

GRADO EN INGENIERÍA MULTIMEDIA



VNIVERSITAT
DE VALÈNCIA

TRABAJO FIN DE GRADO

DONA'M MÓN: APLICACIÓN MÓVIL PARA
VISIBILIZAR LA VIDA Y OBRA DE MUJERES EN
PUNTOS GEOLOCALIZADOS DE VALENCIA.

AUTOR:
IVANA STANISLAVOVA IVANOVA

TUTOR:
MANUEL PÉREZ AIXENDRI



VNIVERSITAT
DE VALÈNCIA [○*] Escola Tècnica Superior
d'Enginyeria ETSE-UV

TRABAJO FIN DE GRADO

DONA'M MÓN: APLICACIÓN MÓVIL PARA
VISIBILIZAR LA VIDA Y OBRA DE MUJERES EN
PUNTOS GEOLOCALIZADOS DE VALENCIA.

AUTOR: IVANA STANISLOVA IVANOVA

TUTOR: MANUEL PÉREZ AIXENDRI

Declaración de autoría:

Yo, Ivana Stanislavova Ivanova, declaro la autoría del Trabajo Fin de Grado titulado “DONA’m MÓN: Aplicación móvil para visibilizar la vida y obra de mujeres en puntos geolocalizados de Valencia.” y que el citado trabajo no infringe las leyes en vigor sobre propiedad intelectual. El material no original que figura en este trabajo ha sido atribuido a sus legítimos autores.

Valencia, 26 de junio de 2025

Fdo: Ivana Stanislavova Ivanova

Resumen:

Este es el resumen del TFM. Debe ser corto (máximo media página) y cubrir los aspectos principales del TFM.

Abstract:

This is the abstract of the TFM. It must be short and cover the main aspects of the TFM.

Resum:

Aquest és el resum del TFM. Ha de ser curt (màxim mitja pàgina) i cobrir els aspectes principals del TFM.

Agradecimientos:

Índice general

1. Introducción	21
1.1. Introducción	21
1.2. Motivación	21
1.3. Objetivos	22
1.3.1. Alineación con los Objetivos de Desarrollo Sostenible (ODS)	23
1.4. Organización de la memoria	24
2. Estado del arte	27
2.1. Historia, evolución e impacto de las aplicaciones móviles	27
2.1.1. Origen y evolución de los dispositivos móviles	27
2.1.2. Ecosistemas y sistemas operativos móviles	28
2.1.3. Tipos de aplicaciones móviles	28
2.1.4. Impacto económico y social	29
2.2. Evolución de la Realidad Aumentada en el Contexto de las Aplicaciones Móviles	30
2.2.1. Fundamentos y primeras ideas (1960–1990)	30
2.2.2. Definición formal y primeros prototipos (1990–2000)	31
2.2.3. Herramientas abiertas y salto a lo portátil (1999–2010)	31
2.2.4. Consolidación y tendencias actuales (2010–2025)	32
2.3. La visibilización de las mujeres en el espacio urbano: una cuestión de justicia histórica	33
2.3.1. Referentes femeninos en la ciudad: biografías y puntos de interés . .	34
2.4. Análisis de aplicaciones similares	36
2.4.1. Pokémon GO	36
2.4.2. Streetmuseum	36
2.4.3. CultuAR	37
2.4.4. Historypin	37
2.5. Tecnologías	38
2.5.1. Plataformas de desarrollo de aplicaciones móviles	38
2.5.2. Geolocalización	40

2.5.3. Realidad Aumentada (RA)	41
2.5.4. Base de Datos	43
2.5.5. Arquitectura de persistencia y gestión de datos	43
3. Requisitos, especificaciones, coste, riesgos, viabilidad	47
3.1. Requisitos	47
3.1.1. Requisitos funcionales	47
3.1.2. Requisitos funcionales	47
3.1.3. Requisitos no funcionales	49
3.2. Especificaciones	51
3.3. Ciclo de vida del software	53
3.4. Planificación temporal	54
3.4.1. Descomposición de tareas por fases	54
3.4.2. Estimación temporal	57
3.4.3. Diagrama de Gantt	58
3.5. Estimación de costes	60
3.5.1. Costes directos de personal	60
3.5.2. Asignación temporal por perfil	60
3.5.3. Costes directos de material	61
3.5.4. Costes indirectos	61
3.5.5. Coste total del proyecto	62
3.6. Viabilidad del proyecto	62
3.6.1. Viabilidad económica	62
3.6.2. Viabilidad legal	63
3.7. Análisis de riesgos	63
3.7.1. Principales riesgos identificados	63
3.7.2. Matriz de riesgos	65
3.7.3. Resumen de evaluación de riesgos	65
3.7.4. Estrategias de mitigación	65
3.7.5. Planes de contingencia	66
4. Análisis	67
4.1. Diagramas de casos de uso	67
4.2. Especificación de casos de uso	70
4.2.1. Caso de uso 1: Login de usuario	70
4.2.2. Caso de uso 2: Registro de usuario	71
4.2.3. Caso de uso 3: Ver mapa con puntos de interés	72

4.2.4. Caso de uso 4: Ver listado de puntos ordenado por cercanía	73
4.2.5. Caso de uso 5: Ver ficha del lugar	74
4.2.6. Caso de uso 6: Ver rutas y consultar puntos de una ruta	75
4.2.7. Caso de uso 7: Escanear QR de punto de ruta	76
4.2.8. Caso de uso 8: Ver puntos RA disponibles	77
4.2.9. Caso de uso 9: Visualizar contenido RA	78
4.2.10. Caso de uso 10: Editar perfil de usuario	79
4.2.11. Caso de uso 11: Ver historial de lugares visitados y rutas completadas	80
4.2.12. Caso de uso 12: Cerrar sesión	81
4.2.13. Caso de uso 13: Iniciar sesión como administrador	82
4.2.14. Caso de uso 14: Gestionar mujeres históricas	83
4.2.15. Caso de uso 15: Gestionar puntos geolocalizados	84
4.2.16. Caso de uso 16: Gestionar rutas temáticas	85
4.2.17. Caso de uso 17: Consultar lista de usuarios	86
4.2.18. Caso de uso 18: Gestionar materiales multimedia	87
4.2.19. Caso de uso 19: Consultar métricas de uso	88
4.2.20. Caso de uso 20: Gestionar áreas de investigación	89
4.3. Diagramas de Actividad	90
4.4. Diagramas de Estado	94
5. Diseño	97
5.1. Diseño de base de datos	97
5.1.1. Elección de la base de datos y justificación	97
5.1.2. Tablas principales y justificación de diseño	97
5.1.3. Relaciones entre tablas	99
5.1.4. Esquema de tablas	99
5.1.5. Justificación global del diseño	99
5.2. Diseño de la arquitectura del frontend	101
5.2.1. Diagrama de clases del frontend	101
5.3. Diagramas de secuencia	103
5.3.1. Diagrama de secuencia del caso de uso 1: Login de usuario	103
5.3.2. Diagrama de secuencia del caso de uso 2: Registro de usuario	104
5.3.3. Diagrama de secuencia del caso de uso 3: Ver mapa con puntos de interés	104
5.3.4. Diagrama de secuencia del caso de uso 4: Ver listado de puntos ordenado por cercanía	105
5.3.5. Diagrama de secuencia del caso de uso 5: Ver ficha del lugar	105

5.3.6. Diagrama de secuencia del caso de uso 6: Ver rutas y consultar puntos de una ruta	106
5.3.7. Diagrama de secuencia del caso de uso 7: Escanear QR de punto de ruta	106
5.3.8. Diagrama de secuencia del caso de uso 8: Ver puntos RA disponibles	107
5.3.9. Diagrama de secuencia del caso de uso 9: Visualizar contenido RA .	107
5.3.10. Diagrama de secuencia del caso de uso 10: Editar perfil de usuario .	108
5.3.11. Diagrama de secuencia del caso de uso 11: Ver historial de lugares visitados y rutas completadas	108
5.3.12. Diagrama de secuencia del caso de uso 12: Cerrar sesión	109
5.3.13. Diagrama de secuencia del caso de uso 13: Iniciar sesión como administrador	109
5.3.14. Diagrama de secuencia del caso de uso 14: Gestionar mujeres históricas	110
5.3.15. Diagrama de secuencia del caso de uso 15: Gestionar puntos geocalizados	110
5.3.16. Diagrama de secuencia del caso de uso 16: Gestionar rutas temáticas	111
5.3.17. Diagrama de secuencia del caso de uso 17: Consultar lista de usuarios	111
5.3.18. Diagrama de secuencia del caso de uso 18: Gestionar materiales multimedia	112
5.3.19. Diagrama de secuencia del caso de uso 19: Consultar métricas de uso	112
5.3.20. Diagrama de secuencia del caso de uso 20: Gestionar áreas de investigación	113
6. Implementación y pruebas	115
6.1. Implementación del backend	115
6.1.1. Herramientas, frameworks y lenguajes utilizados	115
6.1.2. Estructura del proyecto y organización del código	115
6.1.3. Modelado de datos	116
6.1.4. Gestión de imágenes	118
6.1.5. Paginación de la API	118
6.1.6. Seguridad y permisos	119
6.1.7. Gestión de variables de entorno y configuración segura	119
6.1.8. Serialización y exposición de la API REST	119
6.1.9. Vistas, lógica de endpoints y ejemplos de lógica relevante	121
6.1.10. Administración y gestión de datos	122
6.1.11. Cambio de contraseña del usuario administrador de Django	123
6.1.12. Gestión de migraciones y consistencia de la base de datos	123
6.1.13. Integración con el frontend y gestión de CORS	124
6.1.14. Integración y despliegue con Docker	125

6.1.15. Pruebas automáticas	125
6.1.16. Gestión de emails y notificaciones	126
6.1.17. Internacionalización y localización	127
6.1.18. Gestión de archivos estáticos	127
6.1.19. Gestión de geolocalización y decisión sobre campos GIS	127
6.2. Implementación del frontend	128
6.2.1. Herramientas, frameworks y lenguajes utilizados	128
6.2.2. Estructura del proyecto y organización del código	128
6.2.3. Navegación y estructura de pantallas	129
6.2.4. Gestión de usuarios y autenticación	130
6.2.5. Visualización de mujeres y lugares	131
6.2.6. Detalle de lugar y gestión de visitas	131
6.2.7. Mapas y geolocalización	133
6.2.8. Gestión de rutas temáticas	133
6.2.9. Historial de lugares y rutas visitadas	134
6.2.10. Notificaciones y proximidad	134
6.2.11. Realidad aumentada	134
6.2.12. Gestión de estilos y diseño	135
6.2.13. Gestión de almacenamiento local	135
6.2.14. Integración con la API y manejo de errores	135
6.2.15. Pruebas y depuración	136
7. Conclusiones	137
7.1. Revisión de costes	137
7.2. Conclusiones	137
7.3. Trabajo futuro	137
A. Apéndice	139
A.1. Ejemplos del lenguaje de marcado Latex	139
Bibliografía	140

Capítulo 1

Introducción

1.1. Introducción

En la actualidad, la tecnología se ha convertido en una herramienta clave para transformar la forma en la que interactuamos con nuestro entorno, permitiendo nuevas maneras de aprender, explorar y conectarnos con la historia y la cultura. Sin embargo, la construcción de los relatos históricos y culturales ha tendido a otorgar mayor protagonismo a figuras masculinas, dejando en la sombra numerosas contribuciones femeninas de gran relevancia en ámbitos como la ciencia, el arte, la política o la educación.

Este Trabajo de Fin de Grado se centra en el desarrollo de una aplicación móvil interactiva basada en geolocalización y realidad aumentada (RA), que tiene como objetivo visibilizar el legado de mujeres con relevancia histórica que han nacido, vivido, desarrollado parte de su labor o tenido relevancia en Valencia. Mediante esta aplicación, los usuarios podrán desbloquear contenido educativo y multimedia al visitar puntos de interés geolocalizados, conectando los lugares emblemáticos de la ciudad con las historias de estas mujeres, conociendo su vida y obra.

La aplicación lleva por nombre DONA'm MÓN, una frase en valenciano con doble significado en su primera palabra: “dona” puede leerse tanto como sustantivo (“mujer”) o como forma verbal (“dame”). Así que el título leído como “Dame mundo”, esconde un juego de palabras que representa el propósito del proyecto: ofrecer a las mujeres este espacio real y simbólico en el mundo, un lugar en el mapa donde su historia pueda ser descubierta y reconocida.

DONA'm MÓN ofrece una experiencia educativa y lúdica que invita a reflexionar sobre la igualdad de género. Este proyecto pretende ser un puente entre el pasado y el presente, ayudando a los usuarios a redescubrir la ciudad desde esta nueva perspectiva inclusiva.

1.2. Motivación

El desarrollo de este proyecto surge de una doble motivación: por un lado, mi interés personal en el desarrollo de tecnologías con un propósito educativo y social, y por otro, el deseo de poner en valor las aportaciones de las mujeres a lo largo de la historia.

Durante mi recorrido académico como estudiante de Ingeniería Multimedia y en mi máster actual en Tecnologías web, Aplicaciones móviles y Computación en la nube, he desarrollado un fuerte interés por aplicar el conocimiento técnico en proyectos que tengan

un impacto positivo en la sociedad, orientados a transformar la manera en la que accedemos y transmitimos el conocimiento.

Además, como mujer en STEM, considero importante que este proyecto consiga ayudar en la visibilización del papel femenino en todas las áreas del saber y a toda clase de público. Aunque se han producido avances significativos en materia de igualdad, los datos actuales evidencian que la brecha persiste: solo el 33 % de los investigadores a nivel mundial son mujeres, y en campos como la inteligencia artificial esta cifra desciende al 22 % [1]. En el contexto español, apenas el 24 % de las cátedras universitarias están ocupadas por mujeres, según el informe “Científicas en Cifras 2021” del Ministerio de Ciencia e Innovación [2]. Estas cifras reflejan una infrarrepresentación que no solo afecta a la ciencia, sino también a otros ámbitos clave del desarrollo cultural y social.

Valencia, como muchas otras ciudades, alberga una historia rica y compleja en la que también han participado numerosas mujeres valiosas, desde científicas y artistas hasta activistas y educadoras. Muchas de ellas siguen siendo poco conocidas fuera de círculos especializados. Este proyecto representa una oportunidad para darles mayor protagonismo, aprovechando tecnologías como la realidad aumentada y la geolocalización para generar experiencias de aprendizaje activas, accesibles e innovadoras.

Por todo esto, el objetivo de este trabajo no es solo técnico, sino también ético y cultural: contribuir, desde la tecnología, a construir una memoria más justa e inclusiva.

1.3. Objetivos

El objetivo principal de este proyecto es desarrollar una aplicación móvil interactiva que utilice tecnologías de geolocalización, realidad aumentada (RA) y una base de datos centralizada para visibilizar y divulgar las historias de mujeres destacadas en la ciudad de Valencia, promoviendo su reconocimiento histórico y cultural a través de una experiencia inmersiva, educativa y accesible para los usuarios.

Descomponiendo este objetivo principal, se definen los siguientes objetivos específicos:

■ **Investigación y selección de mujeres invisibilizadas:**

- Identificar figuras femeninas históricas relacionadas con Valencia que hayan destacado en campos como la ciencia, el arte, la política o los derechos sociales.
- Asociar a cada figura un punto geolocalizado en la ciudad con un vínculo significativo con su historia.

■ **Creación y gestión de una base de datos con MySQL:**

- Diseñar y desarrollar una base de datos relacional en MySQL, estructurada para almacenar de forma eficiente información sobre las mujeres seleccionadas, sus ubicaciones geográficas y el contenido multimedia vinculado.
- Implementar un backend utilizando el framework Django, asegurando la integridad, consistencia y seguridad de los datos mediante su ORM, y exponiendo una API RESTful que permita su acceso y manipulación desde la aplicación móvil.

■ **Desarrollo de funcionalidades tecnológicas:**

- Implementar un sistema de geolocalización que permita a los usuarios desbloquear contenido interactivo al acercarse a los puntos de interés.
 - Integrar elementos de realidad aumentada desarrollados con A-Frame, que permitan visualizar objetos, imágenes o recreaciones 3D relacionadas con las mujeres seleccionadas al enfocar con la cámara del dispositivo móvil.
 - Implementar un sistema de notificaciones que alerte a los usuarios al aproximarse a un punto de interés relevante, mejorando la interacción con el entorno urbano y fomentando la exploración activa.
- **Diseño de una experiencia de usuario intuitiva y atractiva:**
- Crear una interfaz accesible y adaptada al entorno móvil con React Native y Expo Go, que facilite la navegación y el uso de las funcionalidades principales de la aplicación.
 - Incluir un mapa interactivo con los puntos de interés marcados, accesibles según la ubicación del usuario.
- **Generación de contenido multimedia:**
- Diseñar materiales interactivos y educativos como textos, audios, imágenes, videos y modelos 3D que permitan a los usuarios conocer la vida y las contribuciones de las mujeres seleccionadas.
 - Desarrollar un sistema de recompensas o logros para incentivar la exploración de todos los puntos geolocalizados.
- **Compatibilidad multiplataforma y pruebas de usabilidad:**
- Garantizar que la aplicación sea funcional en dispositivos Android e iOS mediante el uso de herramientas multiplataforma como React Native y Expo Go.
 - Realizar pruebas de usabilidad con un grupo de usuarios para evaluar la experiencia de uso y verificar que se cumplen los objetivos educativos y tecnológicos planteados.

1.3.1. Alineación con los Objetivos de Desarrollo Sostenible (ODS)

DONA'm MÓN se relaciona de forma directa con algunos de los Objetivos de Desarrollo Sostenible propuestos por la ONU dentro de la Agenda 2030, especialmente aquellos centrados en la igualdad, la educación, la sostenibilidad urbana y el bienestar. A través de una experiencia tecnológica que combina cultura, memoria y participación ciudadana, el proyecto contribuye a acercar estos objetivos al contexto local. [3, 4, 5]

- **ODS 5: Igualdad de género.** Este objetivo busca eliminar todas las formas de discriminación y violencia contra mujeres y niñas, y promover su plena participación en todos los ámbitos. DONA'm MÓN se centra precisamente en recuperar y visibilizar figuras femeninas que han sido olvidadas o ignoradas. Reivindicar sus historias es una forma de justicia simbólica y de construir referentes para las nuevas generaciones.

- **ODS 4: Educación de calidad.** Este ODS promueve el acceso a una educación inclusiva, equitativa y de calidad. La aplicación funciona como una herramienta educativa alternativa que permite aprender fuera del aula, en el espacio urbano, utilizando recursos multimedia y tecnologías actuales. Es una forma distinta de acercarse a la historia local desde una perspectiva crítica y feminista.
- **ODS 11: Ciudades y comunidades sostenibles.** Uno de los enfoques de este objetivo es garantizar que las ciudades sean más inclusivas y accesibles. DONA'm MÓN propone una forma de recorrer Valencia nueva, resignificando lugares que muchas veces pasan desapercibidos. También invita a caminar y descubrir el entorno, fomentando un uso más consciente del espacio público.
- **ODS 3: Salud y bienestar.** Aunque no es el objetivo principal del proyecto, se puede decir que la aplicación promueve indirectamente hábitos saludables, ya que para desbloquear los contenidos de las rutas es necesario moverse físicamente por la ciudad. Esto puede motivar a las personas usuarias a caminar, explorar y mantenerse activas mientras aprenden.

1.4. Organización de la memoria

Esta memoria se estructura en siete capítulos principales, además de un apéndice y una bibliografía final. A continuación se describen brevemente los contenidos de cada uno de ellos:

Capítulo 1 – Introducción: contextualiza el trabajo realizado, presentando la motivación personal y social del proyecto, así como los objetivos generales y específicos que se pretenden alcanzar. También incluye esta sección de organización de la memoria para orientar al lector sobre la estructura del documento.

Capítulo 2 – Estado del arte: recoge un análisis comparativo de aplicaciones similares ya existentes, con el objetivo de identificar puntos fuertes y carencias que puedan aprovecharse como referencia para el desarrollo del proyecto. Además, se detallan las tecnologías empleadas, justificando su elección en función de las necesidades de la aplicación.

Capítulo 3 – Requisitos, especificaciones, costes, riesgos y viabilidad: define los requisitos funcionales y no funcionales del sistema, junto con sus especificaciones técnicas. También se lleva a cabo una estimación de los costes asociados, se identifican posibles riesgos y se valora la viabilidad del proyecto desde una perspectiva técnica y económica.

Capítulo 4 – Análisis: describe el análisis de la solución propuesta, incluyendo aspectos como la estructura general de la aplicación, la lógica funcional y la interacción entre sus componentes principales.

Capítulo 5 – Diseño: presenta el diseño de la aplicación a diferentes niveles (arquitectura, interfaz de usuario, estructura de datos, etc.), sirviendo como puente entre el análisis conceptual y la posterior implementación técnica.

Capítulo 6 – Implementación y pruebas: documenta el proceso de desarrollo, indicando cómo se ha llevado a cabo la implementación de la solución y qué herramientas se han utilizado. A su vez, se incluyen las pruebas realizadas para verificar el correcto funcionamiento del sistema, abarcando pruebas funcionales, de rendimiento y de usabilidad.

Capítulo 7 – Conclusiones: ofrece una reflexión final sobre los resultados obtenidos, revisa los costes reales frente a los estimados, valora el cumplimiento de los objetivos y propone posibles mejoras o líneas de trabajo futuro.

Bibliografía: recoge todas las fuentes utilizadas para la elaboración del trabajo, tanto en su fase de investigación como en el desarrollo e implementación del proyecto.

Esta organización busca facilitar la lectura y comprensión del documento, guiando al lector a través del proceso completo de conceptualización, desarrollo y evaluación de la aplicación DONA'm MÓN.

Capítulo 2

Estado del arte

2.1. Historia, evolución e impacto de las aplicaciones móviles

2.1.1. Origen y evolución de los dispositivos móviles

El desarrollo de las aplicaciones móviles no puede entenderse sin revisar primero la evolución histórica de los dispositivos que las alojan. La telefonía móvil nace en 1946 con el sistema móvil de AT&T, que permitía realizar llamadas desde vehículos utilizando dispositivos de más de 30 kg. Este sistema inicial, basado en la tecnología Push-To-Talk y gestionado por operadoras humanas, tenía una cobertura muy limitada y altos costes de uso [6].

Durante las décadas siguientes se sucedieron avances importantes: la introducción del sistema RCC en 1960, el Improved Mobile Telephone Service de AT&T en 1965, y finalmente, la llegada del primer teléfono móvil portátil, el Motorola DynaTAC, en 1983. Sin embargo, no fue hasta la década de 1990 cuando los dispositivos comenzaron a incorporar funcionalidades propias de asistentes personales, como los Apple Newton o los Palm Pilot, marcando así el inicio de las primeras “aplicaciones” de uso cotidiano, como agendas, calendarios y juegos como Snake de Nokia [7].

El cambio de paradigma llegó en 2007 con la aparición del iPhone, que introdujo una interfaz multitáctil intuitiva y eliminó el teclado físico, inaugurando así la era del smartphone moderno. Un año más tarde, Apple lanzó la App Store, permitiendo a desarrolladores externos distribuir aplicaciones fácilmente. Ese mismo año, Google presentó Android junto con el HTC Dream, diversificando el ecosistema y facilitando el acceso masivo al desarrollo y consumo de aplicaciones móviles [8].



Figura 2.1: Evolución histórica de los dispositivos móviles.

2.1.2. Ecosistemas y sistemas operativos móviles

El auge de las aplicaciones móviles está estrechamente vinculado al desarrollo de los sistemas operativos (SO) móviles. Durante los primeros años del siglo XXI, Symbian OS dominaba el mercado, especialmente en terminales de Nokia. Sin embargo, fue rápidamente desplazado por iOS (2007) y Android (2008), que ofrecían interfaces gráficas avanzadas, tiendas de aplicaciones centralizadas y acceso optimizado a hardware [9].

Actualmente, Android posee una cuota de mercado superior al 70 % a nivel global, mientras que iOS concentra cerca del 28 % del mercado, especialmente en países desarrollados [10]. Otros sistemas operativos como Windows Phone, BlackBerry OS o Tizen han desaparecido o mantienen una presencia marginal. La consolidación de este duopolio ha propiciado estándares de desarrollo bien definidos, facilitando la expansión del mercado de aplicaciones.

Cabe destacar que la fragmentación del ecosistema Android sigue representando un reto técnico. Según un estudio de Android Police [11], el tiempo medio de soporte de actualizaciones es de 21 meses en dispositivos Android, frente a los 41 meses de Apple, lo que obliga a los desarrolladores a implementar múltiples versiones y adaptaciones para garantizar la compatibilidad.

2.1.3. Tipos de aplicaciones móviles

Las aplicaciones móviles pueden clasificarse en función de su arquitectura en cinco categorías principales:

- **Aplicaciones web:** ejecutadas dentro del navegador, sin necesidad de instalación. Utilizan tecnologías como HTML5, CSS y JavaScript. Son portables pero con acceso limitado al hardware [?].

- **Aplicaciones web progresivas (PWA)**: ofrecen una experiencia cercana a las nativas, pueden instalarse desde el navegador y ejecutarse sin conexión, gracias a tecnologías como los service workers.
- **Aplicaciones híbridas**: encapsulan una web app dentro de una aplicación nativa mediante motores como Cordova o Ionic. Permiten cierto acceso a funciones del dispositivo, pero con rendimiento inferior a las apps nativas.
- **Aplicaciones nativas**: desarrolladas específicamente para cada plataforma, usando lenguajes como Swift (iOS), Kotlin o Java (Android). Ofrecen el mejor rendimiento, pero requieren múltiples desarrollos independientes.
- **Aplicaciones nativas multiplataforma**: permiten usar un único lenguaje y base de código para generar versiones para iOS y Android. Ejemplos incluyen React Native, Flutter y Xamarin, ampliamente utilizados en el desarrollo moderno [12].

	App Nativa	App Híbrida	App Web (Progresiva)	App Web
Experiencia de usuario	EXCELENTE	EXCELENTE	BUENA	NORMAL
Velocidad y Rendimiento	MUY RÁPIDA	MUY RÁPIDA	BUENA	NORMAL
Seguridad	ALTA	ALTA	BUENA	BUENA
App Stores	SÍ	SÍ	NO	NO
Función Offline	SÍ	SÍ	NO	NO
Tiempo de Desarrollo	ALTO	MEDIO	BAJO	BAJO
Mantenimiento	ALTO	MEDIO	BAJO	BAJO
Coste de Desarrollo	ALTO	MEDIO	MEDIO	MEDIO

Cuadro 2.1: Comparación entre tipos de aplicaciones móviles

Cada enfoque presenta ventajas y limitaciones en términos de rendimiento, coste, mantenimiento, compatibilidad y acceso a recursos del sistema.

2.1.4. Impacto económico y social

El mercado de aplicaciones móviles ha alcanzado dimensiones económicas y sociales extraordinarias. En 2023 se descargaron más de 257 mil millones de aplicaciones a nivel

mundial, con un gasto dentro de estas y suscripciones que superó los 170 mil millones de dólares [13]. El tiempo medio diario de uso del móvil ha aumentado a 5 horas, consolidando a las aplicaciones como principales mediadoras de nuestra relación con el entorno digital [14].

Además, su impacto trasciende lo económico. Las aplicaciones han revolucionado sectores clave como la salud (mHealth), la educación (mLearning), el transporte, el comercio electrónico, la cultura y la participación ciudadana. Por ejemplo, en el ámbito de la salud, aplicaciones como MySugr o Glucose Buddy permiten a pacientes con diabetes gestionar su enfermedad mediante el seguimiento de glucosa, dieta y ejercicio. En educación, plataformas como Duolingo o Khan Academy han universalizado el acceso al aprendizaje, permitiendo a millones de personas adquirir nuevas habilidades desde cualquier lugar [15].

En cuanto a la accesibilidad, los dispositivos móviles permiten una conexión constante con contenidos digitales, lo que favorece procesos de inclusión digital en poblaciones diversas. Esta capacidad de mediación sociotécnica hace que las aplicaciones móviles sean herramientas importantes para la innovación social [15].

2.2. Evolución de la Realidad Aumentada en el Contexto de las Aplicaciones Móviles

La Realidad Aumentada (RA) es una tecnología que permite combinar contenido digital generado por ordenador con la percepción del mundo físico, proporcionando una experiencia enriquecida y contextual. Aunque su auge reciente está vinculado al ecosistema móvil, su desarrollo abarca más de medio siglo, con avances teóricos, técnicos y aplicados que han configurado su estado actual como tecnología madura.

2.2.1. Fundamentos y primeras ideas (1960–1990)

Los antecedentes de la RA preceden a la era digital. En 1962, Morton Heilig presentó el Sensorama, un sistema inmersivo que combinaba estímulos visuales, sonoros y olfativos para simular experiencias sensoriales integradas, aunque sin capacidades de interacción. A nivel tecnológico, el verdadero precursor de la RA fue el Head-Mounted Display (HMD) diseñado por Ivan Sutherland en 1968, apodado "Sword of Damocles". Este sistema óptico-mecánico, con seis grados de libertad, permitió visualizar objetos generados por ordenador superpuestos al entorno real [16].



Figura 2.2: Primer sistema RA: "Sword of Damocles" (1968)

No obstante, el concepto formal de RA se desarrolló más tarde. En 1992, Caudell y Mizell acuñaron el término Augmented Reality en un contexto industrial, para referirse al uso de displays que asistían a operarios durante tareas complejas. Este enfoque pragmático orientó la investigación hacia sistemas útiles para tareas del mundo real, diferenciándose de la realidad virtual (RV), que propone la inmersión total en entornos digitales.

2.2.2. Definición formal y primeros prototipos (1990–2000)

Ronald Azuma estableció en 1997 la definición académica más citada de RA, basada en tres criterios fundamentales: (1) combinación de elementos reales y virtuales, (2) interacción en tiempo real y (3) registro espacial tridimensional preciso [16]. En paralelo, Paul Milgram y Fumio Kishino introdujeron el Continuo Realidad-Virtualidad, que posiciona la RA entre el entorno físico puro y la RV completa, permitiendo caracterizar distintos grados de mezcla entre ambos mundos [17].



Figura 2.3: Sistema KARMA para asistencia con RA (1993)

Durante esta década se desarrollaron los primeros sistemas funcionales de RA. En 1993, el sistema KARMA, creado por Steve Feiner y su equipo, permitía la superposición de información contextual durante tareas físicas [18]. En 1997, se presentó el prototipo MARS, un sistema de RA móvil diseñado para guiar a turistas mediante visualización aumentada en tiempo real. Estas iniciativas fueron posibles gracias al desarrollo de dispositivos portables y técnicas de tracking visual.

2.2.3. Herramientas abiertas y salto a lo portátil (1999–2010)

Uno de los hitos más relevantes en esta etapa fue el desarrollo de ARToolKit por Hirokazu Kato y Mark Billinghurst en 1999. Esta librería de código abierto permitió detectar marcadores visuales en video y superponer contenido digital 3D en tiempo real [19]. Supuso una democratización de la RA, facilitando la experimentación por parte de investigadores y desarrolladores independientes.

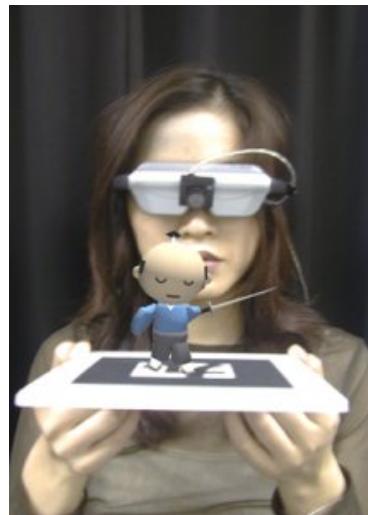


Figura 2.4: ARToolKit y marcadores visuales (1999)

Entre 2001 y 2004, surgieron los primeros sistemas de RA sobre dispositivos móviles, como PDAs y teléfonos inteligentes de primera generación. En 2003, Wagner y Schmalstieg presentaron un sistema de RA completamente autónomo en un dispositivo de mano [20]. Otros experimentos destacados fueron ARQuake (2000), un videojuego de RA en primera persona, y Invisible Train (2004), una aplicación colaborativa con varios usuarios simultáneos.

2.2.4. Consolidación y tendencias actuales (2010–2025)

Durante la última década, la Realidad Aumentada ha pasado de ser una tecnología emergente a consolidarse como una herramienta madura y transversal en múltiples sectores. El desarrollo de kits como ARToolKit, ARCore (Google) y ARKit (Apple) permitió la integración de funcionalidades de RA en dispositivos móviles de forma nativa, facilitando su adopción masiva y el desarrollo de aplicaciones accesibles sin necesidad de hardware especializado [21, 22].

Uno de los catalizadores clave de esta expansión fue el éxito comercial de Pokémon GO (2016), que demostró el potencial de la RA móvil como forma de interacción lúdica y geolocalizada [23]. Paralelamente, surgieron dispositivos avanzados como Microsoft HoloLens y Magic Leap One, que ofrecieron experiencias inmersivas mediante visores ópticos, reconocimiento espacial y control gestual [24, 25].

En el plano técnico, los avances en visión por computador y en algoritmos de localización y mapeo simultáneo (SLAM) mejoraron notablemente la precisión del registro espacial tridimensional [26]. También se consolidaron nuevas formas de interacción como la RA tangible y la RA colaborativa, que permiten manipular objetos físicos para modificar contenidos digitales o interactuar con varios usuarios en tiempo real [27, 28].

Actualmente, la RA se aplica con éxito en campos como la educación, la salud, la industria, la arquitectura o el comercio, y su desarrollo se orienta hacia una fusión más natural entre el mundo físico y el digital mediante técnicas de spatial computing y nuevos dispositivos como el Apple Vision Pro (2024) [29].



Figura 2.5: HoloLens 2 de Microsoft en labores de asistencia remota y formación.

2.3. La visibilización de las mujeres en el espacio urbano: una cuestión de justicia histórica

La representación de las mujeres en el espacio urbano continúa siendo escasa. Según el proyecto europeo “Women’s Legacy” [30], impulsado desde la Conselleria d’Educació, Cultura i Esport de la Generalitat Valenciana, solo un 10 % de las calles con nombres de personas en muchas ciudades españolas están dedicadas a mujeres, y este patrón se repite en monumentos, placas conmemorativas y espacios simbólicos.

A pesar de esta disparidad, en los últimos años han surgido iniciativas locales e institucionales orientadas a revertir esta situación. En Valencia, destaca el proyecto “Dones de Ciència” [31], una colaboración entre la Universitat Politècnica de València (UPV) y Las Naves, que ha creado más de 30 murales urbanos en centros educativos para homenajear a mujeres científicas de todo el mundo. Esta intervención artística no solo embellece el entorno, sino que actúa como una herramienta pedagógica para inspirar a las nuevas generaciones. Asimismo, el Ayuntamiento de València ha promovido campañas de renombramiento de calles con nombres femeninos y la inclusión de mujeres en los itinerarios culturales y turísticos de la ciudad [32].

Por otra parte, diversas asociaciones vecinales y colectivos feministas han impulsado rutas autogestionadas por la ciudad que visibilizan las huellas femeninas en el espacio urbano. Entre ellas destaca la Ruta Violeta de Mujeres en Valencia”, organizada por la Asociación Por Ti Mujer, que recorre espacios emblemáticos ligados a la historia de las mujeres en la ciudad, promoviendo un reconocimiento público de sus contribuciones sociales y culturales [33].

Asimismo, la Assemblea Feminista de València ha organizado la Ruta pels espais del Patronato de Protección a la Mujer”, que recorre espacios vinculados a esta institución franquista, la cual operó durante gran parte del siglo XX bajo la tutela de la Iglesia y el Estado. Su función no era tanto proteger a mujeres víctimas de violencia como controlar y castigar a aquellas que transgredían los roles tradicionales de género. Esta ruta permite evidenciar cómo las estructuras institucionales legitimaron la exclusión, la represión y la estigmatización de mujeres consideradas “desviadas” por la moral conservadora, visibilizando así dinámicas de resistencia y memoria colectiva en la ciudad [34].

Además, la Ruta de Memorias Lesbianas”, también organizada por la Assemblea Feminista de València, recorre lugares significativos para la comunidad lesbica, con el objetivo

de visibilizar los derechos y experiencias de las mujeres lesbianas en la Valencia de los años 70, 80 y 90, destacando la importancia de reconocer y valorar estas luchas en la construcción de la historia urbana [35].

Estas rutas permiten redescubrir barrios desde una óptica feminista, incorporando historias personales, luchas sociales y legados invisibilizados. En este sentido, la plataforma DONA'm MÓN podría ser un recurso útil a implementar durante estas rutas, integrando información geolocalizada, recursos multimedia y facilitando la participación activa de las personas en la construcción de esta memoria feminista urbana.

2.3.1. Referentes femeninos en la ciudad: biografías y puntos de interés

La aplicación propuesta busca poner en valor la figura de mujeres que, desde distintos campos del saber y la acción, han contribuido de manera decisiva al progreso de los derechos, el pensamiento y la cultura. Aunque muchas de ellas han nacido o trabajado en Valencia, su legado trasciende el ámbito local y constituye una aportación fundamental a la historia del feminismo y la igualdad. A través de la geolocalización de lugares clave asociados a sus vidas y obras, se propone una experiencia que va más allá de la información enciclopédica, para convertirse en un ejercicio de reconocimiento social y emocional de su presencia en la ciudad.

Una de las figuras destacadas es Concepción Aleixandre Ballester (1862–1952), una de las primeras mujeres médicas en España y pionera en el ámbito de la ginecología. Su activismo feminista se centró en la defensa del acceso de las mujeres a la educación y al ejercicio profesional de la medicina, algo profundamente innovador en su tiempo. Estudió en la Universidad de Valencia, donde enfrentó obstáculos institucionales por su condición de mujer, y desarrolló una intensa carrera médica y científica. El edificio histórico de la Facultad de Medicina de la Universitat de València es un lugar simbólico para recordar su figura y su lucha por la igualdad en el ámbito sanitario [36].

Otra mujer fundamental en la historia reciente de Valencia es Carmen Alborch (1947–2018). Fue ministra de Cultura, escritora, catedrática de Derecho Mercantil y una de las voces más influyentes del feminismo institucional en España. Su obra literaria, como *Solas* (1999), promovió un feminismo humanista accesible y empático. En Valencia, dejó una impronta profunda como directora del Instituto Valenciano de Arte Moderno (IVAM), donde impulsó una política cultural abierta, plural y comprometida con la igualdad. El IVAM, por tanto, no solo representa su labor como gestora cultural, sino también su apuesta por un feminismo integrador desde las instituciones [37].

En el ámbito de los derechos civiles, Clara Campoamor (1888–1972) ocupa un lugar esencial por su incansable lucha por el sufragio femenino, que logró defender con éxito en las Cortes Constituyentes de 1931. Si bien no tuvo una vinculación directa con Valencia, su legado ideológico inspiró a generaciones de mujeres en toda España, incluida la Comunidad Valenciana. La existencia de centros educativos como el CEIP Clara Campoamor subraya la importancia de su figura como referente pedagógico y su impacto en la formación de una conciencia feminista desde edades tempranas [38].

Retrocediendo a la Edad Media, encontramos a Isabel de Villena (1430–1490), abadesa del Real Monasterio de la Trinidad y considerada la primera escritora en lengua valenciana. Su obra *Vita Christi* constituye una relectura humanista del relato evangélico desde una perspectiva femenina, resaltando la figura de la Virgen y otras mujeres bíblicas. Isabel de Villena transformó el espacio monástico en un lugar de producción intelectual y espiritual

liderado por mujeres, algo excepcional en su época. El Monasterio de la Trinidad, donde vivió y escribió, es un lugar clave para reivindicar la aportación femenina a la literatura y la espiritualidad valencianas [39].

En la medicina, Manuela Solís Clarás (1888–1936) destaca por ser una de las primeras mujeres médicas de la ciudad. Ejerció en el Hospital Provincial de Valencia y fue una firme defensora del acceso de las mujeres a la formación sanitaria. Además de su práctica médica, se implicó en la divulgación científica y en la promoción de redes de apoyo entre profesionales sanitarias, en un momento en que la medicina era un espacio profundamente masculinizado [40]. El Hospital Provincial es, por tanto, un espacio idóneo para recordar su legado.

La aplicación también incluye representaciones artísticas contemporáneas de mujeres de gran relevancia internacional. Por ejemplo, Mae Jemison, primera mujer afroamericana en viajar al espacio, es homenajeada con un mural en el CEIP Torrefiel. Su figura simboliza la superación de barreras raciales y de género en los campos de la ciencia y la exploración espacial. Del mismo modo, Vandana Shiva, ecofeminista india y activista medioambiental, cuenta con un mural en el CEIP Ballester Fandos. Su lucha por la justicia ambiental, la soberanía alimentaria y los derechos de las mujeres rurales conecta con un feminismo global que también debe tener presencia en las narrativas educativas locales [41].

“DONA’m MÓN” busca ser un vehículo para visibilizar la presencia de estas figuras en el espacio urbano, contribuyendo así a una representación más justa y equitativa de la historia y promoviendo un futuro en el que la diversidad y la igualdad sean valores fundamentales.

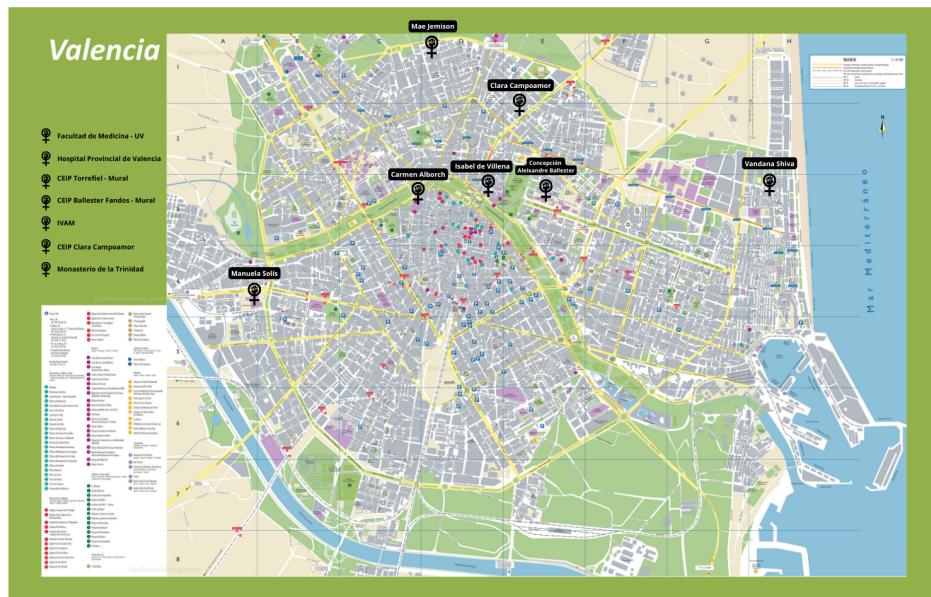


Figura 2.6: Boceto inicial del mapa de Valencia con los puntos geolocalizados de las mujeres referentes incluidos en la aplicación.

2.4. Análisis de aplicaciones similares

En el desarrollo de “DONA’m MÓN”, resulta esencial analizar aplicaciones existentes que, de manera similar, combinan geolocalización y realidad aumentada (RA) para ofrecer experiencias inmersivas y educativas. Este análisis nos permite identificar características, estrategias y tecnologías clave que podrían ser adaptadas para cumplir con los objetivos específicos del proyecto. A continuación, se presentan algunas aplicaciones relevantes cuyos enfoques, funcionalidades y tecnologías han inspirado aspectos concretos del diseño conceptual de este TFG.

2.4.1. PokéMon GO

Una de las referencias más notables es PokéMon GO, que revolucionó el mercado de aplicaciones móviles al combinar geolocalización y RA para motivar a los usuarios a explorar el mundo real. En este caso, los jugadores buscan y capturan criaturas virtuales que aparecen en puntos geográficos específicos. La aplicación también implementa un sistema de recompensas y niveles que fomenta la participación constante. Aunque el enfoque principal de PokéMon GO es el entretenimiento, su éxito demuestra cómo las tecnologías inmersivas pueden transformar la forma en que las personas interactúan con su entorno. En “DONA’m MÓN”, se busca adaptar esta dinámica de exploración y descubrimiento, pero con un enfoque educativo y cultural, utilizando los puntos de interés como portales hacia las historias de mujeres que dejaron una huella significativa en Valencia.



Figura 2.7: Interfaz de PokéMon GO mostrando un entorno real aumentado con la aparición de un PokéMon virtual y también el entorno virtual del juego.

2.4.2. Streetmuseum

Por otro lado, Streetmuseum es una aplicación que también aprovecha la RA, pero con un objetivo más centrado en la historia y la cultura. Esta herramienta permite a los usuarios visualizar imágenes y escenas históricas superpuestas al paisaje actual, ofreciendo una perspectiva del pasado en tiempo real. La capacidad de conectar visualmente el entorno moderno con eventos y figuras históricas es una inspiración directa para el proyecto. En nuestro proyecto, esta idea se adapta al permitir que los usuarios accedan a contenido multimedia –como imágenes, textos y modelos 3D– que contextualice la vida y obra de las mujeres en cada ubicación geolocalizada.



Figura 2.8: Ejemplo de Streetmuseum: visualización de una escena histórica superpuesta sobre el entorno actual de la ciudad.

2.4.3. CultuAR

CultuAR, desarrollada por la empresa española AR Vision, es una aplicación que combina la geolocalización con la realidad aumentada para ofrecer visitas culturales interactivas en más de 200 municipios. Mediante la cámara del dispositivo, los usuarios pueden visualizar reconstrucciones históricas, personajes del pasado o información multimedia superpuesta al entorno real. Una de sus características más valiosas es la capacidad de personalizar rutas culturales por municipios, integrando elementos inmersivos que enriquecen la experiencia turística y patrimonial. Esta aplicación destaca por su integración fluida con el entorno urbano y su enfoque didáctico, lo que representa una clara inspiración para DONA'm MÓN en cuanto a accesibilidad y diseño de experiencias geolocalizadas con RA.



Figura 2.9: Captura de CultuAR mostrando información sobre un monumento en el entorno real.

2.4.4. Historypin

Finalmente, Historypin se presenta como una plataforma colaborativa que permite a los usuarios subir y explorar fotografías, eventos y recuerdos históricos asociados a ubicaciones específicas. Lo más destacable de esta aplicación es su enfoque participativo, donde los usuarios pueden contribuir activamente al contenido disponible. Este modelo de colaboración inspira la posibilidad de que, en futuras versiones de “DONA'm MÓN”, los usuarios también puedan agregar información o enriquecer las narrativas existentes sobre mujeres de Valencia, convirtiendo la aplicación en una herramienta dinámica y viva.



Figura 2.10: Interfaz de Historypin donde se visualizan fotografías históricas asociadas a ubicaciones reales en el mapa y su detalle de contenido.

Estas cuatro aplicaciones ilustran diversas maneras de utilizar la tecnología para crear experiencias inmersivas e interactivas. Cada una de ellas presenta fortalezas únicas que se alinean, en mayor o menor medida, con los objetivos de DONA'm MÓN. De PokéMon GO tomamos la dinámica de exploración y gamificación; de Streetmuseum, la idea de superponer contenido histórico en el entorno actual; de CultuAR, el uso efectivo de RA para la divulgación cultural local; y de Historypin, el potencial de colaboración comunitaria. Esta combinación de estrategias y conceptos pretende hacer de DONA'm MÓN una aplicación única que conecte a los usuarios con la historia de manera inclusiva, educativa e innovadora.

2.5. Tecnologías

En este apartado se analizan las tecnologías disponibles y las soluciones existentes en el mercado que pueden ser aplicadas al desarrollo de la aplicación móvil. Este análisis incluye una revisión de herramientas de geolocalización, realidad aumentada, bases de datos y frameworks de desarrollo multiplataforma. La selección de las tecnologías se justifica considerando la funcionalidad, compatibilidad y eficiencia necesarias para cumplir con los objetivos del proyecto.

2.5.1. Plataformas de desarrollo de aplicaciones móviles

Una de las decisiones fundamentales en cualquier proyecto de ingeniería de software es la elección de la tecnología base sobre la cual se construirá la aplicación. Esta decisión afecta directamente a la arquitectura general, la experiencia de usuario, la mantenibilidad del código y la escalabilidad futura del sistema. En el contexto de este proyecto, se evaluaron varias opciones para el desarrollo de una aplicación móvil multiplataforma que integrase geolocalización, funcionalidades de realidad aumentada, y una interfaz moderna, accesible y responsive.

Unity. Unity es un motor de desarrollo multiplataforma ampliamente utilizado en los ámbitos del videojuego, simulación y experiencias interactivas en tiempo real. Su potencia radica en su motor gráfico, que permite renderizar contenido 3D de alta calidad, así como en su flexibilidad para implementar lógica de aplicación mediante scripts en C#. Unity ofrece soporte directo para tecnologías de realidad aumentada a través de

módulos como AR Foundation, que actúan como capa de abstracción sobre ARKit (iOS) y ARCore (Android), permitiendo así desarrollar una única base de código RA para múltiples plataformas.

En lo relativo a interfaces de usuario, Unity ha evolucionado en los últimos años. Su sistema tradicional basado en el Canvas permitía crear interfaces gráficas bidimensionales, pero con limitaciones de estilo y escalabilidad. Como respuesta a estas deficiencias, se introdujo UI Toolkit, un nuevo sistema de interfaz basado en un modelo declarativo, inspirado en tecnologías web como HTML y CSS. UI Toolkit permite separar la lógica de presentación del comportamiento, definiendo componentes visuales reutilizables y estilos en archivos dedicados. A pesar de estas mejoras, el desarrollo de interfaces ricas, formularios interactivos y navegación compleja continúa siendo menos eficiente que en frameworks específicos para aplicaciones móviles.

React Native. React Native es un framework desarrollado por Meta (anteriormente Facebook) para el desarrollo de aplicaciones móviles multiplataforma utilizando JavaScript y el modelo de componentes de React. Permite crear interfaces de usuario nativas reutilizando componentes declarativos que se renderizan mediante puentes hacia los componentes nativos de Android e iOS. Una de las grandes ventajas de React Native es su arquitectura reactiva, que permite gestionar estados complejos de forma eficiente, manteniendo la UI sincronizada con los datos de la aplicación.

En combinación con Expo, un conjunto de herramientas que facilita la creación y prueba de aplicaciones React Native, es posible desarrollar aplicaciones sin necesidad de compilar ni configurar código nativo. Esto permite utilizar herramientas como Expo Go para probar la aplicación directamente en un dispositivo físico mediante código QR, lo que acelera significativamente el flujo de desarrollo. React Native es especialmente potente para construir interfaces modernas, personalizadas y adaptables, con soporte para navegación compleja, formularios, animaciones y estilos dinámicos.

Flutter. Flutter es un framework desarrollado por Google para construir aplicaciones multiplataforma desde una única base de código. Utiliza el lenguaje Dart, y su arquitectura se basa en un sistema de renderizado propio que dibuja todos los elementos de la UI desde cero, lo que garantiza uniformidad visual entre plataformas. Flutter ofrece una amplia colección de widgets personalizables y anima la creación de interfaces complejas de forma declarativa y eficiente.

Aunque Flutter no fue la opción elegida para este proyecto, se considera una alternativa competitiva frente a React Native. Sin embargo, su adopción requiere dominar el lenguaje Dart, y su integración con tecnologías de realidad aumentada aún está en evolución en comparación con la madurez del ecosistema de Unity o la compatibilidad simplificada que ofrece React Native con herramientas web como AR.js y A-Frame.

Desarrollo de interfaz de usuario y elección final. Una parte crucial del proyecto es la interfaz gráfica de usuario, dado que la aplicación está destinada a un público general y debe facilitar la navegación intuitiva, la visualización de datos geográficos y el acceso a contenidos de realidad aumentada. Si bien Unity, especialmente con UI Toolkit, ofrece un entorno funcional para construir interfaces gráficas, su enfoque está más orientado a interacciones en entornos tridimensionales y menos a la gestión de flujos de navegación o formularios comunes en aplicaciones móviles.

Por el contrario, React Native destaca precisamente en este tipo de interfaz. Su modelo declarativo, su sistema de componentes reutilizables, su integración con herramientas modernas de desarrollo (como React Navigation, Redux o Context API), y su compatibi-

lidad con Expo, permiten construir aplicaciones móviles escalables y con una experiencia de usuario optimizada.

Por estos motivos, y tras un análisis técnico detallado, se decidió utilizar React Native con Expo como entorno principal de desarrollo para esta aplicación, reservando el uso de tecnologías gráficas (como la realidad aumentada) a componentes embebidos que se integran dentro de esta arquitectura general. Esta elección permite combinar la robustez de React para la UI con la flexibilidad de integrar experiencias multimedia mediante tecnologías web.

2.5.2. Geolocalización

La geolocalización es uno de los pilares fundamentales de la aplicación, ya que permite situar puntos de interés específicos en un mapa, facilitando que los usuarios exploren la ciudad de Valencia mientras descubren la historia de mujeres destacadas. Para implementar esta funcionalidad, existen diversas herramientas ampliamente utilizadas en el desarrollo de aplicaciones móviles:

- **Google Maps API:** Es una solución robusta y versátil que permite integrar mapas interactivos, obtener datos de ubicación en tiempo real y personalizar la visualización según las necesidades del proyecto. Ofrece documentación detallada y soporte técnico, lo que facilita su integración. Sin embargo, su principal desventaja es que su plan gratuito únicamente se ofrece durante un periodo de prueba de corta duración y su coste se incrementa en proyectos con un gran volumen de usuarios o solicitudes [42].
- **Mapbox:** Alternativa potente que combina funcionalidades avanzadas, un alto nivel de personalización estética y una política de precios más flexible que Google Maps API, ya que su plan gratuito no tiene un límite de tiempo. Es especialmente útil en aplicaciones que requieren un diseño visual distintivo. No obstante, su integración puede resultar más compleja para desarrolladores con poca experiencia [43].
- **OpenStreetMap (OSM):** Opción de código abierto que permite el acceso y uso gratuito de datos cartográficos, además de ofrecer un alto nivel de personalización. Esta plataforma colaborativa es especialmente adecuada para proyectos con presupuestos limitados o académicos, ya que no implica costes de licencia. Aunque carece de algunas funcionalidades avanzadas nativas, como el enrutamiento en tiempo real o servicios de geolocalización enriquecidos, su flexibilidad y libertad de uso la convierten en una solución funcional y accesible [44].
- **Mapas nativos del dispositivo con React Native:** Esta opción permite aprovechar las capacidades de geolocalización del propio sistema operativo del dispositivo móvil (iOS o Android) mediante bibliotecas como react-native-maps. Es una solución eficiente y ligera, que facilita la integración con el hardware del dispositivo y proporciona una experiencia fluida. Además, evita costes derivados de servicios externos, lo que la convierte en una alternativa atractiva para proyectos con recursos limitados [45].

En este proyecto se ha optado por utilizar mapas nativos con React Native, ya que permiten una integración directa con la aplicación móvil, ofrecen un rendimiento óptimo, y eliminan la dependencia de servicios externos de pago. Esta elección responde a criterios de eficiencia, sostenibilidad económica y control sobre el desarrollo.

2.5.3. Realidad Aumentada (RA)

Una vez definida la arquitectura general de la aplicación y confirmado el uso de React Native como entorno principal de desarrollo, fue necesario analizar detalladamente qué tecnologías de realidad aumentada (RA) podían integrarse de forma eficiente, respetando las limitaciones del ecosistema de desarrollo adoptado. Uno de los objetivos fundamentales era conservar la compatibilidad con Expo Go, evitando procesos de compilación nativa que dificultasen el flujo de desarrollo iterativo. Bajo esta premisa, se realizó un estudio de las distintas alternativas existentes, evaluando tanto su madurez técnica como su grado de integración con React Native.

Realidad aumentada con tecnologías nativas. Las soluciones nativas ofrecen, en general, el mayor grado de precisión, rendimiento y acceso a capacidades avanzadas del dispositivo. Entre estas destacan ARKit (para dispositivos iOS) y ARCore (para Android), los kits de desarrollo oficial proporcionados por Apple y Google, respectivamente. Ambas plataformas permiten el seguimiento del entorno, la detección de planos horizontales y verticales, la oclusión de objetos virtuales con respecto al entorno físico, el anclaje espacial y la iluminación adaptativa. Estas herramientas están diseñadas para crear experiencias inmersivas de alta calidad y son ampliamente utilizadas en entornos profesionales y comerciales.

Sin embargo, una de sus limitaciones fundamentales es su naturaleza monoplataforma. Para desarrollar una aplicación de RA que funcione tanto en Android como en iOS, sería necesario implementar dos versiones independientes, o bien utilizar un entorno unificador como Unity, que permite compilar una única base de código para múltiples sistemas operativos.

Unity, a través de su módulo AR Foundation, permite abstraer las diferencias entre ARKit y ARCore, y desarrollar experiencias RA multiplataforma con una arquitectura común. Unity es particularmente potente en términos de renderizado gráfico, gestión de modelos tridimensionales, simulación física y animación en tiempo real, por lo que constituye una de las herramientas más robustas para el desarrollo de RA avanzada.

No obstante, su integración con aplicaciones desarrolladas en React Native es compleja. El enfoque convencional implicaría desarrollar toda la aplicación en Unity, lo cual, como se ha discutido anteriormente, penaliza la usabilidad y escalabilidad de la interfaz. Por este motivo, se exploró también la posibilidad de utilizar Unity como una librería nativa embebida dentro de una aplicación React Native.

Unity como librería nativa integrada en React Native. Unity permite exportar un proyecto como una librería nativa —un archivo .aar para Android o .framework para iOS— que puede integrarse dentro de una aplicación móvil desarrollada con otras tecnologías, como React Native. Este enfoque, conocido como Unity as a Library, ofrece la posibilidad de desarrollar únicamente la parte de RA en Unity y exponerla como un módulo invocable desde la aplicación principal.

Mediante este patrón arquitectónico, es posible mantener la lógica general y la interfaz de usuario desarrolladas en React Native, y delegar en Unity únicamente el renderizado y gestión de la escena RA. La comunicación entre ambos entornos se realiza mediante puentes nativos que deben implementarse en Java/Kotlin (para Android) y Objective-C/Swift (para iOS).

Este enfoque ha sido demostrado en proyectos como react-native-unity, que ofrecen ejemplos funcionales de integración, así como en la documentación oficial de Unity (Unity

as a Library). No obstante, esta integración presenta múltiples desafíos técnicos:

- Es necesario ejectar el proyecto Expo, abandonando así la posibilidad de utilizar Expo Go y el flujo de desarrollo simplificado que ofrece.
- La configuración del proyecto requiere conocimientos avanzados en desarrollo nativo para cada plataforma.
- La integración y sincronización de eventos entre Unity y React Native debe implementarse manualmente.

Por tanto, aunque Unity como librería nativa representa una solución muy potente desde el punto de vista gráfico y funcional, su coste de integración y mantenimiento dentro de una arquitectura basada en React Native la convierte en una opción poco viable para este proyecto, cuyo foco está en la accesibilidad, rapidez de desarrollo y compatibilidad multiplataforma sin compilación nativa.

Realidad aumentada en React Native con tecnologías web. En busca de una alternativa más ligera y plenamente compatible con React Native y Expo, se optó por explorar las soluciones basadas en tecnologías web. Una de las más destacadas es AR.js, una librería de código abierto que permite implementar experiencias de RA directamente en navegadores móviles, sin necesidad de acceso a capas nativas. AR.js se basa en WebGL, Three.js y A-Frame, y permite implementar RA sobre marcadores visuales (marker-based AR) mediante la detección de patrones gráficos como Hiro, Kanji o diseños personalizados.

Complementando AR.js, el framework A-Frame facilita la construcción declarativa de escenas 3D y RA mediante HTML. Permite importar modelos .glb, añadir iluminación, cámaras, interacciones y animaciones sin necesidad de escribir código complejo de bajo nivel. Esta aproximación reduce significativamente la barrera de entrada al desarrollo de RA, y se adapta perfectamente al modelo de desarrollo web moderno.

Las escenas construidas con A-Frame y AR.js pueden ser desplegadas como aplicaciones web estáticas (por ejemplo, en GitHub Pages), e integradas dentro de la aplicación React Native utilizando el componente WebView, que actúa como un navegador embebido. Este enfoque permite mantener la lógica general y la interfaz de la aplicación dentro del ecosistema de React Native, mientras que la experiencia de RA se encapsula como una unidad independiente, reutilizable y fácilmente actualizable.

Viro React. Otra alternativa evaluada fue Viro React, un framework orientado a la creación de experiencias inmersivas en RA y realidad virtual directamente en React Native. Viro permite cargar modelos 3D, definir cámaras, aplicar animaciones, detectar superficies, y renderizar objetos virtuales en función del entorno físico. Su enfoque declarativo está alineado con la filosofía de React y ofrece un entorno de desarrollo intuitivo para experiencias visuales.

No obstante, Viro no es compatible con Expo Go, ya que depende de módulos nativos que requieren compilar la aplicación. Para utilizarlo, es necesario ejectar el proyecto y utilizar herramientas como EAS Build para generar versiones personalizadas de la app. Esta necesidad de compilación rompe el flujo de desarrollo ágil basado en Expo y contradice uno de los principios técnicos fundamentales del proyecto: evitar configuraciones nativas complejas para preservar la portabilidad y escalabilidad del sistema.

Elección final. Después de un análisis riguroso, se optó por utilizar AR.js combinado con A-Frame, integrados en la aplicación React Native mediante un componente WebView.

Esta solución permite cumplir los objetivos funcionales de la aplicación (visualización de modelos 3D sobre marcadores físicos), respetando al mismo tiempo los requisitos técnicos del entorno (compatibilidad con Expo, desarrollo ágil, y arquitectura limpia).

La elección de esta tecnología responde a criterios técnicos sólidos:

- Portabilidad total: la experiencia RA se ejecuta en el navegador y es accesible desde cualquier dispositivo moderno.
- Desacoplamiento arquitectónico: la escena RA es independiente del código base de la aplicación.
- Facilidad de mantenimiento: se pueden actualizar las escenas de RA sin necesidad de recompilar la app.
- Compatibilidad con Expo Go: se preserva el flujo de desarrollo iterativo y multiplataforma.

En resumen, esta arquitectura permite combinar las ventajas del desarrollo móvil moderno (mediante React Native) con la flexibilidad de tecnologías web para RA, consiguiendo una solución técnica equilibrada, funcionalmente completa y sostenible a largo plazo.

2.5.4. Base de Datos

2.5.5. Arquitectura de persistencia y gestión de datos

En el desarrollo de esta aplicación se ha optado por una arquitectura desacoplada que separa claramente el frontend y el backend. El frontend se implementa en React Native, lo que permite el despliegue multiplataforma en dispositivos móviles, mientras que el backend se desarrolla en Django, un framework web de alto nivel que facilita la creación de aplicaciones robustas y seguras [46].

Persistencia de datos en entornos relacionales y no relacionales

La persistencia de datos es un aspecto fundamental en el desarrollo de aplicaciones que requieren almacenar información de manera duradera. Existen dos enfoques principales para gestionar esta persistencia: las bases de datos relacionales, que estructuran la información en tablas con relaciones definidas mediante claves, y las bases de datos no relacionales, que permiten una mayor flexibilidad al almacenar los datos en formatos como documentos, grafos o pares clave-valor [47].

Las bases de datos relacionales son especialmente adecuadas cuando los datos presentan una estructura definida, relaciones entre entidades y necesidad de integridad referencial, mientras que las no relacionales ofrecen ventajas en contextos donde la escalabilidad horizontal, la alta disponibilidad o la variabilidad en los esquemas de datos son prioritarias [48].

Django y la arquitectura MVT

Django se basa en la arquitectura Modelo–Vista–Plantilla (MVT), un patrón similar al clásico MVC (Model–View–Controller), adaptado al ecosistema web de Python [49]. Este patrón organiza el código en tres componentes:

- **Modelo (Model):** Define la estructura de los datos y su comportamiento a través de clases Python, que se traducen automáticamente en tablas mediante el ORM (Object-Relational Mapping) de Django.
- **Vista (View):** Gestiona la lógica de negocio y responde a las solicitudes HTTP, procesando la información que llega desde el frontend.
- **Plantilla (Template):** Presenta la información al usuario utilizando HTML enriquecido con etiquetas dinámicas, aunque en este proyecto su papel es mínimo, ya que la interfaz se renderiza completamente en React Native.

La interacción entre Django y la base de datos está mediada por su ORM, lo que permite manipular los datos mediante objetos Python en lugar de consultas SQL explícitas. Esta capa de abstracción incrementa la seguridad —previniendo ataques como la inyección SQL— y facilita el mantenimiento del código [49].

Sistema de Gestión de Bases de Datos: MySQL

Para este proyecto se ha elegido MySQL como sistema de gestión de bases de datos relacional (SGBDR), debido a su madurez, rendimiento y plena compatibilidad con Django [50]. MySQL permite estructurar los datos de manera eficiente y soporta operaciones complejas con múltiples relaciones entre tablas, lo que resulta fundamental para gestionar tanto la información relacionada con las figuras históricas como los datos de geolocalización, multimedia y analíticas del uso de la app.

La configuración entre Django y MySQL se realiza mediante el uso de controladores compatibles (como `mysqlclient`), lo que asegura una integración fluida y un rendimiento óptimo.

Comparativa entre sistemas de bases de datos

A continuación se muestra una tabla comparativa entre los principales sistemas de gestión de bases de datos considerados:

Característica	SQLite	MySQL	MariaDB	MongoDB
Tipo	Relacional	Relacional	Relacional	No relacional
Escalabilidad	Baja	Alta	Alta	Muy alta
Rendimiento	Medio	Alto	Muy alto	Alto
Integridad referencial	Sí	Sí	Sí	No
Compatibilidad Django	Nativa	Completa	Completa	Parcial (con plugins)
Flexibilidad de esquema	Baja	Media	Media	Alta
Facilidad de configuración	Muy fácil	Media	Media	Media
Comunidad y soporte	Moderada	Muy activa	Activa	Muy activa

Cuadro 2.2: Comparativa entre SQLite, MySQL, MariaDB y MongoDB [47, 51, 52].

Consideraciones sobre otras tecnologías: MariaDB, SQLite, MongoDB y Cassandra

Aunque inicialmente se contempló el uso de MariaDB, una bifurcación de MySQL con mejoras en rendimiento y licencias completamente abiertas, finalmente se optó por MySQL por razones de compatibilidad institucional y soporte técnico [51]. SQLite, si bien está soportada nativamente por Django y resulta útil en fases de desarrollo o en aplicaciones de bajo requerimiento, no ofrece la escalabilidad ni la gestión de concurrencia necesarias para este proyecto [53].

En cuanto a las bases de datos no relacionales, cabe mencionar tecnologías como MongoDB y Cassandra. MongoDB, basada en documentos JSON, es ampliamente utilizada en aplicaciones que requieren estructuras de datos flexibles y cambios frecuentes en el esquema [52]. Cassandra, por su parte, está orientada a entornos distribuidos con grandes volúmenes de datos y alta disponibilidad [54]. Sin embargo, el enfoque relacional fue considerado más adecuado para las necesidades estructuradas y normalizadas del presente trabajo.

Capítulo 3

Requisitos, especificaciones, coste, riesgos, viabilidad

3.1. Requisitos

3.1.1. Requisitos funcionales

Los requisitos funcionales describen las acciones que el sistema debe poder realizar para satisfacer las necesidades del usuario y del administrador. En este proyecto, se centran en el uso de geolocalización, realidad aumentada, gestión de contenido y experiencia interactiva.

Lista de requisitos funcionales:

3.1.2. Requisitos funcionales

RF1: La aplicación debe permitir a los usuarios visualizar un mapa interactivo con los puntos geolocalizados de interés.

RF2: El sistema debe detectar la ubicación actual del usuario mediante GPS.

RF3: Al acercarse físicamente a un punto geolocalizado, la aplicación debe desbloquear el contenido asociado.

RF4: Cada punto geolocalizado debe estar vinculado a una mujer histórica, con contenido educativo multimedia.

RF5: La aplicación debe mostrar una ficha informativa con texto, imágenes y/o audio sobre cada figura femenina.

RF6: La app debe permitir visualizar elementos de realidad aumentada (como imágenes, objetos 3D o figuras animadas) al enfocar con la cámara.

RF7: La lista de puntos y el mapa deben permitir filtrar los lugares por el ámbito de investigación de las mujeres representadas.

RF8: El usuario debe poder ver qué puntos ha desbloqueado y cuáles no.

RF9: La aplicación debe permitir consultar la lista completa de mujeres disponibles con sus respectivas ubicaciones.

RF10: El usuario debe recibir una notificación cuando esté cerca de un punto no visitado.

- RF11: Debe existir un sistema de rutas que conecten diferentes puntos de interés temáticamente.
- RF12: Al pulsar sobre un punto en el mapa o la lista, debe abrirse su ficha detallada, desde la cual el usuario podrá marcar el lugar como visitado o no.
- RF13: El sistema debe permitir guardar el progreso del usuario (puntos visitados).
- RF14: El usuario debe disponer de un historial donde se registre la fecha y hora de cada punto visitado.
- RF15: La aplicación debe ser funcional tanto en dispositivos Android como iOS.
- RF16: Debe existir un panel de administración web para gestionar los puntos, contenidos y figuras históricas.
- RF17: El administrador debe poder añadir, editar o eliminar mujeres, ubicaciones y materiales multimedia desde el panel.
- RF18: El sistema debe permitir importar imágenes, vídeos y modelos 3D desde el panel de administración.
- RF18b: El administrador debe poder gestionar áreas de investigación y rutas temáticas desde el panel de administración.
- RF18c: El administrador debe poder consultar métricas y estadísticas de uso, como puntos más visitados y usuarios activos.
- RF18d: El administrador debe poder gestionar los permisos de otros administradores si existen varios roles.
- RF19: El sistema debe validar que no se puede desbloquear un punto si el usuario no está físicamente cerca.
- RF20: La app debe ofrecer retroalimentación visual y sonora al desbloquear contenido.
- RF21: El sistema debe enviar datos al backend para mantener sincronizado el progreso del usuario.
- RF22: La aplicación debe cargarse en menos de 5 segundos desde su apertura inicial.
- RF23: La aplicación debe permitir a los usuarios registrarse y acceder mediante un sistema de inicio de sesión.
- RF24: Los usuarios deben poder iniciar sesión con un correo electrónico y contraseña.
- RF25: La aplicación debe mantener la sesión iniciada hasta que el usuario decida cerrarla.
- RF26: El usuario debe poder editar los datos de su perfil desde la aplicación.
- RF27: El administrador debe poder acceder, desde el panel de gestión, a la lista de usuarios registrados.
- RF28: El administrador debe poder visualizar el perfil de cada usuario, incluyendo su progreso (puntos visitados).

- RF29: El sistema debe permitir al administrador eliminar usuarios o restablecer sus datos si es necesario.
- RF30: Al escanear un código QR de un lugar, se debe actualizar el progreso de rutas temáticas dentro de la aplicación.
- RF31: El sistema debe permitir filtrar los puntos por estado de visita (visitados o no visitados).
- RF32: Los filtros por estado de visita y por ámbito deben poder aplicarse de forma combinada.
- RF33: El listado de puntos debe estar ordenado por cercanía al usuario.
- RF34: La distancia en tiempo real entre el usuario y cada punto debe mostrarse junto a cada entrada del listado.
- RF35: La lista de puntos debe mostrar de forma resumida información del lugar y de la mujer representada.
- RF36: Desde la ficha detallada de un punto debe poder compartirse su contenido por redes sociales.
- RF37: El usuario debe poder acceder a una sección de rutas temáticas disponibles y ver los puntos que componen cada una.
- RF38: Los puntos de una ruta no pueden marcarse como visitados manualmente: sólo se desbloquean escaneando el código QR correspondiente.
- RF39: La ruta temática se debe marcar como completa y registrar en el historial del usuario al visitar todos los puntos
- RF40: Debe existir una pestaña específica con los puntos que disponen de realidad aumentada, accesibles si el usuario está dentro de un radio determinado.
- RF41: Al seleccionar un punto con RA disponible, debe abrirse la cámara y mostrarse el modelo correspondiente.
- RF42: El usuario debe poder consultar una sección de perfil donde editar sus datos personales, ver su historial de lugares visitados y rutas completadas, y cerrar sesión.

3.1.3. Requisitos no funcionales

Los requisitos no funcionales indican cómo debe comportarse el sistema más allá de lo que hace: establecen criterios de calidad, restricciones técnicas o normativas, y condiciones necesarias para su correcto desarrollo, despliegue y uso.

Requisitos no funcionales del producto

Estos requisitos definen las propiedades internas del sistema y sus comportamientos esperados, como rendimiento, usabilidad, seguridad o compatibilidad técnica.

- RNF1: La aplicación debe cargarse completamente en menos de 5 segundos desde su apertura.

RNF2: El sistema debe ser capaz de gestionar al menos 100 usuarios simultáneos sin caída de rendimiento.

RNF3: Los modelos 3D deben estar optimizados para dispositivos móviles, con un tamaño inferior a 10 MB por unidad.

RNF4: El consumo de batería durante el uso continuo no debe superar el 10 % por hora.

RNF5: La interfaz debe ser intuitiva y permitir su uso sin formación previa.

RNF6: La tipografía y botones deben adaptarse automáticamente a diferentes resoluciones de pantalla.

RNF7: La aplicación debe ofrecer un modo accesible con texto ampliado y audio narrado.

RNF8: La navegación debe ser posible con una sola mano (diseño mobile-first).

RNF9: Compatible con Android 10+ y iOS 13+.

RNF10: El panel de administración debe ser accesible desde navegadores modernos (Chrome, Firefox, Safari, Edge).

RNF11: Las contraseñas de los usuarios deben almacenarse cifradas en la base de datos.

RNF12: La comunicación entre cliente y servidor debe estar cifrada mediante HTTPS.

RNF13: El backend debe estar desarrollado en Django y utilizar PostGIS (extensión de PostgreSQL para datos geoespaciales) como sistema gestor de bases de datos.

RNF14: La aplicación debe estar desarrollada con A-Frame, integrando geolocalización y RA mediante tecnologías WebXR y bibliotecas compatibles con A-Frame.

RNF15: La arquitectura debe ser modular y permitir añadir nuevos puntos o contenidos sin modificar el código base.

RNF16: Se debe estructurar el código y documentarlo adecuadamente para facilitar mantenimiento y ampliación.

Requisitos no funcionales de la organización

Estos requisitos están relacionados con decisiones internas del equipo desarrollador o institución que afectan a los métodos de trabajo, tecnologías permitidas o estructura del proyecto.

RNF17: El sistema debe permitir su despliegue en un servidor que soporte Django y PostGIS (PostgreSQL) según los medios disponibles en la UV.

RNF18: Debe evitarse el uso de servicios de alto coste por uso intensivo, priorizando soluciones open source cuando sea posible (por ejemplo, evitar cuotas de Google Maps si es viable usar OpenStreetMap).

RNF19: El panel de administración debe incluir autenticación de administradores con control de permisos.

RNF20: El proyecto debe desarrollarse siguiendo buenas prácticas de ingeniería del software: control de versiones, pruebas, documentación técnica y separación por capas.

Requisitos no funcionales externos

Son aquellos impuestos por normativas legales, estándares externos o políticas públicas que el sistema debe cumplir.

RNF21: El sistema debe cumplir con el Reglamento General de Protección de Datos (RGPD) en lo referente al tratamiento de datos personales.

RNF22: Los usuarios deben aceptar la política de privacidad y condiciones de uso antes de registrarse.

RNF23: El acceso a los datos de usuario debe estar restringido y protegido contra accesos no autorizados.

RNF24: El almacenamiento de datos personales (nombre, email, localización) debe hacerse de forma segura y limitada a lo estrictamente necesario.

3.2. Especificaciones

Una vez definidos los requisitos funcionales del sistema, es posible descomponerlos en funcionalidades concretas que formarán parte de la aplicación y su panel de gestión. A continuación se detalla el conjunto de acciones que los usuarios (tanto visitantes como administradores) podrán realizar.

Funcionalidades relacionadas con la geolocalización y el mapa

- F1. Visualizar un mapa interactivo con los puntos de interés distribuidos por Valencia.
- F2. Detectar en tiempo real la ubicación actual del usuario.
- F3. Mostrar puntos cercanos en función de la ubicación.
- F4. Notificar al usuario cuando se acerque a un punto no visitado.
- F5. Restringir el acceso al contenido si el usuario no está físicamente cerca del punto.

Funcionalidades educativas y de contenido

- F6. Acceder a la ficha de una mujer histórica al llegar a un punto.
- F7. Mostrar contenido multimedia asociado: texto, imágenes, audios o vídeos.
- F8. Activar realidad aumentada al enfocar con la cámara del dispositivo.
- F9. Visualizar objetos 3D o elementos históricos en RA.
- F10. Consultar un listado general de todas las mujeres incluidas en la app.
- F11. Reproducir contenido offline si ya ha sido descargado previamente.

Funcionalidades de interacción y gamificación

- F12. Desbloquear logros al visitar puntos o completar rutas temáticas.

F13. Mostrar estadísticas personales (puntos visitados, logros obtenidos).

F14. Ofrecer retroalimentación visual y sonora al desbloquear un nuevo punto.

Funcionalidades de usuario

F15. Registrar un nuevo usuario mediante correo electrónico y contraseña.

F16. Iniciar sesión en la aplicación.

F17. Mantener la sesión activa entre usos.

F18. Cerrar sesión manualmente.

F19. Aceptar política de privacidad y condiciones de uso al registrarse.

Funcionalidades del panel de administración

F20. Iniciar sesión como administrador desde el backend web.

F21. Añadir nuevas figuras históricas con nombre, descripción y material multimedia.

F22. Asociar cada mujer con un punto geolocalizado.

F23. Cargar imágenes, audios, vídeos y modelos 3D desde el panel.

F24. Editar o eliminar puntos o figuras ya creadas.

F24b. Gestionar áreas de investigación y rutas temáticas (crear, editar, eliminar).

F24c. Consultar y descargar métricas de uso: puntos más visitados, número de usuarios activos, etc.

F24d. Gestionar permisos y roles de administradores (si aplica).

F25. Ver la lista de usuarios registrados.

F26. Consultar el progreso de cada usuario (puntos desbloqueados, logros).

F27. Eliminar usuarios si fuese necesario.

F28. Consultar métricas de uso: puntos más visitados, número de usuarios activos, etc.

Funcionalidades generales

F29. Adaptar la interfaz a distintos tamaños de pantalla.

F30. Activar modo accesible con texto grande y audio.

F31. Navegar por la app de forma intuitiva y con accesibilidad mobile-first.

F32. Garantizar el uso fluido y sin errores tanto en Android como en iOS.

Funcionalidades relacionadas con rutas y QR

- F33. Ver una lista de rutas temáticas con sus nombres y descripciones.
- F34. Consultar los puntos que forman parte de cada ruta.
- F35. Escanear códigos QR para desbloquear puntos de una ruta.
- F36. Impedir que los puntos de una ruta se marquen como visitados sin escanear su QR.
- F37. Registrar en el historial del usuario las rutas completadas.

Funcionalidades de realidad aumentada (RA)

- F38. Ver una lista filtrada de los puntos con contenido RA disponible.
- F39. Mostrar únicamente los puntos con RA que estén a una distancia determinada del usuario.
- F40. Al seleccionar uno de estos puntos, abrir automáticamente la cámara y cargar el modelo RA asociado.

Funcionalidades del perfil de usuario

- F41. Acceder a la pantalla de perfil desde el menú principal.
- F42. Editar los datos personales del usuario.
- F43. Consultar el historial de puntos visitados, con fecha y hora.
- F44. Consultar las rutas temáticas que el usuario ha completado.

3.3. Ciclo de vida del software

Para el desarrollo del proyecto **DONA'm MÓN**, se ha adoptado el **modelo de desarrollo en cascada**, una metodología secuencial en la que cada fase del ciclo de vida del software debe completarse completamente antes de pasar a la siguiente. Este enfoque estructurado es especialmente útil en proyectos en los que los requisitos están claramente definidos desde el inicio, como es el caso de esta aplicación, cuya funcionalidad ha sido minuciosamente detallada y delimitada en la fase de análisis.

Ventajas del modelo en cascada

- **Claridad y estructura:** permite dividir el proyecto en fases bien diferenciadas, con objetivos y entregables específicos en cada una.
- **Planificación precisa:** facilita la estimación de tiempos y recursos al establecer un flujo lineal de trabajo sin iteraciones constantes.
- **Documentación completa:** promueve una elaboración detallada de documentación desde las primeras etapas del proyecto.
- **Control de calidad:** permite realizar revisiones sistemáticas al finalizar cada fase, asegurando la solidez del sistema antes de avanzar.

Justificación de su elección

El modelo en cascada ha resultado adecuado para este proyecto debido a la existencia de una base de requisitos bien definidos y una visión clara de las funcionalidades esperadas desde el principio. La aplicación combina tecnologías como geolocalización, escaneo de códigos QR y realidad aumentada, pero todas estas funcionalidades están organizadas en módulos claramente separados, lo que ha permitido planificar su desarrollo de forma lineal. Además, el hecho de que el proyecto haya sido desarrollado por una sola persona favorece un enfoque ordenado y controlado, minimizando el riesgo de desviaciones.

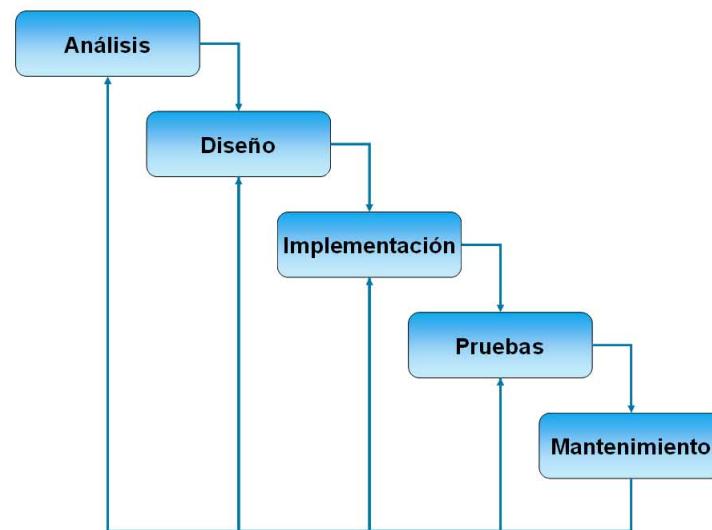


Figura 3.1: Representación del modelo de desarrollo en cascada.

3.4. Planificación temporal

3.4.1. Descomposición de tareas por fases

La planificación se ha dividido en siete bloques principales: **Inicio**, **Estado del arte**, **Análisis**, **Diseño**, **Implementación**, **Pruebas** y **Documentación**, en concordancia con el modelo incremental utilizado.

1. Inicio

- Estudio del problema que resuelve la app
- Definición de objetivos del TFG
- Planificación inicial del proyecto

2. Estado del arte

- Revisión de apps similares educativas, con RA y mapas
- Estudio de mujeres y sus lugares en Valencia
- Estudio de frameworks de RA, mapas y escáneres QR

- Selección de tecnologías definitivas (React Native, Django, AFrame)

3. Análisis

- Definición detallada de requisitos funcionales y no funcionales
- Modelado inicial del sistema: navegación y casos de uso
- Modelado entidad-relación de la base de datos
- Definición de métricas y rutas

4. Diseño

- Diseño de la arquitectura frontend y backend
- Diseño de interfaz móvil y estilo visual
- Diseño de pantallas clave: login, listado, mapa, RA, etc.

5. Implementación backend

- Creación del proyecto Django y configuración Docker
- Creación de modelos de datos y migraciones
- Implementación de API REST para usuarios, lugares y rutas
- Sistema de login y sesiones
- Registro de visitas, logros y progreso

6. Implementación frontend

- Configuración base y navegación
- Registro e inicio de sesión de usuarios
- Implementación del listado y filtros por ámbito y visitas
- Implementación del historial de usuario
- Implementación de mapa con geolocalización
- Escaneo de QR y navegación a detalle
- Sincronización de datos con backend

7. Implementación de realidad aumentada

- Configuración de AR.js y marcador personalizado
- Activación de RA según la ubicación (menos de 1km)
- Visualización de modelo 3D y contenido RA educativo

8. Pruebas y validación

- Pruebas funcionales de cada módulo
- Pruebas completas con usuarios externos
- Corrección de errores y mejoras detectadas

9. Documentación

- Redacción progresiva de la memoria (paralela a todo el desarrollo)
- Revisión completa, maquetación y anexos finales

3.4.2. Estimación temporal

La técnica de estimación utilizada es la media beta de probabilidad. Para cada tarea se ha asignado un tiempo optimista (TO), pesimista (TP) y más probable (TM), y se ha calculado la duración estimada (TE) con la siguiente fórmula:

$$TE = \frac{TO+4(TM+TP)}{6}$$

Cuadro 3.1: Estimación por tres valores de las tareas del proyecto

Tarea	TO	TP	TM	TE
Estudio del problema	1	3	2	2.00
Definición de objetivos	1	2	1	1.17
Planificación inicial	1	2	1	1.17
Revisión apps similares	2	4	3	3.00
Estudio lugares mujeres	1	3	2	2.00
Frameworks RA y QR	2	4	3	3.00
Selección tecnologías	1	2	1	1.17
Requisitos funcionales	2	4	3	3.00
Casos de uso y navegación	1	3	2	2.00
Modelo E-R BD	1	3	2	2.00
Def. métricas y rutas	1	2	1	1.17
Diseño arquitectura	2	4	3	3.00
Diseño interfaz visual	2	5	3	3.17
Pantallas clave	2	4	3	3.00
Django y Docker	1	3	2	2.00
Modelos y migraciones	1	3	2	2.00
API usuarios y lugares	3	5	4	4.00
Login y sesiones	1	3	2	2.00
Registro visitas y logros	1	3	2	2.00
Conf. navegación base	1	3	2	2.00
Registro e inicio sesión	1	3	2	2.00
Listado y filtros	2	4	3	3.00
Historial de usuario	1	3	2	2.00
Mapa geolocalizado	2	4	3	3.00
Escaneo QR	1	3	2	2.00
Sincronización datos	1	3	2	2.00
Config. AR.js y marcador	1	3	2	2.00
Desbloqueo por distancia	1	3	2	2.00
Visualización RA	2	4	3	3.00
Pruebas funcionales	2	4	3	3.00
Pruebas con usuarios	1	3	2	2.00
Corrección errores	1	3	2	2.00
Redacción progresiva	4	6	5	5.00
Revisión y entrega final	2	4	3	3.00
Total estimado	73	127	96	100.00 días

3.4.3. Diagrama de Gantt

Para representar la planificación temporal del proyecto de forma visual, se ha elaborado un diagrama de Gantt (véase la Figura 3.2) utilizando Microsoft Project, teniendo en cuenta las dependencias y la posibilidad de solapar tareas como la documentación progresiva.

A pesar de que la estimación por tres valores del proyecto arroja una duración total de aproximadamente **100 días** (equivalente a **800 horas**), el desarrollo real se ha planificado a lo largo de **80 días laborables** (como puede observarse en el diagrama de Gantt), lo que supone un total de **640 horas efectivas** de trabajo, distribuidas en jornadas de **8 horas al día**.

Cabe destacar que:

- El proyecto se ha realizado en paralelo con el *Máster en Tecnologías Web, Computación en la Nube y Aplicaciones Móviles*.
- Además, durante este periodo se estaban llevando a cabo *prácticas extracurriculares en el Instituto de Robótica y Tecnologías de la Información y la Comunicación (IRTIC)*, donde también se ha avanzado el desarrollo del TFG.
- No se ha trabajado durante los fines de semana, por lo que los 80 días corresponden exclusivamente a días laborables.

La fecha de inicio del proyecto fue el **3 de marzo de 2025**, y la fecha de finalización fue el **20 de junio de 2025**. El proyecto ha sido planificado y ejecutado con el objetivo de ser presentado en la **primera convocatoria**.

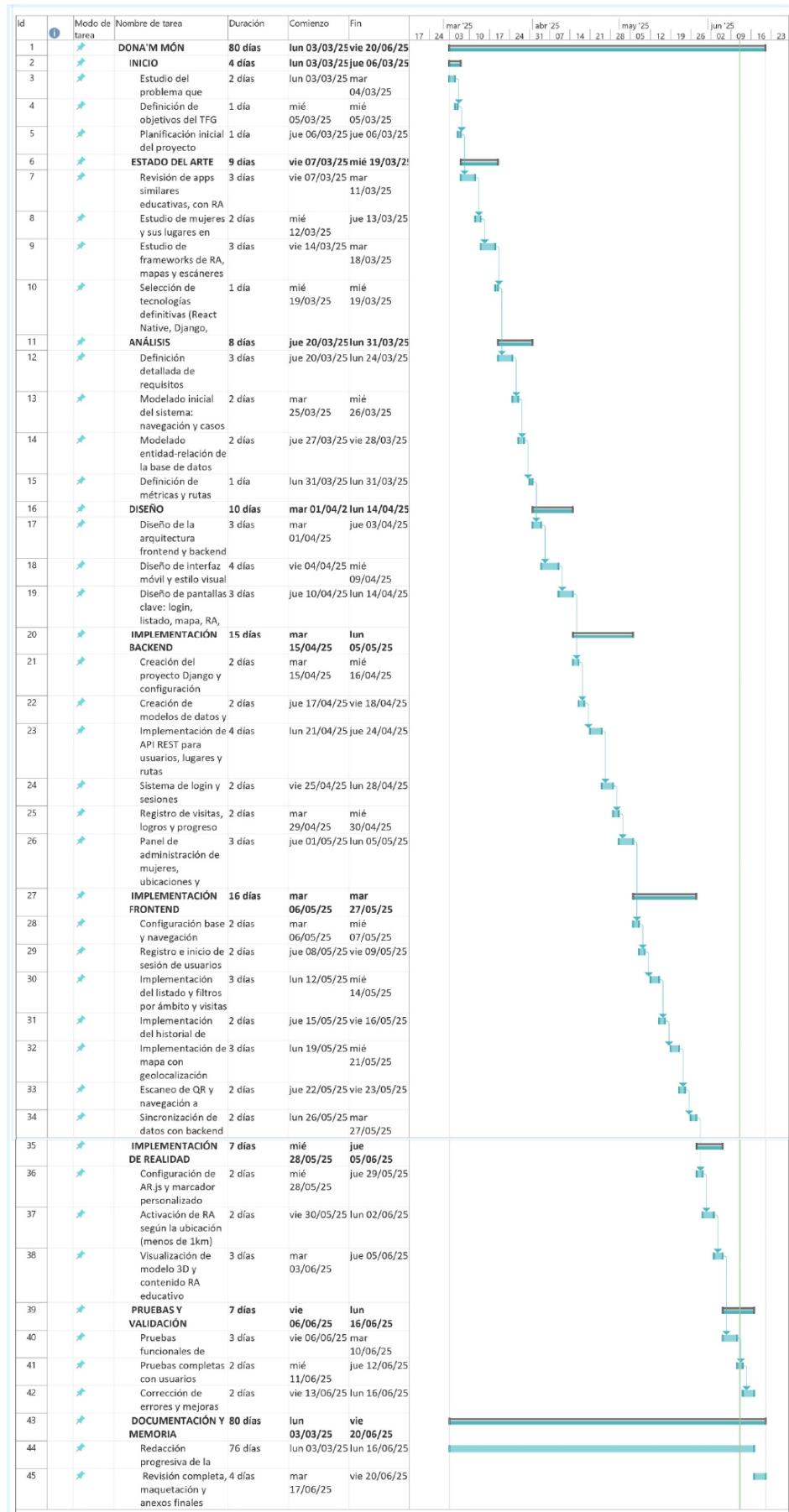


Figura 3.2: Diagrama de Gantt detallado del proyecto.

3.5. Estimación de costes

La estimación económica del proyecto se ha realizado distinguiendo entre:

- **Costes directos de personal:** incluyen el salario bruto anual para cada perfil profesional, junto con los costes de Seguridad Social correspondientes (aproximadamente un 30 % adicionales). Se han considerado perfiles distintos (Jefe de Proyecto, Desarrollador Backend, Desarrollador Frontend, Desarrolador de Realidad Tester/-QA y Modelador 3D), aunque el proyecto sea llevado a cabo por una sola persona.
- **Costes directos de material:** contemplan licencias, equipos y amortización del hardware y software utilizado.
- **Costes indirectos:** coeficiente adicional del 20 % aplicado sobre el total de los costes directos, para cubrir gastos generales asociados (electricidad, internet, etc.).

Este desglose permite dimensionar correctamente los recursos implicados y simular una estructura empresarial.

3.5.1. Costes directos de personal

Se han obtenido los sueldos medios anuales en España para cada perfil profesional a partir del portal *Glassdoor* [55, 56, 57, 58, 59, 60, 61]. A cada salario bruto se le añade un 30 % adicional correspondiente a cotizaciones sociales: contingencias comunes, desempleo, formación y Fogasa.

Se asume una jornada laboral de 8 h/día, 20 días laborables por mes y 11 meses de trabajo al año, siendo 220 días laborales al año. La planificación se basa en 80 días efectivos de trabajo entre el **3 de marzo y el 20 de junio de 2025**, distribuidos entre los distintos perfiles según su implicación en cada fase.

Cuadro 3.2: Perfiles y costes de personal (según cotización SS 2025)

Perfil	Bruto/año (€)	+31.4 % SS (€)	Total (€)	€/día	€/h
Jefe de Proyecto	40 500	12 717.00	53 217.00	270.92	33.87
Desarrollador Frontend	40 500	12 717.00	53 217.00	270.92	33.87
Desarrollador Backend	30 000	9 420.00	39 420.00	200.10	25.01
Desarrollador RA	23 000	7 222.00	30 222.00	153.68	19.21
Tester / QA	23 500	7 379.00	30 879.00	156.71	19.59
Modelador 3D	12 000	3 768.00	15 768.00	79.84	9.98
Diseñador gráfico	26 000	8 164.00	34 164.00	173.32	21.67

3.5.2. Asignación temporal por perfil

Se ha asignado a cada perfil el número estimado de jornadas necesarias para cubrir las tareas planificadas según el cronograma de trabajo. A continuación, se presenta la estimación de coste directo de personal en función de esas jornadas.

Cuadro 3.3: Asignación de tareas y costes por perfil

Perfil	Días	€/día	Tareas	Coste (€)
Jefe de Proyecto	50	270.92	Estudio inicial, definición de objetivos, planificación, selección de tecnologías, coordinación general	13,546.00
Desarrollador Frontend	16	270.92	Diseño arquitectura frontend, desarrollo UI, navegación, QR, filtros, mapa	4,334.72
Desarrollador Backend	15	200.10	Creación API Django, modelo de datos, autenticación, sincronización backend	3,001.50
Desarrollador RA	7	153.68	Integración AR.js, activación por proximidad, carga de modelos RA	1,075.76
Tester / QA	7	156.71	Pruebas funcionales, pruebas con usuarios, validación y corrección de errores	1,096.97
Modelador 3D	15	79.84	Diseño y preparación de modelos 3D para RA	1,197.60
Diseñador gráfico	20	173.32	Diseño interfaz, estilo visual, iconos y logotipos	3,466.40

3.5.3. Costes directos de material

Los costes directos de material comprenden los recursos físicos y licencias digitales utilizados durante el desarrollo del proyecto. Para estimar su impacto económico real, se ha calculado la amortización proporcional al tiempo de uso, aplicando la fórmula:

$$\text{Amortización} = \frac{\text{Precio unidad} \times \text{Días de uso}}{\text{Días de vida útil}}$$

Cuadro 3.4: Elementos utilizados y coste amortizado durante el desarrollo (80 días)

Elemento	Precio (€)	Vida útil	Uso (días)	Coste amortizado (€)
Portátil Asus ROG Strix G15	999.00	5 años (1825 días)	80	43.79
iPhone 11	749.00	4 años (1460 días)	80	41.03
Ratón Logitech G203	20.99	5 años (1825 días)	80	0.92
Visual Paradigm (mensual)	30.00	30 días	30	30.00
Microsoft 365 Business Standard	12.50	30 días	30	12.50
Visual Studio Code (profesional)	45.00	365 días	80	9.86
Docker Pro	99.00	365 días	80	21.70
GitHub Copilot	100.00	365 días	80	21.92
Overleaf Premium	140.00	365 días	80	30.68
Total amortizado				212.40 €

Cabe destacar que varias de estas herramientas (como Visual Studio Code, Docker o GitHub) disponen de versiones gratuitas o de acceso académico, pero se ha optado por reflejar el coste real de uso profesional para una estimación más realista.

3.5.4. Costes indirectos

Los costes indirectos se han estimado aplicando un coeficiente adicional del 20 % sobre el total de los costes directos (personal y material). Este porcentaje se considera repre-

sentativo para cubrir gastos generales como el consumo eléctrico, conexión a internet, mantenimiento del equipo, uso de espacios compartidos y otros recursos no imputables directamente al desarrollo del proyecto, pero imprescindibles para su ejecución.

$$\text{Costes indirectos} = (\text{Costes directos de personal} + \text{Costes directos de material}) \times 0,20$$

$$\text{Costes indirectos} = (27\,718,95 + 212,40) \times 0,20 = 5\,586,27$$

3.5.5. Coste total del proyecto

A partir de la suma de los distintos componentes estimados (personal, material e indirectos), se obtiene el coste total simulado del desarrollo del proyecto en un entorno profesional.

Cuadro 3.5: Resumen de costes del proyecto

Concepto	Coste (€)
Costes directos de personal	27 718,95
Costes directos de material	212,40
Costes indirectos (20 %)	5 586,27
Coste total estimado	33 517,62

Esta estimación permite cuantificar de forma realista los recursos necesarios para ejecutar un proyecto de características similares en un entorno empresarial. Aunque el trabajo haya sido desarrollado por una única persona, se ha optado por realizar una simulación profesional con perfiles especializados para reflejar adecuadamente la complejidad técnica y el valor añadido del producto final.

3.6. Viabilidad del proyecto

3.6.1. Viabilidad económica

El análisis de costes realizado en la sección anterior ha permitido cuantificar los recursos necesarios para el desarrollo de la aplicación *DONA'm MÓN*. El coste total estimado, que incluye los costes directos de personal (27.718,95 €), costes directos de material (212,40 €) y un 20 % adicional en concepto de costes indirectos (5.586,27 €), asciende a **33.517,62 €**.

Dado que el proyecto se ha desarrollado sin fines comerciales, el desarrollo ha sido ejecutado por una única persona, utilizando licencias gratuitas, versiones académicas o software de código abierto. Por tanto, los costes no se han asumido en su totalidad. No obstante, se ha realizado una estimación económica completa con el objetivo de evaluar su viabilidad en un contexto profesional o empresarial real, permitiendo así dimensionar correctamente los recursos necesarios en caso de una futura implementación a escala.

El presupuesto puede considerarse razonable si se contempla una posible financiación pública (subvenciones a proyectos culturales, feministas o tecnológicos), financiación universitaria, patrocinio institucional (ayuntamientos, universidades o fundaciones), o apoyo

por parte de entidades interesadas en promover la visibilización de mujeres en entornos urbanos y educativos. Además, el impacto social y educativo del proyecto puede justificar su financiación mediante convocatorias de innovación social o género.

3.6.2. Viabilidad legal

Desde el punto de vista legal, el desarrollo de esta aplicación requiere considerar varias cuestiones relevantes:

- **Protección de datos personales (RGPD):** la aplicación permite el registro de usuarios y almacena información relacionada con su ubicación geográfica y su histórico de visitas. Por ello, se deberá garantizar el cumplimiento del Reglamento General de Protección de Datos (RGPD), asegurando la obtención del consentimiento explícito del usuario, el almacenamiento seguro de datos y la posibilidad de acceder, rectificar o eliminar la información personal.
- **Política de privacidad y condiciones de uso:** será necesario incluir una política clara y accesible que informe al usuario sobre el tratamiento de sus datos, el uso de la geolocalización y la finalidad educativa de la aplicación.
- **Derechos de autor y licencias:** todos los modelos 3D, imágenes, textos y recursos utilizados deben contar con licencia libre (Creative Commons, dominio público) o haber sido creados expresamente para el proyecto. En caso contrario, sería necesario solicitar los permisos adecuados.
- **Política de uso de servicios externos:** la aplicación hace uso de tecnologías como *WebView* para integrar experiencias de realidad aumentada desarrolladas con A-Frame y AR.js, por lo que se han respetado sus licencias de uso (ambas de código abierto bajo licencia MIT), garantizando la legalidad del uso de estas herramientas dentro del proyecto.

No se han identificado restricciones legales insalvables que impidan la realización o publicación de la aplicación. En todo caso, se recomienda aplicar buenas prácticas en el desarrollo seguro, accesible y ético del software.

3.7. Análisis de riesgos

Durante el desarrollo del proyecto *DONA'm MÓN* se han identificado diversos riesgos que pueden afectar negativamente a su cumplimiento en tiempo, coste o calidad. Para su evaluación se ha utilizado un enfoque cualitativo que combina la estimación de la probabilidad de ocurrencia y el impacto en el proyecto, permitiendo establecer niveles de riesgo y su clasificación prioritaria.

3.7.1. Principales riesgos identificados

En la Tabla 3.6 se recogen los principales riesgos detectados, su probabilidad estimada, el impacto medido en días de retraso potencial, su nivel de riesgo (NR = Probabilidad × Impacto), la clasificación resultante y la fase del proyecto en la que podrían manifestarse.

Cuadro 3.6: Principales riesgos identificados

Riesgo	Prob. (%)	Impacto (días)	NR	Clasificación	Fase
Retrasos en la integración de la RA con React Native	40	8	3.20	Inaceptable	Ejecución
Cambios en dependencias externas (Mapas, RA, etc.)	30	5	1.50	Alto	Ejecución
Falta de experiencia previa con tecnologías clave	35	6	2.10	Alto	Planificación/Ejecución
Definición ambigua del alcance inicial	25	7	1.75	Alto	Inicio
Estimaciones optimistas en tareas críticas	20	6	1.20	Moderado	Planificación
Problemas legales (datos, localización, RGPD)	15	5	0.75	Moderado	Inicio
Baja disponibilidad de recursos y tiempo personal	30	6	1.80	Alto	Ejecución
Errores funcionales en fases finales	10	5	0.50	Bajo	Cierre

A continuación se describen los riesgos con mayor nivel de prioridad:

1. **Definición ambigua del alcance del proyecto:** Un alcance mal delimitado puede derivar en objetivos poco claros y decisiones incorrectas en fases posteriores del desarrollo.

Probabilidad: Media — Impacto: Alto — Nivel de riesgo: Inaceptable

2. **Estimaciones temporales y de costes imprecisas:** Una planificación optimista puede provocar desviaciones significativas respecto al cronograma o sobrecostes imprevistos.

Probabilidad: Alta — Impacto: Alto — Nivel de riesgo: Inaceptable

3. **Falta de experiencia con tecnologías específicas:** La falta de familiaridad con herramientas clave como AR.js o A-Frame podría ralentizar la implementación de la funcionalidad de realidad aumentada.

Probabilidad: Media — Impacto: Alto — Nivel de riesgo: Inaceptable

4. **Retrasos en la ejecución de tareas clave:** La elevada carga de trabajo asumida por una sola persona incrementa la probabilidad de bloqueos o demoras.

Probabilidad: Alta — Impacto: Medio — Nivel de riesgo: Alto

5. **Dependencia de recursos externos:** El uso de librerías de terceros (como AR.js, Mapbox, A-Frame) puede verse afectado por cambios inesperados o falta de mantenimiento.

Probabilidad: Media — Impacto: Medio — Nivel de riesgo: Medio

6. **Problemas de compatibilidad entre componentes:** La integración entre tecnologías heterogéneas (React Native, Django, AR.js) podría causar errores o conflictos difíciles de prever.

Probabilidad: Baja — Impacto: Alto — Nivel de riesgo: Alto

- 7. Riesgos legales:** A pesar de que no se almacena información sensible, el tratamiento de datos de geolocalización y el uso de usuarios registrados exige el cumplimiento de la normativa vigente en materia de protección de datos (RGPD).

Probabilidad: Baja — Impacto: Medio — Nivel de riesgo: Medio

3.7.2. Matriz de riesgos

La figura 3.3 muestra la matriz de evaluación de riesgos utilizada, que relaciona la probabilidad de ocurrencia con la magnitud del impacto. Esta herramienta facilita la clasificación de los riesgos y la priorización de acciones preventivas.

		IMPACTO EN EL PROYECTO		
		BAJO	MEDIO	ALTO
PROBABILIDAD DE QUE OCURRA (%)	ALTA [70 - 100]	RIESGO MEDIO	RIESGO ALTO	RIESGO INACEPTABLE
	MEDIA [35 - 70]	RIESGO BAJO	RIESGO ALTO	RIESGO INACEPTABLE
	BAJA [10 - 35]	RIESGO BAJO	RIESGO MEDIO	RIESGO ALTO

Figura 3.3: Matriz de priorización de riesgos utilizada

3.7.3. Resumen de evaluación de riesgos

Cuadro 3.7: Resumen cualitativo de evaluación de riesgos

Riesgo	Probabilidad	Impacto	Clasificación
Definición ambigua del alcance	Media	Alto	Inaceptable
Estimaciones temporales imprecisas	Alta	Alto	Inaceptable
Falta de experiencia con RA	Media	Alto	Inaceptable
Retrasos en tareas clave	Alta	Medio	Alto
Dependencia de recursos externos	Media	Medio	Medio
Problemas de compatibilidad tecnológica	Baja	Alto	Alto
Riesgos legales (RGPD)	Baja	Medio	Medio

3.7.4. Estrategias de mitigación

Para minimizar el impacto de los riesgos identificados en el proyecto, se han definido una serie de medidas preventivas orientadas a reducir su probabilidad de ocurrencia o su

impacto en caso de materializarse. Estas estrategias se alinean con las buenas prácticas en la gestión de proyectos tecnológicos y tienen carácter proactivo.

- **Definición ambigua del alcance del proyecto:** Se ha trabajado desde el inicio con una delimitación clara del alcance y los objetivos, apoyada por una estructura modular de tareas y entregables. La revisión periódica del cronograma y del backlog funcional permite detectar desviaciones tempranas y corregir posibles ambigüedades.
- **Estimaciones temporales y de costes imprecisas:** La planificación se ha fundamentado en la técnica de estimación por tres valores (optimista, más probable y pesimista), reduciendo la subjetividad y permitiendo incorporar márgenes de seguridad en tareas críticas. Además, se han utilizado referencias de proyectos similares como guía de validación.
- **Falta de experiencia con tecnologías específicas:** Durante la fase de análisis técnico se han realizado pruebas de concepto con las tecnologías clave (AR.js, A-Frame, integración con React Native) para identificar dificultades de forma anticipada. Asimismo, se ha recurrido a documentación oficial, foros especializados y comunidades activas.
- **Retrasos en la ejecución de tareas clave:** Dado que el proyecto es desarrollado por una única persona, se han asignado holguras temporales en tareas críticas y puntos de control semanales para verificar el avance real. El uso de herramientas como Microsoft Project ha permitido detectar desviaciones y replanificar en tiempo real.
- **Dependencia de recursos externos:** Se ha priorizado el uso de herramientas open source con comunidades activas y mantenidas (por ejemplo, AR.js, OpenStreetMap). Además, se han definido posibles alternativas ante fallos o incompatibilidades en recursos externos.
- **Problemas de compatibilidad entre componentes:** La arquitectura del sistema se ha diseñado de forma desacoplada, favoreciendo la integración por medio de interfaces RESTful y estándares abiertos. Esto reduce la posibilidad de conflictos entre tecnologías heterogéneas.
- **Riesgos legales (RGPD):** Aunque no se almacenan datos sensibles, se garantiza el cumplimiento normativo mediante el uso de comunicaciones seguras (HTTPS), el consentimiento del usuario para acceder a la ubicación y la ausencia de almacenamiento persistente de información personal.

3.7.5. Planes de contingencia

Se han previsto planes de contingencia específicos para los riesgos más relevantes. Por ejemplo, si se detectan incompatibilidades graves entre tecnologías, se contemplaría el uso de una solución alternativa basada únicamente en mapas interactivos sin RA, manteniendo la funcionalidad básica de exploración de contenidos educativos geolocalizados.

Capítulo 4

Análisis

Introducción

En este capítulo se realiza el análisis del sistema desde el punto de vista de la Ingeniería del Software. El objetivo es especificar, de manera estructurada y rigurosa, cómo debe comportarse la aplicación y qué funcionalidades debe ofrecer a los distintos tipos de usuarios, siguiendo las metodologías y herramientas propias de la disciplina.

El análisis comienza con la identificación de los roles principales del sistema y las tareas asociadas a cada uno. En el caso de esta aplicación, los roles contemplados son: usuario registrado, usuario no registrado y administrador. Cada uno de ellos dispone de diferentes permisos y accesos a funcionalidades, siendo el administrador quien puede gestionar todos los contenidos y usuarios, mientras que los usuarios registrados acceden a la experiencia completa de la app.

A continuación, se presentan los diagramas de casos de uso, que permiten visualizar de forma global y por rol las interacciones posibles con el sistema. Posteriormente, se detallan los casos de uso más relevantes mediante tablas descriptivas, especificando actores, propósito, precondiciones, postcondiciones y flujos de eventos. Según la naturaleza de la aplicación, también se incluyen diagramas de actividad y de estados para ilustrar los procesos dinámicos y los posibles cambios de estado de los objetos principales.

4.1. Diagramas de casos de uso

A continuación se presentan los diagramas de casos de uso para los diferentes roles del sistema:

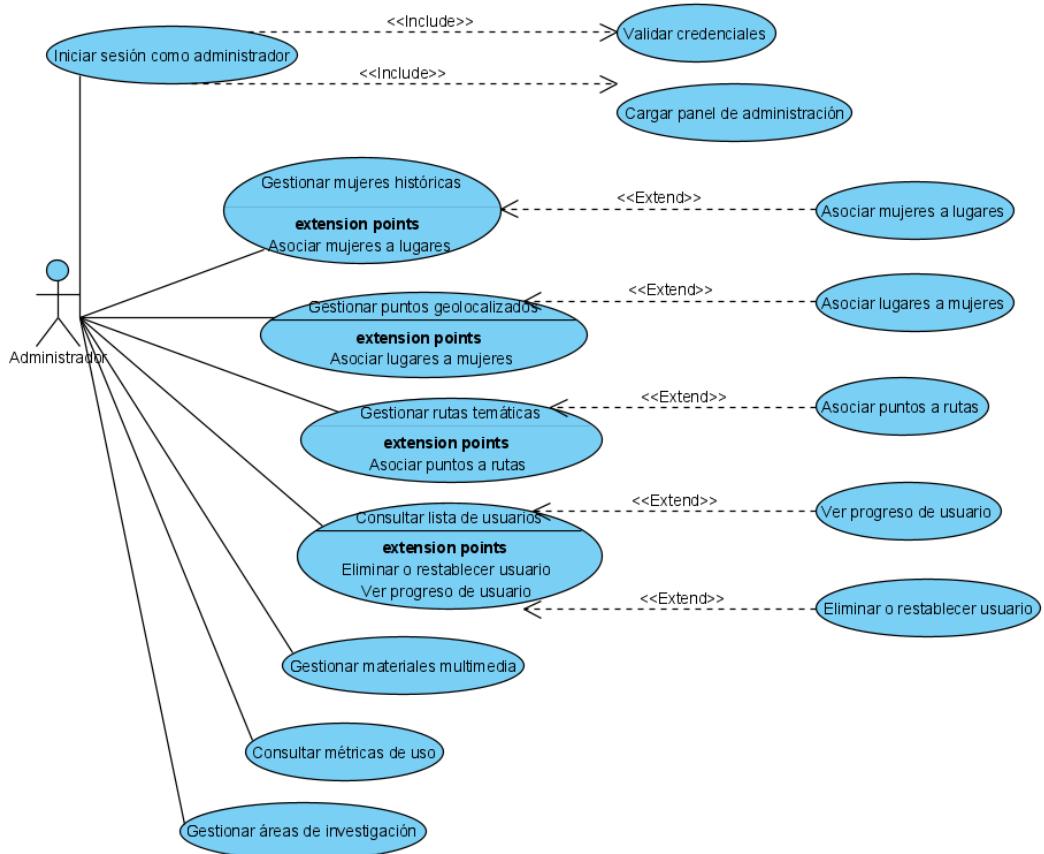


Figura 4.1: Diagrama de casos de uso: Administrador

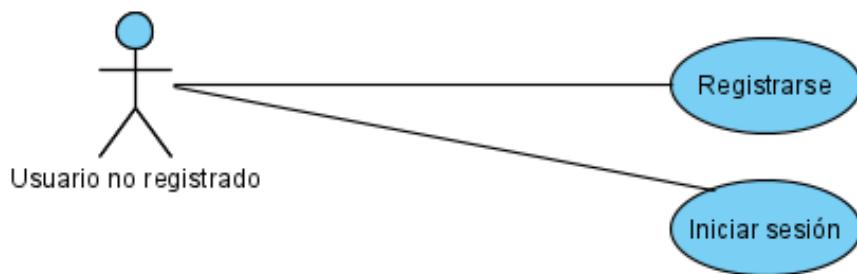


Figura 4.2: Diagrama de casos de uso: Usuario no registrado

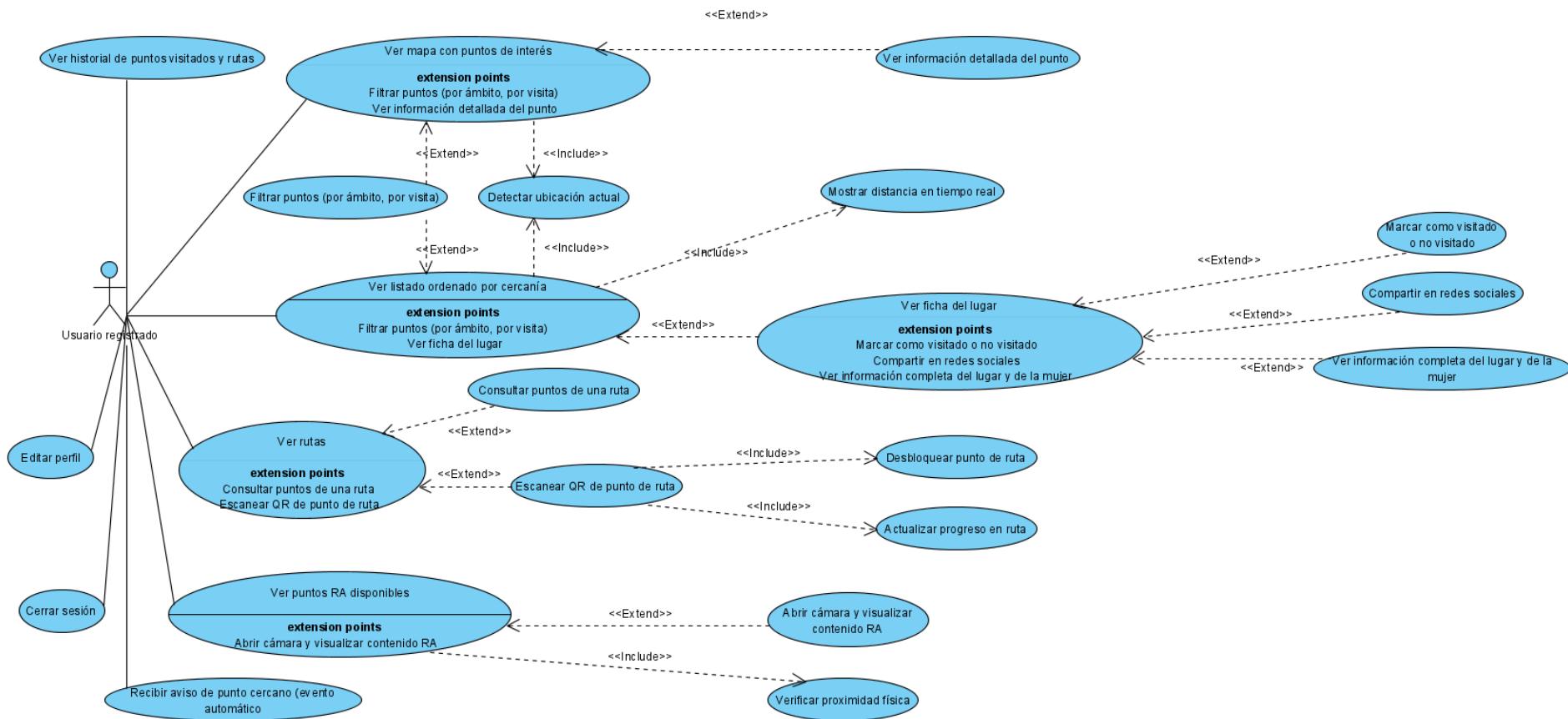


Figura 4.3: Diagrama de casos de uso: Usuario registrado

4.2. Especificación de casos de uso

A continuación se detallan los casos de uso principales de la aplicación:

4.2.1. Caso de uso 1: Login de usuario

Cuadro 4.1: Caso de uso 1 - Login de usuario

Nombre del caso de uso	Login de usuario
Actores	Usuario no registrado (principal), Sistema (secundario)
Propósito	Permitir que el usuario acceda a su cuenta personal mediante correo electrónico y contraseña.
Resumen	Este caso de uso describe el proceso de inicio de sesión mediante credenciales para acceder a las funcionalidades personalizadas de la app.
Tipo	Primario
Referencias	RF23, RF24, RF25
Precondiciones	El usuario debe tener una cuenta creada previamente. El sistema debe estar operativo y conectado a internet.
Postcondiciones	El usuario inicia sesión correctamente y accede a la pantalla principal de la aplicación, manteniendo la sesión iniciada hasta que decida cerrarla.
Flujo de eventos principal	
Actor	Sistema
1. El usuario abre la aplicación DONA'm MÓN.	
	2. El sistema muestra la pantalla de login con los campos de email y contraseña.
3. El usuario introduce su email y contraseña.	
	4. El sistema valida las credenciales en la base de datos.
	5. Si las credenciales son válidas, se genera un token y se accede a la pantalla principal.
	6. Si las credenciales no son válidas, se muestra un mensaje de error y se vuelve al paso 2.

4.2.2. Caso de uso 2: Registro de usuario

Cuadro 4.2: Caso de uso 2 - Registro de usuario

Nombre del caso de uso	Registro de usuario
Actores	Usuario no registrado (principal), Sistema (secundario)
Propósito	Permitir que nuevos usuarios creen una cuenta personal mediante un formulario con correo electrónico y contraseña.
Resumen	Este caso de uso describe el proceso por el cual un usuario puede registrarse en la aplicación, aceptando las condiciones de uso y política de privacidad.
Tipo	Primario
Referencias	RF23, RF24, RF25, RNF21, RNF22
Precondiciones	El sistema debe estar operativo y conectado. El usuario no debe tener una cuenta registrada con ese correo.
Postcondiciones	El usuario queda registrado y accede automáticamente a la pantalla principal con sesión iniciada.
Flujo de eventos principal	
Actor	Sistema
1. El usuario pulsa sobre Registrarse. ^{en} la pantalla de login.	
	2. El sistema muestra el formulario de registro con los campos de correo, contraseña y checkbox de aceptación legal.
3. El usuario introduce sus datos y acepta la política de privacidad.	
	4. El sistema verifica que el correo no exista previamente en la base de datos.
	5. El sistema guarda los datos cifrados y crea la nueva cuenta.
	6. El sistema genera una sesión automática y redirige a la pantalla principal.
	7. Si el correo ya existe o hay errores, se muestra un mensaje al usuario.

4.2.3. Caso de uso 3: Ver mapa con puntos de interés

Cuadro 4.3: Caso de uso 3 - Ver mapa con puntos de interés

Nombre del caso de uso	Ver mapa con puntos de interés
Actores	Usuario registrado (principal), Sistema (secundario)
Propósito	Permitir al usuario visualizar un mapa interactivo con los puntos geolocalizados disponibles en la ciudad de Valencia.
Resumen	Este caso de uso describe la carga del mapa, la detección de la ubicación actual, la visualización de los puntos y la posibilidad de aplicar filtros o acceder a la ficha de un punto.
Tipo	Primario
Referencias	RF1, RF2, RF3, RF7, RF8, RF31, RF32
Precondiciones	El usuario debe haber iniciado sesión y tener permisos de geolocalización habilitados.
Postcondiciones	El usuario ve los puntos en el mapa y puede filtrar, recibir notificaciones o acceder a la ficha de un punto.
Flujo de eventos principal	
Actor	Sistema
1. El usuario accede a la pestaña del mapa.	
	2. El sistema obtiene la ubicación actual mediante GPS.
	3. El sistema carga los puntos geolocalizados en el mapa.
2. El usuario activa un filtro por ámbito o estado de visita.	
	4. El sistema actualiza la visualización del mapa según los filtros.
3. El usuario pulsa sobre un punto del mapa.	
	5. El sistema abre la ficha detallada del punto seleccionado.
	6. Si el usuario se aproxima a un punto no visitado, se lanza una notificación.

4.2.4. Caso de uso 4: Ver listado de puntos ordenado por cercanía

Cuadro 4.4: Caso de uso 4 - Ver listado de puntos ordenado por cercanía

Nombre del caso de uso	Ver listado de puntos ordenado por cercanía
Actores	Usuario registrado (principal), Sistema (secundario)
Propósito	Permitir al usuario consultar todos los puntos disponibles organizados automáticamente por distancia desde su ubicación actual.
Resumen	Este caso de uso permite al usuario visualizar un listado de lugares asociados a mujeres históricas, ordenados por cercanía, con filtros por ámbito y estado, y acceso a la ficha de cada uno.
Tipo	Primario
Referencias	RF7, RF8, RF9, RF31, RF32, RF33, RF34, RF35
Precondiciones	El usuario debe tener sesión iniciada y el GPS activado.
Postcondiciones	El usuario ve una lista actualizada y filtrada de puntos, con acceso a sus fichas.
Flujo de eventos principal	
Actor	Sistema
1. El usuario accede a la pestaña de listado.	2. El sistema detecta la ubicación actual del usuario. 3. El sistema obtiene todos los puntos disponibles. 4. El sistema calcula y muestra la distancia en tiempo real a cada punto. 5. El sistema ordena el listado por cercanía ascendente.
2. El usuario aplica un filtro por ámbito o por estado de visita.	6. El sistema actualiza el listado aplicando los filtros seleccionados.
3. El usuario pulsa sobre un punto del listado.	7. El sistema abre la ficha detallada del punto.

4.2.5. Caso de uso 5: Ver ficha del lugar

Cuadro 4.5: Caso de uso 1.5 - Ver ficha del lugar

Nombre del caso de uso	Ver ficha del lugar
Actores	Usuario registrado (principal), Sistema (secundario)
Propósito	Permitir al usuario visualizar toda la información del lugar y de la figura femenina representada, incluyendo contenido educativo multimedia.
Resumen	Este caso de uso describe el acceso a la ficha completa de un punto, con texto, imágenes, biografía de la mujer, ubicación y la posibilidad de marcar como visitado o compartir por redes sociales.
Tipo	Primario
Referencias	RF4, RF5, RF12, RF13, RF14, RF36
Precondiciones	El usuario debe haber accedido desde el mapa o listado, seleccionando un punto.
Postcondiciones	El usuario ha consultado la información y puede haber marcado el punto como visitado o compartido el contenido.
Flujo de eventos principal	
Actor	Sistema
1. El usuario pulsa sobre un punto desde el mapa o el listado.	
	2. El sistema carga los datos completos del lugar y de la mujer histórica asociada.
	3. El sistema muestra el contenido educativo (texto, imágenes y/o audio).
2. El usuario pulsa “Marcar como visitado”.	
	4. El sistema actualiza el estado del punto como visitado y lo guarda en el historial.
3. El usuario pulsa “Compartir”.	
	5. El sistema lanza el menú para compartir el contenido en redes sociales.

4.2.6. Caso de uso 6: Ver rutas y consultar puntos de una ruta

Cuadro 4.6: Caso de uso 6 - Ver rutas y consultar puntos de una ruta

Nombre del caso de uso	Ver rutas y consultar puntos de una ruta
Actores	Usuario registrado (principal), Sistema (secundario)
Propósito	Permitir al usuario visualizar las rutas temáticas disponibles y consultar qué puntos forman parte de cada una.
Resumen	Este caso de uso describe cómo el usuario accede a la lista de rutas temáticas (por ejemplo, mujeres científicas), y cómo puede consultar los puntos que debe desbloquear escaneando su QR correspondiente.
Tipo	Primario
Referencias	RF11, RF37
Precondiciones	El usuario debe haber iniciado sesión.
Postcondiciones	El usuario ha accedido a la información de una ruta y conoce qué puntos contiene.
Flujo de eventos principal	
Actor	Sistema
1. El usuario accede a la pestaña de rutas.	
	2. El sistema muestra una lista de rutas temáticas disponibles, con su nombre y descripción.
2. El usuario pulsa sobre una de las rutas.	
	3. El sistema muestra los puntos geolocalizados que componen esa ruta.
	4. El sistema indica visualmente cuáles han sido desbloqueados y cuáles no.

4.2.7. Caso de uso 7: Escanear QR de punto de ruta

Cuadro 4.7: Caso de uso 7 - Escanear QR de punto de ruta

Nombre del caso de uso	Escanear QR de punto de ruta
Actores	Usuario registrado (principal), Sistema (secundario)
Propósito	Permitir al usuario desbloquear un punto dentro de una ruta temática mediante el escaneo de un código QR.
Resumen	Este caso de uso permite que el usuario avance en una ruta únicamente si ha escaneado correctamente el QR asociado a un punto. Una vez validado, se actualiza su progreso.
Tipo	Primario
Referencias	RF30, RF38, RF39
Precondiciones	El usuario debe haber iniciado sesión y tener la cámara habilitada. Debe haber accedido previamente a la ruta.
Postcondiciones	Si el código escaneado es válido, el punto se marca como desbloqueado en la ruta y se actualiza el historial del usuario.
Flujo de eventos principal	
Actor	Sistema
1. El usuario accede a una ruta y pulsa sobre un punto no desbloqueado.	
2. El usuario pulsa el botón "Escanear QR".	
	3. El sistema abre la cámara y espera a que el usuario enfoque el código QR.
3. El usuario escanea el QR.	
	4. El sistema valida que el QR corresponde al punto correcto.
	5. Si es válido, el sistema desbloquea el punto y lo marca como visitado.
	6. El sistema actualiza el progreso del usuario en esa ruta.
	7. Si el código QR no es válido, se muestra un mensaje de error.

4.2.8. Caso de uso 8: Ver puntos RA disponibles

Cuadro 4.8: Caso de uso 8 - Ver puntos RA disponibles

Nombre del caso de uso	Ver puntos RA disponibles
Actores	Usuario registrado (principal), Sistema (secundario)
Propósito	Mostrar al usuario solo los puntos que tienen contenido de realidad aumentada disponible y que están dentro del radio permitido.
Resumen	Este caso de uso describe cómo el sistema filtra automáticamente los puntos que disponen de RA y están dentro del rango de distancia del usuario para que puedan visualizarse.
Tipo	Primario
Referencias	RF40, RF41
Precondiciones	El usuario debe tener sesión iniciada, el GPS activado y conexión a internet.
Postcondiciones	El usuario visualiza un listado de puntos RA disponibles que puede seleccionar para abrir el contenido RA.
Flujo de eventos principal	
Actor	Sistema
1. El usuario accede a la pestaña de RA.	
	2. El sistema detecta la ubicación actual del usuario.
	3. El sistema filtra los puntos con contenido RA.
	4. El sistema calcula qué puntos RA están dentro del radio permitido.
	5. El sistema muestra el listado de puntos RA disponibles cercanos.

4.2.9. Caso de uso 9: Visualizar contenido RA

Cuadro 4.9: Caso de uso 9 - Visualizar contenido RA

Nombre del caso de uso	Visualizar contenido RA
Actores	Usuario registrado (principal), Sistema (secundario)
Propósito	Permitir al usuario visualizar los modelos 3D o elementos interactivos en RA sobre el entorno real desde la cámara del dispositivo.
Resumen	Este caso de uso describe el proceso de abrir la cámara desde un punto RA cercano y mostrar el contenido de realidad aumentada asociado, como un modelo 3D vinculado a una mujer histórica.
Tipo	Primario
Referencias	RF41
Precondiciones	El usuario debe haber accedido a la pestaña RA, seleccionando un punto válido y concedido permiso de cámara.
Postcondiciones	El usuario visualiza el contenido RA correctamente alineado sobre el marcador o la ubicación.
Flujo de eventos principal	
Actor	Sistema
1. El usuario pulsa sobre un punto RA disponible.	2. El sistema solicita permisos para acceder a la cámara. 3. El sistema abre la cámara del dispositivo. 4. El sistema detecta la ubicación o el marcador visual. 5. El sistema carga el modelo RA correspondiente (imagen, animación o 3D). 6. El sistema muestra el modelo RA en tiempo real superpuesto al entorno.

4.2.10. Caso de uso 10: Editar perfil de usuario

Cuadro 4.10: Caso de uso 10 - Editar datos del perfil

Nombre del caso de uso	Editar datos del perfil
Actores	Usuario registrado (principal), Sistema (secundario)
Propósito	Permitir al usuario modificar sus datos personales como nombre, correo o contraseña desde la sección de perfil.
Resumen	Este caso de uso describe cómo un usuario puede acceder a su perfil, modificar la información personal y guardar los cambios, los cuales se actualizan en la base de datos.
Tipo	Primario
Referencias	RF26, RF42
Precondiciones	El usuario debe tener sesión iniciada. Debe estar en la sección de perfil.
Postcondiciones	Los datos del perfil del usuario quedan actualizados correctamente en el sistema.
Flujo de eventos principal	
Actor	Sistema
1. El usuario accede a la pestaña de perfil.	
2. El usuario pulsa el botón "Editar perfil".	
3. El usuario modifica los campos deseados.	
4. El usuario pulsa el botón "Guardar cambios".	
	5. El sistema valida los nuevos datos.
	6. El sistema actualiza los datos del usuario en la base de datos.
	7. El sistema muestra un mensaje de confirmación.

4.2.11. Caso de uso 11: Ver historial de lugares visitados y rutas completadas

Cuadro 4.11: Caso de uso 11 - Ver historial de lugares visitados y rutas completadas

Nombre del caso de uso	Ver historial de lugares visitados y rutas completadas
Actores	Usuario registrado (principal), Sistema (secundario)
Propósito	Permitir al usuario consultar qué puntos ha visitado y qué rutas ha completado, con su respectiva fecha y hora.
Resumen	Este caso de uso permite al usuario ver un resumen personalizado de su progreso en la app, incluyendo puntos visitados, rutas completadas y fechas correspondientes.
Tipo	Primario
Referencias	RF14, RF42, F43, F44
Precondiciones	El usuario debe tener sesión iniciada.
Postcondiciones	El usuario visualiza su historial actualizado, sin modificar ningún dato.
Flujo de eventos principal	
Actor	Sistema
1. El usuario accede a la pestaña de perfil.	
2. El usuario pulsa sobre “Historial de visitas y rutas”.	
	3. El sistema recupera los datos del historial del usuario desde la base de datos.
	4. El sistema muestra la lista de puntos visitados con fecha y hora.
	5. El sistema muestra también las rutas completadas por el usuario.

4.2.12. Caso de uso 12: Cerrar sesión

Cuadro 4.12: Caso de uso 12 - Cerrar sesión

Nombre del caso de uso	Cerrar sesión
Actores	Usuario registrado (principal), Sistema (secundario)
Propósito	Permitir al usuario finalizar su sesión actual y cerrar su acceso a la app hasta el próximo inicio de sesión.
Resumen	Este caso de uso describe cómo el usuario puede cerrar manualmente su sesión desde la sección de perfil, eliminando su token de acceso y volviendo a la pantalla de login.
Tipo	Primario
Referencias	RF25, RF42
Precondiciones	El usuario debe tener sesión iniciada.
Postcondiciones	La sesión se cierra y el usuario vuelve a la pantalla de login.
Flujo de eventos principal	
Actor	Sistema
1. El usuario accede a la pestaña de perfil.	
2. El usuario pulsa el botón “Cerrar sesión”.	
	3. El sistema elimina el token de autenticación almacenado.
	4. El sistema redirige al usuario a la pantalla de login.

4.2.13. Caso de uso 13: Iniciar sesión como administrador

Cuadro 4.13: Caso de uso 13 - Iniciar sesión como administrador

Nombre del caso de uso	Iniciar sesión como administrador
Actores	Administrador (principal), Sistema (secundario)
Propósito	Permitir al administrador acceder al panel de administración web mediante autenticación segura.
Resumen	El administrador introduce sus credenciales (usuario y contraseña) en la pantalla de login del panel de administración. El sistema valida los datos y, si son correctos, permite el acceso a las funcionalidades administrativas.
Tipo	Primario
Referencias	RF16, F20, RNF19
Precondiciones	El administrador debe estar registrado en el sistema y disponer de credenciales válidas.
Postcondiciones	El administrador accede al panel de administración con los permisos correspondientes.
Flujo de eventos principal	
Actor	Sistema
El administrador accede a la URL del panel de administración.	
El administrador introduce su usuario y contraseña.	
El administrador pulsa el botón de “Iniciar sesión”.	
	4. El sistema valida las credenciales introducidas.
	5. Si las credenciales son correctas, el sistema carga el panel de administración.
	6. Si las credenciales son incorrectas, el sistema muestra un mensaje de error.

4.2.14. Caso de uso 14: Gestionar mujeres históricas

Cuadro 4.14: Caso de uso 14 - Gestionar mujeres históricas

Nombre del caso de uso	Gestionar mujeres históricas
Actores	Administrador (principal), Sistema (secundario)
Propósito	Permitir al administrador crear, editar o eliminar figuras históricas femeninas dentro del sistema.
Resumen	Este caso de uso describe las acciones que puede realizar el administrador para mantener actualizada la base de datos de mujeres históricas: crear nuevas entradas, modificar datos existentes, o eliminarlas. Cada mujer puede estar asociada a uno o varios lugares geolocalizados.
Tipo	Primario
Referencias	RF17, RF18, F21, F22, F24
Precondiciones	El administrador debe haber iniciado sesión correctamente en el panel de administración.
Postcondiciones	Se actualiza la base de datos de figuras históricas femeninas según las acciones realizadas.
Flujo de eventos principal	
Actor	Sistema
1. El administrador accede a la sección “Mujeres históricas” del panel.	
2. El administrador pulsa sobre “Añadir mujer”, “Editar” o “Eliminar”.	
3. El administrador introduce o modifica datos como nombre, biografía, fechas, y áreas de investigación.	
	4. El sistema valida los campos requeridos.
	5. El sistema guarda los datos nuevos o actualizados en la base de datos.
	6. Si se ha eliminado una figura, se borran también sus asociaciones con puntos.
	7. El sistema muestra un mensaje de confirmación de los cambios.

4.2.15. Caso de uso 15: Gestionar puntos geolocalizados

Cuadro 4.15: Caso de uso 15 - Gestionar puntos geolocalizados

Nombre del caso de uso	Gestionar puntos geolocalizados
Actores	Administrador (principal), Sistema (secundario)
Propósito	Permitir al administrador crear, editar o eliminar lugares geolocalizados y asociarlos a mujeres históricas.
Resumen	El administrador puede mantener actualizada la base de datos de lugares: crear nuevos puntos, modificar su información (nombre, descripción, ubicación, foto), o eliminarlos. Cada lugar puede estar vinculado a una mujer histórica.
Tipo	Primario
Referencias	RF17, RF18, F22, F24
Precondiciones	El administrador debe haber iniciado sesión correctamente en el panel de administración.
Postcondiciones	Se actualiza la base de datos de lugares y sus asociaciones con mujeres históricas.
Flujo de eventos principal	
Actor	Sistema
1. El administrador accede a la sección “Lugares” del panel.	
2. El administrador pulsa sobre “Añadir lugar”, “Editar” o “Eliminar”.	
3. El administrador introduce o modifica datos como nombre, descripción, ubicación, foto y mujer asociada.	
	4. El sistema valida los campos requeridos.
	5. El sistema guarda los datos nuevos o actualizados en la base de datos.
	6. Si se elimina un lugar, se eliminan también sus asociaciones.
	7. El sistema muestra un mensaje de confirmación de los cambios.

4.2.16. Caso de uso 16: Gestionar rutas temáticas

Cuadro 4.16: Caso de uso 16 - Gestionar rutas temáticas

Nombre del caso de uso	Gestionar rutas temáticas
Actores	Administrador (principal), Sistema (secundario)
Propósito	Permitir al administrador crear, editar o eliminar rutas temáticas y asociar puntos a cada ruta.
Resumen	El administrador puede definir rutas temáticas, asignarles nombre y descripción, y asociarles puntos geolocalizados. Puede modificar o eliminar rutas existentes.
Tipo	Primario
Referencias	RF11, RF18b, F24b, F33, F34
Precondiciones	El administrador debe haber iniciado sesión correctamente en el panel de administración.
Postcondiciones	Se actualiza la base de datos de rutas temáticas y sus asociaciones con puntos.
Flujo de eventos principal	
Actor	Sistema
1. El administrador accede a la sección “Rutas temáticas” del panel.	
2. El administrador pulsa sobre “Añadir ruta”, “Editar” o “Eliminar”.	
3. El administrador introduce o modifica datos como nombre, descripción y puntos asociados.	
	4. El sistema valida los campos requeridos.
	5. El sistema guarda los datos nuevos o actualizados en la base de datos.
	6. Si se elimina una ruta, se eliminan también sus asociaciones.
	7. El sistema muestra un mensaje de confirmación de los cambios.

4.2.17. Caso de uso 17: Consultar lista de usuarios

Cuadro 4.17: Caso de uso 17 - Consultar lista de usuarios

Nombre del caso de uso	Consultar lista de usuarios
Actores	Administrador (principal), Sistema (secundario)
Propósito	Permitir al administrador visualizar la lista de usuarios registrados, su progreso y gestionar sus cuentas.
Resumen	El administrador puede ver todos los usuarios registrados, consultar su progreso (puntos visitados, logros), y eliminar o restablecer cuentas si es necesario.
Tipo	Primario
Referencias	RF27, RF28, RF29, F25, F26, F27
Precondiciones	El administrador debe haber iniciado sesión correctamente en el panel de administración.
Postcondiciones	Se actualiza la base de datos de usuarios si se elimina o restablece alguna cuenta.
Flujo de eventos principal	
Actor	Sistema
1. El administrador accede a la sección “Usuarios” del panel.	
2. El administrador selecciona un usuario para ver su perfil y progreso.	
3. El administrador puede eliminar o restablecer la cuenta del usuario.	
	4. El sistema muestra la información del usuario y su progreso.
	5. Si se elimina o restablece, el sistema actualiza la base de datos y muestra confirmación.

4.2.18. Caso de uso 18: Gestionar materiales multimedia

Cuadro 4.18: Caso de uso 18 - Gestionar materiales multimedia

Nombre del caso de uso	Gestionar materiales multimedia
Actores	Administrador (principal), Sistema (secundario)
Propósito	Permitir al administrador importar, editar o eliminar imágenes, vídeos y modelos 3D asociados a mujeres o lugares.
Resumen	El administrador puede subir, modificar o eliminar archivos multimedia que se mostrarán en la app, asegurando que el contenido esté actualizado y sea relevante.
Tipo	Primario
Referencias	RF18, F23, F24
Precondiciones	El administrador debe haber iniciado sesión correctamente en el panel de administración.
Postcondiciones	Se actualiza la base de datos y el repositorio de archivos multimedia.
Flujo de eventos principal	
Actor	Sistema
1. El administrador accede a la sección de gestión multimedia.	
2. El administrador selecciona “Añadir”, “Editar” o “Eliminar” un archivo multimedia.	
3. El administrador sube o modifica el archivo y lo asocia a una mujer o lugar.	
	4. El sistema valida el formato y tamaño del archivo.
	5. El sistema guarda o elimina el archivo y actualiza la base de datos.
	6. El sistema muestra un mensaje de confirmación de los cambios.

4.2.19. Caso de uso 19: Consultar métricas de uso

Cuadro 4.19: Caso de uso 19 - Consultar métricas de uso

Nombre del caso de uso	Consultar métricas de uso
Actores	Administrador (principal), Sistema (secundario)
Propósito	Permitir al administrador visualizar estadísticas y métricas sobre el uso de la aplicación.
Resumen	El administrador puede consultar datos como puntos más visitados, número de usuarios activos, rutas más populares, etc., para tomar decisiones informadas sobre el contenido y la gestión.
Tipo	Primario
Referencias	RF18c, F24c, F28
Precondiciones	El administrador debe haber iniciado sesión correctamente en el panel de administración.
Postcondiciones	No aplica (consulta de información).
Flujo de eventos principal	
Actor	Sistema
1. El administrador accede a la sección de métricas del panel.	
2. El administrador selecciona el tipo de métrica o informe a consultar.	
	3. El sistema muestra los datos solicitados en pantalla.

4.2.20. Caso de uso 20: Gestionar áreas de investigación

Cuadro 4.20: Caso de uso 20 - Gestionar áreas de investigación

Nombre del caso de uso	Gestionar áreas de investigación
Actores	Administrador (principal), Sistema (secundario)
Propósito	Permitir al administrador crear, editar o eliminar áreas de investigación asociadas a mujeres históricas.
Resumen	El administrador puede mantener actualizada la lista de áreas de investigación, que luego pueden asociarse a las mujeres históricas para facilitar la búsqueda y filtrado en la app.
Tipo	Primario
Referencias	RF18b, F24b
Precondiciones	El administrador debe haber iniciado sesión correctamente en el panel de administración.
Postcondiciones	Se actualiza la base de datos de áreas de investigación.
Flujo de eventos principal	
Actor	Sistema
1. El administrador accede a la sección de áreas de investigación.	
2. El administrador pulsa sobre “Añadir”, “Editar” o “Eliminar” área.	
3. El administrador introduce o modifica el nombre del área.	
	4. El sistema valida el campo requerido.
	5. El sistema guarda o elimina el área en la base de datos.
	6. El sistema muestra un mensaje de confirmación de los cambios.

4.3. Diagramas de Actividad

Los diagramas de actividad son una herramienta utilizada en el modelado de procesos y flujos de trabajo dentro de un sistema. Permiten representar gráficamente la secuencia de actividades, decisiones y flujos alternativos que pueden ocurrir durante la ejecución de un proceso. Son especialmente útiles para visualizar el comportamiento dinámico de un sistema y entender cómo interactúan los distintos componentes o usuarios.

El diagrama de la Figura 4.4 muestra, de forma paralela, todas las funcionalidades principales a las que puede acceder un usuario tras iniciar sesión en la aplicación. Cada partición corresponde a una sección clave del menú (mapa interactivo, listado de lugares, rutas temáticas, realidad aumentada y perfil personal) o a las notificaciones de proximidad. En cada bloque se detallan las acciones posibles y las decisiones que puede tomar el usuario.

Este diagrama de actividad (véase la Figura 4.5) corresponde al panel de administración. Ilustra los principales procesos y decisiones que puede realizar un administrador tras iniciar sesión en la plataforma web. El flujo se divide en varias particiones, cada una dedicada a una funcionalidad específica: gestión de mujeres históricas, gestión de lugares geolocalizados, gestión de usuarios, gestión de visitas e historial, y gestión de rutas temáticas. En cada sección se muestran las operaciones CRUD (crear, leer, actualizar, eliminar) correspondientes, así como acciones adicionales como el registro de nuevos usuarios, la consulta y eliminación de históricas de visitas y rutas, y el reinicio de visitas en rutas para usuarios concretos.

En este diagrama se observan los primeros pasos que sigue un usuario al acceder a la aplicación, pasando por las opciones de registro o inicio de sesión, la validación de datos para llegar a la visualización de la aplicación (véase la Figura 4.3).

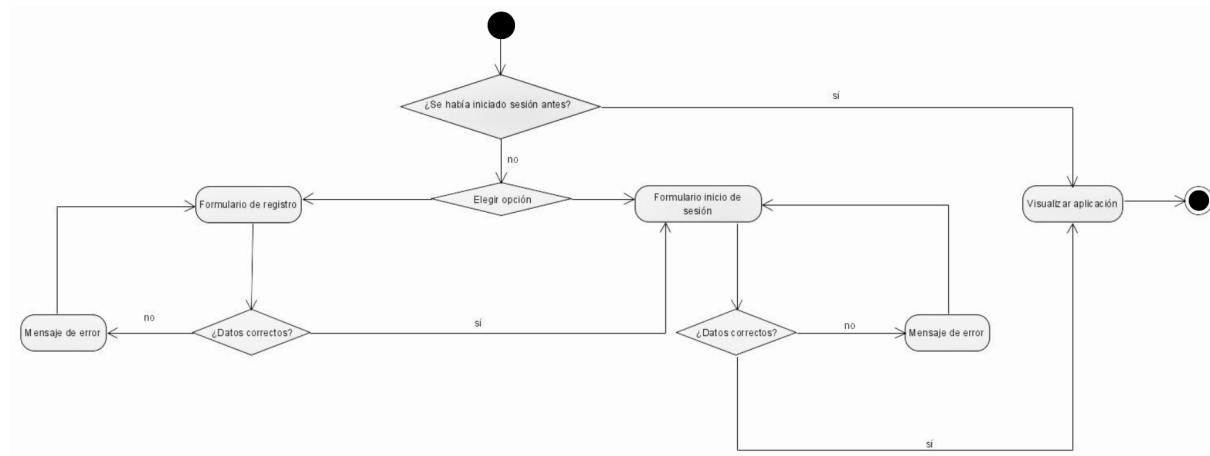


Figura 4.3: Diagrama de actividad de usuario no registrado

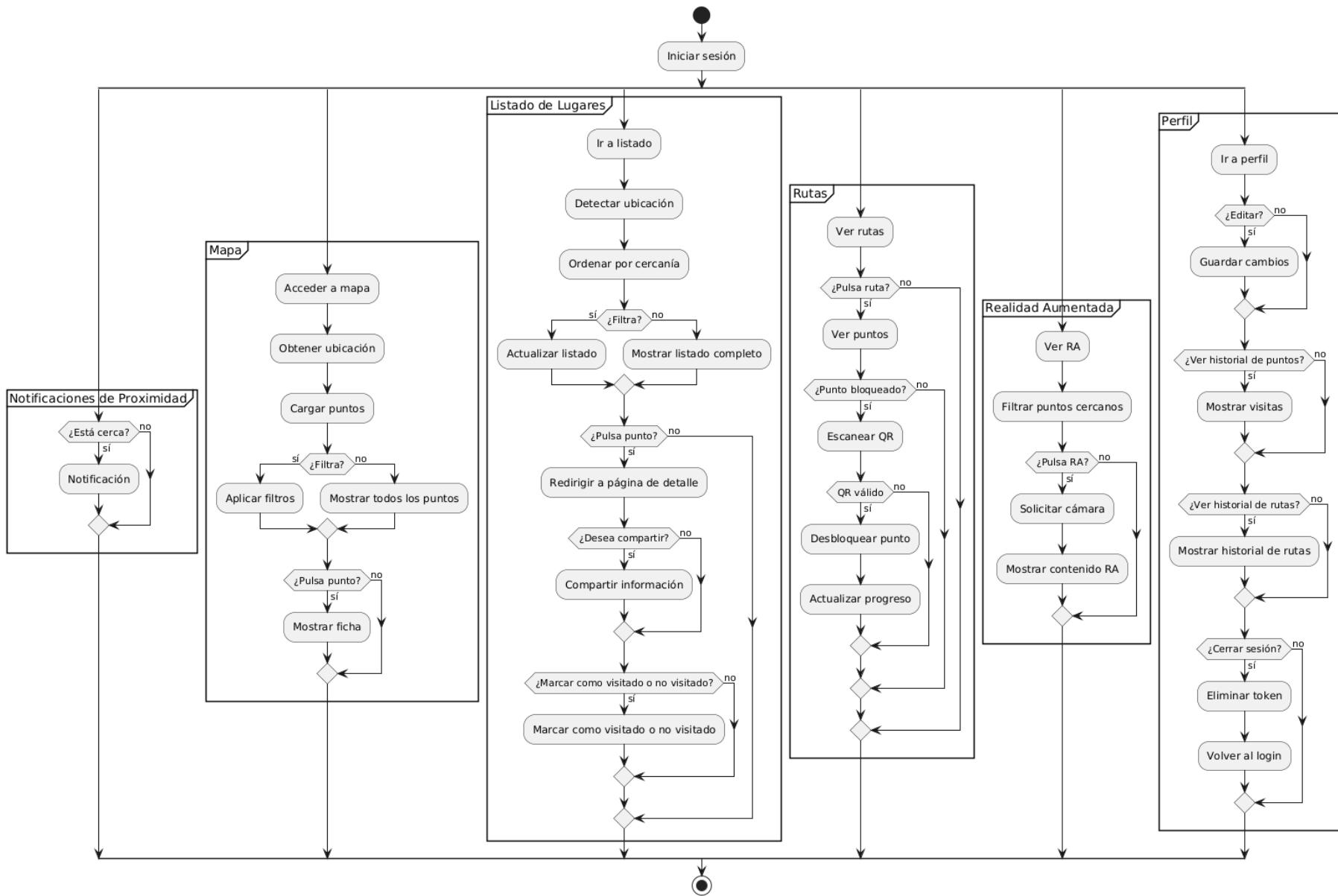


Figura 4.4: Diagrama de actividad de usuario registrado

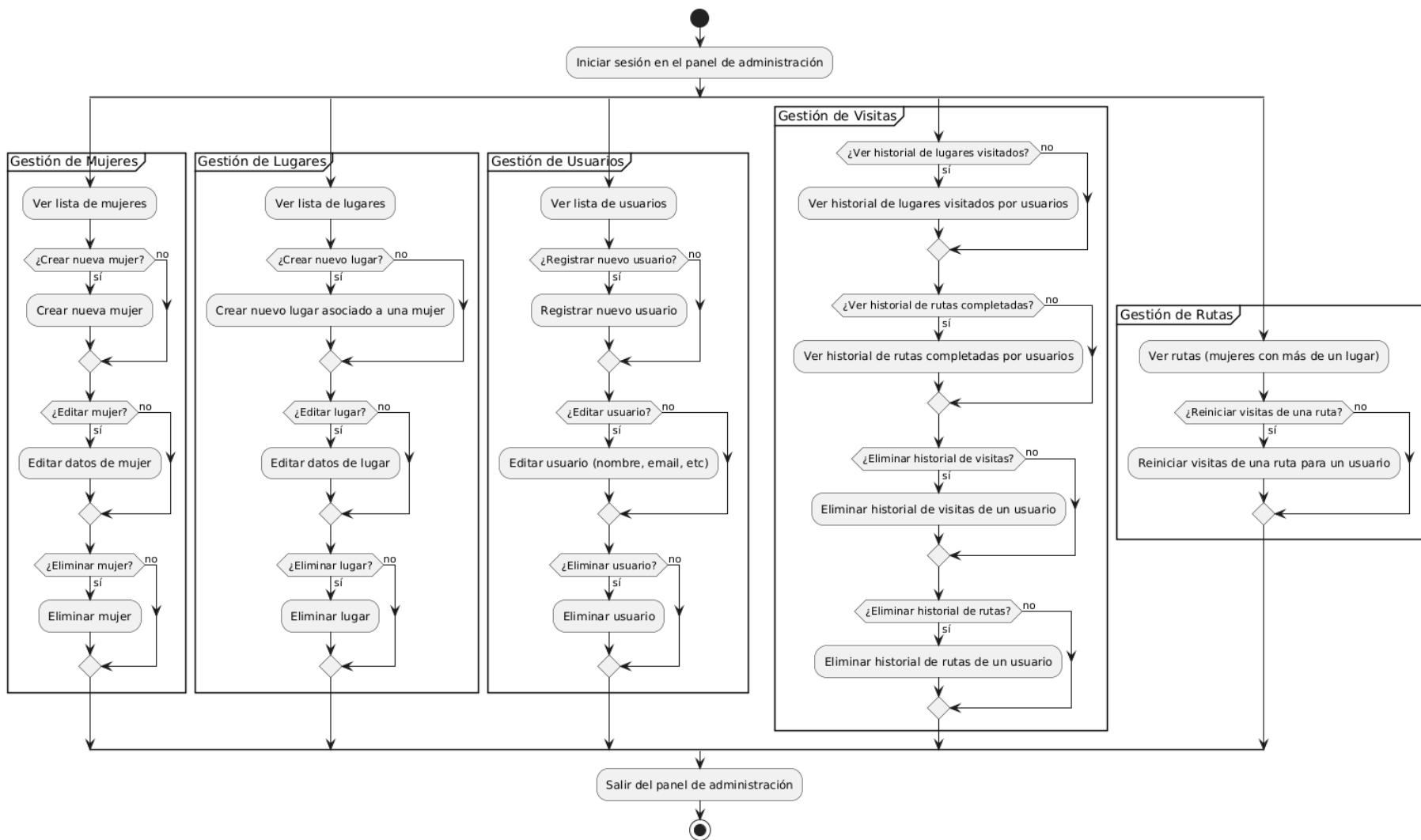


Figura 4.5: Diagrama de actividad del panel de administración

4.4. Diagramas de Estado

Los diagramas de estado son una herramienta fundamental en el modelado de sistemas software, especialmente útil para describir el comportamiento dinámico de un objeto o componente a lo largo de su ciclo de vida. Un diagrama de estado representa los diferentes estados que puede adoptar un elemento del sistema y las transiciones entre ellos, que se producen como respuesta a eventos o acciones. Estos diagramas permiten visualizar de manera clara cómo reacciona el sistema ante distintas situaciones, facilitando la comprensión y el diseño de su lógica interna.

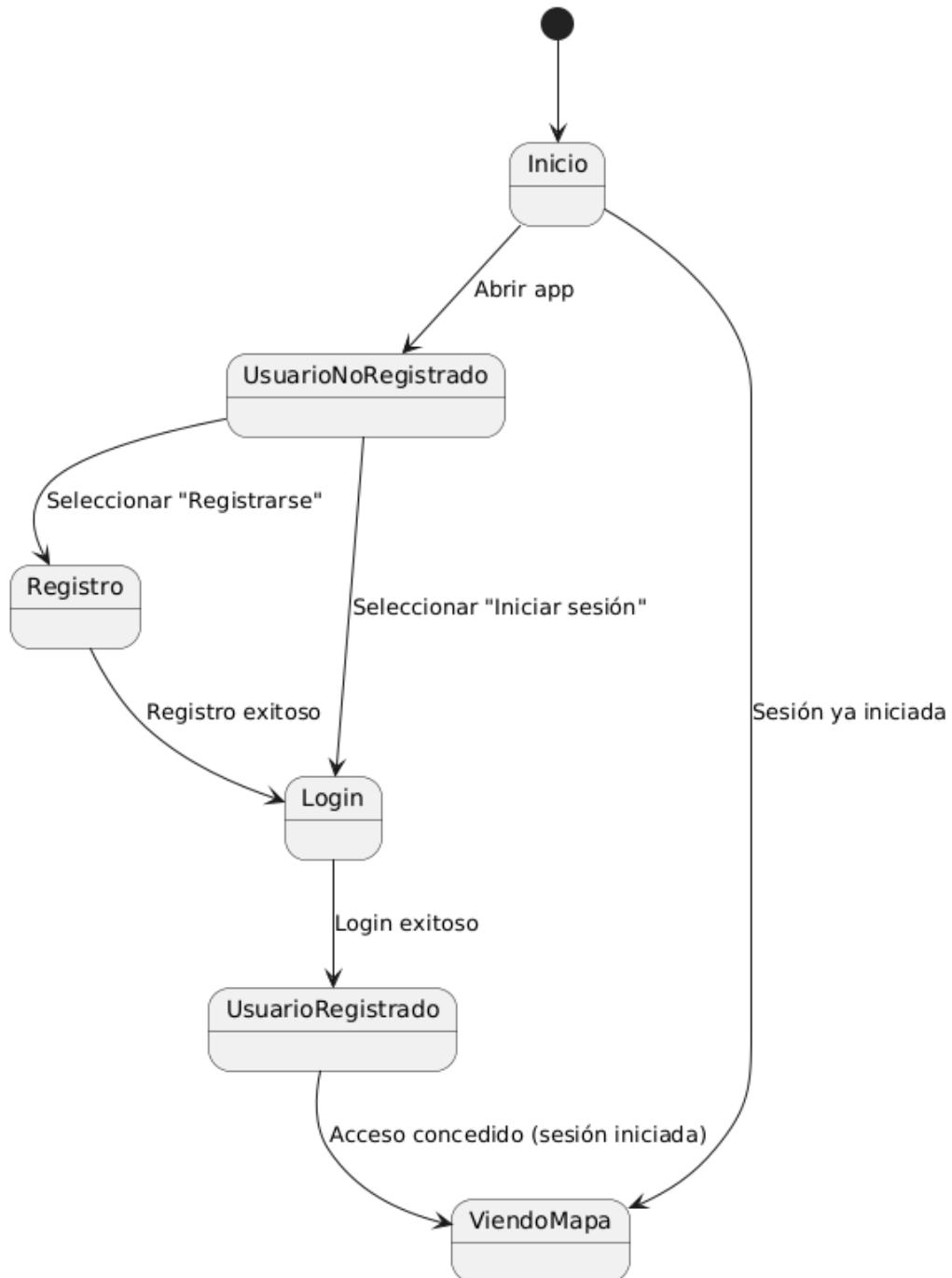


Figura 4.6: Diagrama de estado de usuario no registrado

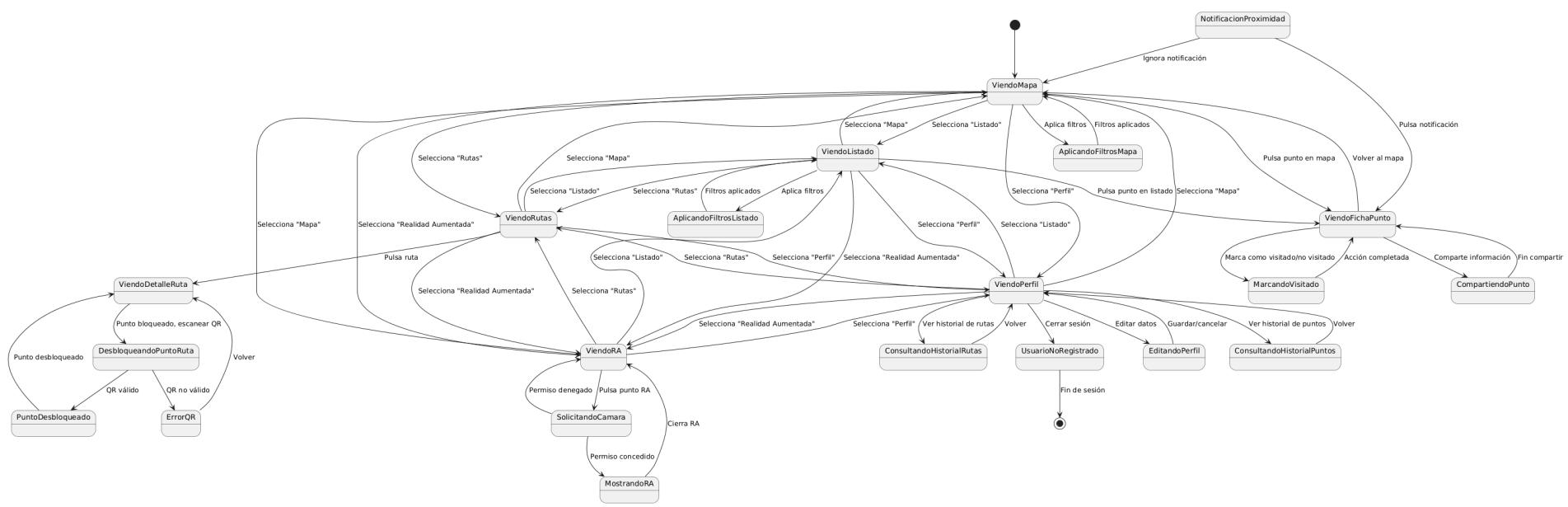


Figura 4.9: Diagrama de estado de usuario registrado

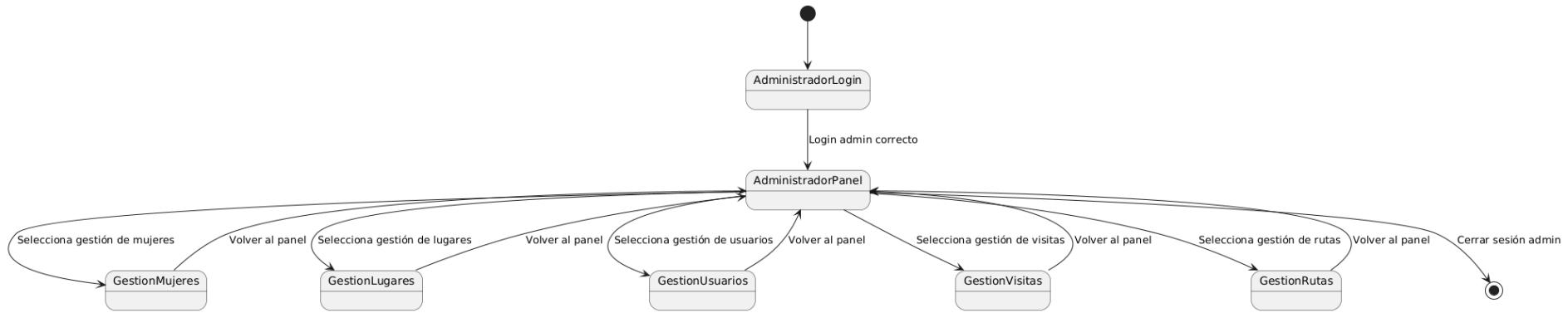


Figura 4.10: Diagrama de estado del panel de administración

Capítulo 5

Diseño

5.1. Diseño de base de datos

La base de datos de la aplicación está diseñada para almacenar información sobre mujeres, sus lugares asociados, rutas temáticas, usuarios y el historial de visitas. El diseño sigue una estructura relacional y está implementado en **PostgreSQL** mediante modelos definidos en Django.

5.1.1. Elección de la base de datos y justificación

Se ha seleccionado **PostgreSQL** como sistema gestor de base de datos por su robustez, soporte avanzado de tipos de datos, integridad referencial y compatibilidad nativa con Django. PostgreSQL permite almacenar listas de valores en un solo campo, lo que resulta útil para atributos como las áreas de investigación asociadas a cada mujer, sin perder la capacidad de realizar consultas eficientes.

5.1.2. Tablas principales y justificación de diseño

A continuación se describen las tablas principales que conforman el modelo de datos, junto con la justificación de las decisiones tomadas:

- **AreaInvestigacion:** Permite mantener un catálogo normalizado de áreas de investigación, facilitando la gestión y evitando duplicidades.
 - **id** (clave primaria)
 - **nombre:** nombre único del área.
- **Mujer:** Representa a una mujer relevante. El campo `areas_investigacion` se implementa como un `ArrayField` de cadenas, aprovechando las capacidades de PostgreSQL para almacenar múltiples áreas directamente en el registro, simplificando la consulta y el filtrado desde la API.
 - **id** (clave primaria)
 - **nombre**
 - **descripcion**
 - **foto**

- **areas_investigacion:** lista de cadenas de texto (`ArrayField`) con los nombres de las áreas de investigación asociadas.
 - **fechas:** información de fechas relevantes.
- **Lugar:** Almacena información geolocalizada y la relación con una mujer. Se justifica la relación uno-a-muchos para poder asociar varios lugares a una misma mujer.
- **id** (clave primaria)
 - **nombre**
 - **descripcion**
 - **latitud**
 - **longitud**
 - **foto**
 - **ar_url:** enlace a contenido de realidad aumentada.
 - **mujer_id** (clave foránea): referencia a la mujer destacada.
- **Ruta:** Agrupa lugares y mujeres en recorridos temáticos. Se emplean relaciones ManyToMany para maximizar la flexibilidad y permitir que una ruta incluya múltiples mujeres y lugares, y viceversa.
- **id** (clave primaria)
 - **nombre**
 - **descripcion**
 - Relación de tipo ManyToMany con las tablas **Mujer** y **Lugar**.
- **UserProfile:** Extiende el modelo de usuario de Django para almacenar información adicional relevante para la aplicación, como la fecha de nacimiento.
- **id** (clave primaria)
 - **user_id** (relación uno a uno con User)
 - **birth_date**
 - **email**
- **VisitedLugar:** Permite registrar el historial de visitas de los usuarios a lugares concretos, facilitando la personalización y la obtención de estadísticas.
- **id** (clave primaria)
 - **user_id** (clave foránea)
 - **lugar_id** (clave foránea)
 - **visited_at:** fecha y hora de la visita.
- **VisitedLugarRuta:** Permite registrar visitas de usuarios a lugares dentro de rutas específicas, lo que posibilita analizar el recorrido de los usuarios y mejorar la experiencia de gamificación.
- **id** (clave primaria)
 - **user_id** (clave foránea)

- **ruta_id** (clave foránea)
- **lugar_id** (clave foránea)
- **visited_at**: fecha y hora de la visita.

5.1.3. Relaciones entre tablas

- Un **Lugar** pertenece a una **Mujer** (relación uno a muchos).
- Una **Mujer** puede tener varios **Lugares** asociados.
- Una **Ruta** puede incluir varias **Mujeres** y varios **Lugares** (relaciones ManyToMany).
- Un **UserProfile** extiende a un **User** de Django (relación uno a uno).
- **VisitedLugar** y **VisitedLugarRuta** permiten registrar el historial de visitas de los usuarios, tanto de lugares individuales como de lugares dentro de rutas.

5.1.4. Esquema de tablas

El siguiente esquema resume la estructura de las tablas principales de la base de datos. Para mejorar la visualización y evitar cortes de línea, se utiliza el entorno `tabularx`:

Tabla	Campos principales
Mujer	id, nombre, descripcion, foto, areas_investigacion (ArrayField), fechas
Lugar	id, nombre, descripcion, latitud, longitud, foto, ar_url, mujer_id (FK)
Ruta	id, nombre, descripcion, mujeres (M2M), lugares (M2M)
UserProfile	id, user_id (O2O), birth_date, email
VisitedLugar	id, user_id (FK), lugar_id (FK), visited_at
VisitedLugarRuta	id, user_id (FK), ruta_id (FK), lugar_id (FK), visited_at
AreaInvestigacion	id, nombre

Cuadro 5.1: Esquema resumido de las tablas principales de la base de datos

5.1.5. Justificación global del diseño

El modelo de datos relacional implementado garantiza la integridad y la eficiencia en las consultas. El uso de `ArrayField` en **Mujer** permite una mayor flexibilidad y rendimiento en la consulta de áreas de investigación, sin necesidad de realizar joins adicionales, lo que es especialmente útil para filtros rápidos en la API. Las tablas de historial de visitas están separadas para facilitar la obtención de estadísticas, la personalización de la experiencia del usuario y la escalabilidad futura (por ejemplo, para añadir logros o recomendaciones personalizadas).

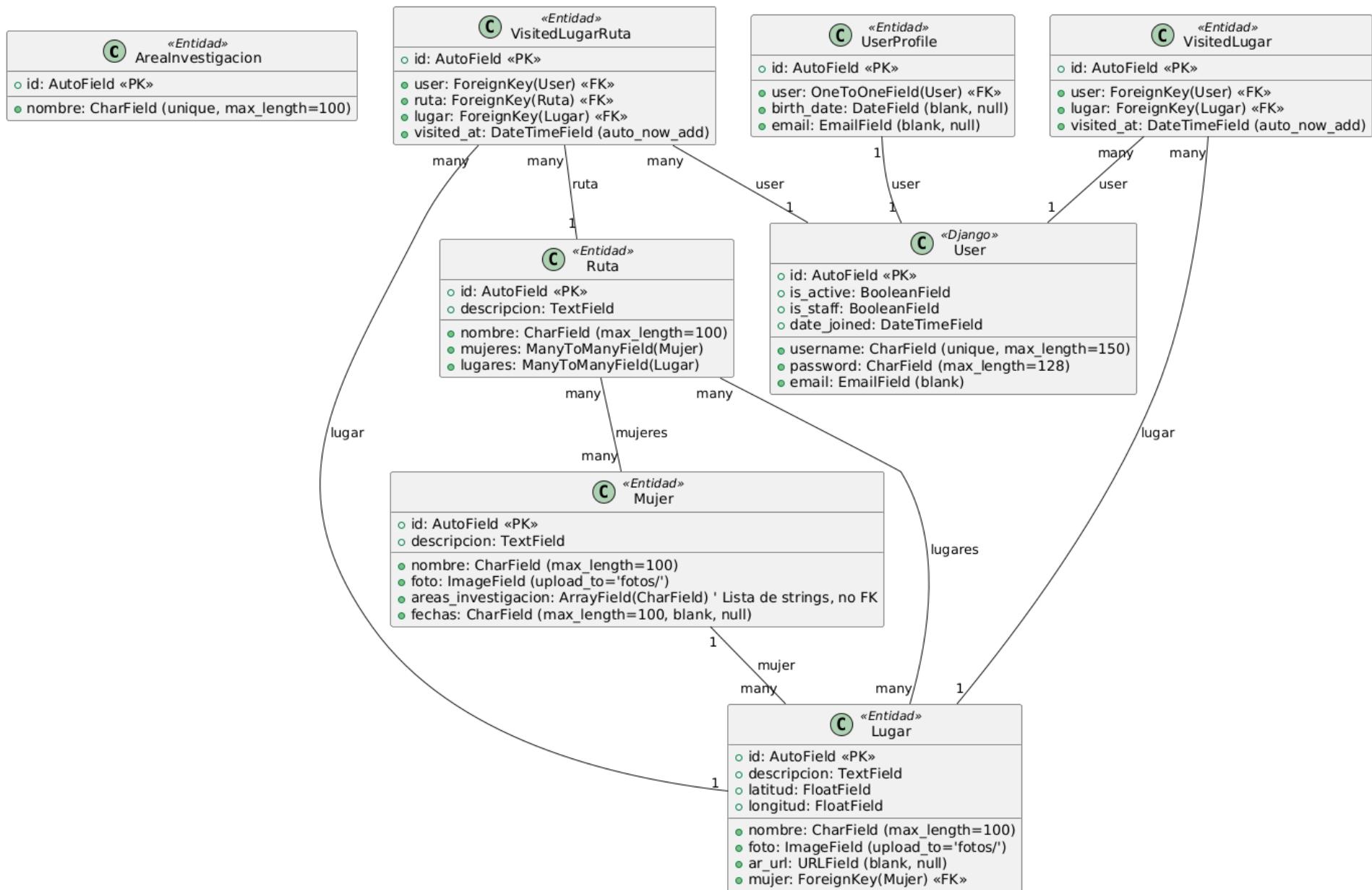


Figura 5.1: Diagrama de clases de la base de datos

5.2. Diseño de la arquitectura del frontend

El frontend de la aplicación está desarrollado en **React Native**, siguiendo una arquitectura basada en componentes funcionales y el uso de contextos para la gestión eficiente del estado global. El hecho de que el diseño sea modular y esté desacoplado hace que sea más sencillo de mantener y escalar, además de permitir añadir nuevas funcionalidades sin complicaciones. Esto también ayuda a que la experiencia de usuario sea más fácil de adaptar en un futuro con nuevas necesidades.

5.2.1. Diagrama de clases del frontend

La Figura 5.2 muestra el diagrama de clases que representa la arquitectura principal del frontend. Los componentes se agrupan en paquetes según su responsabilidad funcional: contextos, navegación, usuario, rutas, lugares y utilidades.

- **Contextos:** El componente `FiltrosContext` centraliza la gestión del estado global de los filtros aplicados en la aplicación, proporcionando métodos para su actualización y reseteo. Este contexto es consumido por componentes como `Filtros`, `Rutas` y `Mapa`, permitiendo la sincronización de los criterios de filtrado en toda la interfaz.
- **Navegación:** El componente `NavigationApp` implementa la navegación principal mediante un *stack navigator*, gestionando el flujo entre pantallas y facilitando la transición entre las distintas funcionalidades de la app.
- **Componentes de Usuario:** Incluyen las pantallas de autenticación (`Login`, `Register`), el perfil de usuario (`PerfilUsuario`) y el acceso al historial de lugares y rutas visitadas (`HistorialLugares`, `HistorialRutas`). Cada uno de estos componentes gestiona su propio estado y lógica de interacción, garantizando la personalización y seguridad de la experiencia de usuario.
- **Componentes de Rutas y Lugares:** `Rutas` y `Mapa` permiten al usuario explorar rutas temáticas y lugares geolocalizados, respectivamente. Los componentes `Detail`, `HistorialLugares` y `HistorialRutas` permiten consultar información detallada de los lugares y el registro de visitas de estos, integrando la lógica de interacción con el backend.
- **Componentes de Utilidad:** Incluyen `Filtros`, `Mujeres` (visualización combinada de mujeres y lugares), la pantalla de bienvenida (`Welcome`), la lógica de realidad aumentada (`AR`), y la lógica de notificaciones de proximidad (`NotificacionProximidad`), que se ejecuta de forma invisible para el usuario y activa notificaciones según la localización.

Las relaciones entre componentes reflejan tanto el consumo de contexto (*consume*), como la navegación entre pantallas (*navega a*), y el uso de lógica auxiliar (por ejemplo, la integración de notificaciones en el componente principal `Main`).

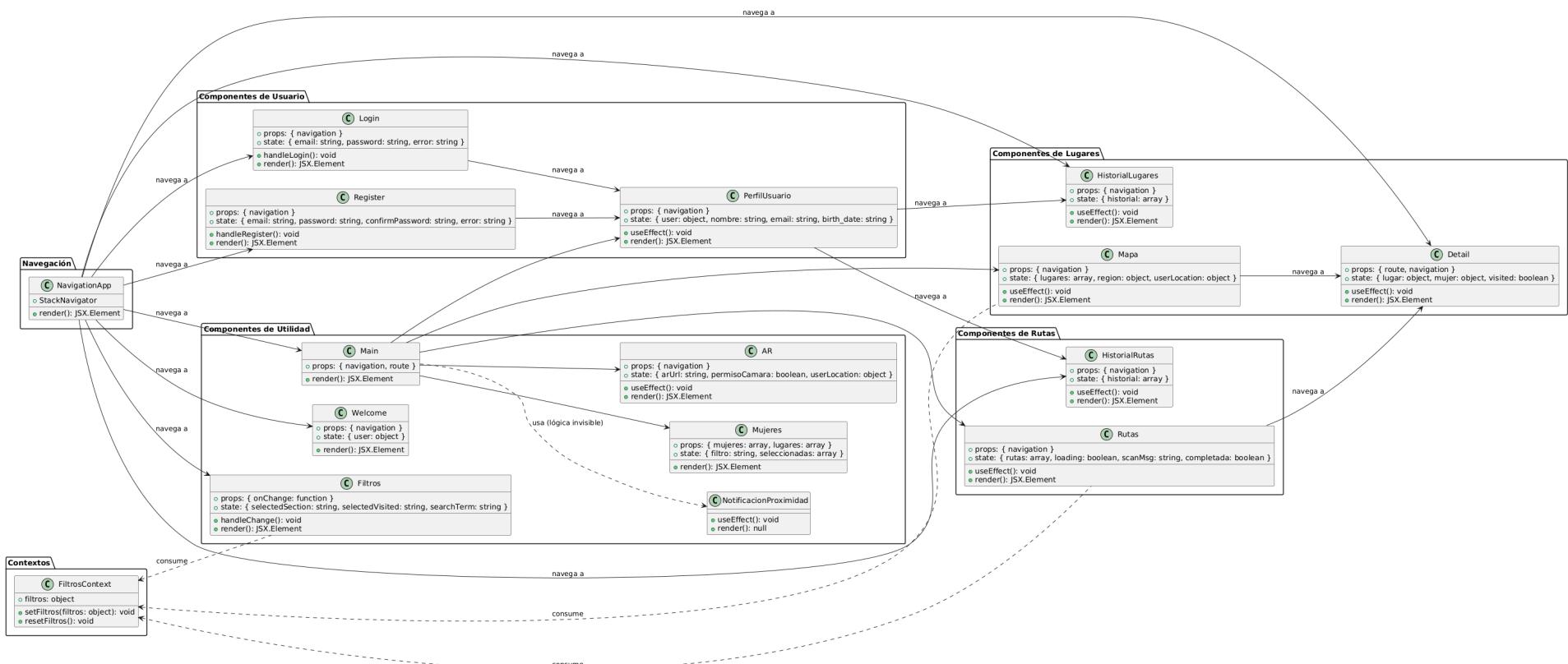


Figura 5.2: Diagrama de clases del frontend React Native: estructura de componentes y relaciones principales.

5.3. Diagramas de secuencia

Los diagramas de secuencia muestran cómo interactúan los distintos elementos del sistema a lo largo del tiempo para llevar a cabo un caso de uso concreto. Representan visualmente el flujo de mensajes entre actores (como el usuario o el administrador) y los componentes internos del sistema (como la aplicación, la base de datos o servicios de autenticación).

En esta sección se presentan los diagramas de secuencia correspondientes a cada uno de los casos de uso definidos en el análisis de requisitos del sistema, descritos en el apartado 4.2 (Especificación de casos de uso).

5.3.1. Diagrama de secuencia del caso de uso 1: Login de usuario

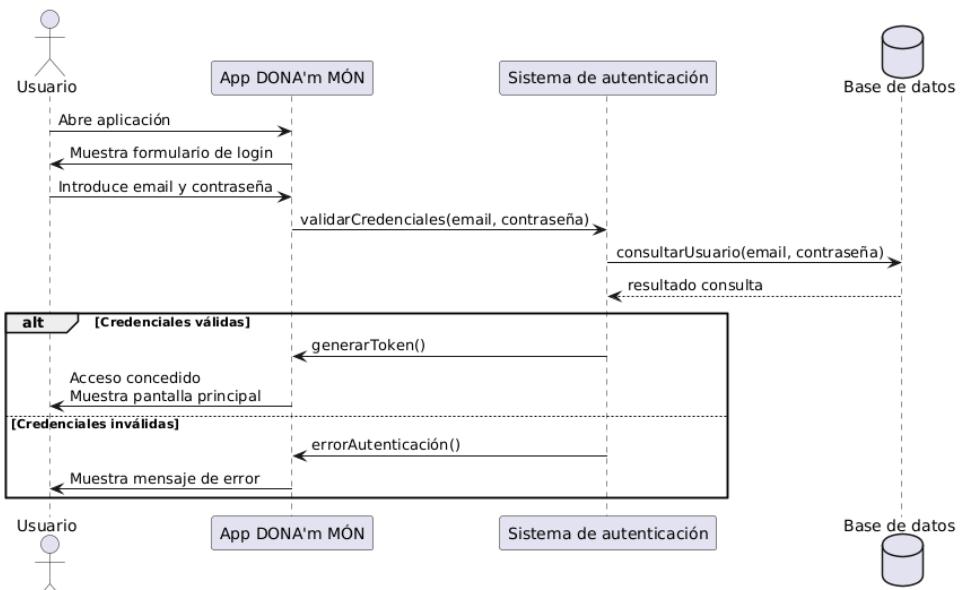


Figura 5.3: Diagrama de secuencia del caso de uso 1: Login de usuario

5.3.2. Diagrama de secuencia del caso de uso 2: Registro de usuario

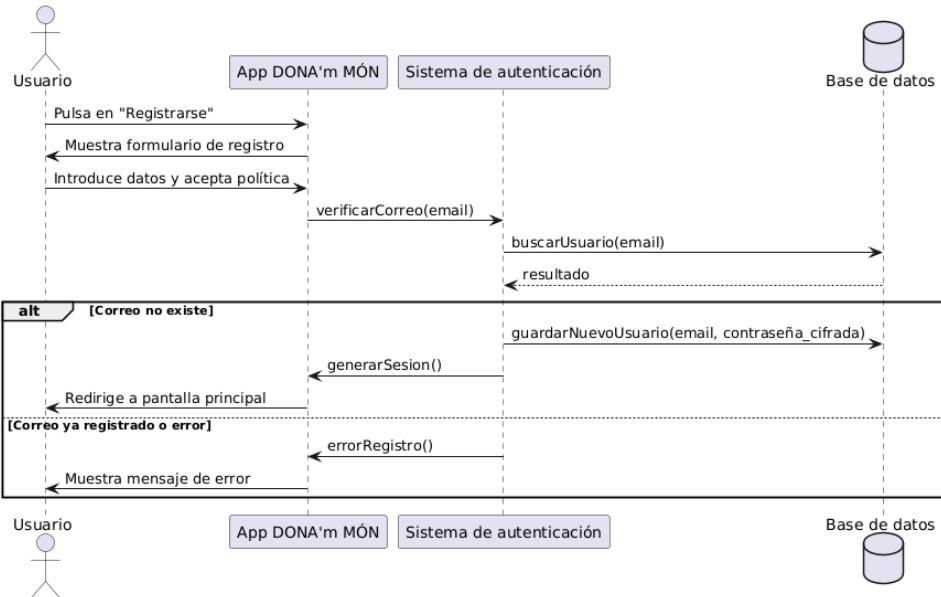


Figura 5.4: Diagrama de secuencia del caso de uso 2: Registro de usuario

5.3.3. Diagrama de secuencia del caso de uso 3: Ver mapa con puntos de interés

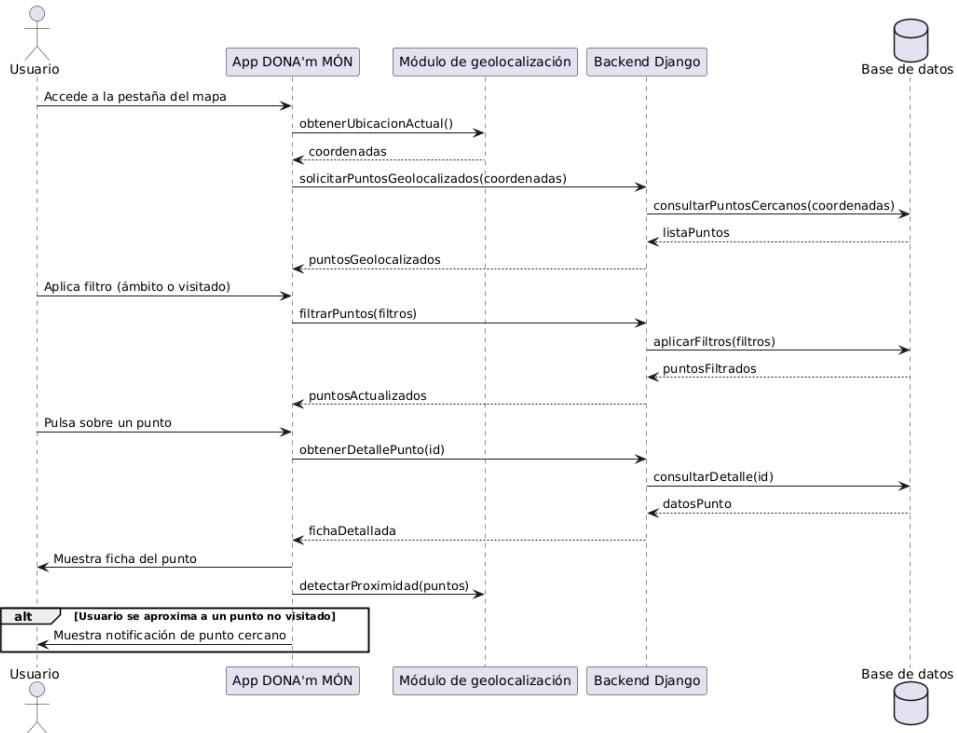


Figura 5.5: Diagrama de secuencia del caso de uso 3: Ver mapa con puntos de interés

5.3.4. Diagrama de secuencia del caso de uso 4: Ver listado de puntos ordenado por cercanía

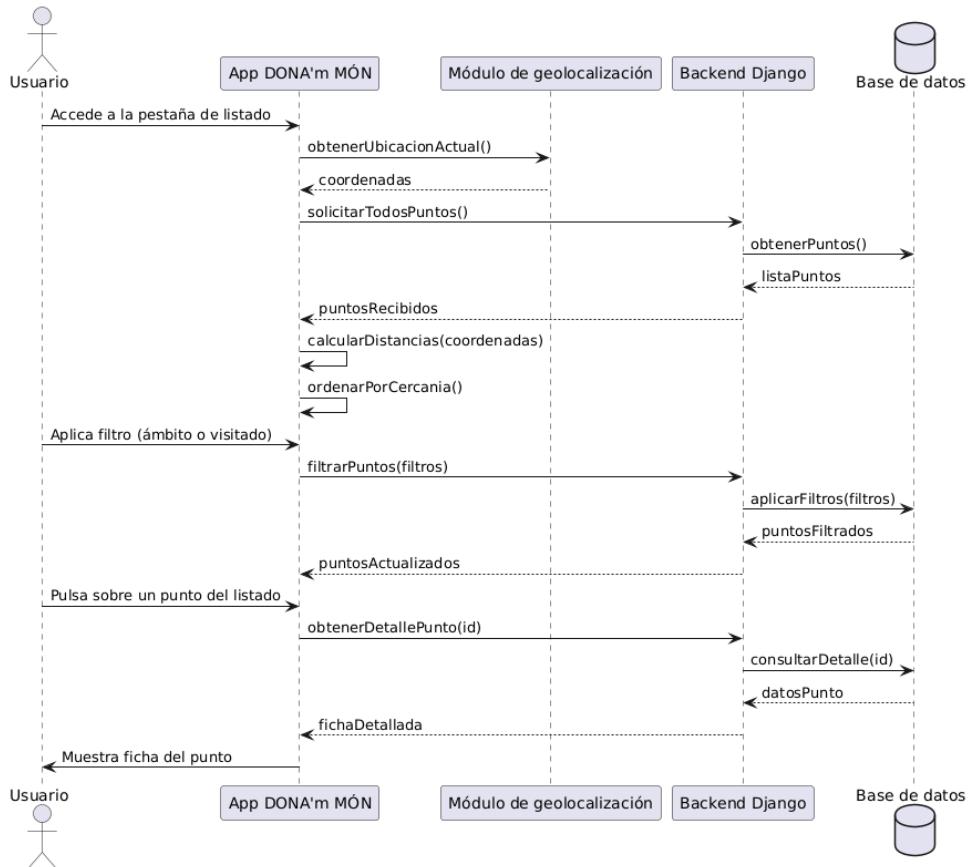


Figura 5.6: Diagrama de secuencia del caso de uso 4: Ver listado de puntos ordenado por cercanía

5.3.5. Diagrama de secuencia del caso de uso 5: Ver ficha del lugar

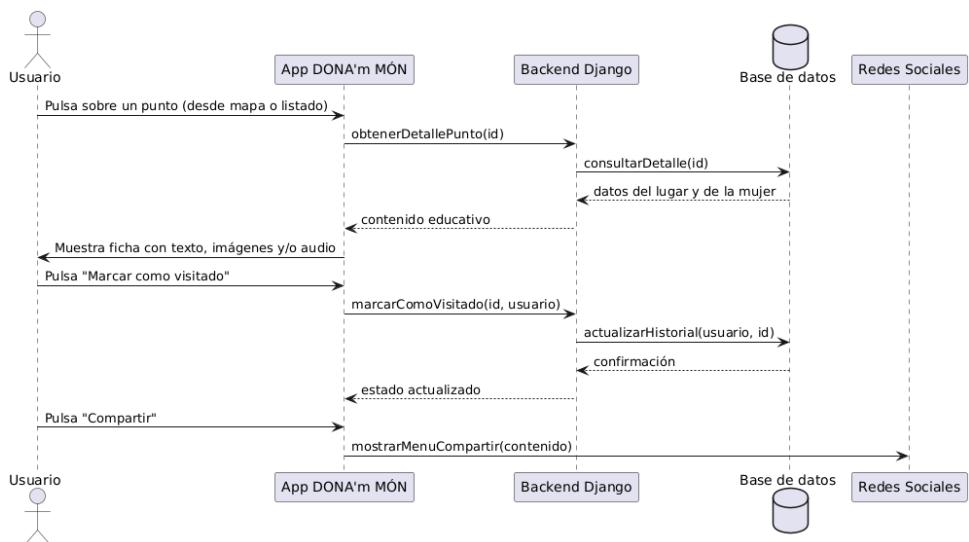


Figura 5.7: Diagrama de secuencia del caso de uso 5: Ver ficha del lugar

5.3.6. Diagrama de secuencia del caso de uso 6: Ver rutas y consultar puntos de una ruta

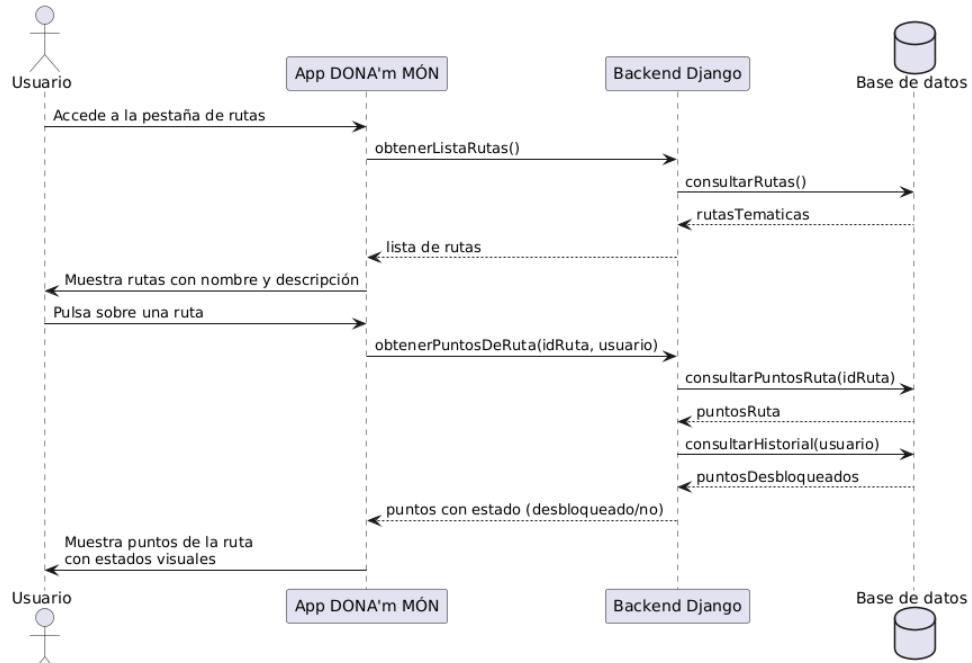


Figura 5.8: Diagrama de secuencia del caso de uso 6: Ver rutas y consultar puntos de una ruta

5.3.7. Diagrama de secuencia del caso de uso 7: Escanear QR de punto de ruta

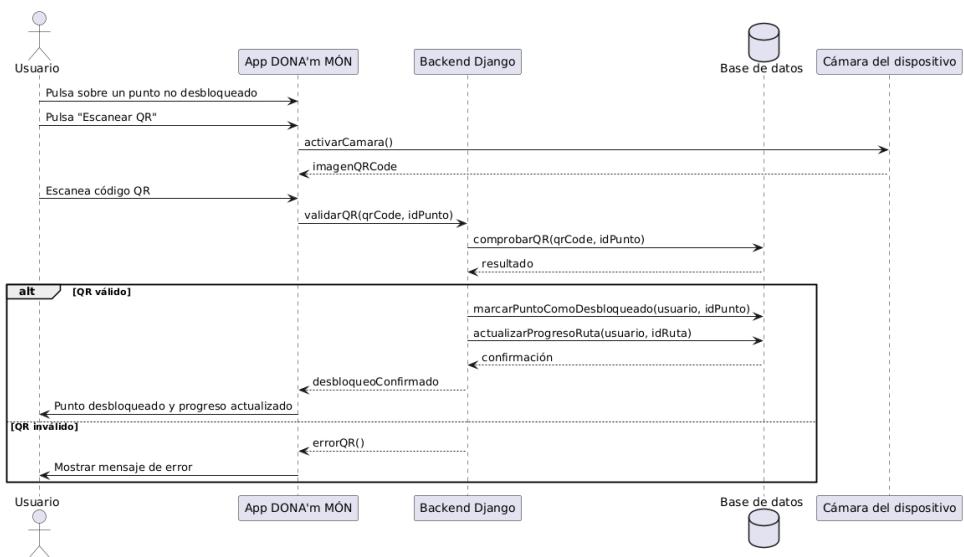


Figura 5.9: Diagrama de secuencia del caso de uso 7: Escanear QR de punto de ruta

5.3.8. Diagrama de secuencia del caso de uso 8: Ver puntos RA disponibles

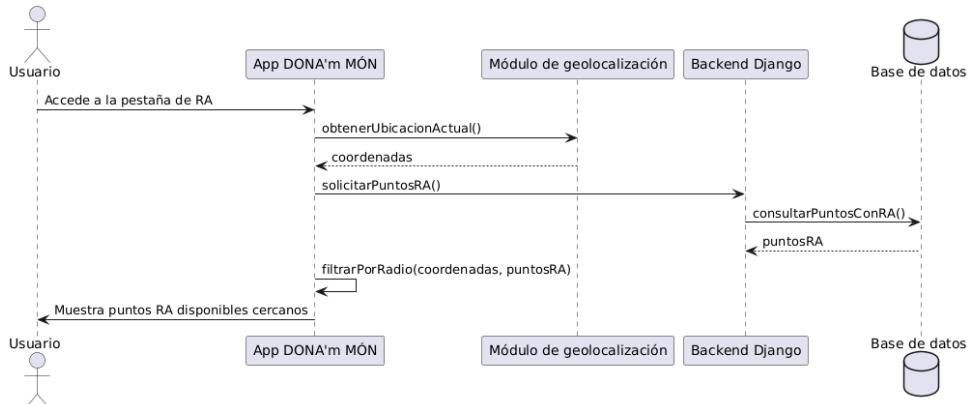


Figura 5.10: Diagrama de secuencia del caso de uso 8: Ver puntos RA disponibles

5.3.9. Diagrama de secuencia del caso de uso 9: Visualizar contenido RA

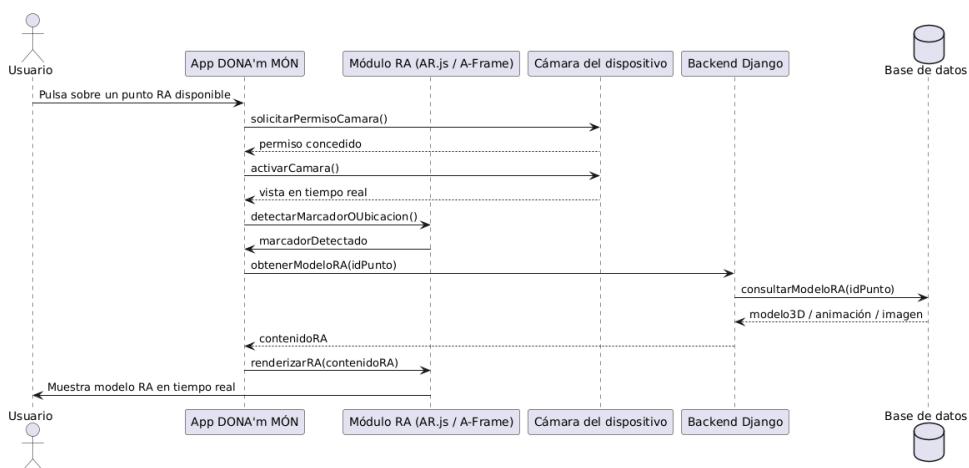


Figura 5.11: Diagrama de secuencia del caso de uso 9: Visualizar contenido RA

5.3.10. Diagrama de secuencia del caso de uso 10: Editar perfil de usuario

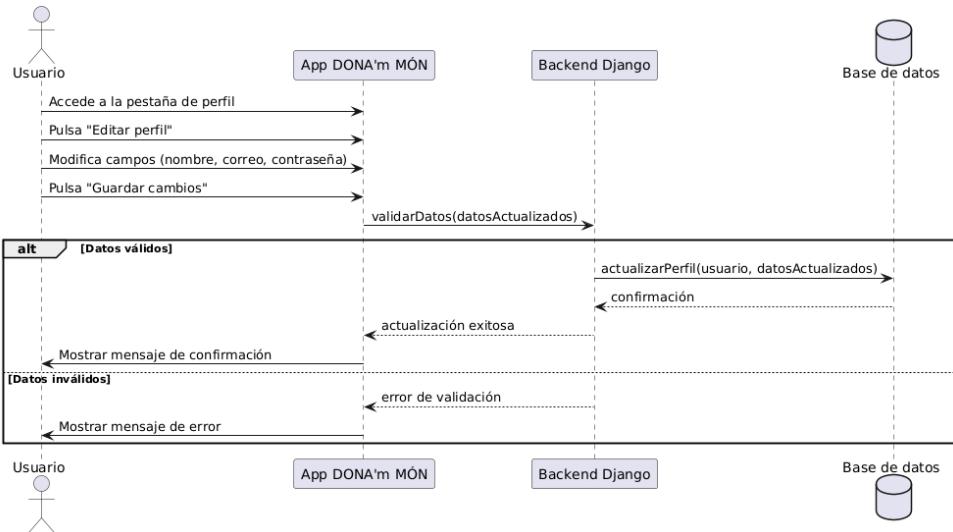


Figura 5.12: Diagrama de secuencia del caso de uso 10: Editar perfil de usuario

5.3.11. Diagrama de secuencia del caso de uso 11: Ver historial de lugares visitados y rutas completadas

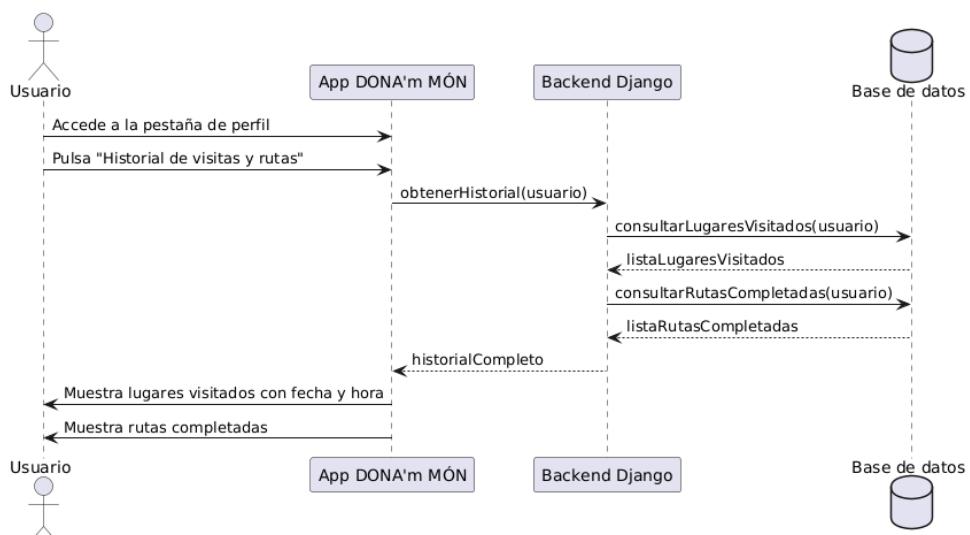


Figura 5.13: Diagrama de secuencia del caso de uso 11: Ver historial de lugares visitados y rutas completadas

5.3.12. Diagrama de secuencia del caso de uso 12: Cerrar sesión

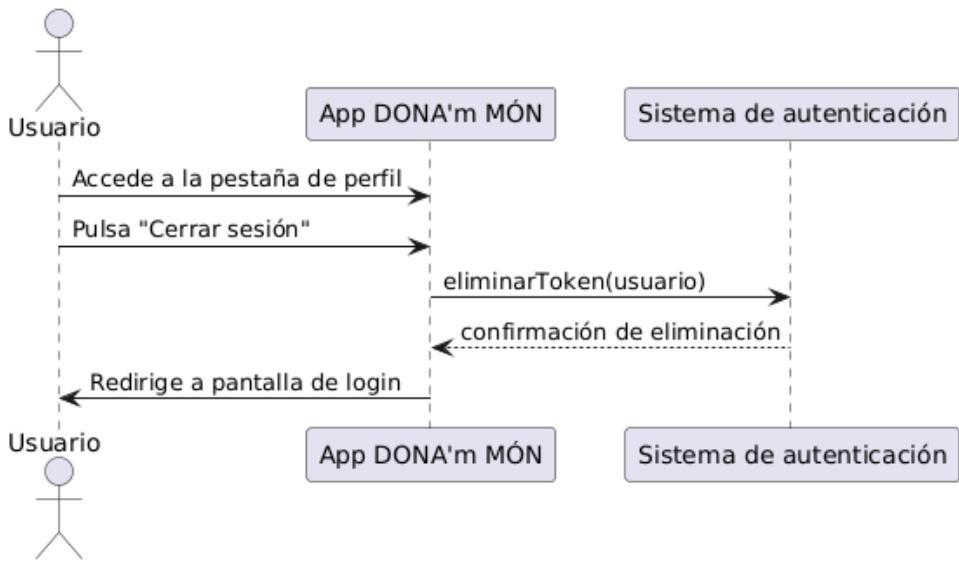


Figura 5.14: Diagrama de secuencia del caso de uso 12: Cerrar sesión

5.3.13. Diagrama de secuencia del caso de uso 13: Iniciar sesión como administrador

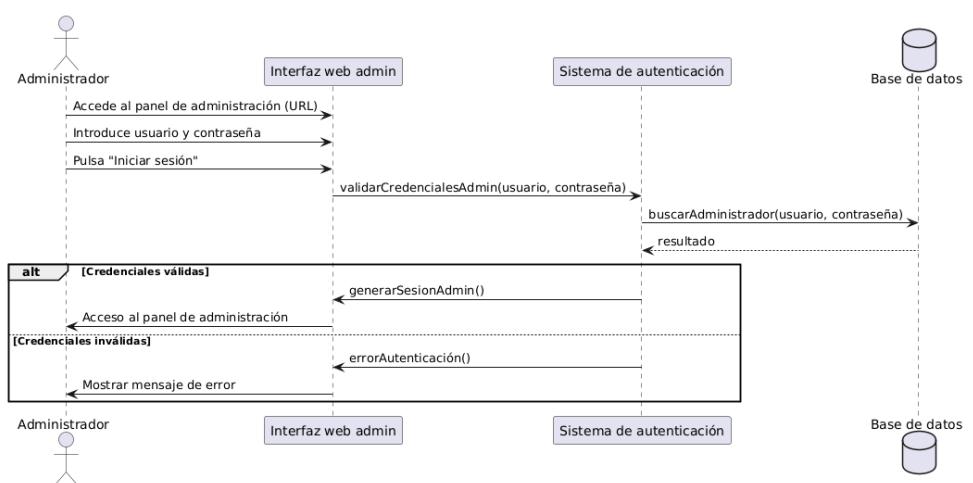


Figura 5.15: Diagrama de secuencia del caso de uso 13: Iniciar sesión como administrador

5.3.14. Diagrama de secuencia del caso de uso 14: Gestionar mujeres históricas

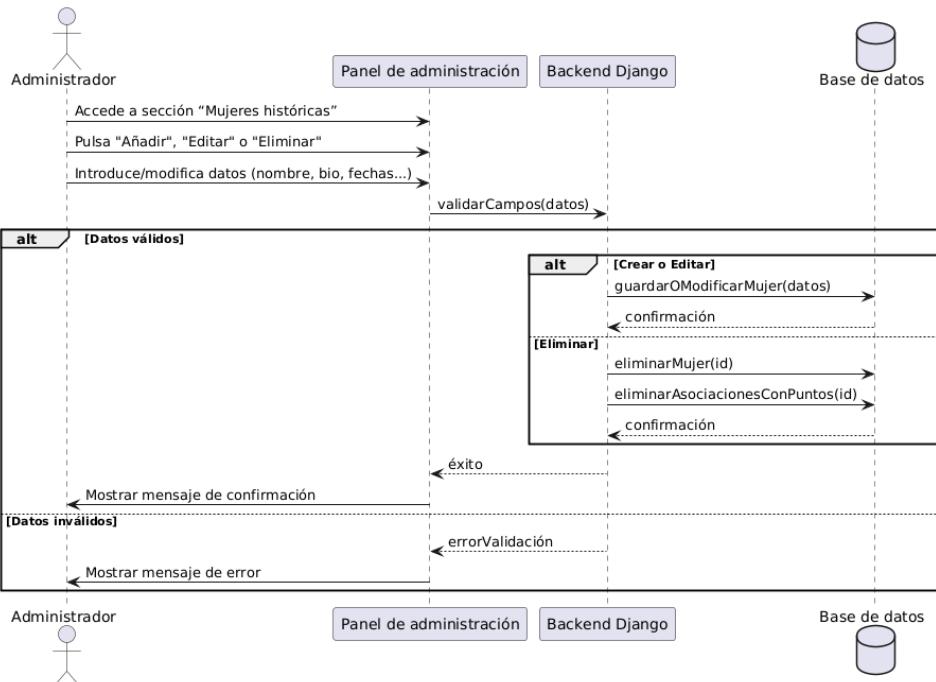


Figura 5.16: Diagrama de secuencia del caso de uso 14: Gestionar mujeres históricas

5.3.15. Diagrama de secuencia del caso de uso 15: Gestionar puntos geolocalizados

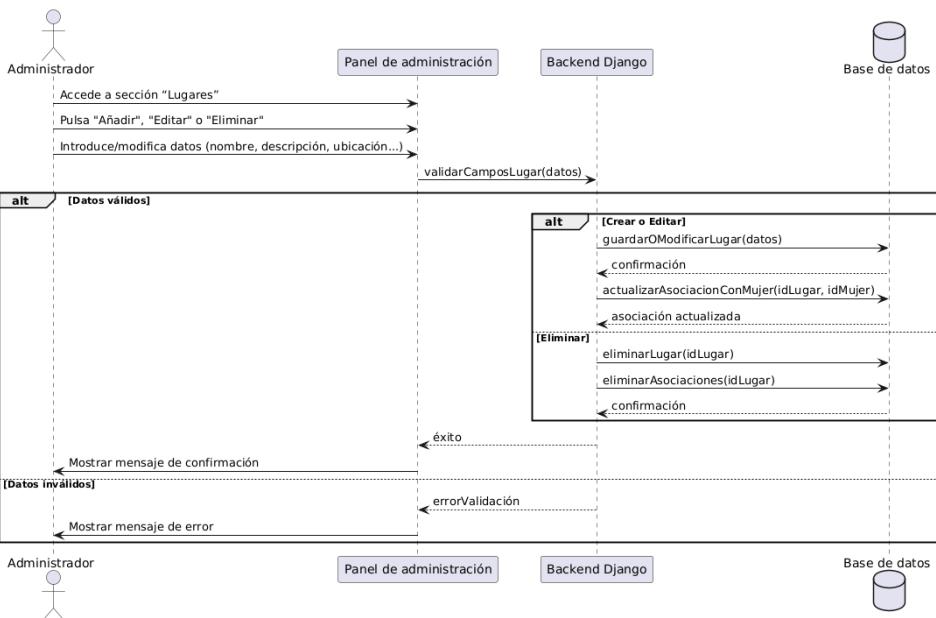


Figura 5.17: Diagrama de secuencia del caso de uso 15: Gestionar puntos geolocalizados

5.3.16. Diagrama de secuencia del caso de uso 16: Gestionar rutas temáticas

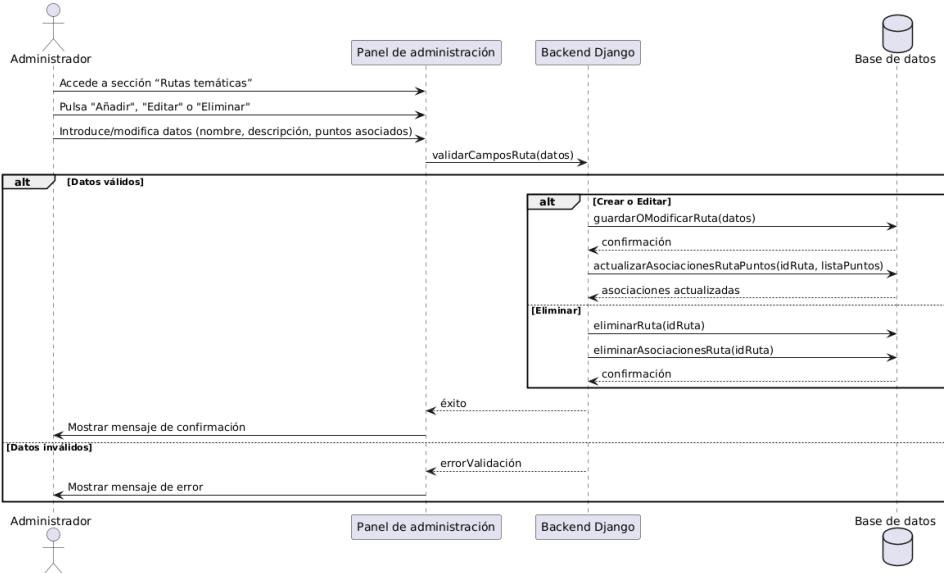


Figura 5.18: Diagrama de secuencia del caso de uso 16: Gestionar rutas temáticas

5.3.17. Diagrama de secuencia del caso de uso 17: Consultar lista de usuarios

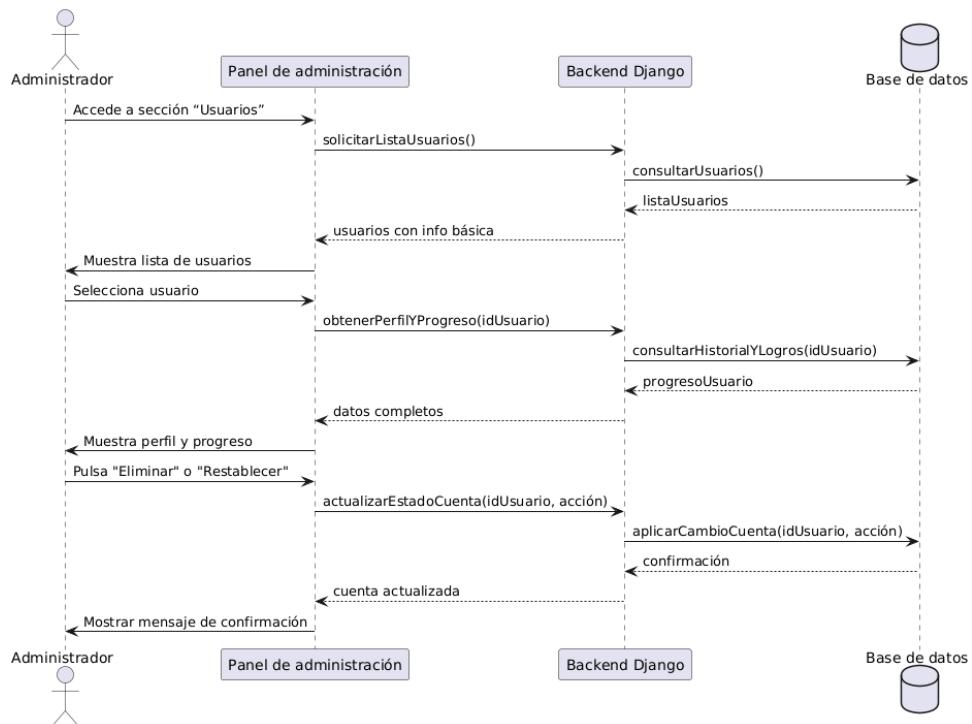


Figura 5.19: Diagrama de secuencia del caso de uso 17: Consultar lista de usuarios

5.3.18. Diagrama de secuencia del caso de uso 18: Gestionar materiales multimedia

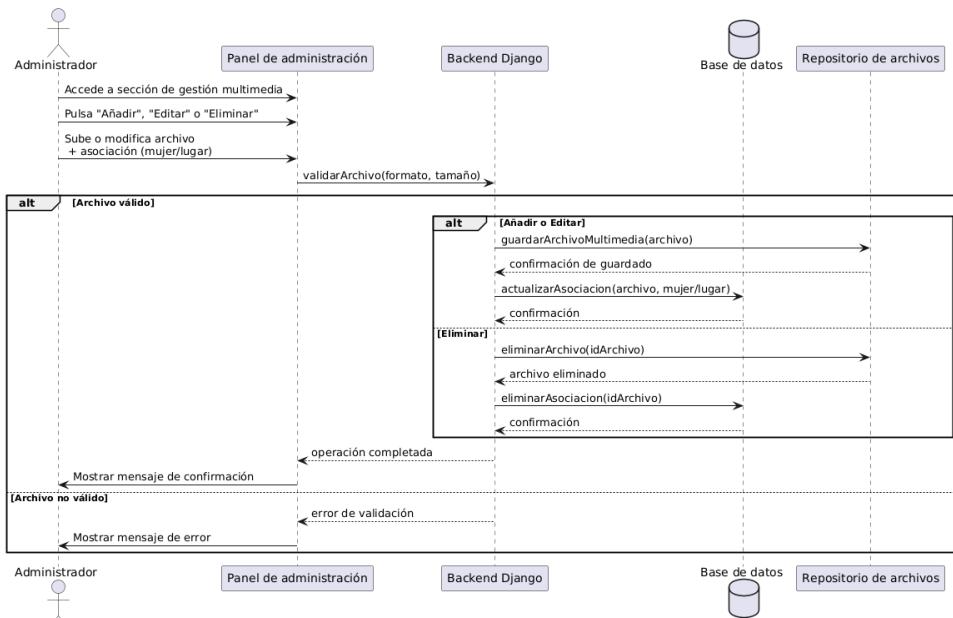


Figura 5.20: Diagrama de secuencia del caso de uso 18: Gestionar materiales multimedia

5.3.19. Diagrama de secuencia del caso de uso 19: Consultar métricas de uso

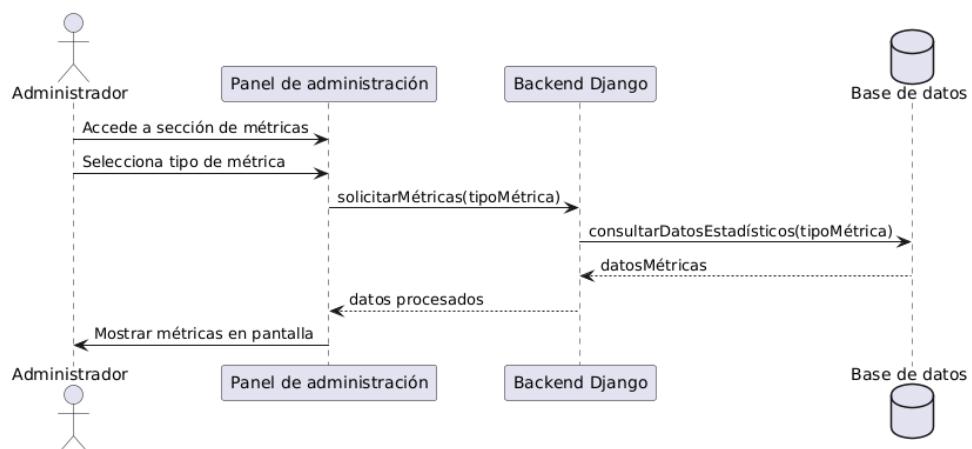


Figura 5.21: Diagrama de secuencia del caso de uso 19: Consultar métricas de uso

5.3.20. Diagrama de secuencia del caso de uso 20: Gestionar áreas de investigación

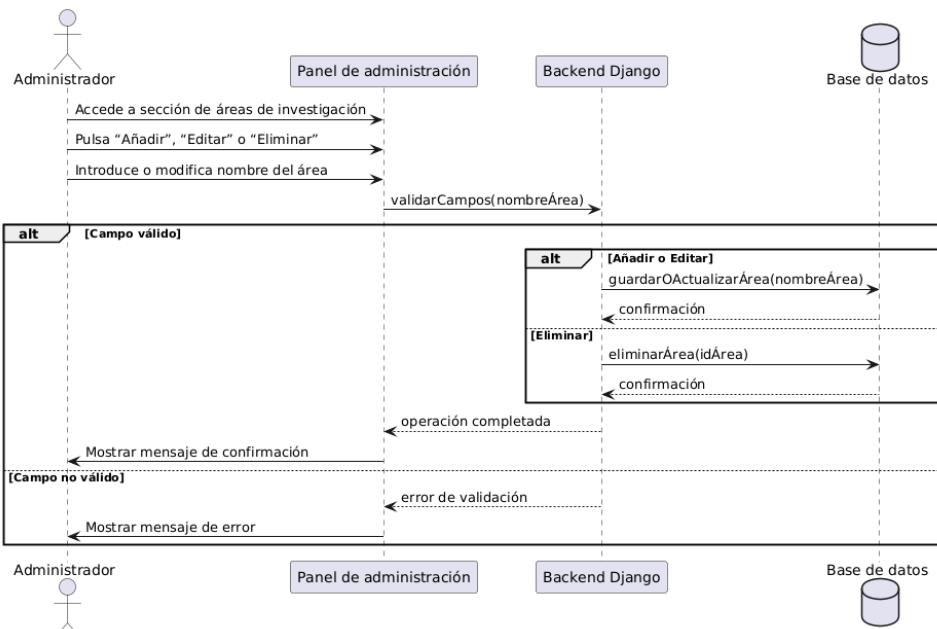


Figura 5.22: Diagrama de secuencia del caso de uso 20: Gestionar áreas de investigación

Capítulo 6

Implementación y pruebas

6.1. Implementación del backend

6.1.1. Herramientas, frameworks y lenguajes utilizados

El backend de la aplicación se ha desarrollado utilizando el framework **Django** (versión 5.2.2) junto con **Django REST Framework** para la construcción de la API. Django facilita la definición de modelos, la gestión de usuarios y la exposición de endpoints RESTful de manera segura y eficiente.

Para la persistencia de datos se ha empleado **PostgreSQL** como sistema gestor de bases de datos relacional. El entorno de desarrollo y despliegue se ha gestionado mediante **Docker** y **Docker Compose**, lo que permite la portabilidad y la replicabilidad del entorno en diferentes máquinas.

Las principales herramientas y librerías utilizadas son:

- **Python 3.10**
- **Django 5.2.2**
- **Django REST Framework (djangorestframework)**
- **Docker** y **Docker Compose** para la gestión de contenedores
- **psycopg2-binary** para la conexión con PostgreSQL
- **django-cors-headers** para permitir peticiones desde el frontend
- **Pillow** para el manejo de imágenes en Django

6.1.2. Estructura del proyecto y organización del código

El código fuente del backend se encuentra en la carpeta `donam-mon-backend/`, que contiene la configuración de Docker, los archivos de requisitos y el proyecto Django propiamente dicho. La estructura principal es la siguiente:

Listado 6.1: Estructura principal del backend

```
donam-mon-backend/
  docker-compose.yml
```

```

Dockerfile
requirements.txt

backend/
    manage.py
    db.sqlite3

config/
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py

mujeres/
    __init__.py
    admin.py
    apps.py
    models.py
    serializers.py
    tests.py
    views.py

migrations/

```

config/: Contiene la configuración global del proyecto Django (ajustes, rutas principales, etc.).

mujeres/: Es la aplicación principal, donde se definen los modelos, vistas, serializers, administración y lógica de negocio.

6.1.3. Modelado de datos

El modelado de datos es uno de los aspectos más relevantes del backend. Se han definido modelos para representar mujeres relevantes, lugares asociados, rutas temáticas, usuarios y el historial de visitas. A continuación se muestra un fragmento representativo del modelo **Mujer**:

Listado 6.2: Modelo Mujer

```

class Mujer(models.Model):
    nombre = models.CharField(max_length=100)
    descripcion = models.TextField()
    foto = models.ImageField(upload_to='fotos/', blank=True, null=True)
    areas_investigacion = ArrayField(
        models.CharField(max_length=100), blank=True, default=list, verbose_name=
        'Áreas de investigación'
    )
    fechas = models.CharField(max_length=100, blank=True, null=True)

```

Se ha optado por utilizar **ArrayField** para el campo **areas_investigacion** en el modelo **Mujer**, aprovechando la capacidad de PostgreSQL para almacenar listas de cadenas en un solo campo. Esta decisión permite simplificar el modelo, evitar tablas intermedias y realizar búsquedas y filtrados eficientes desde la API, facilitando la integración con el

frontend. Se evaluó el uso de `ManyToManyField`, pero se descartó por la simplicidad y rendimiento que ofrece `ArrayField` en este caso.

El modelo `Lugar` representa los lugares geolocalizados asociados a una mujer, e incluye campos para la latitud, longitud, foto y un enlace a contenido de realidad aumentada. En lugar de almacenar los modelos 3D directamente en la base de datos o en el propio backend, se ha optado por subir los archivos 3D y el HTML necesario a un repositorio externo (GitHub Pages) y guardar únicamente la URL de acceso en el campo `ar_url`. De esta forma, la base de datos y el backend no se ven saturados con archivos grandes ni con peticiones de contenido estático. Además, al utilizar servicios externos optimizados para servir archivos estáticos y contenido web se mejora tanto la velocidad de carga como la escalabilidad de la aplicación. Así, el campo `ar_url` actúa como referencia a la experiencia de realidad aumentada, permitiendo que el frontend acceda directamente al recurso externo de forma eficiente y desacoplada del backend. Esta estrategia también facilita la actualización y gestión de los modelos 3D y sus visores HTML, ya que no es necesario modificar ni desplegar el backend cada vez que se realiza un cambio en estos recursos.

Listado 6.3: Modelo Lugar

```
class Lugar(models.Model):
    nombre = models.CharField(max_length=100)
    descripcion = models.TextField()
    latitud = models.FloatField(blank=True, null=True)
    longitud = models.FloatField(blank=True, null=True)
    foto = models.ImageField(upload_to='fotos/', blank=True, null=True)
    ar_url = models.URLField(max_length=300, blank=True, null=True,
        verbose_name='URL de Realidad Aumentada')
    mujer = models.ForeignKey(Mujer, on_delete=models.CASCADE, related_name='lugares')
```

El modelo `Ruta` agrupa lugares y mujeres en recorridos temáticos, utilizando relaciones `ManyToManyField` para maximizar la flexibilidad:

Listado 6.4: Modelo Ruta

```
class Ruta(models.Model):
    nombre = models.CharField(max_length=100, unique=True)
    descripcion = models.TextField(blank=True)
    mujeres = models.ManyToManyField(Mujer, related_name='rutas', blank=True)
    lugares = models.ManyToManyField(Lugar, related_name='rutas', blank=True)
```

Para la gestión de usuarios se utiliza el modelo estándar de Django, extendido mediante un modelo `UserProfile` que almacena información adicional relevante para la aplicación:

Listado 6.5: Modelo UserProfile

```
class UserProfile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE, related_name='profile')
    birth_date = models.DateField(blank=True, null=True)
    email = models.EmailField(blank=True, null=True)
```

El historial de visitas se gestiona con los modelos `VisitedLugar` y `VisitedLugarRuta`, que permiten registrar tanto visitas individuales como visitas dentro de rutas temáticas:

Listado 6.6: Modelo VisitedLugar

```
class VisitedLugar(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE, related_name='visited_lugares')
    lugar = models.ForeignKey(Lugar, on_delete=models.CASCADE, related_name='visitas')
    visited_at = models.DateTimeField(auto_now_add=True)
```

Listado 6.7: Modelo VisitedLugarRuta

```
class VisitedLugarRuta(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE, related_name='visited_lugares_ruta')
    ruta = models.ForeignKey(Ruta, on_delete=models.CASCADE, related_name='visitas')
    lugar = models.ForeignKey(Lugar, on_delete=models.CASCADE, related_name='visitas_ruta')
    visited_at = models.DateTimeField(auto_now_add=True)
```

6.1.4. Gestión de imágenes

Para la gestión de imágenes, se utiliza el campo `ImageField` en los modelos. En la configuración de Django se define la ruta de almacenamiento y la URL de acceso a los archivos subidos:

Listado 6.8: Configuración de imágenes en settings.py

```
MEDIA_URL = '/fotos/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'fotos')
```

En `config/urls.py` se añade la siguiente configuración para servir archivos en desarrollo:

Listado 6.9: Configuración de media en urls.py

```
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    # ... rutas ...
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

6.1.5. Paginación de la API

Para mejorar la eficiencia y la experiencia de usuario, la API implementa paginación por defecto:

Listado 6.10: Paginación en settings.py

```
REST_FRAMEWORK = {
    'DEFAULT_PAGINATION_CLASS': 'rest_framework.pagination.PageNumberPagination',
    ,
    'PAGE_SIZE': 10,
}
```

6.1.6. Seguridad y permisos

La autenticación de usuarios se gestiona mediante el sistema estándar de Django, complementado con autenticación basada en tokens para la API. En la configuración de Django REST Framework se especifica el uso de `TokenAuthentication` y los permisos por defecto:

Listado 6.11: Configuración de autenticación y permisos

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework.authentication.TokenAuthentication',
    ],
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.IsAuthenticated',
    ],
}
```

Nota: En producción, `DEBUG` debe estar siempre en `False` y el `SECRET_KEY` debe obtenerse de una variable de entorno. Además, `CORS_ALLOW_ALL_ORIGINS = True` es solo para desarrollo; en producción se recomienda restringir los orígenes permitidos.

6.1.7. Gestión de variables de entorno y configuración segura

Para evitar exponer credenciales sensibles en el código fuente, se utiliza un archivo `.env` para definir variables de entorno como las credenciales de la base de datos. El archivo `docker-compose.yml` y la configuración de Django acceden a estas variables para una gestión segura y flexible.

Listado 6.12: Archivo `.env`

```
POSTGRES_DB=donamdb
POSTGRES_USER=donamuser
POSTGRES_PASSWORD=donampass
```

6.1.8. Serialización y exposición de la API REST

La serialización de los modelos se realiza mediante clases `ModelSerializer`, que permiten transformar los objetos de la base de datos en formatos JSON fácilmente consumibles por el frontend y validar los datos recibidos antes de almacenarlos. En este proyecto, los serializers también gestionan relaciones entre modelos y añaden campos personalizados, como URLs absolutas para las imágenes.

Por ejemplo, el serializador para el modelo `Mujer` incluye la serialización anidada de los lugares asociados y un campo calculado para la URL de la foto:

Listado 6.13: Serializador `Mujer`

```
class MujerSerializer(serializers.ModelSerializer):
    lugares = LugarSerializer(many=True, read_only=True)
    foto_url = serializers.SerializerMethodField()

    class Meta:
        model = Mujer
        fields = ('id', 'nombre', 'descripcion', 'foto', 'foto_url', 'lugares', 'areas_investigacion')
```

```

def get_foto_url(self, obj):
    request = self.context.get('request')
    if obj.foto:
        url = obj.foto.url
        url = url.replace('/fotos/fotos/', '/fotos/')
        if request is not None:
            return request.build_absolute_uri(url)
        else:
            return url
    return None

```

El serializador para el modelo Lugar también incluye la relación con la mujer asociada (mostrando su nombre), así como un campo personalizado para la URL de la foto:

Listado 6.14: Serializador Lugar

```

class LugarSerializer(serializers.ModelSerializer):
    mujer = serializers.StringRelatedField()
    foto_url = serializers.SerializerMethodField()

    class Meta:
        model = Lugar
        fields = ('id', 'nombre', 'descripcion', 'latitud', 'longitud', 'mujer',
                  'foto', 'foto_url', 'ar_url')

    def get_foto_url(self, obj):
        request = self.context.get('request')
        if obj.foto:
            url = obj.foto.url
            url = url.replace('/fotos/fotos/', '/fotos/')
            if request is not None:
                return request.build_absolute_uri(url)
            else:
                return url
        return None

```

El serializador de perfil de usuario permite exponer y validar los datos adicionales del usuario:

Listado 6.15: Serializador UserProfile

```

class UserProfileSerializer(serializers.ModelSerializer):
    class Meta:
        model = UserProfile
        fields = ['birth_date', 'email']

```

El serializador de registro de usuario permite crear un usuario y su perfil asociado en una sola petición:

Listado 6.16: Serializador UserRegister

```

class UserRegisterSerializer(serializers.ModelSerializer):
    profile = UserProfileSerializer()
    class Meta:

```

```

model = User
fields = ['username', 'password', 'profile']
extra_kwargs = {'password': {'write_only': True}}


def create(self, validated_data):
    profile_data = validated_data.pop('profile')
    user = User.objects.create_user(username=validated_data['username'],
                                    password=validated_data['password'])
    UserProfile.objects.create(user=user, **profile_data)
    return user

```

El serializador de visitas a lugares incluye la información completa del lugar visitado:

Listado 6.17: Serializador VisitedLugar

```

class VisitedLugarSerializer(serializers.ModelSerializer):
    lugar = serializers.SerializerMethodField()
    class Meta:
        model = VisitedLugar
        fields = ['id', 'lugar', 'visited_at']

    def get_lugar(self, obj):
        return LugarSerializer(obj.lugar, context=self.context).data

```

El serializador de visitas a lugares dentro de rutas:

Listado 6.18: Serializador VisitedLugarRuta

```

class VisitedLugarRutaSerializer(serializers.ModelSerializer):
    lugar = LugarSerializer(read_only=True)
    mujer = serializers.StringRelatedField(read_only=True)

    class Meta:
        model = VisitedLugarRuta
        fields = ('id', 'user', 'mujer', 'lugar', 'visited_at')

```

Y el serializador de rutas, que expone tanto las mujeres como los lugares asociados:

Listado 6.19: Serializador Ruta

```

class RutaSerializer(serializers.ModelSerializer):
    mujeres = serializers.StringRelatedField(many=True)
    lugares = LugarSerializer(many=True, read_only=True)
    class Meta:
        model = Ruta
        fields = ('id', 'nombre', 'descripcion', 'mujeres', 'lugares')

```

6.1.9. Vistas, lógica de endpoints y ejemplos de lógica relevante

Las vistas de la API se han implementado principalmente mediante `ModelViewSet`, lo que permite disponer de endpoints CRUD (crear, leer, actualizar, borrar) de forma automática y segura. Por ejemplo, la vista para el modelo `Mujer` es:

Listado 6.20: Vista MujerViewSet

```

class MujerViewSet(viewsets.ModelViewSet):

```

```
queryset = Mujer.objects.all()
serializer_class = MujerSerializer
```

Para la lógica de visitas, se han implementado endpoints que permiten registrar una visita y consultar el historial de visitas de un usuario. Un ejemplo de lógica relevante en la vista podría ser la validación para evitar duplicados en el historial:

Listado 6.21: Lógica para evitar duplicados en visitas

```
class VisitedLugarViewSet(viewsets.ModelViewSet):
    queryset = VisitedLugar.objects.all()
    serializer_class = VisitedLugarSerializer

    def perform_create(self, serializer):
        user = self.request.user
        lugar = serializer.validated_data['lugar']
        if VisitedLugar.objects.filter(user=user, lugar=lugar).exists():
            raise ValidationError("Ya has registrado una visita a este lugar.")
        serializer.save(user=user)
```

6.1.10. Administración y gestión de datos

Django proporciona una interfaz de administración (*Django Admin*) que ha sido personalizada para facilitar la gestión de los modelos principales de la aplicación. En el archivo `admin.py` de la app `mujeres`, se han registrado los modelos y se han configurado opciones de visualización y filtrado para mejorar la experiencia de los administradores:

Listado 6.22: Registro del modelo Mujer en Django Admin

```
from django.contrib import admin
from .models import Mujer, Lugar, VisitedLugarRuta, Ruta

class LugarAdmin(admin.ModelAdmin):
    list_display = ('nombre', 'descripcion', 'latitud', 'longitud', 'ar_url')
    search_fields = ('nombre',)

class MujerAdmin(admin.ModelAdmin):
    list_display = ('nombre', 'descripcion')
    search_fields = ('nombre',)

class RutaAdmin(admin.ModelAdmin):
    list_display = ('nombre', 'descripcion')
    search_fields = ('nombre',)
    filter_horizontal = ('mujeres', 'lugares')

admin.site.register(Mujer, MujerAdmin)
admin.site.register(Lugar, LugarAdmin)
admin.site.register(VisitedLugarRuta)
admin.site.register(Ruta, RutaAdmin)
```

6.1.11. Cambio de contraseña del usuario administrador de Django

Si se olvida la contraseña del usuario administrador o es necesario restablecerla, Django permite cambiarla fácilmente mediante un comando de gestión. Dado que el backend se ejecuta en un contenedor Docker, el proceso recomendado es el siguiente:¹

Listado 6.23: Cambio de contraseña del usuario administrador en Docker

```
# Acceder al contenedor donde corre Django (normalmente llamado 'web')
docker compose exec web python manage.py changepassword <nombre_usuario>
```

Por ejemplo, para el usuario ivana:

```
docker compose exec web python manage.py changepassword ivana
```

El sistema solicitará la nueva contraseña de forma interactiva. El servicio se llama `web`, tal y como se indica en el archivo `docker-compose.yml`.

6.1.12. Gestión de migraciones y consistencia de la base de datos

Django utiliza un sistema de migraciones para versionar y aplicar cambios en el esquema de la base de datos. Cada vez que se modifica un modelo, se genera una nueva migración que puede ser aplicada en cualquier entorno, garantizando la coherencia entre desarrollo, pruebas y producción.

En este proyecto, el campo `areas_investigacion` del modelo `Mujer` ha pasado de ser un `ArrayField` a un `ManyToManyField` y de nuevo a `ArrayField`, lo que se refleja en las migraciones generadas automáticamente por Django. Este proceso ha permitido comparar el rendimiento y la simplicidad de ambas aproximaciones, optando finalmente por `ArrayField` por su integración directa con PostgreSQL y la naturaleza de los datos.

Listado 6.24: Fragmento de migraciones reales

```
# 0001_initial.py
migrations.CreateModel(
    name='Mujer',
    fields=[
        ('id', models.BigAutoField(primary_key=True)),
        ('nombre', models.CharField(max_length=100)),
        ('descripcion', models.TextField()),
        ('foto', models.ImageField(blank=True, null=True, upload_to='')),
        ('areas_investigacion', django.contrib.postgres.fields.ArrayField(
            base_field=models.CharField(max_length=100), blank=True, default=list
            , verbose_name='Áreas de investigación')),
        ('fechas', models.CharField(blank=True, max_length=100, null=True)),
    ],
)
# 0002: Cambio a ManyToManyField (prueba de diseño)
migrations.RemoveField(model_name='mujer', name='areas_investigacion'),
migrations.AddField(
    model_name='mujer',
    name='areas_investigacion',
```

¹Nota: Si se ejecuta Django fuera de Docker, se usa el mismo comando pero directamente en la terminal.

```

        field=models.ManyToManyField(blank=True, related_name='mujeres', to='mujeres
            .areainvestigacion'),
),
# 0003: Vuelta a ArrayField (por simplicidad y rendimiento)
migrations.RemoveField(model_name='mujer', name='areas_investigacion'),
migrations.AddField(
    model_name='mujer',
    name='areas_investigacion',
    field=django.contrib.postgres.fields.ArrayField(
        base_field=models.CharField(max_length=100), blank=True, default=list,
        verbose_name='Áreas de investigación'),
),

```

Nota: El reinicio de migraciones (eliminando archivos y tablas intermedias) solo debe realizarse en entornos de desarrollo, nunca en producción, para evitar la pérdida de datos.

Listado 6.25: Pasos para reiniciar migraciones

```

# 1. Eliminar los archivos de migración antiguos en la carpeta mujeres/
migrations/
rm mujeres/migrations/*.py

# 2. Borrar las tablas intermedias o problemáticas directamente en la base de
# datos PostgreSQL.
# (Ejemplo usando psql)
DROP TABLE mujeres_mujer_areas_investigacion;

# 3. Ejecutar los comandos de migración
python manage.py makemigrations
python manage.py migrate

```

6.1.13. Integración con el frontend y gestión de CORS

Para permitir la comunicación entre el frontend (desarrollado en React Native) y el backend, se ha configurado el middleware `django-cors-headers`, que habilita las peticiones CORS (Cross-Origin Resource Sharing). Esto es fundamental para que la aplicación móvil pueda consumir la API REST sin restricciones de origen.

En el archivo `settings.py` se ha añadido la configuración necesaria:

Listado 6.26: Configuración de CORS en Django

```

INSTALLED_APPS = [
    # ... otras apps...
    'corsheaders',
    'rest_framework',
    'mujeres',
]

MIDDLEWARE = [
    'corsheaders.middleware.CorsMiddleware',
    # ... otros middlewares...
]

```

```
CORS_ALLOW_ALL_ORIGINS = True # Para desarrollo; en producción se recomienda
    restringir los orígenes permitidos
```

6.1.14. Integración y despliegue con Docker

El backend se ejecuta en un contenedor Docker. El archivo `docker-compose.yml` define los servicios necesarios, incluyendo la base de datos PostgreSQL y el servidor de la aplicación Django.

Listado 6.27: `docker-compose.yml`

```
version: '3.9'

services:
  db:
    image: postgis/postgis:14-3.3
    restart: always
    env_file: .env
    ports:
      - "5432:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data

  web:
    build: .
    command: sh -c "python manage.py migrate && python manage.py runserver
      0.0.0.0:8000"
    volumes:
      - ./backend:/app
    ports:
      - "8000:8000"
    depends_on:
      - db
    env_file: .env

volumes:
  postgres_data:
```

Esto permite levantar todo el entorno de desarrollo y pruebas con un solo comando, garantizando la coherencia entre entornos y facilitando la colaboración.

6.1.15. Pruebas automáticas

Para garantizar la calidad y robustez del backend, se han implementado pruebas automáticas utilizando el framework de testing de Django. Un ejemplo de test para el modelo `Mujer` es:

Listado 6.28: Test unitario para el modelo Mujer

```
from django.test import TestCase
from .models import Mujer

class MujerModelTest(TestCase):
    def test_creacion_mujer(self):
```

```

mujer = Mujer.objects.create(
    nombre='Ada Lovelace',
    descripcion='Pionera de la programación',
    foto='fotos/ada.jpg',
    areas_investigacion=['Matemáticas', 'Computación'],
    fechas='1815-1852'
)
self.assertEqual(mujer.nombre, 'Ada Lovelace')
self.assertIn('Computación', mujer.areas_investigacion)

```

Listado 6.29: Test de API para el endpoint de mujeres

```

from rest_framework.test import APITestCase
from django.urls import reverse
from .models import Mujer

class MujerAPITest(APITestCase):
    def test_listado_mujeres(self):
        Mujer.objects.create(nombre='Ada', descripcion='Pionera',
            areas_investigacion=['Matemáticas'])
        url = reverse('mujer-list') # Asegúrate de que el router DRF registre
            este nombre
        response = self.client.get(url)
        self.assertEqual(response.status_code, 200)
        self.assertEqual(len(response.data), 1)

```

6.1.16. Gestión de emails y notificaciones

Para permitir el envío de notificaciones por correo electrónico (por ejemplo, confirmación de registro o recuperación de contraseña), se ha configurado el backend para utilizar un servidor SMTP. En `settings.py` se definen los parámetros necesarios:

Listado 6.30: Configuración de email en `settings.py`

```

EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_PORT = 587
EMAIL_USE_TLS = True
EMAIL_HOST_USER = 'tu_email@gmail.com'
EMAIL_HOST_PASSWORD = 'tu_contraseña'

```

El envío de emails se realiza mediante la función `send_mail` de Django:

Listado 6.31: Ejemplo de envío de email

```

from django.core.mail import send_mail

send_mail(
    'Bienvenido a la app',
    'Gracias por registrarte.',
    'tu_email@gmail.com',
    [usuario.email],
    fail_silently=False,
)

```

6.1.17. Internacionalización y localización

El backend está preparado para soportar varios idiomas, facilitando la internacionalización de la aplicación. En `settings.py` se configuran los parámetros de idioma y zona horaria:

Listado 6.32: Internacionalización en `settings.py`

```
LANGUAGE_CODE = 'es-es'
TIME_ZONE = 'Europe/Madrid'
USE_I18N = True
USE_L10N = True
USE_TZ = True
```

6.1.18. Gestión de archivos estáticos

Para servir archivos estáticos (CSS, JS, imágenes no subidas por usuarios), se ha configurado la ruta correspondiente en `settings.py`:

Listado 6.33: Archivos estáticos en `settings.py`

```
STATIC_URL = '/static/'
STATIC_ROOT = BASE_DIR / 'backend' / 'static'
```

Durante el despliegue, se ejecuta el comando `python manage.py collectstatic` para recopilar todos los archivos estáticos en la carpeta definida.

```
• PS C:\Users\isiva\Documents\tfg\donam-mon-backend> python manage.py collectstatic
  163 static files copied to 'C:\Users\isiva\Documents\tfg\donam-mon-backend\backend\backend\static'.
```

Figura 6.1: Recopilación de archivos estáticos con `collectstatic` en Django

6.1.19. Gestión de geolocalización y decisión sobre campos GIS

Aunque el proyecto tiene incluida la librería `djangorestframework-gis` (puedes verla en el archivo `requirements.txt`), en realidad no se está usando porque no se han definido campos GIS como `PointField` o `PolygonField`, ni tampoco se utiliza el backend geoespacial de Django (`django.contrib.gis.db.backends.postgis`).

Para la geolocalización, simplemente se guardan las coordenadas de latitud y longitud como campos `FloatField` en el modelo `Lugar`:

Listado 6.34: Fragmento del modelo `Lugar`

```
class Lugar(models.Model):
    nombre = models.CharField(max_length=100)
    descripcion = models.TextField()
    latitud = models.FloatField(blank=True, null=True)
    longitud = models.FloatField(blank=True, null=True)
    # ... otros campos...
```

Esta solución es suficiente para lo que necesita la aplicación ahora mismo: permite guardar y consultar posiciones geográficas sin complicaciones y sin depender de librerías adicionales ni configuraciones avanzadas. Además, facilita la integración y el despliegue

usando el backend estándar de PostgreSQL (`django.db.backends.postgresql`), sin tener que recurrir a PostGIS.

Ventajas de este enfoque:

- El modelo de datos es más sencillo y fácil de mantener.
- La integración y el despliegue funcionan en cualquier entorno PostgreSQL, sin requisitos especiales.
- Se evitan dependencias y configuraciones extra.

En una posible evolución, si en algún momento se necesitan consultas espaciales más avanzadas o funcionalidades GIS (por ejemplo, búsquedas por proximidad o rutas optimizadas), se podría migrar a campos GIS y aprovechar tanto PostGIS como `djangorestframework-gis` sin demasiada dificultad.

6.2. Implementación del frontend

6.2.1. Herramientas, frameworks y lenguajes utilizados

El frontend de la aplicación se ha desarrollado utilizando el framework **React Native** para asegurar la compatibilidad multiplataforma (Android, iOS y web). Se ha utilizado **Expo** para simplificar el desarrollo, pruebas y despliegue, así como para acceder a funcionalidades nativas como la cámara, la geolocalización y las notificaciones.

Las principales herramientas y librerías utilizadas son:

- **React Native** (v0.79.2)
- **Expo** (v53.0.9)
- **React Navigation** para la gestión de la navegación entre pantallas
- `@react-native-async-storage/async-storage` para almacenamiento local seguro
- `axios` para peticiones HTTP
- `expo-location` y `expo-notifications` para geolocalización y notificaciones push
- `react-native-maps` para mostrar mapas y marcadores
- `react-native-vector-icons` para iconografía
- `expo-camera` para escaneo de QR en rutas
- `@react-native-picker/picker` para selectores personalizados

6.2.2. Estructura del proyecto y organización del código

El código fuente del frontend se encuentra en la carpeta `donam-mon-frontend/my-app/`, que contiene los componentes, utilidades, estilos y recursos gráficos. La estructura principal es la siguiente:

Listado 6.35: Estructura principal del frontend

```

my-app/
  App.js
  apiConfig.js
  index.js
  assets/
    # Imágenes y recursos gráficos
    ...
  Donammon/
    Main.js
    Mapa.js
    Mujeres.js
    PerfilUsuario.js
    Detail.js
    RutaDetail.js
    HistorialLugares.js
    HistorialRutas.js
    NotificacionProximidad.js
    AR.js
    Filtros.js
    FiltrosContext.js
    Login.js
    Register.js
    Welcome.js
    NavigationApp.js
    Rutas.js
    ...
  styles/
    styles.js
    styles-detail.js
    styles-filtros.js
    styles-perfil.js
    ...
  ...

```

App.js: Punto de entrada principal, configura el proveedor de filtros y la navegación.

Donammon/: Carpeta principal de componentes de pantalla y utilidades.

assets/: Imágenes y recursos gráficos usados en la interfaz.

6.2.3. Navegación y estructura de pantallas

La navegación principal se gestiona mediante un **Bottom Tab Navigator** definido en **Main.js**, que organiza la aplicación en pestañas: Mapa, Mujeres, Rutas, Realidad Aumentada (RA) y Perfil.

Listado 6.36: Definición de pestañas en Main.js

```

<Tab.Navigator
  screenOptions={{
    tabBarActiveTintColor: '#5f68c4',
    tabBarInactiveTintColor: '#000000',
    tabBarStyle: { backgroundColor: '#edebff' },
  }}
>

```

```

<Tab.Screen
    name="Mapa"
    component={Mapa}
    initialParams={route.params}
    options={{
        headerShown: false,
        tabBarIcon: ({ color, size }) => (
            <Image source={require('../assets/mapa.png')} />
        ),
    }}
/>
<Tab.Screen name="Mujeres" component={MujeresCombinadas} ... />
<Tab.Screen name="Rutas" component={Rutas} ... />
<Tab.Screen name="RA" component={AR} ... />
<Tab.Screen name="Perfil" component={PerfilUsuario} ... />
</Tab.Navigator>

```

Cada pantalla principal se implementa como un componente independiente, facilitando la escalabilidad y el mantenimiento.

Además, la navegación global de la app se gestiona mediante un **Stack Navigator** definido en `NavigationApp.js`, que permite acceder a pantallas como el login, registro, detalle de lugar, historial de visitas o filtros, además de la pantalla principal con pestañas. El componente `App.js` monta el contexto global de filtros y notificaciones, y envuelve toda la navegación de la aplicación.

6.2.4. Gestión de usuarios y autenticación

La autenticación de usuarios se realiza mediante peticiones a la API del backend. El login y el registro almacenan el token de autenticación en `AsyncStorage` para su uso en futuras peticiones.

Listado 6.37: Ejemplo de login en `Login.js`

```

const validateLogin = async () => {
    try {
        const response = await fetch('http://192.168.1.44:8000/api/api-token-auth',
            {
                method: 'POST',
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify({ username, password }),
            });
        if (response.ok) {
            const data = await response.json();
            await AsyncStorage.setItem('token', data.token);
            navigation.navigate('Main');
        } else {
            Alert.alert('Error', 'Usuario o contraseña incorrectos');
        }
    } catch (error) {
        Alert.alert('Error', 'No se pudo conectar con el servidor');
    }
};

```

El perfil de usuario permite cambiar el nombre de usuario y cerrar sesión, eliminando el token almacenado.

6.2.5. Visualización de mujeres y lugares

La pantalla `Mujeres.js` muestra una lista de lugares destacados asociados a mujeres relevantes, permitiendo filtrar por área, visitados y búsqueda por nombre.

Listado 6.38: Filtrado y renderizado de lugares

```
const filteredLugares = allLugares
  .filter(lugar => selectedSection === 'Todas' || lugar.area ===
    selectedSection)
  .filter(lugar => selectedVisited === 'Todas' || (selectedVisited === 'Visitadas' ? lugar.visited : !lugar.visited))
  .filter(lugar => !searchTerm || lugar.nombre.toLowerCase().includes(
    searchTerm.toLowerCase()));
```

Cada tarjeta de lugar muestra la imagen, nombre, área, distancia y estado de visita, y permite acceder al detalle.

6.2.6. Detalle de lugar y gestión de visitas

La pantalla `Detail.js` muestra la información completa de un lugar, incluyendo la mujer asociada, descripción, dirección, fechas, imagen y estado de visita. Permite marcar el lugar como visitado o no visitado, sincronizando el estado con el backend mediante una petición HTTP autenticada. Además, la pantalla ofrece funcionalidades avanzadas para mejorar la experiencia del usuario:

- **Compartir información del lugar:** El usuario puede compartir el lugar y su imagen a través de las opciones nativas del dispositivo. Se genera un mensaje con los datos relevantes y, si existe, se adjunta la imagen descargada temporalmente.
- **Visualización de imágenes ampliadas:** Al pulsar sobre la imagen de la mujer o del lugar, se muestra la imagen en grande en un modal.
- **Visualización de información enriquecida:** Se muestran áreas de investigación como chips, fechas, ámbito, descripción, dirección, un mapa con la localización exacta y la distancia al usuario.

A continuación se muestra un ejemplo de la función para marcar un lugar como visitado o no visitado:

Listado 6.39: Marcar como visitado

```
const toggleVisited = async () => {
  const updatedLugar = { ...lugar, visited: !lugar.visited };
  navigation.setParams({ lugar: updatedLugar });
  const token = await AsyncStorage.getItem('token');
  if (!token) return;
  if (!lugar.visited) {
    // Marcar como visitado
    try {
      const response = await fetch('http://192.168.1.44:8000/api/visit-
        lugar/', {
```

```

        method: 'POST',
        headers: { 'Content-Type': 'application/json', 'Authorization': `Token ${token}`,
        body: JSON.stringify({ lugar_id: lugar.id }),
      });
      const data = await response.json().catch(() => ({}));
      if (!response.ok) {
        throw new Error(`Error ${response.status}: ${JSON.stringify(data)} `);
      }
    } catch (error) {
      console.error('Error registrando visita:', error);
    }
  } else {
    // Marcar como no visitado (eliminar del historial)
    try {
      await fetch('http://192.168.1.44:8000/api/visited-lugares/', {
        method: 'PUT',
        headers: { 'Content-Type': 'application/json', 'Authorization': `Token ${token}`,
        body: JSON.stringify({ lugar_id: lugar.id }),
      });
    } catch (error) {
      console.error('Error eliminando visita:', error);
    }
  }
}
};

```

Y la función para compartir la información del lugar:

Listado 6.40: Función para compartir la información del lugar

```

const compartirLugar = async () => {
  try {
    let message = "Echa un vistazo a este lugar de la aplicación Dona'm Món!\n\n";
    message += `Nombre del lugar: ${lugar.nombre}\n`;
    if (lugar.mujer_nombre) message += `Mujer destacada: ${lugar.mujer_nombre}\n`;
    if (lugar.mujer_descripcion) message += `Sobre ella: ${lugar.
      mujer_descripcion}\n`;
    if (lugar.direccion) message += `Dirección: ${lugar.direccion}\n`;
    if (lugar.descripcion) message += `Descripción: ${lugar.descripcion}\n`;
    let options = { message };
    if (lugar.foto) {
      // Descargar la imagen a un archivo temporal
      const fileUri = FileSystem.cacheDirectory + 'lugar.jpg';
      const downloadRes = await FileSystem.downloadAsync(lugar.foto,
        fileUri);
      if (downloadRes.status === 200) {
        options = { ...options, url: downloadRes.uri };
      }
    }
    await Share.share(options);
  }
};

```

```

    } catch (error) {
        console.error('Error al compartir el lugar:', error);
    }
};

```

6.2.7. Mapas y geolocalización

La pantalla `Mapa.js` utiliza `react-native-maps` para mostrar todos los lugares en un mapa, con iconos personalizados y colores según el estado de visita. Permite filtrar lugares y ver detalles al pulsar sobre un marcador.

Listado 6.41: Marcadores en el mapa

```

<MapView>
    {filteredLugares.map(lugar => (
        <Marker
            key={lugar.id}
            coordinate={{ latitude: lugar.latitud, longitude: lugar.longitud }}
            pinColor={lugar.visitado ? '#43a047' : '#5f68c4'}
        >
            <Callout>
                <Text>{lugar.nombre}</Text>
                <Text>{lugar.mujer_nombre}</Text>
            </Callout>
        </Marker>
    )));
</MapView>

```

La ubicación del usuario se obtiene con `expo-location`, permitiendo calcular distancias y mostrar solo lugares cercanos si se desea.

6.2.8. Gestión de rutas temáticas

La pantalla `Rutas.js` y `RutaDetail.js` permiten visualizar rutas temáticas, ver los lugares que las componen y marcar cada lugar como visitado mediante escaneo de QR. El progreso de la ruta se muestra visualmente y se puede reiniciar la ruta si se desea.

Listado 6.42: Escaneo de QR y registro de visita

```

const handleBarcodeScanned = async ({ data }) => {
    const lugar = ruta.lugares.find(l => l.nombre.toLowerCase() === data.toLowerCase());
    if (lugar) {
        await fetch('http://192.168.1.44:8000/api/visit-lugar-ruta/', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json', 'Authorization': `Token ${token}` },
            body: JSON.stringify({ ruta_id: ruta.id, lugar_id: lugar.id }),
        });
        setScanMsg('Lugar de ruta marcado como visitado!');
        await fetchVisitedRuta();
    } else {
        setScanMsg('QR no corresponde a ningun lugar de la ruta.');
    }
}

```

```
};
```

6.2.9. Historial de lugares y rutas visitadas

Las pantallas `HistorialLugares.js` y `HistorialRutas.js` muestran el historial de lugares y rutas visitadas por el usuario, permitiendo borrar el historial si se desea.

Listado 6.43: Borrado de historial de rutas

```
const handleClearHistory = async () => {
  const token = await AsyncStorage.getItem('token');
  for (const ruta of rutasCompletadas) {
    await fetch(`http://192.168.1.44:8000/api/visited-lugares-ruta/?ruta_id=${ruta.id}`, {
      method: 'DELETE',
      headers: { 'Authorization': `Token ${token}` },
    });
  }
  setRutasCompletadas([]);
  Alert.alert('Éxito', 'Historial de rutas completadas borrado.');
};
```

6.2.10. Notificaciones y proximidad

La app utiliza `expo-notifications` y `expo-location` para enviar notificaciones cuando el usuario se acerca a un lugar destacado. El componente `NotificacionProximidad.js` gestiona esta lógica, comprobando la distancia a los lugares y mostrando alertas contextuales.

Listado 6.44: Notificación de proximidad

```
if (dist !== null && dist <= 3 && !notifiedIds.current.has(lugar.id)) {
  await Notifications.scheduleNotificationAsync({
    content: {
      title: 'Estás cerca de un lugar destacado!',
      body: `Te encuentras a ${dist.toFixed(2)} km de ${lugar.nombre} (${lugar.mujer_nombre})`,
    },
    trigger: null,
  });
  notifiedIds.current.add(lugar.id);
}
```

6.2.11. Realidad aumentada

La pantalla `AR.js` permite acceder a experiencias de realidad aumentada asociadas a ciertos lugares, abriendo el visor web correspondiente si el usuario está cerca del lugar.

Listado 6.45: Acceso a AR

```
if (distance <= 3) {
  openURL(lugar.ar_url);
}
```

6.2.12. Gestión de estilos y diseño

Los estilos se gestionan en archivos como `styles.js` y `styles-perfil.js`, utilizando `StyleSheet` de React Native para mantener la coherencia visual y facilitar la personalización.

Listado 6.46: Ejemplo de estilos en `styles.js`

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#f9efef8',
    alignItems: 'center',
    justifyContent: 'center',
  },
  filterButton: {
    backgroundColor: '#7196e4',
    paddingVertical: 0,
    paddingHorizontal: 30,
    borderRadius: 10,
    marginTop: 23,
    alignSelf: 'center',
    width: width + 20,
    height: 39,
    alignItems: 'center',
  },
  // otros estilos...
});
```

6.2.13. Gestión de almacenamiento local

Se utiliza `AsyncStorage` para guardar el token de usuario, preferencias y otros datos persistentes, asegurando que la sesión se mantenga entre reinicios de la app.

Listado 6.47: Uso de `AsyncStorage`

```
await AsyncStorage.setItem('token', data.token);
const token = await AsyncStorage.getItem('token');
```

6.2.14. Integración con la API y manejo de errores

Todas las peticiones a la API del backend se realizan mediante `fetch` o `axios`, gestionando los errores y mostrando mensajes claros al usuario en caso de fallo.

Listado 6.48: Manejo de errores en peticiones

```
try {
  const response = await fetch('http://192.168.1.44:8000/api/visited-lugares
  ', { /* ... */ });
  if (!response.ok) throw new Error('Error al cargar lugares visitados');
  const data = await response.json();
  setVisitedCount(data.length);
} catch (error) {
  Alert.alert('Error', 'No se pudo cargar el historial');
}
```

6.2.15. Pruebas y depuración

Durante el desarrollo, se han utilizado logs en consola y mensajes de alerta para depurar la lógica de la app y asegurar el correcto funcionamiento de todas las funcionalidades.

Listado 6.49: Ejemplo de logs y alertas

```
console.log('Detalle lugar:', lugar);
Alert.alert('Éxito', 'Nombre de usuario actualizado');
```

Capítulo 7

Conclusiones

7.1. Revisión de costes

7.2. Conclusiones

7.3. Trabajo futuro

Hablar con Por Ti Mujer y la Asamblea de Mujeres de Valencia por si les interesa que la aplicación sea un recurso para sus rutas. Poder descargar los datos de la base de datos en formato CSV o JSON para su uso en otros proyectos. Poder añadir nuevos puntos de interés a la base de datos desde la aplicación móvil. Poder añadir nuevos puntos de interés a la base de datos desde el panel de administración. Página de contacto para que los usuarios puedan enviar sugerencias o comentarios sobre la aplicación. Tener un apartado de preguntas frecuentes (FAQ) para resolver dudas comunes de los usuarios. Tener una sección de noticias o novedades sobre la aplicación y las mujeres que aparecen en ella. Tener una página para administrar todo lo relaciona con la aplicación. Expandir la aplicación a otras ciudades de España o del mundo.

Apéndice A

Apéndice

A.1. Ejemplos del lenguaje de marcado Latex

Antes de empezar a escribir la memoria debes leerte, al menos, los dos primeros capítulos del clásico de Tobias Oetker [?]¹.

El Apéndice A.1 recoge algunos ejemplos de edición con LATEX.

This document is an example of BibTeX using in bibliography management. Three items are cited: *The LATEX Companion* book [?], the Einstein journal paper [?], and the Donald Knuth's website [?]. The LATEX related items are [?, ?]².

Texto en el párrafo 1.

Texto en el párrafo 2.

Texto en el párrafo 3.

■ Consideración 1

■ Consideración 2

1. Punto 1

2. Punto 2

A continuación se muestra una ecuación:

$$\int_0^1 \frac{1}{x^2 + 1} dx$$

Podemos incluir imágenes en formato: png, pdf o jpg.

En la figura A.1 se muestra un diagrama realizado con <https://www.yworks.com/products/yed>:

¹La última versión puede encontrarse en <http://tobi.oetiker.ch/lshort/lshort.pdf>.

²Esto está tomado de https://www.overleaf.com/learn/latex/Bibliography_management_with_bibtex

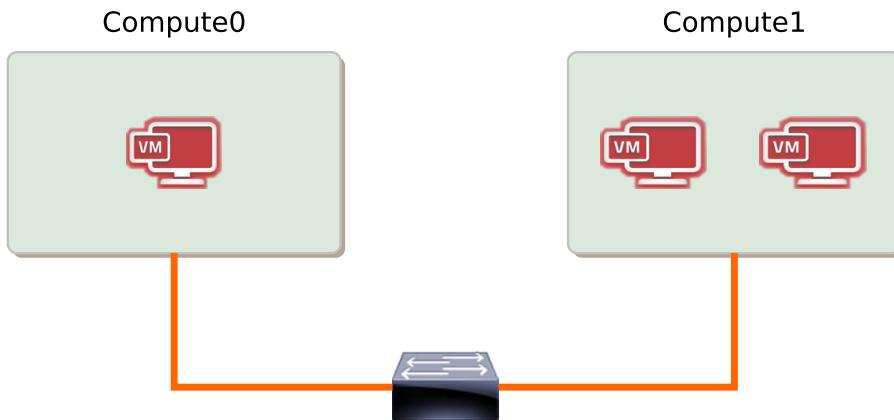
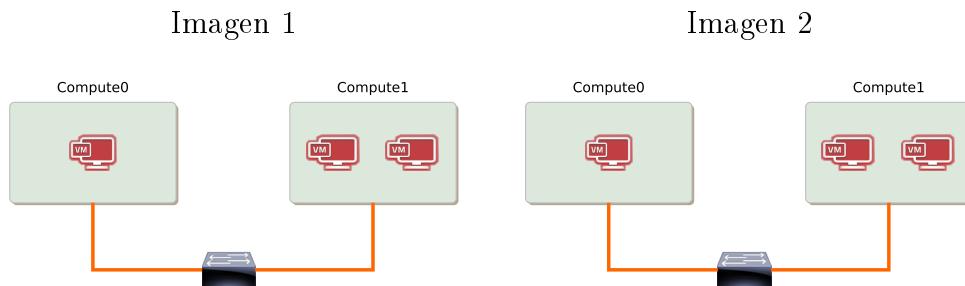


Figura A.1: Esta es una figura que latex decide donde colocar (floating) en el documento.



Este es un ejemplo de una tabla:

Columna 1	Columna 2
1	2

O la misma tabla centrada:

Columna 1	Columna 2
1	2

Para generar el fichero PDF:

```
pdflatex ejemplo-memoria.tex
bibtex ejemplo-memoria
pdflatex ejemplo-memoria.tex
```

También se puede usar `latexmk` que automáticamente regenera la bibliografía.

```
latexmk -pdf ejemplo-memoria.tex
```

Bibliografía

- [1] UNESCO. Science report: The race against time for smarter development. <https://unesdoc.unesco.org/ark:/48223/pf0000377433>, 2021. United Nations Educational, Scientific and Cultural Organization.
- [2] Ministerio de Ciencia e Innovación. Científicas en cifras 2021: Estadísticas e indicadores de la (des)igualdad de género en la formación y profesión científica. https://www.ciencia.gob.es/dam/jcr:dc8689c4-2c47-4aaf-97ce-874bd0b5a081/Cientificas_en_Cifras_2021.pdf, 2022. Secretaría General de Investigación.
- [3] Organización de las Naciones Unidas. Transformar nuestro mundo: la agenda 2030 para el desarrollo sostenible. <https://www.un.org/sustainabledevelopment/es/agenda-2030/>, 2015. Consultado el 2 de junio de 2025.
- [4] ONU Mujeres. Objetivo 5: Lograr la igualdad entre los géneros y empoderar a todas las mujeres y niñas. <https://www.unwomen.org/es/news/in-focus/women-and-the-sdgs/sdg-5-gender-equality>, 2024. Consultado el 2 de junio de 2025.
- [5] ONU-Hábitat. Objetivo 11: Lograr que las ciudades sean más inclusivas, seguras, resilientes y sostenibles. <https://www.un.org/sustainabledevelopment/es/cities/>, 2024. Consultado el 2 de junio de 2025.
- [6] Jon Agar. Constant touch: A global history of the mobile phone, 2004. ISBN 978-1840465413.
- [7] Joel West and Jason Dedrick. Globalization and the evolution of the mobile phone industry. *International Journal of Technology and Globalisation*, 5(1/2):65–80, 2010.
- [8] Jamie Murphy. Mobile marketing: The future of marketing, or just another fad? *International Journal of Mobile Marketing*, 9(1):47–59, 2014.
- [9] StatCounter. Mobile operating system market share worldwide. <https://gs.statcounter.com/os-market-share/mobile/worldwide>, 2024.
- [10] Statista. Number of smartphone users worldwide from 2016 to 2024. <https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/>, 2024.
- [11] AndroidPolice. Android versus ios software updates revisited, 2017.
- [12] Gartner. Market guide for multiexperience development platforms, 2022.
- [13] Data.ai. State of mobile 2023, 2023.
- [14] App Annie. State of mobile report 2022, 2022.

- [15] David Parsons and Kathryn MacCallum. *Digital Transformation and Public Services: Societal Impacts in Sweden and Beyond*. Springer, Cham, 2020.
- [16] Ronald T. Azuma. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385, 1997. <https://doi.org/10.1162/pres.1997.6.4.355>.
- [17] Mark Billinghurst, Adrian Clark, and Gun Lee. A survey of augmented reality. *Foundations and Trends in Human–Computer Interaction*, 8(2–3):73–272, 2015. <https://doi.org/10.1561/1100000049>.
- [18] Steven Feiner, Blair MacIntyre, and Dorée Seligmann. Knowledge-based augmented reality. *Communications of the ACM*, 36(7):53–62, 1993. <https://doi.org/10.1145/159544.159587>.
- [19] Hirokazu Kato and Mark Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR’99)*, pages 85–94, 1999. <https://doi.org/10.1109/IWAR.1999.803809>.
- [20] Daniel Wagner and Dieter Schmalstieg. First steps towards handheld augmented reality. In *Proceedings of the 7th IEEE International Symposium on Wearable Computers*, pages 127–135, 2003. <https://doi.org/10.1109/ISWC.2003.1241409>.
- [21] Apple. Arkit: Augmented reality for ios. <https://developer.apple.com/augmented-reality/>, 2017.
- [22] Google. Arcore: Google developers. <https://developers.google.com/ar>, 2018.
- [23] J. Paavilainen, J. Hamari, J. Stenros, and J. Kinnunen. The pokémon go experience: A location-based augmented reality mobile game goes mainstream. In *Proceedings of CHI 2017*, 2017.
- [24] Microsoft. Hololens product page. <https://www.microsoft.com/en-us/hololens>, 2016.
- [25] D. W. F. van Krevelen and R. Poelman. A survey of augmented reality technologies, applications and limitations. *The International Journal of Virtual Reality*, 9(2):1–20, 2010.
- [26] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *ISMAR 2007*, 2007.
- [27] Hiroshi Ishii and Brygg Ullmer. Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 234–241, 1997. <https://doi.org/10.1145/258549.258715>.
- [28] M. Billinghurst and H. Kato. Collaborative mixed reality. In *ISMAR 2002*, 2002.
- [29] Apple. Introducing apple vision pro. <https://www.apple.com/apple-vision-pro/>, 2024.
- [30] Women’s legacy. proyecto europeo cofinanciado por la unión europea, 2022. Conselleria d’Educació, Cultura i Esport de la Generalitat Valenciana.
- [31] Dones de ciència: Murals que inspiren, 2020. Las Naves & UPV.
- [32] Callejero con perspectiva de género: iniciativas y actuaciones, 2023. Ayuntamiento de València.

- [33] Asociación Por Ti Mujer. Ruta violeta de mujeres en valencia, 2023. Recuperado de <https://asociacionportimujer.org/ruta-violeta-de-mujeres-en-valencia>.
- [34] Assemblea Feminista de València. Ruta pels espais del patronato de protección a la mujer, 2023. Recuperado de <https://www.feministas.org/ruta-pels-espais-del-patronato-de.html>.
- [35] Assemblea Feminista de València. Ruta de memorias lesbianas en valencia: visibilización y lucha feminista, 2024. Recuperado de <https://www.levantemv.com/valencia/2024/10/27/mapa-memorias-lesbianas-valencia-lesbianismo-lgtbi-assemblea-feminista-feminismo-110410679.html>.
- [36] R. Guijarro-Garzón et al. Pioneras de la medicina: Concepción aleixandre y la visibilización de la mujer en la ciencia. *Revista de Historia de la Medicina*, 12(2):134–148, 2021.
- [37] Carmen Alborch. *Solas*. Espasa Calpe, 1999.
- [38] Amelia González. *Clara Campoamor: el sufragio femenino en España*. Cátedra, 2006.
- [39] Carmen Rueda Ramos. *Isabel de Villena y la espiritualidad femenina en la Valencia del siglo XV*. Tirant lo Blanch, 2013.
- [40] J. Martínez Pérez. *Médicas en Valencia: mujeres que abrieron camino*. Editorial UV, 2019.
- [41] Vandana Shiva. *Earth Democracy: Justice, Sustainability, and Peace*. South End Press, 2005.
- [42] Google Developers. Google maps platform documentation. <https://developers.google.com/maps/documentation>.
- [43] Mapbox. Mapbox documentation. <https://docs.mapbox.com/>.
- [44] OpenStreetMap Foundation. Openstreetmap wiki. <https://wiki.openstreetmap.org/>.
- [45] React Native Maps Contributors. react-native-maps: React native mapview component for ios + android. <https://github.com/react-native-maps/react-native-maps>.
- [46] Django Software Foundation. Django documentation, 2024.
- [47] Ramez Elmasri and Shamkant B. Navathe. *Fundamentals of database systems*. Pearson, 7th edition, 2017.
- [48] Michael Stonebraker and Rick Cattell. 10 rules for scalable performance in ‘simple operation’ datastores. *Communications of the ACM*, 54(6):72–80, 2011.
- [49] Antonio Mele Vincent. *Mastering Django: Core*. Independently published, 2020.
- [50] Oracle Corporation. Mysql 8.0 reference manual, 2023.
- [51] MariaDB Foundation. Mariadb vs mysql, 2023.
- [52] Inc. MongoDB. Mongodb manual, 2024.
- [53] Django Software Foundation. Using sqlite with django, 2024.
- [54] Apache Software Foundation. Apache cassandra documentation, 2023.

- [55] Glassdoor. Sueldo de jefe de proyecto. https://www.glassdoor.es/Sueldos/jefe-de-proyectos-sueldo-SRCH_KO0,17.htm, 2025. Consultado el 11 de junio de 2025.
- [56] Glassdoor. Sueldo de desarrollador react native. https://www.glassdoor.es/Sueldos/react-native-developer-sueldo-SRCH_KO0,22.htm, 2025. Consultado el 11 de junio de 2025.
- [57] Glassdoor. Sueldo de desarrollador python/django. https://www.glassdoor.es/Sueldos/django-developer-sueldo-SRCH_KO0,16.htm, 2025. Consultado el 11 de junio de 2025.
- [58] Glassdoor. Sueldo de qa tester. https://www.glassdoor.es/Sueldos/qa-tester-sueldo-SRCH_KO0,9.htm, 2025. Consultado el 11 de junio de 2025.
- [59] Glassdoor. Sueldo de modelador 3d. https://www.glassdoor.es/Sueldos/modelador-de-3d-sueldo-SRCH_KO0,15.htm, 2025. Consultado el 11 de junio de 2025.
- [60] Glassdoor. Sueldo de diseñador gráfico. https://www.glassdoor.es/Sueldos/disenador-grafico-sueldo-SRCH_KO0,17.htm, 2025. Consultado el 11 de junio de 2025.
- [61] Glassdoor. Sueldo de desarrollador de realidad aumentada/virtual. https://www.glassdoor.es/Sueldos/virtual-reality-developer-sueldo-SRCH_KO0,25.htm, 2025. Consultado el 13 de junio de 2025.