PRÁCTICA 4

Elección de una arquitectura

PROGRAMACIÓN DE APLICACIONES MÓVILES NATIVAS

SARA SÁNCHEZ GARCÍA

CURSO 2023/24

Introducción

A continuación, se muestran diferentes supuestos ejemplos de aplicaciones y para cada uno de ellos se ha elegido una arquitectura software. Además, se explica porque la arquitectura elegida es adecuada para el supuesto expuesto.

Supuesto 1: Aplicación de E-commerce para una PYME

```
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.
```

En este primer escenario estamos ante un proyecto pequeño para una PYME, con presupuesto limitado. Esta empresa quiere lanzar su tienda online a través de una aplicación móvil nativa y consta de un solo desarrollador. Por tanto, una arquitectura adecuada sería la MVC puesto que separa claramente la lógica de negocio (Model), la presentación de la interfaz de usuario (Vista) y la gestión de la interacción del usuario (Controlador). Esto permite una mejor organización y mantenimiento del código, y hace que el proceso de desarrollo sea más eficiente. Además, al separar la lógica de negocio del resto del código, se facilita la reutilización de código en diferentes partes de la aplicación, característica ideal para el desarrollo de la aplicación con un solo desarrollador. Esto significa que una función que realiza una tarea específica en el modelo puede ser utilizada en diferentes vistas y controladores, sin necesidad de escribir código adicional. Así que el programador no se saturaría de trabajo.

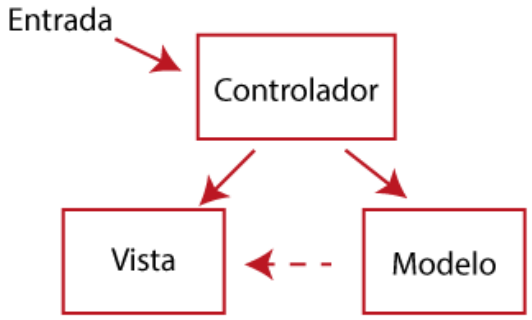

Diagrama 1: Modelo vista controlador.

Cabe resaltar que el patrón MVC puede ser difícil de entender y aplicar correctamente para los desarrolladores novatos. Así que sería indispensable que el desarrollador tuviese experiencia en este tipo de arquitecturas, ya que se requiere una comprensión sólida de la arquitectura y la estructura de la aplicación, así como de los roles de los diferentes componentes del patrón.

Supuesto 2: Aplicación Social Interactiva para una Startup

```
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.
```

En el supuesto 2 estamos ante una aplicación social interactiva para una startup. Además, consta de seis meses para desarrollarlo y tienen un equipo de 3 desarrolladores. Por tanto, la aplicación tiene características interactivas como chat en tiempo real así que la utilización de arquitectura MVP (Model-View-Presenter) es una opción acertada. Dado que la aplicación necesita ser rápida y eficiente este patrón les permite separar la lógica de la interfaz de usuario de la lógica de negocio.

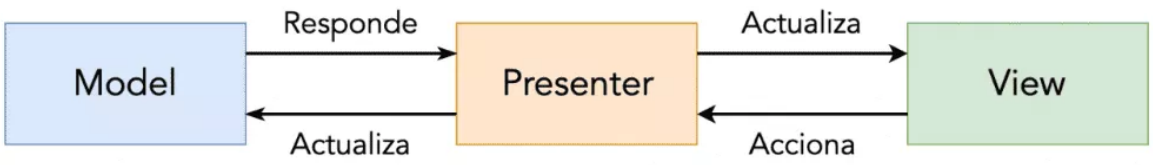

Diagrama 2: Modelo vista presentador.

Otra ventaja es que la separación de código es muy clara lo que permite hacer testing exhaustivo fácil. La mayor desventaja es que hay código repetitivo que puede resultar cansino de implementar pero al tener un equipo de tres desarrolladores y un tiempo de entrega de 6-8 meses no resulta un problema. Por otro lado, es una arquitectura que al principio es difícil de comprender pero si son desarrolladores con experiencia previa en esta arquitectura la aplicación se desarrollará con mayor soltura.

Supuesto 3: Aplicación Financiera para una Gran Empresa

```
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.
```

Para desarrollar una aplicación financiera para una gran empresa con un alto presupuesto, un plazo de entrega de 10-12 meses y un equipo grande con especialistas en seguridad, es esencial seleccionar una arquitectura de software que garantice la seguridad, eficiencia y escalabilidad. La seguridad es de máxima importancia en este contexto, dado que se manejan datos financieros sensibles.

Para este caso en particular la Clean Architecture es especialmente adecuada. Su enfoque en la separación de capas es ideal para aplicaciones financieras. Además, se puede garantizar una fuerte separación entre la lógica de negocio, la interfaz de usuario y los detalles técnicos. Estas son características beneficiosas para una aplicación compleja y con datos sensibles como es una aplicación financiera.

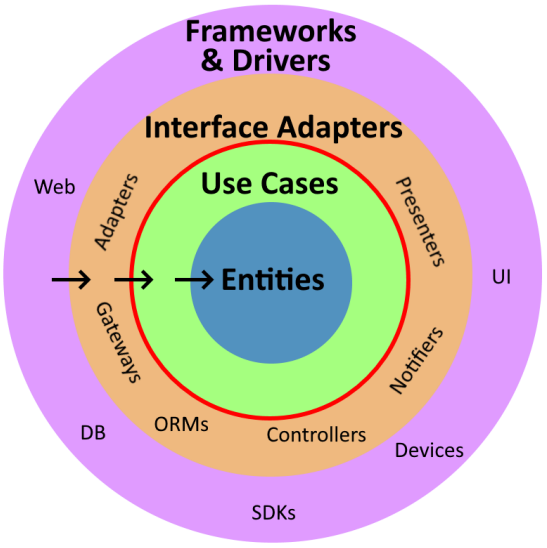

Diagrama 3: Clean Architecture.

La Clean Architecture te permite mantener un alto nivel de seguridad y mantener la eficiencia al separar claramente las capas.
Un inconveniente de la Clean Architecture es que su curva de aprendizaje es muy compleja pero como tiene un equipo grande, con recursos y varios meses para su realización, no es una característica negativa.

Supuesto 4: Plataforma de Salud y Bienestar para Hospitales

```
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.
```

Para desarrollar una plataforma de salud y bienestar para hospitales con un presupuesto muy alto, un equipo multidisciplinario y la necesidad de garantizar un rendimiento constante y alto, así como la seguridad y privacidad de los datos, es crucial seleccionar una arquitectura de software que cumpla con estos requisitos.
Al igual que en el caso anterior la Clean Architecture es muy adecuada para aplicaciones de salud y bienestar, ya que enfatiza la separación de preocupaciones y la gestión de la lógica de negocio y las reglas de negocio de manera independiente. Esto puede ser fundamental para garantizar la seguridad y la privacidad de los datos de salud.
La Clean Architecture promueve una fuerte separación de responsabilidades, lo que facilita la organización de código en capas. Esto mejora la legibilidad y la facilidad de mantenimiento. Además, la lógica de negocio está completamente desacoplada de los detalles de implementación, como frameworks o bibliotecas específicos. Esto hace que la aplicación sea más resistente a cambios en el entorno tecnológico. Otra ventaja, es que facilita las pruebas unitarias, mejorando la seguridad del sistema, algo muy importante en una aplicación médica con la pasión de datos sensibles. También, la Clean Architecture facilita la adición de nuevas características y la escalabilidad de la aplicación. Los nuevos componentes pueden integrarse sin afectar negativamente a los existentes.

Las mayores desventajas son que implementar Clean Architecture puede requerir un esfuerzo adicional en términos de diseño y planificación. Es más compleja en comparación con enfoques más simples como el patrón Modelo-Vista-Controlador (MVC).Además, la creación de una arquitectura sólida siguiendo Clean Architecture puede requerir más recursos y tiempo en las etapas iniciales del desarrollo. Pero al constar de un gran equipo y con gran financiación económica no supone ningún problema.

Supuesto 5: Aplicación Prototipo para un Hackathon

```
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.
```

Dado el contexto de un hackathon con recursos limitados y un objetivo de crear un prototipo funcional en un tiempo muy limitado, es importante elegir una arquitectura de software simple y ágil.
La Arquitectura MVC (Model-View-Controller) puede ser una buena opción pues es una arquitectura clásica que puede ser rápida de implementar en entornos simples. Además, proporciona una separación básica de responsabilidades entre la lógica de negocio, la interfaz de usuario y el control. En un entorno de hackathon, donde el tiempo es escaso, la implementación rápida es esencial.

Conclusión

La elección de la arquitectura adecuada para un proyecto de desarrollo de software es una decisión crucial que puede tener un impacto significativo en la eficiencia, escalabilidad y mantenibilidad de la aplicación. Por ello, es muy importante comprender los requisitos del proyecto, considerar si el proyecto requerirá escalabilidad en el futuro, tener en cuenta si la seguridad es una preocupación principal, se deben buscar arquitecturas que permitan la implementación de medidas de seguridad adecuadas, como la separación de capas en Clean Architecture. También se debe tener en cuenta la mantenibilidad, las habilidades y experiencia de tu equipo de desarrollo, el presupuesto y el tiempo disponible.

Un punto importante es que no hay porque limitarse a una sola arquitectura. En algunos proyectos, puede ser beneficioso combinar elementos de diferentes arquitecturas para satisfacer tus necesidades específicas.

En última instancia, la elección de la arquitectura debe adaptarse a los requisitos y las restricciones del proyecto. No existe una respuesta única para todos los casos, y la decisión dependerá de una evaluación cuidadosa de los factores mencionados anteriormente. Además, es importante recordar que las arquitecturas pueden evolucionar con el tiempo a medida que cambian las necesidades del proyecto, por lo que la flexibilidad es clave.