



universidad  
de león



**Escuela de Ingenierías**  
**Industrial, Informática y**  
**GRADO EN INGENIERÍA INFORMÁTICA**

Trabajo de Fin de Grado

**APLICACIÓN MÓVIL PARA GASTOS PERSONALES**  
**MOBILE APPLICATION FOR PERSONAL EXPENSES**

Autor: Xosé Babío Sabín  
Tutor: José Alberto Benítez Andrades

(Febrero, 2024)

**UNIVERSIDAD DE LEÓN**  
**Escuela de Ingenierías Industrial, Informática y**  
**Aeroespacial**

**GRADO EN INGENIERÍA INFORMÁTICA**  
**Trabajo de Fin de Grado**

**ALUMNO:** Xosé Babío Sabín

**TUTOR:** José Alberto Benítez Andrades

**TÍTULO:** APLICACIÓN MÓVIL PARA GASTOS PERSONALES

**TITLE:** MOBILE APPLICATION FOR PERSONAL EXPENSES

**CONVOCATORIA:** Febrero, 2024

**RESUMEN:**

Cada día es más difícil tener una buena salud financiera siendo joven o recién entrado en el mundo laboral. El objetivo de este proyecto es generar una aplicación de gestión de gastos personales sencilla, con pocas funcionalidades, muy fácil de usar y clara para que cualquier persona pueda gestionar sus finanzas personales sin ningún tipo de apuro. Primero se hace un estudio del problema teniendo en cuenta la posible competencia. Después, se define detalladamente como se va a gestionar el proyecto a lo largo de su vida de desarrollo hasta tenerlo finalizado, así como gestionar sus recursos y el presupuesto. Acto seguido, se detalla la solución, desde las partes más visibles de la aplicación, hasta la parte que no se ve y que hace que funcione. A continuación, se detalla la normativa que ha de tenerse en cuenta para la realización del proyecto. Se muestra el producto final y cómo funciona. Se evalúa el producto final si ha cumplido los objetivos definidos al principio del proyecto. Y finalmente se sacan conclusiones y se opina del resultado.

**ABSTRACT:**

Nowadays it is more difficult to have a good financial health being young or just entering the working world. This project seeks to have a simple application, with few features, very easy to use and clear so that anyone can enjoy their personal finances without any trouble. First, a study of the problem is made taking into account the possible competition. Then, we define in detail how the project will be managed throughout its development life until its completion, as well as how to manage its resources and budget. Next, the solution is detailed, from the most visible parts of the application to the unseen part that makes it work. Next, the regulations that must be taken into account for the implementation of the project are detailed. The final product and how it works is shown. The final product is

evaluated to see if it has met the objectives defined at the beginning of the project. And finally, conclusions are drawn and an opinion is given on the result.

**Palabras clave:** Aplicación móvil, finanzas, Android, economía

**Firma del alumno:**

**VºBº Tutor/es:**

# ÍNDICE DE CONTENIDOS

## Contenido

ÍNDICE DE CONTENIDOS.....	4
ÍNDICE DE FIGURAS .....	7
ÍNDICE DE TABLAS.....	9
INTRODUCCIÓN .....	11
PLANTEAMIENTO DEL PROBLEMA .....	11
OBJETIVOS .....	11
METODOLOGÍA EMPLEADA .....	13
TECNOLOGÍAS Y HERRAMIENTAS .....	14
ESTRUCTURA DEL TRABAJO.....	14
1. ESTUDIO DEL PROBLEMA .....	16
1.1 CONTEXTO DEL PROBLEMA .....	16
1.2 EL ESTADO DEL ARTE.....	16
1.2.1 APLICACIONES SIMILARES .....	16
1.2.2 CRÍTICA AL ESTADO DEL ARTE.....	20
1.2.3 CONCLUSIÓN .....	21
2. GESTIÓN DEL PROYECTO SOFTWARE.....	22
2.1 ALCANCE DEL PROYECTO .....	22
2.1.1 DEFINICIÓN DEL PROYECTO.....	22
2.1.2 OBJETIVOS ESPECÍFICOS DEL PROYECTO .....	23
2.1.3 LÍMITES DEL PROYECTO .....	23
2.1.4 PRODUCTOS ENTREGABLES.....	24
2.1.5 CRITERIO DE ACEPTACIÓN .....	24
2.1.6 RESTRICCIONES.....	24
2.2 PLAN DE TRABAJO.....	25
2.2.1 IDENTIFICACIÓN DE FASES.....	25

---

2.2.2	IDENTIFICACIÓN DE TAREAS .....	25
2.2.3	PLANIFICACIÓN Y ESTIMACIÓN DE TAREAS .....	26
2.3	GESTIÓN DE RECURSOS.....	30
2.3.1	ESPECIFICACIÓN DE RECURSOS .....	30
2.3.2	ASIGNACIÓN DE RECURSOS.....	30
2.4	PRESUPUESTO .....	30
2.4.1	COSTES DE HARDWARE .....	31
2.4.2	COSTES DE SOFTWARE .....	31
2.4.3	COSTES TOTALES .....	31
3.	SOLUCIÓN .....	33
3.1	DESCRIPCIÓN DE LA SOLUCIÓN .....	33
3.1.1	DESCOMPOSICIÓN EN SUBSISTEMAS.....	34
3.2	ANÁLISIS DEL SISTEMA.....	37
3.2.1	DEFINICIÓN DE REQUISITOS .....	37
3.2.2	ESPECIFICACIÓN DE REQUISITOS.....	39
3.2.3	CASOS DE USO .....	45
3.2.4	ESPECIFICACIÓN DE LOS CASOS DE USO .....	46
3.3	DISEÑO DEL SISTEMA.....	50
3.3.1	CONTEXTO .....	50
3.3.2	ARQUITECTURA SOFTWARE .....	52
3.3.3	ARQUITECTURA DEL SISTEMA .....	53
3.3.4	DISEÑO DE PANTALLAS .....	54
3.3.5	DISEÑO DE LA BASE DE DATOS .....	56
3.4	IMPLEMENTACIÓN .....	57
3.4.1	HERRAMIENTAS .....	57
3.4.2	ASPECTOS A DESTACAR.....	58
3.5	PRUEBAS .....	62
3.5.1	PRUEBAS UNITARIAS.....	62

---

3.5.2	PRUEBAS DE CAJA BLANCA.....	62
3.5.3	PRUEBAS DE CAJA NEGRA .....	65
4.	NORMATIVA VIGENTE .....	67
5.	PRODUCTO FINAL .....	69
5.1	PANTALLA PRINCIPAL .....	69
5.2	GESTIÓN DE GASTOS .....	70
5.3	GESTIÓN DE INGRESOS .....	73
5.5	GRÁFICA.....	75
6.	EVALUACIÓN .....	77
7.	CONCLUSIONES.....	78
7.1	APORTACIONES REALIZADAS.....	78
7.2	PROBLEMAS ENCONTRADOS .....	78
7.3	POSIBLES MEJORAS.....	79
7.4	OPINIÓN PERSONAL.....	79
	BIBLIOGRAFÍA .....	<b>¡Error! Marcador no definido.</b>
	ANEXOS.....	85
	MANUAL DE USUARIO .....	85
	PANTALLA PRINCIPAL .....	85
	GESTIÓN DE GASTOS .....	86
	GESTIÓN DE INGRESOS.....	89
	GRÁFICOS .....	90

## ÍNDICE DE FIGURAS

Figura 1.1 Aplicación del Banco Santander .....	17
Figura 1.2 Aplicación MINT.....	17
Figura 1.3 Aplicación YNAB.....	18
Figura 1.4 Aplicación Personal Capital.....	19
Figura 1.5 Aplicación Fintonic.....	20
Figura 2.1: Diagrama de Gantt del proyecto.....	29
Figura 2.2: Gráfico de costes .....	32
Figura 3.1: Subsistemas de la aplicación .....	35
Figura 3.2: Casos de uso del usuario .....	45
Figura 3.3: Casos de uso del sistema .....	46
Figura 3.4: Modelo-Vista-Controlador.....	51
Figura 3.5: Model-View-ViewModel.....	52
Figura 3.6: Arquitectura del sistema.....	54
Figura 3.7: Estructura principal de las pantallas.....	54
Figura 3.8: Gama de colores 1 .....	55
Figura 3.9: Gama de colores 2 .....	55
Figura 3.10: Gama de colores 3 .....	56
Figura 3.11: Dao en este proyecto.....	59
Figura 3.12: El Adapter .....	60
Figura 3.13: El ViewHolder .....	61
Figura 3.14: Ejemplo de Corrutina.....	62
Figura 3.16: Función calc() a probar .....	63
Figura 3.15: Grafo del código anterior.....	63
Figura 5.1: Pantalla principal .....	69
Figura 5.2: Gestión de gastos.....	70
Figura 5.3: Formulario del gasto.....	71
Figura 5.4: Elección en calendario.....	72
Figura 5.5: Actualizar o eliminar gasto .....	72
Figura 5.6: Gestión de ingresos .....	73
Figura 5.7: Editar o eliminar ingreso.....	74
Figura 5.8: Añadir ingreso.....	74
Figura 5.9: Gráfica.....	75
Figura 5.10: Gráfica ampliada .....	76





## ÍNDICE DE TABLAS

Tabla 2.1: Planificación del Inicio del Proyecto.....	26
Tabla 2.2: Planificación de la Iteración 1 .....	26
Tabla 2.3: Planificación de la Iteración 2 .....	27
Tabla 2.4: Planificación de la Iteración 3 .....	27
Tabla 2.5: Planificación de la Iteración 4 .....	27
Tabla 2.6: Planificación de la Iteración 5 .....	28
Tabla 2.7: Planificación de la Revisión y Puesta en marcha.....	28
Tabla 2.8: Especificación de recursos .....	30
Tabla 2.9: Asignación de recursos (por horas).....	30
Tabla 2.10: Costes de hardware.....	31
Tabla 2.11: Costes de software .....	31
Tabla 2.12: Costes totales .....	32
Tabla 3.1: Diseño intuitivo y simple .....	39
Tabla 3.2: Interfaz con XML y View Binding.....	39
Tabla 3.3: Dispositivos Mviles con Android .....	39
Tabla 3.4: Arquitectura MVVM .....	39
Tabla 3.5: Backend en Kotlin.....	40
Tabla 3.6: Base de datos SQLite con Room.....	40
Tabla 3.7: Corrutinas.....	40
Tabla 3.8: Controlador de versiones con Git .....	40
Tabla 3.9: Gráficas con librería MPAndroidChart.....	41
Tabla 3.10: Consulta gastos del mes actual .....	41
Tabla 3.11: Consulta gasto del año actual .....	41
Tabla 3.12: Consulta de recomendación del algoritmo.....	41
Tabla 3.13: Consulta de gastos desglosados del mes elegido .....	42
Tabla 3.14: Cambio de mes de gastos .....	42
Tabla 3.15: Crear gasto .....	42
Tabla 3.16: Modificar gasto .....	42
Tabla 3.17: Borrar gasto .....	43
Tabla 3.18: Creación automática de gasto periódico .....	43
Tabla 3.19: Consulta de ingresos desglosados del mes elegido .....	43
Tabla 3.20: Cambio de mes de ingresos .....	43
Tabla 3.21: Crear ingreso .....	44

---

Tabla 3.22: Modificar ingreso .....	44
Tabla 3.23: Borrar ingreso .....	44
Tabla 3.24: Creación automática de ingreso periódico .....	44
Tabla 3.25: Consulta de gráficas de gasto y ahorro.....	45
Tabla 3.26: Ampliar, reducir y moverse entre la gráfica .....	45
Tabla 3.27: Caso de uso - Crear gasto.....	46
Tabla 3.28: Caso de uso - Actualizar gasto.....	47
Tabla 3.29: Caso de uso - Borrar gasto .....	47
Tabla 3.30: Caso de uso – Crear ingreso.....	47
Tabla 3.31: Caso de uso – Actualizar ingreso.....	48
Tabla 3.32: Caso de uso – Borrar ingreso .....	48
Tabla 3.33: Caso de uso – Consultar gastos.....	48
Tabla 3.34: Caso de uso – Consultar ingresos.....	49
Tabla 3.35: Caso de uso – Consultar gráficas.....	49
Tabla 3.36: Caso de uso – Consultar recomendación.....	49
Tabla 3.37: Caso de uso – Generación automática de gasto periódico .....	49
Tabla 3.38: Caso de uso – Generación automática de ingreso periódico.....	50
Tabla 3.39: Caminos independientes del código .....	64
Tabla 3.40: Pruebas de caja negra .....	65

## INTRODUCCIÓN

Este proyecto se centra en crear una aplicación móvil que agilice la gestión financiera personal para aquellos con ingresos y gastos diversos. El proceso implica identificar el problema, analizar herramientas existentes, definir requisitos y riesgos, desarrollar la aplicación y realizar pruebas para garantizar su calidad.

En este capítulo, nos centraremos en los objetivos, alcance, propósito del proyecto y la organización del trabajo.

## PLANTEAMIENTO DEL PROBLEMA

El problema de finanzas personales que enfrenta mucha gente hoy en día es la falta de conciencia y control sobre sus gastos e ingresos. Con la facilidad de acceso a créditos, tarjetas y servicios financieros, es común caer en patrones de gasto impulsivo.

Es importante también la mejoría en la tecnología de pagos, se puede pagar con tarjeta, con el teléfono, incluso con algunos relojes inteligentes, lo que hace que los pagos sean más fáciles y rápidos, por lo tanto, somos menos conscientes en ellos. Además, ahora se gasta mucho dinero en internet. Tanto las aplicaciones de comida a domicilio, como las compras en aplicaciones de ropa o las suscripciones a servicios en línea (multimedia, almacenamiento en la nube, etc...) constituyen un gasto importante en nuestro día a día.

La inestabilidad económica y la incertidumbre laboral también añaden presión a las finanzas personales. Las emergencias inesperadas pueden desequilibrar rápidamente el presupuesto de alguien.

Las herramientas que intentan ayudar a que una persona categorice y controle sus gastos e ingresos suelen ir acompañadas de otras funcionalidades más complicadas orientadas al mundo de las inversiones y demás. Utilizar aplicaciones que constantemente te recomiende invertir y que dedica gran parte de la interfaz a este ámbito hace que una persona que simplemente quiera controlar sus gastos para tener un ahorro coherente deje de utilizarla.

## OBJETIVOS

El proyecto se enfoca en construir una aplicación móvil con un propósito claro: empoderar a las personas para que gestionen sus finanzas personales de manera efectiva y eviten riesgos económicos. La simplicidad es clave, permitiendo a los usuarios registrar fácilmente sus ingresos y gastos para obtener una visión clara de su situación financiera.

La aplicación ha de permitir de manera clara separar la parte de los gastos con la de los ingresos. En esta primera parte se podrán ver una lista de gastos por meses, en donde se podrá ir navegando mes a mes. Debe permitir añadir gastos de una manera cómoda mediante un formulario y poder editarlos y eliminarlos. En este formulario es donde se especificará que tipo de gasto es, si necesario o no necesario.

La parte de los ingresos funcionará de manera similar a los gastos, pudiendo así listar, crear, modificar y eliminar los ingresos.

Además, en estos ingresos y gastos se han de poder programar una periodicidad, es decir, sin tener que añadirlos de manera manual que se cree el mismo gasto o ingreso pasado un cierto tiempo.

También es necesario añadir alguna pantalla de resumen general, en dónde poder leer las recomendaciones que el algoritmo haga. En este caso será la pantalla principal, en donde se podrá consultar los gastos del mes actual, año actual y la recomendación del algoritmo.

Una característica destacada de la aplicación es la representación visual a través de una gráfica de barras. Esta gráfica ofrece una perspectiva instantánea y comprensible de la distribución de los recursos financieros del usuario. Al enfrentar los gastos importantes con los no importantes y resaltar la parte destinada al ahorro, la aplicación proporciona una herramienta intuitiva para evaluar la salud económica de manera rápida.

Imagina visualizar cómo se distribuyen tus ingresos: una barra para los gastos esenciales, otra para aquellos menos necesarios y, por último, una destinada al ahorro. Esta representación visual permite a los usuarios identificar áreas de mejora, ajustar sus hábitos de gasto y fortalecer su posición financiera.

De forma simple los objetivos del proyecto se pueden definir de la siguiente manera:

- Estudio y análisis comparativo de las competencias de la solución.
- Desarrollar un producto interactivo y rápido para su fácil utilización.
- Desarrollar el algoritmo de recomendación para las finanzas personales.
- Maquetación de gráficas de gastos y ahorro.

Más adelante, nos centraremos en el análisis detallado de los objetivos mientras exploramos los requisitos específicos de la aplicación.

## METODOLOGÍA EMPLEADA

En esta iniciativa, se ha empleado una metodología basada en el desarrollo incremental e iterativo, el cual aborda las deficiencias a un modelo convencional de tipo cascada. Este método ha permitido adaptarse con mayor flexibilidad a los cambios, posibilitando una evolución constante y progresiva del proyecto, en contraposición a las limitaciones rígidas del enfoque en cascada [1].

En la etapa inicial, se lleva a cabo una exhaustiva evaluación del problema planteado y se identifican los diversos requisitos del sistema que se pretenden abordar antes de iniciar el proceso de desarrollo del producto. Tras esta fase analítica, se inicia el proceso de desarrollo fundamentado en la estrategia de incrementos, donde de manera iterativa se van consolidando y perfeccionando las diferentes secciones del producto hasta alcanzar su forma final y completa. Este enfoque iterativo permite una construcción gradual y sistemática, asegurando que cada parte del producto sea refinada antes de integrarse en la versión final, garantizando así un producto robusto y bien elaborado.

Al iniciar cada iteración, se lleva a cabo una concisa especificación de los requisitos junto con un detallado análisis de los objetivos que se pretenden alcanzar con ese incremento particular. Una vez completados los incrementos, los cuales pueden variar en duración temporal, se procede a realizar un análisis exhaustivo de los objetivos logrados. Posteriormente, se lleva a cabo una fase de pruebas en un entorno de virtualización del sistema al que va dirigido el producto. Esta etapa es fundamental para evaluar el desempeño y la funcionalidad del producto en un ambiente simulado, permitiendo identificar posibles fallos o áreas de mejora antes de su implementación definitiva.

Dentro de cada fase de incremento del ciclo de vida, se despliegan múltiples iteraciones que contribuyen al progreso del desarrollo incremental. Estas iteraciones, a su vez, se componen de varias etapas o actividades clave, entre las cuales se incluyen:

- Inicio: Se establece claramente el alcance del próximo incremento, definiendo con precisión los objetivos específicos que se buscan alcanzar durante esa iteración en particular.
- Elaboración: Se lleva a cabo un refinamiento de la visión general del proyecto, profundizando en los detalles y preparando la arquitectura del sistema para iniciar la implementación.

- **Construcción** Se lleva a cabo la implementación práctica de todos los casos de uso definidos, así como la ejecución de pruebas generales para validar el funcionamiento del sistema.
- **Transición:** Se lleva a cabo la validación final y el cierre del sistema actual para prepararlo para el inicio del siguiente incremento del proyecto.

La duración variable de los incrementos en cada fase del proyecto ha estado influenciada por la diferencia en la carga de trabajo y la complejidad de las tareas. Algunas fases demandaron más tiempo para resolver desafíos técnicos, mientras que otras fueron más ágiles en la implementación de funcionalidades específicas. Adaptar la duración de cada incremento según las necesidades de cada fase ha permitido una gestión más eficiente de los recursos y el tiempo disponibles para alcanzar los objetivos establecidos en cada iteración.

## TECNOLOGÍAS Y HERRAMIENTAS

Para la elaboración del proyecto se han utilizado distintas tecnologías en las que se basa el producto y herramientas que facilitaron el desarrollo de este:

- **Backend:** El lenguaje principal del proyecto es Kotlin, con librerías pensadas para el desarrollo de Android nativo. Como herramienta de desarrollo se han utilizado Android Studio en su versión Giraffe 2022 y Visual Studio Code en su versión 1.84.2 [2][3][4].
- **Frontend:** Se ha utilizado el lenguaje XML con librerías de desarrollo de Android. Como tecnología que conecta el Backend y Frontend se ha utilizado View Binding [5].
- **Base de datos:** Como tecnología de persistencia de datos se ha utilizado SQLite, con la librería Room de Kotlin [6].
- **Controlador de versiones:** Se ha utilizado Git como controlador de versiones y como servidor para alojar el código GitHub [7].
- **Documentación:** Para documentar se ha empleado Microsoft Word y Microsoft Excel.
- **Dispositivos:** Tanto para el desarrollo del producto como de la documentación se ha utilizado un ordenador personal con un procesador I7-6700 y 16GB de RAM y para pruebas del producto una virtualización de un Google Pixel 7 con un sistema con la API 34, Android 13.0.

## ESTRUCTURA DEL TRABAJO

Este documento que refleja la memoria del proyecto se divide en cuatro grandes áreas, en dónde se cubren el proceso de vida del trabajo:

1. **Estudio del problema:** Primero se hace un análisis previo al desarrollo del producto, en dónde se contextualiza el problema a resolver, con las distintas soluciones que ofrece ahora mismo el mercado.
2. **Gestión del proyecto:** Tras analizar el problema, se realiza un estudio de las tecnologías actuales y se plantea una gestión del proyecto. En este plan se define el alcance del proyecto, se organiza el plan de trabajo y se intenta buscar un presupuesto tanto real como viable para llevar a cabo la solución planteada. Además de eso es necesario realizar un análisis de riesgos.
3. **Solución:** Este es el apartado clave del documento, en dónde se define detalladamente la solución. Para definir la solución es necesario exponer todos y cada uno de los requisitos del producto, como todos los casos de uso. A continuación, se comenta y expone el diseño y arquitectura del sistema. Como tercer punto, se expone todo lo relacionado al proceso de desarrollo y problemas que se han encontrado a medida que el producto iba tomando forma. Y finalmente, las pruebas, en dónde se explica de que tipo son y cómo se han llevado a cabo.
4. **Evaluación:** Ya con el producto terminado se evalúa la validez de la solución alcanzada, comparándolo con el problema planteado al principio del documento.

Además, al final se exponen distintos temas como las posibles vías de expansión tras la finalización del proyecto, problemas encontrados y conclusiones finales.

# 1. ESTUDIO DEL PROBLEMA

En este apartado se expone el contexto en el desarrolla el proyecto. Es necesario aclarar temas acerca de las tecnologías y aplicaciones móviles y finanzas personales.

## 1.1 CONTEXTO DEL PROBLEMA

El problema de finanzas personales que enfrenta mucha gente hoy en día es la falta de conciencia y control sobre sus gastos e ingresos. El estilo de vida moderno también ayuda a no ser consciente de los gastos por la cantidad de pagos que hacemos con tarjeta, además de los que hacemos en efectivo. Las subscripciones mensuales a servicios de cine y series bajo demanda, servicios de música en línea y almacenamiento en la nube son pagos que muchas veces no tenemos en cuenta a la hora llevar las cuentas personales.

Además del problema de la poca conciencia acerca de nuestros gastos también se suma la inestabilidad económica y la incertidumbre laboral. Las emergencias inesperadas pueden desequilibrar rápidamente el presupuesto de alguien, generando estrés financiero.

La falta de educación financiera también contribuye a este problema. Muchas personas no cuentan con los conocimientos necesarios para administrar eficazmente su dinero, planificar para el futuro o entender los riesgos y beneficios de diversas opciones financieras.

## 1.2 EL ESTADO DEL ARTE

En el mercado de aplicaciones móviles existen muchas opciones de aplicaciones que ayudan al usuario a controlar sus propios gastos e ingresos, así que en este punto vamos a analizar brevemente varias alternativas, focalizando en las ventajas y desventajas.

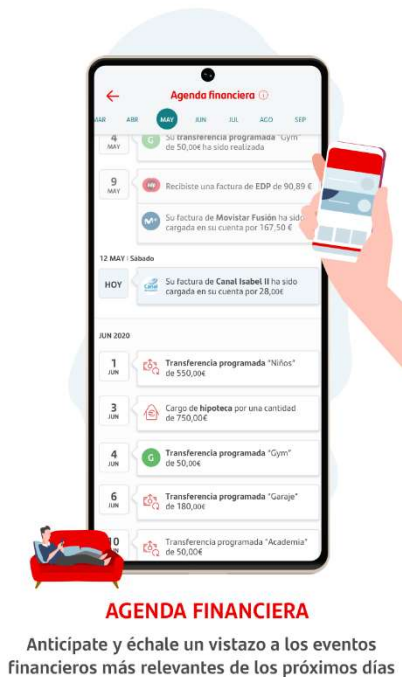
### 1.2.1 APLICACIONES SIMILARES

#### 1.2.1.1 APLICACIONES DEL BANCO

En primer lugar, tenemos la aplicación de nuestro banco. Es cierto que no todas las aplicaciones de banco son iguales así que para resumir un poco vamos a analizar los puntos comunes que tienen estas aplicaciones. La mayor ventaja que tienen estas aplicaciones es que los ingresos y los pagos con tarjeta se registran automáticamente. Además, según el concepto del pago con tarjeta, la aplicación detecta de que categoría es el pago (restaurante, gasolina, transporte...). Las mayores desventajas que tienen estas aplicaciones son que no se puede registrar de forma manual ingresos o pagos que se hacen en efectivo, lo que se registra son los movimientos de una tarjeta, algunas veces el pago no se registra bien en su categoría correcta y el banco no deja de ser un negocio con el dinero del usuario, es probable que te recomiende



planes de pensiones o inversiones sin ninguna garantía de poner en riesgo la situación financiera del usuario.



*Figura 1.1 Aplicación del Banco Santander*

#### 1.2.1.2 MINT: BUDGET & TRACK BILLS

Esta aplicación ofrece seguimiento de gastos, presupuestos personalizados, monitoreo de crédito y recordatorios de facturas. Como ventajas podemos destacar el seguimiento desde un solo lugar, ya que se puede conectar con diferentes bancos, la creación de presupuestos personalizados y el recordatorio de facturas. Como desventajas cabe decir que la sincronización con las cuentas de banco no es muy fina, lo que resulta en retrasos o errores en la actualización de datos. Otra desventaja son los anuncios y recomendaciones de productos financieros, que



*Figura 1.2 Aplicación MINT*

pueden resultar intrusivos o no deseados para algunos usuarios. Y como desventaja final, tiene tantas opciones que el usuario promedio no utiliza que puede resultar difícil de utilizar.

#### 1.2.1.3 YNAB (YOU NEED A BUDGET)

YNAB se enfoca en la creación de presupuestos, ayudando a los usuarios a asignar fondos a categorías específicas y seguir su progreso. Sus ventajas principales son el enfoque en presupuestos, que permite asignar fondos a diferentes categorías, y la educación financiera, proporciona recursos y herramientas educativas para ayudar a los usuarios a comprender mejor la gestión financiera y mejorar sus hábitos de gasto. Como desventajas destacaremos la curva de aprendizaje, para algunos usuarios, el enfoque basado en reglas y la metodología específica de asignación de fondos pueden requerir tiempo para acostumbrarse y comprender completamente y el costo, aunque ofrece una prueba gratuita, YNAB es una aplicación de pago. Esto puede ser una desventaja para aquellos que buscan soluciones gratuitas para la gestión financiera personal.

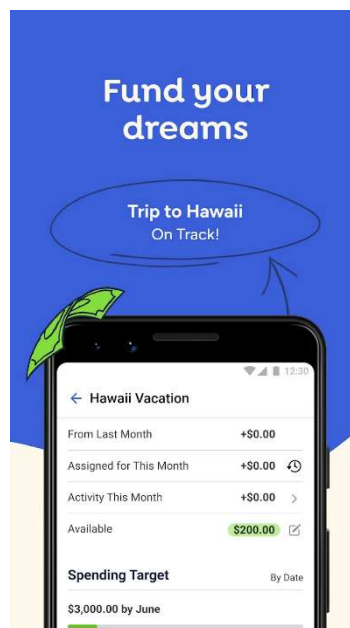
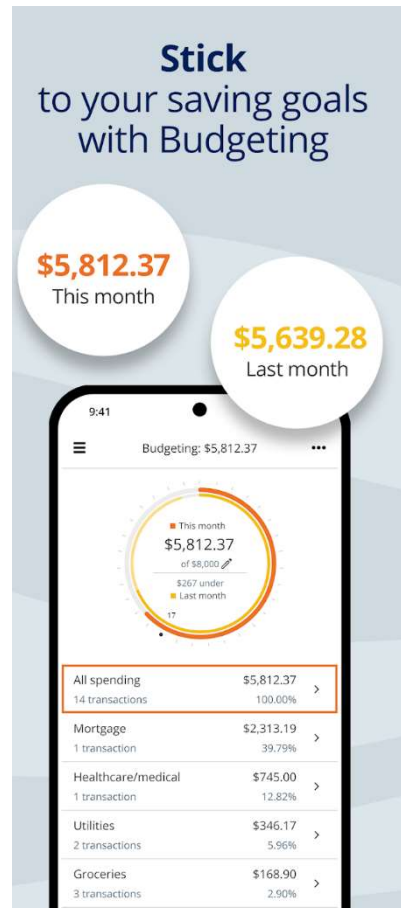


Figura 1.3 Aplicación YNAB

#### 1.2.1.4 PERSONAL CAPITAL

Es una aplicación que combina seguimiento de presupuestos con herramientas de inversión. Proporciona información detallada sobre inversiones, así como seguimiento de activos y pasivos. Tiene un seguimiento integral, permite rastrear y consolidar todas las cuentas financieras, incluyendo cuentas bancarias, inversiones, hipotecas y préstamos. Ofrece un asesoramiento financiero personalizado a través de consultas con asesores financieros certificados. Su sistema de seguridad es de doble factor, lo que asegura a sus usuarios su

información más delicada. En este caso las desventajas son todas con relación al tipo de aplicación, es una aplicación enfocada a las inversiones, por lo que el usuario promedio, que no tiene inversiones y solo quiere controlar sus gastos se ve perjudicado. Además de eso, algunos usuarios consideran que las comunicaciones frecuentes por correo electrónico o los anuncios promocionales de los servicios de gestión de activos pueden resultar intrusivos.



*Figura 1.4 Aplicación Personal Capital*

#### 1.2.1.5 FINTONIC

Fintonic es una aplicación financiera que proporciona servicios de gestión de finanzas personales. A través de esta aplicación, los usuarios pueden realizar un seguimiento de sus gastos, ingresos, y otras transacciones financieras de manera centralizada. Proporciona una visión integral de las finanzas personales al permitir a los usuarios conectar varias cuentas y tarjetas de diferentes entidades bancarias en una única plataforma. La aplicación utiliza algoritmos para categorizar automáticamente los gastos e ingresos, facilitando el seguimiento y análisis de las transacciones financieras. Los usuarios pueden establecer presupuestos personalizados en diferentes categorías, lo que ayuda a controlar y limitar los gastos en áreas

específicas. La eficacia de Fintonic está estrechamente vinculada a su capacidad para establecer y mantener conexiones con las cuentas bancarias de los usuarios, permitiendo la sincronización de datos. Problemas técnicos en estas conexiones pueden incidir en la precisión de la información proporcionada por la aplicación. La aplicación puede no ser compatible con la totalidad de las entidades bancarias, lo que impone restricciones en la accesibilidad para algunos usuarios según la institución bancaria a la que estén afiliados. Aunque Fintonic incorpora medidas de seguridad, algunos usuarios pueden sentir inquietud respecto a la privacidad de sus datos financieros al otorgar acceso a sus cuentas bancarias.

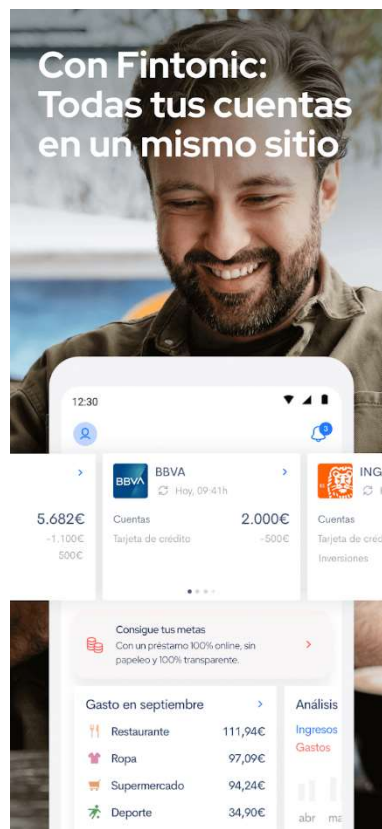


Figura 1.5 Aplicación Fintonic

### 1.2.2 CRÍTICA AL ESTADO DEL ARTE

Es cierto que las inversiones son una parte importante para tener una gran salud financiera y es importante gestionarlas, las inversiones empiezan a cobrar importancia cuando ya tienes una base financiera sólida e incluso holgada. No es muy frecuente encontrar un grupo de usuarios con esa base formada como para poder sacarle partido a esta funcionalidad. Las aplicaciones mencionadas están fuertemente basadas en gestionar tus activos por lo que no creo que sean adecuadas para la mayoría de los usuarios.

Además de eso, el exceso de funcionalidad como poder crear muchas categorías y presupuestos, no hace que la aplicación sea fácil e intuitiva. Lo lógico es separar esas categorías en dos, necesario e innecesario, y después de calcular el porcentaje de tus gastos ya se puede hacer un análisis de subcategorías dentro de ellas.

En cuanto a las aplicaciones del banco, la crítica principal es que no se pueden registrar gastos manualmente, o editar los gastos automáticos que la propia aplicación registra. Este hecho hace que muchos gastos no se categoricen bien ya que lo hacen a través del concepto y el vendedor. Además de eso no se pueden registrar cómo utilizas el dinero en efectivo.

### 1.2.3 CONCLUSIÓN

Después del análisis de las diferentes aplicaciones y la crítica al estado del arte vemos que no hay una alternativa para un usuario joven, con pocos ingresos que quiere formar una base financiera sólida que no está acostumbrado a gestionarse su propio dinero, quizá con varios trabajos, cobrando en efectivo y no la misma cantidad siempre. Además, poder registrar de forma periódica gastos e ingresos, pensado para esas subscripciones mensuales y anuales que no tenemos en cuenta al analizar nuestros gastos.

Es por eso por lo que el objetivo de este proyecto es hacer una aplicación fácil, intuitiva, que ayude al usuario a forjar una base financiera con la que poder vivir tranquilo y tener un porcentaje de ahorro suficiente.

## 2. GESTIÓN DEL PROYECTO SOFTWARE

En este capítulo se expone la información más relevante de la gestión del proyecto mediante una simulación empresarial del proyecto hasta conseguir el objetivo final, ya que el trabajo ha sido realizado de forma personal en el domicilio personal.

Se definirá el alcance del proyecto, el plan de trabajo, cómo se gestionan los recursos y un presupuesto estimado.

### 2.1 ALCANCE DEL PROYECTO

En esta sección, nos centraremos en definir con precisión los elementos clave que determinarán el alcance del proyecto. Estos aspectos incluyen la descripción detallada del proyecto, sus objetivos principales, los límites que lo marcan, los productos entregables, los estándares que se deben cumplir para su aceptación y las restricciones que pueden influir en su desarrollo.

#### 2.1.1 DEFINICIÓN DEL PROYECTO

El proyecto es una aplicación móvil desarrollada para el entorno Android que consiste en la gestión de gastos e ingresos con un algoritmo de recomendación para seguir el principio de finanzas 50-30-20 [8]. Este principio consiste en gastar el 50% de tus ingresos en gastos imprescindibles como facturas, comida, educación y la casa, el 30% en gastos innecesarios, como el ocio, ropa, comidas en restaurantes y deporte, y el 20% en ahorro. Existen muchas formas de administrar las finanzas personales, pero esta es la más básica.

Este proyecto se puede definir como un MVP (Minimum Viable Product), que es la versión más básica de un producto que satisface las necesidades básicas del usuario [9]. Es una versión funcional con características esenciales para lanzar al mercado y obtener retroalimentación temprana de los usuarios. Es una estrategia muy efectiva para establecer los cimientos de un producto que se puede ampliar justo con lo que los usuarios demandan.

La interfaz está desarrollada en español como único idioma. Es una interfaz intuitiva y fácil, en donde en pocos clics se puede llegar a toda la funcionalidad de la aplicación. En la versión final se puede consultar las recomendaciones de la aplicación, consultar un breve resumen, ver y añadir gastos por mes, ver y añadir ingresos por mes, y finalmente consultar una gráfica anual de gastos y ahorro. Además de eso podemos programar gastos e ingresos, de forma mensual, semanal y anual.

### 2.1.2 OBJETIVOS ESPECÍFICOS DEL PROYECTO

En términos generales, para marcar el final del proyecto, se deben alcanzar ciertos logros fundamentales. Estos objetivos específicos son:

- Que la aplicación te recomiende disminuir gastos en cierta categoría según el principio 50-30-20.
- Poder añadir gastos con descripción, importe, regularidad, categoría y fecha.
- La posibilidad de editar gastos.
- Consultar los gastos ordenados de cada mes.
- Poder añadir ingresos con descripción, importe, regularidad y fecha.
- La posibilidad de editar ingresos.
- Que la aplicación calcule tus porcentajes de gasto frente a tus ingresos siguiendo el principio 50-30-20.
- El cálculo de la regularidad de gastos e ingresos, la propia aplicación añade automáticamente gastos e ingresos de forma semanal, mensual y anual.
- El borrado de gastos.
- El borrado de ingresos.
- Consultar el gasto desglosado en categorías y el ahorro durante el año de forma gráfica.
- Desarrollar una interfaz fácil e intuitiva para el usuario promedio.
- Asegurar el cumplimiento de la ley de protección de datos.

### 2.1.3 LÍMITES DEL PROYECTO

A lo largo del desarrollo del proyecto pueden surgir problemas e imprevistos por lo que se han definido unos límites para que no se altere el planteamiento inicial de dicho proyecto. Definiendo estos límites se puede prevenir un aumento de costes, tanto monetarios como de tiempo, no esperados en el desarrollo de este. En este caso los límites se han definido en esta lista:

- La aplicación está desarrollada para teléfonos que utilicen un sistema operativo basado en Android.
- El proyecto es desarrollado mediante la arquitectura MVVM definida más adelante en la memoria [10].
- La instalación de la aplicación se hará posible mediante la instalación de un archivo .apk obtenido únicamente mediante la aplicación y servicio de Google Play.

#### 2.1.4 PRODUCTOS ENTREGABLES

Al no tener un cliente específico y el cliente sea el conjunto de usuarios de teléfonos con un sistema operativo Android, no es necesario hacer un conjunto de entregas a lo largo del desarrollo del proyecto.

El único producto entregable es la versión de la aplicación 1.0.0. Este entregable se despliega en el servicio distribuidor de aplicaciones para Android más famoso, Google Play. Para poder hacerlo es necesario formar parte de los desarrolladores de Google Play, para el cual hay que pagar una licencia de la que hablaremos más adelante en la memoria.

Suponiendo tener un cliente el cual nos demande una serie de entregables a lo largo del desarrollo seguiríamos la siguiente estrategia. Al desarrollar el producto mediante un proceso iterativo nos pone muy fácil la elección de entregables. El proceso iterativo cuenta con 5 iteraciones en dónde se va incrementando la dimensión del producto, hablaremos de estas iteraciones más adelante. Por lo tanto, en cada iteración se le entregará al cliente el producto para que pueda comprobarlo y valorarlo. Además, el cliente recibirá un sexto entregable en dónde obtendrá el producto completo, con los errores corregidos y un manual de usuario.

#### 2.1.5 CRITERIO DE ACEPTACIÓN

Para que el proyecto sea aceptado como válido tiene que cumplir los siguientes aspectos mostrados en esta lista:

- Comprobar el cumplimiento de los distintos Requisitos Funcionales y No Funcionales descritos en el documento de ERS (Especificación de Requisitos de Software).
- Comprobar el cumplimiento de los distintos casos de uso descritos en el Plan de Proyecto y en el Documento de Diseño.
- Las pruebas de caja negra y caja blanca han de ser exitosas obteniendo los resultados esperados en cuanto al funcionamiento [11].
- Todos los objetivos expuestos en el apartado de objetivos de este documento han de estar presentes en el proyecto.

#### 2.1.6 RESTRICCIONES

En esta sección es necesario separar el presupuesto del tiempo. En cuanto al presupuesto, se permite una desviación típica del 5%.

Hablando sobre la estimación del tiempo de desarrollo, se permite una desviación aproximada de 1 mes.



## 2.2 PLAN DE TRABAJO

En esta sección del documento se realizará una identificación de las fases y tareas necesarias para llevar a cabo el proyecto de una manera iterativa y planificada.

### 2.2.1 IDENTIFICACIÓN DE FASES

Para identificar las fases tendremos que realizar un análisis del proyecto de alto nivel. En esta lista se definen las principales fases de alto nivel, que luego detallaremos:

- **Inicio del proyecto:** se definen los distintos casos de uso y requisitos para el desarrollo completo de la aplicación. Además, se prepara el entorno de desarrollo y se realiza un aprendizaje completo de las tecnologías a utilizar.
- **Proceso iterativo:** en esta fase se desarrolla el código del proyecto en distintas iteraciones. Primero se detalla que requisitos se van a implementar en la iteración, y cómo se van a hacer. Al terminar, se realizan las pruebas necesarias para garantizar la calidad de lo desarrollado. Esta fase es la que da forma al resultado final del producto.
- **Revisión y puesta en marcha:** se revisa al completo la aplicación mediante pruebas y se corrigen. En el momento en el que el proyecto cumple con los requisitos necesarios para considerarse finalizado, se lleva a producción y se publica.

### 2.2.2 IDENTIFICACIÓN DE TAREAS

Cada fase de alto nivel la dividiremos en tareas más pequeñas para detallar más el plan de trabajo.

#### 2.2.2.1. INICIO DEL PROYECTO

En esta fase del proyecto se realizan las siguientes tareas: estudio del problema, análisis de programas existentes en Google Play, especificación de requisitos, preparación del entorno de desarrollo y aprendizaje de tecnologías.

#### 2.2.2.2 PROCESO ITERATIVO

Distinguimos el proceso iterativo en 5 iteraciones:

- **Iteración 1:** especificación de requisitos y de los casos de uso para la iteración, diseño e implementación de la base de datos, conexión del backend con la base de datos, definición de la arquitectura, navegabilidad entre pantallas y pruebas.
- **Iteración 2:** especificación de requisitos y de los casos de uso para la iteración, implementación de la pantalla de **gestión de gastos**, consultar, crear, modificar y eliminarlos, y pruebas.

- **Iteración 3:** especificación de requisitos y de los casos de uso para la iteración, implementación de la pantalla de **gestión de ingresos**, consultar, crear, modificar y eliminarlos, y pruebas.
- **Iteración 4:** especificación de requisitos y de los casos de uso para la iteración, implementación de la **pantalla principal**, así como las funciones de regularidad y de cálculo de porcentajes de gasto y pruebas.
- **Iteración 5:** especificación de requisitos y de los casos de uso para la iteración, implementación de la pantalla de **gráficas** y pruebas.

### 2.2.2.3 REVISIÓN Y PUESTA EN MARCHA

Terminado el proceso iterativo llegamos a la fase de revisión, en donde se prueba la aplicación de principio a fin, corrigiendo los fallos encontrados y asegurando todos los requisitos y los casos de uso. Una vez asegurada una calidad de producto se procede a desplegar la aplicación en Google Play y publicándola al mercado.

### 2.2.3 PLANIFICACIÓN Y ESTIMACIÓN DE TAREAS

La planificación se ha realizado siguiendo las fases por incrementos detalladas en los puntos anteriores. Tenemos una tabla y un diagrama de Gantt para estimar el calendario de desarrollo del proyecto [12].

El proyecto comienza el día 03/07/2023 y acaba el 23/11/2023, lo que son 144 días duración. En la siguiente tabla podemos observar la planificación de la primera fase del proyecto con sus tareas desglosadas, con su fecha de inicio, fecha de fin, duración en días y día de comienzo.

*Tabla 2.1: Planificación del Inicio del Proyecto*

TAREAS	FECHA DE INICIO	FECHA DE FIN	DURACIÓN EN DÍAS	DÍA DE COMIENZO
<b>INICIO DEL PROYECTO</b>	03/07/2023	14/07/2023	12	0
CASOS DE USO	03/07/2023	04/07/2023	2	0
REQUISITOS	05/07/2023	07/07/2023	3	2
ENTORNO DE DESARROLLO	08/07/2023	10/07/2023	3	5
APRENDIZAJE DE TECNOLOGÍAS	11/07/2023	14/07/2023	4	8

Continuamos viendo en la siguiente tabla la iteración 1, en dónde vemos que dura 23 días y comienza el 17/07/2023.

*Tabla 2.2: Planificación de la Iteración 1*

TAREAS	FECHA DE INICIO	FECHA DE FIN	DURACIÓN EN DÍAS	DÍA DE COMIENZO
<b>ITERACIÓN 1</b>	15/07/2023	06/08/2023	23	12

REQUISITOS Y CASOS DE USO	15/07/2023	16/07/2023	2	12
BASE DE DATOS	17/07/2023	23/07/2023	7	14
ARQUITECTURA	24/07/2023	30/07/2023	7	21
NAVEGABILIDAD	31/07/2023	03/08/2023	4	28
PRUEBAS	04/08/2023	06/08/2023	3	32

A continuación, en la siguiente tabla observar la planificación de la segunda iteración del proyecto con sus tareas desglosadas, con su fecha de inicio, fecha de fin, duración en días y día de comienzo.

*Tabla 2.3: Planificación de la Iteración 2*

TAREAS	FECHA DE INICIO	FECHA DE FIN	DURACIÓN EN DÍAS	DÍA DE COMIENZO
<b>ITERACIÓN 2</b>	07/08/2023	25/08/2023	19	35
REQUISITOS Y CASOS DE USO	07/08/2023	08/08/2023	2	35
GESTIÓN DE GASTOS	09/08/2023	23/08/2023	15	37
PRUEBAS	24/08/2023	25/08/2023	2	52

Inmediatamente vemos en la siguiente tabla la planificación de la iteración 3, muy parecida a la iteración 2 ya que sus tareas son análogas.

*Tabla 2.4: Planificación de la Iteración 3*

TAREAS	FECHA DE INICIO	FECHA DE FIN	DURACIÓN EN DÍAS	DÍA DE COMIENZO
<b>ITERACIÓN 3</b>	26/08/2023	13/09/2023	19	54
REQUISITOS Y CASOS DE USO	26/08/2023	27/08/2023	2	54
GESTIÓN DE INGRESOS	28/08/2023	11/09/2023	15	56
PRUEBAS	12/09/2023	13/09/2023	2	71

En la iteración 4 la duración es mucho mayor ya que tiene más tareas y el trabajo empleado debe ser mayor. Podemos consultar esa información en la siguiente tabla.

*Tabla 2.5: Planificación de la Iteración 4*

TAREAS	FECHA DE INICIO	FECHA DE FIN	DURACIÓN EN DÍAS	DÍA DE COMIENZO
<b>ITERACIÓN 4</b>	14/09/2023	28/10/2023	45	73
REQUISITOS Y CASOS DE USO	14/09/2023	15/09/2023	2	73
PANTALLA PRINCIPAL	16/09/2023	02/10/2023	17	75
FUNCIONES REGULARIDAD	03/10/2023	15/10/2023	13	92
CÁLCULO DE PORCENTAJES	16/10/2023	26/10/2023	11	105
PRUEBAS	27/10/2023	28/10/2023	2	116

Llegamos a la última fase del proceso iterativo, la iteración 5. En esta iteración nos centramos en la pantalla de la consulta de gráficas, y se divide en 3 tareas, la definición de

requisitos y casos de uso, el desarrollo de las gráficas y finalmente las pruebas. Observemos la siguiente tabla.

*Tabla 2.6: Planificación de la Iteración 5*

TAREAS	FECHA DE INICIO	FECHA DE FIN	DURACIÓN EN DÍAS	DÍA DE COMIENZO
<b>ITERACIÓN 5</b>	29/10/2023	15/11/2023	18	118
REQUISITOS Y CASOS DE USO	29/10/2023	30/10/2023	2	118
PANTALLA DE GRÁFICAS	31/10/2023	13/11/2023	14	120
PRUEBAS	14/11/2023	15/11/2023	2	134

Por último, es necesario definir, planificar y estimar las tareas de la última fase, la revisión y puesta en marcha. Esta fase es distinta al resto ya que se basa en realizar las pruebas finales con el sistema ya completo y el despliegue final. Observamos estas tareas en la siguiente tabla.

*Tabla 2.7: Planificación de la Revisión y Puesta en marcha*

TAREAS	FECHA DE INICIO	FECHA DE FIN	DURACIÓN EN DÍAS	DÍA DE COMIENZO
<b>REVISIÓN Y PUESTA EN MARCHA</b>	16/11/2023	23/11/2023	8	136
PRUEBAS FINALES	16/11/2023	21/11/2023	6	136
DESPLIEGUE	22/11/2023	23/11/2023	2	142

Por último, en la siguiente figura podemos observar el diagrama de Gantt de todas las tareas que conforman el proyecto, así de las distintas fases repartidas desde el primer día del proyecto hasta el último.

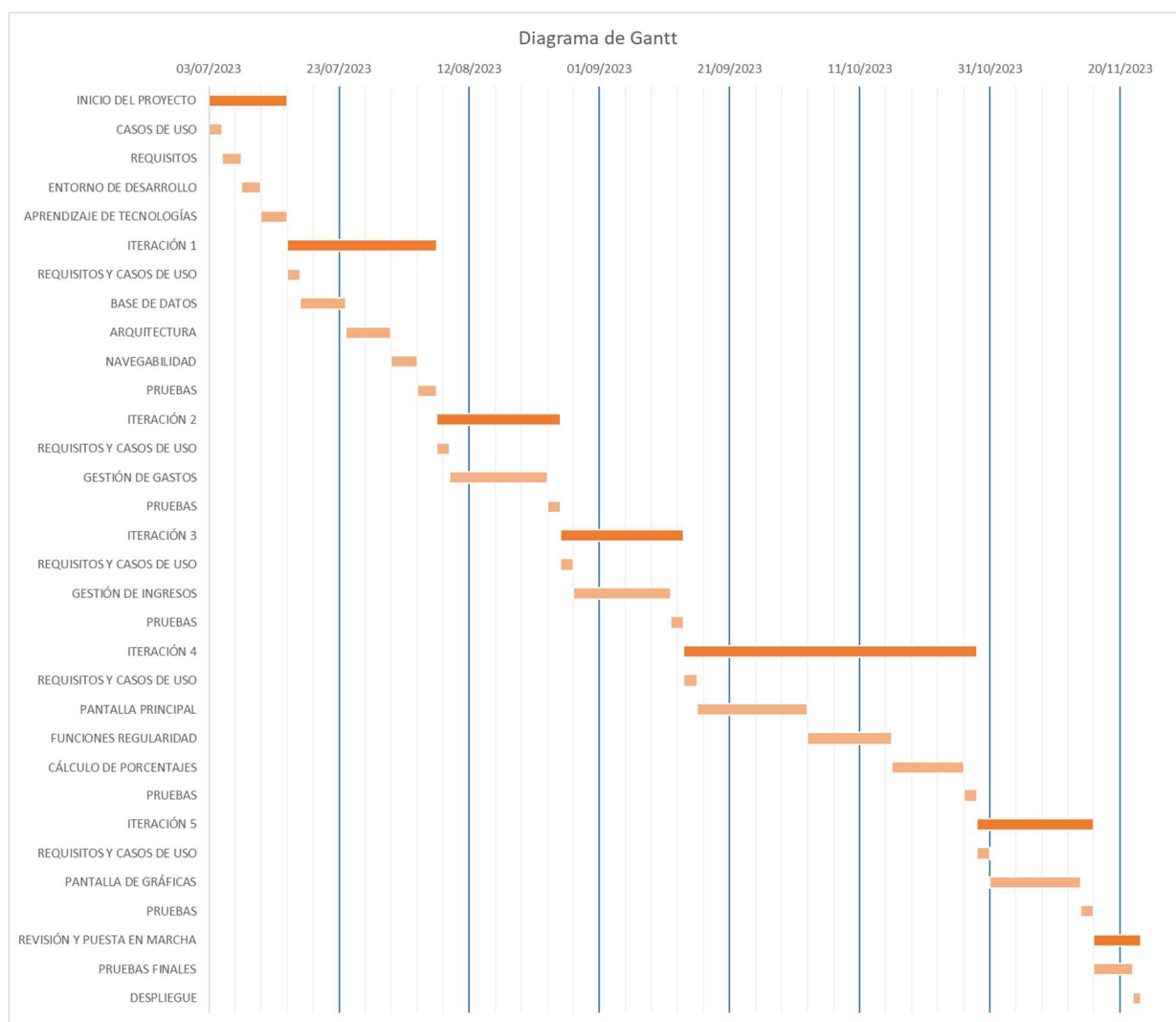


Figura 2.1: Diagrama de Gantt del proyecto

## 2.3 GESTIÓN DE RECURSOS

### 2.3.1 ESPECIFICACIÓN DE RECURSOS

En la siguiente tabla podemos observar la especificación de recursos, viendo la media salarial anual y calculando el salario por hora teniendo en cuenta los días laborables al año.

*Tabla 2.8: Especificación de recursos*

RECURSO	MEDIA SALARIAL (€)	SALARIO/HORA (€)
Diseñador Gráfico	21.350,00 €	10,63 €
Desarrollador Junior	27.985,00 €	13,94 €
Desarrollador Senior	44.607,00 €	22,21 €

Datos recogidos de la web “Indeed Salaries” buscando por puesto en España en 2023. Para calcular el salario por hora se ha utilizado la siguiente fórmula [13]:

$$\frac{\text{Media Salario Anual}}{251 \text{ días laborables} \times 8 \text{ horas diarias}}$$

### 2.3.2 ASIGNACIÓN DE RECURSOS

La asignación de horas no es igual para todos los roles ya que cada rol efectúa tareas totalmente distintas dentro del proyecto. El desarrollador senior se encarga de crear la arquitectura, definir los requisitos, casos de uso y crear el esqueleto en el que construirá el desarrollador junior el resto del proyecto. En cambio, el diseñador gráfico crea los mockups necesarios para que los desarrolladores efectúen la parte gráfica. En la siguiente tabla se muestra el número de horas destinadas a cada rol:

*Tabla 2.9: Asignación de recursos (por horas)*

RECURSO	COMIENZO	FIN	HORAS
Diseñador Gráfico	03/07/2023	14/07/2023	96
Desarrollador Junior	03/07/2023	23/11/2023	1152
Desarrollador Senior	03/07/2023	30/07/2023	152

## 2.4 PRESUPUESTO

En esta sección se lleva a cabo una evaluación global de los gastos asociados al proyecto, abarcando desde los costos generados por la planificación de las actividades hasta los relacionados con el hardware y el software. Se incluye una tabla que resume los costos totales organizados por categorías.

### 2.4.1 COSTES DE HARDWARE

En esta sección se recogen los costes destinados al hardware para llevar a cabo el proyecto. Se tiene en cuenta que trabajan 3 personas simultáneamente. Podemos consultar los costes en la siguiente tabla.

*Tabla 2.10: Costes de hardware*

COMPONENTE	UNIDADES	PRECIO/UNIDAD (€)	PRECIO (€)
PROCESADOR	3	200,00 €	600,00 €
MEMORIA RAM	3	60,00 €	180,00 €
MEMORIA ROM	3	50,00 €	150,00 €
PLACA BASE	3	100,00 €	300,00 €
ALIMENTACIÓN	3	70,00 €	210,00 €
REFRIGERACIÓN	3	40,00 €	120,00 €
CAJA	3	50,00 €	150,00 €
PERIFERICOS	3	120,00 €	360,00 €
<b>TOTAL</b>			<b>2.070,00 €</b>

### 2.4.2 COSTES DE SOFTWARE

En la siguiente tabla se recogen los costes relacionados con el software para que se lleve a cabo el proyecto.

*Tabla 2.11: Costes de software*

SOFTWARE	UNIDADES	PRECIO/UNIDAD (€)	PRECIO (€)
GITHUB COPILOT	2	114,00 €	228,00 €
ILUSTRATOR	1	40,52 €	40,52 €
WINDOWS 11	1	124,00 €	124,00 €
<b>TOTAL</b>			<b>392,52 €</b>

Podemos ver que solo necesitamos 2 licencias de Github Copilot, ya que sólo los desarrolladores utilizarán el servicio. Lo mismo ocurre con las licencias de Windows 11 y Illustrator, que la utilizarán el desarrollador gráfico. Los desarrolladores utilizarán Ubuntu, ya que es una distribución de Linux libre y gratuita.

### 2.4.3 COSTES TOTALES

Aquí se presenta la totalidad de los costes, mostrando la suma de cada uno de los elementos detallados previamente. Como podemos observar hacen un total de 22.915 €.

Tabla 2.12: Costes totales

TIPO	PRECIO (€)
<b>COSTES HUMANOS</b>	20.452,48 €
<b>COSTES HARDWARE</b>	2.070,00 €
<b>COSTES SOFTWARE</b>	392,52 €
<b>TOTAL</b>	22.915,00 €

En el siguiente grafico podemos observar como el coste humano supone el 89 % del coste, el coste hardware el 9% y el coste software el 2% del total.

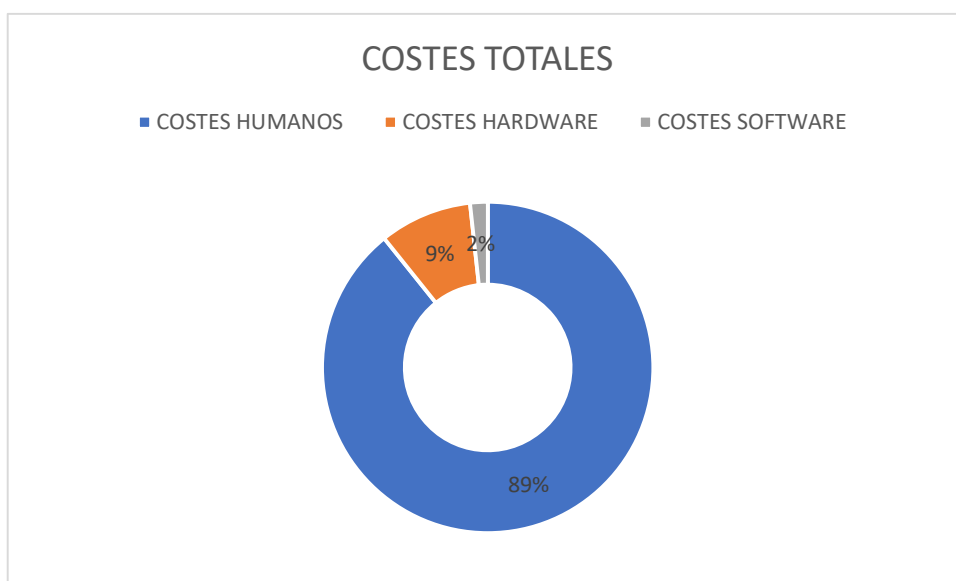


Figura 2.2: Gráfico de costes



### 3. SOLUCIÓN

La solución del proyecto detalla la respuesta a los problemas mencionados en la sección de problemas del documento. Se proporcionará una descripción concisa de la solución propuesta, el proceso de desarrollo y el resultado obtenido en el proyecto.

#### 3.1 DESCRIPCIÓN DE LA SOLUCIÓN

La solución adoptada para resolver el problema se basa en crear una aplicación móvil sencilla e intuitiva en donde los usuarios con muy poca formación y conocimiento financiero puedan tener una fuerte salud económica.

El funcionamiento de la aplicación es muy sencillo, lo que es un objetivo primordial del proyecto. Se basa en 4 pantallas primarias, la pantalla de información, la pantalla de gastos, la pantalla de ingresos y la pantalla de gráficos. Estas pantallas están divididas y son accesibles desde unos botones con icono en un menú inferior muy típico de las aplicaciones de móviles.

En la pantalla de información se encuentran tus gastos del mes actual, tus gastos anuales y la recomendación. En la recomendación se ejecuta un algoritmo que calcula lo siguiente. Primero calcula el porcentaje de tus ingresos que corresponde a los gastos importantes (como pueden ser la comida, facturas o el alquiler) y el porcentaje los gastos no importantes o innecesarios (como pueden ser gastos en restaurantes, fiestas, conciertos, subscripciones a servicios cinematográficos...) mensuales, después calcula lo mismo pero anual. Según el principio financiero que aplicamos en esta aplicación los gastos importantes no pueden superar el 50% de tus ingresos, los gastos innecesarios no pueden superar el 30% de tus ingresos y el 20% deberían de ser ahorro. Si te pasas tanto mensualmente como anualmente en el porcentaje marcado de algunos de los gastos la aplicación te recomienda bajar el gasto en esa categoría para poder seguir el principio aplicado.

Para gestionar esa información están las 2 pantallas siguientes. La primera que nos encontramos es la pantalla de gastos. En esta pantalla se te muestra una lista de los gastos del mes actual desglosados. En esta pantalla se puede navegar entre meses para poder consultar esta información. Esta pantalla tiene un botón flotante en la parte inferior derecha en donde podemos añadir gastos. Emergerá una pantalla con cajas de texto en donde se puede rellenar la información necesaria para crear un gasto. Por resumir, es necesario rellenar el gasto con importe, descripción, fecha, categoría y regularidad. En el caso del importe y descripción se rellena la información en la propia caja de texto. En el caso de la categoría y regularidad es un desplegable que eliges la opción que quieres. Las opciones que hay para elegir son en el caso de

la categoría, necesario e innecesario, y en el caso de la regularidad, puntual, semanal, mensual y anual, de lo que hablaremos luego. Para elegir la fecha hay un botón que al pulsarlo emerge un calendario en donde se puede elegir la fecha del gasto.

Además de poder crear gastos, también se puede modificarlos y eliminarlos. Para eso hay que pulsar en el gasto que se quiere interactuar con él y navegarás a otra pantalla parecida a la de crear el gasto y ahí se puede modificar los datos o simplemente eliminar el gasto.

La regularidad funciona como unos gastos recurrentes que se registran automáticamente en función del tipo de regularidad que hayas elegido al crear el gasto. Un ejemplo perfecto es el alquiler, se crea eligiendo la regularidad mensual y no es necesario registrarlo todos los meses. En el momento que se quiera cancelar ese gasto hay 2 opciones, una actualizar el último registro y ponerlo como gasto puntual, u otra es eliminar el último gasto.

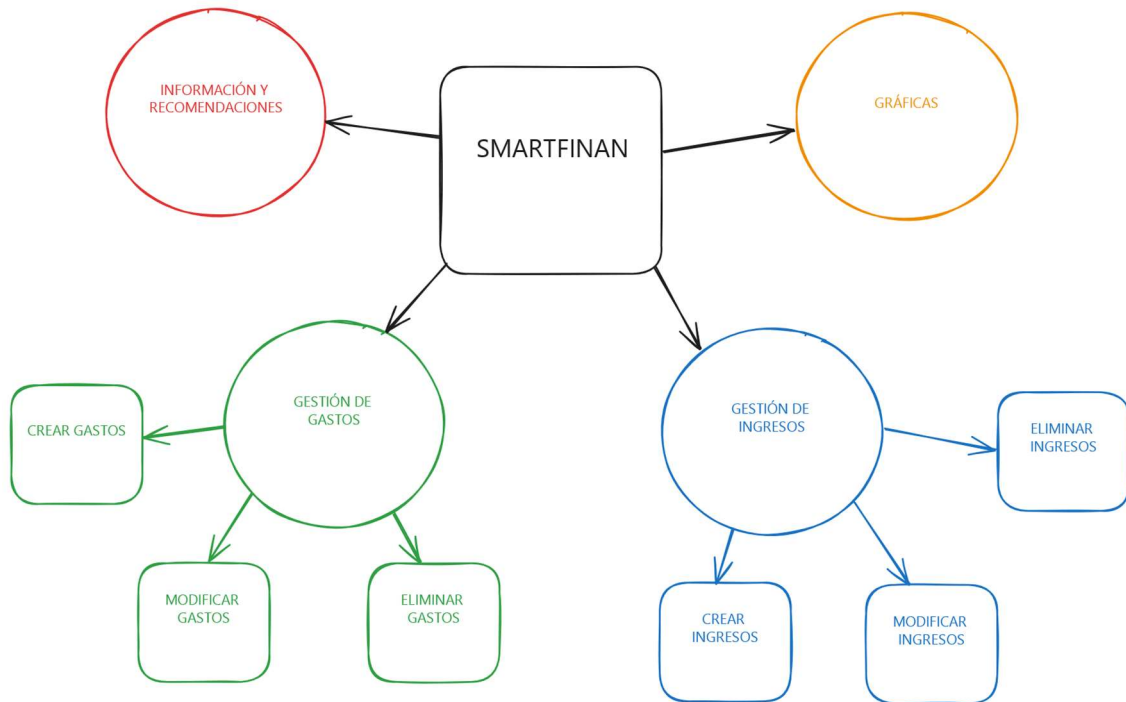
Para explicar el funcionamiento de la gestión de los ingresos es muy fácil ya habiendo explicado cómo funcionan los gastos, es exactamente lo mismo con una única diferencia, no existe la categoría en los ingresos. Para no repetir toda la información consulte cómo funcionan los gastos explicados anteriormente.

Por último, está la pantalla de los gráficos. Esta pantalla es muy sencilla, de forma vertical está representado el importe, y de forma horizontal está representado el tiempo en meses. El gráfico representa el año actual. También tenemos una leyenda donde están referenciado los gastos importantes, gastos no importantes y el ahorro en distintos colores. Teniendo todo esto se forma una gráfica de barras en donde se puede ver tu información financiera de una forma mucho más global y sencilla.

### 3.1.1 DESCOMPOSICIÓN EN SUBSISTEMAS

La forma más fácil de entender el proyecto es dividiéndolo en diferentes subsistemas con sus objetivos y competencias bien definidos.

Se resuelve este problema con la siguiente figura, en donde vemos los siguientes subsistemas bien definidos.



*Figura 3.1: Subsistemas de la aplicación*

A continuación de esta figura en donde se pueden ver las distintas partes del proyecto completo, se explica cada una de las partes de forma detallada.

#### 3.1.1.1 INFORMACIÓN Y RECOMENDACIÓN

En la sección de información, encontrarás un resumen de tus gastos mensuales actuales, así como los gastos totales anuales, junto con una recomendación específica. Esta recomendación se genera a través de un algoritmo diseñado para calcular diferentes porcentajes en relación con tus ingresos. En primer lugar, determina el porcentaje de ingresos dedicado a gastos esenciales, como alimentación, facturas o alquiler, comparado con el porcentaje destinado a gastos no esenciales o superfluos, como salidas a restaurantes, entretenimiento, suscripciones a servicios, etc., tanto a nivel mensual como anual.

Siguiendo el principio financiero adoptado por esta aplicación, se establece que los gastos esenciales no deben sobrepasar el 50% de tus ingresos, los gastos no esenciales no deben exceder el 30% y se sugiere destinar al menos un 20% al ahorro. En caso de que alguno de tus gastos exceda estos porcentajes, ya sea mensual o anualmente, la aplicación te aconsejará reducir el gasto en esa categoría para ajustarte a estos lineamientos financieros.

#### 3.1.1.2 GESTIÓN DE GASTOS

En este subsistema, visualizarás un resumen detallado de tus gastos del mes en curso, organizados de forma desglosada para facilitar su consulta. La interfaz te permite explorar los distintos meses para acceder a la información histórica de tus gastos. En la esquina inferior

derecha, encontrarás un botón interactivo que te permite agregar nuevos gastos de manera rápida. Al presionarlo, se desplegará una pantalla que presenta campos específicos para ingresar los detalles del gasto.

Para agregar un gasto, se requiere proporcionar el importe, descripción, fecha, categoría y periodicidad. En los campos de importe y descripción, simplemente se ingresa la información directamente. En cuanto a la categoría y la periodicidad, se presentan como menús desplegables, donde puedes seleccionar entre "necesario" o "innecesario" para la categoría, y para la periodicidad, las opciones disponibles son "puntual", "semanal", "mensual" y "anual". Para elegir la fecha correspondiente al gasto, se dispone de un botón que abre un calendario, permitiéndote seleccionar fácilmente el día deseado.

Además de la capacidad de agregar nuevos gastos, tienes la posibilidad de modificar o eliminar los existentes. Al seleccionar un gasto específico, serás dirigido a una pantalla similar a la de creación de gasto, donde tendrás la opción de editar los datos o eliminar directamente el gasto.

El concepto de periodicidad representa los gastos recurrentes que se registran automáticamente según la selección de la frecuencia al crear el gasto. En caso de querer cancelar este gasto recurrente, existen dos opciones: la primera es actualizar el registro más reciente y convertirlo en un gasto puntual, y la segunda opción implica eliminar el último registro del gasto en cuestión.

#### 3.1.1.3 GESTIÓN DE INGRESOS

En este subsistema, se detalla el manejo y seguimiento de tus ingresos, siguiendo una estructura similar a la explicada para los gastos, con una diferencia crucial: la ausencia de una categorización para los ingresos.

Esta pantalla presenta un desglose detallado de tus ingresos mensuales y permite la navegación entre distintos periodos para acceder a la información histórica. Además, integra un botón flotante ubicado en la esquina inferior derecha, facilitando la incorporación rápida de nuevos ingresos. Al hacer clic en este botón, se despliega una pantalla que contiene campos para ingresar los detalles del ingreso.

Para añadir un ingreso, simplemente se debe proporcionar el monto, una descripción, la fecha y la periodicidad, siguiendo una metodología similar a la de los gastos. Los campos de monto y descripción se completan directamente, mientras que la fecha se elige mediante un calendario emergente.

Asimismo, al igual que con los gastos, esta sección también permite la modificación o eliminación de los ingresos existentes. Al seleccionar un ingreso específico, se redirige a una pantalla similar a la de creación de ingreso, brindando la opción de editar los datos o eliminar directamente la entrada de ingreso.

El concepto de periodicidad en los ingresos opera de manera análoga a los gastos recurrentes. Por ejemplo, si recibes ingresos mensuales fijos, se registra automáticamente según la periodicidad establecida al crear el ingreso correspondiente.

En definitiva, la gestión de ingresos sigue un proceso similar al de los gastos, pero sin la necesidad de categorización, facilitando el seguimiento y control de tus ingresos de manera eficiente.

#### 3.1.1.4 GRÁFICAS

En el subsistema de gráficos, se presenta una interfaz simple y directa. La información se representa en el eje vertical el importe, mientras que en el eje horizontal se representa el tiempo, en meses, abarcando el año actual. El gráfico ofrece una perspectiva visual de tu información financiera mediante barras.

En esta visualización, se destaca una leyenda que identifica y diferencia los distintos elementos. Los gastos importantes, los gastos no importantes y el ahorro se presentan en colores diferenciados para una fácil interpretación. Esta segmentación permite una comprensión más clara y global de tu situación financiera.

La gráfica de barras ofrece una representación visual dinámica, brindando una visión panorámica de tus datos financieros de manera sencilla y accesible. Esta herramienta te permite analizar rápidamente la distribución de tus gastos importantes, gastos menos esenciales y tu capacidad de ahorro en el transcurso del año actual.

## 3.2 ANÁLISIS DEL SISTEMA

En esta sección del documento se expone la información referente a los requisitos del proyecto, tanto funcionales como no funcionales, así como los casos de uso y su especificación.

### 3.2.1 DEFINICIÓN DE REQUISITOS

En esta sección se definen los requisitos no funcionales, y los funcionales categorizados cada uno en el subsistema del que pertenece para su mejor comprensión y listado [14].

#### 3.2.1.1 REQUISITOS NO FUNCIONALES

Los requisitos no funcionales que conforman este proyecto se encuentran en esta lista.

- **RNF01** – Diseño intuitivo y simple
- **RNF02** – Interfaz con XML y View Binding
- **RNF03** – Dispositivos móviles con Android
- **RNF04** – Arquitectura MVVM
- **RNF05** – Backend en Kotlin
- **RNF06** – Base de datos SQLite con Room
- **RNF07** – Corutinas
- **RNF08** – Controlador de versiones Git
- **RNF09** – Gráficas con librería MPAndroidChart

#### 3.2.1.2 REQUISITOS FUNCIONALES

Los requisitos funcionales que conforman este proyecto están recogidos en las siguientes listas.

##### 3.2.1.2.1 INFORMACIÓN Y RECOMENDACIONES

- **RF 1** – Consulta de gasto de mes actual
- **RF 2** – Consulta de gasto de año actual
- **RF 3** – Consulta de recomendación del algoritmo

##### 3.2.1.2.2 GESTIÓN DE GASTOS

- **RF 4** – Consulta de gastos desglosados del mes elegido
- **RF 5** – Cambio de mes
- **RF 6** – Crear gasto
- **RF 7** – Modificar gasto
- **RF 8** – Borrar gasto
- **RF 9** – Creación automática de gasto periódico

##### 3.2.1.2.3 GESTIÓN DE INGRESOS

- **RF 10** – Consulta de ingresos desglosados del mes elegido
- **RF 11** – Cambio de mes
- **RF 12** – Crear ingreso
- **RF 13** – Modificar ingreso
- **RF 14** – Borrar ingreso
- **RF 15** – Creación automática de ingreso periódico

##### 3.2.1.2.4 GRAFICAS

- **RF 16** – Consulta de gráficas de gasto y ahorro
- **RF 17** – Ampliar, reducir y moverse entre la gráfica

## 3.2.2 ESPECIFICACIÓN DE REQUISITOS

## 3.2.2.1 REQUISITOS NO FUNCIONALES

Tabla 3.1: Diseño intuitivo y simple

RNF01	Diseño intuitivo y simple
Descripción	La aplicación debe ser fácil de utilizar mediante un diseño intuitivo y simple
Prioridad	Alta
Tipo	Apariencia
Estado	Implementado

Tabla 3.2: Interfaz con XML y View Binding

RNF02	Interfaz con XML y View Binding
Descripción	La aplicación debe implementar su interfaz mediante archivos XML y usar View Binding para controlar la interfaz desde el Backend
Prioridad	Media
Tipo	Apariencia
Estado	Implementado

Tabla 3.3: Dispositivos Mviles con Android

RNF03	Dispositivos móviles con Android
Descripción	La aplicación debe poder instalarse en dispositivos móviles con Android
Prioridad	Alta
Tipo	Apariencia
Estado	Implementado

Tabla 3.4: Arquitectura MVVM

RNF04	Arquitectura MVVM
Descripción	La aplicación debe ser implementada mediante la arquitectura MVVM para su mejor mantenimiento y mejoras futuras
Prioridad	Alta
Tipo	Apariencia
Estado	Implementado

Tabla 3.5: Backend en Kotlin

RNF05	Backend en Kotlin
Descripción	El backend de la aplicación debe ser implementado con el lenguaje de programación Kotlin
Prioridad	Alta
Tipo	Apariencia
Estado	Implementado

Tabla 3.6: Base de datos SQLite con Room

RNF06	Base de datos SQLite con Room
Descripción	La base de datos que utiliza la aplicación debe ser una base de datos SQLite implementada con la librería Room
Prioridad	Media
Tipo	Apariencia
Estado	Implementado

Tabla 3.7: Corrutinas

RNF07	Corrutinas
Descripción	La aplicación debe utilizar corrutinas para las peticiones de los datos a nuestra base de datos SQLite [15]
Prioridad	Alta
Tipo	Apariencia
Estado	Implementado

Tabla 3.8: Controlador de versiones con Git

RNF08	Controlador de versiones Git
Descripción	El código fuente de la aplicación debe estar guardado en un repositorio en GitHub y utilizar un controlador de versiones como Git
Prioridad	Media
Tipo	Apariencia
Estado	Implementado



Tabla 3.9: Gráficas con librería MPAndroidChart

RNF09	Gráficas con librería MPAndroidChart
Descripción	La aplicación debe utilizar gráficas implementando la librería llamada MPAndroidChart [16]
Prioridad	Media
Tipo	Apariencia
Estado	Implementado

### 3.2.2.2 REQUISITOS FUNCIONALES

#### 3.2.2.2.1 INFORMACIÓN Y RECOMENDACIONES

Tabla 3.10: Consulta gastos del mes actual

RF 1	Consulta gastos del mes actual
Descripción	Se debe de mostrar los gastos del mes actual en la pantalla principal.
Prioridad	Media
Subsistema	Información y recomendaciones
Estado	Implementado

Tabla 3.11: Consulta gasto del año actual

RF 2	Consulta gasto del año actual
Descripción	Se debe de mostrar los gastos del año actual en la pantalla principal.
Prioridad	Media
Subsistema	Información y recomendaciones
Estado	Implementado

Tabla 3.12: Consulta de recomendación del algoritmo

RF 3	Consulta de recomendación del algoritmo
Descripción	Se debe de mostrar las recomendaciones del algoritmo en la pantalla principal.
Prioridad	Alta
Subsistema	Información y recomendaciones
Estado	Implementado

## 3.2.2.2.2 GESTIÓN DE GASTOS

Tabla 3.13: Consulta de gastos desglosados del mes elegido

RF 4	Consulta de gastos desglosados del mes elegido
Descripción	Se debe de mostrar una lista de cada uno de los gastos en un mes en particular, con su nombre, descripción, categoría e importe
Prioridad	Alta
Subsistema	Gestión de gastos
Estado	Implementado

Tabla 3.14: Cambio de mes de gastos

RF 5	Cambio de mes de gastos
Descripción	Se debe poder cambiar el mes en los gastos pulsando un botón
Prioridad	Alta
Subsistema	Gestión de gastos
Estado	Implementado

Tabla 3.15: Crear gasto

RF 6	Crear gasto
Descripción	Se debe poder crear un gasto a partir de una interfaz en donde se pueden introducir los detalles, el nombre, la categoría, la periodicidad, el importe y la fecha del gasto
Prioridad	Alta
Subsistema	Gestión de gastos
Estado	Implementado

Tabla 3.16: Modificar gasto

RF 7	Modificar gasto
Descripción	Se debe poder modificar un gasto a partir de una interfaz en donde se pueden introducir los detalles, el nombre, la categoría, la periodicidad, el importe y la fecha del gasto
Prioridad	Alta
Subsistema	Gestión de gastos
Estado	Implementado

Tabla 3.17: Borrar gasto

RF 8	Borrar gasto
Descripción	Se debe poder eliminar un gasto elegido
Prioridad	Media
Subsistema	Gestión de gastos
Estado	Implementado

Tabla 3.18: Creación automática de gasto periódico

RF 9	Creación automática de gasto periódico
Descripción	Se debe de crear automáticamente gastos que previamente han sido declarados periódicos. El día que cumpla el ciclo de periodicidad se crea el gasto automáticamente con los mismos datos, cambiando la fecha
Prioridad	Alta
Subsistema	Gestión de gastos
Estado	Implementado

### 3.2.2.2.3 GESTIÓN DE INGRESOS

Tabla 3.19: Consulta de ingresos desglosados del mes elegido

RF 10	Consulta de ingresos desglosados del mes elegido
Descripción	Se debe de mostrar una lista de cada uno de los ingresos en un mes en particular, con su descripción, fecha e importe
Prioridad	Alta
Subsistema	Gestión de gastos
Estado	Implementado

Tabla 3.20: Cambio de mes de ingresos

RF 11	Cambio de mes de ingresos
Descripción	Se debe poder cambiar el mes en los ingresos pulsando un botón
Prioridad	Alta
Subsistema	Gestión de ingresos
Estado	Implementado

*Tabla 3.21: Crear ingreso*

RF 12	Crear ingreso
Descripción	Se debe poder crear un ingreso a partir de una interfaz en donde se pueden introducir los detalles, el nombre, la periodicidad, el importe y la fecha del ingreso
Prioridad	Alta
Subsistema	Gestión de ingresos
Estado	Implementado

*Tabla 3.22: Modificar ingreso*

RF 13	Modificar ingreso
Descripción	Se debe poder modificar un ingreso a partir de una interfaz en donde se pueden introducir los detalles, el nombre, la periodicidad, el importe y la fecha del ingreso
Prioridad	Alta
Subsistema	Gestión de ingresos
Estado	Implementado

*Tabla 3.23: Borrar ingreso*

RF 14	Borrar ingreso
Descripción	Se debe poder eliminar un ingreso elegido
Prioridad	Media
Subsistema	Gestión de ingresos
Estado	Implementado

*Tabla 3.24: Creación automática de ingreso periódico*

RF 15	Creación automática de ingreso periódico
Descripción	Se debe de crear automáticamente ingresos que previamente han sido declarados periódicos. El día que cumpla el ciclo de periodicidad se crea el ingreso automáticamente con los mismos datos, cambiando la fecha
Prioridad	Alta
Subsistema	Gestión de ingresos
Estado	Implementado

## 3.2.2.2.4 GRÁFICAS

Tabla 3.25: Consulta de gráficas de gasto y ahorro

RF 16	Consulta de gráficas de gasto y ahorro
Descripción	Se debe de mostrar unas gráficas en una pantalla específica en donde se pueda ver gráficamente el gasto necesario, innecesario y el ahorro
Prioridad	Alta
Subsistema	Gráficas
Estado	Implementado

Tabla 3.26: Ampliar, reducir y moverse entre la gráfica

RF 17	Ampliar, reducir y moverse entre la gráfica
Descripción	Se debe poder ampliar, reducir y moverse en la gráfica para ver mejor la información.
Prioridad	Media
Subsistema	Gráficas
Estado	Implementado

## 3.2.3 CASOS DE USO

A continuación de dejar claro cuáles son los requisitos del proyecto es hora de definir los casos de uso. Para ello definimos los actores que pueden interactuar en la aplicación. En este caso tenemos 2 actores, el usuario, que es el actor principal del proyecto, y el sistema, que realiza acciones automáticas, por lo tanto, cambia el estado de la aplicación.

En las siguientes figuras veremos definidos los casos de uso de la aplicación.

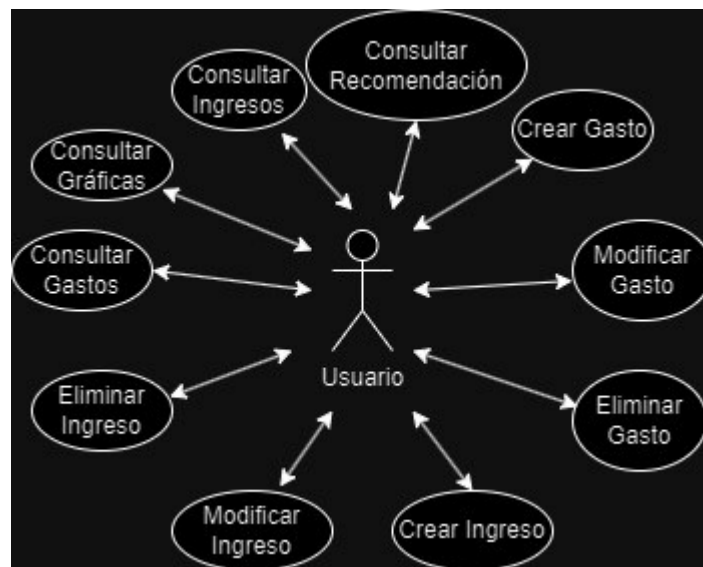


Figura 3.2: Casos de uso del usuario

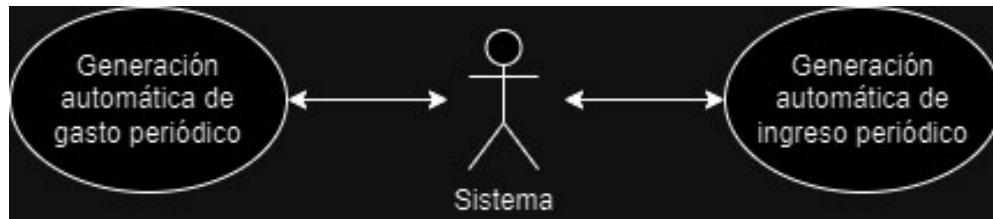


Figura 3.3: Casos de uso del sistema

Ahora que ya tenemos definidos los casos de uso, en la siguiente sección se hará una especificación integral de los mismos.

### 3.2.4 ESPECIFICACIÓN DE LOS CASOS DE USO

En esta sección, para especificar los casos de uso, se describirán los siguientes elementos que forman un caso de uso [17].

- Identificador: se utiliza para identificar cada caso de uso de una manera única. La terminología es “UC-XX” siendo XX un número empezando por 01.
- Nombre: el nombre del caso de uso definido en las anteriores figuras.
- Actor implicado: el nombre del actor que ejecuta el caso de uso.
- Descripción: una breve descripción que explica el caso de uso.
- Precondiciones: define el estado en el que tiene que estar el sistema para que el caso de uso pueda llegar a ejecutarse.
- Escenario básico: indica los pasos que deben seguirse para realizar el caso de uso.
- Postcondiciones: indica las condiciones que se cumplirán al finalizar el caso de uso.

Tabla 3.27: Caso de uso - Crear gasto

Identificador	UC-01
Nombre	Crear gasto
Actor	Usuario
Descripción	Tiene como objetivo crear un gasto
Precondiciones	El usuario debe estar en la pantalla de gastos
Escenario básico	1 - El usuario pulsa el botón flotante de añadir. 2 - El usuario debe introducir los datos del gasto. 3 - El usuario pulsa el botón de confirmar
Postcondiciones	Se crea un nuevo gasto y se puede interactuar con él.

Tabla 3.28: Caso de uso - Actualizar gasto

Identificador	UC-02
Nombre	Actualizar gasto
Actor	Usuario
Descripción	Tiene como objetivo actualizar un gasto
Precondiciones	El usuario debe estar en la pantalla de gastos. El gasto debe de existir.
Escenario básico	1 - El usuario pulsa en el nombre del gasto que quiere actualizar. 2 - El usuario debe introducir los datos del gasto. 3 - El usuario pulsa el botón de confirmar
Postcondiciones	Se actualiza un nuevo gasto y se puede interactuar con él.

Tabla 3.29: Caso de uso - Borrar gasto

Identificador	UC-03
Nombre	Borrar gasto
Actor	Usuario
Descripción	Tiene como objetivo borrar un gasto
Precondiciones	El usuario debe estar en la pantalla de gastos. El gasto debe de existir.
Escenario básico	1 - El usuario pulsa en el nombre del gasto que quiere borrar. 2 - El usuario pulsa el botón de borrar y confirma la decisión.
Postcondiciones	Se borra el gasto y ya no se puede interactuar con él.

Tabla 3.30: Caso de uso – Crear ingreso

Identificador	UC-04
Nombre	Crear ingreso
Actor	Usuario
Descripción	Tiene como objetivo crear un ingreso
Precondiciones	El usuario debe estar en la pantalla de ingresos
Escenario básico	1 - El usuario pulsa el botón flotante de añadir. 2 - El usuario debe introducir los datos del ingreso. 3 - El usuario pulsa el botón de confirmar
Postcondiciones	Se crea un nuevo ingreso y se puede interactuar con él.

Tabla 3.31: Caso de uso – Actualizar ingreso

Identificador	UC-05
Nombre	Actualizar ingreso
Actor	Usuario
Descripción	Tiene como objetivo actualizar un ingreso
Precondiciones	El usuario debe estar en la pantalla de ingresos. El ingreso debe de existir.
Escenario básico	1 - El usuario pulsa en el nombre del ingreso que quiere actualizar. 2 - El usuario debe introducir los datos del ingreso. 3 - El usuario pulsa el botón de confirmar
Postcondiciones	Se actualiza un nuevo ingreso y se puede interactuar con él.

Tabla 3.32: Caso de uso – Borrar ingreso

Identificador	UC-06
Nombre	Borrar ingreso
Actor	Usuario
Descripción	Tiene como objetivo borrar un ingreso
Precondiciones	El usuario debe estar en la pantalla de ingresos. El ingreso debe de existir.
Escenario básico	1 - El usuario pulsa en el nombre del ingreso que quiere borrar. 2 - El usuario pulsa el botón de borrar y confirma la decisión.
Postcondiciones	Se borra el ingreso y ya no se puede interactuar con él.

Tabla 3.33: Caso de uso – Consultar gastos

Identificador	UC-07
Nombre	Consultar gastos
Actor	Usuario
Descripción	Tiene como objetivo consultar los gastos
Precondiciones	El usuario debe estar en la pantalla de gastos
Escenario básico	1 - El usuario busca el mes del que quiere consultar los gastos
Postcondiciones	Ninguna



Tabla 3.34: Caso de uso – Consultar ingresos

Identificador	UC-08
Nombre	Consultar ingresos
Actor	Usuario
Descripción	Tiene como objetivo consultar los ingresos
Precondiciones	El usuario debe estar en la pantalla de ingresos
Escenario básico	1 - El usuario busca el mes del que quiere consultar los ingresos
Postcondiciones	Ninguna

Tabla 3.35: Caso de uso – Consultar gráficas

Identificador	UC-09
Nombre	Consultar gráficas
Actor	Usuario
Descripción	Tiene como objetivo ver tus movimientos de una manera más global
Precondiciones	El usuario debe estar en la pantalla de gráficas
Escenario básico	1 - El usuario navega por la pestaña de la gráfica
Postcondiciones	Ninguna

Tabla 3.36: Caso de uso – Consultar recomendación

Identificador	UC-10
Nombre	Consultar recomendación
Actor	Usuario
Descripción	Tiene como objetivo mejorar los hábitos de gasto del usuario
Precondiciones	El usuario debe estar en la pantalla principal
Escenario básico	1 - El usuario consulta la recomendación que el sistema calcula
Postcondiciones	Ninguno

Tabla 3.37: Caso de uso – Generación automática de gasto periódico

Identificador	UC-11
Nombre	Generación automática de gasto periódico
Actor	Sistema
Descripción	Tiene como objetivo crear un gasto automáticamente
Precondiciones	El usuario debe haber creado un gasto periódico y cumplirse el ciclo de periodicidad.
Escenario básico	1 - El sistema comprueba si el ciclo se ha cumplido 2 - Si el ciclo se ha cumplido, crea un gasto igual al predecesor, pero con la fecha correcta.
Postcondiciones	Se crea un nuevo gasto y se puede interactuar con él.

Tabla 3.38: Caso de uso – Generación automática de ingreso periódico

Identificador	UC-12
Nombre	Generación automática de ingreso periódico
Actor	Sistema
Descripción	Tiene como objetivo crear un ingreso automáticamente
Precondiciones	El usuario debe haber creado un ingreso periódico y cumplirse el ciclo de periodicidad.
Escenario básico	1 - El sistema comprueba si el ciclo se ha cumplido 2 - Si el ciclo se ha cumplido, crea un ingreso igual al predecesor, pero con la fecha correcta.
Postcondiciones	Se crea un nuevo ingreso y se puede interactuar con él.

### 3.3 DISEÑO DEL SISTEMA

En esta sección de la memoria se explica todo lo referente al diseño en el que se construyó todo el producto, con su arquitectura, su explicación de cómo se fueron encontrando las soluciones a los distintos requisitos y cómo se alcanzaron los objetivos expuestos en la fase de análisis.

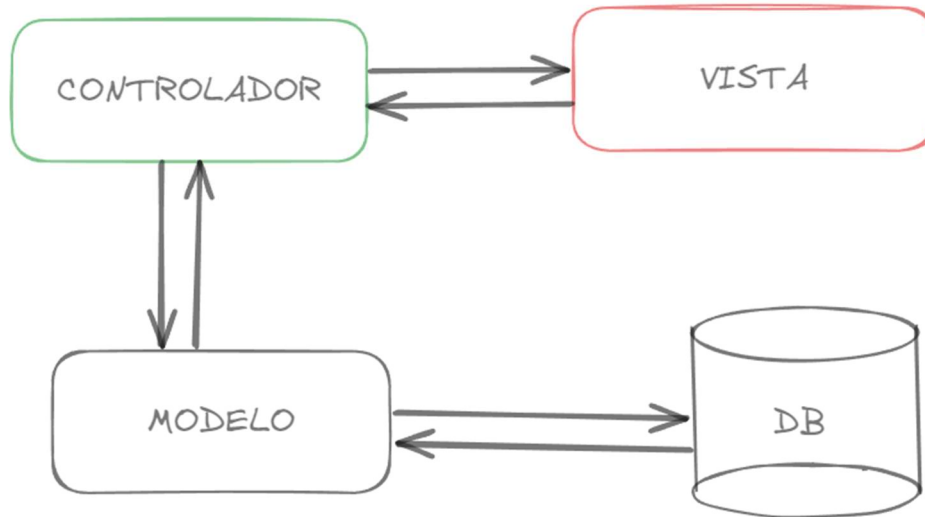
#### 3.3.1 CONTEXTO

Al principio del proyecto se buscaba una aplicación que al desarrollarla cumpliera una serie de conceptos pensados para facilitarle la vida a las personas que trabajasen sobre el código fuente como:

- Separación de responsabilidades: la interfaz gráfica, la lógica de negocio y el acceso a datos son responsabilidades que deberían ser totalmente independientes entre ellas.
- Reutilización de código: es la idea de programar, cumplir con el mayor número de funciones con el menor número de código posible, así es más fácil el día que cambien los requisitos de hacer el cambio.
- Escalabilidad: va de la mano con los conceptos anteriores, si hay poco código y bien organizado es muy fácil que crezca el producto correctamente.
- Mantenibilidad: una parte muy importante de la ingeniería del software. Muchos proyectos dejan de funcionar por la cantidad de trabajo y dinero que requieren su soporte.

Estos conceptos y facilidades y más, son otorgados por el patrón de diseño MVC (Modelo-Vista-Controlador), uno de los patrones de diseño más utilizados en el desarrollo de software [18].

Sobre la siguiente figura nos apoyaremos en explicar este patrón muy importante para este proyecto.



*Figura 3.4: Modelo-Vista-Controlador*

La vista es el conjunto de responsabilidades de la aplicación de comunicarse con el usuario. En esta parte el usuario mediante una serie de elementos como botones, cajas de texto editables, formularios, datepickers, radiobuttons, comboboxes y mucho más, el usuario puede introducirle la información al sistema. La vista también se ocupa de la comunicación opuesta, del sistema hacia el usuario, normalmente mediante texto, imágenes, animaciones o sonidos.

La vista se comunica de manera bidireccional con el controlador, el “cerebro” del programa. El controlador es el que realiza el manejo de las interacciones del usuario. Su función principal es controlar el flujo de la aplicación y coordinar las solicitudes del usuario, realizando operaciones basadas en esas interacciones. A fin de cuentas, es un intermediario entre la vista y el modelo.

El modelo representa la capa de datos y lógica de negocio de una aplicación. Este componente es responsable de gestionar los datos, realizar operaciones sobre esos datos y aplicar la lógica empresarial necesaria para el funcionamiento de la aplicación. Generalmente en el modelo trabaja un ORM (Object-Relational Mapping) para facilitar el acceso a datos y minimizar la cantidad de código SQL que hay que escribir [19].

### 3.3.2 ARQUITECTURA SOFTWARE

La arquitectura utilizada en este proyecto es MVVM (Model-View-ViewModel) que se puede considerar una adaptación al patrón MVC para resolver ciertos problemas encontrados en aplicaciones móviles entre otras.

En MVVM, la Vista sigue siendo responsable de la presentación visual y la interacción con el usuario, similar al componente Vista en MVC. Sin embargo, en lugar de un Controlador, MVVM emplea un ViewModel.

En comparación con MVC, MVVM introduce el concepto de ViewModel, que actúa como un intermediario entre la Vista y el Modelo. El ViewModel contiene la lógica de presentación y facilita la comunicación entre la Vista y el Modelo. La principal diferencia radica en cómo se manejan los cambios de estado y la forma en que se actualiza la interfaz de usuario en respuesta a esos cambios.

En el caso del Modelo de MVVM, la idea es la misma que en MVC, aunque ciertas particularidades que se producen en las tecnologías utilizadas hacen que se ejecute la idea de una manera distinta. Esto se explicará más tarde.

Aunque MVVM comparte algunos conceptos fundamentales con MVC, está diseñado para abordar desafíos específicos que surgen en el desarrollo de aplicaciones móviles y de escritorio, ofreciendo una separación más clara de la lógica de presentación y una forma más efectiva de manejar las interacciones de usuario en esos entornos.

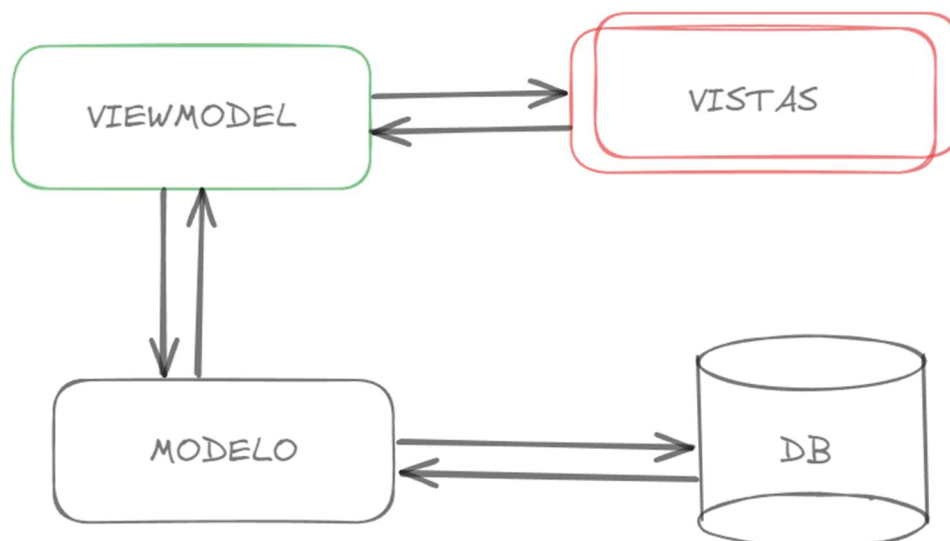


Figura 3.5: Model-View-ViewModel

### 3.3.3 ARQUITECTURA DEL SISTEMA

En esta sección se explica más a fondo, basándonos en la arquitectura software, como está estructurado el programa y qué tecnologías se utilizan en cada parte y como se conectan entre sí.

En primer lugar, tenemos las vistas. En esta parte tenemos archivos XML que definen tanto los componentes gráficos con los que el usuario va a interactuar, como los Layouts que van a definir en gran parte la posición de los componentes. Aquí se utilizan librerías de Android en XML que definen los componentes, como se usan y la variación en forma que se le puede dar.

Además de los archivos XML, tenemos unos archivos Kotlin en donde podemos definir ciertos comportamientos gráficos. Aquí es donde se le da vida a la interfaz, se define que hace cada acción en los componentes y cómo se muestran los cambios de estado en la propia interfaz.

Estos XML están conectados cada uno a su archivo Kotlin mediante la tecnología ViewBinding, una tecnología que definiendo un objeto de esta clase y el id de cada componente se puede acceder a él fácilmente y asignarle comportamiento.

Con la vista definida, pasamos al ViewModel. En esta capa se encuentran archivos que definen las operaciones de la aplicación mediante funciones en el lenguaje Kotlin. En el caso de este proyecto, la gran mayoría de trabajo que hace esta capa es la de definir cómo manipular las listas de datos mostradas en la gestión de gastos e ingresos.

En el entorno Android estas listas se llaman RecyclerView, y se llaman así porque los datos no están cargados al completo, si no que se van cargando solo los datos visibles y cuando unos se ocultan y se muestran otros el componente recicla los recursos de los datos ocultos.

Por último, tenemos la capa Modelo. Aquí se definen los datos con los que vamos a trabajar constantemente. En Kotlin existen un tipo de clases llamadas Data Class que facilitan mucho el trabajo y están pensadas para eso. Si a eso le sumamos la librería Room simplifica considerablemente la interacción con la base de datos SQLite en aplicaciones Android al permitir a los desarrolladores trabajar con objetos en lugar de consultas SQL directas.

Room proporciona un conjunto de anotaciones que se utilizan para definir la estructura de la base de datos y las relaciones entre las tablas, lo que facilita el mapeo de objetos de la aplicación a las tablas de la base de datos. Esto es lo que se llama un ORM (Object-Relational Mapping), ya mencionado anteriormente.

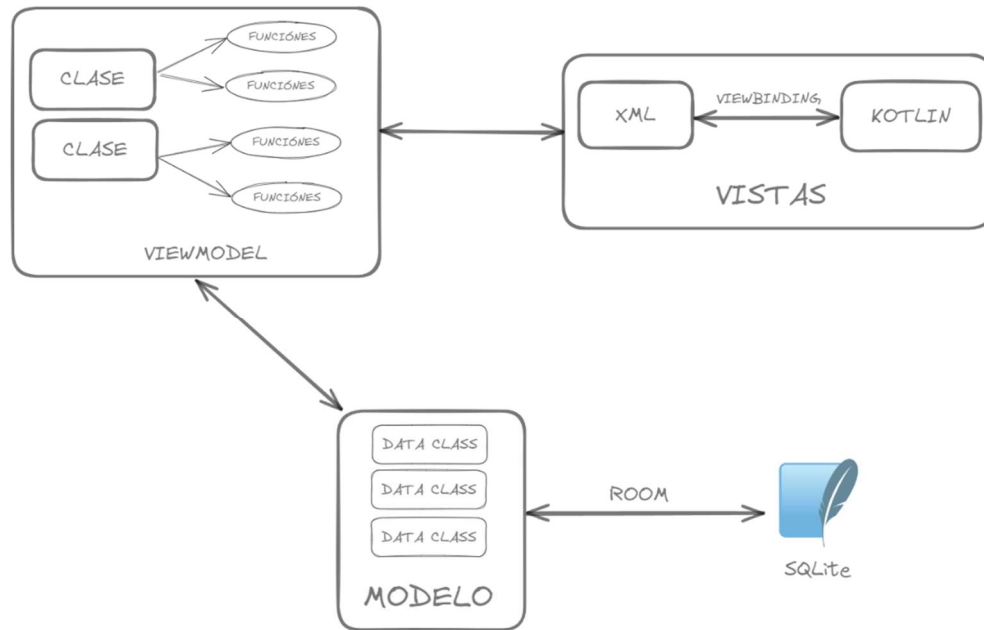


Figura 3.6: Arquitectura del sistema

### 3.3.4 DISEÑO DE PANTALLAS

#### 3.3.4.1 ESTRUCTURA PRINCIPAL

Se busca una estructura sencilla para buscar la sencillez y una aplicación intuitiva, por lo tanto, el minimalismo ha de estar presente.

En este caso, la estructura principal de las pantallas de la aplicación sigue este patrón.



Figura 3.7: Estructura principal de las pantallas

- Un header, en dónde se muestre una información general del contenido de la pantalla, lo primero que se ha de tener en cuenta para saber qué es lo que estás viendo. En algunos casos se puede interactuar con el header, por ejemplo, para cambiar de mes en la gestión de gastos e ingresos.
- El contenido, como su propio nombre indica el contenido más detallado de la pantalla. Aquí está la información real y vital de la pantalla. Se encuentra en el centro de la pantalla.
- La Navbar, una barra de navegación para poder moverse fácilmente dentro de la aplicación en pocos clics y siempre visible.

#### 3.3.4.2 GAMA DE COLORES

La gama de colores elegida es una parte importante dentro del diseño gráfico de la aplicación. Mucha información puede ser enfatizada o relacionada a ciertos colores y mostrar de una manera más clara la información y que no cause confusión.

Empezamos por la parte más visual de la aplicación, las gráficas. En este caso los ahorros, gastos no importantes y gastos importantes están representados por estos colores respectivamente.

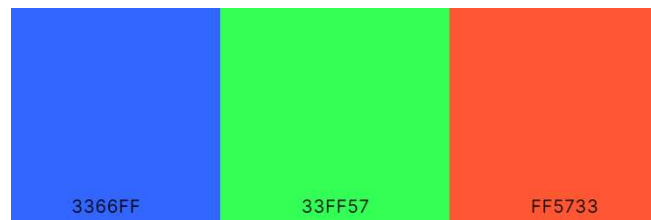


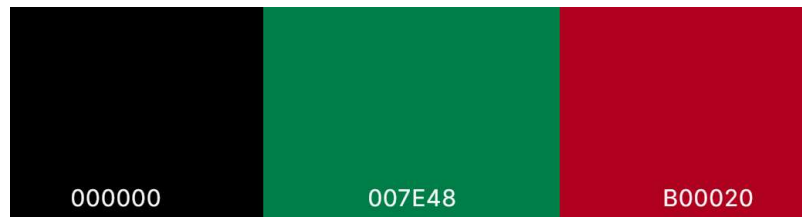
Figura 3.8: Gama de colores 1

Seguimos con el color principal y color secundario de la aplicación. El color principal es el blanco, que es el que se utiliza para el fondo de las pantallas, y el color secundario es un color llamado “Payne’s gray” que se utiliza para Header por ejemplo.



Figura 3.9: Gama de colores 2

Y por último tenemos la gama de colores utilizada para los botones. El primer color se utiliza para los botones que realizan una acción no crítica, como para abrir el calendario. El segundo color se usa para una acción crítica positiva, por ejemplo, guardar un gasto. Finalmente tenemos el color utilizado para realizar una acción crítica negativa, como cancelar un gasto o borrarlo. Estos colores pueden verse en la siguiente figura.



*Figura 3.10: Gama de colores 3*

### 3.3.5 DISEÑO DE LA BASE DE DATOS

En esta sección se expone toda la información referente a la base de datos. Es importante hablar de las tablas que la conforman, todos los campos de cada tabla y para qué sirve, y como se forma la base de datos.

La base de datos es SQLite. SQLite es un sistema de gestión de bases de datos relacional ligero y autónomo que se implementa como una biblioteca escrita en lenguaje C. A diferencia de los sistemas de gestión de bases de datos tradicionales, SQLite es un motor de base de datos compacto y autónomo que no requiere un servidor separado para funcionar, lo que lo hace ideal para aplicaciones que necesitan una base de datos local, como aplicaciones móviles, navegadores web, aplicaciones de escritorio, entre otras.

En esta base de datos se registran datos en dos tablas. La primera es “*Spends*” y “*Incomes*”, que corresponden a gastos e ingresos respectivamente. Por ahora solo se tienen esas tablas ya que para el MVP no es necesario más.

En la tabla *Spends*, se guarda toda la información de los gastos del usuario. A continuación, se lista y se explica todos los campos de la tabla.

- “**spend\_id**”, es el identificador único del gasto. Es la clave primaria, de tipo entero y es un valor auto incremental.
- “**amount**”, es el importe del gasto, un valor decimal.
- “**description**”, en este campo se persiste la información descriptiva del gasto, se podría resumir en el concepto. Es una cadena de texto.



- **“regularity”**, en este campo se guarda la regularidad del gasto. Es un valor enumerable, es decir, hay un grupo de valores posibles. En este caso los valores posibles son: “ONCE”, “WEEKLY”, “MONTHLY” y “YEARLY”.
- **“category”**, se guarda el dato de la categoría. Lo mismo que el anterior campo, es un enumerable con valores posibles de: “NECCESARY” y “UNNECCESARY”.
- **“dateAt”**, que es donde se guarda la fecha del gasto.
- **“checked”**, este es un campo que sirve para saber si ya se hay comprobado si es periódico y si se ha creado el gasto siguiente.

En la tabla Incomes, se guarda toda la información de los ingresos del usuario. A continuación, se lista y se explica todos los campos de la tabla.

- **“income\_id”**, es el identificador único del ingreso. Es la clave primaria, de tipo entero y es un valor auto incremental.
- **“amount”**, es el importe del ingreso, un valor decimal.
- **“description”**, en este campo se persiste la información descriptiva del ingreso, se podría resumir en el concepto. Es una cadena de texto.
- **“regularity”**, en este campo se guarda la regularidad del ingreso. Es un valor enumerable, es decir, hay un grupo de valores posibles. En este caso los valores posibles son: “ONCE”, “WEEKLY”, “MONTHLY” y “YEARLY”.
- **“dateAt”**, que es donde se guarda la fecha del ingreso.
- **“checked”**, este es un campo que sirve para saber si ya se hay comprobado si es periódico y si se ha creado el ingreso siguiente.

## 3.4 IMPLEMENTACIÓN

En esta sección se detallan las herramientas utilizadas para implantar lo diseñado en el punto anterior, así como aspectos a destacar en el código para mejor comprensión.

### 3.4.1 HERRAMIENTAS

En el desarrollo de aplicaciones Android, se emplean diversas herramientas y técnicas que contribuyen a su construcción y funcionalidad. Entre estas herramientas se encuentra Kotlin, un lenguaje de programación moderno y flexible que se ha convertido en el estándar para el desarrollo de aplicaciones en Android debido a su concisión y seguridad, entre otras ventajas.

Para la creación de interfaces de usuario (UI), se hace uso de archivos XML (eXtensible Markup Language). Estos archivos permiten describir la disposición y aspecto visual de los elementos que componen la interfaz de usuario.

Android, el sistema operativo en el que se ejecutan estas aplicaciones, proporciona un entorno robusto y variadas características, como el manejo de eventos, el ciclo de vida de las actividades, servicios y otras funcionalidades fundamentales para la construcción de aplicaciones móviles.

Android Studio, el entorno de desarrollo integrado (IDE) oficial de Android, ofrece herramientas específicas que facilitan la creación, depuración y prueba de aplicaciones. Además, integra emuladores y el SDK de Android, simplificando el proceso de desarrollo.

Para el almacenamiento de datos local, SQLite. Este motor de base de datos relacional, ligero y autónomo, se utiliza para gestionar datos en aplicaciones Android, ofreciendo un almacenamiento eficiente y flexible.

Room, una biblioteca de persistencia en Android, proporciona una capa de abstracción sobre SQLite. Simplifica la interacción con la base de datos a través de objetos y métodos más familiares, facilitando la gestión y manipulación de datos.

Además, las Corrutinas (Coroutines) en Kotlin son utilizadas para manejar la programación asíncrona y concurrente en Android. Permiten escribir código asíncrono de manera más clara y estructurada, evitando la complejidad de los callbacks anidados y mejorando la gestión de hilos.

Por último, cabe destacar el emulador de Android Studio en donde se prueba cada iteración completada en el proceso de desarrollo que permite comprobar que se ha desarrollado el producto correcto y que funciona correctamente.

### 3.4.2 ASPECTOS A DESTACAR

En esta sección se destacan las partes del código, funciones o archivos más importantes para entender el funcionamiento.

#### 3.4.2.1 DAO

Como primer aspecto a destacar, tenemos los archivos DAO. Un DAO (Data Access Object) es un patrón de diseño utilizado en el desarrollo de software para separar la lógica de acceso a datos de la lógica de negocio [20]. El propósito principal de un DAO es proporcionar una interfaz entre el código de la aplicación y la capa de almacenamiento de datos, ya sea una base de datos, un archivo plano u otro mecanismo de persistencia.

El DAO actúa como una capa intermedia que abstrae el acceso a la fuente de datos subyacente, lo que permite a la aplicación interactuar con los datos sin necesidad de conocer los

detalles específicos de cómo se almacenan o recuperan. Proporciona métodos y operaciones estandarizadas para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en la base de datos.

En la siguiente figura podemos observar cómo está implantado en este proyecto.

```
@Dao
interface IncomeDao {

    @Upsert
    fun upsertIncome(income: Income)

    @Delete
    fun deleteIncome(income: Income)

    @Query("SELECT * FROM Income")
    fun getAllIncomes(): MutableList<Income>

    @Query("SELECT * FROM Income where dateAt BETWEEN :from AND :at ORDER BY dateAt DESC")
    fun getIncomesByDate(from: String, at: String): MutableList<Income>

    @Query("SELECT * FROM Income where income_id = :id")
    fun getIncomeById(id: Int): Income

    @Query("SELECT strftime('%Y-%m', dateAt) AS month, SUM(amount) AS amount FROM Income GROUP BY month ORDER BY month")
    fun getIncomesByMonth(): MutableList<ChartInfo>
}
```

*Figura 3.11: Dao en este proyecto*

Podemos observar que Room en Android funciona mediante anotaciones. En este trozo de código diferenciamos cuatro anotaciones. La primera “@Dao” sirve para informar a Room de que esta interfaz va a funcionar como un DAO, y se coloca antes de la declaración de interface.

En segundo lugar, vemos la anotación “@Upsert” encima de la primera función. Esta anotación es muy útil tanto para insertar como para actualizar. En este caso, el objeto income que viene por parámetro se comprueba si su id ya existe en la base de datos, si ya existe actualiza el resto de sus campos, si no, se guarda como un registro nuevo. También existen las anotaciones “@Insert” y “@Update” por si queremos separar estas operaciones en dos funciones distintas, pero en este caso no tenía sentido.

“@Delete” como su propio nombre indica hace que la función borre el objeto que venga por parámetro. Existen modificadores de la anotación que sirven para controlar qué es lo que ocurre si ocurre algún error en el eliminado o si no existe ese registro.

Por último, está la anotación “@Query”, que sirve para ejecutar una consulta en SQL. Esta anotación nos viene muy bien para pedir a base de datos los datos con una serie de condiciones, para poder manejarlos con más facilidad en la aplicación.

### 3.4.2.2 ADAPTER

En el contexto de RecyclerView en Android, un Adapter es una pieza fundamental que actúa como puente entre los datos y la interfaz de usuario que se muestra en un RecyclerView.

El Adapter es responsable de manejar y suministrar los datos que se mostrarán en el RecyclerView. Estos datos pueden provenir de una lista, un array u otra estructura de datos. El Adapter organiza estos datos y los prepara para ser presentados en la interfaz de usuario.

El Adapter crea y mantiene las vistas individuales (llamadas "ViewHolder") que representan cada elemento de la lista dentro del RecyclerView. El método onCreateViewHolder del Adapter se utiliza para inflar o crear las vistas y vincularlas a los ViewHolders.

A través del método onBindViewHolder, el Adapter asocia los datos de cada elemento con las vistas correspondientes en los ViewHolders. Es aquí donde se actualizan las vistas con la información específica de cada elemento de la lista.

Además de gestionar la presentación de datos, el Adapter puede manejar eventos de clic, selección o interacciones con los elementos individuales en el RecyclerView, permitiendo así la interacción del usuario con la aplicación.

```
class IncomeAdapter(  
    private var incomeList: MutableList<Income>,  
    private val itemClickListener: (Income) -> Unit  
) : RecyclerView.Adapter<IncomeViewHolder>() {  
    ▲ Xosé Babío  
    override fun onCreateViewHolder(  
        parent: ViewGroup,  
        viewType: Int  
    ): IncomeViewHolder {  
        val inflater = LayoutInflater.from(parent.context)  
        val view = inflater.inflate(R.layout.item_income, parent, attachToRoot = false)  
        return IncomeViewHolder(view, itemClickListener)  
    }  
  
    ▲ Xosé Babío  
    override fun getItemCount(): Int = incomeList.size  
  
    ▲ Xosé Babío  
    override fun onBindViewHolder(holder: IncomeViewHolder, position: Int) {  
        val item = incomeList[position]  
        holder.render(item)  
    }  
}
```

Figura 3.12: El Adapter

### 3.4.2.3 VIEWHOLDER

Un ViewHolder es una clase que actúa como contenedor de las vistas individuales que componen cada elemento de un RecyclerView en Android.

En un contexto de RecyclerView, el ViewHolder optimiza la eficiencia de desplazamiento al almacenar referencias a las vistas de un elemento de la lista. Cuando se desplaza el RecyclerView y se necesitan nuevas vistas para mostrar nuevos elementos o para reciclar vistas que ya no están en pantalla, el ViewHolder se encarga de mantener la referencia a estas vistas, evitando así buscarlas repetidamente mediante `findViewById()`.

La clase ViewHolder se utiliza para crear y actualizar vistas cuando se infla una nueva vista (en el método `onCreateViewHolder()` del Adapter) o cuando se actualizan datos en una vista existente (en el método `onBindViewHolder()` del Adapter).

```
class IncomeViewHolder(
    view: View,
    private val itemClickListener: (Income) -> Unit
) : RecyclerView.ViewHolder(view) {

    private val binding = ItemIncomeBinding.bind(view)

    fun render(income: Income) {
        binding.tvIncomeAmount.text = income.amount.toString()
        binding.tvIncomeDescription.text = income.description
        binding.tvIncomeDate.text = income.dateAt
        itemView.setOnClickListener { it: View!
            itemClickListener(income)
        }
    }
}
```

Figura 3.13: El ViewHolder

### 3.4.2.4 CORRUTINAS

Las corrutinas en Android, una característica clave de Kotlin, son un mecanismo para facilitar la programación asíncrona y concurrente de manera más sencilla y estructurada. Están diseñadas para manejar tareas asíncronas sin bloquear el hilo principal de la interfaz de usuario (UI), lo que ayuda a evitar bloqueos y a mejorar la capacidad de respuesta de la aplicación.

Las corrutinas permiten escribir código asíncrono de manera secuencial, utilizando estructuras semejantes a funciones normales, pero marcadas con palabras clave específicas de suspensión, como “suspend”. Esto posibilita que el código asíncrono se comporte de manera más secuencial, facilitando su comprensión y mantenimiento. Además, las corrutinas ofrecen funciones como “async” y “await”, que permiten ejecutar múltiples tareas concurrentemente y esperar sus resultados de manera eficiente, lo que hace que las operaciones de red, bases de datos y otras tareas asíncronas sean más simples de implementar y gestionar en aplicaciones Android.

```
private fun checkRegularity() {  
    Log.i( tag: "RegularityChecker", msg: "checkRegularity")  
  
    // Utiliza una coroutine para realizar operaciones en segundo plano  
    GlobalScope.launch(Dispatchers.IO) { this: CoroutineScope  
        val spends = SmartFinanApplication.database.spendDao().getAllSpends()  
        val incomes = SmartFinanApplication.database.incomeDao().getAllIncomes()  
  
        // Regresa al hilo principal para ejecutar checkYearly  
        withContext(Dispatchers.Main) { this: CoroutineScope  
            checkYearly(spends, incomes)  
        }  
    }  
}
```

Figura 3.14: Ejemplo de Corrutina

## 3.5 PRUEBAS

En esta sección hablaremos sobre los distintos tipos de prueba que se ha llevado a cabo.

### 3.5.1 PRUEBAS UNITARIAS

Al final de cada iteración se hacen las pruebas unitarias. Las pruebas unitarias son una metodología de desarrollo de software en la que se verifican y validan unidades individuales de código, como funciones o métodos, para garantizar que funcionen correctamente de manera aislada e independiente [21]. Estas pruebas se centran en probar pequeñas partes del código, comprobando que cada unidad funcione como se espera y cumpla con sus especificaciones, ayudando a identificar errores o fallos de manera temprana en el proceso de desarrollo.

Existen dos tipos de pruebas, las de caja blanca y las de caja negra.

### 3.5.2 PRUEBAS DE CAJA BLANCA

Las pruebas de caja blanca, también conocidas como pruebas de "testing estructural", son pruebas de software que se realizan teniendo en cuenta el conocimiento interno de la estructura y el código fuente del programa. Estas pruebas evalúan cómo funcionan internamente las unidades o componentes del software, revisando la lógica del código, caminos de ejecución

y estructuras de control para validar su funcionamiento correcto. El objetivo es asegurar una cobertura exhaustiva del código y garantizar que se prueben todos los caminos posibles a través de este para detectar posibles errores y mejorar su calidad.

Para demostrar su funcionamiento utilizamos un trozo de nuestro código.

```
private fun calc(): String {
1  val stringBuilder = StringBuilder()

    val importantSpendsPercentage = (totalImportantSpends / totalIncomes) * 100
    val notImportantSpendsPercentage = (totalNotImportantSpends / totalIncomes) * 100
    val monthImportantSpendsPercentage = (totalMonthImportantSpends / totalMonthIncomes) * 100
    val monthNotImportantSpendsPercentage = (totalMonthNotImportantSpends / totalMonthIncomes) * 100
    val monthSavingPercentage = ((totalMonthIncomes - totalMonthSpends) / totalMonthIncomes) * 100

2  if (notImportantSpendsPercentage > 30.0) {
3      stringBuilder.append("Tienes demasiados gastos innecesarios. Este año te pasas en ${(totalNotImportantSpends - (totalIncomes * 0.3))}€.")
    }
4  if (importantSpendsPercentage > 50.0) {
5      stringBuilder.append("Tu nivel de vida es demasiado alto. Superas ${(totalImportantSpends - (totalIncomes * 0.5))}€ en el estándar de ahorro recomendado durante este año.")
    }
6  if (monthNotImportantSpendsPercentage > 30.0) {
7      stringBuilder.append("Este mes has superado en ${(totalMonthNotImportantSpends - (totalMonthIncomes * 0.3))}€ los gastos innecesarios recomendados.")
    }
8  if (monthImportantSpendsPercentage > 50.0) {
9      stringBuilder.append("Tus gastos importantes este mes superan en ${(totalMonthImportantSpends - (totalMonthIncomes * 0.5))}€ a los recomendados, intenta reducirlos.")
    }
10 if (monthSavingPercentage < 20.0) {
11     stringBuilder.append("Este mes has ahorrado ${(monthSavingPercentage)}% de tus ingresos, lo cual es menos del 20% recomendado. Intenta aumentar tus ahorros.")
    }
12 val advise = stringBuilder.toString()

13 if (advise.isEmpty()) {
14     return "No tenemos ninguna recomendación para ti."
    }
15 return advise
}
```

Figura 3.16: Función calc() a probar

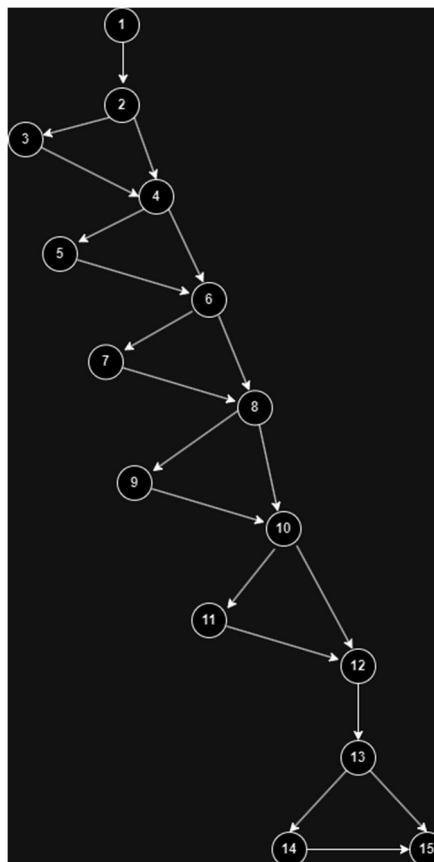


Figura 3.15: Grafo del código anterior

Obtenemos como resultado 15 nodos, 19 aristas y 5 regiones.

Utilizando las fórmulas de la complejidad ciclomática [22]:

- $V(G) = \text{aristas} - \text{nodos} + 2 = 19 - 15 + 2 = 6$
- $V(G) = \text{regiones cerradas} + 1 = 5 + 1 = 6$
- $V(G) = \text{nodos de precisión simple} = 6$

Siendo nodos de precisión simple aquellos de los que emergen 2 aristas.

Esto significa que existen 6 caminos diferentes para recorrer todo el código.

Tabla 3.39: Caminos independientes del código

RECORRIDO	ENTRADA	RESULTADO
1-2-4-6-8-10-12-13-14-15	$\text{notImportantSpendsPercentage} = 20$ $\text{importantSpendsPercentage} = 42$ $\text{monthNotImportantSpendsPercentage} = 25$ $\text{monthImportantSpendsPercentage} = 45$ $\text{monthSavingPercentage} = 30$	Correcto
1-2-3-4-6-8-10-12-13-15	$\text{notImportantSpendsPercentage} = 33$ $\text{importantSpendsPercentage} = 42$ $\text{monthNotImportantSpendsPercentage} = 25$ $\text{monthImportantSpendsPercentage} = 45$ $\text{monthSavingPercentage} = 30$	Correcto
1-2-4-5-6-8-10-12-13-15	$\text{notImportantSpendsPercentage} = 20$ $\text{importantSpendsPercentage} = 55$ $\text{monthNotImportantSpendsPercentage} = 25$ $\text{monthImportantSpendsPercentage} = 45$ $\text{monthSavingPercentage} = 30$	Correcto
1-2-4-6-7-8-10-12-13-15	$\text{notImportantSpendsPercentage} = 20$ $\text{importantSpendsPercentage} = 42$ $\text{monthNotImportantSpendsPercentage} = 32$ $\text{monthImportantSpendsPercentage} = 38$ $\text{monthSavingPercentage} = 30$	Correcto
1-2-4-6-8-9-10-12-13-15	$\text{notImportantSpendsPercentage} = 20$ $\text{importantSpendsPercentage} = 42$ $\text{monthNotImportantSpendsPercentage} = 18$ $\text{monthImportantSpendsPercentage} = 52$ $\text{monthSavingPercentage} = 30$	Correcto
1-2-4-6-8-10-11-12-13-15	$\text{notImportantSpendsPercentage} = 20$ $\text{importantSpendsPercentage} = 42$ $\text{monthNotImportantSpendsPercentage} = 37$ $\text{monthImportantSpendsPercentage} = 45$ $\text{monthSavingPercentage} = 18$	Correcto



Mediante la tabla anterior se puede comprobar que el resultado de las pruebas unitarias para cada uno de los caminos independientes es satisfactorio en todos los casos.

### 3.5.3 PRUEBAS DE CAJA NEGRA

Las pruebas de caja negra son un enfoque de pruebas de software en el que se evalúa la funcionalidad de un sistema sin considerar su estructura interna o el conocimiento del código fuente. Se centran únicamente en las entradas y salidas del sistema, verificando si el software produce resultados correctos en respuesta a diferentes conjuntos de datos de entrada, sin tener en cuenta cómo se procesan o manejan esos datos dentro del programa. Este tipo de pruebas se basa en especificaciones externas y se realiza sin información detallada sobre el diseño interno del software, simulando las acciones de un usuario externo.

A continuación, se muestra una lista de pruebas de caja negra pertenecientes a los casos de uso. Estas pruebas se definen de forma resumida ya que sería mucha cantidad de información la que detallar.

*Tabla 3.40: Pruebas de caja negra*

IDENTIFICADOR	NOMBRE	RESULTADO ESPERADO	RESULTADO OBTENIDO
UC-01	Crear gasto	Se crea el gasto en la tabla de gastos	Correcto
UC-02	Actualizar gasto	Se actualiza el gasto en la tabla de gastos	Correcto
UC-03	Borrar gasto	Se borra el gasto de la tabla de gastos con el identificador elegido	Correcto
UC-04	Crear ingreso	Se crea un nuevo ingreso en la tabla de ingresos	Correcto
UC-05	Actualizar ingreso	Se actualiza el ingreso elegido	Correcto
UC-06	Borrar ingreso	Se borra el ingreso con identificador elegido en la tabla de ingresos	Correcto
UC-07	Consultar gastos	Se consultan los gastos	Correcto
UC-08	Consultar ingresos	Se consultan los ingresos	Correcto
UC-09	Consultar gráficas	Se consulta las gráficas con la información correcta	Correcto
UC-10	Consultar recomendación	Se consulta la recomendación del sistema	Correcto
UC-11	Generación automática de gasto periódico	Se crea un gasto que se registra automáticamente	Correcto
UC-12	Generación automática de ingreso periódico	Se crea un ingreso que se registra automáticamente	Correcto

---

Las pruebas de caja negra para cada uno de los casos de uso son correctas, por lo que se puede deducir que el producto es el correcto, y funciona de manera correcta.

## 4. NORMATIVA VIGENTE

Las aplicaciones deben cumplir con las leyes de protección de datos aplicables, como el Reglamento General de Protección de Datos (RGPD) de la Unión Europea y leyes similares en otras regiones [23].

Debe garantizar la protección de los datos personales y financieros de sus usuarios y obtener su consentimiento explícito para la recopilación y el procesamiento de datos. El Reglamento General de Protección de Datos (RGPD) es un reglamento de la Unión Europea (UE) que establece reglas y estándares para la protección y el procesamiento de datos personales de personas dentro de la Unión Europea y el Espacio Económico Europeo (EEE). Entró en vigor el 25 de mayo de 2018 y tiene como objetivo reforzar y unificar la protección de datos para todos los ciudadanos de la UE y las empresas que operan en este sector.

RGPD introduce una serie de principios y requisitos importantes que las organizaciones que recopilan, almacenan o procesan datos personales deben seguir.

**Consentimiento explícito:** Las organizaciones deben obtener el consentimiento claro y específico de las personas para recopilar y procesar datos personales. El consentimiento debe darse de forma libre, informada e inequívoca mediante una acción afirmativa clara.

**Derechos individuales:** RGPD otorga a las personas una serie de derechos, incluido el derecho a acceder, rectificar, borrar y restringir el procesamiento de sus datos personales. También tiene derecho a la portabilidad de sus datos y derecho a oponerse al tratamiento de sus datos en determinadas circunstancias.

**Obligación de transparencia:** Las organizaciones deben proporcionar información clara y fácilmente accesible sobre cómo se recopilan, almacenan y utilizan los datos personales. Debe proporcionar una política de privacidad y términos de uso que sean transparentes y fáciles de entender.

**Seguridad y protección de datos:** Las organizaciones deben tomar medidas técnicas y organizativas adecuadas para proteger los datos personales del acceso no autorizado, la divulgación, la pérdida o la destrucción.

**Notificación de violación de datos:** En caso de una violación de seguridad que pueda afectar los derechos y libertades de las personas, las organizaciones deben notificar a las autoridades pertinentes y a las personas afectadas dentro de un cierto período de tiempo.

Responsabilidad y cumplimiento: Las organizaciones son responsables de cumplir con los principios del RGPD y deben poder demostrar dicho cumplimiento a través de la documentación adecuada, evaluaciones de impacto de la protección de datos y auditorías internas.

RGPD se aplica a todas las empresas y organizaciones, independientemente de su ubicación física, cuando recopilan o procesan datos personales de personas en la UE. Su principal objetivo es garantizar una mejor gestión y protección de los datos personales y promover una cultura de protección de datos más sólida y transparente. Las empresas que no cumplan con el RGPD pueden enfrentarse a fuertes multas. Por lo tanto, es importante que las empresas garanticen el cumplimiento de esta normativa.

En este caso, la aplicación no recopila datos del usuario porque se almacenan localmente en el propio teléfono del usuario. Esta es una información muy clara, pero debería tenerse en cuenta en futuras mejoras de la aplicación en cuanto cambie el tipo de almacenamiento de datos.

## 5. PRODUCTO FINAL

En este capítulo vamos a adentrarnos en cómo es el producto final, adjuntar capturas de que es lo que se ve, y cómo interactuar con el sistema para que sea funcional.

### 5.1 PANTALLA PRINCIPAL

En primer lugar, se puede ver como el logo de la aplicación está presente en el header, y cómo están presentados los botones de navegación en la navbar.

Es esta sección se puede ver cómo con unos datos simulados cumplimos con el cometido de la aplicación, que nos haga recomendaciones para tener unos mejores hábitos financieros y no correr ningún riesgo en este ámbito.



*Figura 5.1: Pantalla principal*

En esta pantalla no hay nada más con lo que podamos interactuar, sólo con los botones de la navbar inferior. Se puede ver la palabra “Inicio” debajo de la opción que tenemos seleccionada, pero del resto solo vemos el icono.

## 5.2 GESTIÓN DE GASTOS

En esta sección vemos como en el header se nos muestra el gasto producido este mes, el propio mes y unos botones con los que podemos navegar entre los meses.

En el contenido de la pantalla vemos una tabla con los gastos registrados, mostrándonos el importe del gasto, la descripción, la categoría coloreada y la fecha.

Observamos un botón flotante en la esquina inferior derecha del contenido, con ese botón podremos registrar nuevos gastos.

Finalmente se encuentra la navbar.

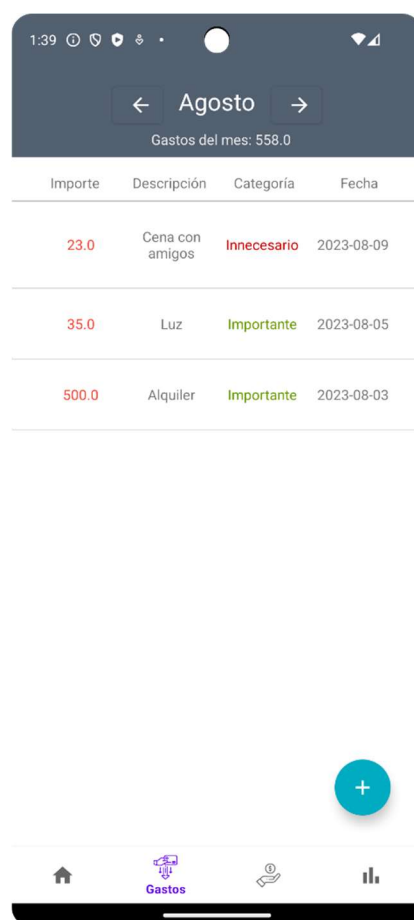
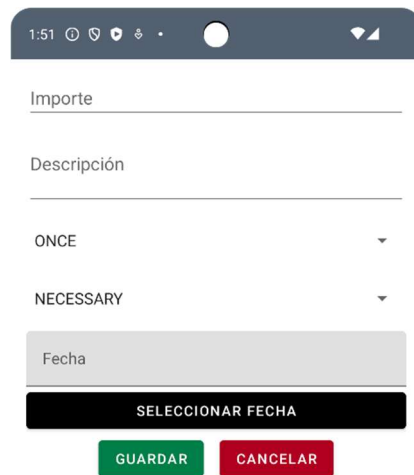


Figura 5.2: Gestión de gastos

Si pulsamos el botón flotante, llegamos a un formulario de registro en donde nos deja elegir el importe del gasto, el concepto de este, la regularidad y la categoría.



The image shows a mobile application interface for recording an expense. At the top, there is a status bar with the time 1:51 and various icons. Below this, the form consists of several input fields: 'Importe' (Amount), 'Descripción' (Description), a dropdown menu currently showing 'ONCE', another dropdown menu currently showing 'NECESSARY', and a date selection field labeled 'Fecha'. Below the date field is a black button with white text that says 'SELECCIONAR FECHA'. At the bottom of the form are two buttons: a green one labeled 'GUARDAR' (Save) and a red one labeled 'CANCELAR' (Cancel).

*Figura 5.3: Formulario del gasto*

Además, tenemos tres botones, “Seleccionar fecha”, “Guardar” y “Cancelar. Con el primer botón se nos abrirá un calendario en dónde podremos seleccionar la fecha asignada al gasto. Una vez elegida la fecha se nos escribirá en la caja de texto no editable Fecha.

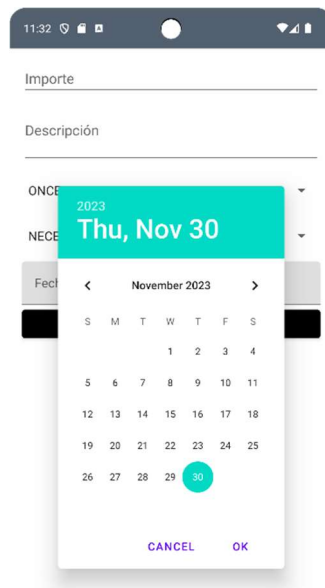


Figura 5.4: Elección en calendario

Una vez guardado el gasto, si se clicla sobre él, se nos abre una pantalla similar en dónde podemos editar los datos o simplemente eliminar el gasto. En la siguiente figura podemos ver cómo están escritos los datos en cada campo, y los botones disponibles.

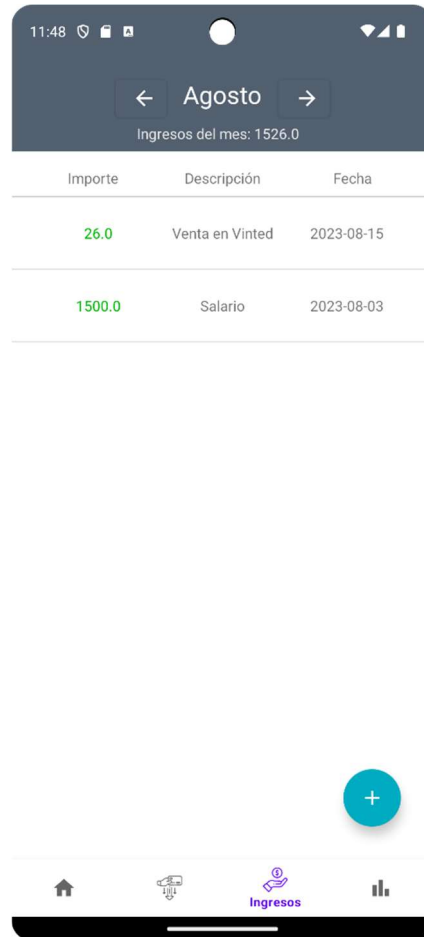


Figura 5.5: Actualizar o eliminar gasto



## 5.3 GESTIÓN DE INGRESOS

La gestión de ingresos funciona prácticamente igual que la gestión de los gastos. En primer lugar, tenemos la lista de los ingresos que hemos registrado hasta el momento.

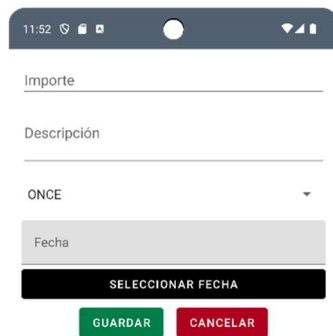


Importe	Descripción	Fecha
26.0	Venta en Vinted	2023-08-15
1500.0	Salario	2023-08-03

Figura 5.6: Gestión de ingresos

Como vemos el importe ahora esta coloreado en verde, por lo que damos a entender que es un valor positivo. Los ingresos al no tener categoría la tabla está dividida en tres, el importe, la descripción y la fecha. Seguimos teniendo el header con la suma de los ingresos de este mes, el mes que estamos mirando y los botones de navegación.

Para añadir el ingreso funciona exactamente igual que los gastos.



11:52

Importe

Descripción

ONCE

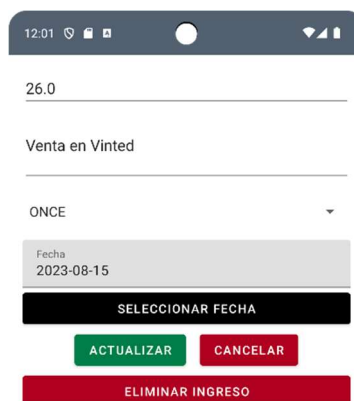
Fecha

SELECCIONAR FECHA

GUARDAR CANCELAR

*Figura 5.8: Añadir ingreso*

Al igual que para editarlo o borrarlo. La diferencia más destacable entre los gastos y los ingresos es que en los ingresos no existe la categoría por lo tanto no está ese campo.



12:01

26.0

Venta en Vinted

ONCE

Fecha  
2023-08-15

SELECCIONAR FECHA

ACTUALIZAR CANCELAR

ELIMINAR INGRESO

*Figura 5.7: Editar o eliminar ingreso*

## 5.5 GRÁFICA

En esta pantalla podemos ver de manera eficaz y rápida todos los gastos y ahorro de cada mes del año. Vemos el año que estamos consultando en el header, y la gráfica en el contenido.

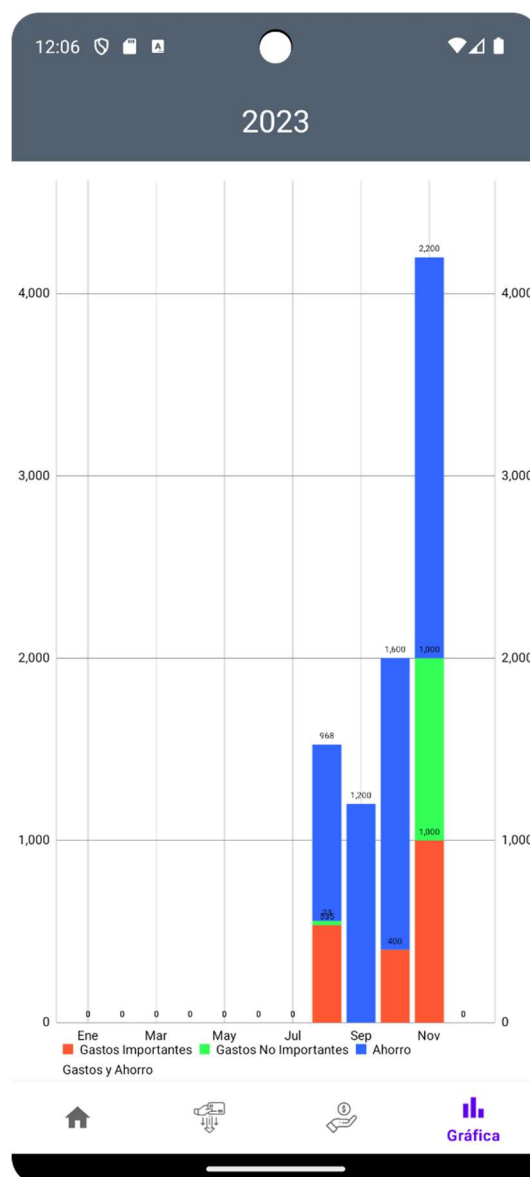


Figura 5.9: Gráfica

Además de eso podemos interactuar con la gráfica ampliando y moviéndonos vertical y horizontalmente como se muestra en la siguiente figura.

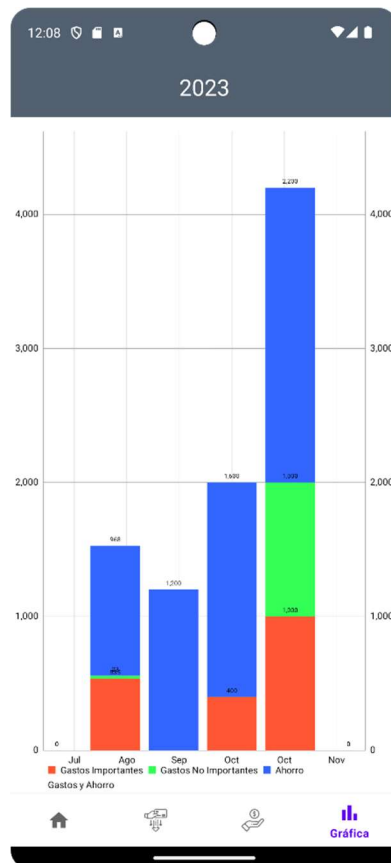


Figura 5.10: Gráfica ampliada

## 6. EVALUACIÓN

Tras haber explicado el producto resultante, es momento de hacer una pequeña evaluación sobre el resultado. Es importante recordar los objetivos del proyecto inicialmente e ir evaluando si la solución resuelve esos problemas planteados.

En primer lugar, vamos a evaluar la sencillez de la aplicación. Es evidente que es una aplicación sencilla, con pocas funcionalidades que cumplen entre ellas el objetivo de darle una salud financiera al usuario. No se tarda mucho en ir registrando los gastos y los ingresos que se van teniendo, sobre todo porque la función de periodicidad nos ahorra mucho trabajo. Muchos de nuestros gastos son semanal, mensual o anualmente los mismos, y en el hipotético caso de que nos cambien la cuota o dejemos de hacerlos, siempre se puede editar esos gastos. Lo mismo pasa con los ingresos, normalmente tendremos nuestro salario registrado mensualmente y podremos ir añadiendo ciertas ventas que podamos hacer, horas extras, trabajos puntuales, etc...

Nos lleva poco trabajo registrar los datos, y poco trabajo consultar los resultados, lo que hace que sea una aplicación a la que se le puede dedicar poco tiempo y cumple su función al groso de la población.

En los objetivos también planteábamos una funcionalidad de consultar nuestros datos de forma gráfica para una evaluación a primera vista. Con nuestra gráfica de barras podemos considerarlo como un objetivo completado. La diferencia cromática entre los tipos de gasto o el ahorro y la disposición de los datos esa gráfica hace que se pueda identificar rápidamente errores graves que cometemos ciertos meses del año y poder corregirlos posteriormente.

Por último, evaluamos lo intuitiva que es la aplicación. En la navbar podemos ver como mediante iconos podemos identificar rápidamente para que sirve cada pantalla. Además, que en las pantallas en donde podemos interactuar con el sistema nos deja claro que el botón flotante con un “+” nos indica que podemos añadir un registro nuevo, y si queremos interactuar con un registro ya existente tocarlo en la tabla sería nuestro primer impulso.

## 7. CONCLUSIONES

### 7.1 APORTACIONES REALIZADAS

La idea del proyecto nace tras mi propia incapacidad de controlarme mis propios gastos. En mi situación personal en dónde empezaba a trabajar y recibir ingresos, era incapaz de ser consciente de todos los gastos innecesarios que iba teniendo, y por lo tanto tener que seguir pidiendo dinero a mis padres.

Tras informarme sobre finanzas personales encontré el principio 50-30-20 que fui llevando durante meses con unos grandes resultados. Hasta el momento utilizaba hojas Excel para controlarme estos gastos, con la ayuda de la aplicación de mi banco. Esto no era cómodo ya que muchos gastos eran en efectivo, y el banco no los registraba, o las suscripciones semanales como Netflix, HBO o Amazon Prime, las tenía que registrar a mano en esas hojas Excel.

Ahí fue donde me di cuenta de que disponer de una aplicación sencilla, con ciertas funcionalidades de periodicidad y un algoritmo que me calculase los porcentajes de gasto, y que la llevase en el teléfono, hoy en día el objeto que llevamos a todos los sitios sería una gran ayuda para mí, y probablemente para mucha más gente que solo busca tener unas cuentas saneadas y nada de inversiones ni conceptos más avanzados.

### 7.2 PROBLEMAS ENCONTRADOS

El problema principal encontrado fue tener que aprender todo lo que tenía que ver con las tecnologías móviles. En primer lugar, no sabía nada de kotlin antes de empezar este proyecto, aunque no ha sido muy complicado aprenderlo ya que mi lenguaje de programación principal es Java y comparten muchas cosas.

Además del lenguaje, aprender todo lo que tiene que ver con patrones de diseño en las tecnologías móviles ha sido más complicado ya que hasta el momento que pasan las iteraciones y va creciendo el programa no te das cuenta de la importancia que tienen los patrones de diseño en la programación. Al final son soluciones encontradas a ciertos problemas muy recurrentes que ya se han encontrado otros programadores y fueron documentando esas soluciones muy efectivas.

Otro problema encontrado fue aprender a utilizar las corrutinas. Al estar acostumbrado a programar para ordenadores o servidores no te das cuenta de que en un teléfono móvil es más importante gestionar bien los recursos ya son más limitados. Además de que el uso de un teléfono es distinto al de un ordenador. Normalmente en un teléfono saltamos de aplicación en

aplicación mucho más frecuentemente, por lo tanto, hay que tener en cuenta el ciclo de vida de esas aplicaciones, lo que pasa cuando la cierras, cuando queda en segundo plano, cuando la abres por primera vez.

### 7.3 POSIBLES MEJORAS

La mejora más clara que tengo en mente es la posibilidad de crearte una cuenta para todo lo que tenga que ver con compartir gastos, crear copias de seguridad, tener múltiples cuentas para gestionar entornos distintos.

Ofrecer la capacidad de sincronizar datos entre dispositivos y realizar copias de seguridad automáticas en la nube para garantizar la seguridad de la información financiera. Permitir a los usuarios administrar gastos e ingresos en múltiples cuentas o categorizar transacciones con mayor detalle para una mejor organización.

Integrar notificaciones cuando se registre un gasto o ingreso automáticamente, así el usuario podrá comprobar que son correctos o editarlos al momento sin problema.

Permitir a los usuarios establecer objetivos de ahorro o presupuestos mensuales y recibir alertas cuando estén cerca de superar esos límites. Esto puede ayudar a mantener un control más estricto de los gastos.

Que los usuarios puedan crear sus propias categorías, así como subcategorías y disponer de más información sobre en dónde gastan más su dinero.

Y por último distintas estrategias financieras, la que está implantada no es la única, es la más sencilla, pero se podría implantar otra y que sea el usuario el que elija como quiere llevar sus finanzas.

### 7.4 OPINIÓN PERSONAL

El hecho de desarrollar una aplicación totalmente desde cero, llevando a cabo todo el estudio, análisis y desarrollo creo que te hace madurar mucho como profesional. Personalmente ya tenía experiencia en el ámbito del desarrollo, pero no tenía mucha experiencia en las demás fases.

Además, me ha hecho adentrarme en el mundo de las tecnologías móviles, que es un mercado muy interesante ya que es muy fácil ser freelance, y subir tus propias aplicaciones a Google Play o App Store y poder vivir de ello, o compaginarlo con trabajar para terceros.

No descarto seguir con este proyecto hasta el punto de publicarlo con cierta estrategia de marketing e intentar hacer de un producto que me produzca ingresos.

Por último, todo el proceso de ir documentando todas las tareas y fases ha sido un gran aprendizaje para mí, y me ha hecho darme cuenta de que no sólo es importante hacer las cosas, sino que es muy importante saber documentarlas y explicarlas al resto. En resumen, si tuviese que decir en una palabra lo que ha sido la realización de este TFG para mí, sería aprendizaje.



## LISTA DE REFERENCIAS

[1] Gutierrez, D. (2011). Métodos de desarrollo de software. *Caracas: Universidad de los Andes*. [Online]. Available:

[https://gc.scalahed.com/recursos/files/r161r/w24792w/ISSO/METODO\\_Andes.pdf](https://gc.scalahed.com/recursos/files/r161r/w24792w/ISSO/METODO_Andes.pdf)

[2] Pérez Ibarra, S. G., Quispe, J. R., Mullicundo, F. F., & Lamas, D. A. (2021). Herramientas y tecnologías para el desarrollo web desde el FrontEnd al BackEnd. In *XXIII Workshop de Investigadores en Ciencias de la Computación (WICC 2021, Chilecito, La Rioja)*. [Online]. Available:

<http://sedici.unlp.edu.ar/handle/10915/120476>

[3] Moskala, M., & Wojda, I. (2017). *Android Development with Kotlin*. Packt Publishing Ltd. [Online]. Available:

<https://books.google.es/books?hl=es&lr=&id=OzkzEAAAQBAJ&oi=fnd&pg=PT16&dq=kotlin&ots=F2UFnttdQf&sig=6ShyDbmlZ5VA4D6PL4OfA0iiU0Y>

[4] Gironés, J. T. (2019). *El gran libro de Android*. Alpha Editorial. [Online]. Available:

[https://books.google.es/books?hl=es&lr=&id=h\\_R5EAAAQBAJ&oi=fnd&pg=PR7&dq=android&ots=Wn6bTHYSOT&sig=tJ0e9Vf3oki\\_BjRkZuxf3lgvoPg](https://books.google.es/books?hl=es&lr=&id=h_R5EAAAQBAJ&oi=fnd&pg=PR7&dq=android&ots=Wn6bTHYSOT&sig=tJ0e9Vf3oki_BjRkZuxf3lgvoPg)

[5] Báez, M., Borrego, Á., Cordero, J., Cruz, L., González, M., Hernández, F., ... & Zapata, Á. (2019). Introducción a android. [Online]. Available:

<https://dspace.itsjapon.edu.ec/jspui/bitstream/123456789/434/1/introduccion-android.pdf>

[6] Mawlood-Yunis, A. R. (2022). Android SQLite, Firebase, and Room Databases. In *Android for Java Programmers* (pp. 475-530). Cham: Springer International Publishing. [Online]. Available:

[https://link.springer.com/chapter/10.1007/978-3-030-87459-9\\_11](https://link.springer.com/chapter/10.1007/978-3-030-87459-9_11)

[7] Blischak, J. D., Davenport, E. R., & Wilson, G. (2016). A quick introduction to version control with Git and GitHub. *PLoS computational biology*, 12(1), e1004668. [Online]. Available:

<https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004668&ref=https://githubhelp.com>

[8] Jorge Jimenez (2023), La regla 50/20/30 que te ayudará a ahorrar en 2024. *Fintonic Blog*. [Online]. Available:

<https://www.fintonic.com/blog/la-regla-502030/>

[9] Lenarduzzi, V., & Taibi, D. (2016, August). MVP explained: A systematic mapping study on the definitions of minimal viable product. In *2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (pp. 112-119). IEEE. [Online]. Available:

<https://ieeexplore.ieee.org/abstract/document/7592786/>

[10] Lou, T. (2016). A comparison of Android native app architecture MVC, MVP and MVVM. *Eindhoven University of Technology*. [Online]. Available:

[https://research.tue.nl/files/48628529/Lou\\_2016.pdf](https://research.tue.nl/files/48628529/Lou_2016.pdf)

[11] Chiu, C. C., Alberto, I., & Carbajal, T. (2015). Las pruebas en el desarrollo de software. *Universidad Nacional Autónoma de México*. [Online]. Available:

[https://www.academia.edu/download/55180608/Las\\_pruebas\\_en\\_el\\_desarrollo\\_de\\_software.pdf](https://www.academia.edu/download/55180608/Las_pruebas_en_el_desarrollo_de_software.pdf)

[12] Wilson, J. M. (2003). Gantt charts: A centenary appreciation. *European Journal of Operational Research*, 149(2), 430-437. [Online]. Available:

<https://www.sciencedirect.com/science/article/pii/S0377221702007695>

[13] Indeed. (s.f.). Salarios en Indeed. [Online]. Available:

<https://es.indeed.com/career/salaries>

[14] Cardozzo, D. R. (2016). *Desarrollo de software: requisitos, estimaciones y análisis*. IT Campus Academy. [Online]. Available:

[https://books.google.es/books?hl=es&lr=&id=tBaYCwAAQBAJ&oi=fnd&pg=PA1&dq=requisitos+de+software&ots=uHpZxk8KI0&sig=fqnUyluccooZaw\\_YekJt8C9bwjl](https://books.google.es/books?hl=es&lr=&id=tBaYCwAAQBAJ&oi=fnd&pg=PA1&dq=requisitos+de+software&ots=uHpZxk8KI0&sig=fqnUyluccooZaw_YekJt8C9bwjl)

[15] Elizarov, R., Belyaev, M., Akhin, M., & Usmanov, I. (2021, October). Kotlin coroutines: design and implementation. In *Proceedings of the 2021 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software* (pp. 68-84). [Online]. Available:

<https://dl.acm.org/doi/abs/10.1145/3486607.3486751>

[16] Jahoda, P. MPAndroidChart. 2015. URI: <https://github.com/PhilJay/MPAndroidChart>. [Online]. Available:

<https://github.com/PhilJay/MPAndroidChart>

[17] Gutierrez, D. (2011). Casos de uso Diagramas de Casos de Uso. *Gutierrez, Demián*, 1, 45. [Online]. Available:

[https://www.academia.edu/download/38141185/UML\\_clase\\_02\\_UML\\_casos\\_de\\_uso.pdf](https://www.academia.edu/download/38141185/UML_clase_02_UML_casos_de_uso.pdf)

[18] González, Y. D., & Romero, Y. F. (2012). Patrón Modelo-Vista-Controlador. *Telemática*, 11(1), 47-57. [Online]. Available:

<https://revistatelematica.cujae.edu.cu/index.php/tele/article/view/15>

[19] Torres, A., Galante, R., Pimenta, M. S., & Martins, A. J. B. (2017). Twenty years of object-relational mapping: A survey on patterns, solutions, and their implications on application design. *information and software technology*, 82, 1-18. [Online]. Available:

<https://www.sciencedirect.com/science/article/pii/S0950584916301859>

[20] Berger, M. (2005). Data Access Object Pattern. *línea*. Available: <https://pdfs.semanticscholar.org/1b11/5f5253567964283e16081488daf6af4f5351.pdf>. [Último acceso: 30 03 2019]. [Online]. Available:

<http://max.berger.name.s3.amazonaws.com/research/silenus/print/dao.pdf>

[21] Bedoya Alzate, E. (2021). Implementación de pruebas unitarias. [Online]. Available:

<https://bibliotecadigital.udea.edu.co/handle/10495/20225>

[22] Cardacci, D. G. (2016). *Refactorización de código y consideraciones sobre la complejidad ciclométrica* (No. 592). Serie Documentos de Trabajo. [Online]. Available:

<https://www.econstor.eu/bitstream/10419/163250/1/867089466.pdf>

[23] “BOE.es - Documento consolidado DOUE-L-2016-80807” [Online]. Available:

<https://www.boe.es/doue/2016/119/L00001-00088.pdf>



## ANEXOS

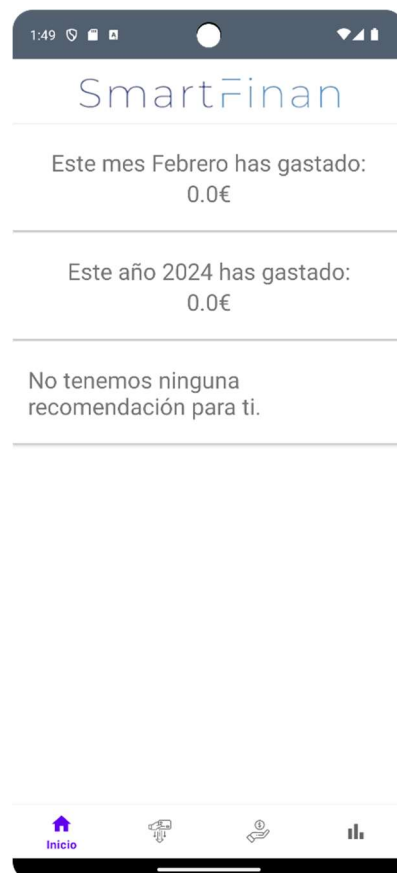
### MANUAL DE USUARIO

En este anexo se recoge un manual de uso de la aplicación, con imágenes, para el correcto uso, explotando al máximo todas las funcionalidades.

#### PANTALLA PRINCIPAL

La pantalla principal está diseñada de manera informativa. En la primera sección de la pantalla encontramos el gasto del mes actual. En esta sección se sumarán todos los gastos asignados al mes actual, independientemente de su categoría. En la segunda sección, justo debajo de la anterior, tenemos los gastos del año actual. Funciona de manera análoga que la anterior, se suman todos los gastos del año actual, sean importantes o no.

Por último, tenemos la sección de la recomendación. Aquí el algoritmo nos recomendará bajar el gasto de ciertas categorías en función de los ingresos que hayamos registrado. Siempre la recomendación prioritaria será la de desajustes anuales, y después las mensuales.



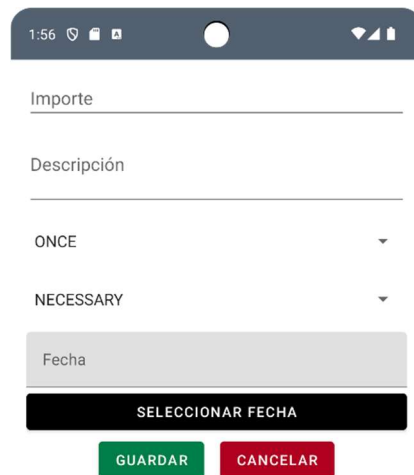
Esto es lo que nos encontramos en el momento de abrir la aplicación por primera vez.

## GESTIÓN DE GASTOS

En la pantalla de gestión de gastos encontramos todos los gastos listados de un mes específico. Por defecto se abrirá el mes actual.



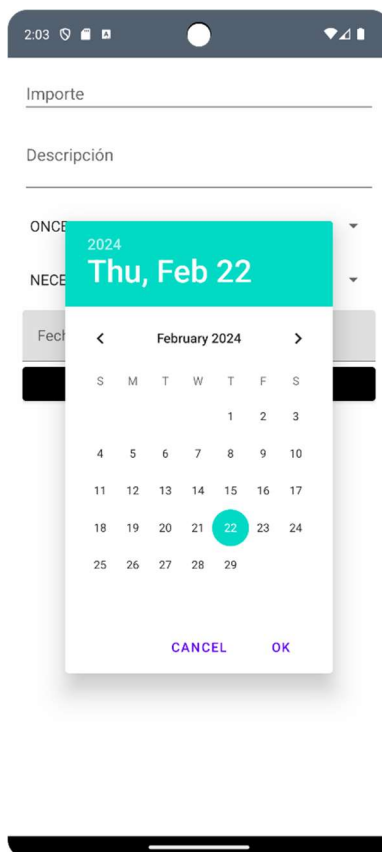
Encontramos al lado del nombre del mes unos botones para poder navegar entre los meses. Además de eso tenemos debajo del mes un pequeño resumen de los gastos del mes. Las celdas para listar los gastos son el importe, descripción, categoría y fecha. Para añadir un gasto debemos pulsar el botón flotante de abajo a la derecha con un “+”.



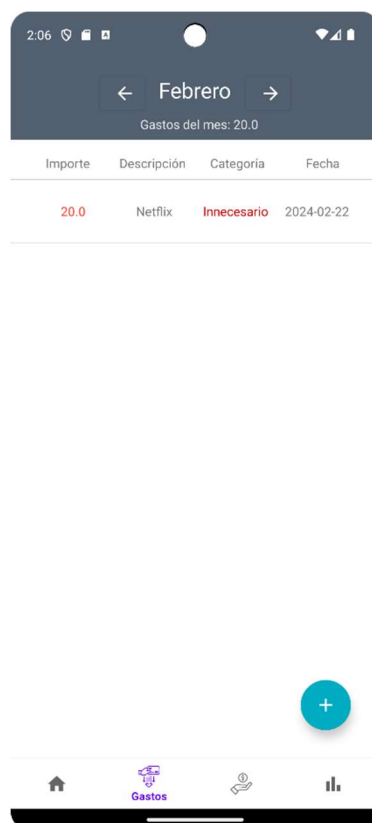
The screenshot shows a mobile application interface for adding an expense. At the top, there is a status bar with the time 1:56 and various icons. Below the status bar, the form consists of several input fields: 'Importe' (Amount), 'Descripción' (Description), a dropdown menu currently showing 'ONCE', another dropdown menu currently showing 'NECESSARY', and a date picker labeled 'Fecha'. Below the date picker is a black button with white text that says 'SELECCIONAR FECHA'. At the bottom of the form are two buttons: a green button labeled 'GUARDAR' (Save) and a red button labeled 'CANCELAR' (Cancel).

Al pulsar el botón encontramos un formulario con los campos necesarios para añadir un gasto. Los campos son los anteriormente comentados y la regularidad. El campo de la regularidad sirve para que el gasto que registremos se repita en función de la regularidad que elijamos. Para que el gasto no se siga repitiendo solo hay que borrar el último registro del gasto que tengamos.

Para seleccionar la fecha debemos de presionar el botón y elegir la fecha en el calendario que aparecerá en la pantalla como aquí se muestra.

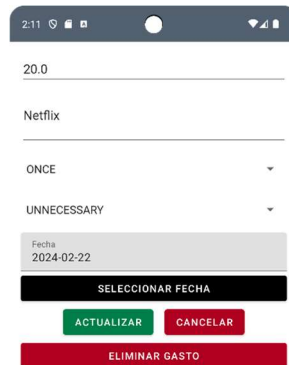


Al elegir la fecha correcta pulsamos el botón “OK” y se escribirá la fecha un campo deshabilitado. Al guardar el gasto se listará en pantalla anterior.





Si queremos editar o borrar el gasto, solo debemos de pulsar en el registro y se nos abrirá un formulario parecido al de crear el gasto. Además de los campos anteriormente mencionados también tenemos el botón de eliminar el gasto.

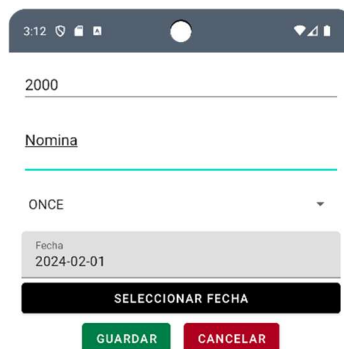


A screenshot of a mobile application interface for editing an expense. The form contains the following fields and controls: a numeric input field with the value '20.0'; a text input field with the value 'Netflix'; a dropdown menu currently showing 'ONCE'; another dropdown menu currently showing 'UNNECESSARY'; a date selection field showing 'Fecha 2024-02-22' with a 'SELECCIONAR FECHA' button below it; a green 'ACTUALIZAR' button; a red 'CANCELAR' button; and a prominent red 'ELIMINAR GASTO' button at the bottom.

Aquí podremos editar el gasto o simplemente borrarlo.

## GESTIÓN DE INGRESOS

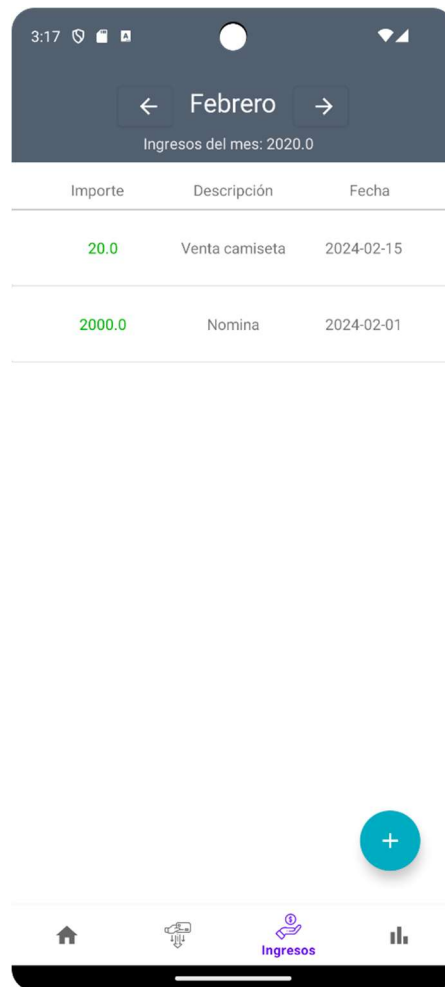
La gestión de ingresos funciona muy parecido a la gestión de gastos. En primer lugar tenemos la lista de los ingresos. Pulsando el botón flotante podemos añadir un ingreso.



A screenshot of a mobile application interface for adding a new income. The form contains the following fields and controls: a numeric input field with the value '2000'; a text input field with the value 'Nomina'; a dropdown menu currently showing 'ONCE'; a date selection field showing 'Fecha 2024-02-01' with a 'SELECCIONAR FECHA' button below it; a green 'GUARDAR' button; and a red 'CANCELAR' button.



Una vez guardado el ingreso podemos editarlo o eliminarlo de la misma manera que los gastos, pulsando en el registro.



Aquí tenemos un ejemplo de como se listan 2 ingresos.

## GRÁFICOS

Por último, tenemos otra pantalla de información, esta vez en forma de gráfica. Aquí se puede hacer un seguimiento de los gastos necesarios, innecesarios y ahorros de cada mes en forma de gráfico de barras.

