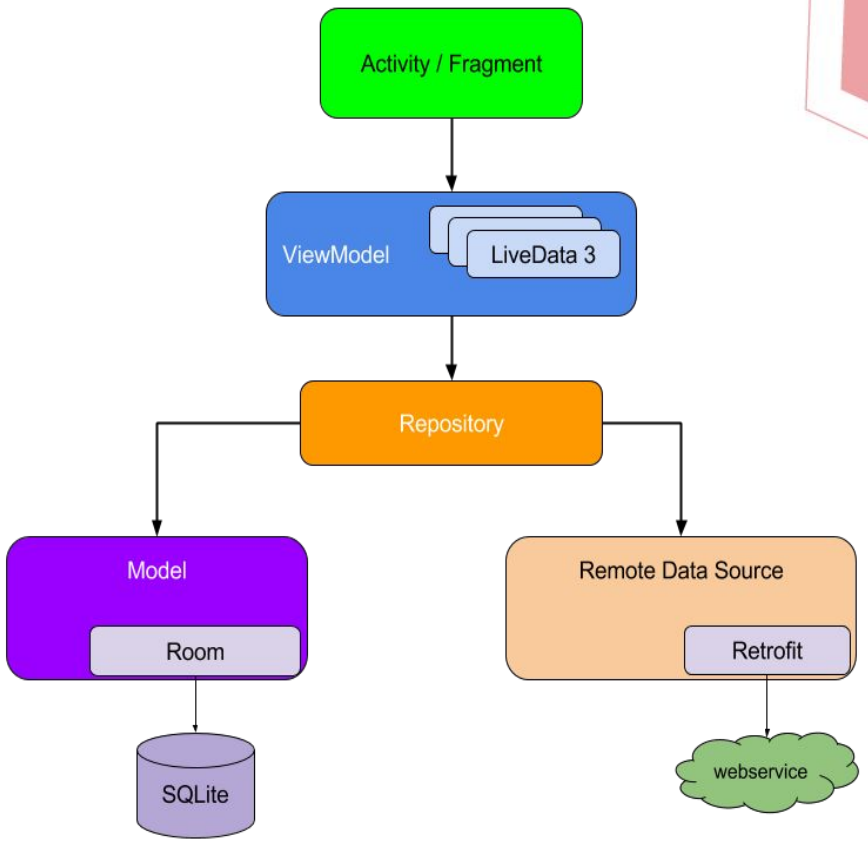
Desarrollo de Aplicaciones para Dispositivos Móviles

Ingeniero Walter Medina
walter.medina@correounivalle.edu.co

Arquitectura MVVM

Model-View-ViewModel, es un patrón arquitectónico utilizado en el desarrollo de software, y ha sido adoptado especialmente en el desarrollo de aplicaciones Android. Este patrón se diseñó para mejorar la modularidad del código, facilitando así el mantenimiento y la escalabilidad de las aplicaciones.



Model: Representa los datos en la aplicación.Los modelos no deben tener conocimiento de la interfaz de usuario.Pueden incluir acceso a base de datos, servicios web, u otras fuentes de datos.

View: Es responsable de la presentación de la interfaz de usuario y de interactuar con el usuario. En Android, la Vista se asocia generalmente con las actividades y fragmentos. La Vista solo debería manejar la lógica relacionada con la interfaz de usuario.

ViewModel: Conectan al Model con la View, la View se suscribe a su respectivo ViewModel y estos por medio de los **Live Data** se percatan que el modelo ha cambiado, por lo tanto se lo notifican a la View. Su propósito principal es manejar la lógica de la aplicación

Repository: Actúa como una capa de abstracción o puente entre los datos de la aplicación que puede ser una base de datos o un servicio web y el ViewModel.

LiveData

LiveData es una clase de contenedor de datos observables. LiveData está optimizado para ciclos de vida, lo que significa que respeta el ciclo de vida de otros componentes de las apps, como actividades, fragmentos o servicios.

La principal característica de LiveData es que puede adaptarse automáticamente a los cambios en los datos subyacentes y notificar a los observadores registrados cuando ocurren estos cambios. Esto facilita la actualización de la interfaz de usuario en respuesta a cambios en los datos.

Base de Datos Local usando Room

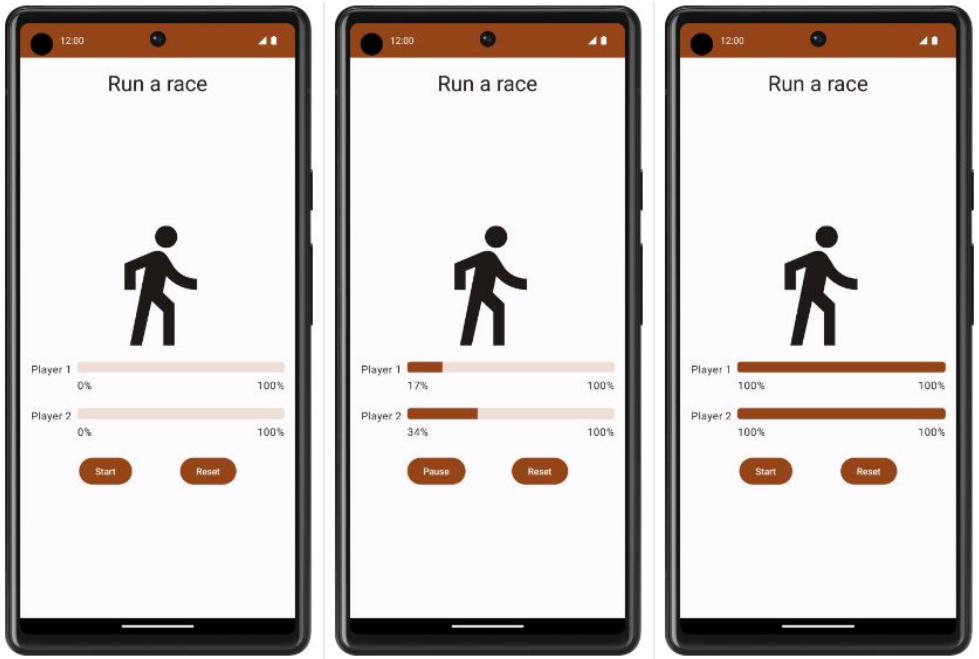
Las apps que controlan grandes cantidades de datos estructurados pueden beneficiarse con la posibilidad de conservar esos datos localmente, por medio de la **base de datos SQLite.**

La biblioteca de persistencias Room brinda una capa de abstracción para SQLite que permite acceder a la base de datos sin problemas y, al mismo tiempo, aprovechar toda la tecnología de SQLite.

Programación de tareas en segundo plano (corrutinas)

Una corrutina es un patrón de diseño de simultaneidad que puedes usar en Android para simplificar el código que se ejecuta de forma asíncrona.

En Android, las corrutinas ayudan a administrar tareas de larga duración que, de lo contrario, podrían bloquear el subproceso principal y hacer que tu app dejara de responder. Más del 50% de los desarrolladores profesionales que usan corrutinas informaron que vieron un aumento en la productividad.



Tarea: Leer sobre corrutinas

- https://developer.android.com/codelabs/basic-android-kotlin-compose-coroutines-kotlin-playground?hl=es_419#0
- https://developer.android.com/kotlin/coroutines?hl=es-419

GRACIAS !!

“El esfuerzo vence al talento cuando el talento no se está esforzando”.
Anónimo

Bibliografía:

- https://developer.android.com/guide/topics/ui/layout/recyclerview?hl=es-419#:~:text=RecyclerView%20es%20el%20ViewGroup%20que,un%20objeto%20contenedor%20de%20vistas.
- https://developer.android.com/training/data-storage/room?hl=es-419
- https://developer.android.com/codelabs/basic-android-kotlin-compose-coroutines-android-studio?hl=es_419#3
- https://developer.android.com/kotlin/coroutines?hl=es-419
- https://developer.android.com/topic/libraries/architecture/livedata?hl=es-419