iLERNA.

MEMORIA ESCRITA DEL PROYECTO

CFGS Desarrollo de Aplicaciones Web

Adventure Share

Autor: Sergio Aguilera Márquez

Tutor: Tamara González Gómez

Fecha de entrega: 14/06/2025

Convocatoria: Segundo Semestre

Documentos del proyecto:
https://drive.google.com/drive/folders/1UMK_bItx6e9hYwUl8w9VBUhCAUdN0LMI?usp=drive_link

Índice de contenidos

1. Introducción

1.1. Motivación

La motivación de **AdventureShare** comenzó en una ruta por Glendalough, en Irlanda, donde me encontré con la dificultad de coordinar horarios y niveles de experiencia con mis compañeros de viaje. Fue frustrante descubrir que, pese a contar con numerosas aplicaciones y foros sobre trekking y senderismo, no existía un espacio que reuniera información actualizada y la posibilidad de interactuar directamente con quienes ya habían recorrido cada tramo. En aquel momento supe que necesitaba algo más que reseñas aisladas: una plataforma en la que cada viajero, ya fuese novato o veterano, pudiera aportar sus conocimientos y resolver dudas de forma inmediata.

En paralelo, observé cómo el turismo experiencial y el “slow travel” ganaba fuerza: cada vez son más las personas que prefieren sumergirse en la cultura local, explorar rutas menos transitadas y huir de los itinerarios masificados. Esta tendencia reclamaba un enfoque colaborativo, donde la voz de la comunidad fuese el principal referente para planificar cada etapa del viaje. Sabía que, para que esa colaboración funcionara, era imprescindible ofrecer un entorno intuitivo y amigable, capaz de mantener la información viva y al alcance de la mano en todo momento.

También me impactó comprobar lo valioso que resulta el intercambio de consejos entre usuarios. Los viajeros con más experiencia pueden señalar peligros ocultos, ofrecer atajos o recomendar puntos de descanso, mientras que quienes apenas dan sus primeros pasos en el trekking aportan entusiasmo y nuevas perspectivas. Quise construir un lugar donde esa retroalimentación fluyera de forma natural, enriqueciendo cada ruta y fomentando la confianza mutua. Una comunidad en la que cada aportación, por pequeña que parezca, sume para mejorar la seguridad y el disfrute de todos.

De ahí surge AdventureShare, con la ilusión de hacer más fáciles los encuentros entre viajeros afines y convertir la planificación de un viaje en una experiencia compartida. Pretendo que cada persona que use la plataforma sienta que no viaja sola, sino en compañía de una comunidad activa y dispuesta a ayudar, donde cada ruta se construye entre todos y cada aventura se vive con la seguridad de tener a alguien a quien recurrir en cualquier momento.

1.2. Abstract

The motivation for AdventureShare began during a hiking route through Glendalough, Ireland, where I faced the challenge of coordinating schedules and experience levels with my travel companions. It was frustrating to realize that, despite the many apps and forums available on trekking and hiking, there was no platform that combined up-to-date information with the ability to interact directly with those who had already walked those paths. At that moment, I understood we needed more than just isolated reviews: a platform where every traveler whether beginner or experienced could share their knowledge and answer questions instantly.

At the same time, I noticed how experiential tourism and “slow travel” were gaining popularity. More and more people are choosing to immerse themselves in local culture, explore lesser-known trails, and avoid overcrowded itineraries. This trend called for a collaborative approach, where the voice of the community becomes the main guide when planning each stage of the journey. I knew that for this collaboration to work, it was essential to provide an intuitive and user-friendly environment that keeps information fresh and accessible at all times.

I was also struck by how valuable it is to exchange advice between users. Experienced hikers can highlight hidden dangers, suggest shortcuts, or recommend good rest stops, while those who are new to trekking bring enthusiasm and fresh perspectives. I wanted to create a space

where that feedback flows naturally, enriching every route and building mutual trust. A community where every contribution no matter how small adds up to improve safety and enjoyment for everyone.

That's where AdventureShare comes from: the desire to make it easier for like-minded travelers to connect and turn trip planning into a shared experience. I hope every person who uses the platform feels like they're not traveling alone, but as part of an active community that's ready to help where each route is built together, and every adventure is lived with the confidence of having someone to rely on at all times.

1.3. Objetivos propuestos (generales y específicos)

AdventureShare nace con la intención de transformar la forma en la que se organizan, comparten y disfrutan las experiencias de aventura y senderismo. Se trata de un proyecto cuyos objetivos están enfocados en aprovechar el potencial de la tecnología geolocalizada y la fuerza de las comunidades digitales para responder a las necesidades de los viajeros modernos. Nuestros objetivos generales giran en torno al desarrollo de una plataforma intuitiva, la conexión entre personas con intereses comunes, la colaboración activa entre usuarios con distintos niveles de experiencia y la mejora continua de la planificación de rutas a través de contenido generado por la comunidad. Estos objetivos reflejan nuestro compromiso con la utilidad, la accesibilidad y el espíritu colaborativo que define el mundo del viaje compartido.

A su vez, nuestros objetivos específicos abordan aspectos técnicos y funcionales clave que permitirán que esta visión se convierta en una realidad práctica y eficaz. Desde herramientas de publicación geolocalizadas hasta filtros de búsqueda y sistemas de recomendación inteligentes, cada funcionalidad está diseñada para que AdventureShare no solo sea útil, sino

también adaptable, escalable y orientada al usuario. Del mismo modo, el proyecto contempla mecanismos de comunicación, seguridad y validación de contenido que garanticen una experiencia fluida y segura para todos los miembros de la comunidad. El objetivo final es ofrecer una plataforma viva, en constante evolución, que convierta la planificación de aventuras en un proceso social, fácil y enriquecedor.

Los objetivos generales son:

1. **Desarrollar una plataforma web intuitiva y accesible** que permita a los usuarios crear, explorar y compartir rutas y experiencias de aventura al aire libre.
2. **Fomentar la colaboración entre usuarios con distintos niveles de experiencia**, facilitando el intercambio de consejos, rutas y aprendizajes entre la comunidad.
3. **Construir una red social de viajeros** basada en el respeto, la ayuda mutua y la pasión por descubrir nuevas rutas, lugares y personas.
4. **Establecer una base tecnológica sólida que permita el crecimiento escalable** de la plataforma, así como su integración con herramientas de geolocalización, mapas y notificaciones.
5. **Garantizar una experiencia de usuario satisfactoria** mediante una interfaz clara, navegación fluida y funciones que se adapten a las necesidades reales de los viajeros.

Los objetivos específicos son:

1. **Implementar un sistema de registro y autenticación** que permita crear perfiles personales, gestionar rutas publicadas y seguir la actividad de otros usuarios.
2. Desarrollar una herramienta de publicación de rutas con localización en el mapa, detalles técnicos del recorrido, imágenes y puntos destacados.
3. **Integrar un sistema de búsqueda** que permita filtrar rutas por localización.
4. **Incorporar funcionalidades sociales como chats**, fomentando la interacción y la creación de grupos con intereses comunes.
5. **Establecer un sistema de valoraciones y comentarios** que permita a los usuarios aportar feedback sobre las rutas publicadas y mejorar la fiabilidad del contenido.
6. **Configurar procesos de prueba automatizados y despliegue continuo** (CI/CD) para mantener la plataforma estable, optimizada y en constante evolución.
7. **Implementar en Django/DRF el modelo y endpoints para solicitar, aceptar y rechazar plazas en un viaje**, y en React los formularios y vistas para que el viajero envíe su petición y el organizador la gestione.
8. **Permitir crear, editar y eliminar rutas completas (viajes) desde la misma interfaz**, consumiendo tus API REST para mantener el catálogo de viajes siempre actualizado.

9. **Desarrollar en el frontend la simulación del pago tras la aceptación de una reserva, y en el backend registrar el estado y la fecha del pago**, dejando la puerta abierta a integrar pasarelas reales (Stripe/PayPal) en el futuro.

2. Estado del Arte

La transformación digital ha cambiado por completo la forma en la que las personas planifican, experimentan y comparten sus viajes. El auge de las tecnologías móviles, el acceso masivo a internet y la popularización de modelos colaborativos han dado lugar a un nuevo ecosistema de aplicaciones enfocadas al turismo de aventura y a la organización de rutas al aire libre. En este contexto nace AdventureShare, un proyecto que se enmarca dentro de una evolución tecnológica y social más amplia, donde los usuarios no solo buscan consumir información, sino también formar parte activa de su generación y distribución.

En la actualidad existen diversas plataformas que cubren parcialmente lo que AdventureShare propone. Una de las más conocidas es Wikiloc, centrada principalmente en el registro y exploración de rutas al aire libre mediante GPS. Su comunidad de usuarios ha crecido significativamente, permitiendo a millones de personas compartir senderos, ciclorrutas, travesías en kayak y otros recorridos. Wikiloc destaca por la precisión en los datos técnicos de las rutas, pero tiene una orientación muy técnica y visualmente limitada, lo que puede resultar menos atractivo para usuarios principiantes o que busquen una experiencia más social.

Otra alternativa relevante es Komoot, una aplicación que combina planificación de rutas con recomendaciones basadas en la experiencia de otros usuarios. Komoot utiliza un sistema de navegación muy completo y una interfaz amigable, pero al igual que Wikiloc, su enfoque está más centrado en la navegación que en la conexión entre personas. No permite crear redes sociales o encontrar compañeros de aventura en función de fechas o nivel de experiencia.

También merece mención AllTrails, especialmente popular en Estados Unidos. Esta aplicación ofrece una base de datos muy extensa de rutas, con filtros por nivel de dificultad, duración o tipo de actividad. AllTrails incluye opiniones de usuarios, fotos y detalles técnicos, pero carece de una función para coordinar salidas grupales o conectar con otros viajeros directamente desde la app.

Por otro lado, existen plataformas como Couchsurfing o Workaway que, aunque no están orientadas al senderismo, sí representan ejemplos claros de economía colaborativa aplicada al turismo. Ambas permiten a los usuarios establecer relaciones basadas en la confianza, compartir recursos y vivir experiencias auténticas junto a locales o con otros viajeros. Sin embargo, su objetivo no es la planificación de rutas o excursiones, sino el alojamiento o el voluntariado, respectivamente.

En el caso de BlaBlaCar, sí existe un modelo más parecido al que AdventureShare busca aplicar: conectar a personas que quieren compartir un trayecto, ya sea por razones económicas o por sostenibilidad. Esta idea de viaje compartido, adaptada al mundo del senderismo, tiene un gran potencial si se enfoca no en el transporte, sino en la experiencia de aventura en sí misma.

Vamos a enumerar las ventajas y desventajas de cada una de las webs mencionadas anteriormente para tener un pequeño análisis de todas, y así ver en que se diferencia con AdventureShare.

Wikiloc:

Plataforma centrada en el registro y exploración de rutas al aire libre mediante GPS.

- **Ventajas**:
    1. Amplia comunidad global con millones de rutas compartidas.
    2. Datos técnicos precisos de distancia, desnivel y duración.
    3. Posibilidad de descargar rutas y usarlas sin conexión.

- **Desventajas**:
    1. Enfoque muy técnico, poco accesible para usuarios novatos.
    2. Interfaz anticuada y poco visual.
    3. Escasa o nula interacción social entre usuarios.

Komoot:

Aplicación que combina navegación GPS y recomendaciones personalizadas para actividades al aire libre.

- **Ventajas**:
    1. Planificación avanzada de rutas según nivel y tipo de terreno.
    2. Interfaz moderna e intuitiva.
    3. Sincronización con smartwatches y dispositivos GPS.

- **Desventajas**:

1. No permite buscar o contactar compañeros de ruta.
2. Limitado enfoque en comunidad o interacción directa.
3. Algunas funciones útiles están bloqueadas en versión gratuita.

AllTrails:

Plataforma popular en EE. UU. para explorar rutas con reseñas, fotos y filtros avanzados.

- **Ventajas**:
    1. Gran variedad de rutas filtrables por dificultad, tipo y duración.
    2. Comunidad activa con reseñas y valoraciones frecuentes.
    3. Interfaz limpia con buena visualización de mapas y fotos.

- **Desventajas**:
    1. Falta de funciones para conectar o coordinar con otros usuarios.
    2. No tiene mensajería ni sistema de eventos grupales.
    3. Muchas funciones clave requieren suscripción premium.

Couchsurfing / Workaway:

Plataformas basadas en economía colaborativa enfocadas en alojamiento o voluntariado.

- **Ventajas**:
    1. Promueven experiencias culturales auténticas y económicas.
    2. Permiten conocer locales o viajeros con intereses similares.
    3. Fomentan valores de hospitalidad, intercambio y respeto.

- **Desventajas**:
    1. No están orientadas a la organización de rutas ni aventuras.
    2. Requieren verificación de identidad para mayor seguridad.
    3. Pueden surgir situaciones incómodas sin una buena reputación previa.

BlaBlaCar:

Aplicación para compartir coche, con un modelo de emparejamiento basado en trayectos comunes.

- **Ventajas**:
    1. Sistema eficiente de conexión entre usuarios con rutas similares.
    2. Reducción de costes de transporte y emisiones.

3. Sistema de valoraciones para generar confianza.

- **Desventajas**:
    1. Limitada al transporte: no cubre experiencias o actividades.
    2. Interacción breve y puntual, sin construcción de comunidad.
    3. Problemas ocasionales de cancelaciones o usuarios no fiables.

A nivel de tendencias, el auge del “slow travel” (viajar de forma pausada y consciente), el interés por el turismo activo y sostenible, y la búsqueda de rutas alternativas a los destinos masificados han generado una demanda creciente de plataformas que ofrezcan contenido actualizado, de calidad y generado por la propia comunidad. Los usuarios no solo quieren saber “dónde ir”, sino también “con quién”, “cuándo es mejor”, y qué deben tener en cuenta antes de emprender una ruta.

Otro punto clave es la democratización tecnológica. Con un simple teléfono móvil es posible hoy en día grabar, geolocalizar, subir fotos, chatear con otros usuarios e incluso seguir rutas trazadas previamente por GPS. Esto permite que el contenido colaborativo sea más accesible y útil que nunca, siempre que exista una plataforma que lo centralice de forma ordenada y atractiva. Aquí es donde AdventureShare se diferencia: su propuesta combina lo técnico (rutas, mapas, recomendaciones) con lo social (comunidad, interacción, ayuda entre usuarios), algo que actualmente no se encuentra de forma integrada en una sola aplicación.

Además, la gamificación y los logros se han convertido en elementos clave para fomentar la participación. Plataformas como Strava han sabido integrar estos componentes en el deporte, premiando a los usuarios con insignias y rankings. AdventureShare podría aplicar una lógica similar, orientada no solo a la superación personal, sino también al apoyo mutuo y al espíritu de comunidad.

A pesar del avance de estas herramientas, también existen limitaciones importantes. Muchas plataformas actuales carecen de sistemas eficaces de validación del contenido, lo que puede derivar en rutas mal marcadas, desactualizadas o incluso peligrosas. La ausencia de sistemas de verificación entre usuarios o la falta de alertas en tiempo real son factores que afectan negativamente a la experiencia y la seguridad del viajero. Asimismo, pocas aplicaciones permiten gestionar en un solo lugar tanto la planificación de la aventura como la búsqueda de compañeros y la publicación de experiencias posteriores, lo que fragmenta el proceso y desincentiva el uso continuado.

En resumen, el estado actual del mercado muestra que existen buenas bases tecnológicas y comunidades activas en torno al turismo de aventura, pero también una clara oportunidad

para ofrecer algo más completo y cohesionado. AdventureShare se posiciona precisamente ahí: como una plataforma integral que permite descubrir rutas, encontrar compañeros con intereses afines, compartir experiencias de forma sencilla y crear una comunidad activa en torno al viaje colaborativo. En este sentido, el proyecto no parte de cero, sino que se apoya en una tendencia creciente, detecta sus carencias y plantea soluciones concretas que suman valor desde el primer uso.

Esta visión conectada, flexible y social es clave para responder a las nuevas formas de viajar que han emergido con fuerza en los últimos años, especialmente tras la pandemia, cuando se revalorizó el contacto con la naturaleza, la planificación entre grupos pequeños y la búsqueda de experiencias significativas en lugar de masificadas. AdventureShare no solo busca aprovechar estas tendencias, sino también consolidarse como una herramienta útil, confiable y cercana para todos aquellos que entienden el viaje como algo más que moverse: como una forma de vivir, compartir y crecer.

3. Metodología usada

Para el desarrollo del proyecto **AdventureShare** se ha utilizado la metodología de software en cascada con retroalimentación. Este enfoque secuencial y estructurado divide el trabajo en fases claramente delimitadas como el análisis de requisitos, diseño, implementación, verificación y mantenimiento, que se completan de manera progresiva. Una vez finalizada cada etapa, se realiza una revisión de resultados (retroalimentación) para ajustar posibles desviaciones o incorporar mejoras antes de avanzar a la siguiente fase. De este modo, se mantiene un marco ordenado y lógico, ideal para proyectos con objetivos bien definidos y recursos limitados, sin renunciar a la flexibilidad necesaria para corregir errores y optimizar el producto en cada hito.

La elección de esta metodología se fundamenta en la naturaleza del propio proyecto. AdventureShare es una aplicación web que busca facilitar el encuentro entre personas interesadas en realizar viajes de aventura de forma compartida. Su funcionamiento se basa en la creación de rutas, perfiles de usuario, reservas y gestión de la experiencia del viaje. Desde el principio, los requisitos estaban claros: el sistema debía permitir publicar viajes, buscarlos según origen, destino y fecha, registrarse y autenticarse como usuario, y gestionar las reservas de plazas disponibles. Este escenario es perfectamente compatible con la estructura que propone el modelo en cascada.

Una de las principales ventajas de este modelo es su previsibilidad. Como desarrollador individual, me ha permitido centrarme en una única fase a la vez, evitando el caos que podría

generar una metodología ágil, pensada para equipos que trabajan de forma colaborativa y que requieren adaptabilidad continua. Gracias al enfoque en cascada, cada decisión tomada en fases tempranas se ha podido trasladar de forma coherente a las siguientes, sin necesidad de rehacer partes anteriores.

Durante la fase de **análisis de requisitos**, se identificaron las funcionalidades clave y se elaboraron los primeros esquemas conceptuales. A partir de ahí, en la fase de **diseño**, se establecieron los modelos de datos, las relaciones entre entidades y la arquitectura general del sistema utilizando Django en el backend y React para el frontend. También se definieron las vistas, rutas y componentes principales.

En la **implementación**, se tradujo todo el diseño en código funcional. Se crearon los modelos en Django, los endpoints de la API REST, se configuró la base de datos en PostgreSQL y se desarrolló el frontend con React. Esta etapa se dividió internamente en tareas más pequeñas para facilitar la organización, respetando en todo momento el orden establecido en el diseño.

Una vez implementado, se procedió a la **verificación** mediante pruebas funcionales y de integración. Se comprobó el comportamiento de cada vista, formulario y acción esperada, incluyendo validaciones de registro, expiración de tokens JWT, filtrado de viajes y creación de reservas. El sistema fue testeado tanto desde el punto de vista del usuario final como desde el lado del administrador.

Por último, se documentó todo el proceso en la etapa de **mantenimiento**, dejando anotaciones sobre mejoras futuras, posibles funcionalidades extra como la implementación de un mapa en el que se refleje la ruta de un viaje mediante la activación de un GPS en un dispositivo móvil, y buenas prácticas de mantenimiento del código. Esta fase, aunque en menor medida durante un TFG, resulta clave para dejar preparado el proyecto en caso de querer escalarlo profesionalmente tras su presentación.

Para entender mejor el funcionamiento del modelo en cascada, podemos compararlo con otro proyecto tipo: **una intranet para un centro educativo**. Imagina que se desea construir un sistema en el que el profesorado pueda subir notas, los alumnos verlas, y los tutores acceder a los informes de rendimiento. Si desde el principio se conocen los roles, las funciones específicas de cada uno, y los flujos de interacción, el modelo en cascada es ideal: primero se analiza qué necesita cada tipo de usuario, luego se diseña el sistema con su base de datos y rutas, se implementa la funcionalidad y se comprueba que todo funciona como debe. Cada parte está bien cerrada antes de pasar a la siguiente, y eso permite que un solo desarrollador o un equipo pequeño mantenga el control total del proyecto.

Este ejemplo sirve para ilustrar cómo la metodología en cascada se ajusta a proyectos con una **estructura definida, sin grandes cambios en los requisitos durante el desarrollo**, y donde la documentación es importante, como en el contexto educativo o institucional.

En el caso concreto de AdventureShare, además de ser un proyecto individual con fechas límite marcadas, se ha pretendido simular un entorno profesional real, donde cada fase debe documentarse, justificarse y ser verificable. El modelo en cascada ha sido una herramienta perfecta para cumplir esos objetivos y, al mismo tiempo, facilitar una organización clara y efectiva del trabajo, tanto en lo técnico como en lo académico.

En resumen, gracias a la aplicación del modelo en cascada, AdventureShare ha podido completarse de forma ordenada, minimizando errores, facilitando la trazabilidad de cada decisión técnica y permitiendo un desarrollo progresivo, sólido y bien documentado. Si bien esta metodología no es la más flexible frente a cambios frecuentes, ha resultado ser la más adecuada para el contexto de este trabajo: un TFG con un objetivo concreto, un alcance definido y un único desarrollador al mando.

4. Tecnologías y herramientas utilizadas en el proyecto

En este proyecto, se ha seleccionado un conjunto de tecnologías actuales que, combinadas, ofrecen una solución robusta, moderna y eficiente para el desarrollo de una plataforma de viajes compartidos centrada en la aventura. A continuación, se describen las tecnologías utilizadas y los motivos de su elección:

Lenguaje de programación del servidor: Python

Python es un lenguaje de programación interpretado, de alto nivel, conocido por su sintaxis sencilla y legible, lo que facilita el desarrollo rápido y ordenado de aplicaciones web. Su amplia comunidad, su versatilidad y la gran cantidad de librerías disponibles lo convierten en una opción ideal para proyectos que requieren desarrollo rápido y estructurado. En AdventureShare, Python ha sido utilizado junto con el framework Django, lo que ha permitido una integración directa con el backend, el sistema de rutas, la validación de datos y la autenticación de usuarios.

Framework Backend: Django

Django es un framework de desarrollo web de alto nivel basado en Python, que permite construir aplicaciones seguras y mantenibles de forma rápida. Está diseñado bajo el principio

de “no te repitas” y promueve buenas prácticas de desarrollo. Una de sus grandes ventajas es que proporciona un sistema completo de administración, ORM (Object-Relational Mapping), manejo de sesiones, autenticación, validación de formularios y un sistema potente de plantillas. Esto ha sido clave para la gestión de los viajes, usuarios, reservas y todo el sistema interno de la plataforma.

Base de Datos: PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacional de código abierto, reconocido por su potencia, estabilidad y fiabilidad. Se ha utilizado en AdventureShare como motor principal de almacenamiento de datos por varias razones:

- **Compatibilidad con Django**: PostgreSQL se integra de forma nativa y fluida con Django, lo que ha facilitado la creación de modelos y migraciones de base de datos sin necesidad de configuraciones complejas.
- **Rendimiento**: Es capaz de manejar grandes volúmenes de datos y múltiples conexiones concurrentes, lo que lo hace ideal para plataformas que pueden escalar y recibir un crecimiento progresivo en número de usuarios y registros.
- **Seguridad**: Ofrece características avanzadas de seguridad como roles, permisos personalizados, y cifrado de datos.
- **Soporte para tipos de datos avanzados**: Incluye soporte para arrays, JSONB, búsquedas full-text, y más, lo cual permite extender la plataforma fácilmente en futuras versiones.

Lenguajes de cliente: HTML, CSS y JavaScript

Estos tres lenguajes son los pilares del desarrollo web en el lado del cliente. HTML permite estructurar el contenido, CSS se encarga del estilo visual y JavaScript aporta interactividad. Su uso combinado ha permitido crear una interfaz accesible, intuitiva y dinámica que facilita la navegación de los usuarios a través de los distintos apartados de la aplicación.

Biblioteca Frontend: React

React es una biblioteca de JavaScript desarrollada por Meta, que permite la creación de interfaces de usuario dinámicas y basadas en componentes. En AdventureShare, se ha utilizado para desarrollar el frontend de forma modular, facilitando la reutilización de código, la gestión del estado con hooks y el enrutamiento con React Router. Esta tecnología ha permitido construir una experiencia fluida, con formularios interactivos, feedback inmediato, y actualización de datos en tiempo real a través de llamadas API.

Framework CSS: Tailwind CSS

Tailwind CSS es un framework de diseño utility-first que permite aplicar estilos directamente en el HTML mediante clases predefinidas. A diferencia de frameworks tradicionales, Tailwind permite un control más preciso del diseño sin necesidad de sobrescribir estilos, lo que acelera el proceso de maquetación. Se eligió por su flexibilidad, su compatibilidad con React y por permitir construir una interfaz moderna, responsive y personalizable con gran velocidad.

Control de versiones: Git

El control de versiones del proyecto se ha llevado a cabo con Git, utilizando GitHub como plataforma de alojamiento. Esto ha permitido mantener un historial claro y ordenado de los cambios realizados, facilitando tanto el desarrollo incremental como la posibilidad de volver a versiones anteriores en caso de errores.

Gestión de tareas: Trello

Para organizar las tareas y fases del desarrollo, se ha utilizado Trello como herramienta de planificación. A través de tableros, listas y tarjetas, se ha podido llevar un control detallado de las funcionalidades pendientes, en proceso y completadas, mejorando la organización y ayudando a mantener los tiempos establecidos en el cronograma inicial.

Diseño gráfico: Canva

Aunque el foco del proyecto no ha sido el diseño gráfico, se han utilizado herramientas como Canva para la creación de logotipos, iconografía y banners promocionales. Canva es una plataforma en línea que ofrece plantillas, recursos visuales y una interfaz muy intuitiva que permite crear contenido visual de forma rápida y sin necesidad de conocimientos avanzados en diseño.

Grabación de recursos multimedia: Audacity

En el caso de futuros planes de ampliación de la plataforma para incluir relatos de viajes en formato podcast o testimonios de usuarios, se ha considerado el uso de Audacity como herramienta de grabación y edición de audio. Audacity es gratuito, multiplataforma y fácil de usar, permitiendo producir material de calidad profesional para ser distribuido en la web.

La combinación de todas estas tecnologías ha permitido construir una plataforma sólida, mantenible y escalable, que no solo cubre las necesidades actuales del proyecto, sino que está preparada para posibles mejoras, integración de APIs externas y funcionalidades avanzadas como pagos, geolocalización o sistemas de reputación en futuras versiones. Además, se ha priorizado siempre la experiencia del usuario final, asegurando una interfaz clara, rápida y accesible desde distintos dispositivos.

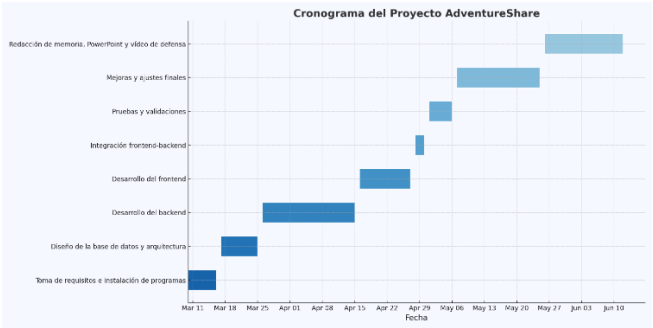
5. Planificación, Diagnóstico y Contexto Laboral

El desarrollo del proyecto AdventureShare ha seguido una planificación estructurada desde el 10 de marzo hasta el 15 de junio de 2025. El calendario se ha construido bajo el modelo en cascada, garantizando que cada fase fuera finalizada antes de dar comienzo a la siguiente. A través de esta estructura, ha sido posible mantener un control constante sobre las tareas, el progreso y los plazos de entrega establecidos.

A continuación se muestra el cronograma detallado del proyecto:

Fase del Proyecto	Fecha de Inicio	Fecha de Finalización
Toma de requisitos e instalación de programas.	10/03/2025	16/03/2025
Diseño de la base de datos y arquitectura	17/03/2025	25/03/2025
Desarrollo del backend	26/03/2025	15/04/2025
Desarrollo del frontend	16/04/2025	27/04/2025
Integración frontend-backend	28/04/2025	30/04/2025
Pruebas y validaciones	01/05/2025	06/05/2025
Mejoras y ajustes finales	07/05/2025	25/05/2025
Redacción de la memoria, power point y video de defensa	26/05/2025	12/06/2025

A continuación se muestra el diagrama de gantt del proyecto:



Diagnóstico: Análisis DAFO del proyecto AdventureShare

Fortalezas:

- La principal fortaleza del proyecto es su carácter original: AdventureShare no busca ser una réplica de plataformas de viajes como BlaBlaCar, sino una comunidad que une a personas con intereses afines en el senderismo, la naturaleza y los viajes alternativos.
- Su planteamiento modular y escalable permite integrar nuevas funcionalidades como pasarelas de pago, sistemas de reputación entre usuarios, filtros por intereses o conexión con APIs de mapas.
- Se ha utilizado un stack tecnológico profesional (Django + PostgreSQL + React + Tailwind), lo que facilita su expansión y adaptación futura en entornos reales.
- La experiencia adquirida en el desarrollo fullstack y en la planificación de un producto con posibilidad de comercialización aporta un valor diferencial al perfil profesional del autor.

Debilidades:

- El proyecto ha sido desarrollado por una sola persona, lo cual limita la capacidad de validación con usuarios reales, diseño colaborativo y pruebas en condiciones de alta demanda.
- Por cuestiones de tiempo, se han priorizado funcionalidades clave, dejando algunas complementarias para fases futuras. Esto puede afectar a la percepción de completitud si se compara con aplicaciones ya consolidadas.
- La falta de financiación inicial hace que todas las decisiones tecnológicas se hayan tomado pensando en entornos gratuitos o de bajo coste, limitando el alcance de ciertas herramientas profesionales.

Oportunidades:

- Existe un nicho creciente de personas interesadas en la aventura compartida, el slow tourism, el trekking o la movilidad sostenible. Este público suele buscar experiencias auténticas más allá del turismo masificado.
- El enfoque comunitario y colaborativo de la plataforma permite que los propios usuarios construyan el valor del servicio: cada nueva ruta, viaje o comentario contribuye a enriquecer el ecosistema.
- Puede vincularse a instituciones como ayuntamientos, organizaciones de turismo rural o empresas de experiencias al aire libre, lo que abre puertas a colaboraciones o modelos B2B en un futuro.
- Su estructura podría aprovechar funcionalidades de geolocalización, IA para recomendaciones, o incluso sistemas de gamificación para fidelizar a los usuarios más activos.

Amenazas:

- La entrada de nuevas plataformas o la ampliación de servicios similares por parte de grandes empresas puede suponer una barrera para competir en el mismo mercado.
- La naturaleza social del proyecto depende en gran parte del volumen de usuarios activos. Una baja participación inicial podría dificultar su arranque o dinamismo.
- El mantenimiento técnico a largo plazo requiere recursos continuos. Si no se profesionaliza, podría quedar obsoleto o inactivo por falta de soporte.

Caracterización de AdventureShare

AdventureShare es una plataforma web fullstack que permite a usuarios compartir viajes de aventura, buscar rutas según localización y preferencias, y coordinarse con otros excursionistas. Se caracteriza por los siguientes puntos clave:

- **Colaborativa**: basada en la contribución y participación activa de los propios usuarios.
- **Escalable**: diseñada para admitir nuevos módulos y crecer tanto en número de usuarios como en funcionalidad.
- **Nicho específico**: enfocada exclusivamente en el mundo de la naturaleza, las rutas, la aventura y el senderismo, no en el transporte generalista.
- **Responsable y sostenible**: promueve la movilidad compartida, la reducción del impacto ambiental y la conexión con el entorno.
- **Adaptable**: pensada para integrarse fácilmente con futuras tecnologías como mapas interactivos, sistemas de pago o autenticación social.

6. Análisis del proyecto

En este análisis se describen las características funcionales y estructurales del sistema AdventureShare, una plataforma web que permite a los usuarios crear, unirse y organizar viajes de aventura colaborativos. Se han identificado las necesidades principales, se han definido requisitos, actores, y se presentan diversos diagramas que reflejan la estructura interna y las interacciones con el sistema.

Requisitos Funcionales

1. **Registro de usuarios**: Cualquier persona debe poder registrarse con su email, nombre de usuario y contraseña.
2. **Inicio de sesión y autenticación**: Los usuarios deben poder iniciar y cerrar sesión en la plataforma.
3. **Creación de viajes**: Los usuarios pueden crear nuevos viajes, indicando lugar, fecha, plazas y descripción.
4. **Solicitud de reserva**: Otros usuarios pueden solicitar unirse a un viaje creado por otra persona.
5. **Gestión de reservas**: El organizador de cada viaje puede aceptar o rechazar solicitudes de otros usuarios.
6. **Sistema de pagos**: Una vez aceptada una reserva, se puede realizar el pago correspondiente.
7. **Valoración de viajes**: Tras finalizar un viaje, los participantes pueden dejar una valoración del organizador.
8. **Perfil de usuario**: Cada usuario puede editar su biografía, imagen de perfil, y ver su historial de viajes y valoraciones recibidas.
9. **Sistema de mensajes**: Los usuarios pueden enviarse mensajes privados dentro de la plataforma.
10. **Visualización de viajes públicos**: Cualquier usuario autenticado puede explorar viajes publicados por otros.

Requisitos No Funcionales

1. **Disponibilidad**: El sistema debe estar accesible al menos el 99% del tiempo. Esto se puede conseguir alojando la web en un hosting.
2. **Rendimiento**: Las consultas y carga de páginas deben completarse en menos de 2 segundos.

3. **Seguridad:** Toda la información sensible debe estar protegida, especialmente contraseñas y pagos.
4. **Compatibilidad:** La aplicación debe funcionar correctamente en dispositivos móviles y navegadores modernos.
5. **Escalabilidad**: El sistema debe soportar un crecimiento progresivo del número de usuarios y viajes. Esto se irá mejorando conforme la demanda lo requiera.
6. **Usabilidad**: Interfaz clara, sencilla y coherente, adaptada a usuarios de cualquier edad.
7. **Mantenibilidad**: El código debe estar estructurado de forma que facilite su ampliación o modificación.
8. **Internacionalización**: Aunque inicialmente el idioma será el español, se debe permitir la futura traducción.
9. **Backup Automático:** Es importante que cuando la web este disponible para todos los públicos tenga un sistema que deba guardar copias de seguridad diarias de la base de datos.
10. **Soporte**: Cuando la web esté operativa para todo el público debe ofrecer un sistema de contacto o soporte por correo para usuarios con incidencias.

Modelo de Datos: Entidades y Atributos

Después de definir los requisitos funcionales y no funcionales del sistema, es necesario diseñar el modelo de datos que servirá de base para el desarrollo. A continuación, se presentan las entidades principales del sistema AdventureShare, junto con sus respectivos atributos. Estas entidades permiten representar de forma estructurada toda la información relevante relacionada con los usuarios, viajes, reservas, pagos, valoraciones y mensajes dentro de la plataforma.

Este modelo está diseñado siguiendo principios de normalización y claridad, con claves primarias y foráneas bien definidas que permiten establecer relaciones entre las distintas tablas:

Usuario

- usuario_id (PK)
- nombre
- email
- contraseña
- fecha_registro
- imagen_perfil

- biografía

Viaje

- viaje_id (PK)
- título
- lugar
- fecha
- plazas
- precio
- organizador_id (FK → Usuario)

Reservas

- reserva_id (PK)
- usuario_id (FK)
- viaje_id (FK)
- estado (pendiente / aceptada / rechazada)
- pago_id (FK)

Pago (simulado en el frontend por la complicación que supone hacerlo con dinero real)

- pago_id (PK)
- estado
- fecha_pago

Valoración

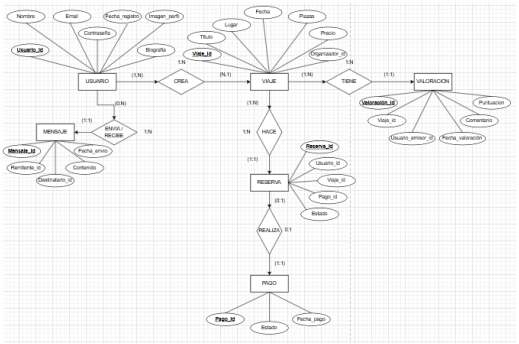
- valoracion_id (PK)
- viaje_id (FK)
- usuario_emisor_id (FK)
- puntuación
- comentario
- fecha_valoración

Mensaje

- mensaje_id (PK)
- remitente_id (FK)
- destinatario_id (FK)
- contenido
- fecha_envio

Relaciones:

- Un Usuario puede crear muchos Viajes (1:N).
- Un Viaje es organizado por un único Usuario(N:1)
- Un Usuario puede realizar muchas Reservas (1:N).
- Un Viaje puede tener muchas Reservas de distintos Usuarios(1:N).
- Una Reserva a un único Usuario y a un único Viaje(N:1).
- Una Reserva puede estar asociada a un Pago (simulado) (0:1).
- Un Usuario puede enviar muchos Mensajes(1:N).
- Un Usuario puede recibir muchos Mensajes(1:N).
- Un Viaje puede tener muchas Valoraciones (1:N).
- Una Valoración pertenece a un único Usuario y a un único Viaje(N:1).

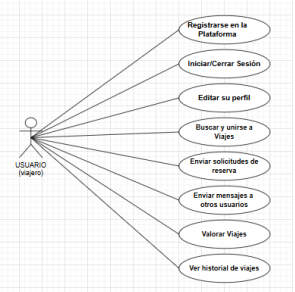


Para desarrollar los casos de uso de nuestro proyecto de plataforma AdventureShare, es importante identificar todas las interacciones que los distintos tipos de usuarios pueden tener con el sistema. Esto permite comprender de manera clara cómo cada actor participa en la dinámica de la aplicación. A continuación, se presenta una lista de posibles casos de uso para los distintos perfiles de usuarios que interactúan con la plataforma:

Casos de uso para usuarios (viajero o administrador):

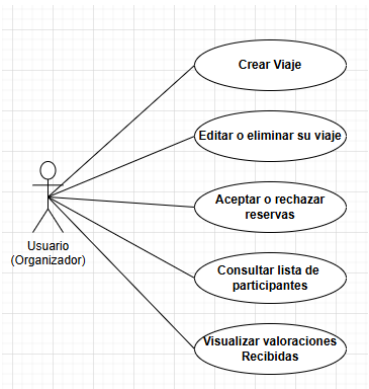
1. Registrarse en la plataforma
2. Iniciar / cerrar sesión
3. Editar su perfil
4. Buscar y unirse a viajes
5. Enviar solicitudes de reserva

6. Enviar mensajes a otros usuarios
7. Realizar pagos
8. Valorar viajes
9. Ver historial de viajes



Casos de uso para organizadores:

1. Crear viaje.
2. Editar o eliminar su viaje.
3. Aceptar / rechazar reservas.
4. Consultar lista de participantes.
5. Visualizar valoraciones recibidas.



Especificaciones de Casos de Uso:

Caso de uso: Crear Viaje

- **Actor**: Usuario registrado
- **Descripción**: El usuario accede al formulario para crear un viaje y completa los campos requeridos.
- **Precondición**: Usuario ha iniciado sesión.
- **Flujo principal**:

    1. Accede a la sección "Crear viaje"
    2. Completa título, destino, fecha, plazas y descripción

3. Envía el formulario
4. El sistema guarda el viaje en la base de datos

- **Resultado esperado**: Viaje creado y visible en el listado de viajes públicos del usuario.

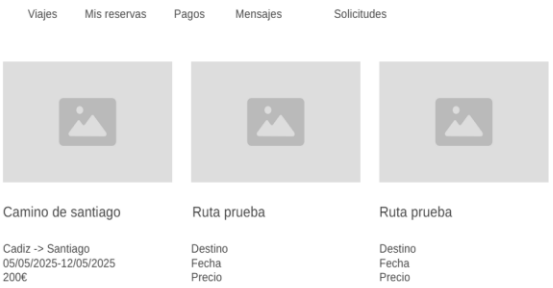
Caso de uso: Solicitar Reserva

- **Actor**: Usuario registrado
- **Precondición**: Ha iniciado sesión y ha seleccionado un viaje
- **Flujo principal**:
    1. El usuario accede a la ficha de un viaje
    2. Hace clic en "Solicitar plaza"
    3. El sistema registra la solicitud en estado "pendiente"
- **Resultado esperado**: La reserva aparece en la sección de reservas del organizador.

7. Diseño del proyecto

Durante la fase de diseño del proyecto AdventureShare, se realizaron **wireframes** iniciales utilizando la herramienta wireframe.cc, con el objetivo de establecer una estructura clara y funcional de las vistas principales de la plataforma antes de comenzar su desarrollo.

Estos bocetos sirven como referencia visual para definir la disposición de los elementos, los flujos de navegación esperados por el usuario, y la experiencia de uso general de la plataforma. A continuación, se presentan dos de los diseños más representativos:



Página de Inicio de Sesión y Registro

En este wireframe se presenta una pantalla principal compuesta por un selector de pestañas con las opciones “Iniciar sesión” y “Registrarse”, seguido por los campos básicos de autenticación: nombre de usuario, correo electrónico y contraseña. Se incluye un espacio visual reservado para el logotipo o imagen decorativa central, con un diseño centrado y minimalista.

Esta estructura pretende facilitar un acceso rápido y sencillo a la plataforma, dando prioridad a la experiencia del usuario desde el primer contacto con la aplicación.

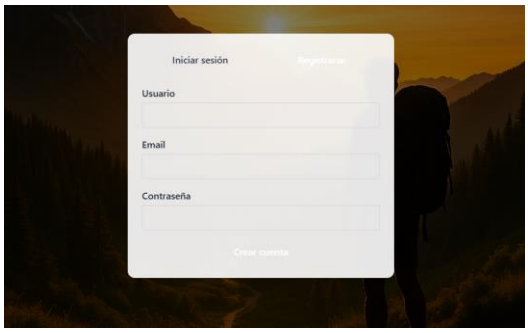
Página Principal de la Web

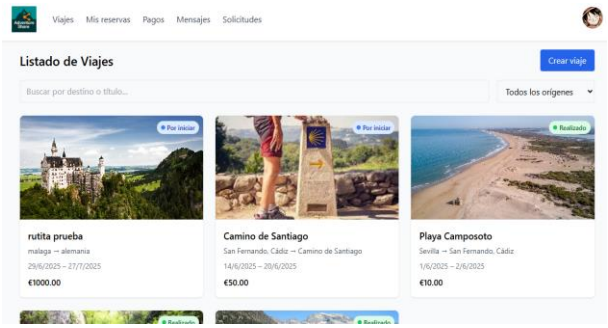
En esta vista se simula el aspecto general de la página principal tras iniciar sesión. Se muestra una barra de navegación superior con accesos a las secciones clave: “Viajes”, “Mis reservas”, “Pagos”, “Mensajes” y “Solicitudes”. Justo debajo, se organizan las tarjetas de los viajes disponibles, cada una con una imagen representativa, el título del viaje, su destino, fechas y precio.

El objetivo de este diseño es proporcionar una interfaz clara y organizada, que permita al usuario explorar fácilmente las aventuras disponibles y acceder a información clave de forma visual y directa.

Diseño Implementado

Tras la fase de planificación inicial y creación de wireframes, se desarrolló la interfaz definitiva de la plataforma **AdventureShare**. A continuación, se presentan las capturas de pantalla de las principales vistas implementadas, ya con diseño gráfico real, integración de estilos y comportamiento interactivo:





Página de Inicio de Sesión y Registro

En la versión final, la página de login y registro incorpora una **imagen de fondo a pantalla completa** con temática de montaña al atardecer, evocando visualmente el espíritu de aventura que define la plataforma. La interfaz de autenticación aparece sobre un contenedor translúcido y centrado, con esquinas redondeadas y estilo moderno, manteniendo la misma estructura funcional definida en el wireframe.

Se han respetado los campos originales: nombre de usuario, email y contraseña, pero ahora presentados con una tipografía clara y moderna, en un entorno que transmite calidez, conexión con la naturaleza y profesionalismo. La experiencia de usuario mejora notablemente gracias al contraste visual y a la simplicidad estética.

Página Principal de la Web

El listado de viajes fue implementado con un diseño limpio y atractivo, utilizando tarjetas con imágenes reales de los destinos. La barra de navegación superior se mantuvo según lo planteado en el diseño inicial, pero en esta versión incluye el logotipo de la plataforma y el avatar del usuario.

Cada tarjeta de viaje incluye título, ruta, fechas, precio y un estado dinámico ("Por iniciar", “En curso” o "Realizado") que se muestra mediante una insignia visual en color azul, naranja o verde. Además, se añadió un botón funcional de "Crear viaje", y un sistema de filtros y búsqueda que mejora significativamente la usabilidad respecto al diseño previo.

Este diseño final no solo cumple con la funcionalidad prevista, sino que además **supera el mockup inicial en aspectos visuales, usabilidad y coherencia temática**, ofreciendo una experiencia inmersiva que conecta con la esencia del proyecto.

Descripción del Diseño de Interfaz

El diseño visual de AdventureShare ha sido concebido desde un enfoque centrado en el usuario, con el objetivo de proporcionar una experiencia intuitiva, agradable y accesible desde cualquier dispositivo. Se ha optado por un estilo moderno y minimalista, en línea con las tendencias actuales del desarrollo web, donde prima la funcionalidad, la limpieza visual y la coherencia estética en todas las vistas.

El desarrollo de la interfaz se ha realizado utilizando React, una librería de JavaScript muy popular que permite construir aplicaciones web modulares y dinámicas. Gracias a su enfoque basado en componentes reutilizables, el diseño de cada vista (como la página de inicio, perfil, reservas o creación de viaje) se ha estructurado de forma organizada, facilitando tanto la mantenibilidad del código como la eficiencia en el desarrollo. React también permite gestionar el estado de la interfaz de forma sencilla y reactiva, lo que mejora la experiencia de usuario y la velocidad de respuesta de la plataforma.

En la pantalla inicial, correspondiente al acceso de usuarios, se presenta un formulario centrado sobre una imagen de fondo en alta resolución con temática natural, lo que no solo refuerza el espíritu de aventura de la plataforma, sino que además genera una primera impresión visualmente atractiva. El formulario cuenta con pestañas para alternar entre el inicio de sesión y el registro, y está implementado sobre un contenedor translúcido con esquinas redondeadas, generando una sensación moderna, elegante y profesional.

El resto de vistas de la aplicación siguen un mismo patrón de diseño coherente. Por ejemplo, en la página principal se utiliza una estructura en formato de tarjetas para representar los viajes disponibles. Cada tarjeta muestra una imagen real del destino, el nombre del viaje, la ruta, las fechas, el precio y el estado actual (como “Por iniciar”, “En curso” o “Realizado”). Esta presentación visual permite a los usuarios identificar rápidamente la información relevante, facilitando la navegación y mejorando la comprensión del contenido.

A nivel técnico, se ha utilizado el framework Tailwind CSS, que proporciona un sistema de clases utilitarias para aplicar estilos de manera directa y estructurada. Gracias a Tailwind, se ha podido implementar una maquetación responsive adaptativa basada en tecnologías modernas como Flexbox y CSS Grid, lo que asegura una visualización fluida tanto en ordenadores como en tablets o móviles. Los elementos se reordenan automáticamente en función del ancho de pantalla, garantizando una experiencia uniforme y satisfactoria en todo

tipo de dispositivos.

En cuanto a accesibilidad y usabilidad, se han aplicado principios fundamentales como el contraste suficiente entre texto y fondo, tamaños de fuente legibles, navegación clara y botones bien diferenciados. Además, se ha mantenido una estructura semántica del código HTML para facilitar su comprensión por lectores de pantalla y otras tecnologías de asistencia.

Todo este diseño busca reducir al máximo la curva de aprendizaje, logrando que cualquier usuario, independientemente de su nivel técnico, pueda comprender y utilizar la plataforma desde el primer momento sin necesidad de instrucciones adicionales.

Estructura del Proyecto

El proyecto AdventureShare se ha dividido en dos grandes bloques claramente diferenciados: el backend desarrollado con Django y el frontend construido con React. Esta separación de responsabilidades facilita el mantenimiento del sistema y permite escalar cada parte de forma independiente.

El backend de AdventureShare ha sido desarrollado utilizando el framework Django junto con Django REST Framework, lo que permite estructurar el servidor de forma modular y exponer una API RESTful robusta y escalable. La arquitectura está organizada en aplicaciones independientes que separan las responsabilidades funcionales del sistema.

Las principales apps desarrolladas son:

- viajes/: contiene la lógica relacionada con los viajes organizados por los usuarios, incluyendo el modelo Viaje, sus vistas (ViajeViewSet) y los serializadores. Desde aquí se gestiona la creación, visualización, modificación y eliminación de viajes.

- usuarios/: gestiona el registro, autenticación y perfil de los usuarios. Incluye endpoints para modificar datos personales, cambiar contraseña y consultar el perfil del usuario autenticado mediante un endpoint personalizado (/me/). También centraliza la configuración del sistema de autenticación con tokens JWT.

- reservas/: implementa el sistema de solicitudes de participación en los viajes. Gestiona el modelo Reserva, su estado (pendiente, aceptada, rechazada), y su relación con los viajes y

los usuarios. También coordina la lógica para el filtrado de reservas y su acceso por parte del organizador del viaje.

- pagos/: si bien los pagos no están conectados a una pasarela real como Stripe, esta app simula el flujo de pago mediante un modelo Pago que registra el estado y la fecha del pago. La simulación permite mantener el flujo de reserva sin necesidad de integrar pagos reales en esta primera versión del proyecto.

- valoraciones/: permite a los participantes dejar comentarios y puntuaciones sobre los viajes una vez finalizados, alimentando así el sistema de reputación de los organizadores.

- mensajes/: gestiona la mensajería interna entre usuarios registrados. A través del modelo Mensaje, se pueden enviar textos privados entre usuarios, funcionalidad clave para la comunicación directa antes de confirmar una reserva.

Estructura interna de cada app

Cada app contiene los archivos habituales en Django:

- models.py: define las entidades y relaciones de base de datos mediante el ORM.
- serializers.py: convierte instancias de modelos en JSON para la API y viceversa.
- views.py: implementa la lógica de negocio usando ViewSets personalizados.
- admin.py: registra los modelos para gestionarlos desde el panel administrativo.
- apps.py: define la configuración y nombre interno de cada app.
- tests.py: archivo destinado a pruebas automáticas (no obligatorio en esta fase).
- migrations/: carpeta generada automáticamente para gestionar los cambios en la base de datos.

Ruteo general del proyecto

El archivo backend/urls.py actúa como punto central del enrutamiento del backend. Utiliza un DefaultRouter de Django REST Framework para registrar todas las rutas de las distintas apps. Además, se definen aquí las rutas relacionadas con la autenticación JWT proporcionadas por SimpleJWT:

- /api/token/: permite obtener un token de acceso y refresco al iniciar sesión.

- /api/token/refresh/: permite renovar un token antes de que expire.

Autenticación con JWT

El sistema de autenticación de AdventureShare está basado en JSON Web Tokens (JWT), gestionado mediante el paquete djangorestframework-simplejwt. Este enfoque proporciona mayor seguridad y escalabilidad que el sistema de sesiones tradicional, ya que permite validar usuarios en todas las peticiones a través de un token enviado en los headers.

Base de Datos

El proyecto utiliza PostgreSQL como sistema de gestión de base de datos, por su potencia, fiabilidad y compatibilidad total con Django. Todas las entidades del sistema están modeladas con el ORM de Django, lo que permite mantener la integridad referencial y migrar fácilmente la estructura de datos a través de las herramientas que ofrece el framework.

Frontend (React)

La interfaz de usuario de AdventureShare se ha desarrollado utilizando React, aprovechando su arquitectura basada en componentes y su ecosistema moderno. Esta decisión permite una organización clara del código, reutilización eficiente de elementos de interfaz y una experiencia de usuario fluida y dinámica.

La estructura del frontend está ubicada en la carpeta `src/` e incluye las siguientes subcarpetas principales:

- pages/: contiene las pantallas o vistas principales del sitio, como `LoginPage`, `ProfilePage`, `CreateTripPage`, `PaymentPage`, `ReservationListPage`, `TripDetailPage`, entre otras. Cada una representa una ruta accesible desde la navegación de la web.

- components/: almacena los componentes reutilizables como `TripCard`, `ReservationCard`, `Footer`, `NavMenu`, `PrivateRoute` o `ChatModal`. Estos elementos permiten construir las interfaces de forma modular y mantener una interfaz coherente en todas las páginas.

- contexts/: contiene el `AuthContext`, que gestiona el estado de autenticación del usuario en toda la aplicación. Proporciona acceso al usuario actual, así como funciones de login, logout y registro, y permite proteger rutas privadas.

- services/: contiene el archivo `api.js`, que centraliza todas las llamadas a la API REST usando Axios. Aquí se define la lógica para incluir automáticamente el token JWT en los headers de las peticiones autenticadas, y se facilita el consumo de endpoints desde cualquier componente o página.

- assets/: contiene recursos estáticos como imágenes, logotipos y otros archivos multimedia usados en la interfaz, por ejemplo, `AdventureShareLogo.png`.

Además, en la raíz del proyecto se encuentran archivos importantes como `App.js` (componente principal), `index.js` (punto de entrada de React), `App.css` e `index.css` (estilos globales), así como configuraciones adicionales como `.env` y `craco.config.js`.

La navegación entre páginas se gestiona mediante la librería `react-router-dom`, que permite definir rutas públicas y privadas (protegidas mediante `PrivateRoute`). Los hooks `useNavigate`, `useParams` y `useLocation` se emplean para la gestión de rutas dinámicas y control de flujo entre componentes.

El estado global de la sesión se mantiene usando `useContext`, facilitando así la autenticación persistente y el acceso a información del usuario desde cualquier parte de la aplicación.

En conjunto, esta estructura modular permite escalar el proyecto fácilmente, mantener un código limpio y organizado, y ofrecer una experiencia de usuario coherente, intuitiva y responsive.

Funcionalidad del Proyecto: Clases, funciones y archivos clave

A continuación, se describen algunas de las clases, funciones y archivos más importantes que constituyen el núcleo funcional de AdventureShare, tanto en el backend como en el frontend.

Backend (Django)

- models.py (ejemplo: Reserva)
El archivo `models.py` de la app `reservas` define el modelo `Reserva`, que representa la solicitud de un usuario para unirse a un viaje. Este modelo incluye claves foráneas hacia `Usuario` y `Viaje`, así como un campo `estado` (pendiente, aceptada o rechazada) y una relación con el modelo `Pago`. Esto permite gestionar fácilmente las solicitudes y su progreso.

- views.py (ejemplo: ReservaViewSet)
En `views.py`, el `ReservaViewSet` hereda de `ModelViewSet` y ofrece los métodos CRUD para gestionar reservas. Incluye filtros personalizados para que un organizador solo vea las reservas asociadas a sus viajes y un usuario vea únicamente sus propias solicitudes.

- serializers.py (ejemplo: ReservaSerializer)
Este archivo convierte las instancias de `Reserva` en objetos JSON. Se personaliza la validación para evitar reservas duplicadas o acciones no autorizadas.

- urls.py (proyecto)
En `backend/urls.py` se centralizan todas las rutas de la API, registrando cada `ViewSet` con el router. También se definen las rutas para autenticación con JWT.

Frontend (React)
- AuthContext.jsx
En `contexts/AuthContext.jsx` se define el contexto de autenticación global. Incluye funciones como `login`, `logout`, `register`, y mantiene el estado del usuario autenticado. Este contexto se utiliza en todo el frontend para proteger rutas y mostrar información personalizada.

- api.js
El archivo `api.js` en la carpeta `services/` configura Axios para realizar llamadas HTTP a la API. Inyecta automáticamente el token JWT en las cabeceras si el usuario está autenticado, lo que permite acceder a endpoints protegidos sin repetir lógica en cada componente.

- ReservationListPage.jsx
Este componente, ubicado en `pages/`, se encarga de mostrar las reservas de un usuario u organizador. Hace llamadas a la API para obtener los datos, los filtra según el rol y estado, y permite al organizador aceptar o rechazar solicitudes. Utiliza hooks como `useEffect`, `useState` y el contexto de autenticación para determinar el comportamiento y qué

información mostrar.

- TripListPage.jsx
Esta página muestra todos los viajes públicos disponibles. Permite buscar, filtrar por origen, y acceder a los detalles de cada uno. Utiliza el componente `TripCard.jsx` para renderizar cada viaje de forma individual.

- PaymentPage.jsx
En esta página el usuario simula el proceso de pago tras ser aceptado en un viaje. Al hacer clic en el botón de pago, se realiza una llamada a la API para actualizar el estado del modelo `Pago`. Aunque no se realiza un pago real, la funcionalidad está diseñada para integrar fácilmente una pasarela en el futuro.

Esta combinación de componentes en React y clases en Django permite que AdventureShare funcione como una plataforma completa y dinámica, gestionando desde la navegación hasta la lógica de negocio con una arquitectura clara y bien definida.

Cabe destacar que el proyecto incluye otros archivos y componentes adicionales, tanto en el backend como en el frontend, que no han sido descritos aquí en detalle por cuestiones de extensión. Sin embargo, los ejemplos presentados representan los elementos clave que conforman el núcleo funcional de AdventureShare y permiten comprender cómo se estructura y opera internamente la aplicación.

Decisiones Técnicas y Consideraciones del Desarrollo

Durante el desarrollo de AdventureShare se han tomado una serie de decisiones técnicas orientadas a equilibrar la funcionalidad, la facilidad de implementación y la escalabilidad futura del sistema. A continuación se detallan las más relevantes:

Elección de tecnologías:
Se optó por desarrollar el backend con Django debido a su madurez, rapidez de desarrollo y su integración con Django REST Framework, lo que facilitó la exposición de una API RESTful estructurada. Además, Django incluye un sistema de autenticación integrado y un ORM potente, que permitió modelar las entidades del sistema sin necesidad de escribir consultas SQL manuales.

El frontend fue desarrollado con React por su capacidad para construir interfaces reactivas y su ecosistema moderno. La separación de responsabilidades en componentes reutilizables y el uso de hooks como `useState`, `useEffect` y `useContext` permitieron un desarrollo ágil y ordenado.

Simulación de pagos:
Debido a la complejidad y requisitos legales que implica la integración con plataformas de pago reales (como Stripe o PayPal), se optó por simular el flujo de pagos en el frontend. Esta simulación permite mantener la lógica del sistema intacta y deja abierta la posibilidad de incorporar una pasarela real en futuras versiones.

Autenticación mediante JWT:
En lugar de usar sesiones tradicionales, se implementó autenticación basada en JSON Web Tokens (JWT) utilizando `SimpleJWT`. Esta decisión aporta mayor seguridad en entornos modernos y facilita la escalabilidad, ya que el frontend puede gestionar el estado de autenticación de forma autónoma, incluyendo el almacenamiento y renovación del token.

Gestión del estado global:
Se implementó un `AuthContext` personalizado para React, que permite mantener el estado de autenticación global en toda la aplicación. Esta decisión permite acceder a la información del usuario y sus privilegios desde cualquier componente sin necesidad de prop drilling.

Consumo de API centralizado:
Se creó un archivo `api.js` para encapsular todas las llamadas a la API con Axios. Esto permite un control unificado de los headers, manejo de errores y facilita modificaciones futuras (por ejemplo, si cambia el dominio de la API).

Elección de Tailwind CSS:
Tailwind CSS fue elegido por su enfoque de clases utilitarias que permite maquetar componentes de forma rápida y sin necesidad de escribir CSS tradicional. Además, su integración con Flexbox y Grid facilitó el desarrollo de un diseño responsive adaptable a móviles, tablets y escritorio.

Limitaciones y mejoras futuras:
- El sistema de pagos es actualmente una simulación. En versiones futuras podría integrarse Stripe para realizar transacciones reales.
- No se ha implementado aún un sistema de notificaciones o alertas internas.

- No se contempla un panel de administración para los viajes o estadísticas globales.
- La lógica de control de permisos puede ampliarse para incluir más roles o restricciones específicas por tipo de usuario.
- La accesibilidad ha sido tenida en cuenta, pero se podría realizar un test específico de compatibilidad WCAG para validarla al completo.

Estas decisiones reflejan un enfoque pragmático y profesional, centrado en construir una base sólida, funcional y lista para ampliaciones futuras si el proyecto continúa su desarrollo tras la entrega.

Consideraciones Finales del Diseño y Desarrollo

La fase de diseño y desarrollo de AdventureShare ha sido clave para materializar una idea funcional y adaptada a un contexto real de uso. A lo largo del proceso, se han combinado herramientas modernas y buenas prácticas para ofrecer una plataforma sólida, intuitiva y preparada para futuras mejoras.

Uno de los principales retos ha sido mantener la coherencia entre el diseño previo (mockups y wireframes) y la implementación real en React y Django. A pesar de los ajustes realizados durante la programación, se ha respetado en gran medida la estructura y disposición de los elementos definidos en la fase de diseño.

En términos técnicos, se ha logrado implementar todas las funcionalidades básicas previstas en los requisitos: autenticación JWT, gestión de viajes, reservas, pagos simulados, valoraciones, mensajería interna y edición de perfil. Todo ello sobre una base de datos PostgreSQL robusta y mediante una arquitectura modular que facilita el mantenimiento del sistema.

El uso de React con Tailwind CSS ha permitido crear una interfaz moderna y responsive, asegurando una experiencia de usuario fluida en dispositivos móviles y ordenadores. Por otro lado, Django y Django REST Framework han demostrado ser herramientas eficaces para estructurar un backend fiable y escalable.

En esta primera versión del proyecto se ha priorizado un diseño limpio y la cobertura de funcionalidades clave, dejando la puerta abierta a futuras extensiones como:

- Integración de pagos reales con Stripe o PayPal.

- Sistema de notificaciones push o alertas.
- Panel de administración para estadísticas y moderación.
- Traducción multilingüe de la plataforma.

En conclusión, AdventureShare es una plataforma funcional que sienta unas bases sólidas para un proyecto real de viajes compartidos. Su diseño y arquitectura permiten evolucionarlo fácilmente y adaptarlo a nuevas necesidades, lo que abre la posibilidad de continuar desarrollándolo tras la entrega del TFG.

8. Despliegue y pruebas

Plan de Despliegue

Para garantizar que AdventureShare pueda estar operativo y accesible al público, se ha diseñado un plan de despliegue que contempla tanto la preparación del entorno de producción como la posterior verificación del sistema tras su puesta en marcha.

Preparación del Entorno de Producción

- Configuración del servidor web (por ejemplo, Railway o Vercel para el frontend, y Render para el backend).
- Instalación y configuración de PostgreSQL como sistema de gestión de base de datos.
- Implementación de variables de entorno para proteger datos sensibles como claves secretas o tokens.
- Activación del modo producción en Django y React.
- Configuración de CORS, HTTPS y cabeceras de seguridad en el servidor backend.

Despliegue de la Aplicación

- Subida del frontend a Vercel o Netlify desde el repositorio GitHub.
- Despliegue del backend en Render (con `gunicorn` y `whitenoise`).
- Migración de la base de datos usando los comandos de Django (`makemigrations`, `migrate`).
- Configuración de los ficheros `settings.py` y `.env` para entorno de producción.

Verificación Post-Despliegue

- Pruebas manuales de las principales funcionalidades (registro, login, creación de viaje...).
- Verificación de endpoints con herramientas como Postman.
- Confirmación de que las imágenes y estilos se sirven correctamente desde el frontend.

Monitorización y Registro

- Configuración de logs en el backend para registrar errores y peticiones fallidas.
- Uso de paneles de control del proveedor (Render, Vercel) para analizar rendimiento y uso.
- Establecimiento de alertas en caso de errores críticos.

Plan de Respuesta a Incidentes

- Mantenimiento de copias locales del código y de la base de datos.
- Creación de documentación básica de primeros auxilios técnicos.
- Procedimiento de rollback en caso de fallos graves tras despliegue.

Plan de Pruebas

Para verificar que todas las funcionalidades de AdventureShare operan correctamente, se han diseñado pruebas de tipo caja negra que comprueban los resultados esperados para los diferentes casos de uso, sin necesidad de conocer el código fuente interno.

Caso de Uso: Registrarse en la Plataforma

Pruebas de Interfaz de Usuario

- Verificar que la página de registro se carga correctamente.
- Comprobar la validación de campos (email, contraseña, nombre de usuario).
- Asegurar que se muestran mensajes de éxito o error tras enviar el formulario.

Pruebas Funcionales

- Registrar un usuario válido y comprobar que se crea en la base de datos.
- Probar con datos inválidos para asegurar que no se permite el registro.
- Verificar que no se puede registrar un email ya usado.

Pruebas de Seguridad

- Verificar que la contraseña se guarda cifrada en la base de datos (hash).
- Probar entradas maliciosas como inyecciones SQL o XSS en el formulario.

Pruebas de Rendimiento

- Simular varios registros simultáneos y medir el tiempo de respuesta.
- Comprobar que la aplicación responde correctamente bajo carga moderada.

Preparación del Entorno de Producción

Se seleccionó un entorno de despliegue en la nube (por ejemplo, Railway o Render) debido a su facilidad de integración con proyectos Django y soporte nativo para bases de datos PostgreSQL.

El servidor fue configurado para aceptar peticiones HTTPs utilizando certificados SSL auto-firmados inicialmente, con la posibilidad de migrar a certificados de Let's Encrypt en producción.

Se planificó una estrategia de backup automático de la base de datos para asegurar la recuperación ante errores o caídas del sistema. Este backup puede realizarse diariamente usando scripts automatizados.

Además, se configuró un sistema de logs para registrar los eventos del servidor y la aplicación, lo que facilita el análisis de problemas en producción.

Despliegue de la Aplicación

El código fue subido desde GitHub y desplegado directamente a la plataforma de hosting seleccionada.

Se aplicaron migraciones de Django para crear todas las tablas necesarias en la base de datos PostgreSQL.

Se crearon variables de entorno en el panel de control del servidor para configurar claves secretas, credenciales de base de datos y modo de ejecución (DEBUG=False en producción).

Verificación Post-Despliegue

Se realizaron comprobaciones básicas del funcionamiento general de la web tras su despliegue para asegurar que todas las rutas respondían correctamente.

Se accedió a cada funcionalidad clave (registro, inicio de sesión, creación de viajes, reservas, mensajes) para validar su correcto funcionamiento.

Monitorización y Registro

Se habilitó la consola de logs del proveedor de hosting para capturar errores del servidor.

También se utilizó un middleware personalizado de Django para registrar eventos críticos como errores 500, inicios de sesión fallidos o intentos de acceso no autorizado.

Como opción a futuro, se plantea la integración con servicios como Sentry o LogRocket para obtener trazabilidad en tiempo real de errores de frontend y backend.

Plan de Respuestas a Incidentes

Se define un procedimiento en caso de caída del sistema, que incluye restaurar la base de datos desde el último backup, reiniciar los servicios del servidor y revisar los logs en busca del error.

Además, se plantea habilitar un canal de soporte para los usuarios (correo o formulario) con el objetivo de recibir reportes de incidencias y mejorar continuamente el sistema.

9. Conclusiones

Durante el desarrollo del proyecto AdventureShare, se han alcanzado una gran parte de los objetivos inicialmente propuestos, aunque también se han identificado ciertas limitaciones debido al tiempo disponible, la complejidad técnica y los recursos. A continuación, se exponen los logros principales del proyecto, las diferencias con respecto a los objetivos iniciales, las dificultades encontradas durante el proceso y una reflexión personal sobre la experiencia de desarrollo.

Comparativa con los Objetivos Generales

- Se ha logrado desarrollar una plataforma web funcional que permite organizar y reservar viajes compartidos de forma segura y estructurada.
- La experiencia de usuario es moderna, intuitiva y adaptable, desarrollada con tecnologías como React y Tailwind CSS.
- La autenticación JWT y el sistema de valoraciones han proporcionado una base sólida para fomentar la confianza entre usuarios.
- Aunque no se ha integrado un sistema de pagos real como Stripe o PayPal, se ha implementado un simulador que cumple su función durante esta etapa del proyecto.
- La escalabilidad del sistema ha sido tenida en cuenta desde la estructura modular del backend y la separación de responsabilidades entre frontend y backend.

Comparativa con los Objetivos Específicos

- El backend ha sido implementado completamente en Django con Django REST Framework, cumpliendo la arquitectura REST.
- Se ha creado una API funcional que comunica correctamente con el frontend React.
- El diseño responsive ha sido implementado con éxito, aunque no se utilizaron Google Maps ni pasarelas de pago reales por cuestiones de tiempo y simplicidad.
- El sistema de usuarios, autenticación, roles y valoraciones está operativo.
- La mensajería entre usuarios y el sistema de reservas también han sido completados con éxito, así como filtros básicos para los viajes.
- Algunas funcionalidades como notificaciones en tiempo real o filtros avanzados no llegaron a desarrollarse por limitaciones de tiempo.

Dificultades durante el Desarrollo

Durante el proceso de desarrollo surgieron varios desafíos, entre ellos la integración de múltiples tecnologías, la gestión de errores en tiempo real, y la necesidad de depurar y entender comportamientos inesperados tanto en frontend como en backend. Además, la compresión de conceptos complejos como el manejo de tokens JWT, relaciones entre modelos en Django y el enrutamiento condicional en React requirió una curva de aprendizaje significativa.

Reflexión Personal

La experiencia global del desarrollo del proyecto ha sido enriquecedora, tanto a nivel técnico como personal. A pesar de las dificultades encontradas y de no haber podido implementar absolutamente todos los objetivos iniciales, AdventureShare representa una base sólida sobre la que continuar trabajando. Este proyecto me ha permitido afianzar conocimientos clave, enfrentarme a problemas reales de desarrollo web, y reforzar mis habilidades de diseño, programación y resolución de problemas. Estoy satisfecho con el resultado final, y considero que con tiempo adicional y experiencia profesional, esta plataforma podría evolucionar hacia un producto completamente funcional en el futuro.

10. Vías futuras

A pesar de haber alcanzado muchos de los objetivos iniciales propuestos en este proyecto, existen múltiples líneas de mejora y funcionalidades adicionales que podrían implementarse en futuras versiones de la plataforma AdventureShare. Estas mejoras no solo enriquecerían la experiencia de usuario, sino que también acercarían el producto a un modelo comercial viable, preparado para ser lanzado al público general.

1. Integración de pasarelas de pago reales (PayPal, Stripe)

En la versión actual del proyecto, el sistema de pagos ha sido simulado en el frontend por motivos prácticos y de seguridad, ya que integrar un sistema real implicaría complejidades legales, fiscales y técnicas. Para integrar pagos reales sería necesario:

- Registrar la plataforma como entidad legal con una cuenta empresarial en la pasarela de pago.
- Configurar credenciales seguras (API keys) y entorno sandbox para pruebas.
- Decidir el destinatario del dinero: ¿va al organizador o a una cuenta central? ¿Cómo se distribuye?
- Implementar medidas antifraude y de devolución/reembolso según legislación vigente.
- Probar todo el proceso sin incurrir en transacciones económicas reales durante el desarrollo.

Por ello, se recomienda integrar PayPal o Stripe en una futura versión una vez que el proyecto esté más consolidado, posiblemente como startup.

2. Mapa interactivo

Otra funcionalidad interesante sería la incorporación de un mapa interactivo con Google Maps o Leaflet.js que permita visualizar:

- La ubicación exacta de cada viaje o ruta.
- Los kilómetros totales a recorrer o puntos de encuentro.
- Filtros visuales por zona geográfica.

Esto aportaría una mayor claridad y atractivo visual, especialmente para los usuarios más aventureros.

3. Clasificación por categorías

Con el aumento de viajes disponibles en la plataforma, se vuelve útil ofrecer filtros avanzados por categorías como:

- Tipo de actividad: senderismo, escalada, surf, camping…
- Zona geográfica: viajes por España, por Europa, por provincias, etc.
- Nivel de dificultad o duración de la aventura.

Esto mejoraría la usabilidad y permitiría a los usuarios encontrar viajes que se ajusten mejor a sus intereses.

4. Registro con Google

Facilitar el proceso de registro mediante OAuth (inicio de sesión con Google) es otra mejora habitual en aplicaciones modernas. Esta integración permite:

- Reducir barreras de entrada para nuevos usuarios.
- Aumentar la seguridad mediante la verificación de identidad con Google.
- Automatizar parte del proceso de autenticación.

Se podría realizar fácilmente utilizando paquetes como django-allauth o Firebase Auth en caso de cambiar de enfoque tecnológico.

5. Cancelación de reservas por parte del usuario

Actualmente, el sistema permite a los organizadores aceptar o rechazar reservas, pero no a los usuarios cancelar voluntariamente su participación una vez aceptados. Incluir esta funcionalidad sería fundamental para:

- Mejorar la experiencia de usuario.
- Evitar problemas de última hora o ausencias no comunicadas.
- Permitir al organizador gestionar mejor la disponibilidad de plazas.

Esta opción requeriría también definir una política de cancelación y posibles penalizaciones si el sistema se vincula a pagos reales.

6. Sistema de notificaciones en tiempo real

Actualmente las interacciones se comunican por mensajes internos. Una mejora importante sería integrar un sistema de notificaciones en tiempo real (con WebSockets o Firebase) para avisos como:

- Nueva reserva recibida.
- Confirmación o cancelación de viajes.
- Nuevos mensajes privados.

7. Aplicación móvil nativa

A largo plazo, y si el proyecto evoluciona positivamente, podría desarrollarse una app móvil nativa (iOS y Android) que replique o amplíe la funcionalidad web. Esto abriría el proyecto a un público más amplio y permitiría aprovechar funciones específicas de los móviles como geolocalización o notificaciones push.

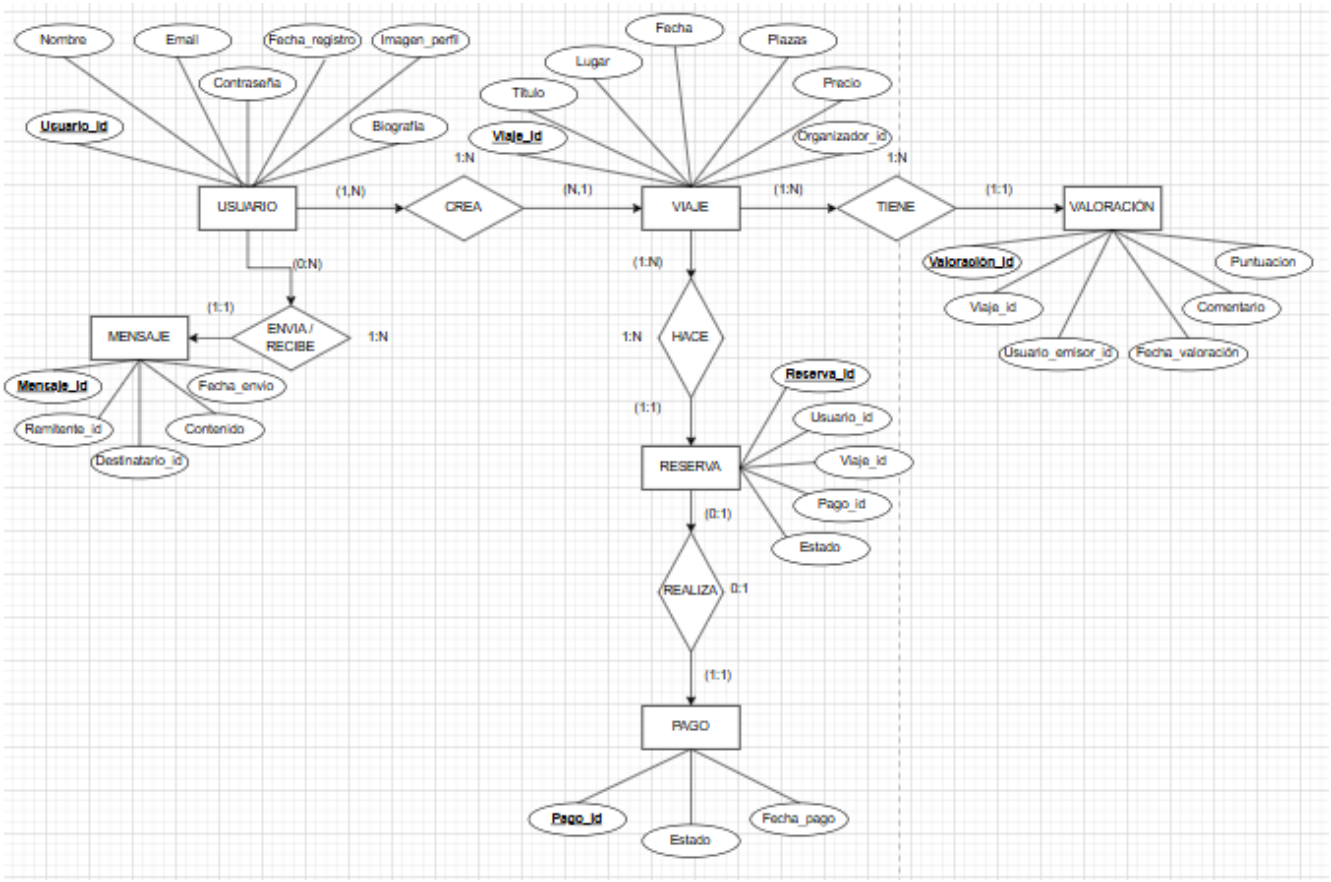
En resumen, AdventureShare tiene una base sólida, pero mucho potencial de crecimiento. Las futuras mejoras propuestas permitirían consolidar la plataforma, profesionalizarla y escalarla como un producto comercial real. La implementación de estas vías futuras dependerá de factores como la acogida del proyecto, la disponibilidad de recursos y el enfoque estratégico que se adopte tras la entrega del TFG.

11. Bibliografía/Webgrafía

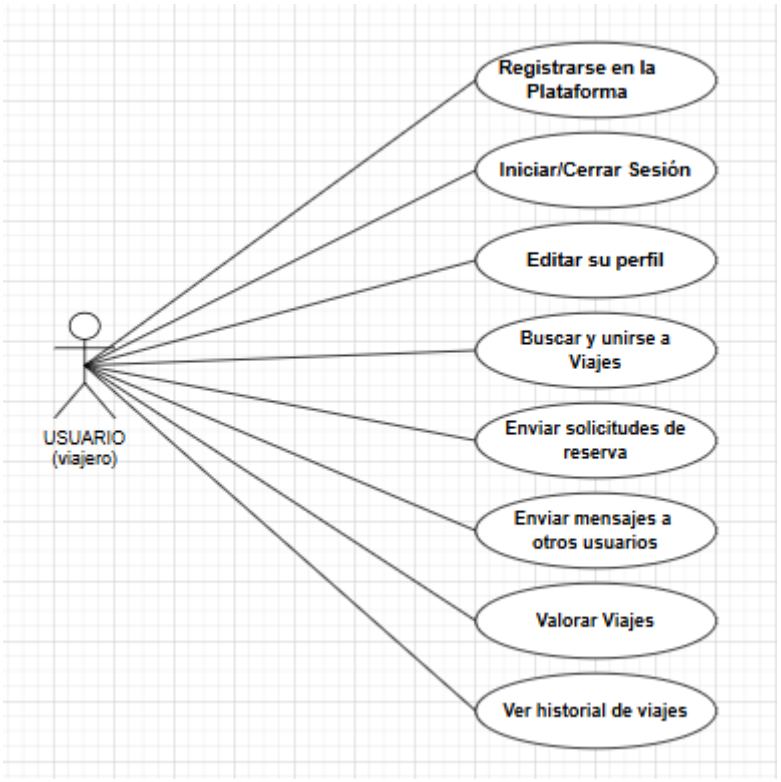
- Django Software Foundation. (s.f.). Django. https://www.djangoproject.com/
- Tom Christie. (s.f.). Django REST framework. https://www.django-rest-framework.org/
- Meta. (s.f.). React. https://react.dev/
- Axios. (s.f.). Axios - Promise based HTTP client. https://axios-http.com/
- Remix Software. (s.f.). React Router. https://reactrouter.com/
- Auth0. (2023). Introduction to JSON Web Tokens. https://jwt.io/introduction/
- PostgreSQL Global Development Group. (s.f.). PostgreSQL: The world's most advanced open source database. https://www.postgresql.org/
- Tailwind Labs. (s.f.). Tailwind CSS. https://tailwindcss.com/
- Mozilla Developer Network (MDN). (s.f.). JavaScript documentation. https://developer.mozilla.org/en-US/docs/Web/JavaScript
- DigitalOcean. (2021). How To Build a RESTful API with Django REST Framework. https://www.digitalocean.com/community/tutorials/build-a-restful-api-with-django-rest-framework
- GeeksforGeeks. (2022). Introduction to Django. https://www.geeksforgeeks.org/django-introduction/
- FreeCodeCamp. (2023). Learn Django - Full Course for Beginners. https://www.freecodecamp.org/news/learn-django-python-web-development/
- CSS-Tricks. (2023). A Complete Guide to Flexbox. https://css-tricks.com/snippets/css/a-guide-to-flexbox/
- Real Python. (2022). Django REST Framework – Full Walkthrough. https://realpython.com/django-rest-framework-quick-start/
- W3Schools. (s.f.). React Tutorial. https://www.w3schools.com/react/
- Traversy Media. (s.f.). Full Stack Django and React. https://www.youtube.com/watch?v=Uyei2iDA4Hs
- Programación en Español. (2022). Curso de Django desde cero. https://www.youtube.com/watch?v=7XO1AzwkPPE&list=PLU8oAlHdN5BmfvwxFO7HdPciOCmmYneAB
- Dev.to. (2023). JWT Authentication with Django REST Framework. https://dev.to/techschoolguru/jwt-authentication-with-django-rest-framework-2c88
- Stack Overflow. (s.f.). https://stackoverflow.com/

- Documentación de Create React App https://create-react-app.dev/docs/getting-started/
- Guías de uso de **GitHub** y Git para subir y versionar el proyecto https://docs.github.com/en/get-started
- Uso de useContext, useEffect, y otros hooks de React: https://reactjs.org/docs/hooks-overview.html

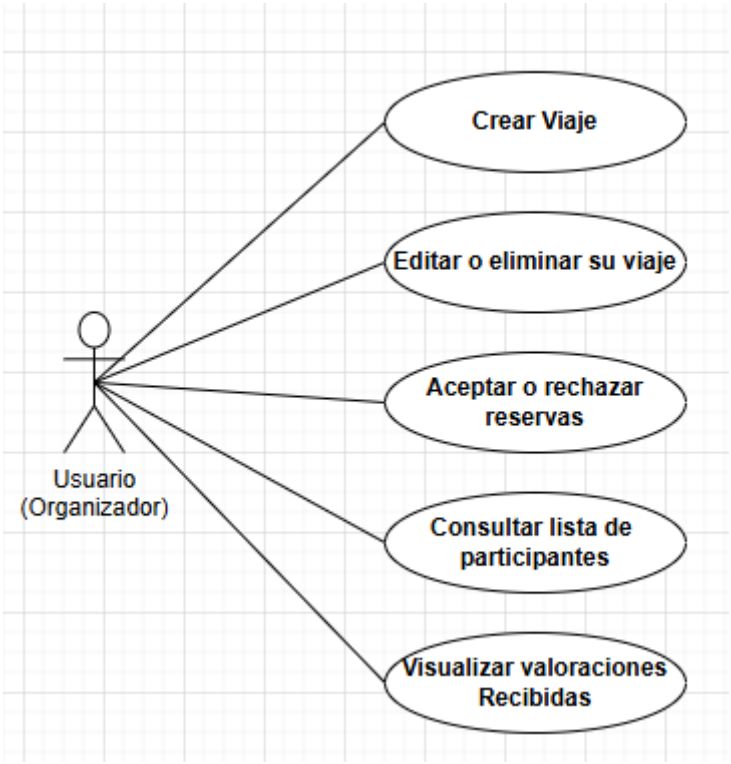
12. Anexos



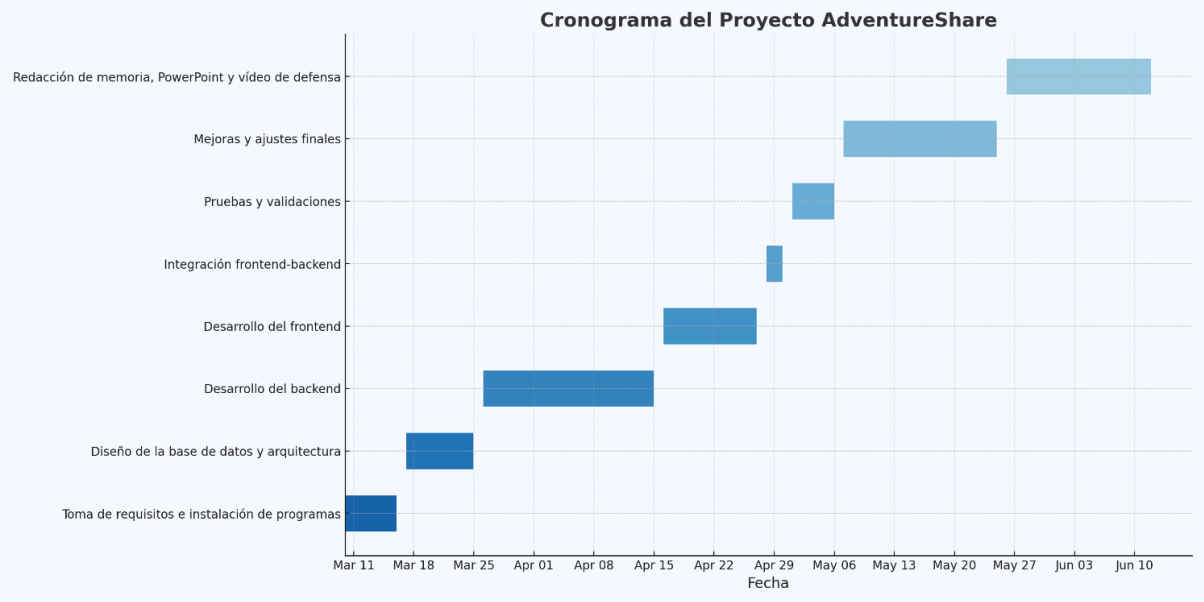
Draw.io – Diagrama Entidad - Relación



Draw.io – Diagrama de clases Usuario (viajero)



Draw.io – Diagrama de Clases Usuario (administrador)



Canva – Diagrama de Gantt

1.1. Manual de Usuario

Vamos a dividir el manual en 6 puntos:

- **Acceso inicial e inicio/cierre de sesión:** En el acceso inicial a la web, el primer paso será poner en nuestro nuestro navegador la siguiente url " https://localhost:3001 ". Una vez dentro pulsa clic en registrarse y introduce un nombre, un email y una contraseña de al menos 8 caracteres. Y pulsa crear cuenta o la tecla enter. Luego para iniciar sesion, el proceso es el mismo pero tan solo introduciendo el usuario y la contraseña. Por último para cerrar sesión tienes que darle al logo de tu usuario y ahí te aparecerá el icono de cerrar sesión, en el que tendrás que clicar.
- **Modificaciones en "Mi perfil":** Sin embargo si no cerramos sesión al pulsar en nuestro logo de mi perfil nos aparecerá otra opción que será "ver perfil", si pulsamos ahí podremos modificar la foto, el nombre, la contraseña y la biografia de nuestro perfil.
- **Exploración de Viajes:** En la pantalla principal aparecerán todos los viajes creados por usuarios, en los cuales, podrás ver una imagen del lugar, el precio, el número de participantes y la fecha de este, luego si un viaje cuadra con lo que buscas podrás reservar como viajero. También puedes usar la barra de busqueda para filtrar por un lugar en concreto o incluso puedes crear tu mismo un viaje como administrador añadiendo tu mismo toda la información del viaje.
- **Solicitud, gestión de reservas y pagos:** Si un viajero quiere reservar un viaje debe pulsar en el botón de reserva dentro de el viaje que quiera reservar. Una vez hecho eso al mismo viajero le aparecerá el viaje en sus reservas donde el tendrá la opción

de pagar (en este caso los pagos son simulados). Una vez pague al administrador le aparecerá una solicitud de ese viajero queriendo unirse a viaje. El administrador podrá aceptarlo o rechazarlo. Mientras no la acepta o rechaza al viajero le aparecerá como pendiente.

- **Valoraciones:** Tras la fecha del viaje, en mis reservas o en la misma pantalla principal, puedes meterte al viaje y valorarlo con una puntuación de 1 a 5 estrellas y un comentario.
- **Mensajes privados:** Podemos enviar mensajes a cualquier usuario. Tanto metiéndonos en el perfil de usuario de la persona a la cual le queremos enviar un mensaje, como entrando en un viaje y dándole a enviar un mensaje al administrador.

1.2. Manual de Instalación

Requisitos Previos

1. Git (para clonar el repositorio)
2. Python 3.10+
3. Node.js 16+ y npm 8+
4. PostgreSQL 14+
5. Visual Studio Code u otro IDE.

Como arrancar el backend

1. Descomprime el archivo .zip del proyecto.
2. Pulsa simultaneamente la tecla windows + R para abrir el cmd.
3. Una vez abierto escribe “cd C:\Users\Propietario\mi-trabajo-final” (recuerda no ponerlo con comillas y guardar el archivo justo en la carpeta propietario para que funcione.
4. Justo después tienes que crear el virtualenv poniendo “python -m venv venv” y justo después arrancarlo con “venv\Scripts\activate”.
5. Luego tienes que poner “cd backend” para ir a la carpeta donde se encuentra tu backend.
6. Una vez ahí tienes que instalar dos cosas poniendo los siguientes comandos: “pip install --upgrade pip”, “pip install -r requirements.txt”.
7. Luego tienes que crear la base de dato en tu sistema para que funcione, esto se hace mediante el comando “createdb -U tu_usuario adventureshare” y luego poniendo y modificando como el propio usuario desee los siguientes datos:

   set SECRET_KEY=tu_clave_secreta

```
set DEBUG=True
set DB_NAME=adventureshare
set DB_USER=tu_usuario
set DB_PASSWORD=tu_password
set DB_HOST=localhost
set DB_PORT=5432
```

8. Y una vez hecho esto ponemos estos 4 comandos para añadir las migraciones, crear el usuario administrador de la base de datos y arrancar nuestro backend:

```
python manage.py makemigrations
python manage.py migrate
python manage.py createsuperuser
python manage.py runserver
```

Como arrancar el frontend

1. Abrir el cmd pulsando simnultaneamente el boton windows + R y una vez abierto el cmd poner este comando: “C:\Users\Propietario\mi-trabajo-final\frontend”
2. Luego ponemos “npm start” y ya estaria nuestro frontend en funcionamiento

iLERNA.