



Universidad Politécnica
de Madrid



**Escuela Técnica Superior de
Ingenieros Informáticos**

Grado en Ingeniería Informática

Trabajo Fin de Grado

**Optimización de una Aplicación Móvil
para una Aplicación de Búsqueda del
Tesoro Orientada a Personas Mayores**

Autor: Sonia Gallego Trapero

Tutor(a): Raúl Alonso Calvo

Madrid, Junio 2025

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado

Grado en Ingeniería Informática

Título: Optimización de una Aplicación Móvil para una Aplicación de Búsqueda del Tesoro Orientada a Personas Mayores

Junio 2025

Autor: Sonia Gallego Trapero

Tutor:

Raúl Alonso Calvo

Departamento de Lenguajes, Sistemas Informáticos e Ingeniería de Software

ETSI Informáticos

Universidad Politécnica de Madrid

Resumen

Este Trabajo Fin de Grado propone el desarrollo de una aplicación móvil orientada a la realización de búsquedas del tesoro, entendidas como rutas a pie con puntos clave en los que los participantes deben registrar su paso mediante la lectura de dispositivos NFC. La finalidad del sistema es fomentar la actividad física y cognitiva, así como la socialización en personas mayores, promoviendo un envejecimiento activo y una mejora integral en su calidad de vida.

El proyecto está tutorizado por Raúl Alonso Calvo y continúa el trabajo iniciado en un TFG del curso anterior, reutilizando la base de datos y la API que dan soporte a las funcionalidades de la nueva aplicación. La solución está pensada para su implementación en asociaciones y centros de mayores, donde se establecen dos roles diferenciados de usuario: jugadores y administradores. Los administradores son los encargados de crear y configurar las actividades —incluyendo la información, la ruta, los dispositivos NFC asociados y la asignación de participantes—, además de poder monitorizar el desarrollo de las actividades en tiempo real. Por su parte, los jugadores acceden a las rutas asignadas desde la aplicación, las siguen mediante un mapa interactivo, y registran su avance a través de interacciones NFC, con soporte adicional mediante geolocalización GPS.

La motivación del proyecto se basa en la evidencia científica que relaciona el sedentarismo en personas mayores con un aumento del riesgo de enfermedades crónicas, deterioro cognitivo, aislamiento social y disminución del bienestar emocional. Frente a ello, el ejercicio físico regular se asocia a múltiples beneficios: mejora de la salud cardiovascular, preservación de la masa muscular y ósea, estimulación de funciones cognitivas como la memoria y la atención, y reducción de síntomas depresivos y de ansiedad. A pesar de estos beneficios, un gran porcentaje de la población mayor no alcanza los niveles recomendados de actividad física. En este sentido, el proyecto busca ser una herramienta tecnológica accesible, lúdica y motivadora que ayude a revertir esta situación.

El desarrollo de la aplicación se ha llevado a cabo siguiendo una metodología estructurada por fases: análisis y definición de requisitos funcionales, diseño de alto nivel de la arquitectura y las interfaces (con atención especial a la usabilidad para personas mayores), diseño de bajo nivel, integración con las herramientas heredadas del proyecto anterior, implementación técnica de la solución y realización de pruebas funcionales.

Además de su impacto positivo a nivel individual y social, la propuesta contribuye a los Objetivos de Desarrollo Sostenible (ODS), en particular al ODS 3 (Salud y bienestar), al promover hábitos saludables y mejorar la calidad de vida de las personas mayores, y al ODS 10 (Reducción de las desigualdades), al facilitar su inclusión en el entorno digital mediante una tecnología accesible y adaptada a sus necesidades.

Abstract

This Final Degree Project proposes the development of a mobile application designed for organizing treasure hunts, understood as walking routes with key points where participants must register their presence by scanning NFC devices. The main objective of the system is to promote physical and cognitive activity, as well as social interaction among older adults, fostering active aging and improving their overall quality of life.

The project is supervised by Raúl Alonso Calvo and continues the work initiated in a Final Degree Project from the previous academic year, reusing and extending existing components such as the database and API that support the new application's functionalities. The solution is intended for use in community centers and associations for the elderly and defines two user roles: players and administrators. Administrators are responsible for creating and configuring the activities — including information, route, associated NFC devices, and participant assignments — as well as monitoring progress in real time. Players can participate in the assigned routes, follow them through an interactive map, and record their progress using NFC interactions and GPS-based location tracking.

The motivation behind the project lies in the well-documented negative effects of physical inactivity in older adults, including increased risk of chronic diseases, cognitive decline, social isolation, and emotional distress. Regular physical activity, on the other hand, is associated with multiple benefits: improved cardiovascular health, preservation of muscle and bone mass, enhanced memory and attention, and reduced symptoms of depression and anxiety. However, a large part of the elderly population does not meet the recommended levels of physical activity. In this context, the proposed application offers an accessible, engaging, and motivating technological tool to help reverse this trend.

The application was developed following a structured methodology, comprising several phases: analysis and definition of functional requirements, high-level system and interface design (with special attention to usability for older adults), low-level design, integration with legacy components, technical implementation, and functional testing.

Beyond its individual and social impact, the project also contributes to the Sustainable Development Goals (SDGs), particularly SDG 3 (Good Health and Well-being), by encouraging healthy habits and enhancing the quality of life of older people, and SDG 10 (Reduced Inequalities), by promoting digital inclusion through accessible and age-appropriate technology.

Tabla de contenidos

1	Introducción.....	1
1.1	Motivación.....	1
1.2	Objetivos	2
2	Estado de la cuestión.....	4
2.1	Fundamentos teóricos de diseño para personas mayores	4
2.2	Tecnologías empleadas	6
3	Diseño	10
3.1	Requisitos	10
3.2	Diseño de alto nivel	17
3.2.1	Diagrama de flujo	17
3.2.2	Diseño de interfaces de usuario	19
3.2.3	Diagrama de arquitectura.....	29
3.3	Diseño de bajo nivel	30
3.3.1	Diseño de modelos de datos	31
3.3.2	Diagrama de secuencia	33
4	Implementación.....	42
4.1.1	Cambios	42
4.1.2	Organización del código	43
4.1.3	Configuración	44
4.1.4	Diagramas UML.....	45
4.1.5	Conexiones con la API.....	47
4.1.6	Uso de Open Street Map (OSMdroid).....	49
4.1.7	Uso de OpenRouteService	50
4.1.8	Documentación del código	51
5	Pruebas y análisis de resultados.....	51
6	Resultados y conclusiones	79
7	Análisis de Impacto	82
8	Bibliografía	84
9	Anexo 1: Informe de Turnitin.....	86

1 Introducción

En este apartado se aborda el contexto que motiva el desarrollo de una aplicación móvil orientada a promover el ejercicio físico en personas mayores, así como los objetivos específicos del proyecto. Se plantea una solución tecnológica que fomente la actividad física y la socialización mediante una experiencia lúdica y accesible.

1.1 Motivación

La inactividad física tiene efectos negativos comprobados en la salud física, cognitiva y emocional de las personas mayores. En el plano físico, el sedentarismo se asocia con un mayor riesgo de desarrollar enfermedades crónicas y con una mayor mortalidad. Según la Organización Mundial de la Salud (OMS), la inactividad es uno de los principales factores de riesgo de muerte por enfermedades no transmisibles, y las personas físicamente inactivas presentan un riesgo de mortalidad entre un 20 % y un 30 % superior al de aquellas suficientemente activas [1].

La vida sedentaria contribuye al desarrollo de enfermedades como cardiopatías, hipertensión, diabetes tipo 2, osteoporosis e incluso ciertos tipos de cáncer [2][3]. En personas mayores, la falta de ejercicio acelera la pérdida de masa muscular (sarcopenia) y la disminución de densidad ósea, lo que se traduce en menor fuerza, equilibrio y, en consecuencia, un mayor riesgo de caídas y fracturas. Por el contrario, mantenerse físicamente activo en la vejez ayuda a prevenir la fragilidad, reduce significativamente el riesgo de lesiones asociadas y mejora el control de factores de riesgo como la presión arterial, la glucemia o el perfil lipídico. Todo ello se asocia con una mayor esperanza de vida [1].

En cuanto a la salud cognitiva, diversos estudios demuestran que la actividad física regular protege el cerebro frente al envejecimiento. El sedentarismo se ha vinculado con un mayor riesgo de deterioro cognitivo y demencia [3][4]. Un meta-análisis indicó que los adultos mayores físicamente activos tenían un 38 % menos de riesgo de deterioro cognitivo en años posteriores, y con actividad moderada la reducción del riesgo fue del 35 % [5]. El ejercicio, especialmente el aeróbico y de fuerza, mejora la irrigación cerebral, favorece la plasticidad neuronal y puede retrasar la aparición de enfermedades neurodegenerativas. Asimismo, se han observado mejoras en funciones ejecutivas como la memoria de trabajo, la atención y la planificación [1].

A nivel emocional, la actividad física se asocia con una disminución de los síntomas de depresión y ansiedad, y con una mejora del bienestar general. El ejercicio estimula la liberación de endorfinas y otros neurotransmisores implicados en la regulación del estado de ánimo. Un estudio reciente con más de 3.200 personas mayores de 60 años evidenció una asociación positiva entre ejercicio físico y salud mental, mediada en parte por el fortalecimiento de los vínculos sociales [6]. En contraste, el sedentarismo se relaciona con mayor incidencia de trastornos del sueño, sensación de soledad y menor calidad de vida emocional.

A pesar de la evidencia sobre sus beneficios, la práctica de actividad física tiende a disminuir con la edad. Datos de la OMS indican que alrededor del 31 % de los adultos en el mundo —unos 1.800 millones de personas— no alcanzan los niveles recomendados de actividad física, siendo esta proporción aún mayor en la población mayor de 60 años. Por ello, combatir el sedentarismo en la tercera

edad se ha convertido en una prioridad de salud pública, ya que contribuye no solo a prevenir enfermedades, sino también a mejorar la calidad y la esperanza de vida. De hecho, las autoridades sanitarias internacionales, como la OMS, se han marcado objetivos como reducir en un 15 % la prevalencia del sedentarismo para el año 2030 (respecto a los niveles de 2010). Aunque el cumplimiento de esta meta aún se ve lejano, refleja el reconocimiento de la urgencia de intervenir en este ámbito. Incluso pequeños aumentos en los niveles de actividad pueden generar beneficios sustanciales para la salud de los adultos mayores. [1]

En este contexto, resulta esencial impulsar propuestas que fomenten la actividad física de forma atractiva, accesible y adaptada a las personas mayores. La aplicación desarrollada en este proyecto busca precisamente contribuir a este objetivo mediante la creación de rutas de búsqueda del tesoro en grupo. Esta dinámica no solo promueve el ejercicio a través del paseo, sino que también refuerza la socialización, ayudando a reducir el aislamiento y generando un sentido de pertenencia. Además, incorpora un componente cognitivo, al requerir orientación en el mapa y la identificación de puntos clave, lo que estimula la mente.

El propósito es ofrecer una herramienta que favorezca un envejecimiento activo desde una perspectiva integral: una aplicación que no solo mejore la condición física, sino que también nutra el bienestar emocional, fortalezca los vínculos sociales y mantenga la mente en forma. El componente comunitario crea un círculo virtuoso en el que las personas mayores se sienten apoyadas, valoradas y útiles dentro del grupo, lo que favorece la constancia y el disfrute de la actividad. En definitiva, se trata de una iniciativa orientada a mejorar la calidad de vida en la vejez, manteniendo tanto el cuerpo como el espíritu activos.

1.2 Objetivos

Este proyecto da continuidad al trabajo iniciado en un Trabajo de Fin de Grado del curso anterior. Parte de una base ya desarrollada, que incluye una base de datos, una API para la gestión de la información y una implementación inicial de la aplicación móvil, aún sin finalizar.

La aplicación tiene como finalidad la realización de búsquedas del tesoro, concebidas como rutas a pie con puntos clave que los participantes deben registrar mediante dispositivos NFC.

Se contemplan dos roles de usuario: jugador y administrador. Los administradores podrán crear actividades —definiendo la información, la ruta y los dispositivos NFC asociados—, registrar a los jugadores en dichas actividades y monitorizar su progreso en tiempo real mientras la actividad esté en curso. Por su parte, los jugadores podrán participar en las actividades asignadas, siguiendo la ruta a través del mapa e interactuando con los dispositivos NFC situados a lo largo del recorrido. Los trayectos se registrarán mediante la localización GPS del dispositivo.

El sistema se integrará con un servicio web REST previamente desarrollado, encargado de almacenar tanto los datos relativos a las actividades creadas como la información sobre la participación de los usuarios. Asimismo, la aplicación contará con una interfaz atractiva, diseñada específicamente para personas mayores —público objetivo en el rol de jugador—, teniendo en cuenta principios de accesibilidad y usabilidad.

Para llevar a cabo este desarrollo, se plantean los siguientes objetivos principales:

- Implementar una aplicación móvil que permita la recogida de datos y la navegación durante las búsquedas del tesoro.
- Establecer la conexión con el servicio web ya existente.
- Desarrollar los métodos y servicios necesarios para almacenar en el servidor los datos generados.
- Realizar el despliegue y pruebas del sistema.

2 Estado de la cuestión

Antes de abordar el desarrollo de la solución propuesta, resulta fundamental comprender el contexto y los principios que la sustentan. El diseño de tecnología orientada a personas mayores presenta una serie de retos específicos que exigen un enfoque centrado en las capacidades, necesidades y hábitos de este colectivo. Conocer los fundamentos teóricos que guían este tipo de diseño no solo permite crear interfaces más accesibles e intuitivas, sino también garantizar que la tecnología realmente aporta valor a sus usuarios.

Asimismo, es importante revisar las tecnologías que se han empleado en el desarrollo del proyecto, ya que estas condicionan tanto las posibilidades técnicas como las decisiones de implementación. Analizar el estado actual de estas herramientas permite situar la solución en un marco tecnológico realista y justificar la elección de cada componente en función de sus capacidades y limitaciones.

Por tanto, este apartado recoge, por un lado, los fundamentos teóricos que orientan el diseño centrado en personas mayores, y por otro, una revisión de las tecnologías utilizadas, aportando una visión completa del punto de partida sobre el que se ha construido este trabajo.

2.1 Fundamentos teóricos de diseño para personas mayores

Uniendo los conocimientos adquiridos a lo largo de la carrera, especialmente en la asignatura de Interacción Persona-Ordenador, con la información contrastada de diversas fuentes [7][8][9], se identifican algunos principios clave para el diseño de interfaces de usuario accesibles y orientadas a personas mayores:

- **Simplicidad en el diseño:** Un diseño simple y limpio es esencial para reducir la carga cognitiva de los usuarios mayores. A medida que se envejece, es común que disminuya la velocidad de procesamiento de la información y que aumente la dificultad para manejar entornos digitales complejos. Eliminar elementos innecesarios —como menús ocultos, animaciones excesivas o información irrelevante— ayuda a que la navegación sea más intuitiva y comprensible. Una interfaz clara, con una jerarquía visual bien definida y una terminología sencilla, permite que los usuarios se concentren en lo que realmente importa y evita que se sientan abrumados o confundidos. Esto favorece la autonomía del usuario, mejora su experiencia general y reduce la necesidad de asistencia externa para utilizar la aplicación.
- **Controles más grandes y espaciados:** Muchas personas mayores experimentan una reducción en la precisión motora fina y pueden tener dificultades visuales relacionadas con la edad, como presbicia o pérdida de agudeza visual. Por ello, es crucial que los elementos interactivos —botones, enlaces, iconos— tengan un tamaño adecuado y estén suficientemente espaciados entre sí. Esto facilita su identificación y selección, reduce errores de interacción (como pulsar el botón equivocado) y mejora la accesibilidad general. Unos controles grandes y bien distribuidos también disminuyen el esfuerzo físico necesario para interactuar con la interfaz, lo cual resulta especialmente útil para

personas con artritis, temblores u otras limitaciones en la movilidad de las manos.

Además de estos principios básicos, hay otros aspectos igualmente importantes a tener en cuenta:

- **Navegación sencilla:** La navegación debe ser directa y enfocada, reduciendo la cantidad de caminos posibles dentro de la aplicación. Diseñar pantallas centradas en una sola tarea ayuda a que los usuarios mantengan la atención. Además, siempre debe existir una opción visible y accesible para retroceder o cancelar una acción, lo cual brinda una sensación de control y seguridad durante el uso.
- **Reducir la complejidad cognitiva:** Es fundamental utilizar un lenguaje claro y familiar y mantener los mensajes breves y directos. Esto facilita la comprensión de las tareas a realizar.
- **Iconos representativos y acompañados de texto:** Los iconos deben ser fácilmente reconocibles y coherentes con su función. No se debe depender únicamente de su interpretación visual; deben ir siempre acompañados de texto que indique su función. Esto evita malentendidos y mejora la accesibilidad para usuarios con menor familiaridad con símbolos digitales.
- **Evitar situar controles muy cerca de los bordes:** Situar elementos interactivos muy próximos a los bordes de la pantalla puede dificultar su acceso, especialmente para personas con menor precisión motora o que usen fundas en el dispositivo. Evitar esas ubicaciones mejora la ergonomía y facilidad de uso.
- **Claridad en los elementos de control:** Los elementos interactivos deben destacarse visualmente para que el usuario los identifique como tales sin dificultad. Un diseño coherente y señalizadores como sombreados o cambios de color al tocar ayudan a reforzar su función.
- **Tipografía legible y de gran tamaño:** El texto debe estar escrito en fuentes claras, con un tamaño suficiente para facilitar su lectura sin necesidad de forzar la vista. Esto resulta clave para personas con presbicia u otras dificultades visuales comunes con la edad.
- **Alto contraste:** Mantener una relación adecuada de contraste entre el texto o elementos gráficos y el fondo mejora la legibilidad. Se recomienda seguir los estándares de accesibilidad definidos por herramientas como *Colour Contrast Analyser*, que establecen:
 - **Contraste mínimo para el texto (AA):** al menos 4.5:1 para texto regular y 3:1 para texto de al menos 18pt o 14pt y negrita. Esto excluye al texto decorativo.
 - **Contraste mejorado para el texto (AAA):** a partir de 7:1 para texto regular y 4.5:1 para texto de 18pt o 14pt y negrita. Esto excluye al texto decorativo.
 - **Contraste de elementos gráficos sin texto (AA):** relación de contraste mínima de 3:1.
- **Interacción sencilla:** Las acciones básicas deben realizarse mediante gestos simples, como tocar la pantalla (tap), evitando gestos complejos que puedan resultar difíciles de ejecutar o recordar. Si se emplean gestos, deben explicarse claramente y ser coherentes a lo largo de la aplicación.
- **Feedback claro y multisensorial:** Cada acción realizada por el usuario debe generar una respuesta perceptible, ya sea visual (como un cambio de color o mensaje), sonora (un tono breve) o táctil (vibración). Esta

retroalimentación inmediata confirma que la acción ha sido registrada y mejora la experiencia de uso.

- **Tiempos de respuesta ampliados:** El sistema debe ofrecer suficiente tiempo para que los usuarios lean, comprendan y actúen ante los mensajes e instrucciones. Evitar límites de tiempo estrictos favorece un uso relajado y sin presión.
- **Entrenamiento inicial:** Para familiarizar a los usuarios con la aplicación, es recomendable ofrecer una breve formación presencial o, en su defecto, un vídeo tutorial claro y guiado. Esto facilita la incorporación y reduce barreras de entrada.
- **Ayuda contextual e instrucciones paso a paso:** Incluir mensajes de ayuda accesibles directamente desde la pantalla o mediante asistentes paso a paso permite a los usuarios resolver dudas en el momento, sin necesidad de abandonar la tarea o buscar información externa.

2.2 Tecnologías empleadas

En este apartado se definen las aplicaciones, servicios web y librerías de código empleadas para el desarrollo del proyecto.

Draw.io:

Herramienta en línea gratuita para la creación de diagramas, que permite diseñar diagramas de flujo, diagramas UML, mapas mentales, esquemas de red, entre otros. Ofrece una interfaz intuitiva y una amplia biblioteca de símbolos y formas personalizables. Es compatible con servicios en la nube, facilitando el trabajo colaborativo. Su uso es común en entornos académicos y profesionales para documentar procesos, estructuras o sistemas. [10]



Figura 2.2.1: Draw.io

Aplicaciones en el proyecto:

- Diseño de diagramas de casos de uso.
- Diseño de diagrama de flujo.
- Diseño de diagrama de arquitectura.
- Diseño de diagramas de modelos de datos.
- Diseño de diagramas de clases UML.

PlantUML:

Herramienta de código abierto que permite generar diagramas a partir de texto simple. Utiliza un lenguaje específico de dominio para crear diagramas UML y otros formatos relacionados con el desarrollo de software, como BPMN, diagramas de bloques, mapas mentales y más. [11]



PlantUML

Figura 2.2.2: PlantUML

Aplicaciones en el proyecto:

- Diseño de diagramas de secuencia.

Figma:

Herramienta de diseño de interfaces y prototipado interactivo que funciona directamente en el navegador. Permite a equipos colaborar en tiempo real, creando diseños de alta fidelidad sin necesidad de instalar software. [12]



Figura 2.2.3: Figma

Aplicaciones en el proyecto:

- Diseño de interfaces (UI).
- Prototipado interactivo sin código.

Colour Contrast Analyser:

Herramienta que permite comprobar la legibilidad del texto evaluando el contraste de color entre elementos gráficos (texto e iconos) y el fondo. Es útil para garantizar la accesibilidad visual de contenidos digitales según los estándares WCAG. Ayuda a diseñadores y desarrolladores a crear interfaces accesibles para personas con baja visión o daltonismo. [13]



Figura 2.2.4: Colour Contrast Analyser

Aplicaciones en el proyecto:

- Análisis de contraste entre texto e iconos y el fondo.
- Visualización de cómo se ve el contenido con distintas deficiencias visuales.

Android Studio:

Entorno de desarrollo integrado (IDE) para crear aplicaciones Android. Está basado en IntelliJ IDEA y proporciona herramientas específicas para diseñar, programar, depurar y probar apps en distintos dispositivos Android. Facilita el desarrollo eficiente gracias a su integración con el SDK de Android y emuladores. [14]



Figura 2.2.5: Android studio

Aplicaciones en el proyecto:

- Desarrollo del código fuente de la aplicación.

Retrofit:

Librería de Android y Java que facilita la comunicación con servicios web mediante el uso de peticiones HTTP. Permite consumir APIs REST de forma sencilla y eficiente, convirtiendo automáticamente las respuestas JSON o XML en objetos Java. Es posible configurar cabeceras, autenticación y manejo de errores, resultando de gran utilidad para aplicaciones que requieren interacción con servidores o servicios externos. [15]

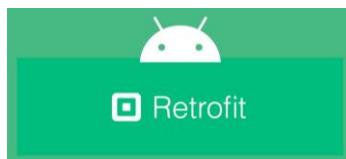


Figura 2.2.6: Retrofit

Aplicaciones en el proyecto:

- Consumo de APIs.

OSMdroid:

Librería de código abierto para Android que permite integrar mapas de OpenStreetMap en aplicaciones móviles. Proporciona funcionalidades similares a las de Google Maps, pero utilizando datos libres y personalizables de OSM. Con OSMdroid, los desarrolladores pueden mostrar mapas, agregar marcadores, trazar rutas y gestionar eventos táctiles. También permite el uso de mapas sin conexión mediante la descarga previa de teselas. [16][17][18]



Figura 2.2.7: OSMdroid

Aplicaciones en el proyecto:

- Visualización de mapa y gestión de eventos táctiles de zoom, desplazamiento y giro.
- Marcadores.
- Ubicación actual del usuario.

Open Route Service (ORS):

Plataforma de servicios de enrutamiento basada en los datos de OpenStreetMap. Proporciona una API que permite calcular rutas entre dos o más puntos, en distintos modos como coche, bicicleta, a pie, transporte público, silla de ruedas, entre otros. Además permite obtener tiempos de viaje, áreas de alcance (isochronas), geocodificación, búsqueda de lugares y análisis geoespacial. [19][20]



Figura 2.2.8: OpenRouteService

Aplicaciones en el proyecto:

- Cálculo de rutas a pie.
- Obtención de tiempos de viaje.
- Geocodificación.

Postman:

Herramienta de desarrollo utilizada para diseñar, probar y documentar APIs de forma sencilla e intuitiva. Permite enviar peticiones HTTP a servidores y ver sus respuestas, facilitando la validación y depuración de servicios web. [21]



Figura 2.2.9: Postman

Aplicaciones en el proyecto:

- Pruebas de peticiones HTTP a las diferentes APIs.

3 Diseño

El desarrollo del proyecto constituye el núcleo del trabajo realizado, en el que se materializa la solución propuesta a partir de una planificación estructurada y basada en principios de ingeniería del software. Este apartado recoge de forma ordenada las distintas fases que han permitido transformar una idea inicial en una aplicación funcional, pasando por la definición clara de sus requisitos, el diseño de su arquitectura de alto y bajo nivel, su implementación y la validación final del sistema.

Cada una de estas fases desempeña un papel clave en la calidad y coherencia del producto final. La **especificación de requisitos** garantiza que el sistema responde a las necesidades del usuario. El **diseño de alto nivel** permite estructurar el flujo funcional, definir las interfaces y establecer la arquitectura general. El **diseño de bajo nivel** detalla los modelos de datos y los comportamientos internos mediante diagramas específicos. La **implementación** convierte los diseños en código real, y las **pruebas de sistema** permiten comprobar su correcto funcionamiento y detectar posibles mejoras.

A través de este recorrido, se busca no solo documentar el proceso seguido, sino también evidenciar las decisiones técnicas tomadas y su justificación, mostrando así la evolución del proyecto desde su concepción hasta su ejecución final.

3.1 Requisitos

A partir de los objetivos planteados para la aplicación, se identifican los siguientes requisitos funcionales:

- **RF_01:** El usuario podrá registrarse.
- **RF_02:** El usuario podrá iniciar sesión.
- **RF_03:** El usuario podrá cerrar sesión.
- **RF_04:** El usuario de tipo jugador podrá visualizar su información de usuario: foto, nombre, email y número de actividades realizadas.
- **RF_05:** El usuario de tipo jugador podrá modificar su información de usuario: foto, nombre, email y contraseña.
- **RF_06:** El usuario de tipo jugador podrá visualizar las actividades pendientes, en proceso o completadas en las que esté registrado.
- **RF_07:** El usuario de tipo jugador podrá visualizar la información detallada de una actividad en la que esté registrado.
- **RF_08:** El usuario de tipo jugador podrá iniciar una actividad en el mapa.
- **RF_09:** El usuario de tipo jugador podrá leer un dispositivo NFC para registrar su paso por ese punto de la ruta.
- **RF_10:** El usuario de tipo administrador podrá crear actividades, con información obligatoria: título, descripción, fecha y hora de realización, usuarios, ruta y dispositivos NFC; y opcionalmente una foto. La actividad puede crearse como pendiente o directamente como activa, en el primer caso solo la información de título, descripción, fecha y hora y ruta son obligatorios.
- **RF_11:** El usuario de tipo administrador podrá visualizar las actividades que ha creado.
- **RF_12:** El usuario de tipo administrador podrá visualizar la información detallada de una actividad que ha creado.

- **RF_13:** El usuario de tipo administrador podrá activar y desactivar una actividad, para activarla debe estar completada toda la información obligatoria: título, descripción, fecha y hora de realización, usuarios, ruta y dispositivos NFC. Si está activa será visible para los usuarios registrados en ella, en caso contrario, no será visible.
- **RF_14:** El usuario de tipo administrador podrá modificar toda la información de una actividad.
- **RF_15:** El usuario de tipo administrador podrá visualizar la información de progreso de los usuarios participantes en una actividad.
- **RF_16:** El usuario administrador podrá eliminar una actividad.

A continuación, se desarrollan los casos de uso:

CU_01: Registrarse	
Actor	Usuario jugador
Precondición	No hay otro usuario registrado con el mismo email
Postcondición	El usuario está registrado
Flujo principal	<ol style="list-style-type: none"> 1. El usuario introduce sus datos: nombre, email, contraseña. 2. El sistema verifica que no hay otro usuario con el mismo email. 3. El sistema registra al usuario.
Flujos alternativos	Si existe un usuario registrado con el mismo email, se muestra un mensaje de error y no se realiza el registro.

Tabla 3.1.1: Caso de uso 01, registro

CU_02: Iniciar sesión	
Actor	Usuario jugador o administrador
Precondición	El usuario está registrado
Postcondición	El usuario entra en la aplicación con la información asociada a su cuenta
Flujo principal	<ol style="list-style-type: none"> 1. El usuario introduce sus datos: email, contraseña. 2. El sistema verifica que existe como usuario registrado. 3. El sistema verifica el rol y abre la aplicación en el formato conforme al rol y con la información del usuario.
Flujos alternativos	Si las credenciales son incorrectas se muestra un mensaje de error y no se inicia sesión.

Tabla 3.1.2: Caso de uso 02, inicio de sesión

CU_03: Cerrar sesión	
Actor	Usuario jugador o administrador
Precondición	El usuario ha iniciado sesión

Postcondición	Sesión del usuario cerrada
Flujo principal	<ol style="list-style-type: none"> 1. El usuario da al botón de cerrar sesión. 2. El sistema elimina la información de la sesión y vuelve a la pantalla de inicio de sesión.

Tabla 3.1.3: Caso de uso 03, cierre de sesión

CU_04: Visualizar información de usuario	
Actor	Usuario jugador
Precondición	-
Postcondición	-
Flujo principal	<ol style="list-style-type: none"> 1. El usuario accede a la pantalla de visualización de datos de usuario. 2. El sistema muestra en la vista los datos del usuario: foto (si la tiene o imagen por defecto), nombre y email.

Tabla 3.1.4: Caso de uso 04, ver información de usuario

CU_05: Modificar información de usuario	
Actor	Usuario jugador
Precondición	El usuario ha iniciado sesión
Postcondición	Modificación de la información de usuario
Flujo principal	<ol style="list-style-type: none"> 1. El usuario accede a la modificación de sus datos. 2. El sistema muestra sus datos actuales con opción a edición. 3. El usuario modifica sus datos con la nueva información: foto, nombre, email, contraseña. 4. El sistema verifica que no hay campos vacíos (a excepción de la foto que es opcional) y que los datos cumplen el formato correcto. 5. El sistema modifica la información del usuario.
Flujos alternativos	En caso de que falte algún dato necesario o tenga un formato incorrecto, muestra un mensaje de error.

Tabla 3.1.5: Caso de uso 05, modificar información de usuario

CU_06: Visualizar actividades en las que está registrado	
Actor	Usuario jugador
Precondición	-
Postcondición	-
Flujo principal	<ol style="list-style-type: none"> 1. El usuario accede a la visualización de actividades. 2. El sistema verifica muestra las actividades en las que el usuario ha sido registrado independientemente del estado de esta: pendiente, en progreso o completada.

Tabla 3.1.6: Caso de uso 06, visualizar actividades del jugador

CU_07: Visualizar actividad	
Actor	Usuario jugador
Precondición	El usuario está registrado en la actividad
Postcondición	-
Flujo principal	<ol style="list-style-type: none"> 1. El usuario accede a la información de una actividad.

Tabla 3.1.7: Caso de uso 07, visualizar actividad de jugador

CU_08: Iniciar actividad	
Actor	Usuario jugador
Precondición	La actividad no ha sido completada antes y está activa
Postcondición	-
Flujo principal	<ol style="list-style-type: none"> 2. El usuario inicia la actividad. 3. El sistema muestra la ruta de la actividad y la posición actual del usuario en un mapa.

Tabla 3.1.8: Caso de uso 08, iniciar actividad

CU_09: Leer dispositivo NFC	
Actor	Usuario jugador
Precondición	Se ha iniciado la actividad
Postcondición	Se ha registrado el punto asociado al NFC como leído en la ruta del usuario para esa actividad
Flujo principal	<ol style="list-style-type: none"> 1. El usuario accede a la lectura de NFC y acerca el NFC al teléfono. 2. El sistema detecta el dispositivo NFC y lo busca entre los asociados a la ruta. 3. El sistema marca el punto como leído y muestra un mensaje de acción completada.
Flujos alternativos	Si en el punto 2 el NFC no es encontrado o hay NFC no leídos anteriores en la ruta, no se registra el punto como leído y se muestra un mensaje de error.

Tabla 3.1.9: Caso de uso 09, leer dispositivo NFC

CU_10: Crear actividad	
Actor	Usuario administrador
Precondición	-
Postcondición	Nueva actividad creada.
Flujo principal	<ol style="list-style-type: none"> 1. El usuario accede a la creación de una nueva actividad. 2. El usuario introduce los datos de la actividad.

	<ol style="list-style-type: none"> 3. El sistema comprueba que ningún campo obligatorio está vacío y los datos cumplen el formato esperado. 4. El sistema registra la nueva actividad.
Flujos alternativos	En caso de que falte algún dato necesario se muestra un mensaje de error.

Tabla 3.1.10: Caso de uso 10, crear actividad

CU_11: Visualizar actividades creadas	
Actor	Usuario administrador
Precondición	-
Postcondición	-
Flujo principal	<ol style="list-style-type: none"> 1. El usuario accede a la visualización de las actividades. 2. El sistema muestra las actividades creadas por ese usuario.

Tabla 3.1.11: Caso de uso 11, visualizar actividades creadas del administrador

CU_12: Visualizar actividad creada	
Actor	Usuario administrador
Precondición	La actividad ha sido creada por este usuario
Postcondición	-
Flujo principal	<ol style="list-style-type: none"> 1. El usuario accede a la información de la actividad.

Tabla 3.1.12: Caso de uso 12, visualizar actividad creada por el administrador

CU_13: Activar y desactivar una actividad	
Actor	Usuario administrador
Precondición	La actividad ha sido creada.
Postcondición	Se modifica el estado de la actividad.
Flujo principal	<ol style="list-style-type: none"> 2. El usuario activa o desactiva una actividad. 3. El sistema cambia el estado de la actividad y la información asociada a este.

Tabla 3.1.13: Caso de uso 13, activar y desactivar actividad

CU_14: Modificar actividad	
Actor	Usuario administrador
Precondición	La actividad ha sido creada.
Postcondición	Actividad modificada.

Flujo principal	<ol style="list-style-type: none"> 1. El usuario accede a la modificación de una actividad. 2. El usuario modifica los datos de la actividad. 3. El sistema comprueba que ningún campo obligatorio está vacío y los datos cumplen el formato esperado. 4. El sistema guarda los cambios de la actividad.
Flujos alternativos	En caso de que falte algún dato necesario se muestra un mensaje de error.

Tabla 3.1.14: Caso de uso 14, modificar actividad

CU_15: Visualizar progreso de los usuarios	
Actor	Usuario administrador
Precondición	La actividad ha sido creada.
Postcondición	-
Flujo principal	<ol style="list-style-type: none"> 1. El usuario accede a la visualización de progreso de una actividad. 2. El sistema muestra los puntos de la ruta leídos por cada usuario registrado en la actividad.

Tabla 3.1.15: Caso de uso 15, visualizar progreso de los jugadores

CU_16: Eliminar actividad	
Actor	Usuario administrador
Precondición	La actividad ha sido creada.
Postcondición	La actividad ha sido eliminada.
Flujo principal	<ol style="list-style-type: none"> 1. El usuario elimina la actividad. 2. El sistema elimina la actividad y la información asociada a esta.

Tabla 3.1.16: Caso de uso 16, eliminar actividad

A continuación, se presentan dos diagramas de burbujas que complementan la información mostrada en los casos de uso. Cada uno refleja gráficamente la relación entre las funcionalidades disponibles para cada tipo de usuario (jugador y administrador), agrupando visualmente las acciones posibles en torno al actor principal. Esta representación permite identificar de forma rápida qué operaciones puede realizar cada usuario y cómo se conectan entre sí.

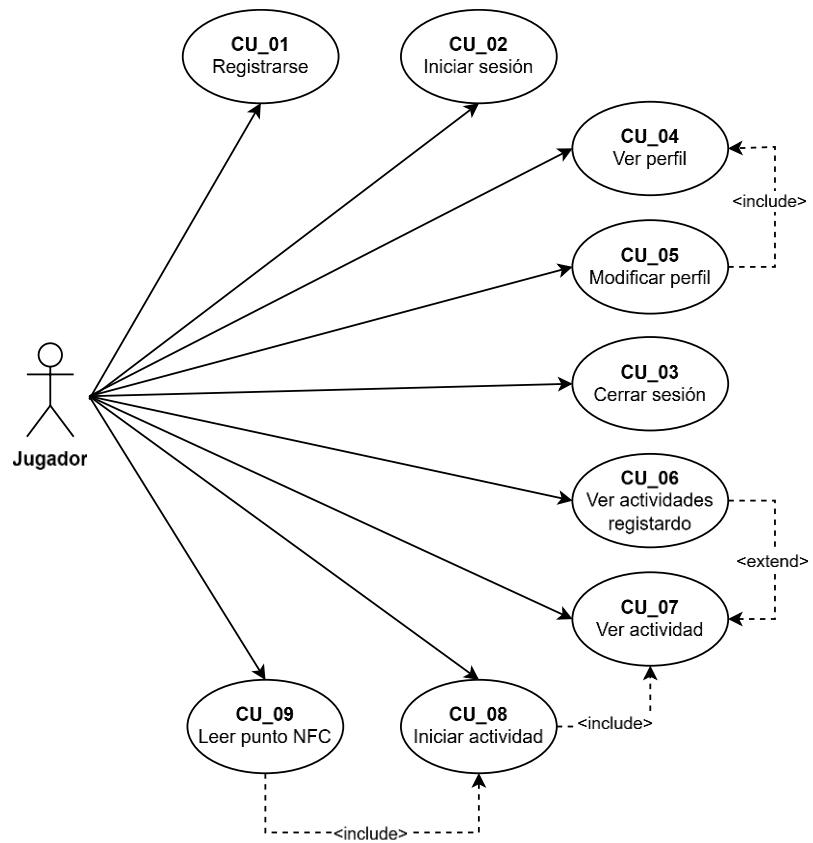


Figura 3.1.1: Diagrama de casos de uso del jugador

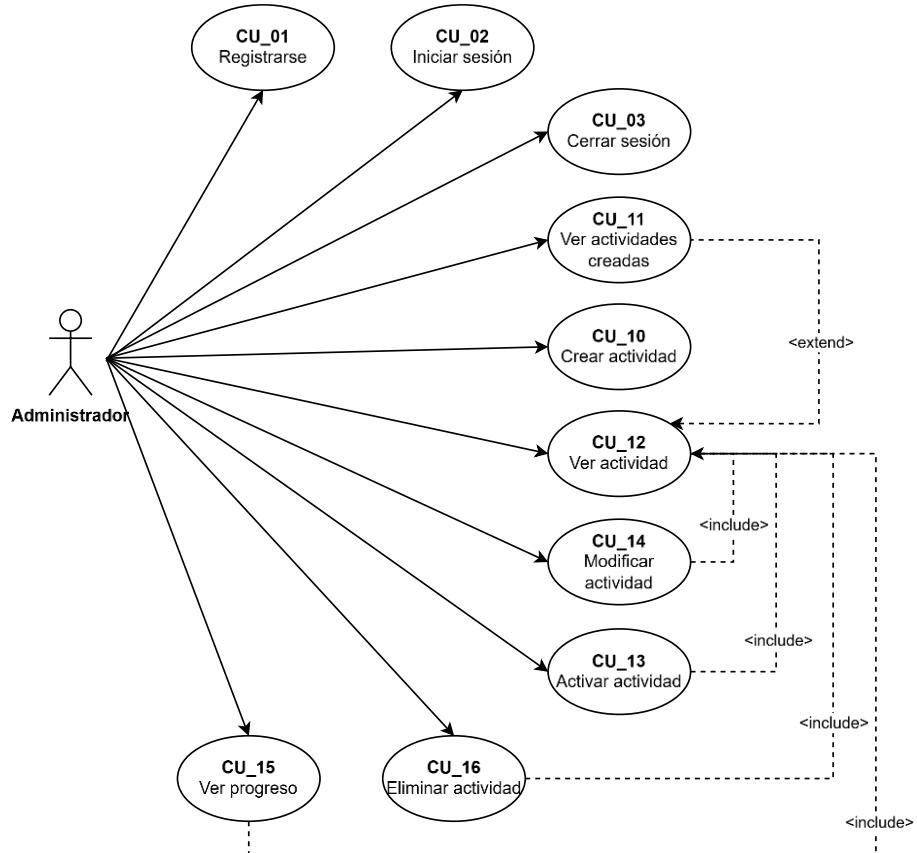


Figura 3.1.2: Diagrama de casos de uso del administrador

3.2 Diseño de alto nivel

Una vez definidos los requisitos funcionales y los casos de uso, se plantea el diseño de alto nivel del sistema con el objetivo de concretar su funcionamiento general y sentar las bases para el desarrollo. Esta etapa incluye tres elementos clave que permiten visualizar el sistema desde distintas perspectivas complementarias:

- El **diagrama de flujo** representa la lógica funcional desde el punto de vista del usuario, mostrando los posibles caminos de interacción y ayudando a identificar decisiones clave.
- El **diseño de interfaces** concreta la experiencia visual e interactiva a través de prototipos en Figma, centrándose especialmente en la accesibilidad para personas mayores.
- El **diagrama de arquitectura** describe la estructura técnica del sistema, organizando sus componentes principales y la comunicación entre ellos.

Con este enfoque, se busca facilitar la comprensión del sistema, guiar la implementación y asegurar la coherencia entre las funcionalidades previstas, la experiencia de usuario y la organización del software.

3.2.1 Diagrama de flujo

Como primer paso del diseño de alto nivel, se presenta un diagrama de flujo (Figura 3.2.1.1) que recoge la lógica del sistema desde la perspectiva del usuario. En él se representan las acciones posibles, las decisiones que pueden tomarse y las rutas que sigue cada tipo de usuario (jugador y administrador) en la aplicación.

Este diagrama es fundamental para visualizar cómo se desarrollan los casos de uso definidos anteriormente y para detectar caminos no contemplados o errores lógicos. Además, sirve como base para las siguientes etapas del diseño: ayuda a definir las pantallas necesarias y apoya la estructuración de los módulos funcionales del sistema.

Leyenda:

- Inicio general
- Autenticación
- Decisiones
- Administrador
- Jugador

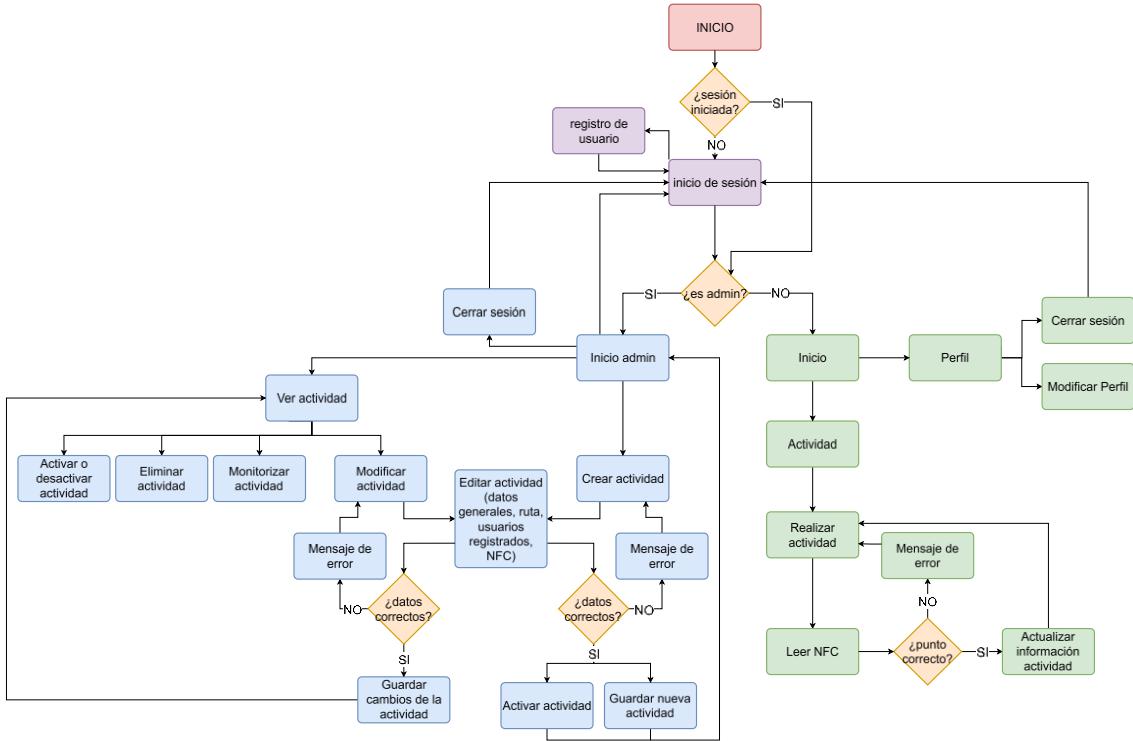


Figura 3.2.1.1: Diagrama de flujo

Al iniciar la aplicación se comprueba si la sesión está iniciada, si es así, se pasa a la siguiente comprobación, en caso contrario se muestra la tarea de inicio de sesión.

Desde inicio de sesión existe la opción de crear una cuenta nueva, tras esta tarea se vuelve a inicio de sesión. Si se completa correctamente el inicio de sesión el sistema comprueba si el usuario tiene rol de usuario jugador o de administrador y muestra las tareas destinadas a ese rol.

En caso de ser **usuario jugador**, se abre la pantalla principal que mostrará listadas las actividades en las que está registrado el usuario. A partir de la pantalla inicial se puede visualizar el perfil del usuario, donde contará con las opciones de modificar sus datos o cerrar sesión.

Se puede ver la información de una actividad desde la lista de estas. Además de los datos, se muestra la opción de iniciar la actividad. Si se acciona esta tarea, se muestra un mapa con la ruta de la actividad. Para completar dicha ruta se leen dispositivos NFC que representan los puntos, si se lee el punto correcto se actualiza la información de la actividad y en caso contrario se muestra un mensaje de error.

Para el **usuario administrador**, la primera pantalla que se muestra también presenta una lista de tareas, pero con aquellas que ha creado este usuario. Aquí también tendrá las opciones de crear una actividad nueva y cerrar sesión.

Al pasar a la tarea de crear una nueva actividad, puede modificar los distintos datos y si son correctos crear y activar la actividad, en caso de tener datos inválidos recibe un mensaje de error y no se completa la tarea.

Si el administrador entra en una tarea desde la pantalla principal, puede ver los datos de la actividad y las opciones de monitorizar el progreso de los usuarios, editar, activar o desactivar y eliminar esa actividad.

La tarea de editar una actividad permite ver y cambiar los datos de igual manera que la tarea de crear una actividad nueva. Si los datos son válidos se puede guardar la actividad modificada, si la nueva información no cumple los requisitos muestra un mensaje de error.

3.2.2 Diseño de interfaces de usuario

A partir de los casos de uso y los módulos representados en el diagrama de flujo, se desarrolla un diseño de la interfaz de usuario (UI) en Figma que satisface todos los requisitos funcionales.

Este diseño permite definir y visualizar la experiencia de usuario (UX) desde una etapa temprana del proyecto. A través de los prototipos creados se representan con precisión las distintas pantallas de la aplicación, su disposición visual, los componentes interactivos y la navegación entre ellas, diferenciando claramente las funcionalidades disponibles para jugadores y administradores. Gracias a esta fase de diseño, fue posible validar decisiones de usabilidad, asegurar coherencia con la arquitectura general del sistema y establecer una base sólida sobre la que desarrollar el frontend de manera más eficiente. Además, al anticipar la estructura y comportamiento de las pantallas, el diseño de interfaces contribuye a una planificación más eficaz del desarrollo y a reducir retrabajos, facilitando así una implementación más eficiente y alineada con los objetivos del proyecto.

Para el desarrollo del diseño se han tenido en cuenta buenas prácticas para **UI orientadas a personas mayores**. En primer lugar, los colores elegidos cumplen en todos los casos el nivel mínimo de contraste (AA) para elementos gráficos y el contraste mejorado (AAA) para el texto, mejorando así la accesibilidad y usabilidad para todos los usuarios. Además, el uso de los colores permite establecer una jerarquía visual mejorando la experiencia del usuario al facilitar la comprensión de la interfaz. Mediante estos colores se consigue, por ejemplo, identificar botones, mantener el foco en la tarea principal de cada pantalla (botones y etiquetas amarillos) o indicar opciones secundarias (botones azules) y cancelación de tareas (botones azul oscuro).

Se mantiene el mismo estilo redondeado y mismo grupo de colores (amarillo, azul y azul oscuro) para todos los botones permitiendo que sean claramente identificables. Todos ellos cuentan con una etiqueta de texto representativa de su función. Para mejorar la accesibilidad se han diseñado botones grandes y con separación suficiente, entre ellos y con el borde de la pantalla, para evitar errores al clicar.

Se proporciona feedback conciso para todas las acciones y mensajes de error aclarativos en caso de que alguna acción no se pueda realizar. Esto ayuda al usuario a comprender qué sucede en la aplicación reduciendo la complejidad cognitiva.

En toda la interfaz se usa un tamaño de letra grande, de al menos 20sp, y una fuente de texto clara y legible.

Se mantiene una distribución intuitiva y sencillas de los elementos en las distintas pantallas.

Por último, se busca una navegación sencilla centrando cada pantalla en una tarea principal, reduciendo los caminos posibles y permitiendo siempre volver atrás y cancelar de forma sencilla y clara.

Todas estas características se aplican especialmente en la parte destinada al usuario jugador. En el caso del usuario administrador, si bien se mantiene la misma línea visual general, se adoptan criterios de diseño distintos, ya que esta sección no está orientada a personas mayores. En su lugar, se prioriza una distribución eficiente de la información y una interacción ágil para facilitar la gestión de tareas.

A continuación, se presentan las distintas pantallas diseñadas.

Registro de usuario

En esta pantalla se presenta el formulario de registro de nuevos usuarios con la opción de ir al inicio de sesión si el usuario ya tiene cuenta.



Figura 3.2.2.1: Pantalla de registro de usuario

Inicio de sesión

Presenta el formulario de inicio de sesión con la opción de crear una cuenta en caso de que el usuario no esté registrado.



Figura 3.2.2.2: Pantalla de inicio de sesión

Pantallas de usuario jugador:

Pantalla principal

Al iniciar sesión como usuario jugador se muestra esta pantalla. Presenta las actividades en las que el usuario a sido registrado, diferenciando aquellas que ya han sido completadas con una capa verde y un ícono de check. Cuenta con un buscador para buscar actividades por el título. En la parte inferior se muestra una barra de navegación entre esta pantalla de inicio y el perfil de usuario que permite un flujo cómodo y rápido entre estas dos pantallas principales.



Figura 3.2.2.3: Pantalla de inicio para usuario jugador

Perfil

Se presenta la información del usuario con las opciones de modificar estos datos o cerrar sesión.



Figura 3.2.2.4: Pantalla de perfil de usuario

Modificar Perfil

Permite modificar los datos del usuario, incluida una foto de perfil. Permite guardar los cambios o cancelar el proceso.

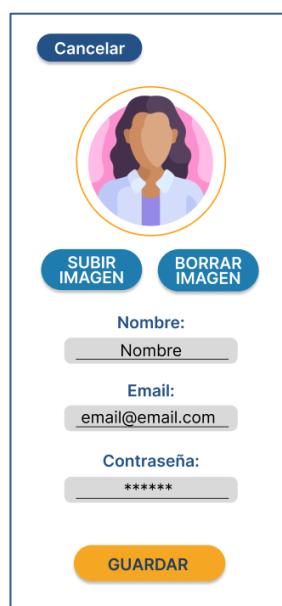


Figura 3.2.2.5: Pantalla de perfil de usuario

Actividad

Presenta la información completa de una actividad. Cuenta con las opciones de iniciar una actividad, si no se ha completado anteriormente, y volver a la pantalla de inicio.

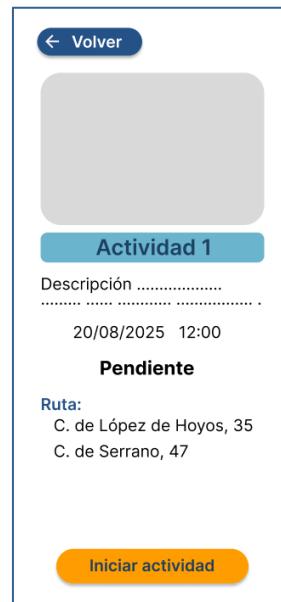


Figura 3.2.2.6: Pantalla de actividad completada

Figura 3.2.2.7: Pantalla de actividad pendiente

Ejecución de actividad

Esta pantalla muestra un mapa con la ruta de la actividad y la posición actual del usuario. Permite leer un NFC para registrar el paso por un punto de la ruta o volver a la información de la actividad.

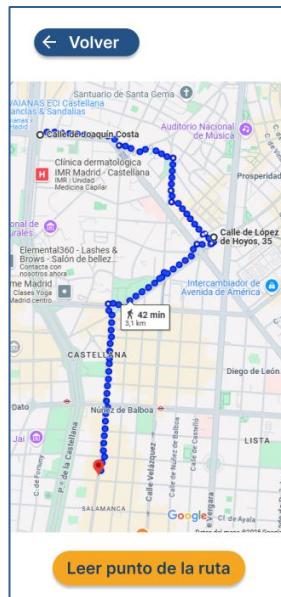


Figura 3.2.2.8: Pantalla de ejecución de actividad

Lectura de NFC

Esta pantalla informa de que la aplicación está intentando leer un NFC, dando la opción de volver al mapa de la actividad cancelando la lectura. En caso de lectura correcta, se muestra la pantalla de la derecha informando de ello.

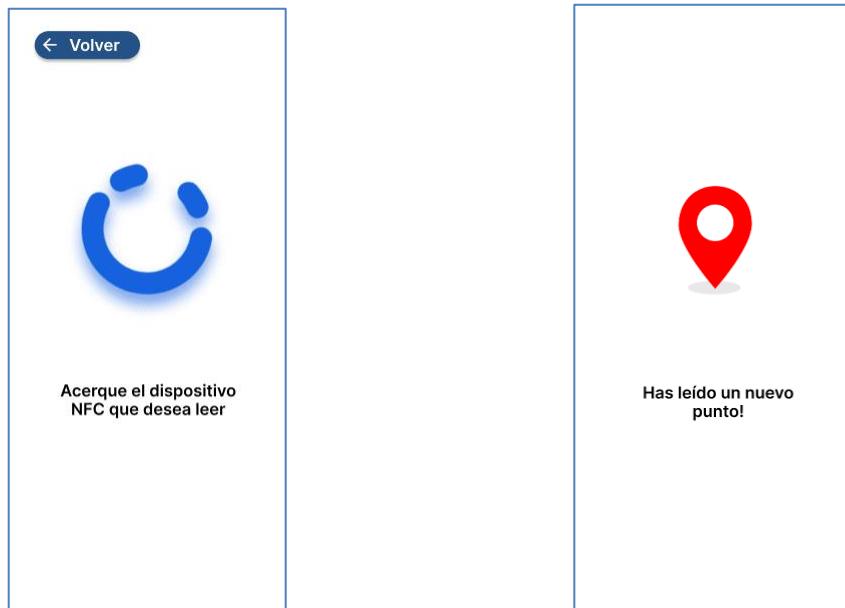


Figura 3.2.2.9: Pantalla de lectura de NFC

Figura 3.2.2.10: Pantalla de confirmación de lectura de NFC

Pantallas de usuario administrador:

Pantalla principal

Al iniciar sesión como usuario administrador se abre esta pantalla. Muestra las actividades creadas por el usuario administrador, diferenciando aquellas que ya han sido completadas con una capa verde y un ícono de check. Cuenta con un buscador para buscar actividades por el título. Permite cerrar sesión.



Figura 3.2.2.11: Pantalla de inicio del administrador

Crear actividad

Permite crear una nueva actividad desactivada o crearla y activarla, dando la opción de cancelar el proceso.

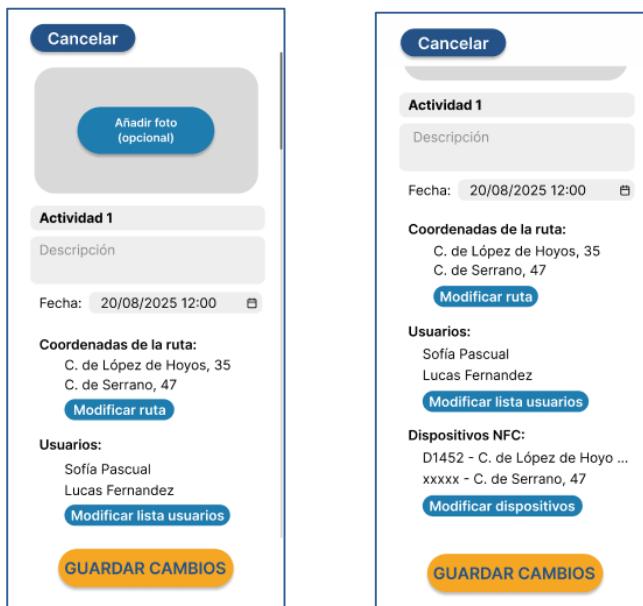


Este formulario es para la creación de una actividad. Se incluye un botón para agregar foto (opcional), campos para Título y Descripción, un selector de Fecha (dd/mm/aaaa), un campo para Coordenadas de la ruta con un botón para añadir puntos de la ruta, secciones para Usuarios y Dispositivos NFC, y dos botones principales: "CREAR ACTIVIDAD" y "CREAR Y ACTIVAR ACTIVIDAD".

Figura 3.2.2.12: Pantalla de creación de actividad

Modificar actividad

Permite cambiar los datos de una actividad existente, dando la opción de guardar los cambios o cancelar el proceso.



Este formulario es para la modificación de una actividad existente. Se muestra el nombre de la actividad (Actividad 1), su descripción, fecha (20/08/2025 12:00), coordenadas de la ruta (C. de López de Hoyos, 35, C. de Serrano, 47) con un botón para modificar ruta, una lista de usuarios (Sofía Pascual, Lucas Fernández) con un botón para modificar lista de usuarios, y dispositivos NFC (D1452 - C. de López de Hoyos ...). Los cambios se guardan en el botón "GUARDAR CAMBIOS".

Figura 3.2.2.13 y 14: Pantalla de modificación de actividad

Modificar fecha y hora

Permite seleccionar una fecha y hora. Estas vistas se usan en la creación y modificación de una actividad.



Figura 3.2.2.15: Widget de selección de fecha

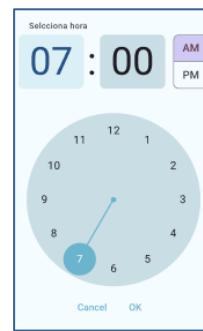


Figura 3.2.2.16: Widget de selección de hora

Modificar usuarios

Permite buscar usuarios por su nombre con un buscador autocompletável y añadirlos a la actividad. Esta vista es usada para la creación y modificación de una actividad.

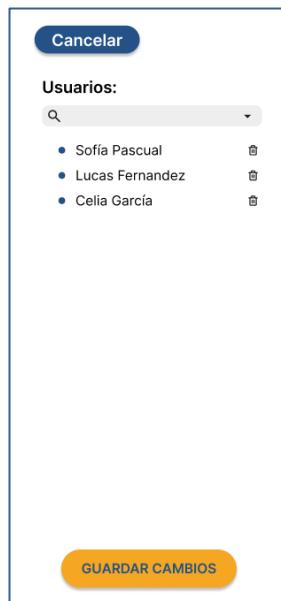


Figura 3.2.2.17: Pantalla de edición de usuarios registrados en una actividad

Modificar ruta

Esta pantalla permite seleccionar un punto en el mapa y añadirlo a la ruta. Se da la opción de guardar la ruta o cancelar el proceso. Esta vista es usada para la creación y modificación de una actividad.

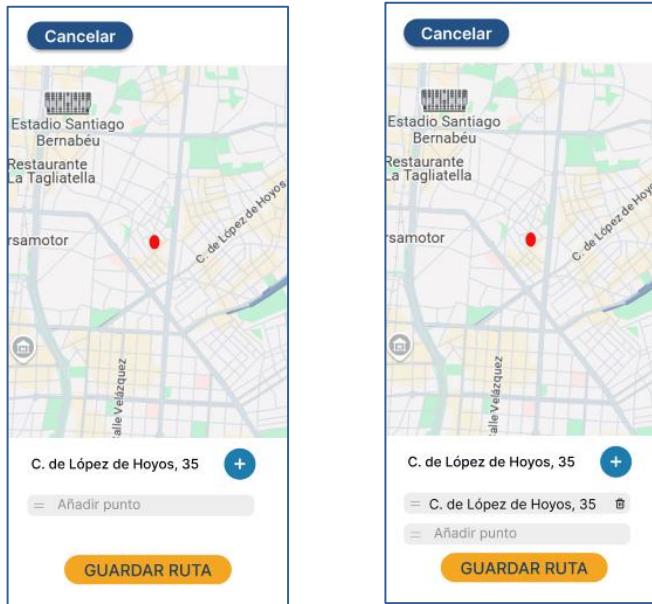


Figura 3.2.2.18 y 19: Pantalla de edición de la ruta

Modificar NFC

Permite leer dispositivos NFC y asociarlo a un punto de la ruta de la actividad. Esta vista es usada para la creación y modificación de una actividad.

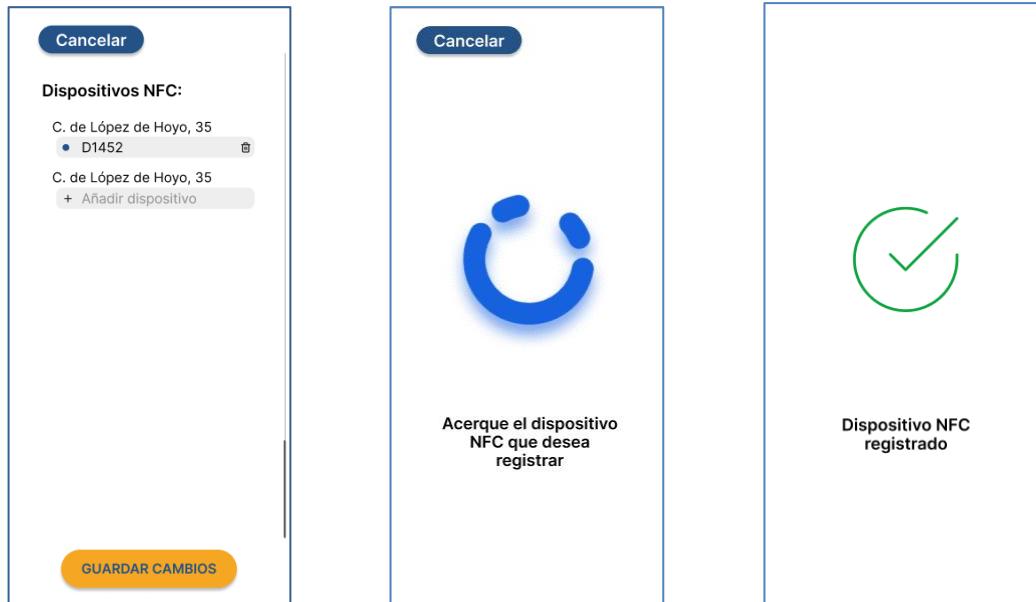


Figura 3.2.2.20:
Pantalla de edición de
NFC asociados a la ruta
de una actividad

Figura 3.2.2.21:
Pantalla de lectura de
NFC para registro en
una actividad

Figura 3.2.2.22:
Pantalla de confirmación
de registro de NFC en
una actividad

Visualizar actividad

Muestra la información de una actividad, dando las opciones de volver a la pantalla de inicio, editar la actividad, monitorizar la actividad, activar o desactivar la actividad y eliminar la actividad.

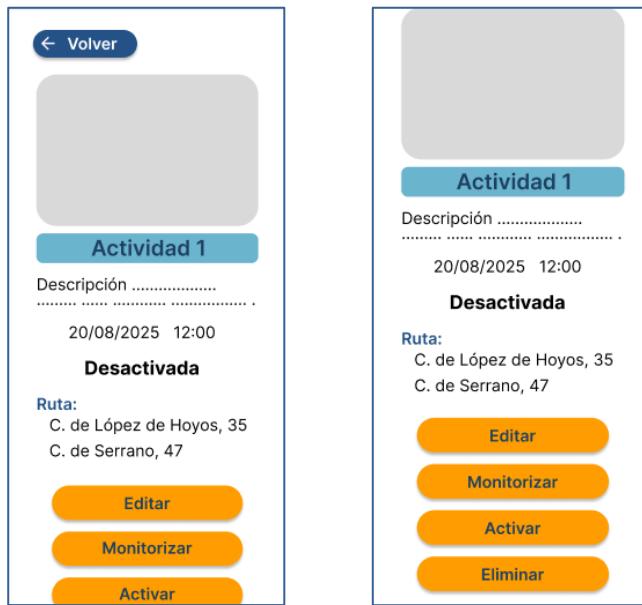


Figura 3.2.2.23 y 24: Pantalla de visualización de actividad de administrador

Monitorizar actividad

Permite monitorizar el progreso de los usuarios registrados en una actividad, muestra los puntos de la ruta en verde, el siguiente en amarillo y los demás en gris.

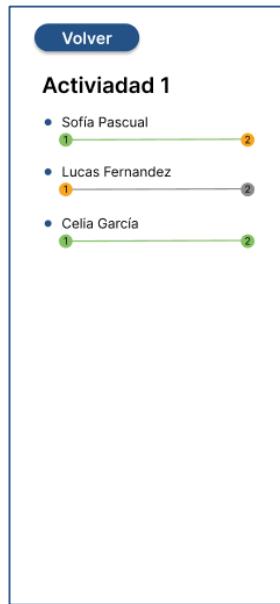


Figura 3.2.2.25: Pantalla de visualización de progreso de los usuarios de una actividad

3.2.3 Diagrama de arquitectura

Para organizar y definir los módulos necesarios para el sistema, se presenta el siguiente diagrama de arquitectura (Figura 3.2.3.1). Este diagrama ofrece una visión de alto nivel de la estructura del sistema, distribuyendo claramente las responsabilidades entre la capa de presentación, la lógica de aplicación y los servicios externos. El diseño mantiene coherencia con el diagrama de flujo de actividades y refuerza la separación de roles entre jugadores y administradores.

Leyenda de las flechas de interacción:

- Input de datos
- Output de datos
- Comunicación interna del sistema
- Comunicación con recursos externos al sistema

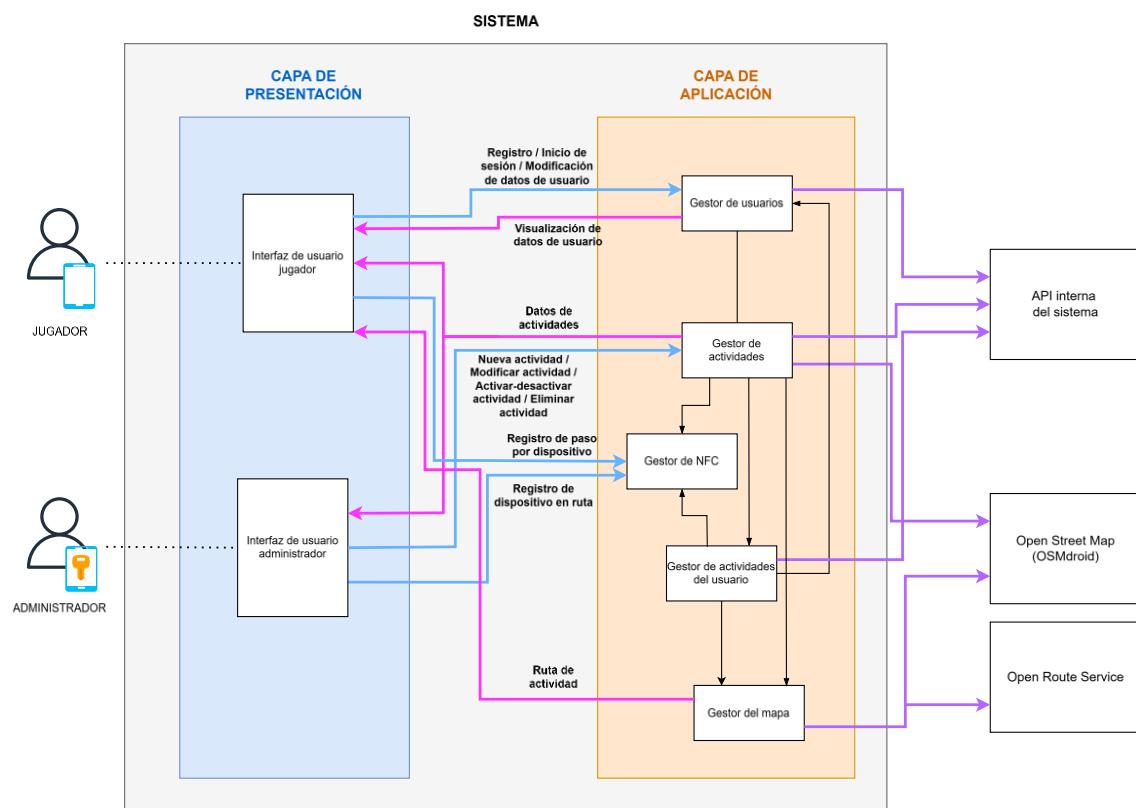


Figura 3.2.3.1: Diagrama de arquitectura

Como se muestra en la Figura 3.2.3.1, el sistema se estructura en tres capas principales: presentación, aplicación y recursos externos.

La **capa de presentación** representa la interfaz con la que interactúan los usuarios, distinguiendo entre jugadores y administradores. Cada tipo de usuario puede acceder a distintas funcionalidades según su rol dentro del sistema.

En la **capa de aplicación** se concentra la lógica del negocio, distribuida en distintos gestores que encapsulan funcionalidades específicas del sistema favoreciendo la separación de responsabilidades. A continuación, se detallan los principales componentes de esta capa:

El Gestor de actividades permite al administrador crear, visualizar, modificar, activar y eliminar actividades. Durante la creación y modificación de actividades, este gestor se comunica con el Gestor del mapa, para definir la ruta sobre el plano; el Gestor de NFC, para asociar los dispositivos NFC de control; y con el Gestor de usuarios y el Gestor de actividades del usuario para registrar y vincular participantes. Toda la información relativa a una actividad se almacena mediante una API interna, mientras que la generación de rutas se apoya en servicios externos como OSMdroid y OpenRouteService.

El Gestor de actividades del usuario se encarga de mantener la relación entre las actividades y el progreso individual de cada jugador dentro de ellas. Gestiona tanto el seguimiento de los puntos superados como la finalización de actividades.

El Gestor de NFC permite leer y procesar la información de dispositivos NFC. Cumple dos funciones: por un lado, permite al administrador registrar dispositivos en las rutas colaborando con el Gestor de actividades; y por otro, permite al jugador validar su paso por un punto de control trabajando con el Gestor de actividades del usuario.

El Gestor de mapa es responsable de la creación y gestión de mapas, incluyendo la localización del usuario, la colocación de marcadores y la conversión entre direcciones y coordenadas geográficas. Para ello, se apoya en OSMdroid, que se encarga de representar visualmente el mapa, y en OpenRouteService, utilizado para calcular rutas óptimas a pie entre los distintos puntos de la actividad.

Finalmente, en la **capa de servicios externos**, el sistema hace uso de tecnologías de terceros para enriquecer su funcionalidad: OSMdroid para la visualización del mapa y la gestión de la ubicación y los marcadores, OpenRouteService para el cálculo de rutas, y una API interna para la persistencia de datos y operaciones de backend.

En conjunto, la arquitectura propuesta garantiza una clara separación de responsabilidades entre las distintas capas del sistema, lo que facilita su comprensión, mantenimiento y evolución. Al apoyarse en servicios externos especializados para funcionalidades como la geolocalización, el mapeo y la persistencia, el sistema puede ofrecer una experiencia rica y precisa a los usuarios sin comprometer su eficiencia ni su capacidad de evolución futura.

3.3 Diseño de bajo nivel

Tras definir la estructura general del sistema en el diseño de alto nivel, esta sección se centra en concretar los aspectos internos que permiten llevar esa estructura a la práctica. El diseño de bajo nivel profundiza en la representación de los datos y en el comportamiento específico de los componentes del sistema.

Por un lado, el modelo de datos describe las entidades principales del sistema, sus atributos y las relaciones entre ellas, sentando las bases para la persistencia y el manejo de la información.

Por otro, los diagramas de secuencia muestran cómo interactúan los distintos objetos o módulos a lo largo del tiempo en escenarios específicos, detallando el flujo de mensajes y la lógica de ejecución.

Este nivel de detalle es fundamental para asegurar una implementación fiel a los requisitos y facilita tanto el desarrollo como el mantenimiento futuro del sistema.

3.3.1 Diseño de modelos de datos

El modelo de datos describe la estructura lógica de la información que gestiona el sistema. A partir de los requisitos identificados y los casos de uso, se definen las entidades principales, sus atributos y las relaciones que las vinculan.

Este diseño proporciona el soporte necesario para las operaciones de almacenamiento, consulta y actualización de datos, asegurando su consistencia e integridad a lo largo del ciclo de vida del sistema.

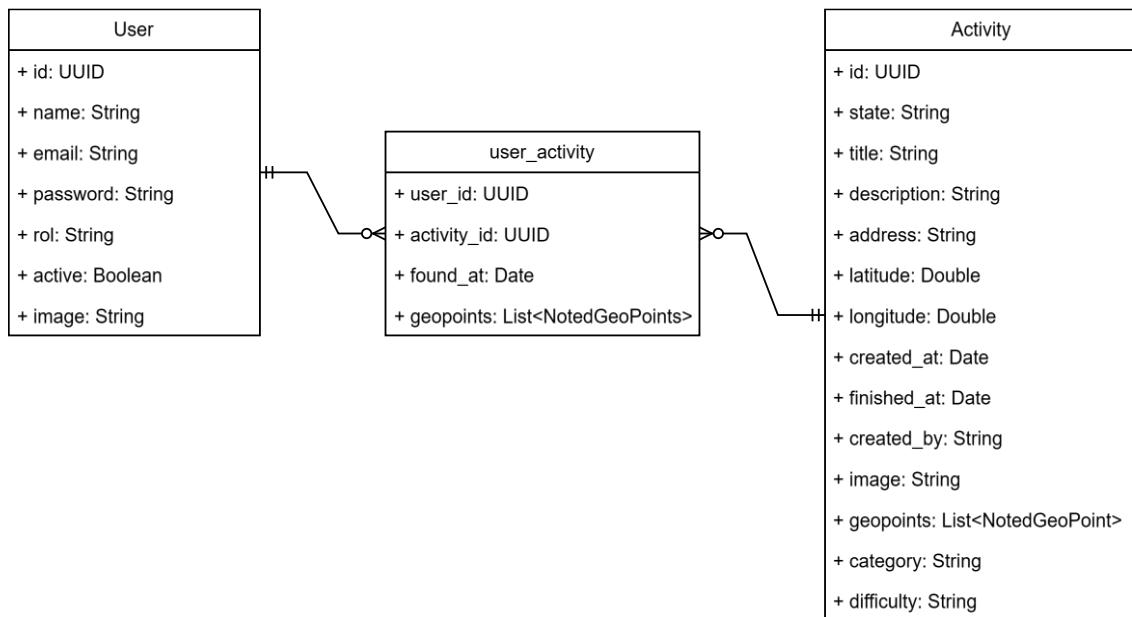


Figura 3.3.1.1: Modelos de datos para API

Como podemos observar en la Figura 3.3.1.1, el sistema cuenta con tres modelos de datos principales: **User**, **User_Activity** y **Activity**.

El modelo **User** representa a un usuario registrado en el sistema. Se identifica mediante un id y contiene los siguientes atributos: nombre, correo electrónico, contraseña, rol, activo e imagen. El rol puede ser de tipo jugador (user) o administrador (admin), este definirá las funciones que puede realizar en la aplicación. La imagen se almacena codificada en formato Base64 y es opcional; en caso de no proporcionarse, se guarda una cadena vacía ("").

El modelo **Activity** representa una actividad disponible en el sistema. Está identificada por un id y cuenta con los siguientes campos: estado (que indica si la actividad es visible para los usuarios), título, descripción, dirección del punto final, coordenadas del punto final (latitud y longitud), fecha de realización (created_at), fecha de finalización (finished_at), identificador del usuario creador, imagen asociada, lista de puntos que conforman la ruta, categoría y nivel de dificultad. Al igual que en el modelo de usuario, la imagen se guarda codificada en Base64 y es opcional. Actualmente, los campos de categoría y dificultad no se utilizan, aunque están previstos para futuras funcionalidades.

Por último, el modelo **User_Activity** representa la relación entre un usuario y una actividad en la que está inscrito. Está definido por los identificadores del usuario y de la actividad, y contiene información sobre la fecha en que el usuario completó la actividad, así como la lista de puntos que componen la ruta asignada al jugador.

Además de los tres modelos principales que estructuran los datos del sistema y se almacenan en la API interna, se define a continuación un tipo de objeto adicional utilizado en las listas de rutas de los modelos Activity y User_Activity.

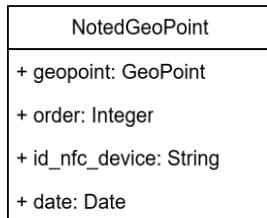


Figura 3.3.1.2: Modelo de NotedGeoPoint

Se trata del modelo **NotedGeoPoint**, que describe cada punto que compone una ruta. Incluye la ubicación geográfica del punto (tipo GeoPoint de java), su posición dentro del recorrido, el identificador del dispositivo NFC asociado y la fecha en la que el usuario ha registrado dicho punto. Este último campo permanece siempre vacío en Activity, y también en User_Activity hasta que el usuario interactúe con el punto correspondiente.

Cómo respuesta de la llamada de inicio de sesión a la API se obtiene un **TokenResponse**, para gestionar esta respuesta se define el siguiente modelo de datos.

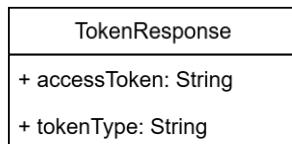


Figura 3.3.1.2: Modelo de TokenResponse

El token que contiene (accessToken), representa un bearer token que permite al usuario hacer otras llamadas a la API y define su sesión.

Por otro lado, para la interacción con la API de OpenRouteService (ORS), se emplean tres modelos de datos adicionales, representados en la Figura 3.3.1.4.

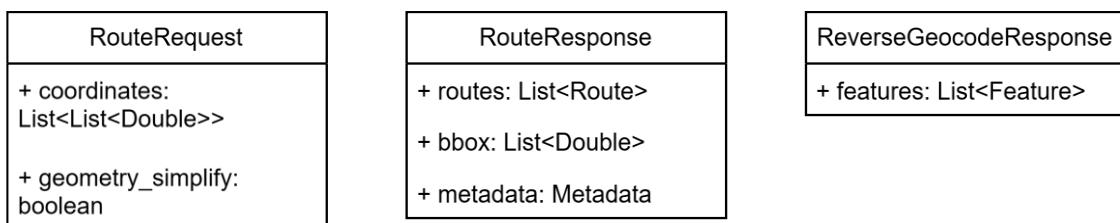


Figura 3.3.1.4: Modelos de datos para ORS

RouteRequest se utiliza como parámetro en la llamada que permite obtener una ruta a partir de una lista de puntos de coordenadas (coordinates).

RouteResponse es la respuesta a esa misma llamada. La ruta se obtiene a través del atributo routes. Las clases anidadas como Route, Metadata y otras no se detallan aquí, ya que su único propósito es mapear la estructura de la respuesta de la API.

ReverseGeocodeResponse es el modelo que representa la respuesta de la llamada encargada de traducir coordenadas en direcciones en formato textual. Al igual que en el caso anterior, las clases Feature y Properties no se explicitan, pues se utilizan únicamente para reflejar la estructura interna de la respuesta. La dirección deseada se obtiene accediendo al atributo label del primer Feature.

3.3.2 Diagrama de secuencia

Los diagramas de secuencia permiten visualizar el comportamiento dinámico del sistema en distintos escenarios clave. Este tipo de diagramas describe cómo interactúan entre sí los objetos o componentes del sistema a lo largo del tiempo, detallando el orden de los mensajes intercambiados para llevar a cabo una funcionalidad concreta.

A partir de los casos de uso y del modelo definido en el diseño de alto nivel, se seleccionan las interacciones más representativas y se detallan paso a paso, lo que resulta especialmente útil como guía directa para la implementación.

Registro de usuario:

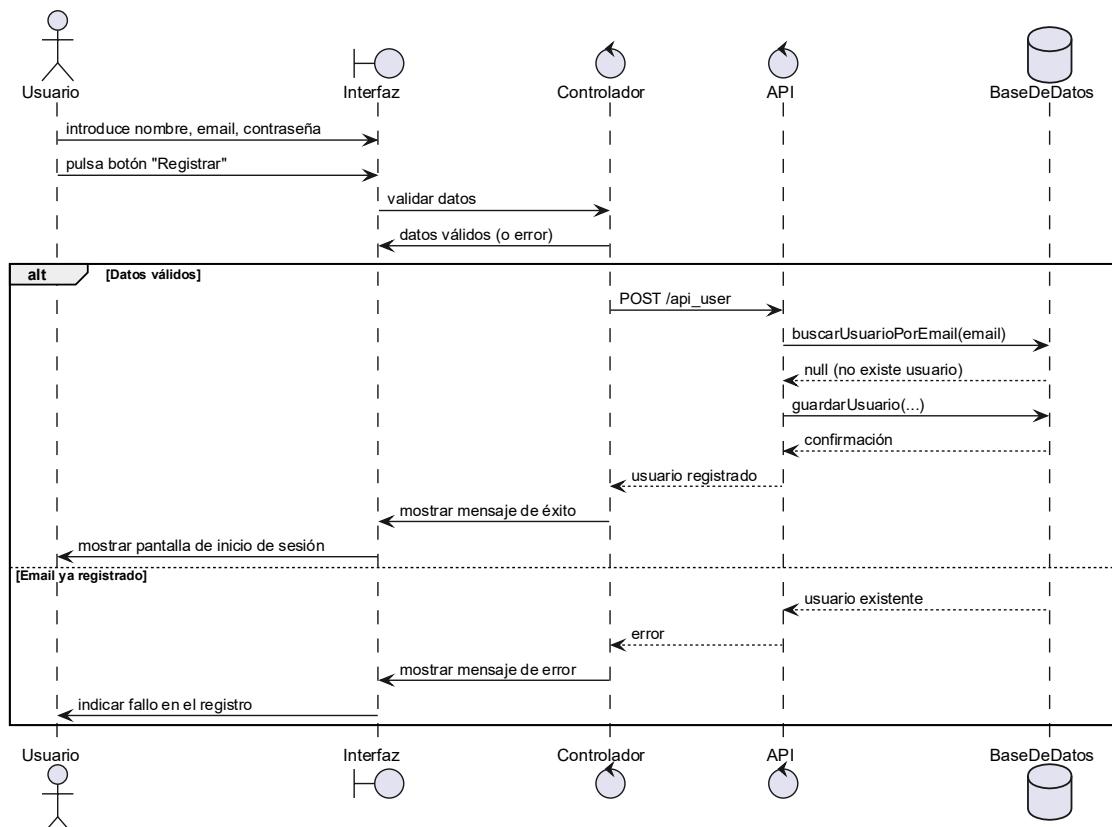


Figura 3.3.2.1: Diagrama de secuencia de registro de usuario

El usuario ingresa sus datos (nombre, email y contraseña) y pulsa el botón de registro. Se verifica que todos los campos estén completos y que cumplan con el formato requerido; en caso contrario, se muestra un mensaje de error.

A continuación, se realiza una llamada POST para crear el nuevo usuario, comprobando que no exista ya otro usuario registrado con el mismo email. Si la creación es exitosa, se muestra un mensaje informativo y se redirige al usuario a la pantalla de inicio de sesión.

Inicio de sesión:

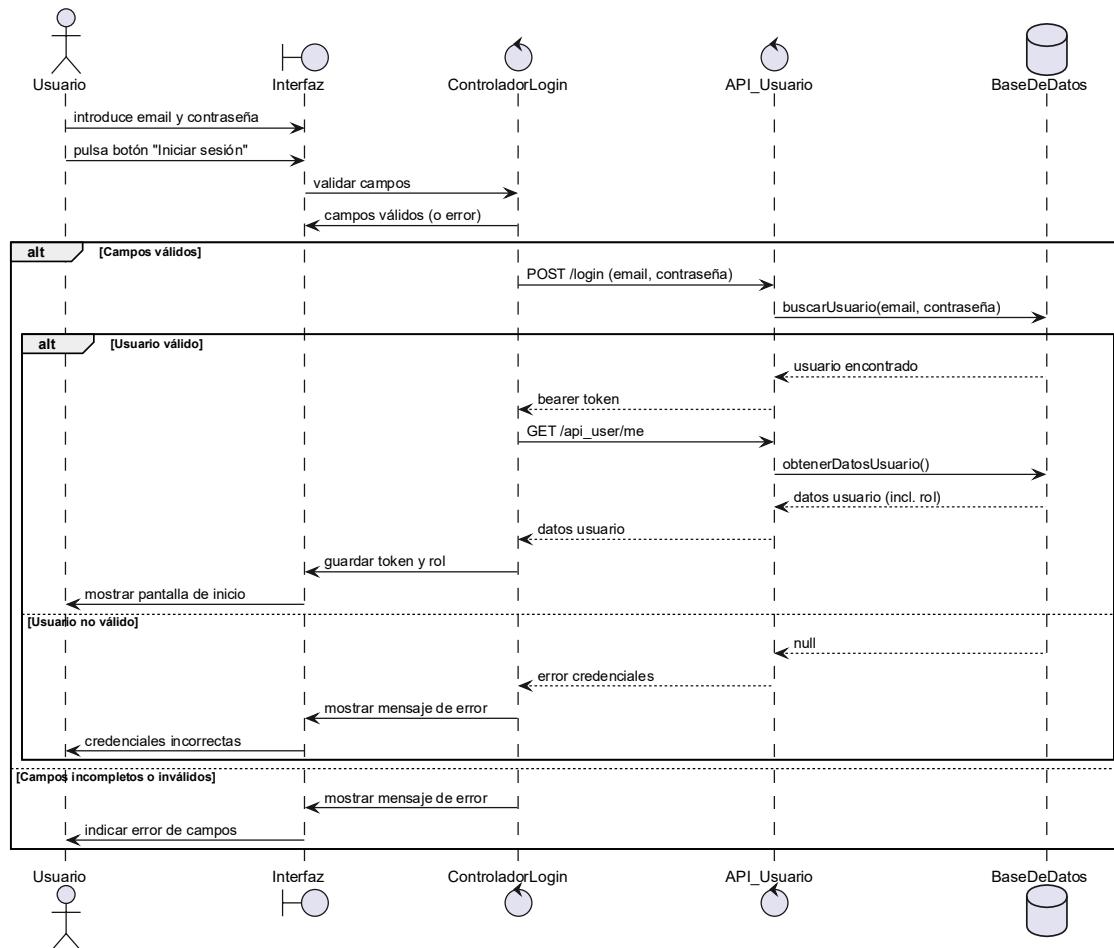


Figura 3.3.2.2: Diagrama de secuencia de inicio de sesión

El usuario introduce sus datos (email y contraseña) y pulsa el botón de iniciar sesión. El sistema verifica que ambos campos estén completos; en caso contrario, muestra un mensaje de error.

Si los campos están correctamente rellenos, se realiza una llamada POST a la API para validar las credenciales. Si estas son válidas, la API devuelve un token Bearer que representa la sesión del usuario.

Una vez autenticado, se realiza una llamada GET para obtener los datos del usuario actual, y la aplicación almacena el token y el rol correspondiente. Finalmente, se muestra la pantalla de inicio adecuada según el rol del usuario.

Si las credenciales son incorrectas, se muestra un mensaje de error y el acceso a la aplicación es denegado.

Visualización y modificación de perfil de usuario:

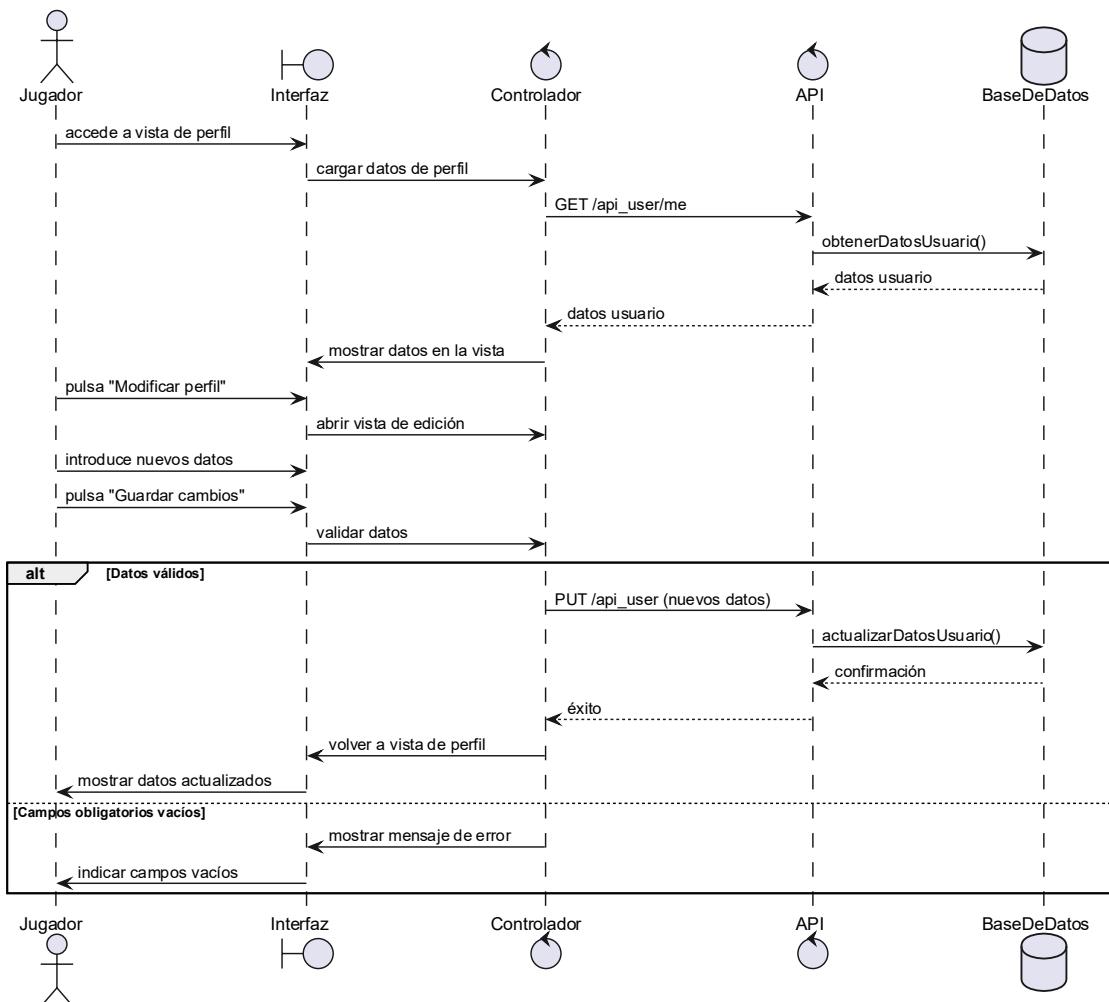


Figura 3.3.2.3: Diagrama de secuencia de visualización y modificación de perfil

El usuario accede a la vista de información de perfil, donde el sistema realiza una llamada GET a la API para obtener los datos del usuario actual y los carga en la interfaz.

Si el usuario selecciona la opción de modificar perfil, se muestra una vista para editar sus datos. En esta, el usuario puede realizar los cambios deseados; antes de guardar, se verifica que ningún campo obligatorio esté vacío. Si algún campo obligatorio falta, se muestra un mensaje de error y no se continúa con el proceso.

Una vez validados los datos, se realiza una llamada PUT a la API para guardar los cambios, y posteriormente se retorna a la vista de visualización del perfil con los datos actualizados.

Visualización de actividades en las que está registrado el usuario:

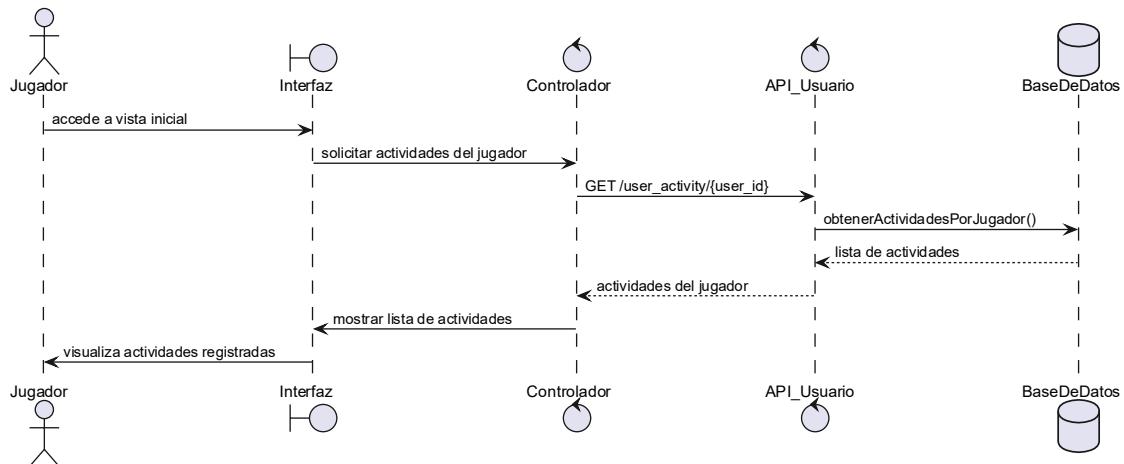


Figura 3.3.2.4: Diagrama de secuencia de visualización de actividades del jugador

El usuario accede a la vista inicial y el sistema realiza una llamada GET a la API para obtener las actividades en las que está registrado. Estas actividades se muestran en la interfaz en formato de lista.

Inicio de actividad:

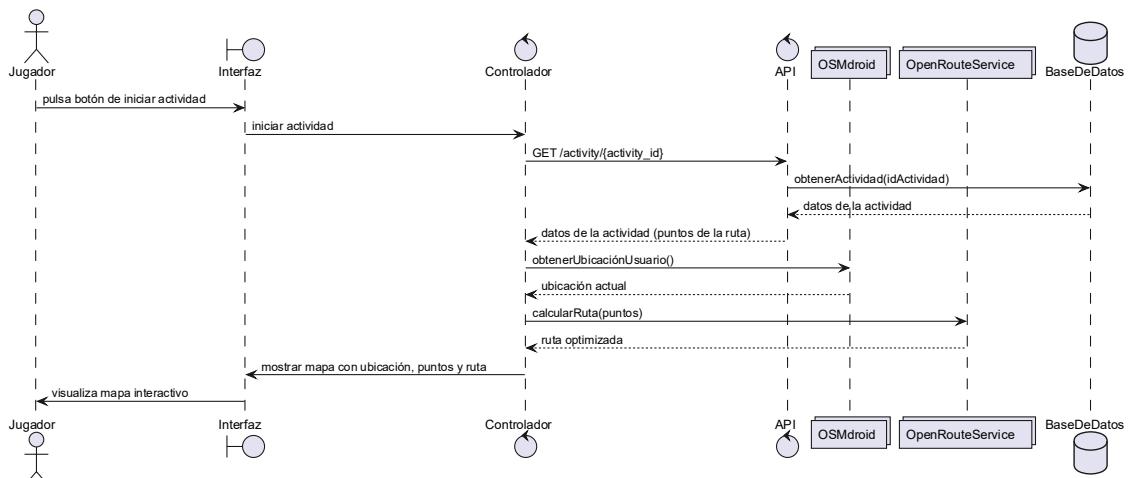


Figura 3.3.2.5: Diagrama de secuencia de inicio de actividad

Cuando el usuario pulsa el botón para iniciar una actividad, el sistema carga una vista con un mapa interactivo. En él se muestra la ubicación actual del usuario junto con los distintos puntos que conforman la ruta, utilizando OSMdroid para la visualización. Además, se traza el recorrido entre los puntos mediante OpenRouteService, ofreciendo al usuario una representación clara del camino que debe seguir.

Lectura de NFC:

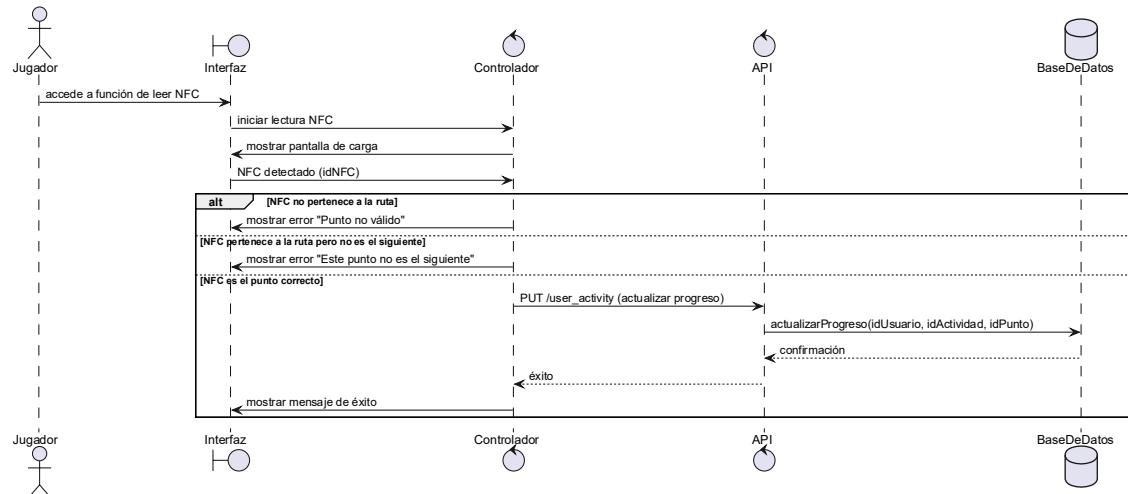


Figura 3.3.2.6: Diagrama de secuencia de lectura de NFC

El usuario accede a la función de lectura de dispositivos NFC. El sistema muestra una pantalla de carga mientras intenta detectar un dispositivo. Si se detecta un NFC que no forma parte de la ruta asignada, se muestra un mensaje de error informando al usuario.

Si el NFC pertenece a la ruta, pero no corresponde con el punto que debe registrarse en ese momento según el orden establecido, también se muestra un mensaje de error.

En caso de que el NFC detectado sea el correcto y esté en el orden esperado, el sistema actualiza el progreso del usuario mediante una solicitud PUT al elemento de registro de la API de un usuario en la actividad y que almacena su progreso, y muestra un mensaje confirmando el registro exitoso.

Visualización actividades creadas por el administrador:

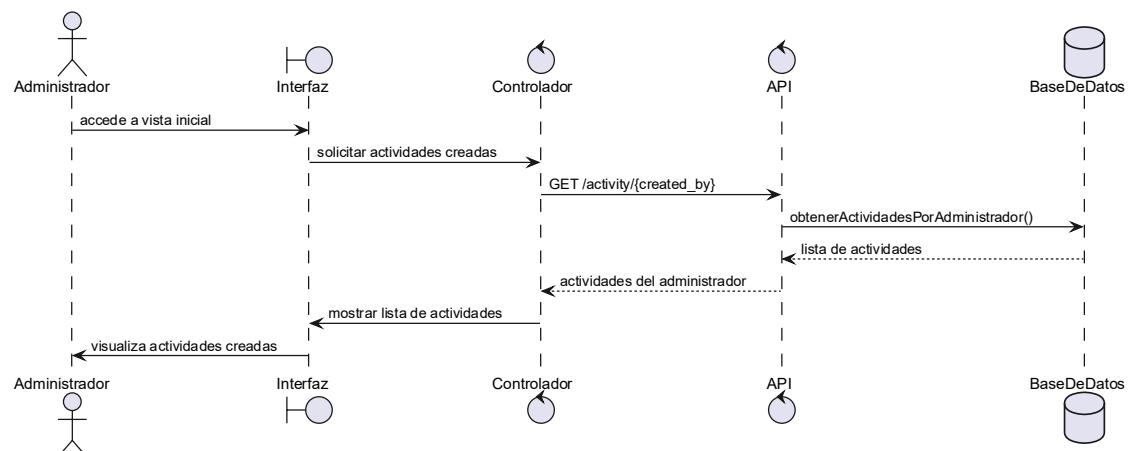


Figura 3.3.2.7: Diagrama de secuencia de visualización de actividades creadas por el administrador

El usuario administrador accede a la vista inicial. El sistema realiza una solicitud GET a la API para obtener las actividades que ha creado. A continuación, la aplicación muestra dichas actividades en formato de lista.

Creación de nueva actividad:

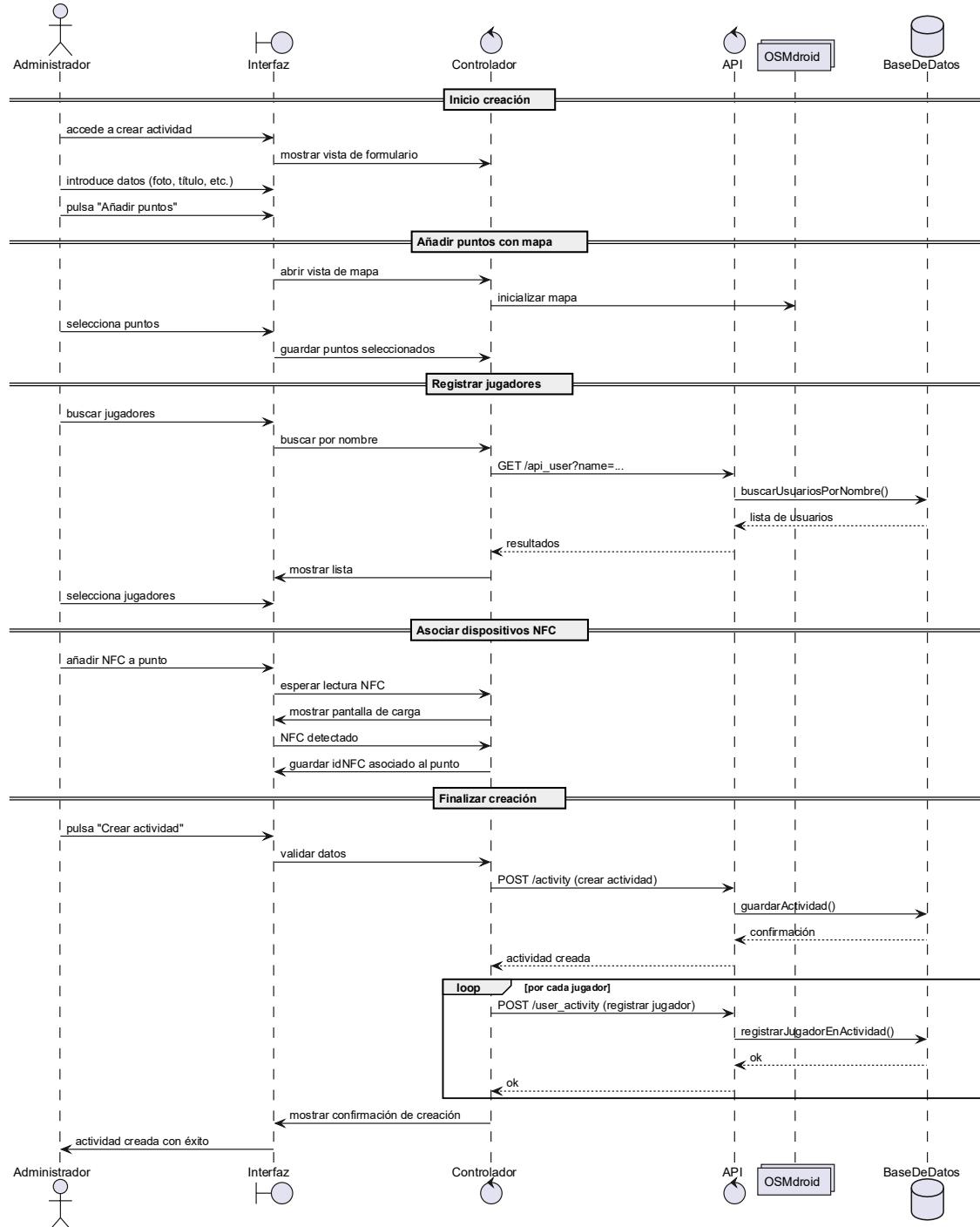


Figura 3.3.2.8: Diagrama de secuencia de creación de actividad

El administrador accede a la opción de crear una nueva actividad, tras lo cual se muestra una vista para introducir los datos generales de la actividad (foto, título, descripción y fecha). A continuación, puede añadir puntos a la ruta

mediante otra vista con un mapa interactivo (basado en OSMDroid), donde selecciona las ubicaciones correspondientes.

El administrador también tiene la opción de registrar jugadores en la actividad. Para ello, puede buscar usuarios por nombre, lo que implica una llamada GET a la API que devuelve una lista de posibles coincidencias.

Además, es posible asociar dispositivos NFC a los puntos de la ruta. Para cada punto, se muestra una vista que permite iniciar la lectura de un NFC. Al activar esta opción, se despliega una pantalla de carga mientras el sistema espera detectar un dispositivo NFC. Una vez detectado, este se registra y se vincula al punto correspondiente.

Una vez completados todos los datos, el administrador puede finalizar el proceso creando la actividad. Esto genera una llamada POST a la API para registrar la actividad, y una llamada POST para crear el registro por cada jugador registrado, vinculándolos con la nueva actividad.

Modificación de actividad existente:

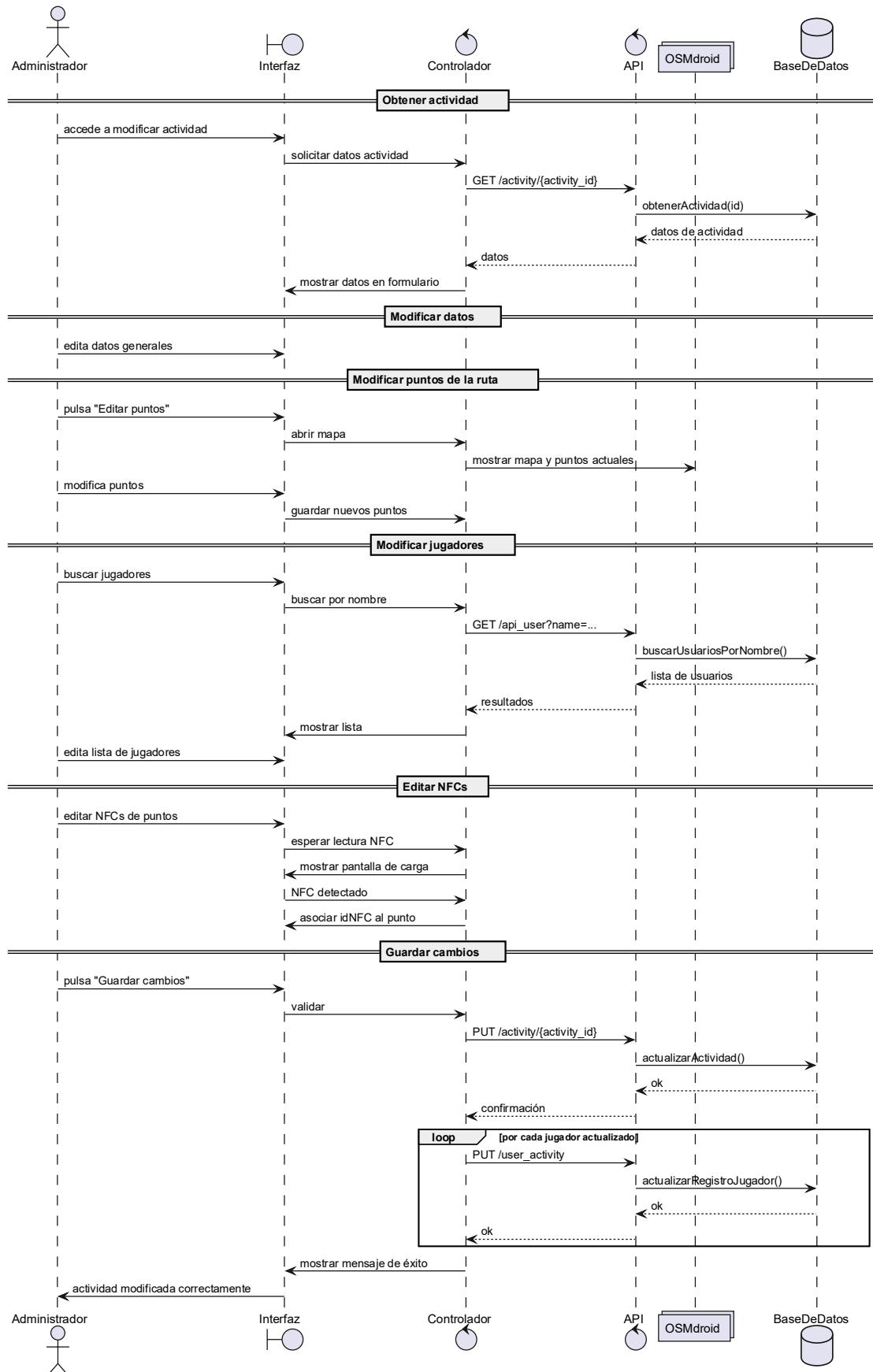


Figura 3.3.2.9: Diagrama de secuencia de modificación de actividad

El administrador accede a la opción de modificar una actividad existente. En primer lugar, el sistema realiza una llamada GET a la API para obtener los datos actuales de la actividad seleccionada. Con esta información, se cargan los datos en la vista de edición.

A continuación, el proceso es similar al de creación de la actividad. Se muestra una vista donde el administrador puede modificar los datos generales de la actividad, como la foto, el título, la descripción y la fecha. También puede actualizar los puntos de la ruta a través de una vista con mapa interactivo (basado en OSMdroid), donde puede añadir, eliminar o mover puntos según sea necesario.

El administrador puede gestionar los jugadores registrados en la actividad, realizando búsquedas por nombre. Estas búsquedas implican llamadas GET a la API para obtener la lista de usuarios coincidentes.

Asimismo, puede asociar o reasignar dispositivos NFC a los puntos de la ruta. Para ello, se muestra una vista con la lista de puntos y la opción de añadir un NFC. Al activar esta función, se muestra una pantalla de carga mientras el sistema espera la lectura de un dispositivo NFC. Una vez detectado, este se registra y se asocia al punto correspondiente.

Una vez realizadas las modificaciones necesarias, el administrador puede guardar los cambios. Esto genera una llamada PUT a la API para actualizar los datos de la actividad, y si se han actualizado los registros de jugadores, se realizan las llamadas POST o PUT necesarias.

Monitorización del progreso de los usuarios en una actividad:

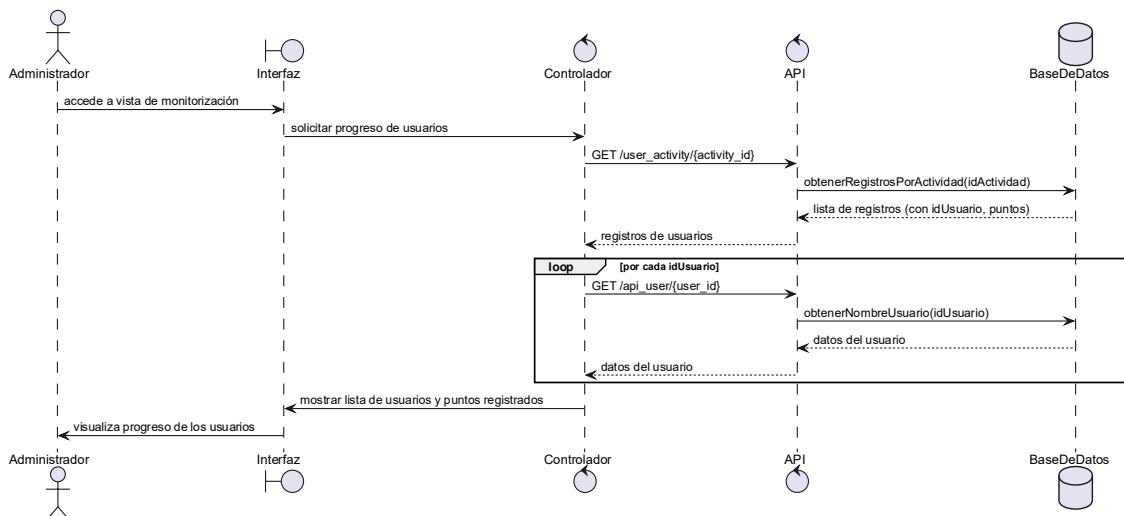


Figura 3.3.2.10: Diagrama de secuencia de monitorización de actividad

El administrador accede a la vista de monitorización de una actividad. El sistema realiza una solicitud GET a la API para obtener las relaciones de registro de todos los usuarios vinculados a dicha actividad.

A partir de esta información, se realizan llamadas adicionales a la API para obtener los nombres de los usuarios a partir de sus identificadores.

Con estos datos, la interfaz muestra una lista de los participantes junto con los puntos de la ruta que cada uno ha registrado, permitiendo así al administrador visualizar el progreso individual dentro de la actividad.

4 Implementación

Una vez completadas las fases de definición de requisitos, diseño de la arquitectura y diseño de interfaces de usuario, se da comienzo a la etapa de implementación. Antes de iniciar el desarrollo de la aplicación, se lleva a cabo un análisis detallado del código y los recursos existentes, con el objetivo de comprender la base tecnológica del proyecto y evaluar tanto las herramientas utilizadas como las posibles limitaciones y oportunidades de mejora de la solución inicial.

El punto de partida incluía una base de datos, una API para el sistema y una versión inicial de la aplicación móvil, que presentaba solo algunas de las funcionalidades. Se asumía que estos elementos eran plenamente funcionales y no requerirían modificaciones. Sin embargo, a medida que avanzó el desarrollo, se identificaron diversas deficiencias que ampliaron significativamente el alcance del proyecto y retrasaron el desarrollo de este. Entre ellas, la ausencia de atributos necesarios en los modelos de datos, métodos no definidos o incorrectamente implementados en la API, así como funcionalidades de la aplicación que, aunque se daban por completas, no funcionaban adecuadamente.

El desarrollo del software se organizó en torno a los dos perfiles principales de usuario: jugador y administrador. Esta división permitió abordar la implementación de forma progresiva y ordenada, enfocándose en cada etapa en un conjunto específico de funcionalidades y facilitando así la prueba y validación independientes de cada módulo.

En una primera fase se desarrollaron las funcionalidades destinadas al jugador, tales como el registro, la autenticación, la lectura de puntos NFC para participar en actividades, y la visualización y edición de su perfil. Estas funciones implicaban una integración estrecha con la base de datos y con algunos recursos del dispositivo, como el lector NFC y el sistema de localización, además del uso de la herramienta OSMdroid, lo que exigió asegurar su correcto funcionamiento en diversos contextos de uso.

Una vez finalizado este bloque, se abordó el desarrollo del perfil de administrador, encargado de crear y gestionar actividades, así como de realizar el seguimiento del progreso de los usuarios. Estas funcionalidades implicaban un mayor control sobre los datos y la creación de interfaces específicas adaptadas a las necesidades de gestión.

4.1.1 Cambios

Durante la implementación surgieron algunos problemas técnicos que requirieron ajustes sobre las decisiones iniciales como:

- **Gestión de mapas y rutas:** Las herramientas que habían sido previstas para la gestión de mapas dejaron de ser válidas debido a restricciones de uso. Para resolverlo, se optó por utilizar **OSMdroid**, una alternativa de código abierto y sin límites de uso, que además se integra fácilmente en

aplicaciones Android. Se usa esta librería para mostrar mapas, ubicación en tiempo real y marcadores.

- **Cálculo de rutas a pie:** Aunque OSMdroid permite visualizar mapas, no gestiona bien las rutas a pie. Por ello, se integró la API de **OpenRouteService**, que permite calcular rutas entre varios puntos optimizadas para peatones, mejorando la experiencia del usuario en actividades al aire libre. También se utiliza esta herramienta para traducir coordenadas a direcciones textuales.
- **Cambios en los modelos de datos:** A medida que se implementaban las funcionalidades, fue necesario modificar los modelos de datos, incorporando nuevos campos para almacenar información adicional (por ejemplo, datos de orden, identificador de dispositivo NFC asociado y fecha de lectura por el usuario, en los elementos de las listas de puntos para las rutas). También se redefinió el uso de ciertos parámetros para mejorar la coherencia de los datos y facilitar las consultas desde la aplicación. Esto implicó cambios en la base de datos, pero esta tarea está fuera del alcance del proyecto.
- **Cambios en la API:** La API inicial no cubría todas las necesidades funcionales del sistema, por lo que el backend fue ampliado con los métodos requeridos por la aplicación. Esta tarea está fuera del alcance del proyecto.

4.1.2 Organización del código

A continuación, se muestran los principales directorios en los que se organiza el código desarrollado:

- App
 - Java/es.upm.etssinf.gib.discoveryfit
 - Activity
 - Admin
 - Player
 - MainActivity
 - Login
 - Signup
 - Model
 - Api
 - ORSApi
 - User
 - Activity
 - UserActivity
 - NotedGeoPoint
 - Adapters
 - Utils
 - Res
 - Colors
 - Drawable
 - Layouts
 - Navigation
 - Values
 - Colors

-  String
- Styles

Dentro del paquete Activity se encuentran las clases relacionadas con las distintas actividades de la aplicación. Las clases MainActivity, Login y Signup, que representan la actividad principal y funcionalidades comunes para todos los usuarios, están ubicadas directamente en la raíz del paquete. Las actividades específicas para cada rol se agrupan en las carpetas Admin y Player, lo que facilita la organización y el mantenimiento del código al separar claramente las funcionalidades destinadas a cada tipo de usuario.

El paquete Model incluye las clases de modelos de datos: User, Activity, UserActivity y NotedGeoPoint. También contiene dos subpaquetes, Api y ORSApi, en los que se definen los clientes y las interfaces necesarias para interactuar con las APIs correspondientes. Además, el directorio alberga clases adaptadoras para mostrar los datos en la interfaz de usuario.

El paquete Utils agrupa clases auxiliares que proporcionan funcionalidades de soporte, como el envío de notificaciones o la clase Utils, que contiene métodos de utilidad empleados en distintos puntos de la aplicación.

El directorio Res contiene los recursos utilizados por la interfaz de usuario. Dentro de Drawable se almacenan las imágenes, iconos y logotipos, siguiendo una convención de nombres que distingue entre imágenes (img_) e iconos (ic_). El subdirectorío Layouts incluye los archivos .xml que definen la estructura y apariencia de las distintas pantallas de la aplicación. En Navigation se encuentra un archivo .xml que especifica la navegación entre las dos pantallas principales del usuario jugador mediante una barra de navegación. Por su parte, el directorio Values agrupa diferentes recursos: Colors y Styles, donde se definen los colores y estilos aplicados al diseño visual; y String, que contiene los archivos de texto en castellano e inglés, lo que permite la localización y adaptación del contenido textual a diferentes idiomas.

4.1.3 Configuración

Para el desarrollo del proyecto se utiliza **Gradle** como sistema de construcción. En el archivo build.gradle se definen las dependencias necesarias para incorporar distintas funcionalidades al proyecto. Algunas de las más relevantes son:

- **Retrofit** y **Gson**, para facilitar la conexión HTTP con APIs REST y la conversión automática de respuestas JSON a objetos Java:

```
implementation("com.squareup.retrofit2:retrofit:2.9.0")
implementation("com.squareup.retrofit2:converter-gson:2.9.0")
implementation("com.squareup.okhttp3:okhttp:4.9.0")
```

- **osmdroid**, para la visualización de mapas basados en OpenStreetMap dentro de la aplicación:

```
implementation("org.osmdroid:osmdroid-android:6.1.18")
• Glide, para la carga y gestión eficiente de imágenes:
implementation("com.github.bumptech.glide:glide:4.16.0")
annotationProcessor("com.github.bumptech.glide:compiler:4.16.0")
```

- **StepView**, para mostrar un componente visual que representa el progreso en pasos:

```
implementation("com.github.shuhart:stepview:1.5.1")
```

Por otro lado, en el archivo `AndroidManifest.xml` se declara la información fundamental que el sistema Android necesita para ejecutar la aplicación. Entre estos elementos se incluyen:

- **Permisos**, como el acceso a Internet, a la ubicación del dispositivo o a la lectura de etiquetas NFC.
- **Componentes** del sistema, como las actividades (Activity), servicios (Service) y receptores (BroadcastReceiver), que deben estar explícitamente registrados para poder ser utilizados.

4.1.4 Diagramas UML

En esta sección se presentan varios diagramas de clases UML que permiten visualizar de forma estructurada cómo se relacionan entre sí algunas de las clases más relevantes del sistema. El objetivo de incluir estos diagramas no es reflejar la totalidad del código, sino ofrecer una visión simplificada y representativa de los principales bloques lógicos involucrados en la implementación.

Se han seleccionado únicamente aquellas clases que intervienen en aspectos clave de la lógica del sistema: la comunicación con APIs externas y la gestión de las preferencias de autenticación. Para mantener la claridad y la legibilidad, los diagramas muestran solo las estructuras esenciales, omitiendo clases auxiliares o detalles internos que no aportan información significativa a nivel estructural, como getters, setters o clases anidadas que solo sirven para mapear respuestas de servicios externos.

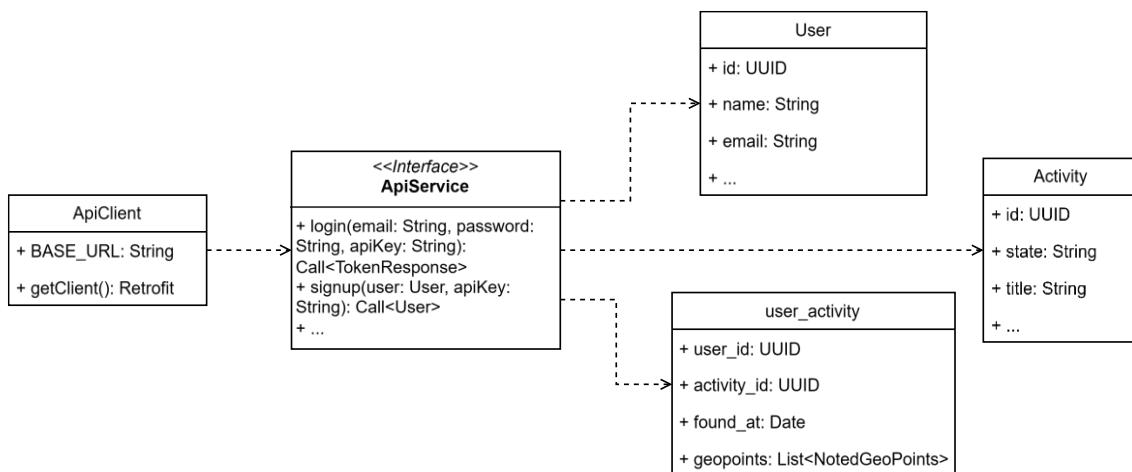


Figura 4.1.4.1: Diagrama de clases para la interacción con la API

La Figura 4.1.4.1 muestra el diagrama de clases correspondiente a la interacción con la **API** propia del sistema. En el centro se encuentra la interfaz `ApiService`, que define los métodos disponibles para el cliente, como el inicio de sesión (`login`) y el registro de usuarios (`signup`). Esta interfaz es utilizada por la clase `ApiClient`, encargada de construir la instancia de `Retrofit` que permite

ejecutar las peticiones HTTP. Los modelos User, Activity y user_activity representan las entidades principales que se intercambian con la API. Las líneas punteadas indican relaciones de uso o referencia entre las clases, permitiendo entender de forma visual cómo se estructuran y vinculan los datos en la lógica de comunicación del sistema.

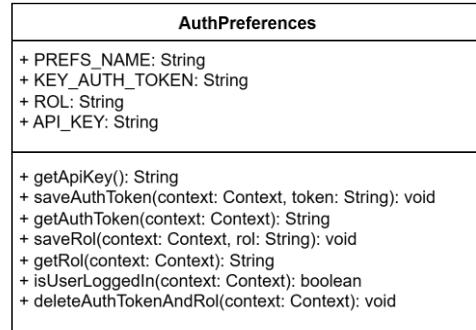


Figura 4.1.4.2: Diagrama de clase de AuthPreferences

En esta figura se representa la clase **AuthPreferences**, que se encarga de almacenar y recuperar la información de sesión del usuario, como el token de acceso a la API o su rol dentro de la aplicación. Aunque no se relaciona directamente con otras clases, su inclusión es relevante por su papel transversal en la gestión del estado de autenticación, funcionando como un componente central en la persistencia de datos sensibles.

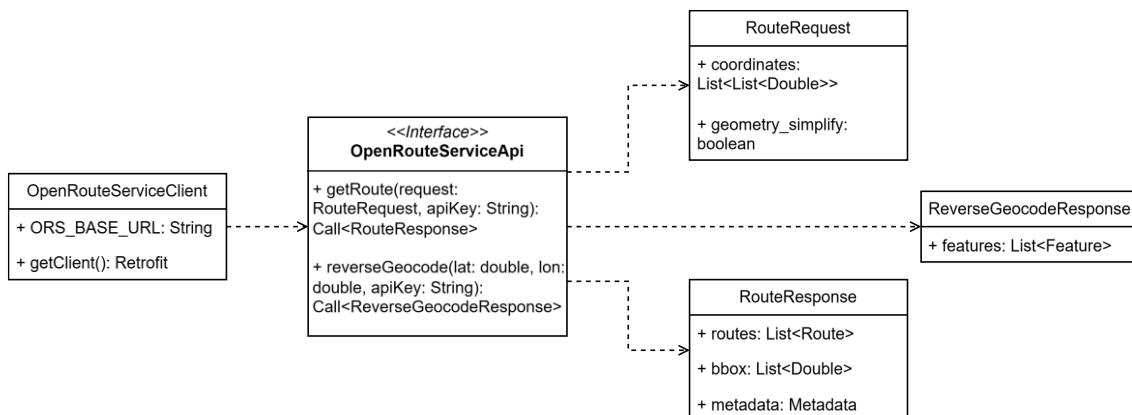


Figura 4.1.4.3: Diagrama de clases para la interacción con la API de OpenRouteService

La Figura 4.1.4.3 representa el diagrama de clases utilizado para la interacción con la API externa de **OpenRouteService**. En el centro se encuentra la interfaz OpenRouteServiceApi, que define los métodos disponibles para solicitar rutas (getRoute) y realizar geocodificación inversa (reverseGeocode). Estos métodos reciben como parámetros los modelos RouteRequest o las coordenadas junto con la clave de API, y devuelven objetos RouteResponse o ReverseGeocodeResponse, respectivamente. La clase OpenRouteServiceClient es responsable de construir la instancia de Retrofit que permite acceder a esta interfaz. Las clases RouteRequest y RouteResponse encapsulan los datos

necesarios para solicitar una ruta y procesar su respuesta, como la lista de coordenadas, las rutas resultantes o los metadatos. Por su parte, ReverseGeocodeResponse contiene una lista de objetos Feature, a partir de los cuales se extraen las direcciones asociadas a coordenadas concretas. Este diagrama permite visualizar de forma clara las dependencias entre los distintos componentes utilizados para comunicarse con este servicio externo.

4.1.5 Conexiones con la API

Todas las funcionalidades de la aplicación dependen en gran medida del acceso y la correcta gestión de los datos proporcionados por la API interna del sistema. Esta API actúa como intermediario entre la aplicación y la base de datos del servidor, permitiendo realizar operaciones esenciales como la autenticación de usuarios, la recuperación de información asociada a las actividades o la actualización del progreso de cada jugador. Por tanto, su integración es clave para el funcionamiento global de la aplicación, lo que justifica dedicar un apartado específico a explicar cómo se han establecido y probado estas conexiones.

Para gestionar la comunicación con la API se ha utilizado **Retrofit**, una biblioteca de cliente HTTP para Android que facilita la implementación de peticiones RESTful. Retrofit permite definir una interfaz con los métodos correspondientes a cada endpoint, simplificando tanto la estructura del código como su mantenimiento. A continuación, se muestra un fragmento de dicha interfaz, donde se declaran los métodos disponibles para interactuar con el servidor:

```
// Given an email and password, it requests a user to log in.  
// If successful, it returns a session token.  
@FormUrlEncoded  
@POST("token")  
Call<TokenResponse> login(@Field("username") String email,  
@Field("password") String password, @Header("apikey") String  
apikey);  
  
// Given a user, request their registration. If successful,  
// return the created user.  
@POST("signup")  
Call<User> signup(@Body User user, @Header("apikey") String  
apikey);  
  
// Requests all existing users. If successful, returns a list  
// of users.  
@GET("api_users")  
Call<List<User>> getAllUsers(@Header("Authorization") String  
token, @Header("apikey") String apikey);  
  
// Given a user ID, request all users with that ID. If  
// successful, return a list of matching users.  
@GET("api_users")  
Call<List<User>> getUserById(@Header("Authorization") String  
token, @Header("apikey") String apikey, @Query("user_id") UUID  
user_id);
```

```

/// Given a user, request to change that existing user's data to
/// the given one. If successful, return the modified user.
@PUT("api_users")
Call<User> editUser(@Body User user, @Header("Authorization")
String token, @Header("apikey") String apikey);

/// Requests the data of the logged-in user. If successful,
/// returns the logged-in user.
@GET("api_users/me")
Call<User> getCurrentUser(@Header("Authorization") String token,
@Header("apikey") String apikey);

/// Requests all existing activities in which the current user
/// is logged in or has created. If successful, it returns a
/// complete list of them.
@GET("activity")
Call<List<Activity>> getAllActivities(@Header("Authorization")
String token, @Header("apikey") String apikey);

/// Given an activity, request the activity to be created. If
/// successful, return the created activity.
@POST("activity")
Call<Activity> addActivity(@Body Activity activity,
@Header("Authorization") String token, @Header("apikey") String
apikey);

/// Given an activity, request to update the existing activity's
/// data with the given data. If successful, return the modified
/// activity.
@PUT("activity")
Call<Activity> updateActivity(@Body Activity activity,
@Header("Authorization") String token, @Header("apikey") String
apikey);

/// Requests all existing user_activities corresponding to the
/// current user. If successful, returns a list of all of them.
@GET("user_activity")
Call<List<UserActivity>> getAllUserActivity
(@Header("Authorization") String token, @Header("apikey") String
apikey);

/// Requests all existing user activities corresponding to the
/// current user and the activity with the given ID. If successful,
/// returns a list of all activities.
@GET("user_activity/")
Call<List<UserActivity>> getUserActivity_activityId
(@Header("Authorization") String token, @Header("apikey") String
apikey, @Query("activity_id") UUID activity_id);

/// Given an user_activity, request the user_activity to be
/// created. If successful, return the created user_activity.
@POST("/user_activity")
Call<UserActivity> addUserActivity(@Body UserActivity
userActivity, @Header("Authorization") String token,
@Header("apikey") String apikey);

```

```

/// Given an user_activity, request to update the existing
user_activity's data with the given data. If successful, return
the modified user_activity.
@PUT("/user_activity")
Call<UserActivity> updateUserActivity(@Body UserActivity
userActivity, @Header("Authorization") String token,
@Header("apikey") String apiKey);

```

Durante el desarrollo, se utilizó la herramienta **Postman** para probar y validar manualmente las distintas peticiones a la API. Esto permitió verificar el comportamiento esperado de los endpoints, depurar posibles errores en las respuestas del servidor y garantizar la estabilidad de las comunicaciones antes de integrarlas en la lógica de la aplicación.

4.1.6 Uso de Open Street Map (OSMDroid)

Para la gestión de mapas dentro de la aplicación se utiliza la librería org.osmdroid.views.MapView, perteneciente al proyecto OSMDroid, una solución de código abierto que permite integrar mapas interactivos en aplicaciones Android sin depender de servicios con restricciones de uso. Esta librería proporciona una interfaz flexible para visualizar mapas con distintas capas de información, como la escala gráfica o la brújula de orientación, que pueden activarse o desactivarse.

Entre sus funcionalidades más destacadas se encuentra la posibilidad de realizar interacciones intuitivas con el mapa, como desplazamiento libre, zoom y rotación, lo que mejora considerablemente la experiencia de navegación. Además, permite mostrar en tiempo real la ubicación del usuario utilizando el sistema de localización del dispositivo, así como añadir marcadores en puntos específicos del mapa, lo cual resulta especialmente útil para representar localizaciones relacionadas con las actividades que ofrece la aplicación.

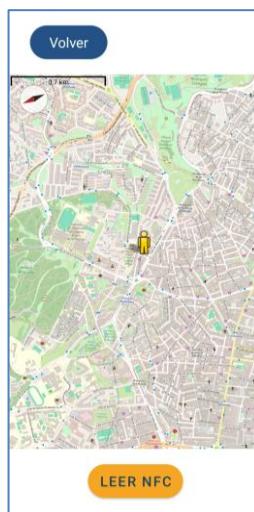


Figura 4.1.6.1: Mapa generado mediante OSMDroid

Gracias a estas capacidades, MapView cumple un papel fundamental en la interfaz de usuario, facilitando la representación geográfica de la información y permitiendo una interacción visual y dinámica con los datos espaciales.

4.1.7 Uso de OpenRouteService

En este proyecto se utiliza la **API de OpenRouteService (ORS)** para incorporar funcionalidades relacionadas con la geolocalización y el cálculo de rutas. Concretamente, se emplean dos de los múltiples servicios que ofrece esta API:

- El cálculo de la **mejor ruta a pie** entre varios puntos geográficos.
- La funcionalidad de **geocodificación inversa**, que permite obtener una dirección textual a partir de unas coordenadas (latitud y longitud).

Estas funcionalidades se integran mediante una interfaz de servicio Retrofit que define las peticiones necesarias. A continuación, se muestran los métodos declarados en dicha interfaz:

```
@POST("v2/directions/foot-walking")
Call<RouteResponse> getRoute(@Header("Authorization") String
apiKey, @Body RouteRequest request);

@GET("geocode/reverse")
Call<ReverseGeocodeResponse> reverseGeocode(@Query("api_key")
String apiKey, @Query("point.lat") double lat,
@Query("point.lon") double lon);
```

El primer método permite obtener una ruta optimizada para caminar entre varios puntos especificados, enviando un objeto RouteRequest en el cuerpo de la petición. El segundo método realiza una consulta inversa para traducir una coordenada geográfica en una dirección legible, como una calle o ciudad.



Figura 4.1.7.1: Ruta generada mediante ORS

4.1.8 Documentación del código

Todo el código desarrollado ha sido documentado siguiendo el estándar **Javadoc**, con descripciones claras de las clases, métodos y parámetros, lo que facilita su comprensión, reutilización y mantenimiento por parte de otros desarrolladores o futuros responsables del proyecto. Esta documentación resulta especialmente útil en clases clave del sistema, como las encargadas de gestionar la lógica de negocio, las peticiones a la API o la manipulación de datos.

Durante el desarrollo se ha utilizado un repositorio en **GitHub** para el control de versiones y la gestión del código fuente. El repositorio incluye un archivo **README** que proporciona una visión general del proyecto, instrucciones de uso y detalles relevantes.

Además, se han incorporado **mensajes de log** en puntos estratégicos del flujo de ejecución, lo que permite monitorizar el comportamiento de la aplicación en tiempo real y detectar rápidamente errores o situaciones inesperadas.

Este enfoque contribuye significativamente a la **mantenibilidad** del sistema, reduce el tiempo necesario para **localizar fallos**, y sienta las bases para una posible **ampliación futura** del proyecto.

5 Pruebas y análisis de resultados

Con el objetivo de verificar el correcto funcionamiento del sistema en su conjunto, se realizaron pruebas de sistema que permitieron comprobar la interacción adecuada entre todos los módulos y el cumplimiento de los requisitos funcionales definidos.

Estas pruebas se ejecutaron manualmente en un entorno controlado utilizando Android Studio, con un dispositivo Oppo Reno 2 como terminal de pruebas. Se adoptó una estrategia de pruebas funcionales de caja negra, basadas en los requisitos establecidos. Además, se llevaron a cabo pruebas de regresión de las tareas principales tras aplicar correcciones, con el fin de garantizar que las funcionalidades previamente implementadas continuaban funcionando correctamente.

A continuación, se presentan las 31 pruebas realizadas, que cubren el 100 % de los casos de uso definidos:

CP_01 Registro de usuario sin datos	
Caso de uso asociado	CU_01: Registrar usuario
Entrada	Nombre “”/”juagdor5”, email “”, contraseña “”
Resultado esperado	Mensaje de error indicando la ausencia de datos necesarios.
Resultado obtenido	Mensaje de error indicando la ausencia de datos necesarios.
Estado	Aprobada.

Tabla 5.1: Caso de prueba 01



Figura 5.1 y 5.2: Imágenes de caso de prueba 01

CP_02 Registro de usuario con formato incorrecto	
Caso de uso	CU_01: Registrar usuario
Entrada	Nombre “jugador5”, email “jugador”, contraseña “”
Resultado esperado	Mensaje de error indicando el error y el formato correcto.
Resultado obtenido	Mensaje de error indicando el error y el formato correcto.
Estado	Aprobada.

Tabla 5.2: Caso de prueba 02

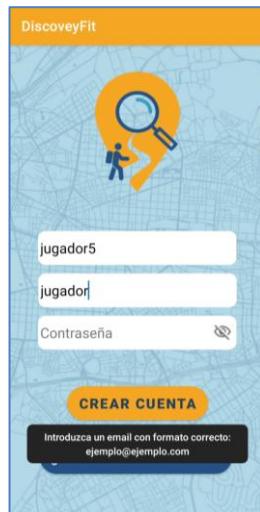


Figura 5.3: Imagen de caso de prueba 02

CP_03 Registro de usuario con correo registrado previamente	
Caso de uso asociado	CU_01: Registrar usuario
Entrada	Nombre “jugador5”, email “jugador2@email”, contraseña “jugador5”
Resultado esperado	Mensaje de error indicando el error y el formato correcto.
Resultado obtenido	Mensaje de error indicando el error y el formato correcto.
Estado	Aprobada.

Tabla 5.3: Caso de prueba 03



Figura 5.4: Imagen de caso de prueba 03

CP_04 Registro de usuario correcto	
Caso de uso asociado	CU_01: Registrar usuario
Entrada	Nombre “jugador5”, email “jugador5@jugador.com”, contraseña “jugador5”
Resultado esperado	Registro correcto, mensaje informativo y paso a la pantalla de inicio de sesión.
Resultado obtenido	Registro correcto, mensaje informativo y paso a la pantalla de inicio de sesión.
Estado	Aprobada.

Tabla 5.4: Caso de prueba 04



Figura 5.5 y 5.6: Imágenes de caso de prueba 04

CP_05 Inicio de sesión con campos vacíos	
Caso de uso asociado	CU_02: Inicio de sesión
Entrada	Email "", contraseña ""
Resultado esperado	Mensaje de error indicando la ausencia de datos necesarios.
Resultado obtenido	Mensaje de error indicando la ausencia de datos necesarios.
Estado	Aprobada.

Tabla 5.5: Caso de prueba 05

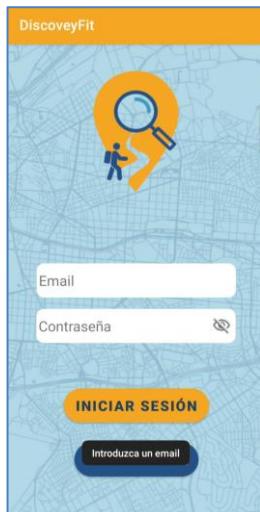


Figura 5.7: Imagen de caso de prueba 05

CP_06 Inicio de sesión con credenciales incorrectas	
Caso de uso asociado	CU_02: Inicio de sesión
Entrada	email “jugador5@jugador.com”, contraseña “jugador3”
Resultado esperado	Mensaje de error indicando credenciales incorrectas.
Resultado obtenido	Mensaje de error indicando credenciales incorrectas.
Estado	Aprobada.

Tabla 5.6: Caso de prueba 06



Figura 5.8: Imagen de caso de prueba 06

CP_07 Inicio de sesión correcto	
Caso de uso asociado	CU_02: Inicio de sesión
Entrada	email “jugador5@jugador.com”, contraseña “jugador5”
Resultado esperado	Mensaje informativo y paso a pantalla inicial de la aplicación según el rol.
Resultado obtenido	Mensaje informativo y paso a pantalla inicial de la aplicación según el rol.
Estado	Aprobada.

Tabla 5.7: Caso de prueba 07

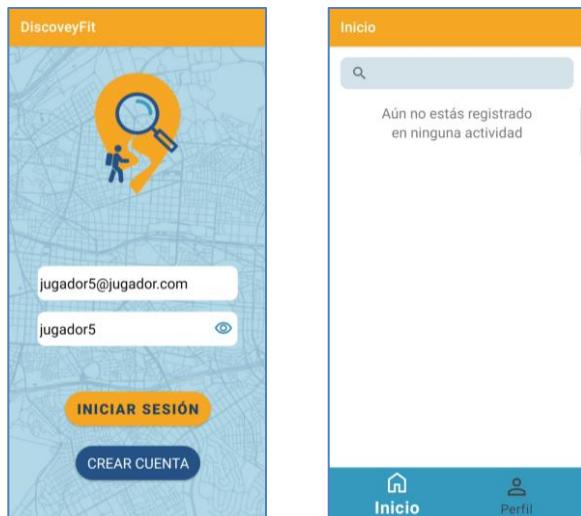


Figura 5.9 y 5.10: Imágenes de caso de prueba 07

CP_08 Cerrar sesión por usuario jugador	
Caso de uso	CU_03: Cerrar sesión
Entrada	-
Resultado esperado	Mensaje informativo y paso a pantalla de inicio de sesión.
Resultado obtenido	Mensaje informativo y paso a pantalla de inicio de sesión.
Estado	Aprobada.
Comentarios	Durante las pruebas se detectó que el mensaje mostrado al cerrar sesión no estaba traducido correctamente. Una vez identificado el error, se procedió a corregirlo y se repitió la prueba para verificar su resolución. A continuación, se muestran las capturas de pantalla del mensaje antes y después de la corrección.

Tabla 5.8: Caso de prueba 08

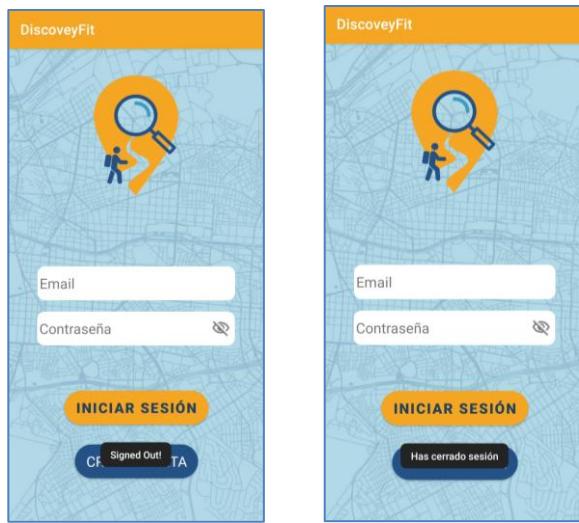


Figura 5.11 y 5.12: Imágenes de caso de prueba 08

CP_09 Cerrar sesión por usuario administrador	
Caso de uso asociado	CU_03: Cerrar sesión
Entrada	-
Resultado esperado	Mensaje informativo y paso a pantalla de inicio de sesión.
Resultado obtenido	Mensaje informativo y paso a pantalla de inicio de sesión.
Estado	Aprobada.
Comentarios	Durante las pruebas se detectó que el mensaje mostrado al cerrar sesión no estaba traducido correctamente. Una vez identificado el error, se procedió a corregirlo y se repitió la prueba para verificar su resolución. A continuación, se muestran las capturas de pantalla del mensaje antes y después de la corrección.

Tabla 5.9: Caso de prueba 09

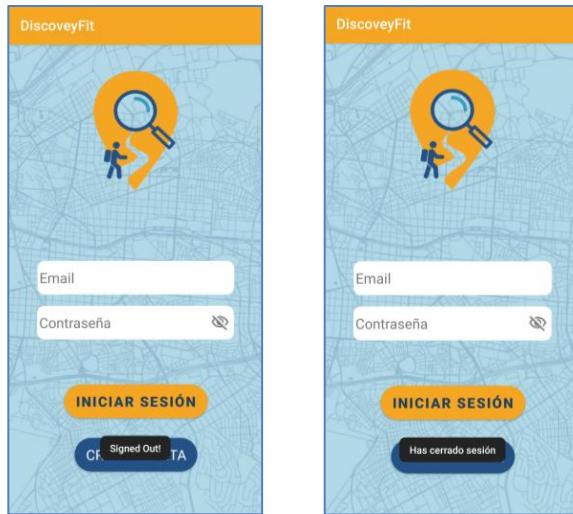


Figura 5.13 y 5.14: Imágenes de caso de prueba 09

CP_10 Visualización de perfil de usuario	
Caso de uso asociado	CU_04: Visualizar información de usuario
Entrada	-
Resultado esperado	Visualización de foto, nombre, email y número de actividades realizadas del usuario.
Resultado obtenido	Visualización de foto, nombre, email y número de actividades realizadas del usuario.
Estado	Aprobada.

Tabla 5.10: Caso de prueba 10



Figura 5.14: Imagen de caso de prueba 10

CP_11 Modificación de perfil de usuario con campos vacíos	
Caso de uso asociado	CU_05: Modificación de información de usuario
Entrada	Nombre “”
Resultado esperado	Mensaje de error indicando la ausencia de datos necesarios.
Resultado obtenido	Mensaje de error indicando la ausencia de datos necesarios.
Estado	Aprobada.

Tabla 5.11: Caso de prueba 11



Figura 5.15: Imagen de caso de prueba 11

CP_12 Modificación de perfil de usuario con campos correctos	
Caso de uso asociado	CU_05: Modificación de información de usuario
Entrada	Nombre “jugador22” e imagen de galería
Resultado esperado	Mensaje informativo, modificación de datos y paso a pantalla de visualización de perfil de usuario.
Resultado obtenido	Se modifican los datos en la base de datos pero la respuesta de la API indica un error y por tanto la lógica de la aplicación opera respecto a dicho error.
Estado	Fallida.
Comentarios	<p>La interacción entre las interfaces y la lógica del código relacionada con esta funcionalidad funciona correctamente.</p> <p>Sin embargo, el método PUT /api_user de la API del sistema no responde de forma adecuada, lo que ha impedido que la prueba se desarrolle como se</p>

	esperaba. Este problema escapa al alcance del presente Trabajo de Fin de Grado.
--	---

Tabla 5.12: Caso de prueba 12



Figura 5.16: Imagen de caso de prueba 13

CP_13 Visualización de actividades por usuario jugador no registrado en ninguna actividad	
Caso de uso	CU_06: Visualizar actividades en las que está registrado
Entrada	-
Resultado esperado	Vista con texto informativo indicando que no hay ninguna actividad que mostrar.
Resultado obtenido	Vista con texto informativo indicando que no hay ninguna actividad que mostrar.
Estado	Aprobada.

Tabla 5.13: Caso de prueba 13

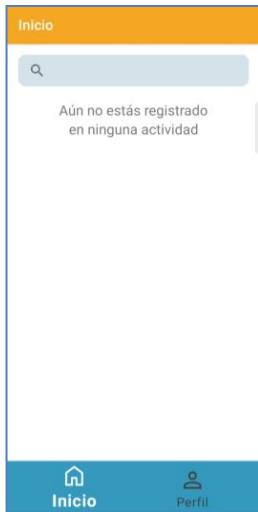


Figura 5.17: Imagen de caso de prueba 13

CP_14 Visualización de actividades por usuario jugador	
Caso de uso	CU_06: Visualizar actividades en las que está registrado
Entrada	-
Resultado esperado	Se muestra la lista de actividades en las que está registrado el usuario, en cada una de ellas se muestra la foto, el título, la fecha y el estado.
Resultado obtenido	Se muestra la lista de actividades en las que está registrado el usuario, en cada una de ellas se muestra la foto, el título, la fecha y el estado. Aquellas completadas, además del texto de estado, aparecen con una capa verde sobre la imagen.
Estado	Aprobada.

Tabla 5.14: Caso de prueba 14



Figura 5.18: Imagen de caso de prueba 14

CP_15 Visualización de actividad	
Caso de uso asociado	CU_07: Visualización de actividad
Entrada	-
Resultado esperado	Se muestra la información detallada de la actividad: foto, título, descripción, fecha, estado y ruta.
Resultado obtenido	Se muestra la información detallada de la actividad: foto, título, descripción, fecha, estado y ruta. Aquellas actividades completadas, además del texto de estado, aparecen con una capa verde sobre la imagen y el botón de iniciar desactivado.
Estado	Aprobada.

Tabla 5.15: Caso de prueba 15



Figura 5.19 y 5.20: Imágenes de caso de prueba 15

CP_16 Iniciar actividad	
Caso de uso asociado	CU_08: Iniciar actividad
Entrada	-
Resultado esperado	Se abre el mapa mostrando la posición actual del jugador y la ruta de la actividad.
Resultado obtenido	Se abre el mapa mostrando la posición actual del jugador y la ruta de la actividad. Para cada punto se puede ver el orden en la ruta si ha sido leído o no.
Estado	Aprobada.

Tabla 5.16: Caso de prueba 16



Figura 5.21: Imagen de caso de prueba 16

CP_17 Leer dispositivo NFC del siguiente punto	
Caso de uso asociado	CU_09: Leer dispositivo NFC
Entrada	NFC
Resultado esperado	Actualización del progreso del usuario y mensaje informativo.
Resultado obtenido	Se muestra una pantalla de carga. Una vez leído el siguiente punto válido de la ruta se muestra un mensaje informativo, si es el último punto de la ruta se indica. Se actualiza el progreso del usuario. Pero no se actualiza el progreso del usuario.
Estado	Fallida.
Comentario	<p>La interacción entre las interfaces y la lógica del código relacionada con esta funcionalidad funciona correctamente.</p> <p>El error en esta prueba se debe a un error en el método PUT en /user_activity de la API del sistema. Este problema escapa al alcance del presente Trabajo de Fin de Grado.</p>

Tabla 5.17: Caso de prueba 17

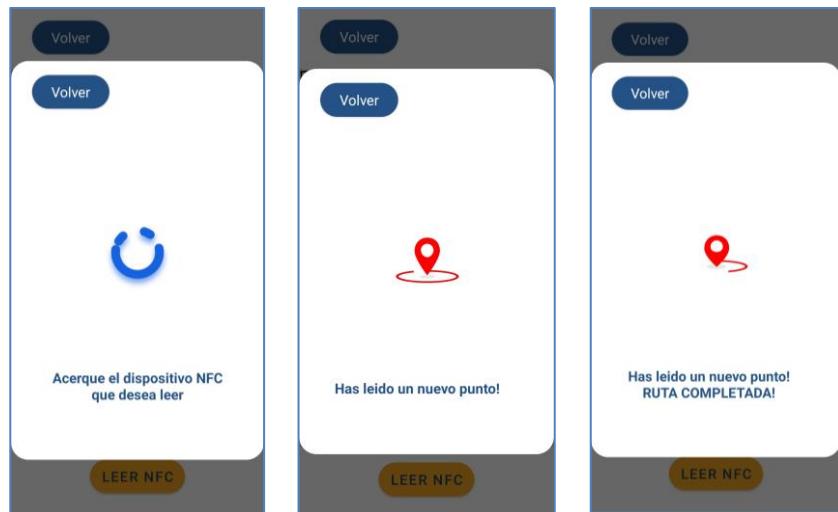


Figura 5.22, 5.23 y 5.24: Imágenes de caso de prueba 17

CP_18 Leer dispositivo NFC de un punto con anteriores pendientes	
Caso de uso asociado	CU_09: Leer dispositivo NFC
Entrada	NFC
Resultado esperado	Mensaje de error indicando que no se puede leer el punto y se informa de los puntos pendientes que se deben leer antes.
Resultado obtenido	Mensaje de error indicando que no se puede leer el punto y se informa de los puntos pendientes que se deben leer antes.
Estado	Aprobada.

Tabla 5.18: Caso de prueba 18

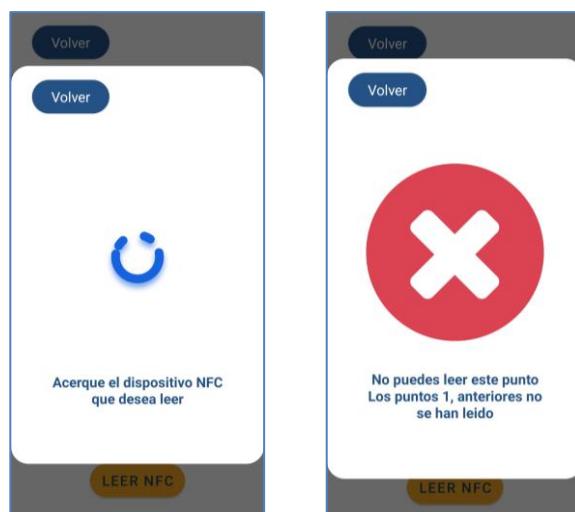


Figura 5.25 y 5.26: Imágenes de caso de prueba 18

CP_19 Leer dispositivo NFC no contenido en la ruta	
Caso de uso asociado	CU_09: Leer dispositivo NFC
Entrada	NFC
Resultado esperado	Mensaje de error indicando que ese dispositivo no corresponde a la ruta.
Resultado obtenido	Mensaje de error indicando que ese dispositivo no corresponde a la ruta.
Estado	Aprobada.

Tabla 5.19: Caso de prueba 19

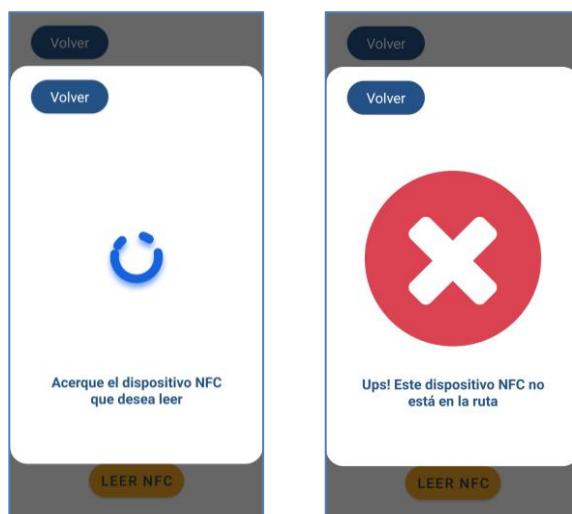


Figura 5.27 y 5.28: Imágenes de caso de prueba 19

CP_20 Creación de actividad con campos vacíos	
Caso de uso asociado	CU_10: Crear actividad
Entrada	-
Resultado esperado	Se muestran mensajes de error indicando los campos obligatorios que deben rellenarse para poder crear la actividad.
Resultado obtenido	Se muestran mensajes de error indicando los campos obligatorios que deben rellenarse para poder crear la actividad.
Estado	Aprobada.

Tabla 5.20: Caso de prueba 20



Figura 5.29, 5.30, 5.31 y 5.32: Imágenes de caso de prueba 20

CP_21 Creación de actividad sin activar	
Caso de uso	CU_10: Crear actividad
Entrada	Actividad con datos correctos: foto, título, descripción, fecha, cuatro puntos, dos jugadores y todos los dispositivos NFC asociados.
Resultado esperado	Se crea la nueva actividad y los registros de los jugadores seleccionados. La actividad no se muestra a los jugadores.
Resultado obtenido	Se crea la nueva actividad y los registros de los jugadores seleccionados. La actividad no se muestra a los jugadores.
Estado	Aprobada.
Comentarios	A continuación, se muestran las pantallas de todo el proceso: la selección de puntos en el mapa, el registro de usuarios, el registro de dispositivos NFC y la actividad completa. Se muestran los mensajes informativos y la vista del administrador.

Tabla 5.21: Caso de prueba 21

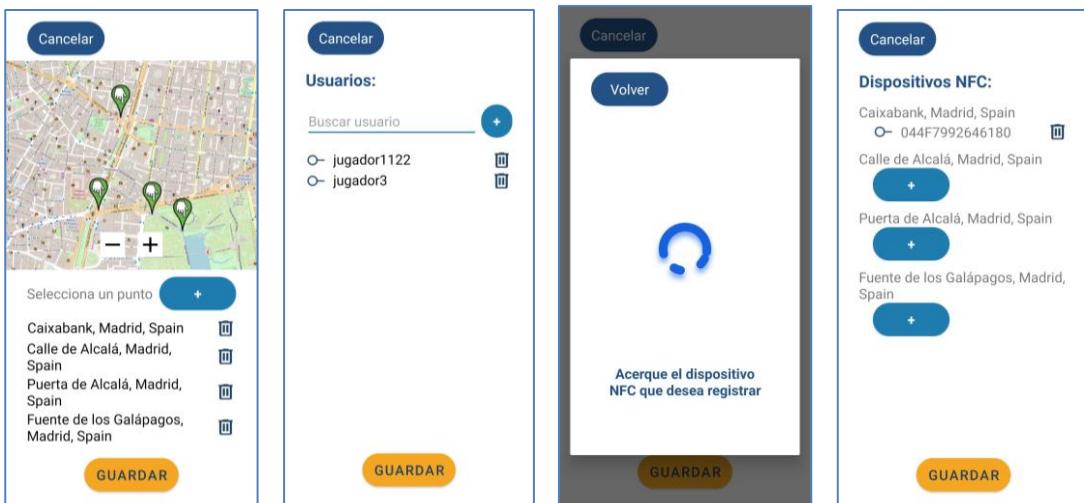


Figura 5.33, 5.34, 5.35 y 5.36: Imágenes de caso de prueba 21



Figura 5.37, 5.38, 5.39 y 5.40: Imágenes de caso de prueba 21

CP_22 Creación de actividad activada	
Caso de uso	CU_10: Crear actividad
Entrada	Actividad con datos correctos: título, descripción, fecha, tres puntos, dos jugadores y todos los dispositivos NFC asociados.

Resultado esperado	Se crea la nueva actividad y los registros de los jugadores seleccionados. La actividad se muestra a los jugadores.
Resultado obtenido	Se crea la nueva actividad y los registros de los jugadores seleccionados. La actividad se muestra a los jugadores.
Estado	Aprobada.
Comentarios	A continuación, se muestran las pantallas de todo el proceso: la selección de puntos en el mapa, el registro de usuarios, el registro de dispositivos NFC y la actividad completa. Se muestran los mensajes informativos y la vista del administrador.

Tabla 5.22: Caso de prueba 22

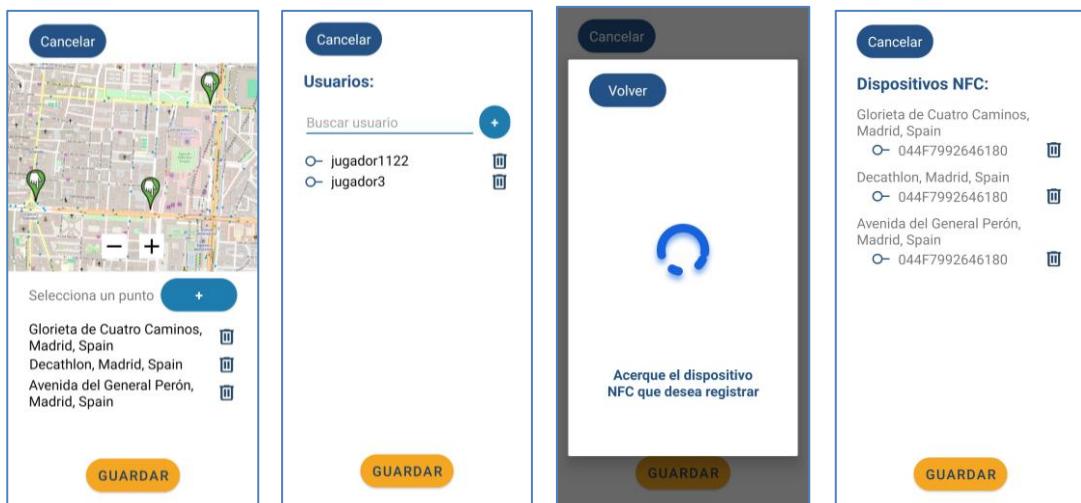


Figura 5.41, 5.42, 5.43 y 5.44: Imágenes de caso de prueba 22



Figura 5.45, 5.46, 5.47 y 5.48: Imágenes de caso de prueba 22

CP_23 Visualización de actividades por administrador que no ha creado ninguna	
Caso de uso	CU_11: Visualizar actividades creadas
Entrada	-
Resultado esperado	Vista con texto informativo indicando que no hay ninguna actividad que mostrar.
Resultado obtenido	Vista con texto informativo indicando que no hay ninguna actividad que mostrar.
Estado	Aprobada.

Tabla 5.23: Caso de prueba 23



Figura 5.49: Imagen de caso de prueba 23

CP_24 Visualizar actividades creadas por el administrador	
Caso de uso	CU_11: Visualizar actividades creadas
Entrada	-
Resultado esperado	Se visualizan las actividades creadas por el usuario. De cada actividad se muestra la foto, título, dirección final, fecha y estado: activa o inactiva.
Resultado obtenido	Se visualizan las actividades creadas por el usuario. De cada actividad se muestra la foto, título, dirección final, fecha y estado: activa o inactiva. El método de la API del sistema devuelve las actividades tantas veces como jugadores registrados tengan. Algunas actividades no se muestran y otras aparecen duplicadas.
Estado	Fallida.
Comentarios	El estado no se traduce correctamente. Se corrige añadiendo el texto a los ficheros de traducción y se repite la prueba. A continuación, se muestran las fotos, con el error y una vez ya arreglado. Respecto al segundo error, la interacción entre las interfaces y la lógica del código relacionada con esta funcionalidad funciona correctamente. El error en esta prueba se debe a un error en el método GET en /activity de la API del sistema para el usuario administrador. Este problema escapa al alcance del presente Trabajo de Fin de Grado.

Tabla 5.24: Caso de prueba 24



Figura 5.50 y 5.51: Imágenes de caso de prueba 24

CP_25 Visualizar actividad creada por el administrador	
Caso de uso asociado	CU_12: Visualizar actividad creada
Entrada	-
Resultado esperado	Se muestra la información detallada de la actividad: foto, título, descripción, fecha, estado y ruta. Además, se presentan cuatro botones: monitorizar, editar, desactivar y borrar.
Resultado obtenido	Se muestra la información detallada de la actividad: foto, título, descripción, fecha, estado y ruta. Además, se presentan cuatro botones: monitorizar, editar, desactivar y borrar.
Estado	Aprobada.

Tabla 5.25: Caso de prueba 25



Figura 5.52: Imagen de caso de prueba 25

CP_26 Activar actividad	
Caso de uso asociado	CU_13: Activar y desactivar una actividad
Entrada	-
Resultado esperado	Se modifica el estado de la actividad en la vista y en la API del sistema.
Resultado obtenido	No se modifica el estado de la actividad y se muestra un mensaje de error.
Estado	Fallida.
Comentarios	<p>La interacción entre las interfaces y la lógica del código relacionada con esta funcionalidad funciona correctamente.</p> <p>El error en esta prueba se debe a un error en el método PUT en /activity de la API del sistema. Este problema escapa al alcance del presente Trabajo de Fin de Grado.</p>

Tabla 5.26: Caso de prueba 26

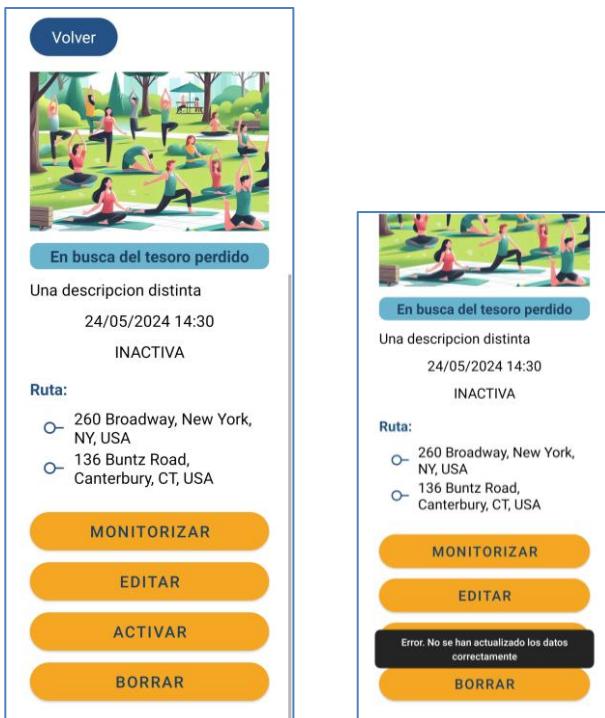


Figura 5.53 y 5.54: Imágenes de caso de prueba 26

CP_27 Desactivar actividad	
Caso de uso asociado	CU_13: Activar y desactivar una actividad
Entrada	-
Resultado esperado	Se modifica el estado de la actividad en la vista y en la API del sistema.
Resultado obtenido	No se modifica el estado de la actividad y se muestra un mensaje de error.
Estado	Fallida.
Comentarios	<p>La interacción entre las interfaces y la lógica del código relacionada con esta funcionalidad funciona correctamente.</p> <p>El error en esta prueba se debe a un error en el método PUT en /activity de la API del sistema. Este problema escapa al alcance del presente Trabajo de Fin de Grado.</p>

Tabla 5.27: Caso de prueba 27

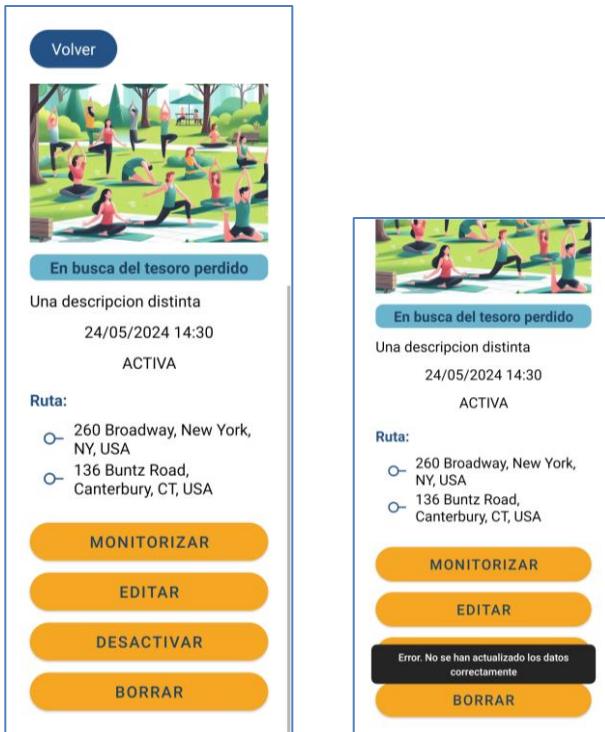


Figura 5.55 y 5.56: Imágenes de caso de prueba 27

CP_28 Modificar actividad con campos vacíos	
Caso de uso	CU_14: Modificar actividad asociado
Entrada	Se deja el campo título vacío.
Resultado esperado	Mensaje de error indicando los campos obligatorios que deben estar completos para guardar la actividad.
Resultado obtenido	Mensaje de error indicando los campos obligatorios que deben estar completos para guardar la actividad.
Estado	Aprobada.

Tabla 5.28: Caso de prueba 28

ACTIVE

Añadir título

Una descripción distinta

Fecha y hora:

24/05/2024 14:30

Ruta: [Editar](#)

- 260 Broadway, New York, NY, USA
- 136 Buntz Road, Canterbury, CT, USA

Usuarios: [Editar](#)

- jugador1122
- Test1

Dispositivos NFC: [Editar](#)

- cndisckn
- Es necesario un título

GUARDAR CAMBIOS

Figura 5.57: Imagen de caso de prueba 28

CP_29 Modificar actividad con datos correctos	
Caso de uso	CU_14: Modificar actividad asociado
Entrada	Actividad modificada con datos correctos. Se modifica la descripción.
Resultado esperado	Se modifican los datos de la actividad que sean necesarios en la API del sistema.
Resultado obtenido	Se guarda la modificación en la API del sistema, pero el método de la API devuelve el código de error 500 y la aplicación muestra un mensaje de error.
Estado	Fallida.
Comentarios	<p>La interacción entre las interfaces y la lógica del código relacionada con esta funcionalidad funciona correctamente.</p> <p>El error en esta prueba se debe a un error en los métodos PUT en /activity y /user_activity de la API del sistema. Este problema escapa al alcance del presente Trabajo de Fin de Grado.</p>

Tabla 5.29: Caso de prueba 29

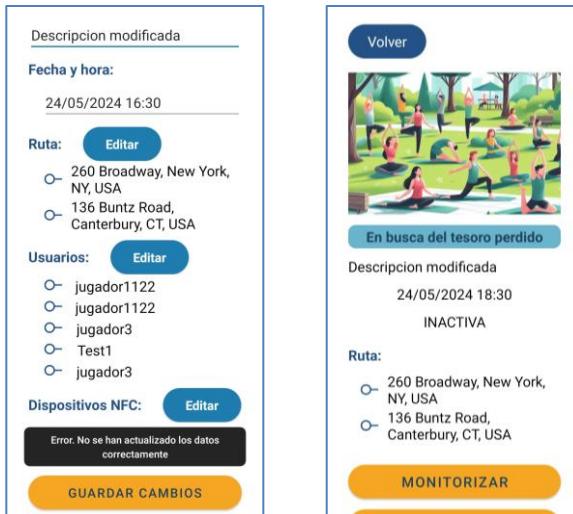


Figura 5.58 y 5.59: Imágenes de caso de prueba 29

CP_30 Visualizar progreso de los usuarios en una actividad	
Caso de uso asociado	CU_15: Visualizar progreso de los usuarios
Entrada	-
Resultado esperado	Se muestran los jugadores de la actividad y su progreso. Se indican en verde con un check los puntos registrados, en amarillo con el número del punto el siguiente punto a leer y sin círculo los siguientes pendientes.
Resultado obtenido	Se muestran los jugadores de la actividad y su progreso. Se indican en verde con un check los puntos registrados, en amarillo con el número del punto el siguiente punto a leer y sin círculo los siguientes pendientes.
Estado	Aprobada.

Tabla 5.30: Caso de prueba 30

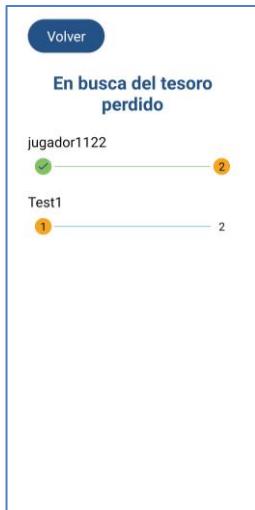


Figura 5.60: Imagen de caso de prueba 30

CP_31 Eliminar actividad	
Caso de uso asociado	CU_16: Eliminar actividad
Entrada	-
Resultado esperado	Eliminación de la actividad y los registros de los jugadores correspondientes.
Resultado obtenido	No se elimina la actividad ni los registros de los jugadores. Se muestran mensajes de error.
Estado	Fallida.
Comentarios	<p>La interacción entre las interfaces y la lógica del código relacionada con esta funcionalidad funciona correctamente.</p> <p>El error en esta prueba se debe a que en la API no están definidos los métodos DELETE en /activity y /user_activity. Este problema escapa al alcance del presente Trabajo de Fin de Grado.</p>

Tabla 5.31: Caso de prueba 31

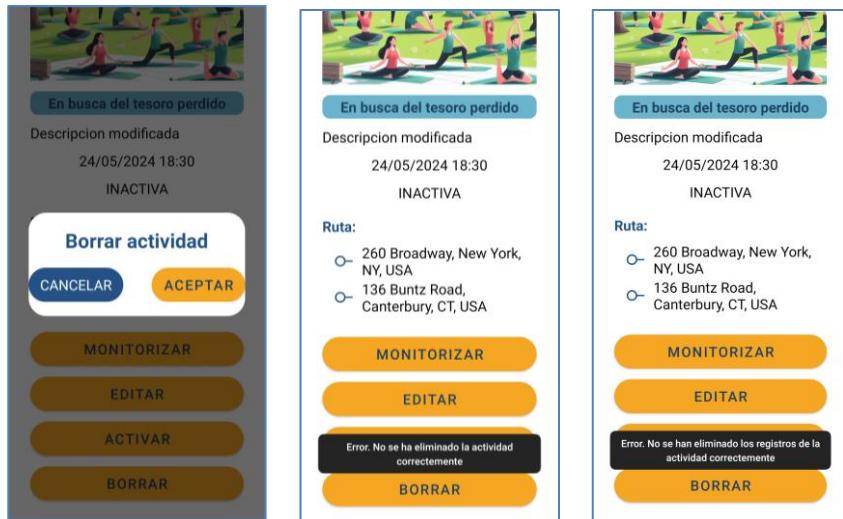


Figura 5.61, 5.62 y 5.63: Imágenes de caso de prueba 31

Como resultado de estas pruebas del sistema, se detectaron y corrigieron los siguientes errores de código:

- Falta de traducción de algunos mensajes al usuario mostrados mediante *Toasts*. Se incluye el texto del mensaje en los ficheros de traducción de ambos idiomas disponibles y se trabaja con la referencia en el código.
- Falta de traducción en el estado de la vista de las actividades de la vista de actividades creadas por el administrador. Se incluye el texto del mensaje en los ficheros de traducción de ambos idiomas disponibles y se trabaja con la referencia en el código.
- Error al intentar añadir un usuario como participante en la creación o modificación de una actividad sin seleccionar a ningún usuario. Se añade código para gestionar de forma adecuada este caso sin provocar errores.

Por último, se han detectado varios errores relacionados con los métodos de la API. Dado que el desarrollo de la API y de la base de datos queda fuera del alcance de este proyecto, y considerando la falta de tiempo derivada del incremento del alcance por funcionalidades que se asumían finalizadas, no ha sido posible lograr que todo el sistema funcione de forma completamente operativa. No obstante, la lógica de la aplicación está implementada y preparada para ejecutar dichas funcionalidades, quedando únicamente pendiente el correcto funcionamiento de las llamadas a la API y los posibles ajustes necesarios para su integración definitiva.

Estas pruebas han permitido comprobar no solo la funcionalidad básica de las tareas que cubre la aplicación, sino también aspectos clave relacionados con la experiencia de usuario. Se ha verificado el correcto flujo de navegación entre las distintas pantallas, así como la disposición adecuada de los elementos en cada vista. Los colores utilizados resultan apropiados y contribuyen a una interfaz clara, accesible y visualmente coherente. Asimismo, se ha comprobado que los textos mostrados son informativos, están correctamente traducidos al español e inglés, y utilizan un tamaño y tipografía legibles. La aplicación proporciona un feedback claro y adecuado para las distintas operaciones realizadas, lo que contribuye a una interacción fluida y satisfactoria por parte del usuario.

6 Resultados y conclusiones

Como resultado del desarrollo de este Trabajo de Fin de Grado, se ha diseñado e implementado una aplicación móvil orientada a la realización de rutas a pie tipo búsqueda del tesoro, con el objetivo de fomentar el ejercicio físico entre personas mayores.

La aplicación implementada cumple con todos los requisitos funcionales definidos inicialmente, a los que se han sumado nuevas funcionalidades durante el proceso, como la modificación de los datos de usuario o la eliminación de actividades. Estas ampliaciones han incrementado el alcance del proyecto, aunque manteniéndose dentro del marco establecido por los objetivos iniciales.

El sistema desarrollado ofrece las siguientes funcionalidades principales:

- **Gestión de usuarios:** Los usuarios pueden registrarse e iniciar sesión. Se contemplan dos roles: jugador y administrador.
- **Rol jugador:** Permite visualizar las actividades en las que está inscrito, iniciar una actividad y leer los puntos que conforman su recorrido. También puede consultar y modificar sus datos personales.
- **Rol administrador:** Ofrece opciones para crear, modificar y eliminar actividades, así como registrar jugadores en ellas. Además, permite monitorizar el progreso de los jugadores en tiempo real.

Para dar soporte a estas funcionalidades, se ha desarrollado la lógica necesaria de la aplicación móvil y se ha establecido la conexión con el servicio web ya existente mediante la librería Retrofit, permitiendo realizar las llamadas correspondientes a la API para almacenar y consultar la información necesaria.

Asimismo, se han llevado a cabo pruebas del sistema con el objetivo de validar el funcionamiento de la aplicación y su diseño, así como determinar el estado del sistema al finalizar el desarrollo.

Respecto a los principales objetivos planteados al inicio del proyecto:

- Implementar una aplicación móvil que permita la recogida de datos y la navegación durante las búsquedas del tesoro.
- Establecer la conexión con el servicio web existente.
- Desarrollar los métodos y servicios necesarios para almacenar en el servidor los datos generados.
- Realizar el despliegue y pruebas del sistema.

Se puede concluir que todos ellos se han cumplido de forma satisfactoria. No obstante, debido a ciertas limitaciones externas al alcance del proyecto —en concreto, problemas detectados en los métodos de la API— no ha sido posible verificar el funcionamiento completo del sistema de manera integrada. Aun así, la lógica de la aplicación está preparada y lista para su correcta integración cuando dichos componentes externos sean corregidos.

A partir de los resultados obtenidos, se proponen **futuras líneas de trabajo** para completar, ampliar y mejorar el sistema desarrollado:

- **Corrección de métodos de la API e integración con el código actual**
Actualmente, la API presenta ciertos errores que impiden el correcto funcionamiento integrado del sistema. Se identifican los siguientes puntos a corregir:

- **Método GET en /activity para el usuario administrador:** muestra las actividades tantas veces como jugadores registrados tenga la actividad. Si no tiene jugadores no aparece y si tiene más de una se muestra duplicada.
- **Métodos PUT en /api_user y /activity:** aunque los cambios se reflejan correctamente en la base de datos, la respuesta del servidor devuelve un código de error 500, impidiendo que la aplicación reconozca la operación como exitosa.
- **Método PUT en /user_activity:** requiere revisión para garantizar su correcto funcionamiento.
- **Eliminación de actividades:** actualmente no existe un método DELETE en /activity y /user_activity. Se propone su implementación o, alternativamente, la incorporación de un atributo en el modelo activity que marque las actividades como eliminadas. Esta segunda opción permitiría mantener un historial sin perder información.

Una vez corregidos estos aspectos, será necesario realizar pruebas y ajustes en el código de la aplicación para asegurar una integración correcta y fluida con la nueva versión del servicio.

- **Pruebas con usuarios reales**

La realización de pruebas con personas mayores, el público objetivo de la aplicación permitiría:

- Validar si el diseño de las interfaces es accesible y fácil de utilizar.
- Detectar posibles mejoras en la experiencia de usuario.
- Identificar nuevas funcionalidades que los propios usuarios consideren útiles.

- **Ampliación de funcionalidades**

Aunque la aplicación cubre los requisitos básicos del sistema, existen múltiples oportunidades para enriquecerla. Algunas propuestas de ampliación incluyen:

- Mostrar información adicional sobre las actividades, como la distancia del recorrido, el tiempo estimado y el tiempo real empleado tras completar la actividad.
- Clasificar las actividades por niveles de dificultad o por categorías temáticas.
- Incluir nuevos modos de actividad, como una búsqueda del tesoro basada en la mecánica de "frío-caliente", utilizando dispositivos Bluetooth beacons. Esta modalidad permitiría su uso tanto en interiores como en exteriores.
- Incorporar elementos de gamificación para hacer la experiencia más atractiva y motivadora. Por ejemplo, asignar puntos al completar actividades o incluir retos breves en los diferentes puntos de una ruta.
- Fomentar la interacción social entre jugadores: permitir que los usuarios puedan agregarse como amigos, visualizar y compartir logros, y participar en actividades en grupo. Esto reforzaría la dimensión social del proyecto, promoviendo la interacción y el sentido de comunidad entre los participantes.

A modo de **conclusión personal**, este proyecto ha representado para mí una experiencia de gran valor académico, profesional y humano.

Me ha permitido consolidar conocimientos adquiridos durante la carrera, como los relacionados con la ingeniería del software, la interacción persona-ordenador (IPO) y el desarrollo de aplicaciones móviles, al aplicarlos de forma práctica en un caso real.

Además, asumir este reto ha sido una oportunidad para investigar de forma autónoma y profundizar en áreas como el desarrollo en Android, la programación en Java y el diseño accesible.

Las tareas desarrolladas a lo largo del proyecto han contribuido al fortalecimiento de competencias fundamentales como la documentación técnica, la capacidad de análisis y la adaptabilidad ante situaciones imprevistas. También he mejorado en aspectos clave como la autonomía de trabajo, la planificación y la gestión eficaz del tiempo, imprescindibles en proyectos de esta envergadura.

Este trabajo me ha hecho reflexionar especialmente sobre la importancia de desarrollar tecnología accesible que contribuya a mejorar la vida de las personas. Me ha resultado muy gratificante poder aplicar mis conocimientos para promover hábitos saludables, como el ejercicio físico, en un colectivo tan importante como el de las personas mayores.

Considero que la usabilidad y la accesibilidad no deberían ser solo valores añadidos, sino criterios fundamentales en el desarrollo de cualquier tecnología orientada a las personas. En un mundo cada vez más digitalizado, y sabiendo que todos seremos mayores algún día, es esencial diseñar soluciones que integren y empoderen a los usuarios de todas las edades. Hacer sentir a nuestros mayores conectados e incluidos es clave para mejorar su bienestar y su calidad de vida.

A nivel personal, me gustaría continuar con el desarrollo de esta aplicación hasta alcanzar una versión completamente funcional e incorporar nuevas funcionalidades que la hagan aún más útil. Creo firmemente que el proyecto tiene una base sólida y un gran potencial de crecimiento. Me motiva especialmente el impacto positivo que puede generar, así como los retos técnicos que plantea, que representan una excelente oportunidad para seguir aprendiendo, especialmente en un área como el desarrollo Android, que me apasiona y en la que deseo seguir profundizando. Continuar trabajando en este sistema significaría, además, seguir explorando maneras de aplicar la tecnología para generar un impacto real y positivo en la sociedad.

7 Análisis de Impacto

Este proyecto ha tenido un impacto muy positivo a nivel **personal**, ya que me ha permitido consolidar conocimientos adquiridos durante la carrera y mejorar habilidades clave como la autonomía, la planificación, la capacidad de análisis o la toma de decisiones. Además, me ha ayudado a desarrollar una mayor sensibilidad hacia la importancia de crear soluciones tecnológicas que realmente respondan a las necesidades de las personas, especialmente de colectivos que a menudo quedan fuera del foco como las personas mayores.

Desde el punto de vista **social**, la aplicación busca fomentar el envejecimiento activo y combatir el aislamiento social, dos de los principales desafíos que enfrentan las personas mayores en la actualidad. A través de propuestas de actividad física accesible y gamificada, se pretende no solo mejorar la salud física de los usuarios, sino también incentivar su participación, autonomía y motivación diaria. La dimensión lúdica y personalizada de las actividades contribuye a que la experiencia sea más atractiva y fácil de integrar en su rutina, reforzando así su bienestar general.

Además, se ha puesto un especial cuidado en aplicar principios de diseño accesible, con el fin de garantizar que la aplicación pueda ser utilizada con facilidad por personas con diferentes capacidades tecnológicas o limitaciones visuales y motoras. Este enfoque tiene como objetivo reducir la brecha digital y hacer la tecnología más inclusiva.

Todo ello se alinea con los Objetivos de Desarrollo Sostenible de la Agenda 2030, en particular con el **ODS 3** [22], centrado en garantizar una vida sana y promover el bienestar para todas las edades, y con el **ODS 10** [23], que busca reducir las desigualdades, tanto sociales como tecnológicas, fomentando una sociedad más inclusiva e igualitaria.

En cuanto al **impacto cultural**, el proyecto promueve una visión más inclusiva y humanista del desarrollo tecnológico, en la que las personas mayores no solo son consideradas usuarias válidas, sino también protagonistas de su propio bienestar. Fomentar su participación en entornos digitales y permitirles sentirse conectados y competentes ayuda a reforzar su autoestima y su calidad de vida, a la vez que contribuye a una sociedad más empática e intergeneracional.

Aunque el **impacto económico** directo de este tipo de proyectos es limitado en su fase inicial, el enfoque adoptado podría llegar a generar valor si se aplicara en colaboración con entidades públicas o del sector salud.

A lo largo del desarrollo del proyecto, se han tomado decisiones conscientes para maximizar su impacto positivo y minimizar cualquier efecto perjudicial. Por ejemplo, se priorizó desde el inicio un enfoque accesible e inclusivo en el diseño de la interfaz, evitando decisiones que pudieran generar barreras tecnológicas para el público objetivo. También se evitó la sobrecarga de funcionalidades o estímulos visuales, teniendo en cuenta las posibles limitaciones cognitivas o visuales de las personas mayores.

Entre los beneficios esperados se encuentran la mejora del bienestar físico y emocional de los usuarios, su inclusión en entornos digitales y el fomento de hábitos saludables de forma autónoma. En cuanto a efectos no deseados, uno de los riesgos podría ser la dependencia tecnológica o la frustración si la aplicación no se adapta correctamente a las necesidades de cada usuario, lo cual refuerza la importancia de realizar pruebas con usuarios reales en fases posteriores. En definitiva, cada decisión ha tratado de responder a una lógica

de impacto positivo, con el objetivo de garantizar que la tecnología no solo sea funcional, sino también ética, útil y verdaderamente humana.

8 Bibliografía

- [1] Organización Mundial de la Salud. (2024, June 26). Actividad física (1st ed.) [Online]. Available: <https://www.who.int/es/news-room/factsheets/detail/physical-activity>
- [2] Infogeriatria. (2024, April 19). *El sedentarismo causa un 6% de las muertes anuales en el mundo* [Online]. Available: <https://www.infogeriatria.com/noticias/20240419/el-sedentarismo-causa-un-6-de-las-muertes-anuales-en-el-mundo>
- [3] Organización Panamericana de la Salud. (2024, June 26). *Cerca de 1800 millones de adultos corren el riesgo de enfermar por falta de actividad física* [Online]. Available: <https://www.paho.org/es/noticias/26-6-2024-cerca-1800-millones-adultos-corren-riesgo-enfermar-por-falta-actividad-fisica>
- [4] Saliha Belmonte Darraz; Ana María González-Roldán; Joaquín de María Arrebola; Casandra Isabel Montoro-Aguilar. (2021, May-June). Impacto del ejercicio fisico en variables relacionadas con el bienestar emocional y funcional en adultos mayores (1st ed.) [Online]. Available: <https://www.elsevier.es/es-revista-revista-espanola-geriatria-gerontologia-124-articulo-impacto-del-ejercicio-fisico-variables-S0211139X21000275>
- [5] American College of Sports Medicine. (2024). *Cognitive Benefits of Physical Activity in Older Adults* [Online]. Available: <https://acsm.org/cognitive-benefits-physical-activity-older-adults>
- [6] D. Giné-Garriga, A. Inzitari, M. L. Guerra-Balic, R. Torres, and M. Jerez-Roig. (2024). *Effects of Physical Exercise Programs on the Health of Older Adults Living in the Community: A Systematic Review of Reviews* [Online]. Available: <https://www.frontiersin.org/journals/public-health/articles/10.3389/fpubh.2024.1385166/full>
- [7] Miguel Gomez-Hernandez; Xavier Ferre1; Cristian Moral; Elena Villalba-Mora. (2023, September 21). Design Guidelines of Mobile Apps for Older Adults: Systematic Review and Thematic Analysis (2nd ed.) [Online]. Available: <https://mhealth.jmir.org/2023/1/e43186>
- [8] Daniel Soler. (2024, April 10). Adaptación del diseño UI/UX para personas mayores (1st ed.) [Online]. Available: https://keepcoding.io/blog/adaptacion-diseno-ui-ux-para-personas-mayores/#Principios_de_diseño_UIUX_para_personas_mayores
- [9] Sam Martín. (2024, September 10). Doble o nada AA: accesibilidad web e inclusión en diseño UI/UX. (1st ed.) [Online]. Available: <https://www.paradigmadigital.com/techbiz/doble-nada-aa-integrando-accesibilidad-web-inclusion-ux-ui/>
- [10] Draw.io [Online]. Available: <https://www.drawio.com/>
- [11] PlantUML [Online]. Available: <https://plantuml.com/es/>
- [12] Figma [Online]. Available: <https://www.figma.com/es-es/design/>
- [13] TPGi [Online]. Available: <https://www.tpgi.com/color-contrast-checker/>

- [14] Android Developers [Online]. Available:
<https://developer.android.com/studio/intro?hl=es-419>
- [15] Square Open Source [Online]. Available:
<https://square.github.io/retrofit/>
- [16] Wiki Open Street Map [Online]. Available:
https://wiki.openstreetmap.org/wiki/Android#Applications_using_OpenStreetMap
- [17] Wiki OSMdroid [Online]. Available:
<https://wiki.openstreetmap.org/wiki/Osmdroid>
- [18] Neil Boyd (2024, November 20) GitHub osmdroid [Online]. Available:
<https://github.com/osmdroid/osmdroid>
- [19] Wiki Openrouteservice [Online]. Available:
https://wiki.openstreetmap.org/wiki/Openrouteservice#Openrouteservice_API
- [20] Andrzej K. Oleś (2025, April 15) GitHub openrouteservice [Online]. Available: <https://github.com/GIScience/openrouteservice?tab=readme-ov-file>
- [21] Postman Docs [Online]. Available:
<https://learning.postman.com/docs/introduction/overview/>
- [22] Pacto Mundial Red España. (n.d.). *ODS 3: Salud y bienestar* [Online]. Available: <https://www.pactomundial.org/ods/3-salud-y-bienestar/>
- [23] Pacto Mundial Red España. (n.d.). *ODS 10: Reducción de las desigualdades* [Online]. Available: <https://www.pactomundial.org/ods/10-reduccion-de-las-desigualdades/>

9 Anexo 1: Informe de Turnitin

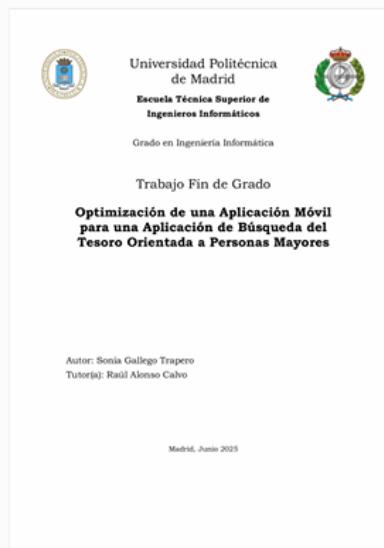


Recibo digital

Este recibo confirma que su trabajo ha sido recibido por Turnitin. A continuación podrá ver la información del recibo con respecto a su entrega.

La primera página de tus entregas se muestra abajo.

Autor de la entrega: SONIA GALLEGOS TRAPERO
Título del ejercicio: Turnitin Memoria Final
Título de la entrega: Memoria_final_TFG_Sonia_Gallego_Trapero.pdf
Nombre del archivo: 10344 SONIA_GALLEGOS TRAPERO_Memoria_final_TFG_Sonia_...
Tamaño del archivo: 6.37M
Total páginas: 91
Total de palabras: 19,357
Total de caracteres: 106,955
Fecha de entrega: 04-jun.-2025 11:25a. m. (UTC+0200)
Identificador de la entrega: 2691923599



Derechos de autor 2025 Turnitin. Todos los derechos reservados.

SONIA GALLEGOS TRAPERO

Memoria_final_TFG_Sonia_Gallego_Trapero.pdf

-  Turnitin Memoria Final
-  TFG ETSIINF (Moodle PP)
-  Universidad Politecnica de Madrid

Detalles del documento

Identificador de la entrega
trn:oid::1:3268376175

91 Páginas

Fecha de entrega
4 jun 2025, 11:25 a.m. GMT+2

19.357 Palabras

Fecha de descarga
4 jun 2025, 12:43 p.m. GMT+2

106.955 Caracteres

Nombre de archivo
10344 SONIA_GALLEGOS_TRAPERO_Memoria_final_TFG_Sonia_Gallego_Trapero_83714_1289672336.pdf

Tamaño de archivo
6.4 MB

4% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- ▶ Bibliography
 - ▶ Quoted Text
-

Top Sources

0%	 Internet sources
0%	 Publications
4%	 Submitted works (Student Papers)
