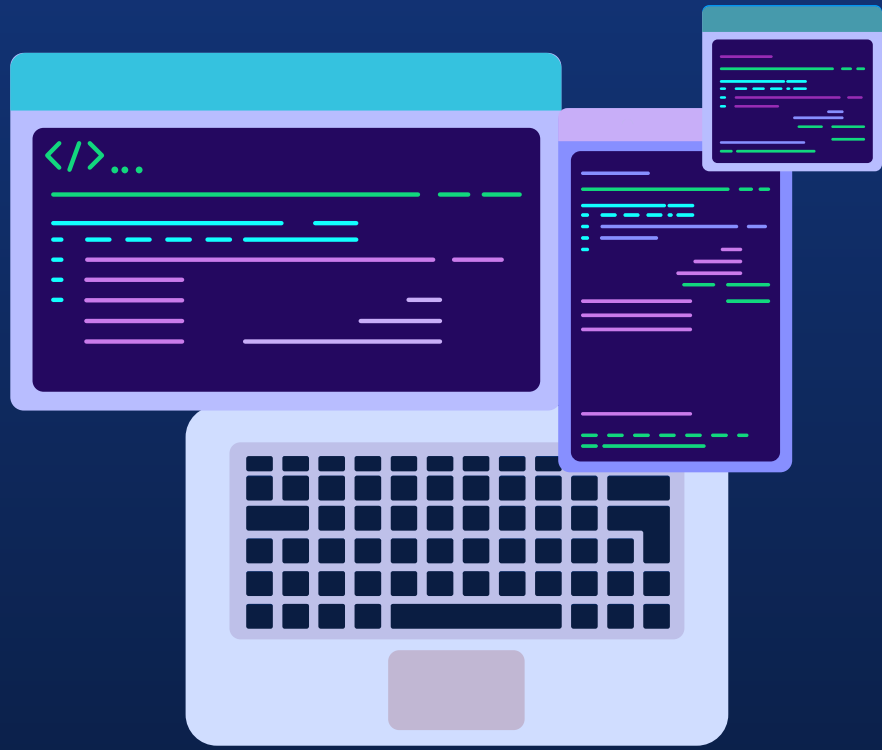
ANDROID ECOMMERCE APP

Ramón García Verjaga
José Alberto Gómez García



ÍNDICE

Visión general de la funcionalidad.

CATÁLOGO

Listado de productos disponibles en el eCommerce que, a través de la API, un administrador puede **crear**, **eliminar** y **actualizar**; y que cualquier **cliente** puede **consultar**.

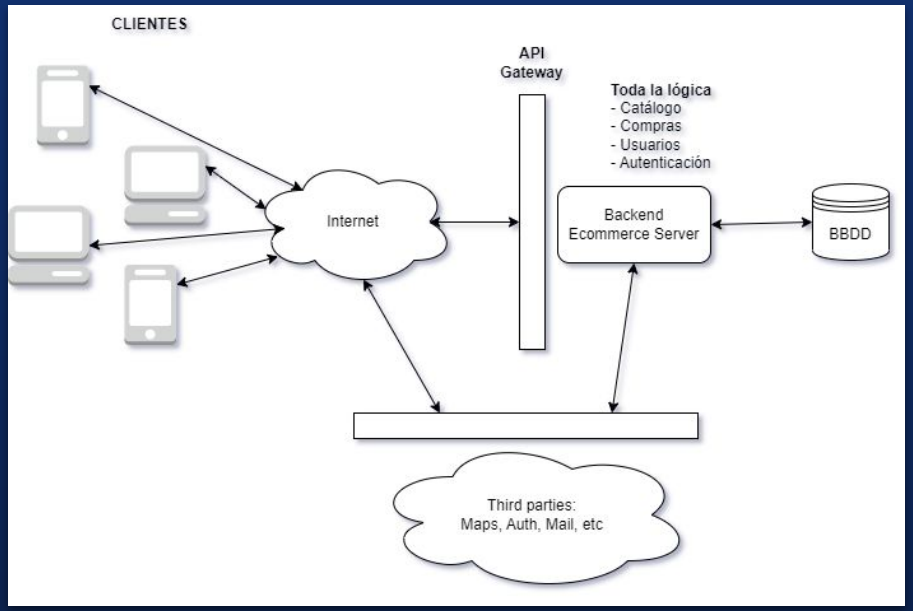
CARRITO

Productos añadidos al **carrito** por el **cliente identificado**.
Adición y **eliminación** de productos.
Posibilidad de comprar los productos.

MAPS

Solicitud de **permisos** para obtener la **localización** actual del **dispositivo**.
Identificación de la **tienda** en la que se **venden** los productos.

Arquitectura del software. Sistema actual



Backend como un monolito

Cliente y servidor débilmente acoplados a través de API

Servicios externos

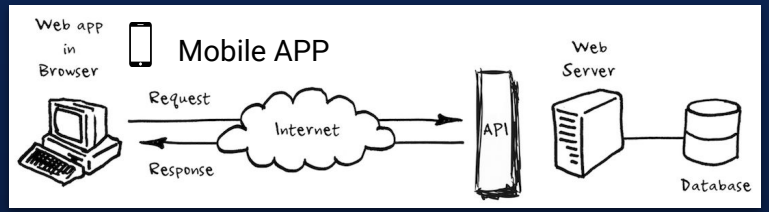


Imagen obtenida de https://cimt.nl/api-services-introduction/

Arquitectura del software. Model View ViewModel (MVVM)

Las capas de código separadas de MVVM son:

- **Model**: Esta capa es responsable de la abstracción de las fuentes de datos. Modelo y ViewModel trabajan juntos para obtener y guardar los datos.
- **View**: El propósito de esta capa es informar al ViewModel sobre la acción del usuario. Esta capa observa el ViewModel y no contiene ningún tipo de lógica de aplicación.
- **ViewModel**: Se encarga de acceder a los datos y realizar las transformaciones necesarias para mostrarlas en la vista. Expone aquellos flujos de datos que son relevantes para la Vista. Además, sirve de enlace entre el Modelo y la Vista.

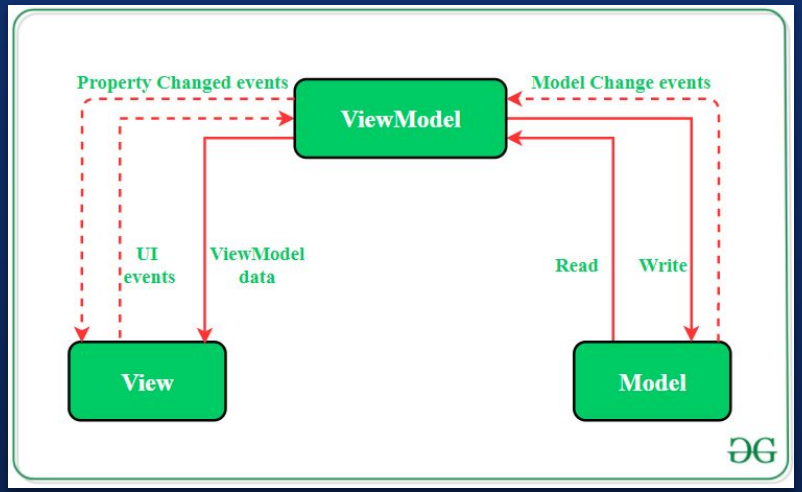
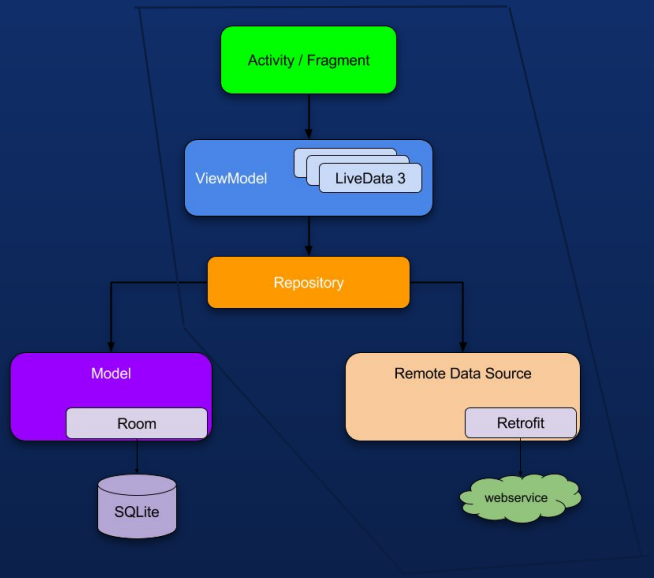


Imagen obtenida de https://www.geeksforgeeks.org/mvvm-model-view-viewmodel-architecture-pattern-in-android/

Arquitectura del software. Model View ViewModel



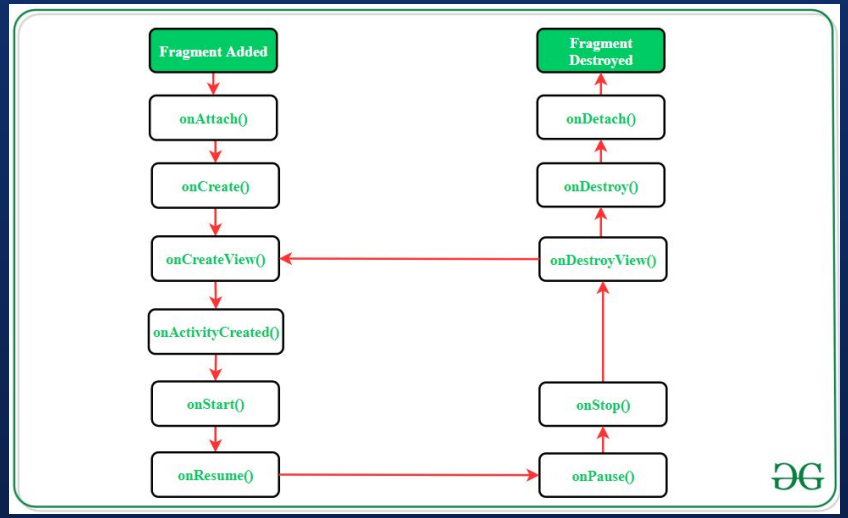


Imagen obtenida de
https://sberoch.medium.com/arquitectura-en-android-hilt-mvvm-kotlin-coroutines-live-data-room-y-retrofit-68bf458ec76e

Patrones de diseño

Adapter	Subclase responsable de proporcionar vistas que representan elementos de un conjunto de datos (estructural).
Fachada	Acceso a información a través de APIs (estructural).
Observer	Actualización de las vistas de forma reactiva en respuesta a eventos (comportamiento).
Singleton	Única instancia de una clase, por ejemplo, de un servicio inyectable (creacional).

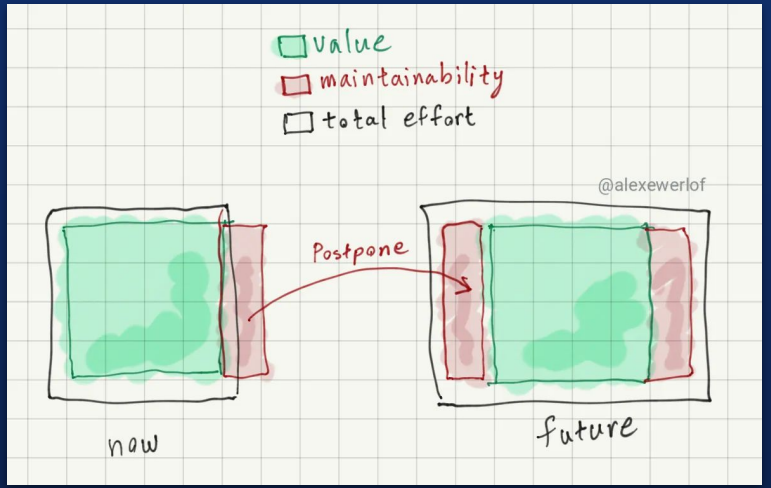
Robustez y mantenibilidad

Robustez

- Manejo de errores y alternativas al happy path
- Reenvío de peticiones (API)

Mantenibilidad (alto grado - app simple)

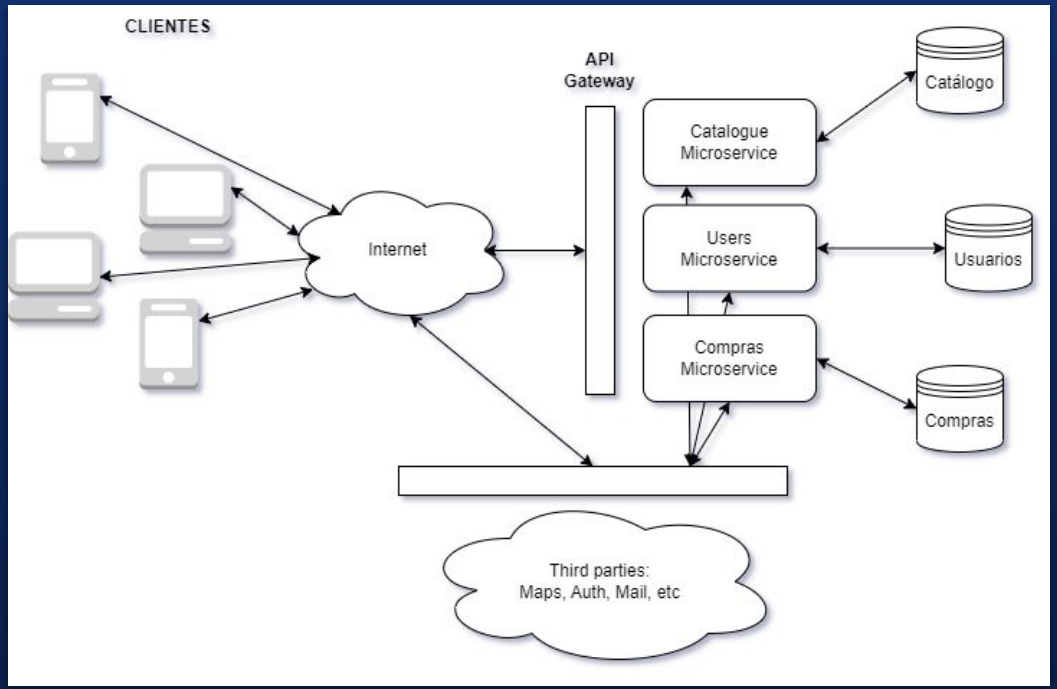
- Reparabilidad
- Evolucionabilidad



Reducir «Deuda técnica»

Imagen obtenida de https://blog.alexewerlof.com/p/tech-debt-day

Futuro. Evolución I. Arquitectura de microservicios



Futuro. Evolución II. Arquitectura orientada a eventos

OBSERVER

Comunicación **reactiva** entre componentes

- Publicación de eventos
- Consumo de eventos

Facilitar el testeo

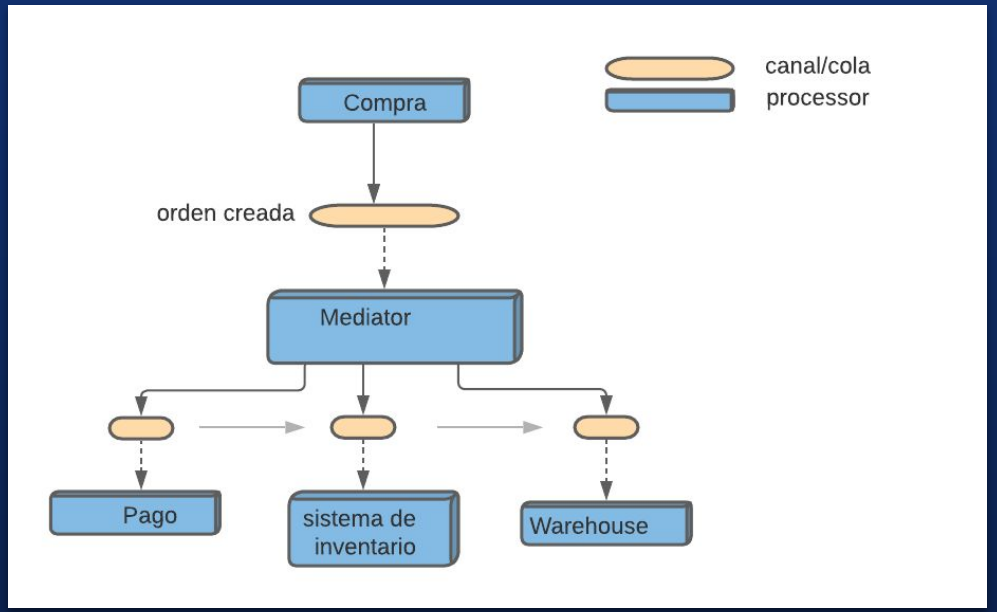


Imagen obtenida de https://hectorpoblete.cl/2021/04/02/arquitectura-orientada-eventos-y-sistemas-complejos/

EXPLICACIÓN DE LA IMPLEMENTACIÓN

DEMOSTRACIÓN

Ejecución de la aplicación, visualización del catálogo de productos y del mapa

BIBLIOGRAFÍA

- Manuel I. Capel. 2022. **Material académico proporcionado en la asignatura Desarrollo de Sistemas Software Basados en Componentes y Servicios** del Máster Profesional Universitario en Ingeniería Informática de la Universidad de Granada.
- Robert C. Martin. 2017. **Clean Architecture: A Craftsman's Guide to Software Structure and Design** (1st. ed.). Prentice Hall Press, USA.
- 2019. **Software Architecture**: 13th European Conference, ECSA 2019, Paris, France, September 9–13, 2019, Proceedings. Springer-Verlag, Berlin, Heidelberg.
- **Guía de arquitectura de apps**. https://developer.android.com/topic/architecture, consultada por última vez el 16/01/2023.
- **Common Design Patterns and App Architectures for Android**. https://www.kodeco.com/18409174-common-design-patterns-and-app-architectures-for-android, consultada por última vez el 16/01/2023.

¡GRACIAS!

¿Alguna pregunta?

Please keep this slide for attribution