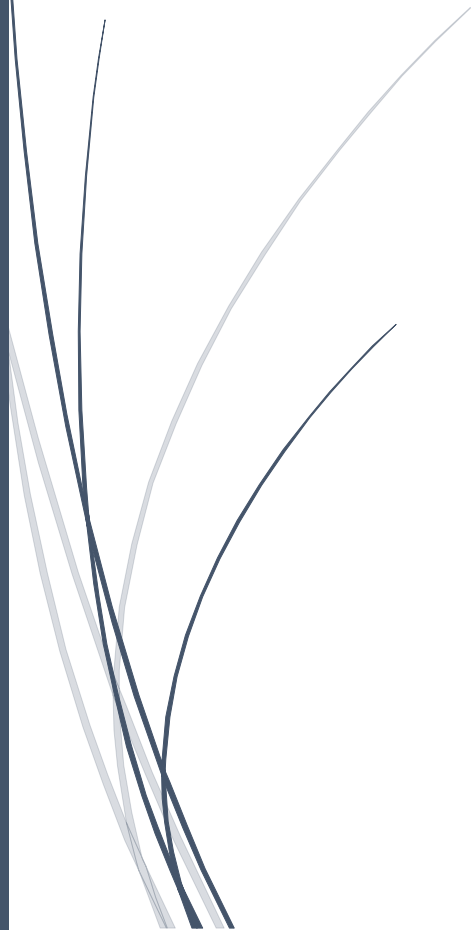
6-10-2023

# Práctica 4: Elección de una arquitectura

Acorán González Moray



Universidad de Las Palmas de Gran Canaria

PROGRAMACION DE APLIACIONES MOVILES NATIVAS

# Contenido

# 1. Introduccion

Este documento proporciona información sobre la importancia de elegir una arquitectura adecuada en el desarrollo de aplicaciones móviles. Se presentan varios escenarios prácticos en los que se deben considerar diferentes restricciones y requisitos, como presupuesto, tiempos de entrega, recursos humanos y rendimiento.

El objetivo es seleccionar la arquitectura más adecuada para cada caso, teniendo en cuenta factores como la seguridad, la escalabilidad y la experiencia del usuario. El documento también establece la fecha límite de entrega de la actividad y menciona que aquellos que no cumplan con la fecha tendrán una penalización en la nota.

# 2. Tipos de arquitectura

Las arquitecturas más comunes aplicadas a la mayorias de proyectos en enornos profesionales son algunas como MVC, MVP, MVVM, MVI, Clean Architecture y arquitectura hexagonal.

**MVC (Model-View-Controller):**

**Modelo:** Representa los datos y la lógica empresarial.

**Vista:** Encargada de la presentación de los datos al usuario.

**Controlador:** Gestiona las interacciones del usuario y actualiza el Modelo y la Vista en consecuencia.

**Casos de uso:** Se utiliza en aplicaciones donde la separación de preocupaciones y la modularidad son importantes. Es común en aplicaciones web y de escritorio.

**MVP (Model-View-Presenter):**

**Modelo:** Almacena la lógica de negocio y los datos.

**Vista:** Representa la interfaz de usuario.

**Presentador:** Actúa como intermediario entre la Vista y el Modelo, maneja la lógica de presentación.

**Casos de uso:** A menudo se utiliza en aplicaciones de escritorio y móviles, donde se necesita una separación clara entre la lógica de presentación y la lógica de negocio.

**MVVM (Model-View-ViewModel):**

**Modelo:** Contiene la lógica de negocio y los datos.

**Vista:** Representa la interfaz de usuario.

**ViewModel:** Actúa como intermediario entre la Vista y el Modelo, maneja la presentación de datos.

**Casos de uso:** Muy común en aplicaciones móviles y aplicaciones de un solo sitio web (SPA) donde se requiere una gestión más sencilla y eficiente de la interfaz de usuario.

**MVI (Model-View-Intent):**

**Modelo:** Almacena el estado de la aplicación y la lógica de negocio.

**Vista:** Representa la interfaz de usuario.

**Intent:** Representa las intenciones del usuario, como acciones o eventos.

**Casos de uso:** Principalmente utilizado en aplicaciones Android para una gestión más predecible y controlada del estado de la interfaz de usuario.

**Clean Architecture:**

Divide una aplicación en capas independientes (Presentación, Dominio, Datos) con reglas estrictas sobre las dependencias entre ellas.

Promueve la separación de preocupaciones y la facilidad de prueba.

**Casos de uso:** Ideal para aplicaciones grandes y complejas donde la mantenibilidad y escalabilidad son críticas.

**Arquitectura Hexagonal:**

También conocida como "Puertos y Adaptadores", se centra en aislar el núcleo de la aplicación de las dependencias externas.

Utiliza puertos (interfaces) y adaptadores (implementaciones) para conectar el núcleo con componentes externos como bases de datos o servicios web.

**Casos de uso:** Adecuada cuando se necesita una alta flexibilidad y portabilidad en la aplicación, lo que es útil en aplicaciones que pueden necesitar cambiar las fuentes de datos o los componentes externos con facilidad.

## 3. Supuestos prácticos

Continuamos analizando cada uno de los supuestos prácticos por separado y de forma independiente:

**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.

**SOL:** La arquitectura más adecuada para el supuesto 1 podría ser la arquitectura MVC (Modelo-Vista-Controlador).

La arquitectura MVC es ampliamente utilizada en el desarrollo de aplicaciones web y móviles, y se caracteriza por separar la lógica de negocio (modelo), la presentación de datos (vista) y el control de las interacciones del usuario (controlador).

En el caso de una aplicación de E-commerce, esta arquitectura permite una clara separación entre la gestión de los productos, el proceso de compra y la interfaz de usuario. Esto **facilita** el mantenimiento y la escalabilidad de la aplicación, ya que los cambios en una capa no afectarán directamente a las demás, aquí señalo la facilidad de aplicar este tipo de arquitectura debido a su baja complejidad, por tanto, aquí nos estamos ajustando tanto al **presupuesto** como a los **recursos humanos** disponibles, además de al timepo de entrega debido a la facilidad de implentación.

Además, la arquitectura MVC es conocida por su capacidad de reutilización de código y su facilidad para realizar pruebas unitarias, lo cual es importante para garantizar la calidad y la estabilidad de una aplicación de E-commerce.

**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.

SOL: La arquitectura más adecuada para el supuesto 2 podría ser la arquitectura hexagonal.

La arquitectura hexagonal, también conocida como arquitectura de puertos y adaptadores, es una arquitectura que se centra en la separación de las capas de la aplicación y en la independencia de las tecnologías utilizadas. Esta arquitectura permite una mayor flexibilidad y escalabilidad, lo que es especialmente importante en una aplicación social con características interactivas y un alto tráfico.

La arquitectura hexagonal se basa en el principio de tener una capa central de dominio que contiene la lógica de negocio y está aislada de las capas externas, como la interfaz de usuario y las bases de datos. Esto permite que la lógica de negocio sea independiente de las tecnologías utilizadas en las capas externas, lo que facilita la evolución y el mantenimiento de la aplicación.

En el caso de una aplicación social interactiva, donde se espera un alto tráfico y se requiere manejar interacciones en tiempo real, la arquitectura hexagonal permite una fácil integración de tecnologías como chats en tiempo real y transmisiones en vivo. Además, al separar la lógica de negocio de las capas externas, se facilita la implementación de pruebas unitarias y la adopción de nuevas tecnologías en el futuro.

**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 múltiples desarrolladores, diseñadores, especialistas en seguridad y analistas.

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

SOL: La arquitectura más adecuada para el supuesto 3 podría ser la arquitectura limpia.

Clean Architecture es una arquitectura que se centra en la separación de responsabilidades y en la independencia de las capas de la aplicación. Esto permite una mayor modularidad y flexibilidad, lo que es especialmente importante en una aplicación financiera que puede requerir cambios y actualizaciones frecuentes.

Además, Clean Architecture promueve la reutilización de código y facilita las pruebas unitarias, lo que es esencial para garantizar la calidad y la seguridad en una aplicación financiera.

En este caso, una gran empresa financiera necesita una aplicación que gestione las finanzas de sus clientes, lo cual implica una gran cantidad de datos y funcionalidades complejas. La arquitectura de Clean Architecture proporciona una estructura clara y escalable para manejar esta complejidad y garantizar la seguridad y el rendimiento de la aplicación.

**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.

SOL: La arquitectura más adecuada para el supuesto 4 podría ser la arquitectura limpia.

Clean Architecture es una arquitectura que se centra en la separación de responsabilidades y en la independencia de las capas de la aplicación. Esto permite una mayor modularidad y flexibilidad, lo que es especialmente importante en un proyecto de esta magnitud.

En el caso de una aplicación para un hospital, es esencial garantizar la seguridad y privacidad de los datos de los pacientes. Clean Architecture facilita la implementación de medidas de seguridad robustas, ya que permite separar claramente las capas de presentación, lógica de negocio y acceso a datos. Esto significa que se pueden aplicar diferentes niveles de seguridad en cada capa, protegiendo así los datos sensibles.

Además, Clean Architecture también facilita la escalabilidad de la aplicación. Dado que se espera un tráfico constante y alto debido a la gran cantidad de pacientes, es importante que la arquitectura sea fácilmente escalable para manejar el crecimiento futuro. Clean Architecture permite agregar nuevas funcionalidades y modificar las existentes sin afectar otras partes de la aplicación, lo que facilita su escalabilidad.

**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

SOL: La arquitectura más adecuada para el supuesto 5 podría ser MVC.

La arquitectura MVC es una de las más comunes y ampliamente utilizadas en el desarrollo de aplicaciones móviles. Proporciona una separación clara de responsabilidades y facilita la modularidad y reutilización del código.

En el caso de este supuesto, donde se tiene un equipo de tres estudiantes con habilidades mixtas, la arquitectura MVC sería una opción adecuada debido a su simplicidad y facilidad de implementación. Permite una clara separación entre la lógica de negocio (modelo), la presentación de datos (vista) y la interacción del usuario (controlador).

Además, la arquitectura MVC es adecuada para un prototipo, ya que permite un desarrollo rápido y flexible. Los estudiantes pueden trabajar de manera independiente en cada componente de la arquitectura y luego integrarlos fácilmente.

## 4. Conclusión

La elección de la arquitectura de software no se limita al presupuesto y recursos humanos, pero algunos patrones son más adecuados para equipos y presupuestos limitados, como MVC y MVP, que son más sencillos y ampliamente conocidos.

MVVM y MVI pueden ser opciones viables con recursos moderados, pero requieren un enfoque más experimentado. En contraste, Clean Architecture y Arquitectura Hexagonal son más complejas y costosas, ideales para proyectos a gran escala con recursos sustanciales y equipos altamente experimentados.

Sin embargo, la elección debe basarse en los requisitos del proyecto, plazos y la experiencia del equipo, no solo en recursos financieros y humanos disponibles. La comprensión y la planificación adecuadas son cruciales independientemente del presupuesto y los recursos.

## 5. Opinión Compañeros

**Opinion sobre elección de Javier Ocón:**

Las explicaciones proporcionadas para cada uno de los supuestos son coherentes y bien fundamentadas, teniendo en cuenta las necesidades específicas de cada proyecto, como seguridad, escalabilidad, presupuesto y recursos disponibles. Las elecciones de arquitectura (MVVM, MVI, MVP) se ajustan adecuadamente a las características y objetivos de cada caso. Esto demuestra un enfoque sólido en la toma de decisiones arquitectónicas considerando las circunstancias particulares de cada proyecto**. Aunque no estoy de acuerdo** en las arquitecturas elegidas en empresas de grandes recursos con enfoque en escalabilidad para este tipo de situaciones recomendaria arquitecturas como la arquitectura limpia o hexagonal.

**Opinion sobre elección de Santiago Emilio Flores:**

Las explicaciones proporcionadas para cada uno de los supuestos son coherentes y bien fundamentadas, teniendo en cuenta las necesidades específicas de cada proyecto, como seguridad, escalabilidad, presupuesto y recursos disponibles. Las elecciones de arquitectura santiago al contrario que javier se adecua mas a las elecciones que he tomado y estoy de acuerdo con sus explicaciones