ULPGC · UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

Práctica 4

Elección de una arquitectura

Programación de Aplicaciones Móviles Nativas
15 de octubre de 2023

Autor:

Alejandro David Arzola Saavedra (alejandro.arzola101@alu.ulpgc.es)

Índice

1. Introducción

Se presentan cinco supuestos prácticos relacionados con el desarrollo de aplicaciones móviles. Cada supuesto tiene sus propias restricciones y requisitos, como presupuesto, tiempos de entrega y hay que justificar qué arquitectura es la más adecuada para cada supuesto

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: Tráfico moderado, es esencial que la aplicación sea rápida y eficiente.

2.1.1. Selección de la Arquitectura

Para una aplicación comentada previamente aunque **MVC es comúnmente utilizado y menos complejo**, para el contexto de **aplicaciones móviles nativas**, la arquitectura **MVVM** (Model-View-ViewModel) podría ser **la opción más apropiada** para 4 meses de desarrollo.

MVC, en su concepción, se aplicó principalmente a aplicaciones de escritorio y web, **no considerando específicamente las peculiaridades de las aplicaciones móviles nativas**.

La elección de **MVVM conlleva varias ventajas**. En primer lugar, ofrece una clara **separación de responsabilidades**, lo cual es fundamental para un **desarrollo eficiente**, especialmente cuando se cuenta con **un solo desarrollador y un presupuesto limitado**. La capacidad de **enlace de datos bidireccional** de MVVM facilita la **actualización dinámica de la interfaz de usuario**, proporcionando una **experiencia fluida y agradable para el usuario**.

La **facilidad de mantenimiento** es otra ventaja significativa. Al **separar las responsabilidades** entre el modelo, la vista y el modelo de vista, se **simplifica la gestión del código**, lo cual es crucial en **equipos pequeños**.

Además, **MVVM** contribuye a mejorar la **experiencia del usuari**o al permitir una **interfaz más receptiva y dinámica**. En un entorno de E-commerce, donde **la velocidad y la fluidez son esenciales**, esta capacidad puede marcar la diferencia.

Comparando con otras arquitecturas, como la **hexagonal** o **clean architecture**, que pueden ser **más complejas y quizás excesivas para un único desarrollador con un tráfico moderado**, **MVVM se presenta como una opción más adecuada** para el caso específico.

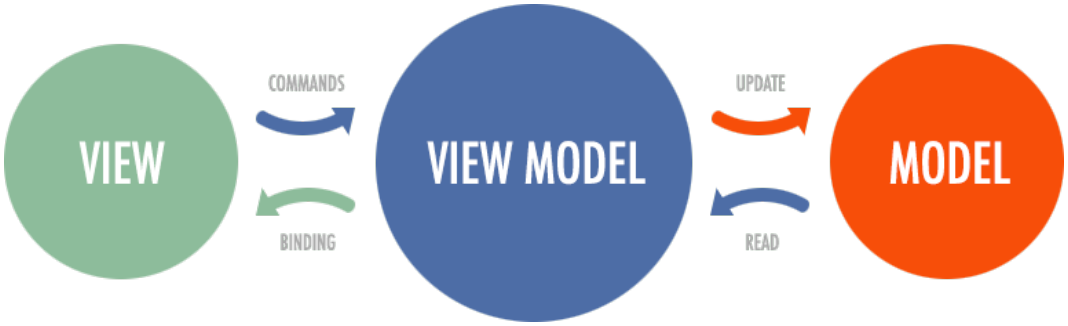

Imagen 1: Arquitectura MVVM

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: Tres desarrolladores, un diseñador y un programador backend.

Rendimiento: Alto tráfico y crucial que maneje interacciones en tiempo real.

2.2.1. Selección de la Arquitectura

En el contexto de una aplicación social con **chats en tiempo real**, donde contamos con un presupuesto moderado y un plazo de desarrollo de 6 a 8 meses, la elección de utilizar **MVP con Clean Architecture** emerge como una opción sólida y bien fundamentada.

Dada la naturaleza interactiva de la aplicación, especialmente con funciones de **chat en tiempo real y transmisiones en vivo**, **el modelo MVP proporciona una clara separación de responsabilidades** entre el Modelo, la Vista y el Presentador. Esta división es esencial para **gestionar flujos de datos reactivos y permitir cambios frecuentes en la lógica de presentación**.

El Presentador en el modelo MVP juega un papel central al manejar las interacciones y actualizar la interfaz de usuario en tiempo real. Esto es crucial para **ofrecer una experiencia fluida a los usuarios, especialmente en un entorno social donde la interactividad es clave**.

La incorporación de **Clean Architetecture refuerza la independencia entre capas**, lo cual es esencial para **adaptarse a cambios rápidos en los requisitos**, una realidad común en startups. La flexibilidad que proporciona Clean Architecture facilita **la adición de nuevas características y modificaciones sin impactar negativamente en otras partes del sistema**.

Considerando el tiempo y presupuesto moderados, la elección de **MVP** se alinea con **un enfoque más liviano en comparación con arquitecturas más complejas**. La combinación de MVP con Clean Architecture establece una base **robusta para la escalabilidad**, preparando la aplicación para crecimientos futuros.

La optimización para el **alto tráfico**, especialmente en el caso de chats en tiempo real, es **facilitada por la estructura de MVP** cuando se combina con tecnologías adecuadas. Este enfoque no solo es **eficiente en términos de rendimiento**, sino que también respalda una **experiencia de usuario efectiva**, un factor clave para **mantener la satisfacción de los usuarios** en una aplicación social interactiva.

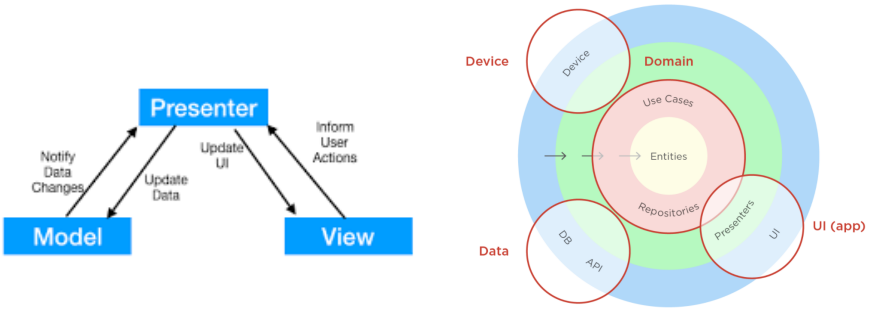

Imagen 2: Arquitectura MVP y Clean Architecture

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: Tráfico muy alto y es esencial que la aplicación sea segura y eficiente.

2.3.1. Selección de la Arquitectura

Con un amplio presupuesto, un extenso plazo de entrega y la presencia de un equipo numeroso de desarrolladores y especialistas en seguridad, la elección más apropiada sería la implementación de **Clean Architecture**.

La prioridad aquí es **la seguridad de los datos**, y **Clean Architecture** proporciona una estructura que permite una **clara separación entre las capas**, facilitando así **la implementación de medidas de seguridad sólidas**. En este contexto, **la capa de negocio puede ser diseñada específicamente para contener tanto la lógica empresarial como las reglas de seguridad**.

Aunque Clean Architecture implica una **inversión inicial** de tiempo debido a su **enfoque modular**, es particularmente **adecuada para proyectos a largo plazo con un presupuesto significativo**. Su facilidad de **mantenimiento y escalabilidad** a medida que evolucionan los requisitos la convierten en una elección acertada.

Con Clean Architecture se **simplifica la colaboración entre especialistas**, ya que cada grupo puede **enfocarse en su área sin interferencias innecesarias**.

En rendimiento, la estructura modular contribuye a **optimizar el rendimiento** de la aplicación, **mejorando su capacidad para gestionar altos volúmenes de tráfico**.

La fortaleza de Clean Architecture se utiliza cuando se espera un crecimiento sustancial de la aplicación. Permite la **adición de nuevas características sin afectar el sistema existente**, lo que es **crucial en proyectos a gran escala con un presupuesto generoso**.

En comparación con MVP o MVI, **Clean Architecture** destaca como una **elección preferida en escenarios donde los recursos no son una limitación**.

Si se implementa adecuadamente, Clean Architecture puede ofrecer una **experiencia de usuario eficiente y segura**.

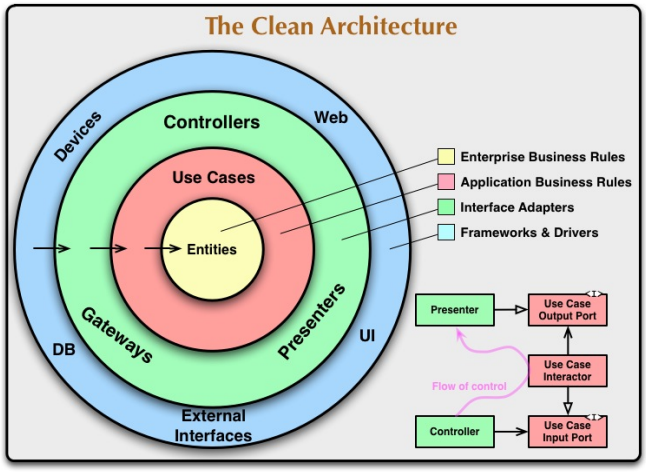

Imagen 3: Clean architecture

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.

2.4.1. Selección de la Arquitectura

En este escenario, donde contamos con un **equipo multidisciplinario** y un **plazo extenso para el desarrollo de la aplicación**, con un enfoque prioritario en la seguridad, la elección de la A**rquitectura Hexagonal se presenta como una opción sólida**.

La **Arquitectura Hexagonal** se destaca por su enfoque en la **separación de capas**, lo que resulta beneficioso para la **implementación de medidas de seguridad**. Al establecer límites claros entre capas, la Arquitectura Hexagonal **controla el acceso a los datos, fortaleciendo la seguridad de la aplicación**.

Dado que disponemos de un presupuesto alto y un plazo de entrega considerable, la Arquitectura Hexagonal se **adapta bien a este contexto**, aunque requiere más tiempo y recursos, **especialmente en las etapas iniciales**, **para establecer la separación de responsabilidades de manera efectiva** al final acaba siendo la mas conveniente en **aplicaciones a largo plazo**.

La mantenibilidad a largo plazo se ve favorecida por esta arquitectura, aprovechando **la colaboración entre especialistas de diversas áreas en un equipo multidisciplinario**. La Arquitectura Hexagonal **facilita la interacción entre desarrolladores** móviles, backend y expertos en seguridad, entre otros.

En términos de **rendimiento**, la Arquitectura Hexagonal muestra eficiencia al **separar la lógica de negocio y las capas**, lo que contribuye a un mejor rendimiento.

La Arquitectura Hexagonal se distingue por su **facilidad para escalar**. A medida que la plataforma crece, **se pueden agregar nuevas funcionalidades e introducir nuevos adaptadores sin afectar la lógica de negocio**, lo que simplifica el proceso de expansión.

La **modularidad** y la clara **separación de responsabilidades** ofrecidas por la Arquitectura Hexagonal **facilitan el desarrollo de interfaces eficientes y libres de errores**.

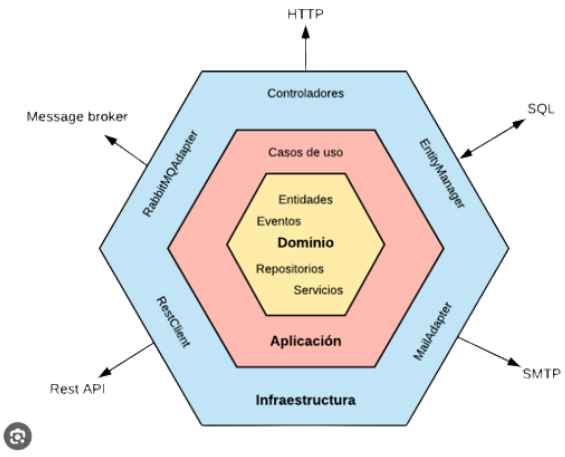

Imagen 4: Arquitectura hexagonal

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.

Tiempos de entrega: 48-72 horas.

Recursos humanos: 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.

2.5.1. Selección de la Arquitectura

Nos encontramos con un grupo de estudiantes inmersos en un hackatón donde los **recursos son limitados, el tiempo es escaso y solo contamos con un desarrollador**. En este contexto, la elección más adecuada podría ser la implementación del patrón de **arquitectura MVP** (Model-View-Presenter).

MVP es una de las arquitecturas más comunes y, en este caso, su **simplicidad**, **rapidez de implementación** pueden ser ventajas significativas. Dada la **falta de experiencia laboral del equipo**, la estructura clara de **MVP** permite que **cada miembro del equipo se enfoque en su área de especialización**.

Al asignar al desarrollador **la lógica de negocio**, al diseñador **la responsabilidad de trabajar en la vista**, y a la persona con **habilidades en el negocio** colaborar en la lógica de presentación,**las fortalezas individuales del equipo se potenciarán**. Además, **MVP facilita la interactividad y la gestión de eventos de usuario**, aspectos fundamentales para demostrar la aplicación en el hackatón.

La simplicidad de **MVP** también permite **adaptarse fácilmente a cambios rápidos**, una característica crucial en un entorno de hackatón donde los **requisitos pueden evolucionar rápidamente**. Al no permitir que la vista interactúe directamente con el modelo, **las modificaciones en la interfaz no afectarán sustancialmente la lógica de negocio**, permitiendo iteraciones más rápidas.

En un hackatón, donde el tiempo de entrega es muy limitado (entre 2 y 3 días) y **no se busca un diseño altamente perfeccionado**, **MVP** destaca por su **capacidad para mejorar la experiencia del usuario**.

En cuanto a la escalabilidad, es cierto que **MVP** puede **tener limitaciones en comparación con algunas arquitecturas más complejas**. Sin embargo, para el contexto del hackatón, ofrece una **solución suficientemente escalable**. **Si la aplicación evoluciona en el futuro, se pueden explorar opciones más escalables y robustas**.

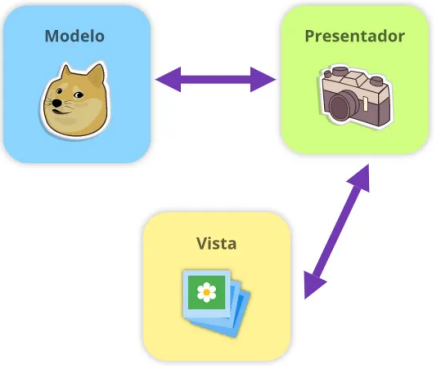

Imagen 5: Arquitectura MVP

3. Conclusión

Como hemos observado, la elección de la arquitectura adecuada para un proyecto depende de varios factores, incluyendo las **necesidades específicas y los recursos disponibles**.

La ventaja de esta diversidad de patrones de arquitectura es que brinda **opciones variadas para abordar distintos tipos de aplicaciones**. La **habilidad para seleccionar la arquitectura más adecuada** es fundamental para el **éxito de un proyecto**.

En última instancia, la clave reside en **tomar decisiones informadas** sobre la arquitectura, basadas en las particularidades y requisitos específicos de cada proyecto.

4. Bibliografía

[1] MVC en android. Recuperado de `https://fahedhermoza.medium.com/por-qu%C3%A9-no-funciona-mvc-en-android-d0b747a823c0`

[2] Arquitectura hexagonal. Recuperado de `https://blog.hubspot.es/website/que-es-arquitectura-hexagonal`

[3] Clean Architecture .Recuperado de `https://wahibhaq.com/blog/clean-architecture-mvp-summary/`

[4] MVP en android. Recuperado de `https://medium.com/@carloslopez_19744/%EF%B8%8F-arquitectura-mvp-en-android-para-principiantes-30b5675ff7b6`

[5] Arquitectura hexagonal. Recuperado de `https://medium.com/@edusalguero/arquitectura-hexagonal-59834bb44b7f`

[6] Clean Software. Recuperado de `https://www.linkedin.com/pulse/clean-software-architecture-key-performance-security-mariam-waheed/`