UNIVERSIDAD DON BOSCO

Desafío practico 3

Arquitectura de software MVVM

Alumnos:

-Teos Arévalo Romeo Alejandro TA191376

-López López Carlos Ricardo LL192148

Docente:

Ing. Alexander Alberto Siguenza Campos

Índice

Introducción

Existen variedad de arquitecturas en Android que nos ayuda a tener un mejor modelo de vista controladora, en el presente trabajo se hablara de la arquitectura MVVM (model-view-viewmodel), El cual tiene como propósito ayudar a los desarrolladores a abordar numeroso problemas y a poder definir la arquitectura de una aplicación, la información brindada en la investigación nos da un mejor enfoque de la funcionalidad, de sus ventajas, desventajas y los pasos a seguir para implementar un patrón de MVVM en nuestros proyectos.

Patrón MVVM

¿Qué es?

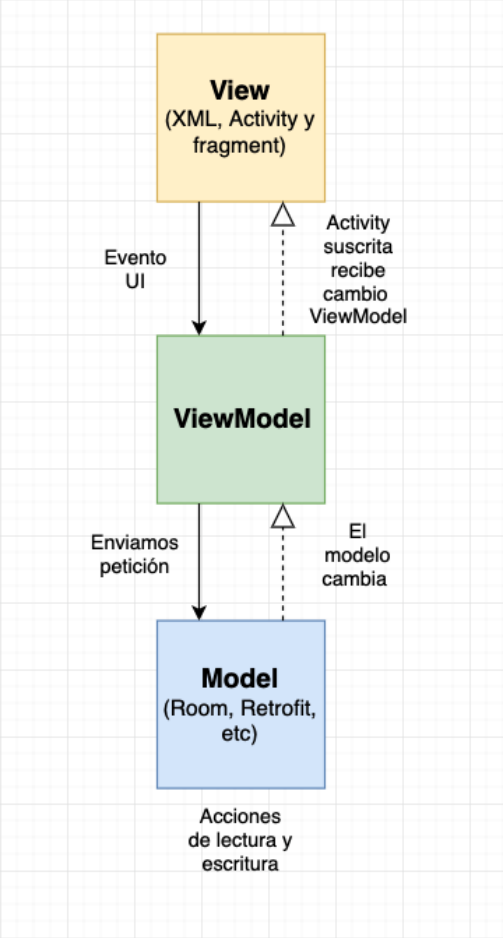
MVVM es una arquitectura desarrollada por Microsoft alrededor de 2004, cuando también se creó Windows Presentation Foundation. Esta arquitectura ha sido adoptada después por otros lenguajes y otras tecnologías, como pueden ser Java con Android o iOS con Apple, y resulta muy potente.

El patrón MVVM ayuda a separar limpiamente la lógica de presentación y negocios de una aplicación de su interfaz de usuario (UI). Mantener una separación limpia entre la lógica de la aplicación y la interfaz de usuario ayuda a abordar numerosos problemas de desarrollo y facilita la prueba, el mantenimiento y la evolución de una aplicación. También puede mejorar significativamente las oportunidades de reutilización del código y permite a los desarrolladores y diseñadores de interfaz de usuario colaborar más fácilmente al desarrollar sus respectivas partes de una aplicación.

Componentes del patrón MVVM

Hay tres componentes principales en el patrón MVVM:

- Modelo (Model), en la cual vamos a ver todo lo que serían los datos, dónde vamos a tener toda la lógica de datos.
- Vista modelo (View Models) que va a ser la encargada de interactuar tanto con el modelo como con la vista.
- La vista (Views) que va a ser la parte visual.



MMVM también nos va a permitir, con mucha facilidad, poder hacer test unitarios (Unit Tests) y poder testear mejor nuestras aplicaciones.

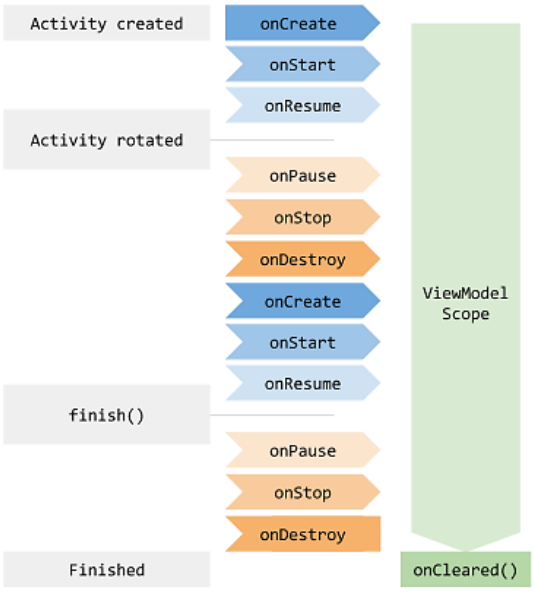
Si estáis familiarizados con el desarrollo, esta arquitectura os recordará al Modelo Vista Controlador (MVC), ya que las bases de ambas arquitecturas son muy parecidas. La diferencia principal entre ambas es que la iteración de MMVVM funciona de otra forma.

El ciclo de vida de un ViewModel

El ciclo de vida de un ViewModel está vinculado directamente a su alcance. Un ViewModel permanece en la memoria hasta que desaparece el ViewModelStoreOwner que determina su alcance. Esto puede ocurrir en los siguientes contextos:

- En el caso de una actividad, cuando termina.
- En el caso de un fragmento, cuando se desvincula.
- En el caso de una entrada de Navegación, cuando se quita de la pila de actividades.

En la Figura, se muestran los distintos estados del ciclo de vida de una actividad a medida que atraviesa una rotación y hasta que termina. La ilustración también muestra el ciclo de vida del ViewModel junto al de la actividad asociada. Este diagrama en particular muestra los estados de una actividad. Los mismos estados básicos se aplican al ciclo de vida de un fragmento.



Ventajas y desventajas usar el patrón MVVM

Ventajas:

- Si una implementación de modelo existente encapsula la lógica de negocios existente, puede ser difícil o arriesgada cambiarla. En este escenario, el modelo de vista actúa como adaptador para las clases de modelo y evita que realice cambios importantes en el código del modelo.
- Los desarrolladores pueden crear pruebas unitarias para el modelo de vista y el modelo, sin usar la vista.

- Los diseñadores y desarrolladores pueden trabajar de forma independiente y simultánea en sus componentes durante el desarrollo. Los diseñadores pueden centrarse en la vista, mientras que los desarrolladores pueden trabajar en el modelo de vista y los componentes del modelo.
- El mantenimiento de los sistemas es simplificado.

Desventajas:

- La curva de aprendizaje para nuevos desarrolladores es un poco superior a los otros modelos que son más simples.
- La distribución de componentes nos obliga a la creación y mantenimiento de un mayor número de ficheros.
- Debes de adaptarte a una estructura predefinida y eso incrementa la complejidad del sistema.

patrón MVVM en kotlin Android

¿cuáles son los pasos que debemos tener en cuenta para la implementación de MVVM en Android?

- Crear un nuevo proyecto en Android Studio según las necesidades de tu app, por ejemplo, el lenguaje de programación kotlin y el SDK que se utilizara.

- Crear la clase Modelo con el fin de que el programa sepa la estructura que va a definir la presentación de los datos de tu app.

- Trabajar con el archivo activity_main.xml para establecer las diferentes entradas y componentes de la app.

- Crear la clase ViewModel para indicar los métodos que se deben llamar para el buen funcionamiento de la app.

- Definir las funcionalidades de View en el fichero de MainActivity y ejecutar la app.

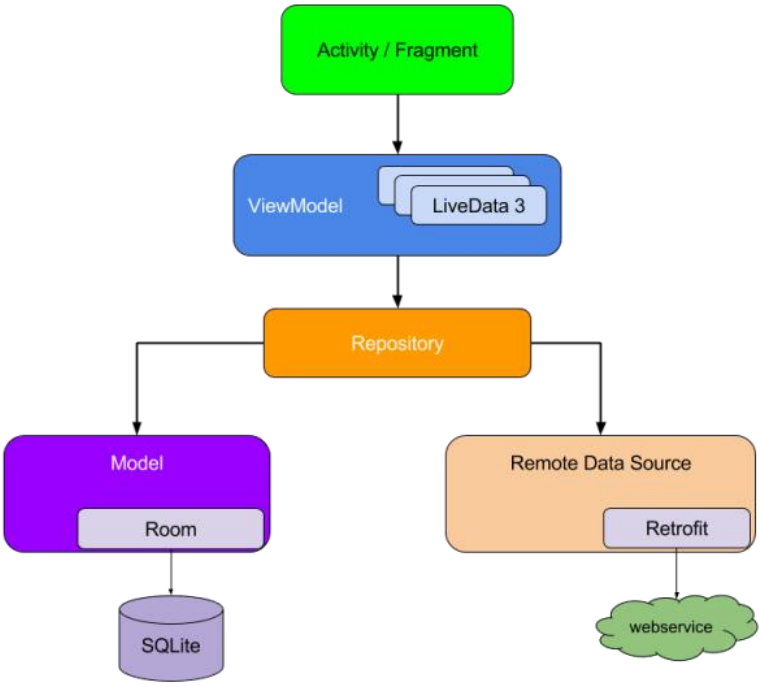
Android nos da una colección de librerías que nos pueden ser muy útiles a la hora de querer mejorar el desarrollo de nuestras apps llamadas Componentes de la arquitectura, en este caso nos centraremos en dos de las clases que estas librerías nos ofrecen.

ViewModel:

Esta clase será el intermediario entre nuestra vista y nuestra lógica del negocio, es la encargada de almacenar la información de la interfaz gráfica, ya que una de sus ventajas más grandes es que no se destruye en el cambio de orientación de nuestra aplicación.

LiveData:

Esta clase nos sirve para compilar objetos de datos que nos permitirán notificar cuando algún valor sea modificado por la parte de la lógica del negocio, logrando con esto que la interfaz se entere y haga los ajustes necesarios.



Anexos

- https://learn.microsoft.com/es-es/dotnet/architecture/maui/mvvm
- https://inmediatum.com/blog/ingenieria/mvvm-que-es-y-como-funciona/
- https://cursokotlin.com/mvvm-en-android-con-kotlin-livedata-y-view-binding-android-architecture-components/
- https://openwebinars.net/blog/la-arquitectura-mvvm-y-sus-componentes/