UNIVERSIDAD DON BOSCO
DESARROLLO DE SOFTWARE PARA MÓVILES
CICLO 01-2023

Trabajo de investigación

INTEGRANTES:

Carlos David Herrera Guardado	-	HG190072
Diego Alejandro Velásquez Gómez	-	VG190501
Irma Gabriela Hernández Martínez	-	HM190181

DOCENTE:

Alexander Alberto Siguenza Campos

29 de abril de 2023

Soyapango, San Salvador

Tabla de contenido

Introducción

Al vivir en un mundo globalizado e interconectado surge la necesidad de crear modelos para la fabricación y realización de productos y servicios, para que estés puedan ser utilizadas en cualquier país y que no se presenten problemas en su utilización.

En el presente trabajo se hablará acerca de estos estándares aplicados al desarrollo móvil, a través de un ejemplo práctico, para así conocer los elementos principales y las ventajas que pueden representar para facilitar el desarrollo y sentar bases para que el mantenimiento de estos sistemas se facilite y pueda ser entendido por futuros equipos.

Al mismo tiempo se explicará algunos de los inconvenientes que estos estándares pueden presentar y que deben ser considerados al momento de escogerlos, para saber si cumplen con los requisitos necesarios para ser aplicados y moldeados al proyecto que se esté desarrollando.

Patrón MVVM

El modelo MVVM o Modelo-Vista-Vista-Modelo, consiste en un estándar para el desarrollo de sistemas, con el objetivo de documentar las funcionalidades y el objetivo del sistema, al mismo tiempo que se ordena el código y se vuelve el eficaz el avance del proyecto, y se garantiza que será entendido por futuros equipos que busquen brindar mantenimiento o añadir funcionalidades y que se convierta en una tarea compleja.

Este modelo guarda similitudes con el modelo MVP, principalmente por la manera en que el código es estructurado, con la diferencia de enlace de datos bidireccionales, más adelante se detalla en que consiste este tipo de enlace.

A continuación, se muestra un diagrama para describir las relaciones que se crean bajo este modelo.

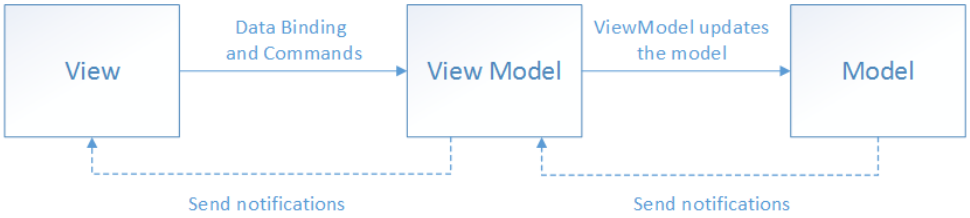

Ilustración 1-Diagrama del modelo MVVM

Componentes principales

Existen tres componentes principales, las cuáles son:

1. **Modelo:** Está capa se encarga únicamente de la conexión a la base de datos.

Por lo tanto, se puede decir que el modelo, es quien maneja la lógica del negocio, debido a esto es que nace la separación en tres capas, de ser necesarios cambios, se intervendrían de manera separada sin poner en riesgo a las demás y en especial al modelo.

2. **Vista:** Está capa es con la que el usuario interactúa, que recibe las indicaciones y datos que serán tratados y también se encarga de presentar los datos que sean solicitados.

La vista define el diseño que verá el usuario y le da la estructura al sistema, para que sea más fácil la interacción. Por ende, es de vital importancia que sea interactiva con las acciones del usuario, como si hay una carga pendiente, deshabilitar opciones que no estés disponibles para su rol o porque hay un proceso en segundo plano que debe ser previamente terminado.

3. **Modelo de vista:** También conocida como *'contenedor de estado a nivel de pantalla'*, esta capa se encargar de crear una barrera entre el modelo y la vista, recibe los datos de la vista, los verifica y traduce para que estos puedan ser recibidos por el modelo, de igual manera se pueden realizar cambios que afecten directamente a la vista.

Por ende, esta capa está encargada de la coordinación de las llamadas o datos que recibe de la vista y que luego de ser procesadas son enviadas al modelo, se mantiene en comunicación con esta a través del envió de notificaciones, manteniendo de esta manera la información sincronizada, siendo esto el enlace de datos bidireccionales que mencionábamos al inicio.

También procesa la información del modelo y la agrupa de manera en que sea entendible y mejor manejable para el usuario y se presente de manera clara a través de la vista.

Bajo está estructura cabe aclarar que la vista es consciente del modelo de vista y esté es consciente del modelo, pero no al contrario y cada uno evoluciona sin necesitar de su predecesor.

MVVM en Android con Kotlin

Android y Kotlin cuentan con enlaces que permiten implementar de manera eficaz el modelo MVVM, con el objetivo de contar con aplicaciones con código ordenado y de fácil mantenimiento, en especial en móviles donde continuamente hay actualizaciones y nuevos requerimientos que obligan a constantemente actualizar y mejorar el código para seguir está línea de crecimiento.

Persistencia. Es una de las principales ventajas de la implementación de este modelo. Consiste en el almacenamiento en caché del estado, esto permite conservarlos durante cambios, como en el cambio de actividades, previniendo que se recargue y convirtiendo más eficiente la aplicación, ya que de lo contrario se debería crear una clase similar al view model y utilizar otras alternativas de almacenamiento para prevenir la pérdida de datos.

Ciclo de vida de un ViewModel

La duración de un ViewModel depende del alcance al que esté destinado, y esto es definido a través del ViewModelStoreOwner, por ende, se pueden definir los siguientes escenarios:

1. En una actividad, cuando está finaliza.
2. En un fragmento, cuando se desvincula.
3. Entrada de Navigation, cuando se termina la pila de actividades.

Por ende, los datos se mantienen vivos y no requieren una recarga continua, mejorando la experiencia del usuario.

Ventajas y Desventajas

Ventajas

- Permite evitar cambios drásticos en los sistemas, debido a que el modelo de la vista funge la función de traductor, este percibe la información de la vista y acomoda o traduce los datos para el modelo, así si la organización crece, únicamente el modelo de la vista sería cambiado, pero no el modelo principal.
- Conserva de manera más eficiente el estado de la IU.

- Permite desarrollar pruebas unitarias sin tomar en cuenta la vista.
- La aplicación puede ser rediseñada sin necesidad de tocar las otras capas.
- Mejor acceso a la lógica empresarial.

Desventajas

- Es complicado aplicar este modelo en aplicaciones ya desarrolladas.
- Dependiendo la complejidad del proyecto puede provocar que las pruebas unitarias crezcan y no pueden ser replicadas fácilmente en otras activity.
- Puede complicarse la aplicación de medidas de seguridad en las capas del modelo, en especial con la vista del modelo que no está diseñada o pensada para esta tarea.

Ejercicio práctico

https://github.com/diegovelasquez-g/DSM-Investigacion-MVVM

Bibliografía

Descripción general de ViewModel. (s. f.). *Android Developers*. https://developer.android.com/topic/libraries/architecture/viewmodel?hl=es-419

KeepCoding, R. (2022, 23 diciembre). ¿Qué es MVC, MVP y MVVM en Android? *KeepCoding Bootcamps*. https://keepcoding.io/blog/que-es-mvc-mvp-y-mvvm-en-android/

Kexugit. (2015, 23 marzo). *Patrones de diseño: Problemas y soluciones con Model-View-ViewModel*. Microsoft Learn. https://learn.microsoft.com/es-es/archive/msdn-magazine/2010/july/design-patterns-problems-and-solutions-with-model-view-viewmodel

Michaelstonis. (2022, 28 noviembre). *Modelo-Vista-Modelo de vista*. Microsoft Learn. https://learn.microsoft.com/es-es/dotnet/architecture/maui/mvvm