Curso:

Programación para Dispositivos Móviles

Docente:

Josue Miguel Flores Parra

Tema:

Proyecto Final

Estudiantes:

Hugo Alonso Youzzueff Diaz Chavez

Miguel Andres Flavio Ocharan Coaquira

Arequipa, 2025

1. Descripción del problema y público objetivo

En el contexto actual, muchas personas registran sus gastos de manera desordenada o no llevan un control claro de su dinero. Esto dificulta identificar en qué categorías se gasta más, cómo evoluciona el gasto mensual y qué hábitos financieros se pueden mejorar.

La aplicación móvil ControlFast busca resolver este problema ofreciendo una herramienta sencilla para registrar, visualizar y analizar los gastos personales de forma rápida y accesible desde el teléfono.

El público objetivo de la aplicación son principalmente estudiantes, jóvenes profesionales y personas que desean empezar a organizar sus finanzas personales sin necesidad de herramientas complejas como hojas de cálculo o software especializado.

2. Características principales de la aplicación

- Registro y edición de gastos

La app permite registrar nuevos gastos indicando el monto, la categoría, la descripción y la fecha. También es posible editar un gasto ya registrado, actualizando sus datos según sea necesario.

- Listado de gastos

En la pantalla principal se muestra un listado de todos los gastos registrados, ordenados cronológicamente. Cada ítem muestra la descripción, la categoría, la fecha y el monto. Un clic corto permite editar el gasto y un clic prolongado abre un cuadro de diálogo para eliminarlo.

- Resumen y estadísticas

La pantalla de resumen presenta el total gastado, un gráfico de pastel con la distribución de gastos por categoría y estadísticas generales como la cantidad total de gastos, el promedio por gasto y la categoría con mayor monto acumulado.

- Eliminación de gastos

El usuario puede eliminar un gasto específico o eliminar todos los gastos mediante opciones con confirmación para evitar borrados accidentales.

- Persistencia de datos con Room

Todos los registros se almacenan de forma local en el dispositivo utilizando la librería Room. Esto garantiza que los datos se mantengan disponibles aunque se cierre la aplicación.

3. Arquitectura y patrón de diseño utilizados

La aplicación ControlFast utiliza la arquitectura MVVM (Model–View–ViewModel), combinada con el patrón Repository para separar la lógica de datos de la capa de presentación.

- Model (Datos)

Incluye las entidades y modelos de datos, como la clase Expense, que representa cada gasto, y el enum CategoriaGasto, que define las categorías predefinidas con sus colores asociados.

- View (Vista)

Está formada por las Activities y los layouts XML: MainActivity, AddExpenseActivity y SummaryActivity. Estas clases son responsables de mostrar la información al usuario y de gestionar las interacciones con la interfaz.

- ViewModel

La clase ExpenseViewModel actúa como intermediario entre la vista y el repositorio de datos. Expone LiveData con la lista de gastos, el total, los totales por categoría y el estado de operaciones. Además, se encarga de llamar a las funciones del repositorio utilizando corrutinas.

- Repository

El ExpenseRepository encapsula el acceso al ExpenseDao y ofrece métodos de más alto nivel para insertar, actualizar, eliminar y consultar gastos. Esto permite cambiar la fuente de datos en el futuro sin afectar a la UI.

- Room (DAO y base de datos)

Se utiliza Room para gestionar la base de datos local. ExpenseDao define las consultas (insertar, actualizar, eliminar, listar, agrupar por categoría, etc.) y AppDatabase proporciona el punto de acceso a la base de datos.

4. Decisiones de diseño y aprendizajes
Durante el desarrollo de ControlFast se tomaron varias decisiones de diseño orientadas a la simplicidad y a la aplicación de buenas prácticas de desarrollo móvil.

- Uso de MVVM y Repository

Se eligió la arquitectura MVVM porque facilita la separación de responsabilidades, mejora la mantenibilidad del código y permite aprovechar LiveData para actualizar la interfaz automáticamente cuando cambian los datos. El patrón Repository se utilizó para centralizar el acceso a datos y mantener desacopladas la UI y la base de datos.

- Uso de Room para persistencia

Room se seleccionó como solución de persistencia por su integración con LiveData y ViewModel, así como por proporcionar una capa de abstracción segura sobre SQLite. Esto permitió manejar operaciones de lectura y escritura en segundo plano usando corrutinas.

- Interfaz con Material Design

Se emplearon componentes de Material Design (MaterialToolbar, MaterialCardView, TextInputLayout, FAB, etc.) para brindar una interfaz moderna, clara y consistente. Se cuidó que los textos, tamaños y colores sean legibles y que el flujo de navegación sea intuitivo para el usuario.

- Gestión de cadenas en strings.xml

Como parte de las buenas prácticas, se migraron las cadenas de texto visibles al usuario hacia el archivo strings.xml. Esto facilita la internacionalización y el mantenimiento del proyecto, evitando textos "quemados" en los layouts o en el código Kotlin.

- Aprendizajes obtenidos

Entre los principales aprendizajes destacan: el uso integrado de ViewModel, LiveData y Room; la organización del proyecto siguiendo MVVM; la importancia de validar datos en formularios; y la generación de visualizaciones gráficas con librerías externas como MPAndroidChart. Además, se reforzó el uso de buenas prácticas en el manejo de recursos, comentarios en el código y estructuración de paquetes.