Elección de una arquitectura

Programación de Aplicaciones Móviles Nativas

Autor: Fernando Sanfiel Reyes

Fecha: 13/10/2023

Índice

Introducción

Teniendo en cuenta que en la ingeniería del software es especialmente relevante elegir una arquitectura adecuada, ya que esto nos facilitará el desarrollo y mantenimiento de la aplicación. También esto influye en su rendimiento, la seguridad y experiencia del usuario. En esta practica se nos plantean varios escenarios prácticos en los cuales debemos elegir la arquitectura más adecuada teniendo en cuenta las circunstancias y restricciones de cada caso en particular.

Supuestos prácticos

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 caso, la arquitectura más adecuada sería MVVM. Esta es una arquitectura que separa la interfaz de usuario del modelo de datos, lo que facilita el desarrollo de aplicaciones móviles nativas con un rendimiento optimizado para ajustarse a los requerimientos de rendimiento. Las principales ventajas de MVVM para este caso es la facilidad de desarrollo que es relevante teniendo en cuenta el tiempo de entrega tan limitado y unos recursos humanos bastante limitados.

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 este caso, el presupuesto y el plazo de tiempo moderados son factores clave a considerar. La arquitectura de microservicios es una arquitectura que permite desarrollar una aplicación social interactiva con un alto rendimiento en un tiempo reducido y con unos recursos humanos y presupuesto moderados. Además, la arquitectura de microservicios es una arquitectura que facilita la escalabilidad y la flexibilidad, lo que es importante para una startup que quiere crear una aplicación que pueda crecer 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.

La arquitectura hexagonal es una opción adecuada para este caso práctico, ya que proporciona una serie de ventajas que la hacen ideal para aplicaciones financieras con un alto volumen de tráfico y requisitos de seguridad estrictos. Teniendo en cuenta que en este caso tenemos un presupuesto alto, un tiempo de entrega bastante amplio y unos buenos recursos humanos no tenemos estos factores como limitantes para elegir una arquitectura compleja como puede ser la arquitectura hexagonal. Por tanto, teniendo en cuenta los demás requisitos es una gran elección usar esta arquitectura debido a su alto grado de seguridad debido a la separación de la aplicación del dominio del sistema operativo y las tecnologías de infraestructura. Esto hace que la aplicación sea más segura, ya que los cambios en el sistema operativo o las tecnologías de infraestructura no afectan a la aplicación. Además es importante destacar que favorece la escalabilidad y el mantenimiento de la aplicación, además de ser especialmente eficiente en lo que a recursos se refiere.

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 este caso es una elección acertada utilizar la *Clean Arquitecture* debido a que el presupuesto, el tiempo de entrega y los recursos humanos no son un factor limitante para poder llevar a cabo un proyecto con una arquitectura compleja como esta. Teniendo en cuenta el requisito limitante principal que es la seguridad implementar la *Clean Arquitecture* es una decisión acertada ya que separa la aplicación del dominio del sistema operativo y las tecnologías de infraestructura. Esto hace que la aplicación sea más segura, ya que los cambios en el sistema operativo o las

tecnologías de infraestructura no afectan a la aplicación. Además de esto es destacable la capacidad de esta arquitectura para facilitar el mantenimiento y escalabilidad de la aplicación.

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 caso tenemos dos factores especialmente limitantes que son el presupuesto y el tiempo de entrega muy corto. Debido a esto, aun que no sea una arquitectura ideal, se debería usar una arquitectura monolítica. Los puntos negativos de esta arquitectura no serán especialmente notables ya que se trata de un prototipo. Por tanto, en este caso practico se a puesto la mayor importancia en la simplicidad, eficiencia y facilidad de desarrollo para la elección de la arquitectura frente a usar una arquitectura mejor pero más compleja.

Bibliografía

Delespierre, B. (04 de 10 de 2021). *Clean Architecture with Laravel*. Obtenido de Clean Architecture with Laravel: https://dev.to/bdelespierre/how-to-implement-clean-architecture-with-laravel-2f2i

HARRIS, C. (10 de 09 de 2023). *Atlassian*. Obtenido de Atlassian: https://www.atlassian.com/es/microservices/microservices-architecture/microservices-vs-monolith#:~:text=¿Qué%20es%20una%20arquitectura%20monolítica,e%20independiente%20de%20otras%20aplicaciones.

Salguero, E. (22 de 06 de 2019). *Medium*. Obtenido de Medium: https://medium.com/@edusalguero/arquitectura-hexagonal-59834bb44b7f

Soni, S. (16 de 03 de 2018). *MVI(Model-View-Intent) Pattern in Android*. Obtenido de MVI(Model-View-Intent) Pattern in Android: https://medium.com/code-yoga/mvi-model-view-intent-pattern-in-android-98c143d1ee7c

Wikipedia. (07 de 07 de 2021). *Modelo–vista–modelo de vista*. Obtenido de Modelo–vista–modelo de vista: https://es.wikipedia.org/wiki/Modelo–vista–modelo_de_vista

Wikipedia. (15 de 04 de 2022). *Modelo–vista–presentador (MVP)*. Obtenido de Modelo–vista–presentador (MVP): https://es.wikipedia.org/wiki/Modelo–vista–presentador