Documento de Arquitectura

Nombre del Proyecto: FinancialCalculatorApp

Hecho y diseñado por: Daniel Ospina Agudelo

Patrón de Diseño Utilizado: MVVM (Model-View-ViewModel)

Este documento describe la arquitectura utilizada en el desarrollo de FinancialCalculatorApp, basada en el patrón MVVM (Model-View-ViewModel).
A continuación, se explican las responsabilidades de cada módulo y su interacción dentro de la estructura del proyecto.

Estructura de Carpetas

La organización del proyecto está estructurada de la siguiente manera:

1. domain

a) models

- Archivo: CalculationModels.kt
- Descripción:
  Define los modelos de datos que representan la información manejada en la aplicación, como entradas y resultados de cálculos financieros.

b) services

- Archivo: CalculationService.kt
- Descripción:
  Contiene la lógica de negocio de la aplicación, como las fórmulas para los cálculos financieros.

2. presentation.viewModels

a) navigation

- Archivos:

AppNavigation.kt, minescreen.kt
- Descripción:
Manejan la navegación de la aplicación, definiendo rutas y pantallas disponibles dentro de la misma.

b) ViewModels

- Archivos:
EmployeeCalculationViewModel.kt, EmployerCalculationViewModel.kt, ProductCalculationViewModel.kt
- Descripción:
Encargados de manejar el estado y la lógica para las pantallas asociadas.

4. ui

a) screens

- Archivos:
EmployeeCalculationScreen.kt, EmployerCalculationScreen.kt, ProductCalculationScreen.kt
- Descripción:
Contienen las interfaces de usuario para cada una de las funcionalidades principales de la aplicación.

b) theme

Define el tema visual de la aplicación, incluyendo colores, tipografía y estilos generales.

5. MainActivity

- Archivo: MainActivity.kt
- Descripción:
Punto de entrada de la aplicación.

Relación entre Componentes (MVVM)

1. Model: Representado por los modelos de datos y los servicios en el módulo domain.
2. ViewModel: Ubicados en presentation.viewModels. Proveen datos a las vistas y reaccionan a sus eventos.
3. View: Implementada en ui.screens. Consume los datos proporcionados por los

ViewModels y muestra la interfaz al usuario.

Ventajas de la Arquitectura MVVM

- Separación de responsabilidades: Cada módulo tiene un propósito claro, facilitando el mantenimiento y la escalabilidad.
- Reutilización de código: Los ViewModels y modelos pueden ser utilizados en diferentes vistas sin modificaciones significativas.
- Facilidad de prueba: Los ViewModels y servicios pueden probarse de manera independiente de la interfaz gráfica.

Conclusión

FinancialCalculatorApp utiliza una arquitectura basada en MVVM para ofrecer una solución modular, mantenible y fácil de escalar.
Esta estructura asegura que la lógica de negocio y la interfaz estén claramente separadas, permitiendo un desarrollo ágil y eficiente.