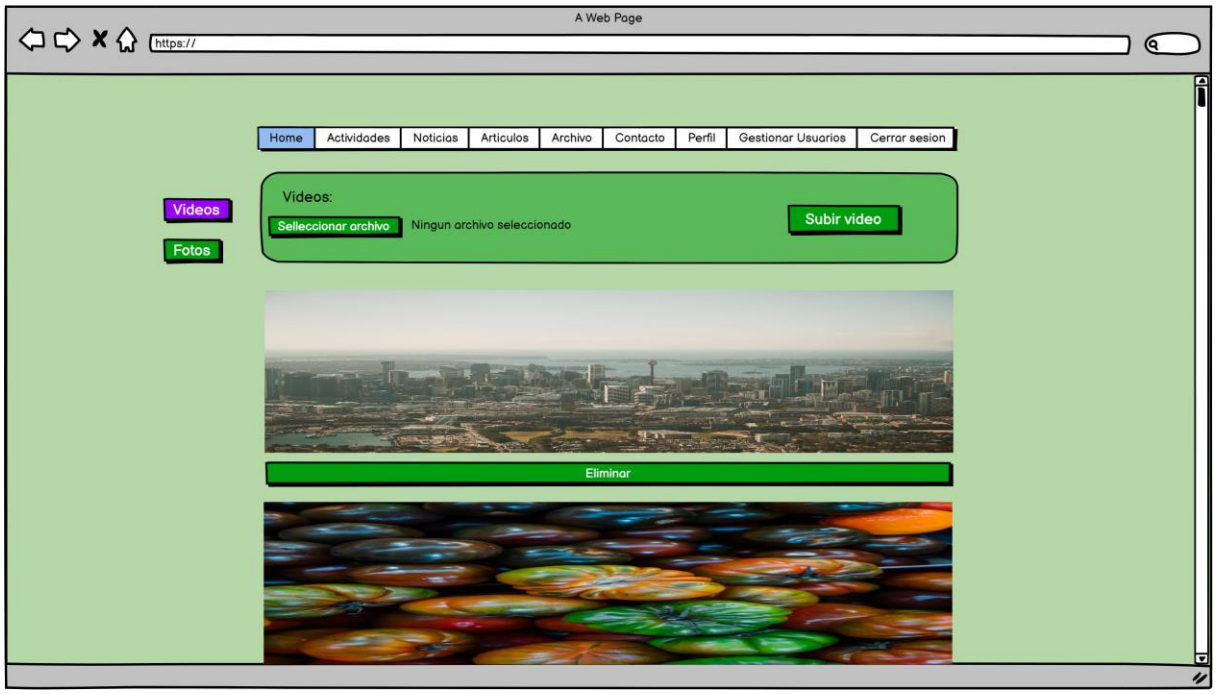

Figura 16: Maqueta Archivo (1)

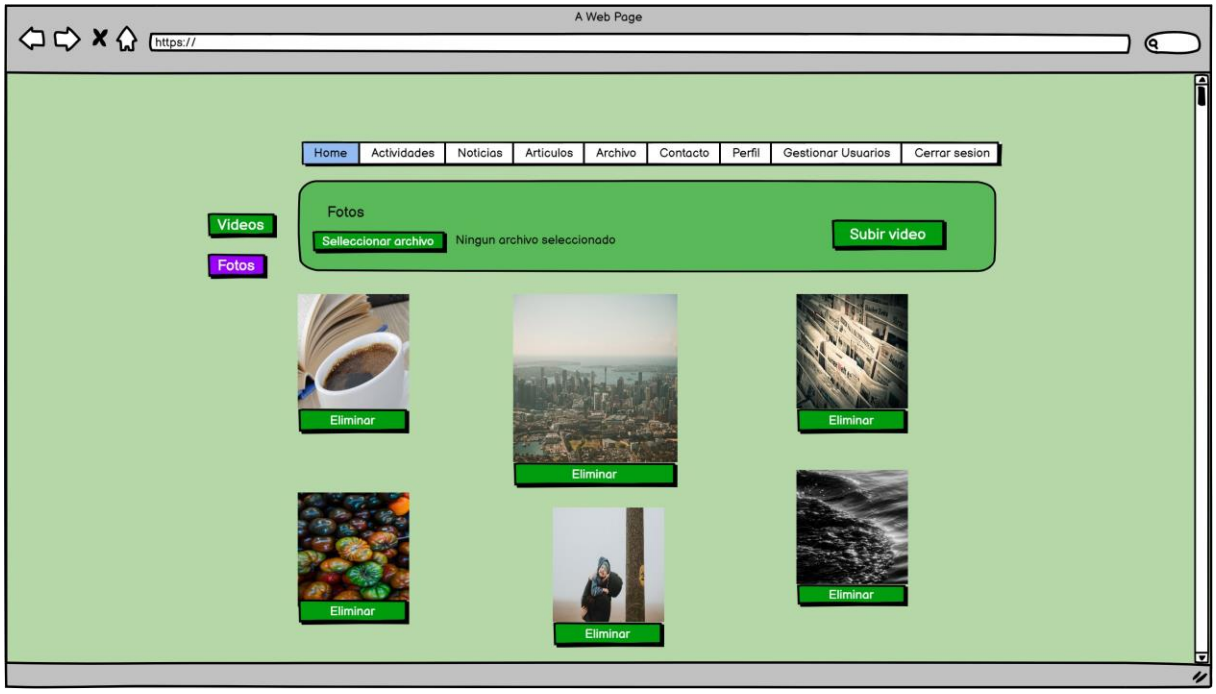

Figura 17: Maqueta Archivo (2)

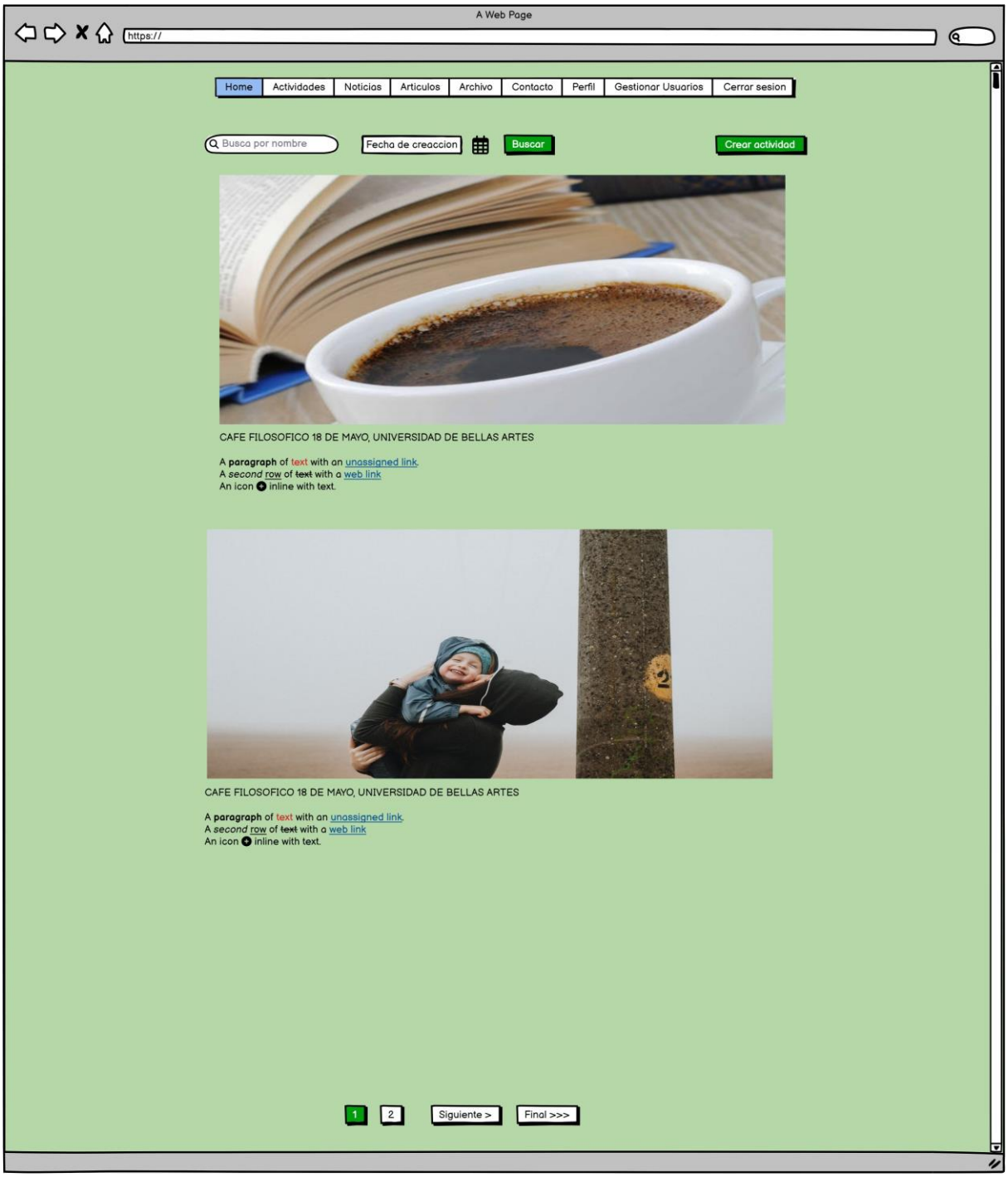

Figura 18: Maqueta actividades

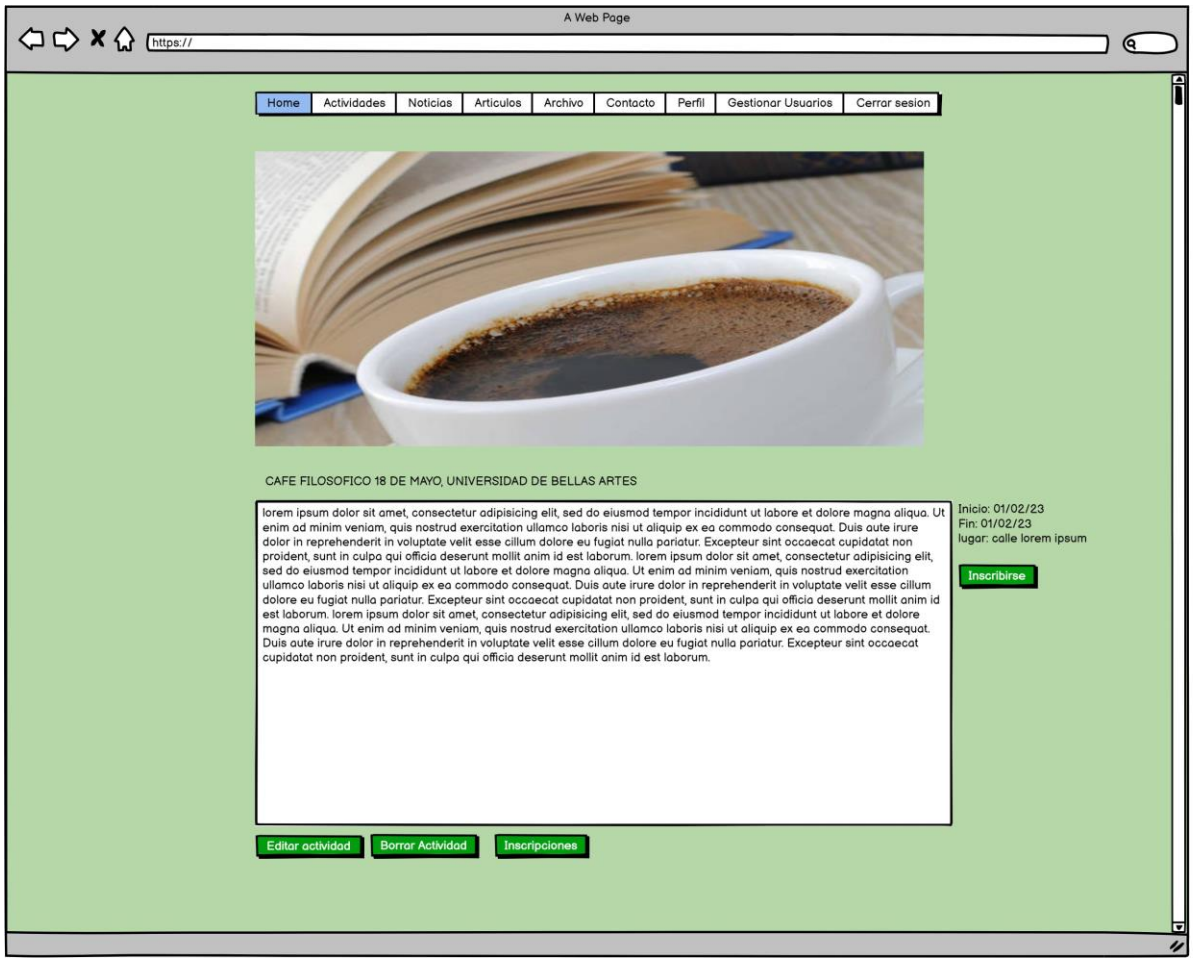

Figura 19: Maqueta interior actividad

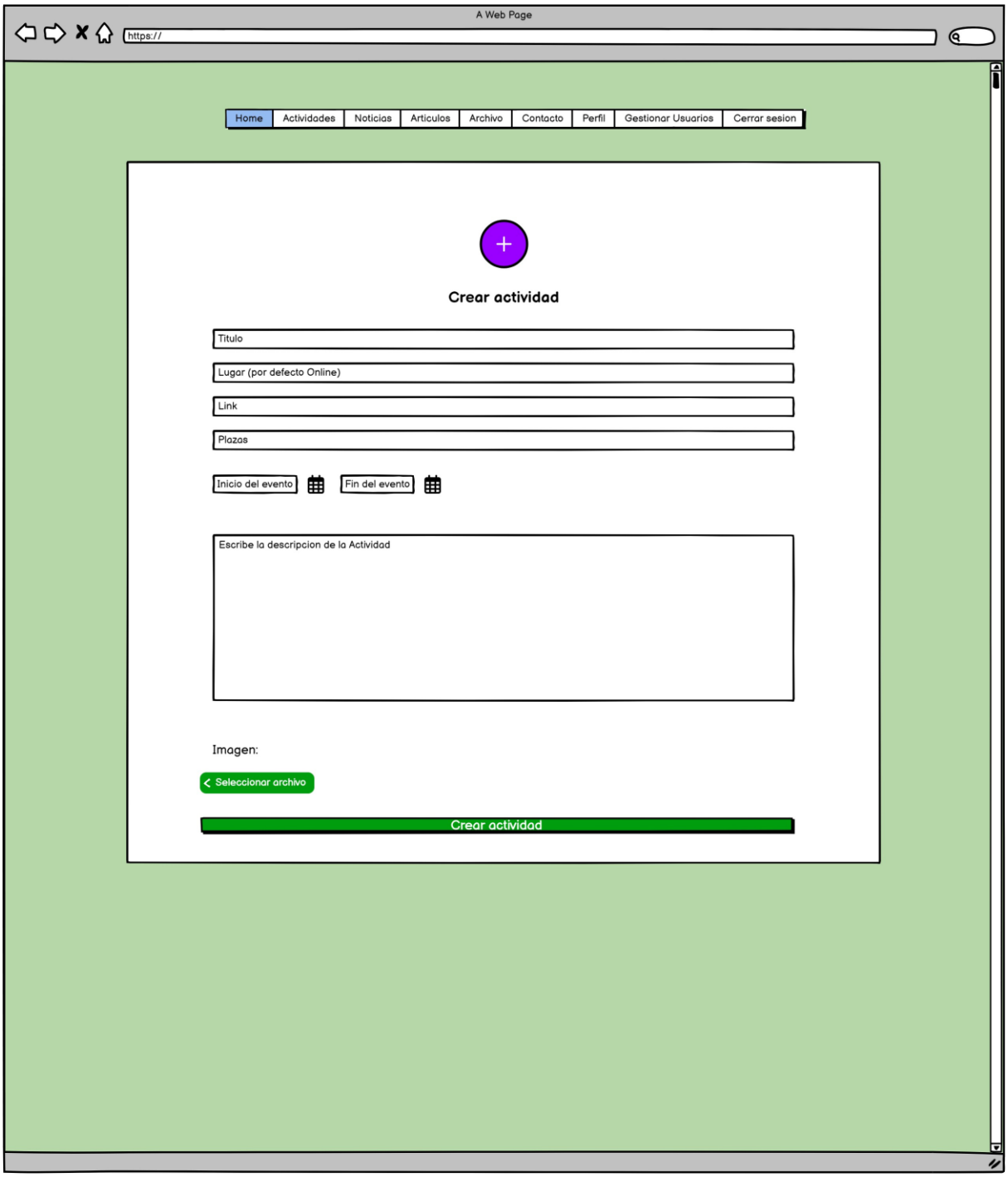

Figura 20: Maqueta crear actividad

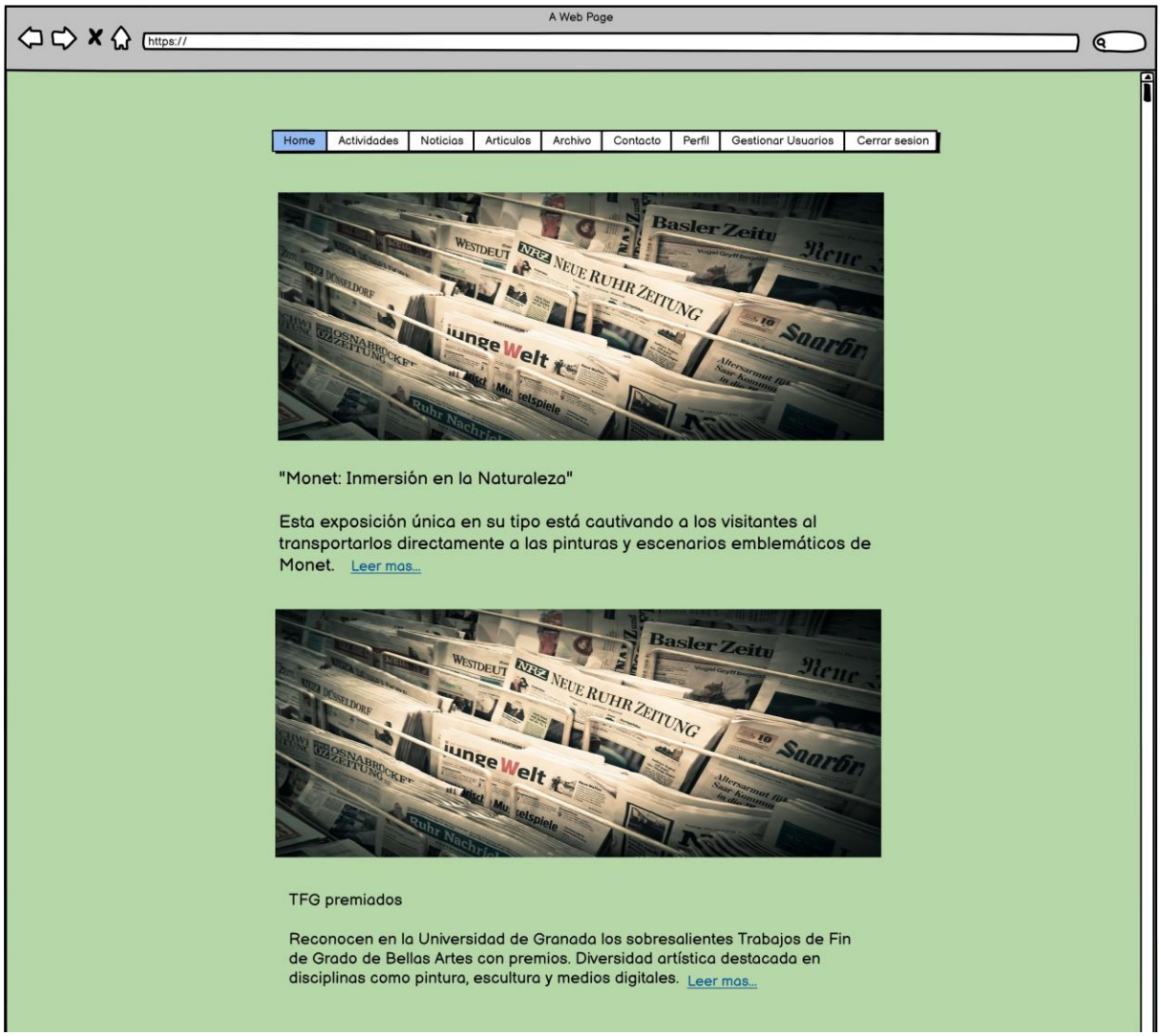

Figura 21: Maqueta noticias

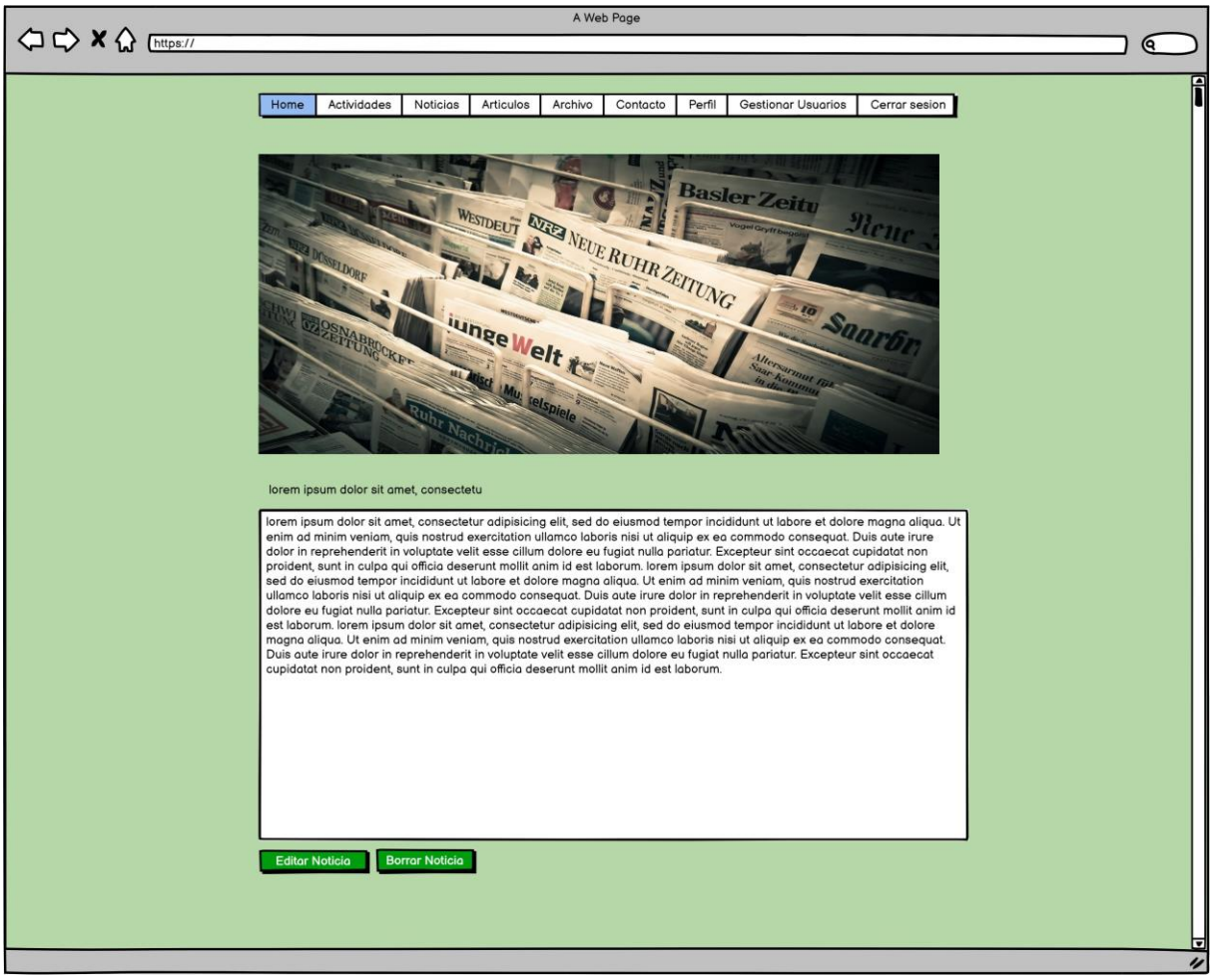

Figura 22: Maqueta interior noticia

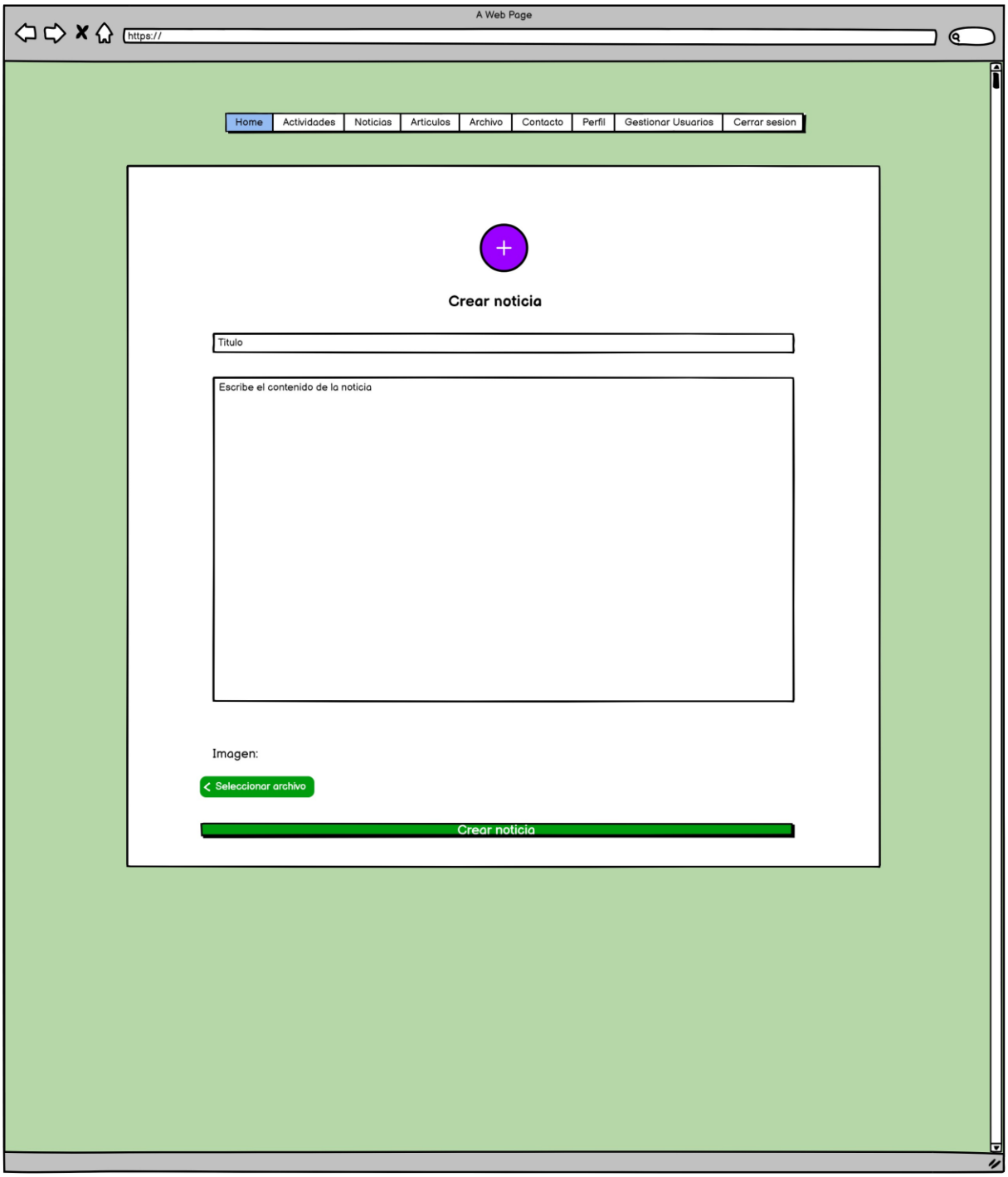

Figura 23: Maqueta crear noticia

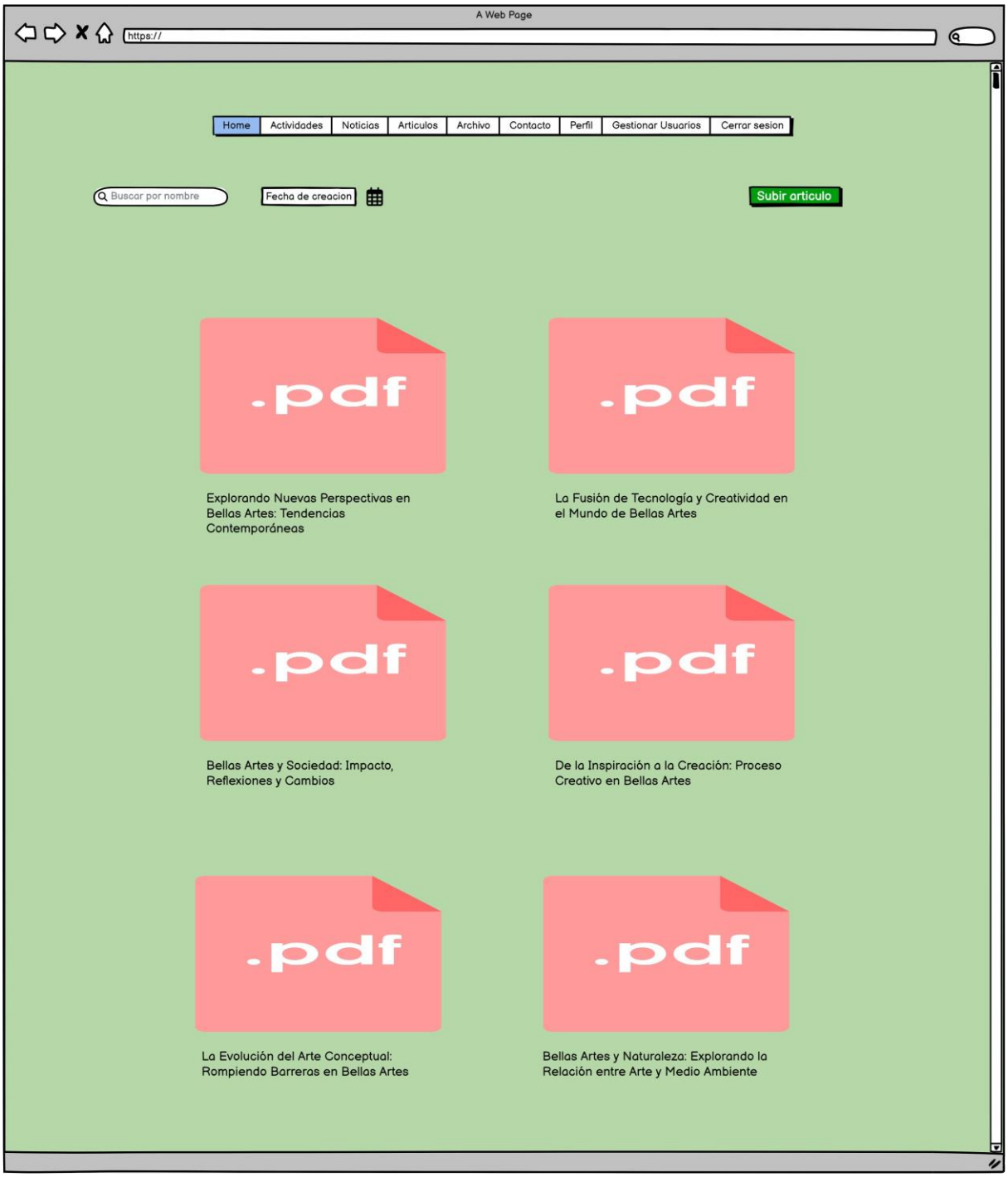

Figura 24: Maqueta artículos

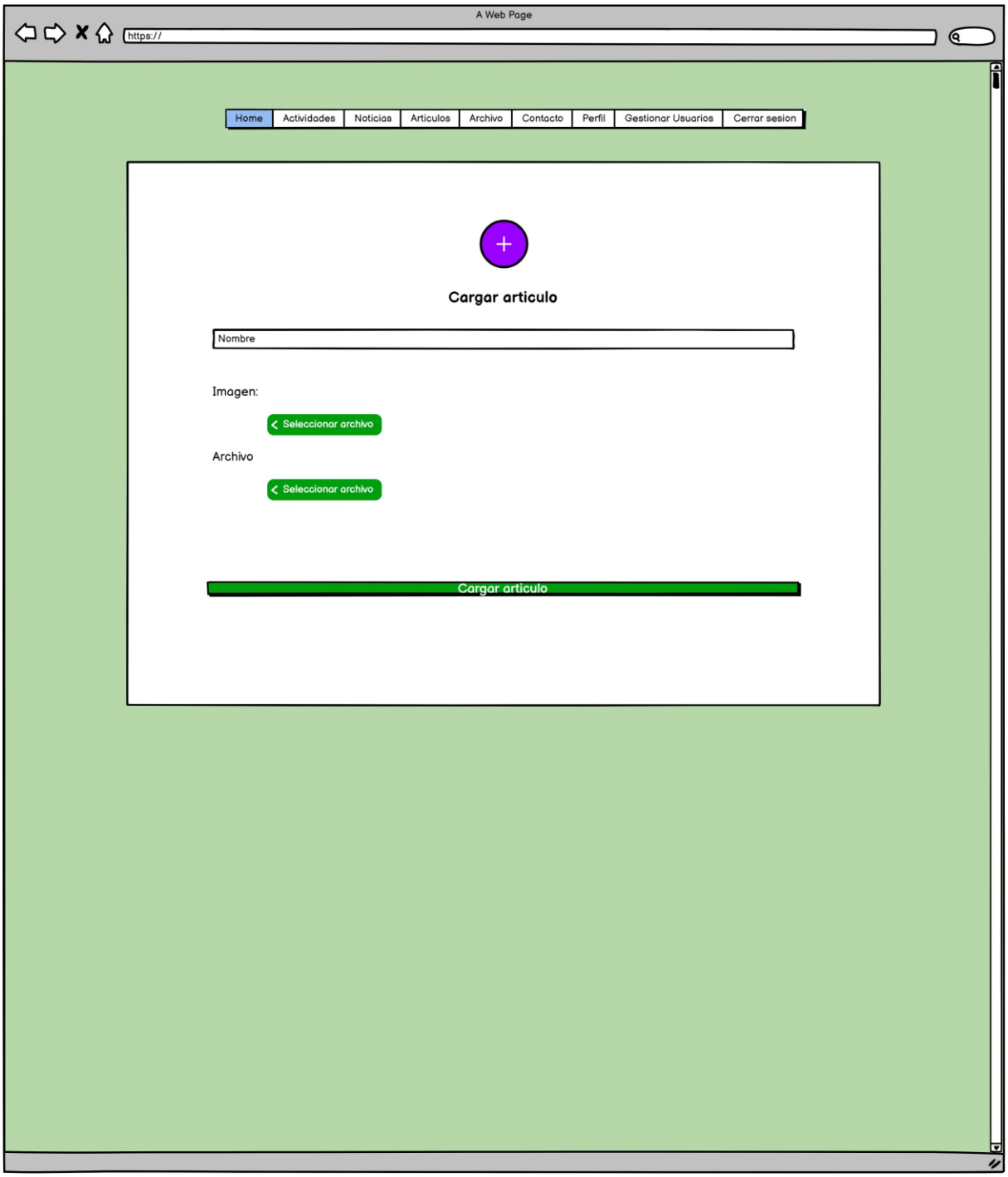

Figura 25: Maqueta subir articulo

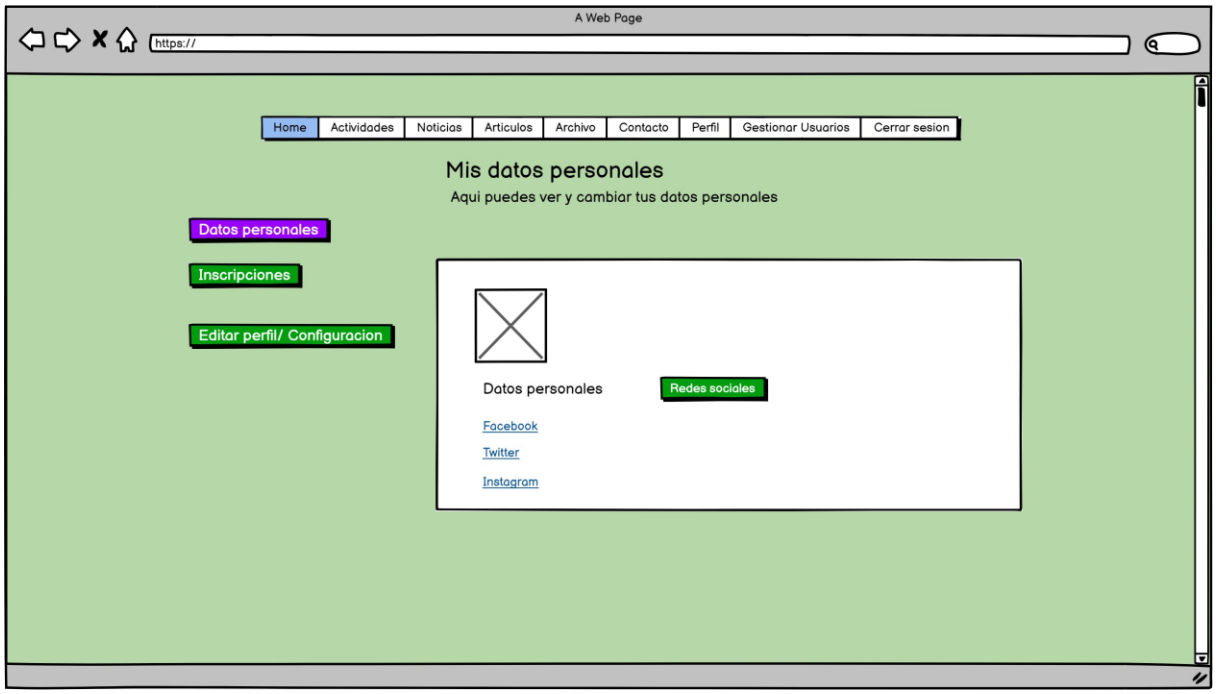

Figura 26: Maqueta perfil

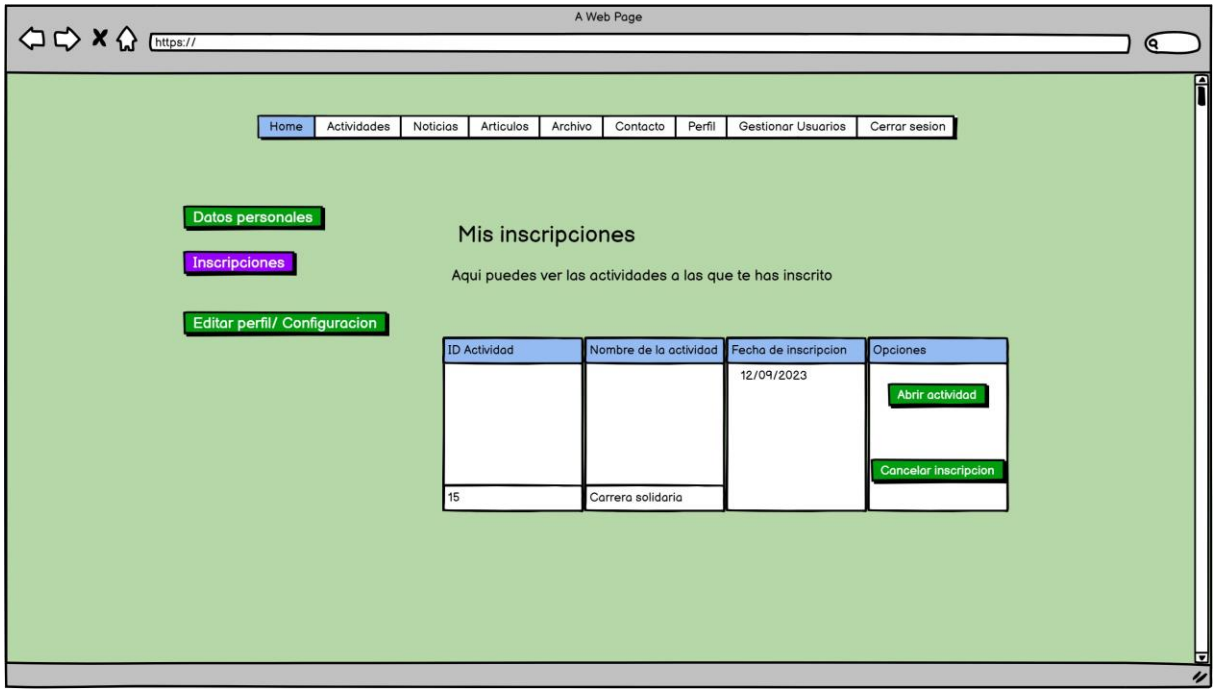

Figura 27: Maqueta inscripciones

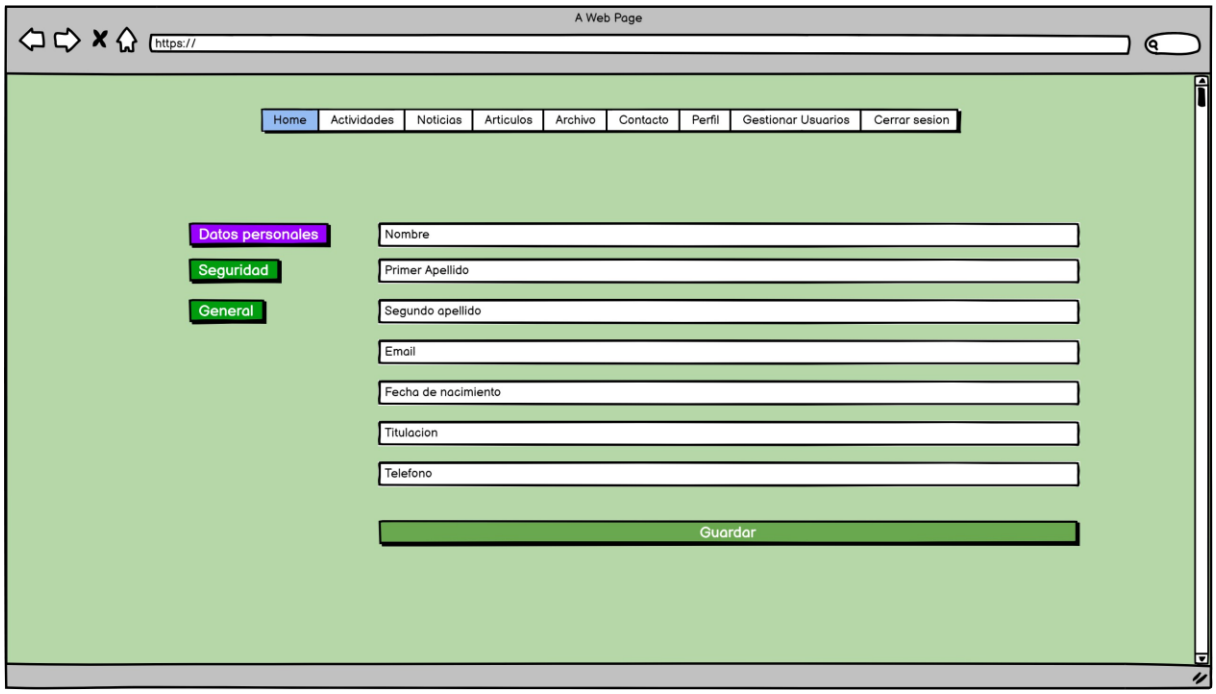

Figura 28: Maqueta editar perfil (1)

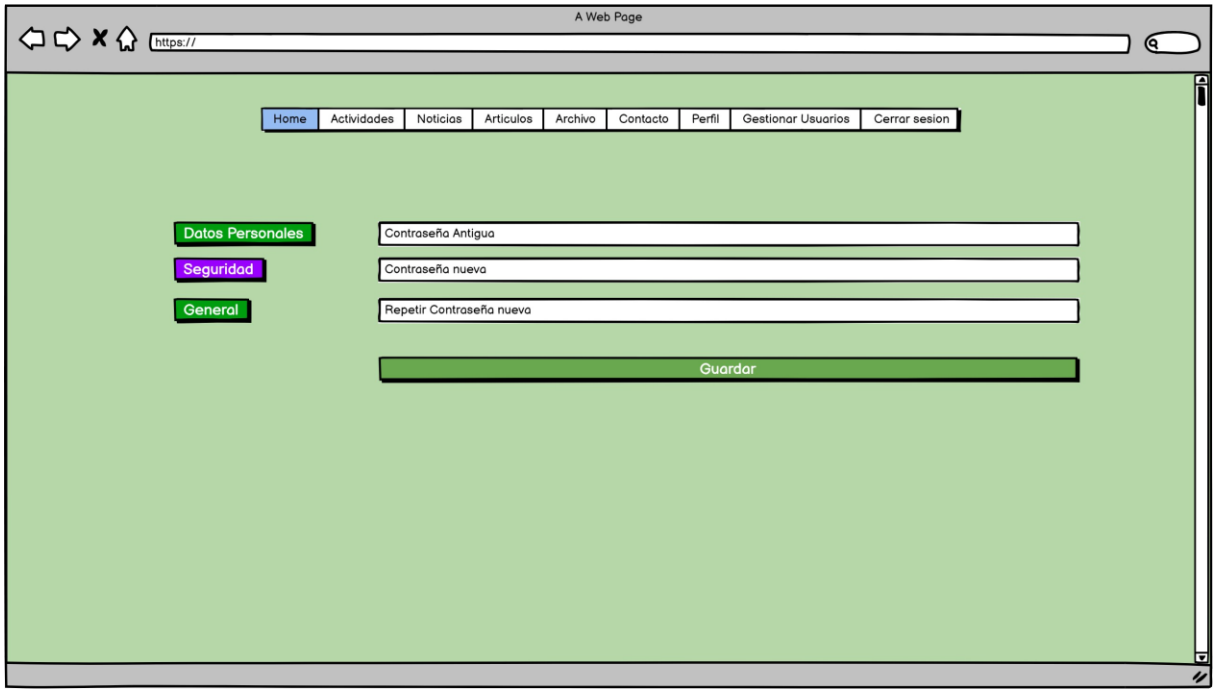

Figura 29: Maqueta editar perfil (2)

Figura 30: Maqueta editar perfil (3)

4

Diseño

A continuación, se detallará los diseños de los diferentes modelos correspondientes al diseño de la aplicación desarrollada.

4.1. Modelos de clases

Se define como modelo de clase a un tipo de diagrama estático que describe la estructura de un sistema mediante la representación de los elementos que la componen [10]

Como la estructura del proyecto sigue la estructura MVC (Modelo-Vista-Controlador) se procederá a enseñar los tres tipos de diagramas que lo componen. Estos son:

- Entidades
- Vistas
- Controladores

4.1.1. Diagrama de clases de Entidad

Indica la relación entre todas las entidades que componen el sistema. Estas contienen atributos propios que indican los elementos que lo forman. Esos son:

- Usuarios
- Noticias
- Actividades
- Artículos
- Inscripciones
- Roles
- Archivo
- Newsletter

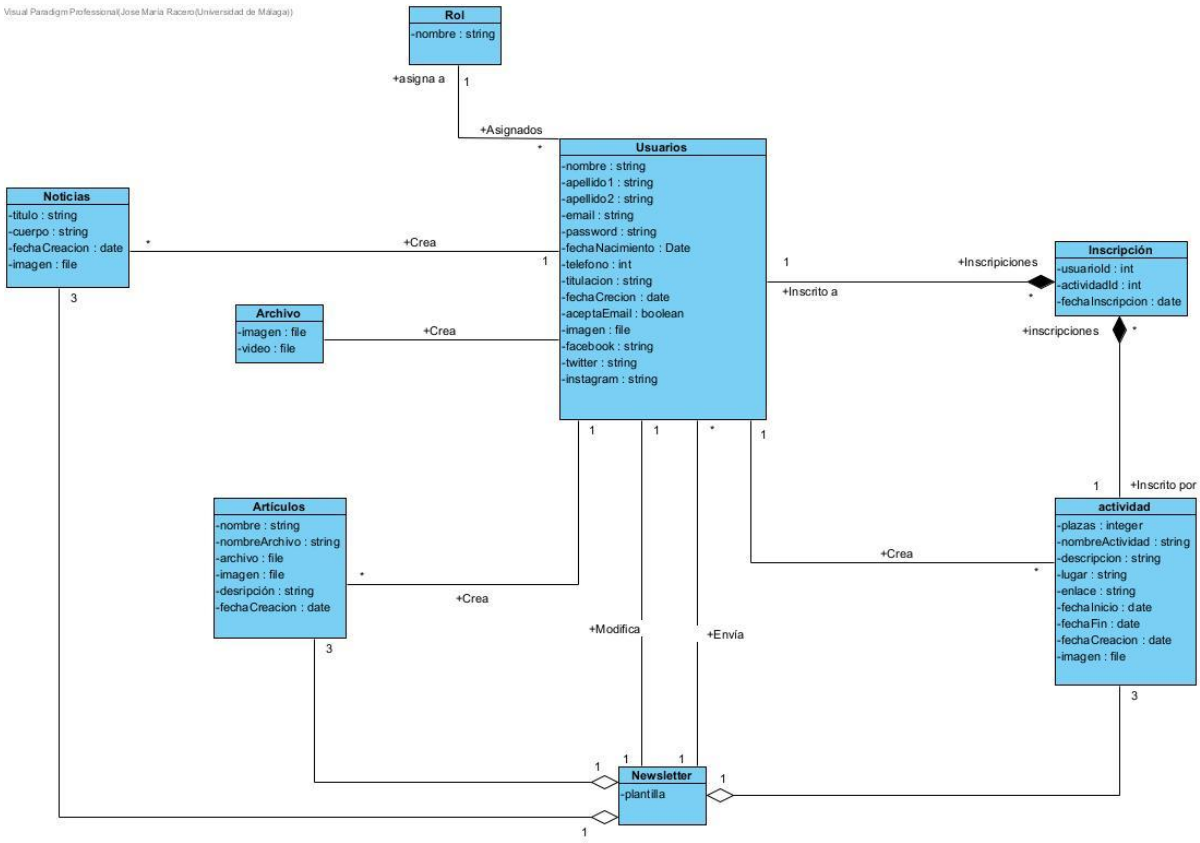

Figura 31: Diagrama de clases de entidad

4.1.2. Diagrama de clases de Control

Indica las clases del tipo control. Estas son las que realizan las acciones lógicas de la aplicación. Cada clase tiene funciones que aplican esta lógica a la aplicación. En esta aplicación siguen la nomenclatura de los proyectos de Django y estarán compuestas por un nombre junto con la palabra View.

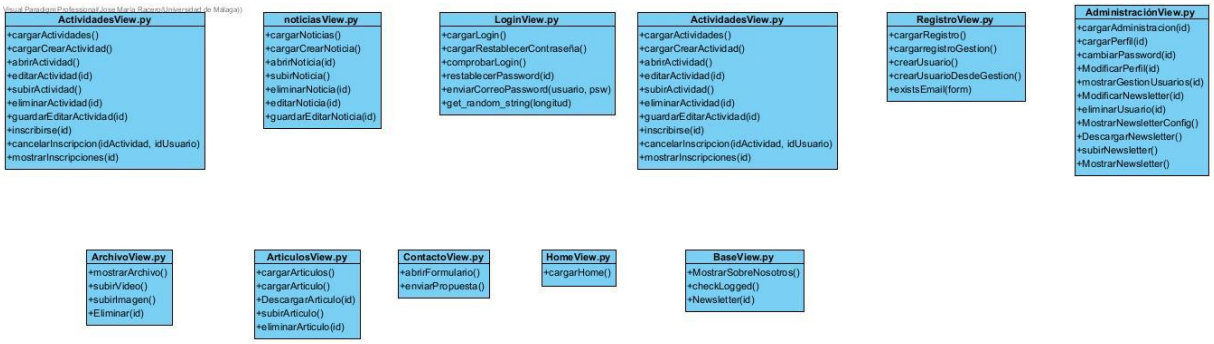

Figura 32: Clases de Control

4.1.3. Diagrama de clases de Interfaz

Indica las clases del tipo interfaz. Estas son las que realizan las comunicaciones entre el actor y la clase de control. Son comúnmente llamadas Vistas.

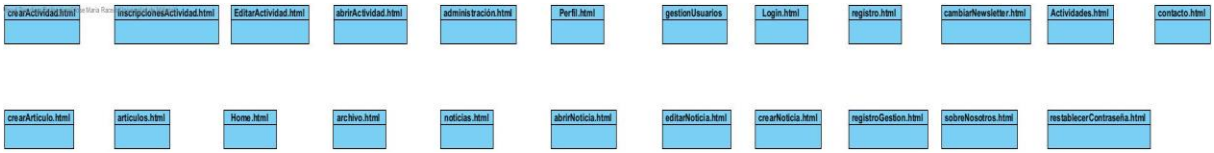

Figura 33: Clases de interfaz

4.2. Modelo físico de datos.

El modelo físico de datos se crea a partir del diagrama de clases de entidad. Este diagrama tiene como objetivo describir como se guardan los datos dentro de la base de datos de la aplicación.

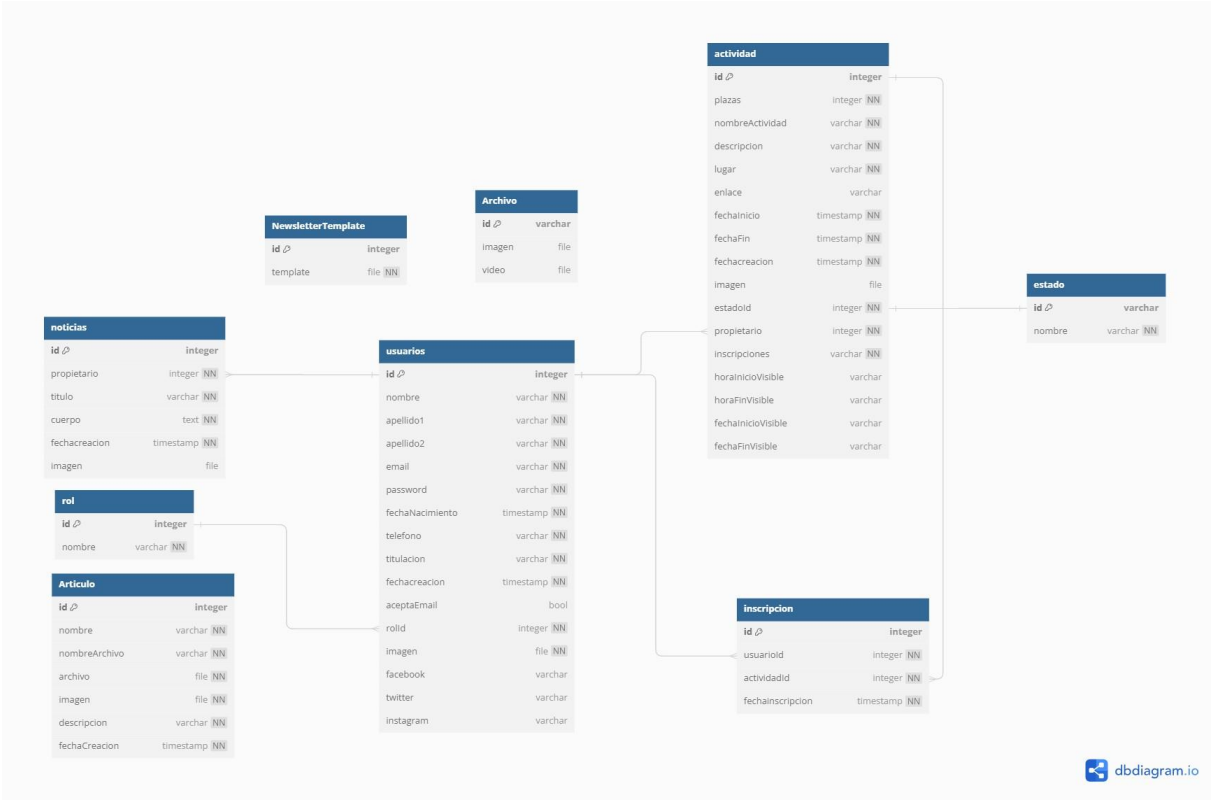

Figura 34: Modelo físico de datos

4.3. Diagrama de secuencia

Un diagrama de secuencia es una representación de como los distintos componentes de una aplicación interactúan entre sí. Estos diagramas se modelan en dos ejes, el eje horizontal y el vertical.

El eje horizontal representa los objetos que interactúan entre sí mientras que el eje vertical representa el espacio de tiempo en el que estas interacciones se llevan a cabo. [11]

En este TFG se han modelado diagramas de secuencia para los casos de uso más importantes de la aplicación.

4.3.1. CRUD Actividades

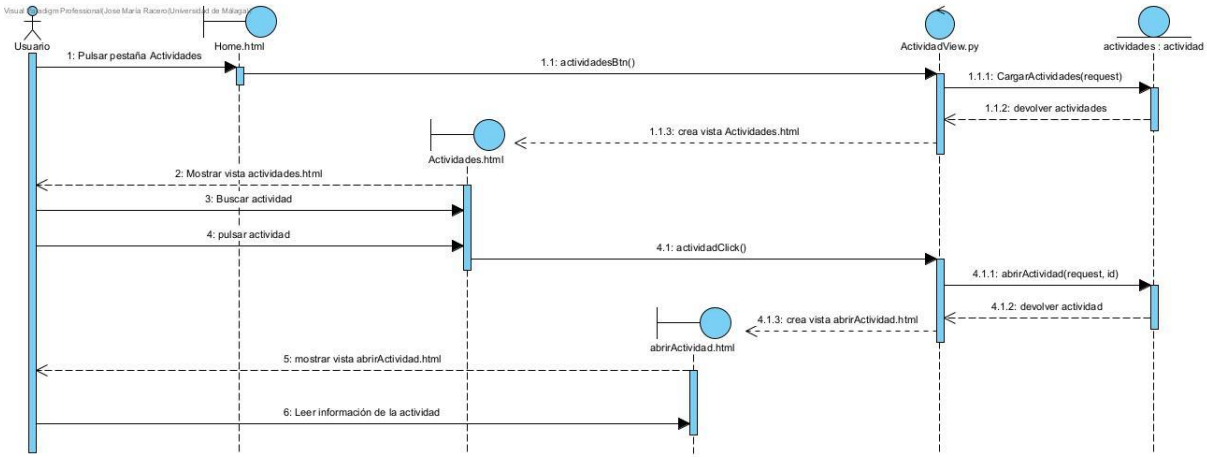

Figura 35: Diagrama de secuencia. Leer actividad

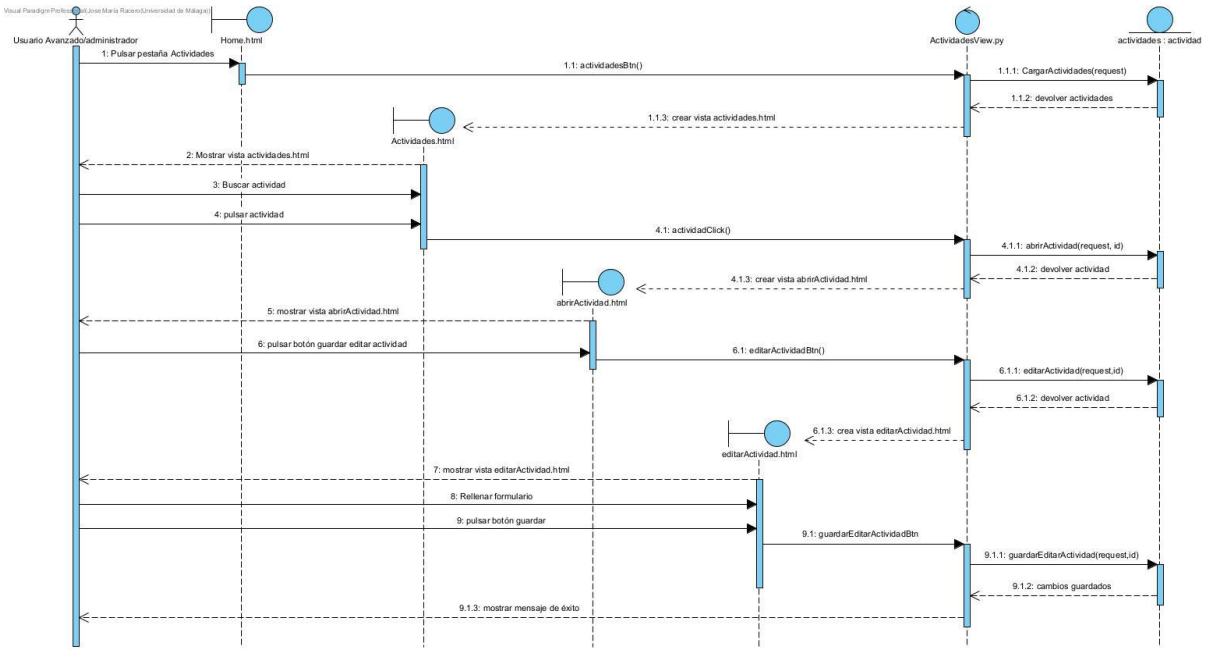

Figura 36: Diagrama de secuencia. Editar actividad

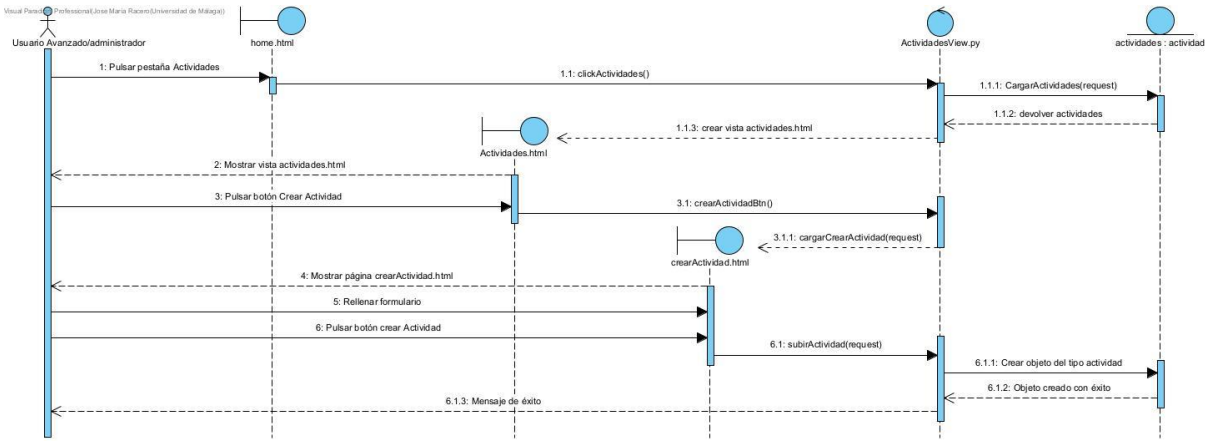

Figura 37: Diagrama de secuencia. Crear actividad

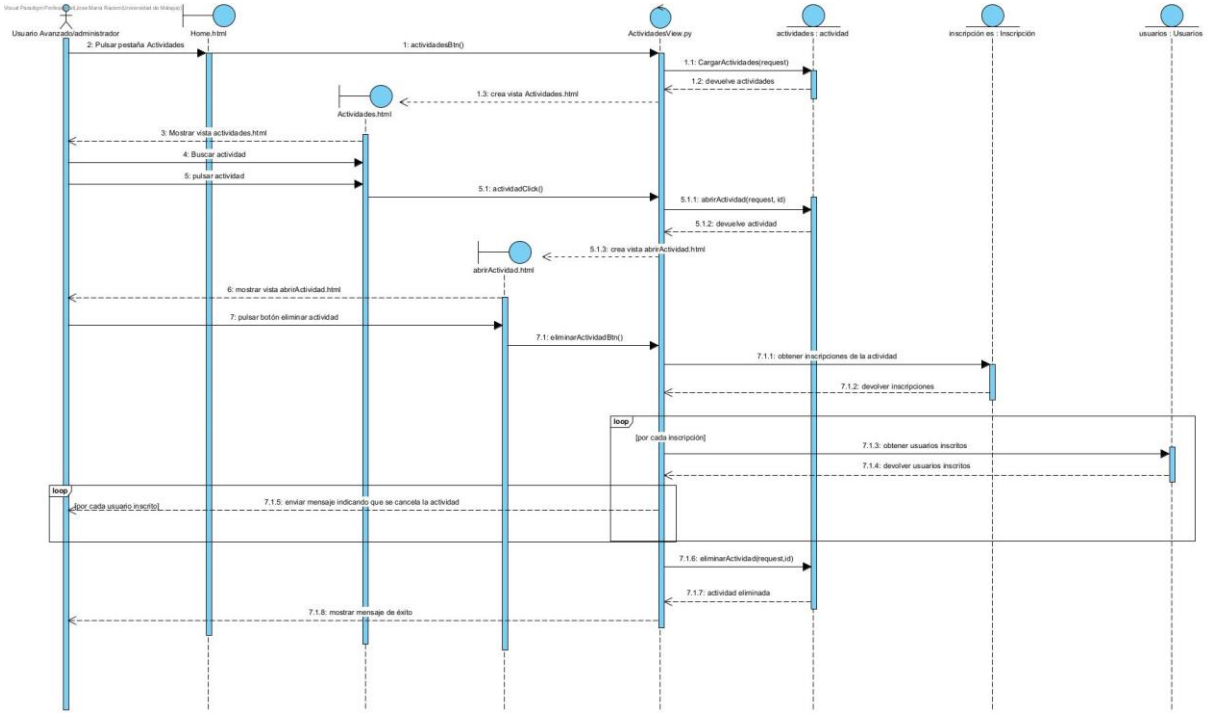

Figura 38: Diagrama de secuencia. Borrar actividad

4.3.2. CRUD Noticias

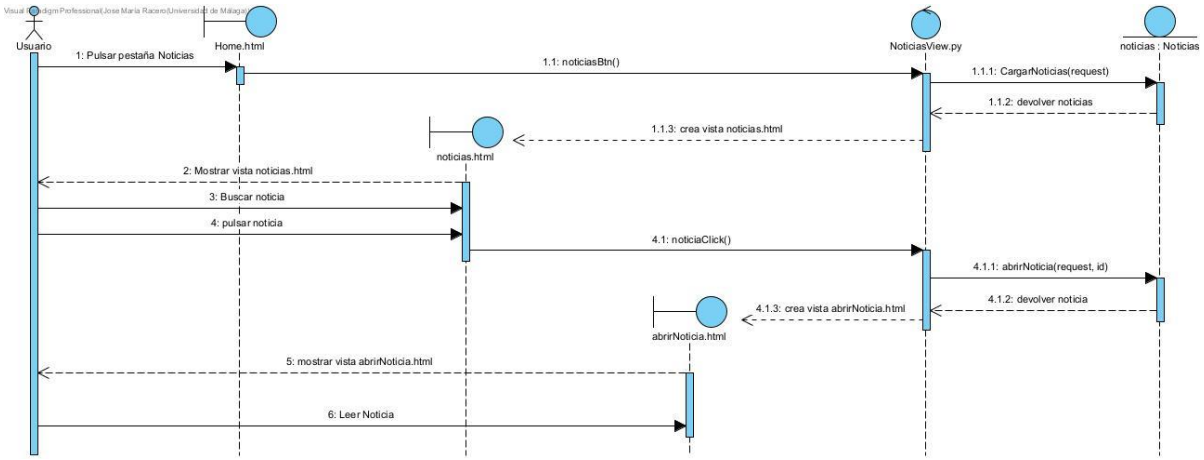

Figura 39:Diagrama de secuencia. Leer noticia

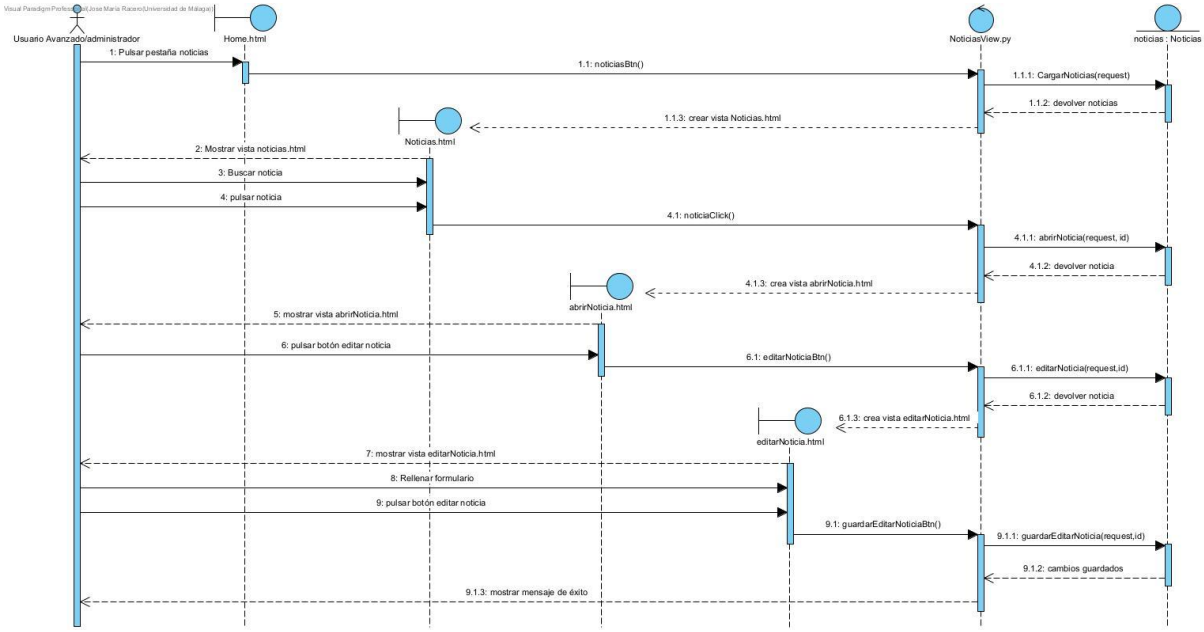

Figura 40:Diagrama de secuencia. Editar noticia

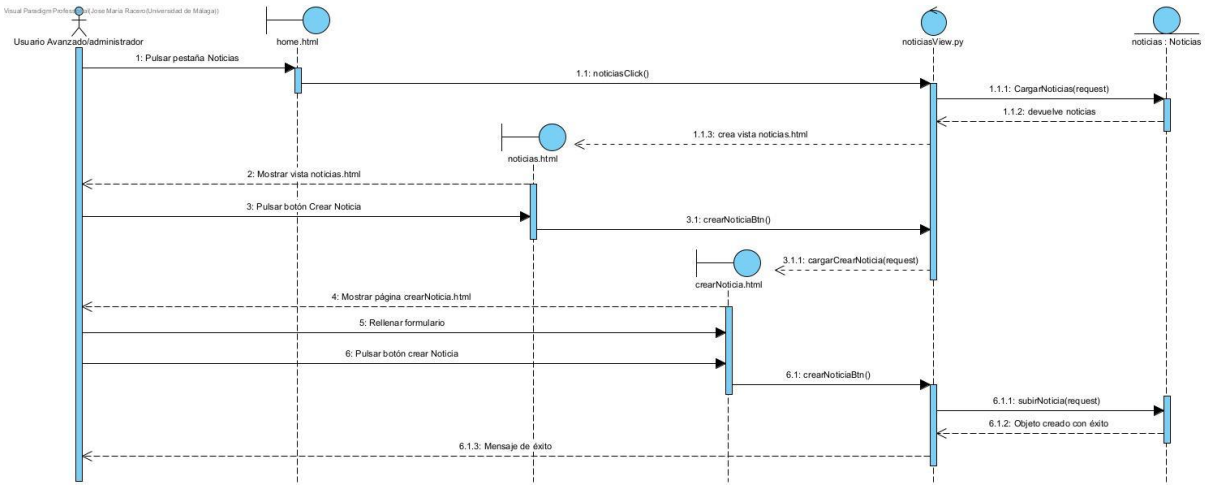

Figura 41:Diagrama de secuencia. Crear noticia

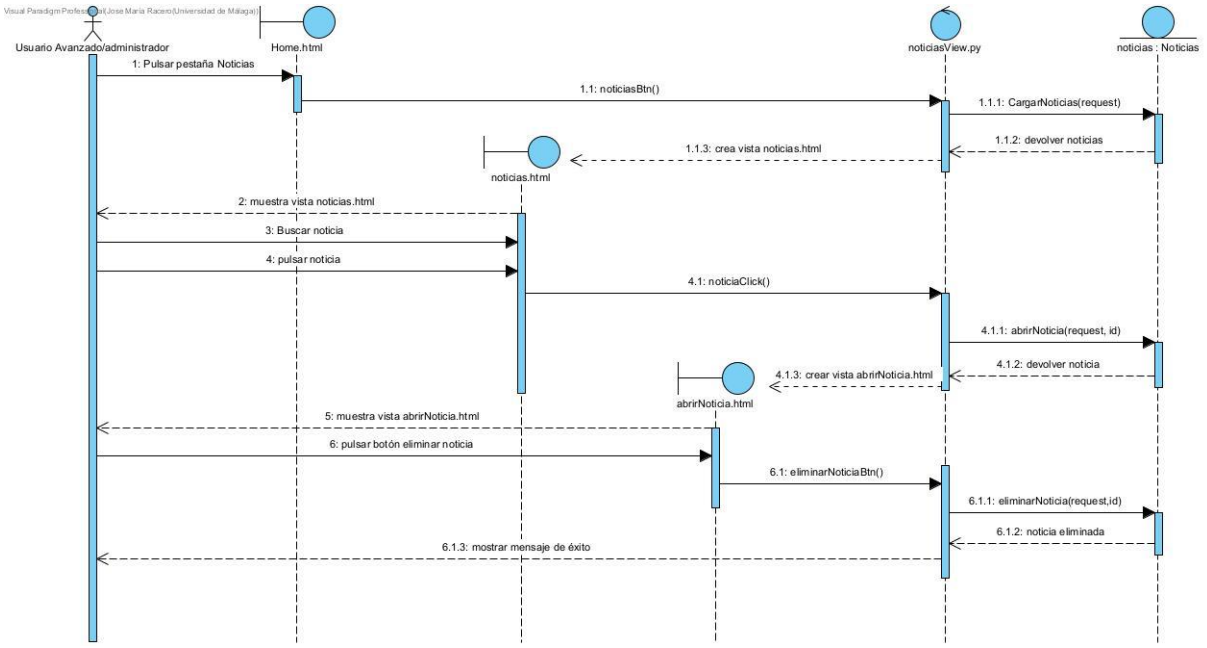

Figura 42:Diagrama de secuencia. Eliminar noticia

4.3.3. CRUD Usuario

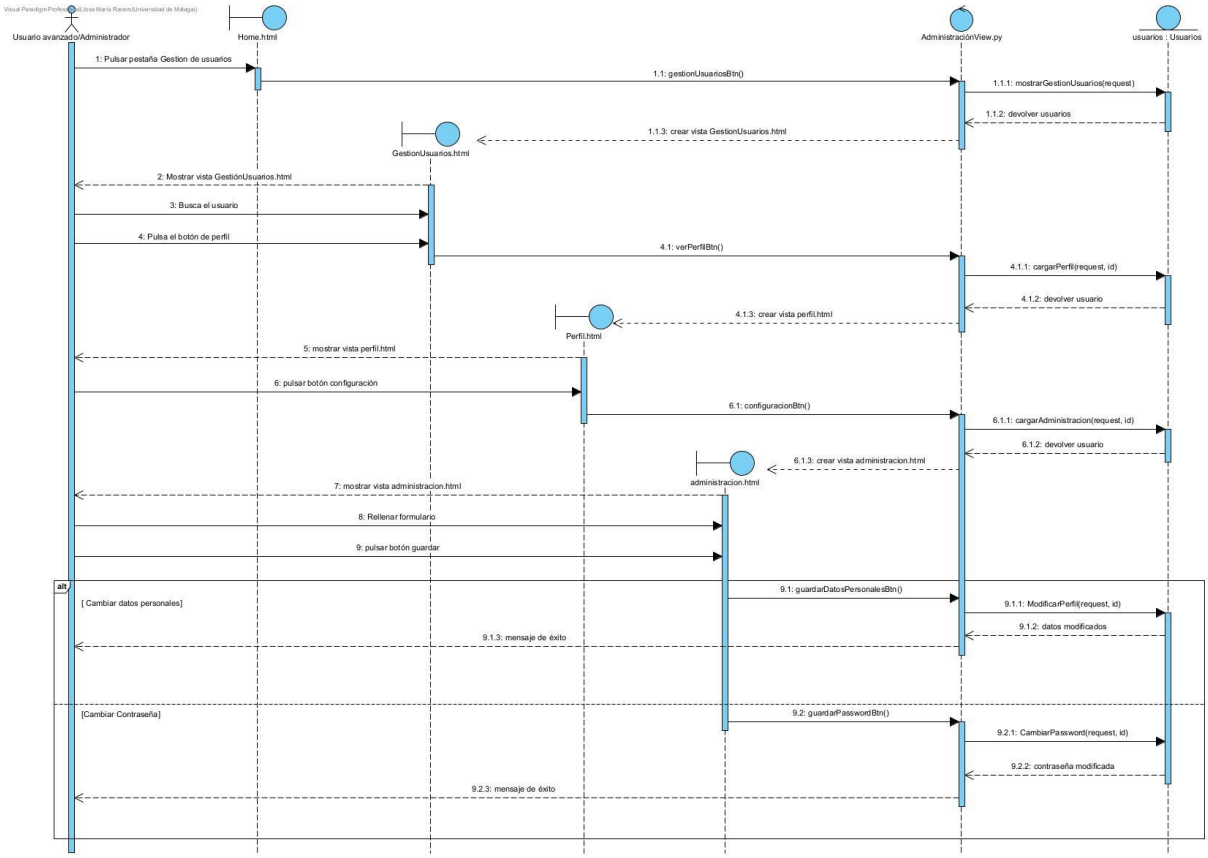

Figura 43:Diagrama de secuencia. Editar usuario

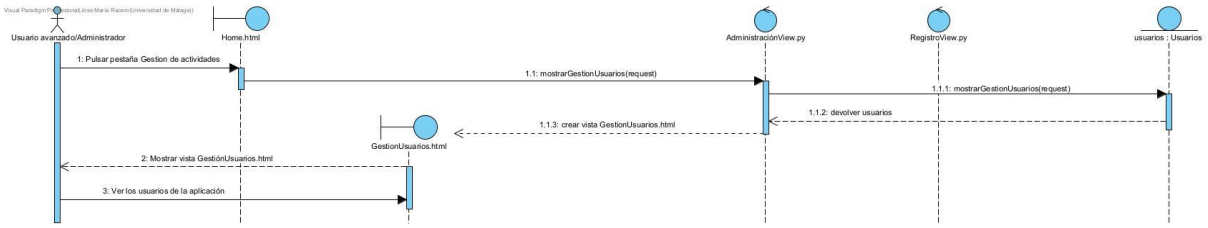

Figura 44:Diagrama de secuencia. Leer usuario

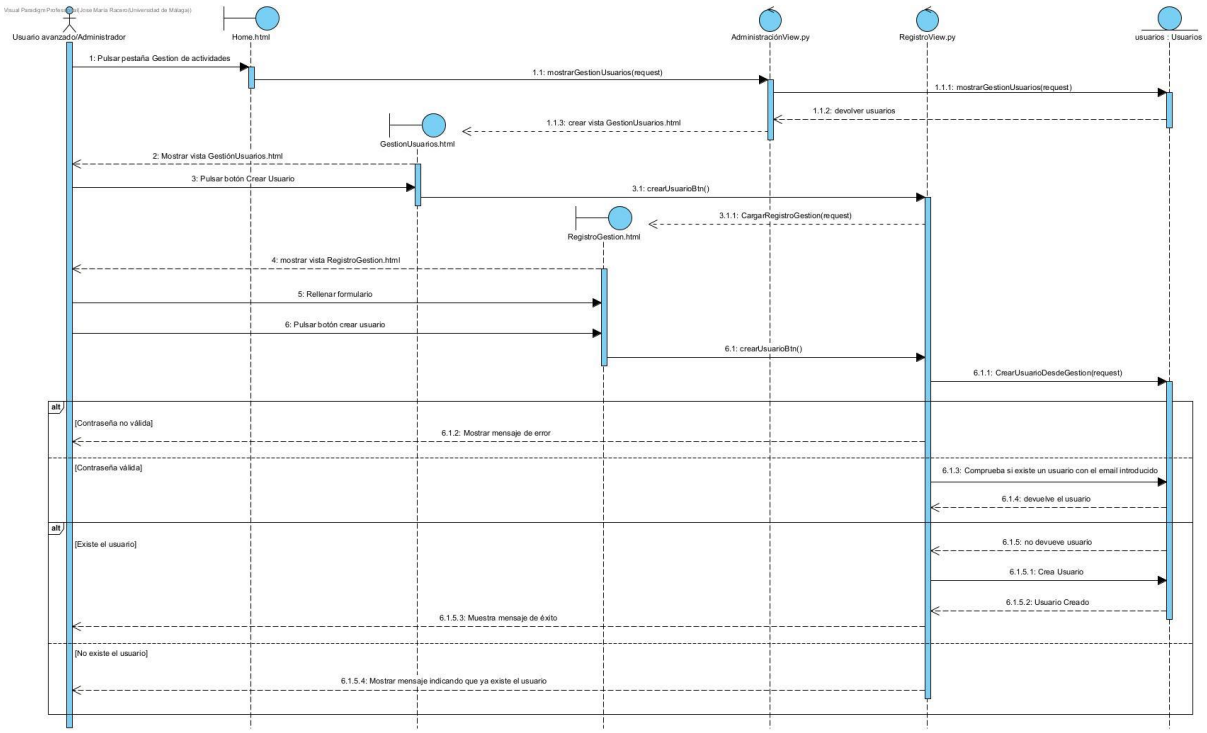

Figura 45:Diagrama de secuencia. Crear usuario

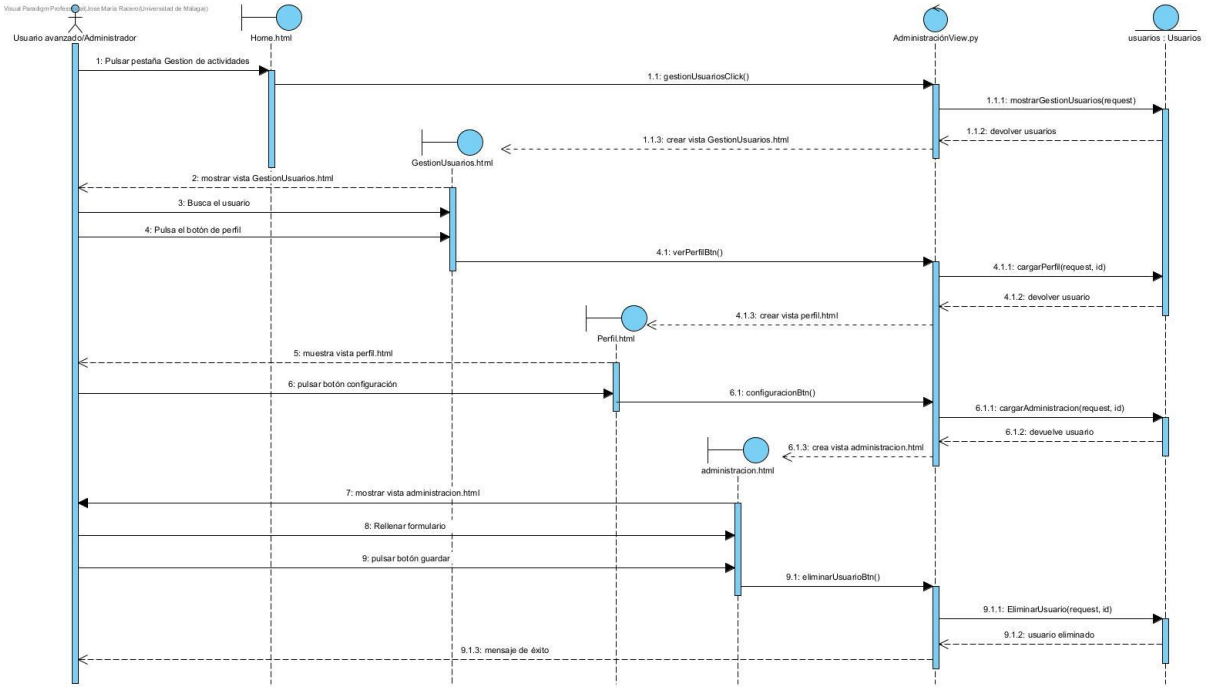

Figura 46:Diagrama de secuencia. Eliminar usuario

4.3.4. Login

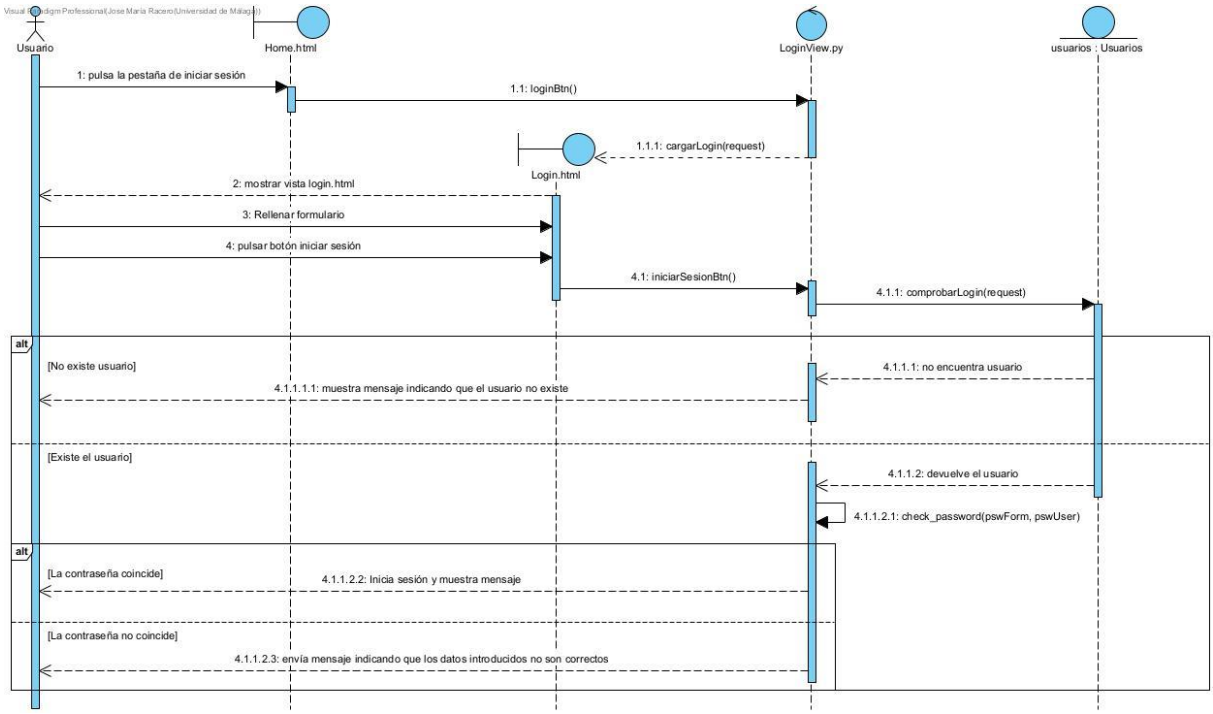

Figura 47:Diagrama de secuencia. Login

4.3.5. Registro

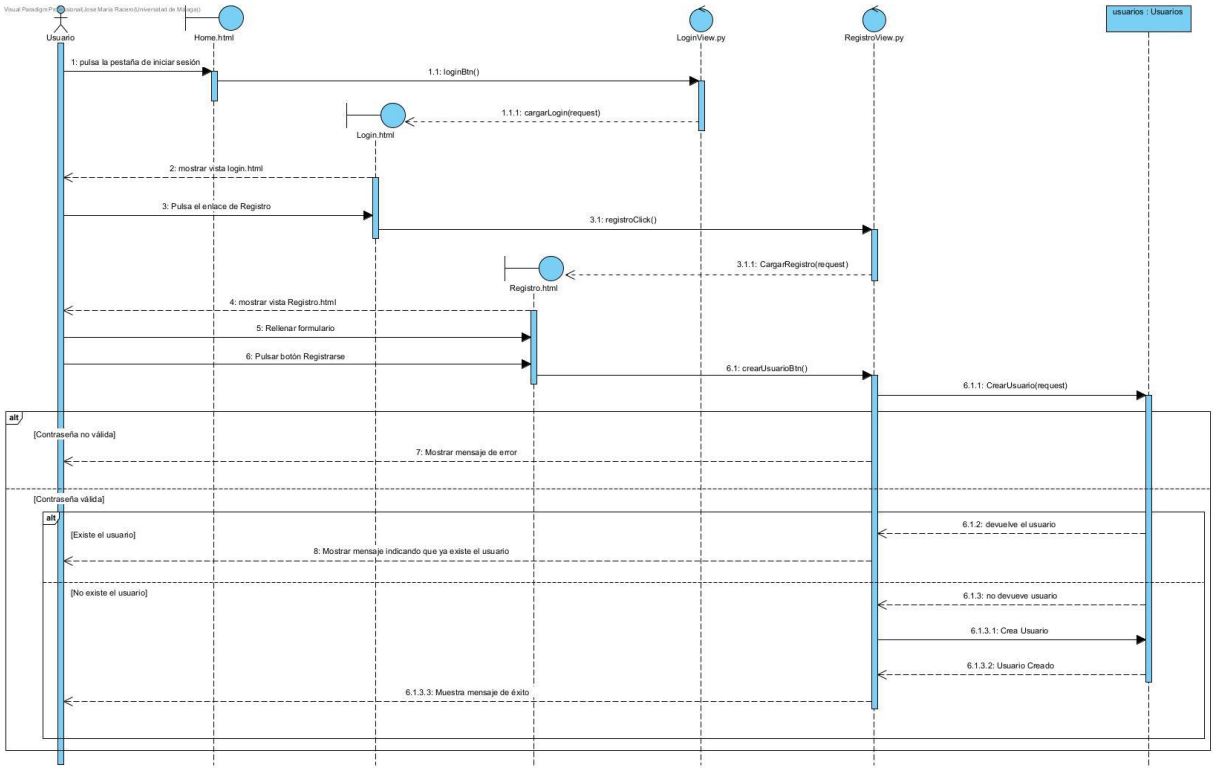

Figura 48:Diagrama de secuencia. Registro

4.3.6. Envío de newsletter

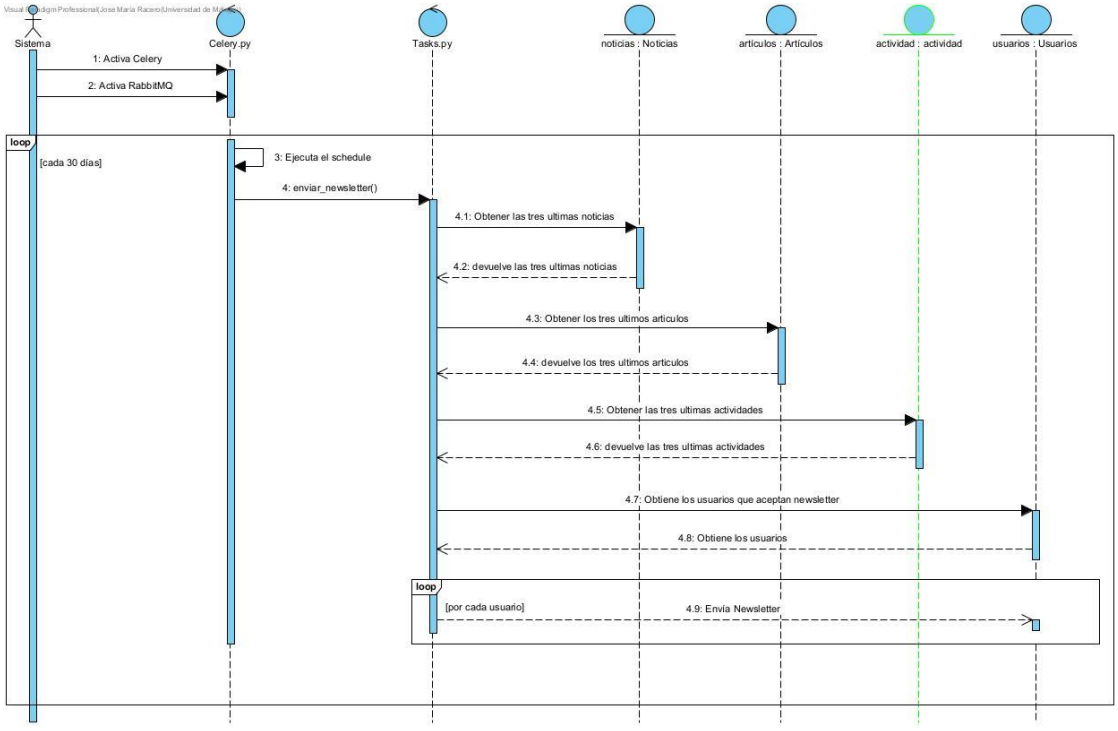

Figura 49:Diagrama de secuencia. Envío de newsletter

4.3.7. Alerta actividad

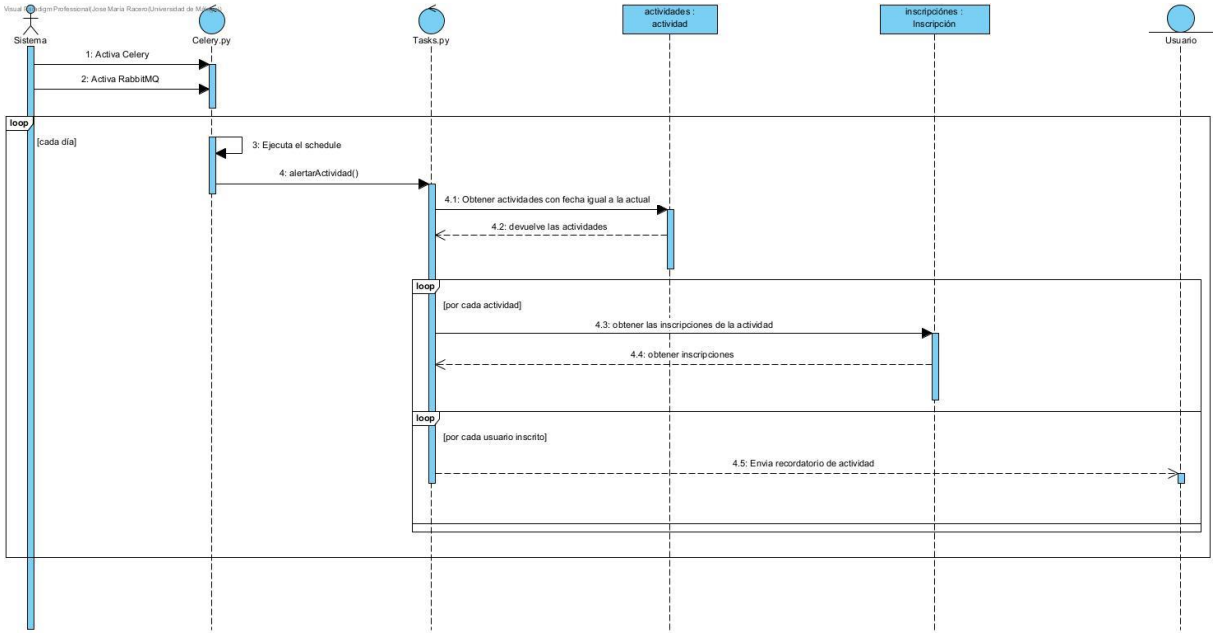

Figura 50:Diagrama de secuencia. Alerta actividad

4.4. Diagrama de arquitectura.

Un diagrama de arquitectura es aquel que muestra la relación entre los componentes software y físicos que hacen posible que la web funcione.

Este proyecto se ejecuta de manera local. Por ende, el usuario se conectará a través de su ordenador al servidor local que habrá debido desplegar previamente. Este servidor se conectará a la base de datos local de MongoDB. Si el usuario desea que se realicen las tareas automáticas deberá iniciar el servicio de RabbitMQ. El diagrama de arquitectura va a lucir de la siguiente forma:

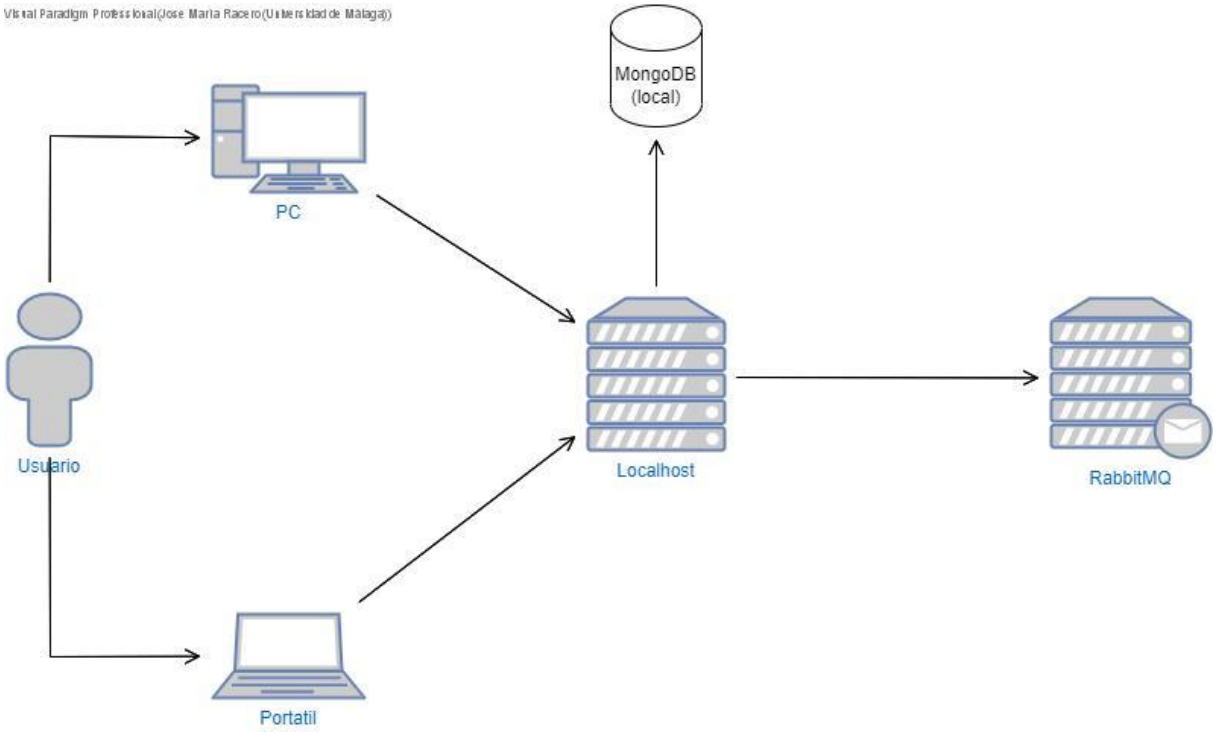

Figura 51: Diagrama de arquitectura

En un futuro se realizará un despliegue de la aplicación. Esto supondrá un cambio en la arquitectura.

El diagrama, una vez desplegada la aplicación, será el siguiente

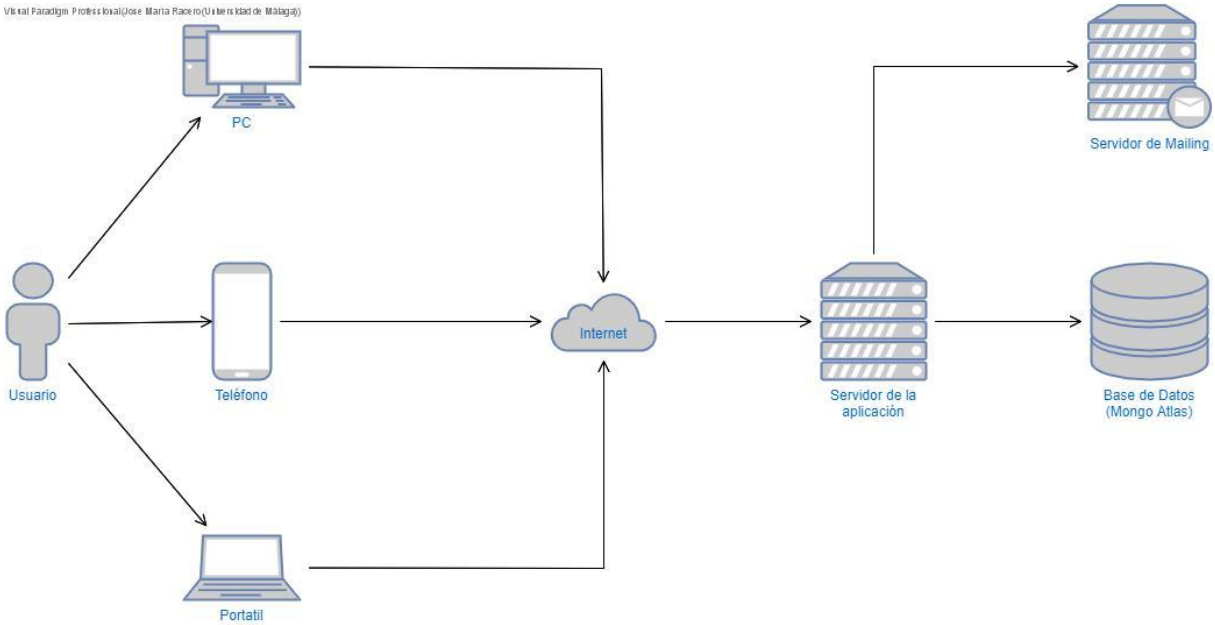

Figura 52: Diagrama de arquitectura una vez desplegado

4.5. Entorno tecnológico

Para la creación de la web se han empleado diferentes herramientas tecnológicas. En este apartado se van a enumerar todas ellas.

4.5.1. Django

Django es un framework de desarrollo web de código abierto escrito en Python. Tiene como finalidad el rápido desarrollo de aplicaciones web. Es altamente escalable, seguro y rápido. [12]

4.5.2. JavaScript

JavaScript es un lenguaje de programación que se compila en tiempo de ejecución. Es el lenguaje más conocido para la creación de páginas web. Está basado en prototipos [13]

4.5.3. HTML 5

HTML es un lenguaje de etiquetas diseñado para la creación de páginas web. Este lenguaje es el estándar para la creación de páginas web. [14]

4.5.4. Bootstrap

Bootstrap es un framework CSS de código abierto que favorece y facilita el desarrollo web [15]

4.5.5. MongoDB

MongoDB es una base de datos no relacional y gratuita orientada al almacenamiento de documentos [16]

4.5.6. Visual Paradigm

Visual Paradigm es una herramienta de diseño y gestión de sistemas tecnológicos, también llamado herramienta UML [17]

4.5.7. Dbdiagram.io

Dbdiagram.io es una web que provee una herramienta gratuita para hacer diagramas entidad-relación [18]

4.5.8. Taiga

Taiga es una herramienta de gestión de proyectos gratuita especializada para metodologías ágiles como Kanban o Scrum [19]

4.5.9. RabbitMQ

RabbitMQ es un bróker de mensajería de código abierto programado en el lenguaje Erlang. Sirve para crear colas de correo. [20]

4.5.10. Celery

Celery es una cola de tareas asíncrona de código abierto para Django. [21]

5

Implementación

En este apartado se va a explicar cómo se estructura el proyecto, como se ha implementado y los casos de prueba que se han realizado.

5.1. Estructura del proyecto

El proyecto sigue la estructura de una aplicación Django estándar. Este consiste en lo siguiente:

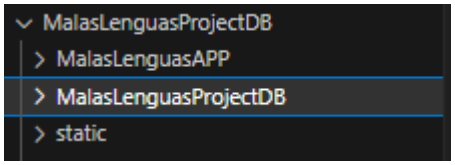

Figura 53: Estructura principal del proyecto

El terminado en APP indica la aplicación, el terminado en DB contiene los archivos de configuración del proyecto y static contiene todos los archivos estáticos de la aplicación. Dentro del proyecto está incluido el archivo settings.py que contiene toda la información relacionada con la aplicación y el entorno. Tanto las conexiones con la base de datos como las variables globales como las aplicaciones instaladas.

Para implementar el proyecto será necesario una serie de objetos. Estos son:

5.1.1. Modelos

En Django los modelos son la fuente de información de los datos dentro de la aplicación. En ella se general los campos y comportamientos de los datos que se almacenan. Estos están vinculados a la base de datos en la cual se almacenan estos datos. [22]

Estos modelos se almacenan en un fichero llamado models.py dentro de la carpeta de la aplicación.

Lo primero es importar el models de Django.db

```
from django.db import models
```

Figura 54: Importación de models

Cada modelo es una clase y se implementa de la siguiente forma:

```
class Noticias(models.Model):
    propietarioId = models.ForeignKey(Usuario, on_delete=models.CASCADE, null=False)
    titulo = models.CharField(max_length=50, default="", null=False)
    cuerpo = models.TextField(null=False)
    fechaCreacion = models.DateTimeField(null=False, auto_now_add=True)
    imagen = models.ImageField(upload_to='static/Noticias/Images/', default="static/Noticias/Images/noticiaGenerica.jpg")
```

Figura 55: Ejemplo de modelo. Noticias

Se llama la clase con el mismo nombre de la tabla a la que referencia introduciéndole models.Model como parámetro.

Dentro de la clase se definen todos los campos que van a almacenar en la base de datos. En este caso las noticias cuentan con los campos

- Propietario
- Titulo
- Cuerpo
- Fecha de creación
- Imagen

El tipo de dato que se almacena se indica llamando a las funciones del models. Por ejemplo, para crear un campo de texto llamaríamos a models.CharField() y dentro

indicaríamos los diferentes atributos como la nulabilidad, el tamaño máximo, el valor por defecto, etc…

5.1.2. Templates

En Django las vistas se llaman templates y se almacenan en la carpeta con el mismo nombre. La estructura de carpetas que he elegido es Templates – Nombre– archivo html

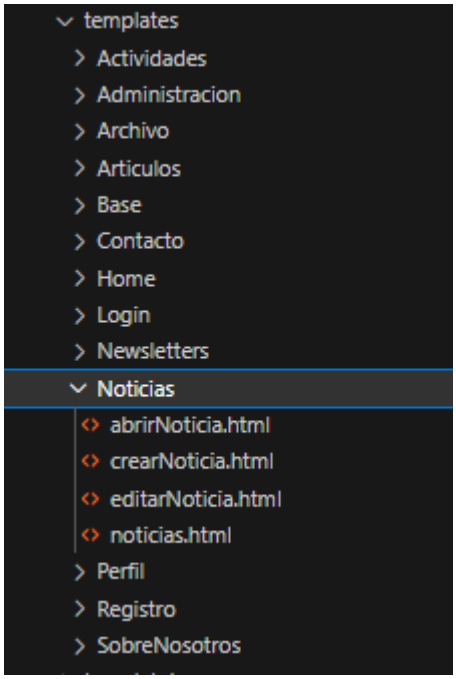

Figura 56: Estructura de vistas

5.1.3. Views

En Django las clases de control tienen el nombre de Views y se almacenan en la carpeta con el mismo nombre. La estructura de carpetas que he elegido es Views – Nombre – Archivo Python.

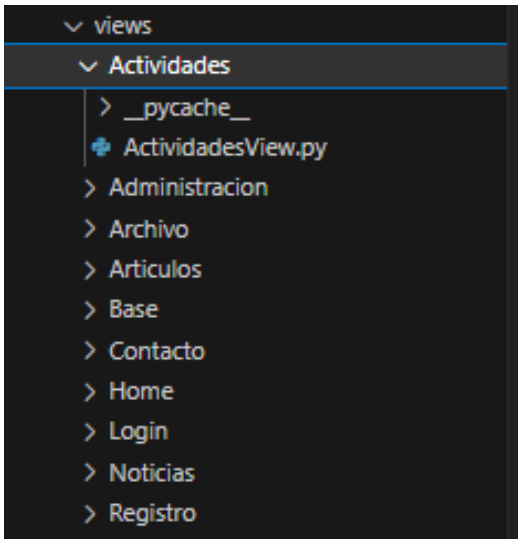

Figura 57: Estructura de Controles

5.1.4. URLs

En Django las llamadas a las funciones del controlador se realizan mediante enlaces. Cada enlace está asociado a una función de los controladores. Estos enlaces se pueden llamar desde otros controladores o desde las vistas.

Estos enlaces se almacenan en el fichero urls.py dentro de la aplicación.

El fichero consiste en una variable llamada urlpatterns en la cual se insertan todos los enlaces con las llamadas a funciones. Por ejemplo, para crear una actividad tendríamos que llamar al siguiente enlace

```
path(r'Actividades/CrearActividad', ActividadesView.cargarCrearActividad),
```

Figura 58: Ejemplo llamada de función

Como se puede observar está compuesto por una dirección y una función del controlador.

5.2. Descripción de la implementación

Para este apartado voy a mostrar la implementación del caso de uso CRUD Actividades.

Las actividades están compuestas por 4 vistas

- abrirActividad.html
- actividades.html
- crearActividad.html
- inscripcionesActividad.html

y un controlador

- ActividadesView.py

5.2.1. Actividades.html

La vista consiste en una serie de cards de Bootstrap con información de las actividades.

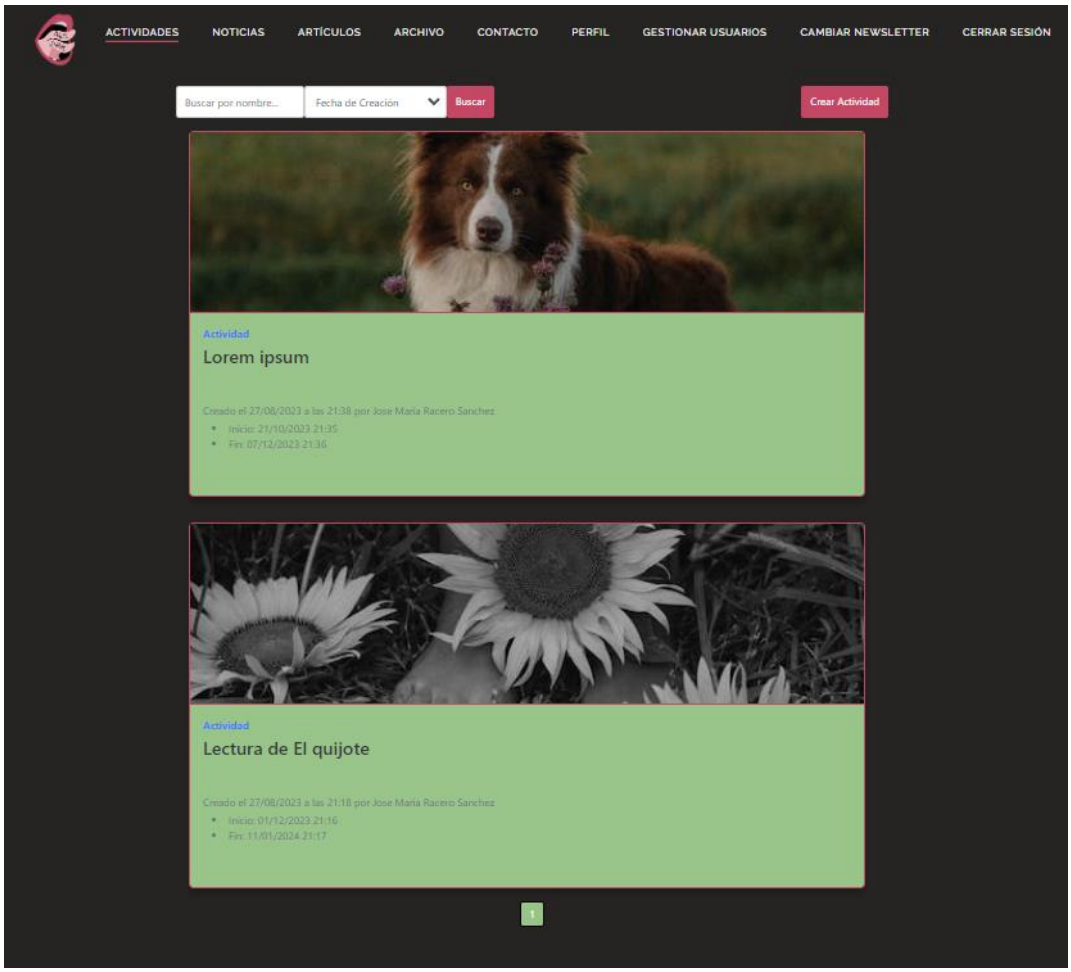

Figura 59: Pagina de actividades

Esta vista se muestra por la función cargarActividades().

```python
def cargarActividades(request):
    context = {}
    if 'idUsuarios' in request.session:
        context['usuarioLogged'] = True

        usuarioId = request.session['idUsuarios']
        usuario = Usuario.objects.get(pk=usuarioId)
        rol = usuario.rolId
        context['rol'] = rol.id
    filtro = request.GET.get('filtro')
    search_query = request.GET.get('q')

    if filtro == 'titulo':
        if search_query:
            actividades = Actividad.objects.filter(nombreActividad__icontains=search_query).order_by('nombreActividad')
            actividades = sorted(actividades, key=lambda x: x.nombreActividad.lower())
        else:
            actividades = Actividad.objects.all().order_by('nombreActividad')
            actividades = sorted(actividades, key=lambda x: x.nombreActividad.lower())
    else:
        if search_query:
            actividades = Actividad.objects.filter(nombreActividad__icontains=search_query).order_by('-fechaCreacion')
        else:
            actividades = Actividad.objects.all().order_by('-fechaCreacion')

    paginator = Paginator(actividades, 8)
    page_number = request.GET.get('page')
    page_obj = paginator.get_page(page_number)

    context['page_obj'] = page_obj
    context['search_query'] = search_query

    return render(request, 'Actividades/actividades.html', context)

```

Figura 60: función cargarActividades

Esta función carga todas las actividades de la base de datos y carga la vista actividades.html junto con las actividades introducidas en el contexto. Además, gracias a la paginación de Django, del paquete django.core.paginator [23] indicamos que se muestren 8 actividades por página.

En esta vista podemos realizar 2 acciones.

- Abrir la actividad
- Crear una actividad

5.2.2. CrearActividad.html

Es la vista que se muestra cuando creamos una actividad. Consiste en un formulario con todos los datos necesarios.

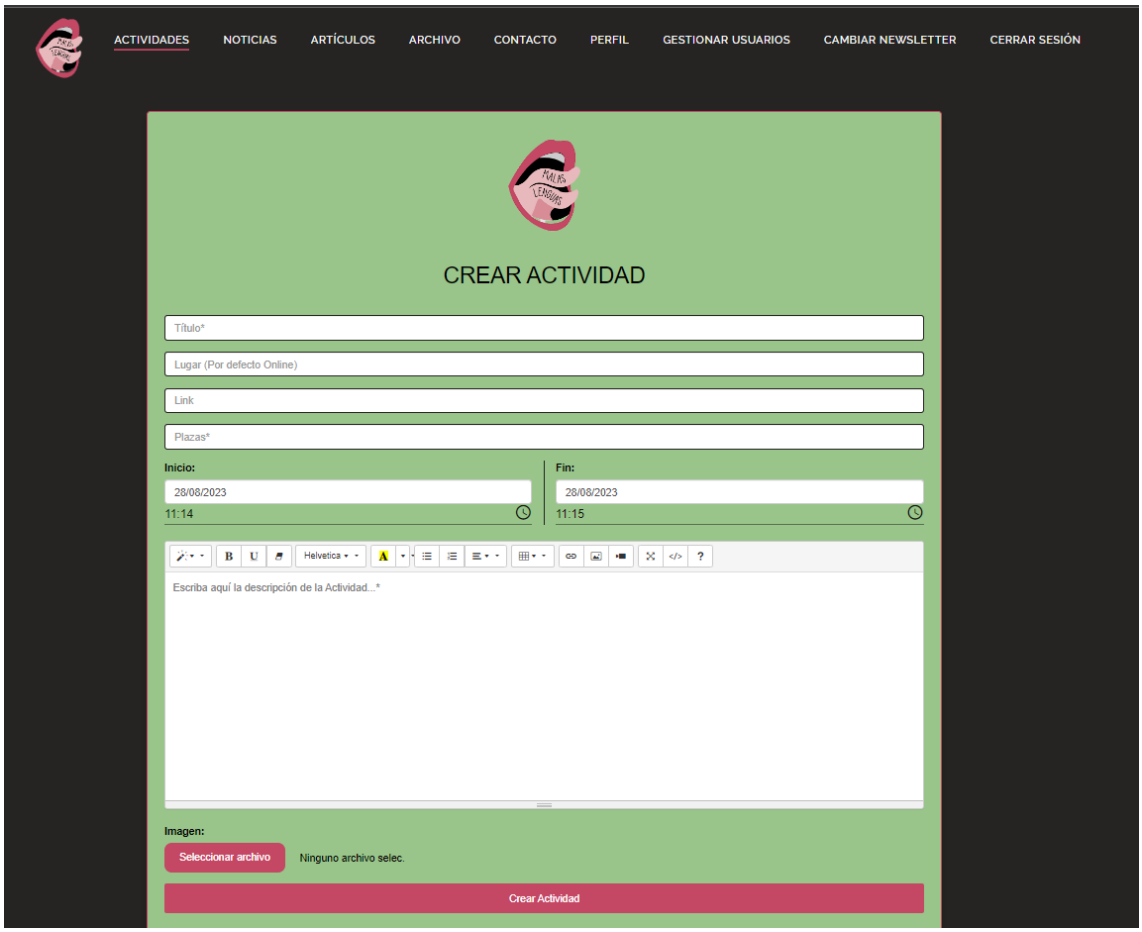

Figura 61: Formulario de creación de actividad

La vista se muestra por la función cargarCrearActividad()

```
54  def cargarCrearActividad(request):
55      context = {}
56      if 'idUsuarios' in request.session:
57          context['usuarioLogged'] = True
58
59      check = checkLogged(request)
60      if check:
61          return check
62
63      if request.session['idRol'] == 2:
64          messages.error(request,"No tienes permisos para realizar esta acción")
65          return HttpResponseRedirect('/Archivo')
66
67
68      return render(request, 'Actividades/crearActividad.html', context)
69
```

Figura 62: Función cargarCrearActividad

La función comprueba que el usuario tenga los roles necesarios para evitar que usuarios básicos accedan a esta página.

Las fechas se manejan mediante un datepicker y las horas con un timepicker. Ambos componentes obtenidos gracias a GIJGO [24]

```
$('#datepicker').datepicker({
    weekStart: 1,
    daysOfWeekHighlighted: "6,0",
    autoclose: true,
    todayHighlight: true,
});
$('#datepicker').datepicker("setDate", new Date());
```

Figura 63: Manejador del datepicker

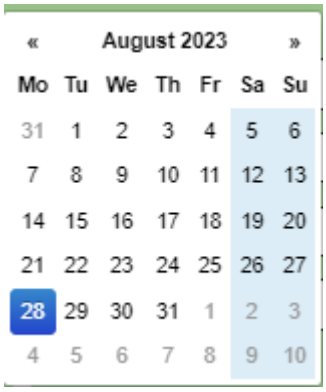

Figura 64: vista del datepicker

```
$('#timepicker').timepicker();
```

Figura 65: Manejador del timepicker

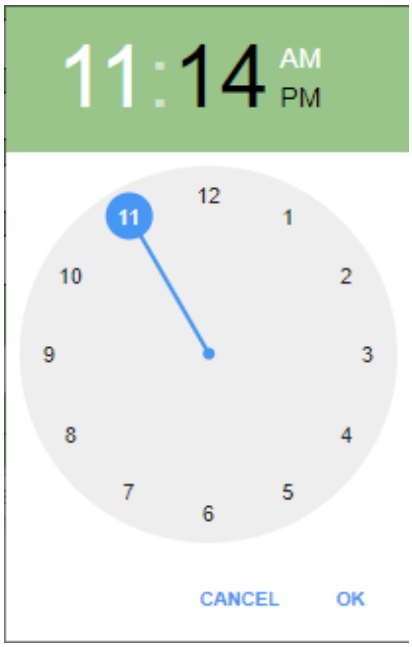

Figura 66: Vista del timepicker

La sección de la descripción de la actividad consiste en un campo de texto implementado con summernote [25]

```
$(document).ready(function () {
  $('.summernote').summernote({
    placeholder: 'Escriba aquí la descripción de la Actividad...',
    height: 300,
    tabDisable: true,
    disableDragAndDrop: true,
  });
});
```

Figura 67: Manejador del summernote

Una vez rellenos todos los datos el usuario pulsará el botón crear actividad. Este botón llama a la función subirActividad()

```
def subirActividad(request):

    check = checkLogged(request)
    if check:
        return check

    if request.session['idRol'] == 2:
        messages.error(request,"No tienes permisos para realizar esta acción")
        return HttpResponseRedirect('/Archivo')


    if 'idUsuarios' in request.session:
        usuarioId = request.session['idUsuarios']
        usuario = Usuario.objects.get(pk=usuarioId)
        nombreActividad = request.POST.get('titulo')
        descripcion = request.POST.get('descripcion')
        imagen = request.FILES.get('imagen')
        fechaInicio = request.POST.get('fechainicio')
        fechaFin = request.POST.get('fechafin')
        horaInicio = request.POST.get('horainicio')
        horaFin = request.POST.get('horafin')
        estadoId = Estado.objects.get(pk=1)
        plazas = request.POST.get('plazas')
        if  request.POST.get('lugar'):
            lugar = request.POST.get('lugar')
        else:
            lugar = "Online"
        enlace = request.POST.get('enlace')

        fechaCreacion = datetime.now()

        fechaInicioParsed = datetime.strptime(fechaInicio, '%d/%m/%Y')
        horaInicioParsed = datetime.strptime(horaInicio, '%H:%M')
        inicioParsed = datetime.combine(fechaInicioParsed.date(), horaInicioParsed.time())

        fechaFinParsed = datetime.strptime(fechaFin, '%d/%m/%Y')
        horaFinParsed = datetime.strptime(horaFin, '%H:%M')
        finParsed = datetime.combine(fechaFinParsed.date(), horaFinParsed.time())

        fechaInicio = inicioParsed.date()
        horaInicio = inicioParsed.time()

        fechaFin = finParsed.date()
        horaFin = finParsed.time()

        fechaInicioParseada = fechaInicio.strftime('%d/%m/%Y')
        fechaFinParseada = fechaFin.strftime('%d/%m/%Y')

        horaInicioParseada = horaInicio.strftime('%H:%M')
        horaFinParseada = horaFin.strftime('%H:%M')

        Actividad.objects.create(nombreActividad=nombreActividad, descripcion=descripcion, imagen=imagen, propietario=usuario, plazas=plazas,
                                 estadoId=estadoId, fechaInicio=inicioParsed, fechaFin=finParsed, fechaCreacion=fechaCreacion, horaInicioVisible=horaInicioParseada,
                                 horaFinVisible=horaFinParseada, fechaFinVisible=fechaFinParseada, fechaInicioVisible=fechaInicioParseada, enlace=enlace, lugar=lugar)
        messages.success(request, "La Actividad se ha subido de manera satisfactoria.")
        return HttpResponseRedirect('/Actividades')
    else:
        messages.error(request, "No ha iniciado sesión")
        return HttpResponseRedirect('/Actividades')
```

Figura 68: Función subirActividad

Esta función obtiene los datos introducidos en el formulario y crea un registro en la base de datos del tipo actividad. Las fechas tienen que almacenarse en el formato de MongoDB, por eso se realizan las conversiones.

5.2.3. abrirActividad.html

Es la vista que se muestra al entrar dentro de una actividad.

Figura 69: Vista dentro de una actividad (1)

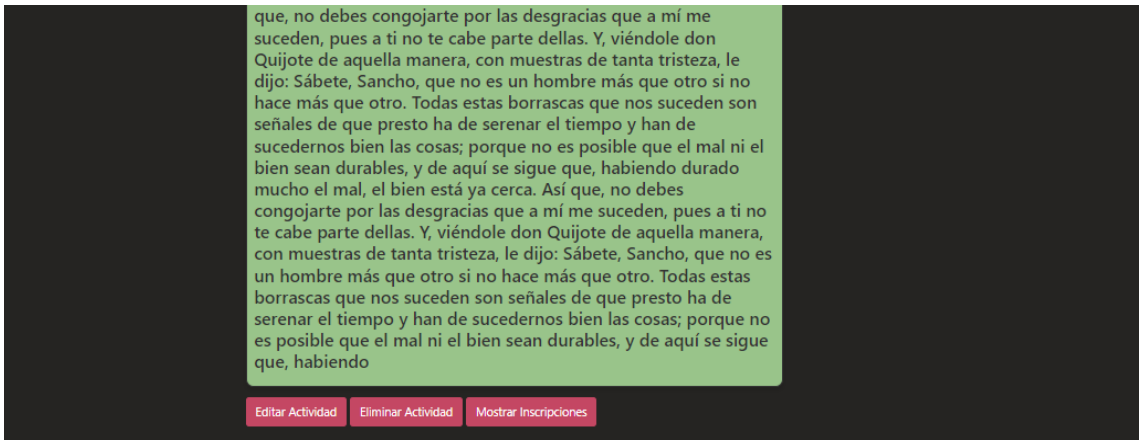

Figura 70: Vista dentro de una actividad (2)

Se muestra por la función abrirActividad(id)

```python
def abrirActividad(request, uidb64):
    id = int(urlsafe_base64_decode(uidb64))
    actividad = get_object_or_404(Actividad, id=id)

    fechaInicio = actividad.fechaInicio.date()
    horaInicio = actividad.fechaInicio.time()

    fechaFin = actividad.fechaFin.date()
    horaFin = actividad.fechaFin.time()

    fechaInicioParseada = fechaInicio.strftime('%d/%m/%Y')
    fechaFinParseada = fechaFin.strftime('%d/%m/%Y')

    horaInicioParseada = horaInicio.strftime('%H:%M')
    horaFinParseada = horaFin.strftime('%H:%M')

    inscritos = actividad.inscripciones.count()
    plazasFinal = actividad.plazas - inscritos

    fechaActual = datetime.now().date()

    puedeInscribirse = fechaInicio>fechaActual

    context = {
        'actividad': actividad,
        'fechaInicio': fechaInicioParseada,
        'fechaFin': fechaFinParseada,
        'horaInicio': horaInicioParseada,
        'horaFin': horaFinParseada,
        'plazas': plazasFinal,
        'fechaActual': fechaActual,
        'puedeInscribirse': puedeInscribirse,
    }

    if 'idUsuarios' in request.session:
        context['usuarioLogged'] = True
        usuarioId = request.session['idUsuarios']
        usuario = Usuario.objects.get(pk=usuarioId)
        rol = usuario.rolId
        context['rol'] = rol.id
        if Inscripcion.objects.filter(actividadId=actividad, usuarioId=usuario):
            context['inscrito'] = True
        else:
            context['inscrito'] = False

    return render(request, 'Actividades/abrirActividad.html', context)
```

Figura 71: Función abrirActividad

En esta función abre la vista abrirActividad.html e incluye en el contexto los datos de la actividad con el id pasado como parámetro, además, debido a que MongoDB almacena las fechas en el formato YYYY-MM-DD, tenemos que pasar por contexto las fechas formateadas en el formato español.

Dentro de esta vista podemos

- Editar la actividad
- Eliminar la actividad
- Inscribirse/cancelar inscripción
- Mostrar inscripciones

5.2.4. Eliminar actividad

llama a la función aliminarActividad(id)

```python
def eliminarActividad(request, uidb64):

    check = checkLogged(request)
    if check:
        return check

    if request.session['idRol'] == 2:
        messages.error(request,"No tienes permisos para realizar esta acción")
        return HttpResponseRedirect('/Archivo')

    id = int(urlsafe_base64_decode(uidb64))

    actividad = get_object_or_404(Actividad, id=id)
    if actividad.imagen:
        actividadPath = actividad.imagen.path
        if os.path.exists(actividadPath):
            os.remove(actividadPath)

    if actividad.fechaInicio >= timezone.now():
        inscripciones = Inscripcion.objects.filter(actividadId=actividad)
        thread = threading.Thread(target=enviarEmailAsincrono, args=(inscripciones, actividad))
        thread.start()
    actividad.delete()
    messages.success(request, "La Actividad se ha eliminado de manera satisfactoria.")
    return HttpResponseRedirect("/Actividades")
```

Figura 72: Función eliminarActividad

Esta función comprueba los roles del usuario que está con la sesión iniciada para después eliminar el registro. Además, envía un mensaje a todos los usuarios inscritos en la actividad para anunciar que la actividad se ha cancelado.

Este envío se realiza de forma asíncrona mediante threads para mejorar la experiencia del usuario reduciendo los tiempos de carga en la aplicación.

```python
def enviarEmailAsincrono(inscripciones, actividad):
    for inscripcion in inscripciones:
        usuario = inscripcion.usuarioId

        contenido = render_to_string('Newsletters/actividadEliminadaTemplate.html', {
            'usuario': usuario,
            'actividad': actividad
        })
        plain_message = strip_tags(contenido)
        subject = "Actividad cancelada"
        send_mail(
            subject,
            plain_message,
            settings.EMAIL_HOST_USER,
            [usuario.email],
            html_message = contenido
        )
```

Figura 73: Función enviarEmailAsincrono

5.2.5. Inscribirse

Llama a la función Inscribirse(id)

```python
def Inscribirse(request, uidb64):

    check = checkLogged(request)
    if check:
        return check

    id = int(urlsafe_base64_decode(uidb64))

    if 'idUsuarios' in request.session:
        usuarioId = request.session['idUsuarios']
        usuario = Usuario.objects.get(pk=usuarioId)
        actividad = get_object_or_404(Actividad, id=id)
        inscritos = actividad.inscripciones.count()
        plazasFinal = actividad.plazas - inscritos
        if plazasFinal <=0:
            messages.error(request, "La Actividad está llena")
            return HttpResponseRedirect("/Actividades/abrirActividad/" + str(uidb64))

        if Inscripcion.objects.filter(usuarioId=usuario, actividadId=actividad).exists():
            messages.error(request, "Ya estas inscrito en esta actividad")
            return HttpResponseRedirect("/Actividades/abrirActividad/" + str(uidb64))

        usuarioId = usuario
        actividadId = actividad
        Inscripcion.objects.create(usuarioId=usuario, actividadId=actividadId)
        messages.success(request, "La Inscripción se ha realizado de manera satisfactoria.")
        return HttpResponseRedirect("/Actividades/abrirActividad/" + str(uidb64))
    else:
        messages.error(request, "No ha iniciado sesión")
        return HttpResponseRedirect("/Actividades/abrirActividad/" + str(uidb64))
```

Figura 74: Función Inscribirse

Esta función obtiene el usuario mediante el id que se le pasa como parámetro y crea un registro de inscripción entre el usuario y la actividad. Antes de esto, comprueba que el usuario esté registrado y que haya suficientes plazas en la actividad.

5.2.6. Cancelar inscripción

Como contraparte existe la opción de cancelar inscripción. Este botón llama a la función CancelarInscripcion(idUsuario, idActividad)

```python
def CancelarInscripcion(request, uidb64, uidb642):

    check = checkLogged(request)
    if check:
        return check

    actividadId = int(urlsafe_base64_decode(uidb64))
    usuarioId = int(urlsafe_base64_decode(uidb642))

    if 'idUsuarios' in request.session:
        usuario = Usuario.objects.get(pk=usuarioId)
        actividad = get_object_or_404(Actividad, id=actividadId)
        inscripcion = Inscripcion.objects.filter(usuarioId=usuario, actividadId=actividad)
        thread = threading.Thread(target=enviarEmailInscripcionAsincrono, args=(inscripcion))
        thread.start()
        inscripcion.delete()
        if usuario.id == request.session['idUsuarios']:
            messages.success(request, "La inscripcion se ha eliminado de manera satisfactoria.")
            return HttpResponseRedirect("/Actividades/abrirActividad/" + str(uidb64))
        else:
            messages.success(request, "La inscripcion se ha eliminado de manera satisfactoria.")
            return HttpResponseRedirect("/Administracion/GestionUsuarios")
    else:
        messages.error(request, "No estás inscrito")
        return HttpResponseRedirect("/Actividades/abrirActividad/" + str(uidb64))

def enviarEmailInscripcionAsincrono(inscripcion):
    usuario = inscripcion.usuarioId
    actividad = inscripcion.actividadId

    contenido = render_to_string('Newsletters/InscripcionEliminadaTemplate.html', {
        'usuario': usuario,
        'actividad': actividad
    })
    plain_message = strip_tags(contenido)
    subject = "Actividad cancelada"
    send_mail(
        subject,
        plain_message,
        settings.EMAIL_HOST_USER,
        [usuario.email],
        html_message = contenido
    )
```

Figura 75: Función cancelarInscripcion y enviarEmaillInscripcionAsincrono

Esta función simplemente obtiene los datos de la actividad y el usuario mediante los id pasados como parámetro y borra el registro de la inscripción además de enviar un email al usuario cuya inscripción se ha cancelado.

5.2.7. Editar actividad

Este botón muestra la vista de edición de la actividad. Llama a la función EditarActividad(id)

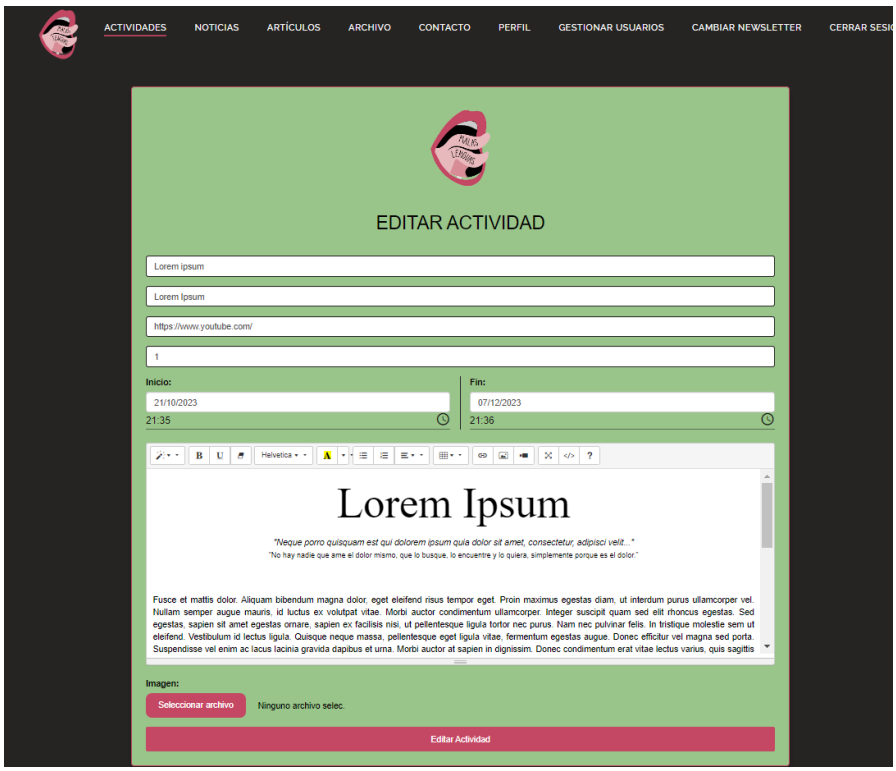

Figura 76: Vista de edición de la actividad

```python
def editarActividad(request, uidb64):

    check = checkLogged(request)
    if check:
        return check

    if request.session['idRol'] == 2:
        messages.error(request,"No tienes permisos para realizar esta acción")
        return HttpResponseRedirect('/Archivo')


    id = int(urlsafe_base64_decode(uidb64))
    actividad = get_object_or_404(Actividad, id=id)

    fechaInicio = actividad.fechaInicio.date()
    horaInicio = actividad.fechaInicio.time()

    fechaFin = actividad.fechaFin.date()
    horaFin = actividad.fechaFin.time()

    fechaInicioParseada = fechaInicio.strftime('%d/%m/%Y')
    fechaFinParseada = fechaFin.strftime('%d/%m/%Y')

    horaInicioParseada = horaInicio.strftime('%H:%M')
    horaFinParseada = horaFin.strftime('%H:%M')


    context = {
        'actividad': actividad,
        'fechaInicio': fechaInicioParseada,
        'fechaFin': fechaFinParseada,
        'horaInicio': horaInicioParseada,
        'horaFin': horaFinParseada
    }
    if 'idUsuarios' in request.session:
        context['usuarioLogged'] = True

    return render(request, 'Actividades/editarActividad.html', context)

```

Figura 77: Función abrirActividad

Esta función al igual que abrirActividad(id) obtiene los datos de la actividad y abre el formulario de edición con los valores de la actividad pasados por contexto.

Una vez se cambian los datos deseados el usuario pulsará el botón Editar Actividad. Este botón llama a la función guardarEditarActividad(id)

```python
def guardarEditarActividad(request, uidb64):

    check = checkLogged(request)
    if check:
        return check

    if request.session['idRol'] == 2:
        messages.error(request,"No tienes permisos para realizar esta acción")
        return HttpResponseRedirect('/Archivo')

    id = int(urlsafe_base64_decode(uidb64))

    actividad = get_object_or_404(Actividad, id=id)
    nombreActividad = request.POST.get('titulo')
    descripcion = request.POST.get('descripcion')
    imagen = request.FILES.get('imagen')
    fechaInicio = request.POST.get('fechainicio')
    fechaFin = request.POST.get('fechafin')
    horaInicio = request.POST.get('horainicio')
    horaFin = request.POST.get('horafin')
    plazas = request.POST.get('plazas')
    lugar = request.POST.get('lugar')
    if imagen:
        actividad.imagen = imagen

    fechaInicioParsed = datetime.strptime(fechaInicio, '%d/%m/%Y')
    horaInicioParsed = datetime.strptime(horaInicio, '%H:%M')
    inicioParsed = datetime.combine(fechaInicioParsed.date(), horaInicioParsed.time())

    fechaFinParsed = datetime.strptime(fechaFin, '%d/%m/%Y')
    horaFinParsed = datetime.strptime(horaFin, '%H:%M')
    finParsed = datetime.combine(fechaFinParsed.date(), horaFinParsed.time())

    fechaInicio = inicioParsed.date()
    horaInicio = inicioParsed.time()

    fechaFin = finParsed.date()
    horaFin = finParsed.time()

    fechaInicioParseada = fechaInicio.strftime('%d/%m/%Y')
    fechaFinParseada = fechaFin.strftime('%d/%m/%Y')

    horaInicioParseada = horaInicio.strftime('%H:%M')
    horaFinParseada = horaFin.strftime('%H:%M')

    actividad.nombreActividad = nombreActividad
    actividad.descripcion = descripcion
    actividad.fechaInicio = inicioParsed
    actividad.fechaFin = finParsed
    actividad.plazas = plazas
    actividad.fechaInicioVisible = fechaInicioParseada
    actividad.horaInicioVisible = horaInicioParseada
    actividad.fechaFinVisible = fechaFinParseada
    actividad.horaFinVisible = horaFinParseada
    actividad.lugar = lugar
    if request.POST.get('enlace'):
        actividad.enlace = request.POST.get('enlace')

    actividad.save()
    messages.success(request, "La Actividad se ha editado de manera satisfactoria.")
    return HttpResponseRedirect("/Actividades/abrirActividad/" + str(uidb64))

```

Figura 78: función guardarEditarActividad

Esta función comprueba que el usuario con la sesión iniciada tenga los roles necesarios para realizar esta función. Después guarda los cambios en la actividad con el id pasado como parámetro.

5.2.8. Mostrar inscripciones

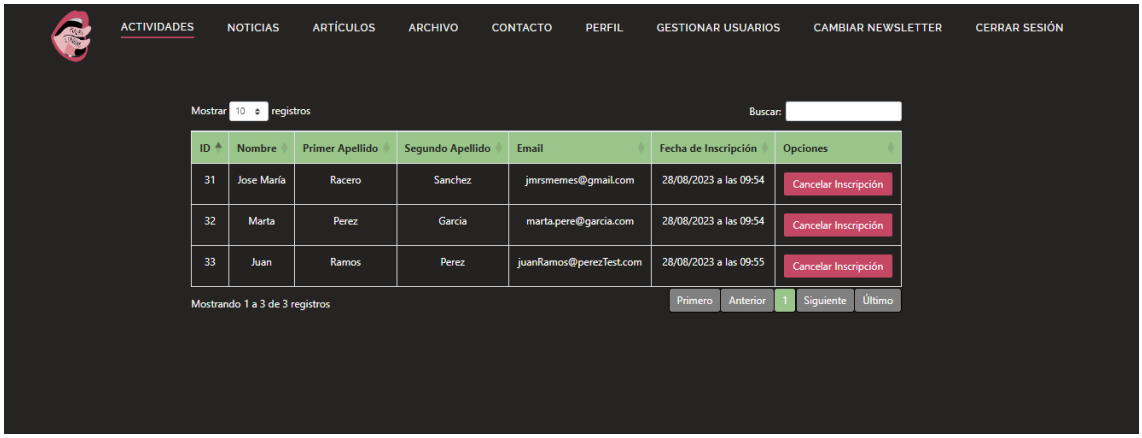

Figura 79: Vista de mostrar inscripciones

Esta vista es la encargada de ver y gestionar las inscripciones de la actividad.

Esta vista se muestra por la función MostrarInscripciones(id)

```
369  def MostrarInscripciones(request, uidb64):
370      context = {}
371      id = int(urlsafe_base64_decode(uidb64))
372      check = checkLogged(request)
373      if check:
374          return check
375
376      if request.session['idRol'] and request.session['idRol'] == 2:
377          messages.error(request,"No tienes permisos para realizar esta acción")
378          return HttpResponseRedirect('/Archivo')
379
380      if 'idUsuarios' in request.session:
381          context['usuarioLogged'] = True
382
383      actividad = Actividad.objects.get(id=id)
384      inscripciones = actividad.inscripcion_set.all()
385      context['inscripciones'] = inscripciones
386      return render(request, "Actividades/inscripcionesActividad.html", context)
```

Figura 80: Función MostrarInscripciones

Esta función comprueba el rol del usuario y abre la vista con las inscripciones de la actividad pasada por parámetro dentro del contexto.

5.3. Casos de prueba

En este apartado se describen algunas de las diferentes pruebas realizadas para los diferentes casos de uso de la aplicación

ID	PU-001
Nombre	Iniciar sesión
Descripción	Comprueba si el sistema comprueba correctamente que el usuario existe en la base de datos.
Componentes afectados	HomeView.py, LoginView.py, login.html, home.html, Usuario
Comportamiento esperado	El usuario rellena los campos de *Email* y de *Contraseña*. Posteriormente pulsa el botón de *iniciar sesión* y el controlador debe verificar que el usuario existe para luego iniciar sesión.
Comportamiento obtenido	Al pulsar el botón de *iniciar sesión* devuelve el resultado correcto y redirecciona a la página principal con la sesión iniciada.
Corrección realizada	Ninguna
Fecha	29/08/2023
Responsable	José María Racero

Tabla 21: Caso de prueba 1

ID	PU-002
Nombre	Registro de usuario.

Descripción	Comprueba si el sistema almacena correctamente en la base de datos un nuevo usuario.
Componentes afectados	HomeView.py, LoginView.py, RegistroView.py, home.html, registro.html, login.html, Usuario
Comportamiento esperado	El usuario pulsa el enlace de *inscríbete*. El sistema debe redireccionar al formulario de registro. El usuario rellena los campos obligatorios. Posteriormente pulsará el botón de *Registrarse*. El controlador debe comprobar que la longitud de la contraseña es de como mínimo 8 caracteres. Luego debe de almacenar la contraseña hasheada y crear el registro de usuario. Después debe redireccionar al formulario de inicio de sesión junto con un mensaje.
Comportamiento obtenido	Al pulsar el botón de *Registrarse* el sistema almacena el registro de usuario de manera correcta. Posteriormente redirecciona con un mensaje de éxito a la página de inicio de sesión.

Corrección realizada	Aumento de la seguridad almacenando la contraseña hasheada con Django
Fecha	29/08/2023
Responsable	José María Racero

Tabla 22: Caso de prueba 2

ID	PU-003
Nombre	CRUD Actividades – Crear
Descripción	Comprueba si el sistema almacena correctamente el registro del tipo Actividad en la base de datos.
Componentes afectados	HomeView.py, ActividadesView.py, home.html, actividades.html, crearActividad.html, Actividad
Comportamiento esperado	El usuario debe de pulsar el botón *Crear Actividad* dentro de la pestaña de actividades. El sistema debe mostrarle el formulario de creación de actividad. El usuario debe de rellenar los campos obligatorios y luego pulsar el botón *Crear Actividad*. Una vez hecho esto el sistema deberá crear un registro del tipo actividad en la base de datos y luego redireccionar a la página de *Actividades* con un mensaje de éxito.

Comportamiento obtenido	Al pulsar el botón de *Crear Actividad* el sistema muestra la vista con el formulario de creación de actividad. Al rellenar los datos y pulsar el botón *Crear Actividad* el sistema almacena los datos de la actividad de manera correcta. Redirecciona a la página de *Actividades* y muestra un mensaje de éxito.
Corrección realizada	Añadidos los componentes Timepicker y Datepicker de Gijgo para mejorar la experiencia de usuario
Fecha	29/08/2023
Responsable	José María Racero

Tabla 23: Caso de prueba 3

ID	PU-004
Nombre	CRUD Actividades – Leer
Descripción	Comprueba que el sistema muestra la actividad correcta al pulsar en ella.
Componentes afectados	HomeView.py, ActividadesView.py, home.html, actividades.html, abrirActividad.html, Actividad
Comportamiento esperado	El Usuario debe de pulsar la actividad. El sistema debe de obtener la actividad mediante el id de la misma para llamar

	a la función que muestra la vista de la actividad buscada.
Comportamiento obtenido	El Usuario al pulsar en cualquier actividad siempre obtiene el contenido de esta.
Corrección realizada	Añadido mapa que indica el lugar donde se va a realizar la actividad para mejorar la experiencia de usuario.
Fecha	29/08/2023
Responsable	José María Racero

Tabla 24: Caso de prueba 4

ID	PU-005
Nombre	CRUD Actividades – Editar
Descripción	Comprueba que el sistema almacena todos los cambios en el registro seleccionado.
Componentes afectados	HomeView.py, ActividadesView.py, home.html, actividades.html, abrirActividad.html, editarActividad.html, Actividad
Comportamiento esperado	El Usuario debe de entrar en la pestaña de *Actividades*. Una vez dentro debe de elegir la actividad que desea editar y pulsar en ella. El usuario una vez pulse en la actividad debe de pulsar el botón *Editar Actividad*. El

	sistema debe de mostrarle el formulario de edición donde modificará los campos que desea editar. Una vez haya editado el contenido, pulsará el botón de *Editar Actividad*. El sistema almacenará los datos modificados en la actividad seleccionada. Redireccionará a la actividad y mostrará un mensaje de éxito
Comportamiento obtenido	Al pulsar *Editar Actividad* el sistema muestra el formulario de edición. Una vez modificados los datos. Al pulsar el botón *Editar Actividad*. El sistema almacena los cambios, redirecciona y muestra el mensaje como debería.
Corrección realizada	Ninguna
Fecha	29/08/2023
Responsable	José María Racero

Tabla 25: Caso de prueba 5

ID	PU-006
Nombre	CRUD Actividades – Borrar
Descripción	Comprueba que el sistema elimina de la base de datos el registro seleccionado.

Componentes afectados	HomeView.py, ActividadesView.py, home.html, actividades.html, abrirActividad.html, Actividad
Comportamiento esperado	El usuario Entra en la pestaña *Actividades*. Pulsará en la actividad que desee borrar. El sistema le mostrará la actividad seleccionada. El usuario pulsará el botón *Eliminar Actividad*. El sistema obtendrá el registro de esa actividad y procederá a eliminarlo de la base de datos. El sistema redireccionará a la pestaña de *Actividades* y mostrará un mensaje de éxito
Comportamiento obtenido	al pulsar el botón de *Eliminar Actividad* el sistema elimina el registro de dicha actividad y redirecciona junto con el mensaje.
Corrección realizada	Ninguna
Fecha	29/08/2023
Responsable	José María Racero

Tabla 26: Caso de prueba 6

ID	PU-007
Nombre	Inscribirse Actividad

Descripción	Comprueba que se crea un registro de inscripción entre el usuario y la actividad en la base de datos.
Componentes afectados	HomeView.py, ActividadesView.py, home.html, actividades.html, abrirActividad.html, Actividad, Usuario, inscripción
Comportamiento esperado	El usuario entra en la actividad en la que desea inscribirse. Una vez dentro, pulsa el botón de *Inscribirse*. Al hacerlo, el sistema crea una relación entre el usuario y la actividad. Redirecciona a la página de la actividad y muestra un mensaje de éxito.
Comportamiento obtenido	Al pulsar el botón *Inscribirse*. El sistema crea la relación entre actividad y usuario y el usuario es redireccionado a la página de la actividad y le muestra un mensaje de éxito
Corrección realizada	Ninguna
Fecha	29/08/2023
Responsable	José María Racero

Tabla 27: Caso de prueba 7

ID	PU-008
Nombre	Cancelar Inscripción

Descripción	Comprueba que se elimina el registro de inscripción entre el usuario y la actividad en la base de datos.
Componentes afectados	HomeView.py, ActividadesView.py, home.html, actividades.html, abrirActividad.html, Actividad, Usuario, inscripción
Comportamiento esperado	El usuario entra en la actividad en la que desea cancelar la inscripción. Una vez dentro, pulsa el botón de *Cancelar Inscripción*. Al hacerlo, el sistema elimina la relación existente. Redirecciona a la página de la actividad y muestra un mensaje de éxito.
Comportamiento obtenido	Al pulsar el botón *Cancelar Inscripción*. El sistema elimina la relación entre actividad y usuario. El usuario es redireccionado a la página de la actividad y le muestra un mensaje de éxito
Corrección realizada	Ninguna
Fecha	29/08/2023
Responsable	José María Racero

Tabla 28: Caso de prueba 8

ID	PU-009

Nombre	CRUD Noticias – Crear
Descripción	Comprueba si el sistema almacena correctamente el registro del tipo Noticia en la base de datos.
Componentes afectados	HomeView.py, NoticiasView.py, home.html, noticias.html, crearNoticia.html, Noticia
Comportamiento esperado	El usuario debe de pulsar el botón *Crear Noticia* dentro de la pestaña de noticias. El sistema debe mostrarle el formulario de creación de noticia. El usuario debe de rellenar los campos obligatorios y luego pulsar el botón *Crear Noticia*. Una vez hecho esto el sistema deberá crear un registro del tipo noticia en la base de datos y luego redireccionar a la página de *Noticias* con un mensaje de éxito.
Comportamiento obtenido	Al pulsar el botón de *Crear Noticia* el sistema muestra la vista con el formulario de creación de noticia. Al rellenar los datos y pulsar el botón *Crear Noticia* el sistema almacena los datos de la noticia de manera correcta. Redirecciona a la página de *Noticias* y muestra un mensaje de éxito.

Corrección realizada	Ninguna
Fecha	29/08/2023
Responsable	José María Racero

Tabla 29: Caso de prueba 9

ID	PU-010
Nombre	CRUD Noticias – Leer
Descripción	Comprueba que el sistema muestra la noticia correcta al pulsar en ella.
Componentes afectados	HomeView.py, NoticiasView.py, home.html, noticias.html, abrirNoticia.html, Noticia
Comportamiento esperado	El Usuario debe de pulsar la noticia. El sistema debe de obtener la noticia mediante el id de la misma para llamar a la función que muestra la vista de la noticia buscada.
Comportamiento obtenido	El Usuario al pulsar en cualquier noticia siempre obtiene el contenido de esta.
Corrección realizada	Ninguna
Fecha	29/08/2023
Responsable	José María Racero

Tabla 30: Caso de prueba 10

ID	PU-011
Nombre	CRUD Noticias – Editar

Descripción	Comprueba que el sistema almacena todos los cambios en el registro seleccionado.
Componentes afectados	HomeView.py, NoticiasView.py, home.html, noticias.html, abrirNoticia.html, editarNoticia.html, Noticia
Comportamiento esperado	El Usuario debe de entrar en la pestaña de *Noticias*. Una vez dentro debe de elegir la noticia que desea editar y pulsar en ella. El usuario una vez pulse en la noticia debe de pulsar el botón *Editar Noticia*. El sistema debe de mostrarle el formulario de edición donde modificará los campos que desea editar. Una vez haya editado el contenido, pulsará el botón de *Editar Noticia*. El sistema almacenará los datos modificados en la noticia seleccionada. Redireccionará a la noticia y mostrará un mensaje de éxito
Comportamiento obtenido	Al pulsar *Editar Noticia* el sistema muestra el formulario de edición. Una vez modificados los datos. Al pulsar el botón *Editar Noticia*. El sistema

	almacena los cambios, redirecciona y muestra el mensaje como debería.
Corrección realizada	Ninguna
Fecha	29/08/2023
Responsable	José María Racero

Tabla 31: Caso de prueba 11

ID	PU-012
Nombre	CRUD Noticias – Borrar
Descripción	Comprueba que el sistema elimina de la base de datos el registro seleccionado.
Componentes afectados	HomeView.py, NoticiasView.py, home.html, noticias.html, abrirNoticia.html, Noticia
Comportamiento esperado	El usuario Entra en la pestaña *Noticias*. Pulsará en la noticia que desee borrar. El sistema le mostrará la noticia seleccionada. El usuario pulsará el botón *Eliminar Noticia*. El sistema obtendrá el registro de esa noticia y procederá a eliminarlo de la base de datos. El sistema redireccionará a la pestaña de *Noticias* y mostrará un mensaje de éxito
Comportamiento obtenido	al pulsar el botón de *Eliminar Noticia* el sistema elimina el registro de dicha

	noticia y redirecciona junto con el mensaje.
Corrección realizada	Ninguna
Fecha	29/08/2023
Responsable	José María Racero

Tabla 32: Caso de prueba 12

ID	PU-013
Nombre	Editar datos personales
Descripción	Comprueba que el sistema guarda los cambios realizados en el registro de usuario.
Componentes afectados	HomeView.py, Administracion.py, home.html, perfil.html, Administracion.html, Usuario
Comportamiento esperado	El Usuario debe de entrar en la pestaña de *Perfil*. Una vez dentro, el usuario debe de pulsar el botón *Editar Perfil/Configuración.* El sistema debe mostrar la vista de *Configuración.* En la pestaña de *Datos personales,* el usuario introducirá todos los valores que desee modificar. Una vez los introduzca pulsará el botón de Cambiar datos. El sistema cambiará los campos en el registro del usuario y

	redireccionará al perfil del usuario con un mensaje de éxito.
Comportamiento obtenido	Al pulsar *Cambiar datos* el sistema almacena los cambios, redirecciona y muestra el mensaje como debería.
Corrección realizada	Ninguna
Fecha	29/08/2023
Responsable	José María Racero

Tabla 33: Caso de prueba 13

ID	PU-014
Nombre	Cambiar contraseña
Descripción	Comprueba que el sistema guarda los cambios realizados en el registro de usuario.
Componentes afectados	HomeView.py, Administracion.py, home.html, perfil.html, Administracion.html, Usuario
Comportamiento esperado	El Usuario debe de entrar en la pestaña de *Perfil*. Una vez dentro, el usuario debe de pulsar el botón *Editar Perfil/Configuración.* El sistema debe mostrar la vista de *Configuración.* En la pestaña de *Seguridad,* el usuario introducirá su contraseña antigua y la nueva que quiera poner. Una vez los introduzca pulsará el botón de

	Cambiar Contraseña. El sistema cambiará el campo en el registro del usuario y redireccionará a la pestaña de *Configuración* del usuario con un mensaje de éxito.
Comportamiento obtenido	Al pulsar *Cambiar contraseña* el sistema almacena los cambios, redirecciona y muestra el mensaje como debería.
Corrección realizada	Oculto para administradores y usuarios avanzados el campo *contraseña antigua* para agilizar el proceso de cambio de contraseña
Fecha	29/08/2023
Responsable	José María Racero

Tabla 34: Caso de prueba 14

ID	PU-015
Nombre	Eliminar Usuario
Descripción	Comprueba que el sistema elimina el registro de usuario.
Componentes afectados	HomeView.py, Administracion.py, home.html, perfil.html, Administracion.html, Usuario
Comportamiento esperado	El Usuario debe de entrar en la pestaña de *Perfil*. Una vez dentro, el usuario debe de pulsar el botón *Editar*

	Perfil/Configuración. El sistema debe mostrar la vista de *Configuración.* En la pestaña de *General,* el pulsará el botón de *Desactivar cuenta*. Una vez pulse el botón, el sistema eliminará el registro del usuario y redireccionará a la pestaña de *Configuración* del usuario con un mensaje de éxito.
Comportamiento obtenido	Al pulsar *Desactivar cuenta* el sistema elimina el registro, redirecciona y muestra el mensaje como debería.
Corrección realizada	Añadida ventana modal de aviso.
Fecha	29/08/2023
Responsable	José María Racero

Tabla 35: Caso de prueba 15

ID	PU-016
Nombre	Enviar newsletter
Descripción	Comprueba que el sistema envía los correos de newsletters a los usuarios
Componentes afectados	tasks.py, Celery.py, Usuario, Actividad, Noticia, Articulo
Comportamiento esperado	El sistema detecta cada 30 días que hay que enviar la newsletter mensual. El sistema ejecuta la función de mensajería que envía a todos los usuarios con la newsletter aceptada un

	correo electrónico con las ultimas noticias, actividades y artículos disponibles.
Comportamiento obtenido	El sistema ejecuta la tarea y envía los correos a los usuarios con el campo a True
Corrección realizada	Ninguna
Fecha	29/08/2023
Responsable	José María Racero

Tabla 36: Caso de prueba 16

6

Conclusiones

Para este proyecto se han empleado muchas tecnologías diferentes que no he tenido la oportunidad de utilizar durante el transcurso de mi formación académica.

He aprendido que analizar y diseñar una aplicación es una tarea muy difícil a la cual hay que echarle muchas horas. Además de un proceso muy importante puesto que, sin ese análisis inicial, empezar a crear la aplicación se volvería un caos.

La elaboración del mismo ha supuesto un esfuerzo personal, una forma de aprender nuevas tecnologías y una forma de implementar todos los conocimientos obtenidos a lo largo del grado. Puede que la universidad no me haya enseñado a utilizar ciertos lenguajes de programación, pero si me ha enseñado una base que puedo utilizar para aprender cualquier lenguaje.

La idea de hacer este proyecto surgió ante la iniciativa de querer aprender a desarrollar una aplicación web, puesto que nunca había creado ninguna y sentía la necesidad de aprender.

Me decidí por Django para forzarme a aprender a programar en Python, un lenguaje muy extendido del que apenas sabía nada.

Tampoco sabía utilizar HTML y CSS, por lo que aprender ha sido una experiencia muy dura pero gratificante.

Los requisitos propuestos se han cumplido y aún hay mucho margen de mejora para aumentar las funcionalidades de la aplicación en caso de que en un futuro se quisiera.

6.1. Posibles mejoras

Como posibles mejoras a implementar estarían las siguientes:

- Desplegado en la nube de la aplicación
- Incluir un sistema de comentarios y valoraciones tanto a las actividades como a las noticias.
- Crear un sistema de mensajería interno dentro de la aplicación
- La inclusión de sistemas de pago en la aplicación para agilizar el proceso de las actividades
- Introducir un editor de plantillas de newsletter dentro de la propia aplicación
- Introducir un motor de búsqueda para las imágenes y videos archivados
- Mejorar la interfaz para hacerla más dinámica
- Añadir un histórico de actividades eliminadas

Bibliografía

[1] Malas Lenguas Disdencias Estéticas (s.f.). El próximo 24 de mayo a las 19:30 tendremoa nuestra Primera Conferencia #5 a manos de Leon [Post]. Facebook. https://www.facebook.com/MalasLenguasDE/

[2] W3techs (s.f.). Site Info - Cacmalaga.eu. https://w3techs.com/sites/info/cacmalaga.eu

[3] Centro de Arte Contemporáneo de Málaga. (4 de octubre de 2021). *Transparencia-Misión*. https://cacmalaga.eu/transparencia/

[4] W3techs (s.f.). *Site Info - Alianzafrancesamalaga.es*. https://w3techs.com/sites/info/alianzafrancesamalaga.es

[5] Alianza Francesa de Málaga (s.f.). *La primera red cultural mundial, con 832 centros en 132 países en los cinco continentes*. https://www.alianzafrancesamalaga.es/academia/que-es-alianza-francesa/#/

[6] Bécares, B. (26 de enero 2022). *Fever es el nuevo unicornio español tras adaptarse a los eventos online de pandemia: predecir el ocio que queremos funciona*. https://www.genbeta.com/actualidad/fever-nuevo-unicornio-espanol-adaptarse-a-eventos-online-pandemia-predecir-ocio-que-queremos-funciona

[7] W3techs (s.f.). *Site Info- Sevillasecreta.co*. https://w3techs.com/sites/info/sevillasecreta.co

[8] Sevilla Secreta (s.f.). *Quiénes somos*. https://sevillasecreta.co/quienes-somos/

[9] Sistemas (s.f.). *Definición Casos de uso*. https://sistemas.com/casos-de-uso.php

[10] Diagramas UML (s.f.) *Diagramas de clases*. https://diagramasuml.com/diagrama-de-clases/?utm_content=cmp-true

[11] Visual Paradigm (s.f.). *What is Sequence Diagram?.* https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/

[12] Django (s.f.) Meet Django. https://www.djangoproject.com/

[13] Mdn Webdocs (24 de julio 2023) *JavaScript.* https://developer.mozilla.org/es/docs/Web/JavaScript

[14] Manz.dev (s.f.). *Introducción a Html5.* https://lenguajehtml.com/html/

[15] Arimetrics (s.f.) *Qué es Bootstrap. https://www.arimetrics.com/glosario-digital/Bootstrap*

[16] MongoDB (s.f.) *¿Qué es MongoDB?.* https://www.mongodb.com/es/what-is-mongodb

[17] Visual Paradigm (s.f.) *Visual Paradigm Product Overview.* https://www.visual-paradigm.com/support/documents/vpuserguide/12/13/5963_visualparadi.html

[18] DbDiagram (s.f.). *Draw Entity-Relationship Diagrams, Painlessly.* https://dbdiagram.io/home

[19] Taiga (s.f.). *Taiga: la herramienta de gestión de proyectos gratuita y open source.* https://taiga.io/es#About

[20] RabbitMQ (s.f.). *RabbitMQ is the most widely deployed open source message broker.* https://www.rabbitmq.com/

[21] Celery (s.f.). *Celery - Distributed Task Queue.* https://docs.celeryq.dev/en/stable/

[22] Django(s.f.) *Documentation- Models.* https://docs.djangoproject.com/en/4.2/topics/db/models/

[23] Django(s.f.). *Documentation- Pagination.* https://docs.djangoproject.com/en/4.2/topics/pagination/

[24] Gijgo (s.f.). Gijgo 1.9.14. *https://gijgo.com/*

[25] Summernote. *Super Simple WYSIWYG Editor on Bootstrap.* https://summernote.org/

[26] Anaconda (s.f.). *Anaconda Distribution.* https://www.anaconda.com/download

[27] MongoDB (s.f.). *Try MongoDB Community Edition.* https://www.mongodb.com/try/download/community

[28] RabbitMQ (s.f.). Installing on Windows. https://www.rabbitmq.com/install-windows.html#installer

[29] Erlang (s.f.). OTP Versions Tree. https://erlang.org/download/otp_versions_tree.html

Apéndice A.

Manual de Implantación

Para la instalación de este proyecto se necesita las siguientes herramientas

- Anaconda
- RabbitMQ
- Erlang
- MongoDB

Lo primero de todo es descargar Anaconda. Esto lo haremos desde la propia página Web. [26]

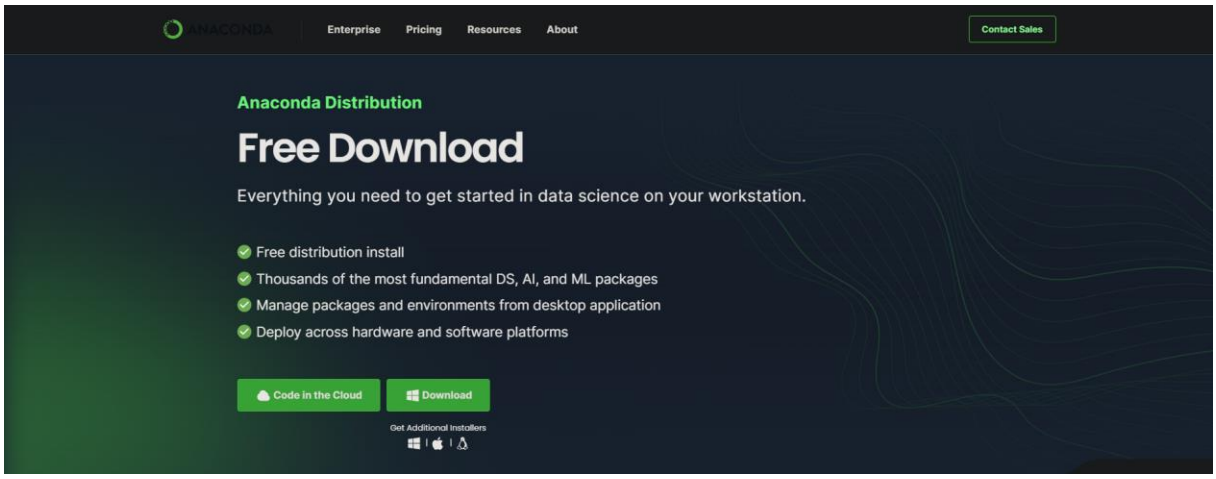

Figura Anexo 1: Página de descarga de Anaconda

Una vez descargado abrimos el ejecutable y aceptemos la licencia nos mostrará la siguiente pantalla.

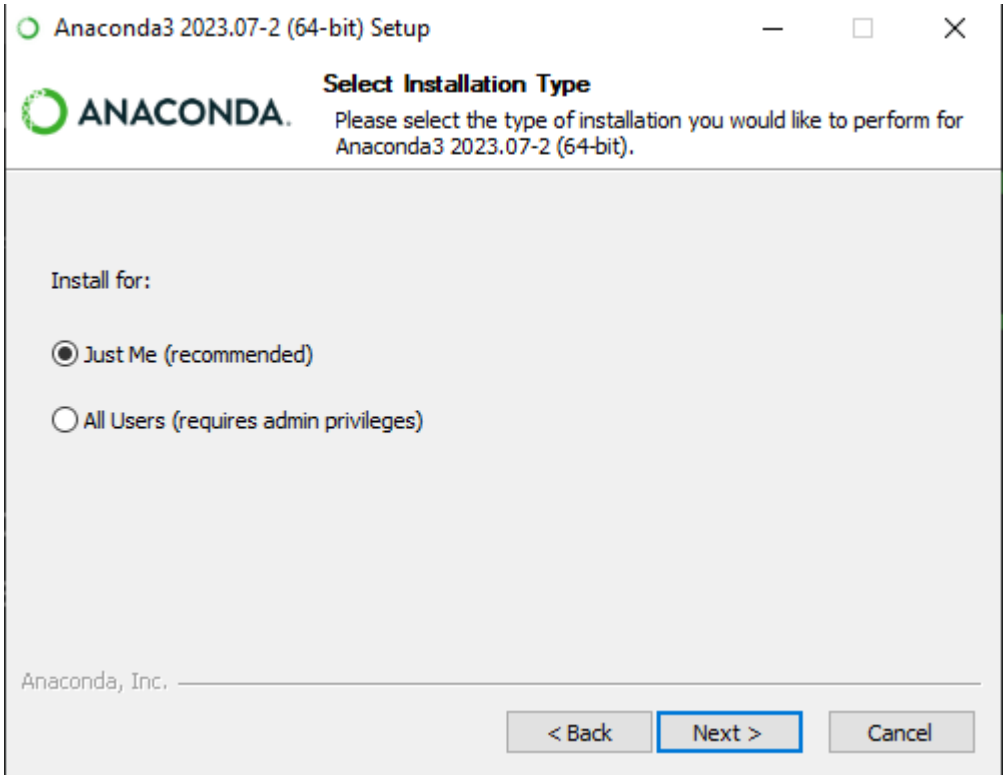

Figura Anexo 2: instalación anaconda (1)

Elegimos just me y pulsamos siguiente

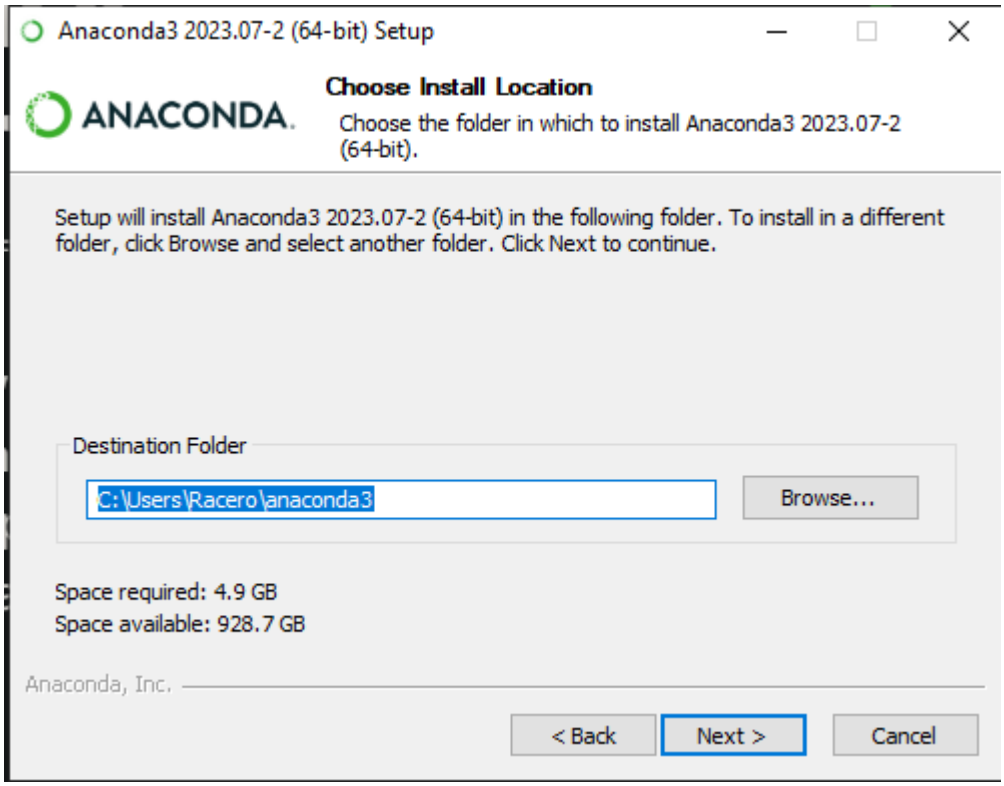

Figura Anexo 3: Instalación anaconda (2)

Elige la carpeta de destino y pulse siguiente sin cambiar ninguna de las opciones posteriores

Una vez instalado el programa anaconda procederá a instalar MongoDB.

Primero entre en la web de MongoDB [27] y pulse el botón verde de Download.

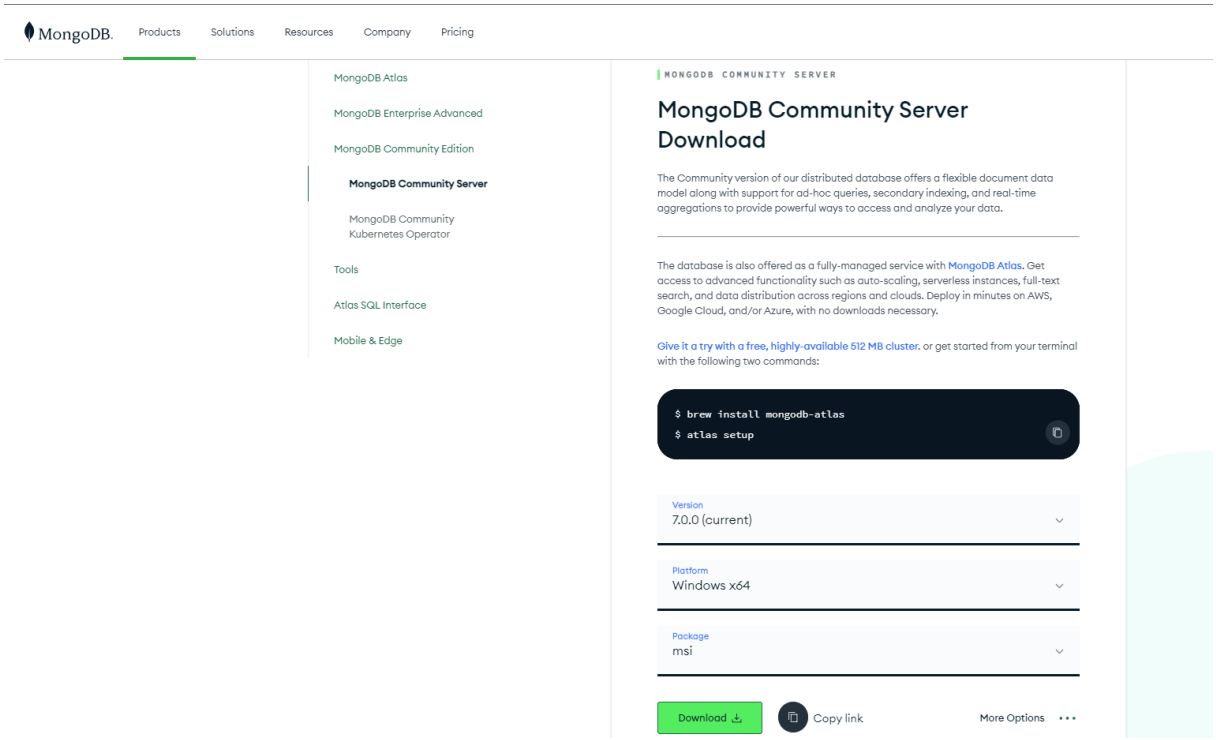

Figura Anexo 4: descarga MongoDB

Una vez descargado abrimos el ejecutable y aceptamos los términos y la licencia. Una vez hecho esto pulsamos en complete

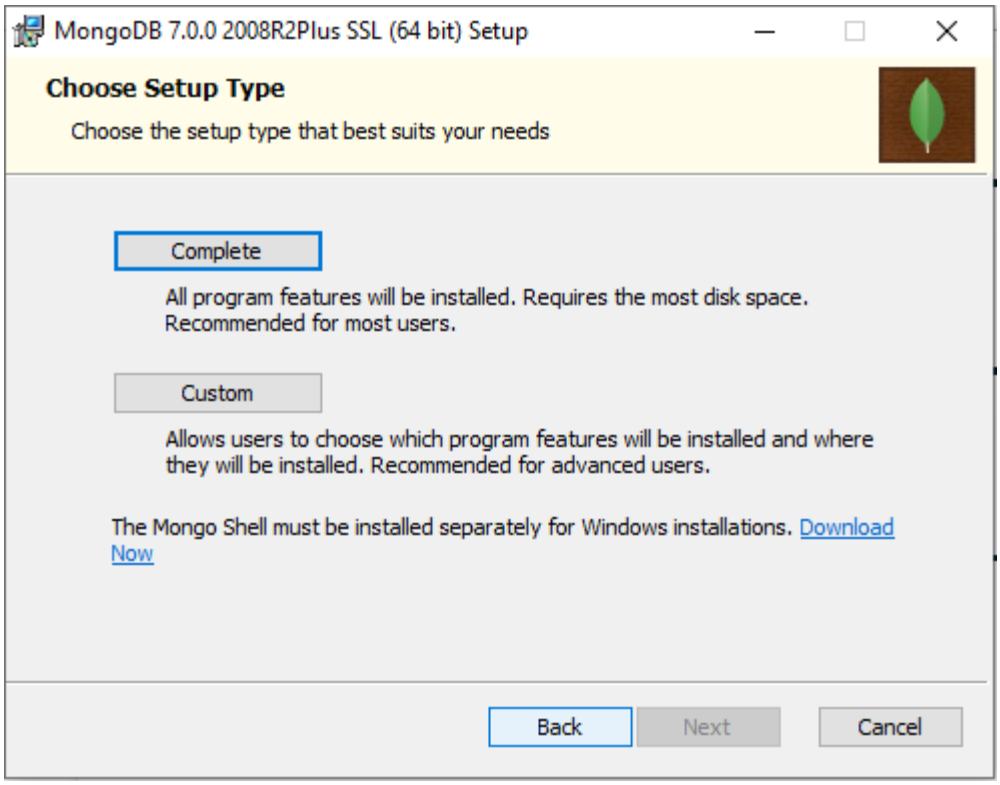

Figura Anexo 5: Instalación MongoDB

en la siguiente pantalla pulsamos next con las siguientes opciones

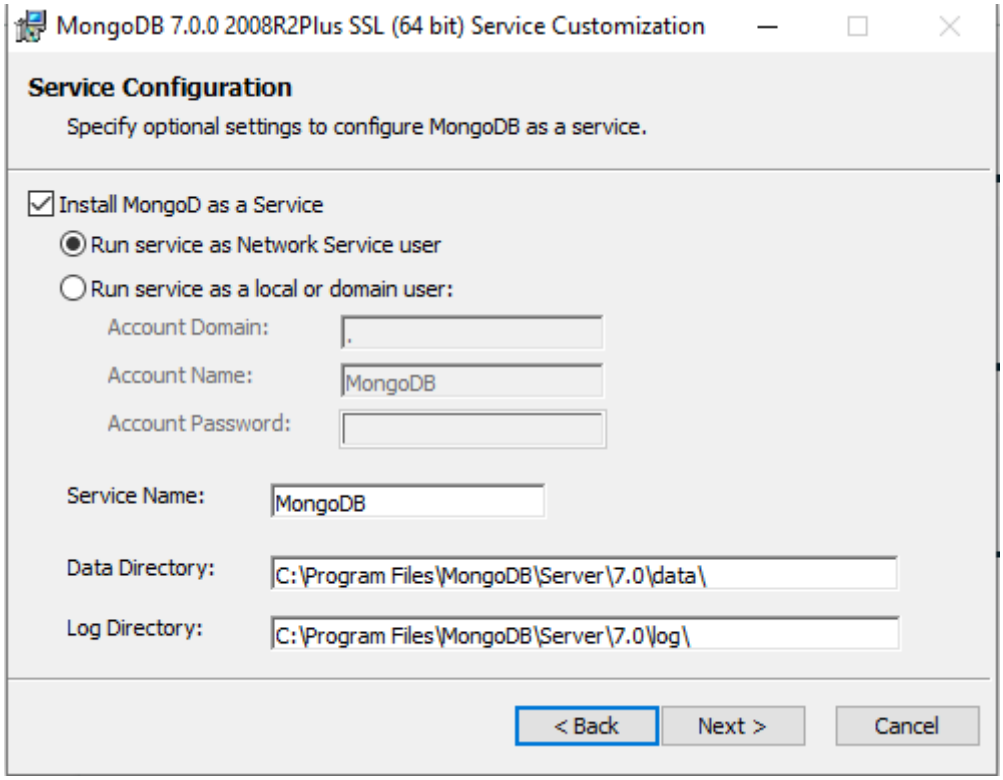

Figura Anexo 6: Instalación MongoDB (2)

En esta pantalla elegimos si queremos descargar mongodb compass. Elige la opción que prefieras. En nuestro caso vamos a pulsar next con el tick marcado.

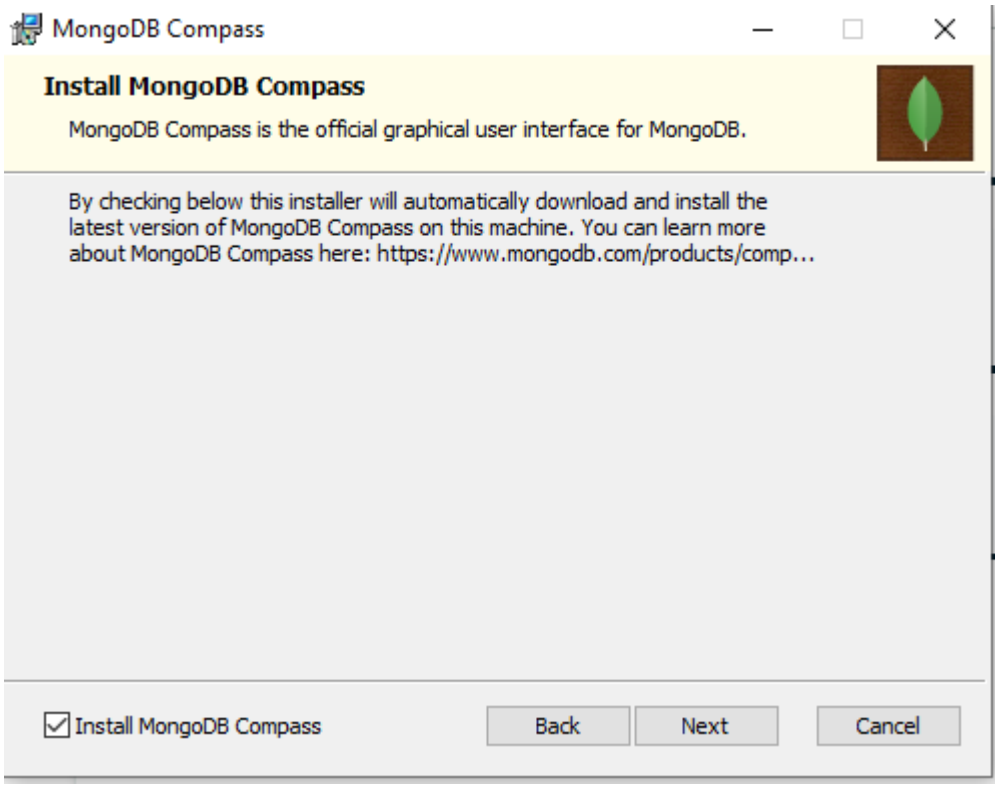

Figura Anexo 7: Instalación MongoDB (3)

En la siguiente pantalla pulsamos instalar y debería empezar la instalación de MongoDB

El siguiente paso será descargar RabbitMQ. Para ello nos vamos a la web oficial y pulsamos el botón de descargar [28]

Figura Anexo 8: Descarga RabbitMQ

Antes de abrir el ejecutable tenemos que descargarnos la última versión de Erlang. Esto lo haremos desde esta web [29]

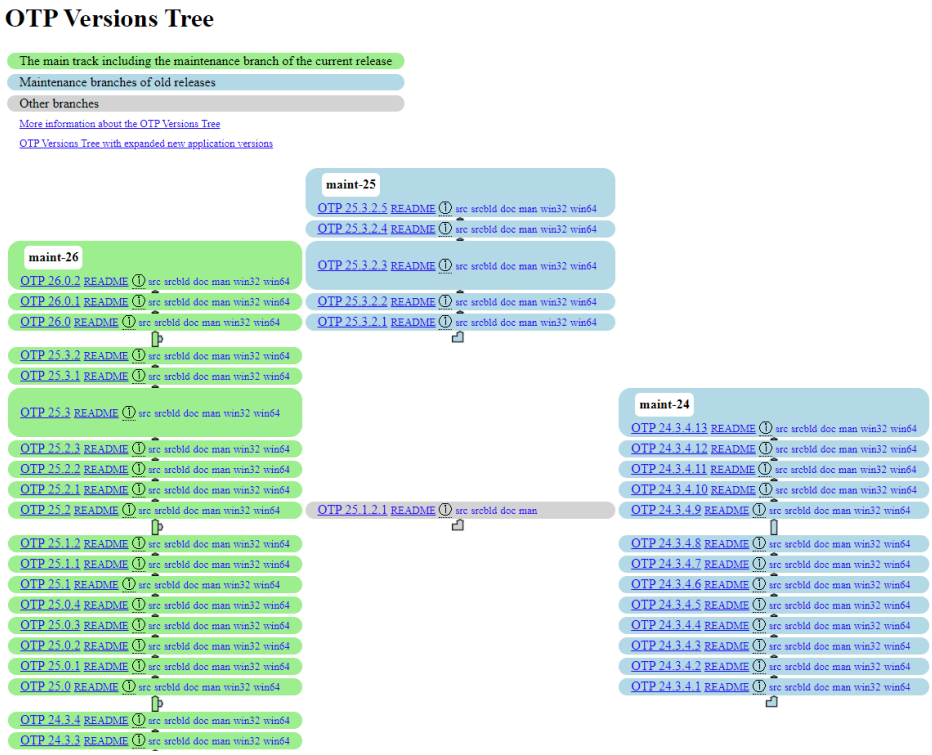

Figura Anexo 9: Descarga Erlang

En ella elegiremos la última versión para el sistema operativo que desee.

Se nos abrirá la siguiente ventana donde indicaremos los componentes a instalar. Instalaremos estos y pulsamos siguientes

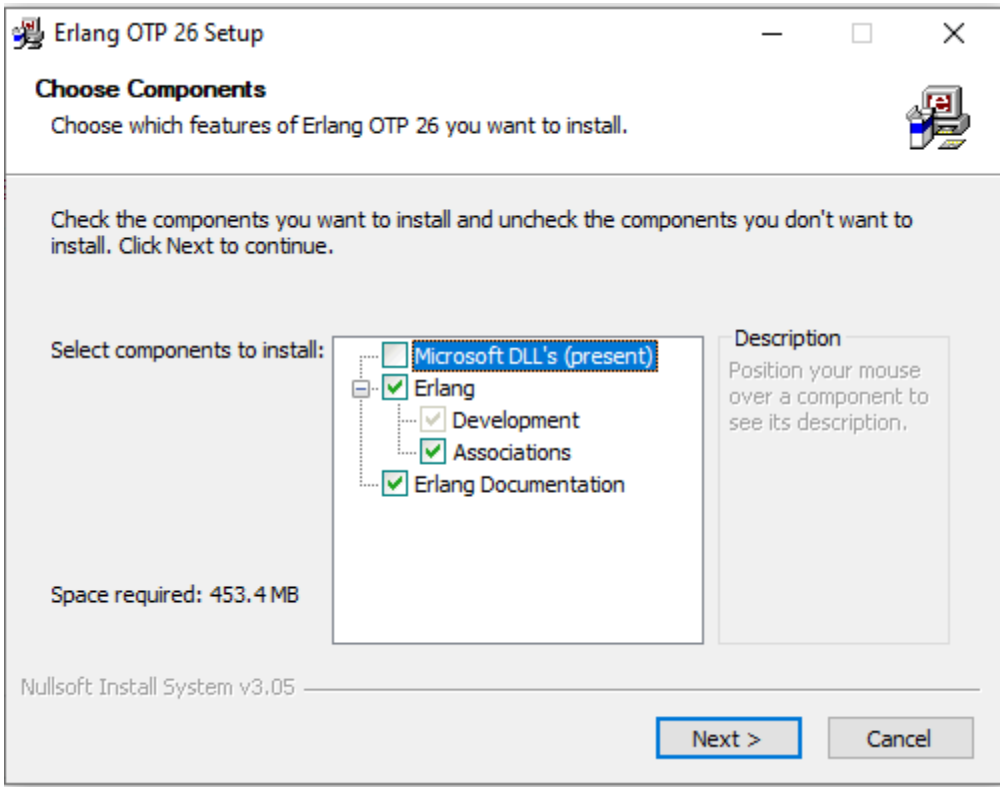

Figura Anexo 10: Instalación Erlang (1)

Elegimos la ruta de la carpeta y pulsamos siguiente y luego instalar para comenzar la instalación.

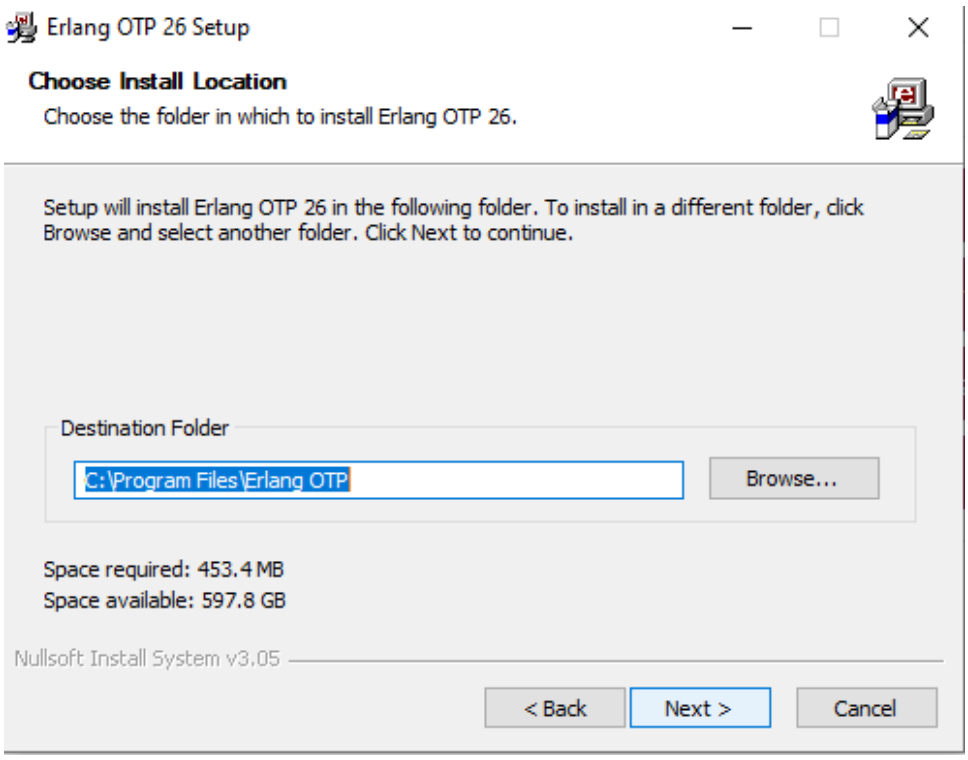

Figura Anexo 11: Instalación Erlang (2)

Una vez descargado Erlang abrimos el instalador de RabbitMQ.

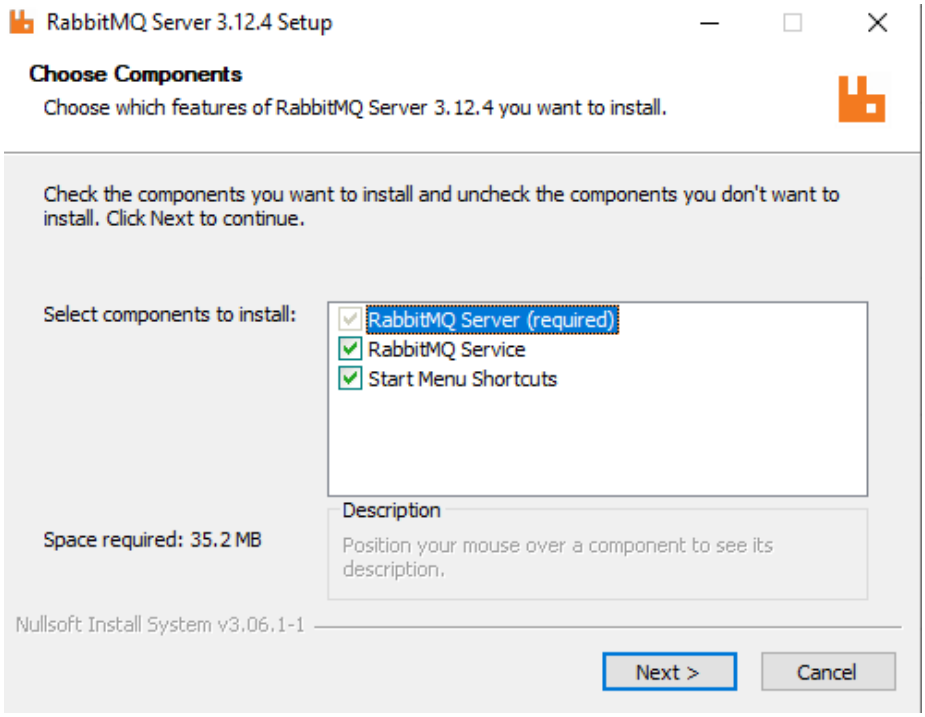

Figura Anexo 12: Instalación RabbitMQ (1)

Elegimos los componentes y pulsamos siguiente

Elegimos la carpeta de destino y pulsamos instalar

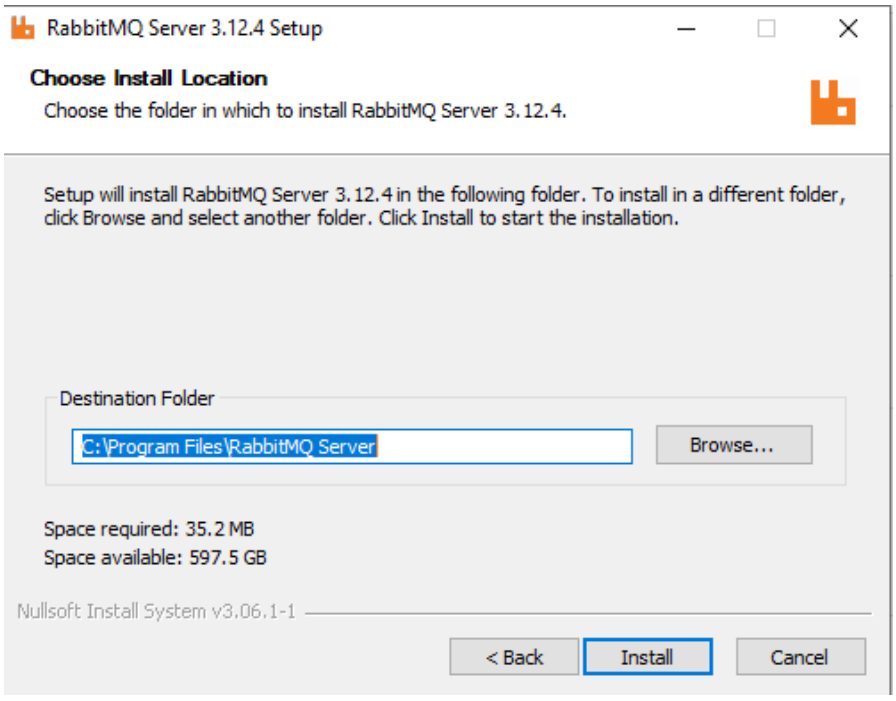

Figura Anexo 13: Instalación RabbitMQ (2)

Una vez descargado todo. Abrimos Anaconda

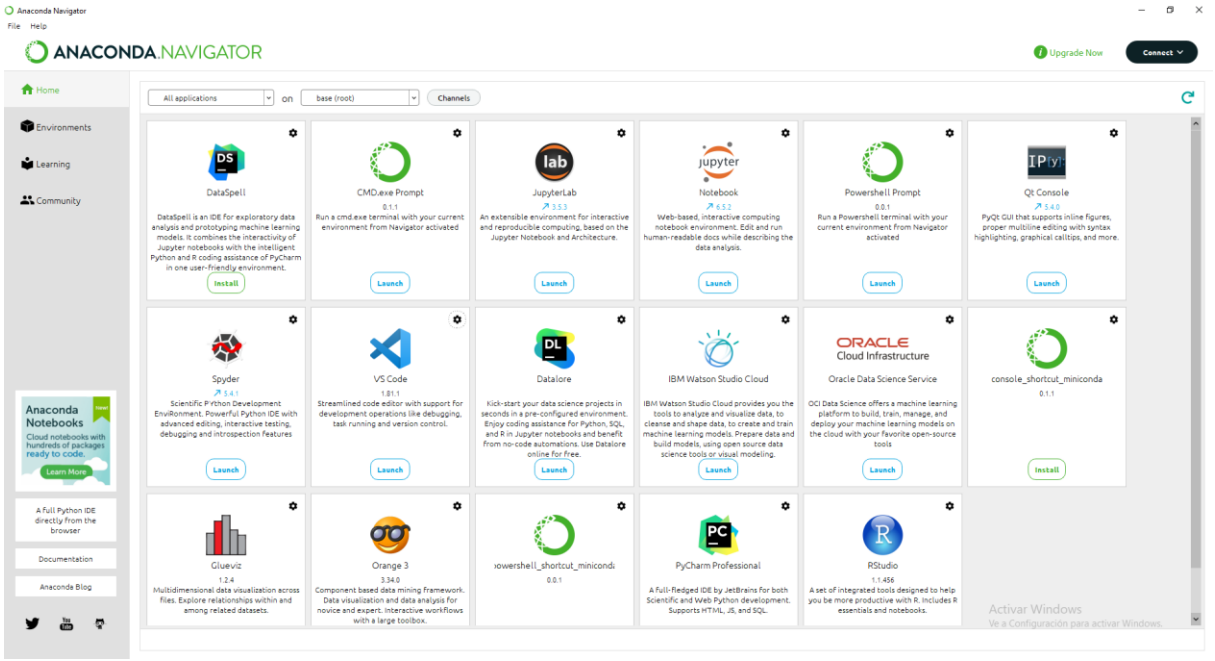

Figura Anexo 14: Anaconda

Una vez abierta la aplicación de anaconda pulsamos el botón de Visual Studio para abrirlo

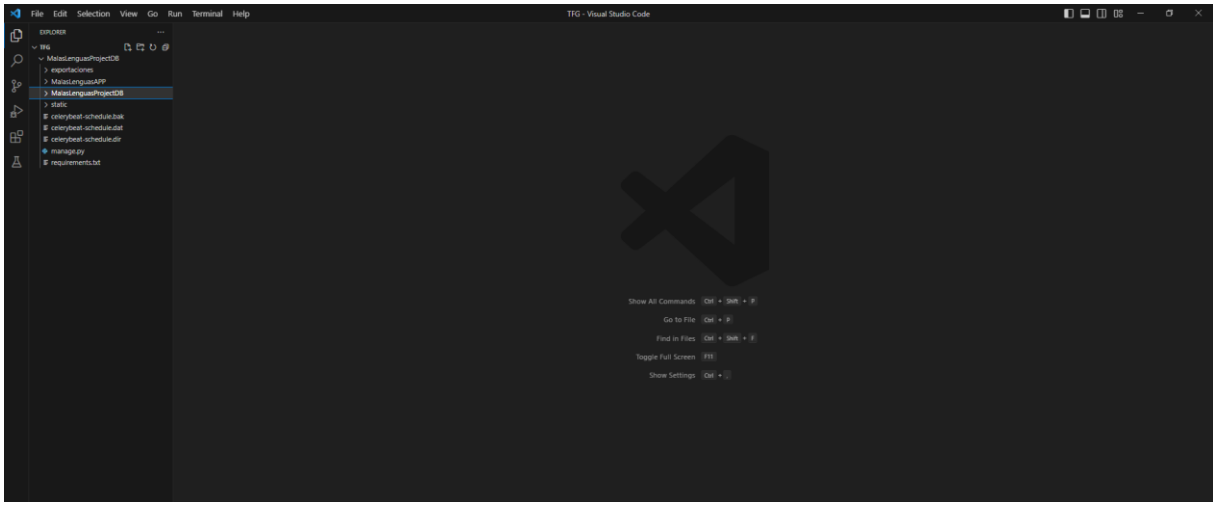

Figura Anexo 15: Visual Studio

Una vez abierto en la consola de comandos escribimos la siguiente línea

```
PS C:\Users\josem\Documents\GitHub\TFG\MalasLenguasProjectDB> pip install -r requirements.txt
```

Figura Anexo 16: instalación de requerimientos

Esto instalará todo lo necesario para que funcione la aplicación.

Para desplegar la aplicación es necesario tener la base de datos iniciada. Por eso debemos buscar la palabra servicios en el explorador de Windows.

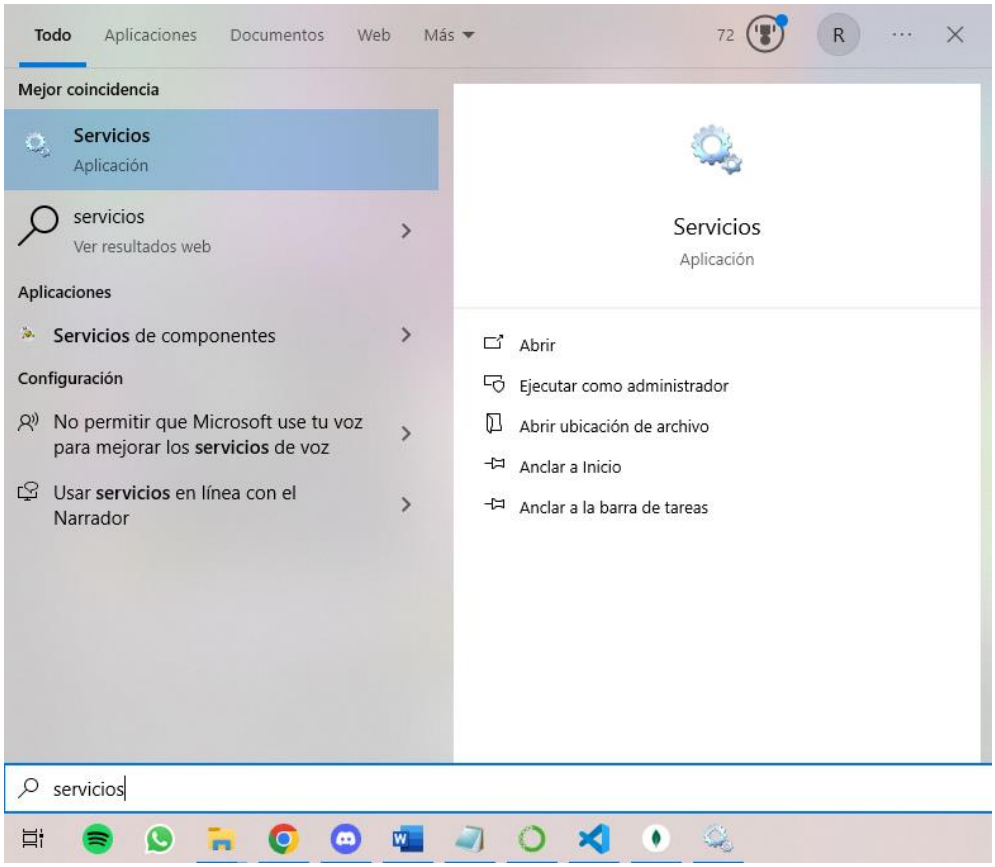

Figura Anexo 17: búsqueda de servicios

Una vez dentro de servicios buscamos el servicio de MongoDB y lo iniciamos

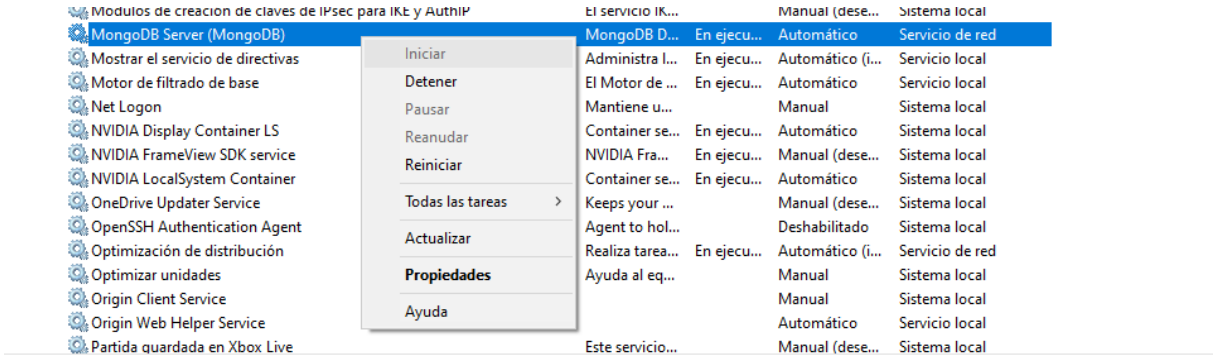

Figura Anexo 18: iniciar server MongoDB

Una vez hecho esto, ejecutamos las siguientes líneas de código en el visual studio.

```
python manage.py makemigrations MalasLenguasAPP
```

Figura 81: Comando makemigrations

```
python manage.py migrate
```

Figura 82: Comando migrate

Estas funciones comprueban los cambios en la base de datos mediante el archivo models.py y lo ejecuta para que se reflejen en la base de datos.

Posteriormente abrimos la aplicación MongoDB Compass y nos conectamos a la base de datos de la siguiente forma

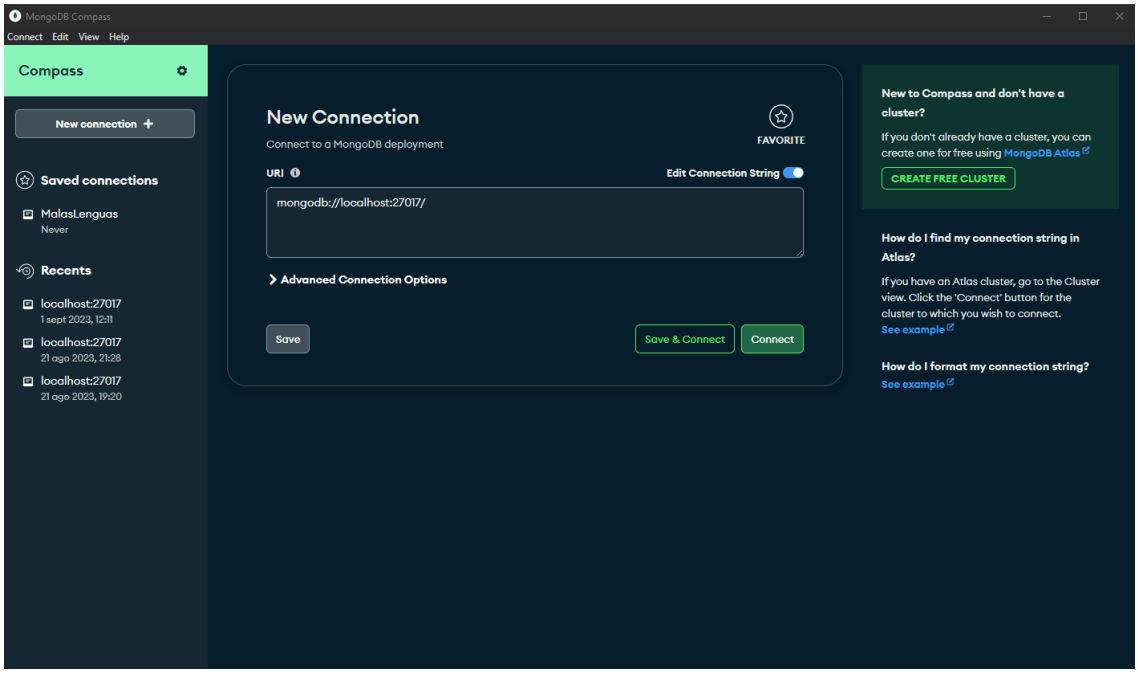

Figura Anexo 19: MongoDB Compass. Conexión a la base de datos

Pulsamos el botón de conectar y esto te conectará a la base de datos. Ahora, por cada archivo en la carpeta exportaciones importamos a su tabla correspondiente

pulsando el botón de add Data y luego import JSON or CSV file

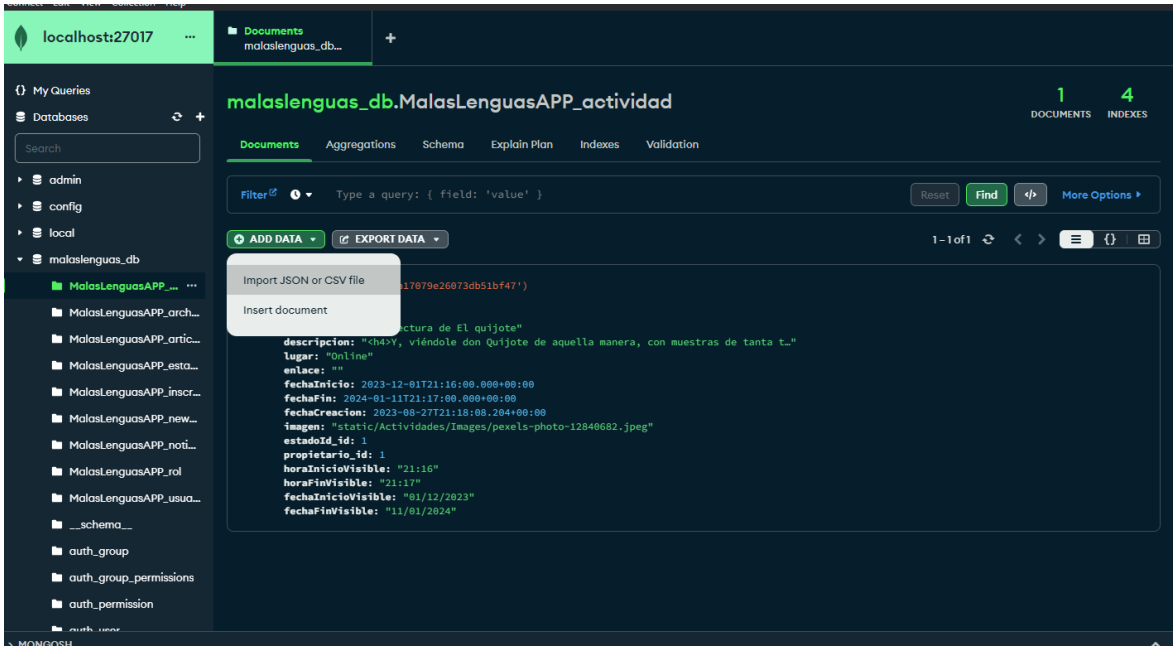

Figura Anexo 20: MongoDB Compass. Importar archivos

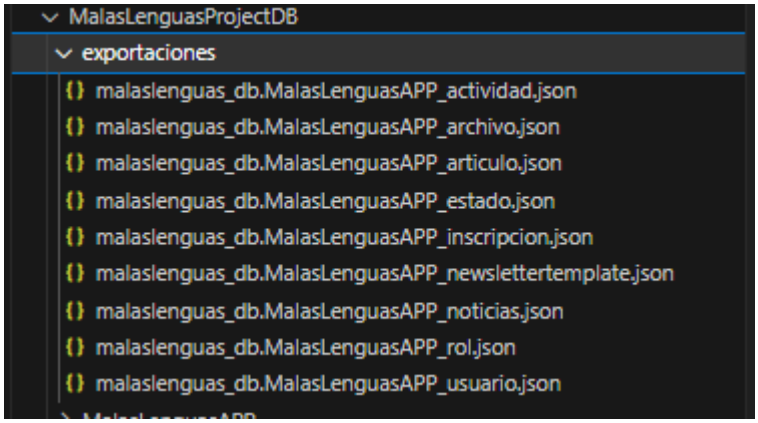

Figura Anexo 21: Carpeta de exportaciones

Hecho esto procedemos a escribir la siguiente línea dentro del visual studio previamente abierto.

```
python manage.py runserver 8080
```

Figura Anexo 22: Poner en marcha MongoDB

Y la aplicación debería estar activa en localhost:8080

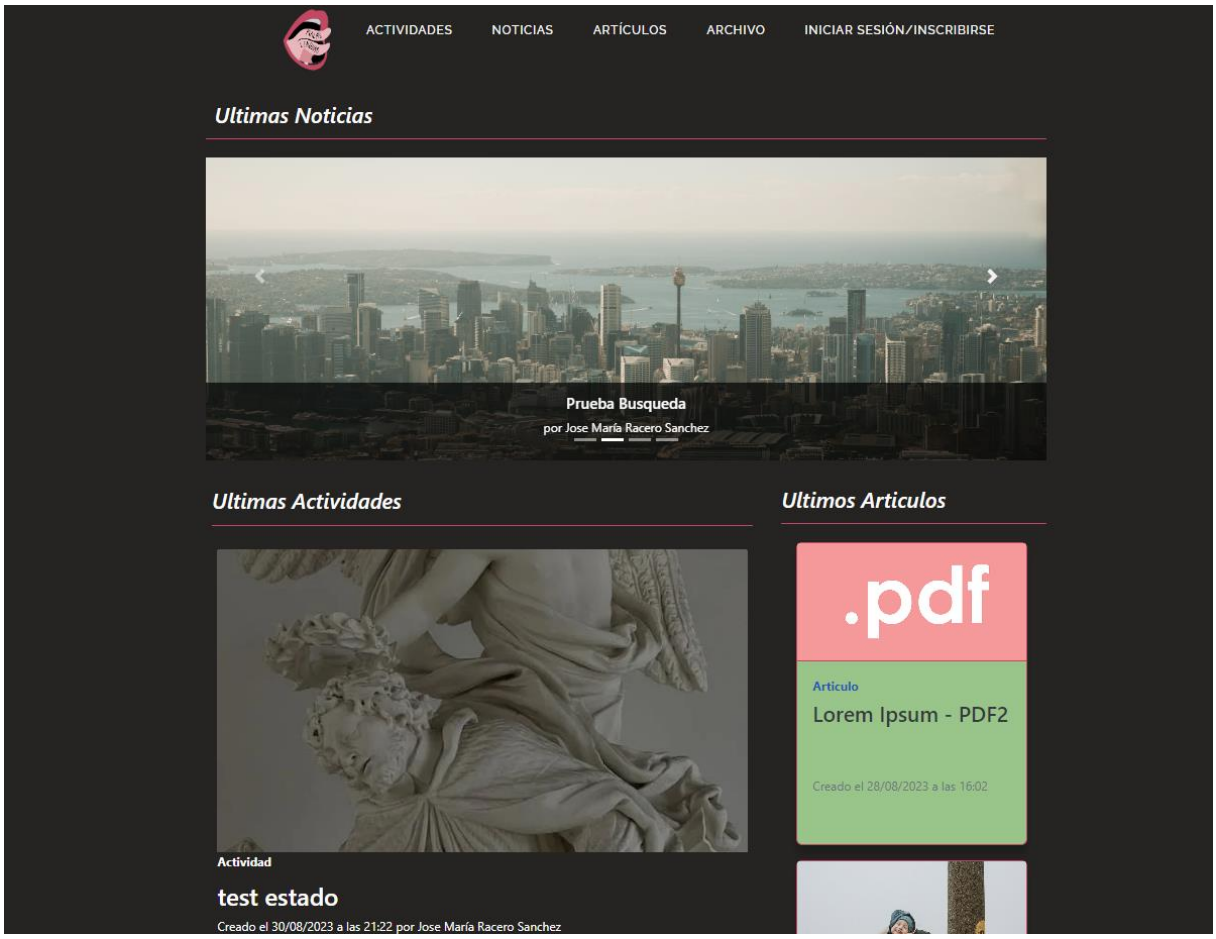

Figura Anexo 23: Página de inicio de la web

Para que el servicio de mensajería funcione debemos activar el servicio de rabbitMQ. Para ello diríjase a la carpeta donde haya instalado RabbitMQ y busque el server para iniciarlo. Por defecto estará en:

C:\Program Files\RabbitMQ Server\rabbitmq_server-3.12.2\sbin

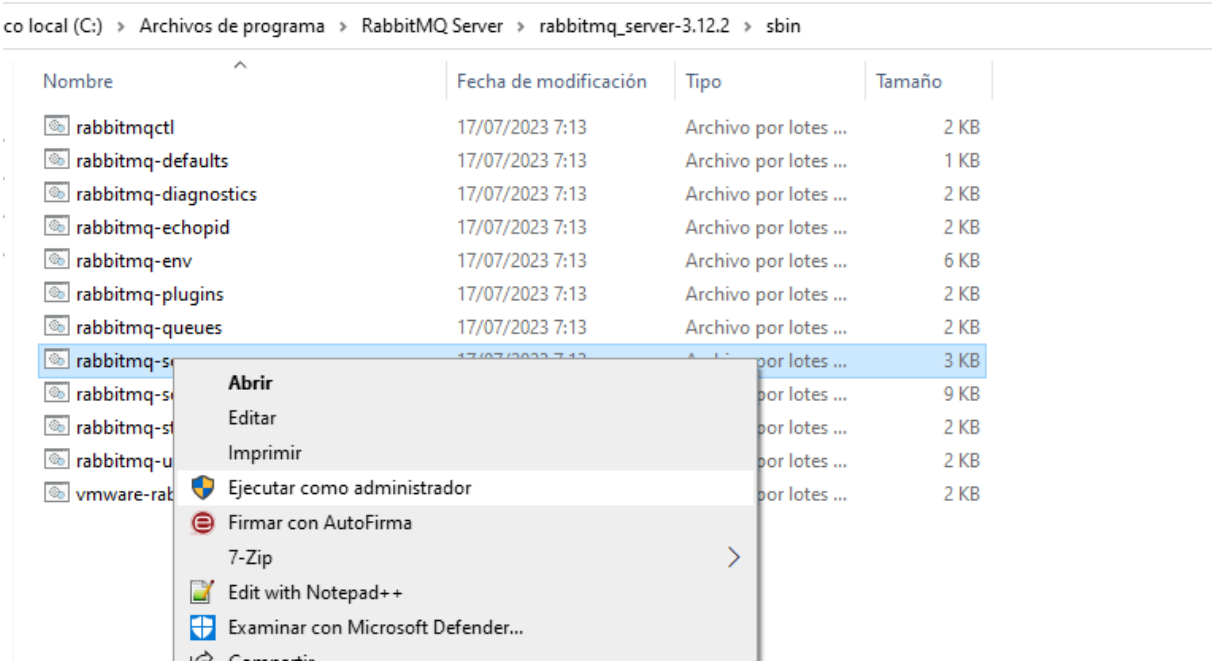

Figura Anexo 24: Iniciar server RabbitMQ

Deberá ejecutar como administrador el servicio.

Una vez realizado este paso lo siguiente será activar el worker y el beat dentro de visual studio

Para ello hay que abrir dos consolas de comando. En una activamos el worker con la siguiente línea

```
celery -A MalasLenguasProjectDB worker -l info --pool=solo
```

Figura Anexo 25: Iniciar worker celery

En la otra activamos el beat con la siguiente línea

```
celery -A MalasLenguasProjectDB beat -l info
```

Figura Anexo 26: Iniciar beat celery

Con todos estos pasos la aplicación estará completamente funcional.

Apéndice B

Manual de Usuario

Abrir la web le mostrará la vista principal de esta

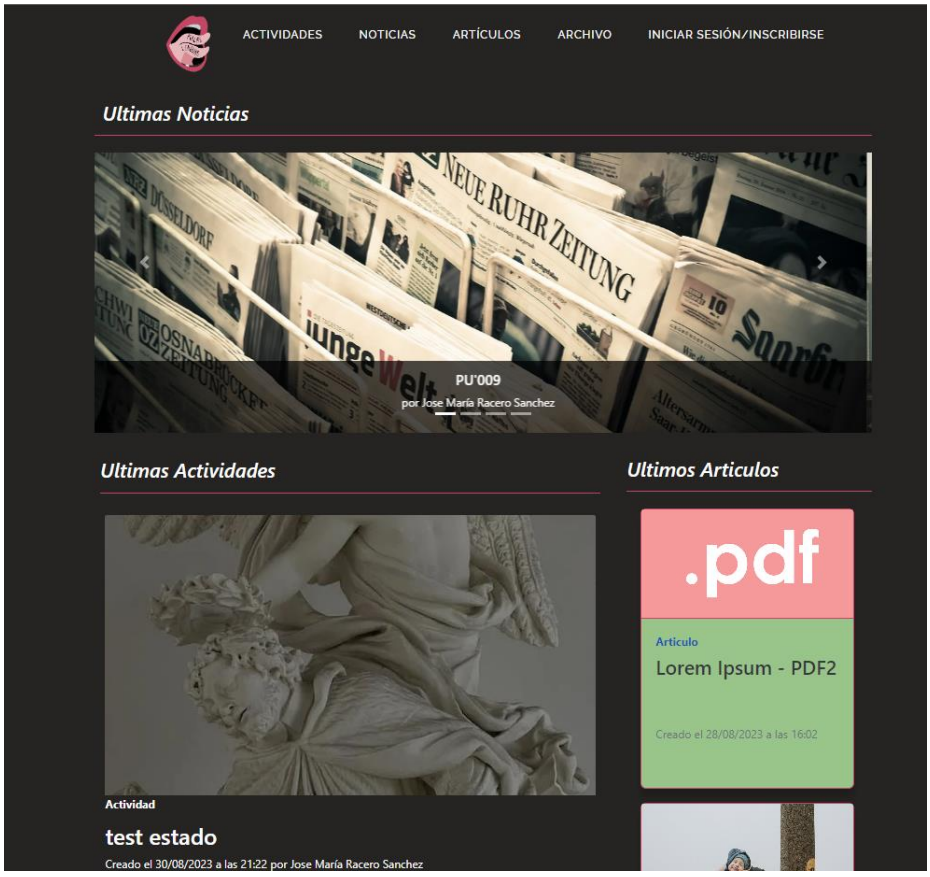

Figura Anexo 27: Página de inicio

Lo primero de todo necesitará iniciar sesión. Para ello

Introducirá las siguientes credenciales:

- Email: Admin@admin.com
- Contraseña Admin123

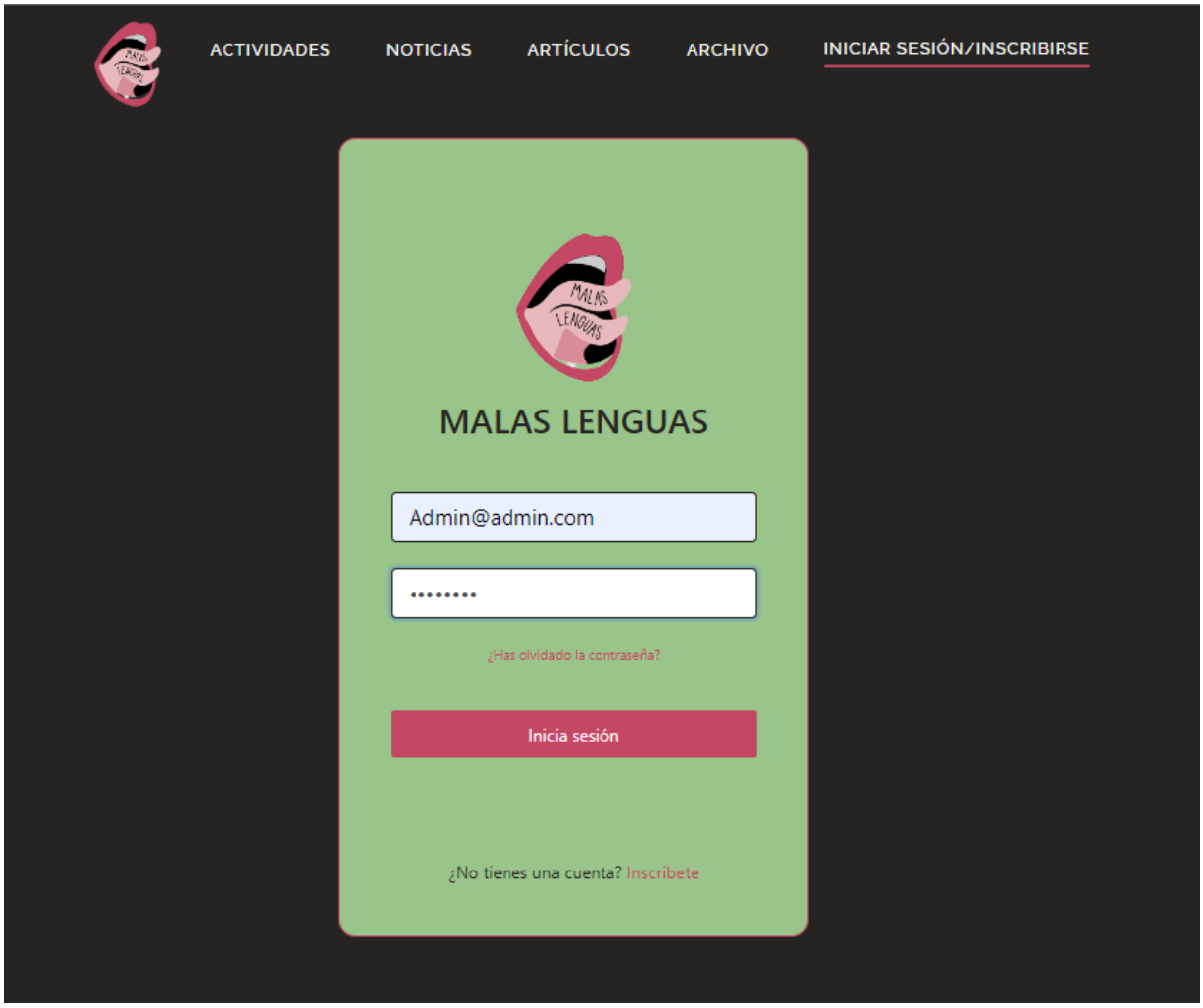

Figura Anexo 28: Página de Login

Una vez haya iniciado sesión podrá empezar a utilizar la aplicación.

Para ver las actividades debe pulsar la pestaña de Actividades en la parte superior de la web

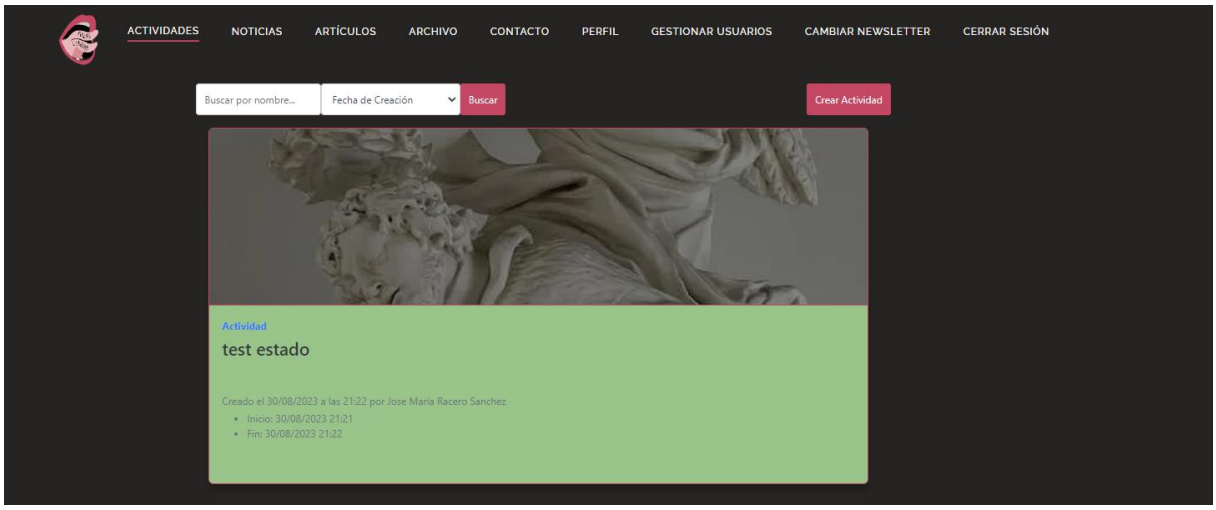

Figura Anexo 29: Página de actividades

Desde aquí podrá crear actividades o mirar las actividades ya creadas.

Para crear actividades debe de pulsar el botón Crear Actividad y rellenar el formulario.

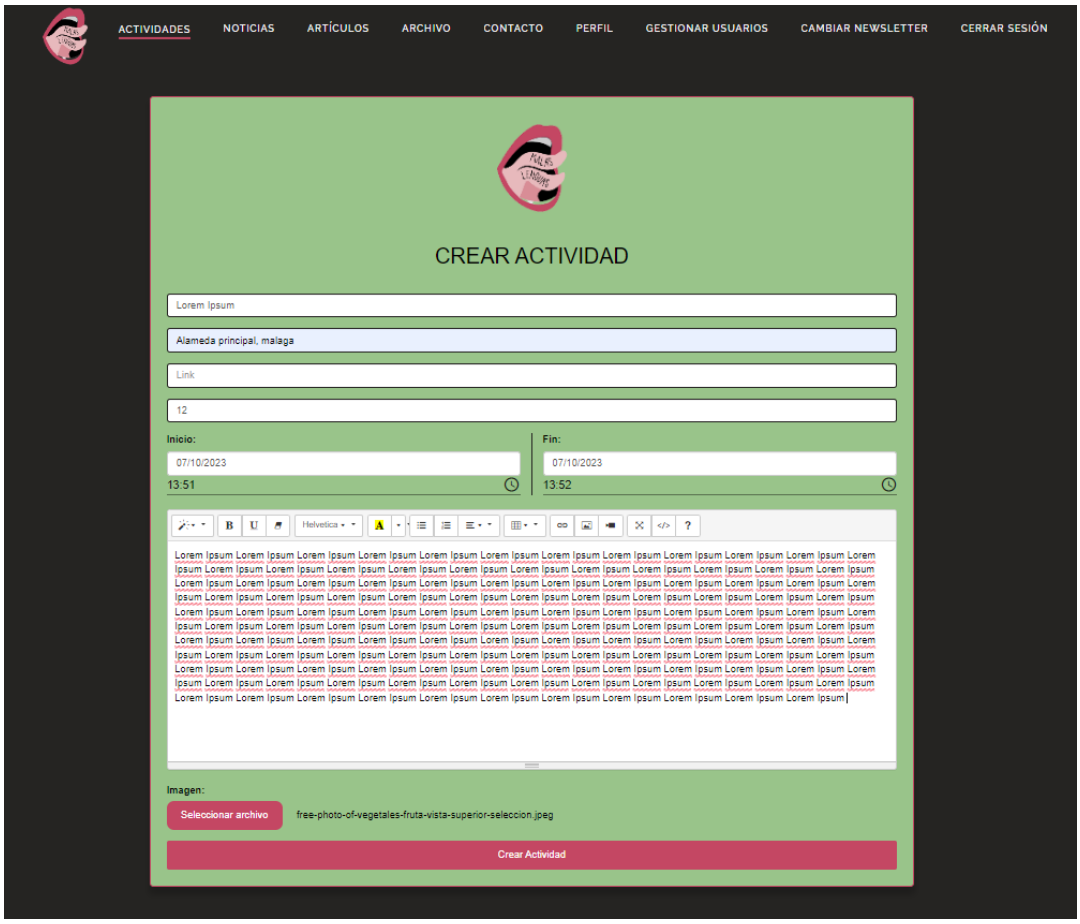

Figura Anexo 30: Formulario de creación de actividad

Una vez relleno el formulario pulsará el botón Crear Actividad y verá la actividad Creada

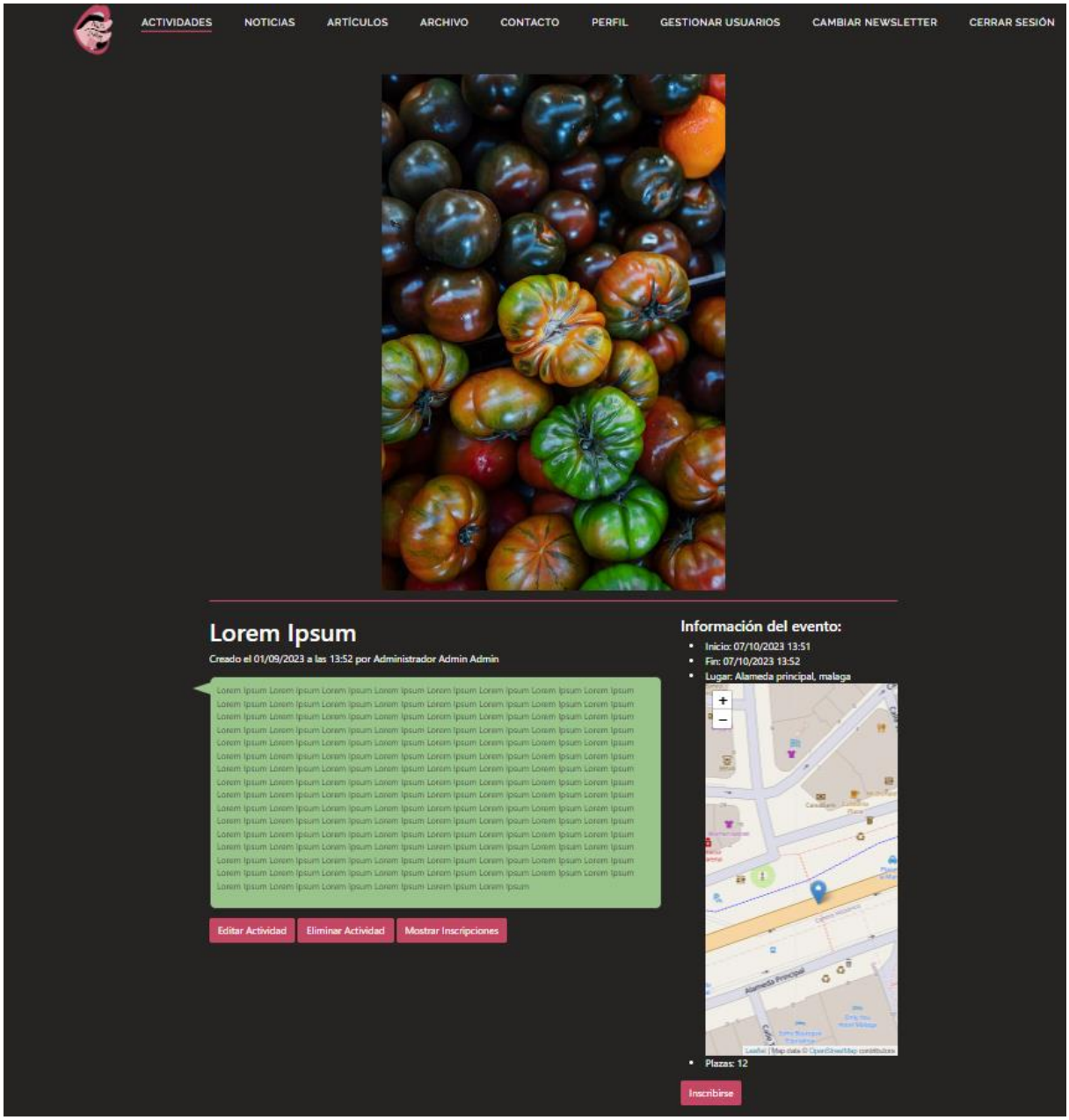

Figura Anexo 31: Dentro de una actividad

Para editar una actividad bastará con pulsar el botón Editar Actividad y cambiar los datos

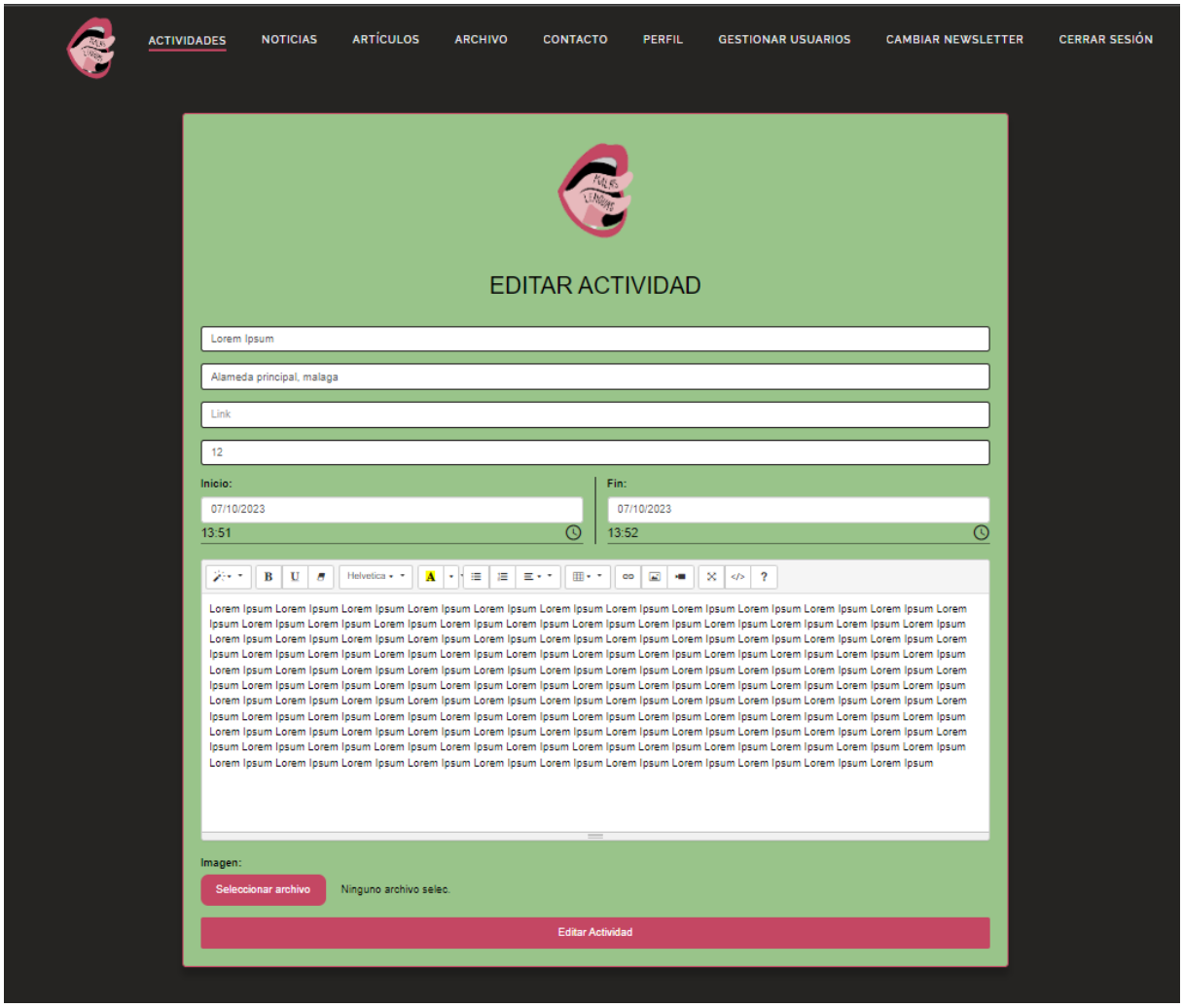

Figura Anexo 32: formulario de edición de actividad

Para eliminar la actividad bastará con pulsar el botón de Eliminar Actividad.

Se le indicará mediante un mensaje que la actividad se ha eliminado y los inscritos recibirán un correo

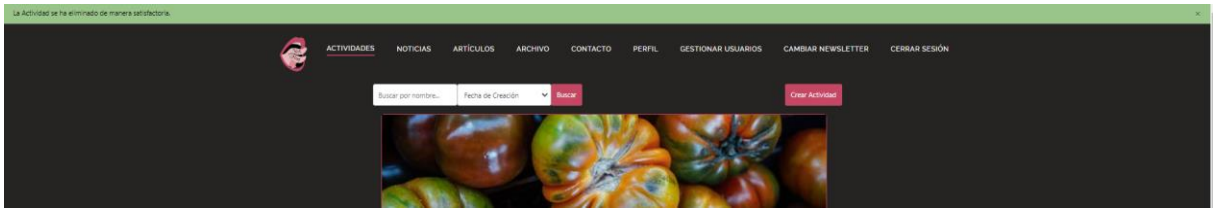

Figura Anexo 33: Mensaje de éxito al eliminar actividad

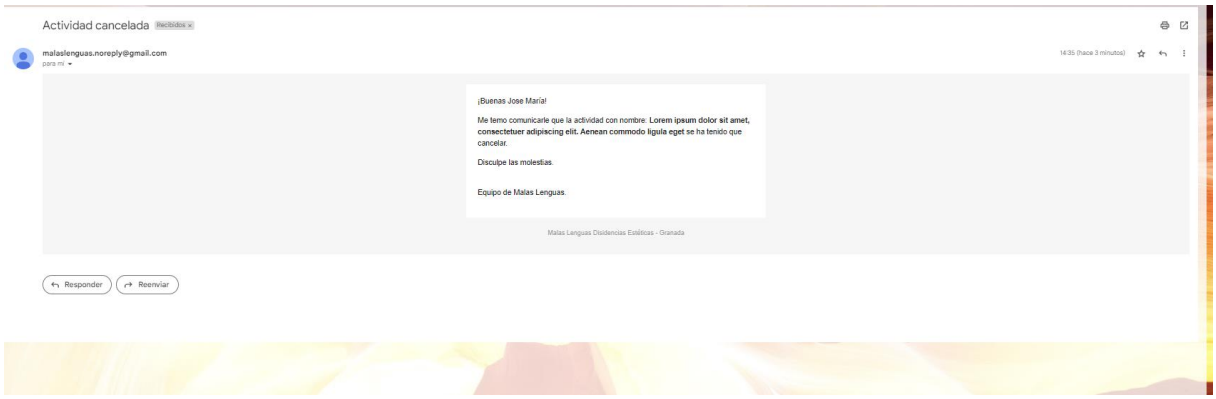

Figura Anexo 34: Correo de eliminación de actividad

Para inscribirse en una actividad bastará con pulsar el botón de inscribirse y recibirá el siguiente mensaje

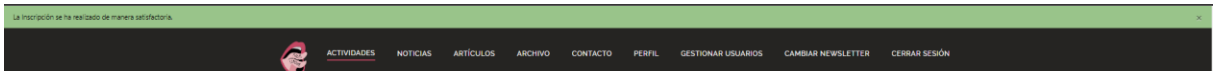

Figura Anexo 35: Mensaje de éxito al inscribirse

Para ver las inscripciones de una actividad bastará con pulsar el botón Mostar inscripciones

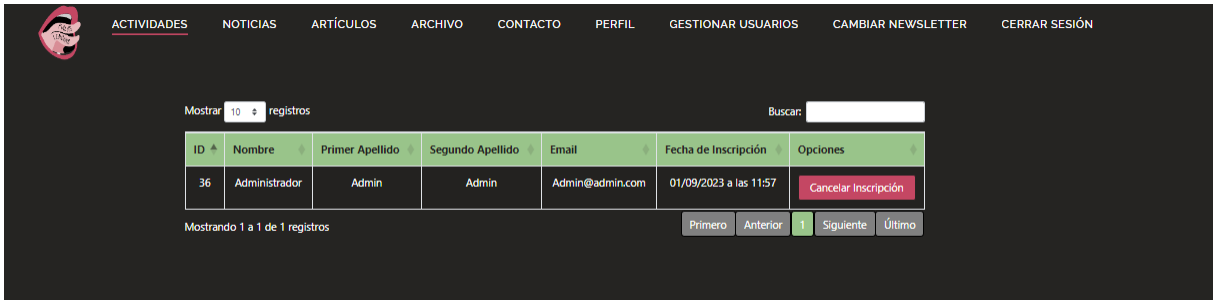

Figura Anexo 36: Vista de inscripciones de actividad

Aquí podrá cancelar las inscripciones de cualquier usuario inscrito en caso de que lo necesite.

Un usuario normal solamente podrá ver sus propias inscripciones, esto se hace dentro del Perfil

Figura Anexo 37: Vista de inscripciones de usuario

El proceso de creación de noticias y de artículos es el mismo que para actividades. Solamente que no existe la opción de inscribirse

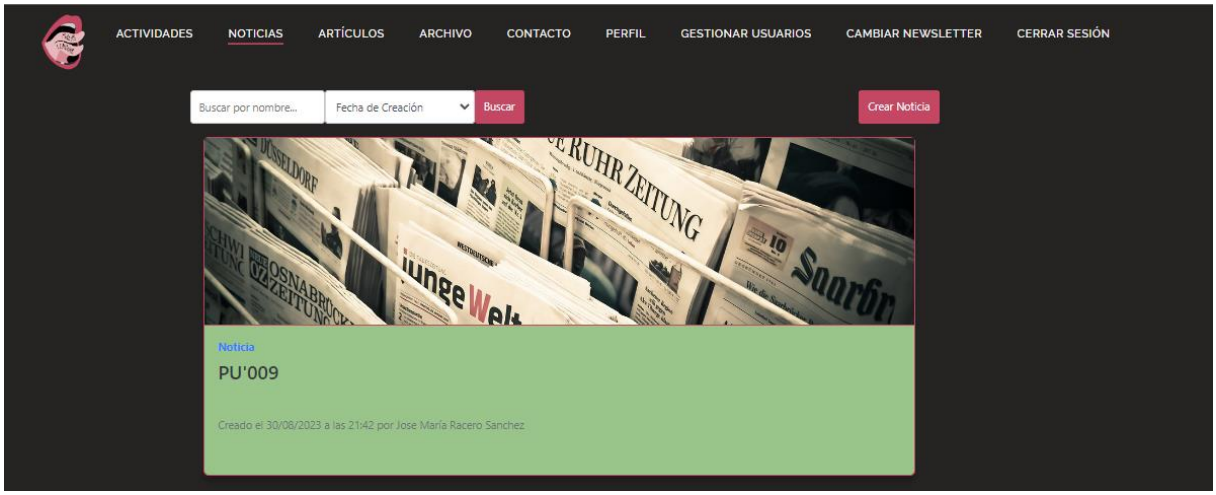

Figura Anexo 38: Vista de noticias

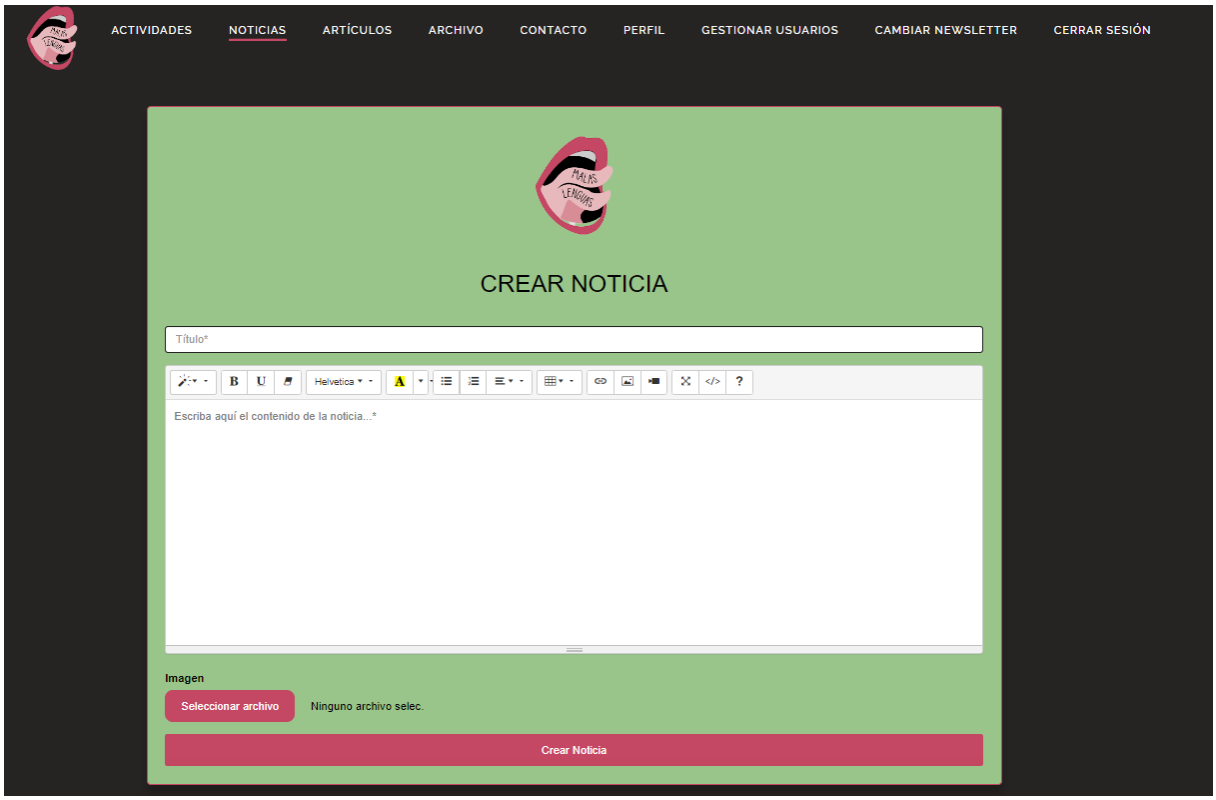

Figura Anexo 39: Formulario de crear noticias

Figura Anexo 40: Interior de noticia

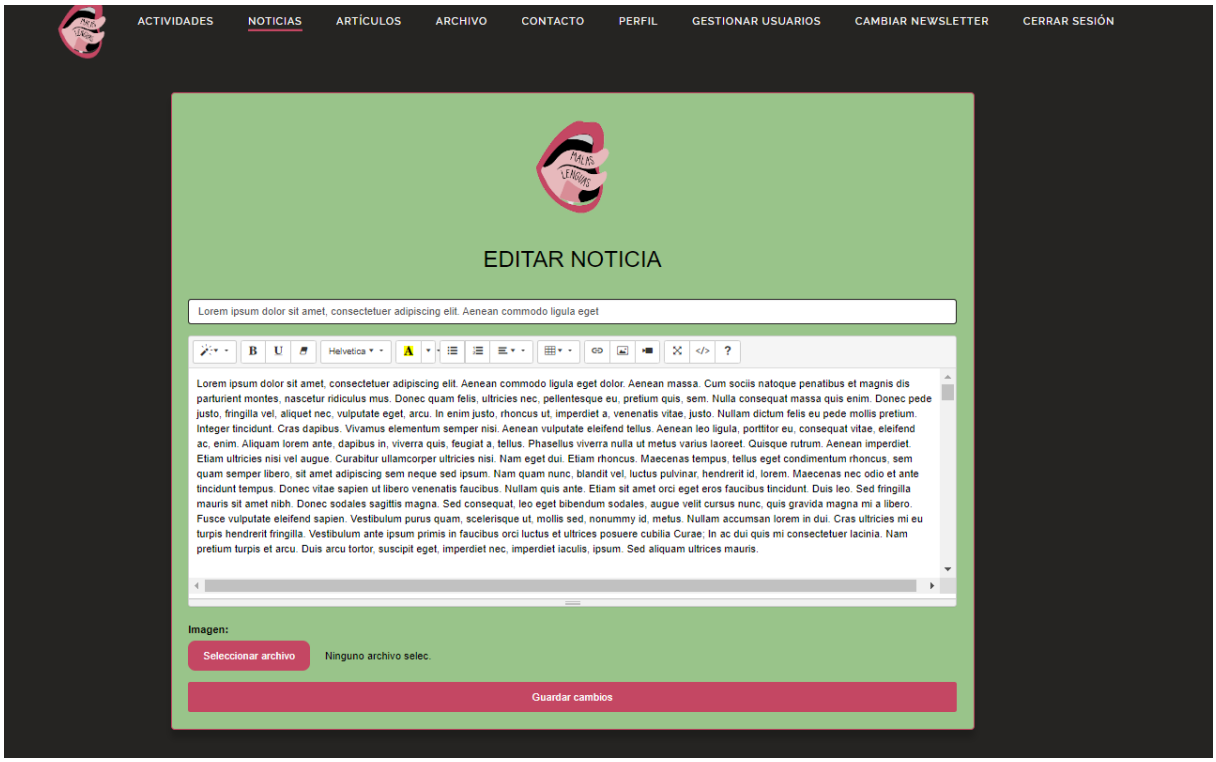

Figura Anexo 41: Formulario de editar actividad

Los artículos en vez de abrirse en una pestaña propia, se abrirán en el lector de pdf o se descargarán dependiendo de cómo tenga el usuario configurado su navegador

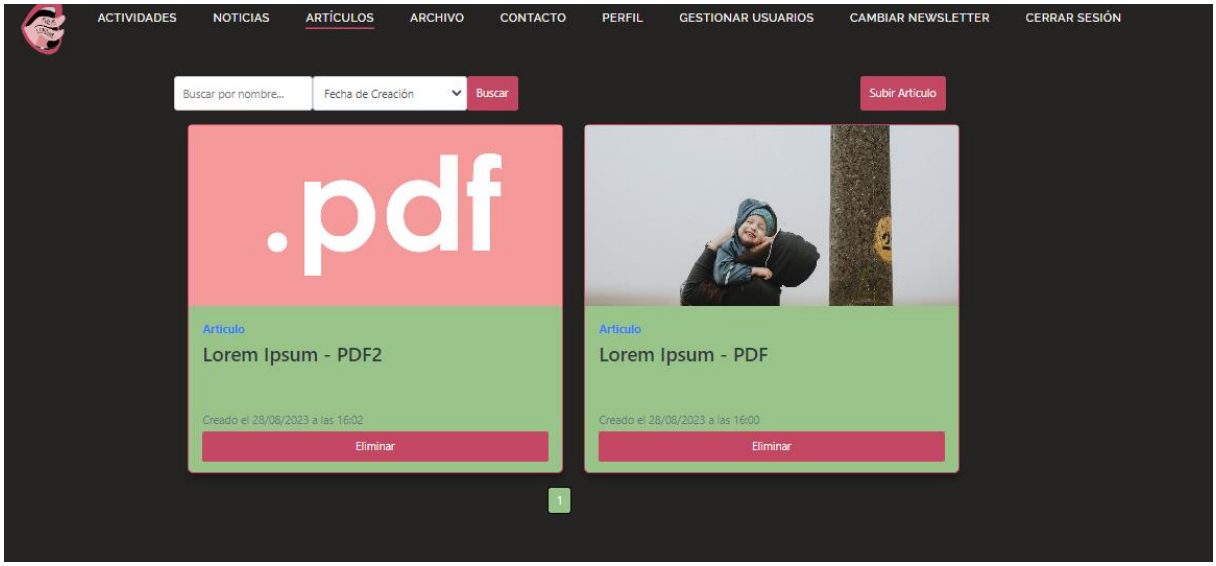

Figura Anexo 42: Vista de artículos

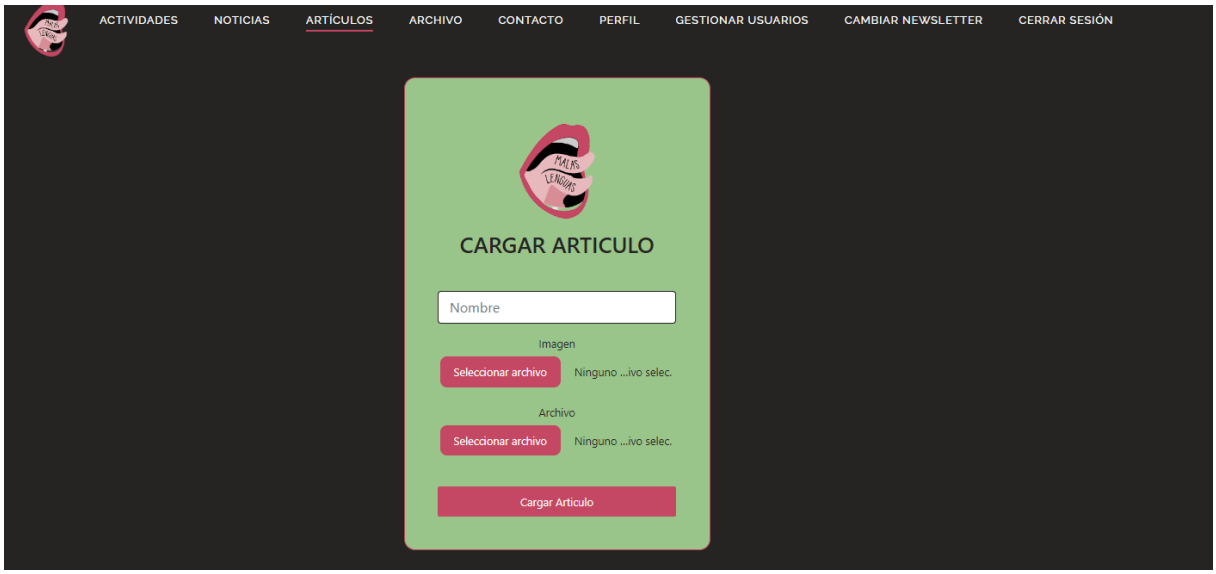

Figura Anexo 43: Formulario de subir articulo

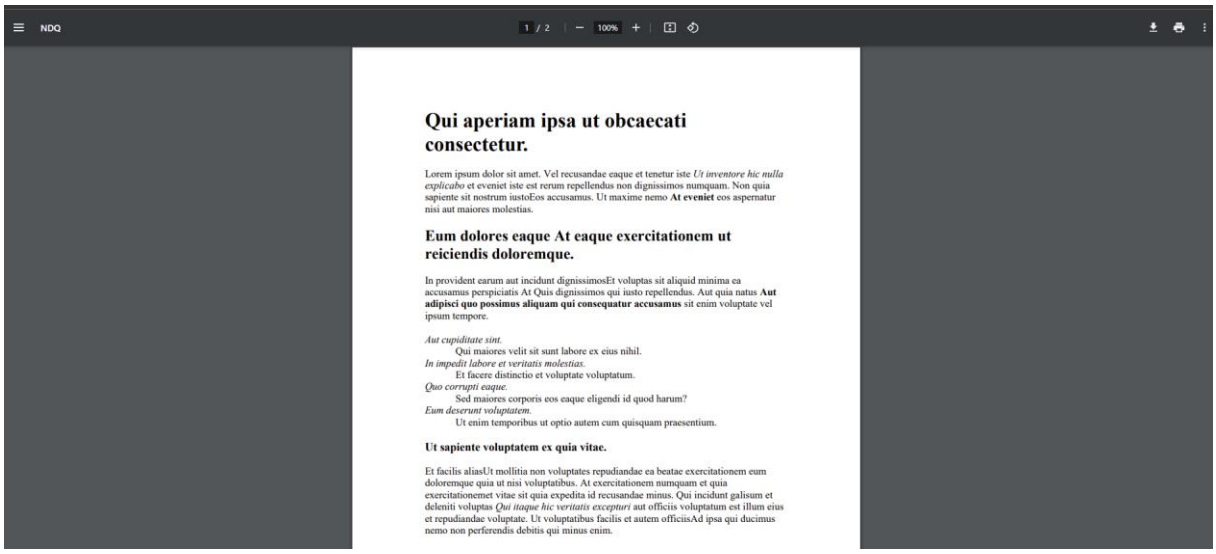

Figura Anexo 44: Ejemplo de abrir artículo en la web

En Archivo encontrará todas las imágenes y videos que suban a la aplicación

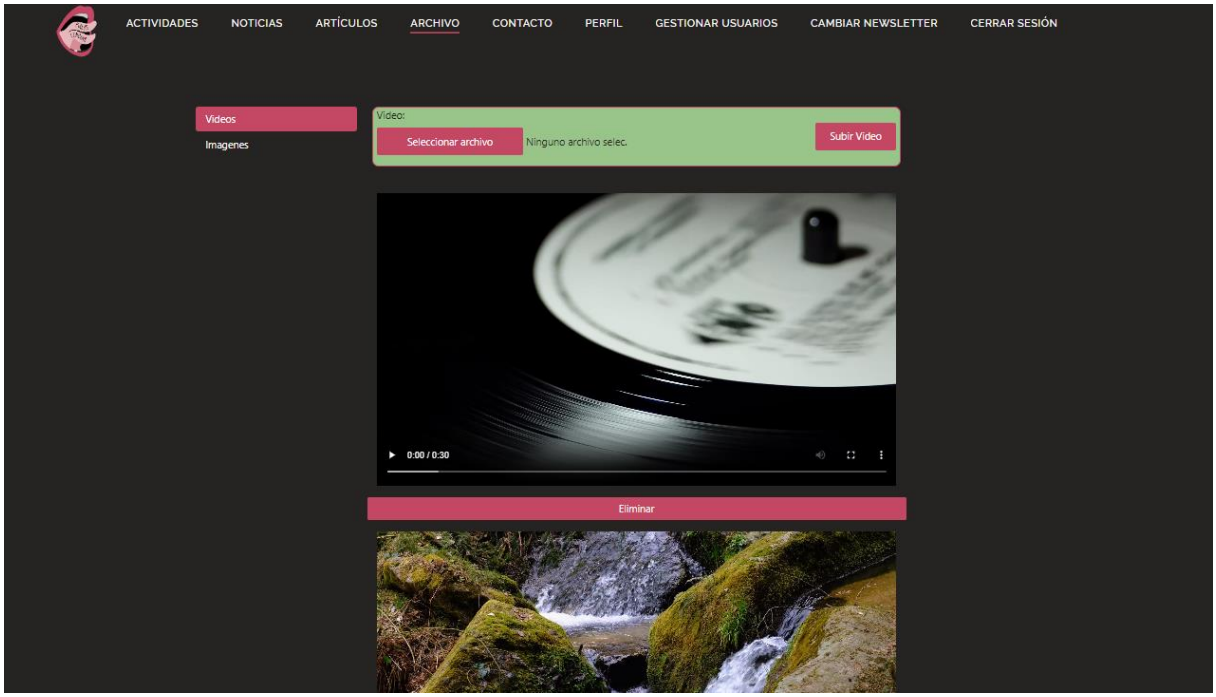

Figura Anexo 45: Vista de videos

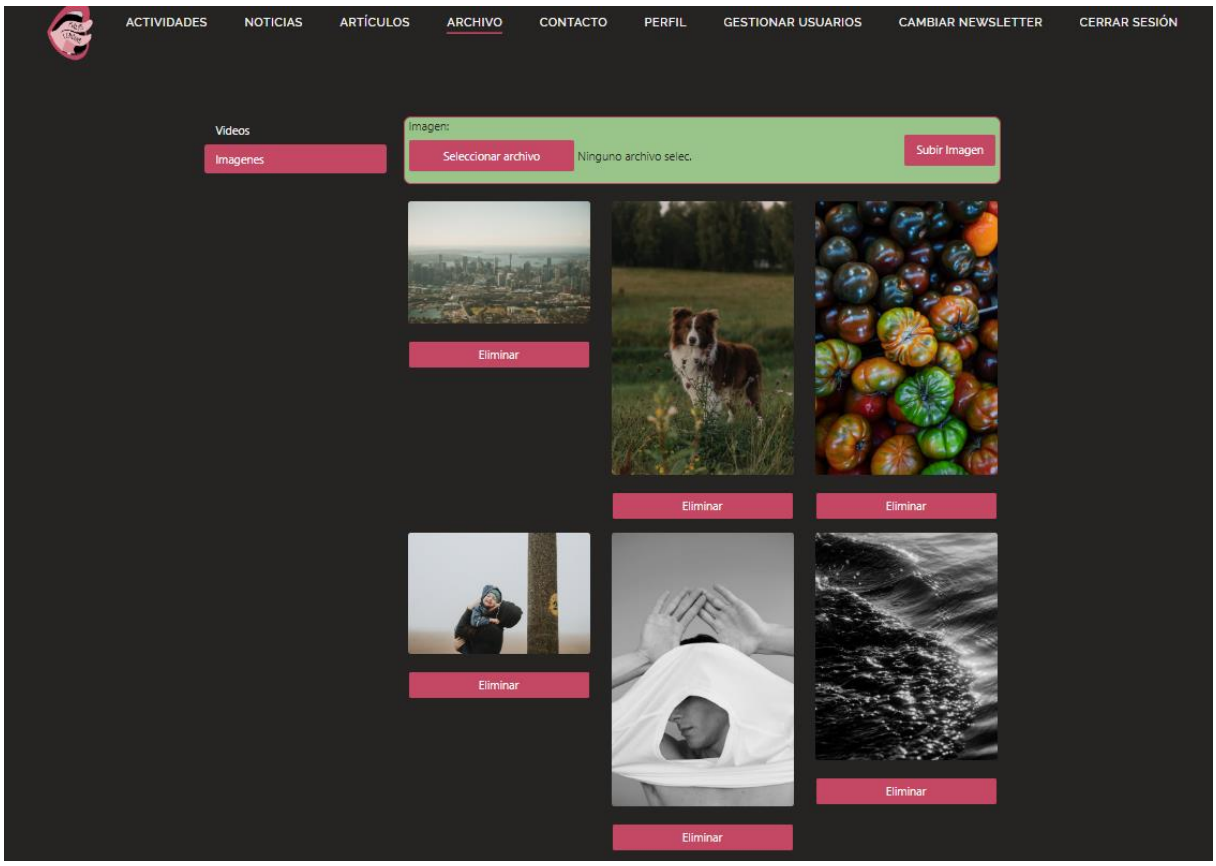

Figura Anexo 46: Vista de imágenes

Para subirlos solo bastará con añadir el archivo y darle al botón de subir.

Para eliminarlo solamente será necesario pulsar el botón de eliminar.

Para enviar propuestas al correo de malas lenguas será necesario rellenar el formulario y darle a enviar propuesta

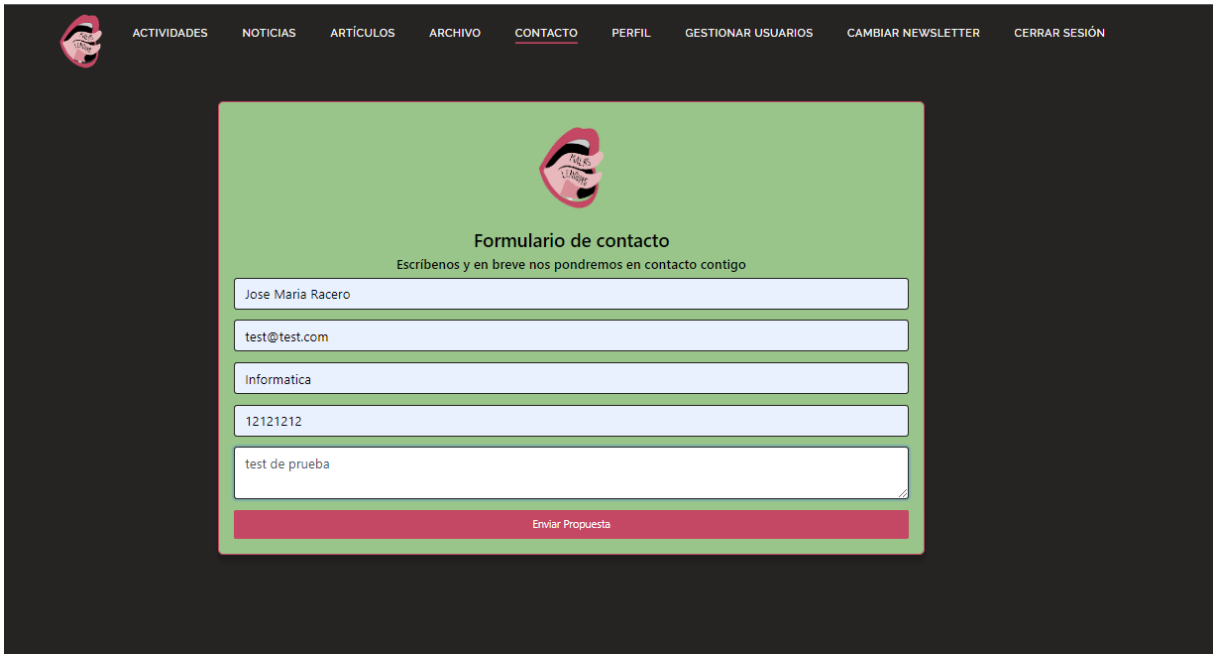

Figura Anexo 47: Formulario de contacto

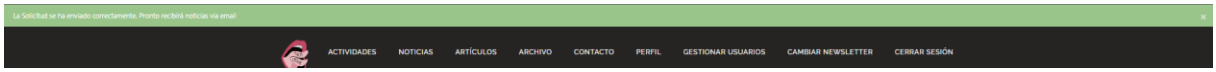

Figura Anexo 48: Mensaje de mensaje enviado

En el buzón de malas lenguas recibirá el correo de la propuesta para su posterior análisis

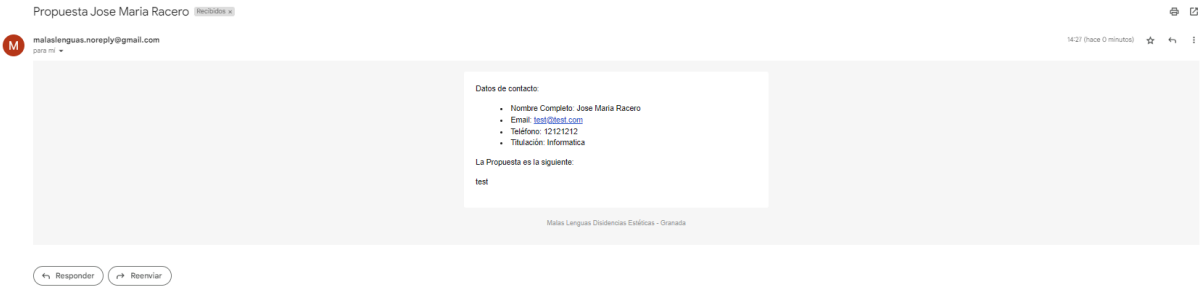

Figura Anexo 49: Email de propuesta

Para gestionar los usuarios tendrá que pulsar en la pestaña Gestionar Usuarios

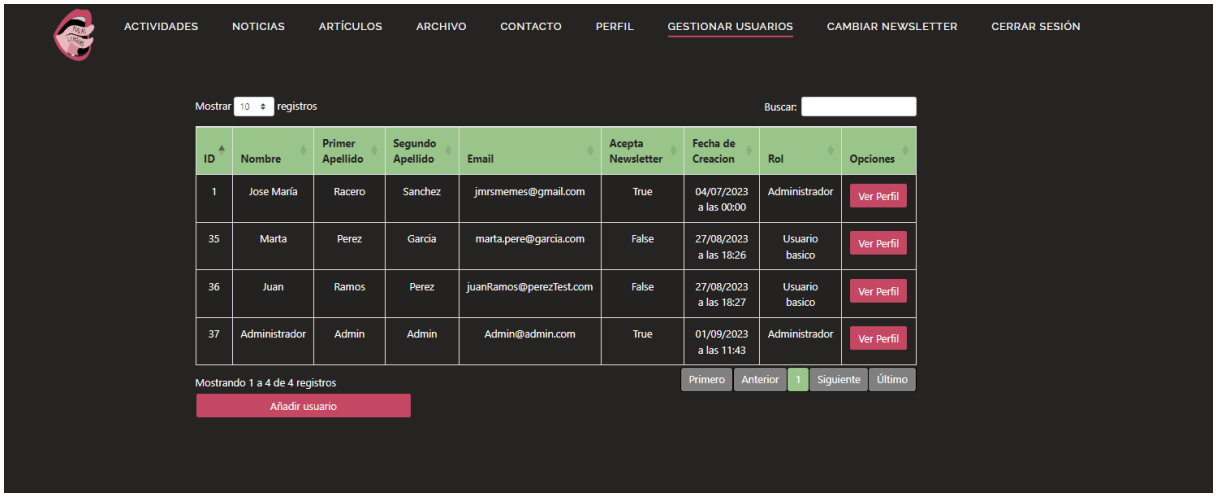

Figura Anexo 50: Vista de administración de usuarios

En esta pestaña podrá ver los perfiles de los usuarios y modificarlos a placer

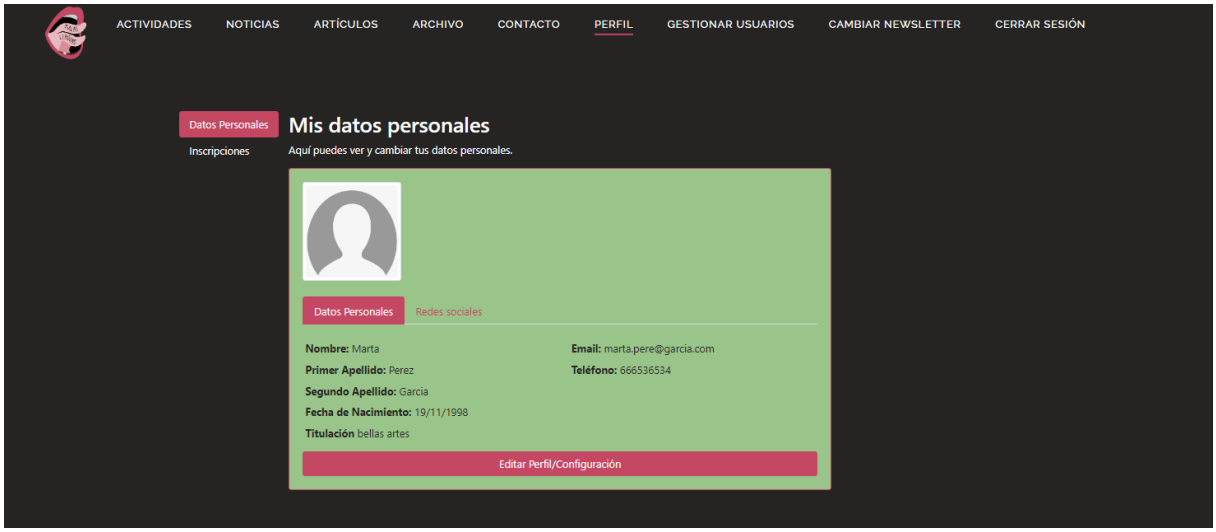

Figura Anexo 51: Perfil de usuario

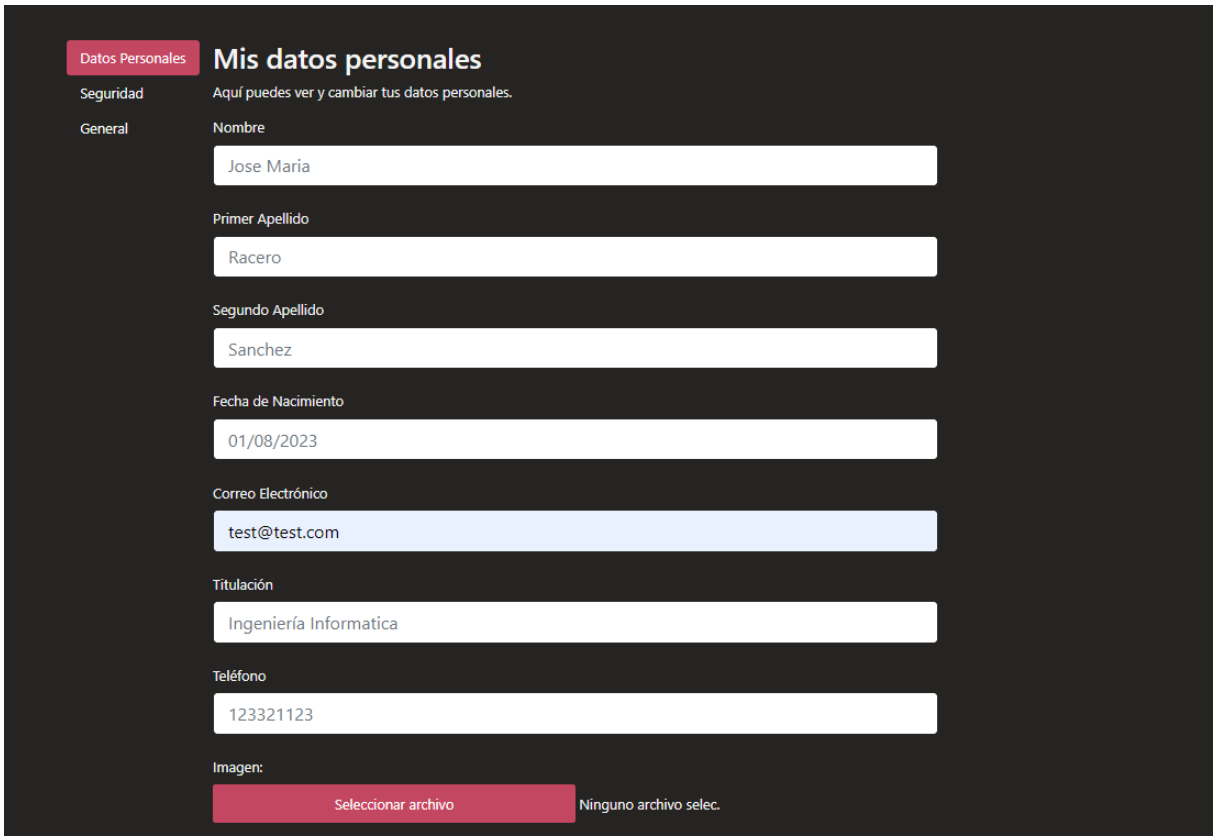

Figura Anexo 52: Formulario de datos personales (1)

Figura Anexo 53: Formulario de datos personales (2)

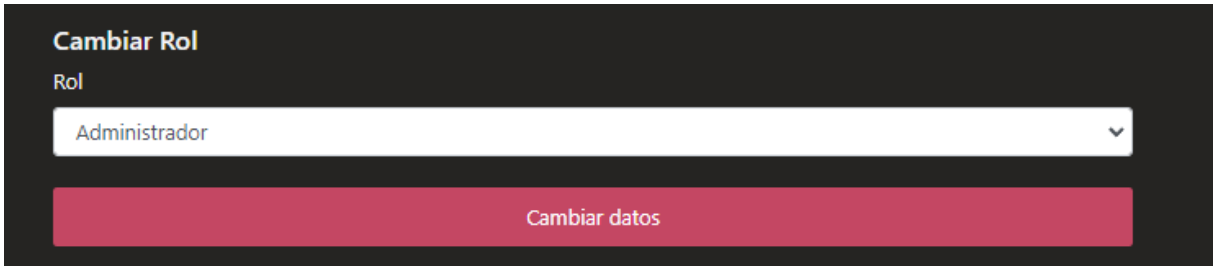

Figura Anexo 54: Formulario de datos personales (3)

En Datos personales podrá cambiar toda la información del usuario, incluido su rol dentro de la misma

En Seguridad podrá cambiar su contraseña

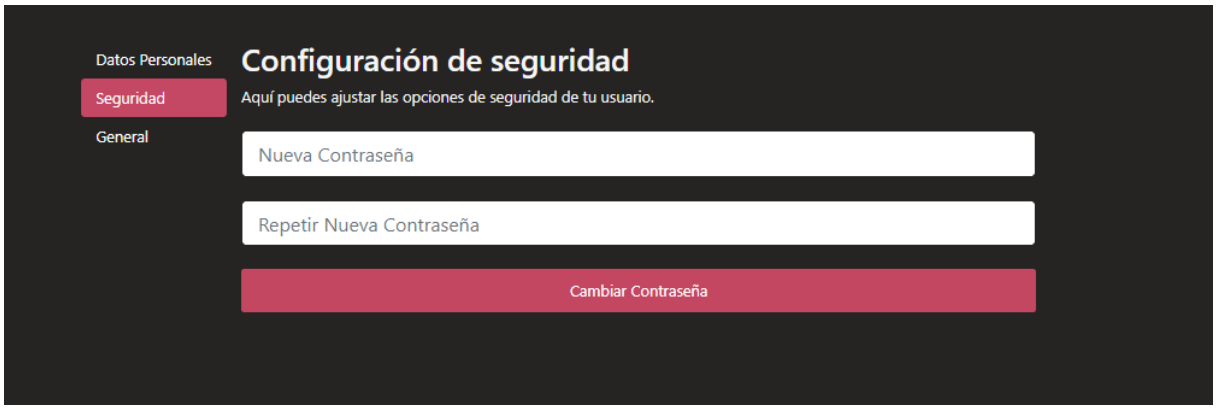

Figura Anexo 55: Formulario de seguridad

En general podrá eliminar la cuenta y cambiar su aceptación en la newsletter

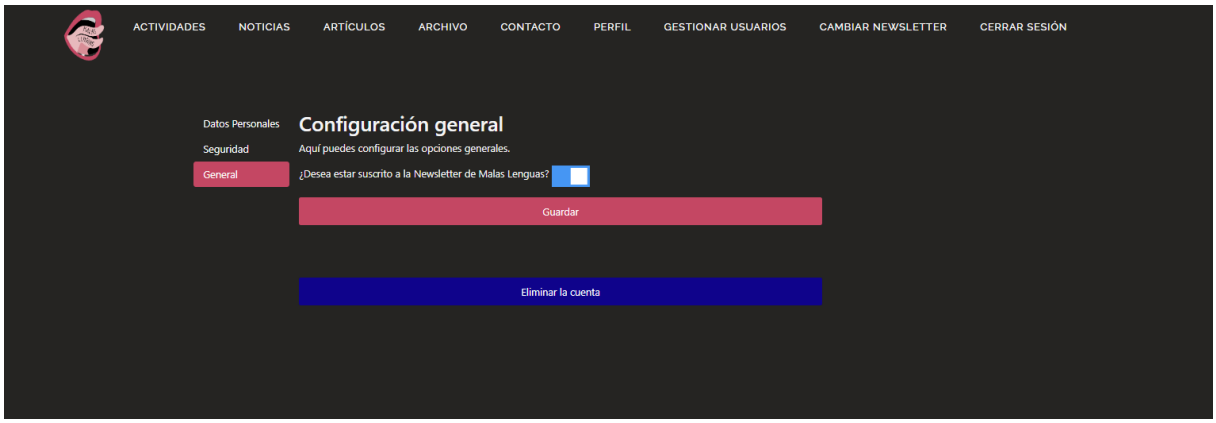

Figura Anexo 56: formulario General

Para cambiar la plantilla de la newsletter deberá acceder a la pestaña de Cambiar Nesletter

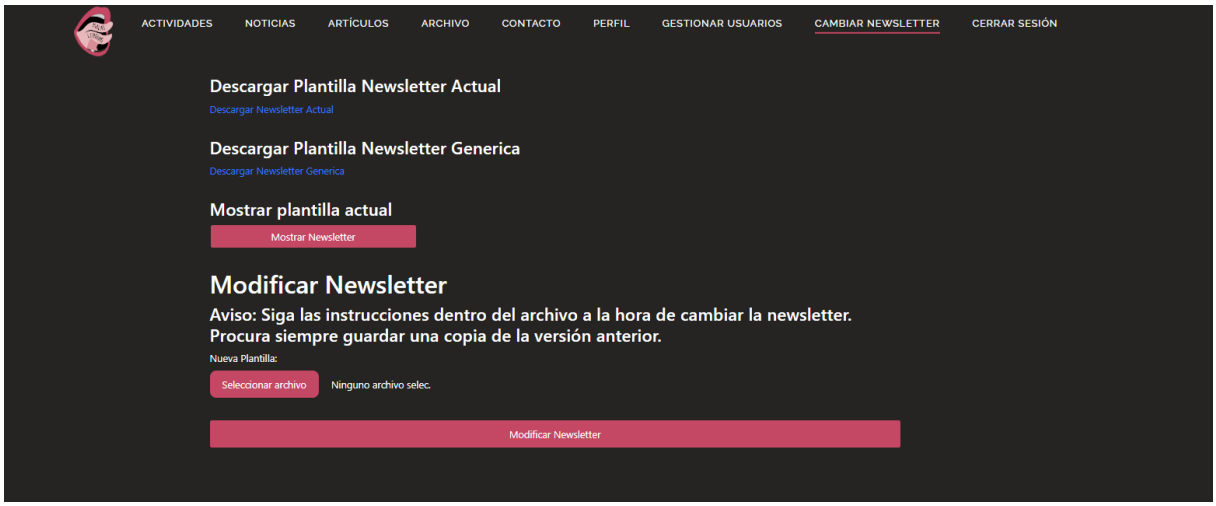

Figura Anexo 57: Vista de cambiar Newsletter

Aquí podrá descargar la plantilla actual para modificarla o insertar una propia.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S. DE INGENIERÍA

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga