

MEMORIA ESCRITA DEL PROYECTO

CFGS Desarrollo de Aplicaciones Multiplataforma

Desarrollo de una Aplicación Android para Guardar Senderos

Autor: Eduardo Fernández Aznar

Tutor: Mario Gago

Fecha de entrega: 12/02/2023

Convocatoria: 2S – 22/23

Documentos del proyecto:

<https://drive.google.com/drive/folders/1RSzvfcoXiXNCnPPfQfI9f7IBKrCW1eo8?usp=sharing>

Enlace GitHub: <https://github.com/eduferaz/AppSendero.git>



Índice de contenidos

1. INTRODUCCIÓN	3
1.1. Motivación	3
1.2. Abstract.....	4
1.3. Objetivos propuestos (generales y específicos)	5
2. METODOLOGÍA USADA	6
3. TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS EN EL PROYECTO	9
4. ESTIMACIÓN DE RECURSOS Y PLANIFICACIÓN	11
5. ANÁLISIS DEL PROYECTO	13
6. DISEÑO DEL PROYECTO	21
7. DESPLIEGUE Y PRUEBAS	28
8. CONCLUSIONES	33
9. VÍAS FUTURAS.....	35
10. BIBLIOGRAFÍA/WEBGRAFÍA	36
11. ANEXOS	38

1. Introducción (3-4 Páginas)

Este documento presenta el proyecto de investigación y desarrollo realizado por Eduardo Fernández Aznar como parte de su Trabajo Fin de Ciclo. Con esta culminación, el estudiante espera completar exitosamente el Ciclo Superior en Desarrollo de Aplicaciones Multiplataformas en el centro ILERNA.

El presente proyecto, *“Desarrollo de una Aplicación Android para Guardar Senderos”*, a la cual he asignado el título de “SENDERISMO SIERRA MORENA”, como nombre de la aplicación, concede la explicación detallada del proceso de programación, diseño y puesta a punto de una app para dispositivos móviles Android

La idea de “SENDEROS SIERRA MORENA” es tener una aplicación, más localista, para los habitantes de la comarca, que quieran conocer mejor la zona o que ya la conozcan y quieran compartir sus conocimientos, de modo que el usuario pueda ir aglutinando en una misma aplicación todos los senderos que recorren la Sierra Morena Sevillana, guarden sus senderos favoritos o aquellos que ya hayan realizado.

Además, el usuario podrá añadir información del sendero, y lo más importante, su localización, esto se conseguirá gracias al sistema GPS del smartphone y a la aplicación de mapas de Google, cuya API se integrará en la aplicación.

1.1. Motivación

En la sociedad actual, la tecnología juega un papel cada vez más importante en nuestra vida diaria, facilitando muchas tareas y mejorando la calidad de vida, especialmente en cuanto al ocio y al tiempo libre se refiere.

Mi motivación como alumno de un Ciclo Superior de Desarrollo de Aplicaciones multiplataformas, es aplicar mis conocimientos adquiridos en los diferentes módulos del ciclo a lo largo de los 2 años, a un proyecto práctico. Mi objetivo es familiarizarme con el uso de servicios externos como Google Maps API, así como profundizar en el del lenguaje de programación Kotlin y el uso del IDE de Android Studio. Además, este proyecto representa un desafío en cuanto a la gestión y la organización de un proyecto, el manejo y los recursos

Como amante del senderismo y la bicicleta de montaña, Licenciado en Ciencias Ambientales y habitante de la Comarca de la Sierra Morena Sevillana, que destaca por su riqueza natural y es considerada Parque Natural, mi objetivo principal es crear una aplicación Android que

satisfaga mis necesidades como senderista. Mi visión es que esta aplicación sea útil para el entorno rural, ya que contribuye al turismo y permite conocer la comarca a través de sus elementos patrimoniales y etnográficos.

En definitiva, mi aplicación se trata de una idea original y creativa que cubre mi necesidad de disfrutar de la naturaleza y el senderismo en una zona con gran valor ecológico y cultural. Para mí, enfrentarme a este desarrollo ha sido un reto técnico y profesional que va a implicar el aprendizaje y uso de diversas herramientas y tecnologías para conseguir que mi aplicación Android sea funcional, atractiva y segura. Se podría decir que se trata de una oportunidad de aprendizaje y mejora continua para poner en práctica los conocimientos adquiridos durante el ciclo formativo y ampliarlos con nuevas competencias y habilidades.

En resumen, además de aprender y mejorar mis conocimientos técnicos, mi motivación es contribuir a favorecer la actividad del senderismo en una tierra donde muchos de esos senderos de acceso público se encuentran ocultos entre las fincas privadas y en el futuro, según vaya mejorando la aplicación, los conocedores del terreno podrán compartir con otras personas toda la información sobre cualquier sendero, dando a conocer aquellos más escondidos.

1.2. Abstract

The project, "Development of an Android Application to Save Hiking Trails", which the student has titled "SENDERISMO SIERRA MORENA" as the name of the application, provides a list of existing trails, a description of them, the duration of the route, and its difficulty, helping the user to search for various types of trails for hiking.

The idea of "SENDEROS SIERRA MORENA" is to have a more localized application for residents of the region who want to better understand the area or who already know it and want to share their knowledge. Thus, the user can aggregate in the same application all the trails that cross the Sevillian Sierra Morena, save their favorite trails, or those they have already completed.

I believe that this type of application would be useful for the rural environment, since we can consider that hiking contributes to rural tourism and allows us to get to know the region little by little, through its heritage and ethnographic elements.

1.3. Objetivos propuestos (generales y específicos)

El objetivo general de “SENDEROS SIERRA MORENA” es desarrollar una aplicación móvil para Android que permita a los usuarios acceder a la información y localización de senderos por municipio. La aplicación ofrecerá al usuario una experiencia de uso fácil e intuitiva que permita a los usuarios guardar, modificar, eliminar, marcar como favorito o marcar como completado uno o muchos senderos

Los objetivos específicos serían:

- Establecer un registro de usuario, para que cada usuario pueda iniciar sesión con sus datos
- Implementar una base de datos local para almacenar los municipios y senderos guardados por el usuario.
- Desarrollar una interfaz de usuario clara y sencilla que muestre la lista de senderos disponibles y permita a los usuarios interactuar con ella.
- Integrar la ubicación de los senderos en Google Maps para facilitar la navegación y la búsqueda.
- Permitir al usuario guardar información adicional para cada sendero, como una foto, una descripción, etc. Además de relacionar cada sendero con el municipio al que pertenece
- Implementar la funcionalidad de marcar como favorito o completado para los senderos y proporcionar una forma sencilla de acceder a la lista de senderos favoritos o completados.
- Proporcionar la opción de modificar o eliminar los senderos guardados por el usuario.
- Garantizar la seguridad y la privacidad de los datos del usuario, implementando medidas de seguridad adecuadas y siguiendo las mejores prácticas en el desarrollo de aplicaciones móviles

2. Metodología usada (2-4 páginas)

Para realizar este proyecto he escogido una metodología ágil, son las más usadas a la hora de realizar y gestionar proyectos. De esta manera puedo observar el progreso del proyecto de manera constante, además la metodología ágil se adapta fácilmente a los cambios del proyecto, esto me resultará muy útil ya que puedo encontrarme con problemas o cambios inesperados en el proceso de desarrollo. Finalmente, uno de los motivos principales por lo cual he escogido una metodología ágil es que me permite trabajar de forma autónoma y tener cierta flexibilidad para trabar en el proyecto a mi ritmo, organizando el tiempo de manera eficiente para cumplir con los plazos.

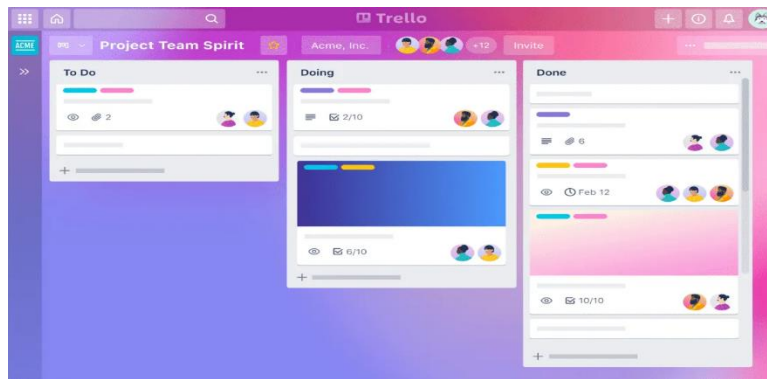
Para mi proyecto es muy útil porque en lugar de planificar todo el proyecto de antemano y seguir un plan rígido, en la metodología ágil se divide el proyecto en iteraciones más pequeñas llamadas sprints, de este modo, puedo gestionar el tiempo, los recursos y los riesgos del proyecto, entregando resultados funcionales en cada iteración

Cada sprint tiene un objetivo específico y se enfoca en entregar un conjunto de características o funcionalidades que puedan ser utilizadas por la aplicación. Después de cada sprint, se revisa el progreso y se hacen ajustes para el siguiente sprint, basándose en las necesidades de mi aplicación. (Zendesk, 2023).

Cómo metodología ágil he escogido Kanban, ya que estoy familiarizado con ella. Kanban es una metodología de gestión de proyectos que se utiliza para visualizar el flujo de trabajo en progreso y maximizar la eficiencia y la productividad, más concretamente he utilizado la aplicación tanto de escritorio como para Android de Trello, una herramienta con la que estoy bastante familiarizado y que me resulta muy útil

En Trello, se utiliza un tablero Kanban que consiste en columnas que representan diferentes etapas del proceso de trabajo, como “Por hacer”, “En proceso” y “Terminado”. Posteriormente puedo mover las tarjetas a lo largo del tablero a medida que avanzo en el trabajo.

Kanban también se enfoca en limitar el trabajo en progreso, lo que significa que se establece un límite en la cantidad de tarjetas que se pueden tener en cada columna, esto ayuda a evitar la sobrecarga de trabajo y me obliga a centrarme en completar una tarea antes de comenzar otra. (Kanbanize, 2023)



1 Imagen obtenida de Trello.com

Como he dicho anteriormente, la herramienta utilizada para poder llevar a cabo esta metodología ha sido Trello, al ser la herramienta que mejor se adapta a mi proyecto y con la que estoy familiarizada.

Es una herramienta muy amigable, Trello utiliza tarjetas y tableros personalizables que me permite organizar las tareas de manera visual. Esto hace que sea fácil de entender y seguir el flujo de trabajo del proyecto. Me proporciona una vista general del progreso, de esta manera puedo monitorizar y actualizar fácilmente el estado de las tareas y ver como se está avanzando en el proyecto. (Solomon, 2022)

En cuanto al ciclo de vida de mi aplicación, será el siguiente.

- **Planificación:** en esta etapa, se define el alcance del proyecto y se establecen los objetivos y requisitos de mi aplicación. Se identifican los recursos necesarios y el cronograma del proyecto. En esta etapa, se realiza la planificación de las iteraciones y se establecen los requisitos de cada uno de ellos.
- **Diseño:** En esta etapa, se define la arquitectura de mi aplicación, se realiza el diseño de la interfaz de usuario y se establece la estructura de la base de datos. También se definen las funcionalidades principales de la aplicación y se crean los diagramas de flujo y los diagramas de clase.
- **Desarrollo:** En esta etapa, se crea el código fuente de la aplicación al mismo tiempo que se va elaborando la memoria escrita. Se desarrolla cada módulo de la aplicación y se van integrando los componentes. Además, vamos realizando pruebas para cada iteración, para controlar el comportamiento de la aplicación.
- **Pruebas:** En esta etapa, se realizan pruebas de calidad para asegurarse de que la aplicación funciona correctamente y cumple con los requisitos establecidos en la etapa

de planificación. Se realizan pruebas de integración, pruebas de sistema y pruebas de aceptación para garantizar que la aplicación esta lista para su entrega.

- Seguimiento y control: Durante esta etapa se monitoriza el progreso del proyecto y se realiza un seguimiento del tiempo y recursos para asegurarse de que se cumplan los objetivos del proyecto y la aplicación sea totalmente funcional
- Cierre: En esta etapa se finaliza el proyecto y se entregan proyecto y aplicación al centro. También se realiza una evaluación del proyecto para determinar si se cumplió con los objetivos establecidos en la etapa de planificación y se identifican lecciones aprendidas y vías futuras para mi aplicación.

3. Tecnologías y herramientas utilizadas en el proyecto

(2-3 páginas)

En este apartado definiré las herramientas y tecnologías usadas para el desarrollo de una aplicación Android:

- **Git:** Git es un software que se encarga del control de versiones de manera distribuida, el cual ha sido utilizado para realizar una réplica del proyecto en un repositorio completo de control de versiones. (Jacobs, 2023). Este repositorio me permite trabajar sin conexión o de forma remota con facilidad y, a continuación, sincronizar mi copia del repositorio con la copia en el servidor y tener el proyecto guardado y accesible desde cualquier lugar.
- **Android:** Android es un sistema operativo móvil desarrollado por Google que se utiliza en una amplia variedad de dispositivos, como smartphones, tabletas, televisores y relojes inteligentes. Proporciona una plataforma para que los desarrolladores de aplicaciones creen y distribuyan aplicaciones móviles a través de la tienda de aplicaciones de Google Play. Android también ofrece una interfaz de usuario personalizable y una amplia gama de características y funcionalidades para los usuarios de dispositivos móviles. (Adeva, 2023). Me he decantado por este sistema operativo por su sencillez y porque es el sistema operativo más usado en la actualidad.
- **Android Studio:** He utilizado Android Studio porque es el entorno de desarrollo integrado (IDE, por sus siglas en inglés) oficial de Google diseñado específicamente para desarrollar aplicaciones para el sistema operativo Android. Proporciona una serie de herramientas y características para simplificar el proceso de desarrollo de aplicaciones móviles, como la edición de código, la depuración, la compilación y la emulación de dispositivos virtuales. (Wikipedia, 2023). Me he decantado por Android Studio ya que incluye plantillas de proyecto predefinidas y una amplia documentación para ayudar a los desarrolladores a crear aplicaciones de alta calidad para Android.
- **Kotlin:** Es un lenguaje de programación de alto nivel y orientado a objetos que se ejecuta en la máquina virtual de Java (JVM) y también puede compilarse en código nativo para plataformas específicas. Fue desarrollado por JetBrains y presentado en 2011 como una alternativa más moderna y concisa al lenguaje Java, aunque se integra sin problemas con código Java existente. Kotlin es interoperable con Java, lo que significa que los desarrolladores pueden utilizar código Kotlin y Java juntos en un

proyecto. Además, Kotlin incluye características adicionales que no se encuentran en Java, como la seguridad contra nulos, el soporte de extensiones de funciones y la inferencia de tipos mejorada, lo que hace más fácil de leer y escribir. Kotlin se ha convertido en un lenguaje de programación popular para el desarrollo de aplicaciones Android, aunque también se utiliza para la creación de aplicaciones de servidor, aplicaciones web y otro tipo de proyectos. (Czura, s.f.)

- XML: es el acrónimo de Extensible Markup Language, es decir, es un lenguaje de marcado que se utiliza para almacenar y transportar datos. En el contexto de Android se utiliza para el diseño de la interfaz de usuario, cada Activity se define en archivos XML, utilizando elementos como vistas, diseños, colores y estilos. XML permite una estructuración jerárquica y una fácil legibilidad del código, lo que hace que sea más fácil para los desarrolladores de Android crear y mantener las interfaces de usuario de sus aplicaciones. Además, los recursos de una aplicación de Android, como las cadenas de texto, las imágenes y los archivos de sonido, se definen en archivos XML separados, esto permite que los recursos sean reutilizados en toda la aplicación y que se puedan mantener y modificar fácilmente. (Souza, 2019)
- Room: Es una biblioteca de persistencia de datos de Android que proporciona una capa de abstracción sobre SQLite, lo que hace que sea más fácil para los desarrolladores de Android trabajar con bases de datos. Room utiliza anotaciones de Java para definir la estructura de la base de datos. Room está diseñado para trabajar bien con otras bibliotecas de Android como LiveData y ViewModel, lo que permite que los desarrolladores implementen patrones de arquitectura recomendados, como la arquitectura de componentes de Android. Room también proporciona herramientas para realizar operaciones de base de datos en segundo plano, lo que mejora el rendimiento de la aplicación. (Leiva, 2021)
- AVD: Un dispositivo virtual de Android (AVD), es un dispositivo que se utiliza para emular un dispositivo físico en el que se ejecuta una aplicación de Android. Los AVD se ejecutan en un emulador de Android, que es un programa que imita el comportamiento de un dispositivo Android en una computadora. El emulador de Android utiliza la misma arquitectura de CPU que el dispositivo físico, lo que permite que las aplicaciones de Android se ejecuten en el entorno virtual de manera similar a como la harían en un dispositivo físico real. Los AVD se crean utilizando el Administrador de AVD en Android Studio. (Android Developers, 2023)

4. Estimación de recursos y planificación (1-2 páginas)

Al ser mi primer proyecto de estas características, la estimación de recursos la he realizado sin mucho conocimiento. He evaluado los recursos que necesitaré, tales como las herramientas de desarrollo o el hardware necesario. Al ser un proyecto para una evaluación de un ciclo de formación, creo que el apartado más importante es el de planificación.

Esta etapa implica la definición de las tareas a realizar, los plazos y las fechas de entrega determinadas por el centro.

Me he centrado en la creación de un cronograma para asegurarme que el proyecto se desarrolla dentro de los plazos establecidos. Para realizar este cronograma he utilizado un diagrama de Gantt, ya que es una herramienta muy útil.

He dividido el diagrama en los dos meses principales, marzo y abril. En el diagrama de Gantt podemos observar las distintas tareas divididas y el tiempo estimado para realizarlas.

Al enfrentarme a un nuevo lenguaje de programación como es Kotlin y al desarrollo de Android, he calculado que voy a necesitar dedicar más tiempo a la tarea de aprender a manejar Kotlin y Android.

Para sacar adelante esta tarea y compaginarla con mi trabajo, me establecí una rutina de trabajo de entre 3 y 4 horas diarias. Sin embargo, debido al reto de aprender a utilizar Android, necesité más tiempo del que tenía previsto, sobre todo tiempo dedicado a corregir los múltiples errores que iban apareciendo y que tenía que solventar atendiendo a las iteraciones marcadas en mi planificación.

La memoria del proyecto será también el apartado que más tiempo me llevará, ya que la iremos elaborando desde el principio del proyecto hasta su final, pues iremos documentando todo el proceso.

También estimé que el desarrollo de la lógica y del código de la aplicación sería el segundo apartado que más tiempo me llevaría, pues aquí incluí todo lo relacionado con la corrección de las pruebas que iba llevando a cabo después de cada iteración.

Consulta la [imagen 3](#) del Anexo para ver un ejemplo del diagrama de Gantt estimado.

Finalmente podremos observar en el diagrama de Gantt real que será necesario más tiempo del pensado para la parte del desarrollo del código de la aplicación, rozando casi la fecha límite de entrega, pues van apareciendo errores que se deben ir solventando durante el

proceso. Lo mismo sucede con la memoria del proyecto, ya que debo registrar todos los cambios hasta el final del desarrollo de mi aplicación.

Por el contrario, el diseño, desarrollo e implementación de la base de datos, tardó menos tiempo de lo que tenía estimado, fue menos complejo de lo que tenía pensado.

Consulta la [imagen](#) 4 del Anexo para ver un ejemplo del diagrama de Gantt real

5. Análisis del proyecto (8-16 páginas)

En este apartado describiremos los requisitos, tanto funcionales como no funcionales, así como los diferentes diagramas.

- **Requisitos funcionales:** Son aquellos que describen las funciones o características específicas que debe tener una aplicación para satisfacer las necesidades de los usuarios. Son las especificaciones de lo que la aplicación debe hacer para cumplir con los objetivos del proyecto.

	Requisito	Descripción
RF-1	Inicio de sesión	El usuario debe iniciar sesión con su usuario y contraseña para poder ingresar a la aplicación
RF-2	Registro	El usuario debe poder registrarse en el sistema, con un usuario y contraseña única para posteriormente poder iniciar sesión con sus datos
RF-3	Pantalla principal	El usuario, en caso de no realizar el inicio de sesión, sólo debe poder acceder al registro sin poder acceder a la página principal de la aplicación
RF-4	Vista senderos	El usuario debe poder navegar por el listado de todos los senderos establecidos
RF-5	Crear ítem	El usuario debe poder crear nuevos senderos y rellenar todos los campos con la información del sendero
RF-6	Error al crear el ítem	Algunos campos son obligatorios, en caso de no rellenarlos, el usuario no podrá crear el sendero
RF-7	Marcar como favorito	El usuario debe poder marcar como favorito los senderos que desee

RF-8	Desmarcar como favorito	El usuario debe poder desmarcar como favoritos los senderos marcados
RF-9	Marcar como completado	El usuario debe poder marcar como completado los senderos que desee
RF-10	Desmarcar como completado	El usuario debe poder desmarcar como completado cualquiera de los senderos marcados
RF-11	Vista información	El usuario debe poder acceder a los atributos con la información de cada sendero
RF-12	Vista Mapa	El usuario debe poder acceder a localización del sendero en Google Maps
RF-13	Vista menú desplegable	Desde el menú desplegable se debe poder navegar hasta la vista inicio de la aplicación
RF-14	Vista menú desplegable	Desde el menú desplegable se debe poder acceder al listado de senderos guardados como favoritos
RF-15	Vista menú desplegable	Desde el menú desplegable se debe poder acceder al listado de senderos guardados como completados
RF-16	Cerrar sesión	Desde el menú desplegable el usuario debe poder cerrar su sesión y volver a la pantalla de inicio de sesión

- **Requisitos no funcionales:** Son aquellos que no está directamente relacionados con la funcionalidad de la aplicación, sino que se centran en otros aspectos, como la calidad, la usabilidad y la seguridad.

	Requisito	Descripción
RNF-1	Mensaje de error	La aplicación debe mostrar un mensaje de error ante fallos de ejecución

RNF-2	Rendimiento	El tiempo de respuesta entre las pantallas de la aplicación no debe superar los 3 segundos
RNF-3	Mantenibilidad	La aplicación debe ser fácil de mantener y actualizar, con un código limpio y estructurado.
RNF-4	Usabilidad	La aplicación debe ser fácil de usar y comprender, con una interfaz de usuario intuitiva
RNF-5	Eficiencia	Para que la información se muestre de manera eficiente y dinámica, se utilizará RecyclerView
RNF-6	Disponibilidad	La aplicación permitirá ejecutar tareas asíncronas y operaciones a la Base de Datos, sin interrumpir la fluidez del hilo principal, a través de corrutinas. Debe estar disponible en todo momento y funcionar correctamente
RNF-7	Programación	La aplicación se desarrollará enteramente en el lenguaje de programación Kotlin, usando Android Studio
RNF-8	Seguridad	La aplicación debe ser segura y proteger los datos del usuario, evitando que terceros no autorizados accedan a ellos
RNF-9	Diseño	La aplicación debe aplicar la normativa de diseño de Material Design proporcionadas por Google
RNF-10	Escalabilidad	La aplicación debe ser escalable, lo que significa que debe poder manejar un número creciente de usuarios y datos sin afectar a su rendimiento.

- **Diagrama Entidad-Relación**

Un diagrama entidad-relación (ER) es una herramienta de modelado de datos utilizada para representar la estructura lógica de una base de datos. El ER muestra las entidades relevantes para el sistema, sus atributos y las propiedades entre ellas.

Aquí se presenta la representación del diagrama entidad-relación y el diseño relacional empleado en la creación de la base de datos. Se ha llevado a cabo una normalización de las tablas hasta la Tercera Forma Normal con el objetivo de prevenir la duplicación de información y garantizar una estructura sólida y escalable de la base de datos. (Ilerna Online, 2019)

Esta sería la relación de tablas tras la normalización.

Senderos: (id, nombre, imagen, descripción, ubicación, distancia, id_municipios)

Municipios: (id, nombre)

Usuarios: (id, nombre, contraseña, email)

Favoritos: (id_usuarios, id_senderos)

Completados: (id_usuarios, id_senderos)

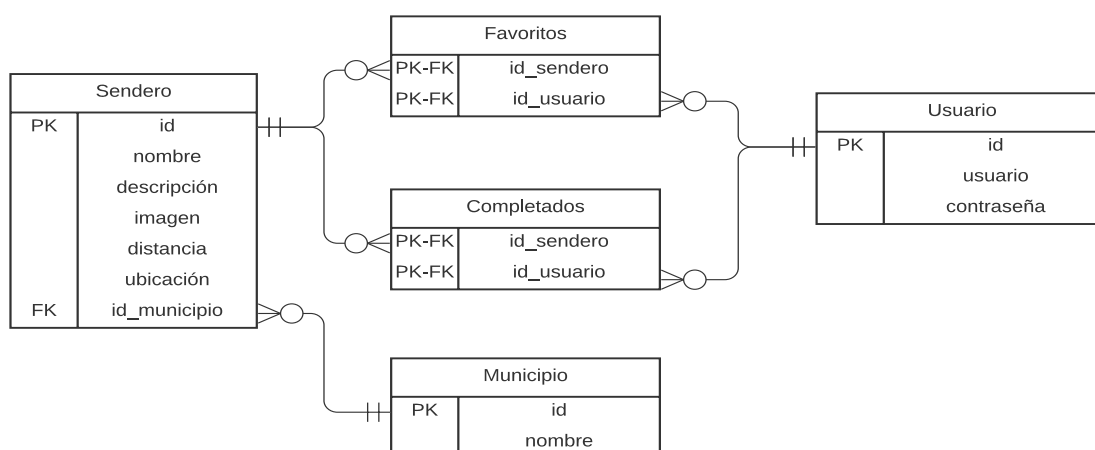
La base de datos Room presente en la aplicación almacena el nombre introducido por el usuario y su contraseña. También se almacenan los senderos introducidos por el usuario y los municipios a los que pertenece cada sendero, estos municipios también son introducidos por el usuario. Todas las operaciones y relaciones llevadas a cabo dentro de la aplicación, se relacionan por medio del Id que se genera automáticamente. Cada tabla tiene su propio Id autogenerado y permite a cada usuario tener disponible un par de listas de senderos guardados, los favoritos y los completados.

Consulta la [imagen](#) 5 del Anexo para ver el esquema Entidad-Relación

- **Modelo relacional**

Un modelo relacional es un modelo de datos utilizado en la gestión de bases de datos relacionales. Los datos se organizan en tablas, que consisten en filas y columnas. Cada tabla representa una entidad y cada fila de la tabla representa una instancia de esa entidad, mientras que cada columna representa un atributo de la entidad.

Las tablas están relacionadas entre sí mediante claves primarias y claves foráneas. Una clave primaria es una columna o conjunto de columnas que identifica de manera única cada fila de una tabla. Una clave foránea es una columna o conjunto de columnas en una tabla que se refiere a la clave primaria de otra tabla, estableciendo una relación entre ellas. (Sanchez, s.f.)



2 Imagen Modelo Relacional. Elaboración propia en Lucidchart

• Diagrama de casos de uso

Un diagrama de casos de uso es una herramienta de modelado que se utiliza para describir el comportamiento funcional de un sistema o una aplicación desde la perspectiva del usuario, en otras palabras, es una representación gráfica de los distintos roles o actores que interactúan con el sistema y los diferentes escenarios de uso que se pueden dar. (Siriwardhana, 2022)

Consulta la [imagen 6](#) del Anexo para ver el diagrama de casos de uso

En la imagen se representan los casos de uso que tienen lugar en la aplicación. El usuario puede seguir una serie de flujos en función de cómo interactúe con la aplicación. La primera pantalla es la pantalla de inicio de sesión, desde aquí podemos, o bien iniciar sesión, si somos un usuario registrado, o registrarnos. Si estamos registrados e iniciamos sesión, se comprobará que somos un usuario registrado y se avanzará hacia la pantalla principal. Por el contrario, si seleccionamos la opción de registro, se abrirá una pantalla para introducir nuestros datos, se comprobará su validez y posteriormente pasaremos a la pantalla de inicio de sesión donde podremos iniciar sesión con nuestros datos.

En la pantalla principal se muestran los senderos que hayamos ido guardando, o si no hemos guardado ninguno, aparecerá en blanco. Desde esta pantalla podemos crear un nuevo sendero y después, o bien crear un nuevo municipio, o si ya ha sido creado y aparece en la lista, seleccionarlo.

Si hay senderos guardados, aparecen en la pantalla principal y podemos seleccionarlos, una vez seleccionado podemos editar el sendero, sus especificaciones, así como el municipio al que pertenece.

Además de todo esto, podemos marcar un sendero como favorito o como completado

A continuación, vamos con la relación de tablas de casos de uso

- **Tabla de especificación casos de uso**

A continuación, se presenta una lista de casos de uso para mi aplicación:

- Registrar usuario: Un usuario debe poder crear una cuenta en la aplicación proporcionando su información personal.
- Iniciar sesión: Un usuario registrado debe poder iniciar sesión en la aplicación con su correo electrónico y contraseña.
- Ver lista de senderos: Un usuario registrado debe poder ver una lista de senderos disponibles en la aplicación.
- Crear sendero: Un usuario registrado debe poder crear un nuevo sendero proporcionando información como nombre, descripción, imagen, distancia y ubicación.
- Modificar sendero: Un usuario registrado debe poder editar la información de un sendero existente, como su nombre, descripción o ubicación.
- Eliminar sendero: Un usuario registrado debe poder eliminar un sendero que haya creado previamente.
- Asociar municipio a sendero: Un usuario registrado debe poder crear un nuevo municipio y asociarlo a un sendero existente.
- Ver lista de municipios: Un usuario registrado debe poder ver una lista de municipios asociados a los senderos.
- Escoger un municipio para un sendero: Un usuario registrado debe poder escoger un municipio de una lista para asociarlo a un sendero existente.
- Editar la lista de municipios: Un usuario registrado debe poder editar la lista de municipios asociados a los senderos.
- Marcar un sendero como favorito: Un usuario registrado debe poder marcar un sendero como favorito para acceder a él fácilmente en el futuro.
- Marcar un sendero como completado: Un usuario registrado debe poder marcar un sendero como completado para llevar un registro de los senderos que ha recorrido.

- Eliminar un municipio de la lista: Un usuario registrado debe poder eliminar un municipio de la lista de municipios asociados a los senderos.
- Eliminar un sendero de la lista: Un usuario registrado debe poder eliminar un sendero de la lista de senderos disponibles.

En el [anexo](#) se encuentra la tabla de especificación de casos de uso

- **Diagrama de Clases**

Consulta la [imagen](#) 7 del Anexo para ver el diagrama de clases

En la imagen podremos observar las distintas clases, así como sus relaciones que explicaremos aquí brevemente:

Entidades

En el modelo de datos que se representa en el diagrama, hay cinco entidades principales:

- SenderoEntidad: representa los detalles de un sendero, como su nombre, descripción, imagen, distancia, ubicación y municipio. La entidad tiene una relación muchos a uno con la entidad Municipio y una relación muchos a muchos con las entidades Favorito y Completado.
- Municipio: representa un municipio y tiene una relación uno a muchos con la entidad SenderoEntidad.
- Favorito: representa la relación entre un usuario y los senderos que ha marcado como favoritos. Tiene una relación muchos a uno con la entidad SenderoEntidad y con la entidad Usuario.
- Completado: representa la relación entre un usuario y los senderos que ha completado. Tiene una relación muchos a uno con la entidad SenderoEntidad y con la entidad Usuario.
- Usuario: representa un usuario y tiene una relación muchos a muchos con las entidades Favorito y Completado.

Interfaz DAO

La interfaz SenderoDAO representa la capa de acceso a datos para la entidad SenderoEntidad. Define las operaciones básicas de CRUD (create, read, update, delete) que se pueden realizar sobre los registros de SenderoEntidad en la base de datos.

Además, la interfaz SenderoDAO define dos métodos adicionales para obtener los senderos favoritos y completados de un usuario.

La interfaz UsuarioDAO define los métodos necesarios para acceder y manipular la tabla Usuario en la base de datos. Los dos métodos definidos en esta interfaz permiten guardar un objeto Usuario en la base de datos y buscar un objeto Usuario por su Id, si no encontrara un usuario con el Id especificado, se devolvería un valor nulo

Relaciones

En el diagrama UML, las relaciones entre las entidades se representan mediante líneas entre las entidades. Las líneas se etiquetan con el tipo de relación que existe entre las entidades.

- La entidad SenderoEntidad tiene una relación muchos a uno con la entidad MunicipioEntidad. Esto significa que muchos senderos pueden estar asociados a un solo municipio, pero un sendero solo puede pertenecer a un municipio.
- La entidad SenderoEntidad tiene una relación muchos a muchos con las entidades Favorito y Completado. Esto significa que un sendero puede ser marcado como favorito o completado por muchos usuarios, y un usuario puede marcar muchos senderos como favoritos o completados.
- La entidad Favorito tiene una relación muchos a uno con la entidad SenderoEntidad y con la entidad Usuario. Esto significa que un favorito está asociado a un solo sendero y a un solo usuario, pero un usuario puede tener muchos favoritos.
- La entidad Completado tiene una relación muchos a uno con la entidad SenderoEntidad y con la entidad Usuario. Esto significa que un sendero completado está asociado a un solo sendero y a un solo usuario, pero un usuario puede completar muchos senderos

Métodos

Cada entidad e interfaces DAO tienen métodos asociados que representan las acciones que se pueden realizar en el modelo de datos. Algunos métodos están implícitos, como los “getters” y “setters” para los atributos de una entidad, mientras que otros métodos son específicos del modelo de datos, como los métodos “save()”, “login()” y “registro()” definidos en la entidad Usuario, además de “findOneById(id:Long):Usuario?” el cual busca un objeto Usuario en la tabla Usuario por su ID y devuelve el objeto encontrado. Si no se encuentra un objeto con el ID especificado, se devuelve un valor nulo.

6. Diseño del proyecto (6-14 páginas)

El inicio de la aplicación arrancará con la pantalla principal, la cual nos pedirá iniciar sesión mediante un usuario único y una contraseña. En caso de no disponer de una cuenta, mediante el botón de registro, podemos registrarnos y acceder al sistema.

Una vez iniciada sesión, el usuario puede acceder a la pantalla principal donde se muestra un listado de los senderos que el usuario haya ido añadiendo. Además, podremos marca y desmarcar cada sendero como favorito cualquiera de los senderos que deseemos.

También hemos incluido la opción de que el usuario vaya registrando todos los senderos que vaya realizando, para llevar una cuenta de los senderos completados, para ello, el usuario puede marcar y desmarcar un sendero como completado. Pulsando sobre alguno de los senderos se abrirá una ventana nueva donde se nos ofrece información sobre el sendero, municipio al que pertenece y un enlace para acceder a su localización GPS abriendo un *fragment* usando la API de Google Maps.

Desde esa pantalla lista de senderos, el usuario también podrá añadir nuevos senderos mediante un botón de añadir, al añadir un sendero tendrá que añadir también el municipio al que pertenece o si ya existe, seleccionarlo de un menú desplegable. También podemos editarlo y eliminarlo.

Adicionalmente, en la parte superior izquierda, dispondremos de un menú desplegable, este tendrá las opciones de cerrar la sesión del usuario, visualizar el estado global o pantalla de inicio, un acceso a la pantalla de favoritos y otro acceso a la pantalla con la lista de senderos que el usuario ha completado.

Consulta la [imagen 8](#) en el anexo para ver un boceto con el diseño de la aplicación

La interfaz del usuario se diseña a través de *layouts*, como el *ConstraintLayout*, el que más hemos utilizado. Este *layout* nos permite posicionar y establecer relaciones entre los distintos elementos de la interfaz de usuario. Estas relaciones permiten una mayor flexibilidad y precisión en la disposición de los elementos, lo que a su vez permite una mejor adaptación a diferentes resoluciones de los elementos, posibilitando de esta manera un mayor control sobre su apariencia, para ello configuramos las dimensiones, colores y temas, todos ellos se disponen en el directorio *resources* de la aplicación. Precisamente para mantener la disposición en pantalla hemos creado unas guías con un margen a cada uno de los lados de un 10%, de modo que nuestra aplicación mantenga una coherencia en todas las pantallas.

Adicionalmente hemos usado otro tipo de *layout* en la aplicación, para los diferentes *RecyclerView* y *diálogos*, el *CardLayout*, el cual se utiliza para mostrar tarjetas con la información de nuestros senderos

Además, existe una clase crucial que hereda de *Application*, posee características importantes como, por ejemplo, será la primera clase en ejecutarse al iniciar la aplicación, y, además, esta clase *Application*, se ejecuta de forma automática, para ello se establece en el fichero *AndroidManifest.xml* bajo el atributo *name*.

En nuestra aplicación, esta clase resultará útil a la hora de acceder a las *SharedPreferences* y guardar las diferentes preferencias del usuario para la aplicación, es decir, guardaremos en ella las diferentes preferencias del usuario para la aplicación, como el inicio de sesión, así cuando ejecutemos la aplicación, si ya hemos iniciado sesión, no tendremos que volver a hacerlo.

Registro: Su lógica consiste en recoger la información del usuario con el formulario de registro y enviar la entidad usuario generada al *ViewModel*. Cuando el usuario rellena el formulario se hacen dos comprobaciones, por un lado, todos los campos del formulario deben estar rellenos y, por otro lado, el usuario debe introducir la contraseña dos veces para asegurarse de que no hay ningún error y el sistema debe verificar que ambas contraseñas son iguales. Si todos los pasos se completan correctamente, mediante un *Toast* se informará al usuario.

Los *ViewModel* son clases pequeñas que se encargan de almacenar y recuperar la información necesaria para la interfaz de usuario. Cada interfaz debe tener su propio *ViewModel*, pero a veces varias interfaces pueden compartir uno. Esto permite que la información se comparta entre ellas, lo que acelera significativamente la actualización de los datos visuales.

Después de enviar la información al *ViewModel* y de que este interactúe con Room, guardaremos el ID del nuevo usuario y su nombre en las *SharedPreferences*, y lo marcaremos como conectado automáticamente. Si intentáramos crear un usuario con el mismo nombre dos veces, Room lanzaría una excepción, ya que hemos establecido que el nombre de usuario debe ser único. En tal caso, informaríamos al usuario de que no puede volver a registrarse

Login: De manera similar a como lo hicimos para la actividad de registro, comenzamos diseñando la interfaz gráfica para la pantalla de inicio de sesión. Ambas interfaces son casi idénticas, ya que lo único que cambia es la ausencia de un segundo campo para confirmar la

contraseña. En cuanto a la lógica, la diferencia es mínima. Lo primero que hacemos es comprobar, mediante *SharedPreferences*, si el usuario ya ha iniciado sesión o no. Si ya ha iniciado sesión, redirigimos al usuario a la pantalla *HomeActivity* y cerramos la actividad actual para evitar que quede en la pila de memoria. Si el usuario no ha iniciado sesión, mostramos la interfaz de inicio de sesión para que el usuario pueda acceder a su cuenta. Después de este paso, el sistema comprobará que no hay ningún campo en blanco, enviará la información de los campos al *ViewModel* y este se conectará con la base de datos para comprobar que los datos existen y son correctos.

HomeActivity

La *HomeActivity* es una de las actividades más cruciales en una aplicación, ya que es responsable de administrar prácticamente toda la lógica restante. En primer lugar, actúa como la actividad base que aloja el *NavigationDrawer*, que es un patrón común en las aplicaciones de Android. Este patrón proporciona un menú lateral que se puede desplegar para acceder a las distintas secciones de la aplicación.

NavigationDrawer

Para configurar el *NavigationDrawer* en nuestra aplicación, es necesario utilizar varios ficheros y componentes adicionales. Estos son los que usaremos nosotros:

- **Menú:** es el fichero que almacena los distintos elementos que se mostrarán en el *NavigationDrawer*. Cada elemento se compone de un nombre, un icono y un identificador único.
- **Navigation:** es el fichero que acompaña al menú y donde se define la correspondencia entre los elementos y las pantallas de la aplicación. Aquí es donde se establece la vinculación entre los elementos del menú y las diferentes secciones de la aplicación.
- **NavigationView:** es el componente visual que se utiliza para añadir el *NavigationDrawer* a la interfaz de usuario.
- **AppBarLayout:** es el componente que crea una barra superior en la que se encuentra el botón que permite abrir el *NavigationDrawer*.
- **NavHostFragment:** es un componente de Android que permite cambiar el fragmento que se está mostrando en pantalla en cualquier momento.

Fragment

Utilizamos los fragmentos *HomeFragment*, *FavoritosFragment* y *CompletadosFragment* para recuperar todos los senderos, son muy similares, pero existen diferencias entre ellos a la hora de recuperar los senderos, así tenemos en el “Home” todos los que tengamos añadidos y aparecerán en la vista principal, en el “Favoritos”, todos los que seleccionemos como favoritos y en el “Completados”, todos los que hayamos seleccionado como senderos ya completados. Para recuperar los senderos, se realizan consultas distintas en Room.

La entidad principal es “SenderoEntidad”, de la cual se requiere toda su información, esta entidad tiene dos índices, uno en el campo nombre con la restricción de que debe ser único y otro en el campo “id_municipio”. Además, tiene una relación de clave foránea con la entidad “Municipio”, que se establece a través de un “ForeignKey”. La clave foránea se establece mediante los campos “id”.

La opción *onUpdate* establece que si se actualiza el “id” en la entidad “Municipio”, también se actualizará el “id_municipio” en la entidad “SenderoEntidad”. La opción *onDelete* establece que, si se elimina un registro de “Municipio”, también se eliminarán los registros correspondientes de “SenderoEntidad”. La entidad “Favorito” mostrará, haciendo uso de un booleano, si el usuario ha dado me gusta al sendero en cuestión. La entidad “Completado” indica si el usuario ha marcado como completado el sendero.

Otro fragment que usaremos también en nuestra aplicación es el *MapsFragment*, con los datos de ubicación establecidos, gracias al uso de las librerías de Google Maps que nos permite, lanzar la ubicación del sendero que seleccionemos. Se abrirá el fragment de Google Maps en el punto donde se encuentra dicho sendero localizado geográficamente, además he implementado la funcionalidad de que al hacer un *onLongMapClickListener()* en el mapa, aparece un *Toast* con las coordenadas geográficas donde se ha hecho clic.

Para poder abrir Google Maps, es necesario otorgar permisos de ubicación a nuestro dispositivo, eso lo haremos en el archivo *Manifest* para posteriormente implementar la gestión de permisos de ubicación y la visualización de un mapa en la aplicación

La función *createMapFragment()* crea un fragmento de mapa y lo asigna a un elemento con ID “map”. Luego, llama a *getMapAsync(this)* para iniciar la carga asíncrona del mapa. La función *isLocationPermissionGranted()* verifica si se han otorgado los permisos necesarios para acceder a la ubicación del dispositivo, después de esto, la función *enableLocation()* habilita la ubicación en el mapa si se han otorgado los permisos de ubicación, en caso de que no se hayan otorgado los permisos, llama a *requestLocationPermission()* para solicitarlos,

después, la función *requestLocationPermission()* muestra un mensaje *Toast* al usuario si no se han otorgado los permisos y los solicita a través de *ActivityCompat.requestPermissions()*.

Por último, el método *onRequestPermissionsResult()* es llamado después de que el usuario haya respondido a la solicitud de permisos y verifica si los permisos se han otorgado, si ha sido así, habilita la ubicación en el mapa y si no se han otorgado, muestra un mensaje *Toast* al usuario.

RecyclerView

Para presentar adecuadamente los datos recuperados en cada fragment anteriormente explicado, necesitamos generar un listado de senderos, sin embargo, nos enfrentamos a una lista dinámica de elementos, lo que dificulta la creación de una interfaz gráfica adaptable para mostrar todos los senderos correspondientes de manera efectiva. Dependiendo del uso que el usuario haga de la aplicación, habrá más o menos tarjetas correspondientes a senderos. Para solucionar este problema en Android, podemos emplear dos soluciones: *ListView* o *RecyclerView*. Nos centraremos en este último, ya que es una evolución del primero. El *RecyclerView* es un componente que permite la generación dinámica de elementos en la interfaz gráfica, ajustándose en función de cuántos elementos existan y creando solo los necesarios. Si nuestra aplicación recupera simultáneamente una gran cantidad de senderos (por ejemplo, 100), no se crearán todos al mismo tiempo, ya que esto podría sobrecargar la memoria del dispositivo. En su lugar, se generará un rango de elementos para que al usuario le parezca que están todos cargados, pero en realidad se irán creando y destruyendo elementos internamente.

Por esta razón, el *RecyclerView* necesita de un adaptador que, como mínimo, debe definir tres funciones y una clase. La clase interna, generalmente llamada *ViewHolder*, es la encargada de vincular los datos de un elemento de la lista con una tarjeta, ya que es la vista que posee la información de un elemento en particular.

Las funciones esenciales del adaptador *RecyclerView* son las siguientes:

- *getItemCount()*: Devuelve la cantidad máxima de elementos que deben existir en la lista que se creará.
- *onCreateViewHolder()*: Crea instancias de la clase *ViewHolder* cuando sea necesario.
- *onBindViewHolder()*: Envía información a la instancia creada para que pueda dibujarse en pantalla correctamente.

Además, se ha creado un *listener* vinculado a los diferentes fragmentos principales para controlar las acciones que ocurren en el *RecyclerView*. Este listener tiene las siguientes acciones:

- Añadir a favoritos: Permite guardar el sendero en la lista de favoritos del usuario.
- Eliminar de favoritos: Permite eliminar el sendero de la lista de favoritos.
- Añadir al listado de senderos completados: Permite añadir el sendero a la lista de senderos completados.
- Eliminar del listado de senderos completados: Permite eliminar el sendero de la lista de senderos completados.
- Abrir ubicación: Permite abrir el fragment de Maps con la ubicación guardada del sendero.
- Ver detalles: Abre un diálogo con información detallada del sendero.
- Edición: Abre la actividad correspondiente a la edición del sendero.

Visualización detalle de sendero

Cuando hacemos clic en un elemento del *RecyclerView* para ver su información detallada, se abrirá un cuadro de diálogo emergente que mostrará toda la información necesaria. En Android no hay un componente que permita hacer esto de manera directa, por lo que debemos programarlo nosotros mismos extendiendo la lógica de la clase *DialogFragment*. Este componente es un fragmento flotante que podemos superponer en la interfaz gráfica en cualquier momento de nuestras aplicaciones.

Los pasos de creación de este diálogo emergente son muy sencillos, creamos el diseño que se desea mostrar al crear el diálogo, luego creamos una clase que extienda de *DialogFragment* y sobrescribimos el método *onCreateDialog*, que es similar al *onCreate* de cualquier actividad. Después de esto, utilizamos el patrón *builder* para describir el comportamiento del diálogo e iniciamos la vista mediante un *ViewBinding*.

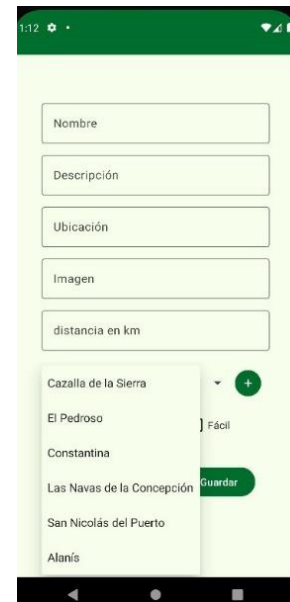
Creación y edición de un sendero

Después de recuperar la información necesaria, se procede a rellenar el *spinner* con la información correspondiente, en el caso de nuestra aplicación, la información que contiene son los nombres de los municipios que el usuario vaya añadiendo. Además, se ha



implementado la funcionalidad de un botón para poder crear nuevos municipios, los cuales se guardan en la base de datos para poder ser seleccionados posteriormente.

Una vez completados los campos del formulario, se procede a guardar la información del sendero en la base de datos o a actualizar la información en caso de estar editando un sendero existente. Para ello se han implementado las funciones correspondientes en la clase que gestiona la base de datos.



7. Despliegue y pruebas (4-10 páginas)

Para ir controlando el proceso de creación correctamente, se han ido haciendo pruebas constantemente a medida que se iban añadiendo *activities*, siguiendo el modelo en cascada con retroalimentación. Esto ha facilitado identificar y solventar los errores durante el proceso, ahorrando trabajo al llegar al final del proyecto.

Para asegurar que las operaciones de lectura y escritura en la base de datos se estaban realizando correctamente, he seguido las normas de diseño del ORM Room en Android.

- **Uso de entidades:** Utilizar las entidades como se muestra en el modelo relacional (Imagen 2, pág. 17). Cada Entidad representa una tabla y en ellas se observan sus atributos
- **Uso de DAO:** En el diagrama de clases ([imagen 7 del Anexo](#)), se observan las consultas a la base de datos de cada DAO de la aplicación. Cada DAO se encarga de realizar una tarea específica, como insertar, actualizar o eliminar datos
- **Uso de LiveData:** Room integra el componente LiveData de Android que nos proporciona una forma eficiente y fácil de monitorear los cambios en la base de datos
- **Optimización de consultas:** Mediante uso de SQL y el uso de Entidades Base y DAOs base, en el caso de nuestra aplicación, “Base Entity” y “Base DAO”, optimizamos las consultas a la base de datos

Para garantizar que la aplicación cumpla con los requisitos funcionales, hemos realizado una serie de pruebas de aceptación, que son pruebas que se realizan para verificar si la aplicación cumple con las expectativas.

En la siguiente tabla se detallan las pruebas realizadas para comprobar que la aplicación funciona correctamente y que ofrece una buena experiencia de usuario. Hemos definido, respecto a los requisitos funcionales, una serie de pruebas que queremos que la aplicación pase.

#	RF	Pruebas unitarias
1	Inicio de sesión	Verificar que el usuario puede iniciar sesión con credenciales válidas y no puede iniciar sesión con credenciales inválidas

		Pruebas realizadas: Iniciar sesión con usuarios registrados y usuarios sin registrar
2	Registro	<p>Verificar que el usuario puede registrarse en el sistema y que los datos ingresados se almacenan correctamente</p> <p>Pruebas realizadas: Registrar un usuario sin rellenar todos los campos y registrar un usuario rellenando todos los campos y posteriormente iniciar sesión para verificar que los datos se han guardado correctamente</p>
3	Acceder sólo al registro, sin iniciar sesión	<p>Verificar que el usuario sólo puede acceder a la página de registro si no ha iniciado sesión</p> <p>Pruebas realizadas: Intentar iniciar sesión con los datos de un usuario no registrado</p>
4	Navegar por el listado de senderos	Pruebas realizadas: Verificar que el usuario puede navegar por el listado de senderos y que se muestran los senderos de manera correcta
5	Crear nuevos senderos	<p>Verificar que el usuario puede crear un nuevo sendero y que se guarda correctamente en el sistema</p> <p>Pruebas realizadas: Crear un nuevo sendero rellenando todos los campos requeridos y volver a la pantalla principal para verificar su creación</p>
6	Avisar de un error al rellenar incorrectamente los datos del sendero o en caso de existir ya dicho sendero	<p>Verificar que aparece un mensaje de error en caso de que el usuario haya introducido datos incorrectos. Verificar que aparece un mensaje avisando de que hay campos vacíos en caso de que el usuario haya dejado algún campo vacío. Y, por último, verificar que aparece un mensaje avisando de que el sendero que se quiere guardar ya existe</p> <p>Pruebas realizadas: Crear un sendero con datos de un municipio no existente.</p> <p>Crear un sendero sin rellenar todos los campos.</p> <p>Crear un sendero con un nombre ya existente en la lista de senderos</p>

7	Marcar sendero como favorito	<p>Verificar que el usuario puede marcar un sendero como favorito y que se muestra correctamente en el listado de senderos favoritos</p> <p>Prueba realizada: Marcar un sendero como favorito tanto en la vista lista de senderos como en la vista información del sendero</p> <p>Navegar desde el menú desplegable a la lista de senderos favoritos</p>
8	Desmarcar sendero como favorito	<p>Verificar que el usuario puede desmarcar correctamente un sendero como favorito y que se elimina correctamente del listado de senderos favoritos</p> <p>Prueba realizada: Desmarcar un sendero como favorito tanto en la vista lista de senderos como en la vista información del sendero</p> <p>Navegar desde el menú desplegable a la lista de senderos favoritos</p>
9	Marcar sendero como completado	<p>Verificar que el usuario puede marcar un sendero como completado y que se muestra correctamente en el listado de senderos completados</p> <p>Prueba realizada: Marcar un sendero como completado tanto en la vista lista de senderos como en la vista información del sendero</p> <p>Navegar desde el menú desplegable a la lista de senderos completados</p>
10	Desmarcar sendero como completado	<p>Verificar que el usuario puede desmarcar un sendero como completado y que se elimina correctamente del listado de senderos completados</p> <p>Prueba realizada: Desmarcar un sendero como completado tanto en la vista lista de senderos como en la vista información del sendero</p> <p>Navegar desde el menú desplegable a la lista de senderos completados</p>
11	Acceder a la información de cada sendero	<p>Verificar que el usuario puede acceder a los atributos con la información de cada sendero y que la información se muestra correctamente</p>

		Prueba realizada: Pulsar sobre un sendero de la lista
12	Acceder a la ubicación del sendero en Google Maps	<p>Verificar que el usuario puede acceder a la ubicación del sendero y se abre correctamente la ventana con la información proporcionada por Google Maps</p> <p>Prueba realizada: Pulsar sobre la pestaña ubicación de cada sendero</p>
13	Navegar hasta la vista principal de la aplicación	<p>Verificar que el usuario puede navegar hasta la vista principal de la aplicación desde las opciones del menú desplegable</p> <p>Prueba realizada: Abrir el menú desplegable y pulsar sobre la opción de Home</p>
14	Acceder al listado de senderos favoritos	<p>Verificar que el usuario puede acceder al listado de senderos guardados como favoritos desde las opciones del menú desplegable</p> <p>Prueba realizada: Abrir el menú desplegable y pulsar sobre la opción de Favoritos</p>
15	Acceder al listado de senderos guardados como ya realizados	<p>Verificar que el usuario puede acceder al listado de senderos guardados como completados desde las opciones del menú desplegable</p> <p>Prueba realizada: Abrir el menú desplegable y pulsar sobre la opción de Completados</p>
16	Cerrar sesión desde las opciones del menú desplegable y volver a la pantalla de inicio de sesión	<p>Verificar que al acceder al menú desplegable existe una opción para cerrar la sesión del usuario</p> <p>Verificar que, al presionar sobre la opción de cerrar sesión, el usuario es redirigido a la pantalla de inicio de sesión.</p> <p>Verificar que los datos de la sesión se han borrado correctamente y que el usuario no puede acceder a la aplicación sin iniciar sesión nuevamente</p> <p>Prueba realizada: Abrir el menú desplegable y pulsar sobre la opción de Cerrar Sesión</p>

En el apartado de despliegue, se llevan a cabo varias tareas. En primer lugar, se debe compilar y empaquetar la aplicación en un archivo APK, que es el formato de archivo utilizado para distribuir e instalar aplicaciones en dispositivos Android.

Para ello seguimos estos pasos:

- Compilar el proyecto haciendo clic en “Build” y comprobar que no existen errores
- Revisar los avisos que aparecen en la consola como pueden ser
- Acceder a la pestaña “Generate Signed Bundle/APK”.
- En la ventana “Generate Signed Bundle or APK”, selecciona “APK
- Hacemos clic en “Create new” para crear una nueva clave de firma, o seleccionamos una clave existente
- Completamos el formulario de configuración de la clave de firma, incluyendo la información de certificado y la contraseña
- Hacemos clic en “Next” y seleccionamos el módulo de la aplicación para el cual se desea generar el archivo APK
- Seleccionamos la opción de “release build type”
- Seleccionamos la versión de Android y el nivel mínimo para los cuales la aplicación está diseñada
- Hacemos clic en “Finish” para generar el archivo APK y una vez terminado el proceso, la APK se guarda en la ubicación especificada en la ventana “Generate Signed Bundle or APK”

Una vez desplegada la aplicación, consideramos también interesante comentar aquí el mantenimiento de la aplicación, ya que es la etapa posterior al despliegue de la aplicación, en la cual realizaremos actividades destinadas a garantizar el correcto funcionamiento y calidad de la aplicación en el tiempo. Es necesario destacar, que nuestra aplicación es muy sencilla y que su desarrollo se enmarca en la realización de un proyecto final de estudios, por lo tanto, no va a requerir de un gran mantenimiento, ni la API que vamos a utilizar va a requerir que estemos constantemente pendiente de su métrica ni sus cuotas, ya que estimamos que las solicitudes que se van a hacer a la API serán pocas, no obstante, sí debemos estar pendientes, de que las solicitudes a la API por parte de los usuarios, se lleven a cabo correctamente, y en caso de errores, actualizar la API y realizar las mejoras pertinentes para su rendimiento.

8. Conclusiones (1-2 páginas)

Al principio de este proyecto se plantearon una serie de objetivos a lograr que se han alcanzado con éxito, además, ya que mi propuesta de proyecto podría resultar bastante simple para demostrar los conocimientos adquiridos durante el curso, decidí aumentar la complejidad de la aplicación aumentando el número de tablas en la base de datos. No obstante, este hecho tuvo como consecuencia el retraso en la entrega del proyecto y que no pudiera terminar algunos aspectos que me hubiera gustado.

A pesar de todo, he de añadir que he aprendido bastante sobre el desarrollo Android y el uso de bases de datos en Room, sobre todo a la hora de diseñar y modelar las tablas de una base de datos para que sean útiles y fáciles de usar para los usuarios de la aplicación.

También he aprendido a utilizar varias bibliotecas de Android, como RecyclerView, LiveData y ViewModel, para crear una interfaz de usuario dinámica y receptiva. También a integrar diferentes componentes de la aplicación, como la autenticación de usuario y el seguimiento de la ubicación.

Uno de los mayores retos que enfrenté durante el desarrollo de este proyecto fue la gestión de la complejidad de la base de datos y la comunicación entre diferentes componentes de la aplicación. La creación de una base de datos con múltiples tablas y relaciones se me hizo complicada ya que requería una planificación cuidadosa para evitar errores y garantizar la eficacia de la aplicación.

Otro desafío fue encontrar una manera de almacenar las coordenadas seleccionadas por el usuario en el mapa, sin afectar significativamente el rendimiento de la aplicación. Para empezar, mi idea era que el usuario pudiera seleccionar un punto en el mapa (un sendero), guardarlo en la base de datos y que quedara reflejado en el campo ubicación mediante SharedPreferences, para posteriormente, desde el fragment donde aparece toda la información del sendero, al clicar sobre el campo ubicación, lanzara un intent al fragment de Google Maps y se abriera la ubicación del sendero. Además, he implementado la funcionalidad de, al hacer un `onLongMapClick()` en el mapa, aparece un Toast con las coordenadas geográficas donde hemos hecho clic. La idea era almacenar esas propiedades para posteriormente lanzar la ubicación. Explicaré esto mejor en “Vías Futuras”.

En definitiva, ha sido un reto, tanto técnico como de gestión del tiempo. Al enfrentarme a una tecnología que desconocía, he necesitado de muchos tutoriales para aprender a desarrollar

en Android, sobre todo ha sido muy frustrante no poder cumplir con el objetivo que tenía con la API de Google Maps.

Pero ya una vez con la aplicación terminada, tengo que decir que estoy conforme con lo conseguido y sobre todo muy contento con lo aprendido, además considero que mi aplicación cumple con todos los requisitos solicitados por el curso, ya que tiene más de las tres tablas de bases de datos.

9. Vías futuras (1-2 páginas)

Mi objetivo es seguir con el desarrollo de este proyecto e implementar las funcionalidades que se quedaron a medias. Tal vez pueda ser una alternativa ligera, fácil de manejar y más localista, a Wikyloc.

Mi idea es continuar con el proyecto e implementar mejoras, como una base de datos en la nube. De este modo podría integrar un sistema de recomendaciones de senderos, basado en las preferencias de los usuarios, que sugiera nuevos senderos relevantes para ellos. Además esta opción me permitiría crear también un ranking de dificultad, donde los usuarios pudieran organizar los senderos en orden de dificultad y compartir el ranking y los senderos completados según su dificultad.

Otra opción que me gustaría incluir en el futuro sería agregar una función de comentarios y calificaciones para los senderos, lo que permitiría a los usuarios compartir sus experiencias y opiniones sobre los diferentes senderos. Esta opción podría generar información de utilidad para los responsables del Parque Natural para mantener y mejorar las rutas senderistas.

Además, he de seguir con la correcta implementación de la API de Google Maps, para conseguir que el usuario seleccione un sendero desde el propio mapa y se guarde en la aplicación. De modo que cuando acceda al sendero, se abra la ubicación guardada.

Otras posibles mejoras para mi aplicación serían, mejorar el diseño de la interfaz, incluir Figma para implementar una interfaz más “vistosa” y se podría considerar también, la expansión de la aplicación a otras plataformas, como iOS y la web, para llegar a una audiencia aún más amplia y permitir a los usuarios acceder a la aplicación desde diferentes dispositivos.

En resumen, existen muchas posibilidades para este proyecto en el futuro y creo que algunas de estas opciones pueden ayudar a mejorar la experiencia de usuario y hacer la aplicación aún más valiosa y útil para los amantes de la naturaleza y los entusiastas del senderismo.

10. Bibliografía/Webgrafía (1-2 páginas)

- Adeva, R. (2023, 02 02). *Qué es Android: todo sobre el sistema operativo de Google*. From ADSL ZONE: <https://www.adslzone.net/reportajes/software/que-es-android/>
- Android Developers. (2023, 02 07). *Cómo crear y administrar dispositivos virtuales*. From Android Developers: <https://developer.android.com/studio/run/managing-avds?hl=es-419>
- Czura, T. (n.d.). *Introducción a Kotlin: Programación de Android Para Seres Humanos*. From Toptal: <https://www.toptal.com/software/introduccion-a-kotlin-programacion-de-android-para-seres-humanos>
- Ilerna Online. (2019, 11 12). *El modelo Entidad-Relación: el esquema de una base de datos*. From El Blog de Ilerna Online: <https://www.ilerma.es/blog/informatica-comunicacion/modelo-entidad-relacion-base-de-datos/>
- Jacobs, M. (2023, 02 09). *¿Qué es Git?* From Microsoft: <https://learn.microsoft.com/es-es/devops/develop/git/what-is-git?source=docs>
- Kanbanize. (2023). *¿Qué es Kanban? Explicación para principiantes*. From Kanbanize: <https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban>
- Leiva, A. (2021). *Room, la librería de Base de datos de Android*. From Devexperto: <https://devexperto.com/room-la-libreria-de-base-de-datos-de-android/>
- Ryte Wiki. (n.d.). *Modelo en Cascada*. From https://es.ryte.com/wiki/Modelo_en_Cascada.
- Sanchez, J. (n.d.). *El modelo relacional*. From Manual de gestión de bases de datos: <https://jorgesanchez.net/manuales/gbd/modelo-relacional.html>
- Siriwardhana, S. (2022, 09 29). *Tutorial de diagrama de casos de uso*. From Creately: <https://creately.com/blog/es/diagramas/tutorial-diagrama-caso-de-uso/#:~:text=El%20diagrama%20de%20caso%20de,roles%20interact%C3%BAan%20con%20el%20sistema.>
- Solomon, K. (2022, 06 26). *Por lo tanto, ¿qué es el gestor de tareas Trello y para qué sirve?* From Trello: <https://blog.trello.com/es/que-es-trello>
- Souza, I. d. (2019, 07 12). *XML: ¿qué es y para qué sirve este lenguaje de marcado?* From Rockcontent.

Wikipedia. (2023, 02 02). *Android Studio*. From Wikipedia:
https://es.wikipedia.org/wiki/Android_Studio

Zendesk. (2023, 02 14). *¿Qué es la metodología ágil y cuáles son las más utilizadas?* From
Blog de Zendesk: <https://www.zendesk.com.mx/blog/metodologia-agil-que-es/>

11. Anexos

Manual de instalación

El manual de instalación constaría de los siguientes pasos:

1. Descarga e instala Android Studio: Para poder instalar la aplicación, primero se necesita tener instalado Android Studio en la computadora. Puede descargarlo desde el siguiente enlace: <https://developer.android.com/studio>
2. Descarga el código fuente: una vez que tenga Android Studio instalado, debe descargar el código fuente de la aplicación. Esto puede ser un archivo ZIP o un repositorio Git que contenga todo el código necesario para la aplicación. Puede descargarlo desde el siguiente enlace: <https://github.com/eduferaz/AppSendero.git>
3. Abrir el proyecto en Android Studio: Después de descargar el código fuente, debe abrir Android Studio y seleccionar "Open an existing Android Studio project" (Abrir un proyecto existente de Android Studio). Luego, seleccione la carpeta que contiene el código fuente descargado
4. Descargar las dependencias: Antes de poder ejecutar la aplicación, es posible que tenga que descargar algunas dependencias. Esto se puede hacer desde el archivo "build.gradle" en Android Studio. Haga clic en "Sync Now" (sincronizar ahora) en la barra de estado para descargar las dependencias necesarias.
5. Conectar un dispositivo Android o usar un emulador: Antes de ejecutar la aplicación debe conectar un dispositivo Android o usar un emulador. Para usar un emulador, puede crear uno desde el menú "AVD Manager" (Administrador de AVD) en Android Studio
6. Ejecutar la aplicación: Finalmente, puede ejecutar la aplicación haciendo clic en el botón "Run" (Ejecutar) en la barra de herramientas de Android Studio. La aplicación se ejecutará en el dispositivo o emulador conectado.
7. Instalación en el dispositivo: también puede instalarla en el dispositivo para usarla fuera de Android Studio. Para hacer esto, haga clic en el botón "Build" en la barra de herramientas de Android Studio y seleccione "Generate signed APK". Luego, siga las instrucciones para generar un archivo APK firmado y cópielo en su dispositivo Android para instalarlo.

Instalación en el dispositivo: también puede instalarla en el dispositivo para usarla fuera de Android Studio:

1. Para hacer esto, haga clic en el botón “Build” en la barra de herramientas de Android Studio y seleccione “Generate signed APK. Luego siga las instrucciones para generar un archivo APK firmado y cópielo en su dispositivo Android para instalarlo
2. Activar la opción de “origen desconocido”: Antes de instalar el APK, debes activar la opción de “origen desconocido” en tu dispositivo. Para hacer esto, debes ir a “Configuración” > “Seguridad” y luego activar la opción de “Orígenes desconocidos”.
3. Instalar el APK: Una vez descargado el APK, simplemente haz clic en el archivo descargado para iniciar la instalación. El archivo puedes encontrarlo en la carpeta “Descargas”
4. Aceptar los permisos: Durante la instalación, debes aceptar los permisos requeridos por la aplicación para funcionar correctamente.
5. Finalizar la instalación: Después de aceptar los permisos, la instalación debería completarse en poco tiempo. Una vez finalizada, puedes encontrar la aplicación en tu menú de aplicaciones.

Recuerda que para instalar una APK en un dispositivo Android, es necesario tener una versión compatible de Android en tu dispositivo. Si tienes dudas o problemas con la instalación, puedes consultar la página de soporte del desarrollador o buscar soluciones en línea.

Manual de usuario

La aplicación es muy intuitiva a la hora de interactuar con ella:

1. Regístrate como usuario
2. Pulsa sobre el botón añadir sendero para añadir tu nueva ruta
3. Rellena los campos con la información de la que dispongas sobre tu ruta
4. Cuando termines, dale a guardar
5. Ahora te aparece tu primer sendero en la lista de senderos
6. Puedes guardarlo como favorito
7. Si ya has realizado el sendero, puedes guardarlo como completado
8. Si haces clic sobre la pestaña de ubicación se abrirá un mapa con la ubicación del sendero

Tabla de especificación de casos de uso

Caso de uso Login	Número CU-1		
Actores	Usuario no Logueado		
Descripción	El usuario debe ser capaz de iniciar sesión en el sistema y acceder a los datos internos de la aplicación		
Precondición			
Secuencia Normal	Paso	Acción	
		Actor	Sistema
	1	Accede a la sesión login	
	2	Introduce los datos	
	3		El sistema envía a comprobar los datos – CU-2

Caso de uso Comprobación Login	Número CU-2		
Actores	Usuario no Logueado		
Descripción	Comprobación de Login		
Precondición	Existencia de datos para comprobar		
Secuencia Normal	Paso	Acción	
		Actor	Sistema
	1		El sistema comprueba los datos introducidos por el usuario
	2		El sistema da la bienvenida al usuario
	3		El sistema permite el acceso a la sección privada
Casos Alternativos	Paso	Acción	
		Actor	Sistema

	2a		El sistema informa de los datos incorrectos
	2b	El usuario introduce nuevos datos	
	2c		Vuelta al paso CU-1

Caso de uso Registro		Número CU-3	
Actores	Usuario no Logueado		
Descripción	El usuario debe ser capaz de registrarse en el sistema añadiendo los datos requeridos		
Precondición			
Secuencia Normal	Paso	Acción	
		Actor	Sistema
	1	Accede a la sección de registro	
	2	Rellena el formulario	
	3		El sistema envía a comprobar los datos a CU-4

Caso de uso Comprobación Registro		Número CU-4	
Actores	Usuario no Logueado		
Descripción	El sistema debe comprobar los datos de registro		
Precondición	Existencia de datos para comprobar		
Secuencia Normal	Paso	Acción	
		Actor	Sistema
	1		El sistema comprueba los datos introducidos por el usuario
	2		El sistema informa que el usuario se ha registrado correctamente

	3		El sistema redirige a la sección de login CU-1
Casos Alternativos	Paso	Acción	
		Actor	Sistema
	2a		El sistema informa de los datos incorrectos
	2b	El usuario introduce nuevos datos	
	2c		Vuelta al paso CU-4

Caso de uso Crear Sendero		Número CU-5	
Actores	Usuario Logueado		
Descripción	Crear nuevo sendero		
Precondición			
Secuencia Normal	Paso	Acción	
		Actor	Sistema
	1	Accede a la sección crear senderos	
	2	Rellena los campos solicitados	
	3	Selecciona un municipio – CU-7	
	4	Envía los datos	
	5		El sistema comprueba los datos
	6		El sistema guarda el sendero creado
Casos Alternativos	Paso	Acción	
		Actor	Sistema
	3a	Selecciona crear un municipio – CU-6	
	2b		El sistema informa de que los datos son incorrectos

	2c	Introduce nuevos datos	
	2d		El sistema se dirige al CU-6

Caso de uso Crear municipio		Número CU-6	
Actores	Usuario Logueado		
Descripción	Crear nuevo municipio		
Precondición	El municipio no existe		
Secuencia Normal	Paso	Acción	
		Actor	Sistema
	1	Accede a la sección nuevo municipio	
	2	Rellena los datos solicitados	
	3		El sistema comprueba los datos
	4		El sistema guarda los datos creados
Casos Alternativos	Paso	Acción	
		Actor	Sistema
	3a		El sistema informa de que el municipio ya existe

Caso de uso Seleccionar municipio		Número CU-7	
Actores	Usuario Logueado		
Descripción	Seleccionar un municipio		
Precondición	El municipio ha sido creado		
Secuencia Normal	Paso	Acción	
		Actor	Sistema
	1		El sistema muestra un desplegable con la lista de municipios disponibles

	2	El usuario selecciona el municipio deseado	
--	---	--	--

Caso de uso Seleccionar sendero		Número CU-8	
Actores	Usuario Logueado		
Descripción	Seleccionar un sendero		
Precondición	El sendero ha sido creado		
Secuencia Normal	Paso	Acción	
		Actor	Sistema
	1		Muestra los senderos guardados por el usuario
	2	Selecciona un sendero de la lista	
	3		Muestra la información del sendero

Caso de uso Editar sendero		Número CU-9	
Actores	Usuario Logueado		
Descripción	Editar un sendero		
Precondición	Tener creado un sendero		
Secuencia Normal	Paso	Acción	
		Actor	Sistema
	1	Selecciona editar un sendero	
	2	Edita los campos deseados	
	3	Selecciona un municipio – CU-7	
	4	Envía los datos	
	5		El sistema comprueba los datos
	6		El sistema guarda el sendero

	7		El sistema informa que el sendero se ha guardado correctamente
Casos Alternativos	Paso	Acción	
		Actor	Sistema
	3a	Selecciona crear un municipio – CU-6	
	4a		El sistema informa de que los datos son incorrectos
	6a	Introduce nuevos datos	
	6b		Redirige al punto CU-5

Caso de uso Eliminar sendero		Número CU-10	
Actores	Usuario Logueado		
Descripción	Eliminar un sendero		
Precondición	Tener creado un sendero		
Secuencia Normal	Paso	Acción	
		Actor	Sistema
	1	Selecciona eliminar sendero	
	2		Muestra un mensaje de confirmación
	3	Acepta	
	4		Elimina el sendero del sistema

Caso de uso		Número	
Marcar favorito		CU-11	
Actores	Usuario Logueado		
Descripción	Permite marcar un sendero como favorito		
Precondición	Tener creado un sendero		
Secuencia Normal	Paso	Acción	
		Actor	Sistema

	1	Marcar sendero como favorito	
	2		Muestra un mensaje de confirmación

Caso de uso Eliminar favorito		Número CU-12	
Actores	Usuario Logueado		
Descripción	Permite desmarcar un sendero como favorito		
Precondición	Tener creado un sendero y marcado como favorito		
Secuencia Normal	Paso	Acción	
		Actor	Sistema
	1	Desmarcar sendero como favorito	
	2		Muestra un mensaje de confirmación

Caso de uso		Número	
Marcar como completado		CU-13	
Actores	Usuario Logueado		
Descripción	Permite marcar un sendero como completado		
Precondición	Tener creado un sendero		
Secuencia Normal	Paso	Acción	
		Actor	Sistema
	1	Marcar sendero como completado	
	2		Muestra un mensaje de confirmación

Caso de uso		Número	
Desmarcar como completado		CU-14	
Actores	Usuario Logueado		
Descripción	Permite desmarcar un sendero como completado		
Precondición	Tener creado un sendero y marcado como completado		
Secuencia Normal	Paso	Acción	
		Actor	Sistema

	1	Desmarcar sendero como completado	
	2		Muestra un mensaje de confirmación

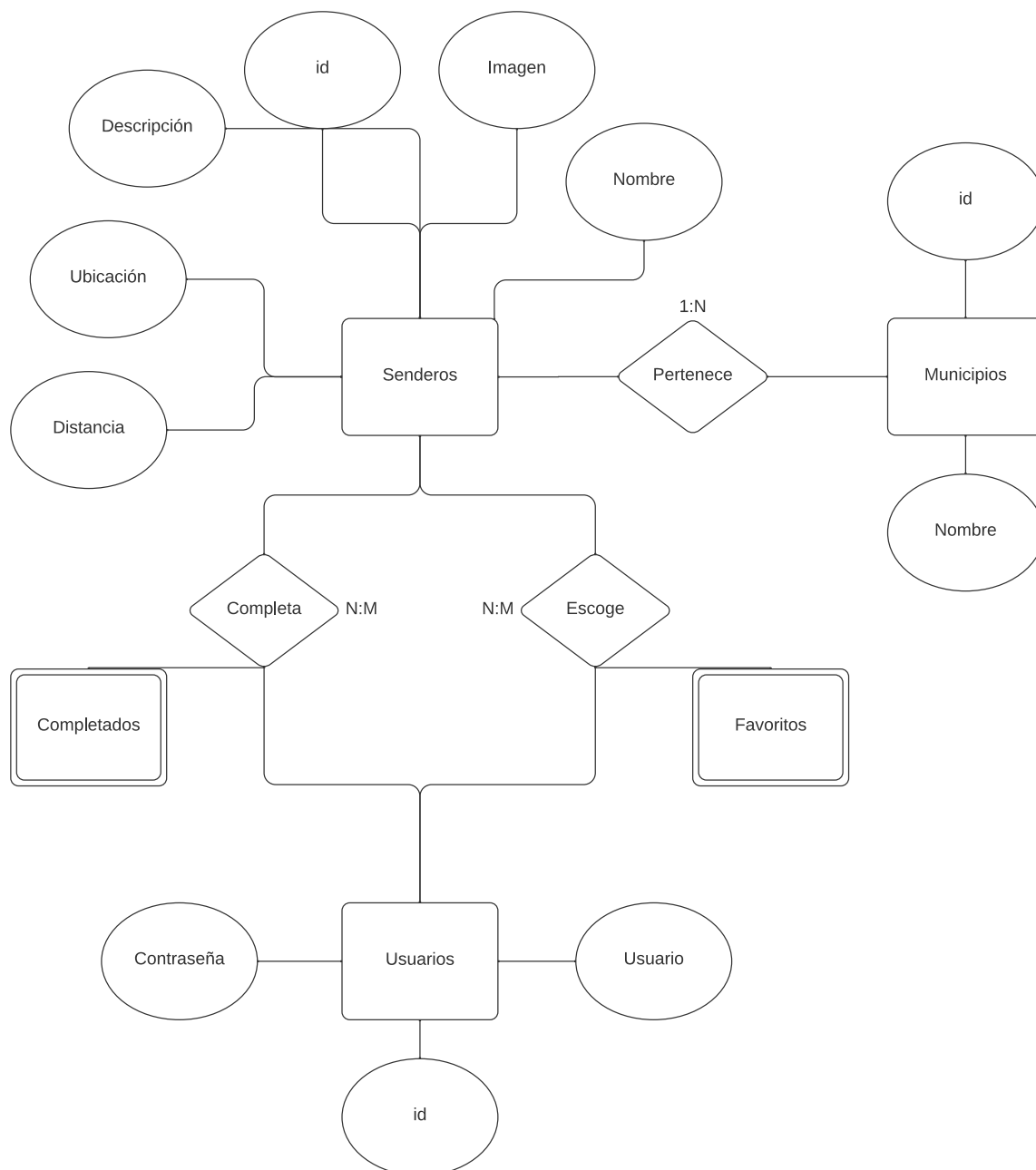
Relación de imágenes del proyecto



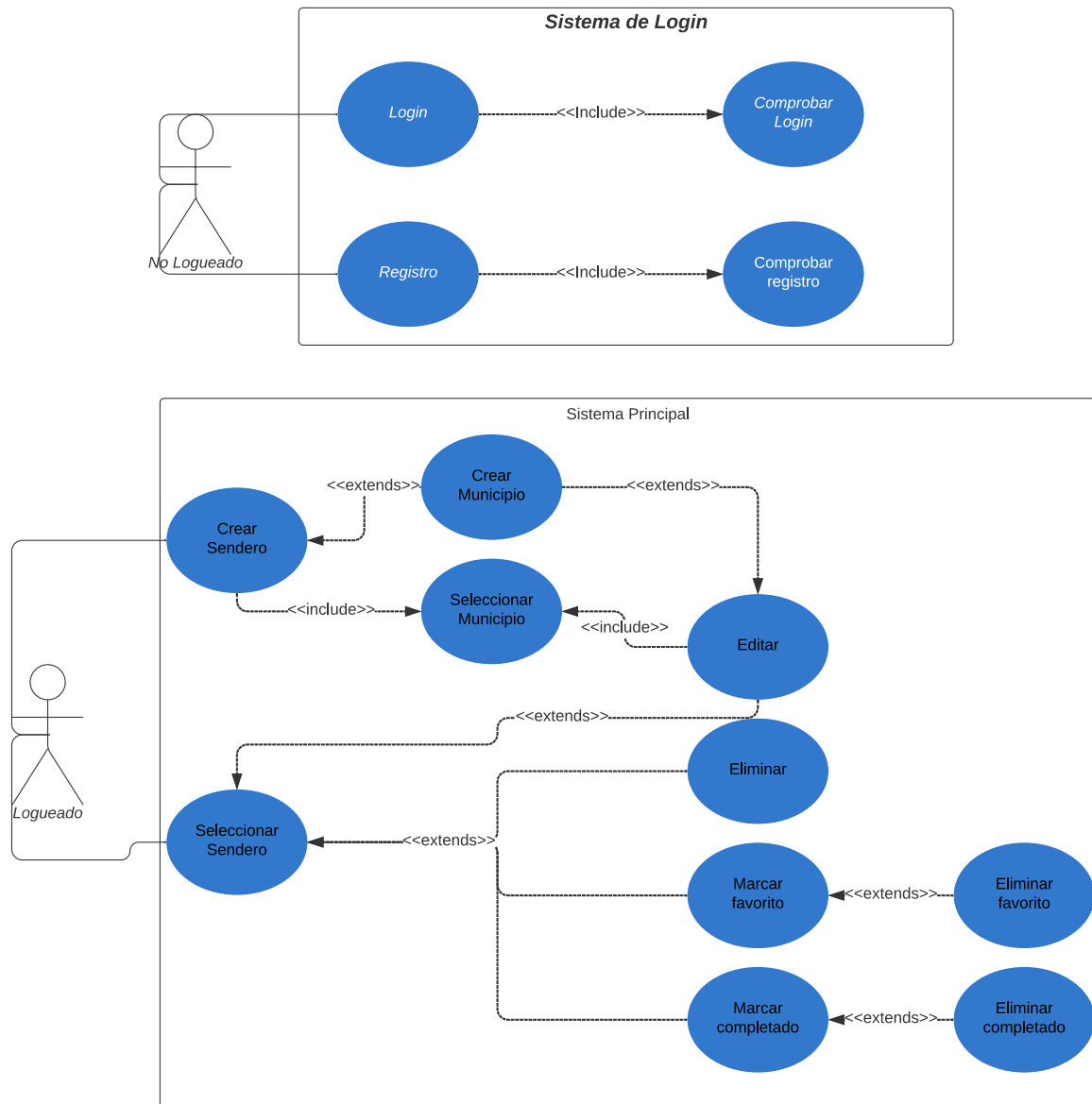
3 Imagen Diagrama de Gantt estimado. Fuente: Realización propia en canva.com



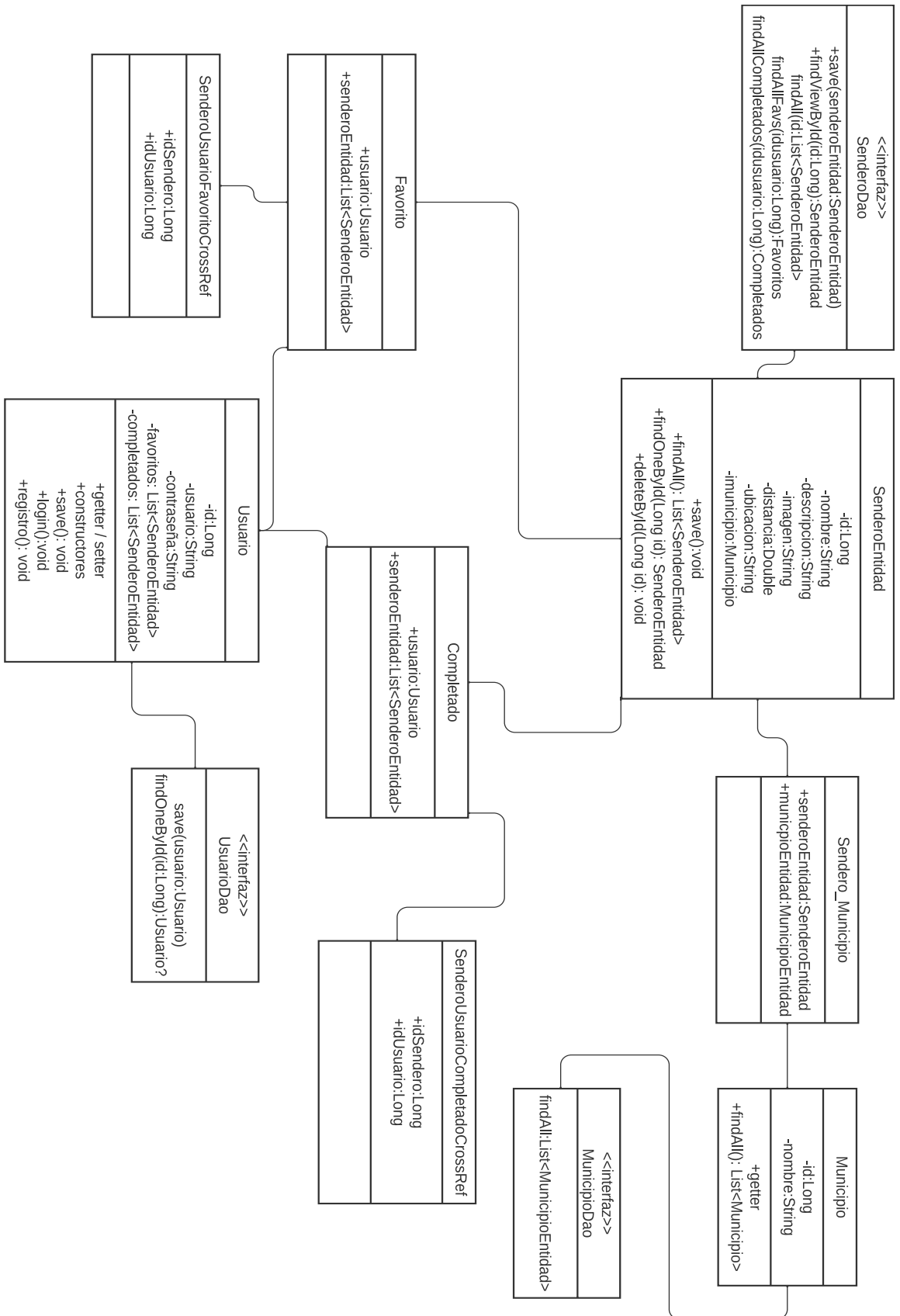
4 Imagen Diagrama de Gantt real. Fuente: Realización propia en canva.com



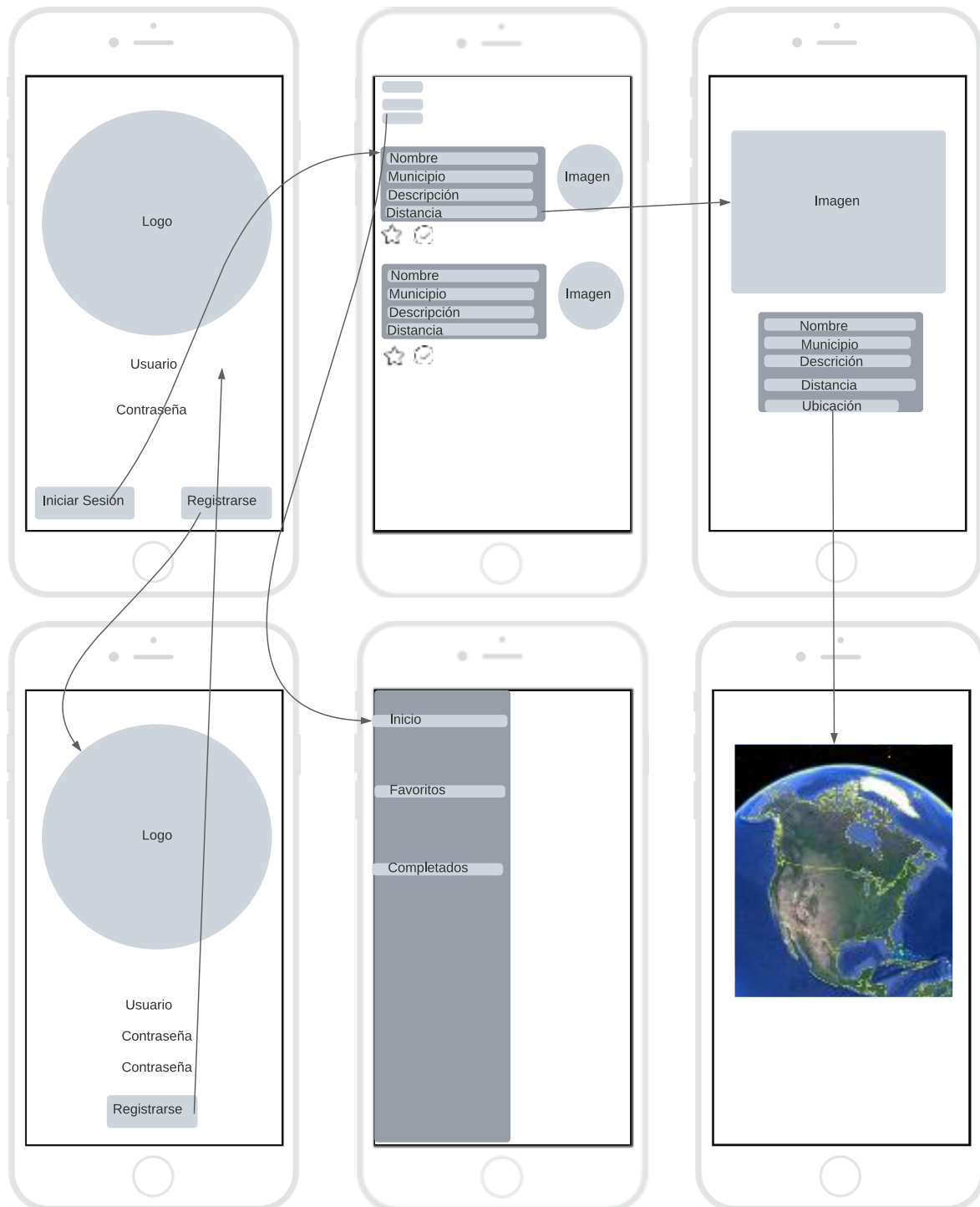
5 Imagen Diagrama ER. Elaboración propia en Lucidchart



6 Imagen Diagrama de Casos de Uso. Realización propia en Lucidchart



7 Imagen Diagrama de Clases. Elaboración propia en Lucidchart



8 Imagen Wireframe para dispositivos móviles. Realización propia en Lucidchart

