Institut Puig Castellar
Santa Coloma de Gramenet

FriendsGo

(Proyecto de desarrollo)

CFGS Desarrollo de Aplicaciones Multiplataforma

Erik Saldaña, Jaider Cabarcas

DAW2

2024-2025

Resumen del proyecto:

El proyecto se basa en la implementación de las funciones de diversas redes sociales como Instagram, Omegle y Tinder para conseguir una sola red social que pueda satisfacer al usuario en prácticamente cualquier ámbito, entretenimiento observando las publicaciones de otras personas, ya sean conocidos, 'influencers' o personas con gran popularidad; conocer nuevas personas y relacionarse con ellas a partir de un chat en directo, llamadas o videollamadas.

El desarrollo de esta aplicación tiene como objetivo ofrecer una única opción ligera y adaptada a cada usuario, evitando así la necesidad de recurrir a varias aplicaciones y simplificando la búsqueda de entretenimiento. Con un diseño intuitivo y adaptable, busca brindar una experiencia fluida y personalizada que maximice la comodidad y el disfrute del usuario.

Cómo se piensa conseguir el objetivo de este proyecto es dividirlo en pequeñas partes, utilizando la metodología Scrum para trabajar de una manera más eficiente y organizada para aprovechar lo máximo posible el tiempo disponible para este proyecto, así se asegura de que cada semana se va avanzando en el proyecto comenzando por diseñar los diagramas y la interfaz de usuario.

Palabras claves:

- FriendsGo
- Red Social
- Videollamadas Aleatorias
- Interacción
- Multiplataforma

Abstract:

The project is based on implementing features from various social networks such as Instagram, Omegle, and Tinder to create a single social network that can satisfy users in practically any area: entertainment through viewing posts from others, whether they are acquaintances, influencers, or popular figures; and meeting new people and interacting with them through live chat, calls, or video calls.

The aim of developing this application is to offer a single, lightweight option tailored to each user, thereby eliminating the need for multiple applications and simplifying the search for entertainment. With an intuitive and adaptable design, it seeks to provide a smooth, personalized experience that maximizes user comfort and enjoyment.

The objective will be achieved by breaking the project into smaller parts, using the Scrum methodology to work more efficiently and in an organized manner, maximizing the available time for this project. This ensures weekly progress, beginning with the design of diagrams and the user interface.

Keywords (entre 4 y 8):

- FriendsGo
- Social Network
- Random Video Calls
- Multiplatform
- Interaction
- Real-Time Interaction

Índex

1. Introducción

En un mundo digital cada vez más interconectado, las formas de comunicación han evolucionado hacia dinámicas basadas en la inmediatez, la globalización y la interacción virtual. En este contexto nace **FriendsGO**, una plataforma web que busca revolucionar la experiencia social en línea mediante la integración de videollamadas aleatorias con funciones propias de una red social moderna. El objetivo es crear un espacio que combine entretenimiento, expresión personal y conexión humana genuina.

Inspirada en plataformas como **Omegle** e **Instagram**, FriendsGO fusiona la conexión instantánea con desconocidos y la publicación de contenido multimedia en una sola experiencia digital. Los usuarios pueden realizar videollamadas aleatorias con personas de cualquier parte del mundo, compartir publicaciones, seguir a otros perfiles y valorar interacciones.

La propuesta busca responder a una necesidad creciente: evitar la fragmentación del uso de múltiples aplicaciones para socializar, publicar y comunicarse. FriendsGO centraliza estas funciones en una única plataforma accesible, ligera y personalizada. Así, se convierte en una solución integral para quienes desean explorar nuevas conexiones, expresarse libremente y formar parte de una comunidad digital activa y respetuosa.

Más allá de su funcionalidad, FriendsGO representa un nuevo enfoque en la interacción social online. Apoyada en tecnologías modernas y desarrollada con metodologías ágiles, la plataforma está diseñada para crecer y adaptarse junto a sus usuarios. Con este proyecto, se da un paso hacia una red social innovadora.

1.1 Contexto

FriendsGO es un proyecto que surge de la necesidad de simplificar y mejorar la experiencia de interacción social digital. Actualmente, los usuarios utilizan múltiples aplicaciones para realizar diferentes acciones: hacer videollamadas, compartir contenido, chatear o conocer nuevas personas. Esta fragmentación obliga a cambiar constantemente de plataforma, lo cual puede resultar poco práctico y disperso. FriendsGO propone una solución unificada que combine estas funcionalidades en un solo lugar.

El concepto parte de observar como plataformas como Omegle, Instagram o incluso Tinder cubren necesidades sociales específicas, pero no ofrecen una experiencia combinada. La idea de este proyecto es tomar elementos clave de cada una como la conexión con desconocidos a través de videollamadas o la publicación de fotos y videos y reunirlos en una misma plataforma. Esto permitiría a los usuarios explorar nuevas conexiones, compartir momentos y comunicarse sin necesidad de depender de varias herramientas.

FriendsGO también pretende responder al interés de muchas personas por conocer otras culturas, ampliar su círculo social o simplemente entablar conversaciones de forma espontánea, sin barreras geográficas. La opción de realizar videollamadas aleatorias es una de las funciones principales del proyecto, pensada para fomentar ese tipo de encuentros digitales que otras redes no ofrecen de manera tan directa.

Este proyecto representa una oportunidad para explorar nuevas formas de interacción online, basadas en funcionalidades reales y necesidades observadas en el uso cotidiano de las redes sociales actuales. La propuesta no busca competir directamente con plataformas consolidadas, sino ofrecer una alternativa que combine varias de sus características más valoradas.

1.2 Justificación

Este proyecto surge de la necesidad de una plataforma que centralice las funcionalidades más deseadas de diversas redes sociales. Mientras que las redes actuales ofrecen funciones limitadas y dispersas, FriendsGO busca ofrecer todo lo necesario en una sola plataforma: videollamadas aleatorias, publicación de contenido, y la posibilidad de interactuar de manera fluida y personalizada. Así, se busca eliminar la frustración de los usuarios que deben navegar entre múltiples aplicaciones para satisfacer sus necesidades de entretenimiento y conexión.

Informe sobre el Tiempo Promedio de Uso de Redes Sociales en España y la Cuestión del Uso Excesivo

Según datos del informe **Digital 2025: Global Overview Report**, una publicación elaborada por las agencias **We Are Social** y **Meltwater**, y basada en análisis de datos de la plataforma **GWI** (GlobalWebIndex), el tiempo promedio diario que los usuarios de internet de 16 a 64 años en **España** dedican al uso de redes sociales es de **1 hora y 56 minutos** (We Are Social & Meltwater, enero de 2025).

Este promedio, aunque representa una media de la población usuaria, ya indica una inversión de tiempo considerable en estas plataformas. Es importante destacar que, al ser un promedio, existe un segmento de la población cuyo uso supera ampliamente esta cifra, pudiendo derivar en un patrón de uso excesivo.

Informe sobre el Número de Redes Sociales Utilizadas por Persona

Según datos del informe **Digital 2024: Global Overview Report**, elaborado por *We Are Social* en colaboración con *Meltwater*, y basados en información de la plataforma de análisis **GWI (GlobalWebIndex)**, el usuario promedio a nivel mundial utiliza **6.86 redes sociales activamente cada mes** (DataReportal, 2024).

Este dato refleja una tendencia creciente hacia la multicanalidad en el uso de plataformas digitales. El usuario típico no se limita a una o dos redes, sino que interactúa con múltiples servicios sociales, ya sea por motivos personales, laborales o de entretenimiento.

En este contexto, la creación de FriendsGO se convierte en una propuesta innovadora que busca reunir esas funcionalidades tan solicitadas, pero que aún no han sido centralizadas en una única red social. La posibilidad de realizar videollamadas aleatorias, compartir contenido, personalizar perfiles y evaluar el comportamiento de otros usuarios dentro de una misma plataforma, responde a una creciente demanda de servicios integrados y fáciles de usar.

Con este proyecto, se busca transformar la manera en que las personas interactúan en línea, proporcionando una red social intuitiva y segura que ofrezca una experiencia completa y accesible en un solo lugar. Este enfoque no solo mejora la experiencia del usuario, sino que también abre nuevas oportunidades para la conexión entre personas y organizaciones con intereses y valores comunes.

1.3 Objetivos

1.3.1 Objetivo general

Durante el desarrollo de esta aplicación web, se ha optado por centrarse en construir una red social que sea agradable para todos los usuarios, independientemente de su edad, etnia o género. Esto se logrará mediante un diseño intuitivo y una navegación clara. Además, se busca que la plataforma sea robusta, escalable y fácil de usar para cualquier tipo de usuario.

Asimismo, se persigue el crecimiento profesional de los desarrolladores mediante la adquisición de nuevos conocimientos, el aprendizaje de diversas tecnologías, el desarrollo de habilidades en el diseño de interfaces amigables y el refuerzo de los conocimientos previos.

1.3.2 Objetivos específicos

A continuación se muestra el listado de objetivos específicos:

De la aplicación:

- Crear un sistema de videollamadas con personas aleatorias.
- Interfaz sencilla.
- Creación de una base de datos robusta.
- Creación de una API que pueda comunicar al cliente con el servidor.
- Sistema de mensajes en línea.
- Publicación de archivos en la red social.
- Control de registro/inicio de sesión

De los desarrolladores:

- Reforzar el conocimiento de tecnologías como Typescript, HTML, CSS, etc.
- Adquirir conocimientos sobre el uso de frameworks como React y ExpressJS
- Familiarizarse sobre el uso de tecnologías como WebRTC, Socket.io y NodeJS.

1.4 Estrategia y planificación del proyecto

La estrategia adoptada para el desarrollo de FriendsGO consiste en crear una aplicación completamente nueva desde cero, lo que permite tener control total sobre su estructura técnica y funcional. Esta decisión responde a la necesidad de integrar en una sola plataforma funcionalidades específicas como videollamadas aleatorias, publicaciones, valoraciones y chat en tiempo real, sin depender de herramientas externas que puedan limitar su implementación.

La planificación del proyecto se ha dividido en fases claras, empezando por el diseño de la interfaz mediante wireframes y continuando con el desarrollo de las funcionalidades principales: gestión de usuarios, publicación de contenido, videollamadas y mensajería. Aunque no se han realizado pruebas formales de accesibilidad, se ha procurado mantener un diseño claro y comprensible, siguiendo patrones comunes en redes sociales actuales.

A continuación, se muestra un desglose de las fases del proyecto, indicando su duración estimada y los objetivos principales de cada etapa:

Tarea / Fase	Duración estimada	Descripción
1. Análisis del proyecto	1 semana	Definición de objetivos generales y específicos, análisis del problema y redacción del enfoque del proyecto.
2. Investigación de referencias	1 semana	Análisis de plataformas similares (Omegle, Instagram) y definición de funcionalidades deseadas.
3. Diseño de wireframes	1 semana	Bocetos de pantallas clave utilizando Excalidraw para escritorio y móvil.
5. Elección y configuración de tecnologías	1 semana	Selección de React, Node.js, WebRTC, PostgreSQL y configuración inicial del entorno de desarrollo.
6. Diseño de arquitectura del sistema	1 semana	Creación del esquema de arquitectura frontend/backend y comunicación en tiempo real.

7. Desarrollo de autenticación	1 semana	Registro, inicio de sesión, validación de usuario y recuperación de contraseña.
8. Gestión de perfiles y edición	1 semana	Visualización y edición de perfil, incluyendo imagen, biografía y configuración básica.
9. Sistema de publicaciones	2 semanas	Creación, edición y eliminación de publicaciones multimedia (texto, imágenes, video).
10. Feed y visualización de contenido	1 semana	Implementación de feed de publicaciones personalizadas por usuario.
11. Desarrollo de videollamadas aleatorias	3 semanas	Emparejamiento aleatorio de usuarios y conexión vía WebRTC + Socket.io.
12. Valoraciones tras videollamada	1 semana	Sistema de puntuación e historial de calificaciones tras cada interacción por video.
13. Chat entre usuarios	2 semanas	Implementación de mensajería en tiempo real y almacenamiento de historial.
14. Notificaciones en tiempo real	1 semana	Envío de notificaciones por solicitudes, mensajes y valoraciones.
15. Pruebas internas y correcciones	2 semanas	Validación manual de funcionalidades, detección de errores y mejoras visuales.
16. Redacción de la memoria del proyecto	2 semanas	Estructuración de los apartados del documento final con capturas y explicaciones técnicas.
17. Creación de anexos y presentación final	2 semanas	Revisión de documentación, recopilación de imágenes, anexos y preparación para la defensa.

Este enfoque se considera adecuado y viable, ya que permite adaptar la aplicación a los objetivos del proyecto y facilita su evolución futura, con una integración coherente entre todas sus partes.

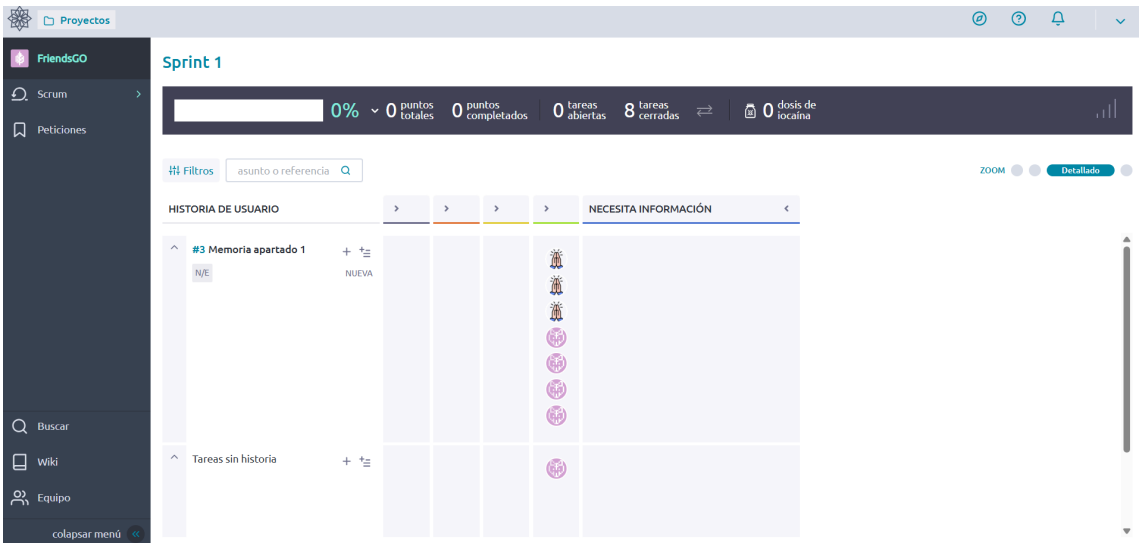
1.5 Metodología de trabajo

La metodología seleccionada para el desarrollo del proyecto es Scrum. Permite gestionar y organizar proyectos en bloques pequeños de trabajo llamados sprints, los cuales, en nuestro caso, tendrán una duración de una semana.

Durante cada sprint se definirán objetivos concretos, como la implementación de nuevas funcionalidades, la corrección de errores o el desarrollo de componentes. Para asignar las tareas correspondientes, se realizará una breve reunión en la que se decidirá qué se debe completar en ese sprint y quién será responsable de cada tarea.

La elección de Scrum se debe a que permite una alta capacidad de adaptación frente a cambios, fomentando una respuesta rápida y eficaz del equipo, al mismo tiempo que mantiene un ritmo constante de trabajo orientado a la entrega continua de valor.

Para la gestión y seguimiento de las tareas se utilizará la plataforma Taiga.io, que facilitará el control visual y organizado del progreso del proyecto. A continuación, se muestra una vista del Sprint 1 del proyecto FriendsGO en la herramienta de gestión Taiga.io:



Dado que el equipo de desarrollo de FriendsGo es reducido, no se realizarán reuniones diarias o semanales de seguimiento. En su lugar, se optará por una comunicación directa y constante entre los miembros del equipo, lo que permitirá una mayor flexibilidad y eficiencia en la toma de decisiones.

1.6 Estudio económico y presupuestario

Se ha hecho un cálculo para estimar los costes necesarios, incluyendo los recursos técnicos, humanos y materiales requeridos, así como los gastos de mantenimiento. El presupuesto resultante permite valorar la viabilidad del proyecto desde una perspectiva profesional.

1.6.1 Análisis de tareas y componentes

Las tareas contempladas para el desarrollo de la plataforma incluyen:

- Las principales tareas previstas en el desarrollo del proyecto incluyen:
- Diseño de la interfaz directamente en código.
- Programación del frontend (React y Remix).
- Desarrollo del backend (Node.js, Express).
- Implementación de videollamadas con WebRTC y Socket.io.
- Configuración de la base de datos (PostgreSQL).
- Desarrollo de funcionalidades como publicaciones, chat en tiempo real, búsqueda de usuarios y control de acceso.
- Pruebas funcionales básicas en entorno local.

1.6.2 Publicidad y visibilidad

Para facilitar la adopción de la plataforma, se plantea una inversión inicial en publicidad digital, especialmente en redes sociales, como estrategia de difusión. Esta campaña tendría un enfoque simple y directo, acorde con el estilo de redes como el Facebook clásico.

1.6.3 Estimación de costes

A continuación se muestra una tabla de precios en contexto a nuestra aplicación.

Concepto	Estimación de horas	Precio por hora(€)	Coste aproximado (€)
Desarrollo Frontend	130	15	1.950 €
Desarrollo Backend	120	15	1.800 €
Integración WebRTC + Chat	60	15	900 €
Gestión de base de datos	40	15	600 €
Diseño en código	25	15	375 €
Pruebas funcionales	30	15	450 €
Formación técnica	30	12.5	360 €
Publicidad básica (opcional)	20	12	250 €
Total	**455**	**—**	**6.685 €**

2 Descripción del proyecto

2.1 Análisis de requisitos

Como requisitos mínimos que se han propuesto para presentar el proyecto de forma atractiva, se ha considerado tener múltiples funcionalidades, plenamente operativas de cara a la entrega del proyecto.

- Opción de inicio de sesión y registro de un usuario
- Publicación de imágenes y textos
- Poder eliminar publicaciones propias
- Dar me gusta a una publicación y que se quede registrado
- Videollamadas con personas aleatorias con opciones de finalizar o hacer “match”
- Si se hace match generar un chat entre estas dos personas
- Al finalizar la llamada dar la opción de evaluar la videollamada con la otra persona
- Búsqueda de usuarios
- Editar perfil
- En el perfil se muestra la valoración de las videollamadas de la persona

2.1.1 Requisitos funcionales

- Registro de usuario: Los usuarios deben registrarse con nombre, apellido, usuario, correo y contraseña.
- Login: Permite a los usuarios registrados acceder con su usuario o correo y contraseña, manteniendo la sesión abierta hasta que cierren sesión.
- Crear publicaciones: Los usuarios pueden compartir contenido como texto o imágenes en la plataforma.
- Retirar publicaciones: Los usuarios pueden eliminar sus publicaciones en cualquier momento.
- Videollamadas aleatorias: Los usuarios pueden realizar videollamadas aleatorias con otros usuarios disponibles.
- Agregar/enviar solicitud de seguimiento mediante la videollamada: Los usuarios pueden enviar solicitudes de amistad durante una videollamada aleatoria.
- Buscar usuario mediante la barra de búsqueda: Permite buscar perfiles por nombre de usuario.
- Enviar solicitud de seguimiento mediante la barra de búsqueda: Permite enviar solicitudes de amistad directamente desde la búsqueda de perfiles.
- Aceptar solicitud de seguimiento: Los usuarios pueden aceptar solicitudes de seguimiento y agregar a otros a su red.
- Remover amistad: Los usuarios pueden eliminar contactos de su lista de amigos o seguidores.
- Valoración después de una videollamada aleatoria: Los usuarios pueden valorar el comportamiento de otro usuario después de una videollamada.
- Notificaciones: Los usuarios reciben notificaciones sobre solicitudes, comentarios y posts.
- Visualización de perfil de usuarios: Los usuarios pueden ver el perfil de otros usuarios, incluidas valoraciones y publicaciones.
- Editar perfil: Los usuarios pueden editar su perfil, excepto las valoraciones de videollamadas, que son permanentes.
- Moderación de contenido: El staff revisa las publicaciones, perfiles o comentarios inapropiados y toma decisiones sobre una sanción.
- Feed personalizado: Los usuarios ven un feed de publicaciones e interacciones relevantes según sus intereses y conexiones.
- Sancionar usuarios: El staff puede imponer sanciones a los usuarios que infringen las normas, como restricciones.

- Remover sanciones: El staff puede remover sanciones a los usuarios si se considera que el comportamiento ha mejorado.
- Enviar mensajes en tiempo real: Los usuarios podrán intercambiar mensajes de texto en tiempo real con otros usuarios con los que hayan interactuado (por ejemplo, desde una lista de amigos)
- Historial de mensajes: Los usuarios podrán consultar el historial de conversaciones anteriores con otros usuarios.
- Gestión de contactos: Solo se podrán enviar mensajes a usuarios agregados como amigos tras ser agregados mediante una solicitud de amistad.

2.1.2 Requisitos no funcionales

La plataforma FriendsGO se desarrollará con el objetivo de ofrecer una experiencia de uso fluida y coherente con los estándares actuales de las aplicaciones web. Aunque no se realizará una evaluación formal de usabilidad, se procurará seguir patrones de diseño presentes en otras redes sociales, con la intención de facilitar una navegación intuitiva.

En lo referente a la seguridad, las contraseñas de los usuarios se almacenarán de forma cifrada mediante algoritmos de hash, evitando así su lectura directa en la base de datos. No se incluirán, por el momento, medidas avanzadas de protección frente a ataques (como la limitación de intentos de inicio de sesión o validaciones exhaustivas del lado del servidor), ya que el enfoque principal se centrará en la implementación funcional en entorno local.

La plataforma será accesible desde navegadores actuales y se ejecutará en un entorno de desarrollo local. Por ello, no se contemplarán configuraciones específicas de producción relacionadas con privacidad, soporte técnico o monitorización.

2.3 Tecnologías

2.3.1 Comparativa de las tecnologías valoradas y elegidas

Las tecnologías valoradas para el desarrollo del proyecto han sido varias, con diversos cambios a medida que el proyecto ha ido avanzando, tanto en el FrontEnd, BackEnd y en el lenguaje para la base de datos.

Tecnologias de FrontEnd:

Aquí se optó por elegir entre React, Angular y Vue, ya que para el FrontEnd se buscaba usar TypeScript.

	SINTÁXIS	APRENDIZAJE	OPTIMIZACIÓN	COMUNIDAD
	Usa JavaScript puro con separación entre HTML y lógica. Los templates de Vue tienen una sintaxis específica que permite manipular dinámicamente el contenido. Combina HTML casi puro con instrucciones que Vue procesa automáticamente.	Inicialmente accesible, pero el uso de JSX y los conceptos propios de React pueden volverse complicados conforme avanza el desarrollo.	Destaca por su Virtual DOM, que lo hace eficiente en aplicaciones dinámicas con actualizaciones frecuentes. Su enfoque modular permite optimizaciones específicas según las necesidades.	Tiene la comunidad más amplia y activa, respaldada por Meta (Facebook), lo que garantiza acceso a recursos, bibliotecas y soporte continuo.
	Similar a Vue, pero con una separación estricta entre los archivos de HTML y lógica (JavaScript+TypeScript). Usa TypeScript para añadir funcionalidad adicional, convirtiéndose en un lenguaje más estructurado.	Tiene una curva de aprendizaje más pronunciada debido a su complejidad, uso obligatorio de TypeScript, y la separación de archivos para lógica y templates.	Aunque más pesado para aplicaciones pequeñas, es extremadamente eficiente en proyectos complejos gracias a su compilación AOT (Ahead-of-Time), que optimiza tiempos de carga y ejecución.	Es respaldado por Google y cuenta con una comunidad activa y soporte corporativo sólido. Sin embargo, su popularidad ha disminuido ligeramente en proyectos modernos.
	Combina HTML y JavaScript mediante JSX, que permite escribir contenido HTML dentro de funciones o clases JavaScript. No hay separación entre lógica y presentación, como en Vue o Angular.	Considerado el más fácil de aprender. Su sintaxis es intuitiva, permite usar JavaScript puro o TypeScript, y su estructura no requiere separación estricta de archivos. Ideal para empezar rápidamente.	Similar a React con su uso del Virtual DOM, pero más ligero, lo que lo hace rápido y adecuado para aplicaciones pequeñas y medianas, con capacidad de escalar según el proyecto.	Cuenta con una comunidad más pequeña, está creciendo rápidamente y es altamente valorada por startups y desarrolladores independientes. Ha sido apoyado por empresas como Alibaba, lo que muestra su fiabilidad en producción.

Al final se decidió usar React porque su curva de aprendizaje no es tan elevada para lo que ofrece, además de todos los recursos disponibles al ser una comunidad tan grande y por su DOM, el cual puede ser de gran utilidad en el desarrollo de la aplicación web y sus futuras actualizaciones.

Después de haber elegido React para el FrontEnd, se buscó un framework que se adapte a las necesidades de nuestra aplicación web.

	CASOS DE USO	APRENDIZAJE	FLEXIBILIDAD	ESCALABILIDAD
	Ideal para aplicaciones web orientadas al SEO, como blogs, páginas corporativas y tiendas en línea. Su soporte para renderizado híbrido lo hace perfecto para sitios dinámicos y estáticos.	Integra características avanzadas como SSR y SSG de forma nativa, lo que reduce la configuración inicial. Su sistema de rutas basado en archivos es simple, pero puede ser desafiante para quienes solo conocen React puro	Es altamente flexible gracias a su soporte integrado para SSR, SSG y APIs internas, lo que permite personalizar tanto el front-end como el back-end. Sin embargo, está más centrado en aplicaciones web que móviles.	Soporta aplicaciones altamente escalables mediante SSR, SSG e ISR (Incremental Static Regeneration). Es ideal para aplicaciones que requieren SEO sólido y manejan contenido dinámico en gran escala, como e-commerce y blogs.
	Excelente para el desarrollo de aplicaciones móviles multiplataforma que requieren acceso a características nativas, como cámaras o geolocalización. Se usa ampliamente en aplicaciones empresariales y de consumo	Aunque tiene una curva de aprendizaje moderada, es fácil para desarrolladores React que quieran construir aplicaciones móviles multiplataforma	Proporciona una gran flexibilidad para personalizar componentes y optimizar aplicaciones móviles. Soporta tanto módulos nativos como bibliotecas externas para cubrir necesidades avanzadas.	Perfecto para aplicaciones móviles escalables, permite usar bibliotecas externas para agregar funcionalidades y optimizar el rendimiento en plataformas específicas, como Android e iOS.
	Se utiliza principalmente para proyectos donde la prioridad es una interfaz de usuario rápida y responsiva. Es ideal para prototipos o sitios web con necesidades de diseño estándar	Es intuitivo y práctico para quienes buscan crear interfaces rápidamente, gracias a sus componentes predefinidos basados en Bootstrap	Ofrece componentes listos para usar que son fácilmente personalizables. Aunque su flexibilidad es limitada en comparación con herramientas más genéricas, es útil para desarrollar interfaces rápidamente.	Es excelente para proyectos pequeños y medianos, donde el enfoque está en una interfaz rápida y estética. No es ideal para aplicaciones grandes con requisitos altamente personalizados debido a sus recursos predefinidos.

A la hora de elegir entre estos frameworks/bibliotecas para tener un desarrollo más efectivo en el frontEnd se optó por Next JS, ya que con las opciones que se disponen se ha valorado que es una mejor opción para el desarrollo de aplicaciones web, además de su buen rendimiento gracias al SSR y SSG y la facilidad de actualizar constantemente la aplicación web.

Después de una búsqueda aún más intensa se ha llegado a la conclusión de que NextJS no cumple con los estándares a día de hoy, se han conocido nuevas alternativas a Next JS, las cuales són Astro y Remix.

	RENDIMIENTO	CASOS DE USO	APRENDIZAJE	ESCALABILIDAD
NEXT.JS	Potencialmente excelente con optimización. Riesgo de lentitud (builds, bundle) en apps grandes si no se cuida.	Desde sitios estáticos a aplicaciones full-stack complejas. Ideal para proyectos que evolucionan.	Moderada-alta. Abundancia de características y cambios (ej. App Router) pueden resultar complejos	Alta (serverless, edge). Proyectos grandes exigen planificación para mantener rendimiento y coherencia
R	Muy bueno. Minimiza JS en cliente, uso eficiente de caché. Transiciones de datos ágiles	Fuerte en apps con mucha interacción de datos/formularios (e-commerce). Flujo de datos simplificado	Moderada. Base React + principios web. Convenciones claras pueden simplificar vs. sobrecarga de opciones	Alta. Diseño basado en estándares web y comunicación eficiente cliente-servidor
A	Excelente por defecto (cero JS enviado para contenido estático). "Islands Architecture" para interactividad óptima.	Ideal para sitios web ricos en contenido y centrados en el rendimiento (blogs, portfolios, docs, marketing).	Baja-moderada. Sencillo para sitios estáticos. Integrar "islas" de React es directo. Menos complejo para su nicho.	Buena para sitios de contenido; escalabilidad de funciones dinámicas complejas depende de integraciones o API routes.

Indagando, viendo la opinión de diversos usuarios y algunos foros, se ha decidido finalmente usar Remix debido a que Astro se utiliza más para proyectos con páginas estáticas.

Tecnologías de BackEnd:

Para el BackEnd se observó de distintos entornos de ejecución, entre ellos Node JS, Deno y Bun. La decisión de hacer uso de uno de estos entornos es debido a que se busca programar el BackEnd en TypeScript para así disminuir la cantidad de lenguajes de programación utilizados en el proyecto.

Se consideró a Deno y Bun como alternativas a pesar de su reciente lanzamiento gracias a sus especificaciones tecnológicas y fortalezas.

Deno en seguridad y Bun en rendimiento

	RENDIMIENTO	MANEJO DE DEPENDENCIAS	SEGURIDAD	ECOSISTEMA
	Ofrece un rendimiento sólido para aplicaciones maduras, aunque es superado por Bun en tareas específicas	Usa npm, con gran ecosistema, aunque puede generar redundancias y problemas de seguridad en dependencias grandes	Introdujo permisos básicos en la versión 20, pero aún depende de configuraciones adicionales para mayor seguridad	Es el entorno más maduro y ampliamente adoptado, utilizado en innumerables proyectos en producción. Ofrece estabilidad y soporte continuo de la comunidad
	Rendimiento decente con V8, aunque ligeramente más lento que Node.js y Bun en benchmarks específicos	Descarga dependencias desde URL, eliminando gestores externos y aumentando la transparencia en el proyecto	Diseñado con permisos explícitos para acceso a red y sistema de archivos, lo que aumenta la seguridad por defecto	Aunque más reciente, se ha consolidado como una alternativa seria, con características modernas y un estándar más limpio. Su ecosistema es más pequeño que el de Node.js
	Es significativamente más rápido gracias a su motor optimizado en Zig, ideal para desarrollo ágil y tareas repetitivas	Incluye un gestor ultrarrápido que combina gestión de dependencias, empaquetado y ejecución en una sola herramienta	En desarrollo, carece de un modelo de seguridad robusto	Es el más nuevo de los tres y está en versión beta. Aunque tiene características prometedoras, aún no alcanza el nivel de madurez y estabilidad de Node.js o Deno

Se ha elegido Node JS debido sobre todo a que es el entorno con una comunidad más extensa y recursos de los cuales se puede hacer uso, además de que ofrece un buen rendimiento, aun así quedándose por detrás de Bun. También por su compatibilidad con la mayoría de sistemas y servidores.

La elección de **ExpressJS** frente a otras opciones como **NextJS** o **NestJS** se basa en su sencillez, flexibilidad y bajo nivel de complejidad, lo cual lo convierte en una solución adecuada para proyectos de tamaño medio o pequeño. Al tratarse de un framework minimalista, ExpressJS permitirá una mayor personalización del backend sin imponer una estructura estricta, facilitando así la integración con tecnologías como Socket.io o WebRTC. Además, al contar con una amplia documentación y comunidad activa, se facilitará su implementación y resolución de problemas durante el desarrollo.

Para videollamadas: Socket.io y WebRTC

Para la implementación del servicio de videollamadas en la red social objeto de este proyecto, se ha optado por una arquitectura basada en la combinación de **WebRTC (Web Real-Time Communication)** y **Socket.io**. Esta decisión se fundamenta en la búsqueda de una solución robusta, eficiente y que se integre de manera nativa en la experiencia web del usuario, al tiempo que ofrece las capacidades necesarias para una comunicación en tiempo real de alta calidad.

WebRTC se ha seleccionado como el pilar para la transmisión de audio y vídeo debido a sus ventajas inherentes para este tipo de funcionalidad:

- Permite establecer **conexiones directas (peer-to-peer)** entre los usuarios. Esto resulta crucial para minimizar la latencia, ofreciendo una experiencia de conversación más fluida y natural, y reduce significativamente la carga en la infraestructura de los servidores de la plataforma al no tener que procesar la totalidad de los flujos multimedia.
- Proporciona **comunicaciones seguras por defecto**, ya que el cifrado es un componente obligatorio de la especificación. Este aspecto es fundamental para garantizar la privacidad de las interacciones de los usuarios.
- Al ser un **estándar web abierto y soportado por los principales navegadores**, elimina la necesidad de que los usuarios instalen software adicional o plugins, facilitando la adopción y el acceso universal al servicio desde múltiples dispositivos.

Por su parte, **Socket.io** ha sido elegido para gestionar la capa de señalización y la comunicación en tiempo real auxiliar:

- Su capacidad para establecer **canales de comunicación bidireccionales y persistentes** entre el cliente y el servidor es esencial para el proceso de "señalización". Este proceso implica el intercambio de metadatos de control necesarios para que dos (o más) clientes puedan iniciar y configurar una conexión WebRTC. Socket.io simplifica enormemente esta compleja tarea.
- Ofrece una **abstracción robusta sobre WebSockets**, con mecanismos de fallback y reconexión automática, asegurando una comunicación fiable incluso en condiciones de red variables.
- Más allá de la señalización específica para las videollamadas, la naturaleza versátil de Socket.io permite **reutilizar esta misma infraestructura para otras funcionalidades en tiempo real** de la red social (como chats, notificaciones instantáneas, etc.), optimizando así el desarrollo y la coherencia tecnológica de la plataforma.

Bases de datos:

	RENDIMIENTO	SEGURIDAD	FLEXIBILIDAD	CASOS DE USO
	Excelente para lecturas y escrituras complejas, transacciones ACID robustas. Puede ser un cuello de botella con escrituras masivas concurrentes sin una afinación cuidadosa	Robusto, con control de acceso basado en roles (RBAC), SSL, autenticación extensible, y cumplimiento de estándares de seguridad. Auditoría detallada.	Relacional, requiere esquemas definidos. Soporta tipos de datos JSON/JSONB para cierta flexibilidad, pero la estructura principal es fija. Alterar tablas grandes puede ser costoso.	Perfiles de usuario detallados, relaciones complejas, gestión de contenido con metadatos ricos, sistemas de monetización y anuncios
MySQL	Buen rendimiento general para cargas de trabajo mixtas (lectura/escritura). Optimizaciones específicas según el motor de almacenamiento	Buenas características de seguridad: usuarios y privilegios, SSL, encriptación en reposo (con extensiones o versiones). Menos granularidad en RBAC nativo comparado con PostgreSQL	Relacional, requiere esquemas definidos. Similar a PostgreSQL, alterar tablas grandes puede llevar tiempo y recursos. Soporte para JSON	Datos de usuario, publicaciones, comentarios, "me gusta", seguidores/seguidos, tablas de metadatos y configuraciones
Cassandra	Excepcional para altas tasas de escritura y lectura distribuida. Optimizado para baja latencia en grandes volúmenes de datos.	Seguridad basada en autenticación y autorización. Encriptación en tránsito y en reposo disponible. Menos funcionalidades de auditoría fina incorporadas por defecto comparado con los RDBMS.	Sin esquema (o esquema en lectura). Muy flexible, permite añadir columnas dinámicamente a las familias de columnas sin afectar otras filas. Ideal para datos que evolucionan rápidamente.	Feeds de actividad en tiempo real, mensajería instantánea, notificaciones, almacenamiento de datos de series temporales (analíticas de uso), contadores masivos, caché de datos.

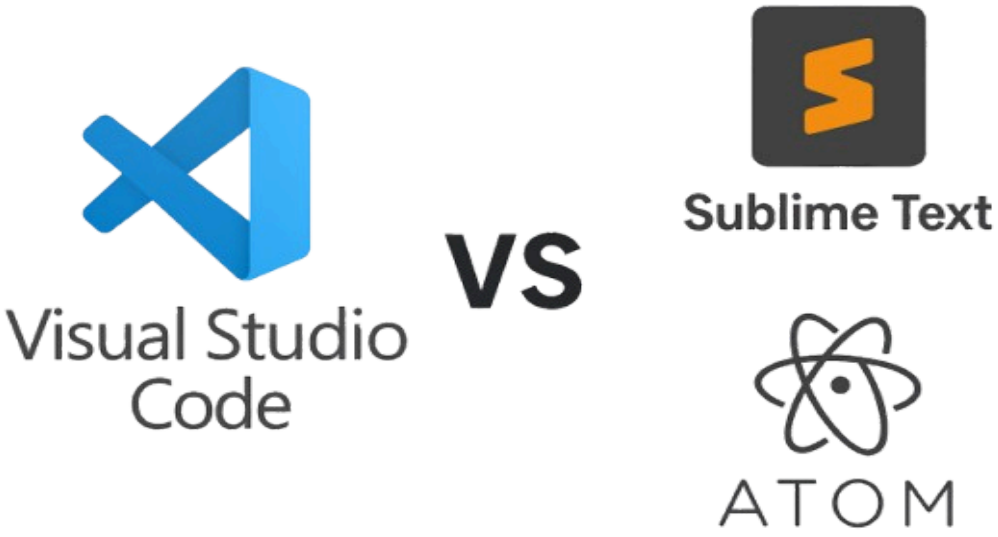
Elegimos **PostgreSQL** como base de datos para nuestra aplicación web debido a su equilibrio entre potencia, consistencia y soporte para relaciones complejas. A diferencia de MySQL, PostgreSQL ofrece funciones avanzadas como tipos de datos JSONB, búsquedas full-text y mejor manejo de integridad referencial. Aunque Cassandra es ideal para grandes volúmenes de escritura y alta disponibilidad, prioriza la disponibilidad sobre la consistencia, lo cual no se ajusta a nuestras necesidades. PostgreSQL nos permite escalar sin perder la estructura relacional ni la fiabilidad de los datos.

Diseño:

Para la creación del wireframe se utilizará Excalidraw, ya que proporciona opciones básicas para poder hacer un diseño base y, a partir de este, ir haciendo mejoras dependiendo de las necesidades finales. Esta herramienta resulta ideal en las etapas iniciales del desarrollo porque permite bocetar rápidamente la estructura visual de la aplicación de forma colaborativa y sencilla. Además, su interfaz minimalista y estilo de dibujo a mano facilitan la comunicación de ideas sin distraer con detalles visuales innecesarios. Gracias a su flexibilidad, podemos modificar los elementos del wireframe de forma ágil a medida que evolucionan los requerimientos del proyecto.

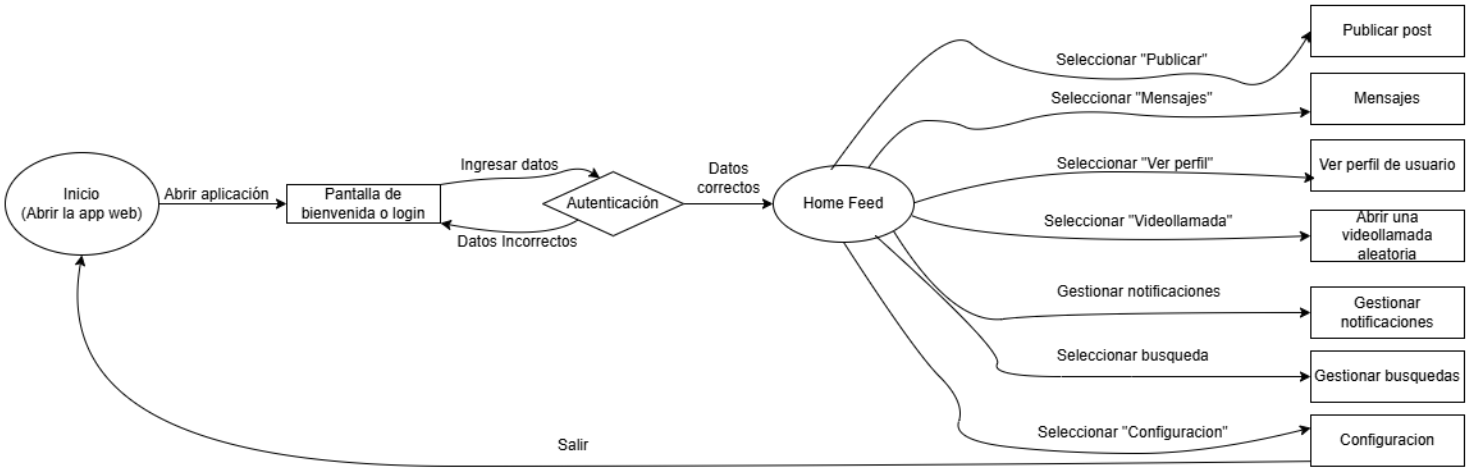
Desarrollo:

Se utiliza Visual Studio Code (**VSCode**) como editor de código principal porque ofrece una combinación ideal de rendimiento, facilidad de uso y extensibilidad, superando a otras opciones como Sublime Text y Atom. A diferencia de Sublime, que requiere licencia para su uso completo, VSCode es totalmente gratuito y de código abierto. En comparación con Atom, VSCode es mucho más rápido y cuenta con un desarrollo activo respaldado por Microsoft, lo que garantiza actualizaciones frecuentes y una comunidad muy activa. Además, su integración con Git, el depurador incorporado, la terminal integrada y la gran variedad de extensiones lo hacen más completo que otros editores livianos, sin llegar a la complejidad de entornos pesados como Eclipse o IntelliJ.



2.4 Estructura del proyecto

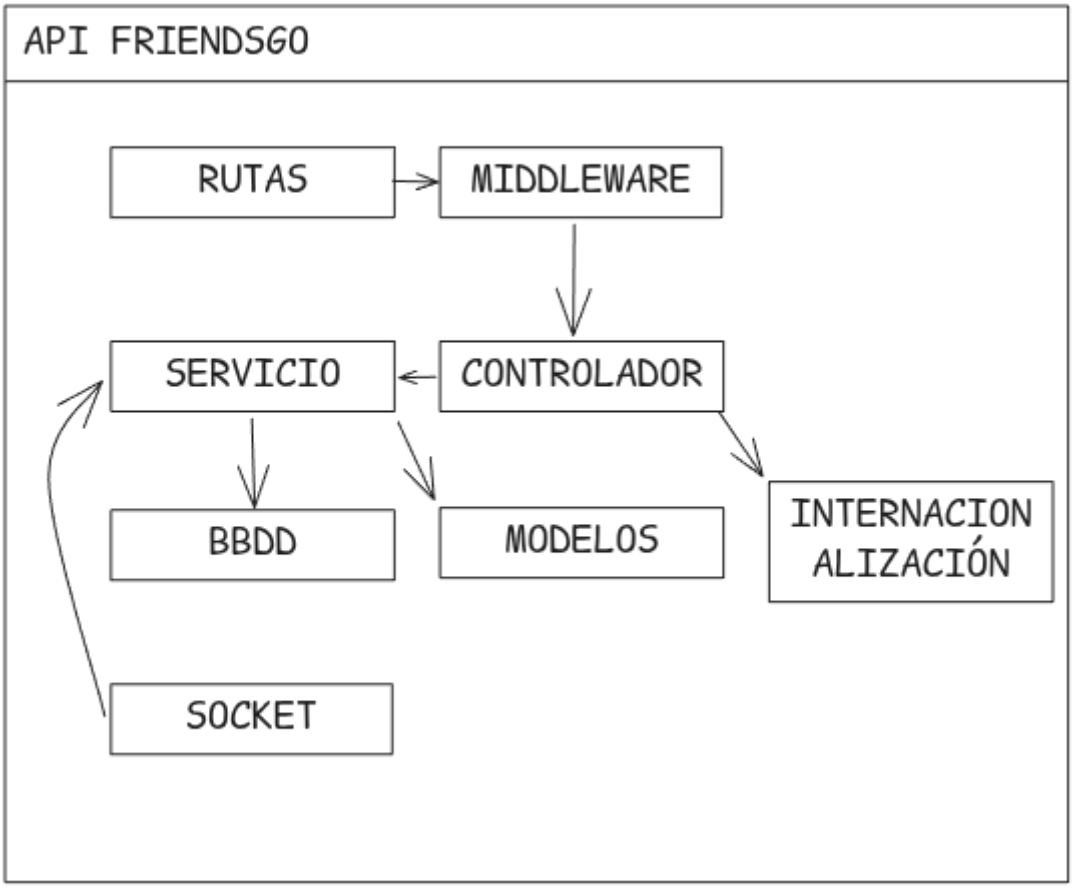
Este diagrama de estado muestra el posible recorrido cuando un usuario abre la aplicación:



El diagrama muestra el flujo de navegación de un usuario en la aplicación web. Al iniciar, el usuario abre la app web y accede a la pantalla de bienvenida o login. Desde ahí, puede autenticarse ingresando sus datos o seleccionar la opción de recuperar contraseña si ha olvidado sus credenciales. Si la autenticación es exitosa, el usuario accede al Home Feed.

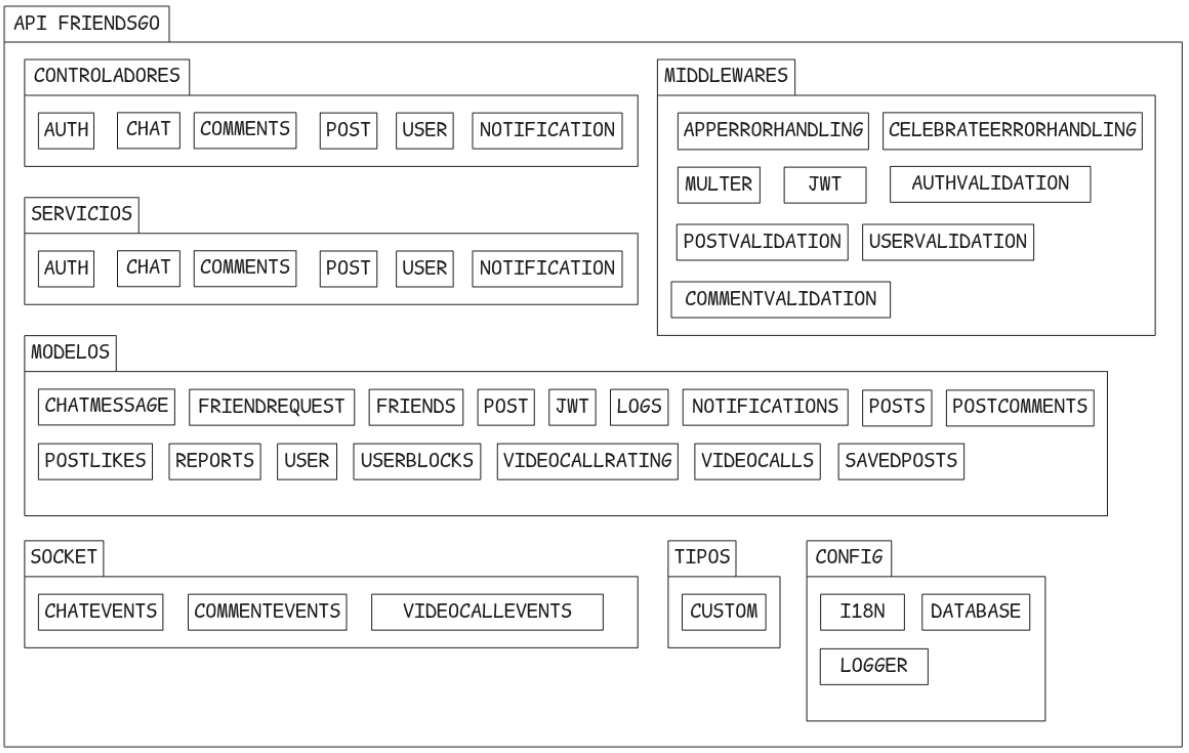
Desde el Home Feed, el usuario puede navegar hacia diferentes funcionalidades, como: publicar un post, ver y enviar mensajes, ver su perfil, abrir una videollamada aleatoria, gestionar notificaciones, realizar búsquedas, acceder a la configuración o cerrar sesión.

2.4.1 Estructura del Backend



El diagrama representa la arquitectura de la API FriendsGo, organizada en módulos bien definidos que siguen una estructura lógica y modular. El flujo inicia en las rutas, que pasan por los middleware encargados de realizar validaciones y procesamientos previos antes de llegar a los controladores. Los controladores actúan como intermediarios entre las rutas y la lógica de negocio, delegando tareas a los módulos de servicio.

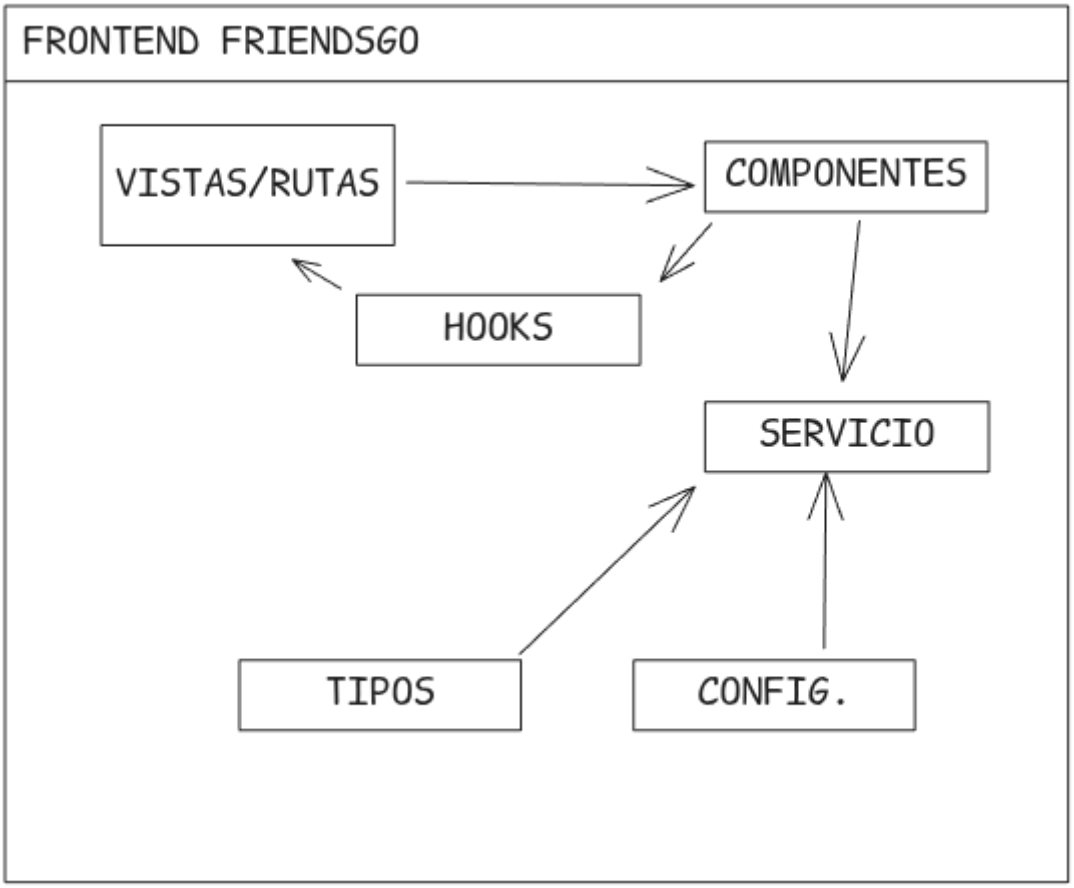
Los servicios gestionan la comunicación con la base de datos (BBDD), utilizan los modelos para estructurar los datos y se apoyan en sockets para funcionalidades en tiempo real. Además, hay un módulo de internacionalización, conectado directamente a los controladores, que permite adaptar la API a distintos idiomas. Esta arquitectura facilita el mantenimiento, la escalabilidad y la integración de nuevas funcionalidades en la aplicación.



La imagen ilustra la organización modular del backend. Se observa un directorio de **controladores** que agrupa la lógica de gestión para las principales funcionalidades como autenticación, chat, comentarios, publicaciones, usuarios y notificaciones. De forma paralela, la sección de **servicios** contiene módulos dedicados a estas mismas áreas temáticas. La carpeta de **middlewares** reúne diversas herramientas, incluyendo componentes para el manejo de errores, validación de datos para distintas entidades, gestión de subida de archivos y mecanismos de autenticación.

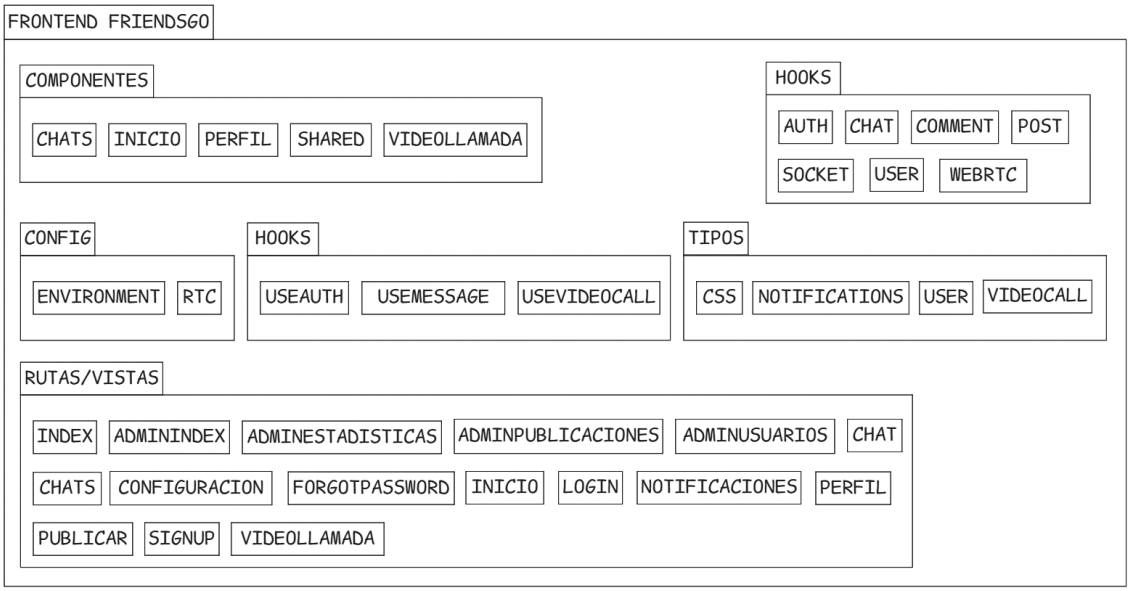
Por otro lado, los **modelos** definen las diversas entidades de datos manejadas por el sistema, incluyendo elementos como mensajes de chat, solicitudes de amistad, información de amigos, publicaciones con sus interacciones (me gusta, comentarios), JSON Web Tokens, registros de actividad, notificaciones, reportes, datos de usuario (incluyendo bloqueos), así como información relativa a videollamadas y sus valoraciones. Para la comunicación en tiempo real, un apartado de **socket** organiza los manejadores de eventos para chats, comentarios y videollamadas. Finalmente, se observan directorios para **tipos**, con definiciones personalizadas, y **config**, que almacena configuraciones esenciales del sistema como la internacionalización, los parámetros de la base de datos y la configuración del sistema de registro (logger). Esta disposición general evidencia un enfoque organizado para la gestión de los diferentes aspectos del backend.

2.4.2 Estructura del Frontend



El diagrama muestra la estructura del Frontend de FriendsGo, organizado en módulos funcionales que facilitan la separación de responsabilidades y la reutilización de código. El flujo parte desde las vistas/rutas, que definen la navegación de la aplicación y se conectan directamente con los componentes, los cuales representan las unidades visuales reutilizables de la interfaz.

Los hooks permiten manejar estados, efectos y lógica reutilizable entre componentes y vistas, fomentando un desarrollo más limpio y eficiente. Los componentes también se comunican con el módulo de servicios, encargado de gestionar las peticiones a la API. Dichos servicios se apoyan en archivos de configuración para centralizar endpoints o claves, y en los tipos, que garantizan la correcta tipificación de los datos, especialmente en entornos con TypeScript. Esta arquitectura modular promueve la mantenibilidad, escalabilidad y claridad en el desarrollo del frontend.



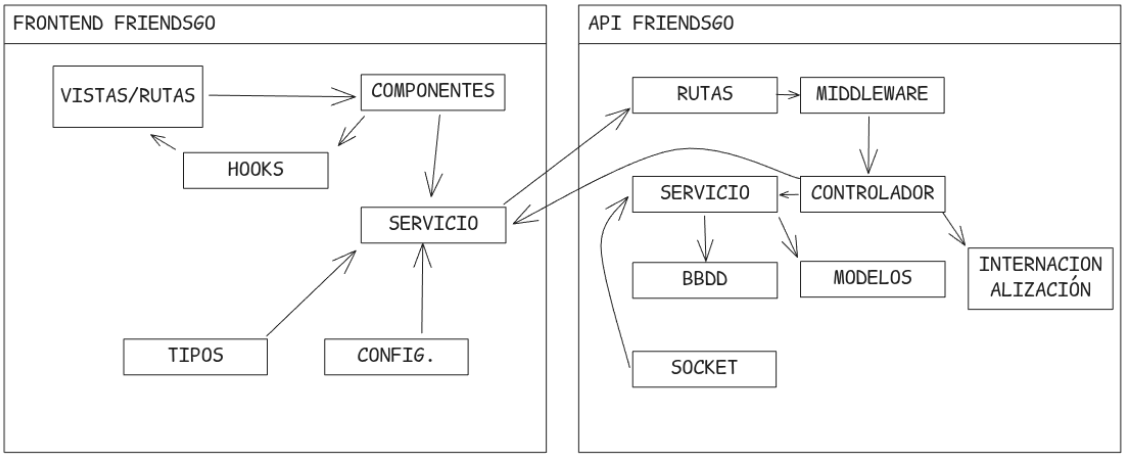
La imagen describe la arquitectura del frontend FriendsGo, organizada para mayor claridad. El directorio **componentes** es central, conteniendo subcarpetas como **chats**, **inicio**, **perfil**, **shared** y **videollamada**; cada una de estas agrupa una colección de componentes de interfaz de usuario dedicados a esa área específica de la aplicación.

La carpeta **config** reúne archivos de configuración clave, como las variables de entorno (**environment**) y ajustes para comunicación en tiempo real (RTC). Incluye también un subdirectorio HOOKS con funciones reutilizables para autenticación (**useauth**), mensajería (**usemessage**) y videollamadas (**usevideocall**), que como se mencionó, ya han sido explicadas.

El directorio RUTAS/VISTAS estructura las diferentes páginas y secciones navegables. Aquí se definen tanto las vistas principales y de usuario (como **inicio**, **perfil**, **login**, **chat**), como las secciones administrativas (**adminindex**, **adminpublicaciones**).

Un directorio **hooks** de nivel superior, ya comentado previamente, organiza lógica reutilizable por temas como **auth**, **chat**, **post**, **socket**, **user** y **webrtc**. Finalmente, la carpeta **tipos** contiene definiciones (posiblemente TypeScript) para **css**, **notifications**, **user** y **videocall**, buscando la robustez y consistencia del código. Esta organización modular favorece el desarrollo y mantenimiento del frontend.

2.4.3 Estructura general del proyecto



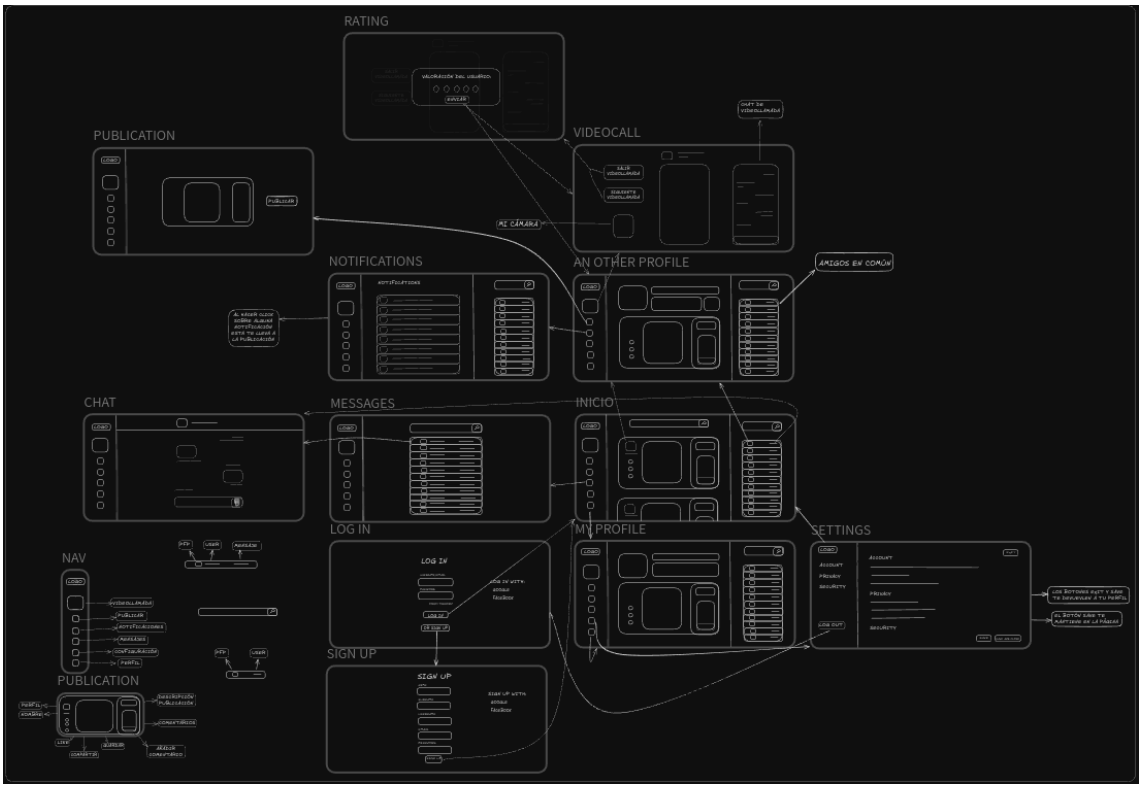
Los diagramas muestran la arquitectura general del proyecto FriendsGo, dividida en dos bloques principales: el frontend y la API backend. En el frontend, los módulos de vistas/rutas, componentes, hooks, tipos y configuración se conectan entre sí a través de un servicio centralizado que gestiona la lógica y la comunicación con la API. Esta estructura permite mantener una separación clara entre la interfaz de usuario y la lógica de negocio.

Por su parte, la API backend está compuesta por rutas, middleware, controladores, servicios, modelos, internacionalización, base de datos y sockets. Las rutas se apoyan en los controladores, que a su vez hacen uso de servicios para acceder a los datos y funcionalidades como sockets en tiempo real. El diagrama también representa cómo el frontend se comunica directamente con la capa de servicios del backend, estableciendo una integración clara y eficiente entre ambas partes del sistema.

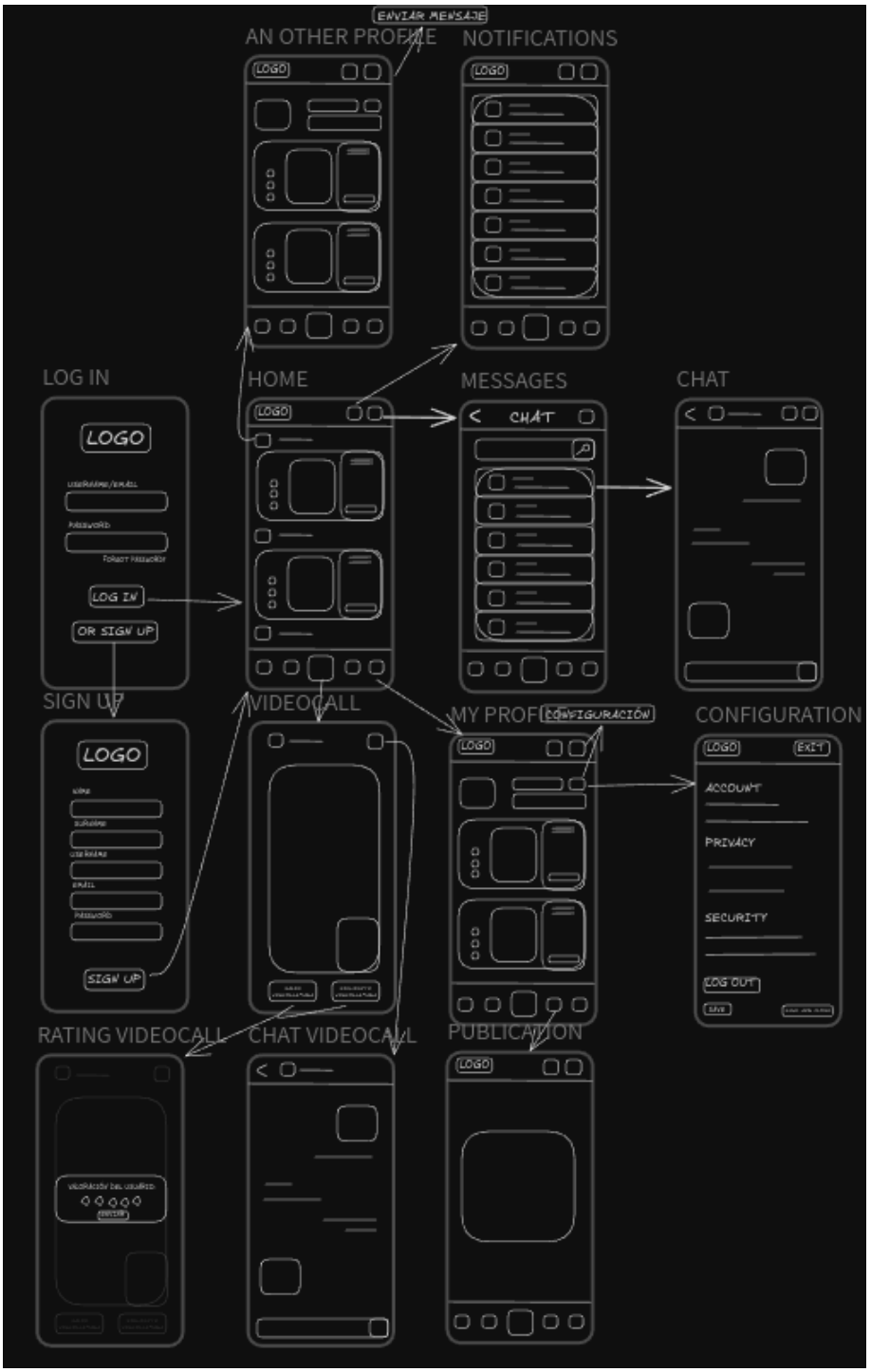
2.5 Descripción de los componentes

Como se menciona en el apartado de tecnologías, se ha utilizado Excalidraw para la creación del wireframe inicial. Esta herramienta permite desarrollar bocetos de forma rápida, clara y colaborativa, lo que facilita la visualización temprana de la estructura general del diseño. A partir de este wireframe se definieron los componentes clave que se implementarán directamente en el código, permitiendo así una transición más eficiente del diseño al desarrollo funcional.

A continuación, se presenta la propuesta de diseño para la versión de escritorio. Esta maqueta refleja la organización visual, jerarquía de la información y disposición de los elementos que compondrán la interfaz. El objetivo es ofrecer una experiencia de usuario clara, y coherente con los principios del diseño responsivo.

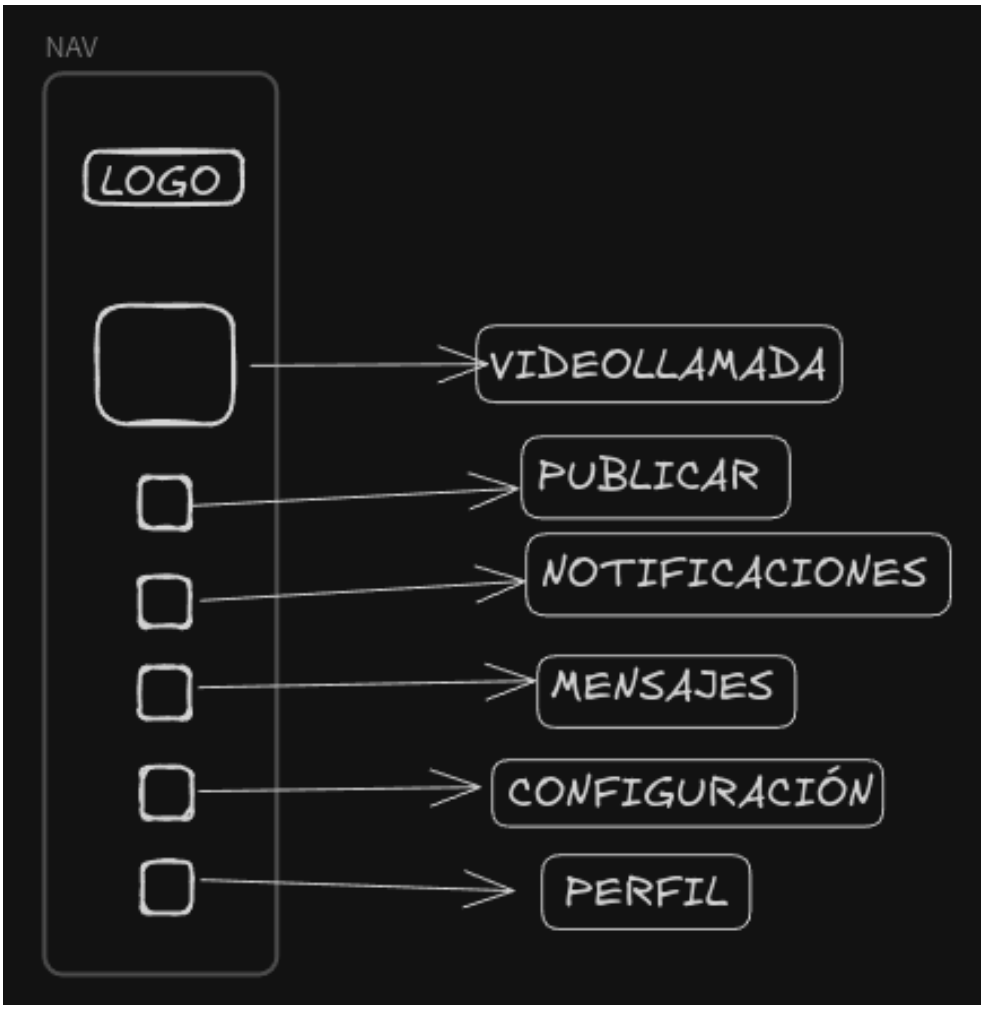


Esta es la propuesta de diseño para dispositivos móviles, adaptada para ofrecer una experiencia cómoda e intuitiva en pantallas pequeñas. Mantiene la coherencia visual con la versión de escritorio, reorganizando los elementos para priorizar la navegación.



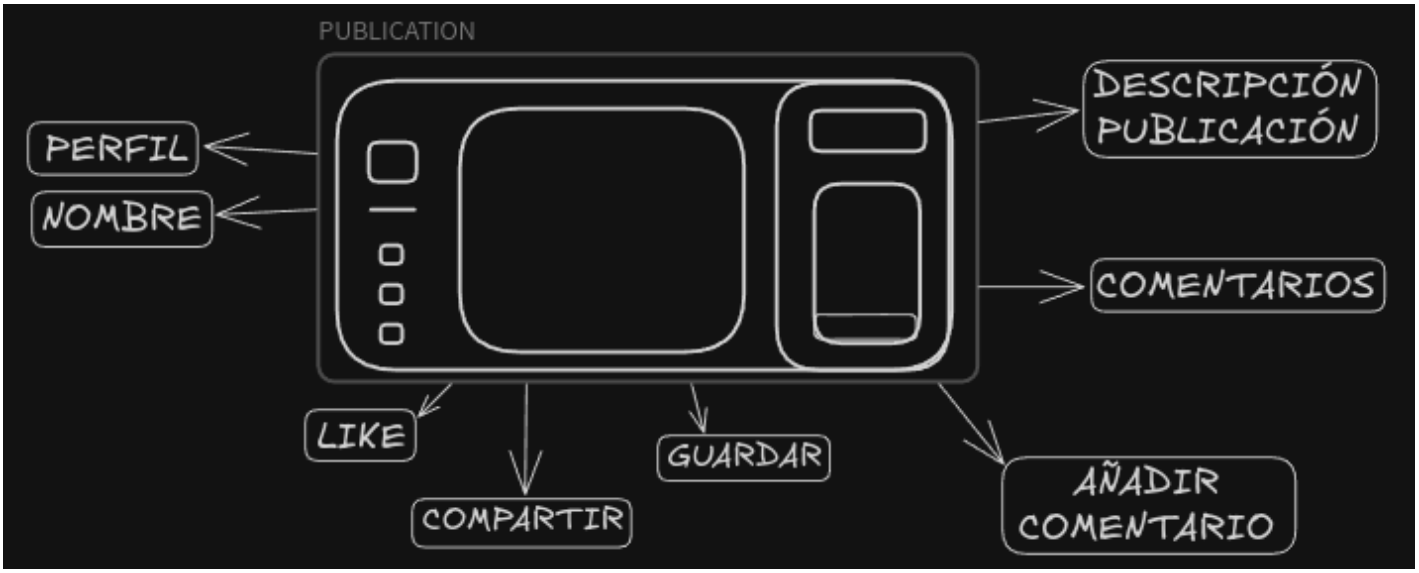
2.5.1 Barra de navegación

La barra de navegación para la versión de escritorio está diseñada para ofrecer un acceso rápido, claro y organizado a las principales secciones de la plataforma. Ubicada en la parte superior o lateral de la interfaz, presenta íconos y/o etiquetas que representan funciones clave como el inicio, perfil, búsqueda, mensajes, notificaciones y ajustes. Su diseño está optimizado para pantallas amplias, permitiendo mostrar más opciones sin sacrificar la limpieza visual. Además, incluye indicadores visuales para mostrar actividades recientes, como mensajes nuevos o notificaciones sin leer, y puede mantenerse fija al desplazarse para facilitar la navegación continua. Esta barra contribuye a una experiencia de usuario intuitiva y eficiente en entornos de escritorio.



2.5.2 Publicación para escritorio

La funcionalidad de publicación en la versión de escritorio permite a los usuarios crear y compartir contenido de forma cómoda y completa desde una interfaz amplia y estructurada. A través de un formulario accesible, los usuarios pueden redactar textos, añadir imágenes y personalizar su publicación antes de compartirla con la comunidad. El diseño está optimizado para aprovechar el espacio disponible en pantallas grandes, facilitando la vista previa del contenido, la carga de archivos y el acceso a herramientas adicionales como etiquetas o configuración de privacidad. Esta versión busca ofrecer una experiencia fluida y eficiente para la creación de contenido desde dispositivos de escritorio.

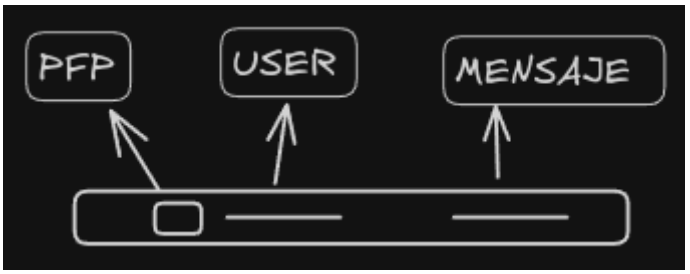


2.5.3 Buscador de escritorio y móvil

El buscador está diseñado para ofrecer una experiencia ágil y consistente tanto en la versión de escritorio como en la móvil. En escritorio, se integra de forma visible en la barra de navegación, permitiendo a los usuarios realizar búsquedas rápidas de perfiles, publicaciones u otros contenidos con facilidad gracias al mayor espacio disponible. En dispositivos móviles, el buscador se adapta a un diseño compacto y accesible, manteniendo su funcionalidad mediante iconos o menús desplegables. En ambas versiones, el sistema ofrece sugerencias automáticas y resultados en tiempo real, optimizando el rendimiento y la precisión de las búsquedas. Esta funcionalidad es clave para facilitar la exploración dentro de la plataforma, independientemente del dispositivo utilizado.

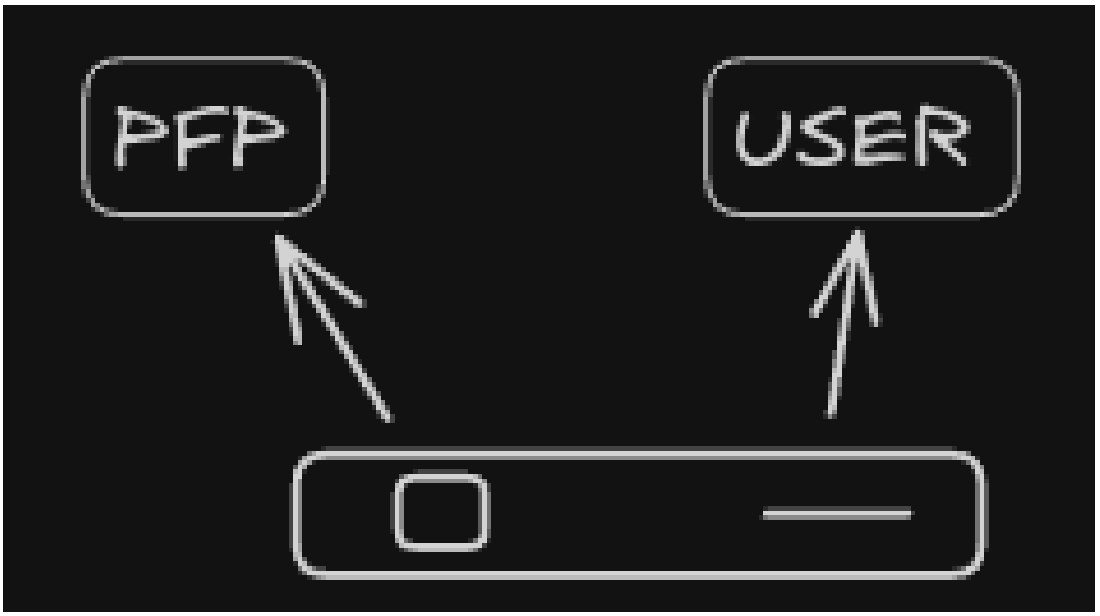
2.5.4 Acceder al chat en escritorio y móvil

El acceso al chat está optimizado para brindar una experiencia fluida tanto en la versión de escritorio como en la móvil. En escritorio, el chat se presenta como una ventana flotante o una sección dedicada dentro de la interfaz, permitiendo mantener conversaciones en tiempo real sin interrumpir la navegación. En dispositivos móviles, el chat se adapta a un diseño de pantalla completa o mediante una interfaz deslizable, garantizando comodidad y claridad en espacios reducidos. En ambas versiones, las conversaciones se actualizan en tiempo real gracias al uso de WebSockets, y se incluye un sistema de notificaciones para alertar sobre nuevos mensajes.



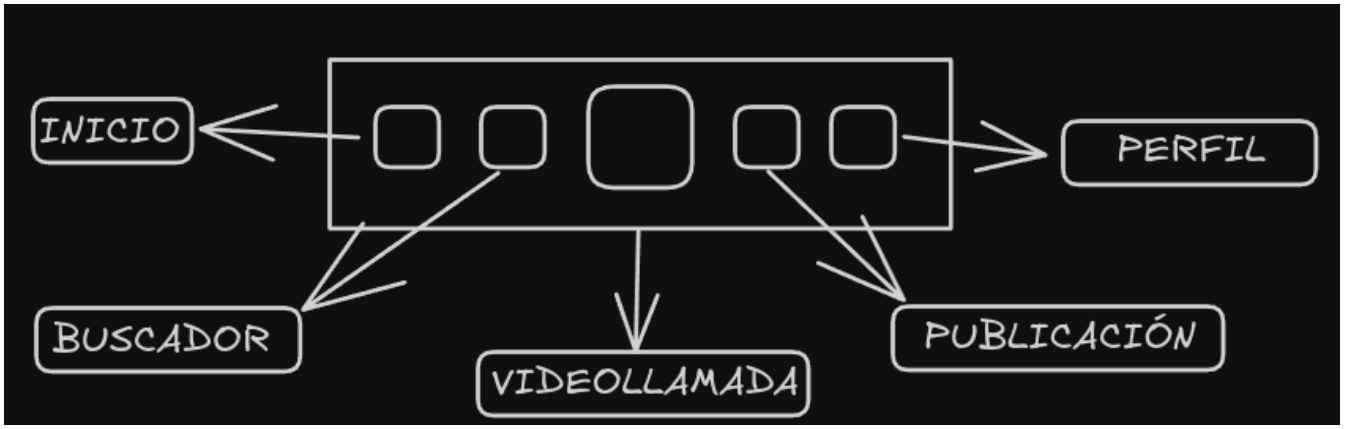
2.5.5 Amigo en escritorio y móvil

La funcionalidad de amigos está diseñada para adaptarse a la experiencia del usuario tanto en escritorio como en dispositivos móviles. En la versión de escritorio, los usuarios pueden acceder a la lista de amigos desde una sección dedicada o a través del menú principal, con una vista amplia que permite gestionar solicitudes, ver perfiles y acceder al chat de manera rápida. En la versión móvil, esta funcionalidad se presenta en un formato más compacto y accesible mediante menús desplegables o pestañas, manteniendo todas las opciones clave, como aceptar o rechazar solicitudes, buscar nuevos amigos y revisar interacciones recientes. En ambos entornos, la interfaz garantiza una navegación intuitiva, promoviendo la interacción social desde cualquier dispositivo.



2.5.6 Navegador inferior de móvil

El navegador inferior en la versión móvil está diseñado para ofrecer un acceso rápido y cómodo a las funciones principales de la plataforma. Ubicado de forma fija en la parte inferior de la pantalla, incluye íconos intuitivos que permiten al usuario moverse fácilmente entre secciones como inicio, búsqueda, publicaciones, notificaciones y perfil. Este diseño mejora la usabilidad en pantallas pequeñas, permitiendo la navegación con una sola mano y reduciendo la necesidad de desplazarse por la interfaz. Además, puede incluir indicadores visuales para notificaciones nuevas, aportando una experiencia más dinámica e interactiva.

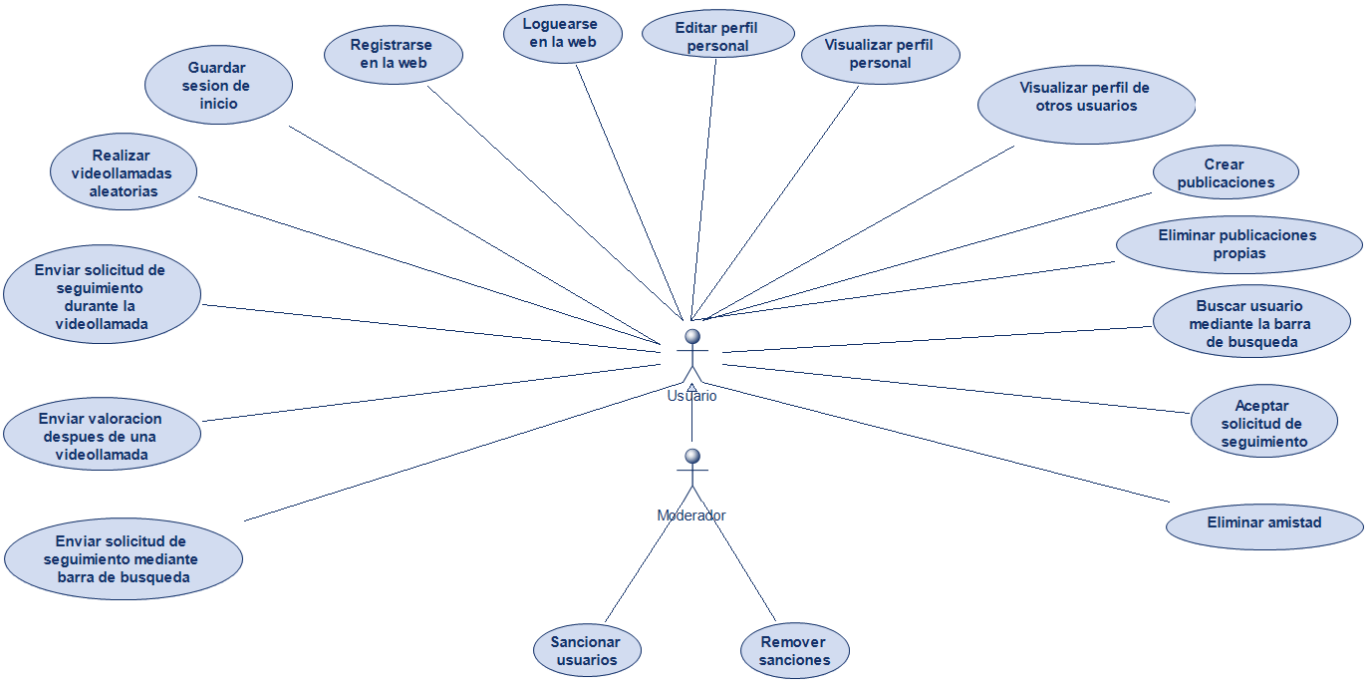


2.5.7 Navegador superior de movil

El navegador superior en la versión móvil está diseñado para facilitar el acceso rápido a funciones esenciales y mejorar la experiencia de navegación en pantallas pequeñas. Ubicado en la parte superior de la interfaz, suele incluir elementos como el logo de la plataforma, un botón de menú desplegable (hamburguesa), íconos de búsqueda, notificaciones y acceso rápido al perfil del usuario. Este diseño compacto y accesible permite a los usuarios acceder fácilmente a opciones y herramientas clave sin ocupar demasiado espacio visual, optimizando la usabilidad y manteniendo una navegación clara y eficiente en dispositivos móviles.

2.7 Definición de las funcionalidades

Para facilitar la comprensión de las funcionalidades principales de la aplicación, se presenta a continuación un diagrama de casos de uso que resume de forma clara las interacciones clave entre los usuarios y el sistema.



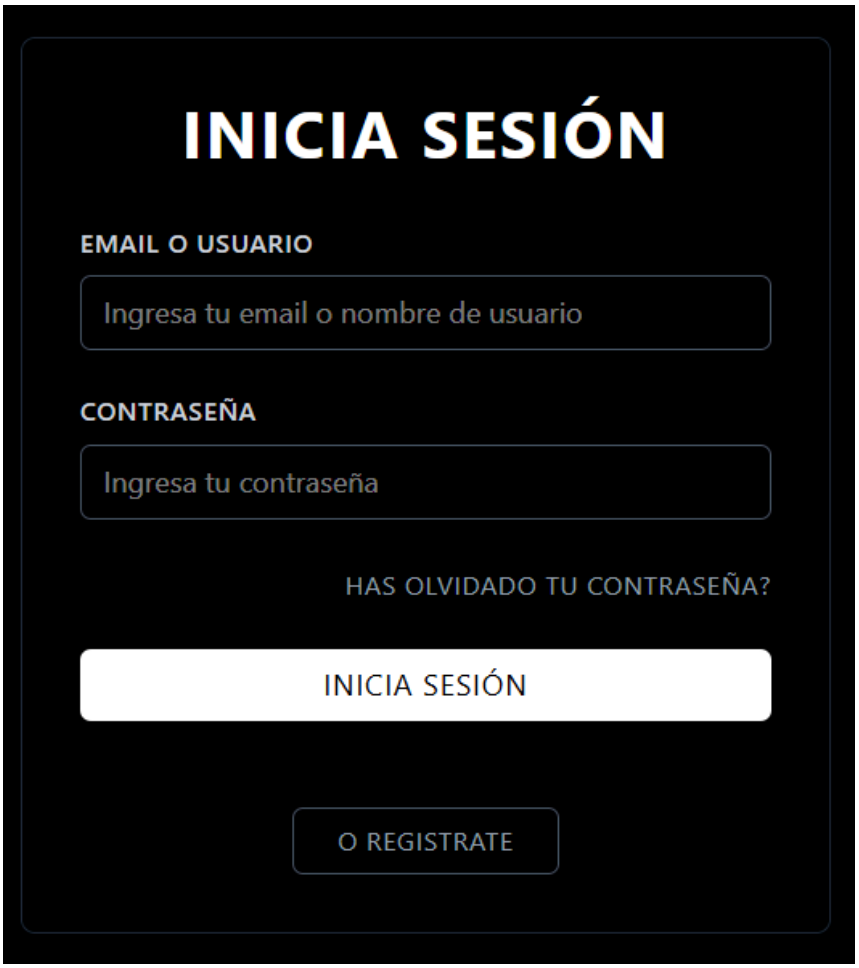
El diagrama de caso de uso ilustra las funcionalidades principales de nuestra aplicación y cómo interactúan los diferentes roles involucrados. El usuario tiene acceso a una variedad de características, como la gestión de su cuenta, la interacción social mediante publicaciones y videollamadas aleatorias, y el envío de solicitudes de seguimiento. Además, puede reportar comportamientos inapropiados y valorar sus interacciones. Por otro lado, el moderador tiene la responsabilidad de supervisar el contenido, revisar reportes y gestionar sanciones, garantizando así un entorno seguro y adecuado para todos los usuarios.

2.7.1 Registro de usuario

El proceso de registro permite a los usuarios crear una cuenta ingresando información básica como nombre de usuario, nombre, apellido, correo electrónico y una contraseña. Una vez completado el formulario, el sistema valida que los datos proporcionados cumplan con los criterios establecidos para garantizar la seguridad y la integridad de la información. En particular, la contraseña debe tener al menos ocho caracteres e incluir al menos una letra mayúscula, un dígito y número, con el fin de fortalecer la protección frente a accesos no autorizados. Este proceso asegura que solo usuarios legítimos puedan acceder a la plataforma.

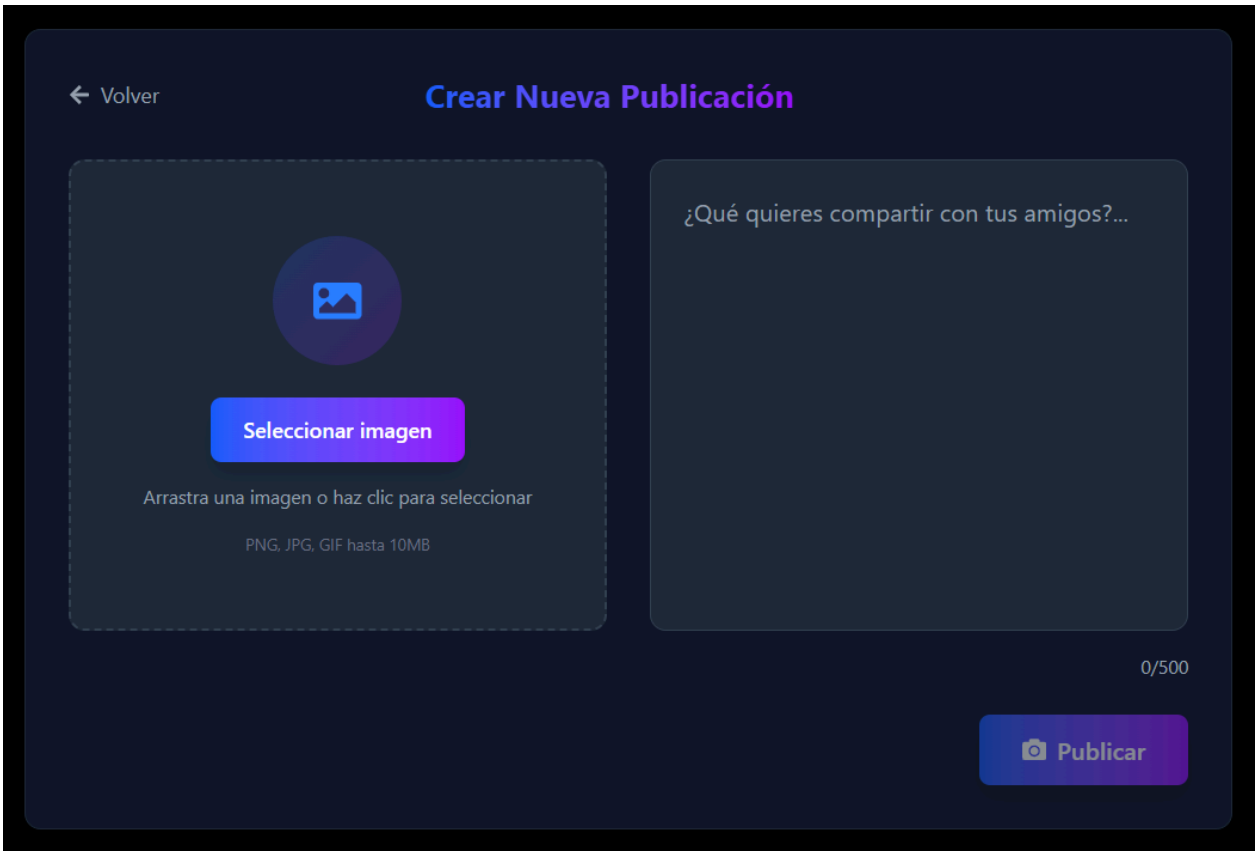
2.7.2 Login

Los usuarios pueden iniciar sesión ingresando su correo electrónico y contraseña, los cuales son validados por el sistema comparándolos con los registros almacenados en la base de datos. Si las credenciales son correctas, se genera un token de sesión que otorga acceso a la plataforma. Este token se almacena de forma segura en el navegador, permitiendo mantener la sesión activa hasta que el usuario cierre sesión manualmente o el token expire. Para garantizar la seguridad de las contraseñas, el sistema emplea algoritmos de hash robustos como bcrypt, lo que impide que las contraseñas se almacenen o transmitan en texto plano.



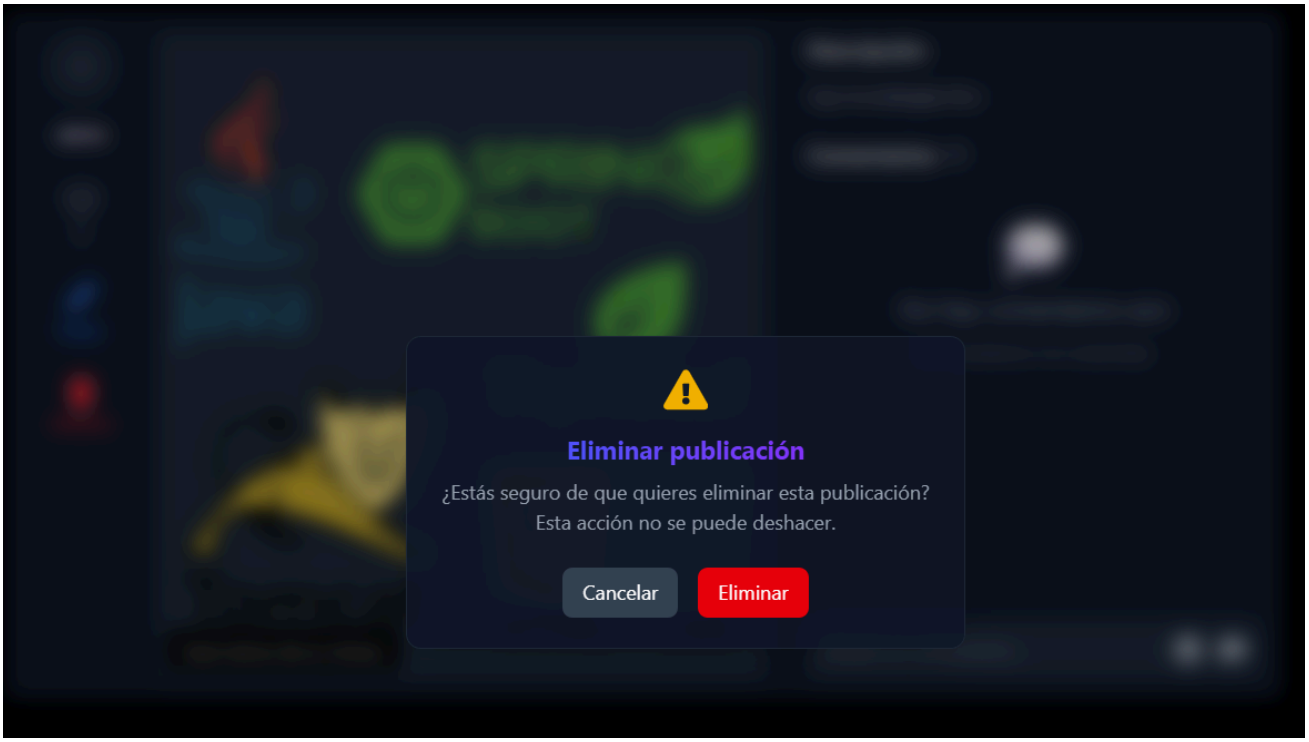
2.7.3 Crear Publicaciones

Los usuarios pueden compartir contenido, como texto o imágenes, a través de una interfaz intuitiva y fácil de usar. Antes de almacenar los archivos, el sistema realiza una validación para asegurar que el formato y el tamaño cumplan con los requisitos establecidos, garantizando así la integridad del contenido y un rendimiento óptimo de la aplicación.



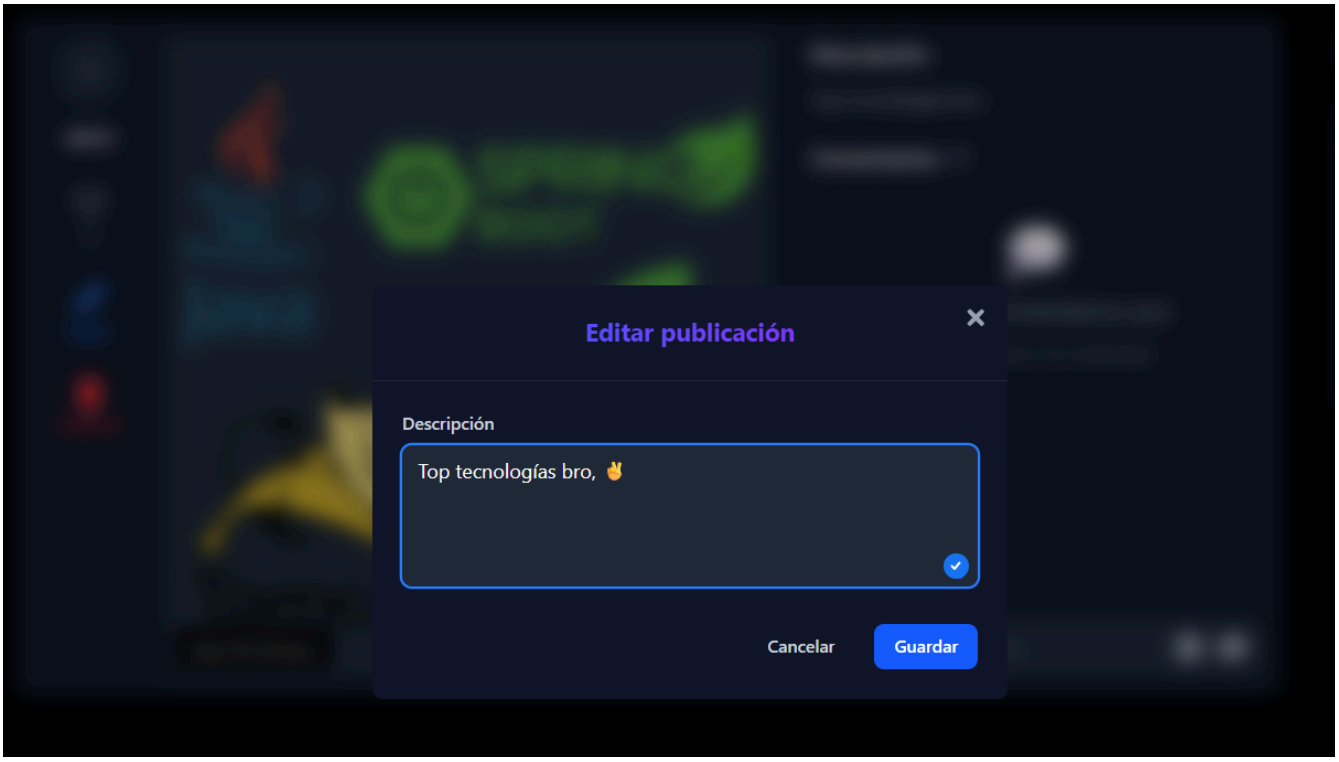
2.7.4 Remover publicaciones

La aplicación permite a los usuarios eliminar sus publicaciones en cualquier momento, brindándoles control total sobre el contenido que comparten. Al seleccionar esta opción, el sistema solicita una confirmación mediante un mensaje emergente, con el fin de prevenir eliminaciones accidentales y asegurar que la acción sea intencional. Esta funcionalidad no solo mejora la experiencia del usuario al ofrecer mayor flexibilidad, sino que también refuerza buenas prácticas de usabilidad al incorporar una capa de verificación antes de ejecutar acciones irreversibles.



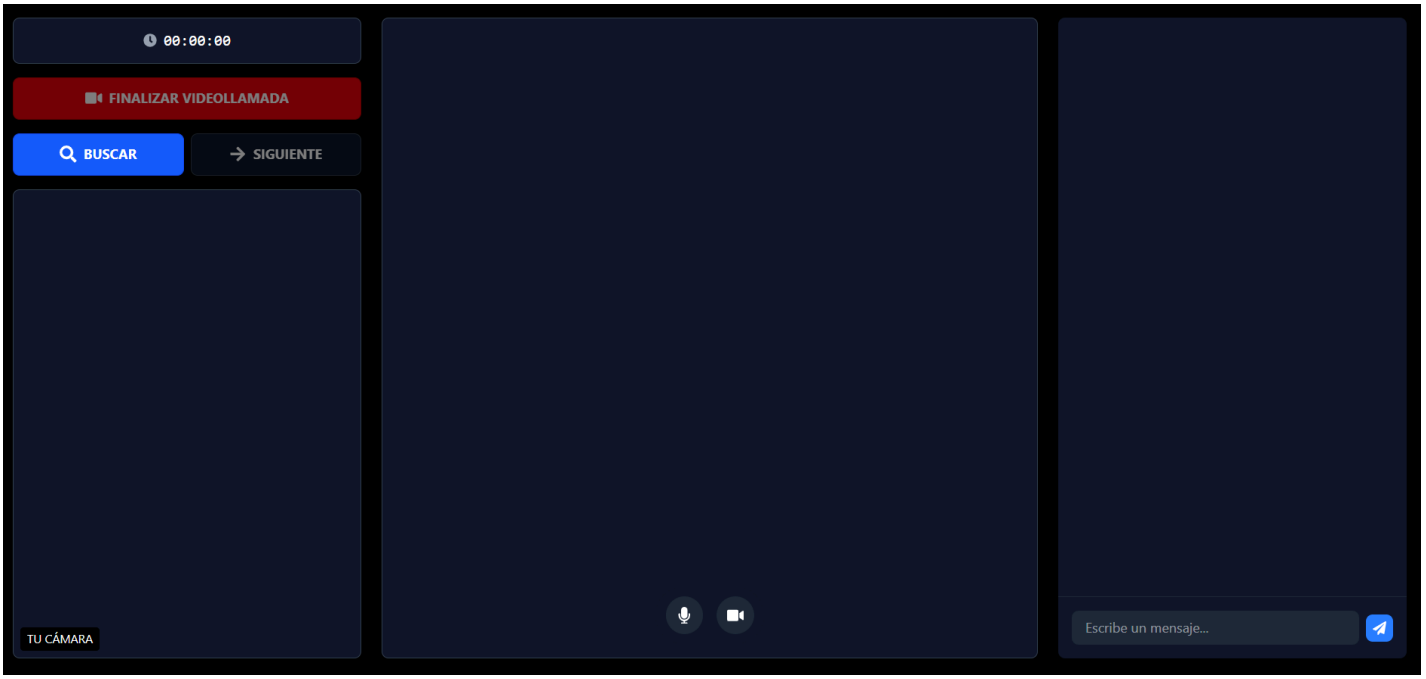
2.7.5 Editar una publicación

La aplicación ofrece a los usuarios la posibilidad de editar sus publicaciones, permitiéndoles modificar la descripción del contenido después de haberlo compartido. Esta funcionalidad resulta útil en caso de errores, cambios de contexto o para actualizar la información publicada. Al seleccionar la opción de edición, se carga un formulario con el contenido actual, facilitando la modificación sin necesidad de rehacer toda la publicación. Una vez guardados los cambios, el sistema actualiza la base de datos y refleja la nueva versión de la descripción, asegurando una experiencia flexible y dinámica para el usuario



2.7.6 Videollamadas Aleatorias

Los usuarios pueden conectarse en videollamadas aleatorias con otros participantes disponibles en la plataforma. Al activar esta opción, el sistema utiliza tecnologías como WebRTC para establecer y gestionar la comunicación en tiempo real, garantizando una conexión estable y de baja latencia. El proceso de emparejamiento se realiza automáticamente, asignando a los usuarios con interlocutores disponibles para ofrecer una experiencia rápida, dinámica y fluida desde el primer momento.

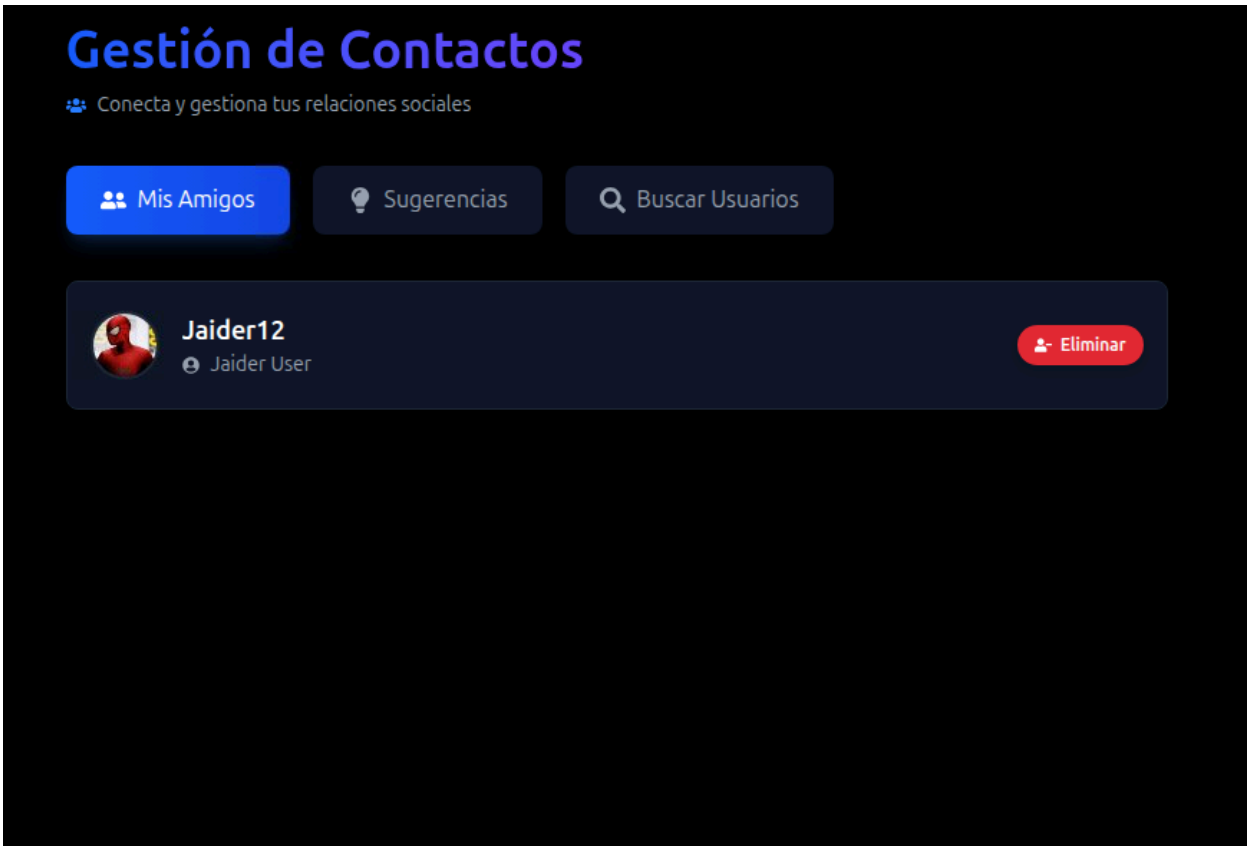


2.7.7 Enviar Solicitud de Seguimiento mediante Videollamada

Durante una videollamada, el sistema ofrece a los usuarios la posibilidad de enviar una solicitud de amistad al interlocutor. Esta funcionalidad permite fomentar conexiones significativas entre los participantes de la llamada. Cuando el usuario selecciona esta opción, el sistema registra la solicitud en una tabla de la base de datos, asignándole un estado inicial de "pendiente". Simultáneamente, el receptor recibe una notificación en tiempo real que le permite aceptar o rechazar la solicitud desde su panel de notificaciones. Este mecanismo asegura una experiencia fluida y organizada para la gestión de nuevas amistades.

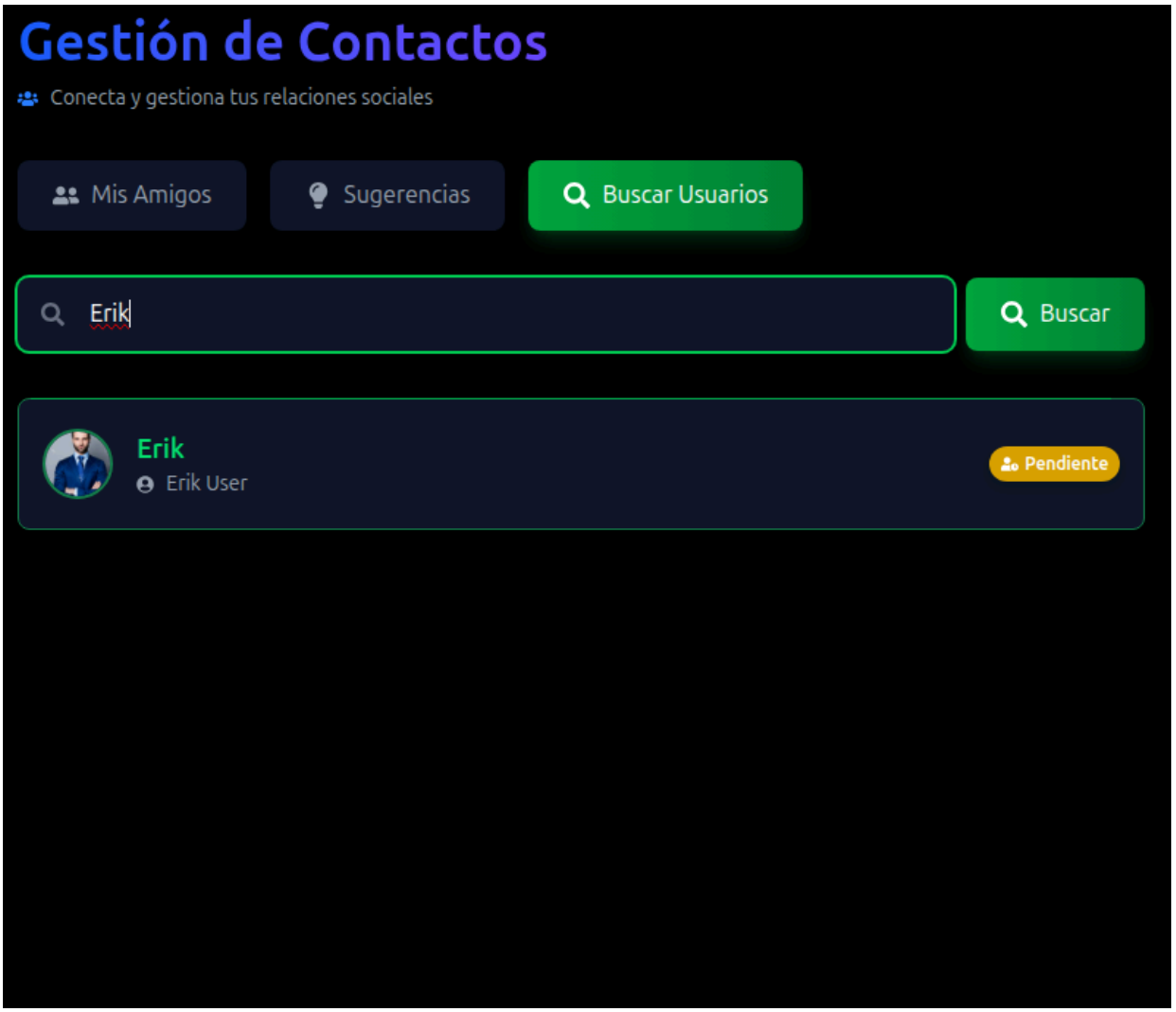
2.7.8 Búsqueda de Usuarios

La barra de búsqueda es una herramienta esencial que permite a los usuarios localizar perfiles específicos de otros miembros de la plataforma. Pueden buscar por nombre o apodo, los resultados se generan de forma casi instantánea, proporcionando una experiencia de búsqueda eficiente y precisa.



2.7.9 Enviar Solicitud de Seguimiento desde la barra de búsqueda

Una vez que los usuarios encuentran un perfil a través de la barra de búsqueda, tienen la opción de enviar una solicitud de amistad directamente desde los resultados. Este proceso es intuitivo: con un solo clic, el sistema registra la solicitud en la base de datos, asegurándose de mantener un historial organizado de solicitudes pendientes, aceptadas o rechazadas. Las solicitudes pendientes son notificadas al receptor, quien puede gestionarlas desde un panel diseñado específicamente para este propósito.

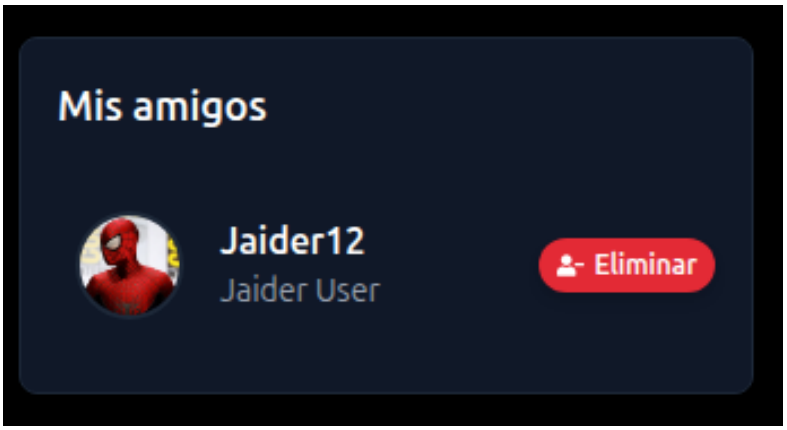


2.7.10 Aceptar Solicitud de seguimiento

Esta funcionalidad permite a los usuarios gestionar las solicitudes de amistad recibidas de manera sencilla. Los usuarios pueden aceptar la solicitud para establecer una conexión bidireccional que se registra en la base de datos o rechazarla si lo consideran oportuno. Este diseño garantiza que las relaciones entre usuarios se construyan de forma consensuada y transparente.

2.7.11 Remover Amistad

Los usuarios tienen la libertad de eliminar amistades si así lo desean. Al eliminar una amistad, el sistema actualiza automáticamente la base de datos, deshaciendo la conexión entre ambas partes. La funcionalidad también asegura que, una vez eliminada la amistad, los usuarios ya no puedan interactuar, respetando los niveles de privacidad establecidos.

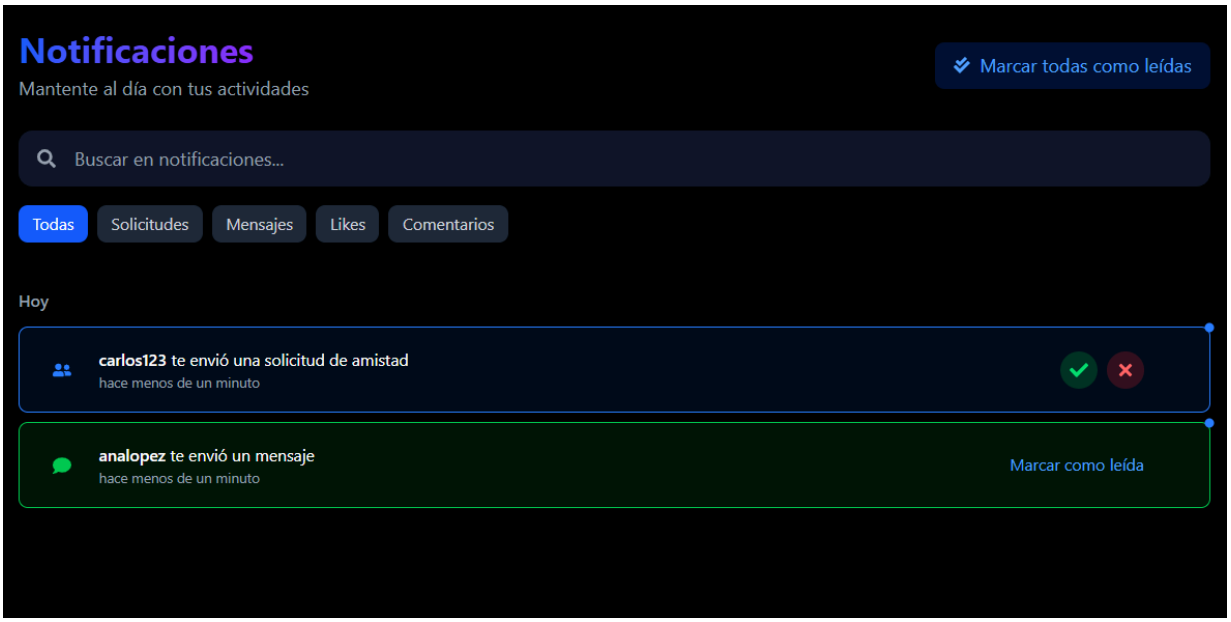


2.7.12 Valoración tras videollamadas

Después de finalizar una videollamada, los usuarios tienen la oportunidad de valorar a su interlocutor utilizando un sistema de puntuación, como estrellas o comentarios. Estas valoraciones son públicas y se reflejan en el perfil del usuario valorado, sirviendo como un indicador de la calidad de sus interacciones. Esta funcionalidad fomenta un comportamiento positivo y proporciona a los usuarios información adicional antes de interactuar con otros.

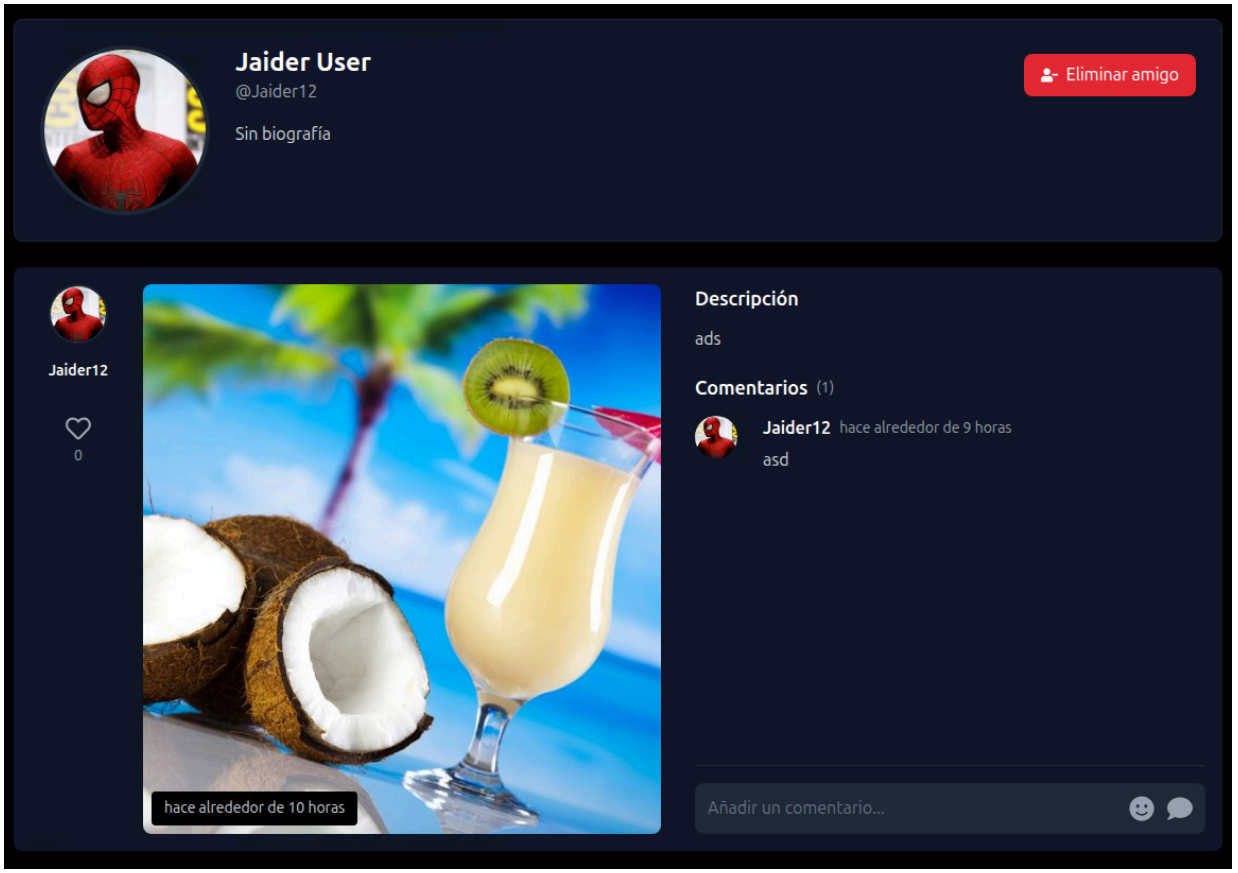
2.7.13 Notificaciones en tiempo real

El sistema genera notificaciones instantáneas para mantener a los usuarios informados sobre eventos importantes, como solicitudes de amistad, comentarios en publicaciones, y más. Estas notificaciones se agrupan en una sección específica para facilitar su visualización. Además, el sistema utiliza tecnologías como WebSockets para garantizar que las notificaciones lleguen en tiempo real, mejorando significativamente la experiencia del usuario.



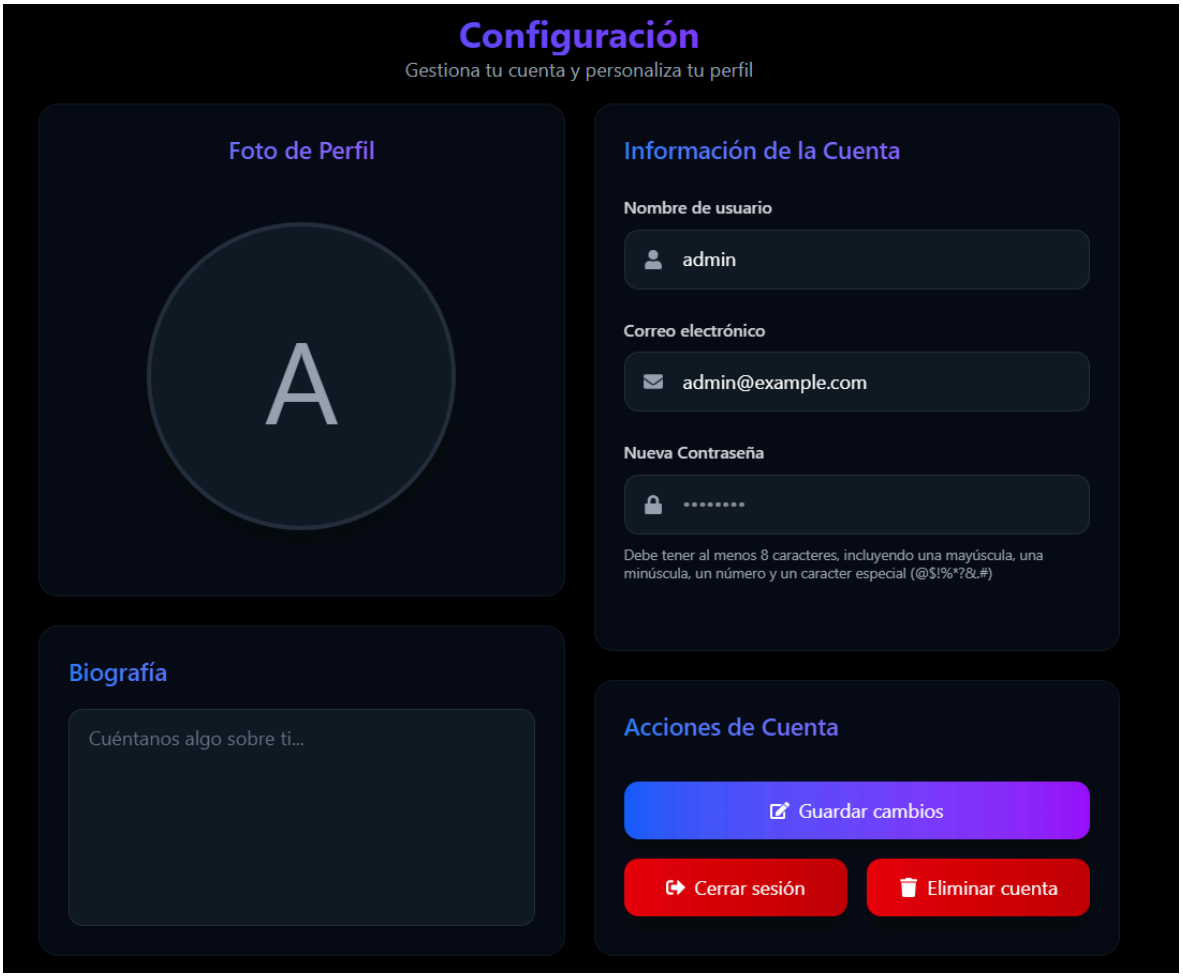
2.7.14 Visualización de Perfiles

Los usuarios pueden explorar libremente los perfiles de otros miembros de la plataforma, lo que fomenta la interacción y el descubrimiento dentro de la comunidad. Cada perfil presenta información clave, como el nombre de usuario, una breve descripción personal, las publicaciones realizadas y las valoraciones recibidas por otros usuarios. Esta vista permite conocer mejor la actividad y reputación de cada miembro, ofreciendo una experiencia más personalizada y enriquecedora al interactuar con la plataforma. Además, el diseño de los perfiles está pensado para ser claro y accesible, facilitando la navegación y visualización del contenido compartido.



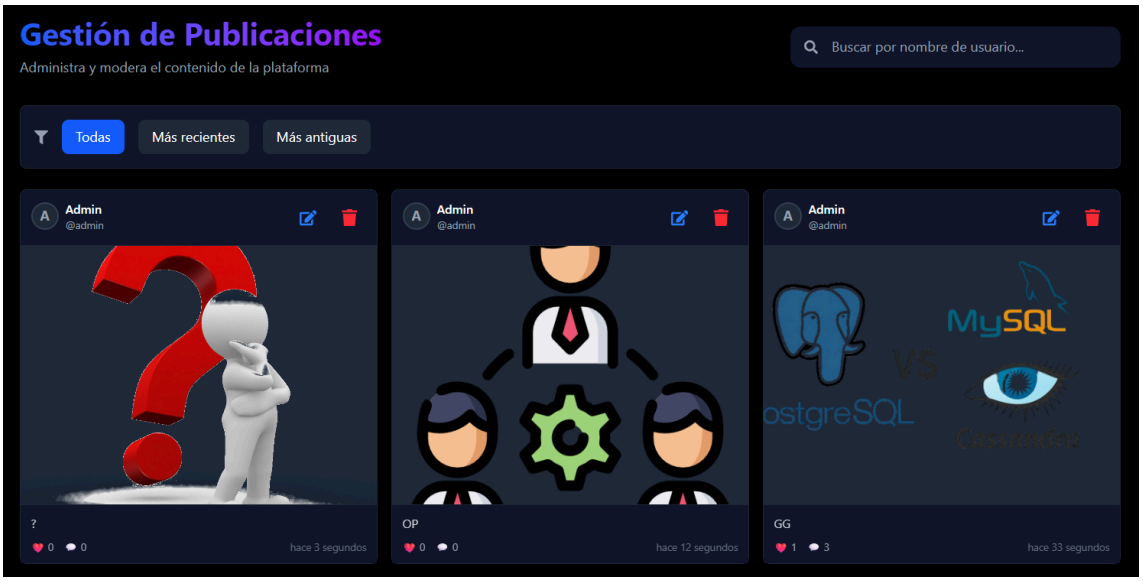
2.7.15 Editar Perfil de usuario

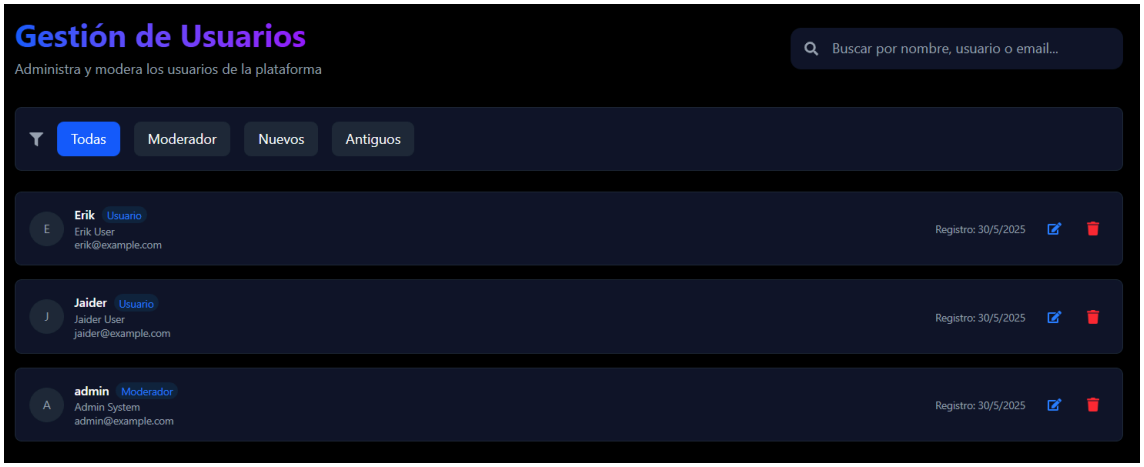
Esta funcionalidad permite a los usuarios personalizar su perfil actualizando información como nombre, biografía y foto de perfil. Los cambios realizados se reflejan inmediatamente en la base de datos y son visibles para otros usuarios según las configuraciones de privacidad.



2.7.16 Moderación de Contenido

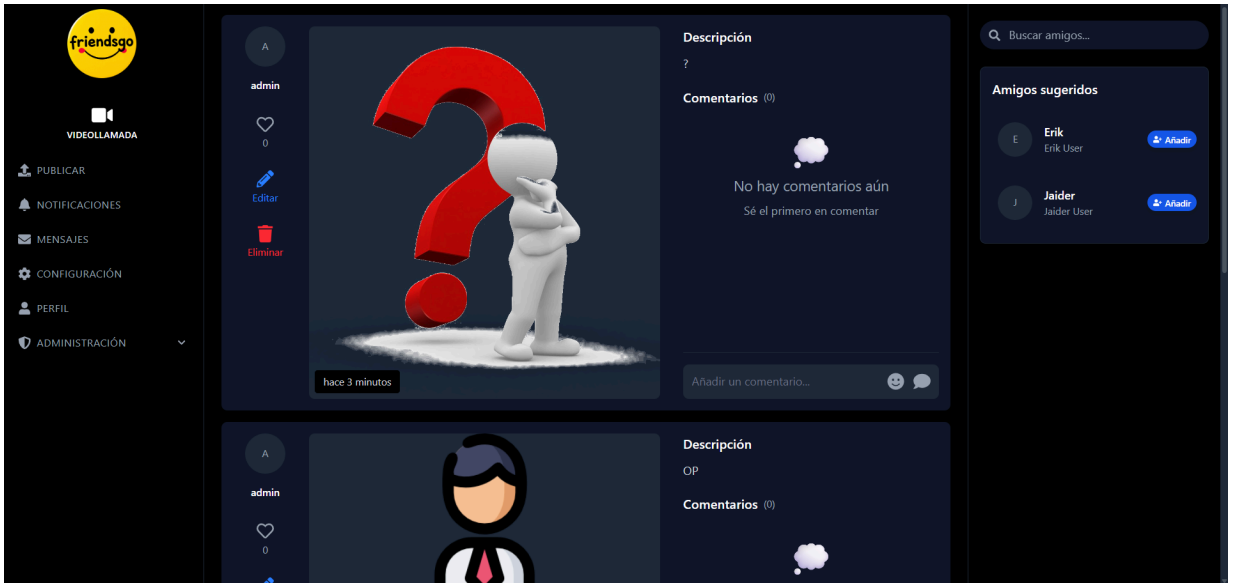
El equipo de staff cuenta con herramientas avanzadas para tomar decisiones sobre contenido inapropiado o usuarios que infringen las normas. Desde un panel administrativo, pueden eliminar publicaciones, suspender cuentas o aplicar sanciones según sea necesario.





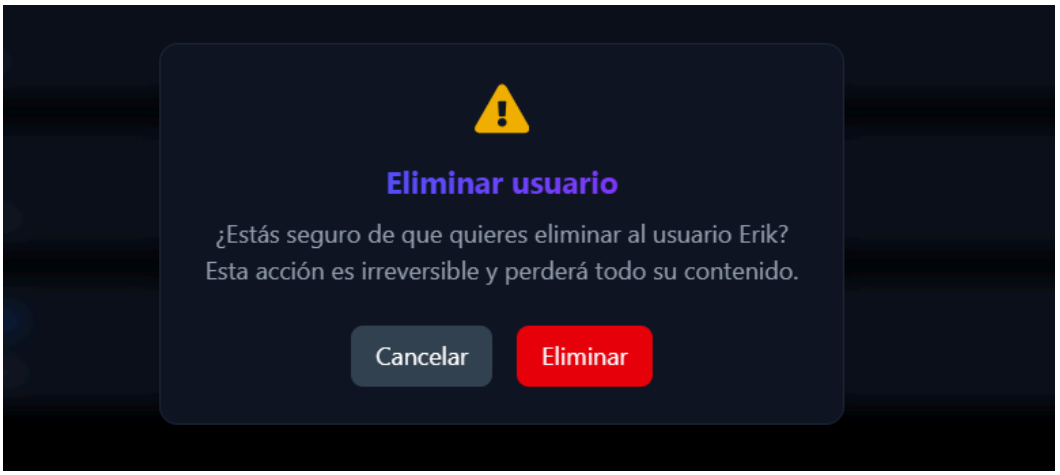
2.7.17 Feed Personalizado

El feed de la plataforma utiliza algoritmos avanzados de filtrado colaborativo para mostrar contenido relevante a cada usuario. Las publicaciones se ordenan en función de las conexiones, intereses y actividad previa del usuario, proporcionando una experiencia única y personalizada que fomenta la interacción.



2.7.18 Sancionas Usuarios

El staff de la plataforma tiene la autoridad para aplicar sanciones a aquellos usuarios que infrinjan las normas de convivencia o los términos de uso. Estas sanciones pueden variar en severidad, incluyendo bloqueos temporales o permanentes, restricciones en el uso de ciertas funcionalidades, y otras medidas disciplinarias según la gravedad de la infracción. Cada sanción se registra de forma detallada en la base de datos, lo que permite mantener un historial claro y transparente del comportamiento del usuario. Además, los usuarios sancionados reciben una notificación con información específica sobre la infracción cometida, la duración o alcance de la sanción, y, en caso de ser posible, las acciones necesarias para resolver la situación. Esta gestión busca mantener un ambiente seguro, respetuoso y justo para todos los miembros de la comunidad.



2.7.19 Remover Sanciones

En caso de que un usuario sancionado demuestre una mejora sostenida en su comportamiento, el equipo de moderación tiene la facultad de eliminar las sanciones que le hayan sido previamente aplicadas. Esta funcionalidad actualiza automáticamente la base de datos, restableciendo el estado del usuario dentro de la plataforma. Con ello, se promueve un entorno más justo y humano, donde se valora el cambio positivo y se brinda una segunda oportunidad a quienes deciden actuar de forma responsable. Esta política no solo fortalece la comunidad, sino que también incentiva el cumplimiento de las normas desde una perspectiva constructiva.

2.7.20 Sistema de chat

El sistema de chat de texto permitirá a los usuarios comunicarse en tiempo real, ya sea después de interactuar mediante videollamadas o al establecer una relación de amistad dentro de la plataforma. Los mensajes se enviarán y recibirán de forma instantánea, garantizando una experiencia fluida y dinámica. Gracias al uso de WebSockets, el sistema ofrecerá notificaciones en tiempo real, asegurando que los usuarios estén siempre al tanto de nuevas interacciones. Todos los mensajes estarán cifrados, lo que garantiza la privacidad y seguridad de las conversaciones. Además, el historial de chats será accesible y persistente, almacenado en una base de datos que permitirá a los usuarios consultar sus conversaciones anteriores en cualquier momento, fortaleciendo la continuidad en la comunicación y mejorando la experiencia general dentro de la plataforma.

3 Conclusiones

3.1 Conclusiones generales del proyecto

El desarrollo del proyecto FriendsGo nos ha aportado un valor significativo tanto a nivel académico como profesional. A lo largo de su realización, hemos tenido la oportunidad de aplicar de forma práctica los conocimientos adquiridos durante el ciclo formativo, integrando tecnologías modernas como React, Remix, Node.js, WebRTC, Socket.io y PostgreSQL en un entorno real y funcional.

Este proceso nos ha permitido reforzar habilidades técnicas clave, como el diseño de interfaces, la programación backend y frontend, el manejo de bases de datos relacionales y la comunicación en tiempo real. Además, hemos ampliado nuestra experiencia en metodologías ágiles como Scrum, gestionando el proyecto mediante sprints semanales que nos han ayudado a organizarnos de forma eficiente y cumplir con los objetivos en los plazos establecidos.

Desde el punto de vista profesional, FriendsGo ha sido una experiencia enriquecedora, ya que nos ha enfrentado a desafíos reales del desarrollo web: desde el diseño inicial hasta la implementación de funcionalidades complejas como videollamadas aleatorias y chats instantáneos. También nos ha permitido trabajar en equipo, tomar decisiones técnicas fundamentadas y adaptar nuestras soluciones a las necesidades de los usuarios.

En definitiva, este proyecto ha consolidado nuestras competencias como desarrolladores y nos ha preparado para afrontar futuros retos en el ámbito profesional con una base sólida, tanto técnica como organizativa.

3.2 Conclusión de los objetivos

A continuación, se presenta un análisis del grado de cumplimiento de los objetivos establecidos al inicio del proyecto, diferenciando entre los objetivos de la aplicación y los objetivos personales como desarrolladores.

Objetivos de la aplicación:

- Se implementó utilizando WebRTC y Socket.io, permitiendo conexiones aleatorias entre usuarios en tiempo real. El sistema funciona correctamente, aunque presenta áreas de mejora en cuanto a estabilidad y control de errores.
- Se logró desarrollar una interfaz clara, ordenada y fácil de utilizar, tanto en escritorio como en dispositivos móviles.
- Se utilizó PostgreSQL con una estructura relacional bien definida, relaciones correctamente establecidas y una normalización adecuada para garantizar la integridad de los datos.
- Se desarrolló una API RESTful con Express que permite una comunicación fluida y eficiente entre frontend y backend.
- Se implementó un sistema de chat funcional en tiempo real usando Socket.io, permitiendo a los usuarios comunicarse de forma directa e instantánea.
- Los usuarios pueden publicar contenido que incluya texto e imágenes, con almacenamiento y visualización adecuados dentro de la plataforma.
- Se implementó un sistema de autenticación que permite el registro de nuevos usuarios y el acceso mediante credenciales validadas, con gestión de sesiones.

Objetivos de los desarrolladores:

- Las tecnologías como TypeScript, HTML, CSS, etc., se usaron de forma intensiva en el desarrollo, reforzando tanto la base técnica como la experiencia práctica.
- Se trabajó activamente con ambos frameworks, enfrentando desafíos reales de integración, estructura y mantenimiento.
- Se logró implementar funcionalidades importantes como videollamadas y chat en tiempo real usando estas tecnologías. Sin embargo, no se ha podido implementar el sistema de notificaciones en tiempo real en todo el ámbito de la aplicación, lo cual queda pendiente para futuras mejoras.

3.3 Valoración de la metodología y planificación

Desde el inicio del desarrollo del proyecto decidimos utilizar una metodología ágil, concretamente Scrum, para organizar el trabajo en sprints semanales. La herramienta utilizada para la gestión de tareas fue Taiga.io, que nos permitió visualizar el progreso de forma clara y repartir el trabajo entre los miembros del equipo.

Esta elección fue muy acertada, ya que nos permitió adaptarnos a los cambios que surgieron durante el desarrollo, ya fueran imprevistos técnicos o nuevas ideas que aportaban valor al proyecto. Gracias a la flexibilidad de Scrum, pudimos reorganizar prioridades sin perder el enfoque general.

Aunque en ciertos momentos fue necesario ajustar la planificación inicial por ejemplo, al encontrarnos con dificultades técnicas o al añadir funcionalidades no previstas, el uso de sprints nos ayudó a mantener un ritmo de trabajo constante y a centrar nuestra atención en los objetivos principales.

En general, la planificación fue respetada en gran medida y resultó fundamental para alcanzar una versión funcional del proyecto dentro del plazo establecido. Consideramos que tanto la metodología como la organización empleada fueron adecuadas para el tipo de proyecto y el tamaño del equipo.

4 Visión de futuro

La visión de futuro del proyecto FriendsGo abarca una gran gama de funcionalidades, algunas son de las que no se han podido implementar al momento de presentar la aplicación web y algunas funcionalidades más que se han ido pensando a medida que pasaba el tiempo.

Estas funcionalidades son:

- Añadir llamadas y videollamadas entre usuarios (amigos)
- Que los eventos de socket para interacción en tiempo de real abarquen toda la aplicación
- Que se puedan enviar imágenes y videos en los chats entre usuarios
- Implementar *historias* (imágenes o videos que publica un usuario y tiene un tiempo de visión de 24 horas)
- Hacer una aplicación móvil, tanto para android como para iOS
- Implementar los reportes de usuarios y publicaciones
- Implementar el guardado de publicaciones
- Inicio de sesión y registro a partir de la API de google y facebook

5. Glosario de términos

- FriendsGo: Nombre de la red social desarrollada en el proyecto, que integra videollamadas aleatorias y funciones clásicas de redes sociales como publicaciones, perfiles y chats.
- Videollamadas Aleatorias: Sistema que conecta de forma automática a dos usuarios al azar para conversar por video en tiempo real.
- WebRTC: Tecnología utilizada para realizar videollamadas directamente entre navegadores sin necesidad de plugins, ofreciendo comunicación peer-to-peer.
- Socket.io: Biblioteca para gestionar la comunicación en tiempo real entre cliente y servidor, esencial para chats y notificaciones.
- Remix: Framework de React elegido para el frontend, con enfoque en rendimiento y renderizado optimizado
- Node.js: Entorno de ejecución para JavaScript usado en el backend del proyecto.
- ExpressJS: Framework ligero sobre Node.js que facilita la creación de APIs y servicios backend.
- PostgreSQL: Sistema de gestión de bases de datos relacional elegido por su potencia y soporte de relaciones complejas.
- Feed Personalizado: Sección de publicaciones adaptadas al perfil y actividad del usuario mediante filtrado colaborativo.
- Match: Función que permite conectar usuarios que se han gustado mutuamente durante una videollamada.
- Moderación: Herramientas administradas por el staff para controlar contenido inapropiado y sancionar usuarios
- Sprint: Unidad de tiempo en Scrum (metodología ágil utilizada), en este caso de una semana, durante la cual se desarrollan funcionalidades.
- Hooks: Funciones reutilizables de React usadas en el proyecto para manejar lógica como autenticación, chat o videollamadas.
- Taiga.io: Herramienta seleccionada para la gestión de tareas y seguimiento del proyecto bajo la metodología Scrum.

6 Anexos

El "Plan de Prevención de Riesgos Laborales de FriendsGO", creado por sus fundadores Jaider Cabarcas Chico y Erik Manuel Saldaña Díaz, busca garantizar la salud y seguridad en su modelo de teletrabajo. Se fundamenta en la legislación española relevante.

La empresa, compuesta por los dos fundadores, autogestiona la prevención, con Jaider Cabarcas como delegado de prevención. Se identifican riesgos principales como ergonómicos, visuales, psicosociales y de sedentarismo.

Las medidas clave incluyen:

- **Prevención y Protección:** Adaptación ergonómica del puesto (sillas, escritorios, pausas activas), higiene, bienestar psicosocial y revisiones médicas recomendadas.
- **Planificación:** Calendario con acciones preventivas diarias, mensuales, trimestrales y anuales (pausas, revisiones, formación).
- **Emergencias:** Protocolos para incendios, accidentes y evacuación.
- **Formación:** Capacitación en EPIs, procedimientos seguros, identificación de riesgos y primeros auxilios.
- **Mejora Continua:** Evaluaciones periódicas mediante inspecciones, encuestas y análisis de incidentes.

El objetivo es mantener un entorno de trabajo remoto seguro y saludable, con un compromiso de evaluación y mejora constantes.

https://drive.google.com/file/d/14zGChgX3W_1LSlBNXyR-tCD8OzFZKqqD/view?usp=drive_link

https://github.com/SoyManoolo/M12