TRABAJO FIN DE GRADO

INGENIERÍA INFORMÁTICA

Sahara SoundScapes

Diseño e implementación de una plataforma pública moderada para concentrar recoger, preservar y divulgar las prácticas culturales y artísticas saharauis.

Autor

Francisco José Fernández Muelas

Directores

Juan Manual Fernández Luna

Sylvia Acid Carrillo

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, Junio de 2022

Sahara SoundScapes

Diseño e implementación de una plataforma pública moderada para concentrar recoger, preservar y divulgar las prácticas culturales y artísticas saharauis.

Autor

Francisco José Fernández Muelas

Directores

Juan Manual Fernández Luna
Sylvia Acid Carrillo

Sahara SoundScapes: Diseño e implementación de una plataforma pública moderada que permita concentrar, recoger, preservar y divulgar las prácticas culturales y artísticas saharauis.

Francisco José Fernández Muelas

Palabras clave: pueblo saharaui, preservación digital, difusión web, recursos colaborativos, plataforma web.

Resumen

En este trabajo se busca dar soporte a un proyecto para la creación, recopilación, intercambio de recursos digitales de naturaleza etnográfica. El objeto central es la cultura y las expresiones artísticas saharauis. Dado el contexto geopolítico en el que se encuentra actualmente los miembros de la comunidad, esta cultura se encuentra en amenaza de desaparición. Para ayudar al encuentro, la divulgación y preservación de este acervo cultural, se ha desarrollado una plataforma web real de dominio: Sahara SoundScapes. El tipo de contenido que se pretende incluir en la plataforma ha determinado el desarrollo de la misma, tanto por el contexto político de conflicto, que supone un control y moderación de cualquier contenido antes de ser publicado, como por las modalidades de participación en la plataforma. Los usuarios potenciales son cualquier miembro de esta comunidad allá donde se encuentre y cualquier persona interesada en colaborar, compartir, consultar una colección de recursos digitales que va a ir creciendo a través del uso de la plataforma. Se trata por tanto del desarrollo de una herramienta tecnológica con aplicaciones sociales.

Sahara SoundScapes: Design and implementation of a moderated public platform to concentrate, collect, preserve and disseminate Saharawi cultural and artistic practices.

Francisco José Fernández Muelas

Keywords: saharawi people, digital preservation, web dissemination, collaborative resources, web platform.

Abstract

This work aims to support a project for the creation, collection and exchange of digital resources of ethnographic nature. The central object is the Saharawi culture and artistic expressions. Given the geopolitical context in which the members of the community currently find themselves, this culture is in danger of disappearing. To help the encounter, dissemination and preservation of this cultural heritage, a real domain web platform has been developed: Sahara SoundScapes. The type of content to be included in the platform has determined the development of the platform, both because of the political context of conflict, which implies a control and moderation of any content before being published, and because of the modalities of participation in the platform. Potential users are any member of this community wherever they are and any person interested in collaborating, sharing, consulting a collection of digital resources that will grow through the use of the platform. It is therefore the development of a technological tool with social applications.

1. Introducción

1.1 Motivación

El territorio conocido como el Sáhara Occidental es una región situada en el norte de África, en el extremo occidental del desierto del Sáhara y que es considerado por Naciones Unidas como territorio no autónomo. Desde hace muchos años este territorio se encuentra inmerso en un constante conflicto [1] debido al fallido proceso de descolonización iniciado en 1975 por España y a la reclamación del territorio y posterior ocupación ilegal por parte de Marruecos. Dicho conflicto, en el que los principales países implicados en primera línea son Marruecos, España y Argelia (debido a su histórica rivalidad con Marruecos) se extiende hasta nuestros días y no se atisba una solución definitiva.

Independientemente de las causas y motivaciones políticas de este conflicto, lo que está claro es que quien sufre y paga las consecuencias del mismo es la población saharaui. Cualquier pueblo que vea inmerso en un conflicto ve minada su calidad de vida, más aún si cabe si la región es pobre en recursos por su situación en un desierto como es el caso del Sáhara Occidental. Después de años de ocupación, conflictos armados y abandono por las instituciones internacionales, el pueblo saharaui se halla en una de las situaciones de más precariedad del mundo.

En estas situaciones, una de las vertientes que menos se tiene en cuenta y que más dañada sale, es el acervo cultural de estos pueblos pequeños. Cuando la situación social es de una emergencia tan grave como en este caso, con campos de refugiados dando asilo a miles de personas, la poca ayuda y atención internacional se centra en ayuda básica para el día a día, dejando de lado proyectos para no perder la cultura de un pueblo que se está diseminando y/o diluyendo en otras culturas.

La digitalización se está convirtiendo en una herramienta fundamental para preservar, proteger y difundir obras culturales en todo el mundo. Existen miles y miles de repositorios digitales tanto públicos como privados, replicados por toda la geografía, que permiten el acceso a cualquier persona con conexión para consultar esos recursos. Pero para tener estos repositorios es necesario un esfuerzo por digitalizar los recursos, y otro esfuerzo para crear y mantener en la red los recursos para que sean accesibles. Esto es un serio problema en pueblos donde, debido a su bajo índice de desarrollo, no ha llegado a tener ni siquiera un mínimo de digitalización.

Teniendo en cuenta que una gran cantidad de la población del Sahara Occidental (alrededor de 175.000 personas [2]) vive en campos de refugiados y que apenas cuentan con tendido eléctrico, es fácil hacerse una idea de los nulos recursos digitales a los que tienen acceso. Es cierto que las tecnologías móviles y el uso de conexión por satélite está empezando a despertar en la zona, pero esto solo ayuda a que la población empiece a tener acceso a la red. De ahí a que sea una sociedad con la madurez digital suficiente como para desarrollar y mantener sus propios sistemas de la información, y en nuestro caso, repositorios de recursos culturales, queda un largo camino a recorrer.

Una piedra más que se encuentran en este camino son las trabas que continuamente pone Marruecos, un país fuerte y con recursos que puede permitirse boicotear constantemente cualquier intento por parte de la sociedad del Sáhara Occidental de progresar y estabilizarse como entidad independiente.

Este proyecto web surge de la idea de crear una herramienta web que dé soporte a Sáhara Soundscapes [3], para concentrar, recoger, preservar y divulgar las prácticas culturales y artísticas saharauis.

Este trabajo aportará a la comunidad saharaui tener una plataforma accesible en la red de manera gratuita para compartir y consultar recursos multimedia relacionados con su cultura. La plataforma será accesible de manera sencilla desde dispositivos móviles, ya que es el único medio medianamente extendido en la población. Además, cuenta con un sistema de registro y

otro de moderación por parte de usuarios expertos, lo que ayudará a dificultar el boicot del contenido por parte de agentes malintencionados.

1.2 Problema a resolver y solución propuesta

La idea es desarrollar una aplicación web que permita el intercambio de recursos culturales saharauis, que actuará como una comunidad digital que permitirá a los usuarios publicar recursos que estén disponibles para el resto, siendo estos recursos moderados por expertos para evitar la intrusión de material no deseado. La plataforma además tendrá otras funcionalidades para darle más utilidad y un componente más social, como la posibilidad de publicar anuncios de eventos y un sistema de mensajería interno.

Una de las principales características de la plataforma será la posibilidad de consultar los recursos en diferentes mapas, dada su geolocalización, lo que permitirá a los usuarios tener una idea visual y espacial de donde están localizados los recursos, o consultar aquellos que pertenezcan a una determinada zona.

La plataforma tiene que ser capaz de permitir a los usuarios la visualización directa del contenido, por lo que, al mismo tiempo que ser un repositorio de recursos, también será un reproductor y visualizador en linea de dichos recursos.

Una de las características más importantes de este trabajo de fin de grado es que se pretende que esta aplicación sea usada en la vida real por un gran número de usuarios, por lo que sus componentes se han de desarrollar usando prácticas de desarrollo profesionales para dar soporte a todas las necesidades de una aplicación desplegada en un entorno de producción. Además, dado el contexto del dominio de la aplicación, usada por personas inmersas en un conflicto social importante, las consideraciones de seguridad son importantes, ya que la plataforma puede verse sometida a ataques y boicots por algunos de los actores del conflicto.

1.3 Objetivos

El objetivo principal de este trabajo de fin de grado, desde el punto de vista académico, es aprender y llevar a cabo todos los pasos del ciclo de vida del desarrollo web. Dentro de este objetivo general, se pueden definir otros objetivos más específicos:

1. Aprender y practicar técnicas de ingeniería de software fuera del desarrollo puro en un escenario más real que el estudiado hasta ahora.

2. Profundizar en el conocimiento de los lenguajes de programación y frameworks que se van a utilizar.

3. Aprender las metodologías del ciclo de operaciones DevOps (gestionar repositorios git, desplegar, mantener un servidor, etc) necesario para tener una aplicación en línea en el mundo real.

4. Aprender metodologías de desarrollo seguro a la hora de realizar un mecanismo de autenticación y autorización, investigando qué métodos hay y sus ventajas e inconvenientes.

Por otro lado, el objetivo principal propio de la aplicación web es crear una aplicación que gestiona una colección de recursos, permitiendo preservar y compartir los mismos y que al final la gestión devenga en la propia comunidad, que en definitiva es la última propietaria de ese acervo cultural.

Para alcanzar este objetivo principal, se proponen los siguientes objetivos específicos:

1. Diseñar interfaces de usuarios fáciles de usar para usuarios que no tengan mucha experiencia en el manejo de herramientas web.

2. Diseñar un sistema de gestión de usuarios que permita la autenticación y autorización en la plataforma.

3. Implementar la aplicación web que permita gestionar recursos multimedia por parte de los usuarios con un mecanismo de moderación, y la difusión de eventos mediante un sistema de anuncios.

4. Añadir a la aplicación web un mecanismo para permitir la comunicación entre usuarios mediante un sistema de mensajería interna.

5. Desplegar y mantener la plataforma en un servidor web, permitiendo que esté disponible online de manera robusta y segura.

1.4 Estado del arte

Actualmente no existe ningún proyecto importante de este tipo que se ocupe de los recursos culturales de la comunidad saharaui. Lo que sí que existen son repositorios de recursos culturales, tanto específicos para un determinado ámbito, como herramientas genéricas para usarlas y adaptarlas a un tema.

Estos son algunos ejemplos entre las implementaciones y plataformas específicas:

- Hispana [4]. Según la descripción en su web, Hispana es un agregador de contenido que pretende “proporcionar acceso al patrimonio cultural y científico español. Para ello, recolecta y hace accesibles los metadatos de los objetos digitales, permitiendo visualizar dichos objetos a través de enlaces que dirijan a las páginas de las instituciones propietarias”. A su vez, es el punto de enlace con Europeana, el agregador de contenido de la Unión Europea.

- Repositorio de recursos culturales del Patronato de La Alhambra [5]. Es un archivo con una gran cantidad de recursos multimedia que permite su búsqueda y consulta, pero no tiene el componente colaborativo y social que pretendemos. Además, su interfaz web no se ha adaptado a las nuevas tecnologías.

- Archivo digital del Instituto Andaluz de Patrimonio Histórico [6]. Similar al anterior, pero con la ventaja de tener una interfaz más usable.

Ninguno de estos repositorios tiene funcionalidades tan colaborativas y sociales como las que queremos para nuestra herramienta.

Por otro lado, algunos ejemplos de herramientas para crear plataformas de gestión de recursos son:

- Omeka [7]: Es una plataforma para colecciones digitales compartidas para compartir recursos multimedia.

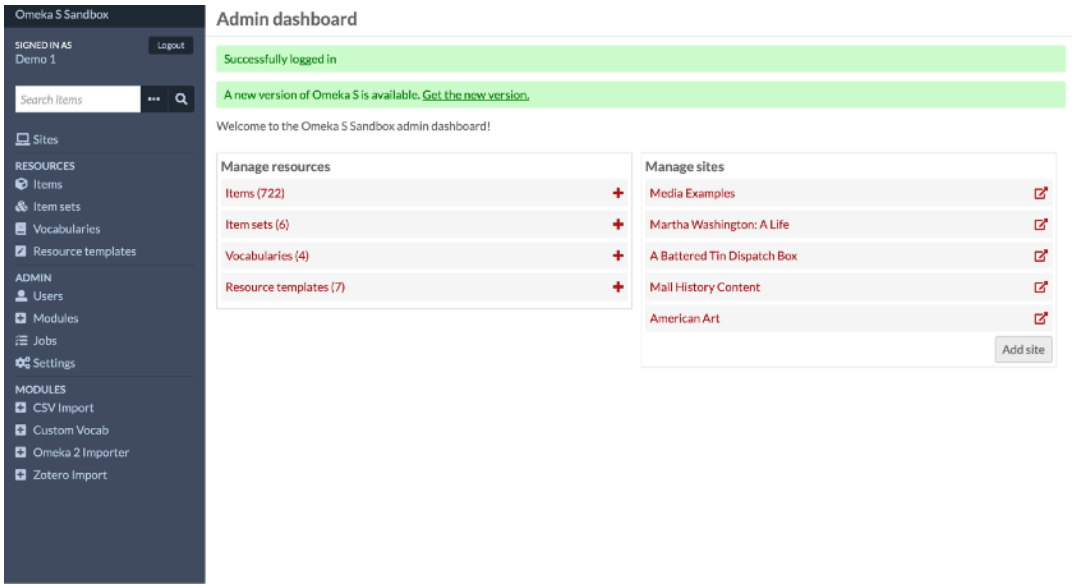


Figura 1.1: Panel de administración de Omeka

- Dspace [8]: Es un software utilizado por organizaciones para construir repositorios digitales.

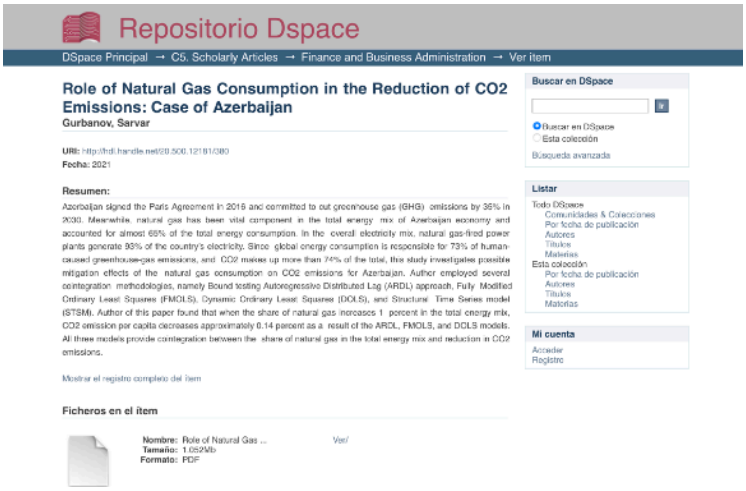


Figura 1.2: Panel de administración de Dspace

- Kune [9]: Otra herramienta similar a las dos anteriores, permite compartir recursos y realizar acciones de ámbito social (mensajes, comentarios, etc) a los usuarios.

Figura 1.1: Panel de administración de Kune

Estas 3 plataformas y otras similares tienen todas las mismas problemáticas:

- Son una herramienta cerrada, que aunque permiten cierto grado de personalización, no se adaptan de la misma manera que el desarrollo de una plataforma específica para nuestro caso.

- La interfaz no es tan amigable y usable como quisiéramos, ya que tienen un diseño similar al de una wiki o un panel de administración complejo, lo cual no sirve para el tipo de usuarios que va a tener nuestra plataforma.

- La cantidad de parámetros a configurar puede ser excesiva para un usuario administrador, el cual no tiene por qué ser experto en informática.

Vistas las alternativas, está claro que se hace necesario un desarrollo a medida desde cero para cumplir con los objetivos del proyecto.

1.5 Estructura del documento

En esta sección describiremos cómo está organizada esta memoria, la cual se compone de los siguientes capítulos:

1. Introducción. Es esta primera sección donde se da una visión general del proyecto, incluyendo sus motivaciones, objetivos y una revisión del estado del arte para ver las posibles alternativas y el por qué no son útiles.

2. Planificación y presupuesto. Se define el plan a seguir durante el desarrollo del proyecto, indicando la metodología de desarrollo que se va a seguir y dando unos estudios iniciales de presupuesto y una estimación de tiempos.

3. Análisis. En esta fase se muestra la especificación de requisitos de la plataforma, utilizando técnicas de ingeniería de software, para tener de forma conceptual todo el alcance. Para ello se definen: requisitos funcionales, no funcionales, de información, de datos y de almacenamiento; así como el estudio completo de los casos de uso.

4. Diseño. En esta sección se documenta el diseño de los diferentes componentes que forman el sistema, que va desde el diseño de la BD para el almacenamiento de la información en la plataforma hasta el esquema de las interfaces gráficas.

5. Desarrollo. La fase de desarrollar la propia aplicación, implementando el código en base a los requisitos definidos en la sección de análisis y utilizando las herramientas definidas en la fase de diseño.

6. Pruebas. Se detallan los diferentes procesos que se han realizado para evaluar la aplicación y comprobar su correcto funcionamiento.

7. Conclusiones. Se explica el resultado final, indicando qué parte de los objetivos se han cumplido, qué dificultades han aparecido y el desvío de los resultados respecto a la planificación inicial.

2. Planificación y presupuesto

A continuación se detallan las fases en las que se ha estructurado el desarrollo del proyecto:

2.1 Metodología de desarrollo

Para la realización de este trabajo he seleccionado una metodología de desarrollo tradicional o en cascada [10]. Esta metodología presenta el proceso de desarrollo de software como una secuencia de etapas claramente diferenciadas. Todas las actividades están previamente planificadas, y el paso de una etapa a la siguiente se realiza de forma secuencial. La principal ventaja de esta metodología es que los pasos que se van dando y las decisiones que se van tomando se hacen sobre seguro, reduciendo así el número de imprevistos. La etapa de implementación es mucho más cómoda si las anteriores etapas se han llevado correctamente.

El motivo por el que he seleccionado esta metodología es porque creo que se adapta mejor al tipo de trabajo que vamos a realizar. En este caso, los requisitos principales de la aplicación vienen definidos por las necesidades que plantea la digitalización de la plataforma Sahara Soundscapes, por lo que con una planificación en cascada podemos definir los “planos” de la aplicación y comprobar con los implicados que todo cumple con lo requerido antes de pasar a su implementación. También considero que esta metodología es mejor para recibir feedback de los tutores y tener un mejor control de los plazos, siendo fácil detectar desvíos.

Las fases que esta metodología plantea para dividir el proceso de desarrollo de software son:

Fase inicial

En esta fase, que en la teoría sobre la metodología en cascada se incluye dentro de la fase de análisis, hemos definido la idea de la aplicación y el plan a seguir con las partes implicadas. Este proyecto va a formar parte de una iniciativa mayor llamada Sahara Soundscapes, por lo que tiene que hacerse de acuerdo a ciertas normas y requisitos que nos marca este “cliente”. Las motivaciones, que han sido definidas en la parte de introducción, son las que marcan estas conversaciones para definir a grandes rasgos cuales van a ser los pasos a seguir. Se selecciona la metodología de desarrollo, que en este caso va a ser una metodología tradicional en cascada.

En términos generales, el objetivo que se decide es realizar una aplicación web responsive que permita gestionar de manera colaborativa los recursos y anuncios, con gestión de usuarios y mensajería interna, utilizando herramientas web modernas y con el límite de entrega el 8 de julio de 2022.

Fase de análisis

En esta fase se específica el comportamiento que se desea en la plataforma de manera minuciosa y que sirvan como especificación del sistema. Para ello, nos valemos de herramientas de ingeniería de software:

- Requisitos funcionales: las funcionalidades que debe permitir la operatividad de la aplicación y la definición de perfiles de usuarios.
- Requisitos no funcionales: las restricciones y requisitos que ha de tener la operativa de aplicación .
- Requisitos de datos, de información y de almacenamiento: una descripción del modelo de datos que se va a recoger, almacenar y mostrar.
- Modelado de casos de uso: una descripción detallada y modelada de cada una de las acciones que va a poder realizar un usuario de acuerdo al perfil de usuario al que está asignado.

Fase de diseño

En esta fase se identifican y describen las abstracciones de software fundamentales de toda la plataforma:

- Base de datos: de acuerdo con los requerimientos de los datos del sistema se identifican las entidades, atributos y relaciones y se realiza el diagrama Entidad-Relación. A partir de él se obtiene el modelo relacional de la base de datos.

- Operaciones: se diseña el comportamiento de las operaciones a realizar, agrupándolas por tipo de usuario en los diagramas de casos de usos y describiendo la secuencia de pasos que realiza cada una con los diagramas de actividad.

- Arquitectura: se define la arquitectura de la aplicación, definiendo qué componentes la forman y como interactúan entre sí. Se seleccionan las herramientas que se van a usar para el desarrollo y se justifica su elección. Los componentes serán un cliente web y API REST, que se encarga de recibir las peticiones del cliente, procesarlas (comunicándose con la base de datos si es necesario) y responder con los datos.

- Diseño de interfaces de usuario, realizando esquemas de la disposición de los componentes de datos de todas las vistas de la aplicación.

Fase de desarrollo

Una vez diseñada la aplicación se comienza con el desarrollo del código. Para una mejor organización, se definen los siguientes bloques:

- Arquitectura inicial de los dos proyectos: API REST y cliente web.

- Autenticación y gestión de usuarios.

- Gestión de recursos.

- Gestión de anuncios.

- Gestión de moderar recursos y anuncios.

- Mensajería interna.

- Panel de administración.

- Despliegue en producción.

Fase de pruebas

Cada una de estas fases no son totalmente independientes, ya que tiene al final un proceso de testing y corrección de errores, y a su vez, durante el desarrollo de una fase surgen problemas que implican hacer cambios en las especificaciones de una fase anterior.

Una vez desarrolladas todas las fases, y teniendo una aplicación 100% funcional en su primera versión se realiza una fase de pruebas generales para comprobar que todo funciona correctamente. Se arreglan los fallos encontrados, se genera una nueva versión, y se inicia el proceso de nuevo hasta que todas las pruebas resultan satisfactorias.

Fase de operaciones y mantenimiento

La plataforma es instalada en un servidor web y puesta en funcionamiento. Esta fase incluye el mantenimiento, que consiste en detectar y solucionar los errores que aparecen con el uso real de la aplicación. Esta fase se extiende más allá del ciclo de este trabajo de fin de grado.

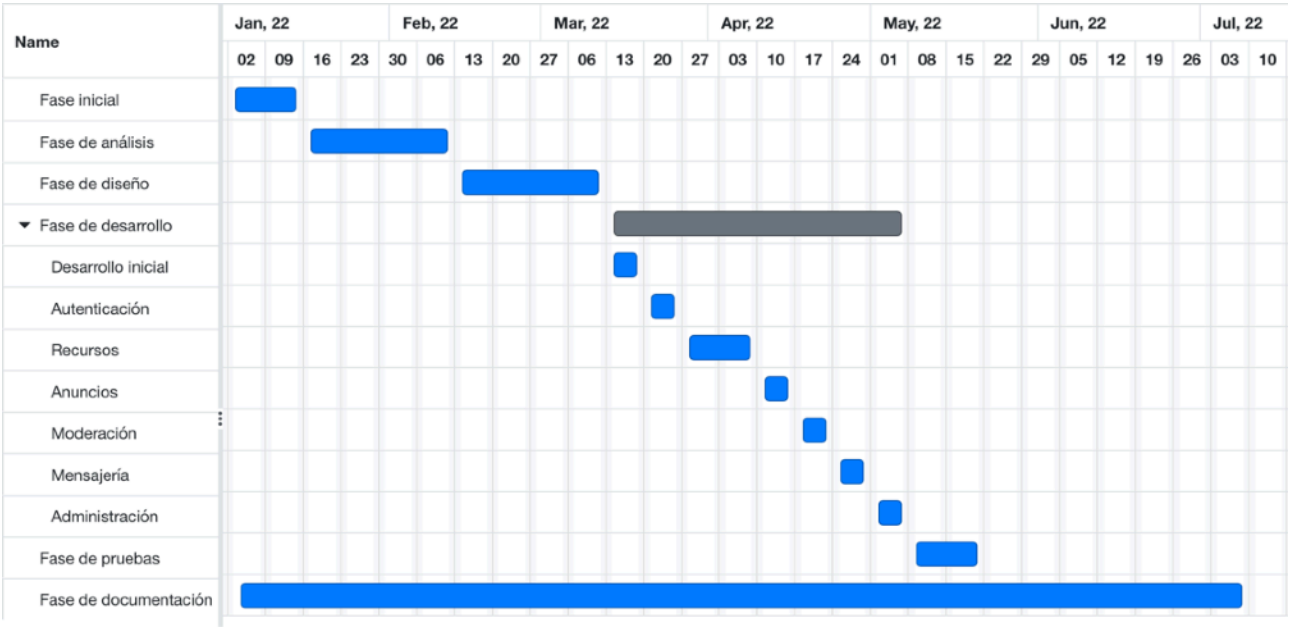
Fase final

En esta fase (que tampoco se incluye en los manuales de la metodología en cascada, pero que añadimos debido a ser un trabajo académico) se analiza el trabajo realizado desde el comienzo, para escribir unas conclusiones sobre lo aprendido. Se revisarán qué objetivos se han cumplido, cuales no y el por qué. Se definirá el futuro trabajo a realizar para mejorar la plataforma. Es también en esta fase cuando se redacta parte de la memoria, ya que se dispone de documentos resultados de las diversas fases del proyecto.

2.2 Planificación temporal

Ahora es necesario definir un plan temporal para saber cuanto tiempo dedicar a cada una de las fases descritas en el punto anterior, para poder cumplir los plazos de entrega de este trabajo. La unidad mínima de trabajo que se va a utilizar va a ser la semana de trabajo. Cada semana se puede traducir en unas 5 horas de trabajo real aproximadamente, ya que el tiempo que puedo dedicar se encuentra limitado por mi empleo actual.

Por lo tanto, se establece el siguiente diagrama de Gantt, una herramienta usada en gestión de proyectos para mostrar tareas o eventos a lo largo del tiempo [11].



En este planteamiento inicial podemos observar los siguientes detalles:

- Se inicia el plan con el inicio del año 2022, y la fecha límite de entrega es el 8 de julio. Por lo tanto, son 6 meses disponibles de trabajo.

- Se planea que cada fase empiece cuando termine la anterior, de acuerdo con el fundamento de la metodología en cascada. La excepción es la fase de documentación, que abarca desde el

inicio al final del proyecto hasta la entrega, ya que dicha documentación se irá actualizando de manera constante durante todas las fases.

- Se proyecta acabar a finales de mayo con el desarrollo y las pruebas, lo que deja más de un mes y medio de margen para cubrir desviaciones que puedan surgir.

2.3 Presupuesto

En esta sección de detalla el presupuesto previsto para el desarrollo de este trabajo, incluyendo gastos materiales, gastos de software y gastos de personal. Además se explican los gastos necesarios para mantener la plataforma operativa una vez acabado el desarrollo.

Presupuesto hardware

- Se necesita de un ordenador para el desarrollo de la aplicación. Los requisitos genéricos de un ordenador para que un trabajo de desarrollo de software, usando las diferentes herramientas necesarias se realice de una manera fluida y ágil han de ser, al menos: 16GB de RAM, un procesador moderno de una potencia similar a un Intel i7 y un almacenamiento en estado sólido de 250Gb. También es preferible que sea un ordenador portátil, para poder llevarlo a las distinta reuniones. Con estas características, el precio suele estar alrededor de los 900€.

- Para hacer más cómodo el desarrollo, se incluye una pantalla extra de 24 pulgadas y teclado y ratón externos.

Presupuesto software

- Todas las herramientas software que se van a utilizar son gratuitas. Como sistema operativo se utiliza Ubuntu 20 y como entorno de desarrollo VSCode.

Presupuesto de personal

- El salario medio en España de un Full-Stack developer senior es de 35000€ brutos anuales. Lo que se traduce en aproximadamente 18€ la hora. El proyecto se desarrolla a lo largo de 6 meses, dedicando un promedio de 5 horas semanales, por lo que el coste de personal sería de 2160€. A esta cantidad hay que añadir cuenta los gastos para una empresa de tener a un trabajador asalariado, como cotizaciones a la SS y seguros, que suelen rondar el 30% del salario bruto, o bien tener en cuenta la cuota de autónomos en caso de ser por cuenta propia.

A este presupuesto, hay que añadirle unos gastos periódicos (mensuales o anuales) de la infraestructura:

- Se necesita contratar un alojamiento VPS para desplegar la aplicación. Dado el volumen de usuarios que se espera tener, se va a contratar un VPS básico con el proveedor Ovh para dar soporte a la fase inicial del la plataforma. En caso de que el número de usuarios crezca, se pueden aumentar las capacidades del VPS. El precio del plan básico es de 3.60€ mensuales [13], que incluye 20GB de almacenamiento, lo que es suficiente para tener diferentes recursos de prueba.

- El registro del dominio. Un dominio estándar libre cuesta alrededor de 10€ anuales [14].

- De manera opcional, un servidor de almacenamiento, para guardar los recursos en caso de que el uso de la plataforma crezca mucho y se opte por la opción de almacenar todos los recursos.

En este caso, el precio depende de la cantidad de almacenamiento que se requiera, pero por poner un precio orientativo, en el proveedor OVH, el servidor de almacenamiento con 2TB de HDD y el rendimiento más bajo cuesta 57€ al mes.

Total

Concepto	Precio
Ordenador portátil (Enlace)	900 €
Pantalla (Enlace)	140 €
Ratón y teclado (Enlace)	17 €
Sueldo empleado	2160 €
Coutas por tener empleado	648 €
VPS x 6 meses	21,6 €
Registro de dominio x 1 año	10 €
Servidor de almacenamiento 2TB x 6 meses	342 €
Total	4238,6 €

2.4 Modelo de negocio

Este trabajo no se plantea como una aplicación que pueda tener un futuro comercial, ya que está hecha para dar soporte a una iniciativa benéfica sin ánimo de lucro. A pesar de esto, sí es cierto que va a tener unos gastos operativos para mantener en la aplicación en línea. Para ello se puede optar entre una o varias de estas opciones:

- Donaciones por parte de particulares, ya sean usuarios de la plataforma o personas que quieran aportar dinero para el mantenimiento.

- Soporte económico por parte de entidades como ONGs, colectivos sociales, empresas, etc.

- Ayudas por parte del estado o instituciones públicas, ya sea en forma de ayuda directa o a través de fondos o becas.

3. Análisis

En esta sección se describe análisis realizado para desarrollar la aplicación.

3.1 Actores y entidades

Seguidamente pasamos a relacionar las entidades principales del proyecto como son los actores, recursos, anuncios etc. El propósito es dotarlas de una semántica y contextualizarlas en el proyecto con su funcionalidades específicas.

3.1.1 Entidades

E1. Usuario. Representa a un usuario de la plataforma.
Atributos: usuario, contraseña (hash), nombre, apellidos, correo electrónico

E2. Rol. Representa uno de los roles disponibles en el sistema: miembro, artista, experto y administrador.
Atributos: identificador, etiqueta

E3. Permiso. Representa cada uno de los permisos que se utilizarán para la autorización en el sistema.
Atributos: identificador, etiqueta

E4. Recurso. Representa los recursos subidos a la plataforma por los usuarios.
Atributos: identificador, tipo multimedia, título, palabras clave, descripción, fuente, tipo de recurso, cobertura, autor, editor, derechos, fecha, formato, URL

E5. Anuncio. Representa un anuncio creado por un usuario en la plataforma.
Atributos: fecha, título, contenido.

E6. Mensaje. Representa un mensaje interno de la plataforma.
Atributos: asunto, contenido, fecha.

3.1.2 Actores

Dentro de las entidad de usuarios, encontramos los actores, que se pueden definir como la abstracción de entidades externas al sistema que interactúan directamente con este.

A1. Invitado. Representa a un usuario que accede a la plataforma sin identificarse para iniciar sesión, registrarse o recuperar la contraseña.

A2. Miembro. Representa a un usuario que está registrado en la plataforma y accede identificándose, pudiendo añadir contenido moderado, esto es, requiere la intervención de un experto que hace de moderador para validar, que el recurso propuesto, antes de ser publicado, es adecuado con el propósito y contenido de la plataforma y no se hace un uso inadecuado de la misma.

A3. Artista. Representa a un usuario que está registrado en la plataforma y accede identificándose, pudiendo añadir contenido sin necesidad de moderación para ser publicado. Esto es así ya que, el artista suele crear nuevo contenido y puede utilizar la plataforma para la divulgación de su material.

A4. Experto. Representa a un usuario que está registrado en la plataforma y accede identificándose, pudiendo añadir contenido sin necesidad de moderación y que puede moderar las propuestas de usuarios miembros y excepcionalmente artistas y expertos.

A5. Administrador. Representa a un usuario que está registrado en la plataforma y accede identificándose, pudiendo realizar tareas de administración, gestión de usuarios y revisión de contenido de la plataforma.

3.2 Especificación de requisitos

Los requisitos son las condiciones de diferente naturaleza que la plataforma ha de contemplar para cumplir con los objetivos establecidos.

3.2.1 Requisitos de datos

Los requisitos de datos son los atributos de cada entidad que se introducen en el sistema para caracterizar una instancia de dicha entidad. Hemos definido los siguientes:

RD1. Datos de usuario:

- Usuario
- Contraseña
- Nombre
- Apellidos
- Email

RD2. Rol. Datos de un rol:

- Etiqueta

RD3. Permiso. Datos de un permiso:

- Etiqueta

RD4. Recurso. El sistema de metadatos de partida que utilizaremos para los recursos será el modelo de Dublin Core. Es un sistema de 15 definiciones semánticas descriptivas que pretenden transmitir un significado semántico a las mismas. Este sistema de definiciones fue diseñado específicamente para proporcionar un vocabulario de características 'base', capaces de proporcionar la información descriptiva básica sobre cualquier recurso, sin que importe el formato de origen, el área de especialización o el origen cultural [15]. Se ha eliminado Relación (posibles relaciones entre recursos) por irrelevante y confuso para un Colaborador.
Datos de un recurso:

- Tipo Multimedia
- Título
- Claves
- Descripción
- Fuente
- Tipo Recurso
- Cobertura
- Autor
- Editor
- Derechos
- Fecha
- Formato
- Identificador (URL)
- Coordenadas

RD5. Anuncio. Se trata de un texto que comunica un evento de interés para la comunidad y va a ser publicado durante periodo de tiempo entre la fecha de inicio y de fin. Datos de un anuncio:

- Fecha inicio
- Fecha fin
- Título
- Contenido

RD6. Mensaje. La plataforma es un lugar de encuentro donde artista divulgan sus trabajos o personas pueden requerir contacto, por lo que existe la necesidad de incluir un sistema de mensajería interna. Esta entidad Mensaje representa un mensaje con información que un usuario envía a otro usuario de la plataforma. Datos de un mensaje:

- Asunto
- Contenido

3.2.2 Requisitos de almacenamiento

Los requisitos de almacenamiento son los atributos de cada entidad que el sistema almacenará de manera esquemática en la base de datos para su procesamiento en la lógica de negocio, incluyendo identificadores únicos.

RA1. Datos de usuario:

- ID_usuario
- Usuario
- Hash
- Nombre
- Apellidos
- Email
- Fecha Alta

RA2. Rol. Datos de un rol:

- ID_rol
- Etiqueta

RA3. Permiso. Datos de un permiso:

- ID_permiso
- Etiqueta

RA4. Recurso. Son los originales del estándar DublinCore al que se ha añadido ID_recurso, y tema. ID_recurso es indispensable para la recuperación de cualquier recurso de la BD de forma única. Y tema es un campo con 6 valores: *Cultura, Territorios, Población, Historia, Política y Economía, Conflicto.* Nos servirá para organizar los recursos y realizar búsquedas selectivas.

Datos de un recurso:

- ID_recurso
- Tipo Multimedia
- Título
- Tema
- Descripción
- Fuente
- Tipo Recurso
- Cobertura
- Autor
- Editor
- Derechos
- Fecha
- Formato

- Identificador (URL)
- Coordenadas

RA5. Anuncio. Datos de un anuncio:

- ID_anuncio
- Fecha inicio
- Fecha fin
- Título
- Contenido

RA6. Mensaje. Datos de un mensaje:

- ID_mensaje
- Asunto
- Contenido
- Fecha

3.2.3 Requisitos de información

Los requisitos de información son los atributos de las entidades necesarios para representarlos en las operaciones que requieren interacción con el usuario, por ejemplo, al mostrarse por pantalla. Son necesarios para describir la disposición de los elementos en las interfaces de usuario o para definir los atributos de los objetos usados en la comunicación entre cliente y servidor.

RI1. Datos de usuario.

- Usuario
- Nombre
- Apellidos
- Email
- Fecha Alta

RI2. Rol. Datos de un rol

- Etiqueta

RI3. Permiso. Datos de un permiso

- Etiqueta

RI4. Recurso. Datos de un recurso

- Tipo Multimedia
- Título
- Claves
- Descripción
- Fuente
- Tipo Recurso
- Cobertura
- Coordenadas
- Autor
- Editor
- Derechos
- Fecha
- Formato
- Identificador (URL)

RI5. Anuncio. Datos de un anuncio

- Fecha inicio
- Fecha fin
- Título
- Contenido

RI6. Mensaje. Datos de un mensaje

- Asunto
- Contenido
- Fecha

3.2.4 Requisitos funcionales

Los requisitos funcionales son los que definen el comportamiento de las operaciones que ha de permitir realizar a un usuario la plataforma, y posteriormente se utilizaran para definir los casos de uso.

RF1. Acceder a la plataforma. El usuario introduce su usuario y contraseña, se valida y accede a la plataforma con el rol que tenga asignado.

RF2. Buscar recursos. Un usuario de cualquier tipo podrá utilizar un formulario de búsqueda para localizar recursos. Podrá ser una búsqueda simple o una búsqueda avanzada por metadatos. Los resultados podrán mostrarse en un listado o geolocalizados en un mapa.

RF3. Explorar recursos en mapa. Un usuario de cualquier tipo podrá explorar todos los recursos disponibles, que aparecerán en un listado y geolocalizados en un mapa.

RF4. Visualizar recurso. Un usuario de cualquier tipo podrá seleccionar un recurso para su visualización. Dependiendo del recurso (texto, audio, vídeo, etc.), aparecerá un tipo de vista u otro.

RF5. Solicitar registro. Un usuario no registrado (invitado) podrá solicitar el registro en la plataforma como Miembro rellenado un formulario que deberá ser aprobado por un Administrador.

RF6. Consultar perfil. Un usuario registrado (miembro, artista, experto, administrador) podrá consultar sus datos personales e información sobre su perfil.

RF7. Editar perfil. Un usuario registrado (miembro, artista, experto, administrador) podrá editar ciertos datos personales de su perfil, mediante un formulario.

RF8. Exportar recurso. Un usuario de cualquier tipo podrá descargar uno o más recursos de los disponibles en la plataforma si los derechos del recurso lo permiten. Existe la posibilidad de descargarlos como archivo comprimido (.zip, .tar)

RF9. Proponer recurso. Un usuario con rol miembro podrá subir un recurso a la plataforma. Para ello, se muestra un formulario para añadir los metadatos, algunos de los cuales vendrán rellenos con valores por defecto por la plataforma para facilitar su uso. Este recurso quedará pendiente de validación por un usuario Experto.

RF10. Editar recurso propuesto. Un usuario miembro podrá editar uno de sus recursos propuestos mientras no se encuentre publicado.

RF11. Proponer edición de recurso aprobado. Un usuario miembro podrá proponer la edición de uno de sus recursos aprobados. Dicha edición deberá ser aprobada por un usuario experto.

RF12. Eliminar recurso propio. Un usuario registrado (miembro, artista o experto) podrá eliminar de la plataforma un recurso que le pertenezca. Se informará explícitamente de que dicho recurso dejará de estar permanentemente en el sistema.

RF13. Proponer anuncio. Un usuario miembro podrá crear un anuncio con información relevante que será mostrado al resto de usuarios. El mensaje quedará pendiente de validación por un usuario experto.

RF14. Editar anuncio propuesto. Un usuario miembro podrá editar uno de sus anuncio propuestos mientras no se encuentre aprobado.

RF15. Proponer edición de anuncio aprobado. Un usuario miembro podrá proponer la edición de uno de sus anuncios aprobados. Dicha edición deberá ser aprobada por un usuario experto.

RF16. Eliminar anuncio propio. Un usuario registrado (miembro, artista, experto) podrá eliminar de la plataforma un anuncio que le pertenezca.

RF17. Solicitar borrado de cuenta. El usuario podrá solicitar de manera automática el borrado de su cuenta, que será aprobada por un Administrador. Se le preguntará si quiere donar los recursos, en caso afirmativo, se asignaran al usuario experto que tiene asignado ese usuario.

RF18. Insertar recurso. Un usuario artista o experto podrá subir un recurso a la plataforma. Para ello, se muestra un formulario para añadir los metadatos, que vendrá rellenado con valores por defecto para facilitar su eso. Este recurso quedará publicado inmediatamente, sin necesidad de validación por otro usuario.

RF19. Editar recurso. Un usuario artista o experto podrá editar determinados metadatos de uno de sus recursos. Los cambios quedarán publicados inmediatamente, sin necesidad de validación por otro usuario.

RF20. Insertar anuncio. Un usuario artista o experto podrá crear un anuncio con información relevante que será mostrado al resto de usuarios. El mensaje quedará publicado inmediatamente, sin necesidad de validación por otro usuario.

RF21. Editar anuncio. Un usuario artista o experto podrá editar uno de sus anuncios. Los cambios quedarán publicados inmediatamente, sin necesidad de validación por otro usuario.

RF22. Editar cualquier recurso. Un usuario experto podrá editar cualquier recurso de la plataforma.

RF23. Eliminar cualquier recurso. Un usuario experto podrá eliminar cualquier recurso de la plataforma.

RF24. Editar cualquier anuncio. Un usuario experto podrá editar cualquier anuncio de la plataforma.

RF25. Eliminar cualquier anuncio. Un usuario experto podrá eliminar cualquier anuncio de la plataforma.

RF26. Aceptar recurso propuesto. Un usuario experto podrá aceptar un recurso pendiente de moderación. Una vez aceptado, el recurso pasa a estado publicado.

RF27. Aceptar anuncio propuesto. Un usuario experto podrá aceptar un anuncio pendiente de moderación. Una vez aceptado, el anuncio pasa a estado publicado.

RF28. Rechazar recurso propuesto. Un usuario experto podrá rechazar un recurso pendiente de moderación. Podrá añadir un comentario sobre el motivo del rechazo.

RF29. Rechazar anuncio propuesto. Un usuario experto podrá rechazar un anuncio pendiente de moderación. Podrá añadir un comentario sobre el motivo del rechazo.

RF30. Aceptar registro. Un usuario administrador podrá validar una solicitud de registro de usuario (miembro o artista).

RF31. Editar usuario. Un usuario administrador podrá editar los datos de cualquier otro usuario (no administrador) del sistema.

RF32. Eliminar usuario. Un usuario administrador podrá eliminar cualquier otro usuario (no administrador) del sistema por encontrar que un usuario hace un uso inadecuado de la plataforma, dado el contexto internacional de la misma. Los recursos de este usuario dejarán de estar en estado publicado y pasarán a estar pendientes de moderación.

RF33. Limitar subida de recurso. Un usuario administrador podrá limitar la subida de recursos de gran tamaño, permitiendo solo añadir enlaces a otras plataformas.

RF34. Consultar mensajes. El usuario podrá acceder a un listado de los mensajes recibidos y enviados por mensajería directa. Podrá filtrarlos por un cuadro de búsqueda.

RF35. Ver mensaje. El usuario podrá ver el contenido de un mensaje recibido o enviado.

RF36. Enviar mensaje. El usuario podrá redactar un nuevo mensaje y enviarlo a un determinado usuario, seleccionándolo de un listado donde aparecen todos los usuarios del sistema.

RF37. Eliminar mensaje. El usuario podrá eliminar de su cuenta un mensaje enviado o recibido que tenga almacenado.

RF38. Bloquear usuario en mensajería. El usuario podrá seleccionar un usuario de la lista de usuarios para bloquearlo y dejar de recibir sus mensajes.

RF39. Solicitar paso a artista. Un usuario miembro podrá solicitar el paso a Artista para así poder publicar recursos sin tener que pasar por moderación.

RF40. Moderar solicitud de paso a artista. Un usuario Experto podrá aceptar o rechazar una solicitud de paso a Artista hecha por un usuario Miembro.

3.2.5 Requisitos no funcionales

Los requisitos funcionales son aquellos que se refieren a restricciones o necesidades para que las operaciones de la plataforma se lleven a cabo con éxito. Para estos requisitos es necesario añadir, en la medida de lo posible, detalles (como tiempos o tecnologías concretas) para evitar hacerlos abstractos y generalistas, facilitando así su comprobación.

Contenido

RNF1. Dado el contexto sociopolítico en el que se inserta la cultura saharaui, la plataforma ha de permitir velar por que el contenido publicado sea escrupulosamente adecuado con su vocación, la de recoger el acervo cultural saharaui, y darle protagonismo a la mujer.

Usabilidad

RNF2. La plataforma ha de tener una interfaz gráfica sencilla, usable para usuarios inexpertos en el uso de webs. El tiempo de aprendizaje ha de ser inferior a 4 horas.

RNF3. La interfaz gráfica ha de ser responsive y compatible con Google Chrome, Mozilla Firefox, Safari, etc.

RNF4. La plataforma mostrará al usuario mensajes de confirmación y de error en cada operación que sean fácilmente interpretables.

RNF5. La plataforma estará disponible en idioma español

Eficiencia

RNF6. La plataforma tiene que ser liviana para ser accesible en entornos con conectividad limitada. No puede tardar más de 10 segundos en cargar toda la interfaz con una conexión 3G.

RNF7. Toda funcionalidad del sistema (exceptuando la carga de nuevos recursos) debe responder al usuario en menos de 5 segundos.

Escalabilidad

RNF8. La arquitectura de la plataforma ha de permitir la escalabilidad vertical y horizontal, para asegurar el servicio en caso de un aumento de usuarios.

Seguridad

RNF9. Las distintas funcionalidades de la plataforma estarán restringidas mediante permisos, que estarán asociados a determinados roles.

RNF10. La plataforma dispondrá de un sistema de copias de seguridad periódicas y redundantes para no perder información.

RNF11. La comunicación entre cliente-servidor se realizará usando el estándar de cifrado HTTPS.

RNF12. La plataforma contará con mecanismos de detección de ataques, permitiendo parar el servicio para mitigar y reparar los daños.

RNF13. El tratamiento de los datos de la plataforma se realizará bajo las directrices del Reglamento general de protección de datos de la UE.

Otros

RNF14. La plataforma debe contar con un manual de usuario accesible, estructurado y sencillo.

3.3 Casos de uso

Los casos de uso son una descripción de todas y cada una de las operaciones que un usuario puede realizar con la plataforma. Se diferencia de los requisitos funcionales en que uno de ellos puede generar uno o más casos de uso, ya que los requisitos funcionales son las características o funciones que deber permitir la plataforma. Los casos de uso pueden ser representados de forma sencilla y esquemática mediante un diagrama de casos de uso o de forma más detallada e individual mediante la especificación de caso de uso [16]

3.3.1 Diagramas de casos de uso

Representan las operaciones que los usuarios pueden hacer, agrupadas por tipo de actor. Una misma operación puede aparecer en uno o más actores.

Invitado

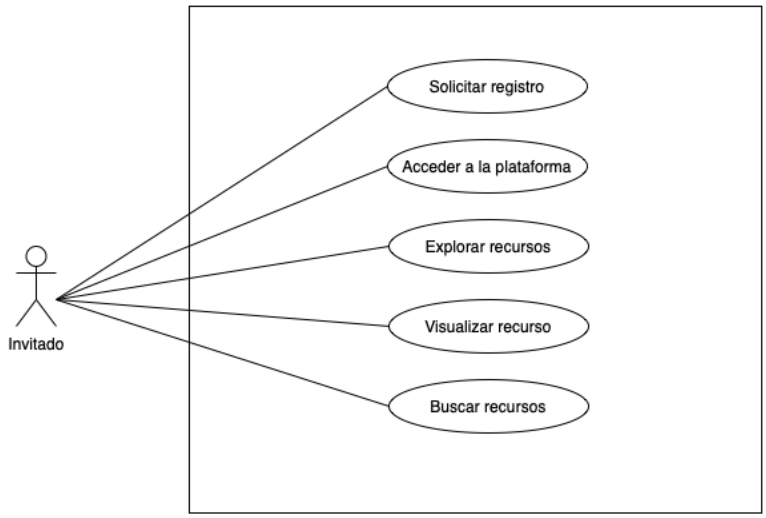


Figura 3.1: Diagrama de Casos de uso de rol Invitado

Miembro

Figura 3.2: Diagrama de Casos de uso de rol Miembro (parte 1)

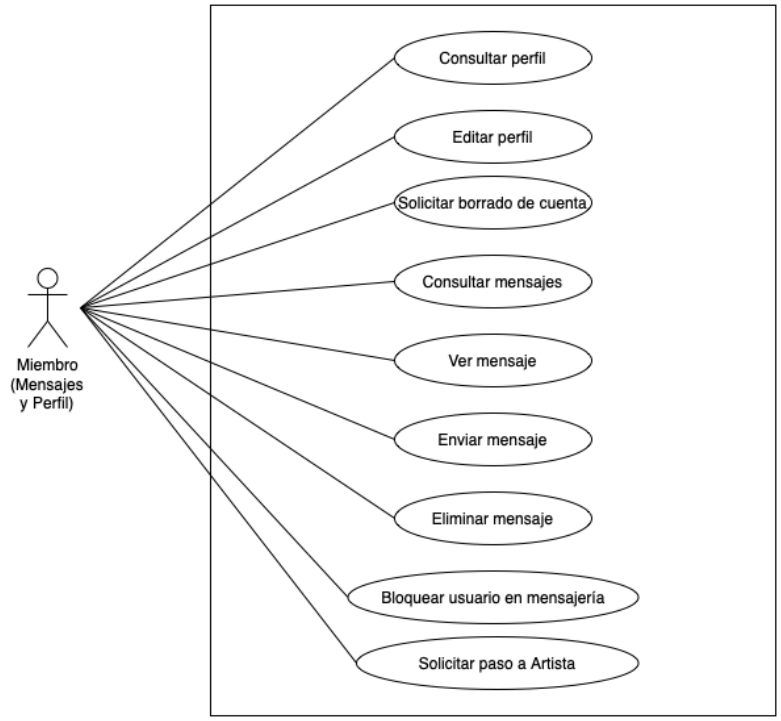


Figura 3.3: Diagrama de Casos de uso de rol Miembro (parte 2)

Artista

Figura 3.4: Diagrama de Casos de uso de rol Artista (parte 1)

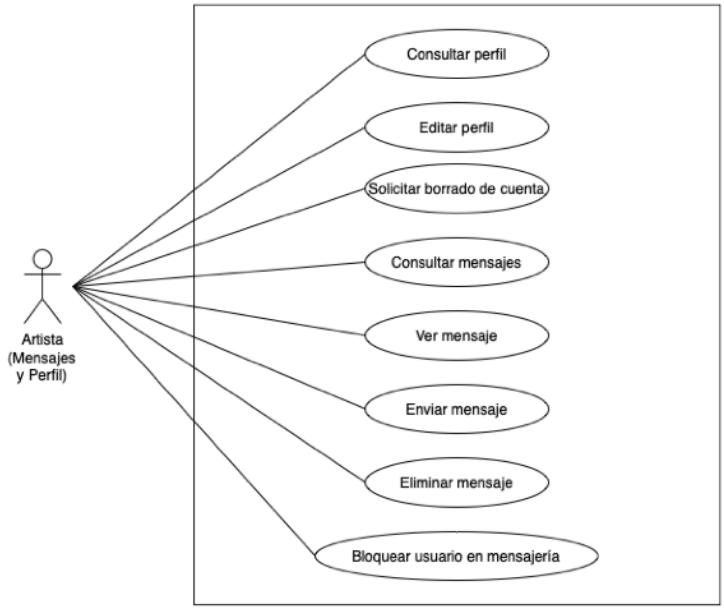


Figura 3.5: Diagrama de Casos de uso de rol Artista (parte 2)

Experto

Figura 3.6: Diagrama de Casos de uso de rol Experto (parte 1)

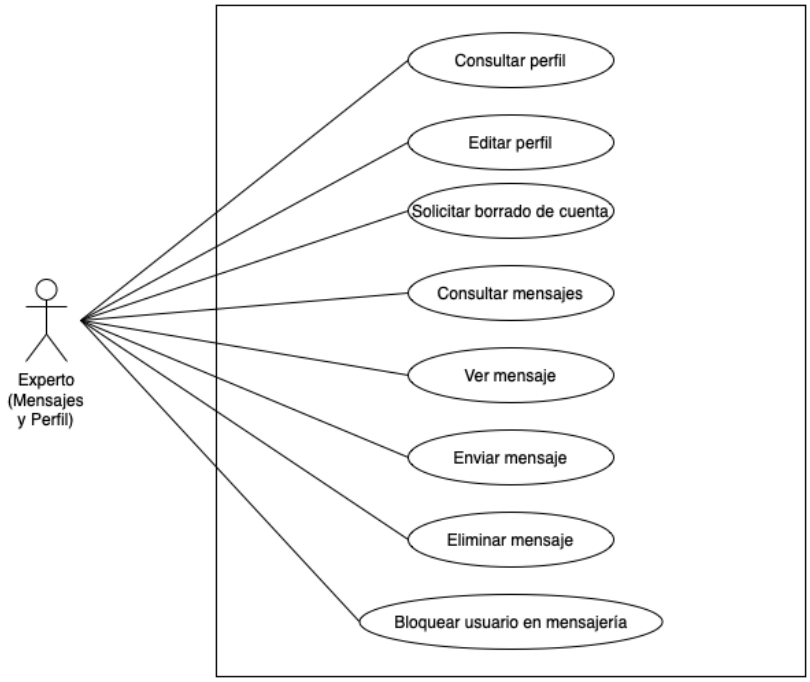


Figura 3.7: Diagrama de Casos de uso de rol Experto (parte 2)

Administrador

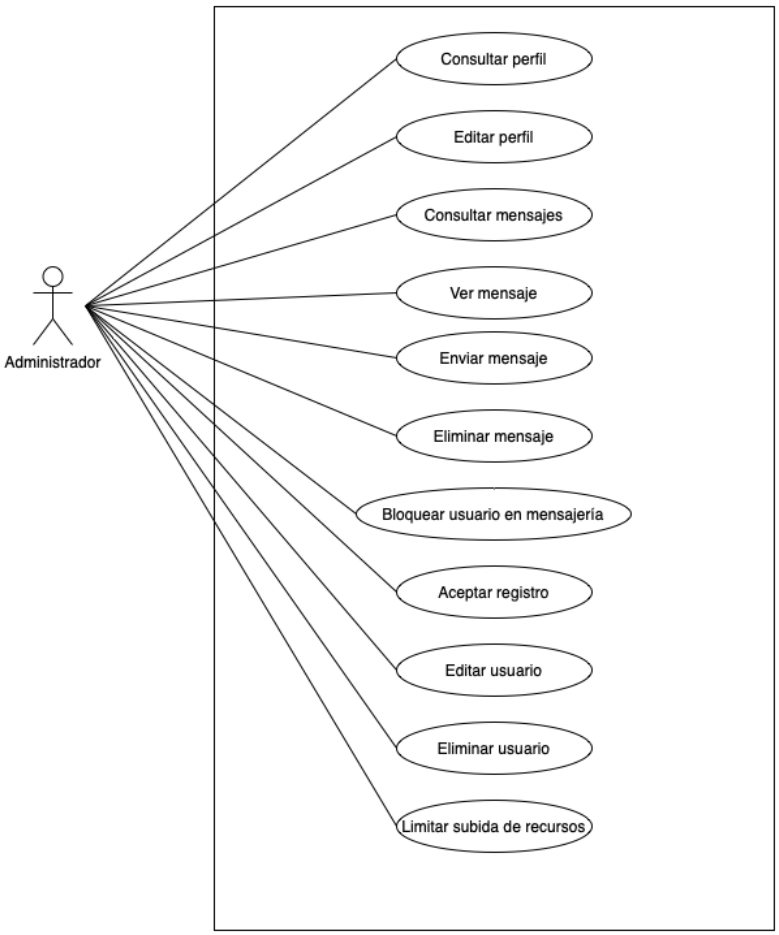


Figura 3.8: Diagrama de Casos de uso de rol Administrador

3.3.2 Especificación de casos de uso

En esta sección se detallan cada uno de los casos de uso nombrados en la sección anterior, utilizando una plantilla estandarizada para indicar los actores que intervienen y una descripción de la secuencia de acciones que se lleva a cabo durante la operación.

Caso de uso	**Acceder a la plataforma**
Actores	Invitado
Tipo	Primario, obligatorio
Precondición	El usuario no tiene una sesión abierta en la plataforma. El usuario está registrado en el sistema
Postcondición	Se inicia una nueva sesión para ese usuario en la plataforma
Requisitos de información	-
Requisitos de datos	RD1
Requisitos de almacenamiento	RA1
Referencias	RF1
Propósito	Permitir a un usuario su identificación como Miembro, Artista, Experto o Administrador
Resumen	El usuario introduce los datos de acceso (usuario y contraseña) en un formulario. El sistema valida los datos, y en caso de éxito, inicia una nueva sesión en el sistema para ese usuario y le mostrará la pantalla de inicio

Caso de uso	**Buscar recursos**
Actores	Invitado, Miembro, Artista, Experto
Tipo	Primario, obligatorio
Precondición	-
Postcondición	Se mostrará una pantalla con los resultados obtenidos
Requisitos de información	RI4
Requisitos de datos	RD4
Requisitos de almacenamiento	RA4
Referencias	RF2
Propósito	Permitir al usuario realizar búsquedas de recursos a partir de ciertos valores
Resumen	El usuario introduce los valores en los campos por los que quiere filtrar. El sistema realiza una búsqueda de aquellos recursos que coinciden con los valores proporcionados y muestra en una pantalla los resultados. Dicha pantalla puede ser en formato lista o en formato mapa

Caso de uso	Explorar recursos en mapa
Actores	Invitado, Miembro, Artista, Experto
Tipo	Primario, obligatorio
Precondición	-
Postcondición	Se mostrará una pantalla con los resultados obtenidos
Requisitos de información	RI4
Requisitos de datos	-
Requisitos de almacenamiento	-
Referencias	RF3
Propósito	Permitir al usuario explorar recursos directamente sobre un mapa
Resumen	El usuario podrá interactuar con un mapa, donde se mostraran iconos por cada uno de los recursos. El usuario podrá filtrar qué tipos de recursos quiere ver, hacer zoom en un área del mapa y consultar los recursos mostrados de manera individual

Caso de uso	Visualizar recurso
Actores	Invitado, Miembro, Artista, Experto
Tipo	Primario, obligatorio
Precondición	-
Postcondición	Se mostrará una pantalla con la información detalla de un recurso y su contenido
Requisitos de información	RI4
Requisitos de datos	-
Requisitos de almacenamiento	RA4
Referencias	RF4
Propósito	Permitir al usuario ver información detalla de un recurso y ver su contenido
Resumen	El usuario selecciona un recurso para verlo en detalle. El sistema muestra una pantalla con todos los metadatos del recurso y un panel para visualizar el contenido. Dependiendo del tipo multimedia del archivo, dicho panel variará entre: un reproductor de audio, un reproductor de video, un visualizador de texto y un visualizador de imágenes (estándar y 360º)

Caso de uso	Solicitar registro
Actores	Invitado
Tipo	Primario, obligatorio
Precondición	El usuario ha de tener una sesión activa en la plataforma
Postcondición	Se mostrará una pantalla con los resultados obtenidos
Requisitos de información	-
Requisitos de datos	RD4
Requisitos de almacenamiento	RA4
Referencias	RF5
Propósito	Permitir al usuario realizar búsquedas de recursos a partir de ciertos valores.
Resumen	El usuario introduce los valores en los campos por los que quiere filtrar. El sistema realiza una búsqueda de aquellos recursos que coinciden con los valores proporcionados y muestra en una pantalla los resultados. Dicha pantalla puede ser en formato lista o en formato mapa

Caso de uso	Consultar perfil
Actores	Miembro, Artista, Experto, Administrador
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	Se mostrará una pantalla con la información del perfil del usuario.
Requisitos de información	RI1
Requisitos de datos	-
Requisitos de almacenamiento	RA1
Referencias	RF6
Propósito	Permitir al usuario ver la información que el sistema tiene almacenado sobre su perfil
Resumen	El usuario accede a la vista de perfil, donde se mostrará un formulario en modo lectura con los campos que la plataforma tiene almacenada sobre su perfil

Caso de uso	Editar perfil
Actores	Miembro, Artista, Experto, Administrador
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	Los datos del perfil del usuario serán actualizados en base de datos con los nuevos valores
Requisitos de información	RI1
Requisitos de datos	RD1
Requisitos de almacenamiento	RA1
Referencias	RF7
Propósito	Permitir al usuario editar la información que el sistema tiene almacenado sobre su perfil
Resumen	El usuario accede a la vista de perfil, donde se mostrará un formulario en modo edición con los campos que la plataforma tiene almacenada sobre su perfil y que son editables

Caso de uso	Exportar recurso
Actores	Miembro, Artista, Experto, Administrador
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema. El usuario se encuentra en la vista de visualización de un recurso
Postcondición	Se inicia una descarga en el navegador web con un archivo comprimido que contiene el recurso seleccionado
Requisitos de información	RI4
Requisitos de datos	-
Requisitos de almacenamiento	RA4
Referencias	RF8
Propósito	Permitir al usuario descargar un recurso en formato comprimido
Resumen	Desde la vista de visualización de un recurso, el usuario selecciona exportar el recurso. El sistema genera un archivo comprimido en formato ZIP e inicia una descarga en el navegador del usuario

Caso de uso	Proponer recurso
Actores	Miembro
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	Se almacenará el recurso y sus metadatos en el sistema, en estado pendiente de moderación
Requisitos de información	-
Requisitos de datos	RD4
Requisitos de almacenamiento	RA4
Propósito	Permitir al usuario de tipo miembro proponer un recurso en la plataforma, para su posterior publicación
Resumen	El usuario accede a la vista de proponer recurso, donde aparecerá un formulario (relleno con valores por defecto en base al perfil del usuario) con los campos a establecer para proponer un recurso. El sistema valida la información y la almacena, pasando el recurso a estado pendiente de validación por un moderador

Caso de uso	Editar recurso propuesto
Actores	Miembro
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema. El recurso que se quiere editar no puede estar publicado (aprobado por un Moderador)
Postcondición	Se almacenarán los nuevos valores del recurso en el sistema
Requisitos de información	RI4
Requisitos de datos	RD4
Requisitos de almacenamiento	RA4
Referencias	RF10
Propósito	Permitir al usuario de tipo miembro editar uno de sus recursos propuestos antes de que sea validado
Resumen	El usuario accede a la vista de edición del recurso propuesto e introduce en el formulario los nuevos valores. El sistema valida la nueva información y la almacena

Caso de uso	Proponer edición de recurso aprobado
Actores	Miembro
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema. El recurso que se quiere editar ha de estar publicado (aprobado por un Moderador)
Postcondición	Se genera una solicitud de edición de recurso que deberá ser aprobada por un usuario Moderador
Requisitos de información	RI4
Requisitos de datos	RD4
Requisitos de almacenamiento	RA4
Referencias	RF11
Propósito	Permitir al usuario de tipo miembro editar uno de sus recursos propuestos una vez que ya ha sido publicado
Resumen	El usuario accede a la vista de edición del recurso publicado e introduce en el formulario los nuevos valores. El sistema valida la nueva información, la almacena, pasa el recurso a estado no publicado para que sea aprobado por un usuario Moderador

Caso de uso	Eliminar recurso propio
Actores	Miembro, Artista, Experto
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	El recurso se elimina del sistema
Requisitos de información	RI4
Requisitos de datos	-
Requisitos de almacenamiento	RI4
Referencias	RF12
Propósito	Permitir a un usuario eliminar uno de sus recursos, independientemente de que esté publicado o no
Resumen	El usuario selecciona eliminar uno de sus recursos de un listado. Se le pide confirmación de la acción y en caso afirmativo, se elimina el recurso del sistema

Caso de uso	Proponer anuncio
Actores	Miembro
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	Se almacenará el anuncio en el sistema, en estado pendiente de moderación
Requisitos de información	-
Requisitos de datos	RD5
Requisitos de almacenamiento	RA5
Referencias	RF13
Propósito	Permitir al usuario de tipo miembro proponer un anuncio en la plataforma, para su posterior publicación
Resumen	El usuario accede a la vista de proponer anuncio, donde aparecerá un formulario con los campos a establecer para proponer un anuncio. El sistema valida la información y la almacena, pasando el anuncio a estado pendiente de validación por un moderador

Caso de uso	Editar anuncio propuesto
Actores	Miembro
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema. El anuncio que se quiere editar no puede estar publicado (aprobado por un Moderador)
Postcondición	Se almacenarán los nuevos valores del anuncio en el sistema
Requisitos de información	RI5
Requisitos de datos	RD5
Requisitos de almacenamiento	RA5
Referencias	RF14
Propósito	Permitir al usuario de tipo miembro editar uno de sus anuncios propuestos antes de que sea validado
Resumen	El usuario accede a la vista de edición del anuncio propuesto e introduce en el formulario los nuevos valores. El sistema valida la nueva información y la almacena

Caso de uso	Proponer edición de anuncio aprobado
Actores	Miembro
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema. El anuncio que se quiere editar ha de estar publicado (aprobado por un Moderador)
Postcondición	Se genera una solicitud de edición de anuncio que deberá ser aprobada por un usuario Moderador
Requisitos de información	RI5
Requisitos de datos	RD5
Requisitos de almacenamiento	RA5
Referencias	RF15
Propósito	Permitir al usuario de tipo Miembro editar uno de sus anuncios propuestos una vez que ya ha sido publicado
Resumen	El usuario accede a la vista de edición del anuncio publicado e introduce en el formulario los nuevos valores. El sistema valida la nueva información, la almacena y pasa el anuncio a estado no publicado para que sea aprobado por un usuario Moderador

Caso de uso	Eliminar anuncio propio
Actores	Miembro, Artista, Experto
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	El anuncio se elimina del sistema
Requisitos de información	RI5
Requisitos de datos	-
Requisitos de almacenamiento	-
Referencias	RF16
Propósito	Permitir a un usuario eliminar uno de sus anuncios, independientemente de que esté publicado o no
Resumen	El usuario selecciona eliminar uno de sus anuncios de un listado. Se le pide confirmación de la acción y en caso afirmativo, se elimina el anuncio del sistema

Caso de uso	Solicitar borrado de cuenta
Actores	Miembro, Artista, Experto
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	Se generará una solicitud para que un usuario Administrador elimine definitivamente la cuenta del usuario
Requisitos de información	-
Requisitos de datos	-
Requisitos de almacenamiento	-
Referencias	RF17
Propósito	Permitir a un usuario solicitar que se elimine su cuenta de la plataforma
Resumen	El usuario selecciona selecciona solicitar que su cuenta sea eliminada. El sistema le pregunta al usuario si desea donar a la plataforma sus recursos. En caso afirmativo, los recursos pasan a su usuario Experto

Caso de uso	Insertar recurso
Actores	Artista, Moderador
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	Se almacenará el recurso y sus metadatos en el sistema, en estado publicado
Requisitos de información	-
Requisitos de datos	RD4
Requisitos de almacenamiento	RA4
Referencias	RF18
Propósito	Permitir al usuario de tipo Artista o Experto publicar un recurso en la plataforma
Resumen	El usuario accede a la vista de publicar recurso, donde aparecerá un formulario (relleno con valores por defecto en base al perfil del usuario) con los campos a establecer para publicar un recurso. El sistema valida la información y la almacena, pasando el recurso a estado publicado

Caso de uso	Editar recurso
Actores	Artista, Experto
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	Se almacenarán y publicarán los nuevos valores del recurso en el sistema
Requisitos de información	RI4
Requisitos de datos	RD4
Requisitos de almacenamiento	RA4
Referencias	RF19
Propósito	Permitir al usuario de tipo Artista o Experto editar uno de sus recursos publicados
Resumen	El usuario accede a la vista de edición del recurso publicado e introduce en el formulario los nuevos valores. El sistema valida la nueva información y la publica

Caso de uso	Insertar anuncio
Actores	Artista, Moderador
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	Se almacenará el anuncio en el sistema, en estado publicado
Requisitos de información	-
Requisitos de datos	RD5
Requisitos de almacenamiento	RA5
Referencias	RF20
Propósito	Permitir al usuario de tipo Artista o Experto publicar un anuncio en la plataforma
Resumen	El usuario accede a la vista de publicar anuncio, donde aparecerá un formulario con los campos a establecer para publicar un anuncio. El sistema valida la información y la almacena, pasando el recurso a estado publicado

Caso de uso	Editar anuncio
Actores	Artista, Experto
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	Se almacenarán y publicarán los nuevos valores del anuncio en el sistema
Requisitos de información	RI5
Requisitos de datos	RD5
Requisitos de almacenamiento	RA5
Referencias	RF21
Propósito	Permitir al usuario de tipo Artista o Experto editar uno de sus anuncios publicados
Resumen	El usuario accede a la vista de edición del anuncio publicado e introduce en el formulario los nuevos valores. El sistema valida la nueva información y la publica

Caso de uso	Editar cualquier recurso
Actores	Experto
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	Se almacenarán y publicarán los nuevos valores del recurso en el sistema.
Requisitos de información	RI4
Requisitos de datos	RD4
Requisitos de almacenamiento	RA4
Referencias	RF22
Propósito	Permitir al usuario de tipo Experto modificar la información almacenada de cualquier recurso de la plataforma, independientemente de quién lo ha publicado
Resumen	El usuario accede a la vista de edición del recurso publicado e introduce en el formulario los nuevos valores. El sistema valida la nueva información y la publica

Caso de uso	Eliminar cualquier recurso
Actores	Experto
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	El recurso se elimina del sistema
Requisitos de información	RI4
Requisitos de datos	-
Requisitos de almacenamiento	-
Referencias	RF23
Propósito	Permitir a un usuario Experto eliminar cualquier recurso de la plataforma, independientemente de quién lo ha publicado
Resumen	El usuario selecciona eliminar uno de los recursos disponibles de la plataforma de un listado. Se le pide confirmación de la acción y en caso afirmativo, se elimina el recurso del sistema

Caso de uso	Editar cualquier anuncio
Actores	Experto
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	Se almacenarán y publicarán los nuevos valores del anuncio en el sistema
Requisitos de información	RI5
Requisitos de datos	RD5
Requisitos de almacenamiento	RA5
Referencias	RF24
Propósito	Permitir al usuario de tipo Experto modificar la información almacenada de cualquier anuncio de la plataforma, independientemente de quién lo ha publicado
Resumen	El usuario accede a la vista de edición del anuncio publicado e introduce en el formulario los nuevos valores. El sistema valida la nueva información y la publica

Caso de uso	Eliminar cualquier anuncio
Actores	Experto
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	El anuncio se elimina del sistema
Requisitos de información	RI5
Requisitos de datos	-
Requisitos de almacenamiento	-
Referencias	RF25
Propósito	Permitir a un usuario Experto eliminar cualquier anuncio de la plataforma, independientemente de quién lo ha publicado
Resumen	El usuario selecciona eliminar uno de los anuncio disponibles de la plataforma de un listado. Se le pide confirmación de la acción y en caso afirmativo, se elimina el anuncio del sistema

Caso de uso	Aceptar recurso
Actores	Experto
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	El recurso pasa de estado “pendiente de moderar” a “publicado”
Requisitos de información	RI4
Requisitos de datos	-
Requisitos de almacenamiento	RA4
Referencias	RF26
Propósito	Permitir a un usuario Experto aceptar los recursos pendientes de publicación que han enviado los usuarios de tipo Miembro
Resumen	El usuario selecciona uno de los recursos pendientes de moderación de un listado. Revisa la información suministrada por el usuario Miembro y pulsa sobre “Aceptar”, pasando el recurso a estado publicado

Caso de uso	Aceptar anuncio
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	El anuncio pasa de estado “pendiente de moderar” a “publicado”
Requisitos de información	RI5
Requisitos de datos	-
Requisitos de almacenamiento	RA5
Procedimiento	
Referencias	RF27
Propósito	Permitir a un usuario Experto aceptar los anuncios pendientes de publicación que han enviado los usuarios de tipo Miembro
Resumen	El usuario selecciona uno de los anuncios pendientes de moderación de un listado. Revisa la información suministrada por el usuario Miembro y pulsa sobre “Aceptar”, pasando el recurso a estado publicado

Caso de uso	Rechazar recurso
Actores	Experto
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	El recurso pasa de estado “pendiente de moderar” a “rechazado”
Requisitos de información	RA4
Requisitos de datos	-
Requisitos de almacenamiento	RA4
Referencias	RF28
Propósito	Permitir a un usuario Experto rechazar los recursos pendientes de publicación que han enviado los usuarios de tipo Miembro
Resumen	El usuario selecciona uno de los recursos pendientes de moderación de un listado. Revisa la información suministrada por el usuario Miembro y pulsa sobre “Rechazar”, pasando el recurso a estado rechazado. Opcionalmente puede incluir un comentario para indicar el motivo del rechazo

Caso de uso	Rechazar anuncio
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	El anuncio pasa de estado “pendiente de moderar” a “rechazado”
Requisitos de información	RI5
Requisitos de datos	-
Requisitos de almacenamiento	RA5
Referencias	RF29
Propósito	Permitir a un usuario Experto rechazar los anuncios pendientes de publicación que han enviado los usuarios de tipo Miembro
Resumen	El usuario selecciona uno de los anuncios pendientes de moderación de un listado. Revisa la información suministrada por el usuario Miembro y pulsa sobre “Rechazar”, pasando el anuncio a estado rechazado. Opcionalmente puede incluir un comentario para indicar el motivo del rechazo

Caso de uso	Aceptar registro
Actores	Administrador
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	Se crea una nueva cuenta de usuario en el sistema
Requisitos de información	RI1
Requisitos de datos	-
Requisitos de almacenamiento	RA1
Referencias	RF30
Propósito	Permitir a un usuario Administrador aceptar la solicitud de registro de una persona para que pase a tener una cuenta de usuario en el sistema
Resumen	El usuario selecciona uno de las solicitudes de registro de usuario de un listado. Revisa la información suministrada por la persona solicitante y si pulsa “Aceptar”, se creará una nueva cuenta para ese usuario y se enviará un email de confirmación al correo electrónico suministrado

Caso de uso	Editar usuario
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	Se almacenarán los nuevos datos del usuario
Requisitos de información	RI1
Requisitos de datos	RD1
Requisitos de almacenamiento	RA1
Referencias	RF31
Propósito	Permitir al usuario de tipo Administrador modificar la información almacenada de cualquier usuario de la plataforma
Resumen	El usuario accede a la vista de edición del usuario seleccionado e introduce en el formulario los nuevos valores. El sistema valida la nueva información y la almacena

Caso de uso	Eliminar usuario
Actores	Administrador
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema. El usuario a eliminar no puede ser tipo Administrador
Postcondición	El usuario se elimina del sistema. Los recursos del usuario eliminado pasarán a estado “pendiente de publicación”
Requisitos de información	RI1
Requisitos de datos	-
Requisitos de almacenamiento	-
Referencias	RF32
Propósito	Permitir a un usuario Administrador eliminar usuarios de tipo Miembro, Artista y Experto
Resumen	El usuario selecciona eliminar uno de los usuarios de la plataforma de un listado. Se le pide confirmación de la acción y en caso afirmativo, se elimina el anuncio del sistema. Los recursos de este usuario dejarán de estar asociados a este usuario eliminado, y pasan a pertenecer al usuario Experto que tiene asociado

Caso de uso	Limitar subida de recurso
Actores	Administrador

Caso de uso	Limitar subida de recurso
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	La posibilidad de publicar recursos multimedia de gran tamaño directamente almacenados en la plataforma quedará activada/desactivada
Requisitos de información	-
Requisitos de datos	-
Requisitos de almacenamiento	-
Referencias	RF33
Propósito	Permitir a un usuario Administrador permitir o limitar el almacenamiento de contenido multimedia de gran tamaño directamente en la plataforma
Resumen	El usuario abre la vista de configuración de la plataforma y modifica el estado del valor actual del campo “Limitar recursos”. Si pasa a estado “Limitar”, no se permitirá a los usuarios subir recursos multimedia de gran tamaño. Si pasa a estado “No limitar”, se permitirá la subida de dichos recursos

Caso de uso	Consultar mensajes
Actores	Miembro, Artista, Experto, Administrador
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	-
Requisitos de información	RI6
Requisitos de datos	-
Requisitos de almacenamiento	RA6
Referencias	RF34
Propósito	Permitir a un usuario ver un listado con sus mensajes recibidos y eliminados, y filtrarlos por determinados campos
Resumen	El usuario abre la vista de mensajes y le aparece un listado con sus mensajes enviados y recibidos. El usuario podrá usar un cuadro de búsqueda para filtrar dichos mensajes en base a ciertos campos

Caso de uso	Ver mensaje
Actores	Miembro, Artista, Experto, Administrador
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	-
Requisitos de información	RI6
Requisitos de datos	-
Requisitos de almacenamiento	RA6
Referencias	RF35
Propósito	Permitir a un usuario ver el contenido de uno de sus mensajes enviados o recibidos
Resumen	El usuario selecciona uno de sus mensajes de un listado, y se muestra una vista con el contenido del mensaje

Caso de uso	Enviar mensaje
Actores	Miembro, Artista, Experto, Administrador
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	Se creará un nuevo mensaje en el sistema, y aparecerá en el listado de enviados del usuario que lo envía, y en el listado de recibidos del usuario destinatario
Requisitos de información	-
Requisitos de datos	RA6
Requisitos de almacenamiento	RA6
Referencias	RF36
Propósito	Permitir a un usuario enviar un mensaje a otro usuario
Resumen	El usuario selecciona entrar a la vista de Nuevo mensaje. Rellena los campos obligatorios del mensaje y pulsa enviar. El mensaje aparecerá en el buzón del usuario destinatario

Caso de uso	Eliminar mensaje
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	El mensaje dejará de estar disponible para el usuario que lo elimina
Requisitos de información	RI6
Requisitos de datos	-
Requisitos de almacenamiento	-
Referencias	RF37
Propósito	Permitir a un usuario eliminar un mensaje de su bandeja de entrada o de salida
Resumen	El usuario selecciona uno de los mensajes de su bandeja de entrada o de salida y pulsa en “Eliminar”. El mensaje desaparece de su bandeja, pero queda en la bandeja del otro usuario hasta que lo elimine

Caso de uso	Bloquear usuario en mensajería
Actores	Miembro, Artista, Experto, Administrador
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	El usuario que ha sido bloqueado no podrá enviar mensajes al usuario que ha realizado la acción
Requisitos de información	RI1
Requisitos de datos	-
Requisitos de almacenamiento	-
Referencias	RF38
Propósito	Permitir a un usuario bloquear a otro para no recibir sus mensajes
Resumen	El usuario selecciona uno de los usuarios de la plataforma desde un listado y selecciona “Bloquear”. El usuario bloqueado no tendrá la posibilidad de enviar mensajes al usuario que ha realizado esta acción

Caso de uso	Solicitar paso a artista
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	Se ha generado una solicitud de paso a artista
Requisitos de información	-
Requisitos de datos	-
Requisitos de almacenamiento	-
Referencias	RF39
Propósito	Permitir a un usuario miembro solicitar el paso al rol de artista
Resumen	El usuario pulsa un botón desde su perfil que genera una solicitud para pasar de rol miembro a rol artista

Caso de uso	Moderar solicitud de paso a artista
Actores	Experto
Tipo	Primario, obligatorio
Precondición	El usuario ha de estar con una sesión activa en el sistema
Postcondición	En caso de aprobar la solicitud, el rol del usuario solicitante pasa a ser Artista
Requisitos de información	-
Requisitos de datos	-
Requisitos de almacenamiento	-
Referencias	RF40
Propósito	Permitir a un usuario Experto aprobar o rechazar la solicitud de paso a Artista de otro usuario
Resumen	El usuario elige una solicitud de paso a Artista y la acepta o la rechaza. En caso de ser aceptada, el usuario solicitante cambia su rol de Miembro a Artista

3.3.3 Diagramas de secuencia

Los diagramas de secuencia son una herramienta del Lenguaje Unificado de Modelado (UML, por sus siglas en inglés) que son utilizados para modelar las interacciones entre los actores y los objetos del sistema, y la interacción de los objetos entre ellos [17]. Un diagrama de secuencia se compone de actores, mensajes enviados por esos actores, valores de retorno e indicaciones de áreas de bucle o iteración. A continuación se muestran los diagramas de secuencia para los casos de uso de la plataforma:

1. Acceder a la plataforma

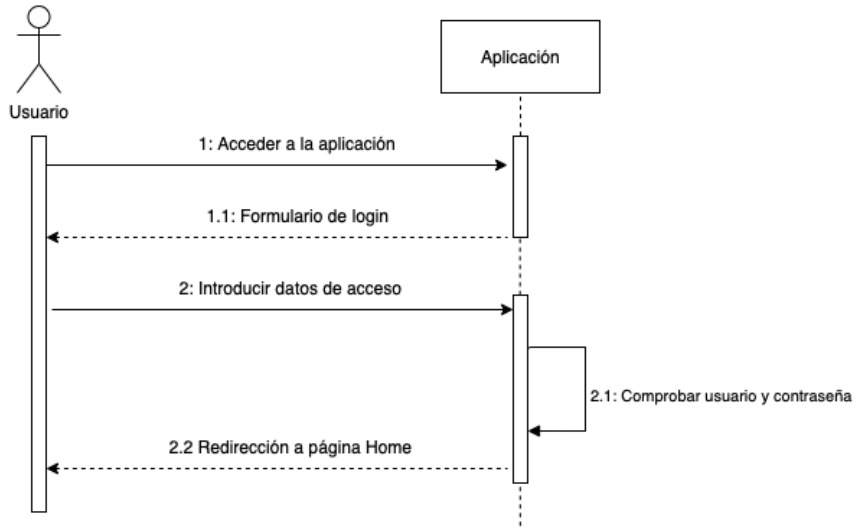


Figura 3.9: Diagrama de secuencia de Acceder a la plataforma

2. Buscar recurso

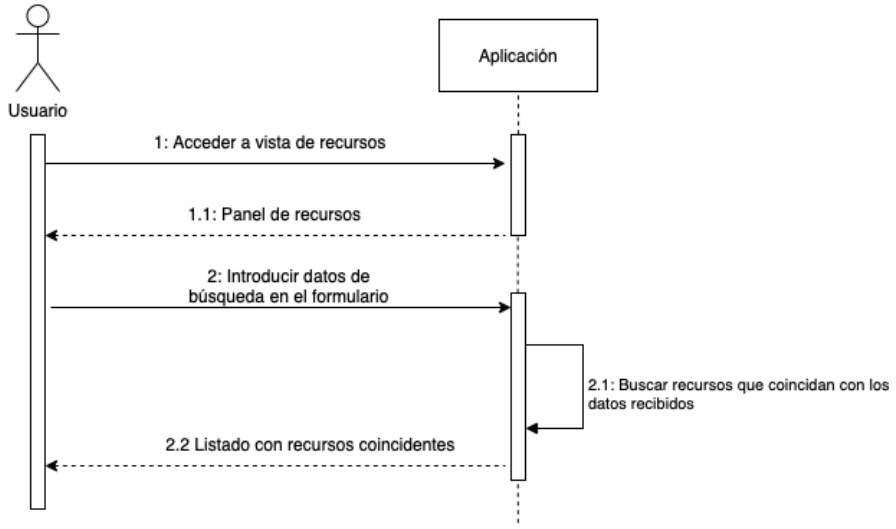


Figura 3.10: Diagrama de secuencia de Buscar recurso

3. Explorar recursos en mapa

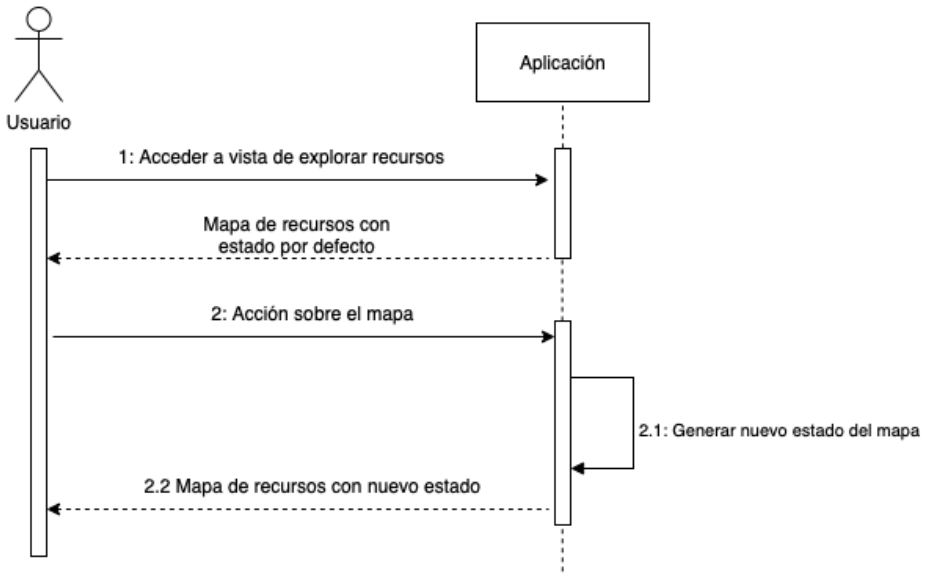


Figura 3.11: Diagrama de secuencia de Explorar recursos en mapa

4. Visualizar recurso

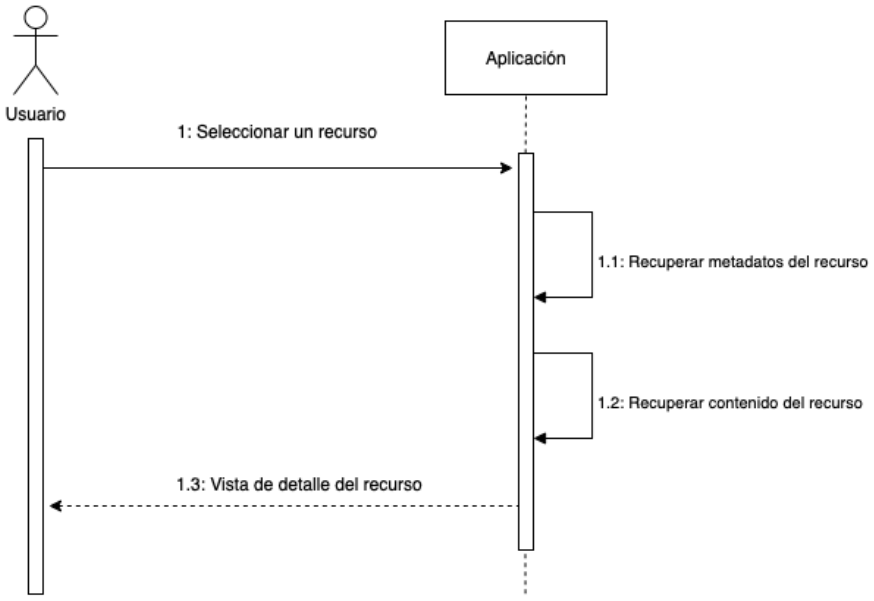


Figura 3.12: Diagrama de secuencia de Visualizar recurso

5. Solicitar registro

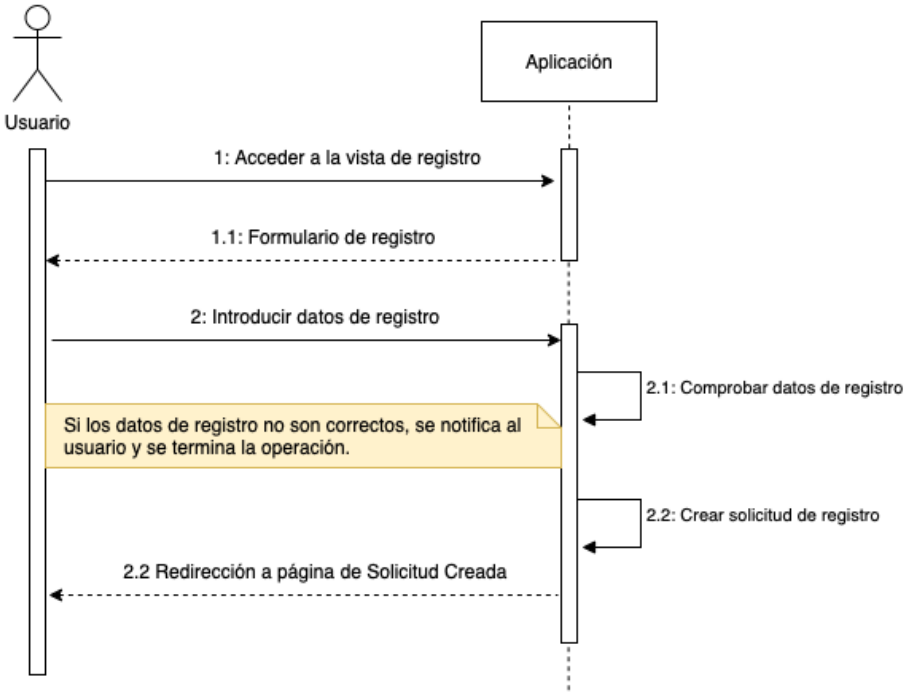


Figura 3.13: Diagrama de secuencia de Solicitar registro

6. Consultar perfil

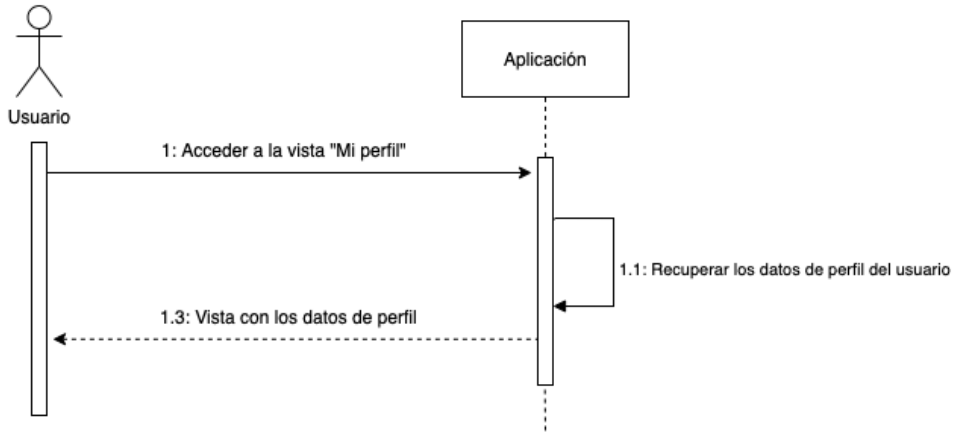


Figura 3.14: Diagrama de secuencia de Consultar perfil

7. Editar perfil

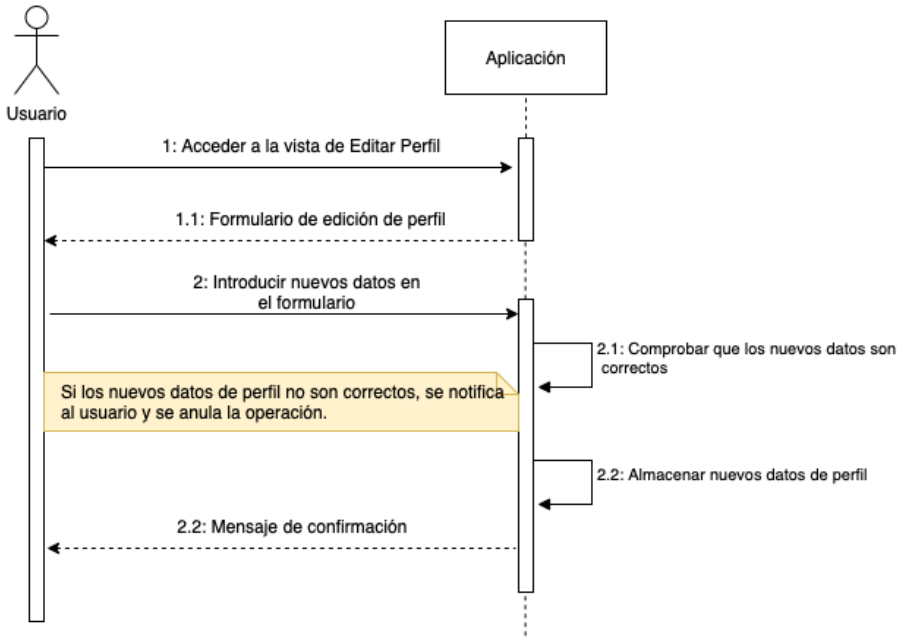


Figura 3.15: Diagrama de secuencia de Editar perfil

8. Exportar recurso

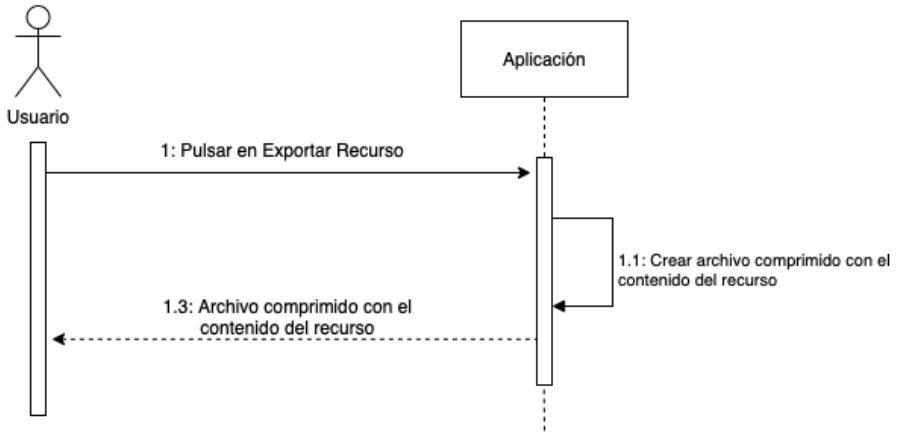


Figura 3.16: Diagrama de secuencia de Exportar recurso

9. Proponer recurso

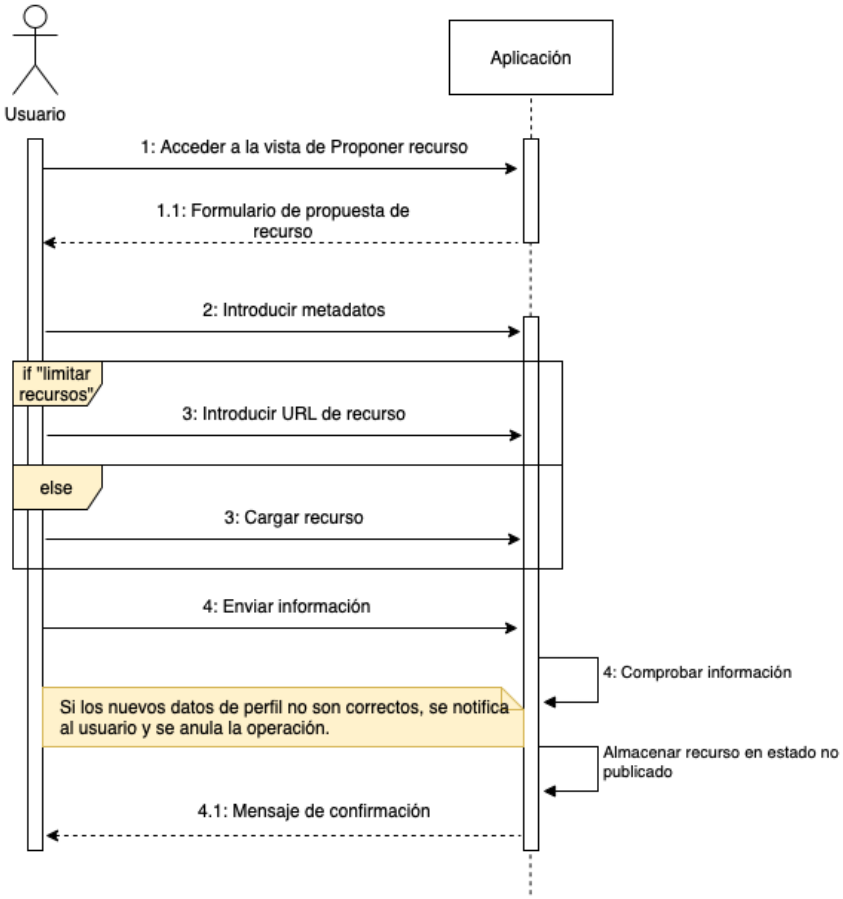


Figura 3.17: Diagrama de secuencia de Proponer recurso

10. Editar recurso propuesto

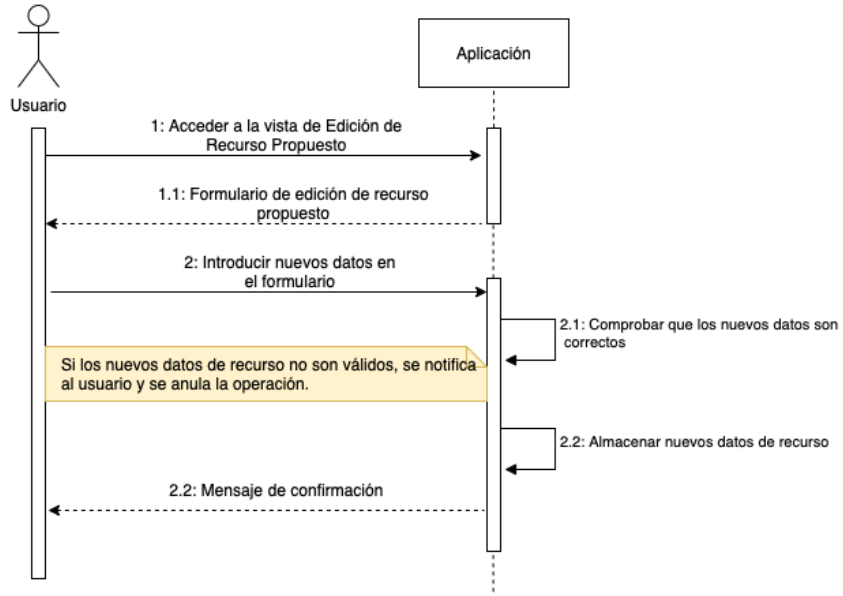


Figura 3.18: Diagrama de secuencia de Editar recurso propuesto

11. Proponer edición de recurso aprobado

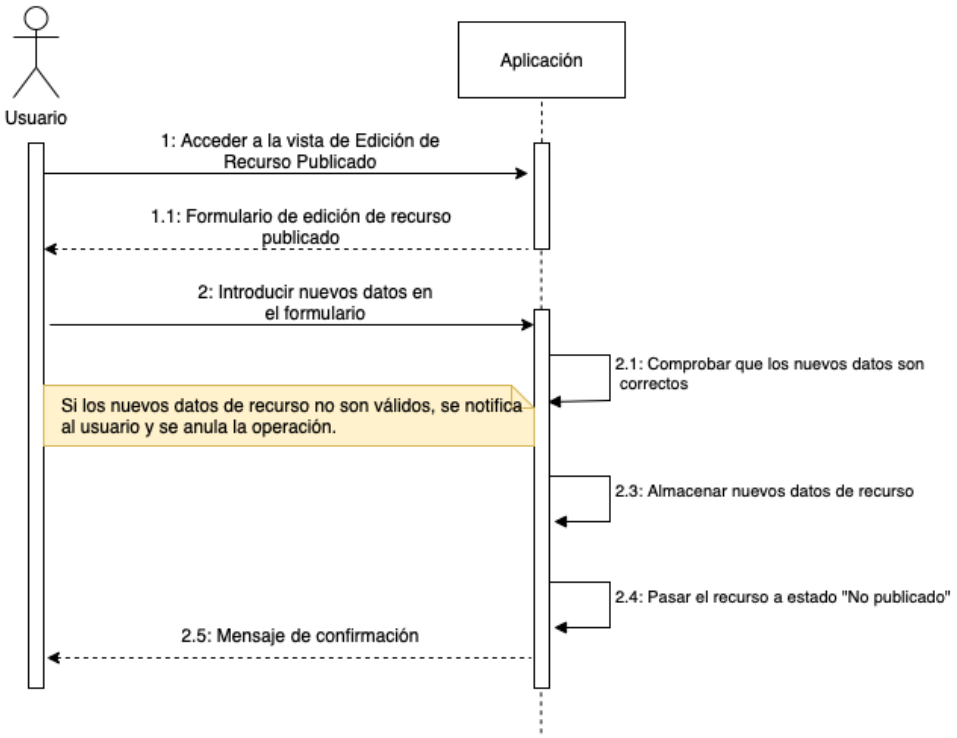


Figura 3.19: Diagrama de secuencia de Proponer edición de recurso aprobado

12. Eliminar recurso propio

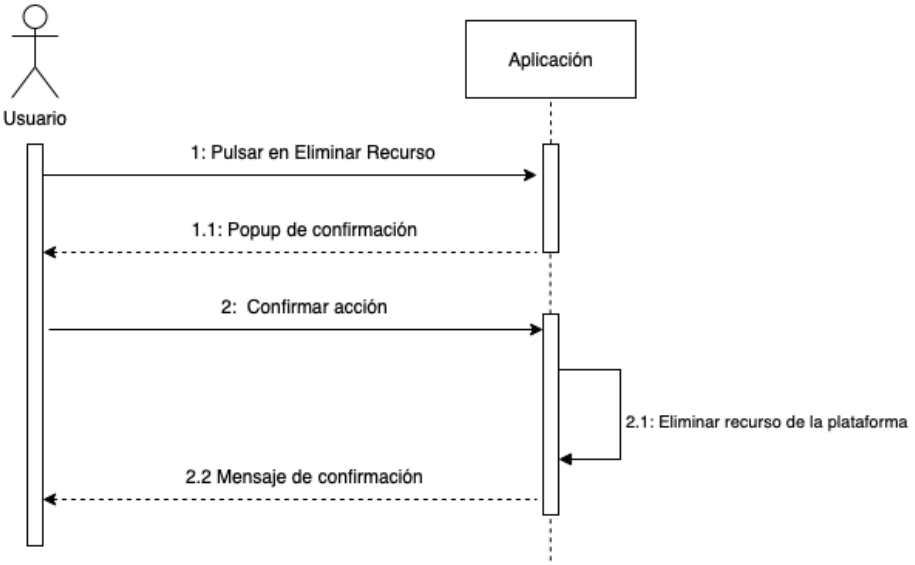


Figura 3.20: Diagrama de secuencia de Eliminar recurso propio

13. Proponer anuncio.

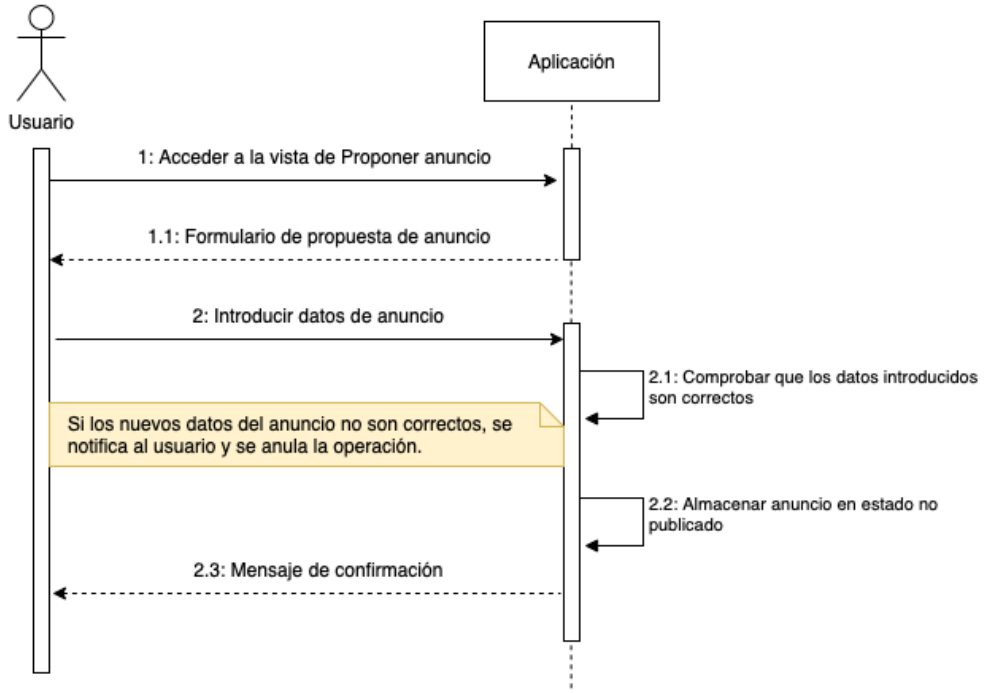


Figura 3.20: Diagrama de secuencia de Proponer anuncio

14. Editar anuncio propuesto

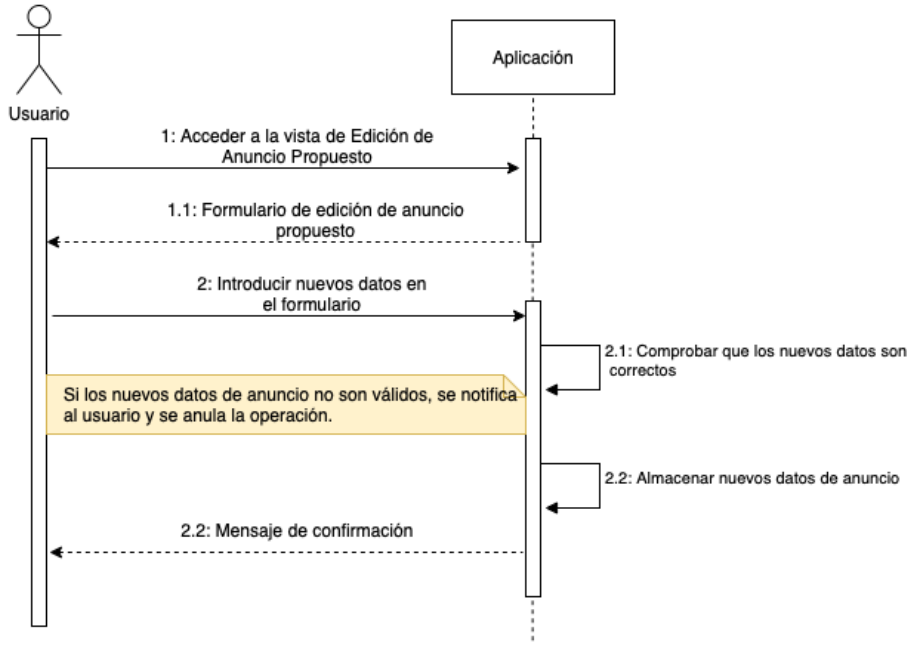


Figura 3.21: Diagrama de secuencia de Editar anuncio propuesto

15. Proponer edición de anuncio aprobado

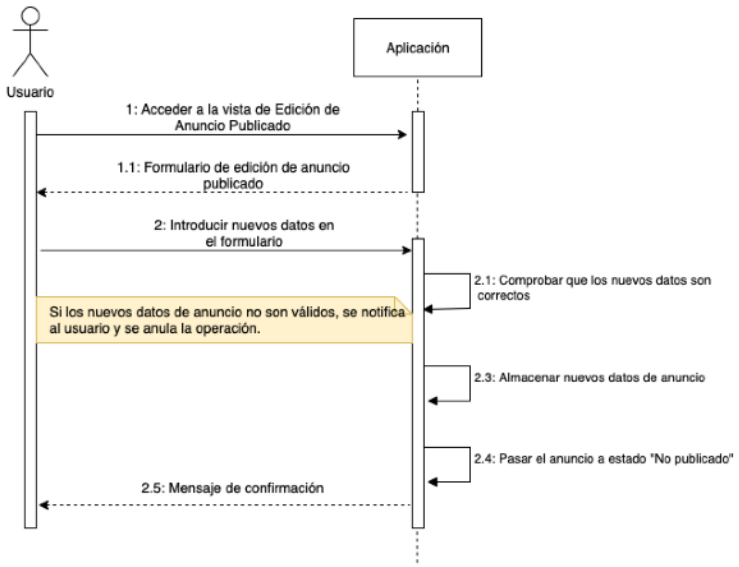


Figura 3.22: Diagrama de secuencia de Proponer edición de anuncio aprobado

16. Eliminar anuncio propio

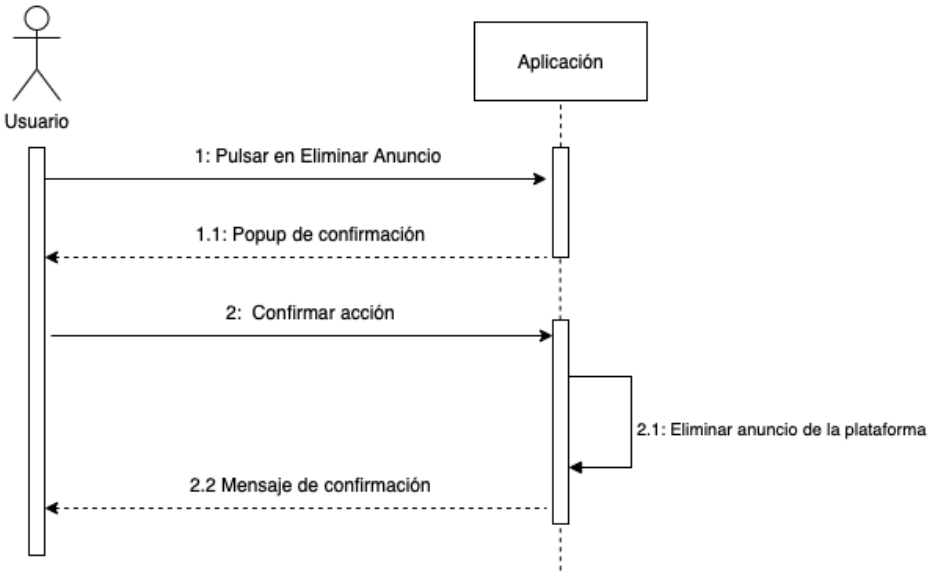


Figura 3.23: Diagrama de secuencia de Eliminar anuncio propio

17. Solicitar borrado de cuenta

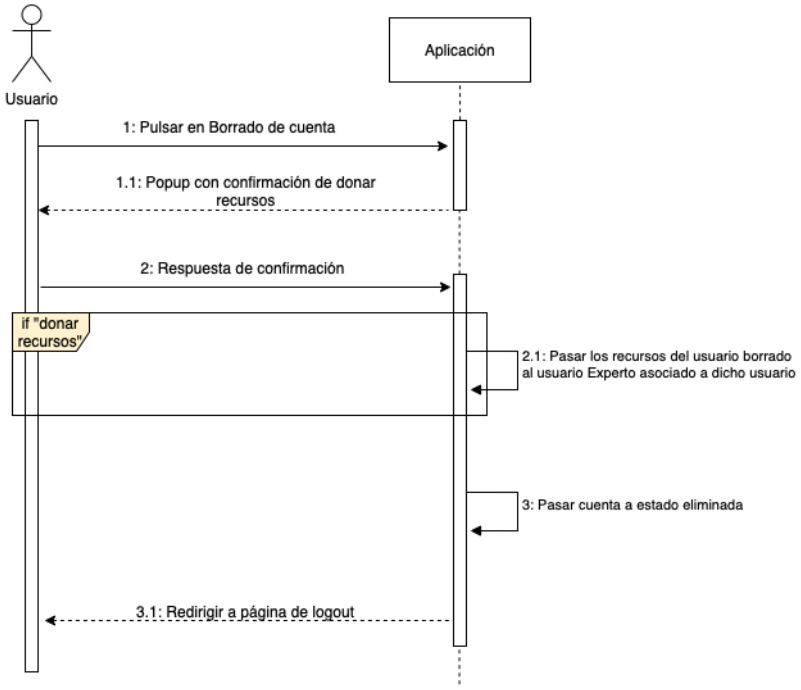


Figura 3.24: Diagrama de secuencia de Solicitar borrado de cuenta

18. Insertar recurso

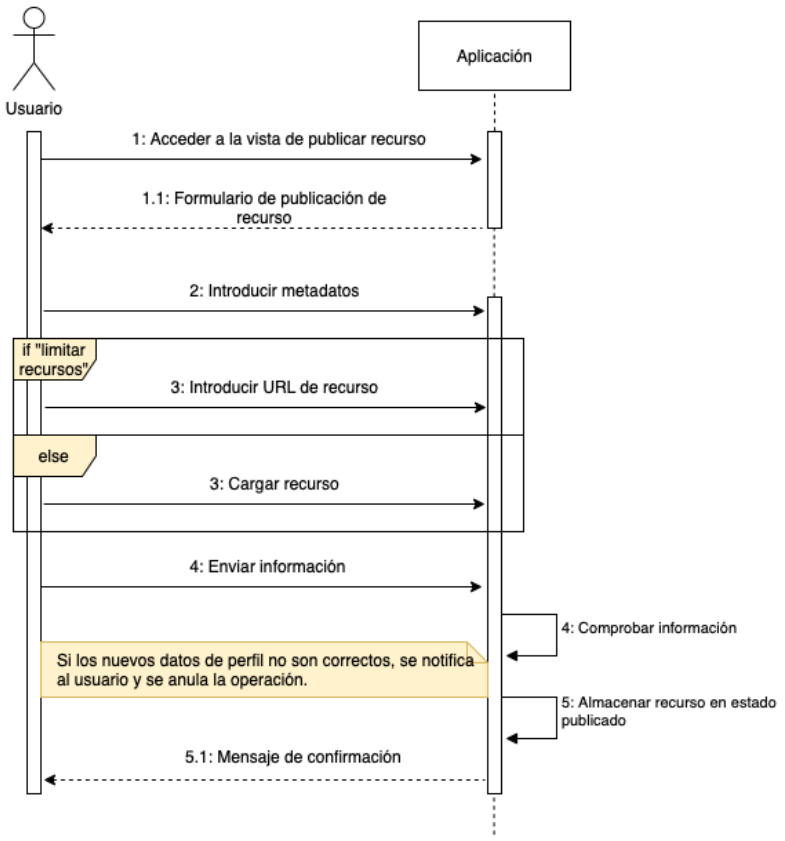


Figura 3.25: Diagrama de secuencia de Insertar recurso

19. Editar recurso

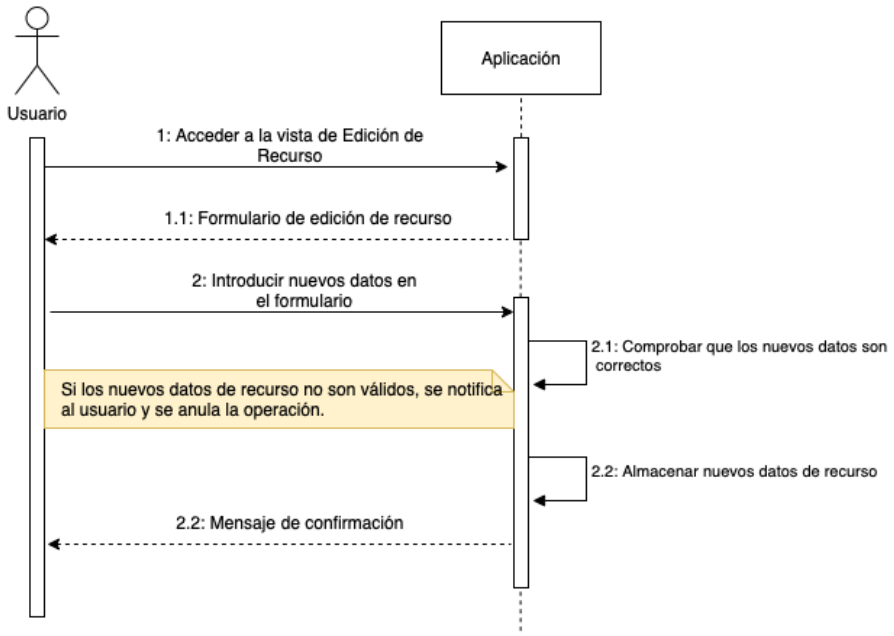


Figura 3.26: Diagrama de secuencia de Editar recurso

20. Insertar anuncio

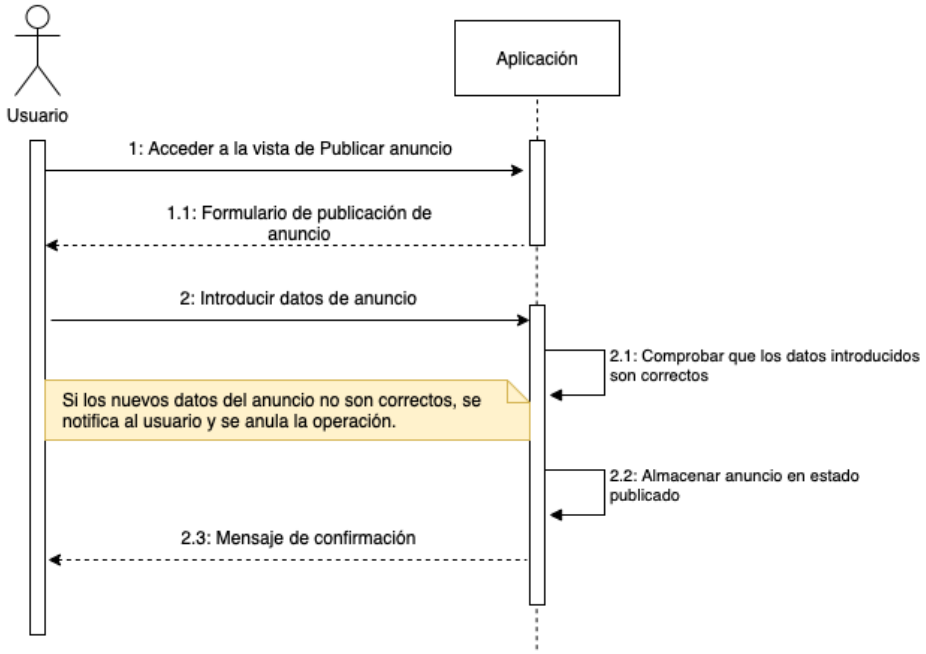


Figura 3.27: Diagrama de secuencia de Insertar anuncio

21. Editar anuncio

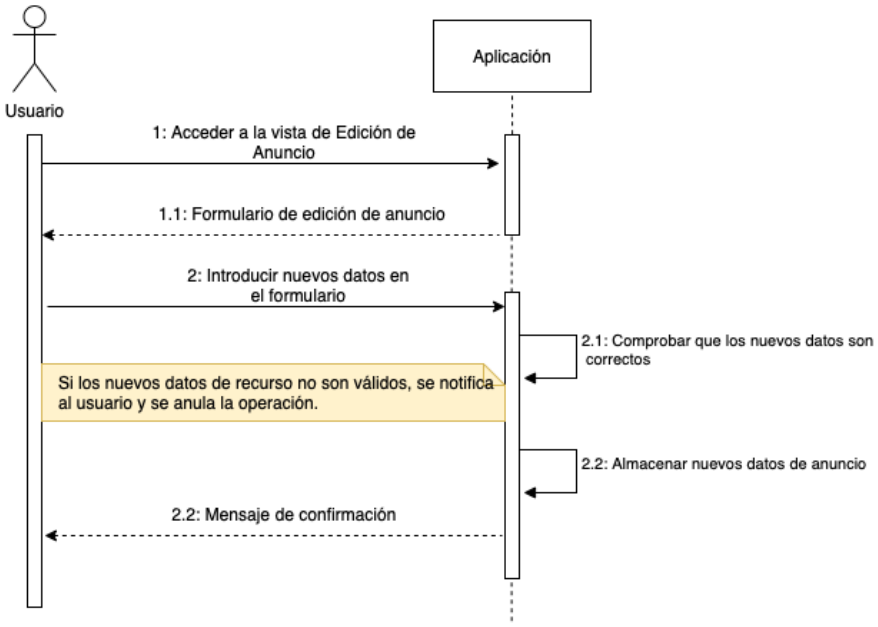


Figura 3.28: Diagrama de secuencia de Editar anuncio

22. Editar cualquier recurso

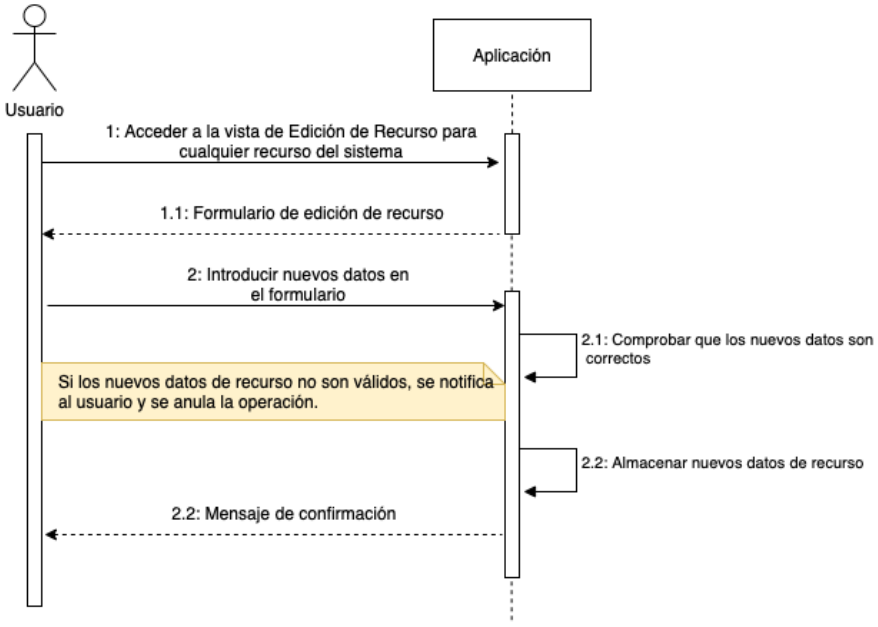


Figura 3.29: Diagrama de secuencia de Editar cualquier recurso

23. Eliminar cualquier recurso

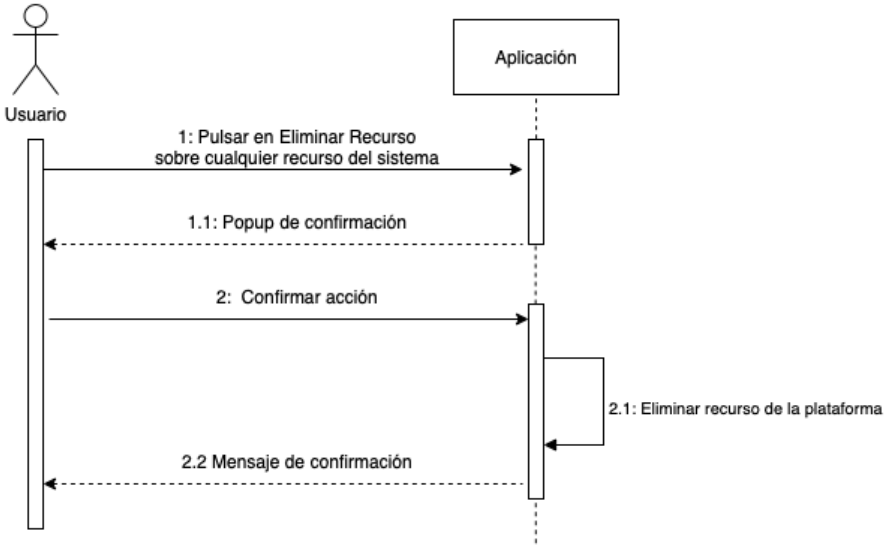


Figura 3.30: Diagrama de secuencia de Eliminar cualquier recurso

24. Editar cualquier anuncio

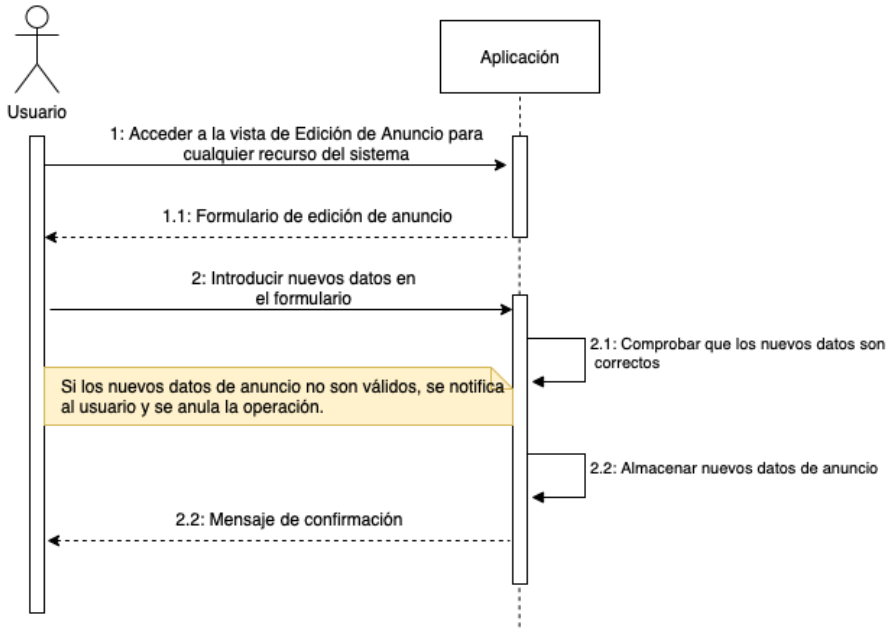


Figura 3.31: Diagrama de secuencia de Editar cualquier anuncio

25. Eliminar cualquier anuncio

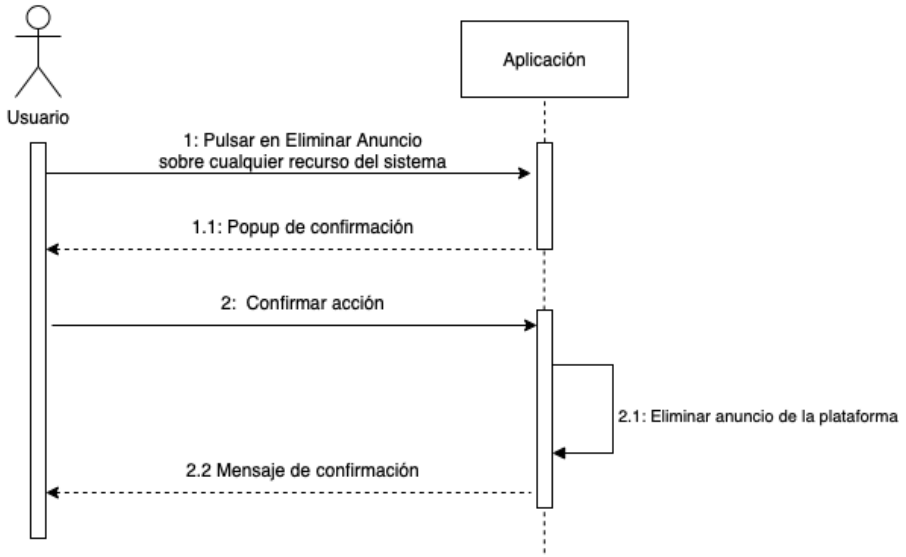


Figura 3.32: Diagrama de secuencia de Editar cualquier anuncio

26. Aceptar recurso

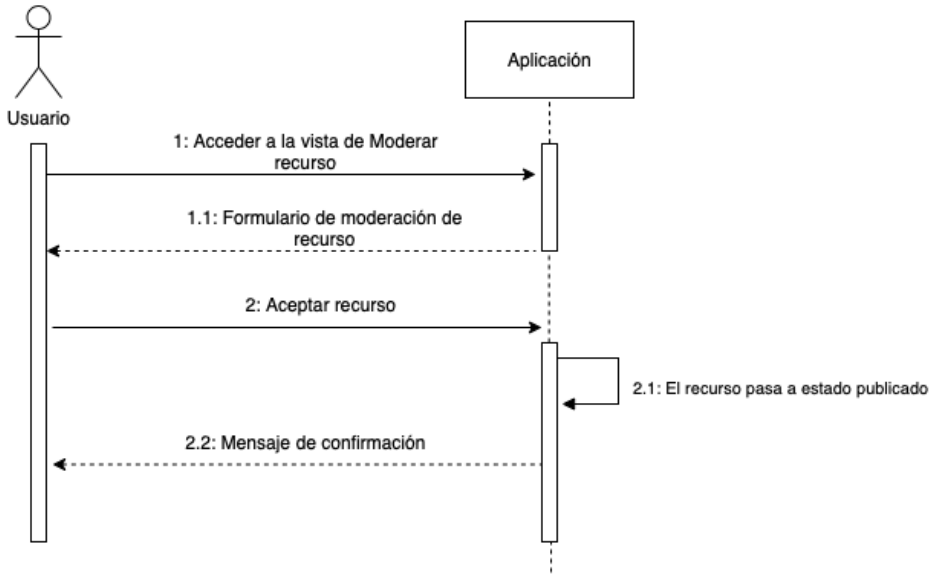


Figura 3.33: Diagrama de secuencia de Aceptar recurso

27. Aceptar anuncio

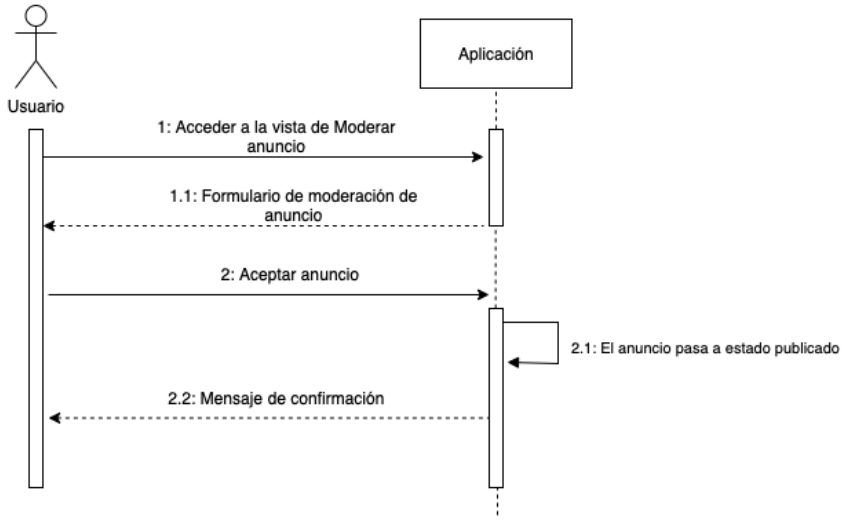


Figura 3.34: Diagrama de secuencia de Aceptar anuncio

28. Rechazar recurso

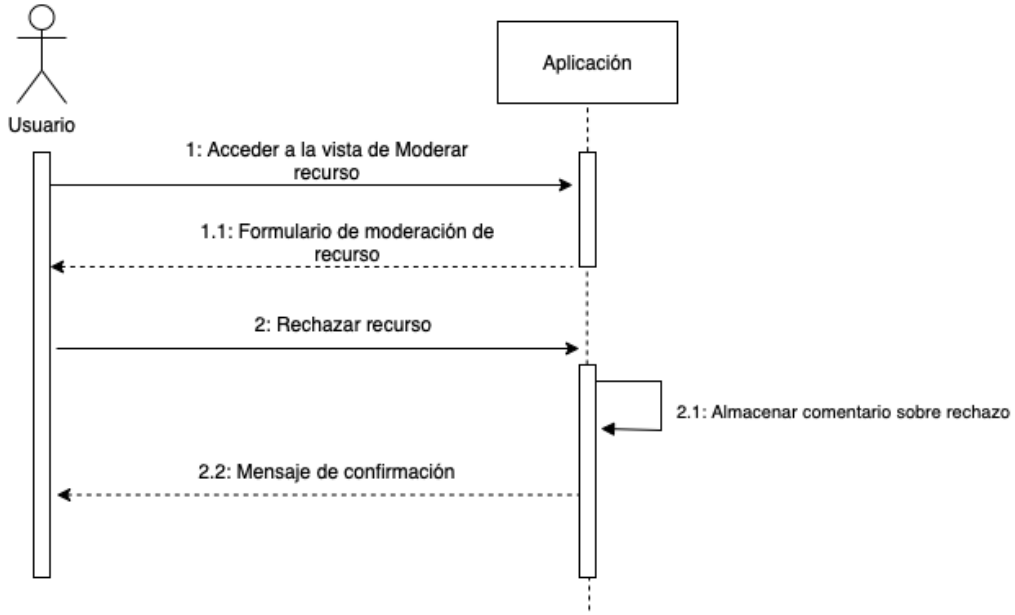


Figura 3.35: Diagrama de secuencia de Rechazar recurso

29. Rechazar anuncio

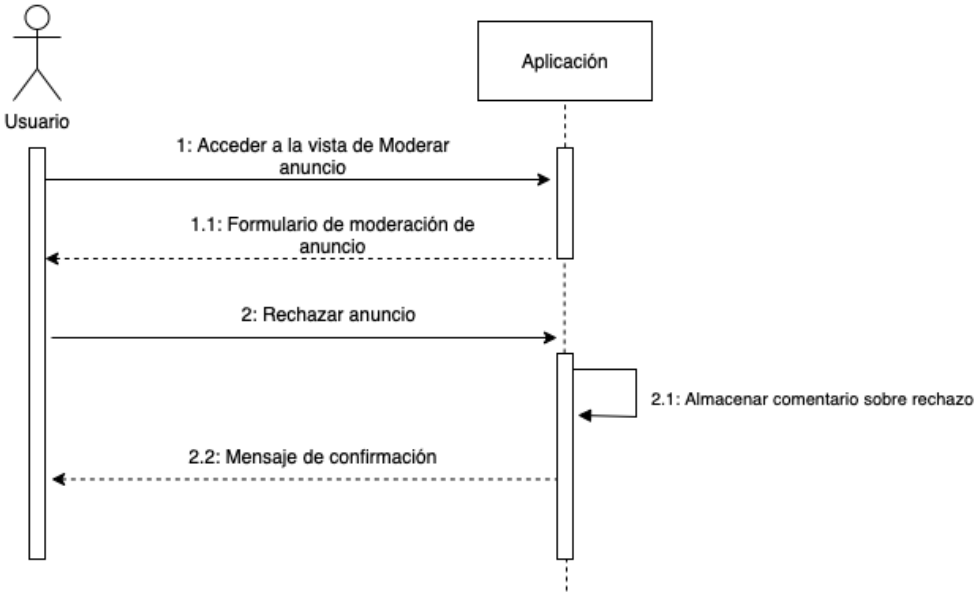


Figura 3.36: Diagrama de secuencia de Rechazar anuncio

30. Aceptar registro

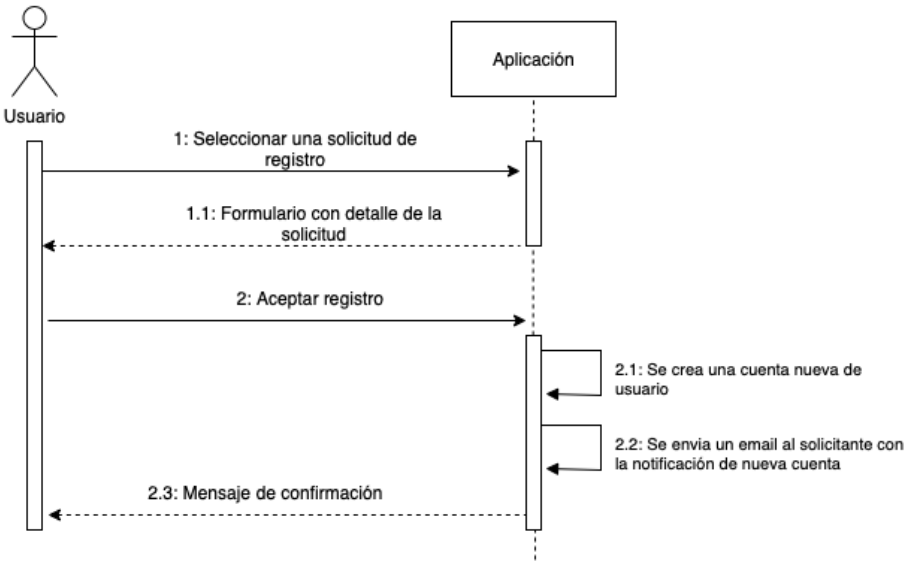


Figura 3.37: Diagrama de secuencia de Aceptar registro

31. Editar usuario

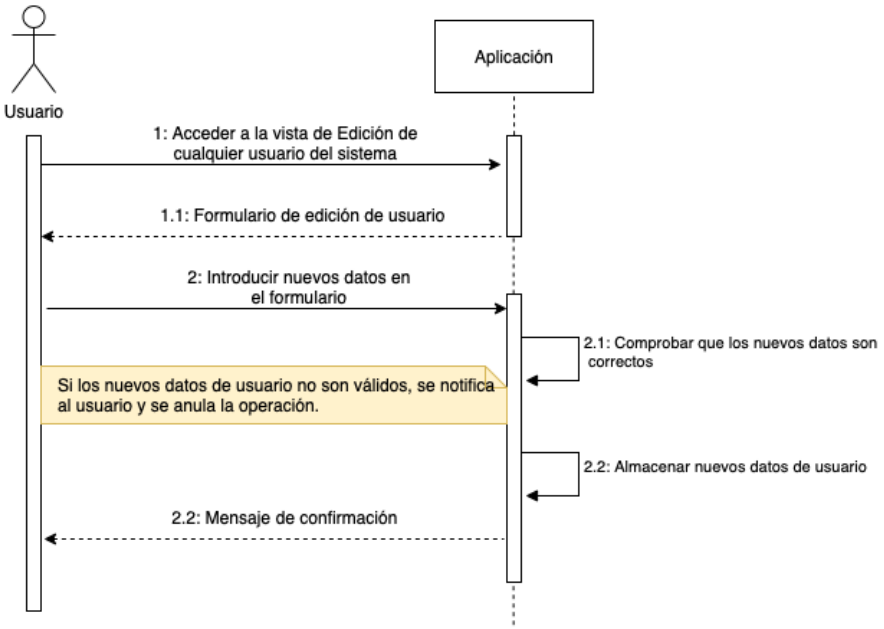


Figura 3.38: Diagrama de secuencia de Editar usuario

32. Eliminar usuario

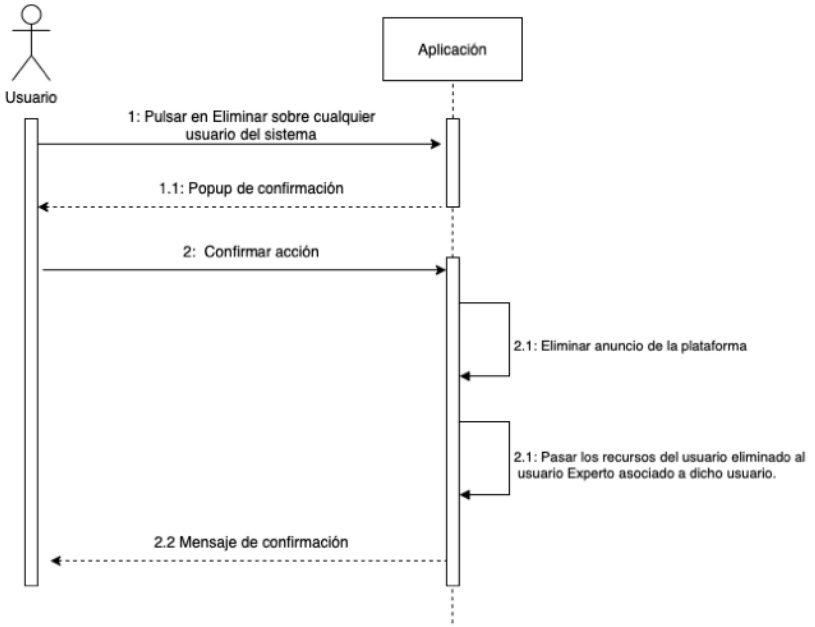


Figura 3.39: Diagrama de secuencia de Eliminar usuario

33. Limitar subida de recurso

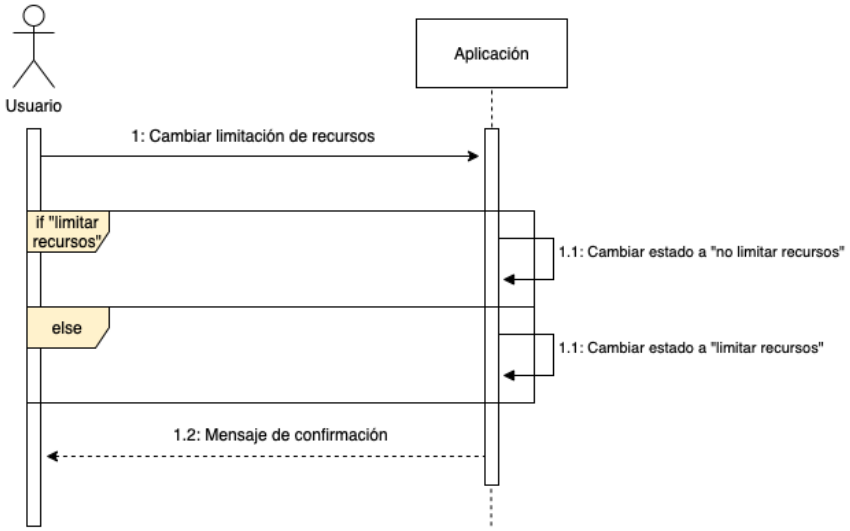


Figura 3.40: Diagrama de secuencia de Limitar subida de recurso

34. Consultar mensajes

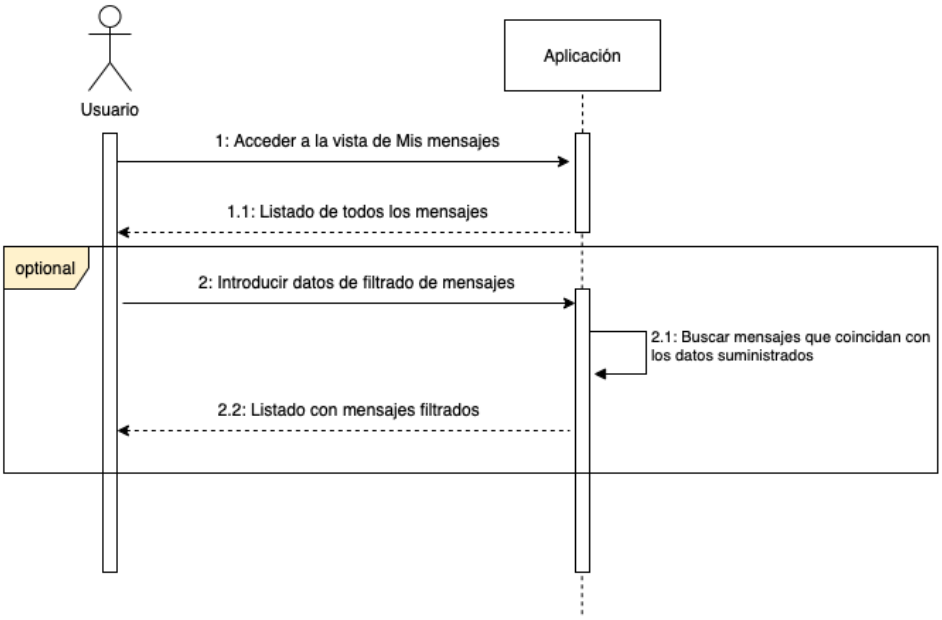


Figura 3.41: Diagrama de secuencia de Consultar mensajes

35. Ver mensaje

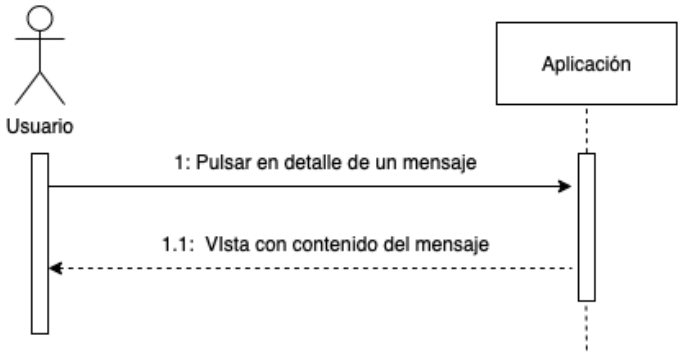


Figura 3.42: Diagrama de secuencia de Ver mensaje

36. Enviar mensaje

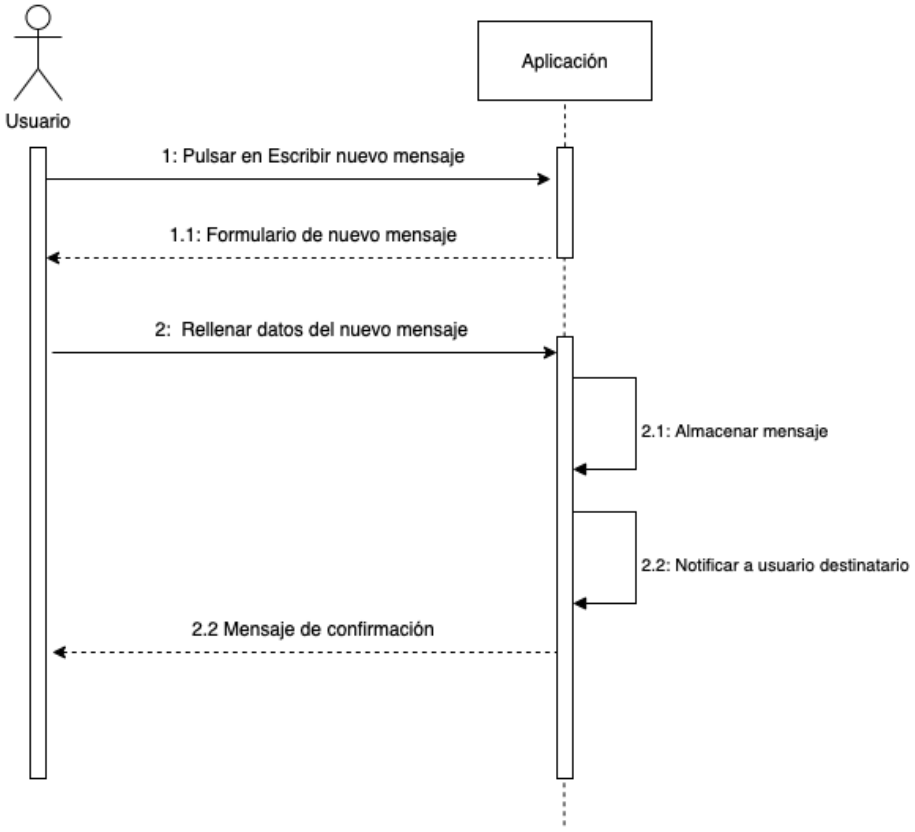


Figura 3.43: Diagrama de secuencia de Enviar mensaje

37. Eliminar mensaje

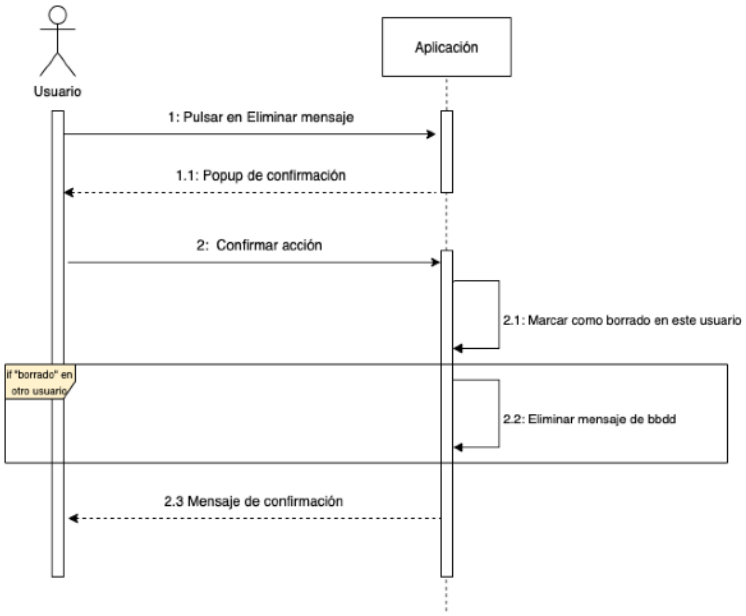


Figura 3.44: Diagrama de secuencia de Eliminar mensaje

38. Bloquear usuario

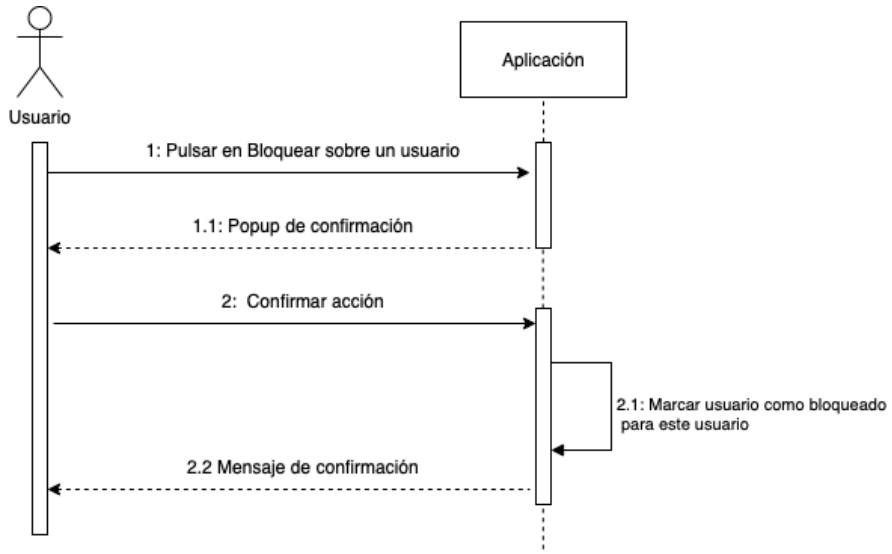


Figura 3.45: Diagrama de secuencia de Bloquear usuario

39. Solicitar paso a Artista

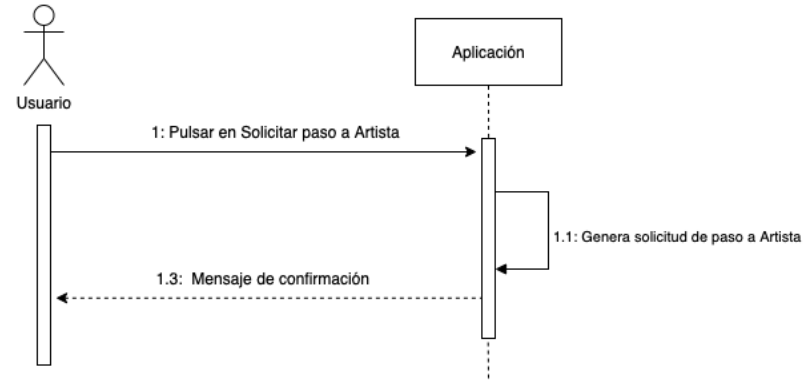


Figura 3.46: Diagrama de secuencia de Solicitar paso a artista

40. Moderar solicitud de paso a Artista

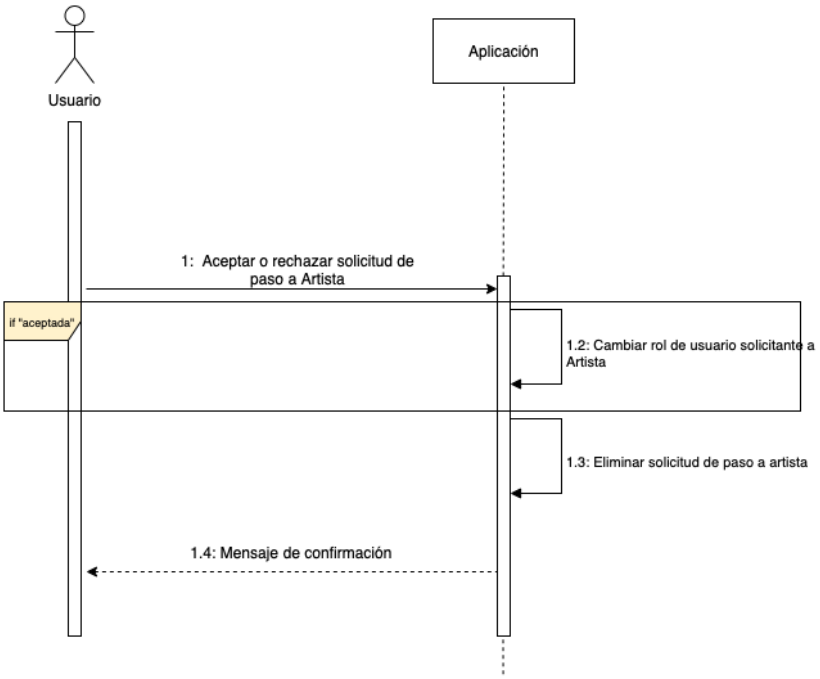


Figura 3.47: Diagrama de secuencia de Moderar solicitud de paso a artista

4. Diseño

Una vez se ha llevado a cabo el análisis de requisitos y la especificación de caso de usos, se comienza la fase de diseño de los componentes de datos del sistema por un lado, y de las interfaces de usuarios por otro. Para almacenar los datos en nuestro sistema se usará una base de datos, que nos va a permitir gestionar éstos de forma eficiente, sin redundancias ni pérdidas de información.

4.1 Diseño de la base de datos

Para definir el diseño de la base de datos, vamos a utilizar un diagrama de entidad-relación. El diagrama resultado se muestra en la figura 4.1. En este diagrama para simplificar su lectura, tan solo se han reflejado las claves primarias de las entidades y aquellos atributos que forman parte las relaciones. En él las entidades se representan utilizando rectángulos, los atributos por medio de círculos o elipses y las relaciones como líneas que conectan las entidades que tienen algún tipo de vínculo [18]

El resultado es el siguiente:

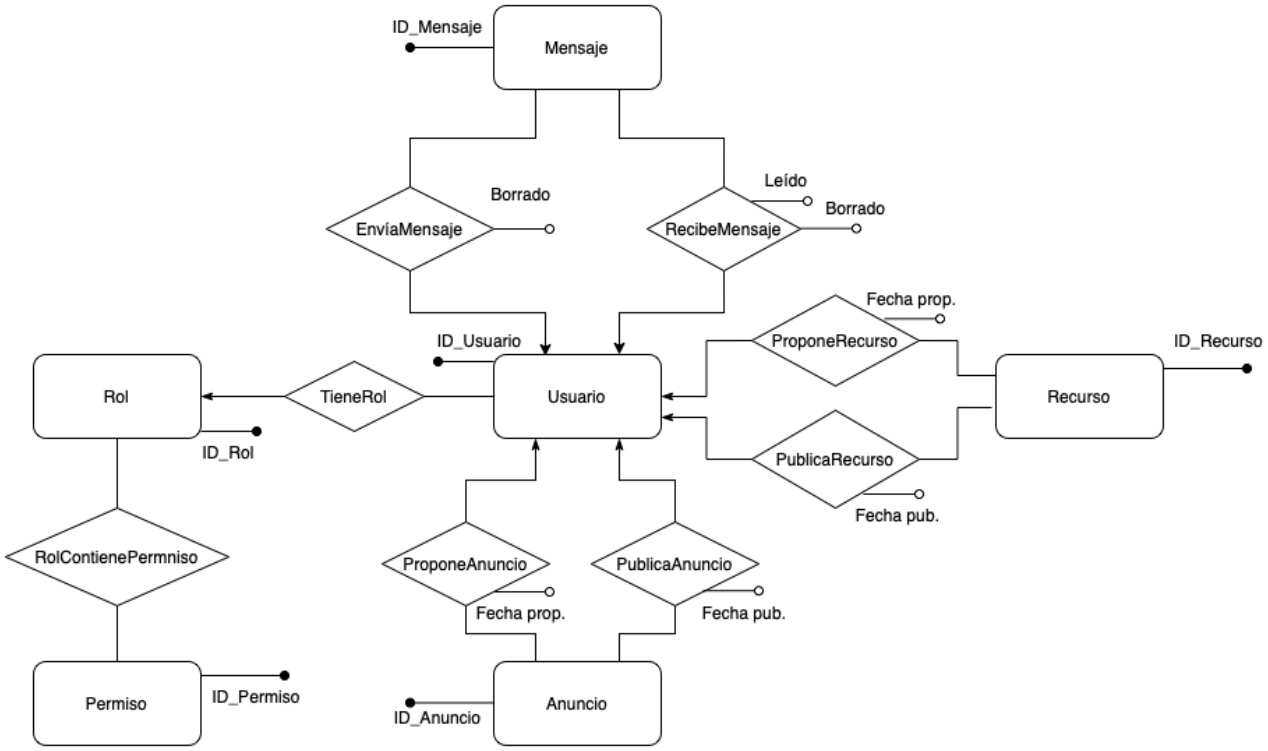


Figura 4.1: Diagrama del modelo de entidad-relación

En el diagrama, solo se muestra el atributo identificador de cada entidad. A continuación se describen el resto de atributos de cada entidad.

Usuario

- username: identificador que elige el propio usuario cuando se registra
- hash: la cadena alfanumérica resultante tras aplicar el algoritmo de hashing SHA-256 a la password proporcionada por el usuario cuando se registra.
- nombre: nombre real del usuario.
- apellidos: apellidos reales del usuario.
- email: dirección de correo electrónico del usuario.
- fecha_alta: timestamp en el que se realizó el registro del usuario

Rol

- etiqueta: nombre del rol en un formato legible, para facilitar su identificación en tareas de administración.

Permiso

- etiqueta: nombre del rol en un formato legible, para facilitar su identificación en tareas de administración.

Recurso

En esta entidad, la mayoría atributos hacen referencia al estándar DublinCore15, como ya se indicó en la sección de requisitos de datos que ahora describimos para determinar el dominio de cada uno de ellos para luego definir su tipo o rango de valores.

- tipo_multimedia: el tipo de contenido multimedia que es (imagen, audio, video, documento, imagen360).

- título: nombre dado al recurso, habitualmente por el autor. Se corresponde con la etiqueta DC.Title del estándar DublinCore15.

- claves: conjunto de palabras clave para facilitar su búsqueda e identificación.

- descripción: una descripción textual del recurso. Puede ser un resumen en el caso de un documento o una descripción del contenido en el caso de un documento visual. Se corresponde con la etiqueta DC.Description del estándar DublinCore15.

- fuente: secuencia de caracteres usados para identificar unívocamente un trabajo a partir del cual proviene el recurso actual. Se corresponde con la etiqueta DC.Source del estándar DublinCore15.

- tipo_recurso: la categoría del recurso. Por ejemplo, página personal, romance, poema, diccionario, etc. Se corresponde con la etiqueta DC.Type del estándar DublinCore15.

- cobertura: es la característica de cobertura espacial y/o temporal del contenido intelectual del recurso. La cobertura espacial se refiere a una región física. La cobertura temporal se refiere al contenido del recurso, no a cuándo fue creado. No confundir este campo con el campo “coordenadas”. Este campo es más genérico y no tiene que seguir un estándar de formato, es meramente descriptivo (por ejemplo, el nombre de un lugar que no aparece actualmente en los mapas por cambio de denominación) y no se utilizará para ubicar el recurso en el mapa. Se corresponde con la etiqueta DC.Coverage del estándar DublinCore15.

- autor: la persona u organización responsable de la creación del contenido intelectual del recurso. Por ejemplo, los autores en el caso de documentos escritos; artistas, fotógrafos e ilustradores en el caso de recursos visuales. No confundir este campo con el identificador interno del usuario de la plataforma que propone este recurso. Se corresponde con la etiqueta DC.Creator del estándar DublinCore15.

- editor: la entidad responsable de hacer que el recurso se encuentre disponible en la red en su formato actual. No confundir este campo con el identificador interno del usuario de la plataforma que aprueba este recurso.Se corresponde con la etiqueta DC.Publisher del estándar DublinCore15.

- derechos: referencia (por ejemplo, una URL) para una nota sobre derechos de autor, para un servicio de gestión de derechos o para un servicio que dará información sobre términos y condiciones de acceso a un recurso. Se corresponde con la etiqueta DC.Rights del estándar DublinCore15.

- fecha: una fecha en la cual el recurso se puso a disposición del usuario en su forma actua, no confundir con la fecha de propuesta o de publicación. Se corresponde con la etiqueta DC.Date del estándar DublinCore15.

- formato: es el formato de datos de un recurso, usado para identificar el software y, posiblemente, el hardware que se necesitaría para mostrar el recurso. Se corresponde con la etiqueta DC.Format del estándar DublinCore15.

- identificador: secuencia de caracteres utilizados para identificar unívocamente un recurso. Ejemplos para recursos en línea pueden ser URLs y URNs. Para otros recursos pueden ser usados otros formatos de identificadores, como por ejemplo ISBN. Se corresponde con la etiqueta DC.Identifier del estándar DublinCore15.

Anuncio

- fecha inicio: fecha de inicio del anuncio.
- fecha fin: fecha de finalización del anuncio.
- titulo: título o encabezamiento del anuncio.
- contenido: texto con el contenido del anuncio.

Mensaje

- asunto: asunto del mensaje.
- Contenido: texto con el contenido del mensaje
- Fecha: timestamp con la fecha de envío del mensaje.

4.2 Paso a tablas

En esta fase, se procede a traducir las entidades, relaciones y atributos definidas en la sección anterior a tablas de la base de datos.

Las tablas iniciales (o de partida) son las siguientes:

Entidades

Usuario

Atributo	Clave primaria	Clave externa	Tipo
ID	Sí	No	ENTERO
email	No	No	CADENA DE CARACTERES
username	No	No	CADENA DE CARACTERES
Hash	No	No	CADENA DE CARACTERES
nombre	No	No	CADENA DE CARACTERES
apellidos	No	No	CADENA DE CARACTERES
fecha_alta	No	No	ENTERO

Rol

Atributo	Clave primaria	Clave externa	Tipo
ID	Sí	No	ENTERO
etiqueta	No	No	CADENA DE CARACTERES

Permiso

Atributo	Clave primaria	Clave externa	Tipo
ID	Sí	No	ENTERO
etiqueta	No	No	CADENA DE CARACTERES

Recurso

Atributo	Clave primaria	Clave externa	Tipo
ID	Sí	No	ENTERO
tipo_multimedia	No	No	CADENA DE CARACTERES
título	No	No	CADENA DE CARACTERES
claves	No	No	CADENA DE CARACTERES
descripción	No	No	CADENA DE CARACTERES
fuente	No	No	CADENA DE CARACTERES
tipo_recurso	No	No	CADENA DE CARACTERES
cobertura	No	No	CADENA DE CARACTERES
coordenadas	No	No	CADENA DE CARACTERES
autor	No	No	CADENA DE CARACTERES
editor	No	No	CADENA DE CARACTERES
derechos	No	No	CADENA DE CARACTERES
fecha	No	No	ENTERO
formato	No	No	CADENA DE CARACTERES
URL	No	No	CADENA DE CARACTERES

Anuncio

Atributo	Clave primaria	Clave externa	Tipo
ID	Sí	No	ENTERO
fecha_inicio	No	No	ENTERO
fecha_fin	No	No	ENTERO
título	No	No	CADENA DE CARACTERES
contenido	No	No	TEXTO

Mensaje

Atributo	Clave primaria	Clave externa	Tipo
ID	Sí	No	ENTERO
fecha	No	No	ENTERO
asunto	No	No	CADENA DE CARACTERES
contenido	No	No	TEXTO

Relaciones

TieneRol (Usuario - Rol)

Atributo	Clave primaria	Clave externa	Tipo
ID_usuario	Sí	Sí	ENTERO
ID_rol	No	Sí	ENTERO

RolContienePermiso (Rol - Permiso)

Atributo	Clave primaria	Clave externa	Tipo
ID_permiso	Sí	Sí	ENTERO
ID_rol	Sí	Sí	ENTERO

ProponeRecurso (Usuario - Recurso)

Atributo	Clave primaria	Clave externa	Tipo
ID_recurso	Sí	Sí	ENTERO
ID_usuario	No	Sí	ENTERO
fecha_propuesta	No	No	ENTERO

PublicaRecurso (Usuario - Recurso)

Atributo	Clave primaria	Clave externa	Tipo
ID_recurso	Sí	Sí	ENTERO
ID_usuario	No	Sí	ENTERO
fecha_publicación	No	No	ENTERO

ProponeAnuncio (Usuario - Anuncio)

Atributo	Clave primaria	Clave externa	Tipo
ID_anuncio	Sí	Sí	ENTERO
ID_usuario	No	Sí	ENTERO
fecha_propuesta	No	No	ENTERO

PublicaAnuncio (Usuario - Recurso)

Atributo	Clave primaria	Clave externa	Tipo
ID_anuncio	Sí	Sí	ENTERO
ID_usuario	No	Sí	ENTERO
fecha_publicación	No	No	ENTERO

EnvíaMensaje (Usuario - Mensaje)

Atributo	Clave primaria	Clave externa	Tipo
ID_mensaje	Sí	Sí	ENTERO
ID_usuario	No	Sí	ENTERO
borrado_en_origen	No	No	ENTERO

RecibeMensaje (Usuario - Mensaje)

Atributo	Clave primaria	Clave externa	Tipo
ID_mensaje	Sí	Sí	ENTERO
ID_usuario	No	Sí	ENTERO
borrado_en_destino	No	No	ENTERO

4.3 Fusión de tablas

En esta fase se revisan las tablas iniciales obtenidas del paso a tabla para comprobar si varias tablas comparten las mismas claves primarias en cuyo caso, se procedería al proceso de fusión. Las tablas que comparten las claves se unen una sola y se elimina una de ellas. Se llega a la conclusión de que se pueden realizar las siguientes fusiones de tablas:

Se realizan las siguientes fusiones de tablas:

- La tabla TieneRol (Usuario - Rol) desaparece, y se fusionan las tablas Usuario con TieneRol, añadiendo el campo id_rol a la tabla de Usuario.

- La tabla ProponeRecurso (Usuario - Recurso) desaparece, y se añaden los campos “id_usuario_propone” y “fecha_propuesta” en la tabla Recurso.

- La tabla PublicaRecurso (Usuario - Recurso) desaparece, y se añaden los campos “id_usuario_publica” y “fecha_publicacion” en la tabla Recurso.

- La tabla ProponeAnuncio (Usuario - Anuncio) desaparece, y se añaden los “id_usuario_propone” y “fecha_propuesta” en la tabla Anuncio.

- La tabla PublicaAnuncio (Usuario - Anuncio) desaparece, y se añaden los campos“id_usuario_publica” y “fecha_publicacion” en la tabla Anuncio.

- La tabla EnvíaMensaje (Usuario - Mensaje) y se añade el campo “id_usuario_envia” y el campo “borrado_en_origen” a la tabla de Mensaje.

- La tabla RecibeMensaje (Usuario - Mensaje) y se añade el campo “id_usuario_recibe” y el campo “borrado_en_destino” a la tabla de Mensaje

Después de la fusión, las tablas resultantes son:

Entidades

Usuario

Atributo	Clave primaria	Clave externa	Tipo
ID	Sí	No	ENTERO
email	No	No	CADENA DE CARACTERES
username	No	No	CADENA DE CARACTERES
Hash	No	No	CADENA DE CARACTERES
nombre	No	No	CADENA DE CARACTERES
apellidos	No	No	CADENA DE CARACTERES
fecha_alta	No	No	ENTERO
id_rol	No	Sí	ENTERO

Rol

Atributo	Clave primaria	Clave externa	Tipo
ID	Sí	No	ENTERO
etiqueta	No	No	CADENA DE CARACTERES

Permiso

Atributo	Clave primaria	Clave externa	Tipo
ID	Sí	No	ENTERO
etiqueta	No	No	CADENA DE CARACTERES

Recurso

Atributo	Clave primaria	Clave externa	Tipo
ID	Sí	No	ENTERO
tipo_multimedia	No	No	CADENA DE CARACTERES
título	No	No	CADENA DE CARACTERES
claves	No	No	CADENA DE CARACTERES
descripción	No	No	CADENA DE CARACTERES
fuente	No	No	CADENA DE CARACTERES
tipo_recurso	No	No	CADENA DE CARACTERES
cobertura	No	No	CADENA DE CARACTERES
coordenadas	No	No	CADENA DE CARACTERES
autor	No	No	CADENA DE CARACTERES
editor	No	No	CADENA DE CARACTERES
derechos	No	No	CADENA DE CARACTERES
fecha	No	No	ENTERO
formato	No	No	CADENA DE CARACTERES
URL	No	No	CADENA DE CARACTERES
id_usuario_propone	No	Sí	ENTERO
fecha_propuesta	No	No	ENTERO
id_usuario_publica	No	Sí	ENTERO

Atributo	Clave primaria	Clave externa	Tipo
fecha_publicación	No	No	ENTERO

Anuncio

Atributo	Clave primaria	Clave externa	Tipo
ID	Sí	No	ENTERO
fecha	No	No	ENTERO
título	No	No	CADENA DE CARACTERES
contenido	No	No	TEXTO
id_usuario_propone	No	Sí	ENTERO
fecha_propuesta	No	No	ENTERO
id_usuario_publica	No	Sí	ENTERO
fecha_publicación	No	No	ENTERO

Mensaje

Atributo	Clave primaria	Clave externa	Tipo
ID	Sí	No	ENTERO
fecha_inicio	No	No	ENTERO
fecha_fin	No	No	ENTERO
asunto	No	No	CADENA DE CARACTERES
contenido	No	No	TEXTO
id_usuario_envía	No	Sí	ENTERO
borrado_en_origen	No	No	ENTERO
id_usuario_recibe	No	Sí	ENTERO
borrado_en_destino	No	No	ENTERO

Relaciones

RolContienePermiso (Rol - Permiso)

Atributo	Clave primaria	Clave externa	Tipo
ID_permiso	Sí	Sí	ENTERO
ID_rol	Sí	Sí	ENTERO

4.4 Normalización

La normalización de la base de datos es un proceso que consiste en aplicar una serie de reglas para comprobar que el modelo relacional cumple una serie de estándares para que la información sea definida sin redundancia y sin pérdida de información., para facilitar posteriormente el funcionamiento del sistema. Para ello se comprueba si el modelo de base de datos cumple con al menos las 3 primeras formas normales.

El esquema está en Primera Forma Normal (1FN):

- Todos los atributos son «atómicos»
- Todas las tablas contienen una clave primaria única, que no contiene valores nulos.
- No existen columnas redundantes.

El esquema está en Segunda Forma Normal (2FN):

- Los campos no clave dependen funcionalmente de la clave.

El esquema está en Tercera Forma Normal (3FN):

- No existe ninguna dependencia transitiva entre los atributos que no son clave.

Por lo tanto, ya podemos asegurar que el modelo de entidad-relación que hemos definido está normalizado.

Por otro lado, podría pensarse que existen dependencias entre los atributos “id_usuario_propone” y “autor”, y entre los atributos “id_usuario_publica” y “editor” de la tabla Recurso. Pero tras hacer un análisis de la situación, se llega a la conclusión de que los 4 campos son necesarios para casos como el siguiente:
Un fichero PDF de una escritora ajena a la plataforma, que un usuario de la plataforma decide subir como recurso. El campo “id_usuario_propone” tendría el valor del id del usuario de la plataforma que sube el recurso; el campo “id_usuario_publica” tendría el valor del id del usuario Experto de la plataforma que lo publica; el campo “autor” sería el nombre de la escritora del libro; el campo Editor, el nombre de la editorial que publicó la edición que se está subiendo.

4.5 Diseño de interfaces de usuario

Para el diseño de las vistas, se ha seguido el ejemplo de las plantillas de paneles de administración que hay disponibles en la red para realizar aplicaciones web de este tipo. Concretamente he seguido el patrón que muestra la plantilla SB Admin 2 [19], de uso muy extendido.

Este tipo de diseño sencillo cumple con el RNF2, ya que es muy común en otras aplicaciones, por lo que los usuarios estarán habituados a ella de forma rápida. Además, un diseño así es fácil de desarrollar de manera responsive, puesto que el menú lateral se puede convertir en desplegable en los dispositivos móviles, cumpliendo con el RNF3. Para cumplir con el RFN6, el número de componentes por vista se ha reducido al mínimo necesario para que cargue en el navegador de manera rápida y no sea necesario el uso de librerías externas.

Estas son las interfaces:

Login

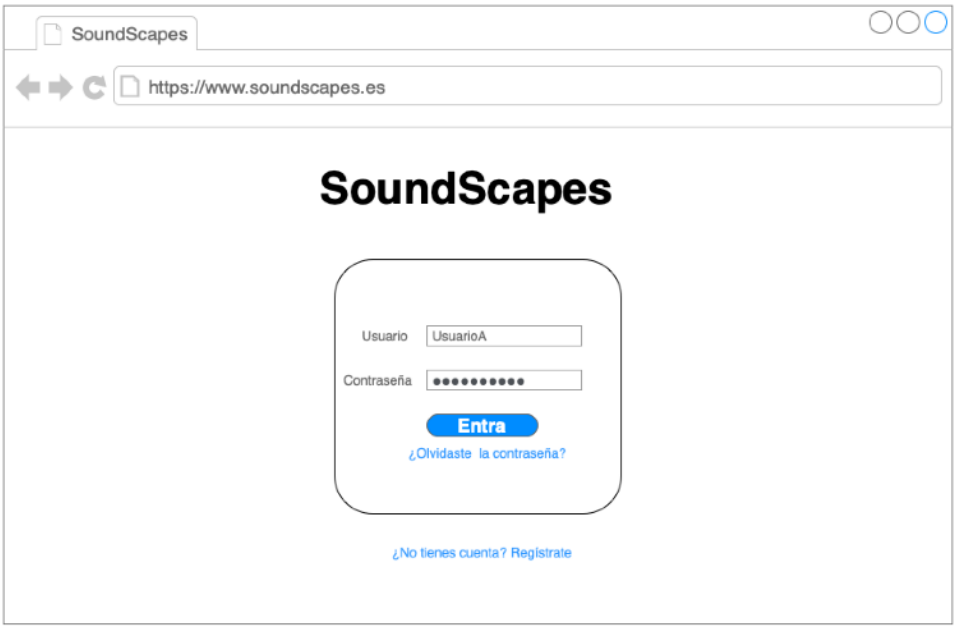


Figura 4.2: Vista de Login

Página de acceso identificado a la plataforma. Se ha diseñado siguiendo el diseño simple que tienen la gran mayoría de aplicaciones web en la actualidad, para facilitar la usabilidad (RNF2), Está formada por un pequeño formulario en la que el usuario introduce sus datos de acceso (usuario y contraseña) y pulsa en Entra para acceder, ejecutando la acción del RF1. También incluye un enlace para restaurar la contraseña y otro para acceder al formulario de registro.

Registro

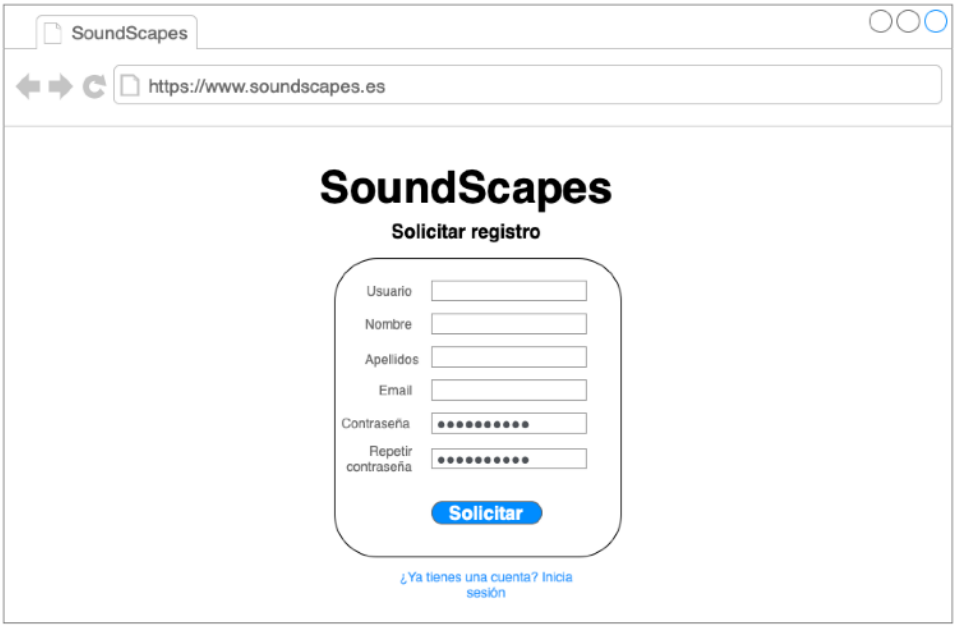


Figura 4.3: Vista de Registro

Página para solicitar el registro en la plataforma. Diseñada de una manera sencilla para facilitar la usabilidad (RNF2), consta de un formulario en el que el usuario introduce sus datos (RD1) para solicitar el registro, permitiendo la acción descrita en RF5. También incluye un enlace para acceder a la pestaña de login.

Editar perfil

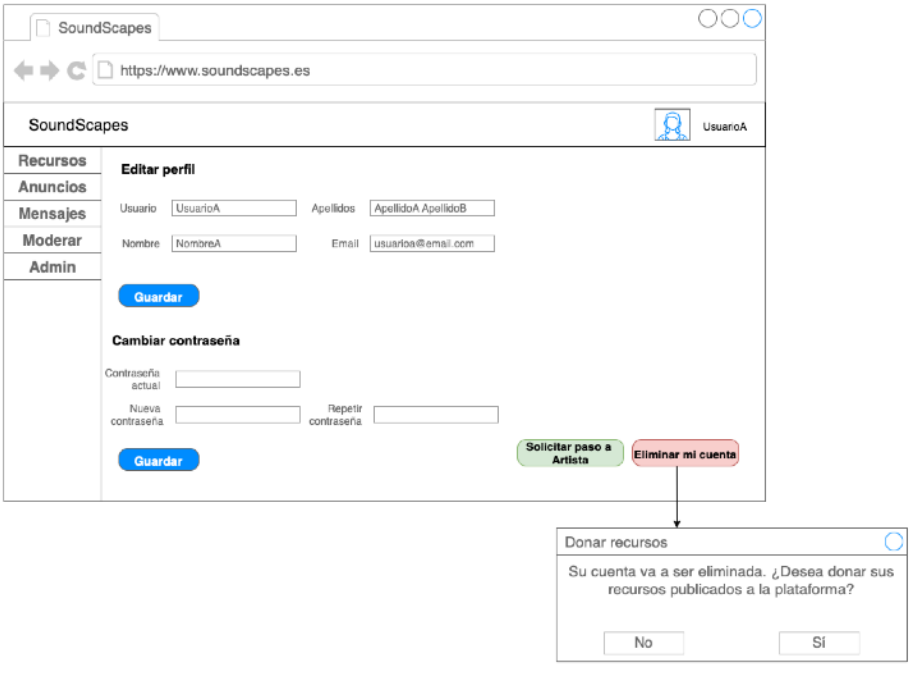


Figura 4.4: Vista de Editar Perfil

Página para editar el perfil del usuario, para permitir la acción del RF7. Se accede pulsando en el botón del avatar situado la parte derecha de la barra cabecera, siguiendo el patrón visto en la mayoría de las aplicaciones web que requieren de identificación. Consta de dos formularios diferenciados, uno para editar la información personal y otro para editar la contraseña de la cuenta. Cumple con el criterio de sencillez ya que solo incluye campos de formulario con sus correspondientes botones de Guardar. Los campos de información personal vendrán rellenos con la información actual del perfil. Además, se incluyen dos botones para solicitar el paso a Artista (RF39) y para eliminar la cuenta (RF17).

Recursos -> Explorar (1)

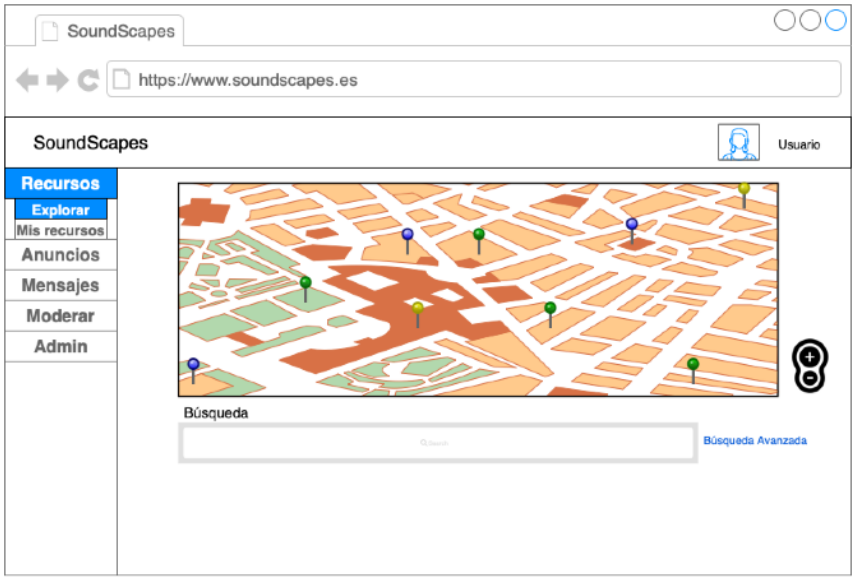


Figura 4.5: Vista de Explorar (1)

Pagina que permite explorar recursos en el mapa (RF3) y buscar recursos (RF2). El mapa que aparece en la parte superior muestra la región del Sahara Occidental, donde se ubicarán todos los recursos en base a sus coordenadas geográficas, y aparecerán como marcadores. El mapa tendrá un panel con botones para hacer zoom, aunque también se puede navegar usando directamente el ratón o el teclado (como ocurre en otras aplicaciones web con mapas como Google Maps). Debajo del mapa hay un cuadro de búsqueda rápida y un botón para abrir el formulario de búsqueda avanzada, descrito abajo.

Recursos -> Explorar (2)

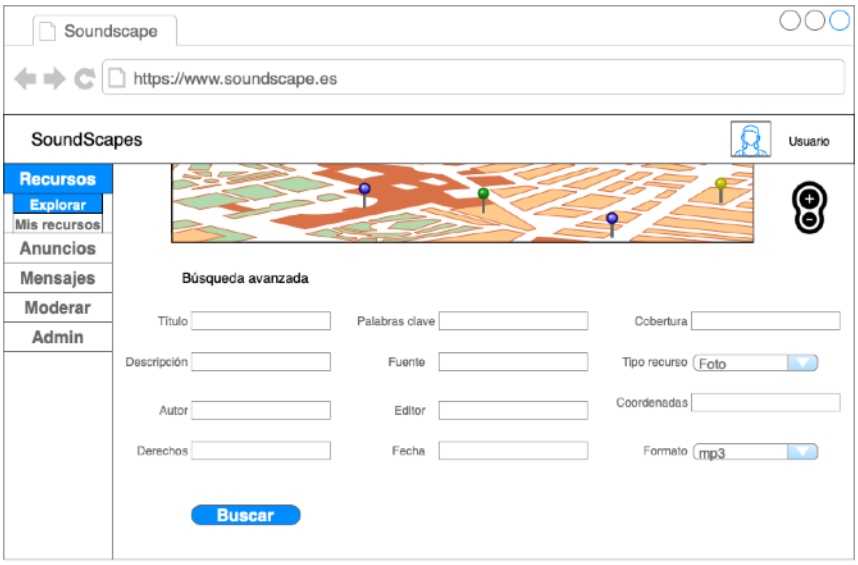


Figura 4.6: Vista de Explorar (2)

En esta imagen se muestra la misma pantalla anterior, pero con el formulario de búsqueda avanzada abierto. Este formulario consta de entradas para que el usuario añada los valores a buscar sobre los campos del recurso, que corresponden a los valores de metadatos almacenados en la plataforma (RD4). Al pulsar en buscar, ya sea usando esta búsqueda avanzada o la búsqueda sencilla mostrada en la imagen anterior, aparecerá un listado con los recursos coincidentes con la búsqueda, descrito en la siguiente imagen.

Recursos -> Explorar (3)

Figura 4.7: Vista de Explorar (3)

En esta imagen se muestra la misma pantalla anterior, pero con el listado de recursos resultado de una búsqueda. Cada ítem de la lista tendrá información básica del recurso a modo de resumen y tendrá un botón “Ver recurso” para acceder a la pantalla de visualización del recurso, donde se verán todos los metadatos y el recurso en sí mismo (RF4). Este listado se ha diseñado así para permitir al usuario ver los resultados de su búsqueda sin abrumarlo con demasiada información y que se destaque la información más relevante, que permita identificarlos identificarlos de manera sencilla.

Recursos - Mis recursos

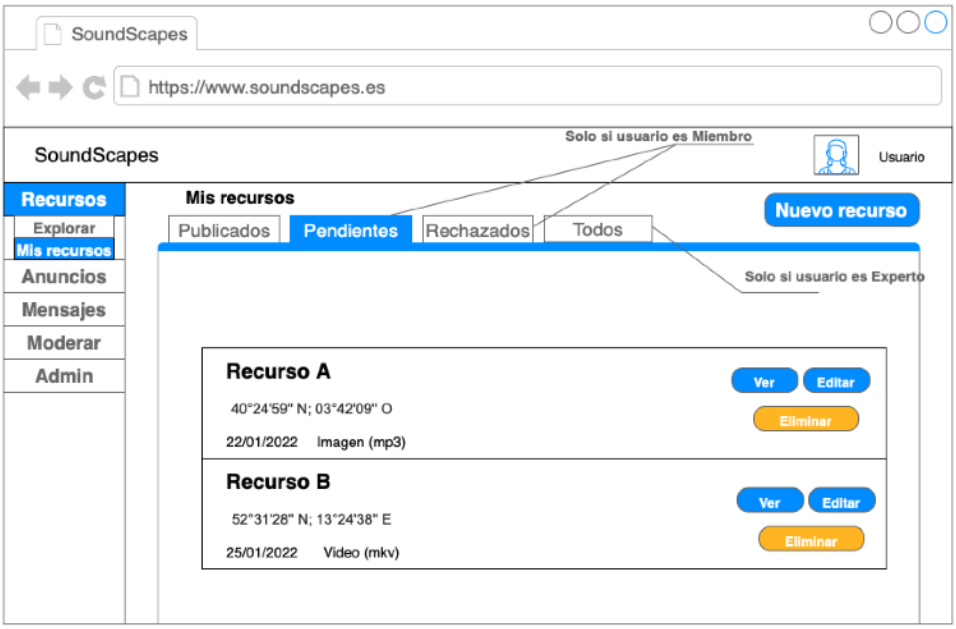


Figura 4.8: Vista de Mis recursos

En esta imagen se muestra la vista de Mis Recursos, en la que se muestran los recursos relacionados con el usuario. Debido a que dependiendo del tipo de usuario se relacionaran diferentes tipos de recursos (Publicados, Pendientes, Rechazados, Todos) se ha optado por una vista con 4 pestañas, además de un botón de Nuevo Recurso (para acceder a RF9 si el usuario es Miembro, o a RF18, si es Artista o Experto). Cada pestaña mostrará el mismo tipo de listado, en el que cada ítem representa un recurso. Este listado se ha diseñado así para permitir al usuario ver los recursos sin abrumarlo con demasiada información y que se destaque la información más relevante y pueda identificarlos de manera sencilla.

- La pestaña Publicados se muestra a usuarios Miembro, Artista y Experto. En ella están aquellos de sus recursos que encuentran en estado Publicado.
- La pestaña Pendientes se muestra a usuarios Miembro. En ella están aquellos de sus recursos que han sido propuestos pero que aún no han sido aceptados por un usuario tipo Experto.
- La pestaña Rechazados se muestra a usuarios Miembro. En ella están aquellos de sus recursos que han sido propuestos pero han sido rechazados por un usuario de tipo Experto.
- La pestaña Todos se muestra a usuarios Experto. En ella están todos los recursos del resto de usuarios de la plataforma.

Los botones que incluye cada ítem de la lista son:

- Ver: permite visualizar el recurso, tanto sus metadatos como el contenido (RF4).

- Editar: permite editar los metadatos o el recurso en sí mismo. Si el usuario es Miembro y el recurso ya se encuentra publicado, esto lleva a la acción descrita en RF11, y si no lo está, a RF10. En el caso de usuarios Artista o Experto lleva a la acción descrita en RF19. Si la pestaña es Todos, permite la acción definida en RF22.
- Eliminar: permite eliminar el anuncio. Si el recurso se encuentra en las pestañas de Publicados, Pendientes o Rechazados, permitirá ejecutar la acción descrita en RF12. Si es en la pestaña Todos, permite ejecutar RF23.

Recursos - Nuevo recurso

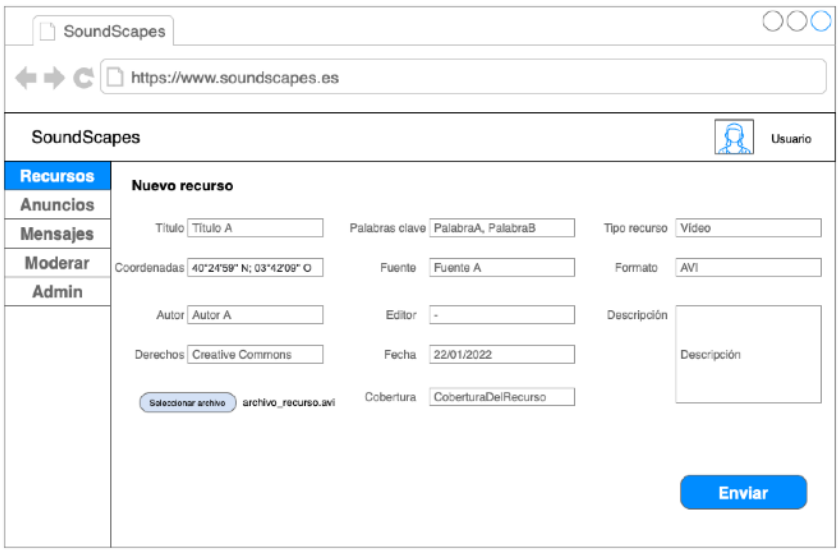


Figura 4.9: Vista de Nuevo recurso

Esta imagen describe la vista desde la que se propone (si el usuario es Miembro, RF13) o se publican (si el usuario es Artista o Experto, RF18) un recurso. Para ello aparece un formulario donde agregar los metadatos del recurso (RD4) y un botón para seleccionar el archivo multimedia. Los campos vendrán rellenos con información por defecto para facilitar el uso de la plataforma a usuarios inexpertos (RNF2).

Recursos - Editar recurso

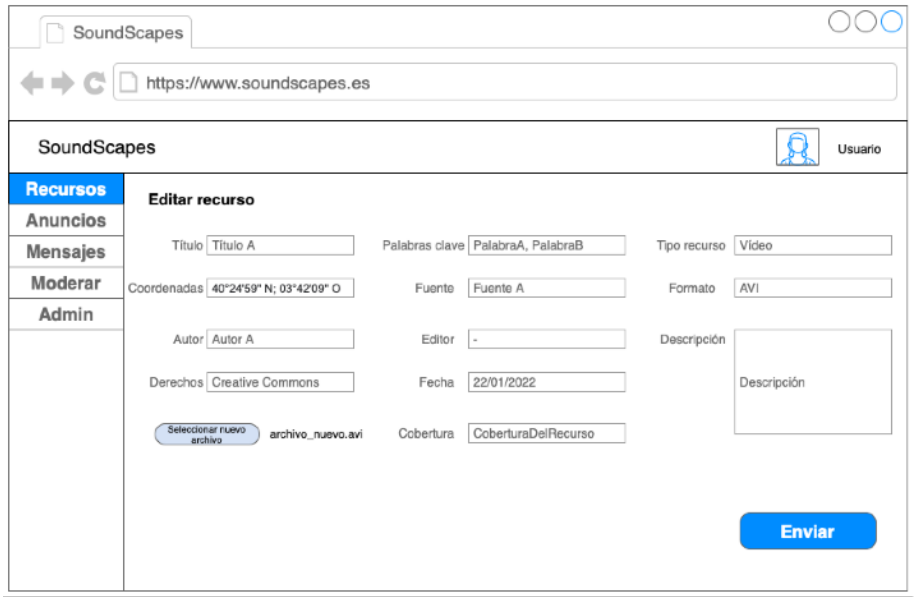


Figura 4.10: Vista de Editar recurso

Esta imagen describe la vista desde la que se propone una edición (si el usuario es Miembro y el recurso está ya publicado, RF11) o se edita directamente (RF10, RF19, RF22) un recurso. Para ello aparece un formulario que vendrá relleno con los metadatos del recurso (RD4) y un botón para seleccionar el archivo multimedia. El formulario es similar al de nuevo recurso, para que resulte más familiar y evitar cambio en la disposición de elementos.

Recursos - Visualizar

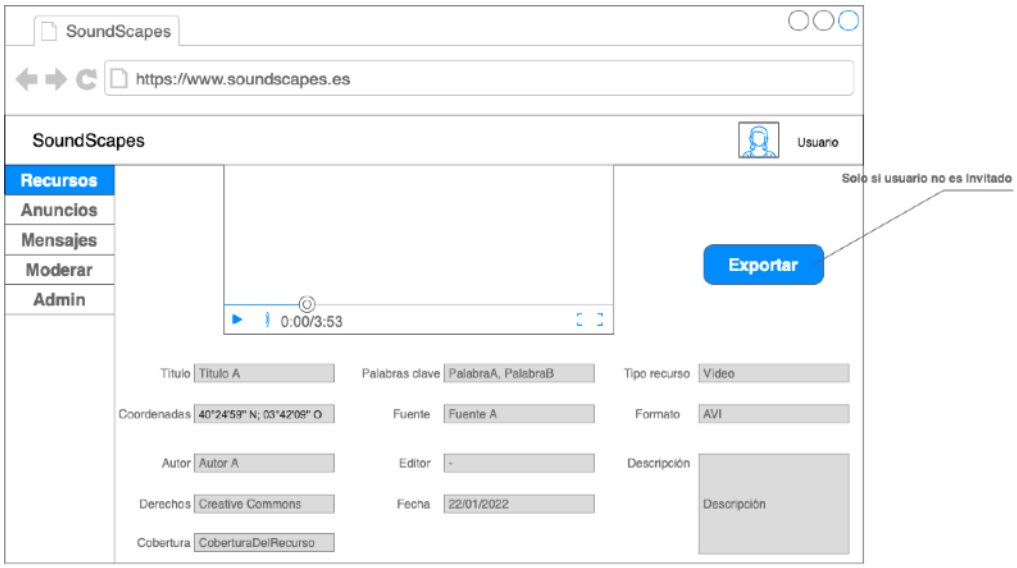


Figura 4.11: Vista de Visualizar

Esta imagen describe la pantalla usada para visualizar un recurso, para cumplir con el RF4. Se muestra un formulario similar al de Nuevo Recurso y Editar recurso, con la diferencia que los campos no son editables. En la parte superior se incluye un visualizador del recurso, que será de un tipo u otro (reproductor de video, reproductor de audio, visualizador de imagen o visualizador de texto) dependiendo del formato del archivo. A la derecha aparece un botón para exportar el recurso, para permitir la acción descrita en el RF8 (solo si el usuario no es Invitado).

Anuncios - Tablón

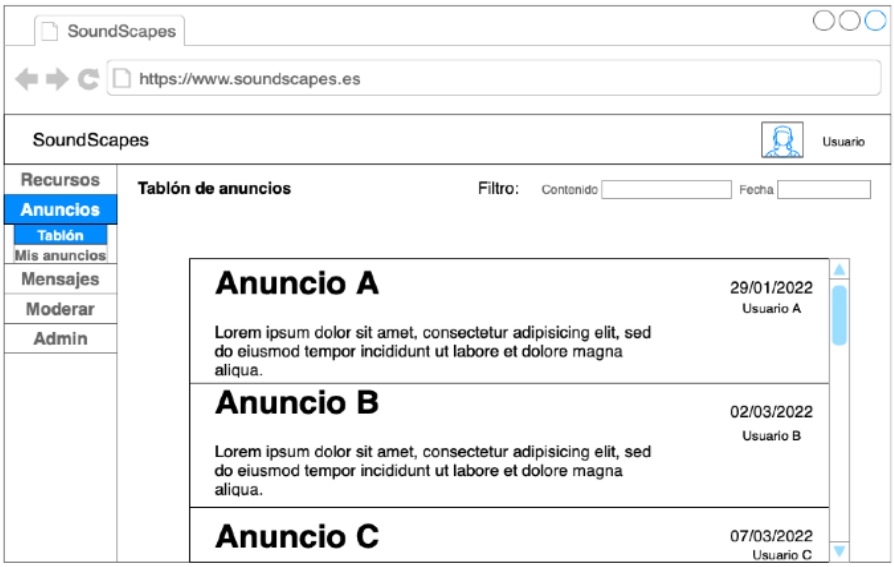


Figura 4.12: Vista de Tablón

Esta imagen muestra la vista de tablón de anuncios, donde se muestran todos los anuncios publicados por usuarios en el sistema. Aparecen como un listado, en el que se ha intentado mantener un diseño similar al de recursos para facilitar el uso y el periodo de adaptación a la plataforma (RNF2). En los ítems de la lista aparece toda la información el anuncio (RI5), ya que solo se compone de título, cuerpo, usuario que lo ha publicado y fecha de publicación. Arriba a la derecha aparecen dos campos para filtrar los anuncios, por contenido (en el título o contenido) o por fecha.

Anuncios - Mis anuncios

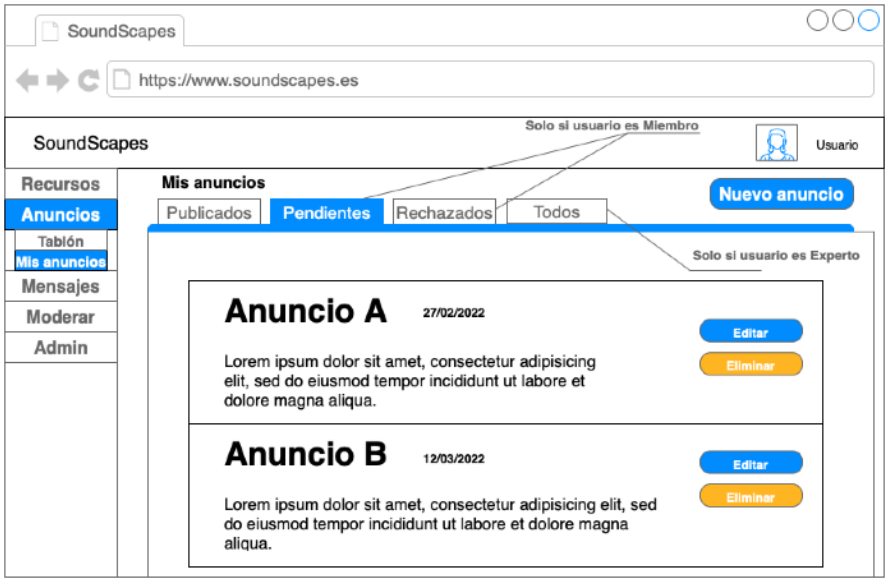


Figura 4.13: Vista de Mis anuncios

En esta imagen se muestra la vista de Mis Anuncios, en la que se muestran los anuncios relacionados con el usuario. Debido a que dependiendo del tipo de usuario se relacionaran diferentes tipos de recursos (Publicados, Pendientes, Rechazados, Todos) se ha optado, al igual que en la pantalla de “Mis recursos” por una vista con 4 pestañas, además de un botón de Nuevo Anuncio (para acceder las acciones descritas en los requisitos RF13 y RF20). Cada pestaña mostrará el mismo tipo de listado, en el que cada ítem representa un anuncio. Este listado se ha diseñado exactamente igual (en funcionalidad y apariencia) que el descrito en la pantalla de “Mis recursos”.

Anuncios - Nuevo anuncio

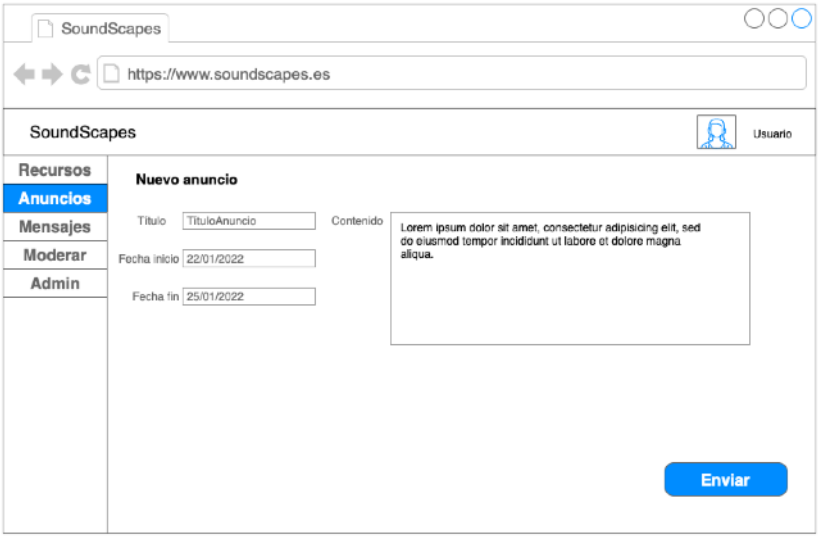


Figura 4.14: Vista de Nuevo anuncio

En esta imagen se muestra la vista de creación de un anuncio, para cumplir con los requisitos RF13 (usuarios Miembro) y RF20 (usuarios Artista y Experto). Se ha diseñado una vista muy sencilla, con un formulario en la que el usuario introduce los campos de un anuncio (RD5) y pulsa el botón de enviar, proponiéndose o publicándose el anuncio.

Anuncios - Editar anuncio

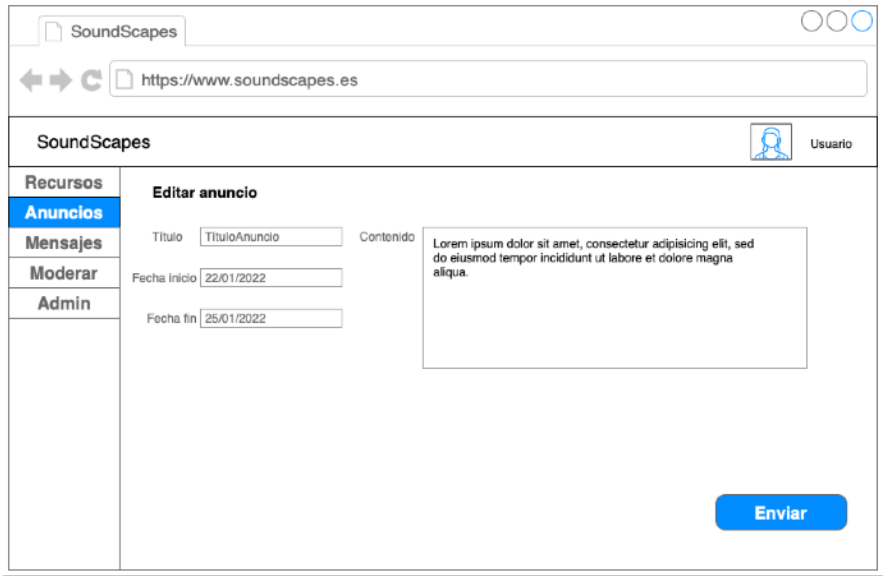


Figura 4.15: Vista de Editar anuncio

En esta imagen se muestra la vista de edición de un anuncio, para cumplir con los requisitos RF14 (usuarios Miembro para anuncios no publicados), RF15 (usuarios Miembro para anuncios publicados), RF21 (usuarios Artista y Experto si se trata de un anuncio propio) y RF24 (usuarios Experto para anuncios que no son suyos). Se ha diseñado una vista muy sencilla, exactamente igual que la de Nuevo anuncio, con un formulario relleno con los datos actuales del anuncio (RD5) y pulsa el botón de enviar cuando los ha editado.

Moderar - Recursos

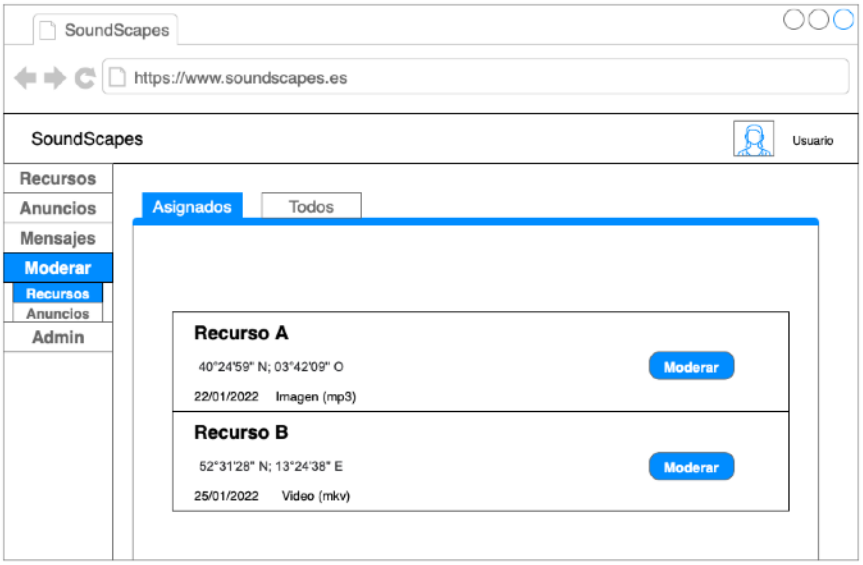


Figura 4.16: Vista de Recursos

Esta pantalla se ha diseñado para permitir a los usuarios tipo Experto ver los recursos pendientes de moderación. El diseño incluye dos pestañas, una para mostrar los recursos pendientes de moderación de los usuarios que tiene asignados el Experto, y otra con el resto de recursos pendientes, ya que también puede acceder a ellos para moderarlos. Cada ítem de la lista muestra información básica del recurso para poder identificarlo rápidamente, y un botón “Moderar”, que le llevará a la siguiente pantalla descrita.

Moderar - Moderar recurso

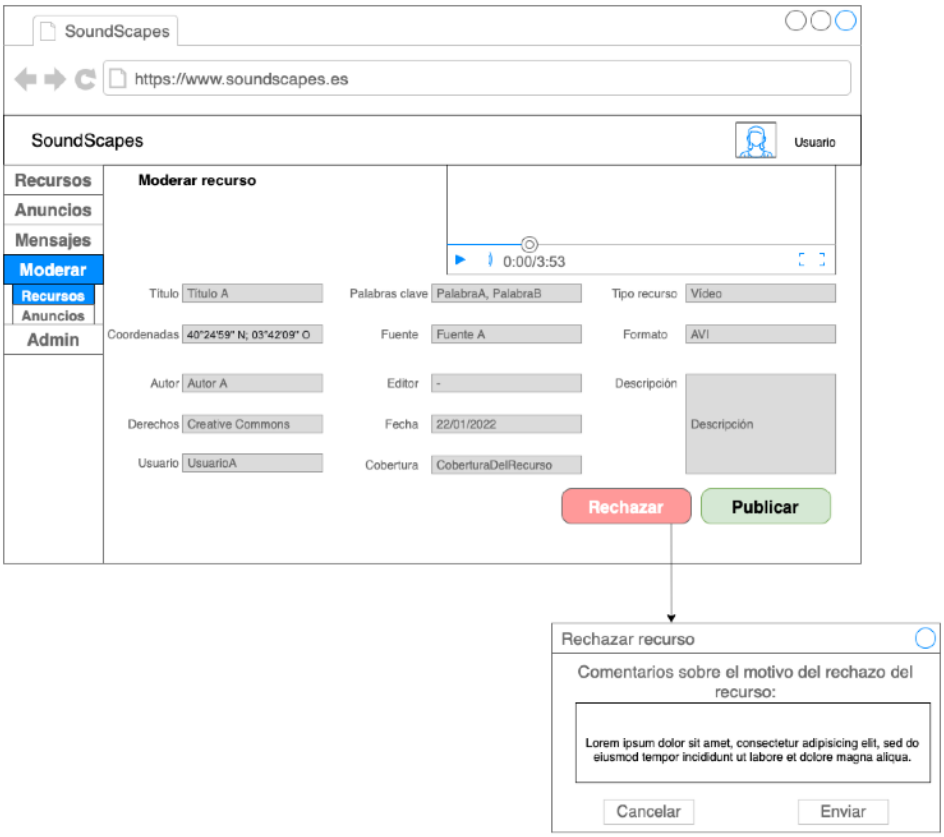


Figura 4.17: Vista de Moderar recurso

Desde esta vista un usuario Experto puede aceptar (RF26) o rechazar (RF28) un recurso propuesto por un usuario de tipo Miembro. Para mantener uniforme el diseño de la plataforma, la estructura es muy similar a la vista de visualización de recurso. En este caso, los campos del formulario están en modo solo lectura, y se incluyen dos botones: uno para rechazar y otro para aceptar. En caso de pulsar en rechazar, el usuario verá un pop-up como se muestra en la figura con el que podrá añadir un comentario indicando los motivos del rechazo.

Moderar - Anuncios

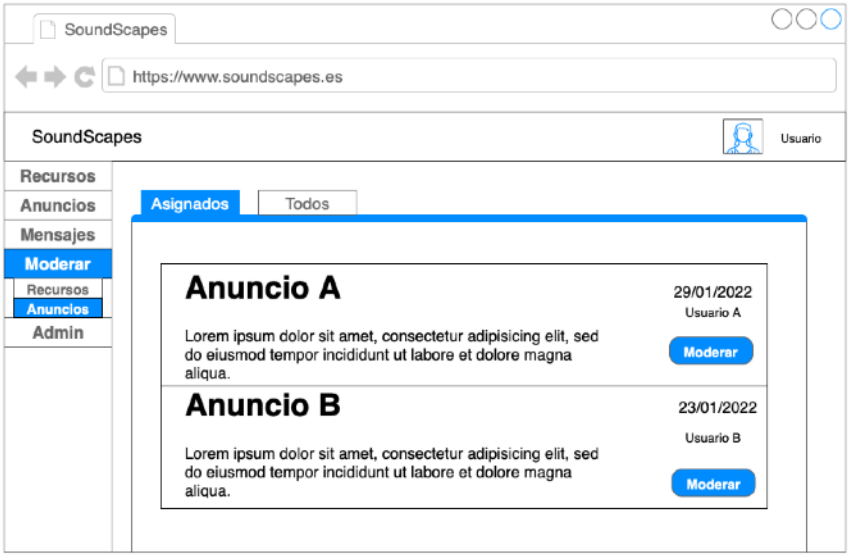


Figura 4.18: Vista de Moderar anuncios

Esta vista se ha diseñado para permitir a los usuarios tipo Experto ver los anuncios pendientes de moderación. El diseño incluye dos pestañas, una para mostrar los anuncios pendientes de moderación de los usuarios que tiene asignados el Experto, y otra con el resto de anuncios pendientes, ya que también puede acceder a ellos para moderarlos. Cada ítem de la lista muestra, al igual que en la vista de Mis Anuncios, la información del anuncio y un botón “Moderar”, que le llevará a la siguiente vista descrita.

Moderar - Moderar anuncio

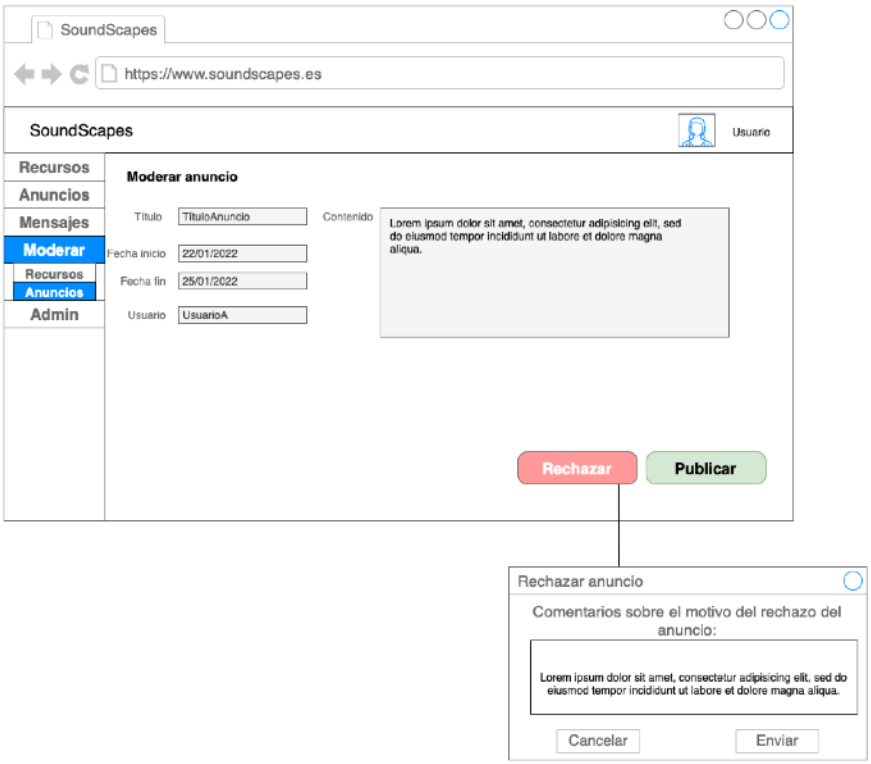


Figura 4.19: Vista de Moderar anuncio

Desde esta vista un usuario Experto puede aceptar (RF27) o rechazar (RF29) un anuncio propuesto por un usuario de tipo Miembro. Para mantener uniforme el diseño de la plataforma, la estructura es muy similar a la vista de creación y edición de anuncio. En este caso, los campos del formulario están en modo solo lectura, y se incluyen dos botones: uno para rechazar y otro para aceptar. En caso de pulsar en rechazar, el usuario verá un pop-up con el que podrá añadir un comentario indicando los motivos del rechazo.

Mensajes - Buzón

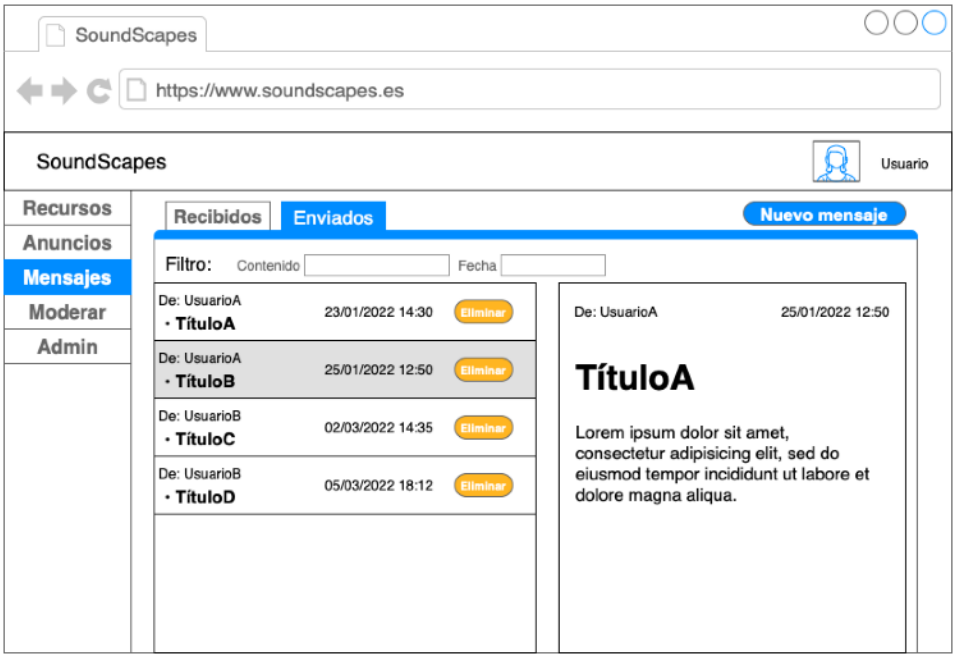


Figura 4.20: Vista de Buzón

En esta imagen se muestra el buzón de mensajes. Se muestran dos pestañas (como en el resto de la plataforma cuando hay dos o mas secciones similares), una para visualizar los mensajes recibidos y otra para los mensajes enviados. También se incluye un botón “Nuevo mensaje” para acceder a la vista de redacción de un mensaje nuevo. En cada una de las pestañas, se ha optado por un diseño similar al de otras aplicaciones de correo: a la izquierda un listado de los mensajes con información básica (RF34) y un botón para eliminar el mensaje (RF37), y a la derecha un visualizador del mensaje seleccionado (RF35). Cada pestaña también incluye dos campos para filtrar por contenido (asunto y contenido del mensaje) o por fecha, para cumplir completamente con el requisito RF34.

Mensajes - Escribir mensaje

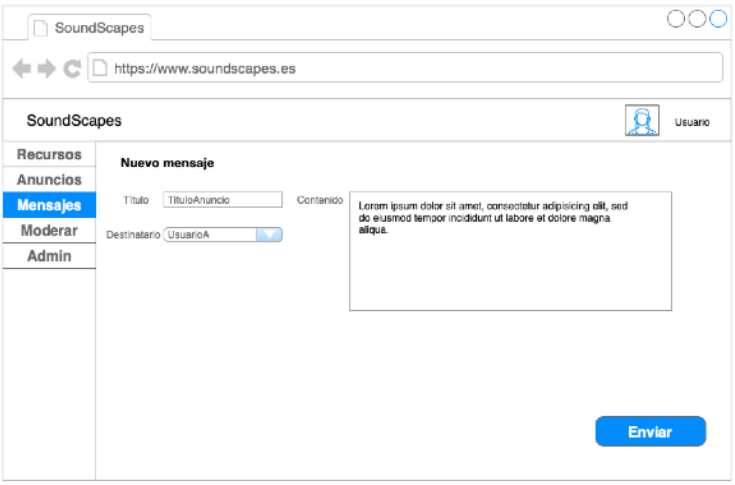


Figura 4.21: Vista de Escribir mensaje

En esta vista, a la que se accede mediante el botón “Nuevo mensaje” de la vista de del buzón, se ha añadido un formulario sencillo para redactar un mensaje, cumpliendo con el requisito RF36. El usuario introduce los datos del mensaje (RD6) y pulsa en el botón enviar.

Admin - Usuarios

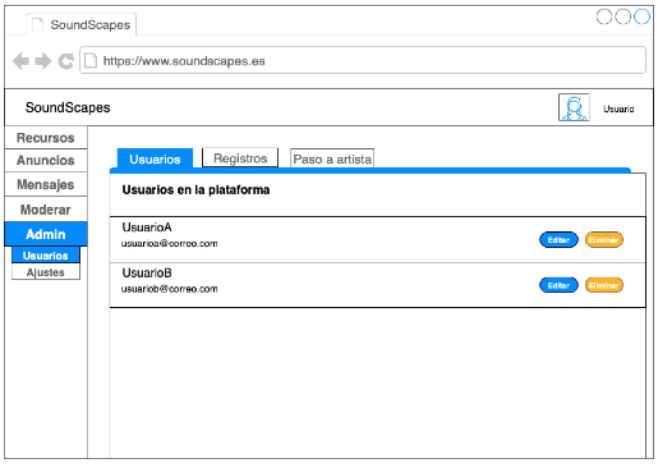


Figura 4.22: Vista de Usuarios (1)

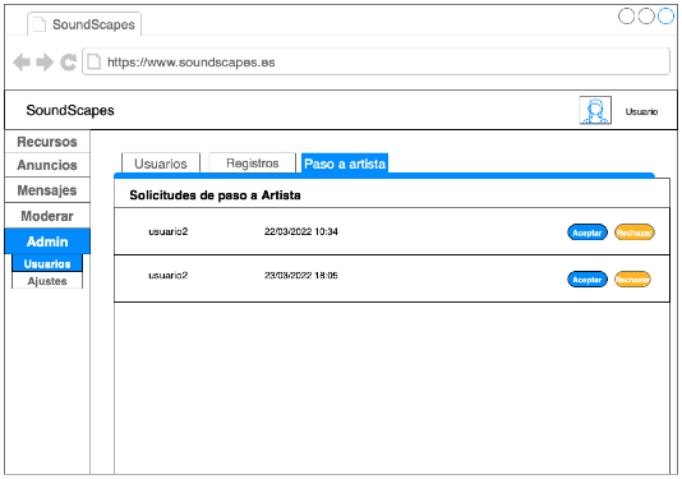


Figura 4.23: Vista de Usuarios (2)

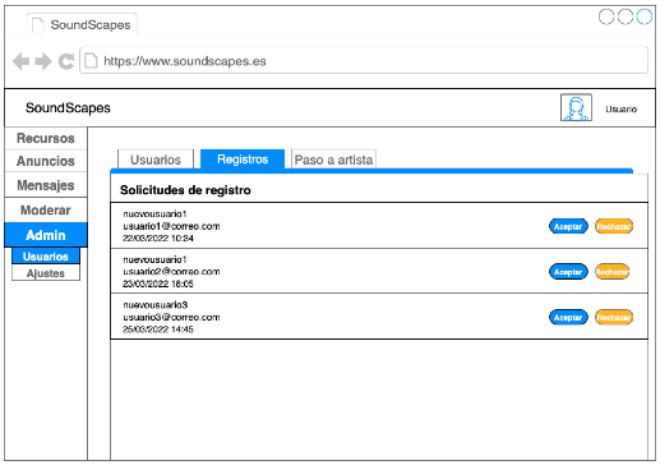


Figura 4.24: Vista de Usuarios (3)

En esta pantalla, que solo está accesible para usuarios tipo Administrador, se le mostrarán al usuario 3 secciones: una con todos los usuarios registrados en el sistema, otra con las solicitudes de registro y otra con las solicitudes de paso a artista. Se ha diseñado así permitiendo al usuario realizar operaciones sobre ellas desde una misma pantalla, sin necesidad de navegar de una a otra. En el primer listado cada ítem es un usuario, en el que se muestra su nombre de usuario y su dirección de email, ademas de los botones de editar (para cumplir con RF31) y eliminar (para cumplir con RF32). En el segundo listado cada ítem es una solicitud de registro, en el que se muestra el nombre de usuario, la dirección de email y la fecha de la solicitud; y dos botones: uno para aceptar el registro y otro para rechazarlo (RF30). En el tercer listado cada ítem es una solicitud de un usuario Miembro de pasar a usuario Artista, en el que se muestra el nombre del usuario, la fecha de la solicitud y dos botones: uno para aceptar la solicitud y otra para rechazarla (RF40).

Admin - Editar usuario

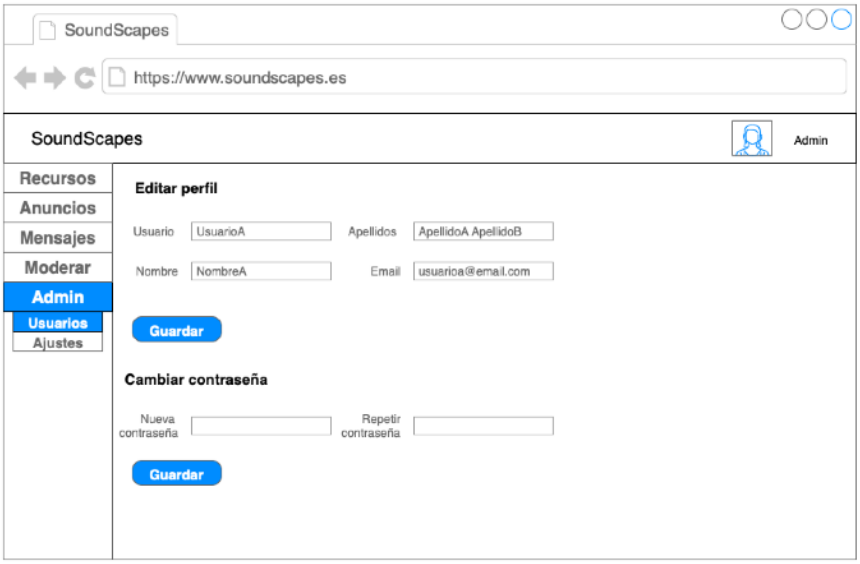


Figura 4.25: Vista de Editar usuario

Página para editar el perfil del usuario seleccionado, para permitir la acción del R31. La vista es exactamente igual que la diseñada para editar el perfil propio: consta de dos formularios diferenciados, uno para editar la información personal y otro para editar la contraseña de la cuenta. Cumple con el criterio de sencillez ya que solo incluye campos de formulario con sus correspondientes botones de Guardar. Los campos de información personal vendrán rellenos con la información actual del perfil.

Admin - Ajustes

Figura 4.25: Vista de ajustes

En esta vista, que solo está accesible para usuarios tipo Administrador, se le mostrará al usuario el único valor disponible para configurar la plataforma. En este caso, es la posibilidad de limitar la subida de recursos multimedia por disponer de poco espacio. Con esta vista se cumple el requisito RF33. Se ha optado por una vista completa para esta configuración de cara a facilitar el añadido de más configuraciones en posibles futuras versiones de la plataforma.

4.6 Arquitectura

La arquitectura de la aplicación, desde el punto de vista de separación de componentes, sigue un modelo cliente-servidor, que es el usado en la inmensa mayoría de las aplicaciones web actualmente y que consiste en una estructura distribuida que divide las tareas o cargas de trabajo entre los proveedores de un recurso o servicio, llamados servidores, y los solicitantes del servicio, llamados clientes [20]. Este tipo de diseño permite que el cliente sea totalmente independiente de la implementación del servidor, por lo que se pueden hacer diferentes tipos de clientes (web, móvil, escritorio, etc) de manera independiente, por diferentes equipos de desarrollo y con diferentes tecnologías. En este trabajo, tenemos el proyecto del API, que actúa como servidor, y el proyecto de cliente web, o frontend, que actúa como cliente realizando las peticiones.

Pero por otro lado, si la plataforma se ve toda como una unidad, la implementación sigue el patrón *MVC* [21]. En dicho modelo, intervienen 3 capas claramente diferenciadas: la capa de presentación (vista), la capa de negocio (controlador) y la capa de datos (modelo). En nuestro caso, la función de la capa de presentación la realiza el cliente web, la capa de de negocio el API y la capa de datos también la realiza el API junto a la base de datos.

Capa de presentación.

La capa de presentación se llama comúnmente interfaz gráfica. Es el componente encargado de hacer intermediario entre el usuario final y los componentes de la capa de negocio, permitiendo la entrada y salida de información.

Este componente contiene la lógica necesaria para recuperar los datos de la API REST, presentar la información a los usuarios, recibir la entrada de datos e interacciones de los usuarios, manipularlas y enviarlas a la API REST. Es agnóstico de la implementación interna de la fuente de datos, solo ha de conocer el diseño de la API: qué rutas hay disponibles (endpoints) y qué formato tienen (esquema de atributos).

En esta aplicación se trata de un cliente web cuyo código estará alojado en un servidor web que se ejecutará en los navegadores web de los usuarios que accedan a la app, ya sea en ordenadores tradicionales o en dispositivos móviles.

Capa de negocio

La capa de negocio es la capa que se encarga de toda la lógica de los servicios que ofrece la aplicación. Recibe los datos enviados por los usuarios a través del cliente web, procesa dicha información, la almacena y atiende a las solicitudes de información del cliente web recuperando y enviando dicha información.

Hay diferentes patrones de diseño a la hora de diseñar la capa de negocio de una aplicación como esta. Se ha elegido el extendido diseño de API REST. En este diseño, se exponen a través de una serie de rutas (endpoints) que aceptan comunicación HTTP. Cuando un cliente accede a una de estas rutas, se activa el mecanismo correspondiente para atender la petición, qué generalmente es una operación CRUD (crear, actualizar, eliminar o consultar un objeto de las entidades que tiene la plataforma) [22]. Esta capa se vale de la capa de datos para la persistencia de la información, comunicándose con ella a través de un driver.

El código de esta aplicación estará alojando en un servidor web que expone a través de internet los endpoints, permitiendo que cualquier cliente web acceda a las rutas. Para que un cliente web

active una ruta de forma satisfactoria, deberá estar debidamente autorizado. Para esta autorización, se ha de implementar un mecanismo de autenticación en esta capa. Para esta aplicación, se ha optado por un sistema de tokens web. Esto consiste en que un cliente web solicita un token de autenticación a la plataforma, suministrando su usuario y contraseña. Si el sistema valida correctamente estos datos, genera un token firmado mediante un algoritmo criptográfico y con una validez temporal y lo envía al cliente web. Cada vez que el cliente realiza una petición a cualquier endpoint del API, incluye en las cabeceras el token [23]

Capa de datos

La capa de datos es el componente donde se almacena de manera persistente la información. En la inmensa mayoría de los casos, es la base de datos. Para esta aplicación se va a utilizar una base de datos relacional, ya que los requisitos de datos que tiene el diseño es totalmente estructurado y sencillo.

En una base de datos relacional se diseñan tablas que contienen una serie de columnas. En dichas tablas se almacena la información en forma de filas, donde cada fila es un registro de la información. La información de manipula mediante consultas SQL, ya sea para consultar, editar, añadir o eliminar los registros.

Para facilitar la comunicación entre la capa de negocio y la base de datos, se utiliza en la capa de negocio un driver que actúa de intermediario entre las dos tecnologías e implementa los métodos de acceso y transformación de los datos.

Resumen de arquitectura

El diseño de la arquitectura de la aplicación, que puede verse en la figura 4.27, va a ser un diseño típico de API REST para el *backend* (el servidor del modelo cliente-servidor), una aplicación web SPA (Single Page Application) para el *frontend* (el cliente del modelo cliente servidor), y una base de datos relacional para almacenar la información.

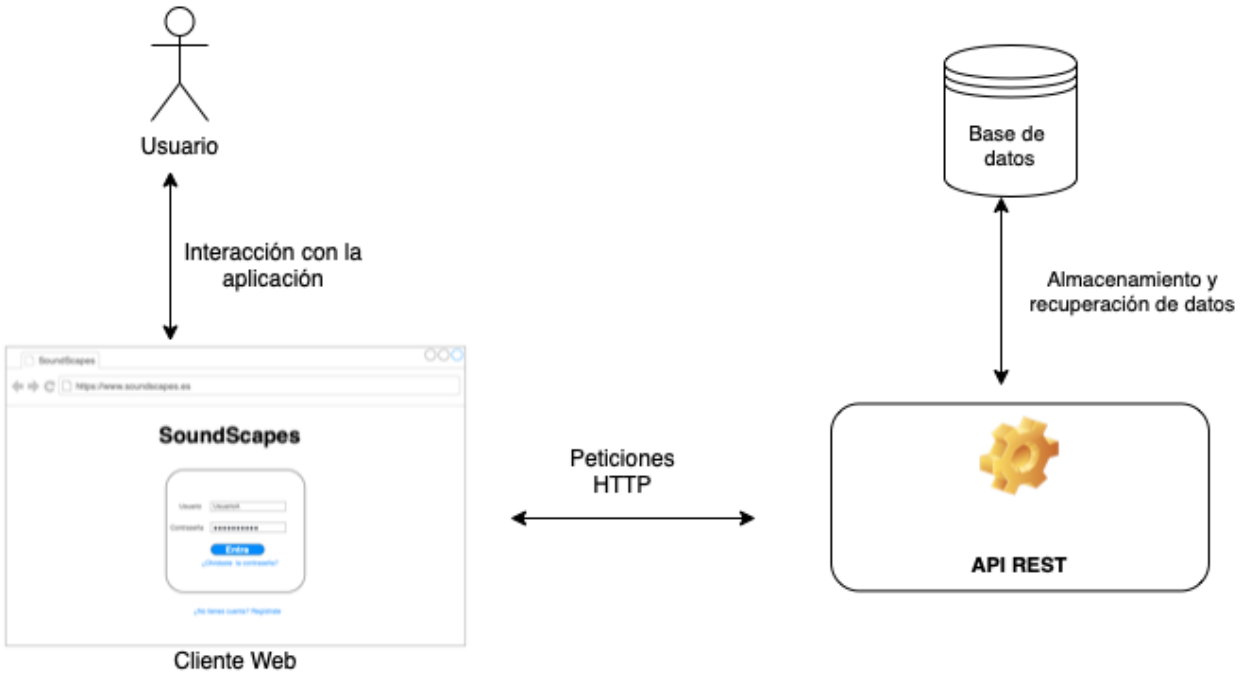


Figura 4.27: Esquema de arquitectura del sistema

5. Implementación

En esta sección se describe la implementación final de la aplicación siguiendo las pautas definidas en el diseño de la fase anterior. Primero se van a detallar las herramientas de desarrollo usadas; después como están estructurados los proyectos de los dos componentes principales: la API REST y el cliente web, mostrando partes principales, organización de ficheros y ejemplificando con fragmentos de códigos qué patrones se han utilizado.

5.2 Herramientas de desarrollo

Cliente web

En este caso, se ha optado por diseñar un cliente web utilizando Angular [24], un framework basado en JavaScript que permite la creación de estos clientes de manera totalmente independiente y agnóstica a la capa de negocio.

La alternativa a este tipo de herramientas, son los framework que permiten un diseño más acoplado entre interfaz web y capa de negocio, como pueden ser Django, Ruby on Rails o Spring. En este tipo de framework, usando el mismo proyecto, se diseñan todos los componentes de la aplicación..

Las ventajas de usar un framework de clientes web como Angular son:

- Más herramientas y funcionalidades específicas para web.an
- Plugins para no repetir código
- Son más flexibles a la hora de diseñar soluciones complejas.
- Se actualizan continuamente.

Las ventajas de usar un framework completo como Django son:

- No es necesario aprender otra tecnología, ya que se usa la misma para backend y frontend.
- Suele ser más sencillo de aprender.
- La instalación y despliegue de la app es mucho más sencilla.

Para una aplicación de diseño sencillo como la desarrollada en este proyecto, el uso de un framework completo como estos, o de elegir un framework específico para la creación de componentes web es cuestión de preferencias personales, ya que ambas opciones son muy usadas en la actualidad y están bien documentadas.

Dentro de los framework específicos para frontend hay 3 opciones principales: Angular, ReactJS y VueJS. Las 3 opciones están muy extendidas en la actualidad a nivel comercial. Todas son gratuitas y open source, y ninguna tiene una ventaja o inconveniente clave para nuestro caso. Se ha optado por Angular debido a mi experiencia usando esa tecnología, por lo que hará más fácil la parte de desarrollo. Angular está desarrollado por Google, por lo que la documentación de sus funciones y el soporte está garantizado [25].

API REST

Al igual que con la capa de presentación, para esta capa también se puede optar por framework completos que incluyen un gran número de herramientas o frameworks específicos para hacer un API REST.
La mayoría de los frameworks completos como Django o Ruby on Rails, incluyen herramientas para el diseño de APIs REST, pero existen alternativas más sencillas y livianas para el diseño de estas.

Se ha optado por el microframework de desarrollo de APIs REST “Flask”. Está desarrollado en Python, y entre sus principales ventajas se incluyen [26]:

- Al ser un “micro”framework, es muy sencillo y ágil empezar a desarrollar una API.
- Facilidad de uso, ya que no es necesario aprender una gran cantidad de elementos o técnicas para desarrollar endpoints básicos como los de esta aplicación.
- Se puede extender usando plugins para determinadas funciones. En nuestro caso, usuarios Flask-SQLAlchemy como ORM, para interactuar de manera sencilla con la base de datos.
- Es OpenSource y su uso está muy extendido, por lo que hay documentación y ejemplos para cualquier problema que surja durante el desarrollo de esta aplicación.

Existen otros microframework muy similares, tanto en Python (FastAPI, Tornado) como en otros lenguajes (Express, Spring REST). Se ha optado por Flask debido a mi experiencia usando esa tecnología, por lo que hará más fácil la parte de desarrollo.

Base de datos

De entre todas las alternativas posibles de base de datos relacionales, para esta aplicación se ha optado por PostgreSQL por estas características [27]:

- Es OpenSource y de licencia totalmente gratuita.
- Está demostrado que tiene una gran escalabilidad y buen rendimiento.
- Es una base de datos relaciónale que usa el estándar SQL.
- Incluye pgAdmin, una herramienta para administrar la base de datos desde un panel web.
- Su uso está muy extendido, por lo que hay una gran cantidad de documentación.

Dentro de las base de datos relacionales, las principales alternativas son MySQL:

- MySQL, una de las base de datos más extendidas en la actualidad. Hay una gran cantidad de documentación sobre ella, pero el principal problema es que es propiedad de Oracle, y aunque incluye una licencia GPU, es necesario pagar para su uso comercial.
- Microsoft SQL Server. Pertenece a Microsoft, es de pago su uso comercial y no es de código abierto.
- SQLite, es una alternativa ligera al resto de opciones. No es un servicio separado sino que se integra a la aplicación, lo que disminuye la latencia. No tiene todas las características que tiene Postgre y es mucho menos robusta, por lo que se desaconseja su uso en producción.

Principalmente se ha elegido por tener una licencia GPU incluso para uso comercial y por su extendido uso, lo que permite que haya una gran cantidad de documentación para posibles problemas que surjan durante la etapa de desarrollo.

5.3 API REST

Tal como se ha definido en el apartado de arquitectura, la API REST es un proyecto utilizando el framework Flask. Este framework se encarga de proporcionar una serie de herramientas para el desarrollo de APIs de este tipo para que el desarrollador no se tenga que preocupar a bajo de nivel de tareas repetitivas.

La estructura del proyecto es la siguiente:

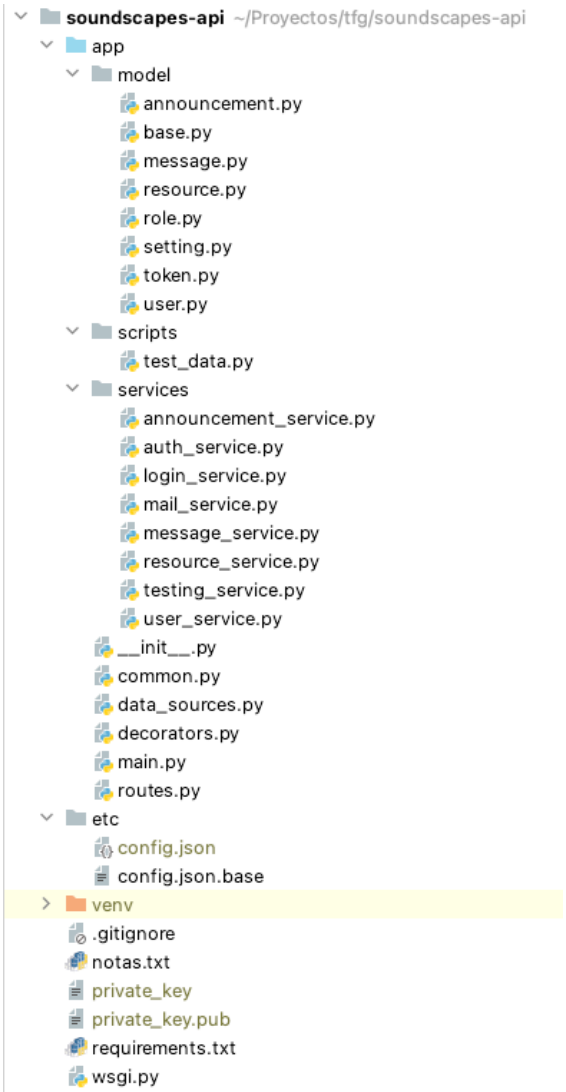


Figura 5.2: estructura del proyecto API REST

En el primer nivel, además de la carpeta app con el contenido de la lógica de la aplicación, que detallaré más adelante, encontramos los siguientes ítems:

- wsgy.py: es el punto de entrada de la aplicación, inicia la API y empieza a escuchar peticiones.
- private_key: es clave (pública y privada) RSA utilizada por el mecanismo de autenticación JWT, para firmar los tokens.

- .gitignore: fichero utilizado para evitar que entren en el control de versiones determinados archivos, como configuraciones o el entorno virtual.

- venv: entorno virtual de python. Los entornos virtuales se utilizan para tener separado el entorno de Python entre diferentes proyectos, para poder tener diferentes librerías en función de los requisitos de cada aplicación.

- requirements.txt: un listado de las librerías usadas por el proyecto y el número de versión instalado.

- etc: carpeta que contiene el fichero con las variables de configuración del proyecto.

La carpeta app es la más importante, ya que es la que contiene la mayor parte del código de la aplicación. La estructura es la siguiente:

model

Aquí se encuentra el modelado de datos, la capa que conecta con la base de datos. Se utiliza el framework SQLAlchemy para crear las clases que representan cada una de las tablas de la base de datos y las clases que actúan a modo de repositorio y de capa de acceso directo a base de datos. Existe una clase de cada uno de estos tipos por cada entidad del sistema, que todas heredan de las tres clases genéricas (Base) para evitar repetir código.

Las clases genéricas tienen este código:

```python
class BaseDao:

    def __init__(self, model):
        self.model = model

    def get_all(self):
        return db.session.query(self.model).all()

    def get_by(self, **kwargs):
        for key in kwargs.copy().keys():
            if key not in self.model.__table__.columns.keys():
                del kwargs[key]
        return db.session.query(self.model).filter_by(**kwargs).all()

    def get_all_by_filters(self, filters):
        return db.session.query(self.model).filter(and_(*filters)).all()

    def get_by_id_list(self, id_list):
        return db.session.query(self.model).filter(self.model.id.in_(id_list)).all()

    def get_by_id(self, id):
        return db.session.query(self.model).get(id)

    def get_distinct_by_field(self, field):
        return db.session.query(getattr(self.model, field)).distinct().all()

    def insert(self, obj):
        db.session.add(obj)
        db.session.commit()

    def insert_all(self, obj_list):
        db.session.bulk_save_objects(obj_list)
        db.session.commit()
```

Figura 5.3: clase BaseDao

```python
class BaseData:

    def __init__(self, dao):
        self.dao = dao

    def save(self):
        self.dao.make_commit()

    def get_all(self):
        return self.dao.get_all()

    def get_by(self, **kwargs):
        return self.dao.get_by(**kwargs)

    def get_by_id(self, id):
        return self.dao.get_by_id(id)

    def get_by_id_list(self, id_list):
        return self.dao.get_by_id_list(id_list)

    def get_distinct_by_field(self, field):
        return self.dao.get_distinct_by_field(field)

    def insert(self, obj):
        self.dao.insert(obj)

    def insert_all(self, obj_list):
        self.dao.insert_all(obj_list)

    def delete(self, obj):
        self.dao.delete(obj)
```

Figura 5.4: clase BaseData

```python
class BaseModel(db.Model):
    __abstract__ = True

    def to_dict(self):
        result = dict()
        for c in self.__table__.columns.keys():
            result[c] = getattr(self, c)

        return result
```

Figura 5.5: clase BaseModel

Por lo tanto, por cada entidad se definirá una clase por cada una de las genéricas, que posteriormente se instanciarán para manejar objetos. Por ejemplo: la clase Message, que se utiliza para representar instancias de mensajes; la clase MessageData, que se utiliza como repositorio en el resto del código para operaciones sobre mensajes; y la clase MessageDao, que se utiliza para el contacto directo con la tabla “Message” de base de datos.

```python
class Message(BaseModel):
    __tablename__ = 'message'
    id = Column(Integer, primary_key=True)
    subject = Column(String)
    body = Column(Text)
    creation_date = Column(Integer)
    deleted_in_src = Column(Boolean)
    deleted_in_dst = Column(Boolean)
    read = Column(Boolean)
    id_user_src = Column(Integer, ForeignKey('user_account.id'))
    id_user_dst = Column(Integer, ForeignKey('user_account.id'))

    def to_dict(self):
        base = super().to_dict()
        base['username_src'] = self.user_send.username
        base['username_dst'] = self.user_receive.username
        return base


class MessageData(BaseData):
    def __init__(self):
        super(MessageData, self).__init__(MessageDao())


class MessageDao(BaseDao):
    def __init__(self):
        super(MessageDao, self).__init__(Message)
```

Figura 5.6: implementación de las clases para Message

services

En esta carpeta se encuentra un fichero por cada uno de los dominios de la lógica de negocio. Cada servicio contiene los métodos que realizan las operaciones necesarias para responder a una petición que llega del cliente.

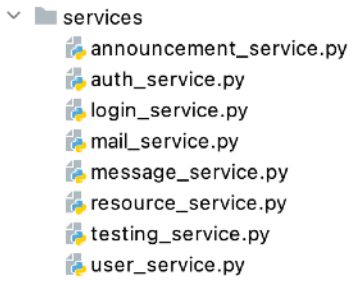


Figura 5.7: carpeta de servicios

En la figura anterior se pueden ver los diferentes servicios. Esta organización es arbitraria, ya que cada servicio puede dividirse en otros más específicos, o podrían estar todos agrupados en el mismo servicio. La división se hace para tener un código más organizado y fácil de mantener. Dentro de cada servicio, se utilizan las instancias de los repositorios del modelo para realizar operaciones CRUD sobre las instancias, junto con otro código para dar soporte a dichas operaciones. Por ejemplo, a la hora de eliminar un recurso, no es suficiente con utilizar el repositorio para eliminar el objeto de la base de datos, también es necesario hacer la

comprobación previa de si el usuario está autorizado a eliminar ese recurso (utilizando un método propio del servicio de autorización) y después eliminar el recurso multimedia que se encuentra almacenado.

```
def delete(username, id):
    resource = resource_data.get_by_id(id)
    user = user_data.get_by(username=username)[0]
    if not auth_service.can_edit_resource(user, resource):
        raise common.LogicException('NOT_ALLOWED')
    if is_local_file(resource.url):
        os.remove(os.path.join(common.resources_path, resource.url.split('/')[-1]))

    resource_data.delete(resource)
```

Figura 5.8: método para borrar recurso

common

En este fichero se encuentran los métodos que se pueden considerar como utilidades genéricas para el resto de la aplicación. Un ejemplo es un método definido para operar con fechas en el formato que se ha decidido almacenar en base de datos:

```
def format_datetime(date=None, with_time=True) -> int:
    if date:
        current_date = date
    else:
        current_date = datetime.datetime.now()
    if with_time:
        final_date = int(current_date.strftime("%Y%m%d%H%M%S"))
    else:
        final_date = int(current_date.strftime("%Y%m%d"))
    return final_date
```

Figura 5.9: método format_datetime para gestión de fechas

routes

En routes se encuentran las rutas que definen el interfaz del API. Aquí es donde se utilizan las funciones principales del framework Flask, que con una simple anotación de Python te permite establecer que un determinado método es un endpoint del API. Para ver los componentes que tiene una ruta en esta aplicación, vamos a ver un ejemplo:

```
@app.route('/resource/reject/<id>', endpoint='reject_resource', methods=['PUT'])
@login_with_role([EXPERTO])
def reject_resource(id, **kwargs):
    try:
        params = request.json
        resource_service.reject(kwargs['username'], id, params['comments'])
        return common.make_response(200, True, 'OK')
    except Exception as err:
        print(traceback.format_exc())
        return common.make_response(500, False, 'INTERNAL_ERROR')
```

Figura 5.10: método de ruta API para rechazar un recurso

Esta es la ruta para rechazar un recurso. Lo primero que vemos es la anotación de Flask para indicar que este método es un *endpoint* del API. Utilizamos tres parámetros:

- El *path* de la ruta: es el *path* que un cliente ha de poner en la url para llegar a este *endpoint*. Este fragmento se añadiría a la URL donde esté alojado el API. Además vemos que el último fragmento de la ruta está entre corchetes angulares, lo que quiere decir que es un parámetro variable que indica el id del recurso que se quiere rechazar.

- Endpoint: es el nombre del *endpoint*. Es un *string* que actúa como etiqueta, necesario solo porque después de la anotación de Flask he utilizado otra creada por mí (login_with_role) para temas de seguridad.

- Methods: Es un array indicando qué métodos REST acepta esta ruta. En este caso solo acepta PUT, pero en la aplicación hay otros con GET, POST y DELETE.

Lo siguiente que vemos es una anotación para indicar que esta ruta está protegida por autenticación y autorización. Se detallará más adelante este mecanismo.

En la definición del método, se indica el parámetro que recibe, que ha de ser el mismo que el indicado en el *path*, además de un diccionario de parámetros extra que se inyectan en la anotación de autenticación.

Dentro del método, vemos un control *try/except* para el control de errores. En el cuerpo del *try*, está la parte principal de la gestión de la ruta: se extraen los parámetros del cuerpo de la petición, que vienen en formato JSON; se llama al método del servicio correspondiente, en este caso el de recursos; y se crea un objeto de tipo de respuesta y se devuelve al cliente. En caso de que algún error ocurra durante el transcurso de estas operaciones, se activa el cuerpo de la parte *except*, se escribe la traza de la excepción y se crea una respuesta de error que se devuelve al cliente.

5.2 Cliente web

El cliente web es el componente con el que interaccionan directamente los usuarios. Está desarrollando utilizando un framework muy extendido, Angular, cuyo uso se ha justificado en el punto 4.

5.2.1 Estructura y componentes básicos

Angular utiliza un patrón de diseño bien definido, por lo que para una aplicación web de interfaz sencilla como es esta, solo hay que seguir los estándares que definen. Para la organización de los archivos, se ha seguido mayormente la línea de la plantilla SB Admin descargada, que da una estructura básica a la aplicación. En la siguiente captura podemos ver el árbol de archivos y directorios:

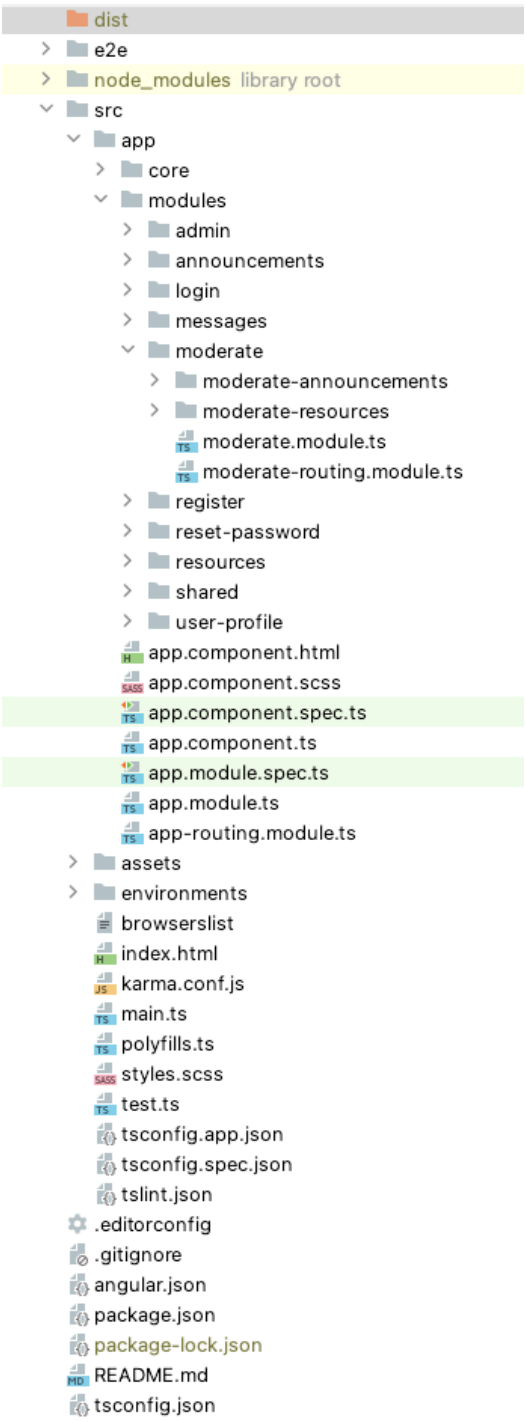


Figura 5.11: estrutura del proyecto cliente web

A continuación se detallan los componentes principales, que están dentro de la carpeta “sec”, ya que además de estos hay varios archivos y directorios generados automáticamente por la herramienta que arranca un proyecto en Angular que no son utilizados por el momento en este proyecto, por lo que no merece la pena describirlos.

App

Carpeta con los archivos fuente del código. Está subdividida en “core”, con los componentes que ya incluía la plantilla, como el componente de sidebar y el de header, y la carpeta “modules” con los módulos desarrollados por mí.

Un proyecto en Angular se estructura en componentes, que tienen que pertenecer a un módulo. Los módulos y los componentes son anidables, por lo que podemos tener un complejo sistema de varios módulos que contienen varios niveles de componentes.

La división por módulos se ha hecho en base a los dominios de la lógica de negocio, además de un módulo shared por componentes compartidos entre varios módulos.

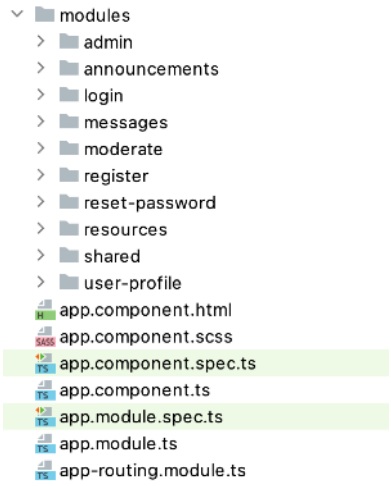


Figura 5.12: estructura de la carpeta de módulos

Como se puede ver en la figura anterior, todos los módulos desarrollados tienen pertenece al módulo raíz, en este caso App. Es el módulo App el encargado de “enrutar” el resto de módulos, esto es, asignar a cada módulo una url dentro del proyecto. Aquí tenemos un ejemplo:

```
{
  path: 'recursos',
  canActivate: [AuthGuard],
  loadChildren: () => import('@modules/resources/resources.module').then(m => m.ResourcesModule),
  data: {title: 'Recursos'}
},
```

Figura 5.13: ejemplo de ruta interna

Se asigna a la url /recursos el módulo de recursos (ResourcesModule). Además se le asigna un componente para controlar la autorización (que será descrito más adelante) y un diccionario para asignarle atributos a la ruta, en este caso solo un título.

Pasamos ahora a ver el contenido típico de un módulo. Como ejemplo vamos a mostrar el de “Moderar”.

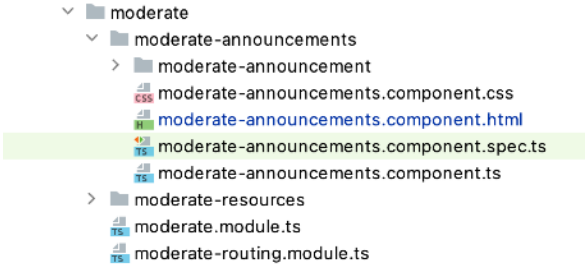


Figura 5.14: estructura del módulo *Moderate*

El módulo contiene dos archivos: uno para definir el propio módulo y qué componentes declara e importa; y otro para el enrutamiento dentro del propio módulo.

Además, vemos que el módulo contiene dos componentes: moderar anuncios y moderar recursos. Cada uno estará asignado a una subruta (recursos y anuncios) por lo que para acceder al de recursos habrá que ir a la ruta global /moderar/recursos.

Cada componente tiene 2 archivos principales:

.ts: es el código de la lógica del componente. Se declara y se definen los métodos que controlan el modelo (los datos) y la vista (presentación). Por ejemplo, el método que obtiene del API los datos de un recurso.

```
getResource() {
  const routeParams = this.route.snapshot.paramMap;
  this.idResource = routeParams.get('id');
  if (this.idResource) {
    this.apiService.getResource(this.idResource).subscribe( next: response => {
      this.resource = response.data;
      this.resource['dc_date'] = apiDateToStringNoTime(this.resource['dc_date']);
    }, error: error => {
      this.toastr.error( message: '', title: 'Error interno obteniendo información.');
    });
  }
}
```

Figura 5.15: método para obtener información de un recurso

Vemos cómo obtiene el id del recurso desde la URL, hace la petición al API utilizando un método de uno de los servicios definidos (que se detallan más adelante), extrae el resultado y lo asigna a variables del modelo.

El otro archivo principal de un componente es el html, que da la estructura a la vista y tiene directivas para controlar la misma. Un fragmento de ejemplo:

```
<div *ngIf="resource" style="..." class="row">
  <div class="col col-12">
    <form>
      <div class="row">
        <div class="col col-12 col-lg-4">
          <div class="form-group">
            <label for="title">Título</label>
            <input [(ngModel)]="resource.title" class="form-control" id="title" name="title"
                   type="text" disabled>
          </div>
          <div class="form-group">
            <label for="format">Formato</label>
            <input [(ngModel)]="resource.format" class="form-control" id="format" name="format"
                   type="text" disabled>
          </div>
          <div class="form-group">
            <label for="title">Autor</label>
            <input [(ngModel)]="resource.author" class="form-control" id="author" name="author"
                   type="text" disabled>
```

Figura 5.16: fragmento de formulario de un recurso

Vemos una estructura de un formulario para mostrar parte de los datos de un recurso. Al principio está directiva *ngIF que controla con un valor booleano si mostrar o no su contenido, en este caso solo se muestra si la petición al API ha acabado y la variable “Resource” ya tiene valor.

Otra directiva que vemos es la de “ngModel”, que enlace un campo de texto html con una variable del modelo, en las dos direcciones, por lo que si modificamos una de las dos partes, la otra se ve afectada.

Todos los módulos siguen una estructura similar a la descrita, excepto el módulo Shared. Este módulo contiene todos los elementos compartidos en la aplicación. Es en este módulo donde se encuentran los componentes que se encargan de reproducir y visualizar el contenido multimedia.

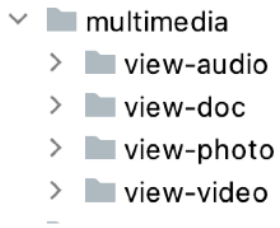


Figura 5.17: diferentes componentes para multimedia

Para los reproductores de audio y vídeo, he utilizado la librería externa VimeJS [28], que permite la reproducción de la mayoría de los formatos más conocidos y cierto grado de personalización del reproductor. He utilizado la configuración básica:

```
<div id="container">
  <vm-player controls>
    <vm-video crossorigin="anonymous">
      <source data-src="{{resource.url}}"/>
    </vm-video>
  </vm-player>
</div>

<style>
  :host {
    width: 100%;
    height: 100%;
    display: flex;
    align-items: center;
    justify-content: center;
  }

  #container {
    width: 100%;
    max-width: 960px;
  }
</style>
```

Figura 5.18: instancia de reproductor de vídeo

Simplemente se instancia un elemento de tipo vm-player, con un controlador dentro tipo vm-video o vm-audio, al cual se le pasa la url del recurso, que tenemos almacenada en el modelo.

Para el visualizador de documentos he utilizado la librería ngx-doc-viewer [29], que permite usar diferentes visualizadores externos. Concretamente, yo he elegido el de google, que permite herramientas y formatos similares al visualizador estándar de Google Drive. El visualizador se instancia muy fácil en su forma más básica:

```
<ngx-doc-viewer
  [url]="resource.url"
  viewer="google"
  style="width:100%;height:70vh;"
></ngx-doc-viewer>
```

Figura 5.19: instancia de visualizador de documentos

Por último, para visualizar fotos simplemente he utilizado un componente que contiene un elemento <img> de HTML.

Dentro del módulo shared también nos encontramos el servicio del API. Un servicio en Angular es un componente que contiene funcionalidad útil para usar en los componentes y que se inyecta en cierta parte de la aplicación para que esté disponible su uso, por lo que no es necesario instanciarlo. Esta técnica se conoce como Dependency Injection (DI) [30]. En este caso, la funcionalidad que ofrece este servicio es proveer de los métodos para llamar a los diferentes endpoints del API.

```
@Injectable({
  providedIn: 'root'
})

export class ApiService {

  private url: string;

  constructor(private httpClient: HttpClient) {
    this.url = environment.api.baseUrl;
  }
  // Auth
  public login(params) {
    return this.httpClient.post( url: `${this.url}/login/`, params);
  }
  public logout(): Observable<any> {
    return this.httpClient.get( url: `${this.url}/logout/`);
  }
  public getCategories(): Observable<any> {
    return this.httpClient.get( url: `${this.url}/categories/`);
  }
  public checkToken(): Observable<any> {
    return this.httpClient.get( url: `${this.url}/user/check_token`);
  }
```

Figura 5.20: servicio para comunicación con API

Cada uno de estos métodos devuelve un objeto de tipo Observable, que es algo similar a las promesas, un patrón común en javascript para la comunicación asíncrona. Los diferentes componentes que utilizan estos métodos tienen que "suscribirse" a estos objetos Observable para esperar la respuesta del API y ejecutar alguna acción con los resultados, pero sin bloquear el resto de la aplicación. Vemos por ejemplo el caso de enviar un mensaje:

```
onClickSend() {
  this.apiService.sendMessage(this.formMessage).subscribe( next: response => {
    if (response.success) {
      this.toastr.success( message: '', title: 'Mensaje enviado con éxito');
      this.router.navigate( commands: ['/mensajes']);
    }
  }, error: error => {
    this.toastr.error( message: '', title: 'Ha ocurrido un error interno');
  });
}
```

Figura 5.21: método para enviar mensaje

Se llama al método sendMessage del servicio, y se suscribe al resultado para esperar a que llegue la respuesta por parte del servidor. Una vez que llega, comprueba si ha habido éxito con la operación y, en caso afirmativo, muestra un mensaje de éxito y redirecciona al usuario al buzón de mensajes. En caso de que la respuesta venga con error, se informa al usuario con un mensaje de error y no se hace nada más.

Otro tipo de componentes a destacar son los modales, que son las ventanas que aparecen emergentes para realizar algún tipo de acción, como una confirmación. Se han realizado usando como base el componente ngbModal de la librería ngBoostrap

5.2.2 Gestión del mapa

Uno de los puntos más importantes de este trabajo consiste en la geolocalización de los recursos en un mapa. El componente que gestiona esta funcionalidad es uno de los más complejos de la aplicación, por lo que se detalla en aparte en esta sección.

La base del componente se ha realizado utilizando el framework OpenLayers [31], una herramienta compleja que tiene una enorme cantidad de posibilidades de personalización. En este trabajo se han incluido las opciones básicas.

Para empezar, hay que definir una capa base que contenga el mapa. Hay muchos tipos de capas de este tipo, pero he elegido la de OpenStreet Maps ya que es completamente open source y su uso está bien extendido, ya que su exactitud con la realidad es suficiente para nuestro problema.

Una vez instanciado el mapa, llega la parte más difícil que es situar puntos en él representando cada uno de los recursos, permitiendo la interacción al hacer click. Para ello, y de forma resumida, hay que crear varias capas vectoriales más, transformar las coordenadas de cada uno de los recursos en objetos que entiende el mapa y utilizar el API que incluye la librería para arrancarlo todo.

El método principal que muestra el mapa de la vista de explorar recursos es este:

```
generateMap() {
  const container = document.getElementById( elementId: 'popup');
  const content = document.getElementById( elementId: 'popup-content');
  const closer = document.getElementById( elementId: 'popup-closer');


  const overlay = new Overlay( options: {
    element: container,
    autoPan: true,
    autoPanAnimation: {
      duration: 250
    }
  });
  this.iconsPoints = this.resources.map(item => {
    const c = item['coordinates'].split(',');
    const cn = [+(c[1]).trim(), +c[0]];
    return new Feature( opt_geometryOrProperties: {
      geometry: new Point(olProj.fromLonLat(cn)),
      name: item['title'],
      type: item['multimedia_type'],
      id_resource: item['id']
    });
  });
  const vectorSource = new VectorSource( opt_options: {
    features: [...this.iconsPoints],
  });

  const vectorLayer = new VectorLayer( opt_options: {
    source: vectorSource,
  });
```

Figura 5.22: método gestión de mapa (1)

```
this.map = new Map( options: {
  target: 'hotel_map',
  layers: [
    new TileLayer( opt_options: {
      source: new OSM()
    }), vectorLayer
  ],
  overlays: [overlay],
  view: new View( opt_options: {
    center: olProj.fromLonLat( coordinate: [-9.8361, 28.9291]),
    zoom: 5
  })
});
this.map.on('singleclick',  (evt: any) => {
  const feature = this.map.forEachFeatureAtPixel(evt.pixel,  callback: function (feature2) {
    return feature2;
  });
  if (feature) {
    const coordinate = evt.coordinate;
    const hdms = feature.get('name');

    content.innerHTML = '<h3>' + feature.get('name') + '</h3>';
    content.innerHTML += '<p><b>Tipo: </b>' +  feature.get('type') + '</p>';
    content.innerHTML += '<a href="/recursos/ver/' + feature.get('id_resource') +  '">Ver recurso</a>';
    overlay.setPosition(coordinate);
  }
});


closer.onclick = function () {
  overlay.setPosition(undefined);
  closer.blur();
  return false;
};
}
```

Figura 5.23: método gestión de mapa (2)

5.3 Autenticación y autorización

La parte autenticación (permitir accesos al sistema) y autorización (controlar qué operaciones puede realizar cada tipo de usuario) merece una sección aparte, ya que involucra componentes del backend (API) y del frontend (cliente web).

En primer lugar, los usuarios están almacenados en base de datos. Cuando un usuario hace una solicitud de registro, se crea una entrada en base de datos almacenando el hash (SHA-256) de la contraseña suministrada. Antes de que el usuario sea válido para el sistema, el proceso tiene que pasar por una serie de pasos:

- Una vez solicitado el registro, se envía un email de confirmación al usuario para comprobar que la dirección de correo suministrada es válida y suya.
- Si el usuario confirma la dirección pulsando en el enlace, la solicitud pasa a pendiente de confirmación por un moderador.
- Si un moderador acepta la solicitud, se envía un email al usuario indicando que ya está confirmado su registro. A partir de aquí el usuario ya puede entrar en el sistema.

El control de inicio de sesión en el sistema y el mantenimiento de la misma se hace usando el mecanismo de JWT comentado en la sección de arquitectura. El proceso es el siguiente:

1. El usuario introduce su usuario y contraseña en el frontal web y se envía la información al API.

2. El API comprueba si el hash de la contraseña es igual al hash almacenado en base de datos para ese usuario.

3. En caso afirmativo, se genera un JSON Web Token con la información que necesita: fecha de caducidad del token, usuario y rol. Esta información se firma con la clave privada de la aplicación.

4. El cliente web recibe el token y lo almacena. Además, almacena el rol que tiene el usuario.

5. A partir de ahora, todas las peticiones HTTP que envíe el cliente web hacia la API incluirán en la cabecera el token. Cuando una petición llega, el API comprueba que el token es válido (no ha sido alterado) utilizando la firma realizada cuando se creó. Si el token es válido, se permite la operación.

6. Si el usuario hace logout o la fecha de caducidad del token llega, el token deja de ser válido y se pedirá al usuario que vuelva a hacer login.

Además de este proceso, existe también de forma intercalada el proceso de autorización. Este consiste en permitir o denegar el uso de determinadas acciones y de visualizar determinadas vistas a un usuario autenticado en base al rol que tiene. Esta comprobación se hace tanto en el cliente web como en la API.

A continuación se describen los principales componentes de código que se encargan de este proceso.

```python
def login_with_role(roles=None):
    def wrap(fn):
        def wrapped_f(*args, **kwargs):
            role_list = roles
            token = request.headers.get("token")
            try:
                private_key = open(common.private_key).read()
                JWT_SECRET = private_key
                JWT_ALGORITHM = "HS256"
                payload = jwt.decode(token, JWT_SECRET, algorithms=JWT_ALGORITHM)
                if role_list and not check_user_is_allowed(payload['user_name'], role_list):
                        result = common.make_response(401, False, 'NOT_ALLOWED')
                        return result

                try:
                    kwargs['username'] = payload['user_name']
                    kwargs['token'] = token
                    return fn(*args, **kwargs)
                except Exception as err:
                    result = common.make_response(500, False, 'INTERNAL_ERROR')
                    return result
            except jwt.ExpiredSignatureError:
                payload = {
                    'status': 'WRONG_TOKEN',
                    'msg': 'Signature expired. Please log in again.'
                }
                result = common.make_response(401, False, payload, '')
                return result
            except jwt.InvalidTokenError:
                payload = {
                    'status': 'WRONG_TOKEN',
                    'msg': 'Signature expired. Please log in again.'
                }
                result = common.make_response(401, False, 'NO_AUTH', payload)
                return result
            except Exception as err:
```

Figura 5.24: anotación para login

La figura 5.24 muestra un fragmento de la implementación del decorador de Python que se añade a todas las rutas del API. Dicho decorador recibe la petición HTTP, extrae el token y hace todas las comprobaciones necesarias para decidir si dejar continuar la acción o no: comprueba que el token es válido, comprueba que el usuario existe y comprueba que el rol que tiene permite realizar esta acción. En caso afirmativo, llama a la ruta que se activó, y en caso contrario devuelve una respuesta negativa que provocará el fin de de la sesión en el cliente web.

Lo métodos encargados de comprobar si un usuario está autorizado a realizar una determinada operación están implementados en el servicio “auth_service”:

```python
def can_edit_announcement(user, announcement):
    return user.id_role == 3 or (user.id == announcement.id_user_proposal)


def can_moderate_announcement(user, announcement):
    return user.id_role == 3 and announcement.id_user_publish is None


def can_edit_resource(user, resource):
    return user.id_role == 3 or (user.id == resource.id_user_proposal)


def can_moderate_resource(user, resource):
    return user.id_role == 3 and resource.id_user_publish is None
```

Figura 5.25: servicio de autorización

En la parte del cliente web, hay un componente llamado TokenInterceptor que se encarga de preprocesar cada petición que va a enviar el cliente hacia el API y le añade como cabecera el token. Además, también preprocesa cada respuesta y comprueba el código de estado. Si es un código 401 (no autorizado), cierra la sesión y redirecciona al usuario hacia la pantalla de login.

```typescript
export class TokenInterceptor implements HttpInterceptor {
    constructor(private router: Router) {}

    intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
        const userToken = localStorage.getItem( key: 'token');
        let request = req;
        if (userToken) {
            request = req.clone( update: {
                headers: req.headers.set('token', userToken),
            });
        }

        return next.handle(request).pipe(tap( next: () => {},
             error: (err: any) => {
          console.log(err);
                if (err instanceof HttpErrorResponse) {
                    if (err.status === 401) {
                        onLoggedout();
                        this.router.navigate( commands: ['/login']);
                    } else {
                      return;
                    }
                }
            }));
    }
}
```

Figura 5.26: implementación de interceptor de tokens

El otro componente importante del cliente web relativo a la autenticación es el AuthGuard. Este componente se añade a las rutas que necesitan autenticación para ser activadas, que en este caso son todas excepción de las de registro, login y reseteo de contraseña. Ya hemos visto en la sección anterior como se añade a una ruta. El componente tiene esta implementación:

```
@Injectable()
export class AuthGuard implements CanActivate {
    constructor(private router: Router, private apiService: ApiService) {}

  canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): Observable<boolean>  {
    if (localStorage.getItem( key: 'isLoggedin') === 'true' && localStorage.getItem( key: 'token')) {
      return this.apiService.checkToken().pipe(
        map( project: e => {
          return true;
        }),
        catchError( selector: (err) => {
          onLoggedout();
          this.router.navigate( commands: ['/login']);
          return of( args: false);
        })
      );
    } else {
      onLoggedout();
      this.router.navigate( commands: ['/login']);
      return of( args: false);
    }

  }

}
```

Figura 5.27: implementación de RoleGuard

Existe otro componente similar, llamado RoleGuard, que hace una tarea similar para comprobar si una ruta puede ser activada para un determinado usuario.

5.4 Despliegue en servidor

Esta aplicación web ha sido desplegada en un Virtual Private Server (VPS) que he contratado en el proveedor Ovh. A continuación se describen algunos de los principales pasos que se han dado para conseguirlo.

1. Instalación de todos los paquetes de software requeridos. He tenido que instalar: el servidor web *Apache;* el gestor de base de datos *PostgreSQL;* la herramienta de control de procesos *supervisor; nodeJS* para generar la *build* de *Angular*; *Python3.8*.

2. Clonar los repositorios de código que tengo alojados en *GitLab*: soundscapes-client y soundscapes-api.

3. Desplegar el API. Para ello, se crea un entorno virtual con la herramienta *virtualenv* y se instalan todas las dependencias con la herramienta *pip*. Después se añade un fichero de configuración para este proyecto en *supervisor* y se arranca. Esta herramienta se encargará de tener activo el proceso, monitorizarlo, generar logs y arrancarlo cada vez que se reinicie la máquina.

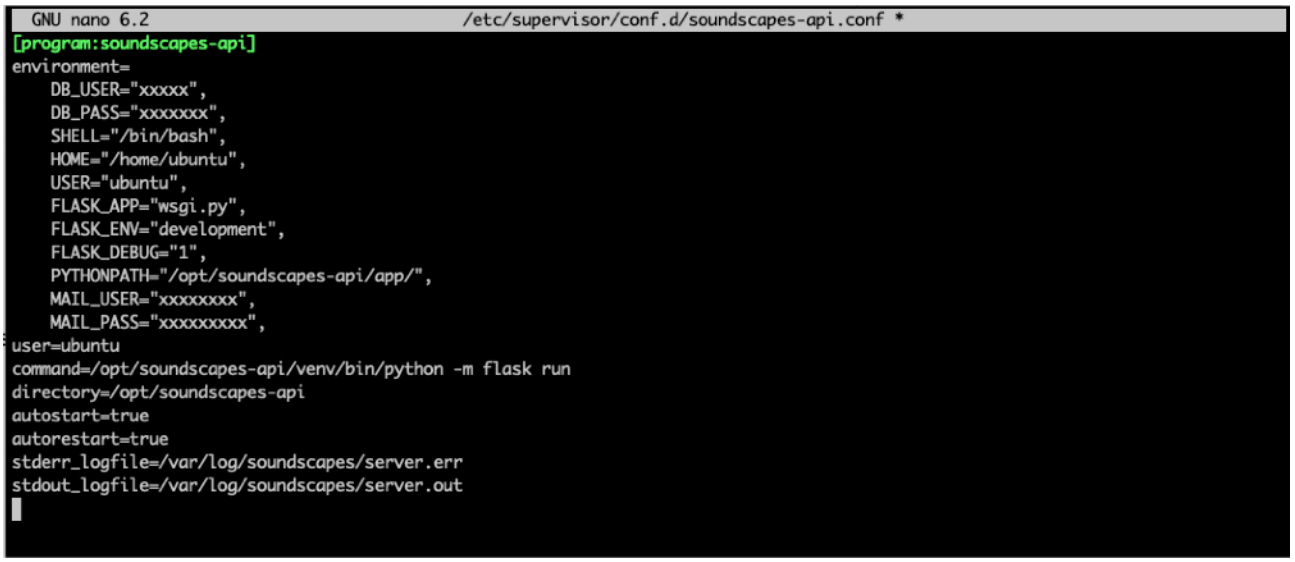


Figura 5.28: archivo de configuración de *supervisor* para la API.

4. Desplegar el cliente web. Para ello se genera una build del proyecto de angular. Una build es una especie de compilado (sin llegar a serlo porque el código sigue siendo JavaScript). Se reduce el tamaño del código y se reorganiza para que sea más eficiente. El resultado es un directorio que hay que poner en el servidor web.

5. Por último, se añade la configuración de Apache. Para mejorar la seguridad, solo se expone el puerto 80 (en el futuro solo será el 443, cuando se tenga el certificado SSL para el dominio), por lo que hay que hacer un proxy reverso de la ruta /api al puerto 5000, que es donde está escuchando el API. Un proxy reverso es una función que permite Apache para redireccionar las peticiones que llegan a una determinada ruta hacia otra parte. En nuestro caso le hemos dicho que todas las peticiones HTTP que le lleguen a la ruta /api, las mande a este mismo servidor, pero a la aplicación que haya escuchando en el puerto 5000 (el API). Además, añadimos en esta configuración de Apache las líneas necesarias para servir el contenido estático de los recursos multimedia.

```
ProxyPass /api/ http://localhost:5000/
ProxyPassReverse /api/ http://localhost:5000/
```

Figura 5.29: configuración apache para proxy reverso

```
Alias /resources "/var/www/html/resources"
<Directory "/var/www/html/resources">
        Options Indexes FollowSymLinks MultiViews
        AllowOverride all
        Order allow,deny
        allow from all
</Directory>
```

Figura 5.30: configuración apache para archivos estáticos

6. Depuración y pruebas

En esta sección se describen las pruebas realizadas a la aplicación, tanto durante el desarrollo como una vez finalizado el mismo.

6.1 Depuración

La fase de depuración consiste en las pruebas que se van haciendo durante el desarrollo para comprobar que las funciones realizadas tienen un comportamiento correcto.

Una vez que se termina de realizar una funcionalidad, se procede a probarla de forma manual, ya sea ejecutando una llamada HTTP a la ruta que activa esa operación, o desde el propio frontal web. Cuando se detecta que una funcionalidad no da el resultado esperado, se inicia un proceso de depuración. Para ello, la principal herramienta es el depurador del IDE, que permite ejecutar un fragmento de código paso a paso, viendo en cada momento el valor de todas las variables y permitiendo la modificación de las mismas para realizar pruebas y encontrar el problema. Otra herramienta fundamental para detectar errores, sobre todo en el cliente web, es la consola del navegador, donde se pueden ver el output de los errores que produce el código de javascript y realizar operaciones similares que con el depurador del IDE.

6.1.1 Ejemplos de depuración de errores

A continuación se detallan dos ejemplos de como ha sido el proceso de depuración de errores: uno para el API (backend) y otro para el cliente web (frontend)

Ejemplo 1: fallo en publicar recurso

Una vez terminados todos los métodos de publicación de recurso, hacemos las pruebas y vemos que todo funciona correctamente excepto que cuando un usuario de tipo Artista o Experto crea un recurso, no pasa directamente a estado publicado, si no que se queda en pendiente de moderar. Esto no debe ser así, ya que este tipo de usuarios puede publicar directamente sus recursos. Al comprobar los logs, no aparece ninguna excepción ni ningún mensaje de error, por lo que el fallo tiene que ser de la lógica desarrollado. Por lo tanto, iniciamos el proceso de depuración.

Lo primero que hay que mirar es si en la propia base de datos el recurso aparece como publicado o como no publicado. Al entrar con la herramienta gráfica y realizar la consulta pertinente, vemos que la columna “publication_date” tiene como valor 0, lo que indica que el recurso no está publicado. Con esta comprobación descartamos que el problema sea de visualización en el cliente web y sabemos que hay un fallo en el código del API que crea el recurso.

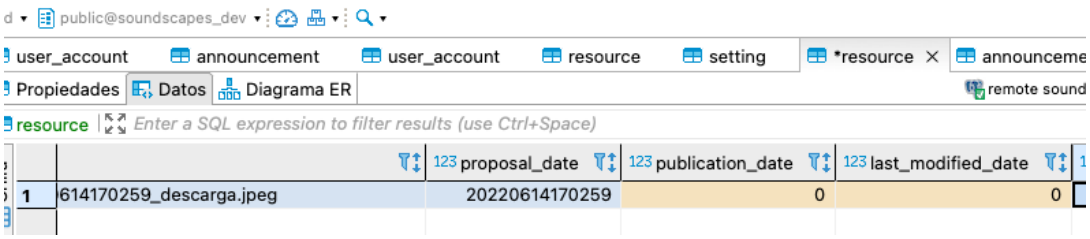


Figura 6.1: registro en BD del recurso buscado.

El siguiente paso es arrancar en modo *debug* el API, poner un punto de ruptura al inicio de la llamada a la ruta de crear un recurso e ir línea a línea para detectar el problema. Al llegar a la línea 119, el curso normal de la ejecución sería entrar dentro del primer “if” para asignarle como fecha de publicación la fecha actual.

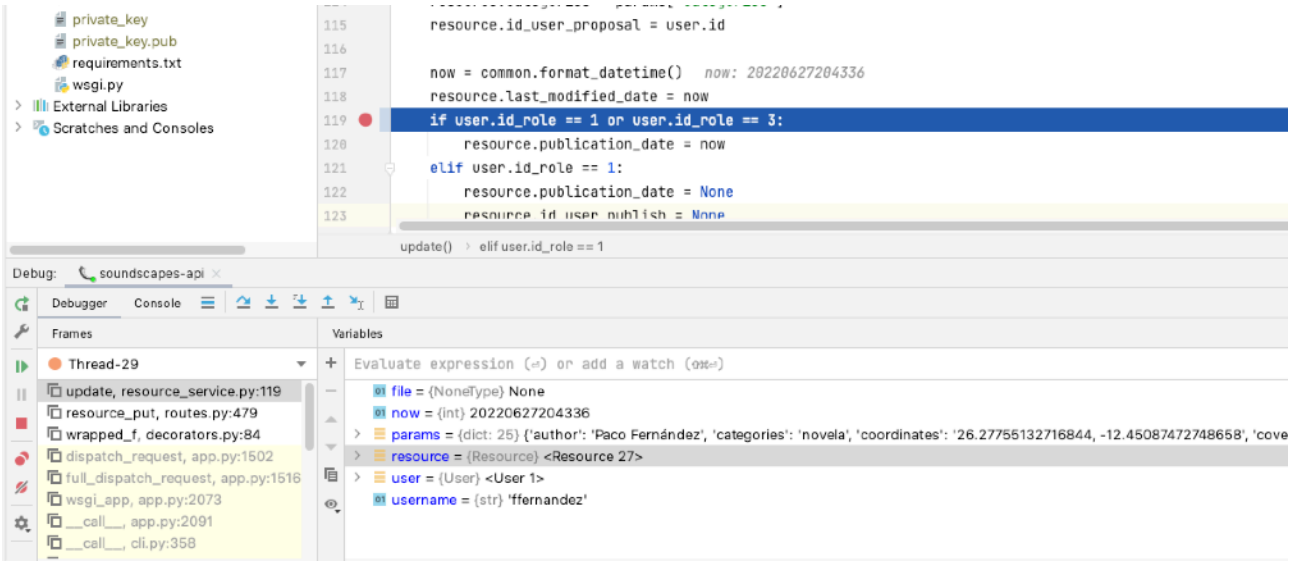


Figura 6.2: IDE en modo depuración en publicar recurso.

Pero al avanzar de línea, no entra en el if. Vemos que los roles por los que comprueba son 1 y 3, y el rol de Artista, que es el tipo de usuario que inició la acción, corresponde con el número 2. Por lo tanto, el bug se soluciona cambiando el 1 de la línea 119 por un 2. Una vez hecho este cambio y reiniciada la app, se realiza la misma operación y ahora sí vemos que funciona todo correctamente.

Ejemplo 2: correo electrónico no aparece en perfil de usuario

Al entrar en la pantalla de edición de perfil de usuario vemos que el campo email aparece vacío, lo cual es imposible porque es un campo básico y obligatorio para el registro:

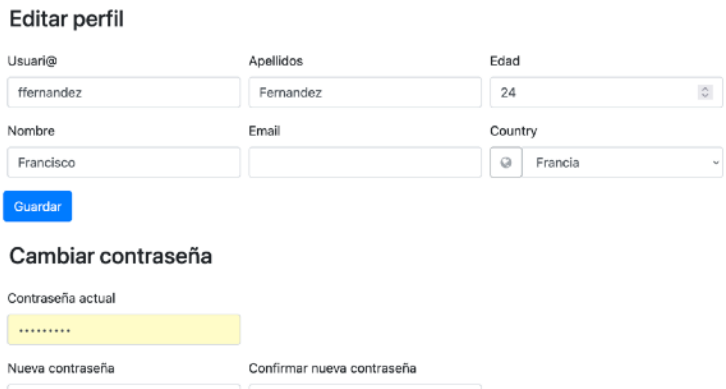


Figura 6.3: formulario de Editar perfil.

Al igual que en el caso anterior, lo primero que se comprueba es que el valor esté correcto en la base de datos.

1	ffernandez	paco@mail.com	a184712a824240be01cac45109027e926f730	Francisco	Fernandez

Figura 6.4: registro en BD del perfil buscado.

Efectivamente, el usuario ffernandez tiene un correcto valor del email en base de datos.

De nuevo realizamos el siguiente paso: comprobar que el API funciona correctamente. Para ello usamos el depurador del IDE para ver, al final de la llamada al método que recupera los datos de un usuario, si el objeto que manda en la respuesta hacia el cliente web contiene los datos correctos.

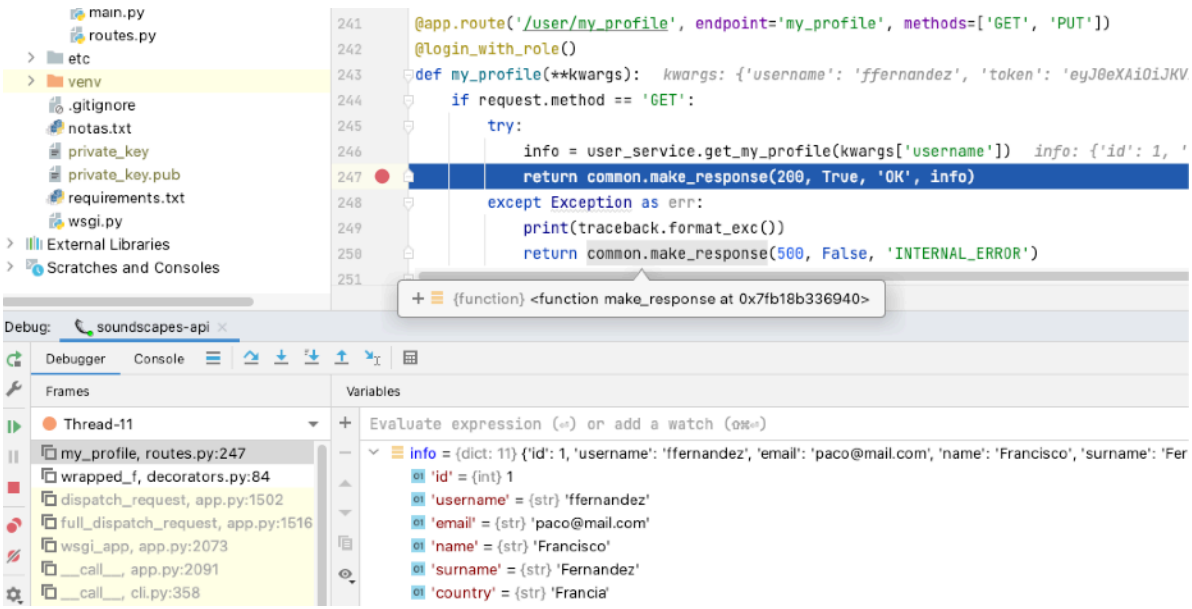


Figura 6.5: IDE en modo depuración en la ruta de recuperar perfil.

Vemos que cuando la ejecución llega a la línea 247, que es la última de la operación, el objeto “info” que contiene los datos que llegarán al cliente web contiene el atributo “email” con el valor “paco@mail.com". Por lo tanto llegamos a la conclusión de que el fallo tiene que estar en el cliente web.

Lo primero que miramos para depurar el cliente web es la consola del navegador, pero no aparece ningún tipo de mensaje de error. Lo siguiente a mirar es el propio código del formulario que muestra la información del usuario, concretamente el campo de email.

```
<div class="form-group col-md-3">
  <label for="surname">Email</label>
  <input type="email" [(ngModel)]="myProfileForm.mail" class="form-control" name="email" id="email">
</div>
<div class="form-group col-md-3">
```

Figura 6.6: código HTML del campo email del formulario de perfil.

En este paso vemos claramente el error: el formulario intenta mostrar un valor del modelo llamado “mail”, mientras que del API llega un atributo llamado “email”. Se cambia este nombre en el fichero HTML, se recarga la vista y ya aparece correctamente el correo electrónico.

6.2 Catálogo de pruebas

Una vez finalizado el desarrollo de la primera versión, se realizan una serie de pruebas end-to-end, que consisten en probar la aplicación desde el punto de vista del usuario final, esto es, realizando el flujo completo de operaciones que un usuario real realizará cuando use la aplicación, y comprobando que el resultado es el esperado. Para realizarlo de manera ordenada, se realiza un catálogo de pruebas con las operaciones que se desean probar:

ID	P-01
Descripción	Explorar recursos
Pasos	• Acceder a la vista de Explorar. • Utilizar el mapa para ver los diferentes recursos. • Introducir valores en el cuadro de búsqueda y pulsar en Buscar. • Ver el listado de resultados y comprobar que coinciden con el criterio de búsqueda. • Comprobar que los recursos del listado corresponden con los recursos del mapa.
Posibles errores	• El componente mapa no funciona. • La búsqueda produce un mensaje de error. • Aparecen recursos que no coinciden con el criterio de búsqueda. • En el mapa no aparecen los recursos filtrados. (aparecen más o menos de los requeridos).
Resultado esperado	El mapa permite navegar correctamente por los recursos, la búsqueda muestra los resultados correctos y estos aparecen en el mapa.

ID	P-02
Descripción	Visualizar un recurso
Pasos	• Acceder a la vista de Ver un recurso desde el mapa, el listado de resultados o el listado de recursos propios. • Comprobar que aparecen todos los campos del recurso en el formulario. • Comprobar que el recurso multimedia se visualiza o se reproduce correctamente. • Si está permitido para ese recurso, comprobar que aparece el botón de descarga y que lo descarga correctamente.
Posibles errores	• No aparecen los valores correctos en los atributos del recurso. • El recurso múltimedia no se visualiza o se reproduce correctamente. • Aparecen recursos que no coinciden con el criterio de búsqueda. • En el mapa no aparecen los recursos filtrados. (aparecen más o menos de los requeridos).
Resultado esperado	El mapa permite navegar correctamente por los recursos, la búsqueda muestra los resultados correctos y estos aparecen en el mapa.

ID	P-03
Descripción	Editar un recurso
Pasos	• Acceder a la vista de Editar un recurso desde el listado de recursos propios o a los que se tiene acceso (al ser Moderador). • Modificar uno o varios atributos del recurso, o el propio archivo multimedia, y pulsar en guardar. • Acceder a la vista de Ver el recurso desde el mapa, el listado de resultados o el listado de recursos propios. • Comprobar que aparecen los nuevos valores en los atributos modificados.
Posibles errores	• No aparecen los valores correctos en los atributos del recurso. • Aparece un mensaje de error a la hora de pulsar el botón de guardar. • Al entrar en visualizar el recurso, no aparecen reflejados los cambios realizados.
Resultado esperado	Aparece un mensaje de confirmación de que todo ha ido correctamente y los cambios realizados se ven reflejados en el recurso.

ID	P-04
Descripción	Eliminar un recurso
Pasos	• Acceder a la vista de listado de recursos propios o a los que se tiene acceso (al ser Moderador). • Pulsar en Eliminar sobre un recurso. • Confirmar el popup que aparece. • Comprobar que el recurso no aparece en el listado de recursos. • Comprobar que el recurso multimedia ha sido eliminando del servidor.
Posibles errores	• El botón de eliminar no produce ninguna acción. • Al pulsar confirmar, aparte un mensaje de error interno. • El sistema indica que el recurso ha sido eliminado, pero sigue apareciendo en el listado de recursos. • El recurso se elimina de base de datos, pero el archivo multimedia sigue apareciendo en el repositorio del servidor.
Resultado esperado	Aparece un mensaje de confirmación de que todo ha ido correctamente y tanto el registro en base de datos del recurso como el archivo multimedia quedan eliminados.

ID	P-05
Descripción	Visualizar tablón de anuncios.
Pasos	• Acceder a la vista de Tablón de anuncios. • Ver los anuncios que aparecen en el tablón los anuncios de la plataforma. • Introducir una cadena de texto en el cuadro de búsqueda y ver que se filtran correctamente.
Posibles errores	• No aparece ningún anuncio en el tablón, o aparecen menos de los que deberían. • El cuadro de búsqueda no filtra correctamente.
Resultado esperado	Aparece un listado con todos los anuncios, y al introducir una palabra en el cuadro de búsqueda, aparecen solo los anuncios que contienen esa palabra en el título o en el cuerpo.

ID	P-06
Descripción	Visualizar un anuncio
Pasos	• Acceder a la vista de Ver un anuncio desde el listado de anuncios propios. • Comprobar que aparecen todos los campos del anuncio en el formulario.
Posibles errores	• No aparecen los valores correctos en los atributos del anuncio.
Resultado esperado	Al pulsar sobre ver anuncio, se abre una nueva vista con la información del anuncio

ID	P-07
Descripción	Editar un anuncio
Pasos	• Acceder a la vista de Editar un anuncio desde el listado de anuncios propios o a los que se tiene acceso (al ser Moderador). • Modificar uno o varios atributos del anuncio y pulsar en guardar. • Acceder a la vista de Ver el anuncio desde el listado de anuncios. • Comprobar que aparecen los nuevos valores en los atributos modificados.
Posibles errores	• No aparecen los valores correctos en los atributos del anuncio. • Aparece un mensaje de error a la hora de pulsar el botón de guardar. • Al entrar en visualizar el anuncio, no aparecen reflejados los cambios realizados.
Resultado esperado	Aparece un mensaje de confirmación de que todo ha ido correctamente y los cambios realizados se ven reflejados en el anuncio.

ID	P-08
Descripción	Eliminar un anuncio
Pasos	• Acceder a la vista de listado de anuncios propios o a los que se tiene acceso (al ser Moderador). • Pulsar en Eliminar sobre un anuncio. • Confirmar el popup que aparece. • Comprobar que el anuncio no aparece en el listado de anuncio.
Posibles errores	• El botón de eliminar no produce ninguna acción. • Al pulsar confirmar, aparte un mensaje de error interno. • El sistema indica que el anuncio ha sido eliminado, pero sigue apareciendo en el listado de anuncios.
Resultado esperado	Aparece un mensaje de confirmación de que todo ha ido correctamente y el anuncio queda eliminado.

ID	P-09
Descripción	Consultar buzón de mensajes
Pasos	• Acceder a la vista de Mensajes • Comprobar que aparecen todos los mensajes recibidos en la pestaña de Recibidos. • Comprobar que aparecen todos los mensajes enviados en la pestaña de Enviados. • Pulsar en cualquier mensaje y ver que aparece su contenido en el área de la derecha.
Posibles errores	• No aparecen todos los mensajes en su correspondiente pestaña. • Al pulsar sobre un mensaje leído, sigue apareciendo con el color de los mensajes no leídos. • Al pulsar sobre un mensaje, no aparece su contenido.
Resultado esperado	Aparece todos los mensajes en su pestaña y al pulsar sobre uno de ellos vemos el contenido.

ID	P-10
Descripción	Enviar un mensaje
Pasos	• Acceder a la vista de mensajes. • Pulsar en el botón de nuevo mensaje. • Rellenar el título, el contenido y elegir un usuario destinatario. • Pulsa en el botón de enviar.
Posibles errores	• Al pulsar en enviar aparece un mensaje de error interno. • Al pulsar en enviar, el mensaje no aparece en el buzón de entrada del destinatario. • No aparecen todos los usuarios en el desplegable de destinatarios.
Resultado esperado	Al pulsar sobre enviar aparece un mensaje de confirmación, y el mensaje aparece como no leído en el buzón de entrada del destinatario.

ID	P-11
Descripción	Eliminar un mensaje
Pasos	• Acceder a la vista de mensajes. • Pulsa en el botón de eliminar. • Confirma en el popup que aparece. • Comprobar que el mensaje deja de aparecen en el listado.
Posibles errores	• El botón de eliminar no produce ninguna acción. • Al pulsar confirmar, aparte un mensaje de error interno. • El sistema indica que el mensaje ha sido eliminado, pero sigue apareciendo en el listado de mensajes.
Resultado esperado	Aparece un mensaje de confirmación de que todo ha ido correctamente y el mensaje queda eliminado.

ID	P-12
Descripción	Moderar un recurso (aceptar)
Pasos	• Acceder a la vista de Moderar un recurso. • Seleccionar un recurso y pulsar en moderar. • Comprobar que los campos aparecen correctamente y que el archivo multimedia se visualiza o reproduce. • Pulsar en Publicar
Posibles errores	• El botón de moderar no produce ninguna acción. • Los atributos no aparecen bien o el archivo multimedia no se visualiza o reproduce correctamente. • Al pulsar en publicar aparece un mensaje de error interno. • Al pulsar en publicar, el recurso no aparece en el listado de publicados.
Resultado esperado	Aparece un mensaje de confirmación de que todo ha ido correctamente y el recurso queda publicado.

ID	P-13
Descripción	Moderar un anuncio (aceptar)
Pasos	• Acceder a la vista de Moderar un anuncio. • Seleccionar un anuncio y pulsar en moderar. • Comprobar que los campos aparecen correctamente. • Pulsar en Publicar
Posibles errores	• El botón de moderar no produce ninguna acción. • Los atributos no aparecen correctamente. • Al pulsar en publicar aparece un mensaje de error interno. • Al pulsar en publicar, el anuncio no aparece en el listado de publicados.
Resultado esperado	Aparece un mensaje de confirmación de que todo ha ido correctamente y el anuncio queda publicado.

ID	P-14
Descripción	Moderar un recurso (rechazar)
Pasos	• Acceder a la vista de Moderar un recurso. • Seleccionar un recurso y pulsar en moderar. • Comprobar que los campos aparecen correctamente y que el archivo multimedia se visualiza o reproduce. • Pulsar en Rechazar. • Rellenar los comentarios con el motivo del rechazo y pulsar en confirmar.
Posibles errores	• El botón de moderar no produce ninguna acción. • Los atributos no aparecen bien o el archivo multimedia no se visualiza o reproduce correctamente. • Al pulsar en rechazar aparece un mensaje de error interno. • Al pulsar en rechazar, el recurso no aparece en el listado de rechazados del usuario. • Los comentarios con el motivo del rechazo no aparecen al editar el recurso.
Resultado esperado	Aparece un mensaje de confirmación de que todo ha ido correctamente, el recurso queda rechazado y aparecen los comentarios con el motivo del rechazo.

ID	P-15
Descripción	Moderar un anuncio (rechazar)
Pasos	• Acceder a la vista de Moderar un anuncio. • Seleccionar un anuncio y pulsar en moderar. • Comprobar que los campos aparecen correctamente. • Pulsar en Rechazar. • Rellenar los comentarios con el motivo del rechazo y pulsar en confirmar.
Posibles errores	• El botón de moderar no produce ninguna acción. • Los atributos no aparecen correctamente. • Al pulsar en rechazar aparece un mensaje de error interno. • Al pulsar en rechazar, el anuncio no aparece en el listado de rechazados del usuario. • Los comentarios con el motivo del rechazo no aparecen al editar el anuncio.
Resultado esperado	Aparece un mensaje de confirmación de que todo ha ido correctamente, el anuncio queda rechazado y aparecen los comentarios con el motivo del rechazo.

ID	P-16
Descripción	Editar usuario
Pasos	• Acceder a la vista de Administración de usuarios. • Pulsar en el botón de editar sobre un usuario. • En la vista que aparece, modificar uno o varios atributos del usuario. • Pulsar en guardar.
Posibles errores	• El botón de editar usuario no realiza ninguna acción. • Los atributos del usuario aparecen mal. • Al pulsar en guardar, aparece un mensaje de error interno. • Una vez guardado, las modificaciones no aparecen reflejadas en el usuario.
Resultado esperado	Aparece un mensaje de confirmación de que todo ha ido correctamente y los datos de usuario quedan modificados.

ID	P-16
Descripción	Eliminar usuario
Pasos	• Acceder a la vista de Administración de usuarios. • Pulsar en el botón de eliminar sobre un usuario. • Confirmar el popup que aparece, eligiendo si donar a la plataforma los recursos del usuario.
Posibles errores	• El botón de eliminar usuario no realiza ninguna acción. • Al pulsar en confirmar, aparece un mensaje de error interno. • Una vez confirmado, el usuario sigue apareciendo en el listado. • Si se ha elegido donar los recursos: los recursos del usuario han quedado eliminados. • Si se ha elegido no donar los recursos: los recursos del usuario siguen apareciendo en la plataforma.
Resultado esperado	Aparece un mensaje de confirmación de que todo ha ido correctamente, el usuario queda eliminado y los recursos permanecen o no en la plataforma en función de la opción seleccionada.

ID	P-17
Descripción	Aceptar registro
Pasos	• Acceder a la vista de Administración de registros. • Pulsar en el botón de aceptar sobre una solicitud de registro. • Confirmar el popup que aparece.
Posibles errores	• El botón de aceptar no realiza ninguna acción. • Al pulsar en aceptar, aparece un mensaje de error interno. • Una vez aceptado, no se crea un usuario nuevo. • No llega un email de confirmación al usuario.
Resultado esperado	Aparece un mensaje de confirmación de que todo ha ido correctamente, el usuario queda registrado en el sistema y recibe un email en su dirección de correo.

ID	P-18
Descripción	Rechazar registro
Pasos	• Acceder a la vista de Administración de registros. • Pulsar en el botón de rechazar sobre una solicitud de registro. • Confirmar el popup que aparece.
Posibles errores	• El botón de rechazar no realiza ninguna acción. • Al pulsar en rechazar, aparece un mensaje de error interno. • Una vez rechazado, la solicitud sigue en el listado. • No llega un email de información al usuario.
Resultado esperado	Aparece un mensaje de confirmación de que todo ha ido correctamente, la solicitud queda eliminada y el usuario recibe un email en su dirección de correo informando de la situación.

ID	P-19
Descripción	Aceptar solicitud de paso a artista
Pasos	• Acceder a la vista de Administración de solicitudes de paso a artista. • Pulsar en el botón de aceptar sobre una solicitud de paso a artista. • Confirmar el popup que aparece.
Posibles errores	• El botón de aceptar no realiza ninguna acción. • Al pulsar en aceptar, aparece un mensaje de error interno. • Una vez aceptado, el usuario no tiene rol de Artista.
Resultado esperado	Aparece un mensaje de confirmación de que todo ha ido correctamente y el usuario cambia su rol de Miembro a Artista.

ID	P-20
Descripción	Rechazar solicitud de paso a artista
Pasos	• Acceder a la vista de Administración de solicitudes de paso a artista. • Pulsar en el botón de rechazar sobre una solicitud de paso a artista. • Confirmar el popup que aparece.
Posibles errores	• El botón de rechazar no realiza ninguna acción. • Al pulsar en rechazar, aparece un mensaje de error interno. • Una vez rechazar, el usuario cambia al de Artista.
Resultado esperado	Aparece un mensaje de confirmación de que todo ha ido correctamente y el usuario mantiene su rol de Miembro.

7. Conclusiones

Tras haber finalizado todo el proceso de desarrollo de la aplicación, se describe en esta sección las conclusiones obtenidas, incluyendo un repaso a los objetivos cumplidos y a los tiempos de ejecución finales, futuras vías de desarrollo y una valoración personal final.

7.1 Objetivos cumplidos

En general, se han cumplido todos los objetivos del trabajo, tanto personales como de aplicación.

A nivel personal:

1. He realizado un profundo repaso a todos los conceptos de ingeniería del software que aprendí durante las asignaturas de la carrera, y que habían quedado en su mayoría olvidados durante los años de trabajo. Esta ha sido la parte en la que más he aprendido (o recordado) herramientas de todo el trabajo.

2. He mejorado mis habilidades con Python y Angular a la hora de implementar la aplicación. He aprendido también el uso de librerías externas, como las de visualización de documentos o reproducción de video, que de otra forma no hubiera conocido.

3. He reafirmado mi conocimiento a la hora de desplegar una aplicación web en la red: comprar un servidor, instalar paquetes (apache, postgreSQL…), generar una *build* de Angular, configurar Apache, arrancar en modo robusto el API, etc.

4. He tenido que investigar sobre cómo hacer de manera segura un mecanismo de autenticación.

A nivel de la plataforma:

1. Se ha diseñado una interfaz web muy sencilla, poco saturada de elementos, que es similar a un panel de administración estándar. Además, es *responsive*, esto es, todas las vista se adaptan a la versión móvil.

2. Se ha diseñado un sistema de gestión de usuarios fiable y probado que permite el registro de usuarios como en cualquier aplicación web profesional, la recuperación de contraseña olvidada y la gestión de usuarios por parte de los administradores. Además se ha implementado de manera funcional un sistema de roles para diferenciar las partes de la aplicación a las que tiene acceso cada usuario dependiendo del rol al que pertenece.

3. Se han implementado todos los componentes necesarios para permitir la gestión de recursos y anuncios, incluyendo el componente de mapa para geolocalizarlos.

4. Se han realizado desde cero todos los componentes necesarios para tener un sistema de mensajería interna totalmente funcional, que permite la comunicación entre usuarios de la plataforma.

5. La aplicación se ha desplegado en un VPS personal, realizando todas las configuraciones necesarias para que esté operativa en línea de manera estable.

Además, las demostraciones en diferentes entrevistas del resultado final al cliente (en nuestro caso, la persona que está desarrollando el proyecto cultural Sahara Soundscapes) han resultado

completamente satisfactorias, recibiendo un feedback positivo en el que se nos ha indicado que la aplicación cumplen con los objetivos esperados por su parte.

7.2 Desarrollo temporal real

La planificación temporal se ha desviado algo de los planteamientos iniciales. En la sección 2 se indicaba que el plan era acabar a finales de mayo, teniendo un mes y medio de sobra para corregir desviaciones. Aunque estaba previsto que la fase de documentación llegara hasta el final del plazo.

En la fase de desarrollo ha habido pequeñas desviaciones. Principalmente la parte de Recursos, que incluía el componente de mapa y los de gestión multimedia se han llevado un poco más de tiempo del previsto. Además se ha corregido la fase de pruebas, que se ha extendido durante toda la fase de desarrollo. El diagrama resultante es el siguiente:

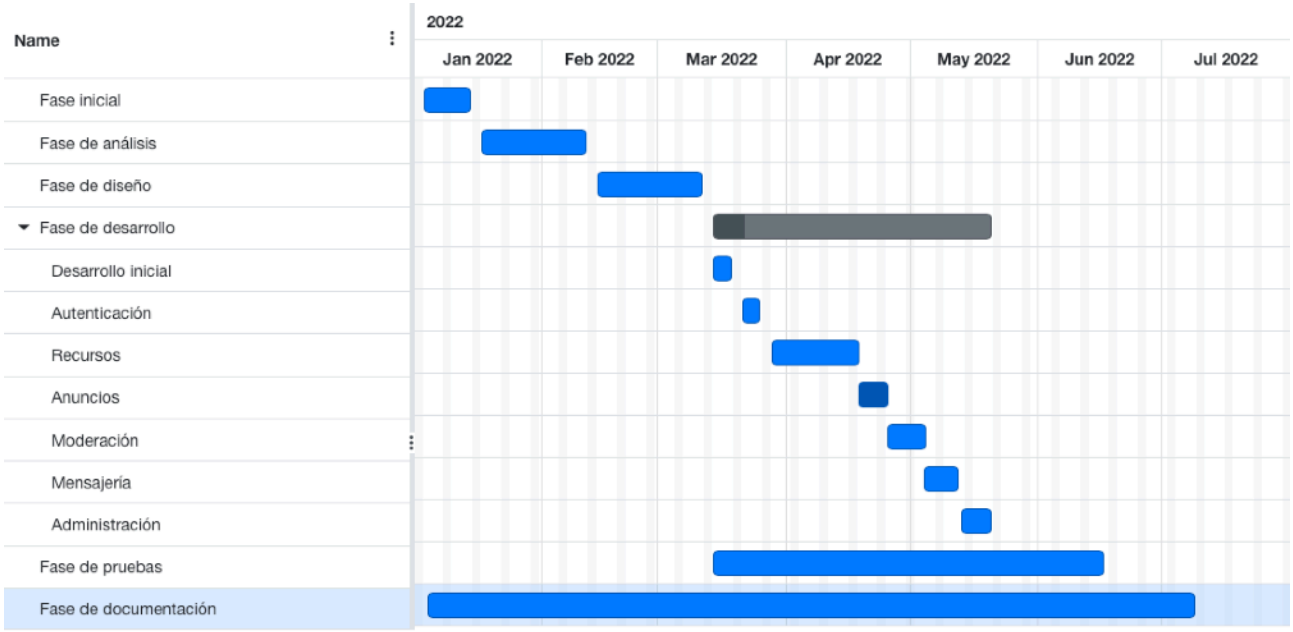


Figura 7.7: Diagrama de Gantt final.

7.3 Futuras vías de desarrollo

Este proyecto se abre a muchas mejoras, las cuales se pueden dividir en dos grupos: las que permiten mejorar las características del estado actual y las que añaden nueva funcionalidad.

Entre las del primer grupo, se pueden destacar:

- Realización de tests, sobre todo tests unitarios que no se han realizado por falta de tiempo pero que son fundamentales en una aplicación profesional. Además, existen otros tipos de tests como los de estrés, que vendrían bien para comprobar el rendimiento del sistema.

- Mejoras de seguridad. Investigar más profundamente en técnicas profesionales de código seguro para evitar ataques elaborados. Realizar estudios de tests de penetración para encontrar vulnerabilidades.

- Mejoras visuales. Los componentes que se visualizan pueden resultar algo básicos ya que se ha usado la librería de componentes de *boostrap*. Se pueden realizar componentes más personalizados para darle al cliente web un aspecto menos genérico.

Entre las del segundo grupo, podemos destacar:

- Una sección de enlaces de interés, donde los moderadores podrían añadir y eliminar enlaces a recursos externos de interés para la comunidad.

- Panel de estadísticas para administradores, donde obtener datos avanzados de uso de la aplicación.

- Integraciones con sistemas de alojamiento externos, como Google Drive, YouTube o Mega, para permitir alojar recursos en esos servicios.

Por último, una evolución interesante de este proyecto es convertirla en un sistema genérico personalizable que sirva a otras comunidades o agrupaciones a gestionar sus recursos y preservar cualquier cultura, no solo la saharaui.

7.4 Valoración personal

Ha sido una buena experiencia desarrollar este trabajo de fin de grado. Mi primer intento de realizar un TFG fue en el año 2017, pero al empezar las prácticas de empresa y comenzar a trabajar lo abandoné, y no ha sido hasta ahora que he encontrado la motivación para matricularme, empezar un nuevo proyecto y poder así terminar el grado. El hecho de que este trabajo sea para dar soporte a un fin social y que vaya a ser utilizado en la vida real para cubrir una necesidad de este tipo ha sido fundamental para encontrar las ganas de hacer todo lo que implica un trabajo de fin de grado.

A nivel técnico, el tener ya varios años de experiencia como desarrollador web, tanto en la parte de *backend* como de *frontend*, ha facilitado mucho la parte de la implementación. No he encontrado dificultades técnicas de gran envergadura que me bloquearan la fase de desarrollo y ha sido divertido desarrollar una aplicación web de este tipo.

La parte de análisis, diseño y documentación han sido sin duda las más laboriosas, ya que aunque esos conceptos se aprenden en las asignaturas, los tenía ya un poco olvidados. Por lo tanto, hacer este trabajo me ha servido para recordar esas herramientas y obtener nuevo conocimiento que puedo aplicar a mi día a día en el puesto de trabajo.

La idea es, aunque el trabajo de fin de grado termine aquí, continuar dando soporte tanto de mantenimiento como de desarrollo a esta aplicación, ya que es una plataforma sin ánimo de lucro que pretende aportar una pequeña ayuda al gran problema que sufre la comunidad saharaui.

Bibliografía y referencias

[1] Artículo en Diario Huffington Post, *España y el conflicto del Sáhara: claves para entender un giro histórico.* Recurso en línea consultado el día 27/06/2022. https://www.huffingtonpost.es/entry/espana-y-el-conflicto-del-sahara-claves-para-entender-un-giro-historico_es_6234f2b5e4b019fd812dec8a

[2] ACNUR (Agencia de la ONU para los refugiados). Recurso en línea consultado el día 14/06/2022. https://eacnur.org/es/actualidad/noticias/emergencias/refugiados-saharauis-campamentos-tinduf

[3] MediaLab Universidad de Granada. Recurso en línea consultado el día 14/06/2022 https://medialab.ugr.es/tag/sahara-soundscapes/

[4] Web oficial de Hispana. Recurso en línea consultado el día 14/06/2022 https://hispana.mcu.es/

[5] Web oficial del Repositorio de recursos culturales del Patronato de La Alhambra. Recurso en línea consultado el día 14/06/2022 https://www.alhambra-patronato.es/ria/

[6] Web oficial del Archivo digital del Instituto Andaluz de Patrimonio Histórico. Recurso en línea consultado el día 14/06/2022 https://repositorio.iaph.es

[7] Web oficial de OMEKA. Recurso en línea consultado el día 14/06/2022 https://omeka.org/classic/

[8] Web oficial de Duraspace. Recurso en línea consultado el día 14/06/2022 https://duraspace.org/dspace/

[9] Web oficial de KUNE project. Recurso en línea consultado el día 14/06/2022 https://kune.ourproject.org/es/

[10] Sommerville, Ian. (2016). *Software Engineering.* Pearson Education Limited.

[11] www.gantt.com. Recurso en línea consultado el día 18/06/2022 https://www.gantt.com

[12] Talent.com, web de empleo. Recurso en línea consultado el día 18/06/2022 https://es.talent.com/salary?job=full+stack

[13] Catálogo de VPS del proveedor OVH. Recurso en línea consultado el día 18/06/2022 https://www.ovhcloud.com/es-es/vps/

[14] Catálogo de dominios del proveedor IONOS. Recurso en línea consultado el día 18/06/2022 https://www.ionos.es/dominios/com-dominio

[15] Web oficial de la DCMI (Dublic Core Metadata Initiative). Recurso en línea consultado el día 22/06/2022 https://www.dublincore.org

[16] Gelperin, David. *Precise Use Cases.* Recurso en línea consultado el día *20/06/2022* https://www.methodsandtools.com/archive/archive.php?id=8

[17] Sommerville, Ian. *Software Engineering, chapter 5, section 5.5.2: Sequence Diagrams.* Pearson Education Limited.

[18] ESIC. *Modelo entidad relación: descripción y aplicaciones.* Recurso consultado en línea el día 22/06/2022 https://www.esic.edu/rethink/tecnologia/modelo-entidad-relacion-descripcion-aplicaciones

[19] StartBoostrap, catálogo de plantillas web. Recurso en línea consultado del día 22/06/2022 https://startbootstrap.com/theme/sb-admin-2

[20] Sun Microsystem. *Distributed Application Architecture (PDF).* Recurso en línea consultado el día 22/06/2022 https://web.archive.org/web/20110406121920/http://java.sun.com/developer/Books/jdbc/ch07.pdf

[21] Microsoft. *Deployment Patterns.* Recurso en línea consultado el día 30/06/2022 https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff648105(v=pandp.10)

[22] Fielding, Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures,* University of California*.* Recurso en línea consultado el día 23/06/2022 https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[23] Internet Engineering Task Force (IETF). *JSON Web Token (JWT) standard.* Recurso en línea consultado el día 23/06/2022 https://datatracker.ietf.org/doc/html/rfc7519

[24] Web oficial de Angular. Recurso en línea consultado el día 30/06/2022 https://angular.io/guide/what-is-angular

[25] Kumar, Ankit. *React vs. Angular vs. Vue.js: A Complete Comparison Guide.* Recurso en línea consultado el día 23/06/2022 https://dzone.com/articles/react-vs-angular-vs-vuejs-a-complete-comparison-gu

[26] Grinberg, Miguel. *Flask Web Development*, 2nd Edition. O'Reilly.

[27] Regina O. Obe, Leo S. Hsu. *PostgreSQL: Up and Running*, 3rd Edition. O'Reilly.

[28] Web oficial de VimeJS Recurso consultado en línea el día 23/06/2022. https://vimejs.com

[29] Paquete ngx-doc-viewer en repositorio oficial NPM. Recurso consultado en línea el día 23/06/2022. https://www.npmjs.com/package/ngx-doc-viewer

[30] w3sDesign. *The Dependency Injection design pattern - Problem, Solution, and Applicability.* Recurso consultado en línea el día 23/06/2022. http://w3sdesign.com/?gr=u01&ugr=proble

[31] Web oficial de OpenLayers. Recurso consultado en línea el día 23/06/2022 https://openlayers.org