Manuel Garcia Muañache

Florida Universitària

Penya Fallera

Proyecto Final de Ciclo
DAM

Índice

1 RESUMEN DEL PROYECTO

Penya Fallera

Proyecto de conexión y gestión de experiencias falleras

Penya Fallera nace para reunir en un único espacio digital a fallas, falleros y visitantes, facilitando tanto la difusión de actividades como la interacción en tiempo real. La plataforma centraliza la publicación de eventos tradicionales (fecha, hora y ubicación) y ofrece chats colectivos asociados a cada acto o agrupación, fomentando la coordinación y la convivencia entre locales y foráneos. Al concentrar toda la programación y la conversación en un mismo sitio, se potencia la visibilidad de las comisiones falleras y se impulsa la participación y el patrocinio de las distintas iniciativas. Además, Penya Fallera ofrece espacios publicitarios para negocios locales, conectando comercios de proximidad con un público altamente comprometido y mejorando la sostenibilidad económica de la plataforma.

2 JUSTIFICACION Y OBJETIVOS DEL PROYECTO

Tras un exhaustivo estudio de mercado, se comprueba que, aunque existen varias aplicaciones dedicadas a las Fallas, ninguna ofrece un espacio donde los usuarios puedan conectarse entre sí y participar activamente en los distintos eventos a través de chats grupales. La idea del proyecto nace para cubrir esa necesidad, fomentando la interacción de la comunidad fallera y facilitando la difusión y comunicación instantánea en torno a cada actividad.

Análisis de soluciones actuales:

Fallas 2025:

Se centra en la geolocalización de las Fallas y en la consulta de eventos programados, pero no incluye funcionalidades de interacción social ni permite a los usuarios contar con un perfil personalizado.

Mundo Fallas:

Permite seguir comunicados oficiales y eventos de las Fallas de tu interés, aunque carece de herramientas para que los participantes puedan compartir contenido propio o entablar conversaciones sobre cada evento.

Falles.app:

Está orientada a la gestión interna de cada falla (control de consumiciones, organización de falleros, etc.), pero no facilita la comunicación ni la relación entre usuarios externos o aficionados.

Mis Fallas:

Ofrece una guía con ubicaciones y descripciones de algunas Fallas, si bien su enfoque es meramente informativo y no contempla ninguna dimensión colaborativa ni de conversación.

Objetivos Principales

Desarrollar una aplicación que combine:

- **Geolocalización y agenda de eventos**
- **Perfiles de usuario:** Personalización de intereses y fallas favoritas.
- **Chats grupales por evento:** Al seleccionar cualquier evento, el usuario accede a un chat en tiempo real donde podrá comentar, preguntar y compartir su experiencia con otros asistentes.

De este modo, la propuesta unifica información útil con comunicación instantánea, creando una experiencia participativa y social en torno a las Fallas.

3 DESARROLLO DEL PROYECTO

3.1 Análisis del mercado y modelo del negocio

Análisis del mercado

- **Competencia existente**
  - *Fallas 2025*: ofrece geolocalización y agenda de actos, pero carece de interacción social y perfiles de usuario.
  - *Mundo Fallas*: centraliza comunicados oficiales, sin posibilidad de compartir contenidos ni chats por evento.
  - *falles.app*: enfocado a la gestión interna de cada comisión (consumos, inventarios), sin espacios colaborativos para visitantes o aficionados.
  - *Mis Fallas*: guía informativa de ubicaciones y datos de Fallas, con un alcance meramente estático.

- **Oportunidad y demanda**
  - ✓ Cada año se instalan más de 700 comisiones en Valencia, movilizando a cientos de miles de asistentes locales y foráneos durante las semanas de Fallas.
  - ✓ El público joven (18–35 años) está acostumbrado a socializar y organizarse a través de apps móviles, y demanda soluciones que integren contenido en tiempo real y chat colectivo.
  - ✓ Los comercios de hostelería y ocio en los entornos falleros buscan canales de publicidad segmentada con retorno inmediato sobre un público muy localizado.

- **Hueco detectado**
  - × No existe una aplicación que aúne información de eventos, geolocalización y grupos de chat por acto o falla, lo que obliga a los usuarios a alternar entre varias plataformas (WhatsApp, Telegram, Instagram).
  - × Las comisiones carecen de un canal oficial propio para interactuar con sus visitantes más allá de los medios tradicionales y las redes sociales generales.

Modelo de negocio

- **Suscripción anual para comisiones falleras**

  Precio fijo por falla o casal, que incluye acceso para todos sus miembros a la gestión de eventos, estadísticas de asistencia y comunicaciones internas.

- **Promociones de negocios locales**

  Venta de espacios destacados (banners, ofertas especiales) dentro de la app, dirigidos a comercios de proximidad y servicios turísticos, con segmentación geográfica por acto o área.

Con esta combinación, Penya Fallera no solo cubre una necesidad clara de la comunidad fallera, sino que crea un ecosistema sostenible donde comisiones y patrocinadores colaboran para enriquecer la experiencia de todos los asistentes.

3.2 Valor añadido de la solución

Penya Fallera aporta un valor diferencial al integrar en una sola plataforma tanto la consulta de la programación como la interacción instantánea entre usuarios, evitando que aficionados y organizadores deban dispersarse entre múltiples aplicaciones o redes sociales (WhatsApp, Telegram, Instagram...).

Gracias al perfil personalizado, cada participante recibe recomendaciones y alertas ajustadas a sus preferencias, lo que maximiza su implicación (mejora a futuro).

Al habilitar chats grupales específicos para cada acto, se fomenta la colaboración y el intercambio de experiencias en tiempo real, estrechando lazos dentro de la comunidad fallera y facilitando la coordinación de planes, la resolución de dudas y la difusión de novedades al momento. De este modo, Penya Fallera no solo centraliza la información, sino que convierte cada evento en un punto de encuentro digital, enriqueciendo la tradición con dinamismo y cercanía.

3.3 Metodologías utilizadas

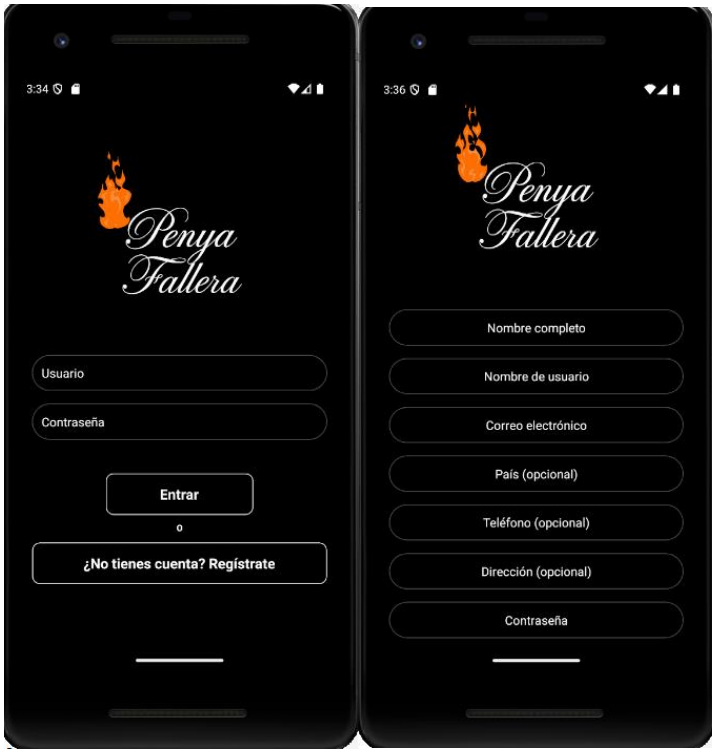
Para el desarrollo de Penya Fallera se ha optado por una metodología ágil, concretamente **Scrum**, con el fin de mantener reuniones periódicas con el tutor del proyecto y cumplir entregas quincenales que simulan los "sprints" propios de Scrum.

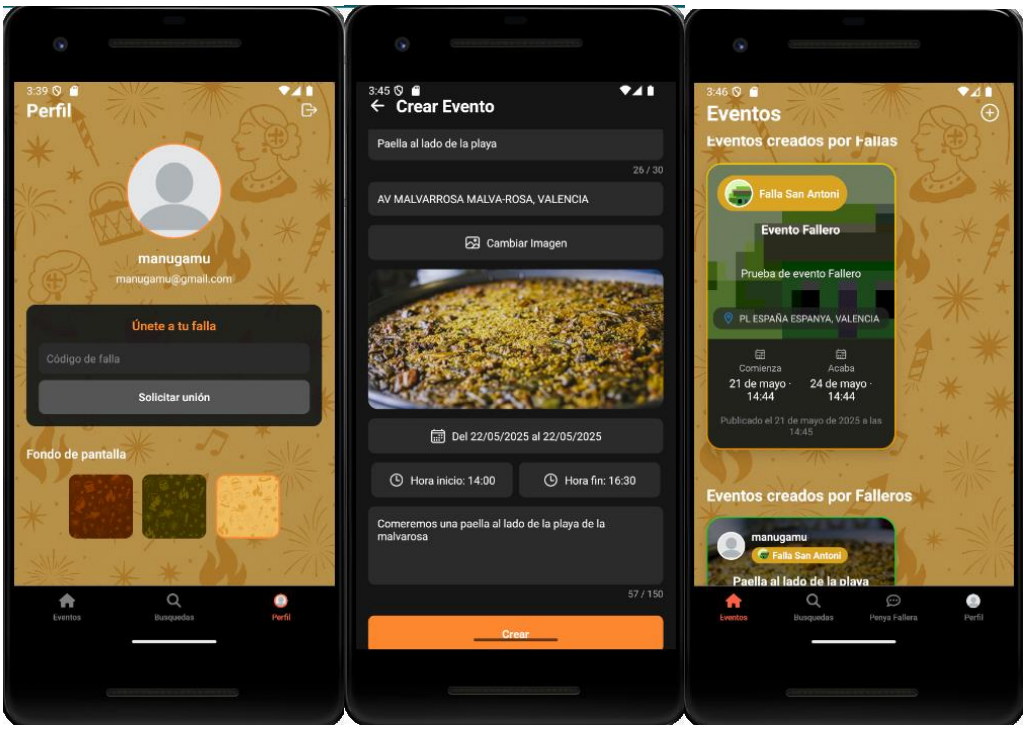
La gestión de las tareas se realiza a través de **Trello**, lo que permite priorizar tanto las funcionalidades críticas como aquellas que aportan un mayor valor al conjunto del proyecto. Gracias a este enfoque, se asegura una adaptación continua a los cambios y una mejora incremental a lo largo de todo el proceso de desarrollo.

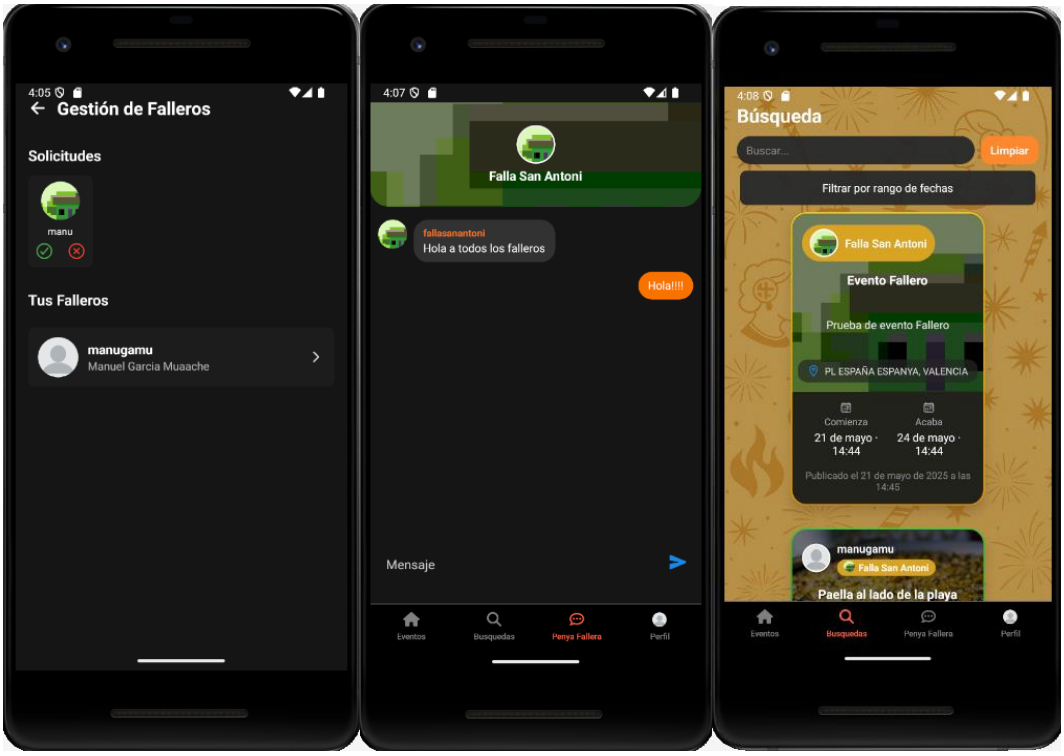
3.4 Descripción de los componentes de la aplicación

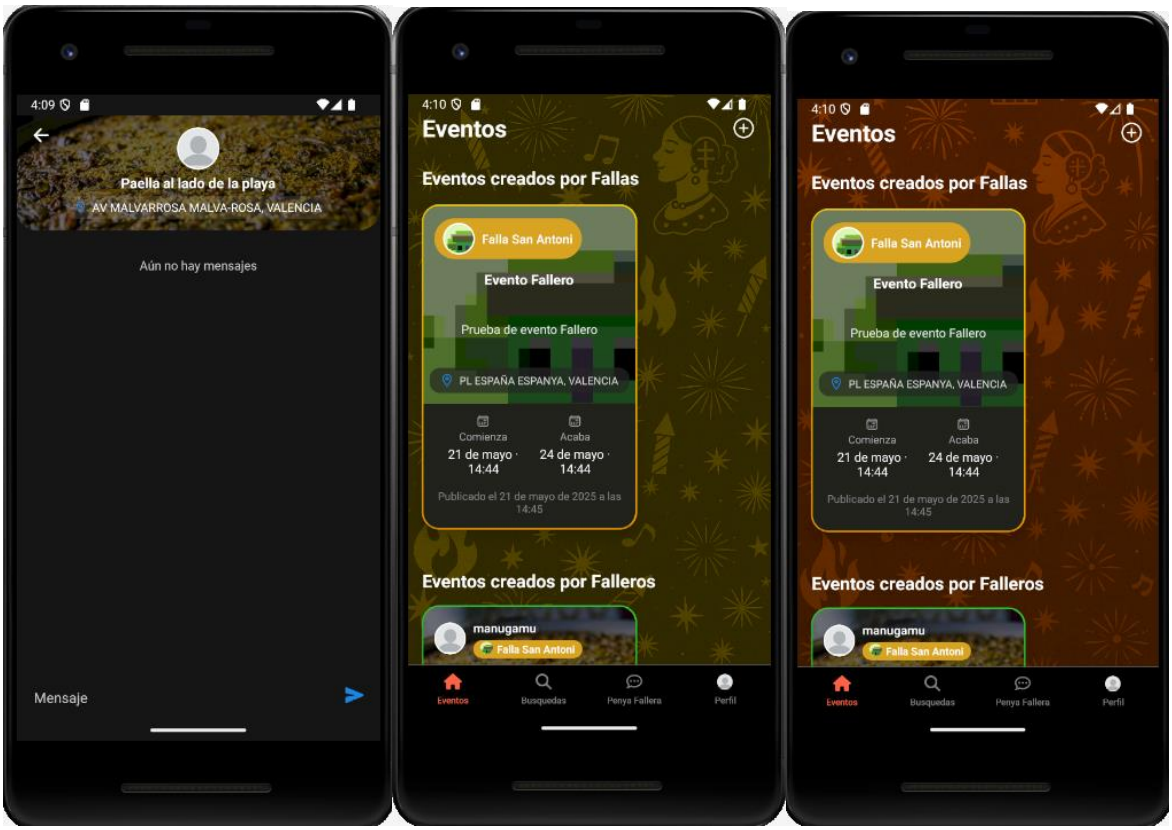
3.4.1 Visión general de la arquit*ect*ura

- **Cliente (App móvil React Native)**
  - Interfaz de usuario que consume la API REST principal y se conecta al servidor de chat vía WebSocket.




Perfil
manugamu
manugamu@gmail.com
Únete a tu falla
Código de falla
Solicitar unión
Fondo de pantalla
Crear Evento
Paella al lado de la playa
26 / 30
AV MALVARROSA MALVA-ROSA, VALENCIA
Cambiar Imagen
Del 22/05/2025 al 22/05/2025
Hora inicio: 14:00
Hora fin: 16:30
Comeremos una paella al lado de la playa de la malvarosa
57 / 150
Crear
Eventos
Eventos creados por Fallas
Falla San Antoni
Evento Fallero
Prueba de evento Fallero
PL. ESPAÑA ESPANYA, VALENCIA
Comienza
21 de mayo · 14:44
Acaba
24 de mayo · 14:44
Publicado el 21 de mayo de 2025 a las 14:45
Eventos creados por Falleros
manugamu
Falla San Antoni
Paella al lado de la playa


Gestión de Falleros
Solicitudes
manu
Tus Falleros
manugamu
Manuel Garcia Muaache
Falla San Antoni
fallasanantoni
Hola a todos los falleros
Hola!!!!
Mensaje
Búsqueda
Buscar...
Limpiar
Filtrar por rango de fechas
Falla San Antoni
Evento Fallero
Prueba de evento Fallero
PL. ESPAÑA ESPANYA, VALENCIA
Comienza
21 de mayo · 14:44
Acaba
24 de mayo · 14:44
Publicado el 21 de mayo de 2025 a las 14:45
manugamu
Falla San Antoni
Paella al lado de la playa



- **Backend principal (Java + Spring Boot)**
    - Se encarga de la gestión de usuarios, eventos, fallas, perfiles, autenticación/seguridad, y sirve todas las rutas REST.
- **Microservicio de mensajería (Node.js + Express + WebSocket)**
    - Servidor independiente dedicado exclusivamente al chat en tiempo real y al almacenamiento histórico de mensajes.
- **Bases de datos y servicios externos**
    - **MongoDB**: persiste usuarios, eventos, fallas y también mensajes del chat (para ambos backends).
    - **Redis**: almacena en memoria la “blacklist” de refresh tokens para gestionar logout e invalidaciones.
    - **Cloudinary**: CDN y transformación de imágenes (avatares de usuario, fotos de eventos).

Esta separación en dos despliegues (Java y Node.js) permite escalar y evolucionar el módulo de mensajería de forma independiente, optimizando el uso de recursos y facilitando el mantenimiento.

3.4.2 Backend principal: Java + Spring Boot

2.1. Arranque y estructura de capas

- **ProyectoFinalApplication.java:** punto de entrada con @SpringBootApplication, arranca el contexto, registra beans y configura el servidor embebido (Tomcat).
- **Patrón en capas**:
  - ✓ **Controladores (@RestController)**: exponen endpoints HTTP y validan entrada.
  - ✓ **Servicios (@Service)**: lógica de negocio (registro, manejo de eventos, orquestación de tokens, llamadas a servicios externos).
  - ✓ **Repositorios (@Repository)**: extienden MongoRepository<T,ID>, gestionan CRUD sobre MongoDB.

2.2. Seguridad y JWT

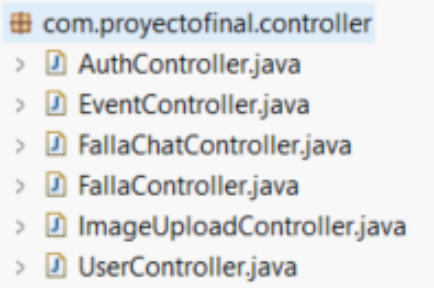
- **Spring Security** configurado en SecurityConfig.java: define rutas públicas (login, registro, subida de avatar) y rutas protegidas.
- **Filtros**:
  - ✓ **JwtAuthFilter:** extrae y valida el access token en cada petición.
  - ✓ **JwtUtil:** genera y verifica firmas de tokens.
- **Refresh tokens y blacklist**: tras autenticar, se emiten un access token (corta validez) y un refresh token (más largo). En logout o ante sospecha, el refresh token se inserta en Redis con fecha de expiración; cualquier intento de usarlo es rechazado.

2.3. Almacenamiento de medios

- **CloudinaryConfig.java:** cliente configurado con credenciales.
- **ImageUploadController.java:** recibe el fichero, lo sube a Cloudinary y guarda la URL resultante en MongoDB.

2.4. Endpoints clave

- **AuthController:** login, registro, refresh token, logout.
- **UserController**: consulta y edición de perfil, subida de avatar.
- **EventController:** CRUD de eventos, consulta de agenda, marcaje de favoritos.
- **FallaController:** administración de fallas (agrupaciones), sus datos y miembros.
- **FallaChatController:** (opcional) gestión de metadatos de salas de chat, pero sin la lógica real de mensajería.



3.4.3 Microservicio de mensajería: Node.js + Express + WebSocket

3.1. Arranque y dependencias

- **ChatServer.js:** inicializa un servidor HTTP con Express y un servidor WebSocket con la librería ws. Conecta a MongoDB (mongoose.connect) para persistir los mensajes.

3.2. Modelo de datos

```
const mongoose = require('mongoose');

const messageSchema = new mongoose.Schema({
  eventoId: { type: String, required: true },
  content: { type: String, required: true },
  createdAt: { type: String, required: true },
  user: { type: String, required: true },
  userId: { type: String, required: true },
  profileImageUrl: { type: String, default: '' }
});

module.exports = mongoose.model('Message', messageSchema);
```

Cada mensaje lleva metadatos para renderizar autor, hora y vínculo con el evento.

3.3. Flujo de WebSocket

- **Conexión**: al conectarse cliente, se le asigna ws.eventoId = null.
- **Unirse a sala** (type: 'join'): el cliente envía { type:'join', eventoId, user } y el servidor almacena ws.eventoId.
- **Enviar mensaje** (type: 'chat'):
  - ✓ El servidor valida campos obligatorios (eventoId, content, user, etc.).
  - ✓ Persiste en MongoDB con Message.create(parsed).
  - ✓ Reenvía el mensaje JSON a todos los clientes con client.eventoId === eventoId.
- **Cerrar conexión**: índice de clientes actualizados, logging en consola.

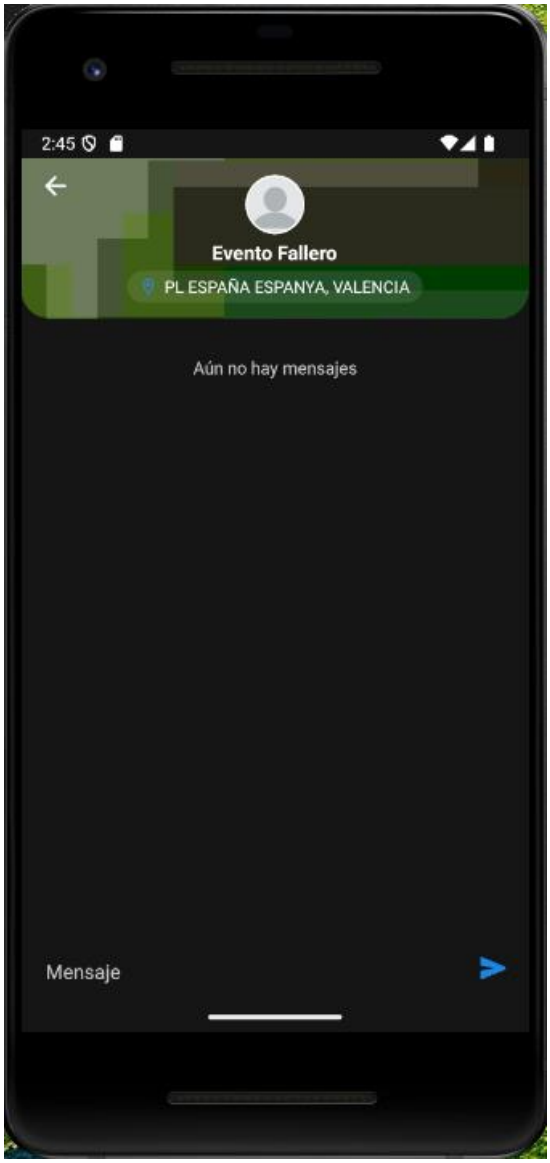
```
if (parsed.type === 'join') {
  ws.eventoId = parsed.eventoId;
  console.log(`🟢 Cliente unido a evento ${ws.eventoId} como ${parsed.user}`);
  return;
}

if (parsed.type === 'chat') {
  const { eventoId, user, content, createdAt, userId } = parsed;

  if (!eventoId || !user || !content?.trim() || !createdAt || !userId) {
    console.warn('❌ Mensaje incompleto o vacío, ignorado');
    return;
  }
```

```
  ws.on('close', () => {
    console.log(`🔴 Cliente desconectado del evento ${ws.eventoId || 'desconocido'}`);
  });
});
```

3.4. Endpoints HTTP auxiliares

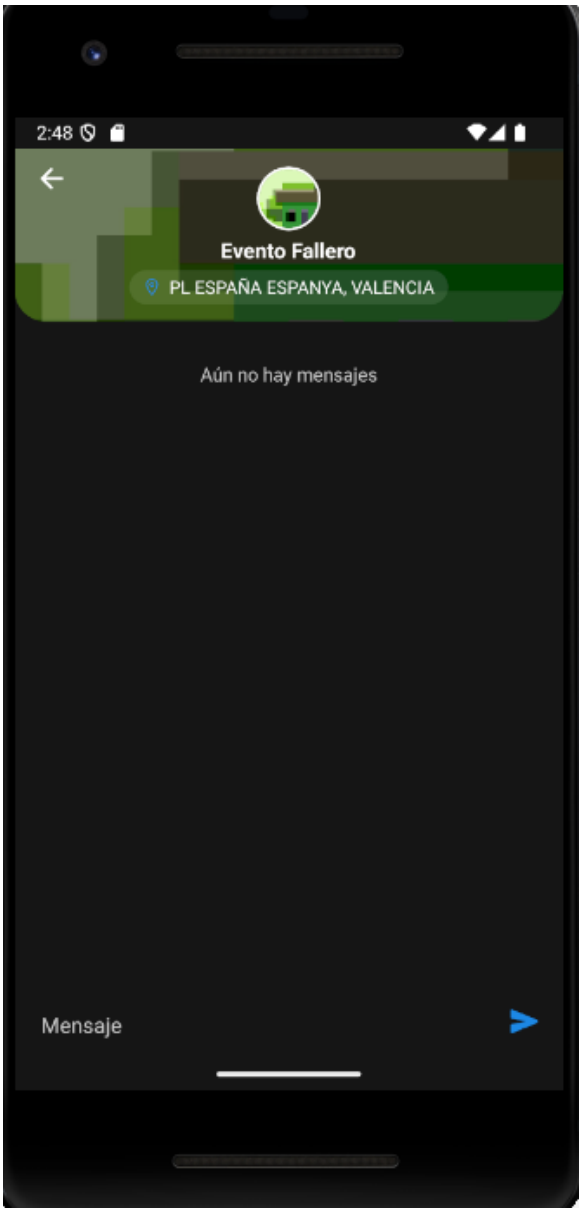
- **GET** /mensajes/:eventoId: recupera historial ordenado de un chat concreto.
- **PUT** /api/chat/update-profile-image: al cambiar avatar en el perfil Java, el frontend notifica aquí el userId y la nueva URL; el servidor actualiza todos los mensajes antiguos y emite un broadcast para refrescar imágenes en clientes conectados.
    - **POST** /api/chat/create-room: reserva una sala “privada” para una falla

A continuación, podemos ver un chat creado por un usuario falla, como se aprecia en la imagen, el usuario no tiene todavía imagen:



Cambiamos su imagen de perfil y volvemos a comprobar el evento creado:

La imagen del creador del evento cambia automáticamente en el chat grupal:



3.4.4 Despliegue y escalabilidad

- **Ejecución en Docker**:
  Toda la plataforma (backend Java, microservicio Node.js y Redis) se despliega como contenedores Docker en un servidor dedicado de Hetzner, garantizando entornos idénticos en desarrollo y producción.
- **Servicios compartidos**:
  - ✓ **MongoDB**: clúster replicado para garantizar persistencia y tolerancia a fallos.
  - ✓ **Redis**: modo clúster o Sentinel para la blacklist de tokens.
  - ✓ **Cloudinary**: servicio SaaS de CDN global para entrega y transformación de imágenes.

Beneficios de esta aproximación

- **Separación de responsabilidades**: cada servicio en su propio contenedor.
- **Escalado independiente**: se pueden aumentar réplicas de chat sin tocar el monolito.
- **Despliegues aislados**: actualizaciones y reinicios por servicio sin afectar al resto.
- **Portabilidad y consistencia**: Docker garantiza entornos homogéneos y facilita la gestión de la infraestructura.

3.5 Dificultades encontradas

Durante el desarrollo, pasar de un entorno de pruebas a uno real me obligó a ser muy cuidadoso con cada paso: un error podía significar deshacer gran parte del trabajo, algo especialmente complicado con plazos tan ajustados.

Configurar todo en un servidor remoto (Hetzner) fue otro reto: nunca había montado un proyecto en un servidor propio, así que aprender a dejarlo operando correctamente me llevó su tiempo.

Crear el chat en vivo exigió muchas pruebas y ajustes para que los mensajes llegaran siempre donde y como debían y el sistema se recuperara solo si había cortes de conexión.

Además, las ganas de añadir nuevas funcionalidades chocaban constantemente con la necesidad de cumplir la fecha de entrega, por lo que tuve que centrarme en entregar primero una versión mínima funcional.

Por último, dedicar tiempo a que la aplicación luciera atractiva consumió varias jornadas de trabajo, reduciendo el tiempo disponible para otras mejoras.

LECCION APRENDIDA: marcar desde el principio un objetivo claro para la versión mínima, y luego balancear con calma funcionalidades y diseño, es clave para terminar a tiempo sin descuidar la calidad.

3.6 Resultados obtenidos

A pesar de la amplitud de ideas en el roadmap, se ha entregado un **MVP robusto y visualmente atractivo** que cumple con los objetivos esenciales:

- **Publicación y geolocalización de eventos**: agenda interactiva con todos los actos y acceso al mapa.
- **Perfiles personalizados**: cada usuario puede configurar sus intereses y fallas favoritas.
- **Chats grupales por evento y falla**: comunicación en tiempo real integrada en cada actividad.
- **Gestión básica de usuarios y seguridad**: registro, login, refresh de tokens y blacklist en Redis.
- **Almacenamiento de medios**: imágenes subidas a Cloudinary con entrega optimizada.

4 LINEAS FUTURAS DE TRABAJO

1. **Login con cuenta de Google**
2. **Eventos generales y posts sociales**
3. **Botón “Asistir” en la ficha de evento**
4. **Indicador de usuarios activos en el chat**
5. **Módulo de promociones para negocios locales**
6. **Filtrado por proximidad (eventos cerca de ti)**
7. **Mejoras estéticas y de usabilidad**

Esta primera version sienta unas bases sólidas sobre las que poder iterar rápidamente, incorporando nuevas características sin comprometer la arquitectura ni el rendimiento de la aplicación.

5 CONCLUSIONES

Penya Fallera ha cumplido con el objetivo de ofrecer un espacio unificado en el que comisiones, falleros y visitantes pueden descubrir y organizarse en torno a las Fallas, al tiempo que conversan en chats grupales específicos para cada acto o agrupación.

La entrega de un MVP sólido y atractivo ha demostrado que es posible combinar geolocalización, agenda de eventos, perfiles personalizados y mensajería en tiempo real bajo una misma interfaz móvil.

El uso de una arquitectura en contenedores (Docker) y la separación de responsabilidades entre el backend Java/Spring Boot y el microservicio de chat en Node.js han facilitado el despliegue en un VPS de Hetzner, así como el escalado independiente de cada componente. Además, la adopción de Scrum y Trello ha garantizado entregas periódicas y una adaptación continua a los requisitos que han ido surgiendo durante el desarrollo.

Durante el proyecto he afrontado retos como la puesta en producción en un entorno desconocido, la configuración de WebSockets para un chat estable y la restricción de tiempo frente a la ambición de implementar nuevas funcionalidades. Estas experiencias me han enseñado la importancia de definir claramente el MVP desde el inicio y de equilibrar diseño, alcance y calidad.

En definitiva, Penya Fallera sienta las bases de una plataforma extensible y escalable, con potencial para crecer añadiendo integraciones sociales, analíticas de uso y nuevos modelos de negocio. La versión actual valida la viabilidad técnica y de mercado, y abre el camino a futuras iteraciones que continuarán enriqueciendo la experiencia fallera digital.

6.0 BIBLIOGRAFÍA

- Cloudinary. (s. f.). *Cloudinary Documentation*. Recuperado de https://cloudinary.com/documentation
- Docker. (s. f.). *Docker Documentation*. Recuperado de https://docs.docker.com/
- Express. (s. f.). *Express API Reference*. Recuperado de https://expressjs.com/
- Java SE. (s. f.). *Java Platform, Standard Edition Documentation*. Oracle. Recuperado de https://docs.oracle.com/javase/
- Mongoose. (s. f.). *Mongoose Documentation*. Recuperado de https://mongoosejs.com/
- MongoDB. (s. f.). *MongoDB Manual*. Recuperado de https://docs.mongodb.com/
- Node.js. (s. f.). *Node.js Documentation*. Recuperado de https://nodejs.org/
- Redis. (s. f.). *Redis Documentation*. Recuperado de https://redis.io/
- React Native. (s. f.). *React Native Documentation*. Recuperado de https://reactnative.dev/
- Scrum Guide. (2020). *The Scrum Guide*. Ken Schwaber & Jeff Sutherland. Recuperado de https://scrumguides.org/
- Spring Boot. (s. f.). *Spring Boot Reference Documentation*. Spring.io. Recuperado de https://spring.io/projects/spring-boot
- Trello. (s. f.). *Trello Help & Documentation*. Recuperado de https://trello.com/
- ws. (s. f.). *ws: a simple to use, blazing fast, and thoroughly tested WebSocket client and server for Node.js*. Recuperado de https://www.npmjs.com/package/ws