Modelo Vista – Vista Modelo

Universidad Don Bosco
Ingeniería de Software

Docente
Ing. Alexander Alberto Siguenza Campos

Presentado por:
Bonilla Avilés, David Alejandro – BA181927
Cruz González, José Roberto – CG181933
Garay Alvarado, Bryan Walberto – GA181935

Tabla de contenido

Introducción

En programación existen varias formas de adecuar lo que se está trabajando en un enfoque más estructurado y ordenado, estas formas de adecuar y trabajar el código nacen debido a la tendencia de los programas a crecer exponencialmente en tamaño y complejidad dificultando en gran medida su edición y actualización a través del tiempo. Para corregir este comportamiento, nacieron patrones que ayudan al software a estructurarse de mejor manera y facilitar la forma de entender y editar el código realizado, estos fueron llamados patrones, estos patrones a su vez son catalogados de diferente manera pueden ser patrones de diseño y patrones de arquitectura.

Los patrones de diseño hacen énfasis en cada uno de los elementos que componen el software realizado, describiendo perfectamente su función y la forma en la que estos se comunican entre sí, definiendo además reglas de uso para restringir sus responsabilidades, de esta manera formas de entrelazar y comprender el código se vuelven estándar y con seguridad de funcionamiento al haber sido utilizados a lo largo del mundo.

Adaptar alguna de estas tecnologías en el FrontEnd para móviles ha sido un reto importante para los desarrolladores, principalmente por cómo se desarrolla la vista y la forma en la que las plataformas establecieron las formas de pintar la vista y comunicarse a nivel de código.

Para facilitar este esquema mayoritariamente en este tipo de programación para móviles se ha venido optando por el patrón MVVM (Modelo Vista Vista-Modelo) que inicialmente suena confuso, pero a lo largo de este reporte se explicará de manera detallada.

Contenido

¿Qué es el patrón MVVM?

El patrón MVVM (Modelo Vista Vista-Modelo), como se describía en la introducción es un patrón de diseño que permite la reutilización de una lógica establecida para la comunicación entre los componentes de software creados, por ejemplo, estructuras de datos, layouts, acciones, mutaciones, etc. La forma de categorizar y organizar todo este tipo de elementos es lo que facilita un patrón de diseño.

El patrón MVVM separa la lógica de la aplicación en tres simples escenarios, bastante parecido a un MVC (Modelo Vista Controlador) pero mucho mejor adoptado para herramientas de desarrollo para móviles como es el caso de Kotlin.

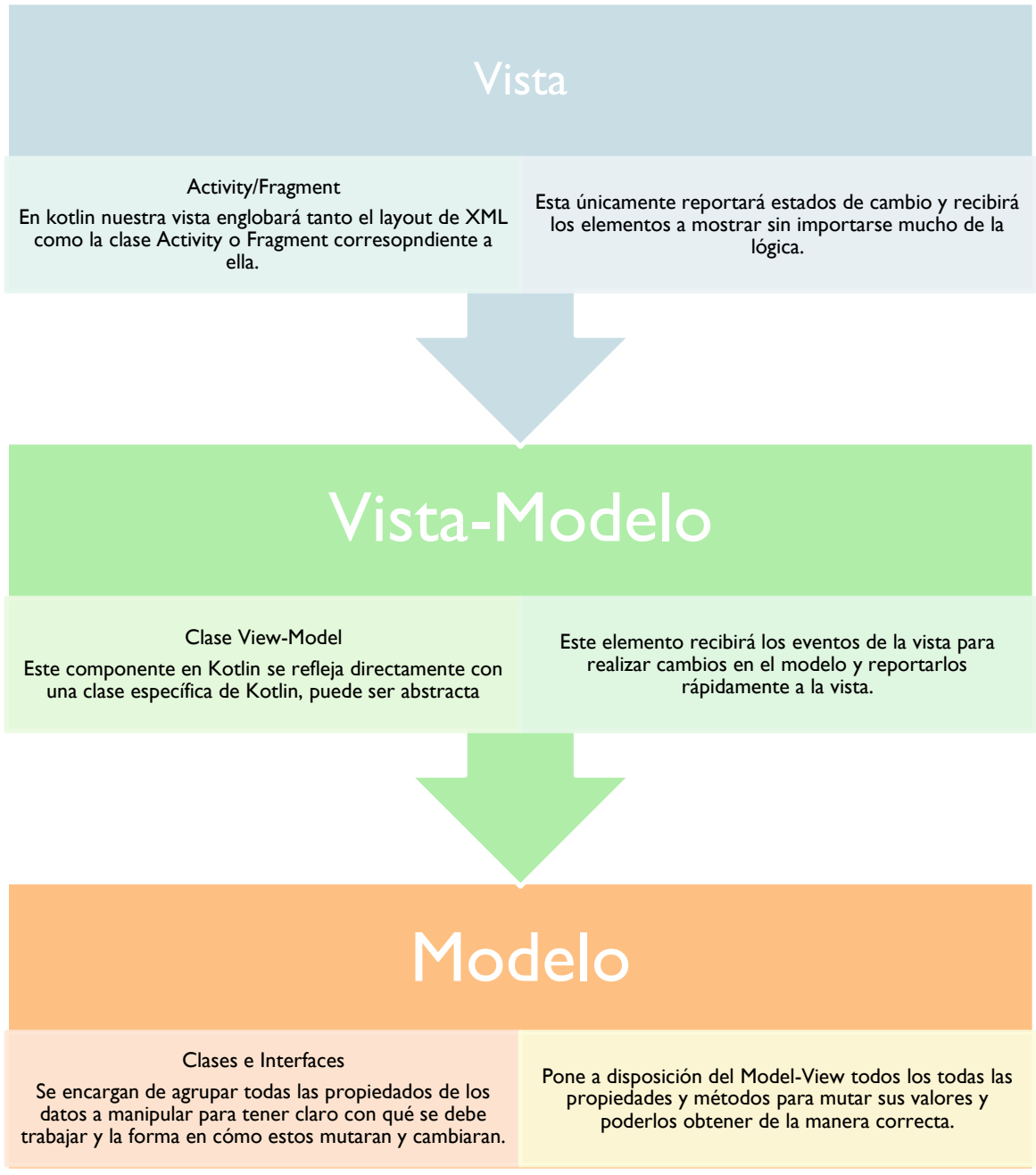
¿Cuáles son los elementos principales y cómo se relacionan entre sí?

Los escenarios y la forma en la que se compone MVVM es:

- **Model:** Modelo que se encarga de ser la representación de la data proveniente de otros sistemas o incluso del mismo sistema, se refiera a datos de tipo estructurales, que suele basarse en la programación orientada a objetos.
- **View:** Es la representación de la vista por completo que se le muestra al usuario es fácil definir muchas veces este componente, por ejemplo, un código php se sabe perfectamente que el controlador enlazará los modelos de la base de datos con código HTML y precisamente estos templates o layouts son los que se definen como vista. En el caso de Android esto es mucho más difícil, debido a que la vista está fuertemente ligada a los Activity que son en sí un controlador mismo del flujo de la aplicación.

- **View-Model**: Este hace referencia al elemento del software que hace posible la presentación de datos en la vista a través del modelo, se encarga de manejar los estados tanto del modelo como de la vista.

Ahora, sabiendo estos elementos es importante resaltar sus roles y responsabilidades, además de la forma en cómo estos se comunican.

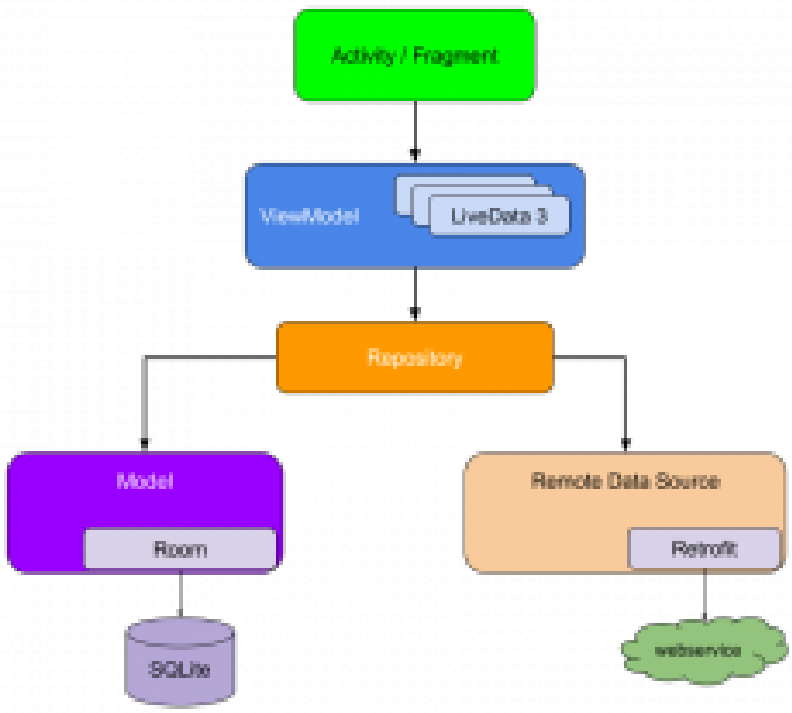


¿Cómo se aplica en Android con Kotlin?

En el apartado anterior se detallan qué clases corresponden a cada elemento del patrón, pero se extenderá más en el cómo en el siguiente apartado.

La forma en la que interactúan los componentes hace mucho más sentido al modelo MVVM en el desarrollo de móviles, debido a que engloba los archivos de Activity/Fragment en la vista, dando mucho más control del manejo de datos y forma en la que se pintan los elementos.

Esta vista puede acceder al View-Model únicamente accede al ViewModel para obtener los datos que necesita para pintar, no debe realizar ninguna acción o ejecutar algún método que permita cambiar el estado de los datos, debido a que su única interacción luego de obtener datos es reportar los eventos que suceden en la ejecución al ViewModel.



El ViewModel es el encargado de actualizar en todo momento los datos dependiendo de lo que sucede en la vista e inclusive de lo que pueda suceder en el modelo en conexión síncronas, por ejemplo. En cada evento reportado se encarga de mutar el estado del modelo.

El Modelo es un estado sin ninguna acción, este dependerá de cómo sea plateado su utilización porque es flexible en su forma de ser creado, se puede auxiliar de otros patrones de diseño para dar vida al modelo y mantener un estado sincronizado entre las diferentes **"DataSources"** que hay en la aplicación para aislar la capa de datos del modelo proporcionado a mvvm.

Ventajas y Desventajas

Ventajas

- La utilización del patrón MVVM permite seguir utilizando el flujo normal de trabajo en la programación de aplicaciones Android.
- Es fácil de implementar en Kotlin.
- Se descentraliza la utilización de un modelo y se pone a disposición de cualquier vista.
- La lógica de negocio para la presentación de los datos se aleja de la vista.

Desventajas

- La forma de lograr un **"testing"** adecuado en la aplicación, dependerá de la forma en la que se presenten los datos en la vista, entre mayor dependencia tenga la vista del ViewModel más complicado se volverá realizar pruebas unitarias.

Bibliografía

Ramos, J. (no date) *Android: ¿Qué es MVC, MVP Y MVVM?*, *Programación y más*. Available at: https://programacionymas.com/blog/android-mvc-mvp-mvvm (Accessed: April 29, 2023).

Reyes, L. (2018) *Aplicando El Patrón de Diseño MVVM*, *Medium*. Medium. Available at: https://medium.com/@reyes.leomaris/aplicando-el-patr%C3%B3n-de-dise%C3%B1o-mvvm-d4156e51bbe5 (Accessed: April 29, 2023).

Rodriguez, E. (2023) *MVVM - Qué es y como funciona.*, *INMEDIATUM*. Available at: https://inmediatum.com/blog/ingenieria/mvvm-que-es-y-como-funciona/ (Accessed: April 29, 2023).