UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERÍA

ESCUELA DE COMPUTACIÓN

CICLO 01 - 2023

Trabajo de investigación

Desarrollo de software para móviles

Catedrático:

Ing. Alexander Sigüenza

Alumnos:

Alexis Josué Cornejo Hernández CH190964

Josafat Norberto Zavaleta Gallegos ZG191275

Fecha de entrega:

29 de abril de 2023

Índice

Introducción. 3

Contenido. 4

Anexos. 6

Bibliografía. 7

Introducción

El patrón MVVM (Model-View-ViewModel) es un patrón de arquitectura de software ampliamente utilizado en el desarrollo de aplicaciones móviles. Proporciona una estructura organizada y clara para separar la lógica de negocio, la presentación de datos y la interfaz de usuario. En el patrón MVVM, el Modelo representa los datos y la lógica de negocio, la Vista es la interfaz de usuario y el ViewModel actúa como un intermediario entre ambos, preparando y exponiendo los datos del modelo para su presentación en la vista. A través de la aplicación del patrón MVVM, se logra una mejor modularidad y mantenibilidad del código. Sin embargo, también es importante considerar las ventajas y desventajas de su implementación, ya que puede introducir una mayor complejidad y dependencia de bibliotecas. En este resumen, exploraremos más a fondo el patrón MVVM, sus componentes principales, su aplicación en el desarrollo de aplicaciones Android con Kotlin, así como las ventajas y desventajas de su uso en el ámbito de aplicaciones móviles.

Contenido

1- ¿Qué es el patrón MVVM?

El patrón MVVM (Model-View-ViewModel) es un patrón de arquitectura de software que se utiliza en el desarrollo de aplicaciones de interfaz de usuario. Fue popularizado por Microsoft para su uso en el desarrollo de aplicaciones de escritorio con tecnologías como WPF y Silverlight, pero también se ha adoptado ampliamente en el desarrollo de aplicaciones móviles y web.

El objetivo principal del patrón MVVM es lograr una separación clara de responsabilidades entre los distintos componentes de una aplicación.

2- ¿Cuáles son sus componentes principales y cómo se relacionan entre sí?

Los componentes principales del patrón MVVM son:

A- Modelo (Model): Representa los datos y la lógica de negocio de la aplicación. El modelo no tiene conocimiento de la interfaz de usuario y se encarga de manejar la obtención y manipulación de los datos.
B- Vista (View): Es la interfaz de usuario de la aplicación. Es responsable de mostrar los datos al usuario y recibir las interacciones del usuario, pero no contiene lógica de negocio.
C- Modelo de vista (ViewModel): Actúa como un intermediario entre el modelo y la vista. Se encarga de preparar y exponer los datos del modelo de una manera que sea fácilmente consumible por la vista. Además, también maneja la lógica de presentación y la interacción con la vista.

La relación entre estos componentes es que la vista observa al ViewModel para obtener los datos necesarios para mostrar, mientras que el ViewModel observa al modelo para obtener los datos actualizados. La comunicación entre la vista y el ViewModel se realiza generalmente mediante el uso de enlaces de datos (data binding).

3- ¿Cómo se aplica el patrón MVVM en Android con Kotlin?

En el desarrollo de aplicaciones Android con Kotlin, el patrón MVVM se puede implementar siguiendo los siguientes pasos:

A- Definir el modelo de datos: El modelo de datos representa la estructura de datos y la lógica de negocio de la aplicación. Puede consistir en clases, objetos u otras estructuras de datos necesarias para la funcionalidad de la aplicación. Es importante separar el modelo de datos de la lógica de presentación y la interfaz de usuario.
B- Crear la vista: La vista se refiere a la interfaz de usuario de la aplicación. En Android, se puede implementar utilizando XML para definir la estructura de la interfaz de usuario y las vistas visuales. La vista debe ser lo más pasiva posible, evitando lógica de negocio y centrándose en la presentación de los datos.
C- Desarrollar el ViewModel: El ViewModel actúa como un intermediario entre la vista y el modelo de datos. Su función principal es preparar y exponer los datos del modelo de una manera que sea fácilmente consumible por la vista. El ViewModel puede contener métodos y propiedades para obtener los datos necesarios del modelo, formatearlos y realizar cualquier procesamiento

adicional antes de presentarlos en la vista. Además, también puede manejar eventos y acciones relacionadas con la interfaz de usuario.

D- Establecer las conexiones entre la vista y el ViewModel: En Android, esto se puede lograr mediante el uso de enlaces de datos (data binding). Los enlaces de datos permiten que los elementos de la interfaz de usuario (como TextViews, EditTexts, etc.) se enlacen directamente con las propiedades del ViewModel. Esto significa que los cambios en los datos del ViewModel se reflejarán automáticamente en la vista, y las interacciones del usuario se propagarán al ViewModel para su procesamiento.

E- Implementar la lógica de interacción: El ViewModel también se encarga de manejar la lógica de interacción de la vista, como los eventos de clic de botones o los gestos táctiles. Puede contener métodos o comandos que se activan cuando se produce una interacción y que realizan las acciones necesarias en el modelo de datos.

4- ¿Cuáles son las ventajas y desventajas de utilizar el patrón MVVM en el desarrollo de aplicaciones móviles?

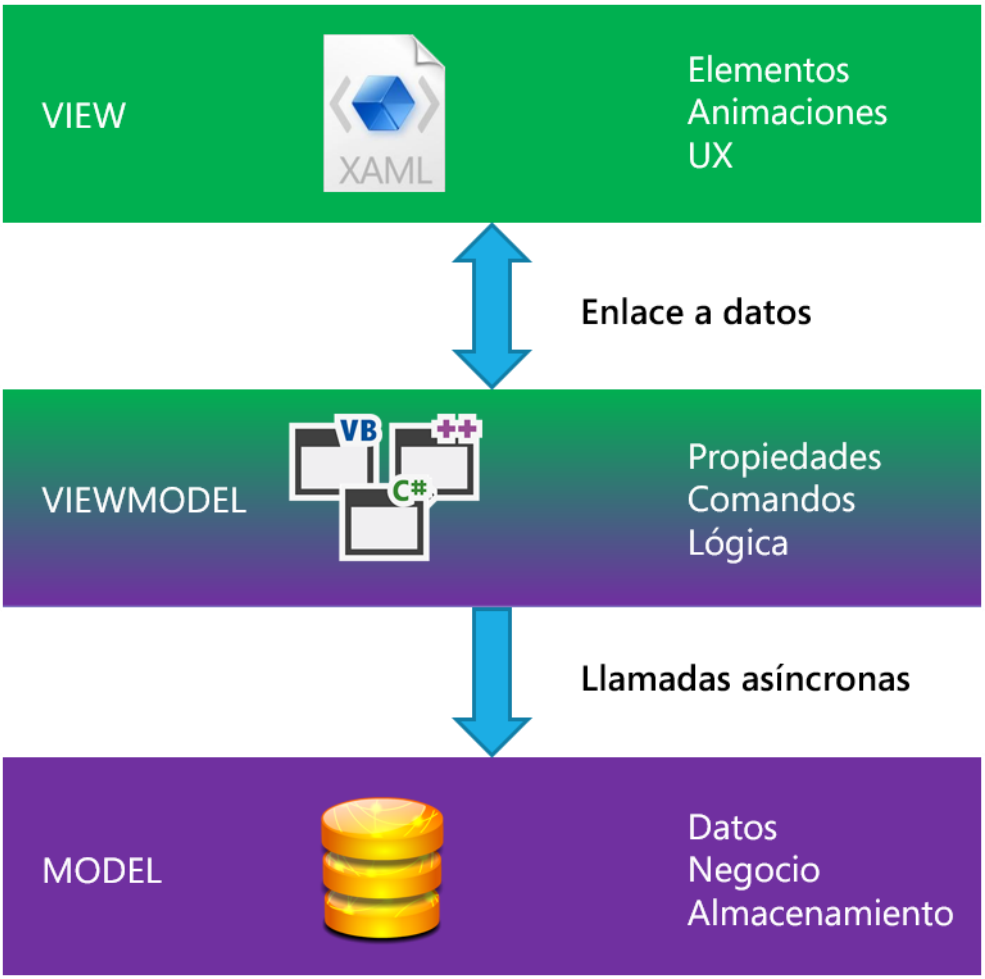
A- Ventajas

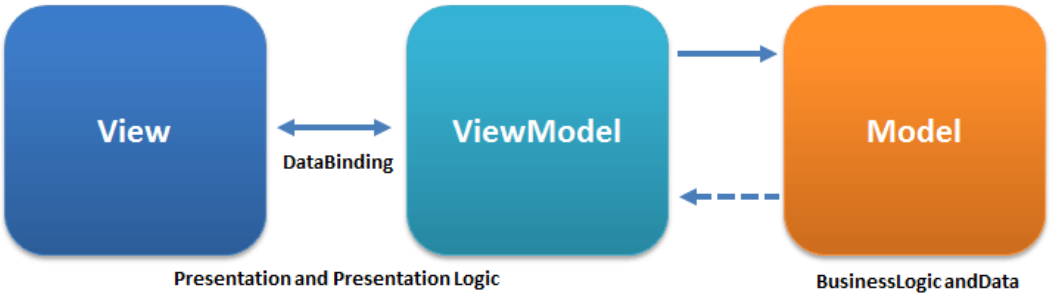
- Separación clara de responsabilidades: El patrón MVVM facilita la separación de la lógica de negocio, la presentación de datos y la interfaz de usuario, lo que hace que el código sea más organizado y mantenible.
- Testabilidad: Debido a la separación de responsabilidades, cada componente del MVVM puede ser probado de manera independiente, lo que facilita la realización de pruebas unitarias y de integración.
- Reutilización de código: El patrón MVVM promueve la reutilización de componentes, ya que el ViewModel puede ser compartido entre diferentes vistas.

B- Desventajas

- Aumento de complejidad: El patrón MVVM puede introducir una mayor complejidad en comparación con enfoques más simples, especialmente para aplicaciones pequeñas o simples.
- Curva de aprendizaje: Si no estás familiarizado con el patrón MVVM, puede llevar tiempo y esfuerzo comprender y aplicar correctamente todos sus conceptos y principios.
- Dependencia de bibliotecas: Algunas implementaciones del patrón MVVM requieren el uso de bibliotecas adicionales, lo que puede aumentar las dependencias del proyecto y el tamaño de la aplicación.
- Sobrecarga de código: En algunos casos, el uso del patrón MVVM puede llevar a una mayor cantidad de código debido a la necesidad de crear los modelos de vista y establecer las conexiones entre la vista y el modelo de vista.

Anexos


VIEW
XAML
Elementos
Animaciones
UX
Enlace a datos
VIEWMODEL
VB
C#
Propiedades
Comandos
Lógica
Llamadas asíncronas
MODEL
Datos
Negocio
Almacenamiento


View
DataBinding
ViewModel
Model
Presentation and Presentation Logic
BusinessLogic andData

Bibliografía

- https://learn.microsoft.com/es-es/dotnet/architecture/maui/mvvm

- https://devexperto.com/mvvm-vs-mvp/

- https://www.abatic.es/arquitecturas-de-software-mvc-y-mvvm/

- Libro electrónico, Patrones de aplicación empresarial con .NET