ULPGC · UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

Práctica 4

Elección de una arquitectura

Programación de Aplicaciones Móviles Nativas
15 de octubre de 2023

Autora:

Ana del Carmen Santana Ojeda (ana.santana152@alu.ulpgc.es)

Índice

1. Introducción

En la siguiente práctica vamos a tratar con **cinco supuestos prácticos** relacionados con el desarrollo de aplicaciones móviles. Cada supuesto tiene sus propias **restricciones y requisitos**, como presupuesto, tiempos de entrega, y justificaremos qué arquitectura es la más adecuada para cada uno.

Primero explicaremos las arquitecturas brevemente para contextualizar en qué nos basaremos en la elección de la arquitectura.

- **MVC (Model-View-Controller)**: Está orientado a aplicaciones donde la interfaz de usuario necesita **interactuar con una lógica de negocio subyacente**. Consiste en tres bloques, el primero es el '**Model**' que representa los datos y la lógica de negocio, luego está el '**View**' que se encarga de la interfaz de usuario y la presentación de datos, y para terminar el último bloque es el '**Controller**' que actúa como intermediario de los dos bloques anteriores, gestionando las interacciones del usuario.

- **MVP** (Model-View-Presenter): Es una variante de MVC, con la diferencia de que separa más la lógica de presentación de la '**View**'. En esta arquitectura también tenemos tres bloques, los dos primeros '**Model**' y '**View**' hacen la misma funcionalidad que en el MVC, mientras que el '**Presenter**', actúa como intermediario en los anteriores bloques, gestionando las interacciones del usuario. Esta arquitectura está pensada para **aplicaciones donde se busca separar la lógica de presentación de la lógica de negocio**.

- **MVVM** (Model-View-ViewModel): Esta arquitectura se utiliza más en aplicaciones móviles (como aplicaciones Android) y aplicaciones de escritorio (como aplicaciones WPF en el entorno .NET). Contiene tres bloques, '**Model**' almacena los datos y la lógica de negocio, '**View**' representa la interfaz de usuario y '**ViewModel**' es el intermediario, proporcionando datos y lógica específica.

- **MVI** (Model-View-Intent): MVI se utiliza para **aplicaciones Android**, que se centra en la **unidireccionalidad de los datos** y las interacciones del usuario. Es la última arquitectura que se divide en tres bloques, '**Model**' y '**View**' son iguales que en las anteriores e 'Intent' define las acciones que el usuario realiza en la interfaz y se encarga de actualizar el Model en consecuencia.

- **Clean Architecture**: Es una arquitectura definida por **capas**, como son la presentación, dominio y de datos. Da una estructura flexible y modular para el desarrollo de aplicaciones, facilitando la independencia de la interfaz de usuario y la lógica de negocio. Esta arquitectura se utiliza para **aplicaciones empresariales y proyectos grandes**, donde se necesita un alto grado de separación de responsabilidades y una arquitectura escalable y mantenible.

- **Arquitectura Hexagonal**: Su filosofía es que una aplicación debe ser independiente de los detalles de su entorno. Se organiza en **hexágonos concéntricos**, donde el núcleo contiene la **lógica de negocio pura**, y los adaptadores proporcionan interfaces para interactuar con el mundo exterior. Se utiliza comúnmente en aplicaciones empresariales y **sistemas donde la modularidad y la prueba unitaria son esenciales**.

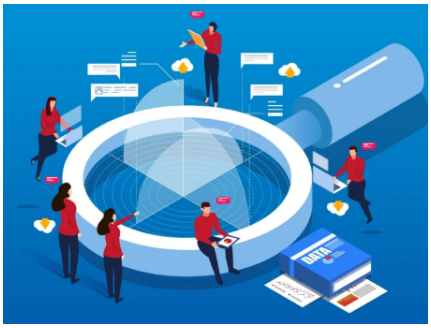

Imagen 1: Arquitectura

2. Desarrollo

2.1. Supuesto 1: Aplicación de E-commerce para una PYME

Una pequeña empresa quiere lanzar su tienda online a través de una aplicación móvil nativa.

Presupuesto: Limitado.

Tiempos de entrega: 4 meses.

Recursos humanos: Un desarrollador principal y un diseñador.

Rendimiento: Se espera un tráfico moderado, pero es esencial que la aplicación sea rápida y eficiente.

Bajo mi punto de vista, considero que es apropiado optar por la arquitectura **MVVM** en este proyecto, debido a las **restricciones con presupuesto y al hecho de contar con un equipo reducido**.

Esta arquitectura, al estar dividida en componentes bien definidos, facilita que tanto el desarrollador como el diseñador puedan **trabajar de manera simultánea y eficiente en sus respectivas áreas**.

Dado que la tienda online requiere un alto nivel de eficiencia y velocidad en su funcionamiento, considero que **MVVM** proporciona una gestión eficaz de los datos y la interfaz de usuario, lo que resultará fundamental para satisfacer al usuarios.

Además, teniendo en cuenta que la empresa actualmente es pequeña, podría crecer en el futuro, **MVVM** es escalable con lo cual permitirá incorporar fácilmente nuevas funcionalidades a medida que la empresa se expanda.

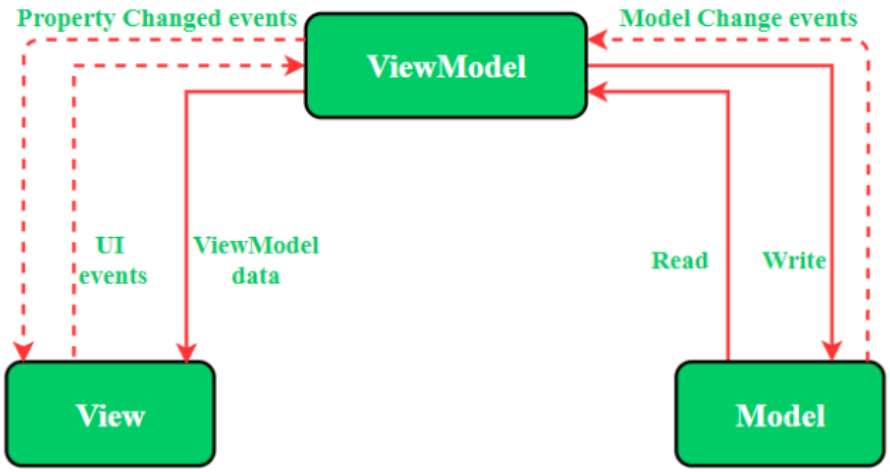

Imagen 2: Arquitectura MVVM

2.2. Supuesto 2: Aplicación Social Interactiva para una Startup

Una startup quiere crear una aplicación social con características interactivas, como chats en tiempo real y transmisiones en vivo.

Presupuesto: Moderado.

Tiempos de entrega: 6-8 meses.

Recursos humanos: Un equipo de tres desarrolladores, un diseñador y un programador backend.

Rendimiento: Se espera un alto tráfico y es crucial que la aplicación maneje interacciones en tiempo real.

Considero que es apropiado elegir la arquitectura **Clean Architecture** para nuestro proyecto. Esta elección se basa en su eficiencia en la gestión de la lógica en tiempo real, ya que esta arquitectura se encuentra **dividida en capas**, incluyendo una capa de dominio que permite separar las preocupaciones y gestionar las interacciones en tiempo real de manera efectiva.

Dado que nuestro equipo ha crecido, Clean Architecture se vuelve aún más relevante. Su división en capas **facilita la asignación de responsabilidades**, permitiendo que los miembros del equipo trabajen de manera coordinada y eficiente.

Además, esta arquitectura es escalable y permite añadir funcionalidades adicionales de manera sencilla. También promueve la independencia tecnológica, con lo cual podemos seleccionar las tecnologías que mejor se adapten a nuestro proyecto.

Además esta arquitectura tiene un enfoque en la modularidad y la **clara separación de responsabilidades**, lo que facilita la mantenibilidad a largo plazo de la aplicación.

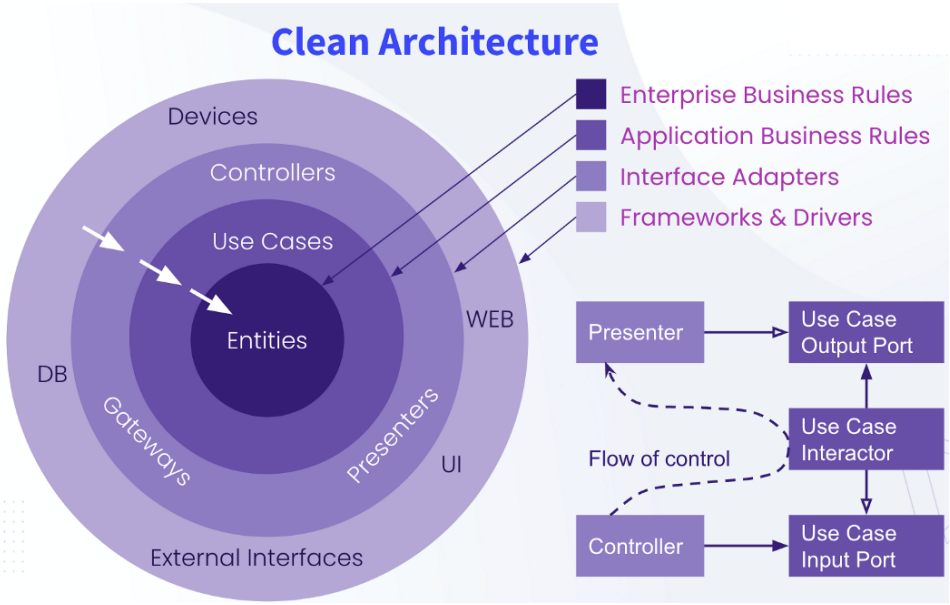

Imagen 3: Clean Architecture

2.3. Supuesto 3: Aplicación Financiera para una Gran Empresa

Una gran empresa financiera quiere desarrollar una aplicación para que sus clientes gestionen sus finanzas, con características como visualización de transacciones, transferencias y análisis financiero.

Presupuesto: Alto.

Tiempos de entrega: 10-12 meses.

Recursos humanos: Un equipo grande con múltiple desarrolladores, diseñadores, especialistas en seguridad y analistas.

Rendimiento: Se espera un tráfico muy alto y es esencial que la aplicación sea segura y eficiente.

Bajo mi punto de vista, considero que es apropiado optar por la **arquitectura Hexagonal** ya que este proyecto tiene un equipo grande y esta arquitectura facilita la colaboración y la asignación de tareas, ya que cada parte del sistema puede ser independiente y tratada como un '**adaptador**' que se conecta a los puertos de la aplicación principal.

Además al tener un presupuesto alto y un plazo largo pues esta arquitectura es **combeniente utilizarla cuando hay plazos largos**.

Esta arquitectura permite la **escalabilidad y la optimización del rendimiento**, lo que es esencial para manejar un tráfico muy alto en una aplicación financiera, además **tiene independencia de la tecnología**, lo que es crucial en un entorno financiero donde las regulaciones pueden cambiar.

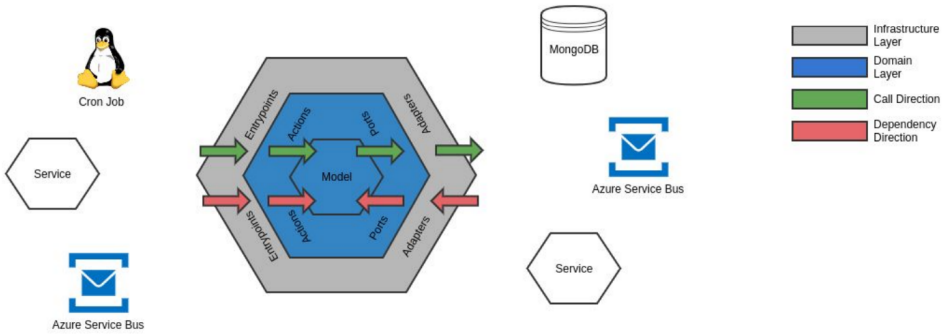

Imagen 4: Arquitectura Hexagonal

2.4. Supuesto 4: Plataforma de Salud y Bienestar para Hospitales

Un hospital de renombre desea desarrollar una aplicación móvil nativa que permita a los pacientes acceder a sus historiales médicos, programar citas, chatear con especialistas y recibir recomendaciones personalizadas basadas en su historial.

Presupuesto: muy alto.

Tiempos de entrega: 12-15 meses.

Recursos humanos: un equipo multidisciplinario compuesto por varios desarrolladores móviles, desarrolladores backend, especialistas en seguridad de la información, diseñadores UX/UI y analistas de sistemas.

Rendimiento: se espera un **tráfico constante y alto debido a la gran cantidad de pacientes**. La seguridad y privacidad de los datos es primordial.

En esta situación, estoy considerando dos opciones: **Clean Architecture y Arquitectura Hexagonal**. Ambas arquitecturas **requieren tiempo en su desarrollo y son altamente adecuadas** para un entorno hospitalario debido a su sólida estructura base.
Personalmente, me inclino hacia la **Arquitectura Hexagonal**, ya que esta **orientada a trabajar de manera independiente** de los detalles específicos de su entorno. Esta

característica genera una mayor confianza en un contexto hospitalario, donde la estabilidad y la fiabilidad son fundamentales.

2.5. Supuesto 5: Aplicación Prototipo para un Hackathon

Un grupo de estudiantes decide participar en un hackathon de 48 horas. Su objetivo es crear un prototipo funcional de una aplicación móvil que ayude a las personas a encontrar compañeros de viaje para compartir gastos en carreteras de peaje.

Presupuesto: Mínimo. Los estudiantes usarán herramientas y recursos gratuitos disponibles.

Tiempos de entrega: 48-72 horas.

Recursos humanos: Un equipo de tres estudiantes con habilidades mixtas: un desarrollador, un diseñador y alguien con habilidades de negocio.

Rendimiento: Como es un prototipo, no se espera un tráfico real. La aplicación debe ser lo suficientemente funcional para demostrar la idea.

En este supuesto pienso que lo más correcto sería utilizar **MVP** al separar la lógica de presentación con la interfaz podría **facilitar el desarrollo rápido** y una interfaz de usuario funcional.

Esta arquitura ayuda a acelerar el proceso de desarrollo al **enfocarse en la funcionalidad esencial y minimizar la complejidad arquitectónica**.

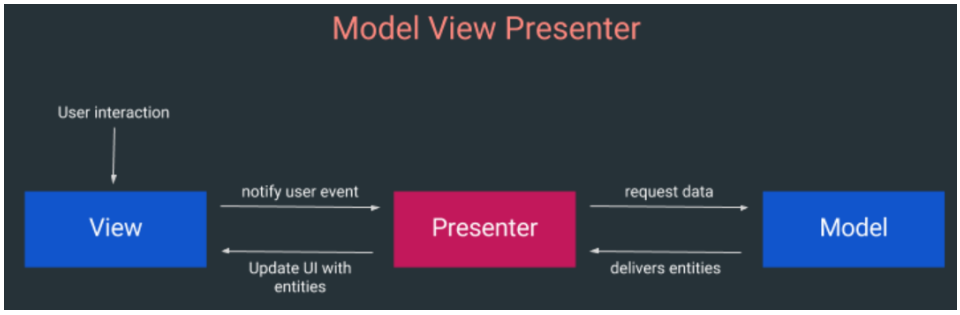

Imagen 5: Arquitectura MVP

4. Bibliografía

[1] Arquitectura y ciclo de vida de aplicaciones móviles nativas. https://aep24.ulpgc.es/pluginfile.php/350666/mod_page/content/14/arquitectura-2023-10-1-12-21-16.pdf

[2] ¿Por qué no funciona MVC en Android. https://fahedhermoza.medium.com/por-qu%C3%A9-no-funciona-mvc-en-android-d0b747a823c0

[3] Arquitectura MVP en Android para principiantes. https://medium.com/@carloslopez_19744/%EF%B8%8F-arquitectura-mvp-en-android-para-principiantes-30b5675ff7b6

[4] ¿Qué es y cómo funciona la arquitectura MVI? https://medium.com/@robercoding/que-es-y-como-funciona-la-arquitectura-mvi-desarrollo-android-kotlin-e6a161e1b2db

[5] ¿Qué es la arquitectura hexagonal? Definición y ejemplos https://apiumhub.com/es/tech-blog-barcelona/arquitectura-hexagonal/

[6] Implement Clean Architecture in Android https://medium.com/simform-engineering/clean-architecture-in-android-12d61c4f5318

[7] How to make a clean architecture Android app, using MVVM, Firestore, and Jetpack Compose? https://medium.com/firebase-tips-tricks/how-to-make-a-clean-architecture-android-app-using-mvvm-firestore-and-jetpack-compose-abdb5