UNIVERSITAT DE BARCELONA

Treball final de grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

Diseño e implementación de una red social en tiempo real basada en el diseño centrado en el usuario

Autor: Alejandro Espinosa Sánchez

Directora: Dra. Mireia Ribera Turró
Realizado en: Departament de Matemàtiques i Informàtica

Barcelona, 20 de junio de 2018

Abstract

User-centered design is a philosophy that aims to place the user at the center of the design in order to meet his needs and improve his experience. In this project this method will be used to design a social network.

A complete iteration of the user-centered design will be performed. For this, first, a research phase will be carried out where the ideal user profile will be defined and social networks will be analyzed to obtain similarities. From the inquiry, scenarios will be defined and the users will follow them to provide information of gaps in the interaction with the system.

Secondly, the sketches will be designed and analyzed using Nielsen's heuristics and finally a high-fidelity prototype will be designed. Based on the prototype, user tests will be carried out in order to detect usability problems and then design the social network.

To design the social network, on the backend, Django and a set of modules will be used to add new functionalities. In particular, Django Channels will be used for real-time communication and Django Rest Framework will be used to design an Application Programming Interface (API) Rest.

Two types of databases will be used, a REmote DIctionary Server (Redis) memory database and a My Structured Query Language (MySQL) database. The first database will store static templates and allow the exchange of messages between users and the second database will have the information of the social network data tables.

On the frontend, to design the interfaces, Bootstrap and the jQuery library will be used to give dynamism to the elements and make modifications in the HyperText Markup Language (HTML) document. Users will receive emails and these will be adaptable and intuitively designed to adapt to any screen resolution.

Keywords. User-centered design, human-computer interaction, user experience, web technologies, software engineering, social networks, real-time communication.

Resumen

El diseño centrado en el usuario es una filosofía que tiene como objetivo situar al usuario en el centro del diseño con la finalidad de satisfacer sus necesidades y de mejorar su experiencia. En este proyecto se utilizará este método para diseñar la red social.

Se realizará una iteración completa del diseño centrado en el usuario. Para ello, en primer lugar, se realizará una fase de investigación donde se definirá el perfil de usuario ideal y se indagará en redes sociales para obtener similitudes. De la indagación, se definirán escenarios que los usuarios deberán realizar y proporcionarán información de carencias en la interacción con el sistema.

En segundo lugar, se diseñarán los esbozos y se analizarán utilizando las heuristicas de Nielsen y finalmente se diseñará un prototipo de alta fidelidad. En base al prototipo, se realizarán pruebas de usuarios con la finalidad de detectar problemas de usabilidad y posteriormente diseñar la red social.

Para diseñar la red social, en la parte del servidor, se utilizará Django y un conjunto de módulos para añadir nuevas funcionalidades. En particular, se utilizará Django Channels para la comunicación en tiempo real y Django Rest Framework para diseñar un API Rest.

Se utilizarán dos tipos de bases de datos, una base de datos en memoria Redis y una base de datos MySQL. La primera base datos almacenará plantillas estáticas y permitirá el intercambio de mensajes entre usuarios y la información de las tablas de la red social.

En la parte del cliente, para diseñar las interfaces se utilizará Bootstrap y la librería jQuery para dar dinamismo a los elementos y realizar modificaciones en el documento HTML. Los usuarios recibirán correos electrónicos y éstos serán adaptables y diseñados intuitivamente para que se adapten a cualquier resolución.

Palabras clave. Diseño centrado en el usuario, interacción persona-ordenador, experiencia de usuario, tecnologías web, ingeniería del software, redes sociales, comunicación en tiempo real.

Resum

El disseny centrat en l'usuari és una filosofia que té com a objectiu situar a l'usuari en el centre del disseny amb l'objectiu de satisfer les seves necessitats i de millorar la seva experiència. En aquest projecte s'utilitzarà aquest mètode per a dissenyar la xarxa social.

Es realitzarà una iteració completa del disseny centrat en l'usuari. Per a això, en primer lloc, es realitzarà una fase d'investigació on es definirà el perfil d'usuari ideal i s'indagarà en xarxes socials per obtenir similituds. De la indagació, es definiran escenaris que els usuaris hauran de realitzar i proporcionaran informació de mancances en la interacció amb el sistema.

En segon lloc, es dissenyaran els esbossos i s'analitzaran utilitzant les heurístiques de Nielsen i finalment es dissenyarà un prototip d'alta fidelitat. Sobre la base del prototip, es realitzaran proves d'usuaris amb la finalitat de detectar problemes d'usabilitat i posteriorment dissenyar la xarxa social.

Per dissenyar la xarxa social, a la part del servidor, s'utilitzarà Django i un conjunt de mòduls per afegir noves funcionalitats. En particular, s'utilitzarà Django Channels per a la comunicació en temps real i Django Rest Framework per dissenyar un API Rest.

S'utilitzaran dos tipus de bases de dades, una base de dades en memòria anomenada Redis i una base de dades relacional MySQL. La primera base dades emmagatzemarà plantilles estàtiques i permetrà l'intercanvi de missatges entre usuaris i la segona emmagatzemarà la informació de les taules de la xarxa social.

A la part del client, per dissenyar les interfícies s'utilitzarà Bootstrap i la llibreria jQuery per donar dinamisme als elements i realitzar modificacions en el document HTML. Els usuaris rebran correus electrònics i aquests seran adaptables i dissenyats intuïtivament perquè s'adaptin a qualsevol resolució.

Paraules clau. Disseny centrat en l'usuari, interacció persona-ordinador, experiència d'usuari, tecnologies web, enginyeria del programari, xarxes socials, comunicació en temps real.

Índice

Índice de figuras

Índice de cuadros

Glosario

Application Programming Interface (API) Application Programming Interface (API) es un conjunto de reglas (código) y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas sirviendo de interfaz entre programas. 1–3, 51

Asynchronous JavaScript And XML (AJAX) Asynchronous JavaScript And XML (AJAX) es un conjunto de tecnologías que permiten hacer páginas dinámicas y solicitar información de forma asíncrona. 55, 61–63, 65, 70, 73, 76–78, 80, 83, 86, 101

Engine X (Nginx) Engine X (Nginx) es un servidor web/proxy inverso ligero de alto rendimiento y un proxy para protocolos de correo electrónico, es software libre y de código abierto. 46

HyperText Markup Language (HTML) HyperText Markup Language (HTML) es un lenguaje que permite describir la estructura y el contenido en forma de texto para la construcción de páginas web. 1–3, 6, 19, 48, 56–58, 60, 63, 73, 76, 77, 80, 83

JavaScript Object Notation (JSON) JavaScript Object Notation (JSON) es un formato en texto plano utilizado para transmitir datos en aplicaciones web. 51, 62, 73, 76, 77, 79, 80, 83, 85

My Structured Query Language (MySQL) My Structured Query Language (MySQL) es una base de datos relacional de código abierto basada en el lenguaje estructurado SQL. 1–3, 46, 59

Object Relational Mapping (ORM) Object Relational Mapping (ORM) es un modelo de programación que abstrae al programador de la base de datos convirtiendo las tablas de una base de datos en un conjunto de entidades y que permite simplificar las consultas y proporcionar seguridad contra ataques al haber una capa de acceso a datos intermedia. 48

Open Authorization (OAuth2) Open Authorization (OAuth2) es un protocolo de autorización que permite que los usuarios autoricen a terceros a acceder a su información sin necesitar conocer las credenciales. 53

REmote DIctionary Server (Redis) REmote DIctionary Server (Redis) es una base de datos libre no relacional en memoria de tipo clave-valor. Permite guardar un conjunto amplio de estructuras de datos. 1–3, 46, 49, 59

Standard Generalized Markup Language (SGML) Standard Generalized Markup Language (SGML) es un estándar que permite definir lenguajes para dar formato a documentos. No especifica ningún formato especifico; más bien, define reglas para etiquetar elementos. 19

URL dispatcher URL dispatcher es un sistema de Django que hace la correspondencia de la dirección URL a un controlador. 48, 51

WebSockets WebSockets es una tecnología que proporciona un canal bidireccional entre el navegador y el servidor, permitiendo enviar y recibir mensajes de forma simultánea. 11, 46, 49, 50, 70, 88, 89, 91

Agradecimientos

Quisiera agradecer a mis padres que me han apoyado, animado y brindado su paciencia tanto en los buenos y malos momentos.

Agradecer a Mireia Ribera por guiarme correctamente durante todo el proyecto, atendiendo a las consultas y tutorías con un trato excepcional.

Finalmente, agradecer a las personas que han contribuido realizando pruebas de usuarios.

1. Introducción

En la actualidad el uso de las redes sociales está muy extendido en un público de todas las edades. Con la llegada de la web 2.0 o también llamada web social, permitió a los desarrolladores cambiar la metodología de desarrollar sus aplicaciones.

Antiguamente, las páginas web eran estáticas y normalmente su contenido no cambiaba. Actualmente, los usuarios interaccionan con los servicios web pudiendo crear contenido y formar parte de sociedades.

Este hecho es interesante, ya que proporcionar al usuario capacidad de comunicación y participación activa ha permitido desarrollar servicios que utilizan la *inteligencia colectiva*.

A raíz de esta evolución tecnológica, han surgido o se han mejorado una serie servicios, entre los que se pueden destacar los blogs, las wikipedias o redes sociales. En particular, se podría decir que las redes sociales son los servicios que más se están beneficiando de estas nuevas tecnologías.

A través de esta evolución tecnológica, las redes sociales han permitido mejorar la forma de establecer contactos y poder comunicarse con grupos de personas dando lugar a dos clasificaciones: redes sociales horizontales y verticales.

Si nos centramos en la primera clasificación de redes horizontales se encontrarían el conjunto de redes que no tienen ninguna temática en particular. Dos ejemplos de este tipo de redes podrían ser *Twitter* o *Facebook*.

Por otro lado, en el grupo de las redes sociales verticales, el contenido está más enfocado a una temática en particular y dirigida a un conjunto de personas con unos intereses o aficiones comunes. Dos ejemplos de este grupo podrían ser LinkedIn e Instagram. El primero está orientado a redes de contactos profesionales y el segundo a fotos, vídeos y aplicar efectos fotográficos para posteriormente poder compartirlas con otros usuarios.

1.1. Motivación

A lo largo de mi trayectoria de estudiante me ha motivado el uso de las nuevas tecnologías y concretamente la programación web, redes sociales y el software distribuido. Además, el incremento en la utilización de las redes sociales como fuente de comunicación y intercambio de información fue un punto motivador a que realizara una red social.

La razón de enfocar el proyecto en la experiencia de usuario es porque no todas las páginas web tienen en cuenta al usuario dando lugar a diseños que no son atractivos y fáciles de usar para el usuario.
Así pues, teniendo en cuenta esa premisa, enfocar el diseño a los usuario me permitirá diseñar los diseños para que sean fáciles de usar, atractivos al usuario y como consecuencia, mejorar la experiencia de usuario.

Como propósito técnico de este trabajo es tener la oportunidad de mejorar el conocimiento en lenguajes de programación orientados a páginas web y expandir el conocimiento en nuevas tecnologías para desarrollar páginas web que están en la actualidad presentes en sector tecnológico. Un ejemplo son los WebSockets que se están extendiendo con rapidez y permiten establecer un canal bidireccional de comunicación entre el servidor y cliente.

1.2. Objetivos

En términos generales, este trabajo de fin de grado tiene como objetivo principal, diseñar e implementar una red social en tiempo real utilizando la filosofía del diseño centrado en el usuario.

Con el fin de lograr el objetivo principal, la red social tendrá los siguientes objetivos específicos.

- Cada usuario tendrá un perfil de usuario que será visible por otros usuarios y en el cual se incluirán campos como el nombre, apellidos, fecha de nacimiento, una imagen de perfil y portada. Esos campos serán modificables y, una vez el usuario haga alguna modificación en su perfil, otros usuarios podrán recibir esas modificaciones en tiempo real.
- Cada usuario podrá crear, visualizar y eliminar mensajes en su propio muro. Asimismo, se dará la funcionalidad que otros usuarios puedan recibir los cambios realizados en el muro justo en el momento de ser efectuados.
- Los usuarios podrán crear álbumes de fotos y eliminarlos.
- Los mensajes y álbumes de fotografías tendrán un nivel de privacidad dependiendo de la visibilidad que el usuario seleccione.
- El usuario podrá interactuar con otros usuarios enviando solicitudes de amistad para que el otro usuario pueda aceptarlas o rechazarlas. Además el otro usuario recibirá un correo que podrá utilizar para aceptar o rechazar la solicitud.
- El usuario podrá buscar en sus amigos y en los amigos de sus amigos.
- El usuario podrá bloquear a otros usuarios y desbloquear aquellos que previamente haya bloqueado.
- El usuario podrá indicar *"Me gusta"* en sus publicaciones o a las de otros usuarios (siempre que sean amigos o sea pública).
- Dar la capacidad a los usuarios para buscar a otros usuarios.
- Disponer de un acceso rápido y en tiempo real de los eventos que sucedan en la aplicación entre las cuales se incluyen publicar un mensaje, me gusta un mensaje y otras destinadas a conocer cuando el usuario recibe una solicitud de amistad.

1.3. Estructura de la memoria

La memoria está estructurada en forma de capítulos con las diferentes fases y partes del proyecto.

- Capítulo 1. **Introducción**. En el primer capítulo, se definirá el proyecto, los objetivos, metodología y finalmente la planificación .
- Capítulo 2. **Estado del arte**. Antecedentes y evolución de las redes sociales.
- Capítulo 3. **Especificación de requisitos**. En este capítulo se especificarán los requisitos funcionales de la aplicación web.
- Capítulo 4. **Diseño centrado en el usuario**. Este capítulo estará enfocado en realizar un proceso iterativo del desarrollo centrado en el usuario.
- Capítulo 5. **Análisis de la aplicación**. Capitulo dedicado a analizar la aplicaciones describiendo los casos de uso, base de datos y estructura de directorios.
- Capítulo 6. **Tecnologías utilizadas**. Capitulo dedicado a explicar las tecnologías que se utilizaron para desarrollar el proyecto.
- Capítulo 7. **Iteraciones**. Capitulo que contiene las iteraciones planificadas para realizar el proyecto.
- Capítulo 8. **Conclusiones**. Conclusiones obtenidas una vez acabado el proyecto.
- Capítulo 9. **Líneas futuras de desarrollo**. Posibles nuevas implementaciones.

1.4. Metodología

Para la realización del proyecto, se optó por la aplicación de una metodología ágil, en concreto **SCRUM**

Este enfoque metodológico posee una serie de características atractivas que las hacen interesantes para desarrollar este proyecto, entre la serie de ventajas que proporciona aplicar esta metodología se encuentra; en primer lugar la **panificación** y en segundo lugar la **gestión de tareas**.

Al margen de eso, se pensó que en un trabajo de fin de grado que es realizado por un estudiante no sería del todo adecuado utilizar este método ya que las metodologías ágiles están más enfocadas al trabajo en equipo. No obstante, desde el punto de vista de la forma de **organización de las historias de usuario** podría ser productivo por el hecho de poder establecer pequeños incrementos quincenales.

Un punto importante que favoreció utilizar esta metodología ágil fue **eliminar los problemas de una metodología en cascada** Inicialmente se pensó como un posible método, pero finalmente se decidió no decantarse por su utilización debido a que no permite el retroceso de una tarea por ser un proceso secuencial.

1.5. Planificación

Para obtener una aproximación del tiempo necesario para realizar las tareas del proyecto se utilizó el diagrama de Gantt. Este tipo de diagrama se utilizó con el objetivo de gestionar y planificar el tiempo de dedicación previsto para cada actividad de una forma visual y contrastar si se realizó una buena planificación una vez finalizado el sprint.

Al inicio de cada sprint se realizó una planificación sobre el tiempo estimado para realizar las unidades mínimas de trabajo del proyecto. Conviene destacar que el proyecto se presentaba en el semestre de primavera pero se empezó a realizar en el semestre de invierno.

Concretando las fechas de inicio y fin de proyecto previstas, la duración del proyecto teniendo en cuenta que la primera reunión fue el 2 de octubre de 2017 y se planificó como fecha de fin de proyecto, el día de la entrega de la memoria, que es el día 27 de Junio de 2018. Con esos intervalos de fechas, la duración del proyecto es de aproximadamente 38 semanas.

Una vez concretadas las fechas de inicio y fin se procedió a organizar el tiempo en actividades en un diagrama de Gannt. Para cada sprint se hicieron controles quincenales para conocer el estado del proyecto.

La duración en horas del trabajo de fin de grado son 18 créditos y cada crédito equivale a 30h de trabajo, se tienen por lo tanto 540 horas de dedicación.

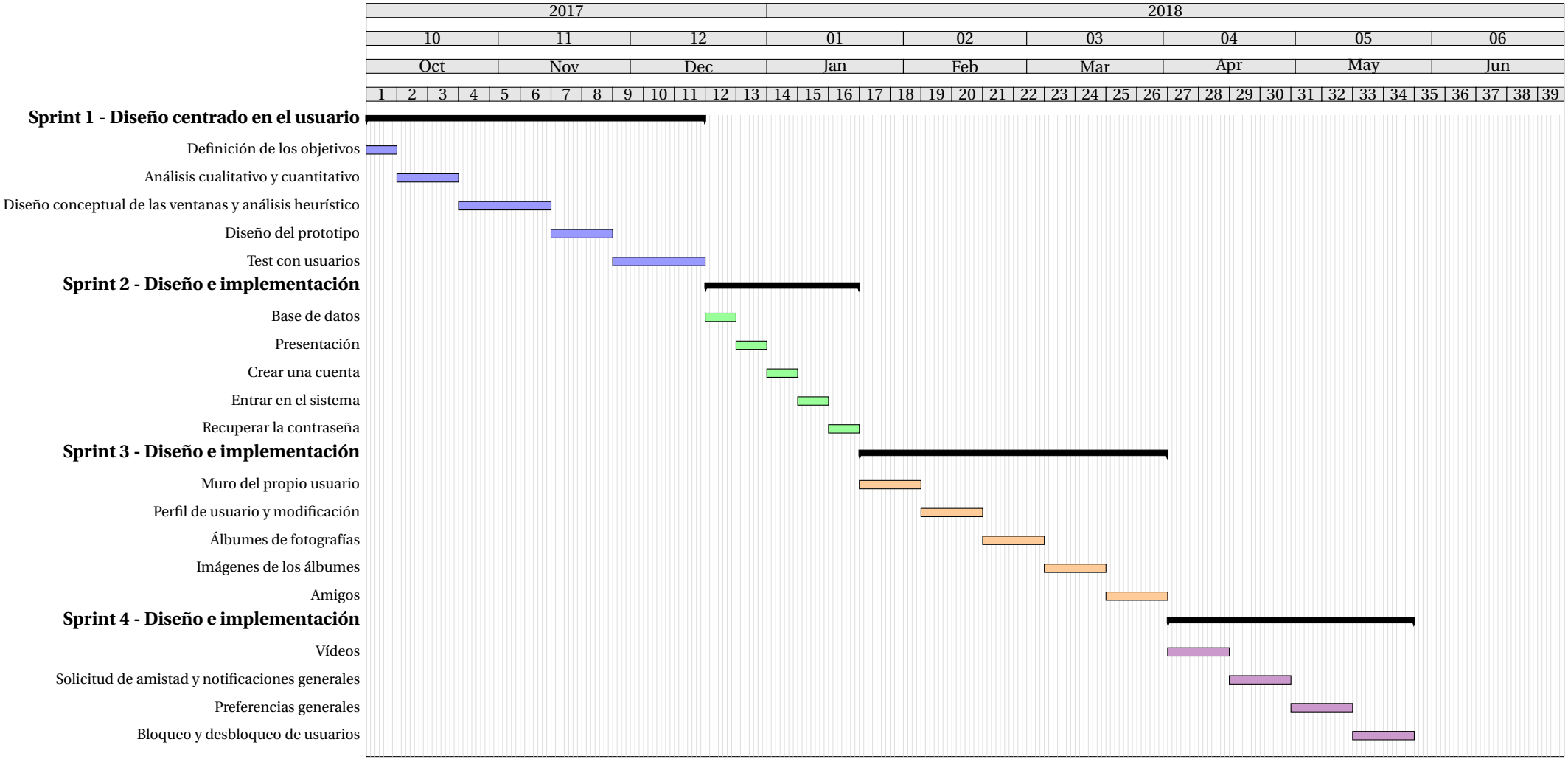

Figura 1: Planificación inicial

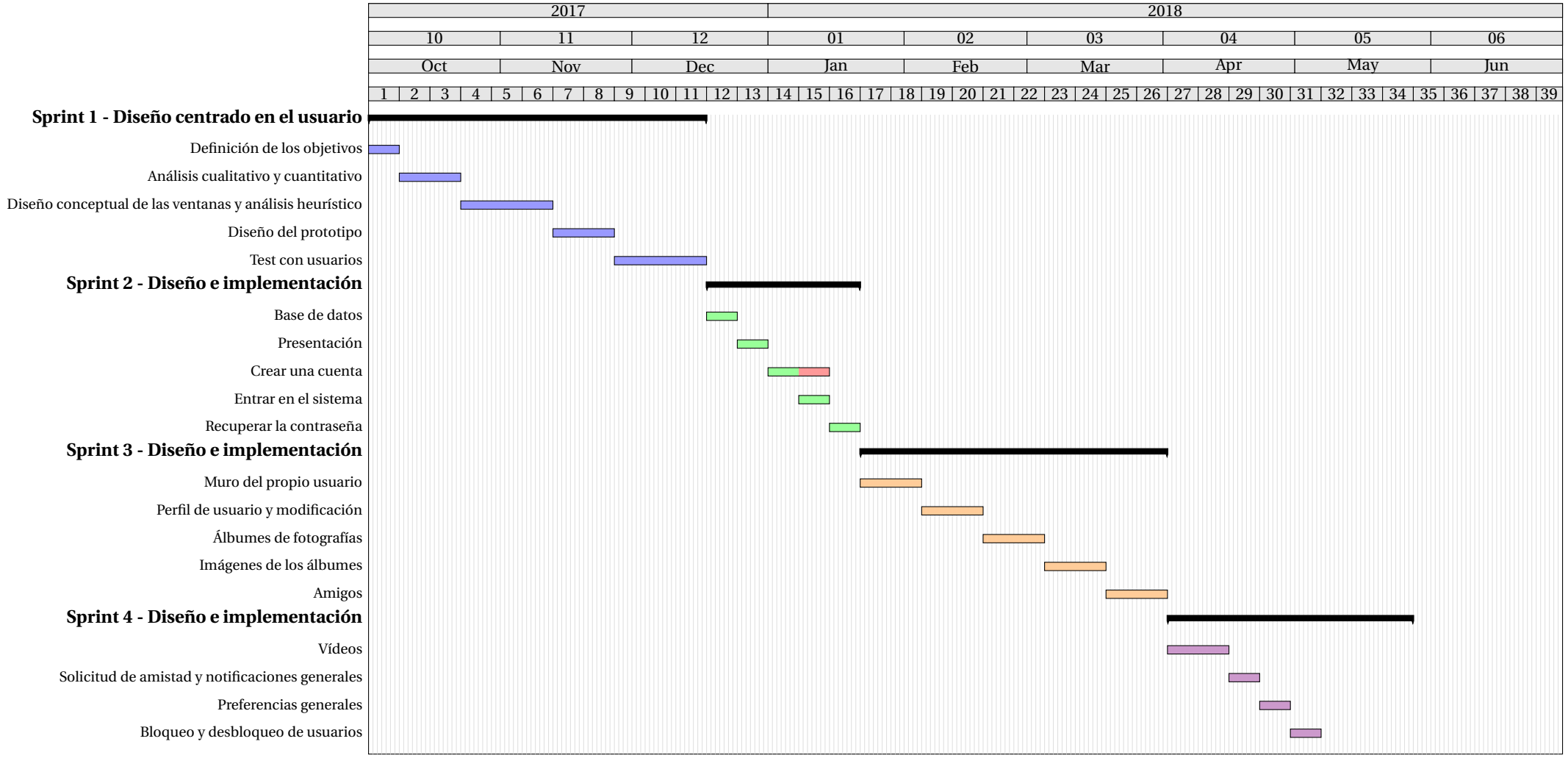

Figura 2: Planificación final

2. Especificación de requisitos

Siguiendo con la metodología ágil, en este capítulo se describen las historias de usuario (HU) que en términos generales son pequeñas descripciones que definen los requisitos del proyecto utilizando el lenguaje común del usuario. Para describir las historias de usuario es importante conocer el rol de usuario que se clasifica de dos formas: visitante y autentificado. El primero para usuarios que no han iniciado sesión y el segundo para usuarios que ya han iniciado sesión.

Una vez definida que es una historia de usuario y conocidos los roles de usuario, se procederá a describir las historias de usuario.

Cabe destacar que debido a un número elevado de requisitos funcionales, en este capítulo se muestran los correspondientes a los visitantes. Por otra parte, en la sección del anexo (véase A.4), se encuentran los requisitos funcionales del usuario autentificado.

Historia de usuario	**Inicio de sesión con usuario y contraseña**.
Descripción	Como usuario visitante, deseo iniciar sesión al sistema para poder hacer uso de las funcionalidades del sistema.
	Criterios de aceptación
Nombre del criterio	**Descripción**
Inicio de sesión exitoso	Cuando el nombre de usuario y contraseña sean válidos y el correo electrónico esté verificado entonces el sistema permitirá el acceso al sistema.
Inicio de sesión no válido	Cuando el nombre de usuario o contraseña no se encuentren en la base de datos entonces el sistema notificará al usuario el mensaje ”Revise el usuario o contraseña introducidos”.
Sin confirmación de email	Cuando el correo electrónico del usuario no haya sido verificado, entonces el sistema notificará al usuario con el mensaje ” Por favor, confirme su correo electrónico antes de iniciar sesión”.

Cuadro 1: HU-01 – Inicio de sesión con usuario y contraseña.

Historia de usuario	**Iniciar sesión con una cuenta social**.
Descripción	Como usuario visitante, deseo iniciar sesión al sistema usando una cuenta social, para poder hacer uso de las funcionalidades del sistema.
	Criterios de aceptación
Nombre del criterio	**Descripción**
Existe el correo electrónico	Cuando exista el correo electrónico en el sistema entonces el sistema iniciará sesión al usuario.
Creación de una cuenta	Cuando no exista un correo electrónico, entonces el sistema iniciará el proceso de creación de una cuenta.
Datos de perfil no encontrados	Cuando falten datos, entonces el sistema iniciará el proceso de recogida de esos datos.

Cuadro 2: HU-02 – Iniciar sesión con una cuenta social.

Historia de usuario	**Creación de una cuenta**.
Descripción	Como usuario visitante, deseo crear una cuenta en el sistema para poder hacer uso de las funcionalidades del sistema.
	Criterios de aceptación
Nombre del criterio	**Descripción**
Validación de los campos	Cuando los campos sean válidos entonces el sistema notificará visualmente al usuario que los campos cumplen con los requerimientos mínimos .
Enviar un correo electrónico	Cuando el visitante se haya creado una cuenta, entonces el sistema le enviará un correo y el sistema mostrará que se ha enviado un mensaje para activar la cuenta.
Crear una cuenta	Cuando el usuario envié sus datos de usuario y perfil, entonces el sistema no permitirá el acceso al sistema mostrando el mensaje “Por favor, verifique su correo electrónico”

Cuadro 3: HU-03 – Creación de una cuenta.

Historia de usuario	**Recuperar la contraseña**.
Descripción	Como usuario visitante, deseo recuperar la contraseña de la cuenta de modo que pueda tener acceso a mi cuenta de nuevo.
	Criterios de aceptación
Nombre del criterio	**Descripción**
Enviar un correo electrónico	Cuando el visitante envié el formulario al sistema, entonces el sistema le enviará un correo electrónico con un formulario para introducir una nueva contraseña.
Correo electrónico no detectado	Cuando el correo electrónico no exista en la base de datos entonces el sistema informará al usuario con el mensaje ”El correo electrónico introducido no se ha detectado en el sistema.”.

Cuadro 4: HU-04 – Recuperar la contraseña.

Historia de usuario	**Creación de una cuenta con una cuenta social**.
Descripción	Como usuario visitante, deseo crear una cuenta utilizando una red social para poder hacer uso de las funcionalidades del sistema.
	Criterios de aceptación
Nombre del criterio	**Descripción**
Existe el correo electrónico	Cuando exista el correo electrónico en la base de datos entonces el sistema iniciará sesión al usuario .
Creación de una cuenta	Cuando no exista un correo electrónico, entonces el sistema iniciará el proceso de creación de una cuenta.
Datos de perfil no encontrados	Cuando falten datos entonces el sistema iniciará el proceso de recogida de esos datos.

Cuadro 5: HU-05 – Creación de una cuenta con una cuenta social.

3. Estado del arte

En este capítulo se hará una breve cronología sobre la evolución de las redes sociales destacando cronológicamente los acontecimientos más importantes que inspiraron a las actuales redes sociales utilizando como fuente de información, la cita a la que se hace referencia [1].

Entrando más en contexto, la primera comunicación se realizó en el año **1971** cuando *Ray Tomlinson* envió un correo electrónico a través de un ordenador en la red ARPANET.

Otro acontecimiento importante se produjo en el **año 1978** cuando se empezó a distribuir un nuevo mecanismo de comunicación basada en tablones de anuncios. Esta tecnología fue el precursor de los foros.

Por otra parte, a principios del **año 1990**, *Tim Berners-Lee* implementó el lenguaje HTML a partir de un subconjunto de etiquetas del metalenguaje Standard Generalized Markup Language (SGML). Inicialmente fue ideado con el objetivo de intercambiar documentos científicos entre miembros del CERN utilizando el sistema de distribución de documentos *World Wide Web* creado por *Tim Berners*.

Avanzando cronológicamente en el tiempo, en el **año 1995**, se creó la primera red social **Classmates**. Una red social de temática general creada por *Randy Conrads* con el objetivo de ayudar a las personas a buscar amigos.

En el **año 1997** se inauguró **SixDegrees**, la primera red social horizontal que destacó por el hecho de permitir crear un perfil de usuario personal, agrupar contactos e intercambiar mensajes entre ellos. Como curiosidad, el nombre **SixDegrees** proviene de una hipótesis que intenta probar que cualquier persona estaba conectada con otra persona en menos de 6 conocidos. Es decir, intenta probar que cualquiera puede estar conectado a cualquier otra persona a través de una cadena de conocidos que no tiene más de cinco intermediarios (conectando a ambas personas con sólo seis enlaces).

La hipótesis fue propuesta por Frigyes Karinthy en el año 1930 y está basada en la idea de que el número de conocidos crece exponencialmente con el número de enlaces en la cadena. [1]

A partir del **año 2000**, las redes sociales nuevas marcaron una diferencia con sus antecesoras. La expansión de internet se hizo una realidad y se volvió una herramienta indispensable para la comunicación entre las personas. Con esto llegó la Web 2.0 o web social que permitió allanar el camino para la llegada de nuevas tecnologías y formas de interacción entre usuarios.

Si nos centramos en el **año 2002** nació **Friendster** que fue la primera red de temática general en implementar un sistema inteligente capaz de relacionar a los usuarios según sus gustos. Un año posterior nacieron varias redes sociales, en particular, **MySpace**, **LinkedIn** y **Facebook**.

En el **año 2005** se inauguró **YouTube** como un servicio de alojamiento de vídeos y un año siguiente **Twitter**, una red social de *microblogging* que permitió a los usuarios enviar y leer mensajes cortos de 140 caracteres llamados *tweets*.

[1] http://cisolog.com/sociologia/teoria-de-los-seis-grados-de-separacion/

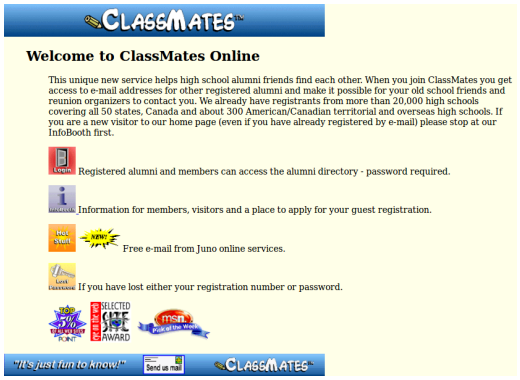

Figura 3: Classmates. Año 1996.

Visualizar

Figura 4: LinkedIn. Año 2005.

Visualizar

Figura 5: Facebook. Año 2005.

Visualizar

4. Diseño centrado en el usuario

El propósito de este capítulo es realizar una iteración completa del diseño centrado en el usuario (DCU) que se describe como una filosofía y un proceso de desarrollo que sitúa las características y necesidades del usuario en cada una de las etapas de diseño con el objetivo de crear productos útiles y usables; es decir, que satisfagan sus necesidades teniendo en cuenta sus características.

Se divide en cuatro etapas, estas son: *investigación y análisis de los usuarios*, *diseño* y *evaluación*. Cabe destacar que se trata de un proceso iterativo, ya que en cada etapa hay una retroalimentación de los usuarios para mejorar y adaptar los diseños.

Para llevar a cabo el diseño centrado en el usuario, en primer lugar se deben comprender a las personas, sus motivaciones y sus necesidades y en segundo lugar, ser consciente que las personas no piensa de la misma forma ante un problema. En la figura 6 se muestra una iteración en la que se observa todo el proceso.

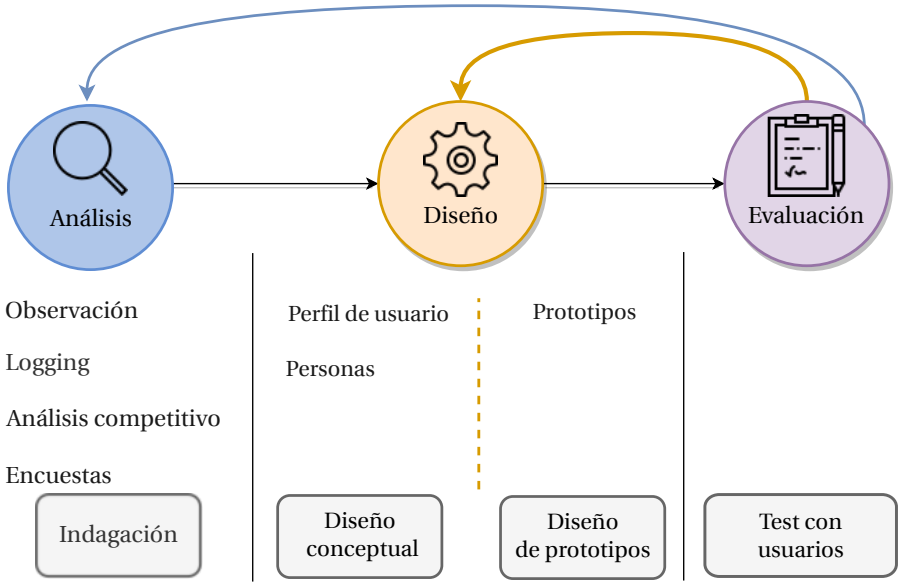

Figura 6: Etapas del diseño centrado en el usuario.

Adaptado de [2]

4.1. Análisis

En esta etapa primeramente se definirán las bases del proyecto a desarrollar, perfil de usuario objetivo y se indagará sobre otras redes sociales sociales.

Posteriormente se realizará un estudio cualitativo haciendo una cuestionario y finalmente un análisis cuantitativo sobre el uso por parte de los usuarios de las interfaces con el objetivo de extraer información a partir de la observación esos usuarios.

4.1.1. Definición del perfil de usuario ideal

Primeramente se empezará definiendo el perfil de persona ideal que estaría interesado en la aplicación. En general, para describir un perfil de usuario se utilizan dos tipos de variables: duras y blandas.

El primer tipo de variable son las denominadas demográficas y describen la edad, sexo y nivel socio-económico. Por otro lado, las variables blandas permiten conocer al individuo por su personalidad, estilo de vida o valores.

Si tenemos en cuenta estas dos variables, el perfil ideal sería una persona joven comunicativa, extrovertida y con ganas de compartir experiencias con el resto de personas.

Si fijamos el perfil de las nuevas tecnologías y comunicación, una persona con unos conocimientos técnicos mínimos debería ser capaz de navegar sin problemas por la aplicación.

4.1.2. Elección justificada de interfaces similares

En la actualidad, hay una gran cantidad de redes sociales con diferentes temáticas pero en este proyecto se eligió Facebook y LinkedIn por ser las dos redes sociales más utilizadas. [2]

Facebook

Es una red social que pertenece al grupo de las redes sociales horizontales y tiene el objetivo de conectar personas con personas. Tiene muchas funcionalidades entre las cuales se incluyen crear álbumes de fotografías, encontrar amigos, compartir experiencias, recursos, organizar grupos o incluso crear encuestas. Además, da soporte de integración con otras aplicaciones.

Después de haber navegado por la aplicación se detectaron algunos puntos fuertes y débiles.

Como puntos fuertes identificados se destacan:

- Proporciona una interfaz limpia y sin ruido visual.
  El tamaño, estética y contraste cumplen los mínimos exigidos y permite leer con claridad y facilidad el texto del servicio web.
- Hace un buen uso el espacio visual y existen zonas en blanco para poder descansar la vista.
- Proporciona un menú en la parte superior con el que permite buscar personas, grupos y páginas. Además proporciona un apartado de ayuda en la parte superior derecha del menú.
- Describe el contenido de un imagen con una etiqueta *alt* proporcionando información para dispositivos que no puedan visualizar la imagen.

[2] http://www.hydrasocialmedia.com/blog/tendencias-en-redes-sociales-en-2018/

Como punto débil detectado sería:

- En la pantalla principal, las cajas de texto que permiten introducir el correo electrónico y contraseña no son lo suficientemente anchas para correos electrónicos de muchos caracteres. Podría darse el caso que un correo electrónico con una longitud superior al ancho de la caja de texto no permitiera visualizar correctamente un correo sin tener que desplazarse por la caja.

LinkedIn

LinkedIn es una red orientada a relaciones profesionales. Dispone de muchas funcionalidades pero se destacan principalmente la creación de conexiones con otros usuarios, grupos de usuarios y también permite validar conocimientos de otras personas y incluir recomendaciones de otros compañeros. Además da la opción de confeccionar un curriculum y crear un videocurriculum.

Como en el caso de Facebook, se analizará para descubrir puntos fuertes y débiles de LinkedIn.

Como puntos fuertes detectados podría destacar:

- El usuario recibe continuamente información sobre el estado actual de los formularios gracias a las comprobaciones que realiza el sistema para comprobar si hay algún error.
- Proporciona un diseño intuitivo y los iconos junto con el texto ofrecen una buena experiencia de usuario.
- La tipografía y el color de los textos es correcta. Además, mantiene el nivel de contraste con el fondo y utiliza pocos tipos de letra para no provocar fatiga visual.
- Utilizan rutas semánticas (amigables) y han evitado utilizar rutas complejas para acceder a recursos.

Se encontró un punto débil que tiene relevancia en el hecho de proporcionar una ayuda que se agradece pero no se localiza fácilmente. En contraste con Facebook que utiliza un icono situado en la parte superior derecha.

4.1.3. Indagación

Una vez definidas las bases, se empezó a indagar sobre el uso de las redes sociales con el propósito de explorar otras redes .

Existen diferentes métodos para llevar a cabo este tipo de investigación: *cuantitativos* y *cualitativos*.

4.1.3.1. Métodos cuantitativos

Se utilizó un cuestionario [3] que es una técnica cuantitativa que consiste en un conjunto de preguntas cerradas.

El cuestionario estaba orientado a conocer los hábitos y usos de las redes sociales con el objetivo de crear un perfil de usuario. Para ello, se realizó el cuestionario a 13 encuestados y se obtuvieron los siguientes resultados.

[3] https://goo.gl/XUh5Uc

La primera pregunta estaba orientada a conocer las edades del público objetivo.
En general, hay un rango de edades comprendidas entre 17 años y 31 años. Las cifras más significativas indican que posiblemente el publico objetivo se encontraría entre 21 y 25 años.

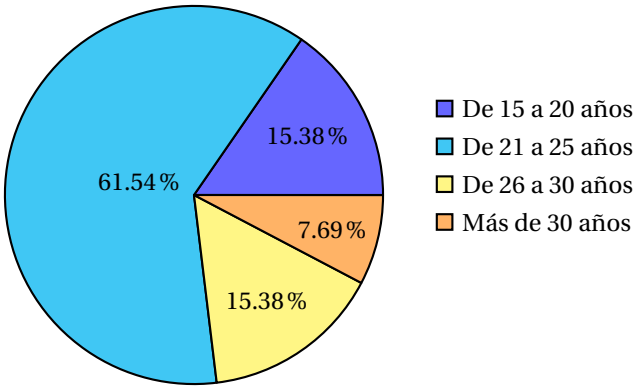

Figura 7: Rango de edades de utilización de redes sociales.

La segunda pregunta estaba orientada a identificar que redes sociales eran las más utilizadas.

A partir de los datos obtenidos, se concluye que la red más utilizada es Facebook y la segunda Instagram. Estos datos indican que las personas que las utilizan se centran en el intercambio de imágenes y vídeos entre usuarios y a comunicarse entre amigos y familiares.

En el caso de Twitter, un 15,4 % de la muestra las utiliza posiblemente por el tipo de comunicación basada en mensajes limitados a 140 caracteres.

Se observa que LinkedIn no está tan extendida entre los usuarios de la muestra o no están buscando trabajo.

Por último, se observa que una minoría no tiene ningún contacto con ninguna red social.

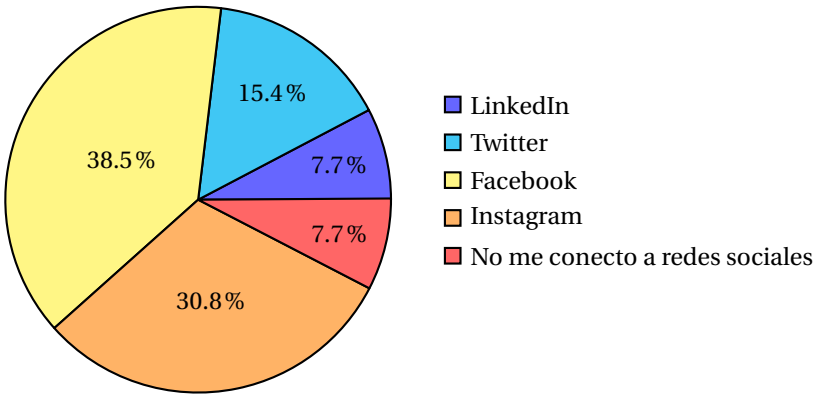

Figura 8: Redes sociales más utilizadas.

La tercera pregunta estaba orientada a analizar el tiempo diario de utilización en las redes sociales.

De la muestra, se observa que un 53,9 % utilizan las redes sociales más de 1 hora por día, un 38,5 % entre 15 minutos y 1 hora por día y una minoría no se conecta a las redes sociales.

A partir de los datos se observa que hay una dependencia a las redes sociales ya sea para comunicarse con otras personas o compartir contenido con otros usuarios.

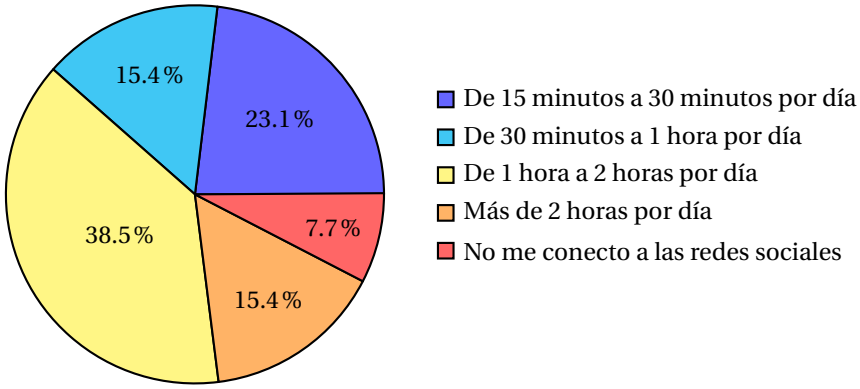

Figura 9: Tiempo diario de utilización de las redes sociales.

La cuarta y última pregunta se trataba de conocer la preocupación por el nivel de privacidad de los usuarios en las redes sociales.

De la muestra, un 53,8 % indicó que les preocupa la privacidad, un 30,8 % afirmó que no les preocupa y un pequeño grupo de los encuestados indicó que es posible que pueda preocuparles.

En general, estos datos reflejan que la privacidad de los datos es un aspecto a considerar en la red social.

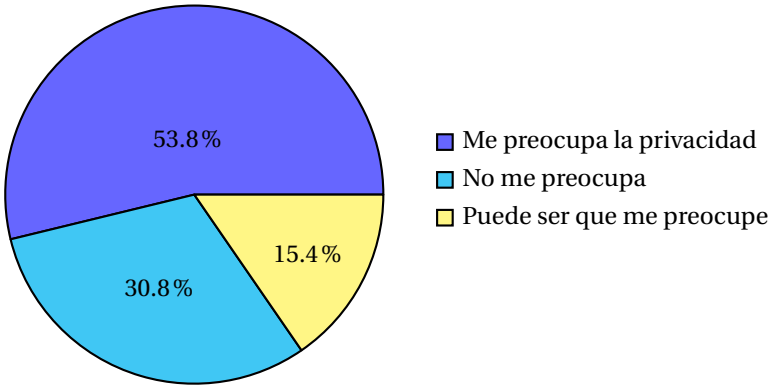

Figura 10: Privacidad de los datos.

4.1.4. Personas

En esta sección se definirán a dos personas ficticias que se utilizarán como guía para el proceso de diseño a partir de la información cualitativa y cuantitativa. El término Persona fue descrito por *Alan Cooper* en el libro *goal-directed design* [3] como una técnica para desarrollar un personaje dependiendo del rol a representar. Con la definición presente, las descripciones de las personas son:

Emilia es una mujer de 25 años que trabaja como profesora en un instituto de Barcelona. Utiliza las redes sociales para comunicarse con sus amigos.
Cuando tiene tiempo libre le gusta subir vídeos y fotografías de platos cocinados por ella a Facebook para compartirlos con sus amigos y amigas.
No le importa la privacidad del contenido multimedia subido a internet siempre que ella no aparezca en la imagen. Cuando ella aparece, le gusta añadir visibilidad a las fotografías. Normalmente utiliza las redes sociales entre dos y tres horas diarias y tiene un conocimiento técnico de informática básico.

Nombre: Emilia.
Edad: 25.
Objetivos: Conocer nuevas amistades.
Preocupaciones: Le preocupa su privacidad.
Personalidad: Extrovertida.
Ocupación: Profesora.
Red social más utilizada: Facebook.
Tiempo de utilización: Más de dos horas diarias.

Luis es un varón de 26 años que trabaja como diseñador de páginas web en una empresa. Normalmente está todo el día consultando su dispositivo móvil buscando nuevas ofertas laborales. Cuando tiene tiempo libre, suele ver las nuevas publicaciones de sus compañeros de trabajo. No le gusta publicar contenido multimedia en las redes sociales. Dispone de un conocimiento técnico en informática elevado ya que recientemente terminó el grado en ingeniería informática.

Nombre: Luis.
Edad: 24.
Objetivos: Buscar otro empleo.
Preocupaciones: Le preocupa su privacidad.
Personalidad: Introvertido.
Red social más utilizada: LinkedIn.
Tiempo de utilización: de una hora a dos horas diarias.

4.1.5. Descripción de los escenarios

En esta sección se describirán los posibles escenarios que los usuarios deberán realizar y que proporcionaran información de valor para la página web a diseñar. Asimismo, cada escenario tendrá un conjunto de tareas representativas agrupadas en escenarios y que los usuarios deberán realizar y permitirá conocer puntos de diseño a tener en cuenta para diseñar las interfaces de usuario además de conocer factores como el número de pasos para realizar las tareas, facilidad de uso y observaciones por parte de otros usuarios.

Cabe destacar que en Facebook el número de tareas que un usuario puede hacer es bastante elevada y se escogieron algunas las más representativas de la aplicación.

De esta forma, se planificaron 4 escenarios que se describen a continuación.

- **Primer escenario.** Enfocado en realizar las acciones de iniciar sesión y registrarse.
- **Segundo escenario.** Consistirá en publicar, eliminar y en indicar "Me gusta" en una publicación.
- **Tercer escenario.** Enfocado en modificar el perfil de usuario.
- **Cuarto escenario.** Englobado en el contexto de la gestión de los álbumes de fotos y imágenes incluyendo la creación, eliminación tanto de los álbumes de fotos como de imágenes.

4.1.6. Conclusiones de los escenarios

Cuando se completaron las diferentes observaciones, se analizaron y las conclusiones obtenidas se muestran a continuación. Cabe destacar que ambos usuarios eran usuarios activos de la aplicación, no obstante, después de realizar los escenarios se obtuvieron datos interesantes de la interacción de los usuarios con la aplicación.

En el primer escenario enfocado al inicio de sesión y registro los dos usuarios se mostraron solventes y no tuvieron problemas en iniciar sesión o en registrarse. Asimismo, ambos usuarios destacaron que el sistema de registro les pareció sencillo y visual al poder acceder rápidamente. Además hubo un participante que destacó que validar los campos a medida que los iba rellenando le pareció una buena forma de detectar errores.

Una objeción de un participante fue que el formulario de inició de sesión del menú superior de la pantalla le resultaba pequeño y desde esa objeción sugirió que en caso de proporcionar al sistema de tal funcionalidad, el formulario tuviese un tamaño más grande.

4.2. Diseño

Cuando se empieza a diseñar una aplicación web es importante diseñar un esbozo conceptual con las funcionalidades y posición de los elementos para mostrar las ideas generales.

Para diseñar los esbozos se utilizaron algunas de las heurísticas de Jakob Nielsen [4].

Antes de empezar a diseñar los diseños conceptuales, se describen brevemente las reglas heurísticas que propuso Jakob Nielsen en el año 1995 tras analizar un número elevado de problemas de usabilidad [4] y llegó a la conclusión después de realizar los análisis que solucionan bastantes problemas de usabilidad. Se agrupan en diez heuristicas:

- **Visibilidad del estado del sistema**. Propone que todo sistema debe proporcionar una retroalimentación de los posibles errores, advertencias o mensajes informativos que sucedan en el sistema. Por ejemplo, en un formulario de inicio de sesión, si se produce un error de validación en los datos introducidos, sería conveniente mostrar un mensaje informativo para indicar al usuario por el tipo de error.
- **Utilizar el lenguaje de los usuarios**. Propone que las palabras utilizadas sean conocidas por el usuario y se eviten tecnicismos o palabras que necesiten de un nivel técnico elevado.
- **Control y libertad para el usuario**. El objetivo de esta regla heurística es proporcionar al usuario de un sistema para abandonar un estado no deseado. Es decir, proporcionar un mecanismo para deshacer la acción realizada.
- **Consistencia y estándares**. Consiste en seguir un estándar para realizar las acciones y mantener una convención de nombres.
- **Prevención de errores**. El sistema debe ser capaz de anticiparse a posibles errores del usuario.
- **Minimizar la carga de la memoria del usuario**. Esta heurística se centra en minimizar la información que un usuario debe recordar para realizar alguna tarea concreta.
- **Flexibilidad y eficiencia de uso**. Ofrecer a usuarios expertos mecanismos para interactuar de una forma optima en el sistema.
- **Diálogos estéticos y diseño minimalista**. Proporcionar información concreta y evitar utilizar muchas unidades de información en la misma página.
- **Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de los errores**. Poder recuperarse de mensajes de errores que deben ser claros y concisos.
- **Ayuda y documentación**. Es posible que el usuario necesite información para realizar alguna tarea de usuario. Esta regla propone ayudar al usuario proporcionando información o una lista detallada con los pasos para realizar una tarea.

4.2.1. Modelos diseñados y análisis heurístico

Esta sección presenta de forma organizada los modelos diseñados. Se analizarán utilizando las heurísticas de Jakob Nielsen y en algunos casos se hará referencia a patrones de diseño. Para cada patrón se especificará donde se utilizó y una descripción del problema que soluciona.

4.2.1.1. Presentación

El primer diseño conceptual corresponde a la presentación 11. La información se organizó en 4 secciones: menú, cabecera, estadísticas y pie de página.

En el menú, concretamente en el logotipo de la aplicación, se aplicó el patrón *HomeLink* [4] para solucionar la necesidad de un usuario de querer volver a visualizar la página de presentación. Es un patrón estándar y adaptado al diseño. Para mantener la consistencia en el diseño y en otros diseños, el menú se mantiene en la parte superior de la pantalla para que, de está forma, el usuario pueda acceder fácilmente.

En la sección estadísticas, la información se organizó en columnas, cada una con un texto breve escritos en un color que contrastara con el fondo y un icono representativo para favorecer la coherencia. Se puso especial importancia en el contraste del color para favorecer la lectura en situaciones de mucha luminosidad o para facilitar la lectura en personas con diversidad funcional visual.

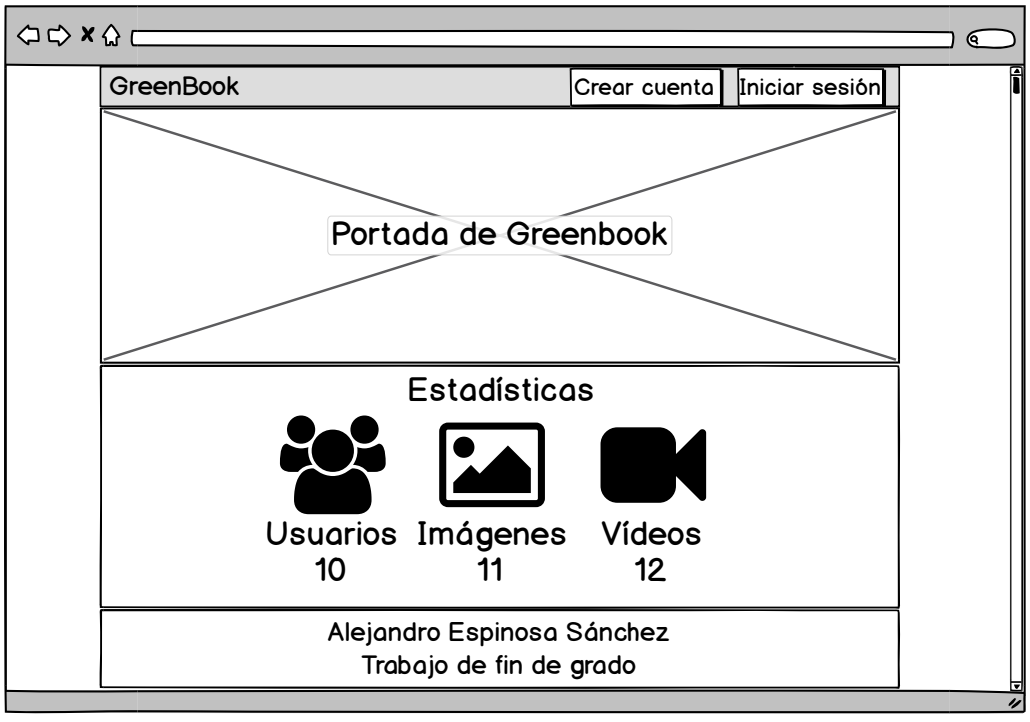

Figura 11: Diseño conceptual de presentación

[4] http://ui-patterns.com/patterns/HomeLink

4.2.1.2. Inicio de sesión

Para diseñar la ventana de iniciar sesión (véase figura 12) se utilizaron varias heurísticas de Nielsen.

En primer lugar, se utilizó la heuristica **visibilidad del estado del sistema** para mantener informado al usuario sobre el estado del sistema y mostrar errores o advertencias. De esta forma, para mostrar el estado del sistema se insertó de forma estratégica un componente informativo para mostrar cualquier mensaje de error. También se utilizó un titulo orientativo para describir en que recurso se encuentra actualmente el usuario y de esta forma informar al usuario del estado actual del sistema.

En segundo lugar, se utilizó la heurística **utilizar el lenguaje de los usuarios** al no utilizar tecnicismos. Por ejemplo, el botón de iniciar sesión si se utilizara un concepto poco familiar como podría ser autentificarse podría causar dificultades al usuario ya que por lo general es un estándar adaptado de otras páginas y adaptado a este diseño. Otro punto importante es la secuencia de acciones para iniciar sesión y la posición se intentó que fuera lo más natural posible manteniendo dos columnas con los campos agrupados.

En tercer lugar, se utilizó la heurística **control y libertad para el usuario** para proporcionar libertar de movimiento para cambiar de recurso y no realizar la acción en la que está actualmente. De esta forma, el usuario podrá seleccionar por una parte iniciar sesión utilizando usuario y contraseña y por otra parte utilizando una cuenta social además de si quiere ir a otro recurso ya sea recuperar la contraseña o crear una cuenta pueda hacerlo pulsando en el botón o hipervínculo correspondiente.

Por último, se utilizó la heurística **diálogos estéticos y diseño minimalista** para evitar la sobrecarga informativa.

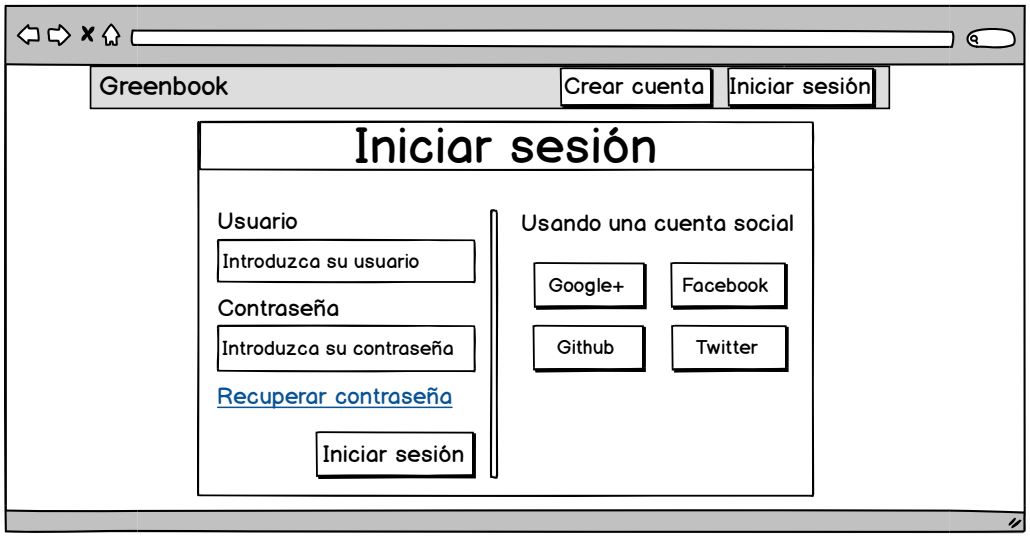

Figura 12: Diseño conceptual de la ventana de inicio de sesión.

Figura 13: Diseño conceptual de un posible error al iniciar de sesión.

4.2.1.3. Creación de una cuenta

El siguiente esbozo a diseñar y analizar corresponde a la creación de una cuenta (véase figura 14).

La primera heurística que se utilizó fue **visibilidad del estado del sistema** para permitir que el usuario pudiera estar informado del estado actual. Así pues, es conveniente seguir un estándar que permita informar al usuario sobre posibles errores que puedan producirse durante alguna acción interna que realice el servidor o usuario y que desencadenen en algún error.

Ese estándar fue mostrar un mensaje de error en el campo donde se hubiera producido y destacar con color rojizo la caja y el texto. Para contrastar la utilización de la heurística se muestra un escenario de error (véase figura 15) al introducir las credenciales. En esa misma linea, seguir ese estándar permitirá adaptarse a usuarios que utilicen lectores de pantalla o dispositivos braille puedan visualizar ese error ya que ese tipo de dispositivos no visualizan los colores correctamente.

La segunda heurística que se utilizó corresponde a **utilizar el lenguaje de los usuarios** para evitar que el formulario tuviera un vocabulario técnico. Además, esta heurística sugiere que los campos de un formulario se presenten de forma ordenada, siguiendo el orden lógico. De esta forma, se presentan de forma ordenada y en columna separando los campos que corresponden al perfil de usuario de los que corresponden a datos de usuario.

La tercera heurística que se utilizó fue **control y libertad del usuario** para permitir que el usuario pudiera cancelar la acción que estaba realizando actualmente y ir a otro tipo de recurso.

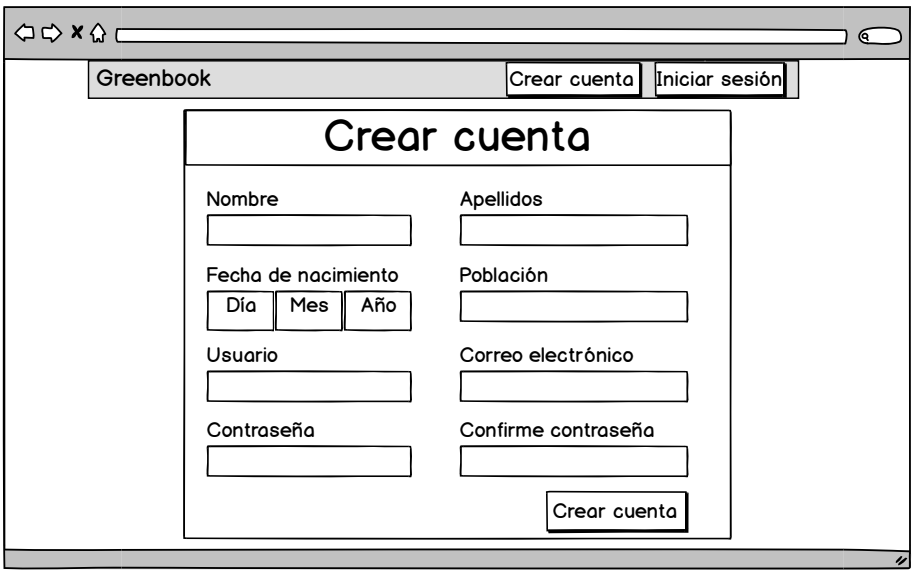

Figura 14: Diseño conceptual de la página de crear cuenta.

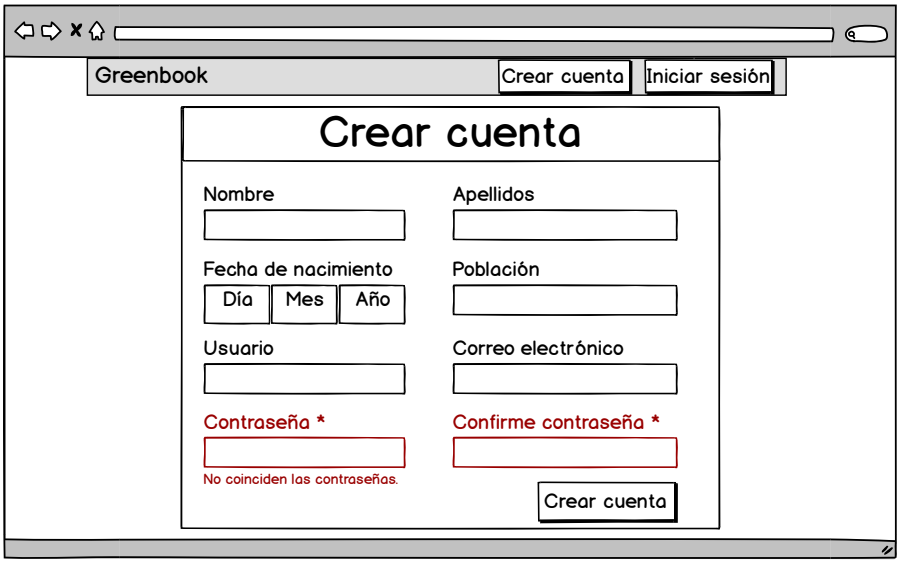

Figura 15: Diseño conceptual de un error en el proceso de crear una cuenta.

4.2.1.4. Muro

El diseño conceptual del muro se organizó en dos bloques:

- **Primer bloque.** Sección de menú, cabecera y botones de navegación.
- **Segundo bloque.** Contenido del muro.

Se utilizaron patrones y estándares de diseño para mantener la consistencia de la aplicación. Se empezarán definiendo los utilizados en el primer bloque y posteriormente los del segundo bloque. En particular, los elementos del primer bloque:

- **Logotipo de la aplicación.** Para resolver el problema que le puede surgir a un usuario que quiera ir a su propio muro, se utiliza el logotipo como hipervínculo. La solución a este problema está descrita en un patrón de navegación *HomeLink* [5].
- **Búsqueda de usuarios.** Las redes sociales permiten buscar a otros usuarios, además de prevenir errores y minimizar la carga de recordar nombres de personas, para ello se utiliza el patrón de búsqueda llamado autocompletar [6].
- **Notificaciones de nuevos eventos y solicitudes de amistad**. Una necesitad de un usuario puede ser recibir notificaciones de ciertos eventos que sucedan en la aplicación. Existe un patrón de diseño llamado *Notification* que resuelve esta necesitad y se adaptó al diseño conceptual [7].
- **Pestañas de navegación.** Un usuario puede tener la necesidad de navegar por diferentes secciones de la aplicación y además saber donde se encuentra.

Una vez descritos los patrones del primer bloque, se describirán los del segundo bloque.

- **Nivel de privacidad.** Cuando el usuario tiene la necesidad de efectuar esta acción puede tener dificultades si se utilizan componentes no estándares u otro mecanismo no conocido por el usuario. Para solucionar esta necesidad, se utiliza el patrón *ShortcutDropdown* [8]. Este componente es estándar y proporciona compatibilidad tanto a dispositivos móviles como a navegadores antiguos.
- **Valor por defecto en la privacidad.** Una necesidad de un usuario podría ser evitar tener que seleccionar la privacidad para cada mensaje. La solución a esta necesidad de encuentra en el patrón de diseño *Good Defaults* [9] que se utiliza cuando el valor por defecto puede ser, en la mayoría de casos, el que el usuario quiere seleccionar de la lista desplegable.
- **Reaccionar a mensajes.** Puede suceder que un usuario quiera reaccionar a un mensaje. Para resolver esa necesidad, se adaptó el patrón social *Reaction* [10] que simplifica la acción haciéndola binaria y tiene la ventaja de no depender de una escala de calificación. De esta forma, los usuarios pueden interpretarlas fácilmente al no precisar de una puntuación.
- **Ventanas laterales** Puede producirse que el usuario quiera acceder a pequeñas porciones de información y que estás puedan variar en su tamaño. Para solucionar esa necesidad, se utiliza el patrón cards [11] que permite agrupar colecciones de elementos homogéneos.

[5] http://ui-patterns.com/patterns/HomeLink
[6] http://ui-patterns.com/patterns/Autocomplete
[7] http://ui-patterns.com/patterns/notifications/examples/18195
[8] http://ui-patterns.com/patterns/ShortcutDropdown
[9] http://ui-patterns.com/patterns/GoodDefaults
[10] http://ui-patterns.com/patterns/reaction
[11] http://ui-patterns.com/patterns/cards

Se utilizó la heurística **diálogos estéticos y diseño minimalista** para proporcionar información concisa y breve. En ambos bloques, se utilizaron márgenes para separar y organizar la información.

Otra heurística corresponde a **prevención de errores** para anticiparse a errores que pudiera hacer un usuario. Por ejemplo en el botón de eliminar mensaje, se podría dar el escenario que intencionadamente se pulsara y eliminara el mensaje sin pedir confirmación. De acuerdo a está heurística, se utiliza una ventana modal

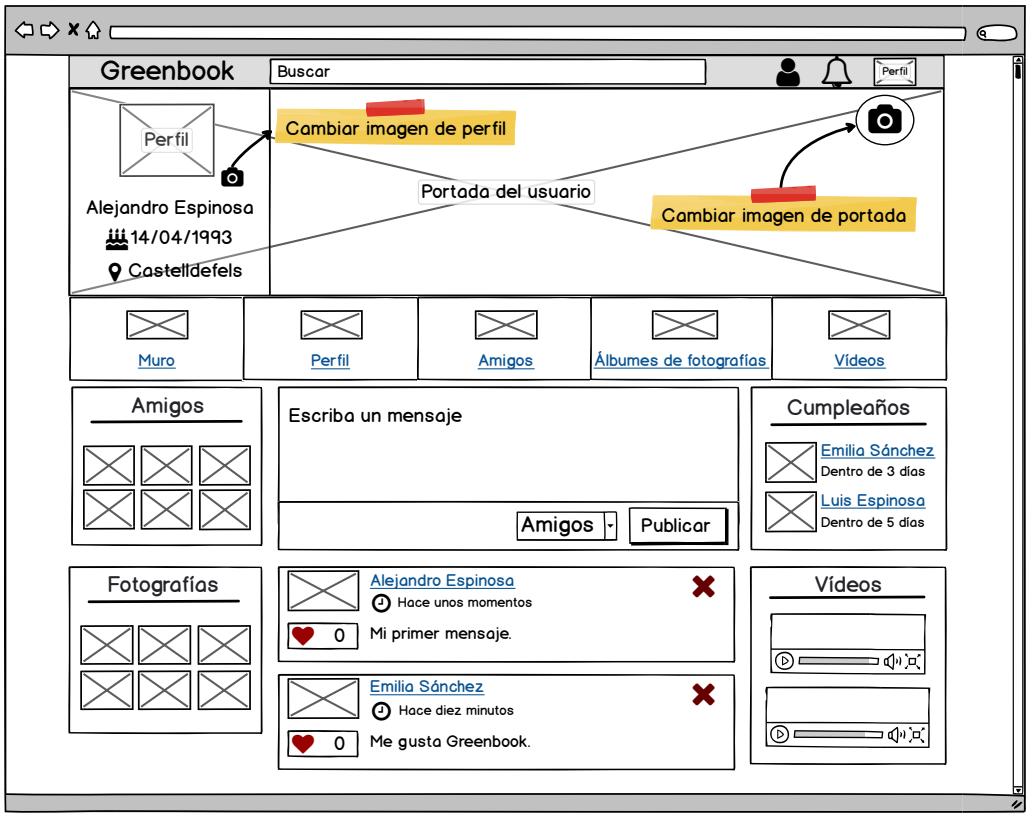

Figura 16: Diseño conceptual del muro

4.2.2. Prototipo

Una vez diseñados los diseños conceptuales se procedió a diseñar un prototipo funcional de alta fidelidad a partir de los esbozos.

Para diseñar las interfaces se utilizó un producto de Adobe llamado Experience Design que es una herramienta que está disponible en los sistemas operativos de MacOS y Windows y está enfocada al prototipado de aplicaciones. Durante la realización del prototipo se utilizó la versión de prueba pero la aplicación es comercial.

Finalmente, se exportaron a la aplicación InVision [12] y se añadió interacción entre ventanas.

Cuando se diseñó el prototipo, se utilizaron diferentes iconos vectoriales de terceros que enumeraré a continuación.

- **Essential Set.** Conjunto de iconos disponibles en diversos estilos: planos, lineales y finalmente rellenos. De este conjunto de iconos utilicé el estilo lineal que contiene únicamente el trazo del icono y el motivo fue para dar un diseño limpio y minimalista.

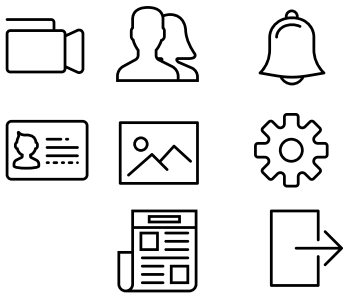

Figura 17: Iconos utilizados del conjunto de iconos Essential Set.

- **Interaction Set.** Consta de un paquete de iconos libres enfocados para la interacción entre personas.
- **Font Awesome.** Para complementar los iconos se utilizó este conjunto de iconos que está enfocado a las páginas web y ofrece un conjunto de iconos bastante detallados.

[12] https://invis.io/9QDH8UYN5

4.3. Test con usuarios

Una vez evaluados heurísticamente en términos de usabilidad, es conveniente probar los diseños con usuarios, con el propósito de detectar posibles carencias en los diseños. Los test con usuarios consistirán en realizar un conjunto de tareas representativas en la aplicación y ver como se comportan los usuarios mientras se graba la pantalla para registrar las acciones del usuario con la interfaz. De esta forma, se podrán detectar problemas y en donde los usuarios encuentran dificultades o en acciones que no sepan realizar de manera intuitiva. Cabe destacar que en este tipo de test se indican objetivos y no como hacer las tareas ya que como se ha explicado anteriormente, la finalidad es encontrar carencias en los diseños.

Antes de realizar los test con usuarios, el primer paso fue diseñar un conjunto de tareas. La organización de las tareas se agrupó de forma estructurada siguiendo un orden ordenado y lógico, en otras palabras, no proponer empezar realizando una tarea de iniciar sesión sin haber creado previamente una cuenta en el sistema.

El segundo paso fue escribir un documento con el fin de informar al usuario que se tomarían datos del usuario a partir de la interacción con el sistema utilizando un programa especifico. Se puede encontrar en el anexo en la sección A.1.

Como último paso, se buscó el programa necesario para grabar la pantalla. Optando por utilizar *JING* que es una herramienta gratuita. Para grabar la voz, se optó por utilizar las aplicaciones nativas del sistema operativo.

Reanudando las tareas, se organizaron en diez escenarios. Concretamente, son:

Primer escenario:

Un amigo te ha hablado de una nueva red social, Greenbook, que es muy interesante. Has decidido apuntarte e iniciar sesión en ella.

Segundo escenario:

Quieres compartir con tus amigos una publicación y para ello la publicas en tu muro. Además te gustaría compartir con todos los usuarios las experiencias del viaje más reciente que hayas realizado. Publica esos mensajes.

Visualizando la publicación que pueden ver los amigos, te has dado cuenta que te gusta y quieres indicarlo.

El mensaje que pueden ver todos los usuarios, no te convence y quieres eliminarlo.

Tercer escenario:

Visualizando tu perfil de usuario, quieres cambiar el nombre, apellidos y fecha de nacimiento. También quieres cambiar la imagen de portada y perfil. Realiza esas modificaciones.

Cuarto escenario:

Este verano saliste de viaje y quieres que tus amigos puedan ver algunas imágenes de ese viaje que hiciste. También quieres que todos los usuarios puedan ver otras imágenes. Crea los álbumes con las imágenes. Una vez subidos, te das cuenta que no quieres que todos los usuarios vean el álbum público y decides eliminarlo.

Quinto escenario:

Hace unos días grabaste dos vídeos y quieres subirlos. Mientras estabas añadiendo el vídeo, te fijas que uno de los dos vídeos no era el que tenías que subir y para solucionar ese problema decides eliminarlo antes de subirlo. Una vez subido, decides limpiar los vídeos subidos y posteriormente visualizar el vídeo.

Sexto escenario:

Envías una solicitud de amistad a la persona que te recomendó la aplicación, eres rechazado pero luego acabas recibiendo una nueva. Acéptala.

Séptimo escenario:

Quieres publicar un mensaje en el muro de la persona que te recomendó la aplicación. Visualizando los mensajes que tiene el usuario, te fijas en un mensaje que publicó y que te parece interesante y lo indicas. Como eres curioso/a quieres saber a qué personas le gusta el mensaje.

Octavo escenario:

Tienes curiosidad por ver los amigos de esa persona y en comprobar si esa persona es amigo de un compañero de la universidad (Pepe) pero no te acuerdas exactamente de su apellido y decides encontrarlo sabiendo que su localidad es (Castelldefels).

Noveno escenario:

Hace unos días quedasteis para ir de excursión y hicisteis algunas imágenes y vídeos. La persona que te recomendó la aplicación te dijo que subió unas imágenes y videos y quieres verlas.

Décimo escenario:

Después de utilizar la red social decides que no puedes gestionar todas las redes sociales en las que estás inscrito y decides eliminar la cuenta con Greenbook.

4.4. Resultados de los tests con usuarios

Una vez descritos los escenarios, el siguiente punto consistió en realizar los tests con usuarios.

Se hicieron cinco tests de usuarios en la Universidad de Barcelona en un entorno tranquilo y se obtuvieron aspectos a mejorar en los diseños y una visión detallada de la interacción con el sistema.

- El proceso de crear una cuenta tiene demasiada información. Podría ser interesante organizar los campos en dos secciones, por ejemplo, una sección para introducir los campos de usuario y otra para el perfil.
- Cuando creo una cuenta me gustaría tener alguna validación del formulario para saber si los campos son válidos o si por ejemplo el correo electrónico o usuario están en uso evitando tener que enviar el formulario.
- Rediseñar el proceso de subir una imagen o vídeo ya que son poco utiles por permitir subir un elemento cada vez. Sería interesante permitir arrastrar un archivo y que se añadiera por ejemplo a una lista de archivos para subir.
- El muro de usuario no permite distinguir a que usuario estoy visitando. Podría ser una buena idea indicarlo.
- Los mensajes indican a que personas le gusta el mensaje pero no que persona concretamente. Sería interesante dar esa funcionalidad.
- Cuando llega una notificación sería interesante, por una parte indicarlo con un sonido acústico y, por otra parte, aplicar algún efecto para poner énfasis en la nueva notificación para que, de esta forma, un usuario pueda ser notificado.

5. Análisis de la aplicación

5.1. Casos de uso

Para la descripción detallada de la funcionalidad se han usado historias de usuario, pero por completitud y para mostrar la información desde diferentes perspectivas se incluye una visión general sobre los casos de uso.

El primer diagrama de caso de uso que se muestra en la figura 18 corresponde al actor visitante que tendrá un acceso limitado en el sistema y podrá iniciar sesión, registrarse o cambiar la contraseña. Además podrá optar por iniciar, registrarse utilizando una cuenta social o en su defecto introduciendo los datos manualmente.

Puede producirse que no se obtengan los datos necesarios para crear una cuenta. En ese caso, se activará el caso de uso introducir datos de perfil para pedir esos datos.

Cuando el usuario intente cambiar la contraseña, el sistema enviará un correo con un enlace para cambiar la contraseña y por último una vez modificada la contraseña el usuario recibirá una notificación indicando que la ha cambiado.

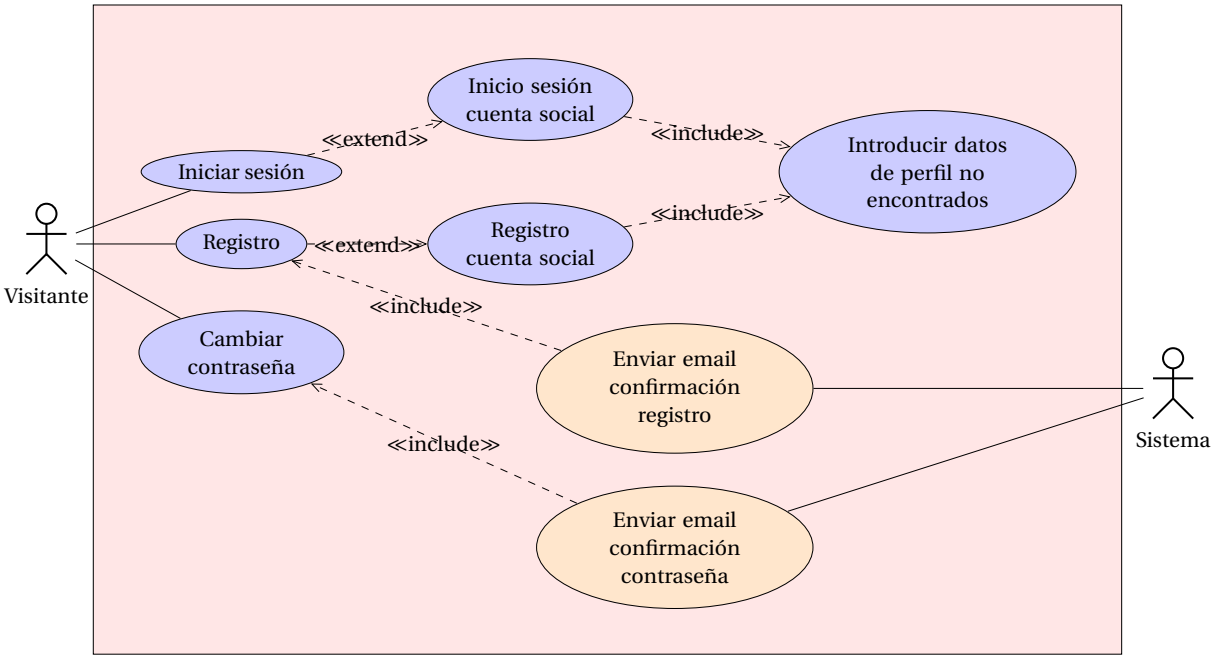

Figura 18: Casos de uso del actor que no ha iniciado sesión.

Una vez vistos los casos de uso de un usuario no autentificado en el sistema, se presentarán los diagramas que corresponden a un usuario autentificado. Los diagramas de casos de uso se dividirán en varias figuras para poder ser preciso en las observaciones.

El segundo diagrama de casos de uso 19 detalla las acciones que podrá hacer un usuario con los mensajes. En particular, en el caso de uso añadir mensaje, se ha optado por especificar un *include* ya que el usuario debe seleccionar la privacidad aunque por defecto se establezca a una concreta.

Cuando se activen los casos de uso añadir, eliminar y los relacionados con los "me gusta", el sistema notificará a otros usuarios en tiempo real de los cambios.

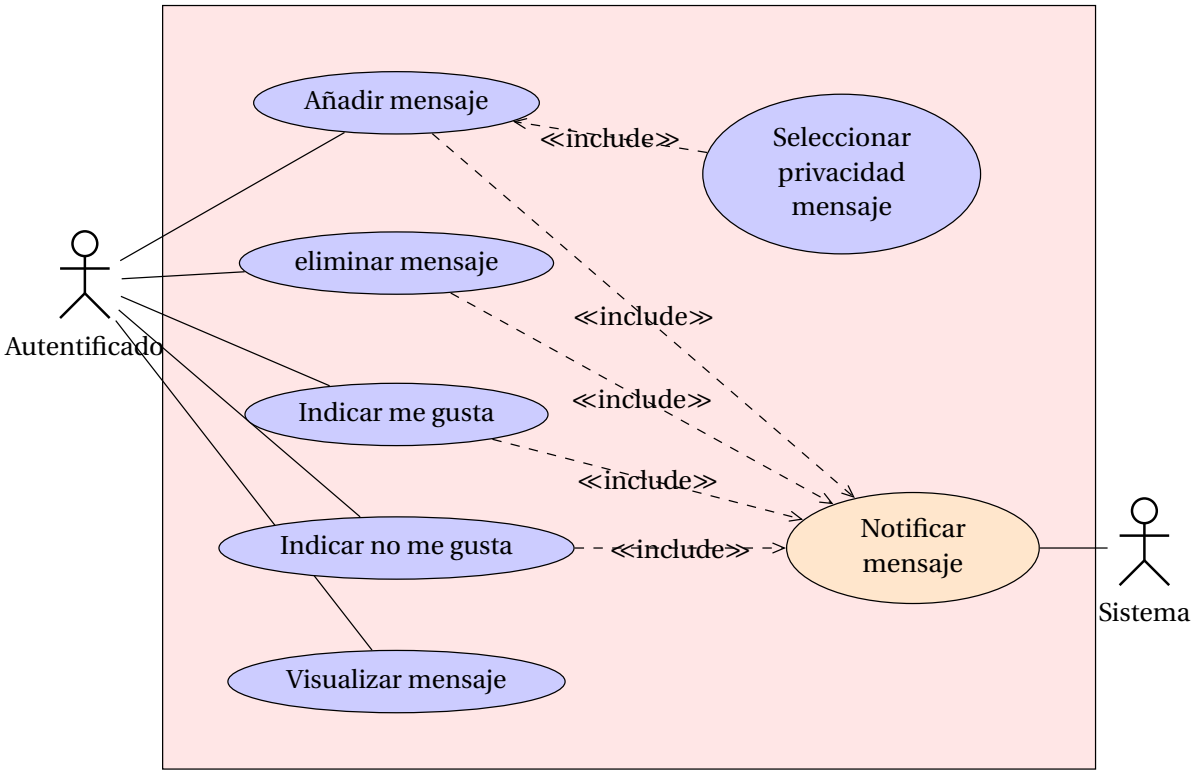

Figura 19: Casos de uso del actor autentificado en los mensajes.

El tercer diagrama de casos de uso 20 corresponde a las posibles acciones que podrá realizar un usuario cuando interaccione con el perfil de usuario.

Se destaca que cuando el usuario modifique el perfil podrá optar por modificar la imagen de portada o perfil. A partir de la modificación de la imagen, el usuario deberá seleccionar una imagen de miniatura. Se ha optado por hacer una inclusión ya que siempre que realice el caso de uso, deberá seleccionar una imagen de miniatura para indicar una zona de interés en la imagen.

Una vez se modifique el perfil, el sistema notificará a los otros usuarios de la modificación y adjuntará los nuevos datos de perfil para que, de esta forma, los usuarios puedan actualizar los cambios en tiempo real.

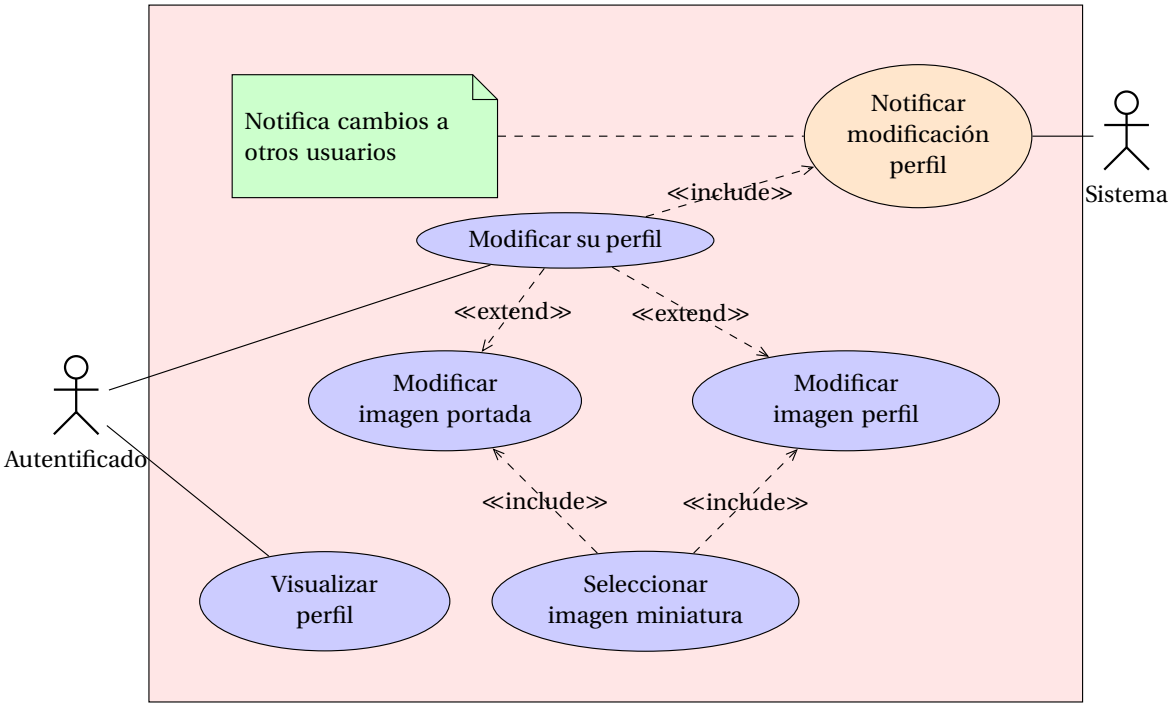

Figura 20: Casos de uso del actor autentificado en el perfil de usuario.

Los casos de uso asociados a los álbumes e imágenes se presentan en la figura 21.

Una observación a destacar se localiza en el caso de uso de crear un álbum de fotos, en el cual se incluye el caso de uso seleccionar una privacidad para que cuando realice esa acción deba seleccionar una privacidad.

Otra observación a destacar se encuentra en el caso de uso añadir imagen álbum, en el que se utiliza el caso de uso seleccionar una imagen de miniatura para seleccionar una zona de interés de la imagen y evitar tener que descargarse la imagen completa.

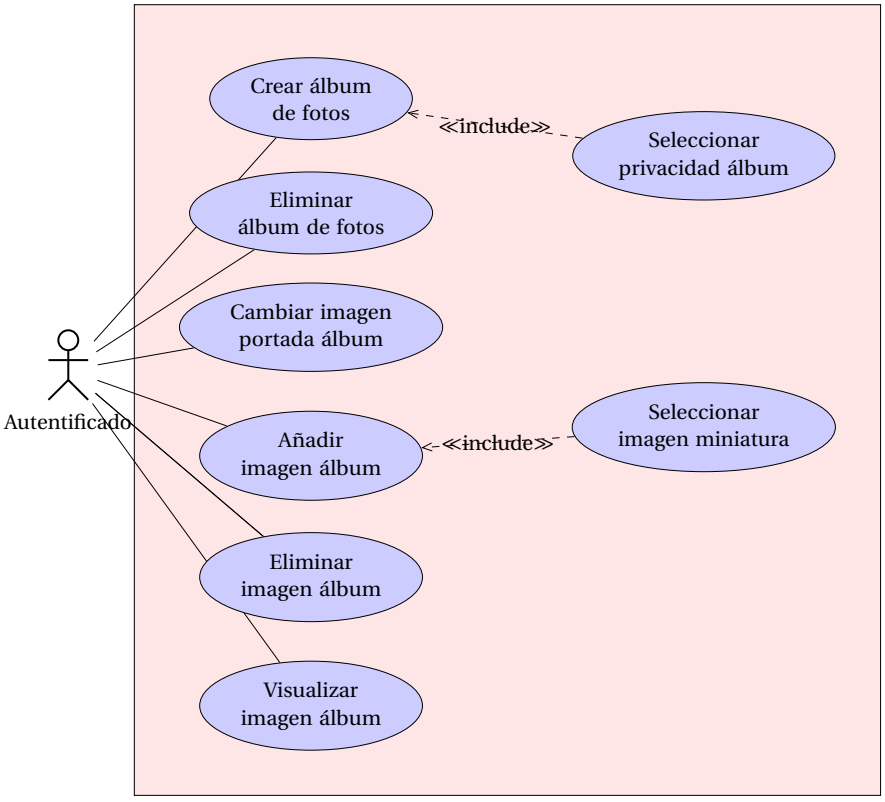

Figura 21: Casos de uso del actor autentificado en los álbumes de fotos e imágenes.

El diagrama de casos de uso 22 definen las acciones que podrá hacer el sistema y el usuario con las solicitudes de amistad y bloqueo entre usuarios. En particular, se observa que si se activan los casos de uso de añadir, aceptar o cancelar una solicitud de amistad, el sistema notificará al otro usuario en tiempo real de las acciones realizadas.

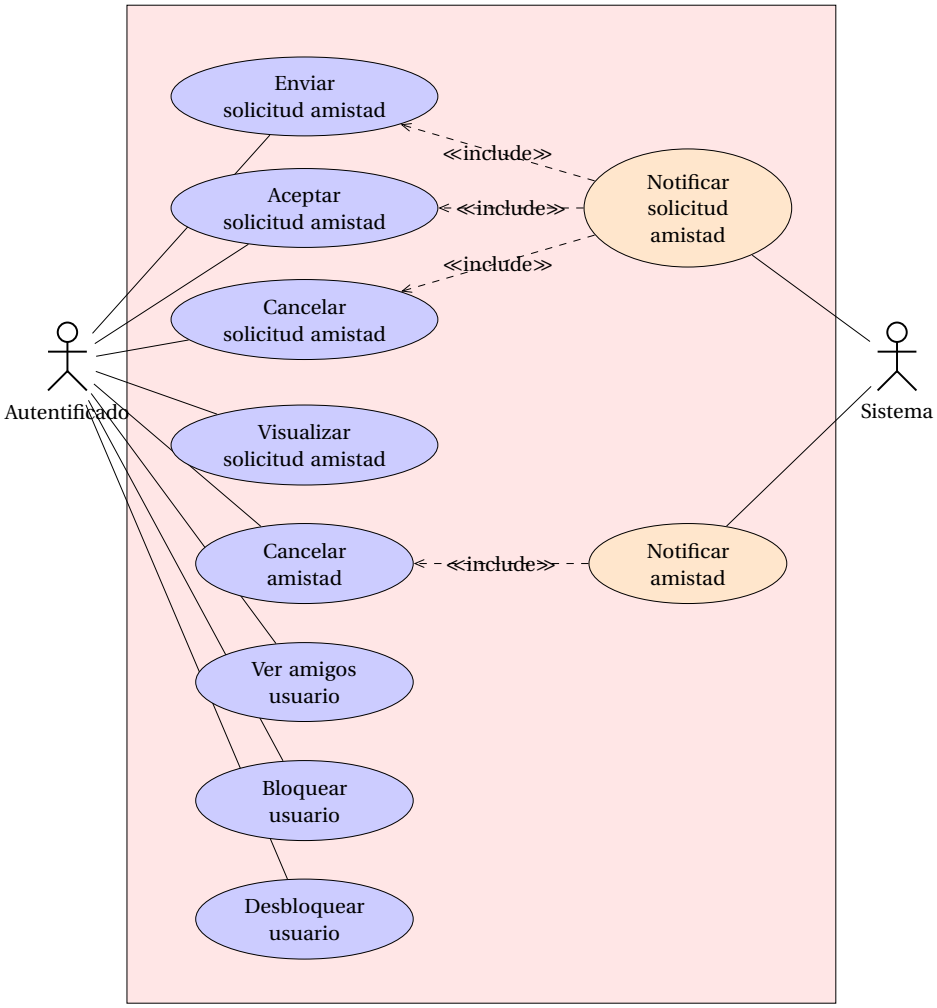

Figura 22: Casos de uso del actor autentificado en la solicitud de amistad y bloqueo.

El caso de uso 23 muestra las acciones que podrá realizar un usuario con las notificaciones que le envíe el sistema en tiempo real de otros usuarios. En particular, se definió el caso de uso *Visualizar notificación* con un *include* a *Notificar notificación* para que el usuario siempre que desee ver las notificacione

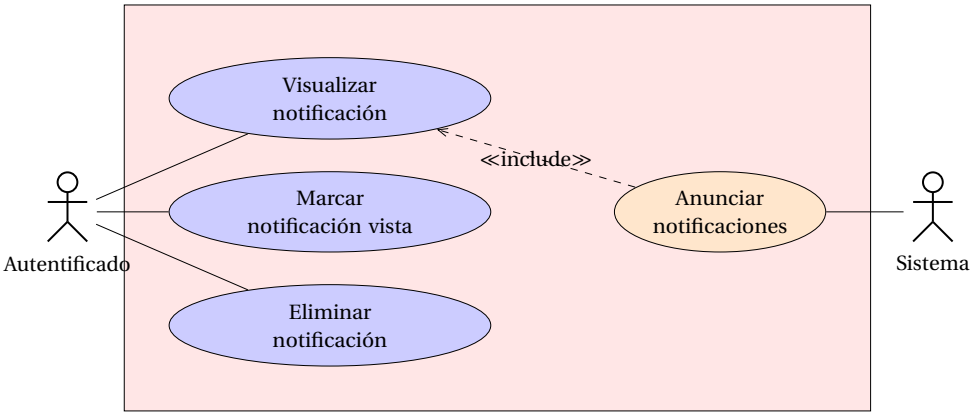

Figura 23: Casos de uso del actor autentificado en las notificaciones generales.

Los casos de uso que podrá realizar un usuario autentificado se muestran en la figura 24. Concretamente el usuario podrá subir un vídeo y eliminarlo. Además, podrá visualizar vídeos que haya subido o de otro usuario.

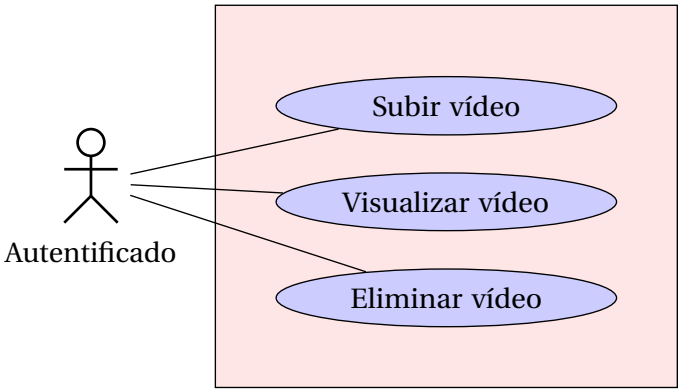

Figura 24: Casos de uso del actor autentificado en el vídeo.

5.2. Base de datos

Una vez presentados los casos de uso se tiene una visión conceptual de las funcionalidades que la aplicación tendrá. El próximo paso será modelar la base de datos diseñando un diagrama entidad relación, que se puede visualizar en la figura 25 de la siguiente página y en esta se darán algunas observaciones sobre las relaciones entre las entidades.

En particular, si se empieza por la entidad Usuario se tienen las relaciones:

- **Relación Usuario - Vídeo.** Representa que un usuario podrá almacenar vídeos.
- **Relación Usuario - Perfil.** Representa que un usuario tendrá como mínimo un perfil asociado y un perfil estará relacionado con un usuario en particular.
- **Relación Usuario - Mensaje.** Representa que un usuario puede publicar mensajes y por otra parte los mensajes son publicados por un usuario.
- **Relación reflexiva, Usuario amigo de otro usuario.** La relación establece que un usuario será amigo de muchos usuarios y viceversa, otro usuario puede ser amigos de esos amigos.
- **Relación reflexiva, Usuario bloqueado**. La relación indica que los usuarios podrán tener muchos usuarios bloqueados y un usuario podrá estar bloqueado por más de un usuario.
- **Relación reflexiva, Usuario notifica**. Con esta relación se indica que cuando se produzcan ciertos eventos, derivados de alguna acción en el sistema, los usuarios podrán ser notificados indicando dos claves foráneas, la primera para el usuario emisor y el segundo para el usuario destino.
- **Relación reflexiva, Usuario solicita amistad**. Representa que un usuario podrá tener solicitudes de amistad pendientes de varios usuarios.

De la entidad Perfil:

- **Relación Perfil - Imagen** Representa que un perfil podrá tener como mínimo ninguna imagen y como máximo dos imágenes: una de portada y otra de perfil. Esas imágenes recíprocamente pertenecerán a un usuario.

Finalmente, de la entidad Álbum:

- **Relación Álbum - Imagen**. Relaciona la entidad álbum con imagen estableciendo que un álbum podrá tener alguna imagen o muchas imágenes y recíprocamente una imagen estará almacenada en un álbum concreto.

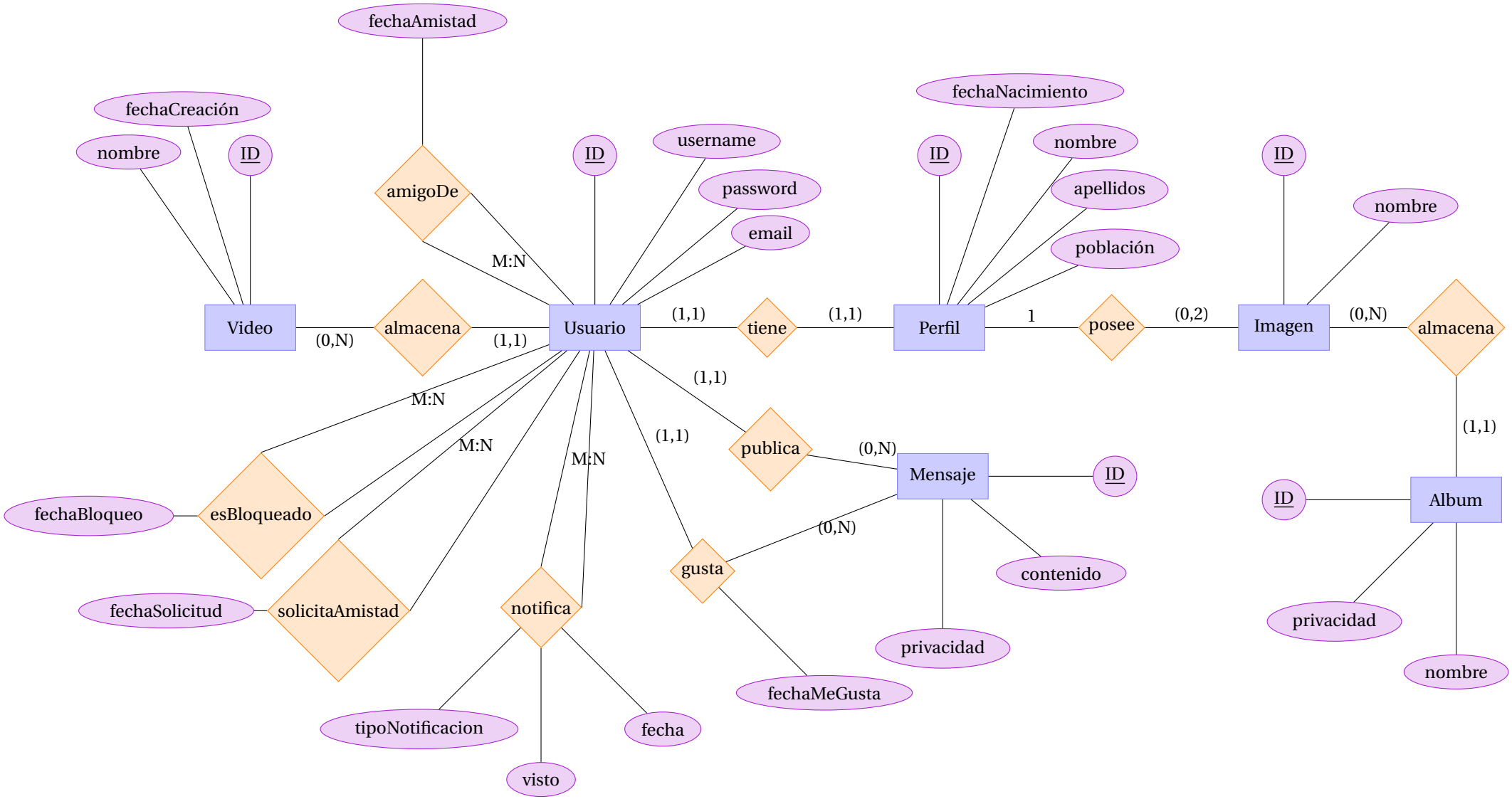

Figura 25: Modelo entidad relación.

5.3. Arquitectura de la aplicación

Para conocer la arquitectura de la aplicación se ilustra en la figura 26 y en ella se pueden los observar los componentes que utiliza para funcionar.

Toda petición la inicia el cliente enviando una petición HTTP o WebSockets al servidor Engine X (Nginx) que es un servidor web muy ligero y de alto rendimiento que se encarga de proporcionar los iconos, imágenes y vídeos de la aplicación. También permite actuar como un proxy inverso para enrutar las peticiones a Django. Si se desea conocer más información sobre la configuración del servidor, se adjunta el fichero de configuración en el anexo (véase A.2).

Por defecto Django no soporta peticiones de WebSockets y fue necesario añadir una capa que permitiera gestionarla. La gestión la controla Daphne que es un servidor que permite negociar la petición que proviene del cliente a través de Nginx.

La capa Channel Layer permite comunicarse con otros clientes conectados al servidor. Como se comentó en el anterior parágrafo, el hecho de no soportar por defecto peticiones WebSockets hizo que fuera necesario añadir una capa que permitiera comunicar consumidores y productores. En este contexto, los productores producen un mensaje y lo guardan en una cola en la base de datos Redis para que los consumidores puedan obtener el mensaje y puedan procesarlo.

La aplicación utiliza la base de datos MySQL para persistir e obtener información de las tablas.

Por otra parte, se utiliza la base de datos Redis para guardar en memoria páginas web comprimidas con la finalidad de no tener que generar la misma plantilla estática para cada petición y obtener un mayor rendimiento. Por ejemplo en las páginas de inicio de sesión y registro, el contenido no varía de un usuario a otro y son candidatos a ser guardados en cache.

La arquitectura es escalable y por lo tanto si fuera necesario añadir nuevos nodos para soportar más peticiones concurrentes se podría realizar. Concretando qué significa añadir un nuevo nodo, sería en particular añadir una entrada en el fichero de configuración de Nginx indicando el socket de Daphne y activar el complemento *load-balanced* para poder distribuir el trafico entre varios servidores.

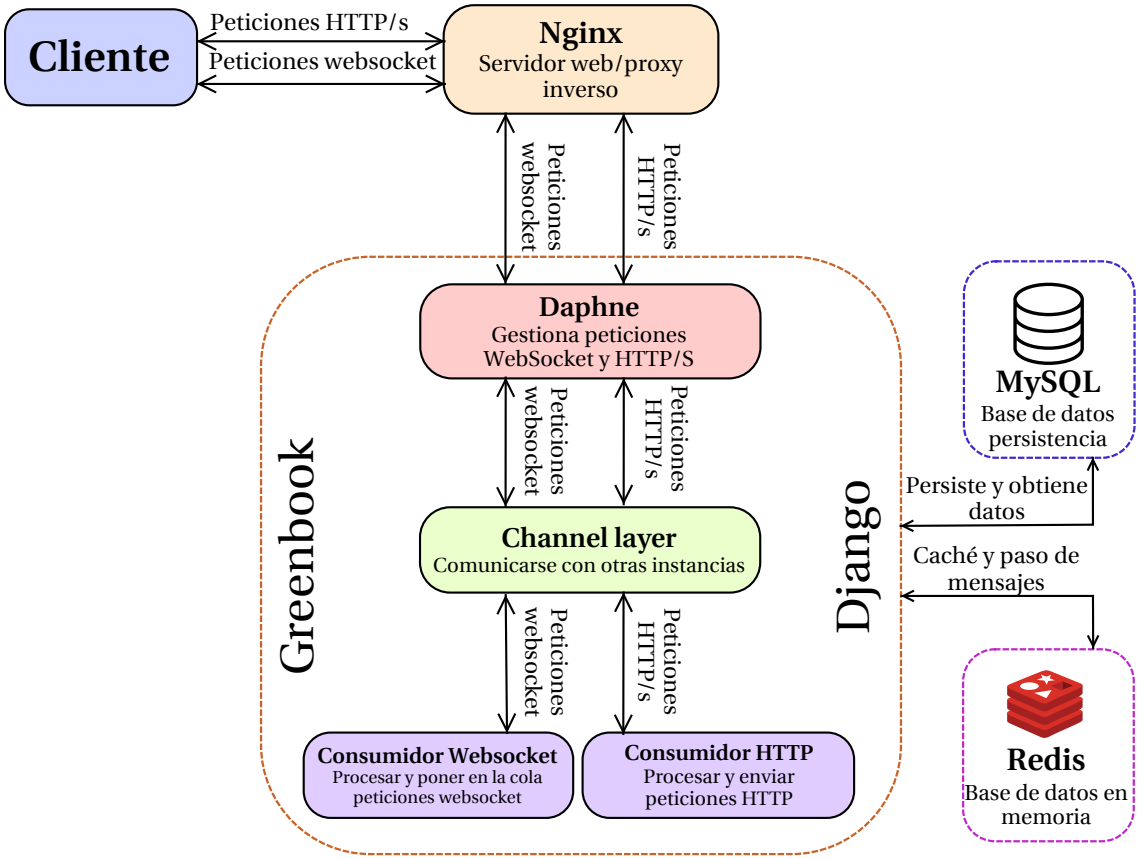

Figura 26: Arquitectura de la red social

6. Tecnologías utilizadas

El propósito de este capítulo será describir las tecnologías que se utilizaron para desarrollar el proyecto. Se clasificaron en dos grupos; en primer lugar tecnologías del lado del servidor y en segundo lugar tecnologías del lado del cliente.

6.1. Tecnologías del lado del servidor

En esta subsección se presentarán las tecnologías no accesibles por el cliente pero utilizadas en el lado del servidor para implementar la lógica interna del servidor.

6.1.1. Django

Para desarrollar la parte del lado del servidor se utilizó Django por la escalabilidad, rendimiento y flexibilidad que proporciona a los desarrolladores de software. Es un proyecto de código abierto y libre que dispone de una buena documentación y por tanto fue un punto a favor de utilizarlo como servidor web. Además otro punto a favor fue que existen extensiones de terceros que se pueden agregar y pueden simplificar el desarrollo de la aplicación web.

Entrando más en detalle, Django es un framework de aplicaciones web escrito en Python. Proporciona un conjunto de herramientas para desarrollar aplicaciones web y entre las cuales se encuentran: un sistema Object Relational Mapping (ORM) para guardar información en la base de datos, un sistema de plantillas para renderizar páginas además de un sistema llamado URL dispatcher para resolver peticiones de los usuarios.

La arquitectura de Django sigue el patrón de diseño MTV (Model Template View) muy similar al patrón MVC que tiene el acrónimo de (Model View Controller).

- **Model**. Es la capa de acceso a la base de datos y contiene toda la información de las tablas. En Django, el modelo es un sistema ORM que tiene la finalidad de traducir los modelos definidos en clases de Django a tablas de bases de datos. De está forma, al estar en clases, se pueden aprovechar las ventajas de la programación orientada a objetos para acceder a los atributos.
- **Template**. Consiste en un sistema de plantilla que normalmente suele ser código HTML que el usuario visualizará en su navegador. La diferencia entre utilizar o no utilizar un sistema de plantilla es que se pueden añadir variables y bucles útiles para estructurar la aplicación y simplificar ciertas operaciones.
- **View**. Es la capa que contiene la lógica para acceder al modelo y enviar los datos a la plantilla adecuada. Cada vista es un controlador, que controla un recurso especifico de la aplicación e introducida en el sistema de rutas, asigna un nombre para poder ejecutarse cada vez que un usuario realice una petición. Cada vista tiene un argumento llamado *request* que contiene el contexto de la petición.

6.1.2. Django Channels

Para desarrollar la parte de interacción en tiempo real se utilizó este módulo desarrollado para Django y que tiene como objetivo extender el framework para añadir una capa que permita gestionar peticiones de WebSockets, conexiones permanentes (HTTP2) y tareas asíncronas.

Django Channels cambia el paradigma de una aplicación de Django a un modelo basado en eventos e introduce cuatro conceptos importantes para entender su funcionamiento. Estos son:

- **Channel** (Canal). Proporciona un sistema de tareas. Los productores introducen los mensajes en el Channel y los entrega en el consumidor que está escuchando el canal y en la arquitectura se utiliza Redis para soportar la capa Channel layer.
- **Consumer** (Consumidor). Permiten consumir eventos de la cola de tareas.
- **Productor** (Productor). Los productores son los encargados de introducir los mensajes en la cola de tareas (channel).
- **Group** (Grupo de difusión). Es una clase que surge para solucionar la carencia de no poder entregar un mismo mensaje a múltiples consumidores. Básicamente, los consumidores se subscriben a un grupo y todos los subscriptores que estén subscritos lo recibirán.

Para mostrar su funcionamiento aplicado en el proyecto, primeramente se definirá como se enlaza Redis con el módulo, seguidamente como se define una ruta, un ejemplo de utilización y finalmente como se conecta un cliente.

La configuración de Redis se puede observar en el código 1. Concretamente en la línea 6, se indica donde se encuentra y el puerto de escucha. Por otra parte, para poder enrutar correctamente, es necesario indicar donde se encuentran los consumidores (línea 8).

```
1  redis_host = os.environ.get('REDIS_HOST', 'localhost')
2  CHANNEL_LAYERS = {
3          "default": {
4          "BACKEND": "asgi_redis.RedisChannelLayer",
5                  "CONFIG": {
6                          "hosts": [(redis_host,6379)],
7                  },
8          'ROUTING': 'ProyectoTFG.routing.rutas_websocket',
9          },
10 }
```

Código 1: Configuración de Redis para el intercambio de tareas en Django Channels.

Para mostrar el funcionamiento, se ha optado por mostrar el consumidor del Muro. En particular, es una clase que extiende de *JsonWebsocketConsumer* e implementa ciertos métodos que se pueden sobrescribir. En particular, en el código 2) se muestra el método *connection_groups* y que retorna los posibles grupos a los que un cliente se conectará cuando intente establecer una conexión.

En el código 2 se muestra ejemplo un ejemplo de como se agrega al usuario autentificado al muro privado.

Del ejemplo de código 2, se puede observar como en la línea 10, se utiliza el canal para entregar un mensaje de error y finalizar rechazar la conexión por no existir el muro en el sistema.

Haciendo un inciso, el lector se podría preguntar cómo se obtienen los datos de sesión del usuario si se utiliza WebSockets en vez de HTTP. Para insertar los datos de sesión, es necesario consultar la línea 2 del código 2 que indica que los datos de sesión se incluirán en la cabecera de la petición.

```
1  class MuroConsumer(JsonWebsocketConsumer):
2      http_user_and_session = True
3      def connection_groups(self, **kwargs):
4
5          grupos_conectar = []
6
7          try:
8              muro_usuario = Muro.objects.get(usuario=kwargs.get('usuario', None))
9          except Muro.DoesNotExist:
10             json_error=json.dumps({"error": "Usuario no encontrado"}
11             self.message.reply_channel.send({"text": json_error), "close": True, })
12             return
13
14         grupos_conectar.append("MuroPrivado" + str(muro_usuario.usuario.id))
15         return grupos_conectar
```

Código 2: Definición del consumidor del Muro.

La clase que implementa la ruta vista anteriormente todavía no puede ser utilizada ya que previamente es necesario hacer la correspondencia de la ruta y con el consumidor del Muro.

```
1  rutas_websocket = [
2          route_class(MuroConsumer, path=r'^/muro/(?P<usuario>[^/]+)/$'),
3  ]
```

Código 3: Definición de las rutas del consumidor del Muro.

Una vez establecida la ruta, los usuarios pueden conectarse al canal y escuchar eventos. Para conectarse es necesario indicar el muro del otro usuario.

Los extractos de código vistos anteriormente no son directamente accesibles por el usuario ya que están implementados en la lógica de la aplicación. Para proporcionar un punto de acceso se proporciona una ruta de conexión. En particular, la definición se encuentra en el código 3 y en el que visualiza la ruta y el argumento necesario para poder establecer una conexión desde el lado del cliente. Cuando el usuario se conecta, se mantiene escuchando peticiones del servidor y cuando recibe una petición ejecuta la función *listen* (líneas 8-11) para procesar el mensaje.

```
1  var ws_scheme = window.location.protocol == "https:" ? "wss" : "ws";
2  var usuario= window['variablesAPI'].otro_usuario;
3  var ws_path = ws_scheme + "://" + window.location.host + "/muro/" +  usuario + "/";
4  console.log("Me estoy conectando --> " + ws_path);
5  var ws = new channels.WebSocketBridge();
6  ws.connect(ws_path);
7
8  ws.listen(function (action, stream) {
9          console.log(action, stream);
10         console.log('Nuevo mensaje');
11 });
```

Código 4: Definición de las rutas del consumidor del Muro.

6.1.3. Django Rest Framework

Proporcionar una interfaz de funciones tanto para el funcionamiento interno de la aplicación como para otras personas que deseen utilizarlas, es una buena manera de reutilizar funciones definidas en el lado del servidor. Con el fin de lograr esa funcionalidad se utilizó este módulo.

El módulo define tres tipos de componentes:

- **ViewSet**. Es una clase contenedora de API que define los métodos sobre un modelo concreto. Integra un sistema de permisos aplicables en métodos para restringir el acceso a ciertas funciones que no cumplan ciertas características. Como por ejemplo, que hayan iniciado sesión.
- **Routers**. Para hacer las correspondencia de los métodos implementados en la clase contenedora, se utiliza este componente. Básicamente es una extensión del sistema URL dispatcher de Django que define un espacio de nombres.
- **Serializers**. Transforma instancias de un modelo de Django a en formato JavaScript Object Notation (JSON) indicando que atributos se mostrarán.

Para mostrar su utilización en el proyecto, se muestra para cada componente, un extracto de código del proyecto.

Inicialmente los modelo de Django no son serializables y por esa razón es necesario realizar un paso previo. Concretamente, ese paso consiste en codificar el modelo utilizando la clase *AlbumSerializer*. Como se observa en el código 5, para poder implementar esa funcionalidad es necesario heredar de la clase ModelSerializer. Finalmente, es necesario indicar una tupla con los atributos del modelo que serán incluidos cuando se codifique. En particular,

```
1 class AlbumSerializer(ModelSerializer):
2     class Meta:
3         model = Album
4         fields = '__all__'
```

Código 5: Codificación del modelo Álbum utilizando la clase ModelSerializer.

El siguiente paso consiste en definir un ViewSet con un método para eliminar un álbum. Antes de ejecutar el método es importante definir que métodos aceptará, la ruta y el nombre. Esa definición se muestra en la línea 7 del código. 6. Como se puede observar en el código, la ruta aceptará el método *DELETE* y el nombre de la ruta será *eliminarAlbum*.

```
class AlbumViewSet(ModelViewSet):
        queryset = Album.objects.all()
        # Indica el serializador
        serializer_class = AlbumSerializer
        # Indica los métodos aceptados y la ruta

        @detail_route(methods=['DELETE'], url_name='eliminarAlbum', url_path='eliminarAlbum')
        def eliminarAlbum(self, request, pk):
            mensajes_error={
            'album_no_existe':'El álbum no existe.',
            'identificador_no_valido':'El identificador no es válido.',
            }
            mensaje_exito= 'El álbum se ha eliminado correctamente.'
            try:
                    album = Album.objects.get(pk=int(pk))
        except ObjectDoesNotExist:
                    return JsonResponse({'exito': 'False','respuesta': mensajes_error['album_no_existe']},
                    status=status.HTTP_500_INTERNAL_SERVER_ERROR)
        except ValueError:
            return JsonResponse({'exito': 'False', 'respuesta': mensajes_error['identificador_no_valido']},
                     status=status.HTTP_500_INTERNAL_SERVER_ERROR)

            album.delete()

            return JsonResponse({'exito': 'True', 'respuesta': mensaje_exito},status=status.HTTP_200_OK)
```

Código 6: Definición del controlador eliminarAlbum.

Una vez definido el controlador, es necesario registrarlo en la aplicación para que pueda ser localizable por el usuario. Como se observa en el cuadro 7, se asocia un espacio de nombres para evitar colisiones con otras rutas.

```
        router = routers.DefaultRouter()
        router.register(r'album', AlbumViewSet)
```

Código 7: Registro del contenedor de métodos en el componente Routers.

6.1.4. Social Django

Con la finalidad de permitir a las personas iniciar sesión o registrarse se utilizó este módulo para Django [13]. El módulo integra Open Authorization (OAuth2) para la comunicación con los proveedores de autentificación y en particular define un conjunto de terminologías que son importantes entender:

- **Servidor de autorización.** Es un término del protocolo OAuth2 para referirse al servidor encargado de generar tokens de acceso, validar usuarios y credenciales.
- **Social User.** Es un modelo definido por el módulo para asociar la información social con el usuario del sistema.
- **Pipeline.** Cada acción que realice el usuario, ya sea para iniciar sesión o registrarse, sigue unas etapas secuenciales definidas por el programador.
- **Partial.** Es un *decorador* que permite guardar el contexto del estado actual del pipeline para posteriormente volver. Puede ser útil para dirigir a un usuario a un controlador para pedir datos de perfil.

Como se ha comentado anteriormente, el proceso secuencial se inicia cuando el usuario accede al recurso pero no se explicó como los usuarios pueden iniciarlo. Para ello, define tres rutas de interacción. En concreto, para iniciar el pipeline, se utiliza la ruta *social:begin* junto con un argumento del proveedor de autentificación.

Para organizar y mostrar las rutas, se proporciona un cuadro 6 que contienen las rutas, parámetros necesarios y una descripción.

Nombre de la ruta	Parámetros	Descripción
social:begin	Nombre del proveedor social	Permite iniciar el proceso de iniciar sesión o registro.
social:complete	Contexto con la información del proveedor social completado	Permite volver a la etapa que se pausó del pipeline.
social:disconnect	Nombre del proveedor social	Permite desconectar al usuario del proveedor social.

Cuadro 6: Rutas que proporciona Social Django

Para aclarar los conceptos, se presenta en el cuadro 8 el prototipo de la función que corresponde a obtener información de Google+.

En particular, se puede observar que la función está definida para ser pausada utilizando el decorador *partial*. Además, el prototipo de la función define dos argumentos *user* y *is_new* para distinguir si el usuario ya se encuentra en la aplicación.

Concretamente, se puede producir el caso que el sistema detecte que el usuario ya esté en el sistema y el argumento *user* tendrá una referencia al modelo del usuario o en caso contrario *AnonymousUser*. Este argumento en conjunción con *is_new* permite diferenciar cuando un usuario quiere asociar varias cuentas a un mismo correo electrónico.

[13] https://github.com/python-social-auth/social-app-django

```python
@partial
def get_informacion_google(backend, strategy, details, response, user=None, is_new=False, *args, **kwargs):
    pass
```

Código 8: Definición de una etapa en el pipeline.

Por último, para proporcionar al lector una visión general de las etapas definidas, se muestra en la figura 27 el orden secuencial de cada etapa.

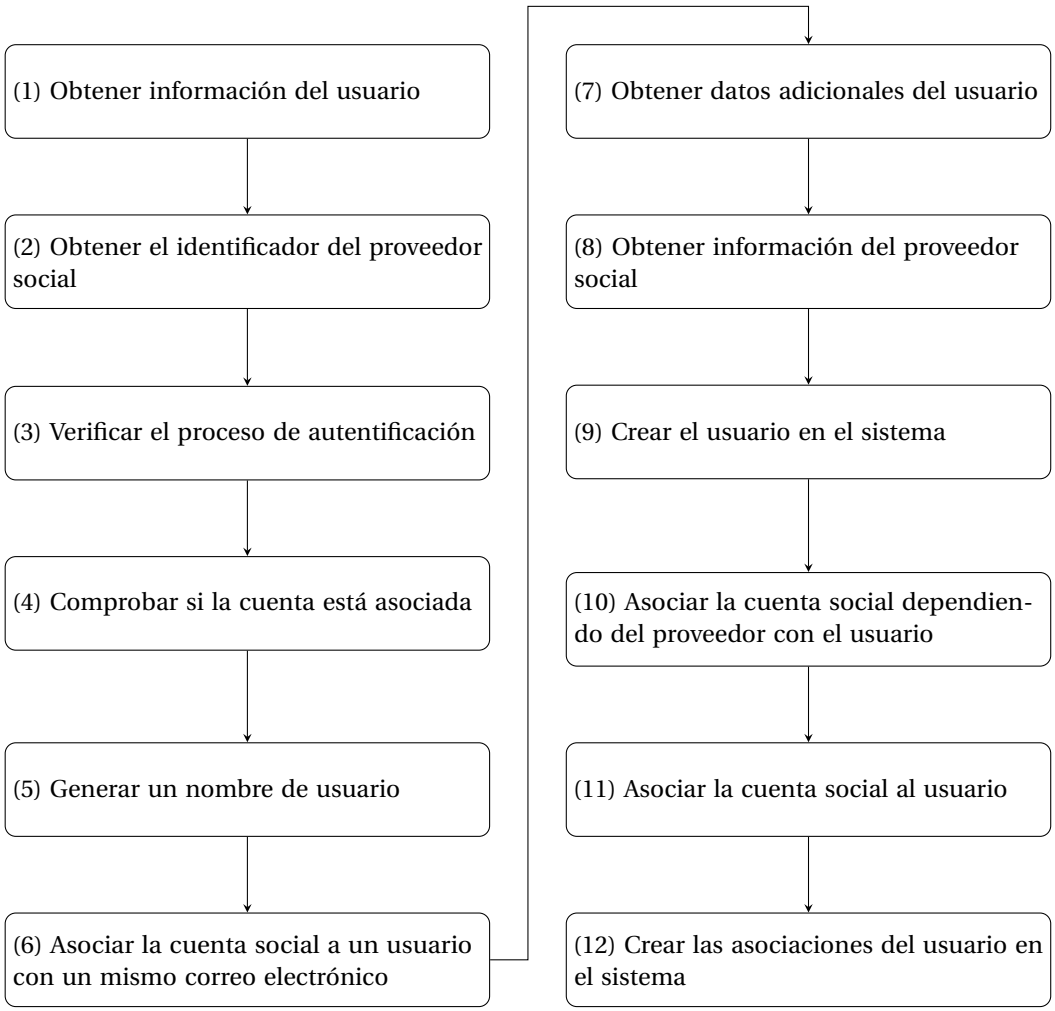

Figura 27: Etapas para generar una cuenta social

6.2. Tecnologías del lado del cliente

Para diseñar la capa visible al usuario fue necesario utilizar las tecnologías que se describen en esta sección.

6.2.1. Bootstrap

Es un framework que facilita la construcción de páginas web adaptando el diseño a la resolución de la pantalla por medio de un sistema de cuadricula con 12 columnas. Está desarrollado por *Twitter* el cual ha publicado el código bajo la licencia *MIT*. [14]

Actualmente se encuentra en la versión 4 y entre las novedades de está versión es la integración del modelo de caja flexible en sus clases para permitir solventar carencias y limitaciones del antiguo modelo basado en elementos flotantes.

En el proyecto se utilizó para que las páginas se adapten a la resolución de la pantalla, por los componentes reutilizables y por las clases de utilidad incluidas y que permiten simplificar el código de estilo.

6.2.2. jQuery

Es una librería diseñada para simplificar la programación en JavaScript y su utilización en el proyecto permitió mejorar la compatibilidad entre navegadores y simplificar funciones que necesitarían un elevado número de líneas de código para ser implementadas. Es una librería con un elevado número de funcionalidades y en este proyecto se utilizó un subconjunto de ellas que se indican a continuación:

- Manipulación de reglas aplicadas en las hojas de estilos.
- Selección de elementos del documento.
- Asignación y gestión de eventos.
- Interacción con la tecnología Asynchronous JavaScript And XML (AJAX).

Para mostrar un ejemplo de esta librería aplicada en el proyecto, se muestran en el código 9 dos de ellas: la selección de elementos del documento y la asignación de eventos.

```
1 SolicitudAmistadController = {
2         // Función que inicializa las interacciones con los botones.
3         inicializar: function () {
4                 $('.nuevaSolicitud').on("click", this.nuevaSolicitudAmistad);
5                 $('.aceptarSolicitud').on("click", this.aceptarSolicitudAmistad);
6                 $('.cancelarSolicitud').on("click", this.cancelarSolicitudAmistad);
7         },
8 });
```

Código 9: Asociación de eventos a botones utilizando la librería jQuery.

[14] https://getbootstrap.com/

6.2.3. MJML

Para diseñar los correos electrónicos se utilizó este framework para facilitar el diseño responsive y que se adapten a cualquier dispositivo. Posee una sintaxis sencilla y una librería de componentes que permiten agilizar el proceso de diseño de un correo electrónico.

Como se puede visualizar en la figura 28, esa sintaxis no es directamente usable en los correos electrónicos ya que es necesario realizar un paso previo que consiste en la conversión a código HTML. Para la conversión se utiliza el motor de MJML que tiene como siglas (MailJet Markup Language) y se encarga de convertir esas etiquetas propias del lenguaje a código HTML para poder ser utilizado en los correos electrónicos. A efectos prácticos, esa conversión se puede visualizar en tiempo real pulsando en *View HTML*.

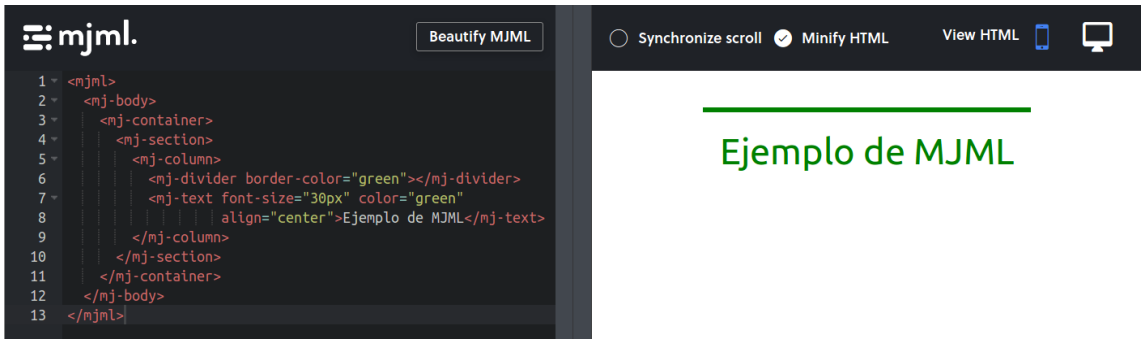

Figura 28: Ejemplo de conversión de etiquetas MJML a código HTML

6.2.4. Flexbox

Antiguamente la alineación de los elementos se diseñaba utilizando propiedades flotantes que dificultaban el posicionado de los elementos. Para solucionar y ayudar a distribuir los elementos se diseñó este modelo unidimensional que, a excepción de las propiedades flotantes que permiten posicionar horizontalmente elementos, este modelo otorga un control total del elemento, esto es de dirección, alineación, orden y tamaño.

El funcionamiento del modelo parte de un contenedor padre que se llama flex-container que puede tener la propiedad *display:flex* o *display:flex-inline*. De esta manera, el contenedor se convierte en un contenedor flexible y los elementos que tienen se convierten en flexibles, es decir, que se adapten al espacio disponible.

Como se puede observar en la figura 29, el modelo de caja flexible es compatible con bastantes navegadores exceptuando antiguas versiones de Internet Explorer que no es del todo compatible y para mantener la compatibilidad es necesario añadir prefijos a las propiedades flex.

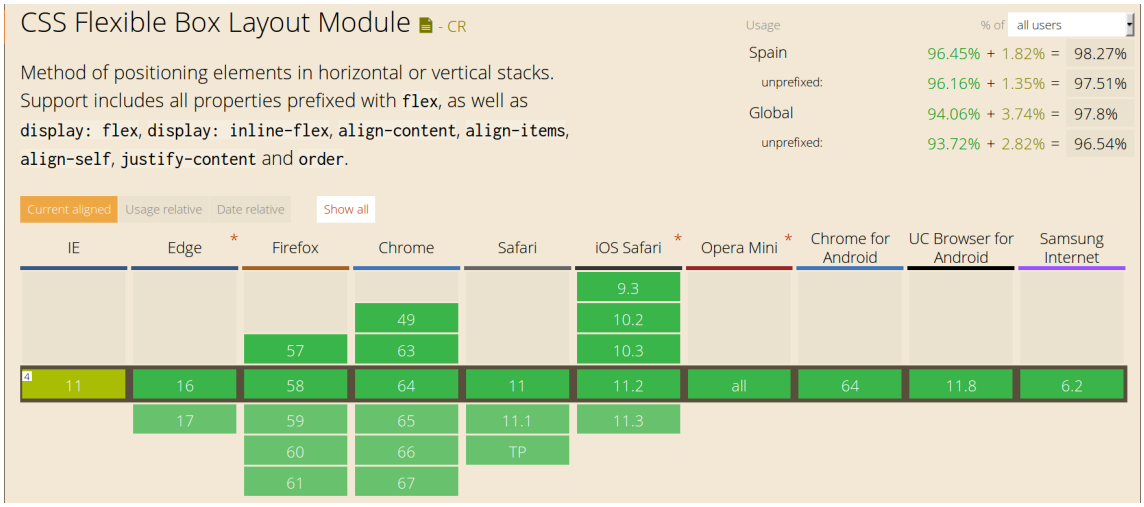

Figura 29: Compatibilidad del modelo de caja flexible en los navegadores web. [15]

6.2.5. SCSS y CSS

Una buena práctica en el desarrollo del proyecto fue desacoplar el contenido de un documento HTML de la definición de estilos, que se define usando CSS. En particular se ha utilizado CSS3, la última versión del estándar que permite la definición de estilos que se aplican a elementos de una web, mediante reglas. De esta forma, las reglas se componen de un selector, que define los elementos a los que se le aplicará el estilo, y un bloque de declaración, en el que se describirán las propiedades que se le aplicarán al elemento.

A pesar de todo, esta tecnología tiene con una serie de limitaciones. Por ejemplo, no cuenta con mecanismos sencillos para favorecer la modularidad del código. Por ello han surgido una serie de *metalenguajes* que añaden nuevas características para facilitar el desarrollo pero que, en la última etapa, acaban generando código CSS compatible. Uno de estos metalenguajes es SCSS utilizado en el proyecto debido a las funcionalidades que añade. En particular añade las siguientes:

- Definición de variables.
- Anidamiento de reglas.
- Uso de estilos reutilizables (mixins).
- Herencia de selectores.
- Uso de bucles.

[15] Fuente: `https://caniuse.com/#feat=flexbox`

7. Seguimiento del proyecto

El propósito de este capítulo es describir los sprints para diseñar e implementar la red social. Para ello, se definieron 4 sprints en la planificación inicial que se describirán brevemente y posteriormente cada una de ellas se detallará en su sección.

- Primer sprint. Definir los objetivos del proyecto y realizar una iteración completa del diseño centrado en el usuario.
- Segundo sprint. Diseñar e implementar las historias de usuario del usuario no autentificado.
- Tercer sprint. Diseñar e implementar las historias de usuario que tienen relación con el muro, perfil de usuario, álbumes de fotos y amigos.
- Cuarto sprint. Diseñar e implementar las notificaciones generales y solicitudes de amistad en tiempo real, los vídeos, preferencias generales y finalmente el bloqueo y desbloqueo de usuarios.

7.1. Primer sprint

En el primer sprint se establecieron los objetivos del proyecto y se realizó una iteración completa del diseño centrado en el usuario. Para mantener el orden se menciona en esta sección 7.1 y en el capítulo 4 se proporciona el proceso.

En síntesis, se definió el perfil de usuario ideal y posteriormente se indagó en dos redes sociales, Facebook y LinkedIn, analizando puntos fuertes y débiles con el objetivo de detectar carencias.

A continuación se realizó un cuestionario para conocer los hábitos y usos de las redes sociales y posteriormente se definieron a dos personas ficticias a partir de la información obtenida.

A continuación se planificaron 4 escenarios y se realizaron pruebas de usuario en esas redes sociales para conocer puntos de diseño a tener en cuenta para diseñar las interfaces de usuario.

Con toda la información, se diseño un prototipo de alta fidelidad a partir de los esbozos con la finalidad de realizar test con usuarios y de evaluar la facilidad de uso del prototipo.

7.2. Segundo sprint

Después de realizar una iteración completa del diseño centrado en el usuario se obtuvo información sobre ciertas partes del diseño que podrían ser mejorables y que se introducirán en el proyecto.

En este sprint se realizarán las historias de usuario que corresponden a las funcionalidades de un usuario no autentificado.

7.2.1. Diseño e implementación de la pantalla de presentación

Para mostrar la página principal, fue necesario definir un controlador encargado de generar un fichero HTML y realizar peticiones a la base de datos para obtener el número de usuarios, imágenes y vídeos. Eso se consiguió añadiendo en el archivo *views.py* el controlador que se observa en el código 10.

```
1  def index(request):
2          if request.user.is_authenticated:
3                  return redirect(reverse('greenbook:muro', args=[request.user.username]))
4
5      num_usuarios = User.objects.get_num_usuarios()
6      num_imagenes = Imagen.objects.get_num_imagenes()
7      num_videos = Video.objects.get_num_videos()
8
9          contexto= {'numUsuarios': num_usuarios, 'numImagenes': num_imagenes, 'numVideos':num_videos}
10         return render(request, 'GreenBook/index.html', contexto )
```

Código 10: Controlador de la página principal.

Como se puede observar en el extracto de código 10, el controlador se dividió en tres bloques. El primer bloque (lineas 2-3) comprueba si el usuario tiene una sesión activa en el navegador, para dirigir al usuario a su muro. Internamente no se permite que un usuario que tenga una sesión activa pueda visualizar la página principal.

El segundo bloque (líneas 5-7) obtienen los usuarios, vídeos e imágenes de la base de datos Redis. Los datos de los usuarios se guardan en memoria en un diccionario y cada vez que se añade o elimina un objeto se incrementa o decrementa utilizando las señales que proporciona Django y que veremos en un momento.

El último bloque (líneas 8-9) genera la plantilla de presentación utilizando las variables del segundo bloque.

Como se comentó en el segundo bloque, cada vez que se añade o elimina de la base de datos MySQL un usuario, imagen o vídeo se activa la señal *post_save* o *post_delete* dependiendo si se está añadiendo o eliminando un objeto.

La función que se muestra en el código 11 implementa la funcionalidad de incrementar el valor del número de usuario en una unidad. Esta función se ejecutará cada vez que se inserte un usuario. La misma funcionalidad se aplica en la señal *post_delete* pero decrementando en una unidad cada vez que un usuario elimine su cuenta.

```
1  @receiver(post_save, sender=User)
2  def post_save_usuario(**kwargs):
3      if kwargs['created']:
4          num_usuarios = cache.get(settings.KEY_TOTAL_USUARIOS, 0)
5          cache.set(settings.KEY_TOTAL_USUARIOS, num_usuarios + 1)
```

Código 11: Señal post_save para actualizar el número de usuarios en la base de datos Redis.

Después de utilizar el decorador para las señales *post_save* y *post_delete* se persiste en la memoria caché pero todavía no se obtiene el número de usuarios. Para obtenerlos, es necesario realizar una consulta a la base de datos Redis utilizando la llave *KEY_TOTAL_USUARIOS*.

El código 12 consulta si la llave se encuentra en memoria y en caso que se encuentre devuelve su valor. En caso que no se encuentre, el sistema realiza una consulta a la base de datos MySQL para obtener el número de usuarios y posteriormente insertarlos en el diccionario. Finalmente la función devuelve el número de usuarios.

```python
class UsuarioManager(models.Manager):
    def get_num_usuarios(self):
        if cache.has_key(settings.KEY_TOTAL_USUARIOS):
            return cache.get(settings.KEY_TOTAL_USUARIOS, 0)

        num_usuarios= User.objects.all().exclude(is_staff=True).count()
        cache.set(settings.KEY_TOTAL_USUARIOS, num_usuarios)
        return num_usuarios
```

Código 12: Obtener el número de usuarios de la memoria Redis o MySQL.

Después de generar la plantilla, es necesario asociar el controlador a una dirección URL para que los usuarios puedan acceder. Para ello, el código 13 muestra la asociación al controlador que mostrará la página principal y la ruta al recurso.

```python
urlpatterns = [ url(r'^$', views.index, name='index'), ]
```

Código 13: Asociación del nombre de recurso con el controlador de la página principal

La figura 30 corresponde a la plantilla HTML de la página principal que genera internamente el gestor de plantillas de Django. En la página se utilizó un plugin llamado particles.js [16] para generar las partículas que se muestran de fondo en el titulo de la aplicación.

En la sección estadísticas, se proporcionan datos estadísticos de los usuarios, imágenes e vídeos.

Cuando un usuario accede a la página de presentación, el gestor de plantillas internamente evalua la variable y la reemplaza por su valor. Para mostrar un ejemplo, en la línea 5 del código 14, se utiliza la variable {{numUsuarios}} que contiene el número de usuarios en la red social y se posiciona en una zona concreta para ser reeemplazada por su valor. En este caso, en la parte inferior de *Usuarios*.

```html
<div class="col-sm-4">
        <div class="estadisticas-item text-center">
                <img src="{% static 'iconos/usuarios.svg' %}" class="img-responsive d-block mx-auto">
                <h3>Usuarios</h3>
                <span class="numero-usuarios">{{ numUsuarios }}</span>
        </div>
</div>
```

Código 14: Extracto de código HTML para mostrar los usuarios en la red social.

[16] https://github.com/VincentGarreau/particles.js/

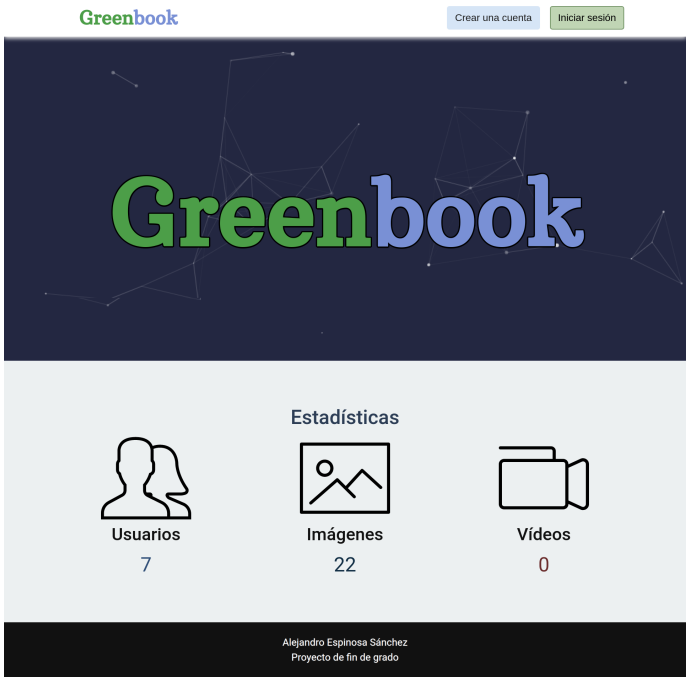

Figura 30: Página principal.

7.2.2. Diseño e implementación de la pantalla de inicio de sesión

La figura 31 permite observar el diseño de iniciar sesión. En particular cuando el usuario pulsa el botón de iniciar sesión, antes de enviar los datos por AJAX al servidor, se activa un evento para validar el formulario comprobando que los campos no estén vacíos.

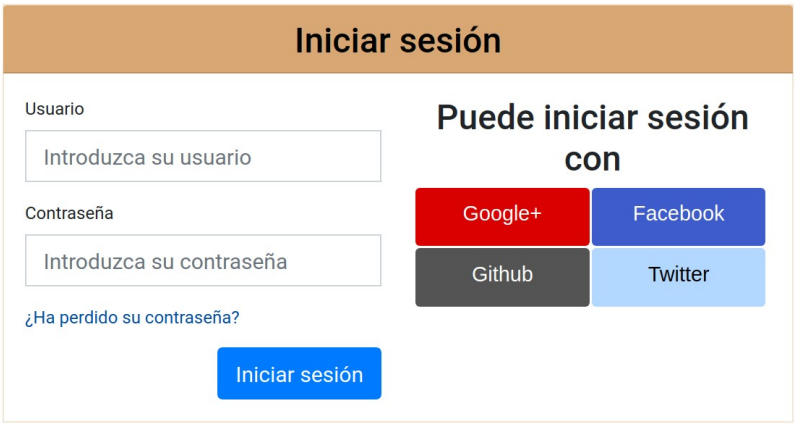

Figura 31: Inicio de sesión.

Cuando el servidor recibe la petición realiza las siguientes comprobaciones:

- Validar que los datos sean correctos.
- Comprobar que el usuario exista y la contraseña coincida con la del usuario.
- Comprobar que el usuario haya validado la cuenta.

Si se produce un error, el servidor envía un JSON informando al usuario del error. Por el contrario, si todas las comprobaciones son correctas, se autentifica al usuario y dirige a su muro.

Para ilustrar los tres posibles errores, se muestra en la figura 32 los posibles mensajes de error o advertencia que puede visualizar el usuario. El primero se muestra cuando los datos no son correctos y el segundo cuando es necesario validar el correo electrónico.

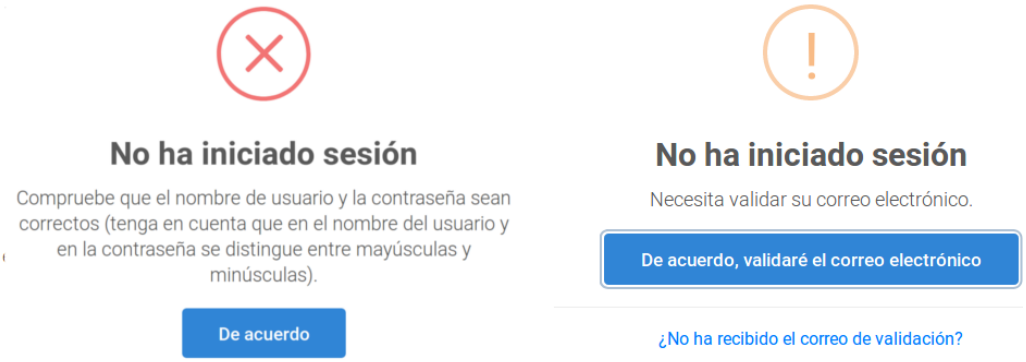

Figura 32: Ventanas modales al pulsar el botón de iniciar sesión

Cuando el usuario visualiza el mensaje de no haber validado la cuenta, se le proporciona una forma de volver a recibir el correo de validación. Internamente, se envía una petición AJAX a un recurso de la aplicación que realiza las siguientes comprobaciones.

- Verificar que el usuario exista y no haya verificado su cuenta.
- Generar una función hash válida durante un día y generada a partir del identificador del usuario, correo electrónico y fecha y hora.
- Generar la plantilla de correo electrónico con información del usuario y el enlace al recurso que permite confirmar el correo electrónico.
- Enviar el correo electrónico al usuario.

El extracto de código 15 define los enlaces a las redes sociales. Si nos fijamos en la línea 2, se puede observar como se enlaza el recurso de iniciar sesión utilizando Google+. Para ello, se hace referencia a la ruta donde se encuentra el controlador y se añade el nombre del proveedor social.

```
<div class="botonesSociales">
   <a href="{% url 'social:begin' 'google-oauth2' %}" class="btn botonSocial botonGoogle">Google+</a>
   <a href="{% url 'social:begin' 'facebook' %}" class="btn botonSocial botonFacebook">Facebook</a>
</div>
<div class="botonesSociales">
   <a href="{% url 'social:begin' 'github' %}" class="btn botonSocial botonGithub">Github</a>
   <a href="{% url 'social:begin' 'twitter' %}" class="btn botonSocial botonTwitter">Twitter</a>
</div>
```

Código 15: Código HTML para iniciar el inicio social.

7.2.3. Diseño e implementación de la pantalla de cambiar la contraseña

La recuperación de la contraseña permite que un usuario que tenga una cuenta pueda cambiar la contraseña. Se pueden producir dos escenarios, que el usuario haya creado una cuenta utilizando una cuenta social o manualmente escribiendo todos los datos. En ambos casos el sistema validará que el correo exista en la base de datos y enviará un correo con un enlace para iniciar el proceso de cambio de contraseña.

El escenario de solicitar cambiar la contraseña de un usuario que se registró utilizando una cuenta social es una excepción. Por defecto, el usuario no proporciona ningún nombre de usuario o contraseña en el sistema y el sistema asigna a un nuevo usuario el nombre de usuario del proveedor social y añade caracteres aleatorios al final del nombre en caso de colisión.

El formulario tiene una alerta informativa de los pasos que el usuario debe realizar para cambiar la contraseña. Por otra parte, el formulario tiene un campo de texto para obtener el correo del usuario y un botón que permite enviar una petición AJAX al servidor.

Figura 33: Diseño del formulario de cambiar la contraseña.

Cuando el correo existe, el servidor genera una plantilla (véase figura 34) con la siguiente información:

- **Ruta para iniciar el proceso de cambio de contraseña.** Internamente se genera una función hash válida durante un día a partir del día actual, identificador del usuario y correo electrónico.
- **Información adicional.** Para proporcionar un punto de seguridad, se muestra información aproximada de la dirección IP e incluyendo la fecha y hora de la petición. Para obtener la ubicación aproximada se utiliza la librería *geoip2*. [17]

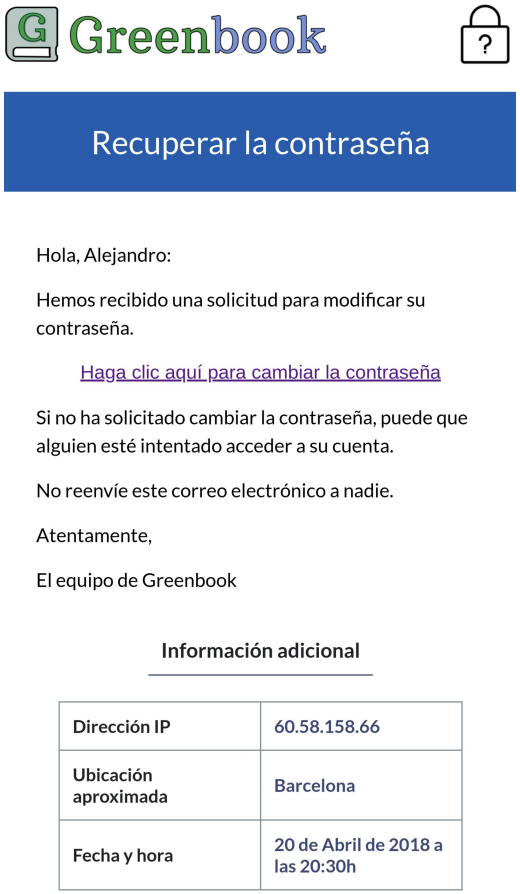

Figura 34: Correo electrónico para modificar la contraseña.

Figura 35: Correo electrónico informando del cambio de contraseña exitoso.

[17] https://docs.djangoproject.com/en/2.0/ref/contrib/gis/geoip2/

7.2.4. Diseño e implementación de la pantalla de crear una cuenta

En las conclusiones de los test con usuarios se cambió el aspecto del diseño conceptual 14, debido a que concentraba los campos de usuario y perfil en un única zona y no resultaba cómodo para introducir los datos. Por otra parte, se permitió que el usuario pudiera crear una cuenta a partir de una red social en vez de tener que escribir los datos de la cuenta y perfil.

La ventana de crear una cuenta está formada por dos paneles, el primer panel contiene información del usuario y el segundo información del perfil de usuario.

Inicialmente el panel de perfil está desactivado. El sistema esperará a que el usuario introduzca los datos y pulse el botón *Continuar* para validar los datos y permitir avanzar a la siguiente pestaña.

En particular, cuando el usuario escribe en el campo nombre de usuario o correo electrónico y el campo de texto pierde el foco (*focusout*) se envía una petición AJAX con el texto para comprobar si el nombre de usuario o correo electrónico están registrados en la aplicación.

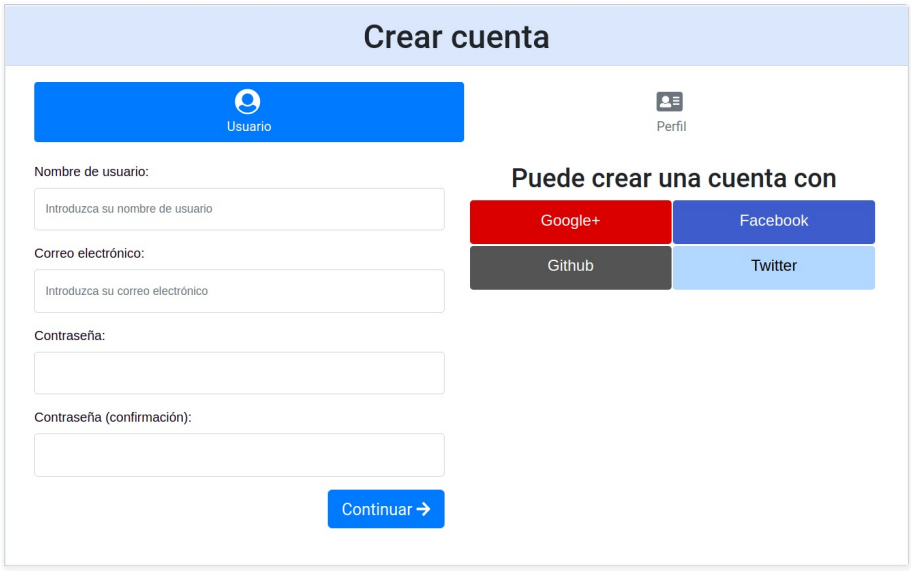

Figura 36: Diseño del formulario de registro en la pestaña de usuario.

Se puede producir el escenario que el formulario no sea válido y siguiendo la heurística de Nielsen **visibilidad del estado del sistema**, se proporciona al usuario una retroalimentación de los campos válidos y no válidos. En particular, para proporcionar un ejemplo de error de validación se adjunta la figura 37.

Para implementar este tipo de validación se utilizaron las funciones definidas de Bootstrap [18].

[18] https://getbootstrap.com/docs/4.0/components/forms/#validation

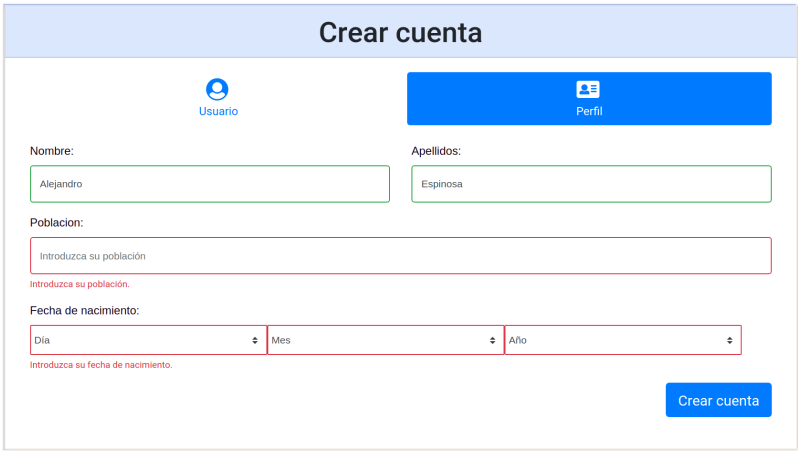

Figura 37: Diseño del formulario de registro en la pestaña de perfil.

Cuando se crea una cuenta, ésta no se activa automáticamente hasta que el usuario valida su correo electrónico. Para ello, el sistema enviará un correo electrónico con un enlace al controlador de activar la cuenta de usuario.

La figura 39 muestra el correo de confirmación. Este mensaje tiene un enlace a un controlador que se utiliza para activar la cuenta del usuario.

Algoritmo 1: Controlador que activa una cuenta de usuario.

```
1  Obtener usuario a partir de los datos de la petición;
2  si No he obtenido usuario entonces
3      Generar el mensaje de error: "Usuario no encontrado";
4      Dirigir al usuario a la página de iniciar sesión para mostrar el error;
5  en otro caso
6      si Ha verificado su cuenta entonces
7          Generar el mensaje de error: "No puedes verificar otra vez la cuenta";
8          Dirigir al usuario a la página de iniciar sesión para mostrar el error;
9      en otro caso
10         si Token del usuario coincide con el proporcionado entonces
11             Activar la cuenta y persistir los cambios;
12             Autentificar al usuario;
13             Generar el mensaje: "Ha validado correctamente su correo electrónico.";
14             Dirigir al usuario a su muro y mostrar el mensaje generado;
15         en otro caso
16             Generar el mensaje de error: "No se ha podido validar su correo electrónico".;
17             Dirigir al usuario a la página de iniciar sesión para mostrar el error;
18         fin
19     fin
20 fin
```

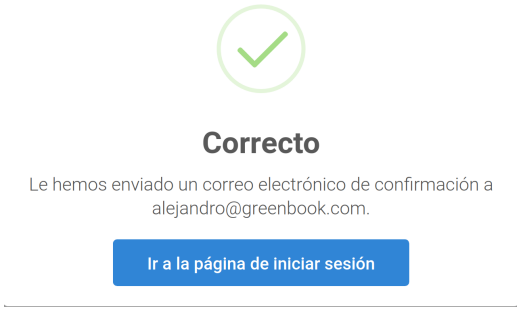

Figura 38: Ventana modal para informar al usuario que debe confirmar su correo electrónico.

Figura 39: Correo electrónico enviado para activar su cuenta.

Aparte de la primera forma que consiste en introducir manualmente los datos, se puede optar por registrarse o iniciar sesión con Google+, Facebook, Github y Twitter. Como se comentó en la sección 6.1.4, cuando el usuario pulse el botón para iniciar sesión con un proveedor concreto, activará las etapas de la figura 27.

Algunas etapas se pueden pausar para obtener información adicional del usuario utilizando el decorador *partial*. En la figura 40 se muestra el escenario de un usuario que quería registrarse pero el sistema no había detectado la fecha de nacimiento y población. Además, también muestra el proceso de verificación de los datos, en particular la fecha de nacimiento comprobando que tenga 13 años o más para crear una cuenta.[19]

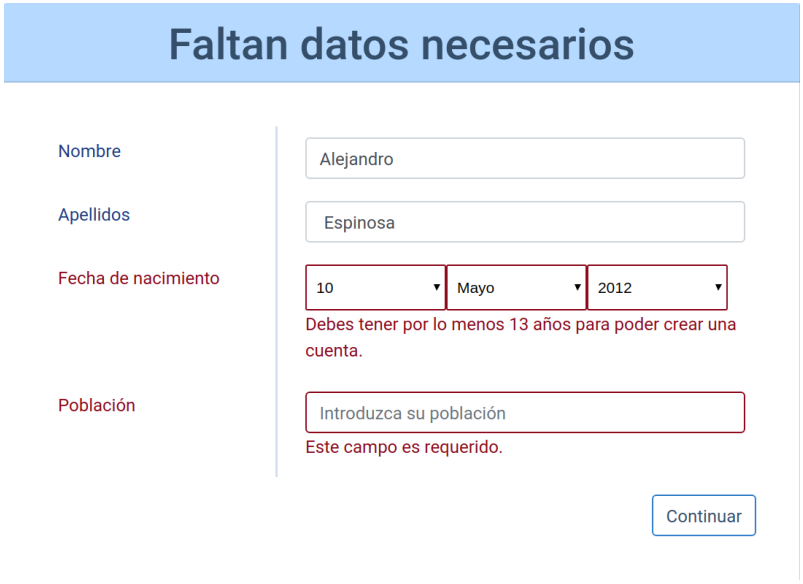

Figura 40: Datos de perfil necesarios pero no encontrados.

[19] Como curiosidad, se fijó la edad mínima a 13 años porque es la edad mínima permitida en Estados Unidos. No obstante, en España, la Ley de protección de datos de carácter personal establece la edad mínima a 14 años.

7.3. Tercer sprint

En el anterior sprint se diseñaron e implementaron las historias de usuario que correspondían a los usuarios visitantes, es decir, aquellos que no iniciaron sesión.

En este sprint se realizarán las historias de usuario que tienen relación con el muro, perfil de usuario, álbumes, imágenes y amigos.

7.3.1. Diseño e implementación del muro

El diseño del muro está dividido en varias secciones: cabecera con información del perfil, menú de navegación y contenido del muro.

El diseño de la cabecera y menú de navegación se puede observar en la figura 41 y en particular para diseñar la cabecera se dividió en capas para poder organizar correctamente la información. Las capas se definieron con propiedades flexibles para que se adaptasen al contenido de la pantalla.

Para diseñar la cabecera se definió un contenedor que englobara todos los elementos interiores y que tuviera un ancho fijado para permitir posicionar los elementos. Para ello se definieron dos columnas, la primera para mostrar información del perfil y la segunda para posicionar el botón de portada. Las columnas se definieron utilizando las clases que ofrece Bootstrap y en particular la primera se definió utilizando *col-2* y la segunda *col-10*.

La primera columna para mantener un buen contraste entre el fondo y letras, se añadió una capa de opacidad con la intención que facilitar la lectura del texto cuando la imagen de portada tiene demasiada luminosidad.

Como se propuso en los test con usuarios 4.4, se aplicó la mejora propuesta de un participante que consistía en añadir un texto informativo, para distinguir cuando un usuario visualizaba el perfil de otro. El texto se añadió en la parte superior de la imagen de usuario tal y como se muestra en la figura 41.

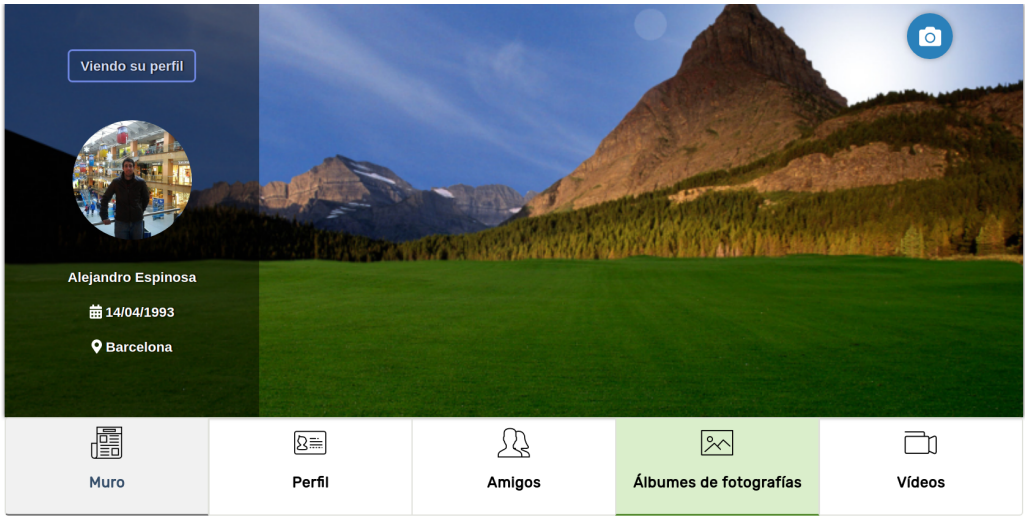

Figura 41: Diseño de la cabecera con información del perfil y menú de navegación.

Los botones de navegación se definieron flexibles para que ocupasen el mismo tamaño. Esa propiedad permitió que en caso de añadir un nuevo botón al contendedor éste se adapte proporcionalmente.

Para proporcionar una retroalimentación del recurso en el que está el usuario se añadió un color de fondo al botón modificando el evento (*onhover*) con propiedades de estilo. En particular se muestra un extracto de código SCSS con estilos aplicados a los botones (véase código 16).

```
.botonMenuActual {
  background-color: #f1f1f1;
  border-bottom: 2px solid darken(#f9f9f9, 50%);

  p {
          font-weight: bold;
          color: #354e6a;
    }
  }
albumFotografias:hover {
  border-bottom: 2px solid rgba(85, 139, 47, 1);
  background-color: lighten(rgba(85, 139, 47, 1), 50%);
  transition: all linear 0.50s;
  }
```

Código 16: Código para dar estilos a los botones de navegación.

El diseño presentado anteriormente es adaptable a cualquier resolución de pantalla. Como se muestra en la figura 42, los elementos de la pantalla e iconos se adaptan al ancho del dispositivo.

Para facilitar la modificación de la imagen de perfil, en dispositivos de poca resolución de pantalla se deja visible el icono para que, de esta forma, puedan pulsar sin necesitar obtener el foco para hacer visible el icono.

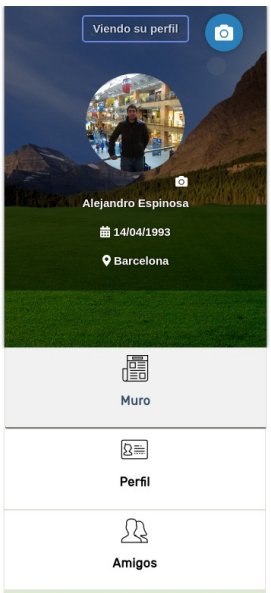

Figura 42: Diseño de cabecera con información de perfil y menú de navegación en dispositivos de poca resolución.

Tras presentar el diseño de la cabecera se presentará el contenido del muro, que está organizado en forma de tres columnas: dos laterales y una central. Los elementos laterales están formados en forma de ventanas y para cada una se aplica un estilo concreto para poder distinguir un contenido de otro. La finalidad de estas ventanas es proporcionar información al usuario y para ello se definieron cuatro tipos:

- **Amigos.** Muestra hasta seis amigos ordenados en orden creciente según la fecha de amistad.
- **Imágenes.** Visualiza las seis imágenes más recientes añadidas por el usuario.
- **Cumpleaños.** Muestra cuantos días y meses quedan para los cumpleaños de los amigos.
- **Vídeos.** Visualizar los seis vídeos más recientes añadidos por el usuario.

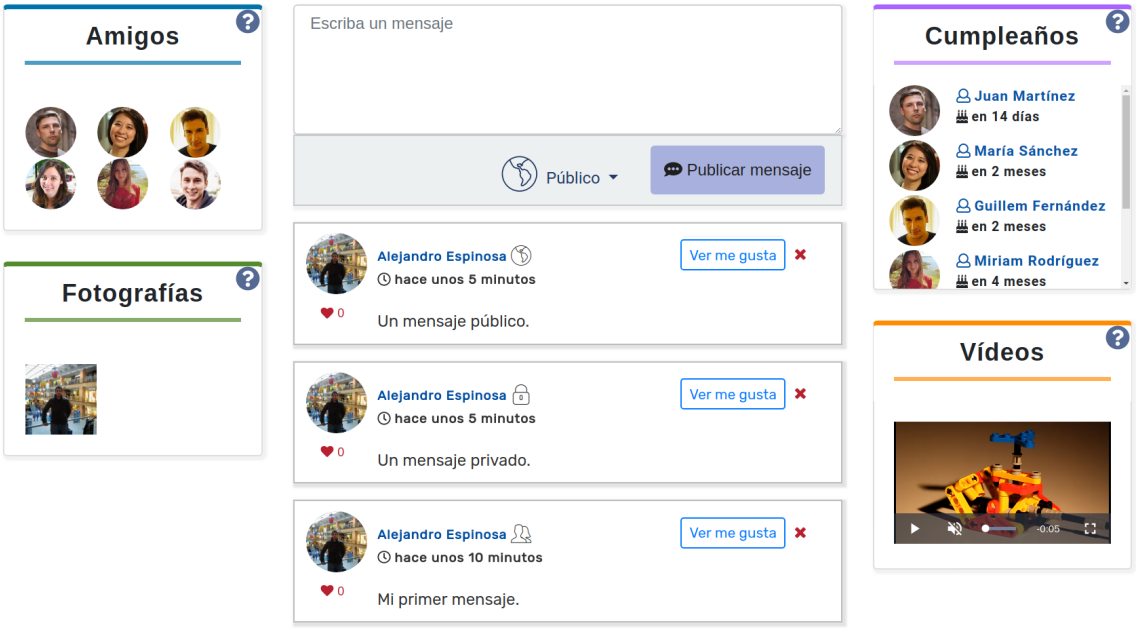

Figura 43: Contenido del muro.

En los mensajes se puede observar que se añadió un icono para diferenciar la privacidad del mensaje. Internamente para gestionar el nivel de privacidad del mensaje, se definió un atributo privacidad en el modelo mensaje.

Para diseñar una página no bloqueante y acelerar la visualización se definió, para cada ventana y los mensajes, una petición AJAX que se ejecuta cuando el usuario accede al muro.

Como se comentó en los test de usuarios 4.4, se implementó la funcionalidad de ver, para cada publicación, las personas a las que le gusta. Para ello, cada publicación tiene una ventana modal adaptada de Bootstrap oculta y no visible si no se pulsa en el botón "'Ver me gusta".

Cuando se visualiza un muro, internamente se envía una petición de conexión WebSockets para conectarse al muro de ese usuario. El servidor procesará la petición agregándolo a ciertos grupos dependiendo del tipo de usuario.

Para ello, se sintetizan en el cuadro 7 los posibles grupos en los que podrá estar un usuario. De este cuadro, la primera columna muestra los tipos de usuarios que podrán conectarse al muro y las tres columnas

siguientes los nombres del grupo a los que se conectarán. Para distinguir los muros de usuario, a cada grupo se concatena el identificador de usuario propietario de muro.

	Nombre del grupo		
Tipo de usuario	**MuroPrivado**	**MuroPublico**	**MuroAmigo**
Usuario propietario del muro	Sí	Sí	Sí
Usuario no amigo	No	Sí	No
Usuario amigo	No	Sí	Sí

Cuadro 7: Grupos a los que se puede agregar a un usuario.

Para actualizar las personas a las que le gusta un mensaje y gestionar las publicaciones, se multiplexa el canal por donde se envía la información añadiendo la acción concreta. De está forma, cuando un usuario envíe o reciba una respuesta podrá saber la acción a realizar.

Figura 44: Personas a las que le gusta una publicación.

7.3.2. Diseño e implementación de la visualización y modificación del perfil de usuario

En la figura 45, se puede observar el diseño del perfil de un usuario. Para mostrarlo fue necesario definir un controlador con un argumento para obtener el perfil de usuario de la base de datos a partir el nombre de usuario.

Figura 45: Diseño del perfil de usuario.

En la figura 46 se observa el formulario de modificación del perfil de usuario. Por temas de seguridad, el acceso a ese recurso se restringe al propio usuario y otro usuario que visualice un perfil de otro usuario no podrá realizar modificaciones. Internamente, el controlador obtiene el modelo de usuario de la sesión y posteriormente el perfil asociado.

Figura 46: Modificación del perfil de usuario.

Se puede producir el escenario que un usuario desee modificar su imagen de perfil y para ello pulse el botón ”Modificar”. En tal caso, se activará una ventana modal para añadir una imagen y posteriormente seleccionar la imagen de miniatura.

El usuario podrá optar por añadir la imagen desde su gestor de archivos o arrastrando y soltando. A continuación podrá continuar a la siguiente pestaña para obtener una imagen de miniatura de una zona concreta de la imagen. Para ello se utiliza un componente llamado Cropper [20]. El área de interés de la

[20] https://fengyuanchen.github.io/cropper/

imagen se transformará a un tipo *blob* y se asignará a una variable, para posteriormente ser enviada en el formulario.

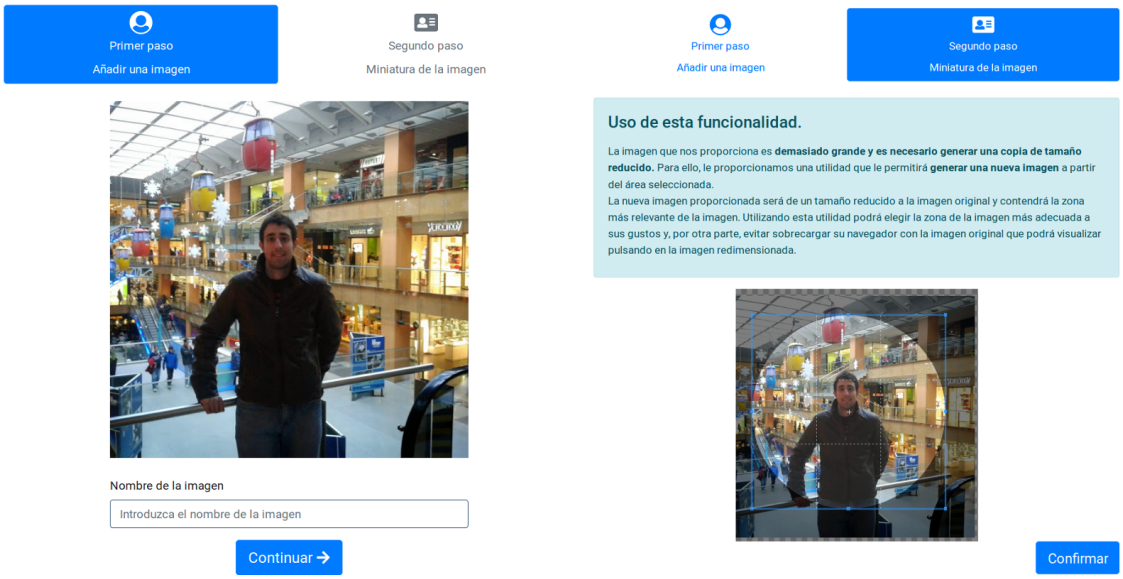

Figura 47: Proceso para generar una miniatura de una imagen de perfil.

Cuando el usuario desee enviar el formulario, se validará comprobando que los campos no estén vacíos. Una vez realizada la comprobación, en la parte del cliente se obtendrá, por una parte, los datos asociados a los campos de texto y por otra, la imagen de perfil y la imagen blob, en caso de existir. Esa información se añadirá en un formulario y se serializará para poder ser enviada por AJAX al recurso de modificar el perfil.

El servidor recibirá la petición y realizará la siguiente secuencia de acciones.

- **Validar el formulario.** Comprobará que los datos sean correctos.
- **Actualizar el perfil.** Actualizará la nueva información en el perfil.
- **Notificar a otros usuarios de la notificación.** Se producirá una vez se actualice correctamente el perfil en la base de datos para que se active la señal *post_save*. En la función, se enviará la nueva información codificada del perfil en formato JSON a todos los usuarios que estén conectados al grupo del perfil del usuario.

La modificación del perfil se realizará en tiempo real y otros usuarios conectados a ese perfil recibirán los nuevos datos del perfil. El lector es posible que se pregunte como se diferencia un perfil de otro usuario y para eso cada perfil tiene un identificador que está insertado como atributo en la etiqueta HTML.

Aparte de eso, es necesario identificar el tipo de atributo del perfil y para ello se inserta de la misma forma que para el identificador de perfil otro atributo. Específicamente, un atributo para cada atributo del modelo perfil.

7.3.3. Diseño e implementación del álbum de fotografías y imágenes de los álbumes

Después de haber diseñado e implementado el perfil de usuario, se implementarán los álbumes de fotografías y las imágenes. Para poder visualizar los álbumes de fotos, se utiliza el nombre de usuario en la dirección URL. Esto permitirá distinguir los álbumes de un usuario de otro usuario.

Por defecto, cuando se crea el usuario, se crean dos álbumes que son el de perfil y portada con un nivel de privacidad público. Estos álbumes no podrán ser eliminados de sistema ni se podrá modificar el nivel de privacidad. En la figura 48, se ilustra el diseño de los álbumes de fotografías.

Figura 48: Diseño de los álbumes de fotos.

Cuando el usuario pulsa el botón *Crear un nuevo álbum de fotografías* será dirigido al recurso que le permitirá añadir un álbum tal y como se muestra en la figura 49. En este recurso no se introducirá ningún nombre de usuario y cualquier usuario que visualice este recurso será para añadir un álbum a su colección de álbumes de fotos.

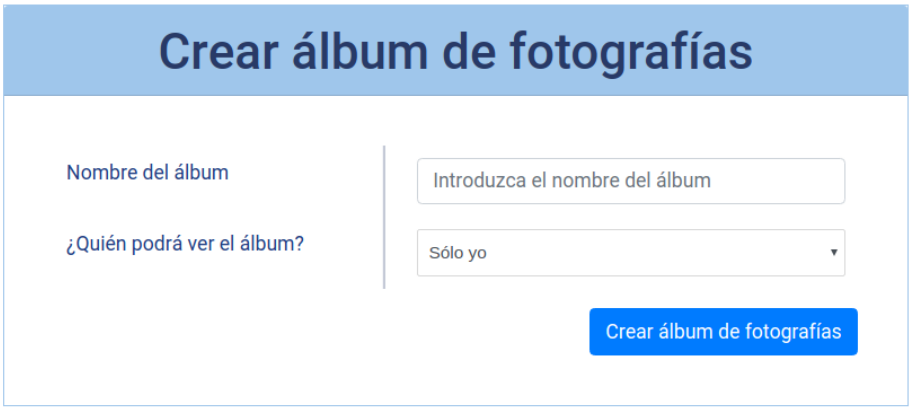

Figura 49: Diseño de crear álbum de fotografías.

```python
def nuevo_album(request, **kwargs):
    album_form = AlbumForm()

    if request.method == 'POST':
        album_form = AlbumForm(request.POST)

        if album_form.is_valid():
            # Genero un nuevo perfil con los nuevos datos
            nuevo_album = album_form.save(commit=False)
            nuevo_album = Album.objects.crear_album(request.user, Album.ALBUM_PERSONAL,
                          nuevo_album.privacidad, nuevo_album.nombre)

            messages.success(request, "Ha creado el álbum %s con éxito." % nuevo_album.nombre)
            return HttpResponseRedirect(reverse('greenbook:imagenes_album', args=[request.user, nuevo_album.id]))

    contexto = {'form_album': album_form}
    return render(request, "GreenBook/nuevoAlbumFotografias.html", contexto)
```

Código 17: Controlador asociado al nuevo álbum.

Una vez creado el álbum, el usuario será dirigido automáticamente al contenido del álbum donde podrá añadir imágenes. Podrá insertar más de una imagen a la lista y antes ser insertadas se comprobarán que sean válidas. Los archivos que no sean imágenes o no cumplan las limitaciones de tamaño no se subirán y serán descartadas.

Cuando se hayan insertado, todavía no se habrán subido al servidor. Previamente será necesario especificar una imagen de miniatura y el nombre de la imagen.

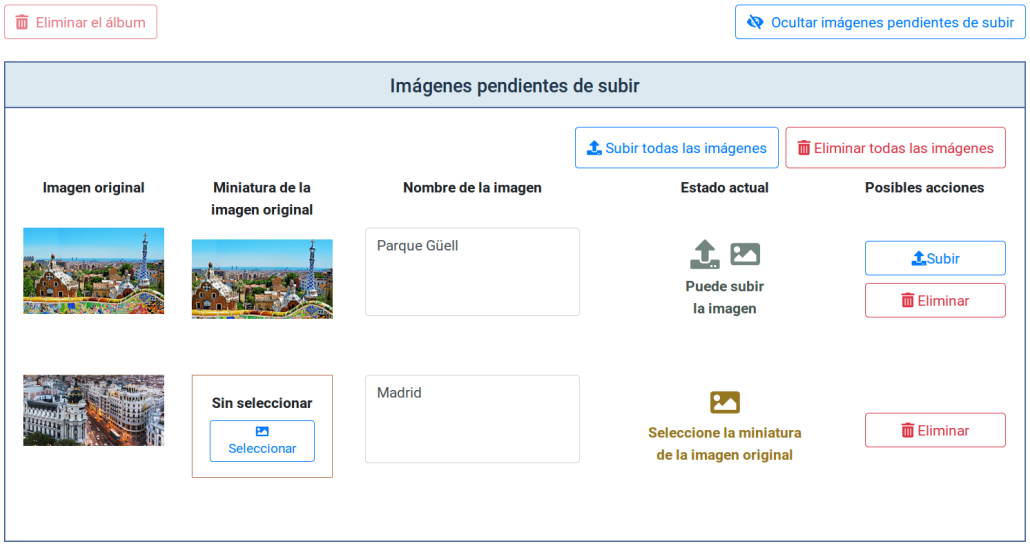

Figura 50: Imágenes pendientes de subir.

7.3.4. Diseño e implementación de la búsqueda de amigos

Para que los usuarios pudieran realizar búsquedas y filtrar a los amigos se diseñó este recurso. Inicialmente, cuando el usuario entra en el recurso aparece en la sección resultados, todos los amigos. Para poder diferenciar los amigos de un usuario de otro, la ruta tiene un argumento que identifica al usuario del cual se mostrarán sus amigos.

Cuando el usuario pulse el botón *Buscar*, antes de enviar la petición AJAX, se obtendrán valores del nombre de la persona y población para posteriormente enviarlos al servidor. A partir de los valores enviados, el servidor devolverá un JSON formado por una lista de plantillas HTML de los usuarios que cumplen los criterios indicados. Esas plantillas todavía no estarán en el documento y serán necesarias añadirlas al contenedor *Resultados*. Para ello, se utilizará la librería jQuery para iterar en los datos e insertarlos.

Un usuario puede necesitar visualizar todos los amigos de un usuario en vez de refrescar la página. Si tiene esa necesidad, puede no introducir ningún nombre y pulsar el botón *Buscar*.

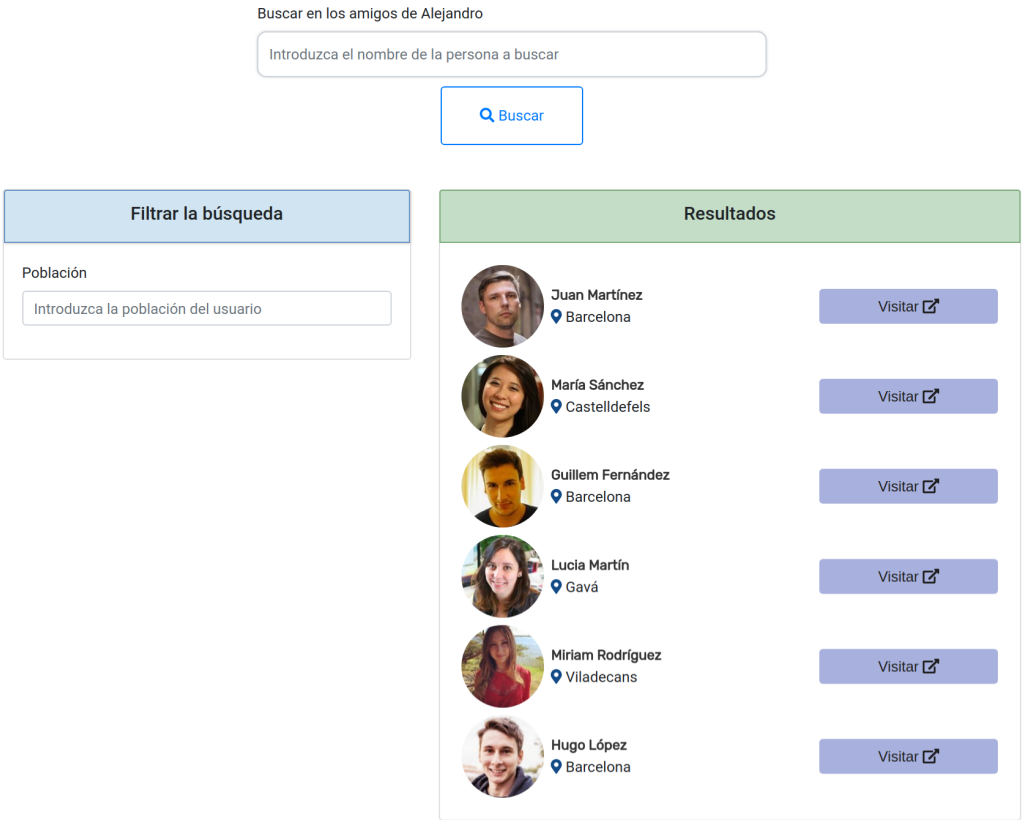

Figura 51: Diseño de la pestaña amigos.

Aparte de buscar en los amigos de un usuario, los usuarios pueden buscar a otros usuarios que estén registrados en la red social. Para buscar usuarios se utilizó la tecnología AJAX y el plugin typeahead [21].

La secuencia de funcionamiento se iniciará en el usuario, cuando escriba en el campo de texto un mínimo de caracteres. Para conocer cuando el usuario quiere realizar la búsqueda, se activará un contador limitado a 500 milisegundos (línea 5).

Cuando se active ese contador, el sistema obtendrá el texto del cuadro y acto seguido la guardará en la variable *query* para posteriormente enviar la consulta al servidor y obtener una estructura JSON con los resultados.

Pueden producirse dos escenarios, que hayan resultados o no. Si la búsqueda tiene resultados, se activará la función *template* para cada elemento de la estructura. Un ejemplo de este escenario se muestra en la figura 53.

Si la búsqueda no genera resultados se activará la función *emptyTemplate* que tendrá una plantilla previamente definida e informará al usuario que no se encontraron resultados. Un ejemplo de esta plantilla, se muestra en la figura 52.

```
1  $.typeahead({
2          input: '.js-typeahead',
3          minLength: 1,  // Número de caracteres necesarios para buscar
4          dynamic: true,
5          delay: 500,    // Milisegundos para procesar la búsqueda
6          order: "asc",  // Orden de los resultado de forma ascendente
7
8          template: function (query, item) {},
9          emptyTemplate: function (query) {},
10         source: {
11                 Usuarios: {
12                         display: "nombre",
13                         ajax: {
14                                 url: '/api/perfil_usuario/getUsuariosByName/{{query}}/',
15                                 dataType: "json",
16                                 path: "data"
17                         }
18                 }
19         },
20 });
```

Código 18: Configuración de la búsqueda de usuarios.

Cuando el servidor envíe la consulta activará la función *template*, si se encontró algún usuario o *emptyTemplate* en caso de no haber encontrado algún usuario. En ambos casos, esas dos funciones devolverán una plantilla en formato HTML. Se muestran dos plantillas, la primera corresponde a una petición sin resultados (véase figura 52) y la segunda a una consulta con resultados (véase figura 53).

[21] http://www.runningcoder.org/jquerytypeahead/

Figura 52: Sin resultados en la búsqueda.

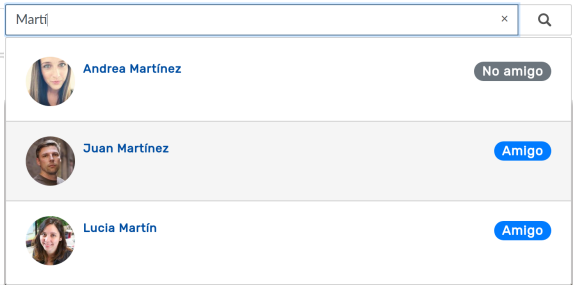

Figura 53: Búsqueda de usuarios.

Por defecto cuando se pulsa el botón buscar del autocompletado, no realiza ninguna acción ya que la petición se efectúa utilizando AJAX. Internamente el componente es un formulario HTTP sin ninguna ruta asociada y cuando el usuario pulsa el botón buscar no sucede ninguna acción.

Para implementar esta acción sin utilizar AJAX, en la parte del servidor se definió un controlador y éste se asoció a una ruta de Django con la finalidad de poder recibir las peticiones de búsqueda del formulario.

Figura 54: Controlador que muestra los usuarios.

7.4. Cuarto sprint

El último sprint se centrará en diseñar e implementar los vídeos y la interacción con otros usuarios. Además se implementará el sistema de notificación de solicitudes de amistad y notificaciones generales y por último las preferencias del usuario.

7.4.1. Diseño e implementación de los vídeos

Aparte de las imágenes, el usuario puede subir vídeos a la aplicación y otros usuarios pueden visualizarlos. Para ello, se definió una pestaña en la que agrupa, por una parte el proceso de subir vídeos y por otra los vídeos subidos.

Un vídeo se puede subir utilizando el componente drag and drop definido en HTML5 o desde el explorador de archivos. Cuando el usuario quiere subir un vídeo, el sistema valida que este cumpla una serie de características:

- **Tamaño del vídeo.** Un vídeo podrá tener un tamaño máximo de hasta 50MB.
- **Extensiones permitidas.** Las extensiones permitidas serán: mpeg, webm y ogg.

Las validaciones se realizan en el cliente como en el servidor. En el cliente, si se detecta que un vídeo no cumple con las características anteriores, el sistema informa al usuario y descarta el vídeo no añadiéndolo a la lista de vídeos pendientes de subir. Por otra parte, en el servidor se realizan las mismas comprobaciones y en caso de error se informará al usuario utilizando un JSON.

Para implementar las comprobaciones, se definió la clase *VideoValidator* que extiende de *BaseValidator* para implementar las validaciones a nivel de formulario y no en el controlador. Esta clase implementa una validación más rigurosa verificando que verdaderamente el vídeo cumple con las características del servidor para ser aceptado. La validación sigue un proceso secuencial que se detalla a continuación:

- **Validar la extensión del archivo.** El primer paso consiste en obtener la extensión del archivo y comprobar que está entre las extensiones válidas. En caso que no se encuentre o no se pueda obtener, el proceso se detiene y se retorna un error de validación con el error encontrado.
- **Verificar el tipo MIME.** El segundo paso consiste en obtener el tipo MIME del vídeo y comprobar que esté entre los permitidos.
- **Verificar las dimensiones del archivo.** El último paso consiste en verificar que el tamaño del archivo no supere al establecido por la aplicación.

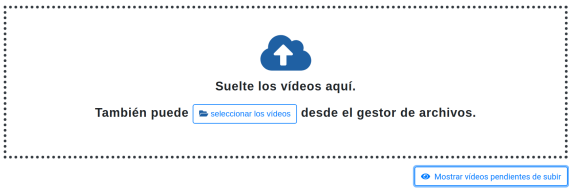

Figura 55: Arrastrar y soltar para subir vídeos.

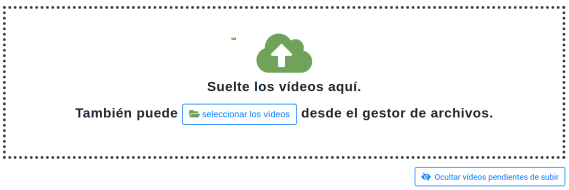

Figura 56: Arrastrar y soltar aceptando el vídeo introducido.

Si la comprobación en el cliente es válida, se añadirá el archivo a un diccionario y se mostrará una entrada en la lista de vídeos pendientes de subir tal y como se muestra en la figura 57.

Para mostrar la entrada, se parte de una plantilla base HTML que se clona y se modifica de forma dinámica añadiendo el vídeo y la posición en el diccionario. Cada plantilla base dispone de estados para proporcionar una retroalimentación del estado en el que se encuentra el vídeo.

Cada estado contiene un icono vectorial que puede ser animado tal y como sucede en el estado de subiendo el vídeo y un texto para proporcionar información del estado actual.

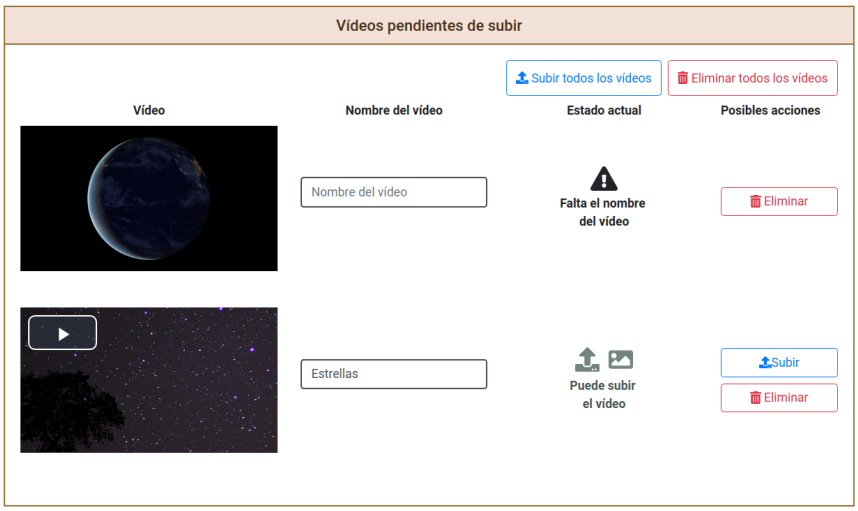

Figura 57: Vídeos pendientes de subir.

Para subir un vídeo, se puede optar por subir todos los vídeos cuyo caso solo se subirán aquellos que tengan un nombre asociado o manualmente, pulsando el botón *Subir* o el icono del estado actual.

La acción provocará que se ejecute un AJAX que tendrá como campos de información, el nombre del vídeo y el archivo. Estos datos se enviarán al servidor en formato JSON al servidor. El servidor, validará que los campos sean válidos y en caso que sean válidos guardará el vídeo en la base de datos.

En caso que el usuario pulse el botón *Eliminar vídeo*, se realizará el mismo procedimiento pero está vez en vez la petición tendrá como dato el identificador del vídeo.

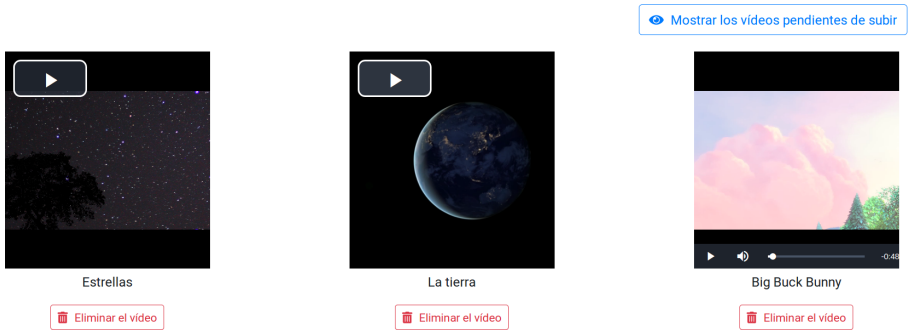

Figura 58: Vídeos subidos.

7.4.2. Solicitudes de amistad

Un usuario que quiera ser amigo de otro usuario deberá enviar una solicitud de amistad yendo al muro del usuario y pulsar el botón *Enviar solicitud de amistad* situado en la figura 59 en el lado inferior izquierdo de la localidad del usuario.

La pulsación del botón enviará una petición asíncrona al servidor y éste comprobará que no exista una solicitud de amistad previa. Si la verificación es válida creará y persistirá un objeto *SolicitudAmistad* que tendrá una referencia al usuario emisor y receptor.

La creación del objeto generará un evento *post_save* que se utilizará para informar la solicitud de amistad al usuario receptor.

Despues de crear el objeto, la solicitud existirá y estará pendiente de ser aceptada por el otro usuario o cancelada. Para saber cuando hay una solicitud de amistad, se consultará en la base de datos si el objeto existe y si son amigos y esa información se enviará al sistema de plantillas donde realizará la gestionar para mostrar los botones. Técnicamente cuando el sistema de plantillas procese la plantilla realizará el siguiente algoritmo 2.

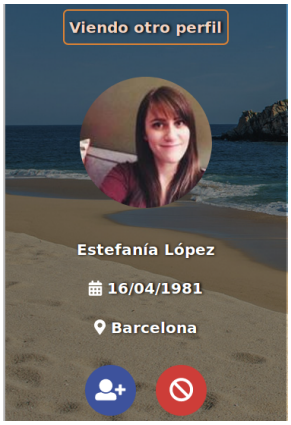

Figura 59: Enviar solicitud de amistad.

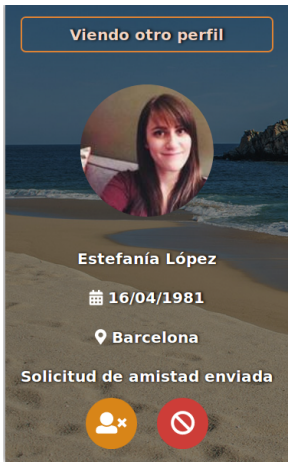

Figura 60: Solicitud de amistad pendiente de aceptar.

El usuario que recibe la solicitud de amistad podrá aceptarla o eliminarla y para ello deberá ir al muro del usuario que emitió la solicitud. Visualmente la cabecera del perfil mostrará tres botones y cada botón tendrá un evento asociado para enviar una petición asíncrona al servidor. Un ejemplo del diseño que

vería el usuario accediera a la cabecera del perfil se ilustra la figura 61.

Algoritmo 2: Comprobaciones para mostrar los botones de la solicitud de amistad o amistad creada.

```
Obtener de la base de datos el usuario, la solicitud de amistad y la amistad;
si no está viendo a otro usuario entonces
    si no son amigos entonces
        si existe la solicitud de amistad entonces
            si la solicitud de amistad va dirigida al usuario que ha iniciado sesión entonces
                Mostrar los botones de aceptar y rechazar la solicitud de amistad;
            en otro caso
                Mostrar el botón de cancelar la solicitud de amistad;
            fin
        en otro caso
            Mostrar el botón de enviar una solicitud de amistad;
        fin
    en otro caso
        Mostrar el botón de eliminar la amistad;
    fin
    Mostrar el botón de bloquear al usuario;
fin
```

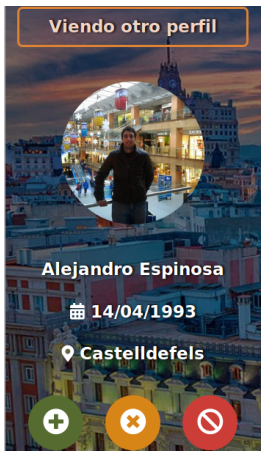

Figura 61: Aceptar o rechazar una solicitud de amistad.

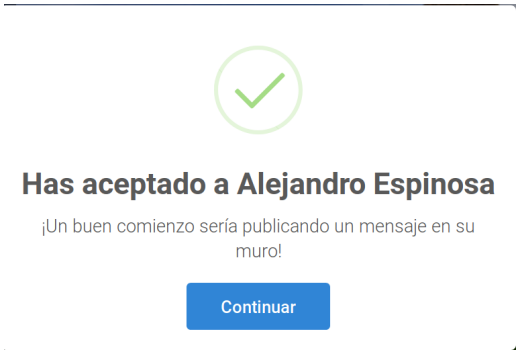

Figura 62: Ventana modal cuando se acepta la solicitud de amistad.

7.4.3. Notificaciones de solicitudes de amistad y generales

Pueden existir notificaciones o solicitudes de amistad y que el usuario quiera ser notificado en el momento de producirse. Para ello, cada usuario autentificado se subscribirá a un canal, uno para las notificaciones y otro para las solicitudes de amistad.

Cada vez que el usuario cambie de recurso, para obtener las notificaciones generales y solicitudes de amistad, se enviará una petición AJAX. El servidor devolverá un JSON que tendrá una lista de plantillas HTML para introducir en el contenedor que corresponda dependiendo de si la plantilla es una solicitud de amistad o una notificación general. Para ilustrar este tipo de plantillas, se muestra en la figura 63 las solicitudes que tiene el usuario pendientes de aceptar y en la figura 64 las notificaciones generales.

Por defecto, las plantillas no tienen ningún evento asociado. Para asociar los eventos se utilizó jQuery con la finalidad de buscar los botones y enlazar el evento. Aparte del evento *onClick* asociado a los botones, cada entrada de las notificaciones generales tiene asociada el evento *mouseover* para que cuando el cursor este en el interior de la entrada, envíe una petición asíncrona para marcar la notificación como vista.

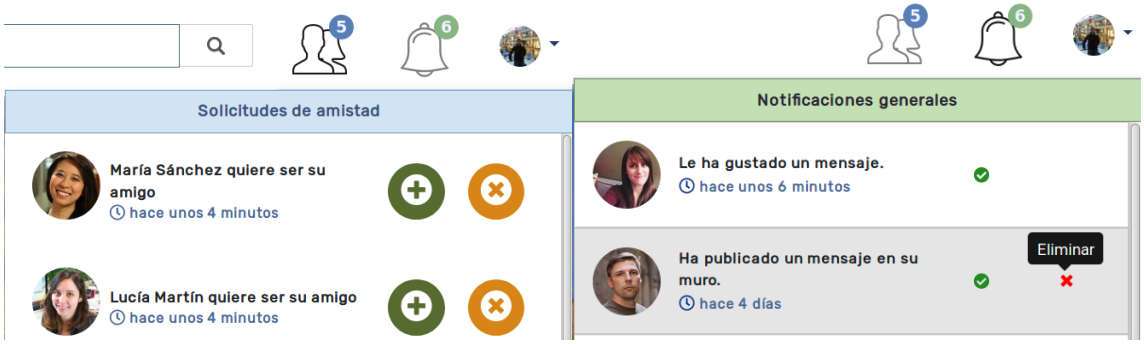

Figura 63: Solicitudes de amistad pendientes de aceptar

Figura 64: Diseño de las notificaciones generales.

Con la finalidad de proporcionar una retroalimentación del estado actual de la petición asíncrona y del número de notificaciones que tiene el usuario se adjuntan dos figuras.

La primera figura 65 ilustra el escenario de no tener ninguna notificación y la segunda figura 66 cuando se envía una petición AJAX y el cliente está esperando la respuesta del servidor.

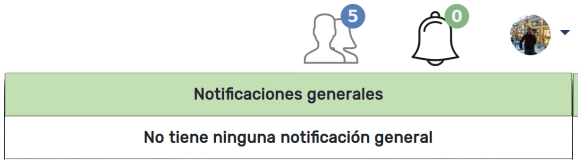

Figura 65: Sin notificaciones generales.

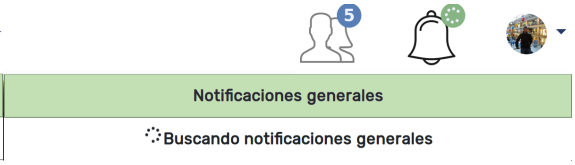

Figura 66: Retroalimentación cuando se están buscando notificaciones generales.

7.4.4. Preferencias generales

En la figura 67 se muestran las posibles acciones que un usuario podrá realizar en sus preferencias: cambiar la contraseña y eliminar su cuenta. La opción de cambiar contraseña aparece ya que el usuario creó una cuenta manualmente.

Aparte de esas opciones, un usuario que se una a la red social a partir de otra cuenta social no tendrá por defecto una contraseña y, en vez de aparecer la opción de cambiar la contraseña, aparecerá un formulario para establecer la contraseña.

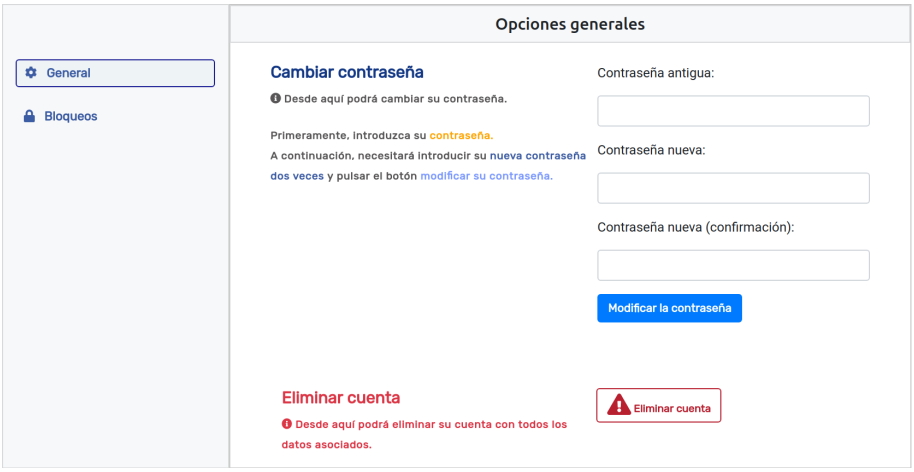

Figura 67: Preferencias del usuario.

La opción de eliminar cuenta permitirá eliminar toda la información del usuario. Para ello se abrirá una ventana modal en la que pedirá introducir el correo electrónico siendo necesario para poder contrastarlo con el correo que el servidor tiene.

Durante el proceso de eliminación pueden producirse errores de validación o internos por no detectar el correo electrónico del usuario. En particular se muestran los dos errores que pueden producirse:

- **Correo electrónico no introducido.** Se produce por no haber introducido un correo electrónico o un formato erronio. Se ilustra en la figura 68, la retroalimentación que visualizará el usuario.
- **Correo electrónico no detectado**. Se produce cuando el servidor detecta que el correo electrónico no coincide con el del usuario. En la figura 69 se ilusta el mensaje de error.

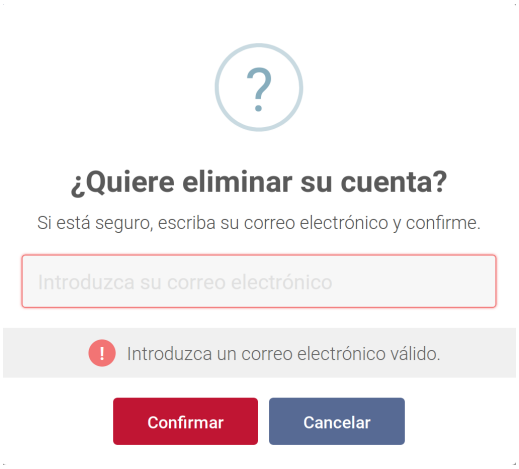

Figura 68: Validación del correo electrónico en el proceso de eliminar la cuenta.

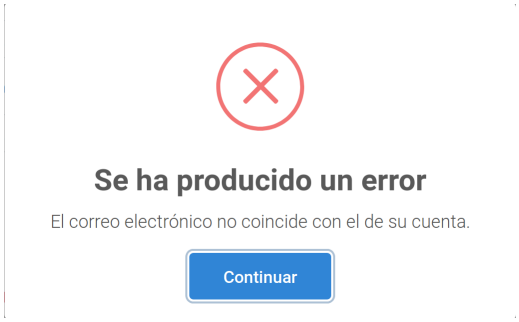

Figura 69: Correo electrónico no encontrado en el proceso de eliminar la cuenta.

Con las figuras que se han presentado anteriormente se han mostrado los errores que pueden producirse durante el proceso de eliminación de la cuenta. Si no se produjera ningún error, el sistema eliminaría el modelo asociado al usuario autentificado y todas las relaciones con las otras entidades. Finalmente respondería con una respuesta JSON indicando que el proceso se completó satisfactoriamente.

Dependiendo de las relaciones con otras entidades y de la carga del servidor, el proceso de eliminación de la cuenta puede demorarse unos segundos y para informar al usuario que necesita esperar se ilustra el figura 70.

Cuando el usuario recibe la respuesta satisfactoria indicando que se ha eliminado correctamente la cuenta entonces redirige al usuario al la página principal tal y como se muestra en la figura 71.

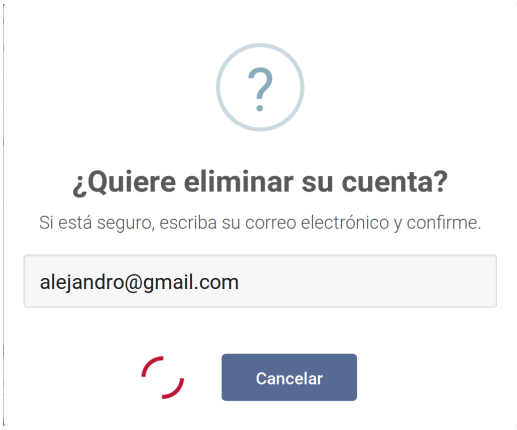

Figura 70: Validación del correo electrónico en el proceso de eliminar la cuenta.

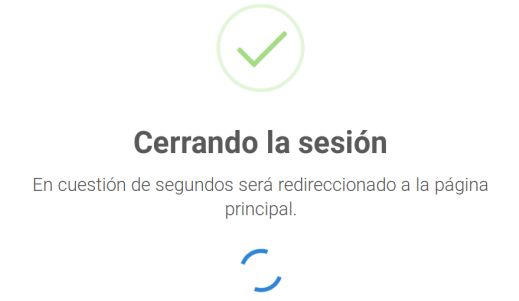

Figura 71: Proceso de eliminación de la cuenta completado.

7.4.5. Bloqueo y desbloqueo de usuarios

Una funcionalidad que se implementó fue bloquear y desbloquear usuarios. Para bloquear a un usuario, el usuario deberá visitar, por ejemplo el muro del usuario y pulsar el botón bloquear.

Cuando el usuario pulse el botón de bloquear activará una ventana modal tal y como se muestra en la figura 72 con la finalidad de comprobar que no se trata de un error. La confirmación generará una petición AJAX que tendrá como parámetro, el identificador del usuario a bloquear y se enviará al controlador de bloquear a un usuario.

El servidor verificará que el usuario a bloquear exista y que no haya un bloqueo previo entre los dos usuarios. Si las comprobaciones son correctas, generará un objeto *Bloqueo* que tendrá a los dos usuarios. Posteriormente persistirá el objeto en la base de datos y por último informará que se ha bloqueado correctamente al usuario (figura 73).

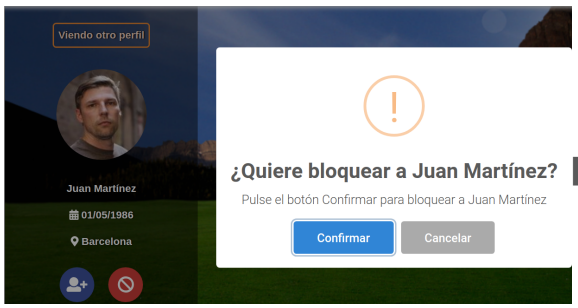

Figura 72: Bloquear a un usuario.

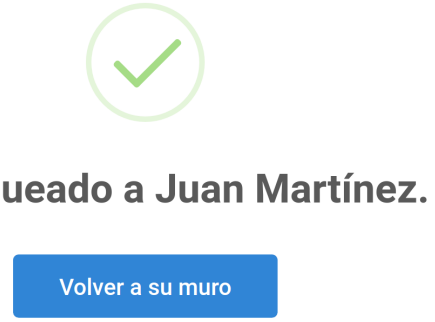

Figura 73: Bloqueo realizado con éxito.

Una vez realizado el bloqueo, los usuarios no podrán acceder a recursos del otro usuario y si intentan acceder se informará que el usuario ha sido bloqueado.

Todos los usuarios bloqueados del usuario estarán en la sección Bloqueos y podrán ser desbloqueados pulsando el botón Desbloquear tal y como se ilustra en la figura 74. La pulsación enviará una petición AJAX con el identificador del objeto Bloqueo y el servidor comprobará que el usuario que generó el bloqueo coincida con el usuario autentificado para poder eliminar el objeto.

Figura 74: Usuarios bloqueados.

8. Conclusiones

En este último capítulo se hará una recapitulación de los objetivos propuestos inicialmente y los resultados obtenidos. Se quiere poner énfasis en el hecho que el objetivo de este proyecto era crear una aplicación web que permitiera a las personas comunicarse en tiempo real y que permitiera el intercambio de imágenes e vídeos entre usuarios. Para que fuera una red social pensada en el usuario, se enfocó en el diseño centrado en el usuario para solucionar problemas no visibles por el diseñador pero presentes en la aplicación.

Para considerar si el objetivo principal se cumplió se recordarán los objetivos específicos haciendo una retrospectiva de los objetivos de la sección 1.2.

- *Cada usuario tendrá un perfil de usuario que será visible por otros usuarios y en el cual se incluirán campos como el nombre, apellidos, fecha de nacimiento, una imagen de perfil y portada. Esos campos serán modificables y, una vez el usuario haga alguna modificación en su perfil, otros usuarios podrán recibir esas modificaciones en tiempo real.*

  Se cumplió este objetivo especifico creando, para cada usuario, un perfil que puede modificar y otros usuarios pueden ver las modificaciones de ese usuario en tiempo real.

- *Cada usuario podrá crear, visualizar y eliminar mensajes en su propio muro. Asimismo, se dará la funcionalidad que otros usuarios puedan recibir los cambios realizados en el muro justo en el momento de ser efectuados.*

  La aplicación cumple esta funcionalidad específica permitiendo que el usuario pueda crear publicaciones con un nivel de privacidad tanto en su muro como en los amigos, visualizar mensajes de amigos y propios. Además puede eliminar mensajes publicados en su muro.

- *Los usuarios podrán crear álbumes de fotos y eliminarlos.*

  El objetivo se cumplió correctamente. Los usuarios pueden agrupar sus imágenes en álbumes y eliminar todas las imágenes cuando se elimine el álbum o específicamente en el álbum.

- *Los mensajes y álbumes de fotografías tendrán un nivel de privacidad dependiendo de la visibilidad que el usuario seleccione.*

  Se cumplió este objetivo definiendo en cada modelo Álbum y mensaje, un atributo para controlar la privacidad.

- *El usuario podrá interactuar con otros usuarios enviando solicitudes de amistad para que el otro usuario pueda aceptarlas o rechazarlas. Además el otro usuario recibirá un correo que podrá utilizar para aceptar o rechazar la solicitud.*

  Se cumplió este objetivo especifico diseño e implementando por una parte, una interfaz agradable y fácil de usar por los usuarios y por otra, una lógica interna para gestionar este tipo de interacción en el sistema.

- *El usuario podrá buscar en sus amigos y de los de otros y a otros usuarios.*

  Gracias a este objetivo específico, los usuarios pueden buscar a otros usuarios e ir a otros muros. Se utilizó la tecnología y un componente de autocompletar en la parte del cliente.

- *El usuario podrá bloquear y desbloquear a otros usuarios que previamente haya bloqueado.*

  La red social cumple este objetivo permitiendo bloquear y desbloquear a usuarios.

- *El usuario podrá indicar "Me gusta" en sus publicaciones o a las de otros usuarios (siempre que sean amigos o sea pública).*

  La red social integra este objetivo específico permitiendo indicar en publicaciones *"Me gusta"* y poder ver a las personas que le gusta la publicación.

- *Dar la capacidad a los usuarios para buscar a otros usuarios.*

  A partir de esta funcionalidad implementada, los usuarios pueden buscar a otros usuarios e ir a su muro. Para cumplir este objetivo especifico se utilizó en el lado del cliente un componente de autocompletar de terceras personas.

- *Disponer de un acceso rápido y en tiempo real de los eventos que sucedan en la aplicación entre las cuales se incluyen publicar un mensaje, me gusta un mensaje y otras destinadas a conocer cuando el usuario recibe una solicitud de amistad.*

  Gracias a esta funcionalidad implementada, los usuarios pueden ser notificados en tiempo real de eventos que sucedan en la aplicación. Se utilizó la tecnología WebSockets para implementar el objetivo específico.

Al haberse cumplido los objetivos específicos se puede afirmar que se ha cumplido el objetivo principal.

8.1. Conclusiones personales

En general, la realización de este proyecto me ha parecido un trabajo muy interesante y variado. He tenido la posibilidad de poner en práctica los conocimientos adquiridos a lo largo del grado.

Me ha servido para aplicar los conceptos, no sólo de una asignatura en concreto, sino de varias de ellas y ver la relación entre las mismas. Esto es algo que no había podido hacer hasta este momento, ya que en otros trabajos realizados durante el transcurso del grado se centraban en la asignatura para la que se desarrollaban o una parte de ella. Además, en ciertas asignaturas, los proyectos se realizaban en equipo y en este proyecto al haberlo realizado sólo he podido apreciar la dificultad de una correcta planificación.

Este trabajo también ha sido el de mayor envergadura que he llevado a cabo. No había realizado un trabajo de tanta magnitud e importancia. Esto ha supuesto todo un reto para mi, sobre todo a la hora de escribir la memoria que ha sido la parte más laboriosa.

9. Líneas futuras de desarrollo

Después de haber realizado el objetivo principal y haber implementado las mejoras propuestas por los usuarios, aún así siempre es posible añadir mejoras y nuevas funcionalidades.

Una mejora aplicable en el muro de usuario sería dar la capacidad de poder añadir emoticonos, vídeos y imágenes en las publicaciones.

Otra mejora sería, en el proceso de subir la imagen, tener la opción de modificarla para añadirle efectos, tal y como sucede en Facebook o Instagram. Se resalta está funcionalidad como una idea que podría incrementar el valor de la aplicación, atrayendo a nuevos usuarios.

Paralelamente, la misma idea se podría extrapolar a los vídeos. También habría la posibilidad de realizar vídeos en directo. Se podría utilizar WebSockets para implementar está funcionalidad.

Por último, se podría implementar un sistema de mensajería privada (sin que hubiera constancia en el muro). Además, esa implementación podría ampliarse pudiendo añadir imágenes o vídeos.

Referencias

[1] Antonio Moreno Muñoz. «Diseño centrado en el usuario (I). Introducción». En: *El Prof. la Inf.* 12.1 (2003), págs. 52-54. ISSN: 1386-6710. DOI: 10.1076/epri.12.1.52.19713. URL: https://openlibra.com/es/book/diseno-centrado-en-el-usuario (visitado 10-10-2017).

[2] Muriel Garreta, Domingo Enric y Mor Pera. «Diseño centrado en el usuario». En: (), pág. 26. URL: https://www.exabyteinformatica.com/uoc/Informatica/Interaccion_persona_ordenador/Interaccion_persona_ordenador_(Modulo_3).pdf (visitado 25-10-2017).

[3] Stefan Blomkvist. «Persona – an overview». En: (). URL: https://www.it.uu.se/edu/course/homepage/hcidist/v04/Persona-overview.pdf (visitado 05-11-2017).

[4] Fernando Finelli. *10 reglas heurísticas de usabilidad de Jakob Nielsen*. 2011. URL: http://www.braintive.com/10-reglas-heuristicas-de-usabilidad-de-jakob-nielsen/ (visitado 29-10-2017).

A. Anexo

A.1. Documento informado

Este documento informado es para el trabajo de fin de grado “Diseño e implementación de una red social en tiempo real basada en el diseño centrado en el usuario“. Este proyecto será llevado a cabo por Alejandro Espinosa, bajo la supervisión de la Dra. Mireia Ribera.

Estimado/a participante:

Gracias por su interés en mi trabajo de fin de grado. Este documento describe lo que se le pedirá que realice en el proyecto y también detalla los permisos necesarios. Por favor, léalo detenidamente y posteriormente firme en la parte inferior si comprende y acepta las condiciones de este estudio. Si tiene alguna pregunta, no dude en hacerla.

Al firmar este documento se conceden los siguientes permisos:

1. **Acepta participar en una entrevista**. La entrevista durará entre 15 minutos y un máximo de 30 minutos y se llevará a cabo en Barcelona o vía Skype.
2. Da permiso a Alejandro Espinosa para que **tome notas de las observaciones** realizadas mientras usa la aplicación sobre su interacción con la interfaz y sus comentarios.
3. Da permiso para que se **utilicen esas notas como fuente de información** para llevar a cabo su proyecto.

Su participación en este estudio es voluntaria. En cualquier momento puede abandonar el estudio. Toda la información obtenida estará restringida a este proyecto y no se proporcionará ninguna información sobre su identidad.

Le agradecemos su ayuda. Es importante que comprenda que este proyecto no sería posible sin su participación. Por favor escriba la fecha y firme en la parte inferior de esta página para indicar que comprende y acepta las condiciones de este estudio.

Muchas gracias por su participación.

Fecha de realización: _____

Nombre del entrevistado	**Nombre del entrevistador**
_____	_____
Firma	*Firma*
_____	_____

A.2. Configuración de Nginx

Para comprender la configuración del servidor, se muestra el fichero de configuración asociado a la configuración de la red social.

Analizando en detalle la configuración las opciones añadidas, se puede visualizar en la línea 2 el puerto de escucha y la línea de abajo limita a 200MB el tamaño de fichero que un usuario podrá subir al servidor.

Con la configuración actual se ha configurado el servidor pero es necesario localizar donde se encontrarán los iconos e imágenes internas de la aplicación (líneas 5-7) y donde se guardarán los archivos que el usuario subirá a la aplicación (líneas 9-12).

El último bloque (líneas 13-27) indican el proxy inverso al servidor Daphne. En ese bloque se indica donde se encontrará el servidor Daphne además de configurar el servidor para que enrute peticiones tanto de WebSockets como HTTP.

```
1  server {
2          listen 80;
3          client_max_body_size 200M;
4
5          location /static {
6          alias /home/alejandro/Escritorio/Greenbook-TFG/GreenBook/static;
7          }
8
9          location /media {
10         alias /home/alejandro/Escritorio/Greenbook-TFG/media;
11         }
12
13         location / {
14         proxy_pass http://0.0.0.0:8001;
15         proxy_http_version 1.1;
16         proxy_set_header Upgrade $http_upgrade;
17         proxy_set_header Connection "upgrade";
18
19         proxy_redirect     off;
20         proxy_set_header   Host $host;
21         proxy_set_header   X-Real-IP $remote_addr;
22         proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
23         proxy_set_header   X-Forwarded-Host $server_name;
24
25         }
26
27 }
```

Código 19: Configuración del servidor Nginx.

A.3. Test de satisfacción

Pregunta	Muy en desacuerdo —— Muy de acuerdo
1. Piensa que le gustaría utilizar este sistema frecuentemente.	☐—☐—☐—☐—☐
2. Encuentro que el sistema es innecesariamente complejo.	☐—☐—☐—☐—☐
3. Piensa que el sistema era fácil de utilizar.	☐—☐—☐—☐—☐
4. Piensa que necesitaré el apoyo técnico para ser capaz de utilizar el sistema.	☐—☐—☐—☐—☐
5. Piensa que hay mucha inconsistencia en el sistema.	☐—☐—☐—☐—☐
6. Ha necesitado aprender muchas cosas antes de poder utilizar el sistema.	☐—☐—☐—☐—☐
7. Cree que la mayoría de gente aprenderá a usar la aplicación rápidamente.	☐—☐—☐—☐—☐
8. Encuentra que el sistema era incómodo de utilizar.	☐—☐—☐—☐—☐
9. Se ha sentido seguro/a utilizando el sistema.	☐—☐—☐—☐—☐

A.4. Requisitos funcionales del usuario autentificado

En esta sección se incluyen los requerimientos de los usuarios que han iniciado sesión.

A.4.1. Mensaje

A continuación se muestran las historias de usuarios que corresponden al mensaje. Los usuarios podrán publicar un mensaje con un nivel de privacidad, eliminar sus propios mensajes publicados en su muro e indicar ”Me gusta” y ’No me gusta” un mensaje de un usuario que pueda visualizar.

Historia de usuario	**Publicar un mensaje con un nivel de privacidad concreto.**
Descripción	Como usuario autentificado, deseo publicar un mensaje para que otros usuarios puedan visualizarlo.
	Criterios de aceptación
Nombre del criterio	**Descripción**
Validación de los campos	Cuando la caja de texto no tenga ningún carácter o el campo visibilidad sea inválido y el usuario pulse en el botón publicar entonces el sistema notificará visualmente al usuario del error detectado.
Muro no amigo	Cuando el usuario autentificado visite el muro de un usuario entonces no podrá publicar ningún mensaje.
Muro amigo	Cuando el usuario autentificado visite el muro de un amigo entonces podrá publicar un mensaje con la privacidad amigos.
Muro propio usuario	Cuando el usuario autentificado visite su muro entonces podrá publicar un mensaje con la privacidad amigos, público y sólo yo.
Notificar publicación	Cuando el usuario autentificado publique el mensaje, el sistema notificará a los usuarios conectados al muro de ese usuario que se ha creado una nueva publicación.
Eliminar el botón que permite suprimir un mensaje en usuarios no propietarios del muro	Cuando el usuario reciba un mensaje, internamente el sistema comprobará si es el propietario del muro y en caso que no sea eliminará el botón *Eliminar*.

Cuadro 8: HU-06 – Publicar un mensaje con un nivel de privacidad concreto.

Historia de usuario	**Eliminar un mensaje.**
Descripción	Como usuario autentificado, deseo eliminar mensajes para eliminarlos de la base de datos y que los usuarios no puedan visualizarlos.
	Criterios de aceptación
Nombre del criterio	**Descripción**
Comprobaciones generales previas	Cuando el usuario pulse en el botón de eliminar el mensaje, el sistema realizará las comprobaciones entre las cuales se destacan verificar que el identificador exista, sea entero además de que el mensaje exista en la base de datos informando en caso de algún error.
Eliminar mensaje muro ajeno	Cuando el usuario autentificado elimine un mensaje de otro muro, el sistema informará al usuario que no puede realizar esa acción.
Notificar eliminación mensaje	Cuando el usuario autentificado elimine un mensaje, el sistema notificará a los usuarios conectados al muro de ese usuario que se ha eliminado el mensaje.

Cuadro 9: HU-07 – Eliminar un mensaje.

Historia de usuario	**Me gusta y no me gusta un mensaje.**
Descripción	Como usuario autentificado, deseo indicar que un mensaje me gusta o no me gusta.
	Criterios de aceptación
Nombre del criterio	**Descripción**
Comprobaciones generales previas	Cuando el usuario pulse el botón de gusta el mensaje, el sistema realizará las comprobaciones entre las cuales se destacan verificar que el identificador exista, sea entero además de que el mensaje exista en la base de datos informando en caso de algún error.
Indicar un "Me gusta"	Cuando el usuario pulse el botón de gusta y el mensaje aún no le haya gustado, el sistema guardará la reacción del usuario en la base de datos e informará con un mensaje de éxito.
Indicar un "No me gusta"	Cuando el usuario pulse el botón de gusta y el mensaje ya le guste, el sistema eliminara la reacción del usuario en la base de datos e informará con un mensaje de éxito.

Cuadro 10: HU-08 – Indicar "Me gusta" o "No me gusta" un mensaje.

Historia de usuario	**Visualizar los gusta de un mensaje.**
Descripción	Como usuario autentificado, deseo visualizar las personas que le gusta un mensaje.
	Criterios de aceptación
Nombre del criterio	**Descripción**
Validación general	Cuando el usuario quiera visualizar los 'Me gusta" de un mensaje y el identificador del mensaje no sea válido, el sistema informará del error producido.
Validación general	Cuando el usuario quiera visualizar los 'Me gusta" de un mensaje y el identificador del mensaje no sea válido, el sistema informará del error producido.
"Me gusta" de un mensaje no visible	Cuando el usuario quiera visualizar los 'Me gusta" de un mensaje y ese mensaje no vea visible por el usuario debido a su nivel de visibilidad, el sistema informará del error producido.

Cuadro 11: HU-09 – Visualizar los "Me gusta" de un mensaje.

A.4.2. Perfil de usuario

Después de haber definidos los requisitos funcionales del mensaje, se definirán los historias de usuario del perfil.

Historia de usuario	**Modificar el perfil de usuario.**
Descripción	Como usuario autentificado, deseo poder modificar los campos del perfil para cambiar algún campo de perfil.
	Criterios de aceptación
Nombre del criterio	**Descripción**
Validación general	Cuando el usuario autentificado envíe el formulario, entonces el sistema validará que los datos introducidos sean válidos y mostrará errores en los campos no válidos.
Notificar modificación perfil	Cuando el usuario modifique su perfil, el sistema informará enviando a los usuarios que estén conectados en el grupo del usuario para recibir la notificación.

Cuadro 12: HU-10 – Modificar el perfil de usuario.

Historia de usuario	**Modificar la imagen de perfil.**
Descripción	Como usuario autentificado, deseo poder modificar la imagen de perfil.
	Criterios de aceptación
Nombre del criterio	**Descripción**
Seleccionar imagen miniatura	Cuando el usuario modifique la imagen de perfil, el sistema abrirá una ventana modal para recortar la imagen y generar una miniatura.

Cuadro 13: HU-11 – Modificar la imagen de perfil.

Historia de usuario	**Modificar la imagen de portada.**
Descripción	Como usuario autentificado, deseo poder modificar la imagen de portada en tiempo real y que otros usuarios puedan ver la modificación.
	Criterios de aceptación
Nombre del criterio	**Descripción**
Seleccionar imagen miniatura	Cuando el usuario modifique la imagen, el sistema abrirá una ventana modal para recortar la imagen.

Cuadro 14: HU-12 – Modificar la imagen de portada.

A.4.3. Álbum de fotografía

Las historias de usuario asociadas al álbum de fotografía están contenidas en esta sección de anexo y son las que se muestran a continuación.

Historia de usuario	**Crear un álbum de fotografías con una visibilidad concreta.**
Descripción	Como usuario autentificado, deseo crear un álbum de fotos para que otros usuarios puedan ver las imágenes organizadas.
	Criterios de aceptación
Nombre del criterio	**Descripción**
Validación de los campos	Cuando la caja de texto no tenga ningún carácter o el campo visibilidad sea inválido y el usuario pulse en el botón publicar entonces el sistema notificará visualmente al usuario del error detectado.

Cuadro 15: HU-13 – Crear un álbum de fotografías con una visibilidad concreta.

Historia de usuario	**Visualizar los álbumes de fotografías.**
Descripción	Como usuario autentificado, deseo visualizar un álbum de fotos de otros usuarios y mis propios álbumes para que los usuarios puedan visualizar las imágenes.
	Criterios de aceptación
Nombre del criterio	**Descripción**
Comprobar la visibilidad del álbum	Cuando el usuario autentificado quiera visualizar un álbum de fotos entonces el sistema comprobará si el usuario tiene permisos. Si no tiene permisos informará del error

Cuadro 16: HU-14 – Visualizar los álbumes de fotografías.

Historia de usuario	**Eliminar un álbum de fotografías.**
Descripción	Como usuario autentificado, deseo eliminar álbumes de fotografías para eliminarlos de la base de datos y que los usuarios no puedan visualizarlos.
	Criterios de aceptación
Nombre del criterio	**Descripción**
Comprobaciones del álbum de fotografías	Cuando el usuario pulse el botón eliminar un álbum de fotografías, el sistema realizará las comprobaciones entre las cuales se destacan verificar que el identificador del álbum exista, sea entero y que exista en la base de datos informando en caso de que se produzca algún error.
Eliminar álbum protegido	Cuando el usuario autentificado elimine un álbum de portada o perfil, el sistema informará al usuario que no puede realizar esa acción.
Eliminar un álbum de otro usuario	Cuando el usuario autentificado elimine un álbum de otro usuario, el sistema mostrará un mensaje de error especificando que no puede realizar esa acción.

Cuadro 17: HU-15 – Eliminar un álbum de fotografías.

A.4.4. Imagen

Después de haber definidos las historias de usuario de los álbumes de fotografías, se definirán las historias de usuario de las imágenes.

Historia de usuario	**Añadir una imagen a su álbum de fotos.**
Descripción	Como usuario autentificado, deseo añadir una imagen a un álbum de fotos concreto un mensaje para guardarlo en el sistema.
	Criterios de aceptación
Nombre del criterio	**Descripción**
Validación de los campos	Cuando falte algún dato requerido o no se envíe algún parámetro necesario al servidor entonces el sistema notificará visualmente al usuario del error detectado.
Álbum no válido o no proporcionado	Cuando el sistema no encuentre el álbum o el álbum no pertenezca a ese usuario entonces el sistema informará al usuario del error.
Imagen en miniatura	Cuando el sistema no encuentre la imagen en miniatura entonces el sistema informará al usuario del error.
imagen no acorde a las limitaciones del servidor	Cuando el sistema valide la imagen en miniatura o de original y encuentre que no cumple las limitaciones, entonces el sistema informará del error al usuario.
Seis primeras imágenes subidas en el muro	Cuando el usuario visualice el muro de un usuario, el sistema mostrará sus seis primeras imágenes

Cuadro 18: HU-16 – Añadir una imagen a su álbum de fotos.

Historia de usuario	**Seleccionar una imagen como imagen del álbum de fotos.**
Descripción	Como usuario autentificado, deseo cambiar la imagen asignada actualmente del álbum de fotos para guardarlo en el sistema.
	Criterios de aceptación
Nombre del criterio	**Descripción**
Validación de los campos	Cuando falte algún dato requerido o no se envíe algún parámetro necesario al servidor entonces el sistema notificará visualmente al usuario del error detectado.

Cuadro 19: HU-17 – Seleccionar una imagen como imagen del álbum de fotos.

A.4.5. Vídeo

En esta sección se detallan las historias de usuarios asociadas a los vídeos. Cada usuario podrá subir vídeos a su espacio reservado, visualizarlos y eliminarlos. Además podrá ver los videos de otros usuarios.

Historia de usuario	**Subir un vídeo.**
Descripción	Como usuario autentificado, deseo subir un vídeo para visualizarlo en otro momento.
	Criterios de aceptación
Nombre del criterio	**Descripción**
Validación de los campos	Cuando falte algún dato requerido o no se envíe algún parámetro necesario al servidor entonces el sistema notificará visualmente al usuario del error detectado.
Extensiones permitidas	Cuando el sistema valide el vídeo y encuentre que no tiene una extensión válida entonces el sistema informará del error.

Cuadro 20: HU-18 – Subir un vídeo.

Historia de usuario	**Visualizar un vídeo.**
Descripción	Como usuario autentificado, deseo visualizar un vídeo de un usuario o propio.
	Criterios de aceptación
Nombre del criterio	**Descripción**
Seis primeros vídeos subidos en el muro	Cuando el usuario visualice el muro de un usuario, el sistema mostrará sus seis primeros vídeos.

Cuadro 21: HU-19 – Visualizar un vídeo.

Historia de usuario	**Eliminar un vídeo.**
Descripción	Como usuario autentificado, deseo eliminar un vídeo que previamente he subido.
	Criterios de aceptación
Nombre del criterio	**Descripción**
Comprobaciones generales previas	Cuando el usuario pulse en el botón de eliminar un vídeo, el sistema verificará que el identificador del vídeo exista y sea entero, Además verificará que exista en la base de datos informando en caso de que se produzca algún error.
Eliminar un vídeo de otro usuario	Cuando el usuario autentificado elimine un vídeo de otro usuario, el sistema mostrará un mensaje de error indicando que no puede realizar esa acción.

Cuadro 22: HU-20 – Eliminar un vídeo.

A.4.6. Solicitud de amistad

Las historias de usuario asociadas a la solicitud de amistad están contenidas en esta sección del anexo y son las que se muestran a continuación.

Historia de usuario	**Enviar una nueva solicitud de amistad.**
Descripción	Como usuario autentificado, deseo enviar una solicitud de amistad a otro usuario para ser su amigo.
	Criterios de aceptación
Nombre del criterio	**Descripción**
Validación general	Cuando el usuario autentificado envíe una solicitud de amistad a otro usuario entonces el sistema comprobará si existe nombre de usuario y mostrará un error en caso que algún parámetro no sea del tipo esperado o no exista.

Cuadro 23: HU-21 – Enviar una nueva solicitud de amistad.

Historia de usuario	**Cancelar una solicitud de amistad.**
Descripción	Como usuario autentificado, deseo eliminar una solicitud de amistad que otro usuario me ha hecho para no aceptar la solicitud .
	Criterios de aceptación
Nombre del criterio	**Descripción**
Comprobar que la solicitud exista	Cuando el usuario autentificado envíe una petición para cancelar la solicitud de amistad, el sistema verificará que la solicitud exista y mostrará un mensaje en caso que no haberla encontrado.
Comprobar que el usuario sea el emisor o receptor	Cuando el usuario cancele la solicitud de amistad, el sistema comprobará que el usuario sea emisor o receptor y, en caso que de no cumplirse la comprobación, el sistema informará al usuario del error.

Cuadro 24: HU-22 – Cancelar una solicitud de amistad.

Historia de usuario	**Visualizar las solicitudes de amistad pendientes de aceptar.**
Descripción	Como usuario autentificado, deseo visualizar las solicitudes de amistad pendientes de aceptar.
	Criterios de aceptación
Nombre del criterio	**Descripción**
Solicitudes de amistad en tiempo real	Cuando se produzca una solicitud de amistad, el usuario la visualizará en el momento de producirse.
Sin solicitudes de amistad pendientes de aceptar	Cuando no hayan solicitudes de amistad, el usuario visualizará "No tiene solicitudes pendientes de aceptar".

Cuadro 25: HU-23 – Visualizar las solicitudes de amistad pendientes de aceptar.

A.4.7. Amigo

Los usuarios podrán agregar a otros usuarios a sus amigos, eliminarlos y realizar búsquedas en sus propios amigos o en los de otros usuarios. A continuación se detallan las historias de usuario y los criterios de aceptación.

Historia de usuario	**Agregar a un amigo.**
Descripción	Como usuario autentificado, deseo aceptar solicitudes de amistad y hacer amigos con otros usuarios.
	Criterios de aceptación
Nombre del criterio	**Descripción**
Eliminar solicitud de amistad	Cuando el usuario acepte la solicitud, ésta será eliminada y se creará el objeto Amistad que permitirá ser amigo del otro usuario.

Cuadro 26: HU-24 – Visualizar a los amigos de un usuario.

Historia de usuario	**Eliminar a un amigo.**
Descripción	Como usuario autentificado, deseo eliminar a un amigo de mi lista de amigos.
	Criterios de aceptación
Nombre del criterio	**Descripción**
Amistad inexistente	Cuando el usuario intente eliminar una amistad que no existe, el sistema informará al usuario del error
Amistad de otro usuario	Cuando el usuario intente eliminar una amistad de otro usuario, el sistema informará al usuario del error.

Cuadro 27: HU-25 – Eliminar a un amigo.

Historia de usuario	**Buscar amigos.**
Descripción	Como usuario autentificado, deseo buscar y filtrar a amigos por nombre y población para crear una lista con los amigos encontrado.
	Criterios de aceptación
Nombre del criterio	**Descripción**
No coincidencias	Cuando la búsqueda no haya tenido coincidencias, el sistema mostrará el mensaje *Sin resultados*
Sin amigos	Cuando el usuario no tenga amigos, el sistema mostrará una plantilla con una imagen y en la parte inferior de la imagen el texto, "Todavía no tienes amigos"
Filtrar búsqueda	Cuando el usuario quiera realizar una búsqueda de un amigo y tenga la necesidad de acotar la búsqueda filtrando por una población especifica, el sistema permitirá realizar ésta búsqueda.

Cuadro 28: HU-26 – Buscar amigos.

A.4.8. Notificación

En esta sección se detallarán las historias de usuarios de las notificaciones. En particular, cada usuario podrá visualizar sus notificaciones generales, eliminarlas y marcarlas como vistas.

Historia de usuario	**Visualizar sus notificaciones generales.**
Descripción	Como usuario autentificado, deseo visualizar a las notificaciones del usuario para saber los acontecimientos ocurridos.
	Criterios de aceptación
Nombre del criterio	**Descripción**
Notificaciones en tiempo real	Cuando se produzca una notificación de un evento ocurrido en el sistema y tenga relación con el usuario, el sistema informará a ese usuario del acontecimiento justo cuando se produzca.

Cuadro 29: HU-28 – Visualizar sus notificaciones generales.

Historia de usuario	**Eliminar una notificación.**
Descripción	Como usuario autentificado, deseo eliminar una notificación de la lista de notificaciones.
	Criterios de aceptación
Nombre del criterio	**Descripción**
Notificación no existe	Mostrar un error cuando se elimine una notificación que no existe.
Eliminar notificación de otro usuario	Mostrar un error si se intenta eliminar una notificación de otro usuario.
Notificaciones en tiempo real	Cuando se produzca una notificación de un evento ocurrido en el sistema y tenga relación con el usuario, el sistema informará a ese usuario del acontecimiento justo cuando se produzca.

Cuadro 30: HU-29 – Eliminar a un amigo.

Historia de usuario	**Marcar una notificación como vista.**
Descripción	Como usuario autentificado, deseo marcar una notificación como vista cuando posicione el cursor en la notificación con el objetivo de llevar un control de las notificaciones he visto.
	Criterios de aceptación
Nombre del criterio	**Descripción**
Generar un error si la notificación ya está vista	Mostrar un error si se intenta marcar una notificación que previamente ya ha sido marcada como vista.
Notificación de otro usuario	Cuando el usuario posicione el cursor enviará una petición AJAX y el servidor comprobará que la notificación pertenezca al usuario informando en caso de error.

Cuadro 31: HU-30 – Marcar una notificación como vista.