



IES
Juan José
Calvo Miguel

Memoria

Live Pulse



Proyecto DAM

Presentado por:
Director/a colectivo/a:
Director/a individual:
Fecha:

Pablo Ovin Fresno
José Luis Arias Cobreros
Julia Paz Triana Toribio
13/12/2024

Agradecimientos

Seguramente me deje a muchas personas en estos agradecimientos, ya que han sido muchas las personas que me han inspirado tanto para adentrarme en el mundo de la programación, como para intentar crear un proyecto de estas características, pero debo destacar a las siguientes:

En primer lugar, a todos los profesores y profesoras del grado superior de desarrollo de aplicaciones multiplataforma del IES Juan José Calvo Miguel de Sotredio, sin los cuales no tendría ni siquiera un punto de partida para llegar a crear esta aplicación.

Mención especial a Julia, que ha supervisado el proceso y siempre me ha dado un feedback positivo, además de ideas y mejoras muy interesantes.

A mis compañeros y compañeras del grado, que han sido soporte y ayuda ya no solo en el proyecto, sino a lo largo de los últimos dos años.

Mención especial a Víctor y Luis, los cuales también aportaron ideas, mejoras, y fueron parte activa de los “test” de mi aplicación.

Por último, tampoco puedo olvidarme de mis amigos externos al grado de DAM, que también fueron parte activa del proceso de “test”, y que me aportaron una visión alejada de la programación, la cual, en muchos aspectos, es más cercana a la del usuario “real” de la aplicación.

Resumen

Este proyecto consiste en una red social centrada en el mundo de la música, diseñada para conectar personas con gustos musicales similares, fomentar interacciones sobre experiencias en conciertos y festivales, y brindar una plataforma donde los usuarios puedan seguir de cerca a sus artistas favoritos y recibir información actualizada sobre próximos eventos.

El objetivo principal de la aplicación es crear una comunidad digital donde los amantes de la música puedan compartir su pasión. Los usuarios pueden encontrar personas con intereses musicales comunes, discutir eventos, intercambiar opiniones y realizar conexiones basadas en sus afinidades. Además, la plataforma ofrece un espacio donde los usuarios pueden consultar y seguir la actividad de sus artistas favoritos y acceder a información sobre eventos futuros que se transmiten en streaming o que están cerca de su ubicación.

La aplicación también permite que los usuarios creen una red de amigos en torno a su interés en la música, comparten experiencias en eventos y comenten publicaciones de otros.

Para el desarrollo de la interfaz se utilizó Angular, que facilita una experiencia de usuario dinámica e interactiva, ideal para una red social moderna. Spring Boot fue la tecnología empleada en el backend de la aplicación, proporcionando una arquitectura robusta y escalable para gestionar la lógica de usuario y otros servicios relacionados. Además, la base de datos Firebase se integra como almacenamiento de datos en la nube, garantizando un rendimiento rápido y seguro y simplificando el manejo de la autenticación.

Para enriquecer la experiencia de usuario, se utilizaron las siguientes APIs:

- **JamBase API:** para consultar información sobre artistas y eventos, manteniendo el contenido de la plataforma actualizado con las últimas novedades musicales.
- **Leaflet:** para la integración de mapas interactivos, lo cual facilita la localización de eventos y conciertos cercanos al usuario.
- **Google Search API:** para mejorar la búsqueda y descubrimiento de artistas y eventos de forma rápida y precisa.

Principales Funcionalidades:

- Registro y autenticación de usuarios
- Sistema de amigos y solicitudes de amistad
- Seguimiento de artistas y eventos favoritos

Palabras Clave

Festivales y Conciertos

Conexión entre usuarios

Seguimiento de Eventos

Índice General

CAPÍTULO 1. MEMORIA DEL PROYECTO.....	7
1.1 RESUMEN DE LA MOTIVACIÓN	7
1.2 OBJETIVOS.....	8
CAPÍTULO 2. INTRODUCCIÓN	9
2.1 JUSTIFICACIÓN DEL PROYECTO.....	9
2.2 ESTUDIO DE LA SITUACIÓN ACTUAL.....	10
CAPÍTULO 3. ASPECTOS TEÓRICOS	11
3.1 CONCEPTOS	11
3.2 TECNOLOGÍAS	12
3.3 HERRAMIENTAS	13
CAPÍTULO 4. ANÁLISIS	15
4.1 DEFINICIÓN DEL SISTEMA.....	15
4.1.1 Determinación del alcance del sistema.....	15
4.2 CATÁLOGO DE REQUISITOS	17
4.2.1 Requisitos funcionales.....	17
4.2.2 Requisitos no funcionales.....	19
4.3 IDENTIFICACIÓN DE ACTORES DEL SISTEMA	20
4.4 ESPECIFICACIÓN DE CASOS DE USO	21
4.5 DIAGRAMA DE CLASES PRELIMINAR DEL ANÁLISIS.....	22
4.6 GUÍAS DE ESTILO	23
4.6.1 Paleta de Colores.....	23
4.6.2 Fuentes Utilizadas.....	24
4.6.3 Estructura y Dimensiones.....	24
CAPÍTULO 5. PLAN DE PRUEBAS.....	25
5.1 INTRODUCCIÓN	25
5.2 DISEÑO Y PLANIFICACIÓN DEL PLAN DE PRUEBAS	25
5.2.1 Pruebas unitarias	25
5.2.2 Pruebas de integración	26
5.2.3 Pruebas de integridad de los datos y la base de datos.....	26
5.2.4 Pruebas funcionales.....	26
5.2.5 Pruebas de interfaz de usuario.....	28
5.2.6 Pruebas de seguridad.....	28
5.2.7 Pruebas de usabilidad y accesibilidad.....	29
5.3 ANÁLISIS E INTERPRETACIÓN DE RESULTADOS OBTENIDOS TRAS LA EJECUCIÓN DEL PLAN DE PRUEBAS	
30	
CAPÍTULO 6. DISEÑO DEL SISTEMA	32
6.1 DISEÑO DE CLASES.....	32
6.1.1 Diagrama de clases Backend.....	32
6.1.2 Diagrama de clases Frontend.....	32
6.2 DIAGRAMAS DE INTERACCIÓN Y ESTADOS	33

6.2.1	<i>Diagramas de interacción (comunicación y secuencia)</i>	33
6.3	DISEÑO DE LA BASE DE DATOS	38
6.3.1	<i>Descripción del SGBD usado</i>	38
6.3.2	<i>Integración del SGBD en nuestro sistema.</i>	39
6.3.3	<i>Diagrama E-R</i>	40
6.4	DISEÑO DE LA INTERFAZ	41
CAPÍTULO 7. IMPLEMENTACIÓN DEL SISTEMA		49
7.1	ESTÁNDARES Y NORMAS SEGUIDOS	49
7.2	LENGUAJES DE PROGRAMACIÓN.....	53
7.3	HERRAMIENTAS Y PROGRAMAS USADOS PARA EL DESARROLLO	56
7.4	CREACIÓN DEL SISTEMA.....	59
7.4.1	<i>Problema 1: Conexión de Firebase con el Frontend (Angular)</i>	59
7.4.2	<i>Problema 2: Almacenamiento de imágenes de perfil en Firebase Storage</i>	59
7.4.3	<i>Problema 3: Barra de búsqueda de eventos</i>	60
7.4.4	<i>Problema 4: Sincronización entre Firebase Authentication y Firestore</i>	60
7.4.5	<i>Problema 5: Manejo de la paginación de eventos y usuarios.....</i>	61
7.4.6	<i>Problema 8: Doble llamada a getUserId durante el login.....</i>	61
7.4.7	<i>Problema 9: Filtro por género de eventos con múltiples géneros musicales</i>	62
7.4.8	<i>Problema 10: Filtro de artistas que requiere al menos un filtro para evitar errores en la API</i>	62
CAPÍTULO 8. MANUALES DEL SISTEMA		63
8.1	MANUAL DE USUARIO	63
CAPÍTULO 9. CONCLUSIONES Y AMPLIACIONES		76
9.1	CONCLUSIONES	76
9.2	AMPLIACIONES	79

Capítulo 1. Memoria del proyecto

1.1 Resumen de la motivación

Este proyecto nace del deseo de conectar a personas con gustos musicales afines en una plataforma social dinámica y especializada. En la era de la música digital y de los eventos masivos como festivales y conciertos, muchas personas buscan formas de interactuar y compartir experiencias, pero las opciones específicas para conectar con personas que comparten sus preferencias musicales son limitadas. Este proyecto pretende llenar este vacío, ofreciendo una plataforma que no solo permite conocer a otros usuarios con intereses similares, sino también seguir de cerca la agenda de conciertos y festivales, y facilitar el seguimiento de artistas.

La estructura del proyecto se basa en una experiencia integral de usuario, donde pueden interactuar con amigos, seguir artistas y consultar eventos desde cualquier dispositivo. Para ello, se ha elegido un enfoque tecnológico que garantiza una aplicación escalable y accesible. Angular se usa en el frontend por su eficiencia y capacidad de crear aplicaciones web interactivas, mientras que Spring Boot en el backend permite manejar la gestión de usuarios de manera segura y robusta. Firebase se selecciona como base de datos por su flexibilidad y facilidad tanto a la hora de autenticar usuarios como a la de guardar datos y archivos en su base de datos online.

Este proyecto aporta, por lo tanto, una plataforma especializada en la conexión social a través de la música. Al enfocarse en el perfil de cada usuario y sus intereses, se crea un espacio para la interacción significativa y el descubrimiento de eventos y artistas relevantes. Además, al integrar APIs como JamBase para eventos y Leaflet para mapas interactivos, el usuario obtiene una experiencia completa y personalizada de su actividad musical.

1.2 Objetivos

Los objetivos principales de este proyecto son los siguientes:

1. **Conectar a personas con gustos musicales similares** mediante un sistema de perfiles y afinidades.
2. **Ofrecer un espacio para la gestión y visualización de eventos musicales** (conciertos y festivales), tanto locales como globales.
3. **Facilitar el seguimiento de artistas favoritos** y recibir notificaciones de sus novedades y conciertos cercanos.
4. **Permitir una interacción dinámica entre usuarios** mediante un sistema de amigos, solicitudes y comentarios en eventos.
5. **Garantizar la seguridad y accesibilidad de los datos de usuario** usando Firebase y Spring Boot para el backend.

Capítulo 2. Introducción

2.1 Justificación del proyecto

Este proyecto surge de la creciente necesidad de contar con una plataforma especializada que conecte a personas que comparten afinidades musicales. Aunque existen redes sociales genéricas y sitios web dedicados a la música, como Wegow, estas plataformas se centran principalmente en la promoción de eventos y en la venta de entradas, pero ofrecen limitadas opciones de interacción social y personalización basada en los gustos del usuario. Este proyecto va más allá de la simple búsqueda de conciertos o eventos musicales: crea un espacio donde los usuarios pueden conectar, interactuar y mantenerse al día sobre eventos, artistas y novedades según sus intereses específicos.

La aplicación se enfoca en permitir a los usuarios construir un perfil detallado, agregar amigos, y seguir artistas y eventos que les interesan, creando así una experiencia única y personalizada.

Además, este proyecto cubre la necesidad de contar con una herramienta que facilite la localización y planificación de eventos mediante mapas interactivos, usando tecnologías como Leaflet, y permita explorar conciertos de forma dinámica y visual. De esta forma, se intenta mejorar la experiencia del usuario en el proceso de planificar su asistencia a eventos y conocer nuevas personas.

Comparada con opciones como Wegow, esta plataforma no solo facilita la consulta de conciertos, sino que se convierte en una red social en torno a la música. Así, brinda a los usuarios una forma de conexión que se extiende más allá de la simple asistencia a eventos, con el propósito de crear una comunidad musical interactiva y especializada.

2.2 Estudio de la situación actual

Para entender el impacto y potencial de nuestra plataforma, es necesario realizar un análisis de sistemas existentes que, aunque no replican nuestro proyecto por completo, sí presentan funcionalidades similares o apuntan a un público similar. A continuación, se analizan algunas de las plataformas más relevantes en este ámbito.

Identificación	Objetivo general	Resultados
Wegow	Facilitar la búsqueda y adquisición de entradas para eventos musicales.	Ofrece información detallada sobre eventos y permite la compra de entradas. Sin embargo, no incluye un aspecto social significativo.
Last.fm	Crear una comunidad de usuarios basada en el seguimiento de sus preferencias musicales.	Recomendaciones de música y artistas basadas en el historial de escucha del usuario. Incluye algunas funcionalidades sociales, pero no se centra en eventos.
Songkick	Informar a los usuarios sobre los conciertos y eventos de sus artistas favoritos en su ubicación.	Sistema de alertas y recomendaciones de eventos locales; no ofrece funciones de interacción social significativas.
Spotify	Plataforma de streaming con recomendaciones personalizadas y listas de reproducción.	Ofrece recomendaciones de música en función de los gustos, pero no facilita la organización de eventos o interacción social fuera del entorno musical.
Eventbrite	Proporcionar una plataforma para la creación y búsqueda de eventos de todo tipo.	Amplia base de datos de eventos y sistema de venta de entradas, pero carece de especialización en eventos musicales y de enfoque en los gustos del usuario.

Cada uno de estos sistemas ofrece funcionalidades específicas que pueden atraer a los amantes de la música, pero presentan carencias significativas en cuanto a la creación de una verdadera red social que permita a los usuarios interactuar y conectar en torno a sus afinidades musicales. Por ejemplo, Wegow y Songkick ofrecen una buena organización de eventos, pero carecen de funcionalidades sociales avanzadas, como la interacción entre usuarios o la formación de comunidades. Last.fm permite una personalización del contenido musical, pero no se centra en los eventos ni en la gestión de relaciones entre usuarios.

Nuestro proyecto busca llenar ese vacío combinando la información sobre eventos y artistas con una verdadera red social donde los usuarios puedan interactuar, comentar experiencias en eventos, hacer amigos y seguir a sus artistas favoritos.

Capítulo 3. Aspectos teóricos

Este capítulo presenta una descripción de los conceptos, tecnologías y herramientas empleados en el desarrollo del proyecto. Esta sección sirve para contextualizar las elecciones tecnológicas y metodológicas, ofreciendo una visión de por qué cada elemento es relevante para la aplicación.

3.1 Conceptos

Redes Sociales y Comunidades Virtuales

Las redes sociales han transformado la forma en que las personas interactúan en línea, ofreciendo plataformas donde los usuarios pueden compartir información, conectar según intereses comunes y colaborar. La importancia de estas redes radica en su capacidad para crear conexiones y permitir la interacción en torno a temas específicos, en nuestro caso, la música. El concepto de "comunidad virtual" es clave para el proyecto, ya que los usuarios pueden interactuar no solo con el contenido musical, sino entre sí, comentando eventos y compartiendo experiencias musicales. Este modelo promueve la interacción y la creación de redes de usuarios basadas en afinidades musicales.

Geolocalización en Aplicaciones Web

La geolocalización es la capacidad de identificar y mostrar la ubicación geográfica de un dispositivo o usuario. En el contexto de la aplicación, la geolocalización permite mostrar eventos cercanos a los usuarios y personalizar las recomendaciones. Este concepto es importante para la funcionalidad de la aplicación, ya que permite ofrecer una experiencia personalizada y basada en la proximidad a los eventos.

3.2 Tecnologías

Angular

Angular es un framework de desarrollo de aplicaciones web de una sola página (SPA) creado por Google. Su estructura modular y su capacidad para gestionar el estado de las aplicaciones en el cliente lo convierten en una herramienta ideal para el desarrollo de interfaces de usuario interactivas y dinámicas. Angular utiliza TypeScript, un lenguaje orientado a objetos que permite una programación más robusta y mantiene la coherencia en la estructura de datos del proyecto. En nuestro proyecto, Angular facilita la creación de un frontend interactivo donde los usuarios pueden navegar, comentar y conectarse con otros usuarios, y seguir sus eventos y artistas favoritos.

Spring Boot

Spring Boot es un framework basado en Java que permite el desarrollo de aplicaciones backend. Su enfoque modular y su estructura escalable han hecho de él una elección popular para la construcción de APIs RESTful, que son la base de la arquitectura de este proyecto. En nuestra aplicación, Spring Boot facilita la comunicación entre el frontend y la base de datos, gestionando las solicitudes relacionadas con el registro de usuarios, la administración de amigos y eventos, y el sistema de notificaciones.

Firebase

Firebase, creado por Google, es una plataforma en la nube que ofrece múltiples servicios para aplicaciones móviles y web, como almacenamiento de bases de datos en tiempo real y autenticación de usuarios. Firebase es ideal para gestionar la base de datos de usuarios, ya que permite el almacenamiento y recuperación de datos de forma eficiente y segura, lo cual es crucial en una red social donde los datos personales deben estar bien protegidos. Firebase ha sido la opción elegida para el proyecto debido a su escalabilidad y facilidad de integración con otros servicios de Google.

JamBase

La API de JamBase permite acceder a información detallada sobre conciertos, festivales y eventos musicales. Esta API es ideal para mantener a los usuarios informados sobre los próximos eventos de sus artistas favoritos. Además de la información básica de cada evento, como su ubicación y fecha, JamBase permite obtener información sobre los artistas y los lugares, lo cual añade valor a la experiencia del usuario.

Leaflet

Leaflet es una biblioteca JavaScript de código abierto para la visualización de mapas interactivos. En el contexto de esta aplicación, Leaflet se emplea para mostrar la localización de los eventos musicales en un mapa, permitiendo al usuario visualizar de forma rápida y gráfica dónde se encuentran los eventos cercanos. Leaflet permite además integrar funciones como marcadores y capas, enriqueciendo la experiencia de los usuarios que buscan eventos basados en su ubicación geográfica.

Google Search

a API de Google Search permite realizar consultas y obtener resultados de búsqueda en tiempo real. En nuestro proyecto, esta API se usa para obtener información adicional sobre eventos o artistas que el usuario puede querer conocer en mayor profundidad. Esta integración enriquece la experiencia de los usuarios, proporcionando enlaces a información relevante y actualizada sin necesidad de salir de la aplicación.

3.3 Herramientas

Visual Studio Code

Visual Studio Code (VS Code) es un editor de código fuente desarrollado por Microsoft, conocido por su gran variedad de extensiones y soporte de múltiples lenguajes de programación. Este editor es utilizado para el desarrollo tanto del frontend (Angular) como del backend (Spring Boot) del proyecto. Su flexibilidad y herramientas de depuración facilitan un flujo de trabajo ágil, además de permitir la integración de Git para el control de versiones.

Postman

Postman es una herramienta que facilita el diseño, prueba y automatización de APIs REST. En este proyecto, Postman se ha utilizado para realizar pruebas exhaustivas de la API desarrollada en Spring Boot, permitiendo verificar el funcionamiento de las diferentes funciones del backend y asegurando la comunicación adecuada con el frontend.

GitHub

GitHub es una plataforma para el alojamiento de código y el control de versiones mediante Git. En este proyecto, GitHub se ha utilizado para gestionar el código fuente, llevar un registro de los cambios realizados en cada versión y colaborar en equipo. Además, GitHub permite la integración con herramientas de despliegue continuo, facilitando el mantenimiento del proyecto a lo largo del tiempo.

IntelliJ IDEA

IntelliJ IDEA es un entorno de desarrollo integrado (IDE) creado por JetBrains, conocido por su capacidad para trabajar con proyectos Java y otros lenguajes de programación.

En el desarrollo del backend de Live Pulse, se ha utilizado IntelliJ IDEA para aprovechar sus características avanzadas, como la finalización de código inteligente, la depuración y las pruebas unitarias. Este IDE ha facilitado un flujo de trabajo eficiente y ha contribuido a la calidad del código.

Este conjunto de conceptos, tecnologías y herramientas ha sido elegido y adaptado para asegurar que el desarrollo del proyecto se alinee con los objetivos planteados, proporcionando una plataforma robusta, escalable y orientada al usuario.

Capítulo 4. Análisis

4.1 Definición del sistema

4.1.1 Determinación del alcance del sistema

Live Pulse es una plataforma de red social centrada en la música que permite a los usuarios conectarse con otras personas que comparten sus gustos musicales, comentar experiencias en eventos (festivales y conciertos), hacer amigos y consultar información sobre sus artistas favoritos. Este apartado delimita el alcance del sistema, especificando claramente lo que se va a incluir en el desarrollo y lo que quedará fuera del mismo.

Alcance del Desarrollo:

1. Registro y Gestión de Usuarios:

- Los usuarios podrán registrarse y crear un perfil personal donde puedan compartir información sobre sus preferencias musicales y experiencias en conciertos.
- La funcionalidad de inicio de sesión y recuperación de contraseña estará implementada para facilitar el acceso a los usuarios.

2. Conexiones entre Usuarios:

- Live Pulse permitirá a los usuarios buscar y agregar a otros usuarios como amigos, fomentando la creación de una comunidad musical.
- Los usuarios podrán comentar y compartir sus experiencias sobre eventos a los que hayan asistido.

3. Consulta de Eventos Musicales:

- La aplicación integrará una API (JamBase) para mostrar información sobre conciertos, festivales y eventos musicales próximos.
- Se ofrecerá una sección donde los usuarios puedan consultar información detallada sobre artistas y eventos, incluyendo fechas, ubicaciones y tipos de música.

4. Interacción con Mapa:

- Se implementará un mapa interactivo utilizando Leaflet, que permitirá a los usuarios localizar eventos musicales en función de su geolocalización.
- Los usuarios podrán ver la ubicación de los eventos en un formato visual y obtener direcciones para asistir.

Límites del Desarrollo:

1. Notificaciones y Recordatorios:

- El sistema de notificaciones sobre actualizaciones en los eventos y artistas seguidos aún no se implementará debido al límite de peticiones diarias que ofrece JamBase de forma gratuita, ya que esto requeriría un WebSocket enviando peticiones de forma continua.

2. Funcionalidad de Streaming:

- Aunque se prevé la posibilidad de incluir información sobre conciertos en streaming, en esta fase del desarrollo no se implementará la funcionalidad para asistir a eventos virtuales directamente a través de la plataforma.

3. Integración con Redes Sociales:

- No se contempla la integración con otras redes sociales para compartir información sobre la plataforma o eventos, lo que limita la difusión de contenido a la aplicación misma.

4. Gestión de Contenido Generado por Usuarios:

- Aunque los usuarios podrán comentar experiencias y compartir opiniones, no se desarrollará una funcionalidad robusta para moderar este contenido ni se implementarán sistemas de puntuación o evaluación.

5. Comercio Electrónico:

- Live Pulse no incluirá en esta fase la venta de entradas o merchandising relacionado con eventos. El enfoque estará centrado en la conexión entre usuarios y la información sobre eventos.

6. Soporte Multilingüe:

- En esta fase inicial, la aplicación solo se desarrollará en español, dejando de lado la posibilidad de ofrecer soporte en otros idiomas.

4.2 Catálogo de requisitos

4.2.1 Requisitos funcionales

RF 1	Registro y Autenticación de Usuarios	Prioridad
RF 1.1	El sistema debe permitir a los usuarios registrarse mediante un formulario	Alta
RF 1.1.1	El formulario de registro debe incluir al menos los campos: Nombre, Correo electrónico y Contraseña	Alta
RF 1.1.2	El sistema debe validar que el correo electrónico no esté ya registrado	Alta
RF 1.1.3	El sistema debe enviar un correo de confirmación tras el registro	Media
RF 1.2	El sistema debe permitir a los usuarios iniciar sesión	Alta
RF 1.2.1	Los usuarios deben poder iniciar sesión utilizando su correo electrónico y contraseña	Alta
RF 1.2.2	El sistema debe proporcionar una opción de recuperación de contraseña	Media

RF 2	Gestión de Perfil de Usuario	Prioridad
RF 2.1	El sistema puede permitir a los usuarios editar su perfil	Alta
RF 2.1.1	Los usuarios deben poder cambiar su nombre, foto de perfil y preferencias musicales	Alta
RF 2.1.2	Los usuarios deben poder eliminar su cuenta	Media

RF 3	Interacción Social	Prioridad
RF 3.1	El sistema debe permitir a los usuarios agregar amigos	Alta
RF 3.1.1	Los usuarios deben poder buscar amigos a través de los distintos eventos	Alta
RF 3.1.2	El sistema debe notificar a los usuarios cuando un amigo es agregado	Media
RF 3.2	El sistema debe permitir a los usuarios comentar sobre eventos	Alta
RF 3.2.1	Los usuarios deben poder crear publicaciones sobre conciertos y festivales	Alta
RF 3.2.2	Los usuarios deben poder reaccionar a publicaciones de otros	Media

RF 4	Consulta de Eventos y Conciertos	Prioridad
RF 4.1	El sistema debe permitir a los usuarios buscar conciertos, festivales y artistas	Alta
RF 4.1.1	Los usuarios deben poder filtrar eventos por género musical	Alta
RF 4.1.2	El sistema debe mostrar detalles de cada evento incluyendo ubicación, fecha y artistas entre otros	Alta
RF 4.2	El sistema debe permitir a los usuarios consultar conciertos en stream	Alta
RF 4.2.1	El sistema debe permitir participar en los chats de los eventos solo si el usuario está logeado	Alta

RF 5	Integración con APIs Externas	Prioridad
RF 5.1	El sistema debe integrar la API de JamBase para obtener información sobre eventos	Alta
RF 5.2	El sistema debe integrar la API Google Search para realizar búsquedas de noticias	Media
RF 5.3	El sistema debe integrar Leaflet para mostrar ubicaciones de eventos en un mapa	Media

Estos requisitos proporcionan una base sólida para el desarrollo de Live Pulse, permitiendo un enfoque claro y ordenado en la construcción del sistema.

4.2.2 Requisitos no funcionales

Los requisitos no funcionales a continuación describen aspectos de calidad del sistema Live Pulse, asegurando que la aplicación no solo cumpla con su funcionalidad, sino que también ofrezca una experiencia de usuario adecuada y satisfactoria.

RNF 1	Usabilidad	La aplicación debe ser fácil de usar y navegar
RNF 1.1	Interfaz Intuitiva	La interfaz de usuario debe ser clara e intuitiva, permitiendo a los usuarios interactuar sin confusión
RNF 1.2	Consistencia en el Diseño	Todos los elementos de la interfaz deben mantener un diseño consistente en términos de colores, tipografía y disposición para mejorar la experiencia del usuario
RNF 2	Rendimiento	El sistema debe garantizar un rendimiento óptimo
RNF 2.1	Tiempo de Carga	La aplicación debe cargar en menos de 3 segundos en conexiones estándar para garantizar una experiencia de usuario fluida
RNF 2.2	Manejo de Cargas Concurrentes	El sistema debe ser capaz de manejar al menos 100 usuarios simultáneos sin degradar el rendimiento
RNF 3	Seguridad	Se deben implementar medidas adecuadas para proteger la información de los usuarios y su privacidad
RNF 3.1	Autenticación de Usuarios	El sistema debe implementar mecanismos de autenticación seguros como OAuth 2.0, para proteger los datos de los usuarios
RNF 4	Escalabilidad	La arquitectura del sistema debe ser diseñada para crecer con el tiempo y soportar un aumento de usuarios
RNF 4.1	Capacidad de Escalamiento	La arquitectura del sistema debe permitir la escalabilidad horizontal, facilitando la adición de más servidores para manejar un aumento en el tráfico de usuarios.

RNF 5	Portabilidad	La aplicación debe ser accesible en diferentes plataformas y dispositivos
RNF 5.1	Compatibilidad de Plataforma	La aplicación debe ser accesible desde navegadores web modernos y dispositivos móviles, asegurando una experiencia consistente en diversas plataformas

Estos requisitos no funcionales son fundamentales para asegurar que Live Pulse no solo funcione correctamente, sino que también ofrezca una experiencia segura, rápida y agradable para los usuarios.

4.3 Identificación de actores del sistema

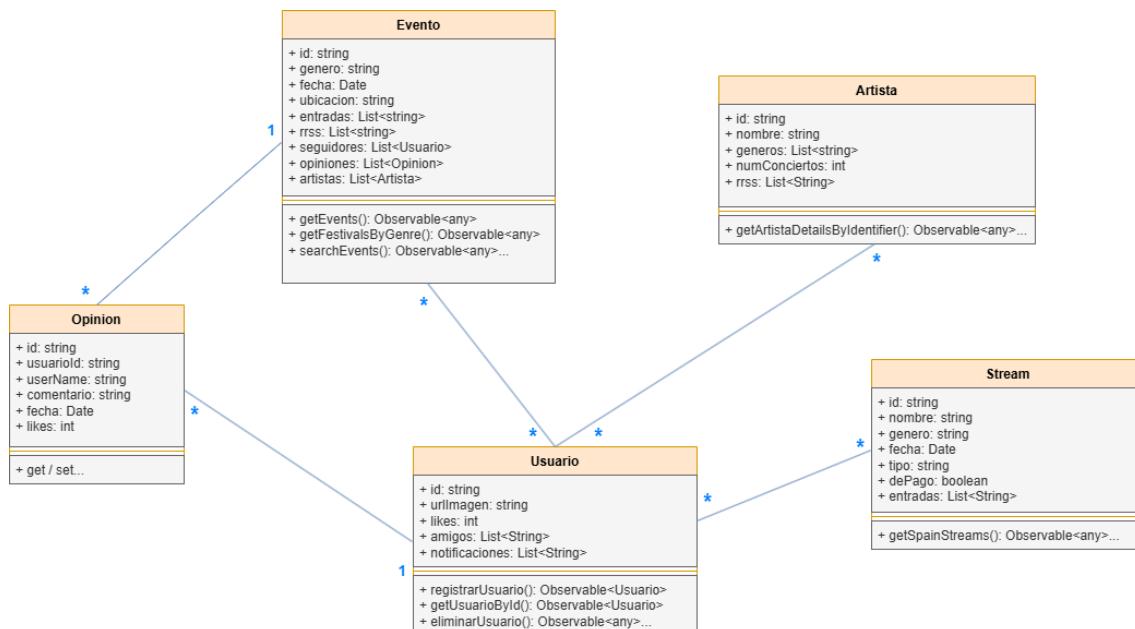
Actor	Descripción
Usuario	Cualquier persona que se registre en la plataforma para descubrir música, conectar con otros usuarios, comentar eventos y recibir actualizaciones sobre conciertos
Sistema de API	Servicios externos que se integran en la aplicación, como JamBase para eventos de música y Google para la búsqueda de noticias
Desarrollador	Programador que mantiene y mejora la aplicación, implementando nuevas funciones y corrigiendo errores.

4.4 Especificación de casos de uso

A continuación, se enumeran los casos de uso del sistema Live Pulse, que describen las interacciones clave que los actores tienen con la aplicación.

Caso de Uso	Descripción
Registro de Usuario	Permite a los nuevos usuarios crear una cuenta en la plataforma proporcionando sus datos básicos
Inicio de Sesión	Los usuarios existentes pueden acceder a sus cuentas utilizando sus credenciales
Perfil de Usuario	Los usuarios pueden visualizar y editar su perfil incluyendo preferencias musicales y amigos
Conexión con Otros Usuarios	Los usuarios pueden buscar y agregar amigos según sus gustos musicales
Publicación de Comentarios	Los usuarios pueden comentar sobre eventos a los que han asistido, compartiendo sus experiencias
Consulta de Artistas	Permite a los usuarios buscar información sobre sus artistas favoritos y seguir su música
Búsqueda de Eventos	Los usuarios pueden buscar conciertos y festivales programados, con información detallada
Chat entre Usuarios	Los usuarios logeados pueden comunicarse entre sí a través de un sistema de chat en tiempo real

4.5 Diagrama de clases preliminar del análisis



4.6 Guías de estilo

4.6.1 Paleta de Colores

La paleta de colores de la aplicación está diseñada para un tema oscuro que proporciona un contraste adecuado y una experiencia visual atractiva. Los colores utilizados son:

- **Color Principal:** #3fa83f (Verde brillante) – Este color se utiliza en elementos clave como botones y enlaces importantes.

```
--primary-color: #3fa83f; /* Color principal usado en la aplicación */
```

- **Color Secundario:** #333333 (Gris oscuro) – Se utiliza como fondo para ciertos componentes y para proporcionar un contraste visual.

```
--secondary-color: #333333; /* Color secundario usado en la aplicación */
```

- **Color de Fondo:** #222222 (Negro suave) – Este color se aplica al fondo general del cuerpo de la página, creando una atmósfera oscura.

```
--background-color: #222222; /* Color de fondo oscuro */
```

- **Color del Texto:** #ffffff (Blanco) – Usado para garantizar que el texto sea legible sobre el fondo oscuro.

```
--text-color: #ffffff; /* Color del texto */
```

- **Color de Bordes:** #444444 (Gris medio) – Utilizado en bordes de botones y otros elementos para definir su contorno.

```
--border-color: #444444; /* Color de los bordes */
```

4.6.2 Fuentes Utilizadas

Las fuentes seleccionadas contribuyen a la legibilidad y estética del diseño:

- **Fuente Principal: 'Signika'**, sans-serif – Utilizada en el cuerpo del texto, brinda una apariencia moderna y amigable.
- **Fuente para Encabezados: 'Bebas Neue'**, sans-serif – Esta fuente se aplica a los encabezados (h1, h2, h3, etc.) para dar un carácter fuerte y llamativo a los títulos.

La elección de estas fuentes complementa la temática oscura de la aplicación, asegurando que el contenido sea accesible y atractivo.

4.6.3 Estructura y Dimensiones

La estructura y dimensiones de los elementos se definen para mantener una apariencia ordenada y coherente:

- **Dimensiones Fijas:**
 - El contenedor de filtros tiene un ancho fijo de 250px y una altura de 775px, proporcionando un espacio definido para la interacción del usuario.
- **Espaciado:**
 - Se utilizan márgenes y rellenos consistentes en los botones, títulos y contenedores para mejorar la jerarquía visual y la facilidad de uso. Por ejemplo, los botones tienen un espaciado interno de 8px 16px, lo que proporciona un área de clic adecuada.
- **Bordes y Sombra:**
 - Muchos elementos incluyen bordes redondeados (border-radius: 5px) y sombras sutiles (box-shadow), que añaden profundidad y separación visual entre los elementos.
- **Interactividad:**
 - Los botones y enlaces incluyen efectos de hover que cambian el color de fondo y aplican sombras, lo que mejora la experiencia del usuario al interactuar con la interfaz.

Capítulo 5. Plan de pruebas

5.1 Introducción

Este plan de pruebas se centra en la evaluación de funcionalidad, seguridad y usabilidad de la aplicación mediante un seguimiento detallado de las pruebas realizadas, con un enfoque en la interpretación de resultados para optimizar el funcionamiento y la experiencia del usuario. Los tipos de pruebas realizadas incluyen pruebas unitarias, de integración, de integridad de datos y base de datos, funcionales, de interfaz de usuario, de seguridad, y de usabilidad y accesibilidad.

5.2 Diseño y planificación del plan de pruebas

Con el fin de detectar errores en el desarrollo de la aplicación se ha diseñará un plan de pruebas dividiendo estas en los distintos tipos de pruebas como son:

- Pruebas unitarias
- Pruebas de integración
- Pruebas de integridad de los datos y la base de datos
- Pruebas funcionales
- Pruebas de interfaz de usuario
- Pruebas de seguridad
- Pruebas de usabilidad y accesibilidad

5.2.1 Pruebas unitarias

Objetivo: Validar el funcionamiento individual de los métodos y componentes principales de la aplicación, tanto en el frontend como en el backend, para garantizar que operan correctamente de forma aislada.

Técnicas:

Mocking: En las pruebas del backend se utilizan objetos simulados (@Mock) para evitar dependencias externas y probar el servicio de forma aislada.

Verificación de respuestas esperadas: Se valida el resultado de métodos asegurando que devuelvan la salida correcta en función de entradas controladas.

Validación de rutas y peticiones HTTP: En las pruebas del frontend se utiliza “HttpTestingController” para interceptar peticiones HTTP y validar el tipo de petición y la respuesta simulada.

5.2.2 Pruebas de integración

Objetivo: Evaluar la interacción entre distintos módulos y servicios de la aplicación, asegurando que se integren correctamente para el funcionamiento esperado.

Técnicas:

Integración de servicios y controladores: Pruebas que validan la integración del frontend con el API del backend.

Simulación de dependencias externas: Se prueban interacciones con bases de datos mediante mocks en el backend.

Validación de la comunicación entre servicios: Se observa que los datos compartidos entre módulos son los esperados, como al verificar el correcto retorno de seguidores y eventos.

5.2.3 Pruebas de integridad de los datos y la base de datos

Objetivo: Comprobar que los datos almacenados en la base de datos cumplen con los requisitos de integridad y consistencia.

Técnicas:

Simulación de operaciones CRUD: Verificar que las operaciones sobre la base de datos reflejan la integridad y el estado esperado en Firestore.

Verificación de actualizaciones de estado: Probar que los cambios sobre campos de la base de datos se actualicen correctamente en ella.

Control de integridad de referencias y relaciones: Se comprueba que al eliminar elementos las relaciones entre datos se actualizan correctamente.

5.2.4 Pruebas funcionales

Registro de un nuevo usuario

Objetivo: Verificar que el sistema permita registrar un nuevo usuario correctamente y que retorne los datos del usuario registrado.

Técnicas:

Pruebas de caja negra y pruebas de integración.

Actualización de usuario

Objetivo: Asegurar que el sistema actualiza correctamente la información de un usuario existente y devuelve los datos actualizados.

Técnicas:

Pruebas de caja negra y pruebas de integración.

Seguimiento de un evento

Objetivo: Asegurar que un usuario puede seguir un evento específico y que el sistema responde con una confirmación adecuada.

Técnicas:

Pruebas de caja negra y pruebas de integración.

Crear una nueva opinión

Objetivo: Asegurarse de que el sistema permite la creación de una nueva opinión y la almacena correctamente.

Técnicas:

Pruebas de caja negra, pruebas de aceptación y pruebas de integración.

Eliminar un like de una opinión

Objetivo: Verificar que el sistema elimina correctamente un "like" de una opinión.

Técnicas:

Pruebas de caja negra y pruebas de integración.

Obtener mensajes

Objetivo: Comprobar que el sistema recupera correctamente los mensajes de un chat específico desde Firestore.

Técnicas:

Pruebas de caja negra y pruebas de integración.

5.2.5 Pruebas de interfaz de usuario

Objetivo: Verificar que la interfaz de usuario de la aplicación sea intuitiva, funcional y cumpla con las especificaciones de diseño, garantizando que los usuarios puedan interactuar con el sistema de manera efectiva.

Técnicas:

Pruebas manuales: Navegar por el menú principal y todas las subopciones para asegurar que cada enlace funcione correctamente y redirija a la página correspondiente. También se puede evaluar la fluidez del desplazamiento y la claridad de los textos y botones.

Pruebas de interacción: Completar un formulario y verificar que se muestre un mensaje de éxito o error adecuado según la información ingresada. También comprobar que los campos obligatorios están correctamente marcados.

Pruebas de compatibilidad: Probar la interfaz en diferentes navegadores (Chrome, Firefox, Opera) para verificar que todos los elementos de la interfaz se visualicen correctamente y que la funcionalidad se mantenga en todos los casos.

Pruebas de rendimiento: Medir el tiempo de carga de la página principal y de otras secciones importantes de la aplicación. Asegurarse de que todas las páginas se carguen en menos de 3 segundos en condiciones normales de conexión a internet.

5.2.6 Pruebas de seguridad

Objetivo: Asegurar que la aplicación esté protegida contra vulnerabilidades comunes y que los datos de los usuarios se mantengan seguros. Esto incluye evaluar la integridad del sistema y la protección contra accesos no autorizados.

Técnicas:

Pruebas de autenticación: Verificar que el sistema implemente medidas de seguridad adecuadas, como el uso de contraseñas seguras. Probar la funcionalidad de "olvidé mi contraseña" para asegurarse de que los enlaces de restablecimiento de contraseña no sean accesibles sin la debida autorización.

Pruebas de cifrado: Verificar que la información sensible se almacene y transmita de forma segura, utilizando protocolos de cifrado adecuados y asegurándose de que las contraseñas estén debidamente cofradas en la base de datos.

5.2.7 Pruebas de usabilidad y accesibilidad

Objetivo: Evaluar la facilidad de uso de la aplicación y garantizar que sea accesible para todos los usuarios, incluidas aquellas personas con discapacidades. Esto incluye verificar que la interfaz sea intuitiva y que cumpla con las pautas de accesibilidad.

Técnicas:

Pruebas de navegación: Pedir a usuarios reales que realicen tareas comunes en la aplicación y observar si pueden completar estas tareas sin dificultades.

Evaluaciones heurísticas: Aplicar principios de usabilidad para evaluar la interfaz. Realizar una revisión de la interfaz con personas ajena al proyecto para identificar problemas de usabilidad.

Pruebas de tiempo de tarea: Medir el tiempo que tarda un usuario en completar tareas específicas para identificar áreas de mejora en la interfaz.

Pruebas de legibilidad: Evaluar el texto en el interfaz para asegurarse de que sea fácil de leer y comprender. Comprobar el tamaño de la fuente, el contraste con el fondo y la claridad del lenguaje utilizado.

5.3 Análisis e interpretación de resultados obtenidos tras la ejecución del plan de pruebas

Tras la ejecución del plan de pruebas diseñado para la aplicación, se ha llevado a cabo un análisis exhaustivo de los resultados obtenidos en cada una de las categorías de pruebas. A continuación, se presenta un resumen de los hallazgos más relevantes:

Pruebas unitarias

- **Resultados:** La mayoría de las funciones y componentes clave de la aplicación superaron las pruebas unitarias. Sin embargo, se detectaron algunos métodos que devolvían respuestas inesperadas bajo ciertas condiciones.
- **Interpretación:** Estos resultados indican que, aunque el núcleo de la aplicación está bien implementado, es necesario realizar ajustes en la lógica de algunos métodos para asegurar un rendimiento óptimo.

Pruebas de integración

- **Resultados:** La integración entre el frontend y el backend funcionó adecuadamente en la mayoría de los casos.
- **Interpretación:** Es crucial optimizar las configuraciones de las API externas y revisar las interacciones entre módulos para garantizar una comunicación fluida y correcta.

Pruebas de integridad de los datos y la base de datos

- **Resultados:** Las operaciones CRUD funcionaron correctamente y la integridad de los datos se mantuvo.

Pruebas funcionales

- **Resultados:** La aplicación pasó satisfactoriamente la mayoría de las pruebas funcionales. Algunas como la eliminación de usuarios y opiniones tuvieron algún fallo inicial que se pudo corregir.
- **Interpretación:** Fue necesario investigar más a fondo las pruebas de eliminación, ya que podrían afectar la experiencia del usuario y la gestión de datos en la aplicación.

Pruebas de interfaz de usuario

- **Resultados:** La interfaz mostró ser intuitiva en su mayor parte, aunque se recogieron comentarios de los usuarios (amigos) que probaron la aplicación sobre la necesidad de mejorar algunos elementos visuales y la organización del menú.
- **Interpretación:** La usabilidad es un aspecto crítico que debe abordarse antes de lanzar la aplicación. Las sugerencias de los usuarios deben ser consideradas para mejorar la experiencia general.

Pruebas de seguridad

- **Resultados:** Las pruebas de autenticación y cifrado revelaron que la aplicación cumple con los estándares de seguridad. Sin embargo, se identificaron algunas vulnerabilidades menores en el manejo de sesiones.
- **Interpretación:** Se requiere un enfoque adicional para fortalecer la seguridad en el manejo de sesiones de usuario, asegurando así la protección de los datos personales.

Pruebas de usabilidad y accesibilidad

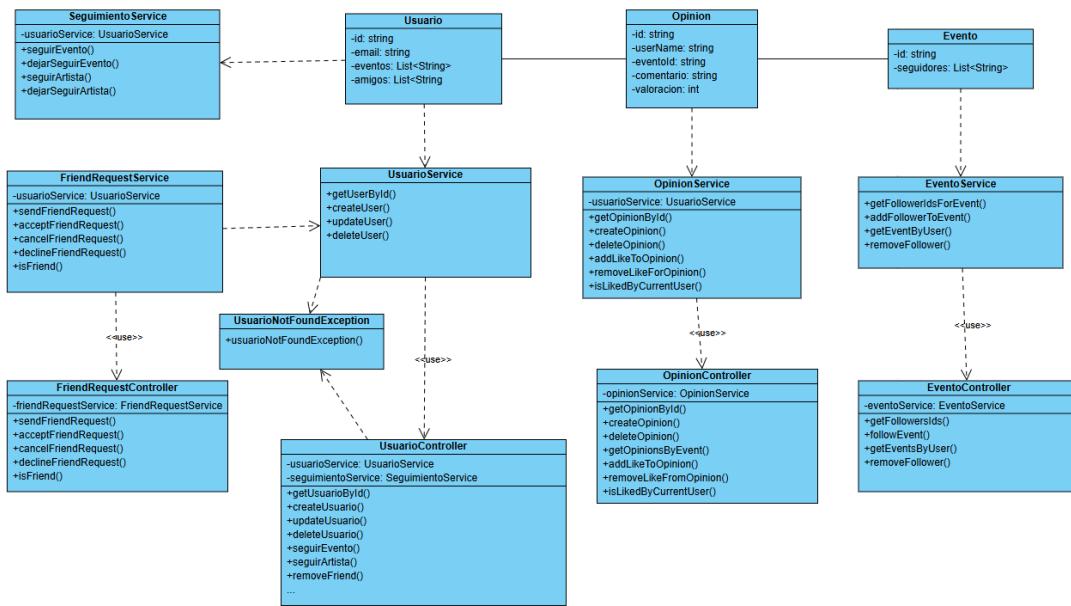
- **Resultados:** Las pruebas revelaron que la aplicación es generalmente accesible y fácil de usar.

El análisis de los resultados de las pruebas realizadas ha proporcionado información valiosa sobre el estado actual de la aplicación. Si bien se observan áreas de éxito en diversas categorías, también hay oportunidades de mejora. La implementación de las recomendaciones resultantes de este análisis será esencial para optimizar el rendimiento, la seguridad y la experiencia del usuario en la aplicación. La iteración y la corrección continua a partir de los hallazgos serán cruciales para el éxito a largo plazo de Live Pulse.

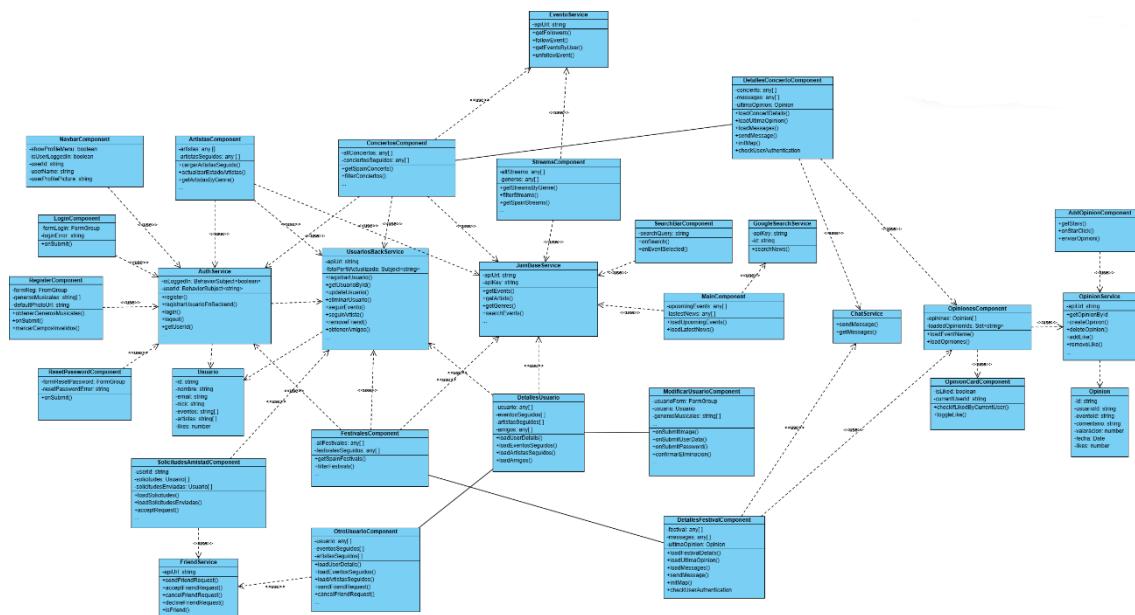
Capítulo 6. Diseño del sistema

6.1 Diseño de clases

6.1.1 Diagrama de clases Backend



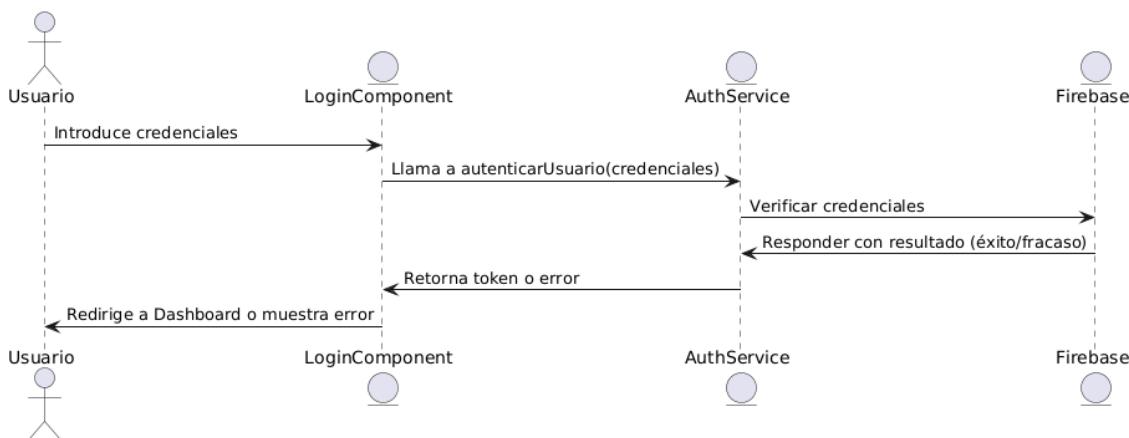
6.1.2 Diagrama de clases Frontend



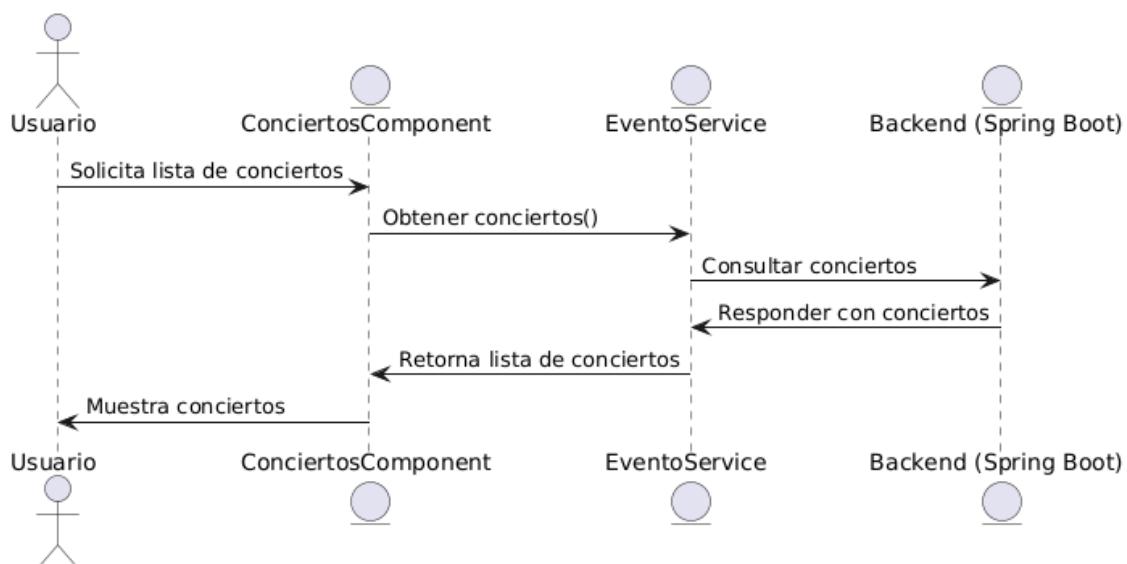
6.2 Diagramas de interacción y estados

6.2.1 Diagramas de interacción (comunicación y secuencia)

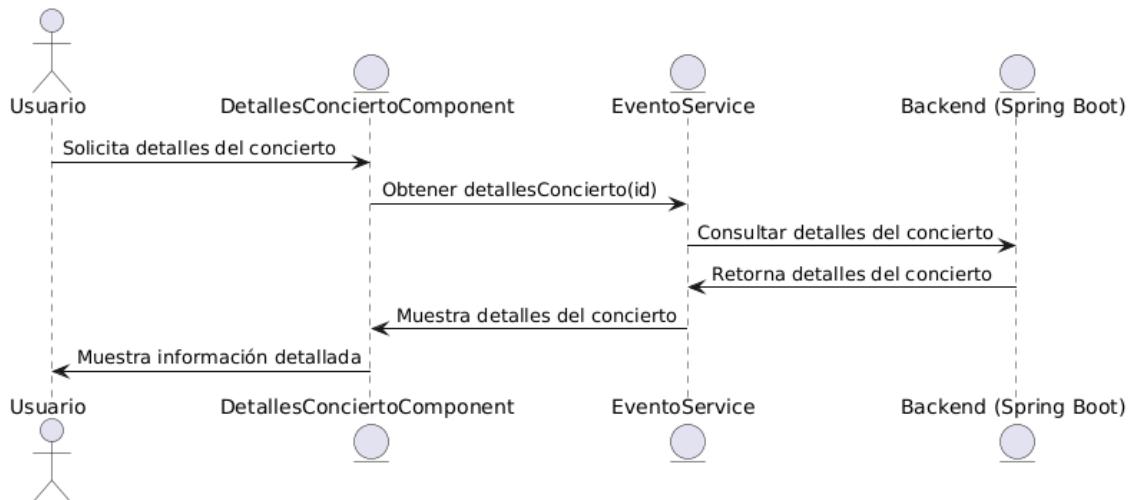
Inicio de sesión	
Precondiciones	El usuario tiene una cuenta registrada en el sistema. El usuario ha accedido a la página de inicio de sesión.
Postcondiciones	Si las credenciales son correctas, el usuario será redirigido a la página principal. Si las credenciales son incorrectas, el sistema mostrará un mensaje de error.
Actores	Usuario, LoginComponent, AuthService y Backend
Descripción	<ol style="list-style-type: none"> El Usuario introduce su nombre de usuario y contraseña. El LoginComponent recoge las credenciales e invoca al servicio AuthService para autenticar al usuario. El AuthService verifica las credenciales a través de Firebase. Si las credenciales son correctas, AuthService devuelve un token de sesión y el LoginComponent redirige al usuario a la página principal. Si las credenciales son incorrectas, AuthService devuelve un error y el LoginComponent muestra un mensaje de error.
Excepciones	Credenciales incorrectas: el sistema muestra un mensaje de error si el nombre de usuario o la contraseña son incorrectas. Error de autenticación: si ocurre un fallo en la autenticación, el sistema muestra un mensaje de error genérico.



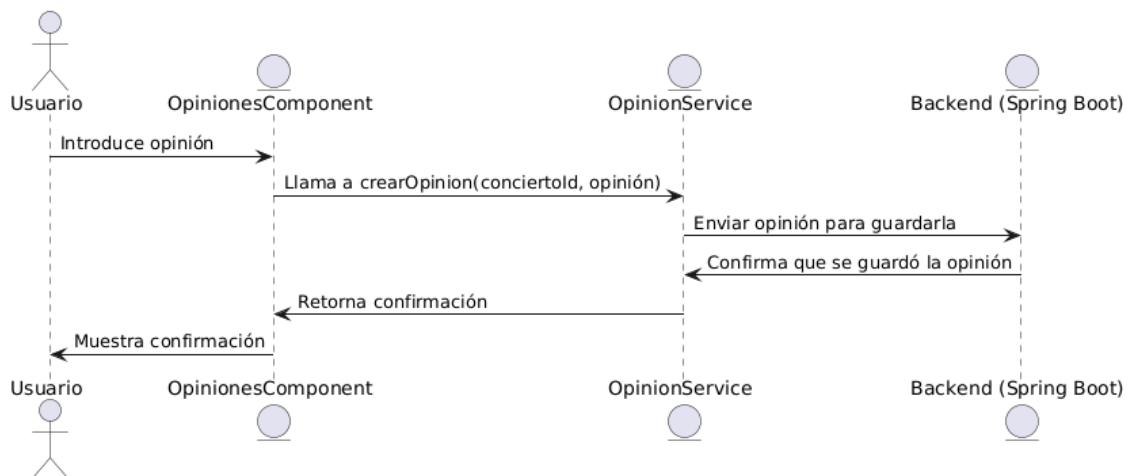
Visualización de Conciertos	
Precondiciones	El usuario ha iniciado sesión y se encuentra en una página de visualización de conciertos. El usuario está navegando por la página que lista los conciertos.
Postcondiciones	El sistema muestra la lista de conciertos disponibles. El usuario puede interactuar con la lista, ver detalles básicos de los conciertos y entrar en ellos.
Actores	Usuario, ConciertosComponent, EventService y Backend
Descripción	<ol style="list-style-type: none"> 1. El Usuario navega a la página de conciertos. 2. El ConciertosComponent invoca el servicio EventService para obtener la lista de conciertos. 3. El EventService consulta el Backend o Firebase para recuperar los datos de los conciertos. 4. El Backend responde con la lista de conciertos. 5. El ConciertosComponent muestra la lista de conciertos al Usuario en la página.
Excepciones	Error en la consulta de conciertos: si el EventService no puede recuperar la lista de conciertos, muestra un mensaje de error.



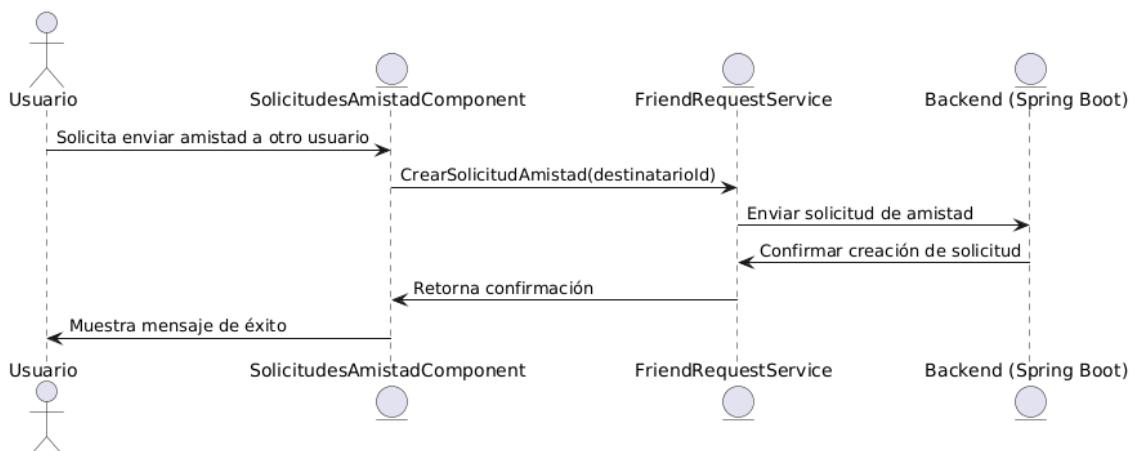
Ver Detalles de un Concierto	
Precondiciones	El usuario ha iniciado sesión y ha seleccionado un concierto de la lista. El DetallesConciertoComponent está cargado y muestra la información detallada del concierto.
Postcondiciones	El usuario ve los detalles completos del concierto seleccionado (fecha, lugar, artistas, etc.). El usuario puede interactuar con los detalles del concierto (añadir opiniones, compartir, etc.).
Actores	Usuario, DetallesConciertoComponent, EventService y Backend
Descripción	<ol style="list-style-type: none"> 1. El Usuario selecciona un concierto de la lista en la página de conciertos. 2. El DetallesConciertoComponent invoca el EventService para obtener los detalles completos del concierto seleccionado. 3. El EventService consulta el Backend para obtener los detalles del concierto. 4. El Backend responde con los datos detallados del concierto (fecha, lugar, artistas, etc.). 5. El DetallesConciertoComponent muestra los detalles al Usuario.
Excepciones	Concierto no encontrado: Si no se encuentra el concierto en la base de datos, muestra un mensaje de error.



Realizar una Opinión sobre un Concierto	
Precondiciones	El usuario ha iniciado sesión. El usuario está en la página de detalles de un concierto y ha visto la información sobre el evento.
Postcondiciones	El usuario ha podido añadir una opinión sobre el concierto. La opinión se guarda correctamente en el sistema y es visible para otros usuarios.
Actores	Usuario, OpinionesComponent, OpinionService y Backend
Descripción	<ol style="list-style-type: none"> 1. El Usuario introduce su opinión sobre el concierto en el formulario del OpinionesComponent. 2. El OpinionesComponent invoca el OpinionService para guardar la opinión en el sistema. 3. El OpinionService envía la opinión al Backend para almacenarla en la base de datos. 4. El Backend guarda la opinión y responde con una confirmación de éxito. 5. El OpinionesComponent muestra un mensaje de confirmación y la opinión se añade a la lista de opiniones.
Excepciones	Error al guardar la opinión: Si no se puede guardar la opinión en la base de datos, el sistema muestra un mensaje de error.



Enviar Solicitud de Amistad	
Precondiciones	El usuario ha iniciado sesión. El usuario quiere enviar una solicitud de amistad a otro usuario.
Postcondiciones	El sistema envía la solicitud de amistad al usuario deseado. La solicitud se almacena y el otro usuario puede aceptarla o rechazarla.
Actores	Usuario, SolicitudesAmistadComponent, FriendRequestService y Backend
Descripción	<ol style="list-style-type: none"> 1. El Usuario selecciona otro usuario para enviarle una solicitud de amistad. 2. El SolicitudesAmistadComponent invoca el FriendRequestService para crear una nueva solicitud de amistad. 3. El FriendRequestService envía la solicitud de amistad al Backend. 4. El Backend almacena la solicitud y responde con una confirmación. 5. El SolicitudesAmistadComponent muestra un mensaje de éxito al Usuario.
Excepciones	Error al enviar la solicitud: Si ocurre un error al crear la solicitud, el sistema muestra un mensaje de error. Usuario no encontrado: Si el usuario al que se le envía la solicitud no existe, muestra un mensaje de error.



6.3 Diseño de la base de datos

El Sistema de Gestión de Bases de Datos (SGBD) utilizado en este proyecto es **Firebase Firestore**, una base de datos NoSQL proporcionada por Firebase que se adapta bien a aplicaciones web y móviles. Además de Firestore, se utiliza **Firebase Authentication** para la gestión de usuarios y **Firebase Storage** para almacenar archivos, como las imágenes de perfil de los usuarios.

6.3.1 Descripción del SGBD usado

- **Nombre del SGBD:** Firebase Firestore
- **Tipo de SGBD:** NoSQL (document-based database)
- **Características clave:**
 - **Escalabilidad automática:** Firestore escala automáticamente según las necesidades de la aplicación, manejando grandes volúmenes de datos sin intervención manual.
 - **Sin esquema fijo:** No requiere que se definan esquemas antes de almacenar los datos, lo que otorga flexibilidad para modificar los datos sin necesidad de migraciones complejas.
 - **Fuerte integración con Firebase:** Firestore está integrado de forma nativa con otros servicios de Firebase como **Firebase Authentication** (para gestionar usuarios) y **Firebase Storage** (para almacenar archivos).
- **Firebase Authentication:**
 - **Gestión de usuarios:** Proporciona un sistema robusto para la autenticación y autorización de usuarios, con soporte para inicio de sesión mediante correo electrónico, redes sociales (Google, Facebook, etc.) y otros proveedores.
 - **Integración sencilla:** El proceso de autenticación se integra fácilmente con las bases de datos y otros servicios de Firebase, permitiendo la creación de usuarios, la recuperación de contraseñas y la gestión de sesiones.
- **Firebase Storage:**
 - **Almacenamiento de archivos:** Se utiliza para almacenar archivos, como imágenes de perfil de usuarios y otros archivos asociados a la aplicación.
 - **Escalabilidad:** Al igual que Firestore, Firebase Storage escala automáticamente según las necesidades, asegurando que los archivos puedan ser almacenados y descargados de manera eficiente.
- **Motivo de elección:** Firebase fue elegido por su integración nativa con otras herramientas del ecosistema, como **Firebase Authentication** para la gestión de usuarios, **Firebase Storage** para manejar archivos, y **Firestore** para la base de datos. Esta elección permite un desarrollo rápido, además de facilitar la usabilidad al estar todas las herramientas integradas.

6.3.2 Integración del SGBD en nuestro sistema

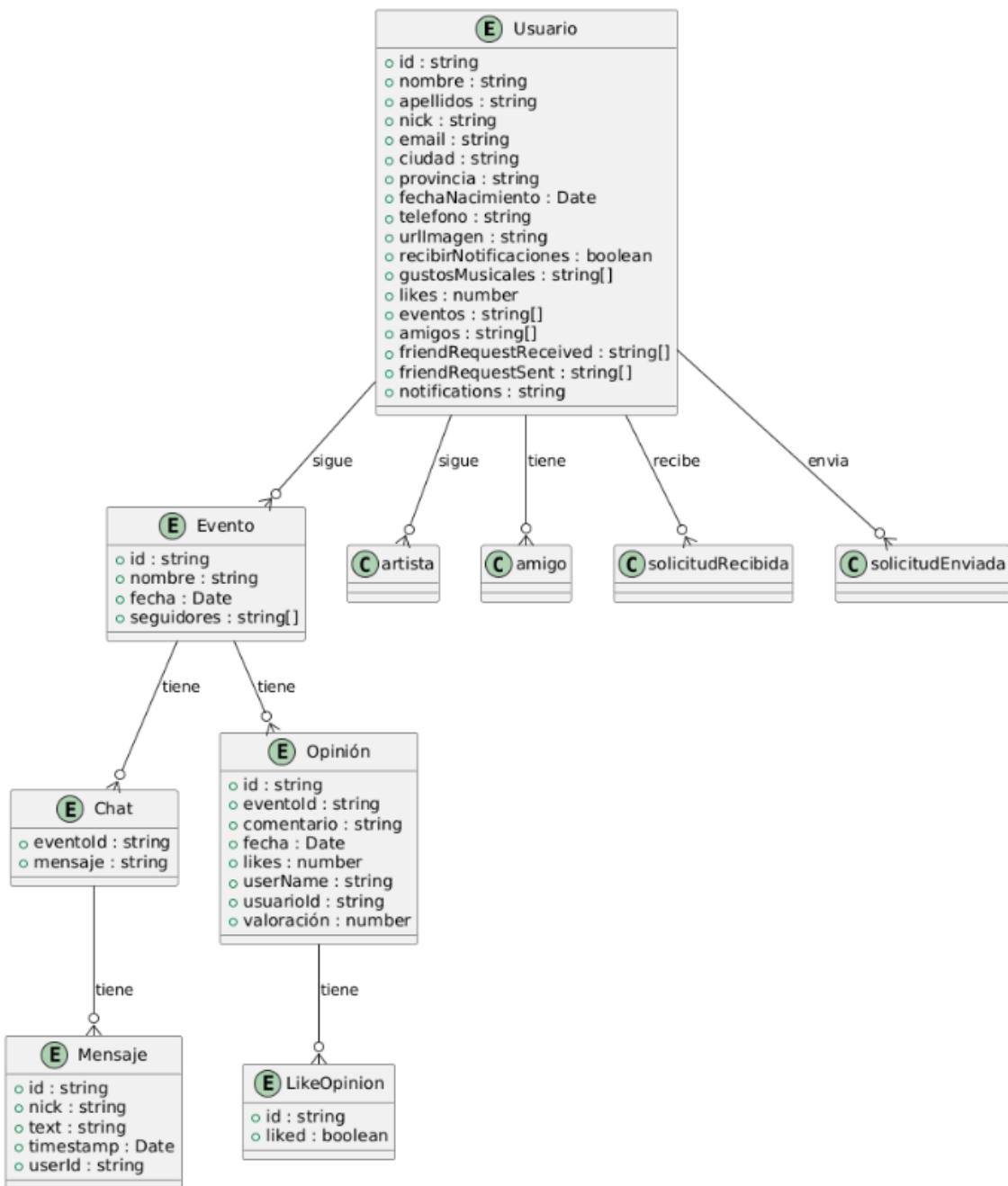
Frontend (Angular):

- **Firebase SDK:** En el frontend, se utiliza el **Firebase SDK** para conectar la aplicación Angular con **Firebase Firestore**, **Firebase Authentication** y **Firebase Storage**.
 - **Firestore:** En Angular, la integración de Firestore se realiza a través de los servicios proporcionados por Firebase. Los datos se gestionan como documentos y colecciones dentro de la base de datos, lo que facilita el manejo de los datos sin necesidad de consultas SQL complejas.
 - **Authentication:** Se utiliza el servicio de **Firebase Authentication** en Angular para permitir el registro de nuevos usuarios, el inicio de sesión y la gestión de sesiones. El flujo de autenticación se implementa en el frontend y luego se comunican los tokens con el backend para la validación de seguridad.
 - **Storage:** Firebase Storage se usa para subir y descargar archivos (por ejemplo, imágenes de perfil de usuario). En Angular, el servicio de **Firebase Storage** se emplea para manejar estos archivos y se guarda la URL de la imagen en Firestore.

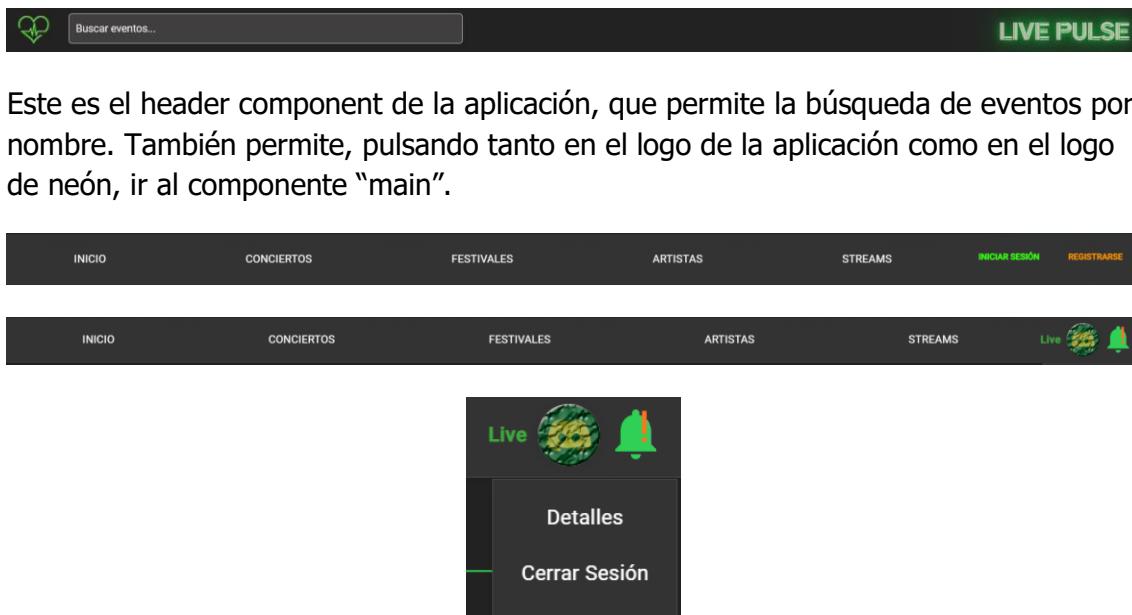
Backend (Spring Boot):

- **Spring Boot** se conecta con **Firestore** a través de la integración de **Firebase Admin SDK**.
 - **Firebase Admin SDK:** Utilizamos este SDK para realizar tareas como la verificación de tokens de autenticación enviados por el frontend, interactuar con Firestore, y subir archivos a Firebase Storage desde el servidor.
 - **Autenticación:** En el backend, se validan los tokens de Firebase enviados desde el frontend para autenticar las solicitudes de los usuarios. Esto se realiza mediante **Firebase Authentication** en combinación con el **Firebase Admin SDK** para verificar la validez de los tokens.
 - **Acceso a Firestore:** El backend puede interactuar con **Firestore** mediante el **Firebase Admin SDK**, realizando operaciones CRUD (crear, leer, actualizar y eliminar) en las colecciones de usuarios, eventos, solicitudes, etc.
 - **Firebase Storage:** Desde Spring Boot, también podemos realizar operaciones de carga y descarga de archivos a **Firebase Storage** mediante el **Firebase Admin SDK**.

6.3.3 Diagrama E-R

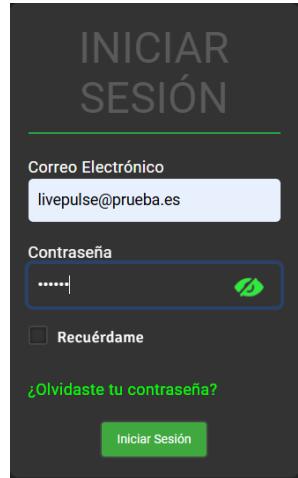


6.4 Diseño de la interfaz



Este es el header component de la aplicación, que permite la búsqueda de eventos por nombre. También permite, pulsando tanto en el logo de la aplicación como en el logo de neón, ir al componente “main”.

También permite acceder al apartado de notificaciones pulsando sobre el ícono de la campana de notificaciones.



El componente de login permite iniciar sesión con una cuenta registrada, además de activar la opción “Recuérdame” para que se guarde la cuenta, y de la opción “¿Olvidaste tu contraseña?” que acceder al componente reset-password.



El componente “reset-password” permite enviar un email para recuperar la contraseña para logearse en la aplicación.

El componente “detalles-usuario” muestra información sobre el usuario que está logeado, incluyendo su Nick, email, el botón “Editar Perfil” que lleva al componente “modificar-usuario” y una lista con los eventos y artistas seguidos por el usuario, así como los amigos del usuario.

MODIFICAR USUARIO

Nombre
Live

Apellidos
Pulse

Nick
LivePulse

Ciudad
Oviedo

Provincia
Asturias

Teléfono
987654321

Notificaciones
Al activar esta opción, estarás aceptando recibir comunicaciones y noticias relevantes por email.

Gustos Musicales

Bluegrass
Blues
Christian
Classical
Country Music

EDM
Folk
Hip Hop & Rap
Indie
Jamband
Jazz

CAMBIAR CONTRASEÑA

Contraseña Actual

Nueva Contraseña

Confirmar Contraseña

Guardar Contraseña

SUBIR IMAGEN

Seleccionar archivo Ningún archivo seleccionado

Guardar Imagen

Eliminar Cuenta

El componente “modificar-usuario” permite modificar, en tres formularios independientes, los datos del usuario, cambiar la contraseña, cambiar la imagen de perfil y eliminar la cuenta.

GÉNERO

- Bluegrass
- Blues
- Christian
- Classical
- Country Music
- EDM
- Folk
- Hip Hop & Rap
- Indie
- Jamband
- Jazz
- K-pop
- Latin
- Metal
- Pop
- Punk
- R&B / Soul
- Reggae
- Rock
- Otros



Falling In Reverse at
Razzmatazz
2024-11-11
Barcelona, Spain

+ Info + Seguir



Angelina Mango at Sala Apolo
2
2024-11-11
Barcelona, Spain

+ Info + Seguir



Trentemøller at Sala Apolo
2024-11-11
Barcelona, Spain

+ Info + Seguir



Pat Metheny at Auditorio
Nacional, Sala Sinfónica
2024-11-11
Madrid, Spain

+ Info + Seguir



Abhir Hathi at Chango Club
2024-11-11
Madrid, Spain

+ Info + Seguir



Russian Circles at Salamandra
2024-11-11
L'Hospitalet de Llobregat, Spain

+ Info + Seguir

El componente “Conciertos” muestra la lista de próximos conciertos comenzando por los más recientes, y permite su filtrado por género.

La diferencia entre estar logeado y no aquí se basa en el botón “+Seguir”, que solo aparece en caso de estar logeados.



En la parte superior del componente “detalles-conciertos” podemos ver información relativa a cada concierto con los datos que podemos ver en la imagen, así como la ubicación del concierto. También desde dentro del propio festival podemos escoger la opción de seguir o dejar de seguir un concierto.

The screenshot shows four sections at the bottom of a page. On the left is a "SEGUIDORES" (Followers) section with a placeholder for follower names. Next is a "CHAT" (Chat) section with an input field and a green "Enviar" (Send) button. Below that is a "ÚLTIMA OPINIÓN" (Last Opinion) section with a message "No hay opiniones disponibles." (No reviews available). Finally, there is a "COMPARTE TU OPINIÓN" (Share Your Opinion) section with a "Comentario" (Comment) input field, a five-star rating placeholder "Tu valoración del evento" (Your event rating), and a green "Enviar Opinión" (Send Opinion) button.

En la parte inferior podemos ver los seguidores que tiene ese concierto, el chat asociado al concierto, la última opinión en caso de tener opiniones y el apartado dónde se añaden las opiniones. Tanto el “chat” como la opción de “comparte tu opinión” solo están habilitados en caso de estar logeados.

El componente “Festivales” funciona igual que el de “Conciertos”, aunque en “detalles-festival” podemos ver ligeras diferencias.



Como se puede apreciar la única diferencia es la inserción de iconos de RRSS asociadas al festival.

A screenshot of the festival review section on the Live Pulse platform. It is divided into four panels: "SEGUIDORES" (Followers) showing a profile picture; "CHAT" (CHAT) with a message input field and a send button; "ÚLTIMA OPINIÓN" (Last Opinion) showing a review from "LivePulse" dated 10/11/2024, rating 5 stars, and a "Ver todas las opiniones" (View all reviews) button; and "COMPARTE TU OPINIÓN" (Share your opinion) with a comment input field, a star rating section, and a "Enviar Opinión" (Send Opinion) button.

Como podemos ver, la parte inferior funciona de igual modo.

GÉNERO

- Bluegrass
- Blues
- Christian
- Classical
- Country Music
- EDM
- Folk
- Hip Hop & Rap
- Indie
- Jamband
- Jazz
- K-pop
- Latin
- Metal
- Pop
- Punk
- R&B / Soul
- Reggae




BILLY STRINGS

24 Conciertos

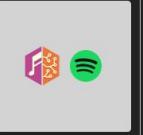
+ Siguendo



OLD CROW MEDICINE SHOW

10 Conciertos

+ Siguendo



TRAMPLED BY TURTLES

16 Conciertos

- Siguendo



THE STEELDRIVERS

6 Conciertos

- Siguendo



ALISON KRAUSS

0 Conciertos

+ Seguir



JERRY GARCIA

0 Conciertos

+ Seguir



THE DEVIL MAKES THREE

17 Conciertos

+ Seguir



WATCHHOUSE

1 Conciertos

+ Seguir



JUDAH & THE LION

2 Conciertos

+ Seguir



ERIC JOHNSON

0 Conciertos

+ Seguir

El componente “artistas” funciona de manera similar a conciertos y festivales, pero no se acceder a los detalles del artista, sino que al pulsar sobre la imagen de alguno de los artistas nos aparecen enlaces tanto al Spotify como al MusicBrainz del artista.

GÉNERO

- Bluegrass
- Blues
- Christian
- Classical
- Country Music
- EDM
- Folk
- Hip Hop & Rap
- Indie
- Jamband
- Jazz
- K-pop
- Latin
- Metal
- Pop
- Punk
- R&B / Soul
- Reggae
- Rock
- Otros



The Jeb Stream

2024-11-11

Stream Mundial

● Comprar Entradas

+ Seguir



Ruthie Collins Stream

2024-11-11

Stream Mundial

● Comprar Entradas

+ Seguir



Nathalie Miranda Stream

2024-11-11

Stream Mundial

● Comprar Entradas

+ Seguir



The Jeb Stream

2024-11-12

Stream Mundial

● Comprar Entradas

+ Seguir



Jazzy Jams Stream

2024-11-12

Stream Mundial

● Comprar Entradas

+ Seguir



Aaron English Band Stream

2024-11-12

Stream Mundial

● Comprar Entradas

+ Seguir

El componente “streams” también es similar a los anteriores, con la peculiaridad de que no tampoco tiene detalles, simplemente enlace a la compra de entradas en caso de que sea de pago. El círculo rojo indica que el stream es de pago, y si aparece en verde que es gratuito. También se ofrece la posibilidad de seguir los streams y filtrarlos por género.

The screenshot shows two sections: 'SOLICITUDES DE AMISTAD' (Friend Requests) and 'SOLICITUDES ENVIADAS' (Sent Requests). Under 'SOLICITUDES DE AMISTAD', there is a placeholder for a profile picture with the number '123' and buttons for 'Aceptar' (Accept) and 'Rechazar' (Reject). Under 'SOLICITUDES ENVIADAS', there is also a placeholder for a profile picture with the number '123' and the text 'Solicitud Enviada' (Sent Request). Below these sections is a 'NOTIFICACIONES' (Notifications) section containing a green message: 'X 123 ha aceptado tu solicitud de amistad.' (User 123 has accepted your friend request.)

El componente “notificaciones” permite ver las solicitudes de amistad enviadas y pendientes, así como las notificaciones, que hasta el momento se limitan a las relacionadas con estas solicitudes de amistad. La “X” de las notificaciones nos permite eliminarlas.

The screenshot displays a grid of five event cards under the heading 'PRÓXIMOS EVENTOS' (Upcoming Events). Each card includes a thumbnail image, the event name, dates, location, and a '+ Info' button.

Event	Date	Location	Action
Falling In Reverse at Razzmatazz	2024-11-11 - 2024-11-11	Barcelona, Spain	+ Info
Trentemøller at Sala Apolo	2024-11-11 - 2024-11-11	Barcelona, Spain	+ Info
Pat Metheny at Auditorio Nacional, Sala Sinfónica	2024-11-11 - 2024-11-11	Madrid, Spain	+ Info
Ahmir Hathi at Chango Club	2024-11-11 - 2024-11-11	Madrid, Spain	+ Info
Russian Circles at Salamandra	2024-11-11 - 2024-11-11	L'Hospitalet de Llobregat, Spain	+ Info

Por último, el componente “main” nos permite ver los próximos 5 eventos (ya sean festivales o conciertos” por orden de fecha más reciente en la parte superior.

The screenshot shows a list of four news articles under the heading 'ÚLTIMAS NOTICIAS' (Latest News), each with a thumbnail image, title, and a brief description.

- Festival Internacional de Música de Morelia: Conciertos en Zamora ...**
14 hours ago ... Algunos de los conciertos serán gratuitos, mientras otros tendrán un costo. Festival de Música de Morelia / Foto: Fernando Maldonado | El Sol de ...
- Diputación de Salamanca · Presentación del III Ciclo de Conciertos ...**
3 days ago ... Presentación del III Ciclo de Conciertos de Órgano de la Provincia de Salamanca. Los Sonidos Centenarios 2023 ... Noticias. IV Festival Familiar de Teatro ...
- El Festival Abierto de Música de Órgano trae doce conciertos**
3 days ago ... El cubano Alejandro Vargas abre hoy el ciclo, que se celebrará hasta el mes de mayo.
- Festival EuroJazz ofrecerá tres conciertos en Pachuca - El Sol de ...**
5 days ago ... El encuentro artístico abrirá el próximo viernes 8 de noviembre a las 18 horas en el Teatro San Francisco.

En la parte inferior nos permite ver noticias relacionadas con festivales y conciertos en España gracias a la API de Google Search.



El componente “footer” nos muestra una serie de información y enlaces que aún no están implementados debido a que aún no se han creado RRSS, términos, condiciones, etc....

Capítulo 7. Implementación del sistema

7.1 Estándares y normas seguidos

1. Uso de Git y GitHub como Sistema de Control de Versiones

Se utilizó GitHub como repositorio central y Git como sistema de control de versiones distribuido. Esto permitió gestionar el código del proyecto de manera eficiente y colaborativa, con un enfoque en los siguientes estándares:

Prácticas seguidas:

- **Estructura de ramas estándar:**
 - Una rama principal (master) utilizada para el código estable y listo para producción.
- **Commits descriptivos:** Cada commit se realizó añadiendo una descripción indicando las últimas funcionalidades operativas. Los commits finales también incluyen fechas.

Beneficios:

- Permite un historial claro del desarrollo del proyecto.
 - Facilita la revisión de código (*pull requests*).
 - Mejora la colaboración entre los desarrolladores.
 - Permite regresar a commits antiguos en caso de errores sin identificar.
-

2. Angular: Organización y Estandarización del Código

En el frontend, desarrollado con Angular, se han seguido normas para mantener el código limpio, organizado y fácil de mantener.

Prácticas seguidas:

- **Convención de nombres:** Archivos, clases y componentes siguen las convenciones de Angular:
 - Componentes: nombre-del-componente.component.ts
 - Servicios: nombre-del-servicio.service.ts
 - Módulos: nombre-del-modulo.module.ts
- **Uso de TypeScript:** La tipificación estática asegura que los errores se detecten en tiempo de desarrollo.
- **Sistema de enrutamiento:** Configuración centralizada en app-routing.module.ts para facilitar la navegación.

Uso de Bootstrap 5:

- Se utilizó **Bootstrap 5** para diseñar la interfaz de usuario, asegurando consistencia visual y diseño responsivo.
- Componentes utilizados:
 - **Navbar:** Implementada como barra de navegación fija y adaptable en todas las pantallas.
 - **Cards:** Usadas para mostrar eventos y opiniones de manera clara.
 - **Formularios estilizados:** Para autenticación y actualización de datos.

Beneficios:

- Garantiza que la aplicación sea accesible y se adapte correctamente a dispositivos móviles y de escritorio.
-

3. Spring Boot: Estilo y Organización del Backend

El backend se desarrolló siguiendo las mejores prácticas de Spring Boot, asegurando que los servicios sean escalables, seguros y fáciles de extender.

Prácticas seguidas:

- **Arquitectura RESTful:** Cada endpoint sigue las convenciones REST, por ejemplo:
 - GET /api/usuarios/{id} para obtener un usuario.
 - POST /api/usuarios para registrar un nuevo usuario.
- **Controladores organizados:** Los controladores (@RestController) manejan las peticiones HTTP, mientras que la lógica de negocio se separa en servicios (@Service).

Beneficios:

- Asegura la separación de responsabilidades.
 - Facilita la integración con el frontend mediante APIs claras.
-

4. Principios de Código Limpio (Clean Code)

Durante el desarrollo, se siguieron los principios de **Código Limpio**, descritos por Robert C. Martin en su libro *Clean Code*, para mantener la calidad del código.

Prácticas aplicadas:

- **Nombres significativos:** Se eligieron nombres claros y descriptivos para variables, funciones, clases y métodos.
- **Funciones pequeñas y específicas:** Cada función realiza una única tarea claramente definida.
- **Reducción de duplicación:** Código común encapsulado en servicios reutilizables.
- **Comentarios útiles:** Se añadieron comentarios únicamente donde el código no era autoexplicativo, evitando redundancia innecesaria.

Beneficios:

- Mejora la legibilidad y mantenibilidad del código.
 - Facilita la incorporación de nuevos desarrolladores al proyecto.
-

5. Validación del Cumplimiento de Estándares

- **Revisión manual:** Durante el desarrollo, cada funcionalidad nueva fue revisada mediante *pull requests* en GitHub, asegurando que cumpliera con las normas establecidas.
- **Pruebas automáticas:**
 - El frontend fue probado mediante Angular Testing Utilities (Jasmine/Karma).

7.2 Lenguajes de programación

En este proyecto se han utilizado diversos lenguajes de programación y herramientas relacionadas, cada uno adaptado a las necesidades específicas de las distintas capas de la arquitectura (frontend, backend y base de datos). A continuación, se describen los lenguajes empleados, las versiones utilizadas y los módulos o complementos destacados:

1. Angular (*TypeScript*)

- **Lenguaje:** TypeScript

TypeScript, una extensión tipada de JavaScript, ha sido utilizado para el desarrollo del frontend. Gracias a sus características como el tipado estático y la compatibilidad con ECMAScript, ha facilitado la creación de un código robusto y mantenible.

- **Versión utilizada:**

- **Angular CLI:** 18.2.1
- **TypeScript:** 5.4.2

- **Complementos y módulos destacados:**

- **Bootstrap 5:** Para diseño y estilos de la interfaz.
- **Angular Material:** Algunos componentes visuales como botones y formularios.
- **RxJS:** Gestión de programación reactiva y manejo de observables.
- **AngularFire:** Biblioteca oficial para integrar Firebase con Angular.
 - Módulos empleados:
 - AngularFireAuth: Integración de Firebase Authentication.
 - AngularFireStore: Acceso a Firestore Database.
 - AngularFireStorage: Manejo de archivos en Firebase Storage.

2. Spring Boot (Java)

- **Lenguaje:** Java

Java ha sido el lenguaje empleado para el desarrollo del backend, siendo Spring Boot el framework principal para gestionar las operaciones, crear una API RESTful y conectar con Firebase.

- **Versión utilizada:**

- **Spring Boot:** 3.3.1
- **Java JDK:** 17

- **Complementos y bibliotecas utilizadas:**

- **Spring Security:** Gestión de autenticación y autorización.
 - **Firebase Admin SDK:** Conexión con Firebase desde el backend.
 - **Jackson:** Serialización y deserialización de datos JSON.
 - **Lombok:** Simplificación del código mediante anotaciones como @Data y @Builder.
 - **Spring Web:** Para la construcción de controladores y endpoints REST.
 - **Spring Data JPA:** Aunque no se emplea una base de datos SQL, algunos conceptos de repositorios se adaptan al manejo de Firestore.
-

3. Firebase

Aunque Firebase no requiere un lenguaje de programación específico, se utiliza un lenguaje similar a JavaScript para definir **reglas de seguridad** y manejar funciones en la **Firestore Database**.

- **Reglas de seguridad (Firestore):**

- Lenguaje de reglas para definir permisos y restricciones sobre la base de datos.

- **Funciones empleadas:**

- Validación de accesos.
 - Control de estructuras de datos permitidas.
-

4. HTML5, CSS3 y SCSS

- **HTML5:**
Lenguaje estándar utilizado para estructurar las vistas de la aplicación.
 - **CSS3 y SCSS:**
Para definir los estilos y apariencia del frontend, con un enfoque modular y escalable.
-

5. Distribución y entorno

- **Frontend:**
El proyecto Angular se desarrolló y compiló para producción utilizando el entorno Node.js (v20.12.1) y NPM (v10.5.0).
 - **Backend:**
Se utiliza una máquina virtual Java y Spring Boot, desplegable en entornos locales o en servidores compatibles con JVM.
 - **Firebase:**
Utilizado como servicio cloud-hosted, accesible mediante las bibliotecas de Firebase en el frontend y backend.
-

6. Razones de la elección de estos lenguajes

- **TypeScript/Angular:** Además de por su conocimiento previo en el grado, por su fuerte tipado y capacidad para construir aplicaciones dinámicas y responsivas.
- **Java/Spring Boot:** Además de su conocimiento previo en el grado, por su robustez, amplia comunidad de soporte y excelente integración con servicios externos como Firebase.
- **Firebase:** Por ser una solución integral que combina base de datos, autenticación y almacenamiento, todo alojado en la nube.

7.3 Herramientas y programas usados para el desarrollo

En el desarrollo de este proyecto, hemos empleado una variedad de herramientas, entornos de desarrollo integrados (IDE), bibliotecas y servicios adicionales que han facilitado la implementación, pruebas y mantenimiento del sistema. A continuación, se detallan las herramientas utilizadas, su versión y propósito:

Visual Studio Code

- **Versión utilizada:** 1.95.3
 - **Propósito:** Este editor de texto se empleó para la implementación del frontend de la aplicación desarrollado en Angular. Se utilizó por su facilidad de configuración, integración con TypeScript y herramientas adicionales como extensiones para Angular y Firebase.
 - **Interacción con el sistema:** Facilitó la escritura del código del frontend, la integración con Firebase, y la ejecución de pruebas locales del mismo.
-

IntelliJ IDEA

- **Versión utilizada:** Ultimate 17.0.10
 - **Propósito:** Se utilizó para la implementación del backend desarrollado en Spring Boot. IntelliJ proporcionó herramientas avanzadas de autocompletado, gestión de dependencias con Maven, y pruebas integradas.
 - **Interacción con el sistema:** Sirvió para gestionar la lógica del servidor, la interacción con Firebase Firestore, y la creación de endpoints RESTful.
-

GitHub

- **Versión utilizada:** Servicio en línea.
 - **Propósito:** Control de versiones y colaboración. GitHub permitió gestionar los cambios en el código fuente, mantener un historial del proyecto y colaborar en diferentes ramas del desarrollo.
 - **Interacción con el sistema:** Se configuró para sincronizar tanto el frontend como el backend.
-

Firebase

- **Servicios utilizados:**
 - **Firestore Database** (para el almacenamiento de datos).
 - **Firebase Authentication** (para la autenticación de usuarios).
 - **Firebase Storage** (para guardar imágenes de perfil y otros archivos).
 - **Propósito:** Proporcionar una base de datos en la nube y servicios asociados que garantizaron una implementación rápida y escalable del backend.
 - **Interacción con el sistema:** Firebase se integró tanto con el frontend como con el backend mediante sus SDK y la configuración de reglas personalizadas de Firestore.
-

Visual Paradigm

- **Versión utilizada:** Web
 - **Propósito:** Diseño y modelado del sistema. Visual Paradigm fue utilizado para la creación de diagramas UML, incluyendo diagramas de clases y de actividad.
 - **Interacción con el sistema:** Los diagramas creados ayudaron a visualizar la estructura del sistema y a documentar su funcionamiento para facilitar el desarrollo colaborativo.
-

PlantUML

- **Versión utilizada:** Web
 - **Propósito:** Creación de diagramas de actividad, secuencia y otros diagramas UML directamente a partir de texto.
 - **Interacción con el sistema:** Permitió documentar los flujos de actividades de los servicios principales de manera clara y rápida.
-

Google Chrome

- **Versión utilizada:** Últimas versiones estables.
 - **Propósito:** Pruebas del frontend. Se utilizó principalmente para probar la interfaz y ejecutar el código Angular.
 - **Interacción con el sistema:** A través del navegador, se verificó la correcta integración entre el frontend y los servicios del backend y Firebase.
-

Postman

- **Versión utilizada:** 11.18.0
 - **Propósito:** Pruebas de las API REST creadas en el backend.
 - **Interacción con el sistema:** Se utilizó para verificar el comportamiento de los endpoints de Spring Boot, comprobar la estructura de las respuestas y simular peticiones con datos de prueba.
-

Leaflet

- **Versión utilizada:** 1.9.4.
 - **Propósito:** Implementación de mapas interactivos en el frontend.
 - **Interacción con el sistema:** Leaflet permitió visualizar ubicaciones dentro de la aplicación, con datos cargados desde Firebase o calculados en tiempo real.
-

Google Search API

- **Propósito:** Proporcionar datos de búsqueda a la aplicación.
 - **Interacción con el sistema:** Se integró en el backend para enriquecer los resultados y la experiencia del usuario.
-

JamBase API

- **Propósito:** Obtener información de eventos musicales y artistas.
 - **Interacción con el sistema:** Se usó como fuente de datos para los eventos mostrados en la aplicación.
-

Bootstrap 5

- **Versión utilizada:** 5.3.2.
 - **Propósito:** Framework de diseño CSS utilizado para crear una interfaz de usuario responsive y estilizada.
 - **Interacción con el sistema:** Bootstrap se usó en el frontend para diseñar componentes de la interfaz de usuario, como menús, formularios y botones, garantizando una experiencia visual atractiva.
-

7.4 Creación del sistema

A lo largo del desarrollo del proyecto, hemos enfrentado varios desafíos técnicos, especialmente relacionados con la integración de diversas tecnologías. A continuación, se detallan algunos de los problemas más destacados:

7.4.1 Problema 1: Conexión de Firebase con el Frontend (Angular)

Descripción del problema:

Uno de los principales problemas que se presentaron fue la conexión de **Firebase** con el frontend de la aplicación en **Angular**. Inicialmente, el frontend no podía comunicarse correctamente con **Firebase Authentication** ni con **Firestore Database**. El error principal parecía ser una configuración incorrecta de las credenciales de Firebase, lo que causaba que las solicitudes a la base de datos y la autenticación no funcionaran correctamente.

Solución:

La solución consistió en revisar detalladamente los archivos de configuración de Firebase en el proyecto de **Angular**. Aunque distintas fuentes parecían indicar que debía indicar las claves API en el archivo “environments.ts”, finalmente lo único que me funcionó fue poner directamente las claves API y configuración de Firebase directamente en el archivo “app.module.ts”.

7.4.2 Problema 2: Almacenamiento de imágenes de perfil en Firebase Storage

Descripción del problema:

Otro desafío importante fue el almacenamiento de las imágenes de perfil de los usuarios. El proceso involucraba subir imágenes a **Firebase Storage** y luego obtener la URL asociada para almacenarla en **Firestore Database**. Inicialmente, no pudimos obtener la URL de las imágenes correctamente después de que se subieran a Storage. El problema era que la URL no se generaba de inmediato y, al intentar almacenarla en la base de datos, se generaba un error de sincronización.

Solución:

La solución fue implementar un manejo asincrónico adecuado de las operaciones de Firebase. Se utilizó la función `getDownloadURL()` de Firebase Storage para obtener la URL una vez que la imagen se cargara completamente. Para ello, usamos un enfoque basado en **Promesas** y **Observables**. Aseguramos que la obtención de la URL solo ocurriera después de que el archivo se hubiera cargado exitosamente en **Firebase Storage**.

7.4.3 Problema 3: Barra de búsqueda de eventos

Descripción del problema:

Uno de los problemas más complejos fue la implementación de la barra de búsqueda de eventos. La función de búsqueda debía filtrar los eventos en **JamBase** según varios criterios, como el nombre del evento y la ubicación. Sin embargo, al principio, la barra de búsqueda no devolvía resultados de manera eficiente y, en algunos casos, ni siquiera mostraba eventos, aunque deberían aparecer en los resultados.

Solución:

Fue necesario modificar la lógica de la barra de búsqueda para hacerla más flexible, permitiendo búsquedas por múltiples parámetros.

7.4.4 Problema 4: Sincronización entre Firebase Authentication y Firestore

Descripción del problema:

Otro problema surgió cuando intentamos sincronizar los datos de usuario entre **Firebase Authentication** y **Firestore Database**. Mientras que la autenticación de usuarios se manejaba correctamente con **Firebase Authentication**, los datos adicionales de los usuarios (como sus eventos, amigos, preferencias, etc.) se almacenaban en Firestore, pero la sincronización no se producía correctamente entre ambas plataformas.

Solución:

Para solucionar este problema, implementamos una lógica en el backend que actualiza automáticamente los datos del usuario en **Firestore** cada vez que un nuevo usuario se registra o inicia sesión.

7.4.5 Problema 5: Manejo de la paginación de eventos y usuarios

Descripción del problema:

Cuando se empezó a manejar un mayor volumen de datos (usuarios y eventos), notamos que la carga de todos los eventos de una sola vez era muy ineficiente y causaba bloqueos en la aplicación, afectando la experiencia del usuario. El desafío consistió en implementar una paginación eficiente tanto para los eventos como para los usuarios.

Solución:

La solución fue implementar un sistema de paginación basado en consultas de JamBase que solo trajeran los datos necesarios para cada página, en lugar de cargar todo de una vez. Utilizamos un máximo de eventos por página para traer un número específico de eventos en cada consulta, reduciendo así la carga y mejorando el rendimiento.

7.4.6 Problema 8: Doble llamada a getUserId durante el login

Descripción del problema:

Uno de los problemas que surgió durante el proceso de autenticación de los usuarios fue que al intentar hacer el login con **Firebase Authentication**, se realizaban dos llamadas al método getUserId en el backend. La primera llamada devolvía un objeto de usuario con los datos vacíos, y la segunda ya traía la información correcta del usuario. Este comportamiento causaba un error en el frontend, ya que los datos del usuario estaban vacíos durante la primera llamada, lo que impedía la correcta visualización de la interfaz de usuario. Además, al ser dos llamadas consecutivas, el sistema se volvía más lento, lo que afectaba la experiencia del usuario.

Solución:

La solución fue investigar el flujo de autenticación en el frontend y backend. Descubrimos que la primera llamada al método getUserId se estaba haciendo innecesariamente debido a un comportamiento asíncrono incorrecto en el flujo de inicio de sesión. Al principio, el frontend hacía una solicitud al backend para obtener los datos del usuario justo después de la autenticación, pero debido a la naturaleza asíncrona del proceso, la primera solicitud se hacía antes de que el usuario estuviera completamente autenticado y sus datos estuvieran disponibles.

Para solucionar esto, ajustamos el flujo de autenticación para asegurarnos de que el método getUserId solo se llamara después de que **Firebase Authentication** confirmara que el usuario estaba correctamente autenticado.

7.4.7 Problema 9: Filtro por género de eventos con múltiples géneros musicales

Descripción del problema:

El filtro por género musical en los eventos causó problemas debido a que muchos eventos tienen más de un género musical asociado. Además, hubo algunos eventos que no tenían ningún género musical asociado en la API. Esto generó complicaciones tanto en la visualización de los resultados como en la lógica del filtro, ya que se esperaba que todos los eventos tuvieran un género, pero algunos no lo tenían y otros tenían varios géneros.

Solución:

La solución fue ajustar la lógica del filtro para manejar múltiples géneros asociados a un evento, así como los eventos que no tienen géneros. Primero, para los eventos que tenían más de un género, decidimos permitir que un evento pudiera coincidir con cualquier género dentro de una lista de géneros seleccionados por el usuario. Para ello, implementamos un filtro que verificara si al menos uno de los géneros del evento coincidía con el género seleccionado en el filtro.

En cuanto a los eventos que no tenían géneros asignados, se decidió que apareciesen en todas las búsquedas, es decir, como si incluyesen todos los géneros.

7.4.8 Problema 10: Filtro de artistas que requiere al menos un filtro para evitar errores en la API

Descripción del problema:

Un problema relacionado con la consulta a la **API de JamBase** fue que, al intentar cargar todos los artistas sin aplicar ningún filtro, la consulta a la API fallaba. La API requería al menos un filtro para devolver resultados, y cuando se intentaba realizar una búsqueda sin filtros, la respuesta era un error o un mal funcionamiento de la API.

Solución:

Para solucionar este problema, decidimos implementar una validación en el frontend para asegurarnos de que siempre se aplicara al menos un filtro antes de hacer la solicitud a la API.

Capítulo 8. Manuales del sistema

8.1 Manual de usuario

Lo primero que nos encontraremos en nuestra aplicación será nuestra página principal, que se divide en dos partes diferenciadas.

Una parte superior con los próximos eventos ordenados cronológicamente, apareciendo los 5 más próximos.



Y una parte inferior con las últimas noticias sobre conciertos y festivales en España.

Conciertos. Noticias sobre conciertos | Diario de Navarra

2 days ago ... gira con su nuevo disco, 'Futuros imposibles'. Este viernes actúan en la Sala Zentral, en el primer concierto del festival Santas Pascuas. Nerea Alejos. 14/11 ...

Un concierto de Navidad en San Miguel, una de las novedades de ...
4 days ago ... festival de música más antiguo de España, que se celebrará el próximo mes de diciembre. Contará con fila cero para los afectados por la dana.

Montoro celebrará su primer Festival de Música Sacra con ...
4 days ago ... Montoro celebrará su primer Festival de Música Sacra con conciertos en los espacios más emblemáticos. Tendrá lugar en el teatro Miguel Romero ...

Diputación de Salamanca · Presentación del III Ciclo de Conciertos ...
5 days ago ... Noticias. IV Festival Familiar de Teatro Provincia a Escena. 02/10/2024. IV ... concierto inaugural en Alba de Tormes de la mano de Saskia Roures (órgano) ...

Continuaremos ahora con el grueso de este manual y su parte de registro. En la parte superior de nuestra web podemos encontrar los botones de “**INICIAR SESIÓN**” y “**REGISTRARSE**”.



Aunque para consultar la mayor parte de la aplicación no necesitamos estar registrados, hay que tener en cuenta que se trata de una especie de red social, en la que compartir, opinar y socializar es la motivación principal.

Una vez elegimos la opción “**REGISTRARSE**” podremos ver un formulario de registro, que nos indica que los campos marcados con (*) son obligatorios. En caso de no llenar todos esos campos, el formulario no será válido y no permitirá registrarse.

The registration form is titled "REGISTRO DE USUARIO" at the top. It includes fields for "Email(*)", "Contraseña(*)", "Nombre(*)", "Apellidos(*)", "Nick(*)", and "Fecha de Nacimiento*". There is also a "Ciudad" field with placeholder "Introduce tu ciudad", a "Provincia" field with placeholder "Introduce tu provincia", a "Teléfono" field with placeholder "Introduce tu teléfono", and a checkbox for "Recibir notificaciones" with a descriptive note below it. On the right side, there is a section titled "Gustos Musicales" containing a grid of genre buttons: Bluegrass, Blues, Christian, Classical, Country Music, EDM, Folk, Hip Hop & Rap, Indie, Jamband, Jazz, K-pop, Latin, Metal, Pop, Punk, R&B / Soul, Reggae, and Rock. A "Regístrate" button is located at the bottom right of the form.

Una vez registrados, podremos iniciar sesión en el botón “**INICIAR SESIÓN**” utilizando el email y contraseña utilizados en el registro (si este ha sido correcto).

The login form is titled "INICIAR SESIÓN". It has fields for "Correo Electrónico" (placeholder: "Ingrese su correo electrónico") and "Contraseña" (placeholder: "Ingrese su contraseña"). There is also a "Recuérdame" checkbox and a "¿Olvidaste tu contraseña?" link. A "Iniciar Sesión" button is at the bottom.

En la pantalla de Iniciar Sesión podemos elegir ver o no nuestra contraseña en el ícono del ojo. También podemos elegir la opción “**Recuérdame**” para que la cuenta de correo se quede guardada permanentemente en nuestro navegador, teniendo en cuenta que la contraseña nunca se guardará. Y también tenemos la opción “**¿Olvidaste tu contraseña?**”, la cual nos llevará a una nueva pantalla en caso de haber olvidado nuestra contraseña.

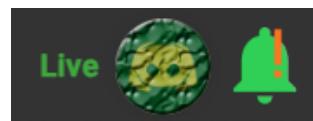


En esta nueva pantalla debemos introducir nuestro correo electrónico, y la aplicación se encargará de enviar un email con el que podremos indicar una nueva contraseña. Aquí tenemos algunas capturas del proceso:

The image contains two screenshots. The left one shows a mobile application's password reset screen with the title "REESTABLECER CONTRASEÑA". It has a field for "Correo Electrónico" containing "pofresno92@gmail.com" and a green "Restablecer Contraseña" button. Below the button, a message says "Se ha enviado un correo electrónico para restablecer la contraseña.". The right screenshot shows an email from "noreply@live-pulse-787b1.firebaseio.com" with the subject "para mí". The email body starts with "Hola," and provides instructions to "Pulsa el siguiente enlace para elegir una nueva contraseña para tu cuenta de Live Pulse." followed by a URL: "https://live-pulse-787b1.firebaseio.com/_auth/action?mode=resetPassword&oobCode=". Below the URL is an ellipsis "...".

Reset your password
for pofresno92@gmail.com
New password
|
SAVE

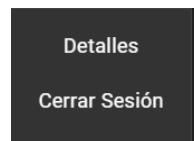
Una vez hemos iniciado sesión con nuestro Usuario y Contraseña, la interfaz cambiará ligeramente. En lugar de los botones de “**INICIAR SESIÓN**” y “**REGISTRARSE**” tendremos la siguiente información: Nombre del Usuario, Imagen de Perfil del Usuario e Icono de Notificaciones.



Veamos las funcionalidades que nos ofrece esto.

En primer lugar, tenemos la campana de notificaciones, la cual varía su ícono en función de si el usuario tiene o no nuevas notificaciones. Podemos ver esa “exclamación” en color naranja si tenemos nuevas notificaciones de cualquier tipo.

Por otro lado, si pulsamos sobre la imagen del usuario, podremos ver el siguiente desplegable que nos permitirá tanto ir a los detalles del usuario, como cerrar la sesión.

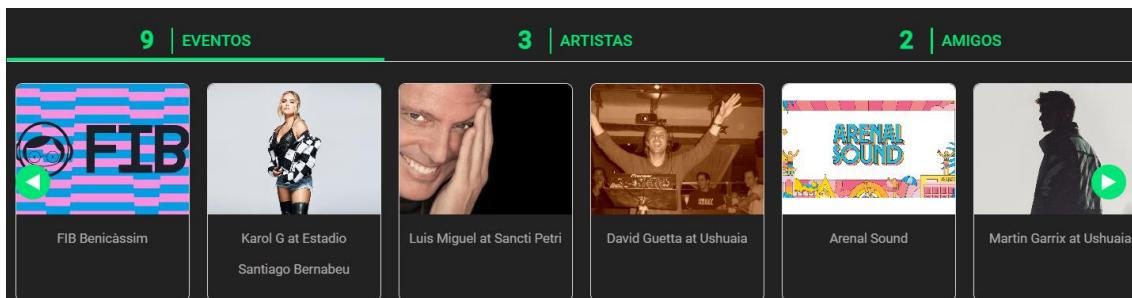


Si cerramos la sesión volveremos al componente “**Login**” para iniciar sesión con una nueva cuenta. En caso de dirigirnos a los detalles, podremos acceder a los detalles del usuario logeado, dónde podremos ver distinta información al respecto que vamos a desglosar.

En la parte superior, tenemos la información básica del usuario, como su “**Nick**”, correo electrónico, imagen de perfil, un check para activar o desactivas las notificaciones, y el botón de “**Editar Perfil**”, que veremos más adelante.



Debajo de estos detalles del usuario, podremos ver tres pestañas en las que se pueden ver los “**Eventos**” y “**Artistas**” seguidos por el usuario logeado, así como los “**Amigos**” del mismo.



Estas pestañas tienen además unas flechas a modo de scroll, en caso de tener más eventos o amigos de los que se pueden ver directamente en la ventana del navegador.

Es importante tener en cuenta que, en lo respectivo a eventos, podemos ir a los detalles de cada evento pulsando directamente en el evento desde esta pantalla.

En cuanto al botón de “**Editar Perfil**” del cual hablamos hace un momento, si pulsamos sobre él, nos llevará a una nueva pantalla con nuevas funcionalidades que explicaremos a continuación:

Lo primero que nos permitirá será cambiar los datos del usuario, los mismos que se eligieron al registrarse con excepción del correo electrónico y contraseña.

MODIFICAR USUARIO

Nombre
Live

Apellidos
Pulse

Nick
LivePulse

Ciudad
Oviedo

Provincia
Asturias

Teléfono
987654321

Notificaciones
Al activar esta opción, estarás aceptando recibir comunicaciones y noticias relevantes por email.

Gustos Musicales

Bluegrass
Blues
Christian
Classical
Country Music
EDM
Folk

Hip Hop & Rap
Indie
Jamband
Jazz
K-pop
Latin
Metal

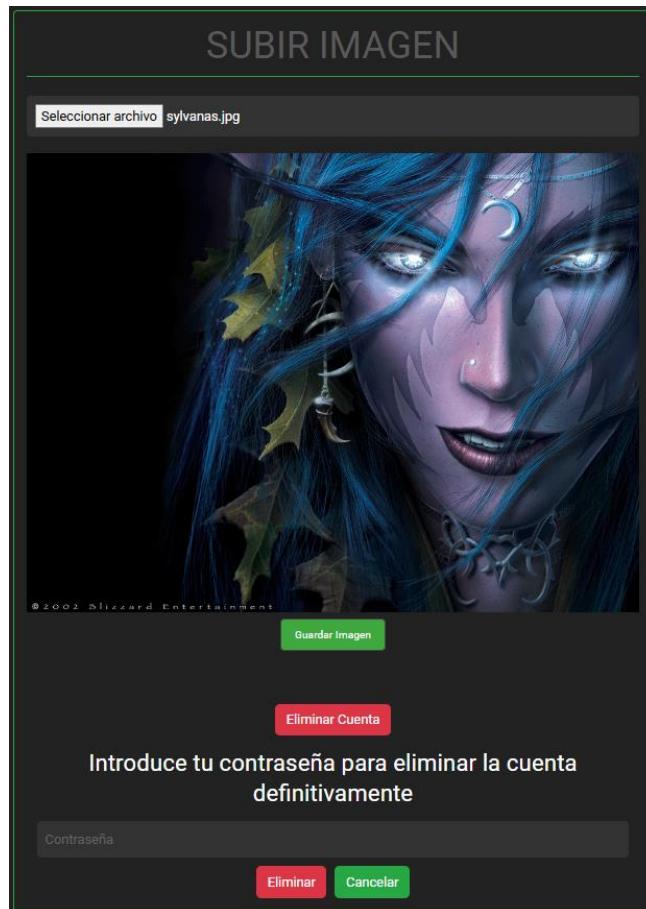
Pop
Punk
R&B / Soul
Reggae
Rock

Guardar Cambios

También nos dará la opción por separado de elegir una nueva contraseña, para lo que deberemos indicar la actual y la nueva dos veces, por temas de seguridad.



Por último, nos ofrece dos funcionalidades más. En primer lugar, la de subir una nueva imagen de perfil, que se podrá previsualizar y luego se adaptará a nuestro icono de imagen de perfil. Y también la opción de "**Eliminar Cuenta**", que requerirá de nuestra contraseña y varias confirmaciones para hacerlo.



Vamos ahora con el resto de apartados de la web, comenzando por la pestaña de “**Conciertos**”.

Esta pestaña nos muestra, por una parte, un filtro, que nos permite filtrar los conciertos en base a géneros musicales.

El grueso de la pantalla son los propios conciertos, que nos ofrecen información básica sobre cada concierto, y dónde podemos ver un botón de “**+Info**” que nos permitirá ver más detalles de cada uno de ellos.

También tenemos, en la parte inferior, una serie de páginas donde se podrán ver más eventos, que están ordenados de forma cronológica, de más próximos a más lejanos en el tiempo.

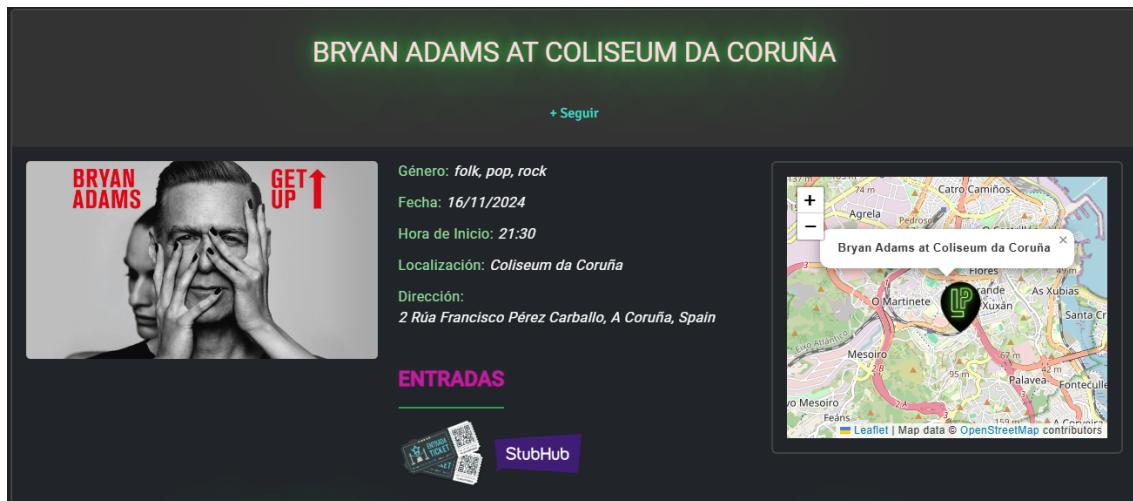
Otro detalle importante a tener en cuenta es el botón “**+Seguir**” o “**-Siguiendo**”, que nos permite seguir o dejar de seguir el concierto, y que solo aparecerá en caso de haber iniciado sesión en la aplicación.

Si pulsamos sobre el botón “**+Info**” podremos ver más detalles acerca del festival, además de otras funcionalidades, que vamos a ir explicando poco a poco.

En primer lugar, comenzando por la parte superior, tendremos el título del concierto, con datos básicos como género, fecha, hora, dirección, etc....

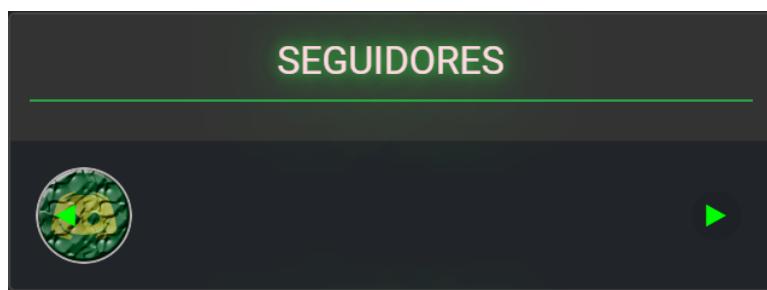
También tendremos de nuevo el botón “**+Seguir**” o “**-Siguiendo**” que nos permite, al igual que en la pantalla anterior, seguir o dejar de seguir el evento.

También tendremos la opción de dirigirnos a webs de compra de entradas, y de ver la ubicación en un mapa.



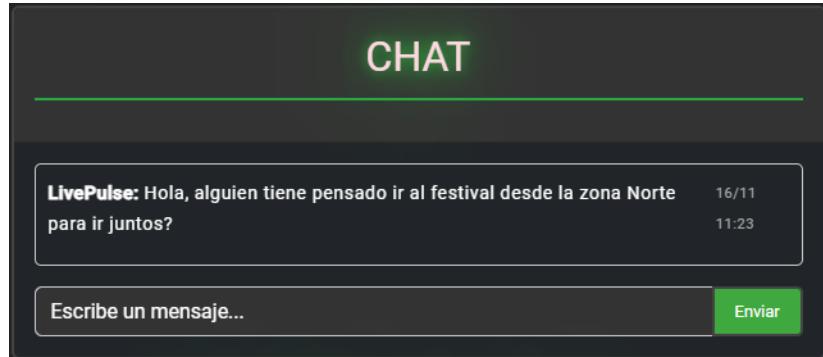
En la parte inferior, tenemos varias funcionalidades que van a depender de si iniciamos sesión o no.

En primer lugar, podemos ver los “**Seguidores**” del evento, es decir, otros usuarios, al igual que nosotros mismos, que han decidido seguir ese evento.

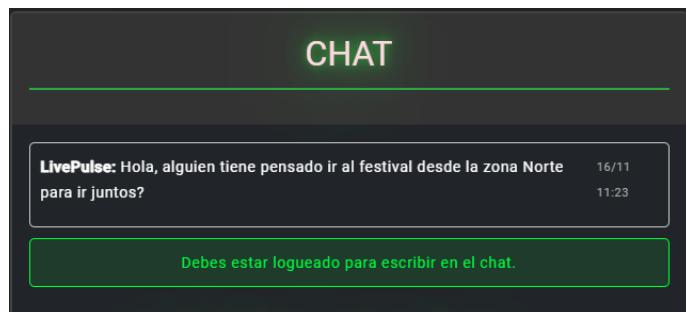


Al pulsar sobre cualquier de esos usuarios, podremos ver algunos detalles del mismo, que luego veremos.

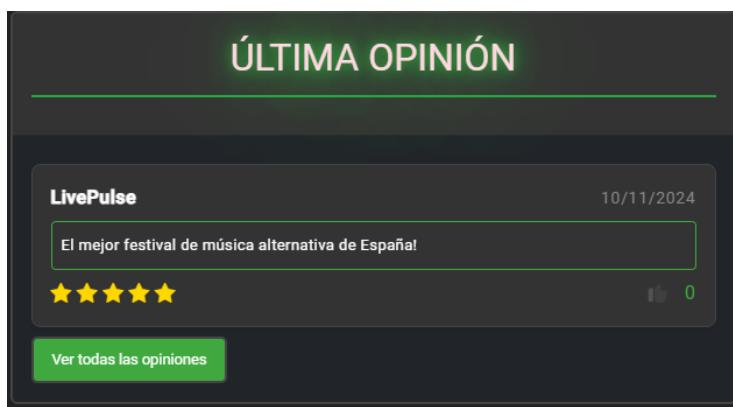
Al lado de los seguidores, tendremos el apartado del “**Chat**”, en el que solo pueden interactuar los usuarios registrados, y dónde podremos ver los mensajes de todos los usuarios. Cada concierto tiene su propio chat independiente del resto.



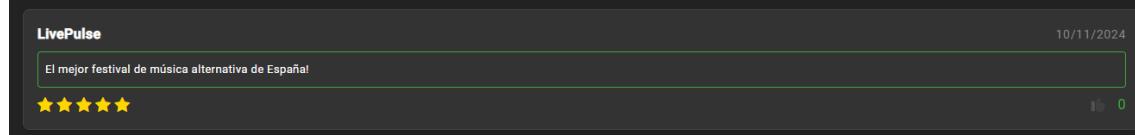
En caso de no estar logeados, nos aparecerá de la siguiente forma:



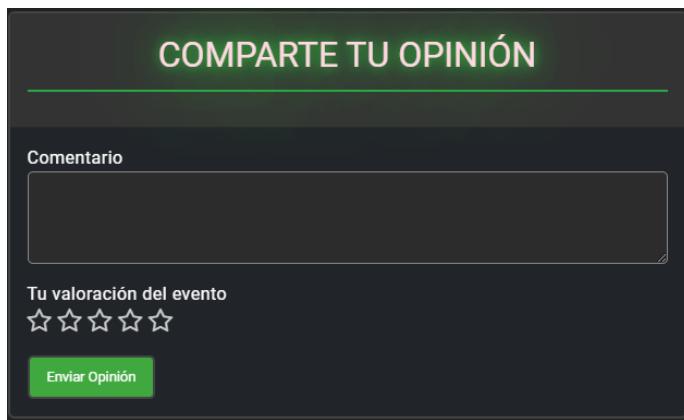
El siguiente apartado que tenemos es el de “**Última Opinión**”, dónde podemos ver la última opinión respecto al concierto, y el botón de “**Ver todas las opiniones**” que nos permite también ver las anteriores. Cada opinión también tiene la opción de recibir “Likes” por parte de otros usuarios. Vamos a ilustrarlo en las siguientes capturas:



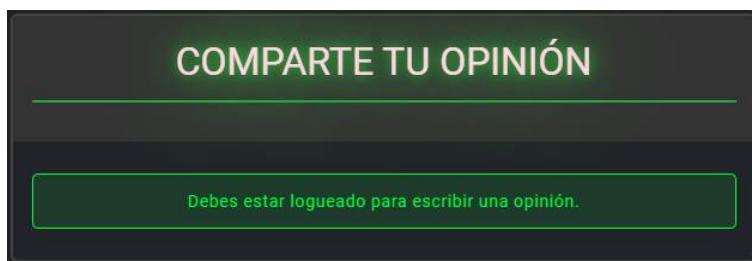
Opiniones sobre "Viña Rock"



El último de los apartados en esta pantalla es el de “**Comparte tu Opinión**”. Que nos permite añadir un comentario u opinión sobre el concierto, unido a una valoración del evento que va de 1 a 5 estrellas.



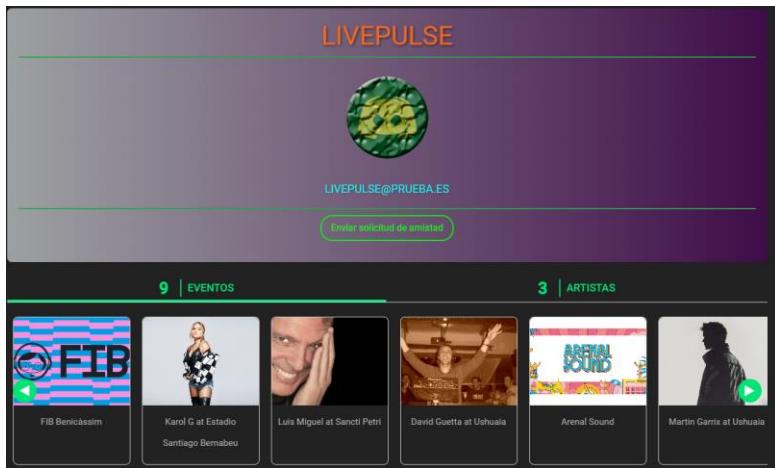
Esta es otra de las funcionalidades reservada a usuarios logeados, por lo que, en caso de no estarlo, veremos lo siguiente, al igual que ocurría en el “Chat”.



Si cambiamos de apartado, en este caso a “**Festivales**”, la funcionalidad y opciones son totalmente iguales, con un mínimo distintivo, y es que debajo de la imagen del Festival, tendremos la opción de acceder a sus distintas redes sociales, como podremos ver en la siguiente imagen:

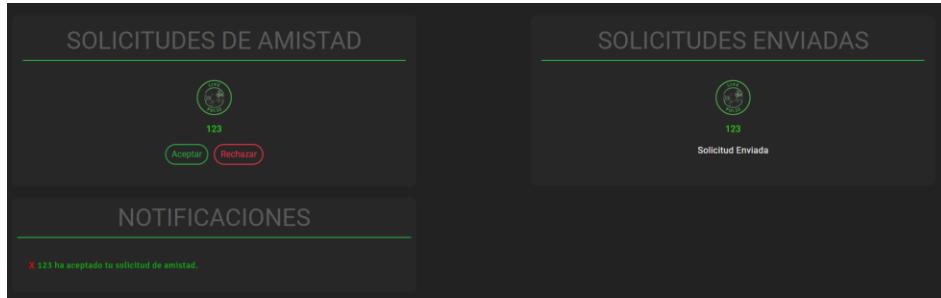


Hace un momento comentábamos que podemos pulsar sobre el ícono de cualquiera de los seguidores de un concierto o festival para ver algunos datos sobre él. Esto nos llevará a una nueva pantalla en la que podremos ver algo similar a lo que vemos en nuestros detalles de usuario, con algunas modificaciones.



Como se puede ver en la captura, podremos ver que eventos y artistas sigue, pero no los amigos que tiene. Además, podremos ver algo de información básica sobre él, como su nombre de usuario, imagen de perfil y correo electrónico, y nos permitirá enviarle una solicitud de amistad, que el luego podrá aceptar o rechazar, y que veremos a continuación.

Cuando un usuario recibe una solicitud de amistad, el ícono de notificaciones que vimos al inicio de este manual, se “activará”, y si pulsamos sobre ese ícono de notificaciones, podremos ver algo como lo siguiente:



Este apartado nos permite tres cosas:

En primer lugar, podemos aceptar o rechazar solicitudes de amistad de otros usuarios.

En segundo lugar, nos permite ver las solicitudes de amistad que tenemos enviadas y pendientes de aceptación o cancelación por parte de otro usuario.

Por último, podemos ver las notificaciones que nos envía la aplicación, por ejemplo, en caso de que un usuario haya aceptado nuestra solicitud de amistad.

Estas notificaciones tienen un ícono "X" que permite eliminarlas, y mientras no las hayamos eliminado todas, el ícono de notificaciones siempre mantendrá el signo de exclamación que indica que tenemos notificaciones.

Ya podemos pasar al siguiente apartado, que se trata del apartado de “**Artistas**”.

GÉNERO

- Bluegrass
- Blues
- Christian
- Classical
- Country Music
- EDM
- Folk
- Hip Hop & Rap
- Indie
- Jamband
- Jazz
- K-pop
- Latin
- Metal
- Pop
- Punk
- R&B / Soul
- Reggae
- Rock



BILLY STRINGS
24 Conciertos
- Siguiendo



OLD CROW MEDICINE SHOW
9 Conciertos
+ Seguir



TRAMPLED BY TURTLES
17 Conciertos
- Siguiendo



THE STEELDRIVERS
8 Conciertos
- Siguiendo



ALISON KRAUSS
0 Conciertos
+ Seguir



JERRY GARCIA
0 Conciertos
+ Seguir



THE DEVIL MAKES THREE
17 Conciertos
+ Seguir



WATCHHOUSE
2 Conciertos
+ Seguir



JUDAH & THE LION
2 Conciertos
+ Seguir



ERIC JOHNSON
24 Conciertos
+ Seguir



THE STRING CHEESE INCIDENT
18 Conciertos
+ Seguir



PUNCH BROTHERS
0 Conciertos
+ Seguir



GREENSKY BLUEGRASS
24 Conciertos
+ Seguir



NICKEL CREEK
15 Conciertos
+ Seguir



TOMMY EMMANUEL
60 Conciertos
+ Seguir

Anterior 1 2 3 Siguiente

Este apartado nos permite ver todos los artistas disponibles y filtrados por género, además de ello, podemos destacar dos funcionalidades.

En primer lugar, el botón de “**+Seguir**” o “**-Siguiendo**” que funciona igual que en los eventos vistos anteriormente, y por otro lado, al pulsar sobre la imagen de cualquiera de los artistas, nos permite acceder tanto a su “**Spotify**” como a su “**MusicBrainz**”.



La última pestaña de la web es la de “**Streams**”, la cual sigue la tónica de las anteriores en cuanto a filtrado por género, pero tiene alguna peculiaridad.

The screenshot shows a sidebar titled "GÉNERO" on the left, listing various music genres with checkboxes: Bluegrass, Blues, Christian, Classical, Country Music, EDM, Folk, Hip Hop & Rap, Indie, Jamband, Jazz, K-pop, Latin, Metal, Pop, Punk, R&B / Soul, Reggae, Rock, and Otros. To the right is a grid of six stream cards, each featuring a microphone and cables background image. The cards are arranged in two rows of three. Each card includes the stream name, date, location, a red circular button, a "Comprar Entradas" button, and a "+ Seguir" button. The cards are: "The Collective Stream" (2024-11-16, Stream Mundial), "Danika & The Jeb Stream" (2024-11-16, Stream Mundial), "Noelle Hannibal Stream" (2024-11-16, Stream Mundial), "Caylia Chaiken Stream" (2024-11-16, Stream Mundial), "Heather Romani Clare Angelini Stream" (2024-11-16, Stream Mundial), and "Izzy Stream" (2024-11-16, Stream Mundial). At the bottom of the grid is a navigation bar with "Anterior" and "Siguiente" buttons, and a page number "1" in a green box.

La página de Streams nos permite seguir y dejar de seguir streams, nos muestra información básica sobre ellos, y tiene dos casuísticas.

En el caso de que el Stream sea gratuito, aparecerá un círculo verde indicándolo, y en caso de ser de pago, aparecerá el círculo rojo junto al enlace para la compra de entradas.

Para completar nuestro manual, señalaremos dos puntos importantes situados en la parte superior e inferior de la web.

En la zona superior tenemos el logo de la aplicación junto a un buscado de eventos, que incluye tanto los conciertos como los festivales.

Si pulsamos sobre el logo, nos llevará de nuevo a la página principal.

The screenshot shows the top navigation bar of the website. On the left is a green heart icon with a white ECG line. Next to it is a search bar containing the text "Primavera". Below the search bar is a button with a globe icon and the text "Primavera Sound Barcelona - Festival - 6/5/25".

Por última, la parte inferior de la web nos permite acceder a recursos como enlaces útiles, redes sociales, etc...

The screenshot shows the footer of the website. It is divided into four main sections: "SOBRE NOSOTROS" (About Us) with links to the school's website and a "Más Información" button; "ENLACES ÚTILES" (Useful Links) with links to Contacto, FAQ, and Blog; "REDES SOCIALES" (Social Networks) with icons for Facebook, Twitter, and Instagram; and "LEGAL" (Legal) with links to Términos y Condiciones (Terms and Conditions) and Política de Privacidad (Privacy Policy). At the bottom center is a small copyright notice: "© 2024 Live Pulse. Todos los derechos reservados."

Capítulo 9. Conclusiones y ampliaciones

9.1 Conclusiones

A lo largo del desarrollo, hemos elaborado una aplicación funcional que permite gestionar eventos y conciertos de manera dinámica, ofreciendo características como la visualización de eventos, el seguimiento de favoritos y la interacción con datos de usuarios. La aplicación cumple con los requisitos esenciales definidos al inicio del proyecto, demostrando un comportamiento estable y ofreciendo una experiencia de usuario básica pero efectiva.

Hemos logrado implementar funciones importantes como:

- **Carga y paginación de eventos**, optimizando el tiempo de respuesta al usuario.
- **Gestión de eventos seguidos**, con una integración adecuada entre cliente y servidor.
- **Diseño interactivo**, que permite a usuarios registrados y no registrados explorar y utilizar la plataforma.

Aunque no se han materializado todas las ideas y mejoras previstas, el sistema construido es funcional y tiene potencial para escalar y ser refinado en el futuro.

Resultados y Expectativas

Los resultados obtenidos son, en general, satisfactorios. La aplicación cumple con los objetivos primarios, como permitir a los usuarios explorar eventos y administrar sus preferencias, interactuar con otros usuarios y amigos, etc. Sin embargo, en algunas áreas clave, como la optimización de la carga de eventos, las notificaciones de eventos y la experiencia de usuario avanzada, los resultados podrían mejorarse con más tiempo y recursos.

En términos de expectativas:

- Se ha cumplido la funcionalidad esencial esperada.
- Algunas funcionalidades avanzadas y optimizaciones (por ejemplo, carga más eficiente de eventos e implementación de ideas sugeridas durante el desarrollo) quedaron fuera del alcance por limitaciones de tiempo o conocimientos técnicos.

Elección de Opciones y Justificación

Cada decisión tomada en el desarrollo del sistema ha buscado equilibrar funcionalidad, simplicidad y viabilidad técnica. A continuación, se resumen las principales elecciones y su justificación:

1. Paginación y Gestión de Datos de la API

- La implementación de paginación para la carga de eventos ha sido clave para reducir el tiempo de respuesta y evitar cargar grandes volúmenes de datos simultáneamente. Aunque la solución inicial es funcional, con más tiempo podríamos explorar métodos más avanzados, como paginación infinita o precarga basada en predicciones.
- La decisión de centralizar la lógica de manipulación de datos en servicios fue acertada para mantener el código modular y manejable.

2. Gestión de Usuarios

- La separación de funcionalidades para usuarios registrados y no registrados ha permitido mantener una experiencia consistente para ambos casos.

3. Estructura del Proyecto

- La elección de tecnologías como Angular ha permitido un desarrollo organizado y modular.

4. Interfaz de Usuario

- Se ha optado por un diseño simple, enfocado en la funcionalidad, que ha permitido priorizar el desarrollo de la lógica del sistema. En el futuro, podrían integrarse animaciones, transiciones y un diseño más atractivo para enriquecer la experiencia de usuario.

Limitaciones y Mejoras Futuras

Durante el desarrollo surgieron ideas de mejora que podrían implementarse con más tiempo y conocimientos:

- **Carga de datos más eficiente:** Explorar técnicas como Lazy Loading o caching para optimizar la interacción con la API.
- **Optimización de imágenes:** Implementar carga perezosa y formatos comprimidos para mejorar los tiempos de carga.
- **Experiencia del usuario:** Introducir notificaciones sobre los eventos seguidos, mensajes de error más claros, y una navegación más intuitiva.
- **Seguridad y robustez:** Validar con mayor rigor los datos del lado del cliente y del servidor.

Reflexión Final

Este proyecto ha sido una experiencia enriquecedora que ha permitido integrar múltiples aspectos del desarrollo web, desde la interacción con APIs hasta la gestión de la lógica de la aplicación en el cliente. Aunque el resultado es funcional y cumple con los objetivos iniciales, queda un camino para llevar la aplicación a su máximo potencial.

El proceso ha dejado claro que, con más tiempo y conocimientos, sería posible implementar soluciones más optimizadas y responder mejor a las necesidades de los usuarios. Este proyecto sirve como un excelente punto de partida para continuar aprendiendo y mejorando en el desarrollo de aplicaciones web.

9.2 Ampliaciones

A continuación, se describen diversas ampliaciones que podrían implementarse en el sistema, mencionando su naturaleza, ventajas y razones por las que no se han incluido en la versión actual.

1. Paginación Avanzada e Interacción con la API

- **En qué consiste:** Implementar paginación infinita (infinite scroll) en lugar de una paginación tradicional con botones de navegación. Esto implicaría cargar más datos automáticamente al llegar al final de la lista visible.
 - **Cómo ampliará el sistema:** Hará que la navegación sea más fluida y natural para los usuarios, eliminando clics adicionales.
 - **Ventajas:** Mejora la experiencia del usuario y permite mantener un diseño más limpio en la interfaz.
 - **Por qué no se incluyó:** Requiere una interacción más sofisticada con la API y una gestión precisa del estado de los datos, que excede el alcance del tiempo disponible.
-

2. Búsqueda y Filtrado Avanzados

- **En qué consiste:** Implementar un buscador con sugerencias en tiempo real (autocomplete) y filtros avanzados, como rango de fechas, ubicación exacta, precio y popularidad.
 - **Cómo ampliará el sistema:** Ofrecerá a los usuarios más opciones para personalizar su búsqueda y encontrar eventos que realmente se ajusten a sus preferencias.
 - **Ventajas:** Incrementa la funcionalidad y mejora la satisfacción del usuario al ofrecerle resultados más relevantes.
 - **Por qué no se incluyó:** El buscador es uno de los componentes que más me ha costado entender y desarrollar, aun teniendo en cuenta que se centra sólo en los eventos. Desarrollar un buscador avanzado requiere más conocimientos por mi parte.
-

3. Recomendaciones Personalizadas

- **En qué consiste:** Implementar un sistema de recomendación basado en el historial de eventos seguidos o visualizados por el usuario.
 - **Cómo ampliará el sistema:** Presentará eventos sugeridos que puedan interesar a los usuarios, aumentando su compromiso con la plataforma.
 - **Ventajas:** Mejora la experiencia del usuario y puede incrementar el uso de la aplicación.
 - **Por qué no se incluyó:** La implementación requiere técnicas avanzadas de análisis de datos y algoritmos que exceden los conocimientos y recursos disponibles actualmente.
-

4. Notificaciones en Tiempo Real

- **En qué consiste:** Incorporar notificaciones para avisar a los usuarios sobre cambios en los eventos que siguen (por ejemplo, cambio de fecha o ubicación) o para informarles sobre nuevos eventos similares a sus intereses.
 - **Cómo ampliará el sistema:** Mantendrá a los usuarios informados y conectados a la plataforma en todo momento.
 - **Ventajas:** Incrementa la retención de usuarios y mejora la percepción del sistema como proactivo y útil.
 - **Por qué no se incluyó:** Requiere consultas periódicas a la API, la cual, al ser gratuita, tiene un límite de peticiones diarias.
-

5. Diseño de Interfaz Mejorado

- **En qué consiste:** Rediseñar la interfaz con un enfoque en animaciones, transiciones suaves y un diseño visual más atractivo.
 - **Cómo ampliará el sistema:** Hará que la aplicación sea más atractiva y moderna, mejorando la experiencia del usuario.
 - **Ventajas:** Incrementa la percepción de calidad y profesionalismo de la aplicación.
 - **Por qué no se incluyó:** Tras aumentar mis conocimientos en Angular, Typescript y conocer Firebase, no pude dedicar tanto tiempo a Bootstrap o CSS, algo que queda pendiente porque me encantan.
-

6. Optimización de Carga de Imágenes

- **En qué consiste:** Implementar técnicas de carga diferida (lazy loading) y formatos optimizados, como WebP, para las imágenes de los eventos.
 - **Cómo ampliará el sistema:** Reducirá los tiempos de carga y el consumo de datos.
 - **Ventajas:** Mejora el rendimiento, especialmente en dispositivos móviles con conexiones limitadas.
 - **Por qué no se incluyó:** Aunque es una mejora significativa, no era esencial para la funcionalidad básica del sistema.
-

7. Estadísticas para los Usuarios

- **En qué consiste:** Proporcionar a los usuarios registrados estadísticas personalizadas, como el número de eventos seguidos, géneros más visualizados o localizaciones frecuentes.
 - **Cómo ampliará el sistema:** Ofrecerá información útil y atractiva para los usuarios, incrementando su compromiso con la plataforma.
 - **Ventajas:** Mejora la personalización y refuerza la conexión del usuario con la aplicación.
 - **Por qué no se incluyó:** Necesita un backend robusto para almacenar y procesar estadísticas, lo cual excede los recursos disponibles.
-

8. Internacionalización y Localización

- **En qué consiste:** Soportar múltiples idiomas y formatos locales para fechas, monedas y ubicaciones.
 - **Cómo ampliará el sistema:** Aumentará la accesibilidad para usuarios de diferentes regiones.
 - **Ventajas:** Incrementa el alcance global de la plataforma.
 - **Por qué no se incluyó:** Se priorizó la funcionalidad principal en un solo idioma para reducir la complejidad.
-

9. Integración con Redes Sociales

- **En qué consiste:** Permitir a los usuarios compartir eventos en redes sociales y registrarse o iniciar sesión a través de cuentas de plataformas como Google o Facebook.
 - **Cómo ampliará el sistema:** Fomentará la difusión de eventos y facilitará la incorporación de nuevos usuarios.
 - **Ventajas:** Incrementa la viralidad y reduce la fricción en el registro.
 - **Por qué no se incluyó:** Requiere configurar integraciones externas y manejar aspectos de seguridad adicionales.
-

Estas ideas ofrecen un panorama amplio de cómo el sistema puede evolucionar en el futuro para atender mejor las necesidades de los usuarios y aprovechar su potencial.