PROGRAMACIÓN DE APLICACIONES MÓVILES NATIVAS

Elección de una arquitectura

Raúl Cruz Ortega
Cuarto / Grupo 41
20/09/2023

Introducción

En el mundo de la ingeniería del software, la elección de una arquitectura adecuada es esencial. Una buena arquitectura no solo facilita el desarrollo y mantenimiento de la aplicación, sino que también puede influir en su rendimiento, seguridad y experiencia del usuario UX.

Por ello en este informe se plantean distintos escenarios a los cuales se les asigna la arquitectura más adecuada según las circunstancias y las restricciones de cada proyecto.

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.

Para este supuesto se plantea implementar la arquitectura **MVP (Model-View-Presenter)** debido a su simplicidad, separación de responsabilidades y facilidad para realizar pruebas. En un entorno con presupuesto y recursos limitados (un desarrollador principal y un diseñador), MVP permite una implementación eficiente y mantenimiento sencillo, mientras se garantiza un rendimiento óptimo para una aplicación de tráfico moderado, cumpliendo así con las consideraciones del supuesto.

La decisión de tomar esta arquitectura se basa en los siguientes puntos:

- **Presupuesto y Recursos Humanos Limitados:** La arquitectura MVP es una arquitectura simple que no requiere inversiones significativas en herramientas o tecnologías costosas. Además, facilita la colaboración y comprensión del código en un equipo pequeño, permitiendo pruebas unitarias eficientes y una implementación más rápida.

- **Tiempos de entrega:** MVP permite separación clara entre la lógica de presentación (el Presentador) y la lógica de negocio (el Modelo) facilitando el desarrollo paralelo y la entrega más rápida de características y funcionalidades. Logrando de este modo adaptarse adecuadamente a entregas más ajustadas en el tiempo.

- **Rapidez y eficiencia:** La arquitectura MVP se enfoca en proporcionar una experiencia de usuario eficiente y rápida. Al separar la capa de presentación (Vista y Presentador) de la lógica de negocio (Modelo), permitiendo optimizar cada componente de manera independiente para lograr un rendimiento mayor.

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.

Para este supuesto se plantea implementar la arquitectura **MVVM (Model-View- ViewModel)** debido a su capacidad para separar la lógica de negocio de la interfaz de usuario en una aplicación social interactiva. Esto permite un desarrollo eficiente con un equipo limitado, manejo de eventos en tiempo real para alto rendimiento y escalabilidad a largo plazo, cumpliendo con el presupuesto y tiempos de entrega moderados.

La decisión de tomar esta arquitectura se basa en los siguientes puntos:

- **Separación de responsabilidades:** MVVM se basa en la clara separación de responsabilidades. En este caso, donde se necesita una aplicación con características complejas, como chats en tiempo real y transmisiones en vivo, la separación de la lógica de negocio y la interfaz de usuario es crucial. Los chats en tiempo real y las transmisiones en vivo requieren

una lógica de negocio sólida y, al mismo tiempo, una interfaz de usuario dinámica.

- **Escalabilidad y mantenimiento:** Aunque el presupuesto es moderado y los tiempos de entrega son relativamente cortos, es fundamental considerar la escalabilidad y el mantenimiento a largo plazo. MVVM promueve una estructura modular y una separación clara de las capas, lo que facilita la expansión de la aplicación y la incorporación de nuevas características en el futuro.

- **Recursos humanos disponibles:** Los desarrolladores pueden especializarse en sus respectivas áreas de competencia, y el diseñador puede trabajar de manera más independiente en la creación de la interfaz de usuario, ya que MVVM facilita la separación de estas responsabilidades.

- **Alto tráfico y tiempo real:** La aplicación debe manejar interacciones en tiempo real y un alto tráfico. MVVM permite gestionar eficazmente eventos en tiempo real al actualizar automáticamente la vista cuando cambian los datos subyacentes.

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 este supuesto se plantea implementar **Clean Arquitecture**, debido a su enfoque en la seguridad, eficiencia, mantenibilidad, y capacidad de adaptación, factores críticos en una aplicación de alta demanda para una gran empresa financiera con un equipo diversificado y un presupuesto alto.

La decisión de tomar esta arquitectura se basa en los siguientes puntos:

- **Seguridad:** Dado que se trata de una aplicación financiera, la seguridad es primordial. La Clean Architecture permite establecer capas bien definidas y controladas, facilitando la implementación de medidas de seguridad en cada nivel del sistema, asegurando que los datos financieros de los clientes estén protegidos.
- **Mantenibilidad a Largo Plazo:** El alto presupuesto sugiere una inversión a largo plazo. La Clean Architecture promueve la mantenibilidad al separar las preocupaciones y brindar claridad en la estructura del proyecto, lo que facilita las actualizaciones y la incorporación de nuevas características a medida que evoluciona el mercado financiero.
- **Eficiencia y rendimiento:** Con un tráfico muy alto previsto, la Clean Architecture permite una estructura flexible que facilita la optimización de capas de datos y negocio para un rendimiento óptimo.
- **Tiempo de entrega razonable:** Aunque el tiempo de entrega es de 10-12 meses, la Clean Architecture permite una gestión más efectiva del proyecto, ya que divide el desarrollo en capas separadas, lo que facilita la entrega incremental y la adaptación a los cambios, permitiendo una respuesta ágil a los requisitos cambiantes.
- **Equipo multidisciplinar:** Con un equipo grande y diverso de desarrolladores, diseñadores, especialistas en seguridad y analistas, la Clean Architecture proporciona una estructura clara con roles definidos en cada capa. Esto facilita la colaboración y permite que cada especialista se enfoque en su área de experiencia.

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 este supuesto se plantea implementar **Clean Arquitecture** debido a su capacidad para garantizar la seguridad y privacidad de los datos en una plataforma de salud, optimizar el rendimiento para el tráfico constante, facilitar la colaboración entre un equipo multidisciplinario, y garantizar la mantenibilidad a largo plazo, aspectos cruciales en un proyecto de salud de alto presupuesto y tiempo de entrega extendido.

La decisión de tomar esta arquitectura se basa en los siguientes puntos:

- **Mantenibilidad a Largo Plazo:** Dado el presupuesto significativo y el tiempo de entrega extendido, es fundamental que la plataforma sea fácilmente mantenible y adaptable a las necesidades cambiantes del sector de la salud. La Clean Architecture promueve la mantenibilidad a largo plazo al separar las preocupaciones y proporcionar claridad en la estructura del proyecto, lo que facilita futuras actualizaciones y mejoras.
- **Seguridad y Privacidad de los Datos:** En el ámbito de la salud, la protección de datos es de suma importancia. La Clean Architecture permite una gestión cuidadosa de la seguridad y la privacidad a lo largo de las capas, garantizando un control detallado sobre cómo se manejan y protegen los datos de los pacientes.
- **Colaboración Multidisciplinaria:** Con un equipo multidisciplinario que incluye desarrolladores móviles, desarrolladores backend, especialistas en seguridad, diseñadores y analistas, la Clean Architecture proporciona una estructura clara con roles definidos en cada capa. Esto facilita la colaboración y la gestión eficiente de un proyecto complejo en el sector de la salud.

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.

Para este supuesto se plantea implementar la arquitectura **MVC (Model-View-Controller)** debido a su simplicidad, velocidad de desarrollo, roles claramente definidos para el equipo de tres estudiantes, y su capacidad para demostrar la idea de manera efectiva en un tiempo limitado, cumpliendo con el presupuesto mínimo y reutilizando recursos disponibles de manera eficiente.

La decisión de tomar esta arquitectura se basa en los siguientes puntos:

- **Sencillez y Velocidad:** El MVC es un patrón de diseño simple y ampliamente utilizado que facilita el desarrollo rápido de aplicaciones. En un hackathon de 48-72 horas, la simplicidad y velocidad de implementación son fundamentales.
- **Herramientas Disponibles:** Dado que se dispone de un presupuesto mínimo y se utilizarán herramientas y recursos gratuitos, el MVC se adapta bien a una amplia variedad de plataformas y frameworks de desarrollo, lo que permite a los estudiantes aprovechar recursos de código abierto y gratuitos para construir su prototipo.
- **Separación de Responsabilidades:** El MVC divide la aplicación en tres componentes claramente definidos: el Modelo (lógica de negocios), la Vista (interfaz de usuario) y el Controlador (gestión de eventos). Esto permite que los estudiantes trabajen de manera eficiente, con roles claramente definidos para cada miembro del equipo.
- **Reutilización de Código:** La división de responsabilidades en MVC permite la reutilización de componentes. Esto es útil en un entorno de desarrollo rápido como un hackathon, donde se necesita construir funcionalidad de manera rápida y eficiente.