Reto App Mercado Libre

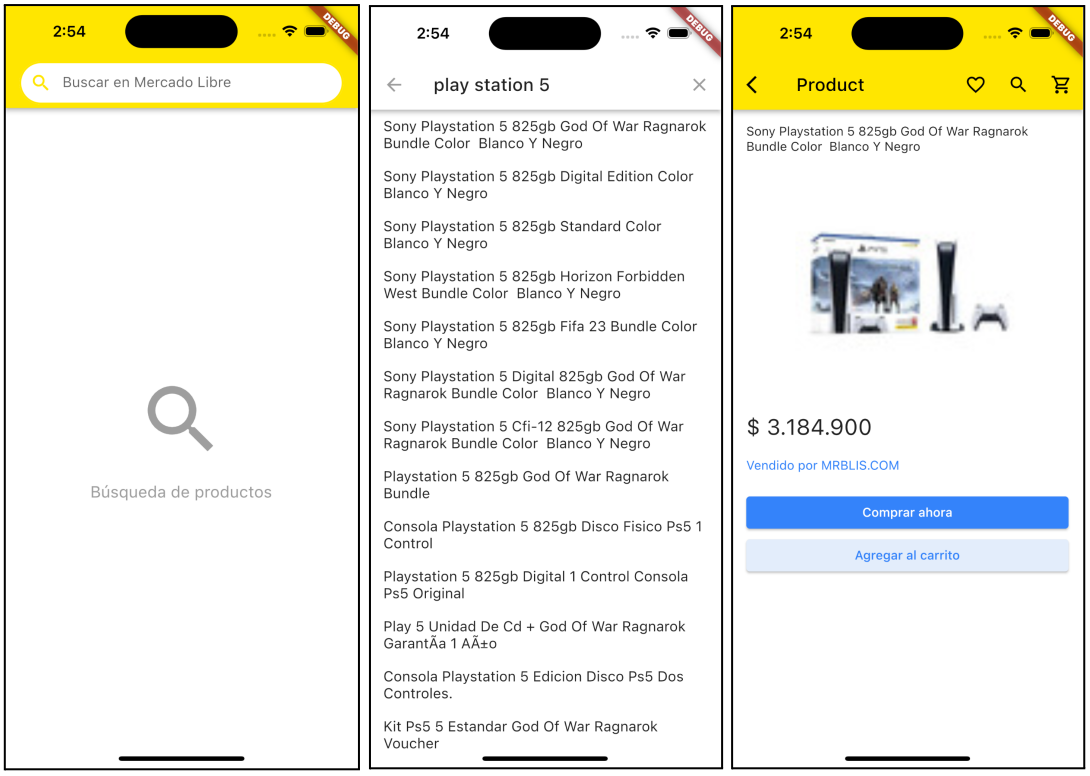
Descripción de la Solución

Análisis del reto

Considero que es un reto sencillo a nivel funcional que podría resolverse con cualquier tecnología para desarrollo de aplicaciones móviles y con patrones de arquitectura y de diseño poco complejos.
Sin embargo, se tiene el objetivo de mostrar capacidades para el proceso de selección y por ello propongo patrones, tecnologías y herramientas utilizadas en proyectos grandes y/o que deben mantenerse en el tiempo.

Interfaz de Usuario

Se propone la siguiente interfaz de usuario para dar solución al reto:



Tecnologías a Utilizar

SDK: No se especifica una tecnología requerida por lo cual elijo una a mi gusto.
Para este reto elijo **Flutter**, por los siguientes motivos:

- Viabilidad técnica para cumplir con los requerimientos del reto actual.
- Viabilidad técnica para cumplir con los requerimientos de un proyecto grande.
- Es la tecnología Mobile en la que me encuentro más actualizado y por ello la realización del reto será más rápida.

Lenguaje de programación: **Dart**. Este es el lenguaje de programación utilizado por Flutter.

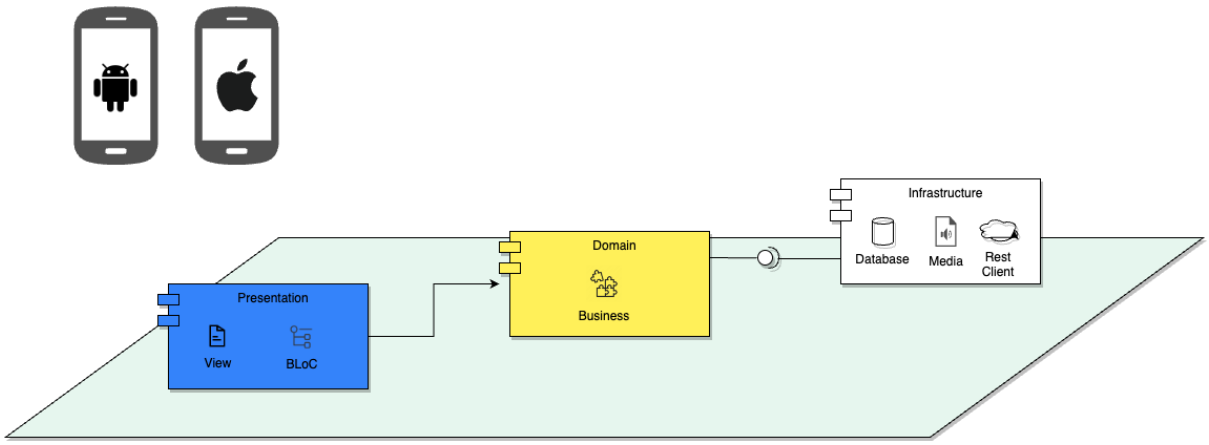
https://dart.dev/

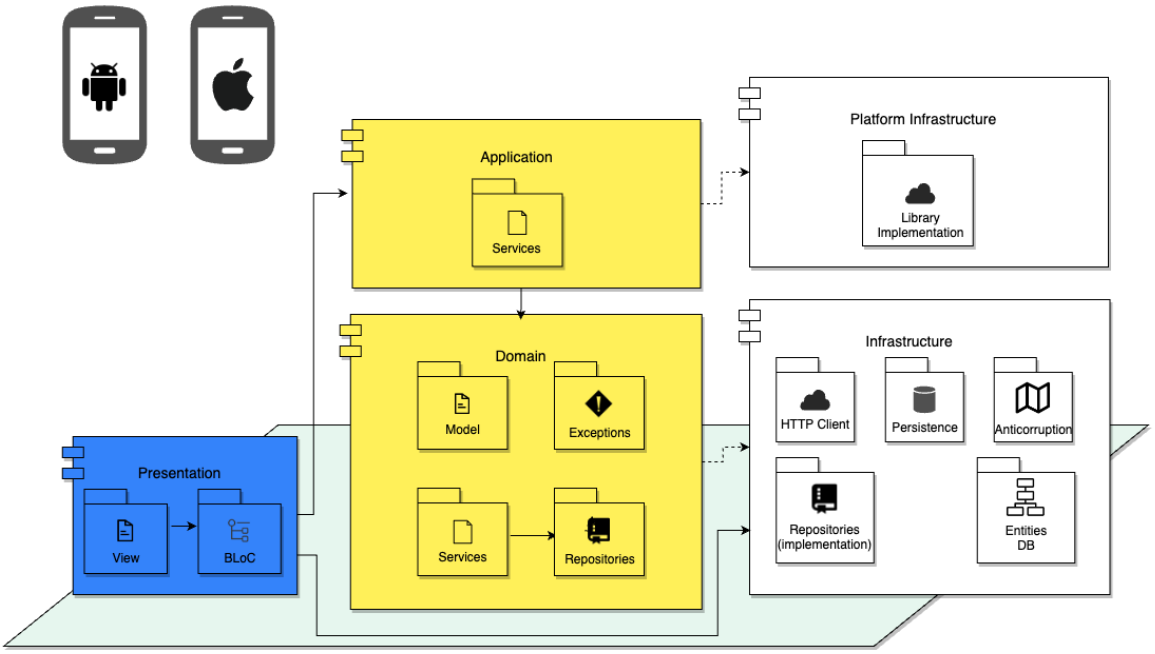
Patrón de arquitectura: **Clean Architecture**. Este patrón de arquitectura nos ayuda a que nuestro software sea mantenible, flexible, testable, a que tenga separadas las responsabilidades. Lo anterior se logra ya que está basado en principios de diseño de software como SoC, SOLID y utiliza POO.

Para una mejor comprensión del patrón de diseño se recomienda leer documentación.

https://medium.com/ruangguru/an-introduction-to-flutter-clean-architecture-ae00154001b0

Diagramas de arquitectura: Se realizará una implementación modularizada basada en los siguientes diagramas:





En el diagrama anterior se evidencian los siguientes elementos:

1. **Presentation**. Este módulo es el encargado de la presentación de la aplicación. Está compuesto por:

    a. La interfaz de usuario.
    b. Los controladores de la vista (Activity, Fragment)
    c. Los Viewmodel o Presenters, los cuales se encargan de manejar toda la lógica de presentación, recibir solicitudes de la vista y respuestas del dominio o la infraestructura.
    d. Componentes propios de la plataforma.

2. **Domain**. Este módulo es el encargado de tener toda la lógica de la aplicación, que se pueda considerar como “lógica de negocio”, tal como validaciones, cálculos, etc. Está compuesto por:

    a. Entities, Value Objects, Aggregates; los cuales me sirven para colocar los objetos importantes dentro del dominio de negocio y la lógica relacionada con ellos. Esta lógica debe ser aquella que cada objeto está en la capacidad de responder sin acceder a ningún otro objeto.
    b. Repositories, que sirven para abstraer el acceso a datos. Ya que son una abstracción, son interfaces y no están acopladas con librerías de terceros, ni siquiera en su nombramiento.
    c. Factories, son opcionales y sirven para la creación de objetos complejos.
    d. Services, son objetos que sirven para gestionar necesidades del dominio que involucran varios objetos o repositorios cuya implementación es externa al dominio.
    e. Pueden agregarse opcionalmente otros tipos de objetos como Excepciones de Negocio.

3. **Infrastructure**. Dependiendo de la arquitectura de referencia implementada, este módulo podría llamarse también Data. Este módulo es el encargado de tener las implementaciones de los repositorios que hay en el Dominio, Implementaciones de Base de Datos, Cliente Http u otras librerías de terceros. Está compuesto por:

    a. Implementaciones de los repositorios.
    b. Manejador de base de datos y componentes propios de la librería utilizada para este fin. No es obligatorio y se hace solamente cuando necesitamos persistir información.
    c. Cliente Http y componentes propios de la librería utilizada para este fin. No es obligatorio y se hace solamente cuando necesitamos consultar información a un servicio web.
    d. Implementaciones de librerías de terceros cuando sea necesario.

La descripción anterior fue bastante pequeña para dar contexto, pero se sugiere tratar de adquirir un buen conocimiento acerca de Domain Driven Design antes de hacer implementaciones.

Patrones de diseño: Para resolver situaciones específicas dentro del proyecto se utilizarán los siguientes patrones de diseño:

- **BLoC (Business Logic Component)**: Es un patrón para manejo de estados que nos sirve como intermediario entre la interfaz de usuario y nuestra lógica y/o el acceso a datos.

- **Repository**: Es un patrón que nos sirve para aislar el acceso a los datos de nuestra lógica de negocio, sin importar el origen de estos datos.

- **Builder**: Es un patrón de diseño creacional que nos sirve para crear objetos completos paso a paso.

- **Triple A**: Este patrón sugiere dividir el código de las pruebas en tres partes, Arrange - Act - Assert, y con ellos dar una mejor legibilidad a éstas.

Integración Continua: La integración continua es un proceso que consiste en hacer integraciones constantes de forma automática para detectar errores en una etapa temprana del proyecto y resolverlos antes de que éste se encuentre en una etapa productiva.

Para ello existen diferentes herramientas, y para este reto escojo **Jenkins**, dada la facilidad de uso, documentación y también la experiencia que tengo con ella.

Análisis de código: Es una estrategia que mediante el análisis del código fuente, busca encontrar errores, vulnerabilidades, fugas de memoria, y también busca que la calidad del código sea buena basándose en un perfil de reglas del lenguaje de programación y/o del framework utilizado en el desarrollo.

Para este reto haré uso de las siguientes herramientas:

- **SonarQube**: Es una herramienta “gratuita” para analizar código muy utilizada y soporta diferentes tecnologías.
  Esta herramienta necesita ser instalada localmente pero puede tener un archivo de configuración del proyecto localmente en el repositorio de código y aprovecharé esa característica.

- **Flutter Analyzer**: Es la herramienta predeterminada para analizar código en Flutter. Me brinda comando por consola para realizar análisis de código.

- **Very Good Analysis**: Es una librería para Flutter que me permite realizar un análisis de código más estricto ya que incluye una mayor cantidad de reglas que Flutter Analyzer.

Requerimientos:

- Versión de Flutter: 3.7.6
- Versión de Dart: 2.19.3
- Versión Mínima Android SDK: 21
- Versión Android Target SDK: 33
- Versión Mínima iOS: 11.0

Librerías utilizadas:

- **Http - Cliente Http**:
  Como cliente HTTP para el consumo de servicios Rest se usó http ya que facilita este trabajo en aplicaciones Flutter y es desarrollada y mantenida por ellos.
  Se agregó un Timeout en cliente Http de 10 segundos. Esto puede cambiar según la necesidad del proyecto.

  Versión: 0.13.5
  https://pub.dev/packages/http

- **Injectable - Inyección de dependencias**:
  Para realizar la inyección de dependencias se usó la librería por medio de etiquetas y auto generación de código, junto con [GetIt](https://pub.dev/packages/get_it).

  Versión: 2.1.0
  https://pub.dev/packages/injectable

- **GetIt - Localizador de servicios**:
  Integrado con [Injectable](https://pub.dev/packages/injectable) permite realizar inyección de dependencias.

  Versión: 7.2.0
  https://pub.dev/packages/injectable

- **Path y Path Provider - Manipulación de rutas**:
  Para manipular y acceder a rutas del sistema de archivos. Principalmente usado para la ubicación de la base de datos local.

  Versión Path: 1.8.2
  https://pub.dev/packages/path

  Versión Path Provider: 2.0.0
  https://pub.dev/packages/path_provider

- **Equatable - Comparación de objetos**:
  Para mejorar la comparación de objetos y evitar posibles errores por diferencia de referencias para objetos con los mismos atributos.

  Versión: 2.0.5
  https://pub.dev/packages/equatable

- **Flutter BLoC - Para manejar el estado de las vistas**:
  Para almacenar y administrar datos relacionados con la IU de manera optimizada para los ciclos de vida.

  Versión: 8.0.0
  https://pub.dev/packages/flutter_bloc

- **Firebase Analytics - Analítica**:
  Es una solución de medición de apps, que proporciona estadísticas sobre el uso de las apps y la participación de los usuarios.

  Versión 10.2.0
  https://pub.dev/packages/firebase_analytics

- **Firebase Core**:
  Un complemento de Flutter para usar Firebase Core API, que permite conectarse a varias aplicaciones de Firebase.

  Versión 2.8.0
  https://pub.dev/packages/firebase_core

- **Firebase Crashlytics - Reporte de errores**:
  Un complemento de Flutter para usar la API de Firebase Crashlytics.

  Versión 3.1.0
  https://pub.dev/packages/firebase_crashlytics

- **Mocktail - Crear Mocks de objetos en Testing**:
  Permite la creación de objetos dobles de prueba en pruebas unitarias automatizadas con el propósito de desarrollo basado en pruebas o desarrollo basado en comportamiento. No necesita generación de código.

Versión: 0.3.0
https://pub.dev/packages/mocktail

- **Very Good Analysis - Análisis de código**:
  Es una herramienta con reglas definidas para analizar el código con el fin de encontrar errores de sintaxis, código incorrecto o malas prácticas, siguiendo las mejores prácticas para el lenguaje Dart.

  Versión: 3.1.0
  https://pub.dev/packages/very_good_analysis

- **json annotation**:
  Define las anotaciones utilizadas por json_serializable para crear código para la serialización y deserialización de JSON.

  Versión: 4.8.0
  https://pub.dev/packages/json_annotation

- **json serializable**:
  Proporciona constructores de Dart Build System para manejar JSON.
  Los constructores generan código cuando encuentran miembros anotados con clases definidas en json_annotation.

  Versión: 6.6.1
  https://pub.dev/packages/json_serializable

Control de Versiones:
Se utiliza un repositorio en Github, el cual está basado en Git.

Repositorio de código:

https://github.com/adsi1407/ProductSearch

Manejo de ramas en el repositorio:
- main:
- development:
- feature/product_search
- feature/quality_tools

Generación de colores:
Se hace uso de la siguiente app web:

https://coolors.co/

Resultados de las pruebas:

```
D9DL60VYCP:domain andres.santacoloma$ flutter pub run test_cov_console
-----------------------------------------|---------|---------|---------|-------------------|
File                                     |% Branch | % Funcs | % Lines | Uncovered Line #s |
-----------------------------------------|---------|---------|---------|-------------------|
lib/                                     |         |         |         |                   |
 domain.dart                             |    0.00 |    0.00 |    0.00 |    no unit testing|
 domain.module.dart                      |    0.00 |    0.00 |    0.00 |    no unit testing|
lib/src/product/exception/               |         |         |         |                   |
 empty_param_exception.dart              |  100.00 |  100.00 |  100.00 |                   |
 negative_param_exception.dart           |  100.00 |  100.00 |  100.00 |                   |
 no_data_product_exception.dart          |  100.00 |  100.00 |    0.00 |                 5|
 short_search_text_exception.dart        |  100.00 |  100.00 |    0.00 |                 5|
lib/src/product/model/                   |         |         |         |                   |
 product.dart                            |  100.00 |  100.00 |  100.00 |                   |
 seller.dart                             |  100.00 |  100.00 |  100.00 |                   |
lib/src/product/repository/              |         |         |         |                   |
 product_repository.dart                 |    0.00 |    0.00 |    0.00 |    no unit testing|
lib/src/product/service/                 |         |         |         |                   |
 product_service.dart                    |  100.00 |  100.00 |  100.00 |                   |
lib/src/shared/exception/                |         |         |         |                   |
 business_exception.dart                 |  100.00 |  100.00 |    0.00 |              2,6,8|
-----------------------------------------|---------|---------|---------|-------------------|
 All files with unit testing             |  100.00 |  100.00 |   85.71 |                   |
-----------------------------------------|---------|---------|---------|-------------------|
```