Universidad Don Bosco

Facultad de ingeniería

Desarrollo de Software para Móviles

ENTREGA DE PROYECTO PRIMERA FASE

Integrantes:

José Rolando Álvarez Mejía AM232553

Diego Enrique Canjura Estrada CE221950

Kevin Roberto Ramírez Montiel RM230143

Oswaldo Alexander Henríquez Miranda HM231279

programa	Link
Figma	Link Figma
Notion	Link notion
Github	Link Github

Fecha de entrega:

21 de Agosto de 2024

Indice

1. Introducción

En un entorno cada vez más dinámico y complejo, la gestión eficaz de las finanzas personales se ha convertido en un aspecto crucial para alcanzar estabilidad y éxito financiero. Las herramientas digitales juegan un papel esencial en facilitar este proceso, ofreciendo a los usuarios la posibilidad de llevar un control detallado y organizado de sus ingresos y gastos. En este contexto, presentamos el desarrollo de una aplicación de finanzas personales diseñada para simplificar y optimizar la administración de los recursos económicos de sus usuarios.

Este documento proporciona una visión completa del desarrollo del sistema, comenzando con el análisis detallado de las necesidades y funcionalidades principales requeridas por los usuarios. A lo largo de las fases de diseño y desarrollo, se han considerado aspectos clave como la gestión de categorías, la implementación de funciones matemáticas para cálculos automáticos, y la posibilidad de establecer y controlar presupuestos. Estos elementos son fundamentales para ofrecer una herramienta efectiva que permita a los usuarios no solo registrar sus gastos e ingresos, sino también planificar y anticipar sus necesidades financieras.

La aplicación será construida utilizando una arquitectura de microservicios basada en API REST, lo que garantiza una alta modularidad y flexibilidad en el desarrollo. El uso de tecnologías modernas como Kotlin y Jetpack Compose asegura que la aplicación no solo sea robusta y eficiente, sino también atractiva y fácil de usar. Además, se han considerado aspectos de seguridad y privacidad para proteger los datos financieros de los usuarios.

En el desarrollo de este sistema, se ha contado con un equipo de estudiantes de Ciencias de la Computación de la Universidad Don Bosco, quienes han asumido roles específicos en el backend, la base de datos, y el frontend de la aplicación. La colaboración efectiva entre los miembros del equipo y la utilización de herramientas avanzadas han sido cruciales para el avance del proyecto.

Este documento también incluye un cronograma detallado que abarca desde la fase inicial de diseño hasta el mantenimiento continuo post-lanzamiento. Además, se proporciona un glosario de términos técnicos utilizados, así como una sección de

encuestas realizadas para entender mejor las necesidades y expectativas de los futuros usuarios de la aplicación.

Esperamos que esta documentación no solo sirva como una guía completa para el desarrollo y la implementación de la aplicación, sino que también ofrezca una base sólida para futuras actualizaciones y mejoras, garantizando así la entrega de una solución eficaz y adaptada a las necesidades del mercado.

2. Objetivos del proyecto

1.1 Objetivo General

Desarrollar una aplicación multiplataforma de seguimiento de gastos que permita a los usuarios registrar, gestionar y analizar sus gastos financieros de manera eficiente y sencilla en dispositivos Android e iOS.

1.2 Objetivos Específicos

1. Desarrollar una Aplicación Multiplataforma
    - Objetivo: Implementar una aplicación que funcione tanto en Android como en iOS utilizando Kotlin Multiplataforma.
    - Descripción: Crear una base de código común que permita desarrollar y mantener una aplicación que funcione en ambas plataformas, asegurando que el 90% del código sea compartido entre Android e iOS.
2. Diseñar una Interfaz de Usuario Intuitiva
    - Objetivo: Diseñar y desarrollar una interfaz de usuario moderna y fácil de usar con Jetpack Compose.
    - Descripción: Utilizar Jetpack Compose para crear una interfaz que permita a los usuarios añadir, editar y visualizar gastos de manera sencilla y atractiva.
3. Implementar Gestión de Datos Eficiente
    - Objetivo: Desarrollar un sistema de gestión de datos que permita a los usuarios almacenar y recuperar información sobre sus gastos.

- Descripción: Implementar un sistema de almacenamiento local y remoto para gestionar los datos de los gastos, asegurando la sincronización entre dispositivos y la persistencia de datos.

4. Aplicar Arquitectura de Software Limpia
    - Objetivo: Utilizar la arquitectura MVVM para estructurar el código de manera modular y mantenible.
    - Descripción: Aplicar el patrón de arquitectura MVVM para separar las responsabilidades de la aplicación, facilitando la escalabilidad y mantenimiento del código.
5. Implementar Funcionalidades de Análisis y Reportes
    - Objetivo: Desarrollar características que permitan a los usuarios analizar y visualizar sus gastos.
    - Descripción: Incluir funcionalidades como gráficos y reportes para ayudar a los usuarios a entender mejor sus hábitos de gasto y tomar decisiones financieras informadas.
6. Garantizar la Calidad y Fiabilidad del Software
    - Objetivo: Asegurar la calidad del software mediante pruebas exhaustivas.
    - Descripción: Implementar pruebas unitarias y de integración para garantizar que la aplicación funcione correctamente y cumpla con los requisitos establecidos.
7. Optimizar el Rendimiento de la Aplicación
    - Objetivo: Mejorar el rendimiento y la eficiencia de la aplicación.
    - Descripción: Realizar optimizaciones para asegurar tiempos de respuesta rápidos y un consumo eficiente de recursos en ambos sistemas operativos.
8. Incorporar Capacidades de Sincronización en la Nube
    - Objetivo: Implementar capacidades de sincronización de datos en la nube para acceso desde múltiples dispositivos.
    - Descripción: Integrar servicios en la nube para que los datos del usuario se sincronicen entre dispositivos, permitiendo el acceso a la información desde cualquier lugar

3. Marco Teórico del proyecto

2.1 Alcances del Proyecto

2.1.1 Desarrollo de Aplicación Multiplataforma

Descripción: El proyecto se centra en el desarrollo de una aplicación que funcionará tanto en Android como en iOS utilizando Kotlin Multiplataforma. El objetivo es compartir la mayor parte del código posible entre las dos plataformas para reducir el tiempo de desarrollo y mantenimiento.

Alcance: La aplicación permitirá a los usuarios gestionar sus gastos, visualizar reportes, y realizar un seguimiento de sus finanzas en ambos sistemas operativos con una única base de código.

2.1.2 Interfaz de Usuario (UI)

Descripción: Se diseñará una interfaz de usuario intuitiva utilizando Jetpack Compose para garantizar una experiencia de usuario uniforme y moderna en ambas plataformas.

Alcance: La UI incluirá funciones para añadir, editar y visualizar gastos, así como gráficos y reportes de análisis.

2.1.3 Gestión de Datos

Descripción: El proyecto incluirá un sistema de gestión de datos que permite almacenar y recuperar información sobre los gastos del usuario.

Alcance: Se implementará almacenamiento local en el dispositivo y sincronización con una base de datos en la nube para mantener la información actualizada y accesible desde múltiples dispositivos.

2.1.4 Arquitectura y Patrones de Diseño

Descripción: Se aplicará el patrón de arquitectura MVVM (Model-View-ViewModel) para estructurar la aplicación de manera modular y mantenible.

Alcance: El proyecto utilizará MVVM para separar las responsabilidades y facilitar el mantenimiento y escalabilidad de la aplicación.

2.1.5 Funcionalidades de Análisis y Reportes

Descripción: La aplicación incluirá funcionalidades para el análisis de gastos y la generación de reportes.

Alcance: Los usuarios podrán visualizar gráficos, informes y estadísticas relacionadas con sus gastos financieros.

2.1.6 Pruebas y Validación

Descripción: Se realizarán pruebas exhaustivas para asegurar la calidad y el rendimiento de la aplicación.

Alcance: Se incluirán pruebas unitarias, de integración y de usabilidad para validar el correcto funcionamiento de todas las funcionalidades.

2.2 Limitaciones del Proyecto

2.2.1 Alcance de las Plataformas Soportadas

Descripción: Aunque la aplicación estará disponible para Android e iOS, no se considerarán otras plataformas como Windows o Linux.

Limitación: La aplicación no será compatible con plataformas adicionales, lo que podría limitar el alcance de usuarios potenciales.

2.2.2 Funcionalidades Avanzadas

Descripción: Algunas funcionalidades avanzadas, como la integración con servicios financieros externos o la gestión de inversiones, no se incluirán en la primera versión de la aplicación.

Limitación: El proyecto se enfocará en el seguimiento de gastos básicos, lo que puede dejar fuera características más avanzadas que los usuarios podrían desear en el futuro.

2.2.3 Compatibilidad con Versiones Antiguas de los Sistemas Operativos

Descripción: La aplicación se desarrollará y optimizará para las versiones más recientes de Android e iOS.

Limitación: La compatibilidad con versiones anteriores de los sistemas operativos puede ser limitada, lo que podría afectar a los usuarios con dispositivos más antiguos.

2.2.4 Recursos y Tiempo

Descripción: El proyecto está sujeto a restricciones de tiempo y recursos humanos.

Limitación: Las limitaciones en el tiempo de desarrollo y el número de desarrolladores pueden afectar la velocidad de implementación y la posibilidad de incluir todas las funcionalidades deseadas.

2.2.5 Sincronización y Conectividad

Descripción: La sincronización de datos en la nube dependerá de la conectividad del usuario a Internet.

Limitación: Los usuarios con conexiones de Internet inestables o lentas podrían experimentar problemas con la sincronización de datos y el rendimiento general de la aplicación.

2.2.6 Seguridad de Datos

Descripción: Aunque se implementarán medidas de seguridad para proteger los datos del usuario, no se garantiza una protección absoluta contra todas las posibles amenazas.

Limitación: La aplicación puede estar sujeta a riesgos de seguridad y privacidad que no se pueden eliminar completamente.

4. Empresa Seleccionada

3.1 Antecedentes del negocio

3.1.1 Interpretación de la lógica del negocio

Métodos utilizados para interpretar la lógica del negocio:

Para comprender la lógica del negocio de esta pequeña empresa, se han utilizado los siguientes métodos:

1. Observación: Se realizó una observación detallada del mercado de aplicaciones financieras, identificando tendencias actuales en la gestión de gastos personales. Esto incluyó el análisis de las aplicaciones más populares, sus funcionalidades, y las áreas donde los usuarios suelen encontrar dificultades o limitaciones.
2. Interacción: Se realizaron entrevistas y encuestas a un grupo de usuarios potenciales para entender mejor sus necesidades y desafíos en la gestión de sus finanzas personales. Estas interacciones proporcionaron información valiosa sobre las funcionalidades más deseadas, como la simplicidad en el seguimiento de gastos, la posibilidad de categorizar los gastos, y la generación automática de reportes.
3. Inspección: Se hizo una inspección detallada de las prácticas actuales en la gestión de finanzas personales, tanto manuales (como hojas de cálculo) como digitales (otras apps disponibles en el mercado). Esta inspección ayudó a identificar áreas donde la automatización y la personalización pueden agregar un valor significativo.

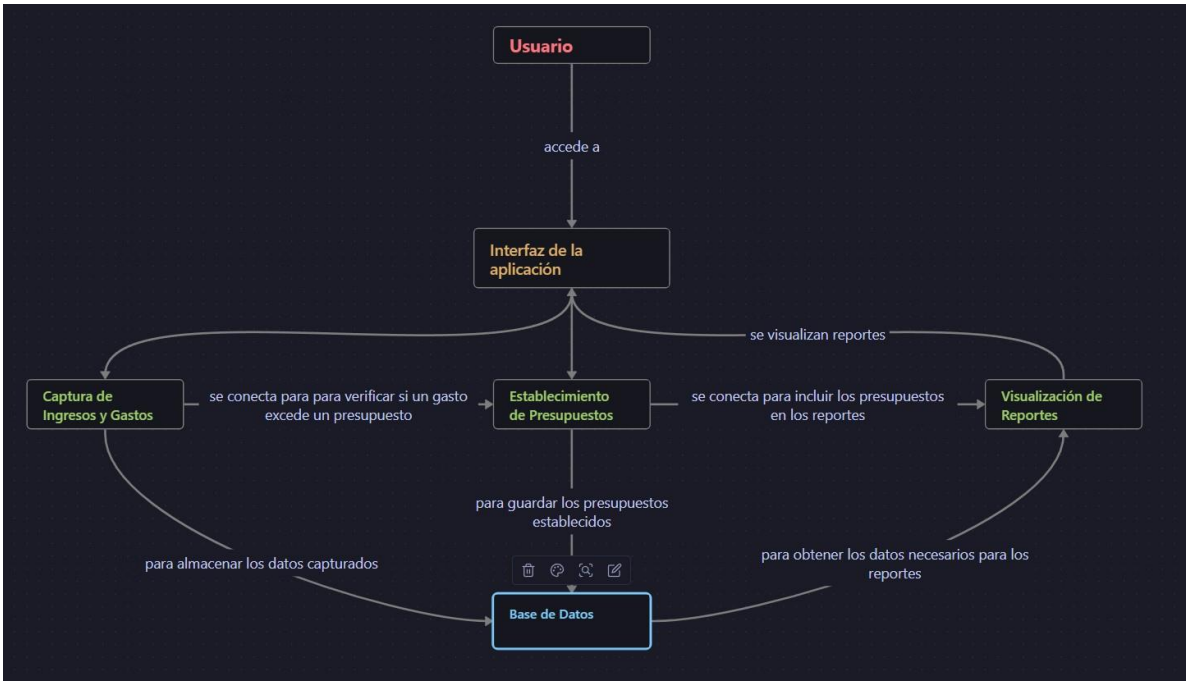
Resultado de la interpretación:

Basado en los métodos anteriores, se desarrolló un diagrama de la lógica del negocio que describe cómo la app gestionará los gastos personales de los usuarios. A continuación se detalla cada proceso a dinamizar:

- Captura de Ingresos y Gastos: El usuario podrá registrar sus ingresos y gastos de manera sencilla, con la opción de categorizar cada transacción. La app permitirá la captura manual o la sincronización automática con cuentas bancarias.

- Establecimiento de Presupuestos: Los usuarios podrán establecer presupuestos para diferentes categorías (como alimentación, transporte, entretenimiento) y la app les notificará si están cerca de exceder sus límites.
- Generación de Reportes: La app generará reportes visuales (gráficos de barras, circulares, etc.) que mostrarán el estado financiero del usuario en cualquier momento, incluyendo un análisis de tendencias y proyecciones futuras.
- Personalización y Alertas: Los usuarios podrán personalizar alertas para recordar pagos, recibir recomendaciones de ahorro, y establecer metas financieras.

Diagrama de la lógica del negocio:

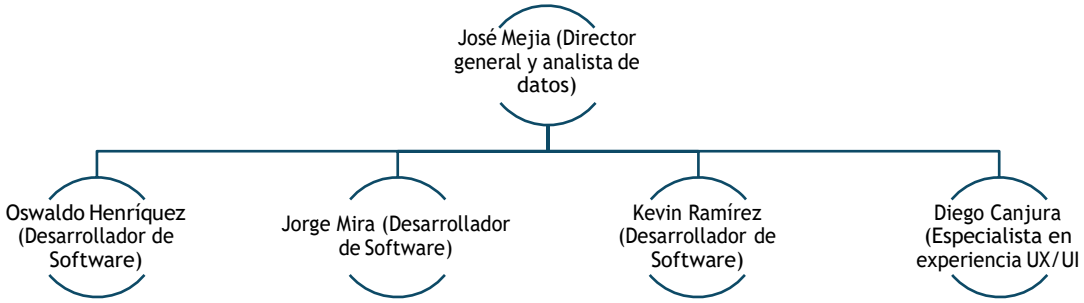


3.1.2 Organigrama de la empresa

En nuestro organigrama empresarial describimos la estructura organizativa y los roles clave dentro de la empresa que desarrollará la app de gestión de gastos personales. Estos son los cargos principales.

- Director General (CEO): Responsable de la visión y dirección estratégica de la empresa. Supervisa todas las operaciones y toma decisiones clave.

- Desarrolladores de Software: Equipo técnico encargado de codificar, probar y mantener la aplicación.
- Especialista en Experiencia de Usuario (UX/UI Designer): Diseña la interfaz y experiencia del usuario, asegurando que la app sea intuitiva y fácil de usar.
- Analista de Datos: Encargado de analizar los datos de uso de la app y proporcionar insights para mejorar continuamente el producto.



3.1.3 Análisis del perfil del cliente

El perfil del cliente ha sido desarrollado utilizando el Lienzo de Mapa de Empatía, que ayuda a comprender mejor las necesidades, deseos y preocupaciones del usuario final.

Aportaciones de los clientes:

1. ¿Qué piensa y siente?
    - Los clientes están preocupados por la falta de control sobre sus finanzas personales y desean una herramienta que les ayude a gestionar sus gastos de manera eficiente.
    - Sienten ansiedad al no tener una visión clara de su situación financiera.

2. ¿Qué ve?
    - Los clientes están expuestos a diversas aplicaciones financieras, pero encuentran que muchas de ellas son complicadas de usar o no ofrecen las funcionalidades que realmente necesitan.
3. ¿Qué dice y hace?
    - Expresan su frustración por no poder mantener un registro constante de sus gastos.
    - Buscan aplicaciones que sean fáciles de usar y que se adapten a su estilo de vida.
4. ¿Qué oye?
    - Escuchan de otros usuarios que la gestión de finanzas personales puede ser simplificada con las herramientas adecuadas.
    - Están influenciados por recomendaciones de amigos y familiares sobre aplicaciones que funcionan bien.

5. ¿Qué esfuerzos realiza?
    - Los clientes tratan de llevar un registro manual de sus gastos o usan aplicaciones que no son completamente satisfactorias.
6. ¿Qué resultados desea?
    - Quieren una aplicación que les permita tener un control total de sus finanzas, con reportes claros y recomendaciones personalizadas para mejorar su situación económica.

5. Planteamiento del problema

Problema Identificado:

La mayoría de las personas no tienen un control efectivo sobre sus finanzas personales, lo que puede llevar a un mal manejo de sus ingresos y gastos, y a la toma de decisiones financieras desinformadas. Aunque existen diversas

aplicaciones en el mercado, muchas de ellas son complicadas de usar, no ofrecen una personalización adecuada o no integran todas las funcionalidades necesarias en un solo lugar.

Solución Propuesta:

Desarrollar una aplicación intuitiva y completa de gestión de gastos personales, que permita a los usuarios registrar sus transacciones, establecer presupuestos, recibir alertas, y generar reportes detallados de su situación financiera. La app estará diseñada para ser fácil de usar, incluso para personas con conocimientos financieros limitados, y se personalizará según las necesidades individuales de cada usuario

6. Descripción Detallada de la Aplicación

Propuesta de Desarrollo

Descripción de la Aplicación

FinancePilot es una innovadora aplicación móvil diseñada para simplificar y optimizar el control de gastos e ingresos personales. En un mundo donde la gestión financiera puede resultar abrumadora, FinancePilot se presenta como una herramienta indispensable para aquellos que están dando sus primeros pasos en el ámbito laboral y para quienes aún no han adoptado un sistema riguroso de seguimiento de sus finanzas. Esta aplicación está pensada para quienes desean tomar el control de su economía personal, brindándoles una plataforma intuitiva y accesible que se ajusta a sus necesidades diarias.

Con una interfaz gráfica amigable y fácil de usar, FinancePilot permite a los usuarios organizar y categorizar sus ingresos y gastos de manera eficiente. Cada transacción se puede clasificar en diferentes categorías predefinidas o personalizadas, facilitando una visión clara de cómo se distribuye el dinero a lo largo del tiempo. Esta categorización no solo ayuda a los usuarios a entender sus patrones de gasto, sino que también les proporciona información valiosa para la planificación financiera a futuro.

Una de las características destacadas de FinancePilot es su capacidad para generar informes y análisis detallados. Los usuarios pueden acceder a gráficos y reportes

que visualizan sus hábitos de gasto, ingresos y ahorros, permitiéndoles identificar áreas donde pueden ajustar su presupuesto y mejorar su salud financiera. Además, la aplicación incluye herramientas de planificación que ayudan a los usuarios a establecer metas financieras y seguir su progreso a lo largo del tiempo.

FinancePilot también se enfoca en la accesibilidad, permitiendo que los usuarios ingresen y revisen sus datos desde cualquier lugar y en cualquier momento. La aplicación está diseñada para ofrecer una experiencia fluida en dispositivos móviles, asegurando que el manejo de las finanzas personales sea conveniente y sin complicaciones.

Funcionalidades Principales

1. **Interfaz Gráfica (UI/UX)**:
    - **Diseño Adaptado a Móviles Android**: La interfaz está diseñada para ser intuitiva y fácil de usar en dispositivos móviles Android. Se emplearán prácticas de diseño de interfaz de usuario modernas para asegurar una experiencia de usuario fluida y agradable.
    - **Interactividad**: La aplicación tendrá una navegación sencilla y una disposición clara de los elementos para facilitar el acceso a las funciones principales sin confusión.
    - **Visualización de Datos**: Utilización de gráficos y tablas para mostrar resúmenes financieros, como ingresos, gastos y presupuestos.
2. **Componentes Dinámicos**:
    - **Formularios Interactivos**:
        - Los formularios permitirán a los usuarios ingresar datos relacionados con gastos e ingresos. Los campos de entrada se ajustarán dinámicamente para facilitar la entrada de información relevante.
    - **Actualización en Tiempo Real**:
        - La aplicación implementará tecnologías para permitir la actualización de datos y la visualización en tiempo real sin necesidad de recargar la página. Esto incluye la sincronización de datos y la actualización de interfaces basadas en las entradas del usuario.
3. **Componentes Estáticos**:
    - **Contenido Informativo**:

        - Se incluirán secciones estáticas como preguntas frecuentes (FAQ), términos y condiciones, y políticas de privacidad para informar a los usuarios sobre el uso de la aplicación.

    - **Recursos Visuales**:
        - La aplicación usará íconos, imágenes y gráficos que mejorarán la estética y funcionalidad, proporcionando una interfaz visualmente atractiva y funcional.

4. **Acceso a Datos**:
    - **Base de Datos**:
        - La aplicación gestionará localmente la información sobre gastos e ingresos, asegurando la integridad y la eficiencia en el manejo de los datos.

- **Seguridad**:
    - Se implementarán medidas de seguridad para proteger la información del usuario, como el cifrado de datos sensibles y autenticación segura.

Propuesta de Desarrollo

1. **Fase de Planificación**:
    - **Definición de Requisitos**:
        - Recopilación de requisitos funcionales y no funcionales mediante entrevistas con usuarios potenciales y análisis de necesidades.
    - **Diseño de Interfaz Gráfica**:
        - Creación de wireframes y prototipos para la interfaz gráfica, definiendo la disposición de los elementos y la navegación.
    - **Elección de Tecnologías**:
        - Selección de tecnologías y herramientas a utilizar, basándose en los requisitos del proyecto y las mejores prácticas.
2. **Fase de Diseño**:
    - **Diseño de la Arquitectura del Sistema**:

        - Estructuración de la aplicación en módulos y servicios, diseño de la base de datos y planificación de la comunicación entre componentes.
    - **Prototipos**:
        - Desarrollo de prototipos interactivos para validar el diseño de la interfaz y la experiencia de usuario.
3. **Fase de Desarrollo**:
    - **Implementación del Frontend**:
        - Desarrollo de la interfaz gráfica utilizando Jetpack Compose y asegurando la compatibilidad con dispositivos Android.
    - **Implementación del Backend**:
        - Programación de los servicios backend utilizando Java, Kotlin y Ktor, y desarrollo de API REST para la comunicación entre el frontend y el backend.
    - **Integración de Funcionalidades**:
        - Implementación de funcionalidades principales como la gestión de categorías, cálculos matemáticos y presupuestos.
4. **Fase de Pruebas**:
    - **Pruebas de Funcionalidad**:
        - Verificación de que todas las funcionalidades de la aplicación operan según lo previsto y cumplen con los requisitos especificados.
    - **Pruebas de Usabilidad**:
        - Evaluación de la facilidad de uso y la accesibilidad de la interfaz gráfica mediante pruebas con usuarios reales.
    - **Pruebas de Seguridad**:
        - Revisión de las medidas de seguridad implementadas para proteger los datos y garantizar la privacidad de la información del usuario.
5. **Fase de Implementación**:
    - **Despliegue Local**:

        - Instalación y configuración de la aplicación en el entorno local, y realización de pruebas finales para asegurar el correcto funcionamiento.
    - **Capacitación**:
        - Provisión de materiales y soporte para capacitar a los usuarios en el uso de la aplicación.

6. **Fase de Mantenimiento**:
    - **Soporte Continuo**:
        - Provisión de soporte técnico para resolver problemas y responder a consultas de los usuarios.
    - **Actualizaciones**:
        - Implementación de mejoras y correcciones basadas en el feedback de los usuarios y los cambios en los requisitos.

7. Modelo de Análisis y Diseño de Sistemas

Análisis de Sistemas

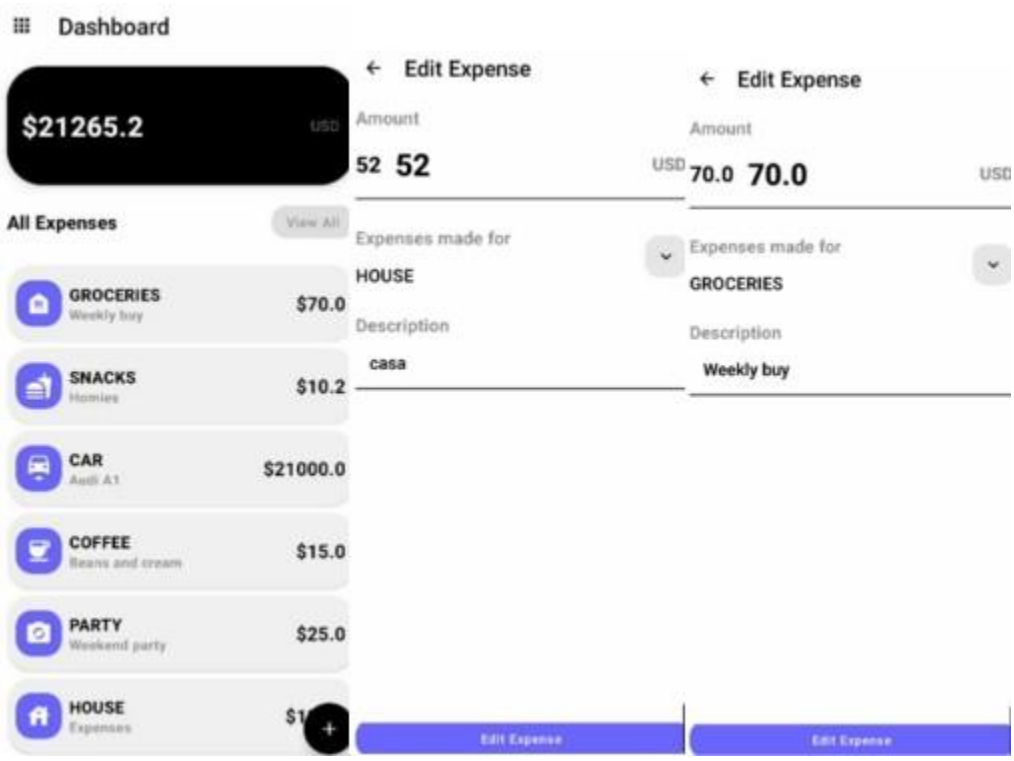
1. **Funcionalidades Principales**:
    - **Gestión de Categorías**:
        - Permitir a los usuarios crear, editar y eliminar categorías para clasificar sus gastos e ingresos. Cada categoría tendrá un nombre y un límite de presupuesto asociado.
    - **Funciones Matemáticas**:
        - Realizar cálculos automáticos para sumar ingresos, restar gastos y calcular balances. Incluirá funciones como sumas, promedios y análisis de variación.
    - **Presupuestos**:

- Permitir a los usuarios establecer presupuestos mensuales o anuales para cada categoría, con alertas cuando se acerquen o superen los límites establecidos.

- **CRUDs**:
  - Implementar operaciones básicas de Crear, Leer, Actualizar y Eliminar (CRUD) para gestionar los registros de gastos e ingresos.

Diseño del Sistema a Realizar

1. **Mockups**:
   - Descripción de los mockups que describen el diseño UX/UI que se implementará para la aplicación.



2. **Diagramas de Flujo y Casos de Uso**:
   - **Diagramas de Flujo**:
     - Visualización del flujo de datos y procesos dentro de la aplicación. Aunque no se tienen diagramas específicos, se

recomienda crear diagramas de flujo para los principales procesos de entrada y salida de datos.

- **Casos de Uso**:
  - Definir los escenarios en los que los usuarios interactúan con la aplicación, como añadir un gasto, crear un presupuesto, o generar un reporte financiero. Estos casos de uso ayudarán a identificar las funcionalidades requeridas y los requisitos del sistema.

Diseño Lógico y Físico del Sistema

1. **Diseño Lógico :**
   - **Arquitectura del Sistema:**
     - **Microservicios con API REST:**
       - **La aplicación adoptará una arquitectura de microservicios, donde cada microservicio será responsable de una funcionalidad específica (por ejemplo, gestión de categorías, cálculo de presupuestos). La comunicación entre microservicios se realizará a través de API REST.**
   - **Diagramas de Clases:**
     - **Representación de las principales clases del sistema, sus atributos y métodos, y las relaciones entre ellas. Estos diagramas ayudarán a estructurar el código y definir la lógica de negocio.**
2. **Diseño Físico:**
   - **Infraestructura Tecnológica:**
     - **Despliegue Local:**
       - **La aplicación se ejecutará en un entorno local sin dependencia de servicios en la nube. Se utilizarán servidores locales o dispositivos para la ejecución de la aplicación.**
   - **Tecnologías Utilizadas:**
     - **Frontend:**

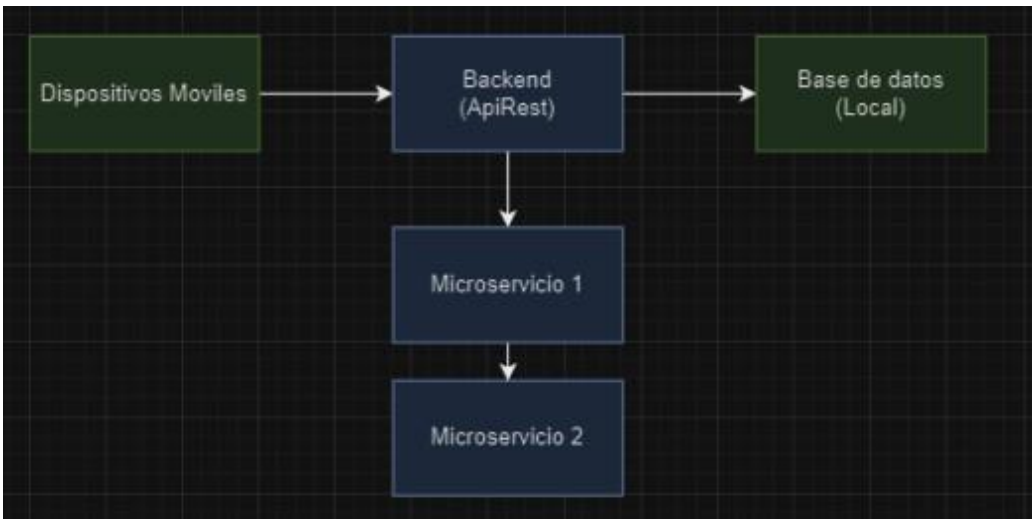
        - **Jetpack Compose: Utilizado para el diseño y desarrollo de la interfaz gráfica en dispositivos móviles Android, proporcionando una experiencia de usuario moderna y eficiente.**

    - **Backend:**

        - **Java y Kotlin: Lenguajes de programación utilizados para desarrollar la lógica del backend. Kotlin se usará para aprovechar sus características modernas y su interoperabilidad con Java.**
        - **Ktor: Framework para crear APIs REST y manejar la comunicación entre el frontend y el backend. Ktor permite la implementación eficiente de servicios web.**
        - **Kotlin Multiplatform (KMP): Tecnología para compartir código entre diferentes plataformas, facilitando la integración de funcionalidades comunes en diferentes entornos.**

Diagramas de Arquitectura del Sistema

- **Diagrama de Arquitectura del Sistema:**
    - **Descripción: Este diagrama muestra cómo se estructuran los diferentes componentes del sistema y cómo interactúan entre sí.**



Diagramas de Casos de Uso

- **Diagrama de Casos de Uso:**
    - **Descripción: Este diagrama muestra los casos de uso principales de la aplicación desde la perspectiva del usuario.**

v

    - **Componentes:**
        - **Actor: Usuario.**
        - **Casos de Uso: Agregar Presupuesto, Agregar Gasto, Ver Reportes, Configurar Presupuesto, etc.**

Componentes

- **Frontend: La aplicación móvil (desarrollada con Jetpack Compose).**
- **Backend: Servicios API REST (desarrollados con Ktor y Kotlin).**
- **Base de Datos: Almacenamiento local para gastos e ingresos.**

8. Roles Específicos de los Integrantes en la Realización del Sistema

Backend y Base de Datos

Integrantes:

- Jorge Nahum Mira Flores
- José Rolando Álvarez Mejía
- Kevin Roberto Ramírez Montiel

Responsabilidades y Tareas:

- Diseñar y desarrollar la lógica del servidor para la aplicación, incluyendo la creación de las API REST necesarias para la comunicación entre el backend y el frontend.
- Desarrollar la lógica para el procesamiento de datos.
- Diseñar, implementar y gestionar la base de datos utilizando SQLDelight, optimizando las consultas SQL y trabajando en la integración de SQLDelight con el resto de la aplicación, asegurándose de que los datos sean almacenados de manera eficiente y segura.
- Asegurarse de que el proyecto se pueda desplegar de manera segura y eficiente.
- Realizar pruebas para garantizar la calidad y estabilidad de la aplicación.

Frontend

Integrantes:

- Oswaldo Alexander Henríquez Miranda
- Diego Enrique Canjura Estrada

Responsabilidades y Tareas:

- Diseñar y desarrollar la interfaz de usuario de la aplicación, creando layouts atractivos y componentes visuales, y asegurando que la experiencia del usuario sea fluida e intuitiva.

- Trabajar con herramientas de diseño como XML y asegurarse de que la interfaz sea responsive en diferentes dispositivos.
- Implementar la lógica de negocio en el lado del frontend, incluyendo la gestión de estados, navegación entre pantallas y la integración con el backend.
- Manejar la interacción entre la interfaz de usuario y el backend utilizando ViewModels, LiveData y otros componentes de arquitectura de Android, asegurando la comunicación con el backend mediante llamadas a APIs y manejando respuestas y errores.

Colaboración y Comunicación

- Participar en reuniones para revisar el progreso y coordinar el trabajo entre frontend y backend.
- Cada miembro documentará su trabajo de manera adecuada para facilitar el mantenimiento y futuras actualizaciones de la aplicación.

9. Técnicas para la Recolección de Información

Técnica Seleccionada: Encuesta

1. ¿Sueles utilizar alguna aplicación para gestionar tus finanzas personales?
    - Sí
    - Raramente
    - No
2. ¿Qué funcionalidad consideras importante en una aplicación de finanzas?
    - Registro de gastos
    - Recordatorios de pagos
    - Creación de presupuestos personalizados
    - Sugerencias de ahorro
3. ¿Qué tan importante es para ti hacer un seguimiento de tus finanzas personales?

- Muy importante
- Importante
- Neutro
- Poco importante
- Nada importante

4. ¿Con qué frecuencia revisas tus ingresos y gastos?
    - Diariamente
    - Semanalmente
    - Mensualmente
    - Nunca
5. ¿Qué tan difícil te resulta mantener un control regular de tus finanzas personales?
    - Muy difícil
    - Difícil
    - Neutro
    - Fácil
    - Muy fácil
6. ¿Cuáles son los principales problemas que enfrentas al manejar tus finanzas personales?
    - Falta de tiempo para llevar un registro
    - Dificultad para mantener un presupuesto
    - No saber cómo ahorrar eficientemente
    - Otro
7. ¿Qué métodos utilizas actualmente para gestionar tus finanzas?
    - Hoja de cálculo
    - Aplicaciones de finanzas
    - En una libreta
    - No utilizo ningún método

8. ¿Qué tan satisfecho estás con los métodos actuales que utilizas para manejar tus finanzas?
    - Muy satisfecho
    - Satisfecho
    - Neutro
    - Insatisfecho
    - Muy insatisfecho
9. ¿Qué tipo de informes o análisis te gustaría recibir de una aplicación de finanzas?
    - Resumen mensual de gastos
    - Análisis de patrones de gastos
    - Proyecciones de ahorro
    - Comparaciones con meses anteriores
    - Otro:
10. ¿Te gustaría que la aplicación te permita establecer metas de ahorros o gastos?
    - Sí
    - Neutro
    - No
11. ¿Qué tan importante es para ti que la aplicación de finanzas sea fácil de usar?
    - Muy importante
    - Importante
    - Neutro
    - Poco importante
    - Nada importante
12. ¿Qué tan cómodo te sientes utilizando tecnología para gestionar tus finanzas?
    - Muy cómodo

- Cómodo
- Neutro
- Incómodo
- Muy incómodo

13. ¿Qué dispositivo prefieres utilizar para gestionar tus finanzas personales?
    - Smartphone
    - Tablet
    - Computadora
    - Otro
14. ¿Qué tan importante es la seguridad de tus datos financieros al usar una aplicación de finanzas?
    - Muy importante
    - Importante
    - Neutro
    - Poco importante
    - Nada importante
15. ¿Te gustaría participar en la beta de nuestra aplicación de finanzas personales?
    - Sí
    - No
    - Tal vez

10. Selección de la Tecnología:

Herramientas Informáticas Utilizadas para el Desarrollo del Sistema

- **Lenguaje de Programación:** Kotlin multiplataforma
- **Gestor de Base de Datos:** SQLDelight
- **APIs:** Ktor con Kotlin

- **Inteligencia Artificial:** Gemini AI (Solo para autocompletado)
- **Herramientas Adicionales:** Jetpack Compose, GitHub

11. Recursos Disponibles en la Empresa para la Implementación del Sistema

Nuestra aplicación será creada con la finalidad de ser totalmente de uso libre y, de ser posible, publicarse gratuitamente en alguna tienda Android o iOS. Sin embargo, los recursos con los que contamos son:

- **Equipo de Desarrollo:** Cinco estudiantes de la carrera de Ciencias de la Computación de la Universidad Don Bosco.
- **Herramientas de Desarrollo:** Las diferentes herramientas a utilizar están mencionadas en el punto anterior, tomando como base principal el lenguaje Kotlin y el IDE "Android Studio", ambos creados por Google y JetBrains.
- **Infraestructura Tecnológica:** La aplicación funcionará completamente en el dispositivo donde sea necesario, no contará con servicios en la nube, solamente con un servicio de IA para autocompletar diferentes enunciados.
- **Tiempo Disponible:** Hasta el 17 de octubre de 2024

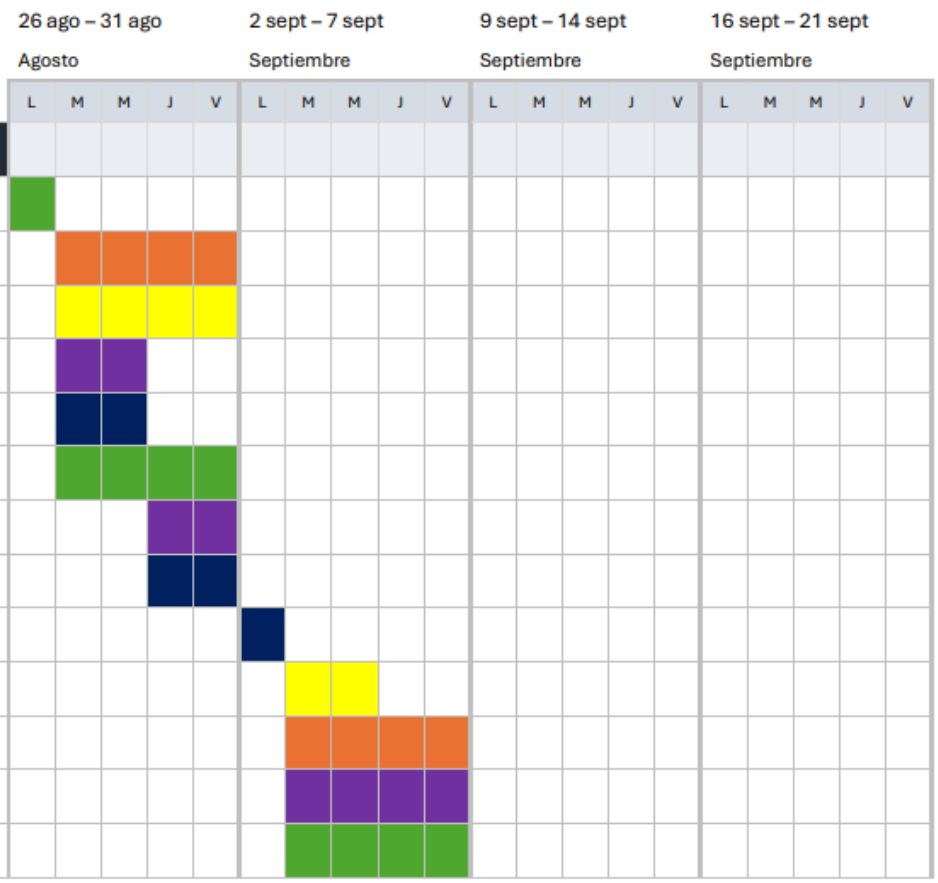
12. Cronograma de Actividades

Proyecto “Seguimiento de gastos Android/iOS”

INICIO DE LA SEMANA

26 AGO – 26 SEPT

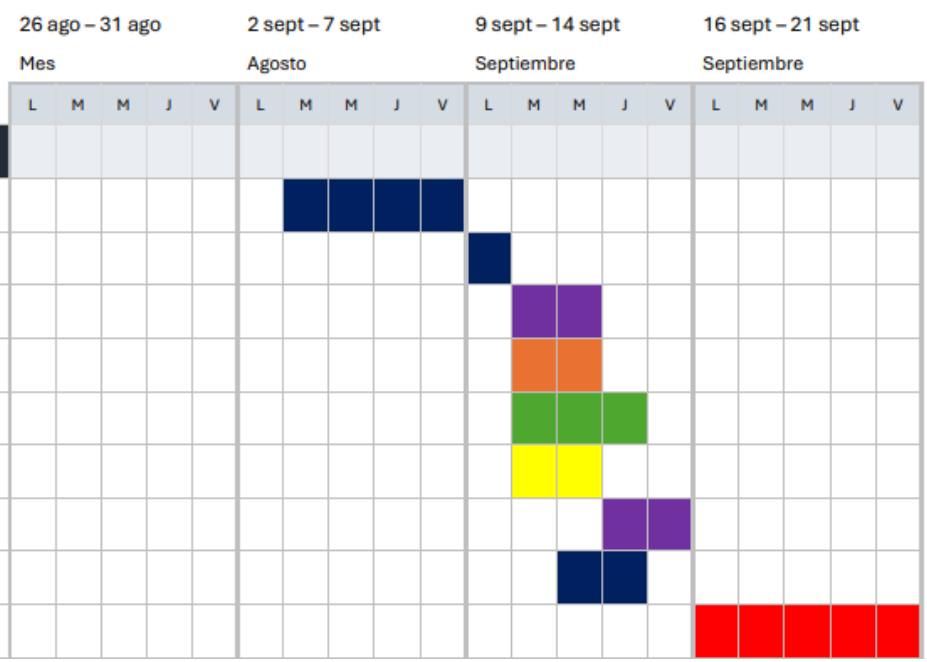
NOMBRE DE LA TAREA	ASIGNADO A	FECHA DE INICIO	FECHA FINAL	DURACIÓN en días	ESTADO
Configuración de entornos	Kevin Ramirez	26 Ago	26 Ago	1	Pendiente
Creación de UI para android	Oswaldo Henriquez	27 Ago	30 Ago	4	Pendiente
Creación de UI para iOS	Diego Canjura	27 Ago	30 Ago	4	Pendiente
Creación de arquitectura app	Jorge Mira	27 Ago	29 Ago	2	Pendiente
Creación y configuración AppTheme	José Mejia	27 Ago	28 Ago	2	Pendiente
Creación de capa de Datos	Kevin Ramirez	27 Ago	30 Ago	4	Pendiente
Creación de capa de presentación - MVVM	Jorge Mira	29 Ago	30 Ago	2	Pendiente
Creación y configuración de TopBar	José Mejia	29 Ago	30 Ago	2	Pendiente
Primer Testing	Jose Mejia	2 Sept	2 Sept	1	Pendiente
Inyección de Dependencias	Diego Canjura	3 Sept	4 Sept	2	Pendiente
Base de datos con SQLDelight	Oswaldo Henriquez	3 Sept	6 Sep	4	Pendiente
Creación de API REST	Jorge Mira	3 Sept	6 Sept	4	Pendiente
Creación de API REST	Kevin Ramirez	3 Sept	6 Sept	4	Pendiente



INICIO DE LA SEMANA

26 AGO – 26 SEPT

NOMBRE DE LA TAREA	ASIGNADO A	FECHA DE INICIO	FECHA FINAL	DURACIÓN en días	ESTADO
Creación de swipe to delete	José Mejia	3 sept	6 sept	4	Pendiente
Testin 2	José Mejia	9 sept	9 sept	1	Pendiente
Configuración multiplataforma	Jorge Mira	10 sep	11 sep	2	Pendiente
Configuración multiplataforma web	Oswaldo Henriquez	10 sep	11 sep	2	Pendiente
Implementación de AI	Kevin Ramirez	10 sept	12 sept	3	Pendiente
Creación de modo oscuro	Diego Canjura	10 sept	13 sept	2	Pendiente
StatusBar color iOS	Jorge Mira	12 sept	12 sept	2	Pendiente
StatusBar Color android	José Mejia	11 sept	12 sept	2	Pendiente
Pruebas finales	Todos	6 sept	21 sept	5	Pendiente



13. Fuentes de información APA

Libros

1. Sommerville, I. (2016). *Software Engineering* (10th ed.). Addison-Wesley.
    - **Referencia completa:** Sommerville, I. (2016). *Software Engineering* (10th ed.). Addison-Wesley.
2. Martin, R. C. (2009). *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall.
    - **Referencia completa:** Martin, R. C. (2009). *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall.

Artículos de Revistas

1. Boehm, B. W. (2002). *Get Ready for Agile Methods, with Care*. *IEEE Computer*, 35(1), 64-69. https://doi.org/10.1109/2.979166
    - **Referencia completa:** Boehm, B. W. (2002). Get ready for agile methods, with care. *IEEE Computer, 35*(1), 64-69. https://doi.org/10.1109/2.979166
2. Poppendieck, T., & Poppendieck, M. (2006). *Lean Software Development: An Agile Toolkit*. *IEEE Software*, 23(3), 24-31. https://doi.org/10.1109/MS.2006.70
    - **Referencia completa:** Poppendieck, T., & Poppendieck, M. (2006). Lean Software Development: An Agile Toolkit. *IEEE Software, 23*(3), 24-31. https://doi.org/10.1109/MS.2006.70

Recursos en Línea

1. JetBrains. (n.d.). *Kotlin*. Retrieved August 20, 2024, from https://kotlinlang.org/
    - **Referencia completa:** JetBrains. (n.d.). *Kotlin*. Retrieved August 20, 2024, from https://kotlinlang.org/
2. Google. (n.d.). *Jetpack Compose*. Retrieved August 20, 2024, from https://developer.android.com/jetpack/compose

    - **Referencia completa:** Google. (n.d.). *Jetpack Compose*. Retrieved August 20, 2024, from https://developer.android.com/jetpack/compose

3. SQLDelight. (n.d.). *SQLDelight Documentation*. Retrieved August 20, 2024, from https://cashapp.github.io/sqldelight/
    - **Referencia completa:** SQLDelight. (n.d.). *SQLDelight Documentation*. Retrieved August 20, 2024, from https://cashapp.github.io/sqldelight/

4. Ktor. (n.d.). *Ktor Documentation*. Retrieved August 20, 2024, from https://ktor.io/
    - **Referencia completa:** Ktor. (n.d.). *Ktor Documentation*. Retrieved August 20, 2024, from https://ktor.io/

Guías Técnicas y Documentos

1. Google. (2023). *Android Development with Jetpack Compose*. Retrieved August 20, 2024, from https://developer.android.com/jetpack/compose/guide
    - **Referencia completa:** Google. (2023). *Android Development with Jetpack Compose*. Retrieved August 20, 2024, from https://developer.android.com/jetpack/compose/guide

2. SQLDelight. (n.d.). *SQLDelight: An Overview*. Retrieved August 20, 2024, from https://cashapp.github.io/sqldelight/
    - **Referencia completa:** SQLDelight. (n.d.). *SQLDelight: An Overview*. Retrieved August 20, 2024, from https://cashapp.github.io/sqldelight/

14. Glosario

API (Interfaz de Programación de Aplicaciones): Conjunto de definiciones y protocolos para desarrollar e integrar software. Permite que diferentes aplicaciones se comuniquen entre sí.

CRUD (Crear, Leer, Actualizar, Eliminar): Operaciones básicas para gestionar datos en aplicaciones. Incluye añadir nuevos datos, leer datos existentes, actualizar datos y eliminar datos.

Frontend: Parte de una aplicación con la que el usuario interactúa directamente. Incluye la interfaz de usuario y la experiencia de usuario.

Backend: Parte del sistema que maneja la lógica del servidor, el procesamiento de datos y la interacción con la base de datos. No es visible para el usuario final.

Jetpack Compose: Biblioteca de Google para el desarrollo de interfaces de usuario en aplicaciones Android, utilizando un enfoque declarativo.

Kotlin: Lenguaje de programación utilizado principalmente para el desarrollo de aplicaciones Android. Ofrece características modernas y es interoperable con Java.

Ktor: Framework para construir aplicaciones web y APIs REST en Kotlin. Permite la creación de servicios web de manera eficiente.

Microservicios: Arquitectura de software en la que una aplicación se divide en pequeños servicios independientes que se comunican entre sí a través de API.

SQLDelight: Herramienta para trabajar con bases de datos SQL en aplicaciones multiplataforma, proporcionando un enfoque seguro y eficiente para manejar consultas SQL.

Diagramas de Clases: Representación gráfica de las clases en un sistema, mostrando sus atributos, métodos y las relaciones entre ellas.

Diagramas de Flujo: Diagramas que muestran el flujo de datos y procesos dentro de un sistema. Ayudan a visualizar la lógica de los procesos.

Jetpack Compose: Biblioteca de Google para diseñar interfaces de usuario en aplicaciones Android de manera declarativa.

Kotlin Multiplatform (KMP): Tecnología que permite compartir código entre diferentes plataformas, facilitando el desarrollo de aplicaciones en múltiples entornos.

Model-View-ViewModel (MVVM): Patrón de diseño que separa la lógica de presentación (ViewModel) de la interfaz de usuario (View) y el modelo de datos (Model) para mejorar la estructura y mantenimiento del código.

ViewModel: Componente del patrón MVVM que gestiona y almacena el estado de la interfaz de usuario, separando la lógica de la UI de la lógica de negocio.

LiveData: Clase en la arquitectura de Android que permite que los datos sean observables, facilitando la actualización de la interfaz de usuario cuando los datos cambian.

Encuesta: Método de recolección de datos a través de preguntas estructuradas, utilizado para obtener información sobre las opiniones y comportamientos de los usuarios.

Sistema Local: Aplicación que se ejecuta en un dispositivo o servidor local sin depender de servicios en la nube.

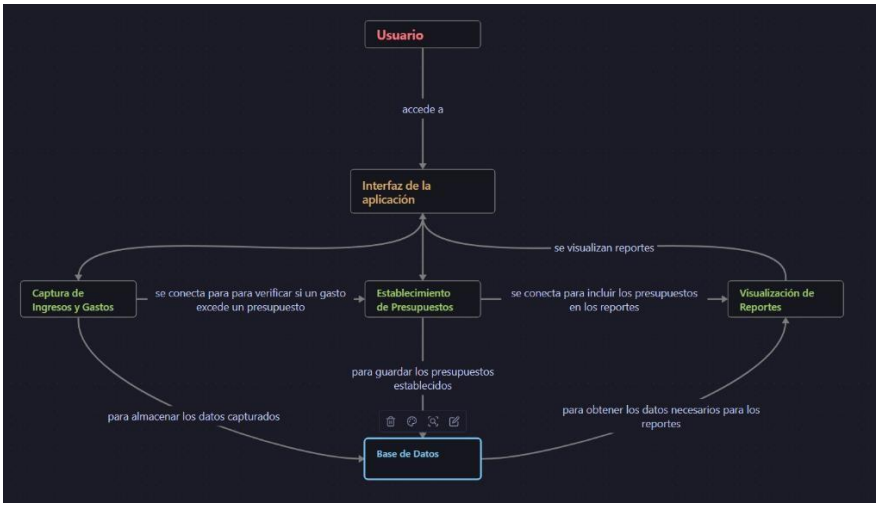
MVP (Model-View-Presenter): Patrón de diseño en el que el Presenter actúa como intermediario entre el Model y la View, facilitando la separación de responsabilidades en la aplicación.

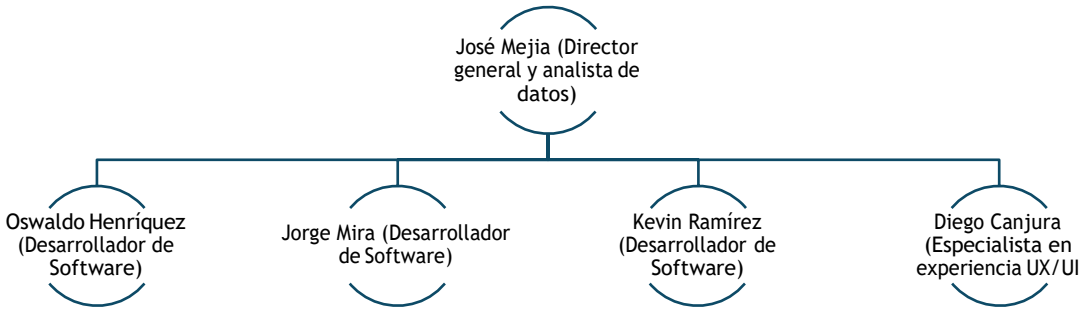
UI/UX (Interfaz de Usuario/Experiencia de Usuario): UI se refiere al diseño visual de la aplicación, mientras que UX se refiere a cómo el usuario experimenta y navega a través de la aplicación.

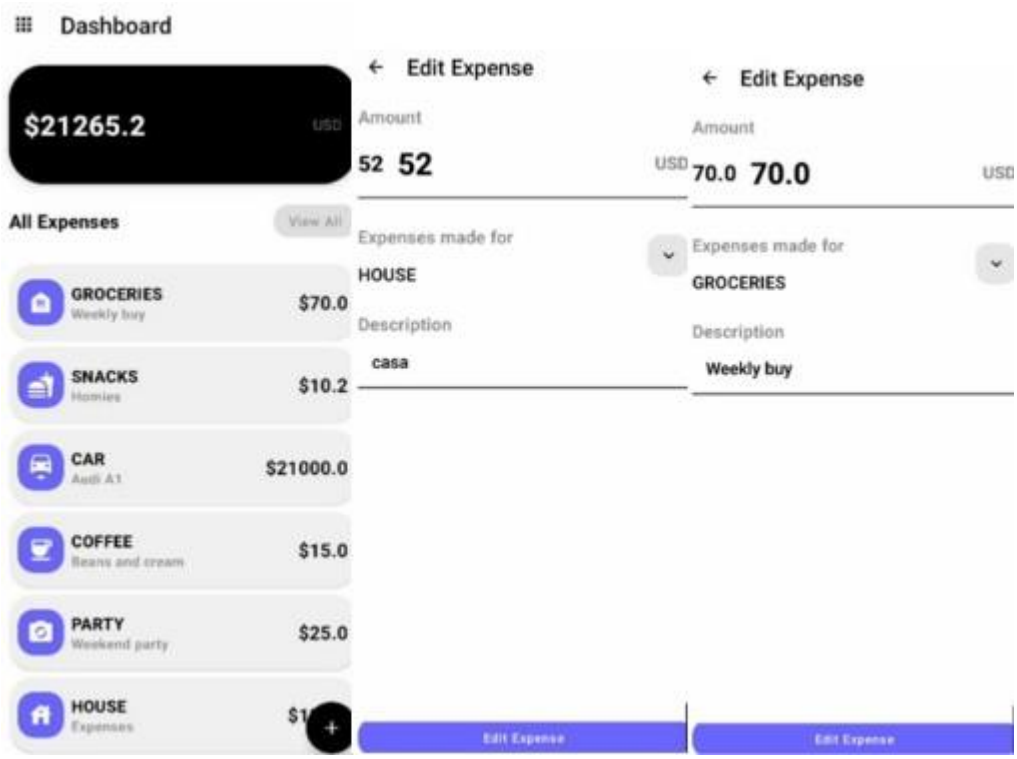
Autocompletado: Función de IA que sugiere o completa automáticamente información basada en patrones y datos previos, mejorando la eficiencia del usuario.

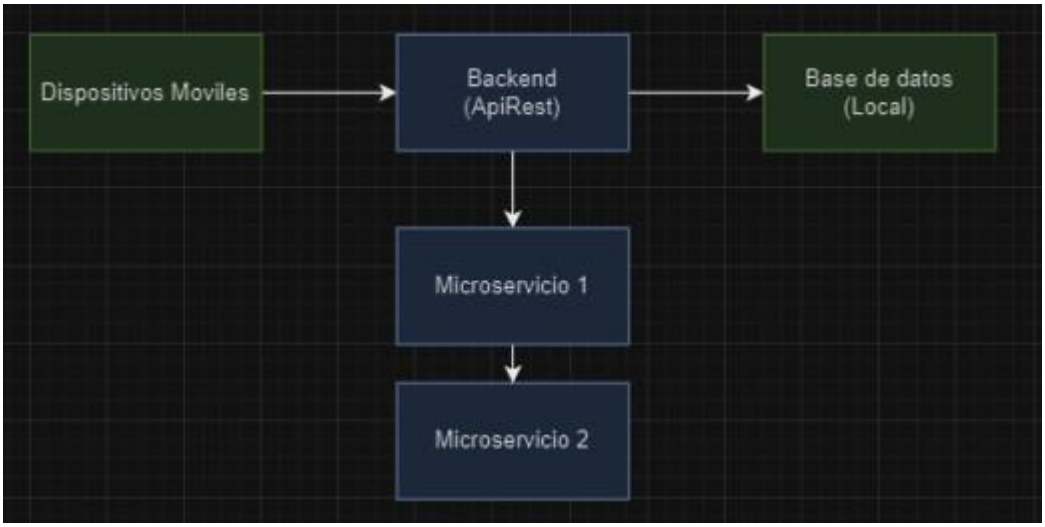
Sprint: Unidad de trabajo en metodologías ágiles como Scrum, donde se completa un conjunto de tareas en un período de tiempo definido.

15. Anexos




José Mejia (Director general y analista de datos)
Oswaldo Henríquez (Desarrollador de Software)
Jorge Mira (Desarrollador de Software)
Kevin Ramírez (Desarrollador de Software)
Diego Canjura (Especialista en experiencia UX/UI


Dashboard
$21265.2
USD
All Expenses
View All
GROCERIES
Weekly buy
$70.0
SNACKS
Homies
$10.2
CAR
Audi A1
$21000.0
COFFEE
Beans and cream
$15.0
PARTY
Weekend party
$25.0
HOUSE
Expenses
Edit Expense
Amount
52 52
USD
Expenses made for
HOUSE
Description
casa
Edit Expense
Edit Expense
Amount
70.0 70.0
USD
Expenses made for
GROCERIES
Description
Weekly buy
Edit Expense

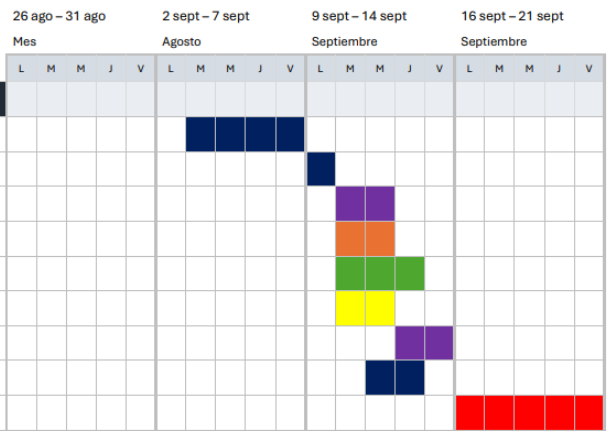

Dispositivos Moviles
Backend
(ApiRest)
Base de datos
(Local)
Microservicio 1
Microservicio 2


Usuario
Agregar presupuesto
Agregar gasto
Ver reporte
Configurar Gastos

INICIO DE LA SEMANA

26 AGO – 26 SEPT

NOMBRE DE LA TAREA	ASIGNADO A	FECHA DE INICIO	FECHA FINAL	DURACIÓN en días	ESTADO
Creación de swipe to delete	José Mejia	3 sept	6 sept	4	Pendiente
Testin 2	José Mejia	9 sept	9 sept	1	Pendiente
Configuración multiplataforma	Jorge Mira	10 sep	11 sep	2	Pendiente
Configuración multiplataforma web	Oswaldo Henriquez	10 sep	11 sep	2	Pendiente
Implementación de AI	Kevin Ramirez	10 sept	12 sept	3	Pendiente
Creación de modo oscuro	Diego Canjura	10 sept	13 sept	2	Pendiente
StatusBar color iOS	Jorge Mira	12 sept	12 sept	2	Pendiente
StatusBar Color android	José Mejia	11 sept	12 sept	2	Pendiente
Pruebas finales	Todos	6 sept	21 sept	5	Pendiente



Proyecto “Seguimiento de gastos Android/iOS”

INICIO DE LA SEMANA

26 AGO – 26 SEPT

NOMBRE DE LA TAREA	ASIGNADO A	FECHA DE INICIO	FECHA FINAL	DURACIÓN en días	ESTADO
Configuración de entornos	Kevin Ramirez	26 Ago	26 Ago	1	Pendiente
Creación de UI para android	Oswaldo Henriquez	27 Ago	30 Ago	4	Pendiente
Creación de UI para iOS	Diego Canjura	27 Ago	30 Ago	4	Pendiente
Creación de arquitectura app	Jorge Mira	27 Ago	29 Ago	2	Pendiente
Creación y configuración AppTheme	José Mejia	27 Ago	28 Ago	2	Pendiente
Creación de capa de Datos	Kevin Ramirez	27 Ago	30 Ago	4	Pendiente
Creación de capa de presentación - MVVM	Jorge Mira	29 Ago	30 Ago	2	Pendiente
Creación y configuración de TopBar	José Mejia	29 Ago	30 Ago	2	Pendiente
Primer Testing	Jose Mejia	2 Sept	2 Sept	1	Pendiente
Inyección de Dependencias	Diego Canjura	3 Sept	4 Sept	2	Pendiente
Base de datos con SQLDelight	Oswaldo Henriquez	3 Sept	6 Sep	4	Pendiente
Creación de API REST	Jorge Mira	3 Sept	6 Sept	4	Pendiente
Creación de API REST	Kevin Ramirez	3 Sept	6 Sept	4	Pendiente