ESCUELA DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

PROGRAMACIÓN DE APLICACIONES MOVILES NATIVAS

CURSO 2023/2024

ELECCIÓN DE UNA ARQUITECTURA

CÉSAR JOSÉ DELGADO SUÁREZ

FECHA: 13/10/2024

Tabla de contenido

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. Hoy, nos enfrentaremos a varios escenarios prácticos en los que tendrán que elegir la arquitectura más adecuada según las circunstancias y restricciones presentadas.

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, sugiero la **Arquitectura Hexagonal (Puertos y Adaptadores)** por varios factores principalmente:

- **Flexibilidad para Cambios**
  La Arquitectura Hexagonal permite una clara separación entre el núcleo de la aplicación y los detalles de implementación. Esto es beneficioso para una aplicación de comercio electrónico, ya que la empresa puede necesitar realizar cambios en las interfaces externas sin afectar la lógica de negocio central.
- **Rendimiento**
  Al ser una aplicación de comercio electrónico, la eficiencia y el rendimiento son esenciales. La Arquitectura Hexagonal permite optimizar y adaptar los puertos (interfaces externas) para maximizar el rendimiento sin afectar el núcleo de la aplicación.
- **Recursos Limitados**
  Dado que el presupuesto y el equipo son limitados, la simplicidad y la facilidad de mantenimiento son cruciales. La Arquitectura Hexagonal facilita la prueba unitaria y la sustitución de componentes, lo que ayuda en el mantenimiento y permite al desarrollador principal trabajar de manera más eficiente posible.
- **Entrega Rápida**
  La Arquitectura Hexagonal permite un desarrollo más rápido y modular. El desarrollador principal puede enfocarse en construir y probar la lógica de negocio sin preocuparse demasiado por los detalles de implementación.
- **Escalabilidad**
  Aunque la empresa es pequeña, la Arquitectura Hexagonal proporciona una base que puede escalar a medida que la empresa crece.

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, sugiero la **Arquitectura MVVM (Model-View-ViewModel)** por varios factores principalmente:

- **Interactividad en Tiempo Real**
  MVVM es una arquitectura que se adapta bien a interfaces de usuario interactivas y dinámicas. Además, la implementación de WebSockets para la comunicación en tiempo real se alinea con la necesidad de chats en tiempo real y transmisiones en vivo.
- **Separación de Responsabilidades**
  MVVM permite una clara separación entre la lógica de negocio (Model), la interfaz de usuario (View), y la lógica de presentación y manipulación de datos (ViewModel). Esto es beneficioso para proyectos donde la interactividad y la presentación de datos son cruciales.
- **Escalabilidad**
  La modularidad de MVVM facilita la escalabilidad del proyecto. A medida que la startup crece y se agregan más características, MVVM permite la expansión y modificación de la aplicación sin afectar otras partes del sistema.
- **Recursos Humanos y Tiempos de Entrega**
  Con un equipo de tres desarrolladores, un diseñador y un programador backend, MVVM facilita la colaboración y la gestión de roles. Los desarrolladores pueden centrarse en la lógica de negocio y en la interfaz, mientras el desarrollador backend implementa servicios como los WebSockets.
- **Presupuesto Moderado**
  MVVM no impone costos adicionales significativos y utiliza patrones bien establecidos que son conocidos en la industria.
- **Facilita Pruebas Unitarias**
  MVVM, al separar claramente las capas de la aplicación, facilita las pruebas unitarias. Esto es crucial para garantizar la estabilidad y el rendimiento de una aplicación social interactiva.

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.

En este caso, sugiero la **Clean Architecture** por varios factores principalmente:

- **Separación de Capas y Responsabilidades**
  Clean Architecture propone una clara separación entre las capas de una aplicación (entidades, casos de uso, controladores, adaptadores, etc.). Esto facilita la comprensión del código, la modularidad y la adaptación a cambios futuros sin afectar otras partes del sistema.
- **Seguridad**
  Dada la naturaleza financiera de la aplicación, la seguridad es de suma importancia. Clean Architecture permite la implementación de medidas de seguridad en capas específicas, como en la capa de casos de uso o en la interfaz externa. Los especialistas en seguridad pueden trabajar de manera efectiva para garantizar la robustez de la aplicación.
- **Escalabilidad**
  Clean Architecture promueve la independencia de frameworks externos, lo que facilita la escalabilidad. Puede adaptarse a un aumento en la complejidad de la aplicación y puede manejar nuevos requisitos a medida que la empresa evoluciona.
- **Facilita Pruebas y Mantenimiento**
  La arquitectura limpia favorece las pruebas unitarias y la facilidad de mantenimiento. Dado el tamaño del equipo y el tiempo de desarrollo prolongado, estas características son fundamentales para garantizar la calidad y la sostenibilidad a largo plazo.
- **Alto Rendimiento**
  La arquitectura limpia no impone restricciones específicas en cuanto a tecnologías y permite la elección de las más adecuadas para garantizar un alto rendimiento. Se pueden implementar técnicas de caching, optimizaciones de bases de datos y otras estrategias para satisfacer las necesidades de tráfico muy alto.
- **Presupuesto Alto**
  Aunque Clean Architecture puede requerir una inversión inicial más alta en términos de planificación y diseño, el presupuesto alto proporciona la flexibilidad necesaria para implementar esta arquitectura de manera efectiva.

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, sugiero la **Clean Architecture**, por varios factores principalmente:

- **Privacidad y Seguridad de los Datos**
  La Clean Architecture permite una clara separación de responsabilidades, lo que es crucial cuando se trata de la privacidad y seguridad de los datos médicos. La capa de casos de uso y la capa de entidades pueden ser diseñadas para gestionar directamente las políticas de seguridad y privacidad.
- **Escalabilidad**
  Dado el presupuesto muy alto y la necesidad de escalabilidad, Clean Architecture proporciona una base sólida para manejar un gran volumen de usuarios y datos. La independencia de frameworks externos facilita la adaptación y expansión del sistema a medida que se agregan nuevas funcionalidades.
- **Colaboración Multidisciplinaria**
  Con un equipo multidisciplinario, Clean Architecture permite una colaboración efectiva entre desarrolladores móviles, desarrolladores backend, especialistas en seguridad de la información, diseñadores UX/UI y analistas de sistemas. Cada equipo puede trabajar en su área específica sin afectar otras partes del sistema.
- **Tiempo de Entrega Extendido**
  Dado el tiempo de entrega extendido, la arquitectura limpia proporciona flexibilidad para realizar cambios y mejoras sin comprometer la integridad del sistema.
- **Rendimiento**
  La Clean Architecture no impone restricciones específicas en términos de tecnologías, lo que permite la elección de tecnologías y estrategias específicas para garantizar un rendimiento constante incluso con un tráfico constante y alto.
- **Facilita Pruebas y Mantenimiento**
  La separación de responsabilidades facilita las pruebas unitarias y el mantenimiento a lo largo del tiempo. Esto es especialmente crítico en aplicaciones de salud, donde la fiabilidad y la precisión son fundamentales.
- **Experiencia del Usuario**
  La capa de interfaz de usuario (UI) es clara en Clean Architecture, permitiendo a los diseñadores UX/UI trabajar de manera efectiva en la creación de una interfaz de usuario intuitiva y amigable para los pacientes

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, sugiero la **Arquitectura MVC (Model-View-Controller)**, por varios factores principalmente:

- **Simplicidad y Rapidez de Desarrollo**
  La arquitectura MVC es conocida por su simplicidad y facilidad de implementación. Dado el tiempo limitado del hackathon, es esencial optar por una arquitectura que permita un desarrollo rápido y eficiente.
- **Roles Definidos para Cada Miembro del Equipo**
  MVC asigna claramente responsabilidades a cada componente (modelo, vista y controlador), lo que facilita la colaboración entre el desarrollador, el diseñador y el miembro de negocios. Cada miembro del equipo puede trabajar en su área específica sin interferir demasiado con los demás.
- **Demostración de la Idea**
  El objetivo principal del hackathon es demostrar la viabilidad de la idea a través de un prototipo funcional. La arquitectura MVC permite una rápida iteración y modificación de la interfaz de usuario para que sea lo suficientemente funcional para la demostración.
- **Presupuesto Mínimo**
  La arquitectura MVC no impone costos adicionales, y los estudiantes pueden utilizar herramientas y recursos gratuitos para desarrollar la aplicación.
- **Adaptabilidad**
  Si durante el hackathon se requieren cambios rápidos en la lógica de negocio, la arquitectura MVC permite ajustes sin afectar drásticamente otras partes del sistema.
- **No se Espera Tráfico Real**
  Dado que la aplicación es un prototipo y no se espera tráfico real, la arquitectura MVC, que está más orientada hacia la presentación y la interacción del usuario, es suficiente para este caso

Enlace a github

https://github.com/CesarJoseDelgadoSuarez/Semana-4-PAMN.git