Anteproyecto DAM

MyMoney es una aplicación de gestión de gastos personales.

Héctor Anido Regueiro

Ciclo de Desarrollo de Aplicaciones Multiplataforma. Proyecto fin de ciclo.

Tutor: Manuel Cayetano Piñeiro Mourazos

Indice.

Resumen del proyecto.

El proyecto consiste en una aplicación móvil para Android, diseñada bajo la arquitectura MVVM, cuyo propósito es gestionar la economía familiar o personal del usuario y ayudar a controlar sus gastos de manera eficiente.

Introducción.

El objetivo principal de la aplicación es ofrecer al usuario una herramienta eficiente para el seguimiento y control de sus gastos personales y familiares ya que hoy en día es normal que una persona tenga múltiples gastos. Por un lado, en los últimos años han aumentado mucho las suscripciones a distintos servicios y productos. También tengamos en cuenta que los cargos o facturas de dichos gastos suelen estar en distintas cuentas bancarias. Asimismo, también hay que tener en cuenta que dichos gatos provienen, cada vez más, de distintas fuentes desde las más clásicas como los recibos de suministros del hogar, pasando por pagos a sanidad privada que cada vez crecen más, todo tipo de seguros: coche, casa, vida, moto, móvil, etc., comercio online, plataformas de pago como Paypal o GooglePay hasta las ya mencionadas suscripciones: a software, a vehículos, a plataformas de cine y televisión, de música, etc. Por ello es importante que podamos juntar todos esos conceptos en una sola aplicación para poder gestionar nuestra economía de forma eficiente y controlada, optimizar recursos o mejorar la productividad.

La aplicación aborda estos desafíos mediante el uso de tecnologías móviles avanzadas, asegurando una experiencia de usuario fluida y una gestión eficiente de los datos financieros.

Objetivos.

El objetivo principal del proyecto es el de desarrollar por completo una aplicación de gestión de gastos a nivel personal/familiar que sea consistente y fácil de manejar. Dicha aplicación será desarrollada en la plataforma Android para que pueda ser accedida a todo su ecosistema y así llegue al máximo número de personas.

Queremos desarrollar una arquitectura sólida basada en el patrón MVVM (Model-View-ViewModel) para asegurar la separación de responsabilidades y mejorar la mantenibilidad del código.

Trataremos de que la interfaz sea lo más sencilla e intutiva que se pueda sin dejar de lado ni el diseño que ha de ser atractivo ni la usabilidad.

Algo que también deberemos tener en cuenta es que la inserción de registros a de ser lo más sencilla y rápida que se pueda, ya que es esencial que el usuario tenga facilidad para que cada vez que tenga un gasto lo pueda introducir en la aplicación sin que ello le suponga una tarea tediosa.

Otro objetivo es el de optimizar la inserción y visualización de los datos, garantizando la integridad y consistencia mediante el uso de tecnologías de persistencia de datos modernas como Room.

Descripción del proyecto.

Tendremos una pantalla de bienvenida con el título de la aplicación y el registro.

La pantalla de bienvenida nos llevará a la pantalla principal donde tendremos el menú inicial con acceso a distintas funcionalidades: gastos, ingresos, añadir un gasto o añadir un ingreso serán las principales, que serán ampliadas a medida que el tiempo de desarrollo lo permita con otras como podrían ser: registro de transferencias, presupuestos y estadísticas.

Cada una de las funcionalidades principales, como la inserción de registros, será gestionada mediante un ViewModel, permitiendo mantener los datos persistentes a través de cambios de configuración de la aplicación. El uso de fragmentos permitirá una mejor organización del código y facilitará futuras expansiones.

Funcionalidades principales:

La opción de “gastos” y la de “ingresos” nos llevará a otra pantalla donde tendremos un menú para escoger qué tipo de gastos o ingresos queremos visualizar.

Las opciones de “añadir gasto” y “añadir ingreso” nos llevará a una pantalla donde podremos introducir un registro en la base de datos.

Las funcionalidades principales como la inserción y visualización de registros se implementarán siguiendo el patrón MVVM, con fragmentos para garantizar una interfaz modular y escalable. El uso de RecyclerView, gestionado a través de un Adapter personalizado, permitirá una presentación optimizada de los datos.

Funcionalidades de ampliación:

Transferencias: permitiría tener en una tabla aparte, los registros con las transferencias realizadas o recibidas por el usuario. Ya que no se pueden considerar como un gasto o un ingreso ya que pueden ser transferencias a uno mismo o a un miembro de la familia considero que deben ir aparte.

Creo que el hecho de que no va a ser posible, por lo menos en esta versión, lograr la integración con otras aplicaciones de banca, hace que sea la funcionalidad que, de realizarse, sería la última en agregar a la app.

Presupuestos: al contrario de la anterior sí que considero que esta funcionalidad es muy interesante por lo que tendrá preferencia sobre las demás. Consistiría en darle al usuario la posibilidad de crear presupuestos personalizados, tanto globales como por tipos de gasto, incluyendo alertas sobre el nivel de presupuesto alcanzado, gráficas, etc.

La funcionalidad de presupuestos implementará un sistema de alertas basado en cálculos en tiempo real con observación de datos desde el ViewModel. Se integrarán gráficos a través de bibliotecas externas, como MPAndroidChart, para una representación visual interactiva de los datos.

Estadística: esta funcionalidad generaría, tomando como datos los que tengamos guardados en la base de datos, estadísticas y/o gráficas mostrando los datos estructurados al usuario sobre sus gastos e ingresos.

Plataformas.

La plataforma escogida es Android. El motivo es que es la plataforma con más usuarios.

Tecnologías a utilizar.

El lenguaje de programación será Kotlin. Actualmente Kotlin es el lenguaje oficial de Android.

El IDE para realizar el proyecto será Android Studio.

Usaremos un enfoque modular mediante fragmentos para la interfaz, pues es la práctica recomendada por Android y la asociación de vistas la haremos mediante viewBinding, todo esto con el objetivo de mantener una aplicación escalable y bien estructurada.

Implementaremos la interfaz usando diferentes layouts: LinearLayout, FrameLayout, RelativeLayout, ConstraintLayout, etc.

Haremos uso de ReciclerView para mostrar los datos de forma dinámica.

Hemos seleccionado Room como capa de persistencia de datos, integrándolo con LiveData para manejar actualizaciones en tiempo real. El uso de KSP en lugar de KAPT optimizará la generación de código y reducirá los tiempos de compilación. Junto con room, usaremos viewModel para la consistencia de datos en la aplicación junto con otras tecnologías como LiveData.

Metodología.

Para el desarrollo de esta aplicación usaremos el IDE Android Studio, respaldado por Git mediante un repositorio en Github para el control de versiones.

Las pruebas de la aplicación se llevarán a cabo mediante test unitarios y pruebas instrumentadas, empleando herramientas como JUnit.

En una primera etapa nos centraremos en el desarrollo de la lógica programática de las principales funcionalidades.

En una segunda etapa, pondremos más atención a toda la parte de la interfaz, intentando ajustar y pulir las vistas que tengamos funcionando. También tendremos que revisar la usabilidad de la aplicación.

En una hipotética tercera etapa el objetivo sería añadir nuevas funcionalidades que completen la aplicación.