***UNIVERSIDAD DON BOSCO***


***FACULTAD DE INGENIERÍA***

***ESCUELA DE COMPUTACIÓN***

***ASIGNATURA:***

Desarrollo de Software para Móviles.

***ACTIVIDAD:***

Trabajo de investigación

***Integrantes:***


| Alumno: | Carné: |
| --- | --- |
| Santos Ronaldo Lemus Torres | LT191211 |

# Índice

## Que es el patrón MVVM

MVVM (Model-View-ViewModel) es un patrón de arquitectura de software que separa la interfaz de la lógica (Model) con el objetivo de que todo aspecto visual (View) sea completamente independiente.

En el patrón MVVM, el modelo representa los datos y la lógica empresarial, la vista representa la interfaz de usuario y ViewModel actúa como intermediario entre el modelo y la vista.

El modelo de vista es responsable de administrar el estado de la vista, exponer los datos y comportamientos del modelo a la vista y manejar la entrada del usuario desde la vista.

El uso de este patrón permite separar las responsabilidades de cada componente de la aplicación, lo que facilita su mantenimiento y su desarrollo, y también facilita la realización de pruebas unitarias de cada componente.

En pocas palabras, el patrón MVVM es una arquitectura útil para el desarrollo de aplicaciones, que permiten la separación clara de las responsabilidades de cada componente de la aplicación y mejora la calidad del código y la experiencia del usuario.

## Principales componentes

Hay tres componentes principales los cuales son:

- View: Representa todo el apartado visual
- Model: Representa los datos y la lógica del programa.
- ViewModel: El recurso de ViewModel, destaca como el componente que se encargará de servir como puente entre la interacción de la Vista (View) y el Modelo (Model).

(keepcoding, keepcoding, 2023)

## Implementación de MVVM con Kotlin

Implementar este patrón no es nada fuera de lo común si hemos trabajado con MVC ya que solo crearemos los componentes necesarios, los pasos serían los siguientes.

- **Crear el proyecto en Android Studio** según las necesidades de tu app, por ejemplo, el lenguaje de programación (Kotlin) y el SDK que vas a usar.
- **Crear la clase Modelo** con el fin de que el programa sepa la estructura que va a definir la presentación de los datos.
- **Trabajar con el archivo activity_main.xml** para establecer las diferentes entradas y componentes.
- **Crear la clase ViewModel** para indicar los métodos que se deben llamar para el buen funcionamiento.
- **Definir las funcionalidades** de View en el fichero de MainActivity.
- **Ejecutar la app.**
  (keepcoding, 2022)

## Ventajas y desventajas

### Ventajas:

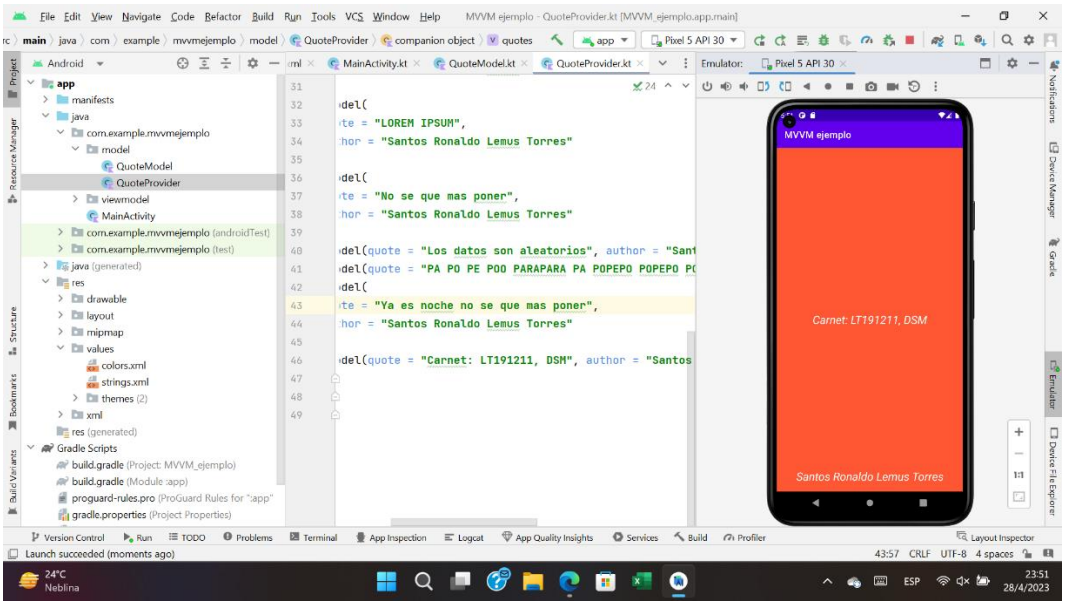
- Admite el trabajo independiente del modelo y las vistas.
- Mejora el mant5enimiento
- Simplifica las pruebas
- Permite un estilo de codificación exploratorio iterativo. Los cambios aislados son menos arriesgados y es más fácil experimentar con ellos.
-

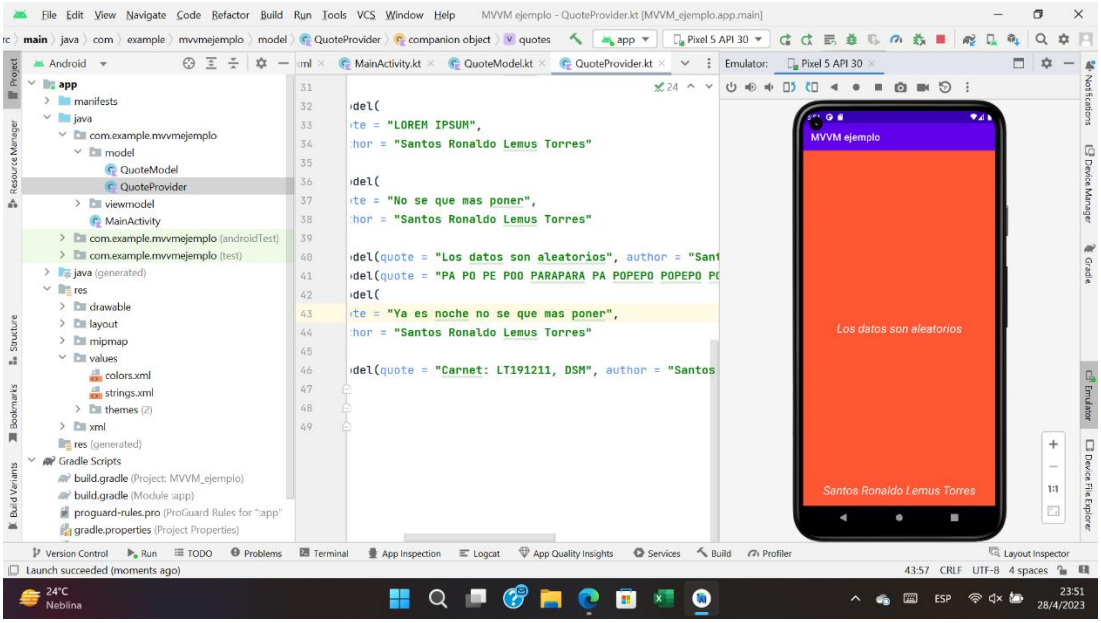
### Desventajas:

- Es difícil adaptar aplicaciones ya hechas a este patrón
- Mayor dedicación al proceso inicial de desarrollo

(learn microsoft, 2022)

# Anexos




*1*

[1] Imágenes de la aplicación Android realizada

# Bibliografía

keepcoding. (23 de Diciembre de 2022). *keepcoding*. Obtenido de keepcoding: https://keepcoding.io/blog/pasos-para-la-implementacion-de-mvvm-en-android/

keepcoding. (21 de Abril de 2023). *keepcoding*. Obtenido de keepcoding: https://keepcoding.io/blog/que-es-el-patron-de-arquitectura-mvvm/#:~:text=El%20patr%C3%B3n%20de%20arquitectura%20MVVM%2C%20tambi%C3%A9n%20conocido%20como%20Model%20View,la%20parte%20l%C3%B3gica%20(Model).

*learn microsoft*. (24 de Septiembre de 2022). Obtenido de learn microsoft: https://learn.microsoft.com/es-es/windows/uwp/data-binding/data-binding-and-mvvm