

**MyFinance**

Autor: Miguel Reyes Gómez

Tutor: David de la Torre Olegario

25/05/2025

Grado Superior Diseño de Aplicaciones Multiplataforma

Índice

[1. Resumen 4](#_Toc198648841)

[2. Abstract 5](#_Toc198648842)

[3. Autenticidad de la información 6](#_Toc198648843)

[3.1 Declaración de Autenticidad del Trabajo 6](#_Toc198648844)

[3.2 Origen del Contenido 6](#_Toc198648845)

[3.3 Proceso de Desarrollo 6](#_Toc198648846)

[3.4 Verificación de la Autenticidad 6](#_Toc198648847)

[3.5 Conclusión 6](#_Toc198648848)

[4. Motivación 7](#_Toc198648849)

[4.1 Motivación personal 7](#_Toc198648850)

[4.2 Motivación profesional 7](#_Toc198648851)

[5. Objetivos del proyecto 8](#_Toc198648852)

[5.1 Objetivo general 8](#_Toc198648853)

[5.2 Objetivos específicos 8](#_Toc198648854)

[6. Estado del arte 9](#_Toc198648855)

[6.1 Money Manager 9](#_Toc198648856)

[6.2 Wallet 9](#_Toc198648857)

[6.3 Diferenciación de MyFinance 9](#_Toc198648858)

[7. Propuesta 10](#_Toc198648859)

[7.1 Contextualización del Dominio y Marco Tecnológico 10](#_Toc198648860)

[7.2 Propuesta de Valor y Contribución al Ecosistema 10](#_Toc198648861)

[7.3 Innovaciones y Características Diferenciadoras 10](#_Toc198648862)

[8. Análisis del problema 11](#_Toc198648863)

[8.1 Especificación de requisitos 11](#_Toc198648864)

[8.2 Requisitos funcionales 11](#_Toc198648865)

[8.3 Requisitos no funcionales 12](#_Toc198648866)

[8.4 Arquitectura Conceptual del Sistema 12](#_Toc198648867)

[8.4.1 Diagrama de casos de uso 12](#_Toc198648868)

[8.4.2 Diagrama de flujo 15](#_Toc198648869)

[8.4.3 Diagramas de secuencia 18](#_Toc198648870)

[9. Identificación y análisis de soluciones posibles 20](#_Toc198648871)

[9.1 Desarrollo nativo 20](#_Toc198648872)

[9.2 Desarrollo Multiplataforma 21](#_Toc198648873)

[9.3 Establecimiento de criterios de selección 21](#_Toc198648874)

[9.4 Selección de la solución adecuada 21](#_Toc198648875)

[10. Solución propuesta 22](#_Toc198648876)

[10.1 Descripción de la solución 22](#_Toc198648877)

[10.2 Fases del desarrollo 23](#_Toc198648878)

[10.3 Implementación y validación 23](#_Toc198648879)

[11. Diseño de la solución 24](#_Toc198648880)

[11.1 Arquitectura de capas 24](#_Toc198648881)

[11.2 Componentes reutilizables 24](#_Toc198648882)

[11.3 Flujo de navegación 25](#_Toc198648883)

[11.4 Gestión de estados y reactividad 25](#_Toc198648884)

[12. Diseño detallado 25](#_Toc198648885)

[13. Tecnologías utilizadas 26](#_Toc198648886)

[13.1 Entorno de desarrollo 26](#_Toc198648887)

[13.2 Lenguaje de desarrollo 26](#_Toc198648888)

[13.3 Sistema operativo 26](#_Toc198648889)

[13.4 Frameworks y bibliotecas 26](#_Toc198648890)

[13.5 Control de versiones 26](#_Toc198648891)

[14. Desarrollo de solución propuesta 27](#_Toc198648892)

[14.1 Proceso de desarrollo 27](#_Toc198648893)

[14.2 Problemas y dificultades 27](#_Toc198648894)

[14.3 Decisiones tomadas 27](#_Toc198648895)

[14.4 Particularidades de la solución final 27](#_Toc198648896)

# Resumen

MyFinance es una aplicación móvil desarrollada para el Trabajo de Fin de Grado, cuyo objetivo principal es facilitar el control financiero personal de forma práctica, accesible e intuitiva.

Esta app está dirigida al público general y permite registrar y categorizar tanto ingresos como gastos, ofreciendo una visión clara del estado económico del usuario mediante estadísticas personalizadas, un historial de movimientos recientes y una “hucha virtual” que se actualiza automáticamente con cada operación. Además, incluye una calculadora de interés compuesto personalizable, gestión de perfil con nombre de usuario y foto, y la opción de alternar entre modo claro y oscuro para mejorar la experiencia.

La metodología de desarrollo la organicé en tres puntos clave. Análisis de requisitos, diseño de interfaz y codificación y validación mediante pruebas.

Para la implementación se emplearon Kotlin y Jetpack Compose en Android Studio, con SQLite como base de datos local. El control de versiones se gestionó a través de GitHub.

En conclusión, este proyecto forma parte del ciclo formativo de Desarrollo de Aplicaciones Multiplataforma (DAM), donde he podido aplicar lo aprendido sobre programación, bases de datos y diseño de interfaces. Más allá del desarrollo técnico, esta aplicación refleja mi interés por crear herramientas útiles que puedan ayudar a otras personas a gestionar mejor su economía del día a día.

# Abstract

MyFinance is a mobile application developed for the Final Degree Project, whose main objective is to facilitate personal financial management in a practical, accessible and intuitive way.

This app is aimed at the general public and allows users to record and categorize both income and expenses, offering a clear view of their financial status through personalized statistics, a history of recent transactions, and a "virtual piggy bank" that updates automatically with each transaction. It also includes a customizable compound interest calculator, profile management with a username and photo, and the option to switch between light and dark mode to improve the experience.

I organized the development methodology into three key points: requirements analysis, interface design, and coding and validation through testing.

I organized the work in three steps: first I analyzed the requirements, then I designed the screens, and finally I built and tested the app. I wrote the code in Android Studio using Kotlin and Jetpack Compose, and I used SQLite for the database. Version control was managed through GitHub and all my code is there.

In conclusion, this project is part of the Desarrollo de Aplicaciones Multiplataforma (DAM). I used what I learned about coding, databases, and design. Beyond the technical side, MyFinance shows my wish to create helpful tools that make everyday money management easier.

# Autenticidad de la información

## 3.1 Declaración de Autenticidad del Trabajo

Yo, Miguel Reyes Gómez [09147403G], autor del Trabajo de Fin de Grado (TFG), declaro que el presente proyecto titulado “MyFinance”, es fruto de mi trabajo personal e inédito. Todos los contenidos, datos y resultados expuestos han sido generados por mí y no provienen de ninguna fuente ajena.

## 3.2 Origen del Contenido

El material presentado en este TFG ha sido desarrollado íntegramente por el autor, sin recurrir indebidamente a ideas, textos o recursos de terceros. Los algoritmos, diseños y soluciones mostrados se han programado y documentado desde cero.

## 3.3 Proceso de Desarrollo

El diseño, la implementación y las pruebas de esta aplicación se han llevado a cabo de forma individual. He dedicado tiempo a investigar las mejores prácticas, diseñar la interfaz, codificar cada módulo y ejecutar pruebas unitarias e integradas para validar su correcto funcionamiento.

## 3.4 Verificación de la Autenticidad

He revisado detalladamente cada sección de este documento y comprobado que no existe plagio ni apropiación indebida de trabajos previos. Asimismo, he sometido el texto a herramientas de detección de similitudes para asegurar su originalidad.

## 3.5 Conclusión

En virtud de lo expuesto, certifico que este Trabajo de Fin de Grado es fruto de mi esfuerzo personal y original, y asumo plena responsabilidad por su integridad. Me comprometo a responder ante cualquier eventualidad relacionada con el cumplimiento de los principios éticos y académicos establecidos.

# Motivación

## 4.1 Motivación personal

Mi motivación para emprender este Trabajo de Fin de Grado (TFG), surge de mi inquietud por profundizar en Kotlin y Jetpack Compose, ambos los descubrí durante el ciclo y me resultaron muy atractivos para el desarrollo móvil.

Al pensar en el tema de mi Trabajo de Fin de Grado, me resultó muy fácil elegirlo: el mundo de las finanzas personales siempre me ha llamado la atención, cómo las personas organizan su presupuesto y qué herramientas utilizan para ello. Quise unir ambas pasiones, la tecnología y las finanzas, en una sola aplicación, y así nació la idea de MyFinance.

Trabajar en un proyecto que combina un lenguaje con el que me siento cómodo y un tema que me apasiona me ha mantenido motivado en cada fase.

En definitiva, MyFinance representa para mí la oportunidad perfecta de consolidar mis habilidades en Kotlin, al tiempo que desarrollo una herramienta útil vinculada a una disciplina que siempre me ha interesado.

## 4.2 Motivación profesional

Desde un enfoque profesional, este proyecto representa una oportunidad única para afianzar mis competencias en el diseño y construcción de aplicaciones Android. La experiencia práctica que obtenga al desarrollar esta app reforzará mi perfil como desarrollador especializado en el ecosistema Android, un sector con alta demanda en la industria tecnológica.

Además, trabajar con Android Studio y aplicar las mejores prácticas de desarrollo en esta plataforma son aspectos esenciales para mi crecimiento profesional. Gracias a este proyecto, no solo pongo en práctica mis habilidades técnicas, sino que demuestro mi capacidad para superar retos complejos y entregar soluciones innovadoras en el ámbito del desarrollo móvil para Android.

# 5. Objetivos del proyecto

## 5.1 Objetivo general

El propósito principal de este Trabajo de Fin de Grado es desarrollar una aplicación móvil que ofrezca una solución integral para la gestión de finanzas personales, combinando una interfaz intuitiva y accesible con funcionalidades avanzadas de análisis y proyección económica. La herramienta permitirá al usuario registrar, categorizar y visualizar sus ingresos y gastos de forma clara y estructurada; planificar ahorros mediante una calculadora de interés compuesto totalmente personalizable; y gestionar su perfil.

## 5.2 Objetivos específicos

1. **Registro y categorización de datos:** Se crearán pantallas de entrada para añadir ingresos y gastos con fecha, monto y categoría. Estos datos se almacenarán en SQLite mediante esquemas bien definidos y clases DAO en Kotlin. Se incluirán validaciones de entrada y manejo de errores para asegurar la calidad de la información. Además, se optimizarán las consultas para mantener un rendimiento fluido con muchos registros.
2. **Diseño de interfaz con Jetpack Compose:** Se aplicará Material Design en Jetpack Compose con un header y una barra de navegación comunes en las cinco pantallas (Home, Historial, Añadir transacciones, Calculadora y Perfil). Cada pantalla compartirá estilos de paleta, tipografía y espaciado para garantizar coherencia visual. Los componentes se desarrollarán de forma modular para facilitar su mantenimiento y adaptación.
3. **Gestión de perfil y preferencias:** Se desarrollará una sección de perfil donde el usuario podrá editar su nombre y subir una foto, guardando la información en la base de datos local. También contará con un selector para alternar entre modo oscuro y claro.

# 6. Estado del arte

La sección de estado del arte revisa las soluciones y conocimientos existentes en el ámbito de las aplicaciones para finanzas personales, sirviendo de base para identificar carencias y oportunidades de mejora.

## 6.1 Money Manager

Money Manager destaca por sus avanzadas herramientas de análisis de gastos, que incluyen gráficos detallados por categoría y períodos. Ofrece plantillas de presupuesto predefinidas y permite sincronizar cuentas bancarias mediante importación automática de CSV. Aunque tiene buena reputación por su profundidad analítica, puede resultar complejo para usuarios comunes debido a la gran cantidad de opciones y ajustes disponibles.

## 6.2 Wallet

Wallet se centra en la gestión colaborativa de presupuestos, facilitando compartir cuentas familiares o de grupo. Incluye recordatorios de facturas y plantillas de presupuestos, así como sincronización con bancos para importación de movimientos. Su interfaz es limpia, pero algunas funciones avanzadas requieren suscripción de pago, lo que puede limitar a usuarios con necesidades básicas.

## 6.3 Diferenciación de MyFinance

A diferencia de Money Manager y Wallet, MyFinance apuesta por un enfoque 100% local, sin depender de la nube ni de suscripciones. Su “hucha virtual” gamificada y la calculadora de interés compuesto están integradas de serie, sin módulos de pago extra. Además, MyFinance ofrece un diseño minimalista basado en Material Design y un esquema de navegación unificado en cinco pantallas, garantizando una curva de aprendizaje suave para cualquier usuario. La alternancia inmediata entre modo claro y oscuro y la gestión de perfil local subrayan su compromiso con la privacidad y la simplicidad.

# 7. Propuesta

## 7.1 Contextualización del Dominio y Marco Tecnológico

Este Trabajo de Fin de Grado se posiciona en el contexto del desarrollo de aplicaciones móviles enfocadas a la gestión de finanzas personales, un área con creciente demanda de soluciones accesibles y seguras para el usuario. Aunque existen numerosas apps que ofrecen registro de transacciones y análisis de gastos, muchas dependen de la nube o presentan interfaces complejas que dificultan su adopción por el público general.

## 7.2 Propuesta de Valor y Contribución al Ecosistema

MyFinance se distingue por ofrecer privacidad total y gran facilidad de uso, cubriendo registro, categorización y visualización de gastos sin depender de suscripciones o servicios en la nube. Al emplear Kotlin y Jetpack Compose, demuestra cómo crear apps Android modulares, accesibles y preparadas para futuras extensiones manteniendo siempre los datos en el dispositivo.

## 7.3 Innovaciones y Características Diferenciadoras

1. **Hucha virtual:** un elemento visual que motiva el ahorro al crecer dinámicamente con cada aportación.
2. **Calculadora de interés compuesto personalizable:** permite al usuario simular escenarios de ahorro.
3. **Diseño uniforme y accesible:** header y barra de navegación constantes en las cinco pantallas principales, con soporte para modo claro/oscuro y elementos de accesibilidad.
4. **Gestión de perfil local:** nombre, foto y preferencias de tema se almacenan en el dispositivo, evitando servicios de terceros y reforzando la confianza del usuario.

# 8. Análisis del problema

El análisis del problema identifica las carencias y dificultades que enfrentan los usuarios al gestionar sus finanzas personales con las soluciones existentes, sirviendo de base para justificar el desarrollo de MyFinance.

## 8.1 Especificación de requisitos

Se llevará a cabo una definición exhaustiva de los requerimientos del sistema, que servirá de guía para el diseño y la implementación de MyFinance. Estos requisitos se estructurarán en dos bloques claramente diferenciados y documentados.

## 8.2 Requisitos funcionales

1. **Registro de transacciones:** El usuario podrá añadir gastos e ingresos, indicando monto, descripción y categoría.
2. **Categorización de operaciones:** Se ofrecerán categorías predefinidas para ingresos y gastos.
3. **Historial de transacciones:** Dispondrá de una pantalla donde se listarán todas las operaciones realizadas, con opciones de filtrado por tipo.
4. **Visualización de estadísticas:** La aplicación generará gráficos que resuman ingresos y gastos.
5. **Calculadora de interés compuesto:** El usuario podrá introducir capital inicial, tasa de interés, plazo y aportación mensual, obteniendo un gráfico con resultados y proyecciones.
6. **Gestión de perfil:** Permitirá editar el nombre de usuario, subir o cambiar foto de perfil y seleccionar entre modo oscuro/claro, almacenado localmente.

## 8.3 Requisitos no funcionales

1. **Privacidad y almacenamiento local:** Todos los datos del usuario deben mantenerse exclusivamente en el dispositivo, sin servicios en la nube.
2. **Rendimiento:** El sistema ofrecerá un rendimiento rápido y fluido en todas las operaciones de lectura y escritura.
3. **Usabilidad y accesibilidad:** Interfaz intuitiva y fácil de usar para todos los usuarios, con soporte de contraste y etiquetas para lectores de pantalla.

## 8.4 Arquitectura Conceptual del Sistema

Antes de iniciar con los diagramas, se llevará a cabo un modelado conceptual del sistema empleando UML (Unified Modeling Language). Se utilizarán diagramas de casos de uso, de secuencia y de flujo para representar gráficamente las estructuras y relaciones internas, así como la interacción entre los usuarios y los distintos componentes del sistema.

### 8.4.1 Diagrama de casos de uso

En este diagrama vemos de un vistazo cómo el usuario se relaciona con las funcionalidades clave de MyFinance y cómo unas conducen a otras:

1. **Actor “Usuario”:** Representa a la persona que usa la app. De él parten líneas hacia todos los casos de uso principales.
2. **Relaciones <<include>>:** Las flechas punteadas con la etiqueta «include» indican funcionalidades obligatorias que siempre se ejecutan dentro de otro caso de uso:
   1. **Registrar Transacción: → Categorizar Transacción:**

Cada vez que el usuario registra una operación, el sistema debe asignarle una categoría.

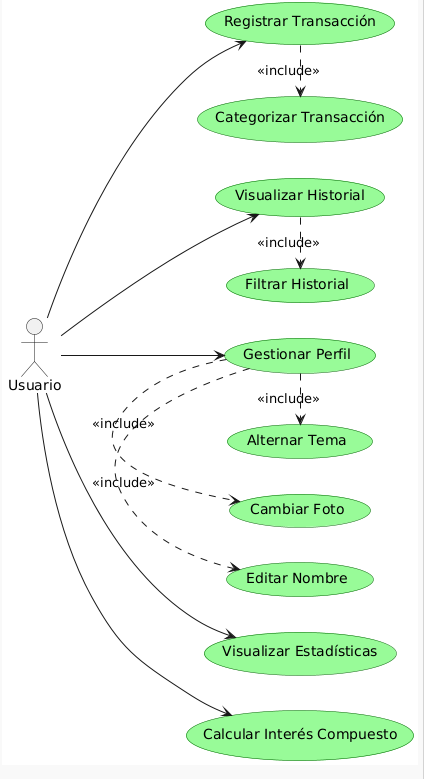
* 1. **Visualizar Historial → Filtrar Historial:**

Al ver el historial, el usuario puede filtrar por tipo de transacción.

* 1. **Gestionar Perfil → Editar Nombre**
  2. **Gestionar Perfil → Cambiar Foto**
  3. **Gestionar Perfil → Alternar Tema:**

Cada una de estas acciones es parte de la gestión del perfil.

Figura 1: Diagrama de casos de uso.



### 8.4.2 Diagrama de flujo

A continuación, describo paso a paso el recorrido del usuario según el diagrama de flujo de MyFinance:

**8.4.2.1 Inicio**: La app se inicia automáticamente en la pantalla de inicio.

**8.4.2.2 Pantalla Inicio:**

* **Función:** punto central de navegación.
* **Gráficos de estadísticas.**
* **Hucha virtual.**
* **Flujos posibles:**
  + Pantalla Historial
  + Pantalla Transacciones
  + Pantalla Calculadora
  + Pantalla Perfil

**8.4.2.3 Pantalla Historial:**

* **Función:** Muestra el listado completo de transacciones
* **Flujo:**
  + Al entrar, carga todas las entradas desde SQLite
  + El usuario puede filtrar por tipo de transacción

**8.4.2.3 Pantalla Transacciones:**

* **Función:** Formulario para registrar una nueva transacción
* **Pasos clave:**
  + El usuario introduce monto, descripción y categoría
  + La fecha es recogida automáticamente
  + Al registrar una transacción, se actualiza automáticamente tanto el historial como las estadísticas y huchas de la pantalla inicio.

**8.4.2.3 Pantalla Calculadora:**

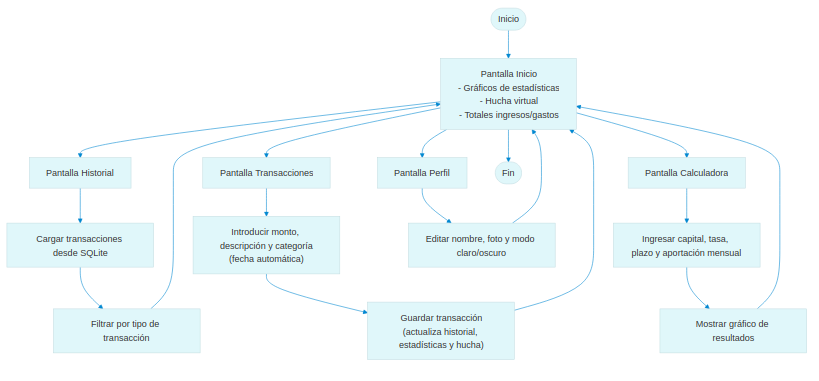
* **Función:** Simulador de interés compuesto
* **Pasos:**
  + El usuario indica el capital inicial, tasa, plazo y aportación mensual
  + Al pulsar en calcular, aparece un gráfico con los datos

**8.4.2.3 Pantalla Perfil:**

* **Función:** Gestión de los datos del usuario y el tema de la app
* **Pasos:**
  + Editar nombre de usuario
  + Cambiar foto de perfil
  + Alternar entre modo claro y oscuro

**8.4.2.4 Finalización del flujo**

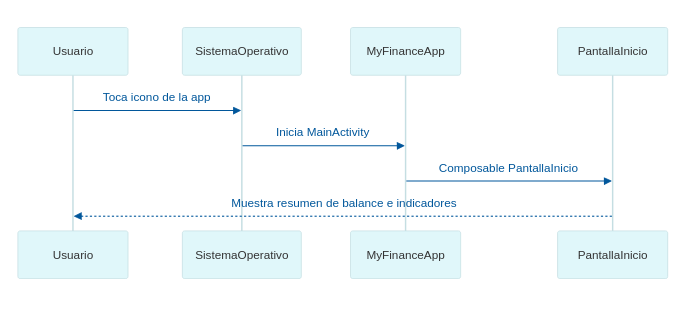
El usuario puede navegar libremente entre todas las pantallas en cualquier momento gracias a la barra de navegación permanente. Cada acción que implique operaciones en la base de datos actualiza automáticamente todos los elementos afectados de la aplicación. De este modo no existe un punto de cierre estricto: el flujo termina cuando el usuario decide cerrar la aplicación.

Figura 2: Diagrama de flujo

### 8.4.3 Diagramas de secuencia

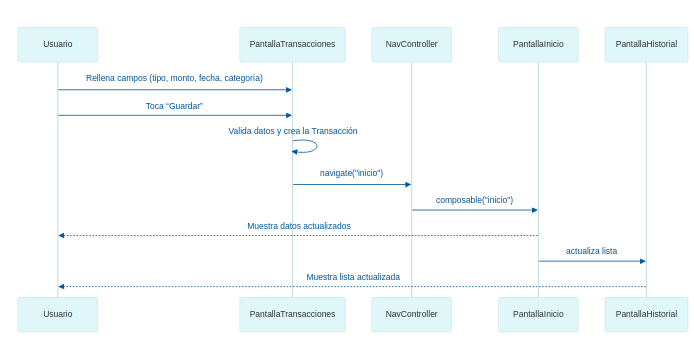
* Abrir la app y ver Home

1. Usuario abre la app
2. El sistema operativo inicia la app
3. MainActivity renderiza la pantalla de inicio
4. PantallaInicio muestra los datos al usuario

****

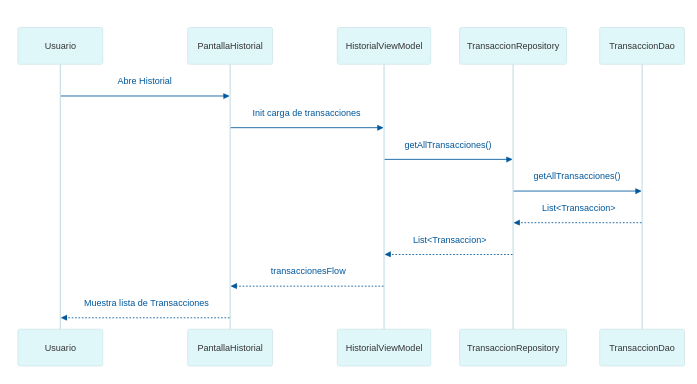
* Crear y guardar una nueva Transacción

1. El usuario rellena los campos (tipo, monto, fecha, categoría)
2. El usuario toca “Guardar”
3. PantallaTransacciones valida datos y crea la Transacción
4. PantallaInicio muestra actualiza sus datos
5. PantallaHistorial muestra la lista actualiza

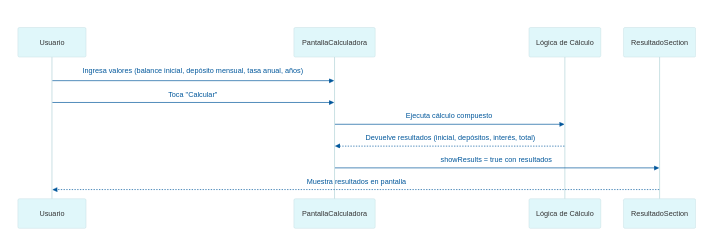
****

* Visualizar la lista de Transacciones en Historial

1. Usuario abre Historial
2. PantallaHistorial solicita transacciones al HistorialViewModel
3. HistorialViewModel invoca getAllTransacciones en TransaccionRepository
4. TransaccionRepository llama a TransaccionDao.getAllTransacciones()
5. TransaccionDao devuelve la lista de transacciones
6. TransaccionRepository retorna la lista a HistorialViewModel
7. HistorialViewModel emite el flujo de transacciones
8. PantallaHistorial muestra la lista de transacciones

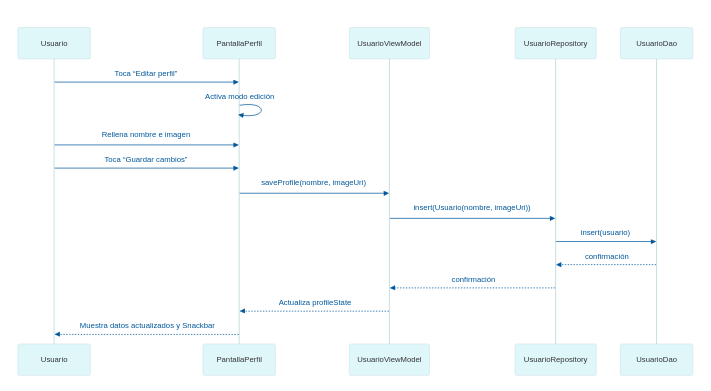
****

* Calcular el resultado en la Calculadora

1. Usuario ingresa valores
2. Usuario toca “Calcular”
3. PantallaCalculadora ejecuta el cálculo compuesto
4. ResultadoSection muestra los resultados calculados

* Actualizar datos en Perfil de Usuario

1. Usuario toca “Editar perfil”
2. PantallaPerfil activa modo edición
3. Usuario modifica nombre y/o foto
4. Usuario toca “Guardar cambios”
5. UsuarioViewModel guarda perfil (inserta o actualiza usuario)
6. PantallaPerfil muestra datos actualizados y un Snackbar de confirmación



# 

# 9. Identificación y análisis de soluciones posibles

# 9.1 Desarrollo nativo

* **Ventajas**
  + Rendimiento óptimo y acceso completo a APIs del sistema
  + Experiencia de usuario muy fluida y coherente con la plataforma
  + Facilidad para implementar actualizaciones incrementales
* **Inconvenientes**
  + Solo accesible en Android
  + Requiere dominar Kotlin/Java

## 

## 9.2 Desarrollo Multiplataforma

* **Ventajas**
  + Con una única base de código se generan apps para Android e iOS.
  + Ahorro de tiempo en desarrollo
* **Inconvenientes**
  + Rendimiento inferior al nativo

## 9.3 Establecimiento de criterios de selección

* **Privacidad y control de datos:** grado en que los datos del usuario permanecen en su dispositivo.
* **Rendimiento y experiencia de usuario:** capacidad de la app para responder sin latencias o bloqueos.
* **Alcance de plataforma:** cobertura de dispositivos (solo Android vs. Android e iOS).
* **Coste y complejidad de desarrollo:** recursos y tiempo necesarios para implementación y mantenimiento.
* **Escalabilidad futura:** facilidad para añadir nuevas funcionalidades o integraciones (API bancarias, análisis predictivo).

## 9.4 Selección de la solución adecuada

* **Privacidad y control de datos:** gana el desarrollo nativo, ya que no requiere backend ni servicios externos.
* **Rendimiento y experiencia:** la opción nativa ofrece los mejores tiempos de respuesta y optimizaciones específicas de Android.
* **Alcance de plataforma:** el multiplataforma cubre más dispositivos, pero el TFG se centra en una aplicación para Android.
* **Coste y complejidad:** el desarrollo nativo simplifica el alcance y reduce las capas tecnológicas, ideal para un proyecto académico.
* **Escalabilidad futura:** aunque multiplataforma facilita exportar a iOS, la arquitectura modular propuesta en Kotlin permite más adelante un incremento de plataforma.

**Solución seleccionada:**

Se elige un desarrollo nativo con Kotlin, Jetpack Compose y SQLite. Esto garantiza la máxima privacidad, un rendimiento fluido y un coste de desarrollo y mantenimiento ajustado a los recursos de un TFG, manteniendo la puerta abierta a futuras migraciones o ampliaciones.

# 10. Solución propuesta

Esta sección detalla la alternativa final seleccionada **desarrollo nativo** y su plan de ejecución, dividiéndola en tres bloques clave: descripción general, fases de desarrollo e implementación y validación.

## 

## 10.1 Descripción de la solución

**MyFinance** es una aplicación Android nativa, desarrollada en Kotlin con Jetpack Compose, que gestiona las finanzas personales íntegramente en local. Emplea Room (SQLite) para el almacenamiento, SharedPreferences para ajustes de tema y perfil, y MVVM para separar capas. Los módulos principales son:

1. **Registro y categorización de ingresos y gastos**

**1.1 Entidades:** Transaccion.kt y Categoria.kt

**1.2 DAO/Repositorios:**

TransaccionDao.kt →TransaccionRepository.kt

CategoriaDao.kt → CategoriaRepository.kt

1. **Visualización de historial y estadísticas**

**2.1 Pantallas:** PantallaHistorial.kt y PantallaInicio.kt

**2.2 ViewModels:** TransaccionViewModel.kt y BalanceViewModel.kt

1. **Hucha virtual y estadísticas totales**

El cálculo del balance total y sumas por categoría en BalanceViewModelFactory.kt y BalanceViewModel.kt

1. **Calculadora de interés compuesto**

**4.1 UI:** PantallaCalculadora.kt

**4.2 Lógica:** el método CalculadoraHelper

1. **Gestión de perfil**

**5.1 Pantalla y ViewModel:** PantallaPerfil.kt y UsuarioViewModel.kt

**5.2 Persistencia:** UsuarioDao.kt → UsuarioRepository.kt y ThemeViewModel.kt para el tema de la app

## 10.2 Fases del desarrollo

* **Planificación y diseño**
  + Definí las funciones de la app y realicé bocetos de cada pantalla: inicio, historial, transacciones, calculadora y perfil.
* **Construcción de la base de la aplicación**
  + Monté la estructura interna para guardar datos y hacerlos fluir correctamente
  + Desarrolle los componentes comunes de las pantallas, la cabecera y la barra de navegación
  + Implementé la pantalla de inicio para mostrar los resúmenes de ingresos y gastos, más tarde añadí el historial de transacciones
* **Desarrollo de las funciones clave**
  + Creé el formulario para añadir nuevas transacciones
  + Implementé la calculadora de interés compuesto
  + Añadí la sección de perfil
* **Pruebas**
  + Probé cada funcionalidad para asegurar que todo sea intuitivo y libre de errores.
  + Ajuste detalles de diseño e implementé el modo claro/oscuro

## 10.3 Implementación y validación

* **Implementación:**
  + Organicé el código en módulos claros para facilitar el mantenimiento.
  + Usé Jetpack Compose para construir las pantallas de forma declarativa y reactiva
  + Configuré Room para gestionar la base de datos local y SharedPreferences para guardar las preferencias de tema.
  + Desarrollé ViewModels que exponen flujos de datos a la interfaz, garantizando que la UI siempre esté sincronizada con la información interna.
* **Validación**
  + Escribí pruebas unitarias para la lógica de cálculo y para asegurar que el acceso a los datos funcionara correctamente.
  + Realicé sesiones de prueba recopilando opiniones sobre usabilidad y corrigiendo los puntos de fricción.

# 11. Diseño de la solución

En esta sección describo cómo he estructurado y organizado la aplicación para garantizar

claridad, escalabilidad y un mantenimiento fácil.

## 11.1 Arquitectura de capas

* **Capa de datos:** se encarga de almacenar y recuperar información. Utilizo Room (SQLite), para hacer transacciones, guardarlas por categorías y almacenar la información del usuario de manera local.
* **Capa de negocio:** contiene la lógica de la aplicación. Aquí tenemos los ViewModels, que son los que coordinan los datos de la base de datos.
* **Capa de presentación:** define la interfaz utilizando Jetpack Compose. Cada pantalla es un componente que se actualiza automáticamente al cambiar datos en la base de datos, gracias al ViewModel.

## 11.2 Componentes reutilizables

Para evitar duplicar código, creé componentes comúnes:

* **Header:** Es la barra superior que contiene el logo y el título de la aplicación
* **Barra de navegación:** Es la barra inferior que contiene los iconos de cada pantalla. Facilita el movimiento entre ellas.

## 11.3 Flujo de navegación

Usé un único NavHost que gestiona las rutas, al pulsar en un icono de la barra de navegación, llama al NavController de NavHost y cambia de pantalla de forma fluida, manteniendo el estado de los datos gracias el ViewModel.

## 11.4 Gestión de estados y reactividad

Cada ViewModel expone su información mediante StateFlow. Cuando se añade una transacción, se cambia el tema de la aplicación o cualquier dato de la base de datos cambia, la interfaz se actualiza automáticamente con los datos correctos.

# 12. Diseño detallado

En esta sección profundizo en cómo se estructura internamente cada módulo y cómo interactúan sus componentes para ofrecer la funcionalidad esperada:

* **Modelo de datos**
  + **Transacción:** objeto con campos **id, tipo, monto, fecha, categoría y descripción.**
  + **Categoría:** cada transacción referencia una categoría con **id, nombre y tipo.**
  + **Usuario:** contiene **id, nombre, email, foto** y preferencias de tema.
* **Repositorios**
  + Actúan como capa intermedia con la base de datos local.
  + Ofrecen métodos para crear, leer, actualizar y borrar entidades
  + Calculan totales y estadísticas y entregan los datos
* **ViewModels**
  + Son el punto de conexión entre los datos y la interfaz.
  + Al cargar una pantalla, piden y muestran la información necesaria.
* **UI y flujos de interacción**
  + Cada pantalla es independiente.
  + **Inicio:** Muestra totales y gráficos que se actualizan automáticamente.
  + **Transacciones:** formulario que permite registrar una transacción.
  + **Historial:** lista filtrable de las últimas transacciones.
  + **Calculadora:** permite calcular el interés que generaría una cuenta.
  + **Perfil:** posibilidad de asignar nombre de usuario, foto y tema de la aplicación.

# 13. Tecnologías utilizadas

## 13.1 Entorno de desarrollo

He utilizado Android Studio como IDE de mi aplicación principalmente porque es el que hemos utilizado durante el curso y me siento muy cómodo con él, además cuenta con una integración nativa con Kotlin y Jetpack Compose.

## 13.2 Lenguaje de desarrollo

He desarrollado la aplicación en Kotlin por un motivo similar al del punto anterior, es el que hemos estado utilizando durante el año y me he sentido cómodo desarrollando con ello.

## 13.3 Sistema operativo

La aplicación está dirigida únicamente a dispositivos Android.

## 13.4 Frameworks y bibliotecas

* + **Jetpack Compose**: para construir la UI de forma declarativa y reactiva.
  + **Room (SQLite)**: la he elegido por su comodidad para tener bases de datos de manera local.

## 13.5 Control de versiones

He trabajado con Github para el control de las versiones de mi aplicación, principalmente porque es la que siempre suelo usar y me resulta muy cómoda.

# 14. Desarrollo de solución propuesta

En este apartado, se detallan los elementos clave del desarrollo, los problemas y obstáculos hallados, además de las medidas adoptadas para superarlos y las especificidades de la resolución final:

## 14.1 Proceso de desarrollo

Dividí el proyecto en sprints semanales: cada semana elegía una pantalla o función (home, historial, formulario, calculadora, perfil), asegurando que no me quedaba estancado.

## 14.2 Problemas y dificultades

El mayor problema que encontré fue la sincronización de datos, me llevó mucho tiempo conseguir que al añadir una transacción se guardará bien y actualizara correctamente tanto el historial como los gráficos de estadísticas.

## 14.3 Decisiones tomadas

Separé la lógica de la gestión de datos y la interfaz para que cada parte tenga su responsabilidad clara y el código no se convierta en un enredo.

Para ello utilicé Room (SQLite) para que la base de datos local envíe automáticamente las actualizaciones.

## 14.4 Particularidades de la solución final

Al hacerlo de esta manera, me aseguro de que la aplicación es 100% offline, toda transacción, cambio en el perfil o cambio de tema de la aplicación, se guarda de manera local y siempre vuelve a aparecer igual incluso después de cerrar la aplicación.