UNIVERSIDAD LAS PALMAS DE GRAN CANARIA

PROGRAMACIÓN DE APLICACIONES MÓVILES NATIVAS

S4 ELECCIÓN DE UNA ARQUITECTURA

PRESENTADO POR: IVONNE YULIETH OJEDA CUERVO

AZ930245

LAS PALMAS DE GRAN CANARIA

OCTUBRE 2023

ANÁLISIS DE 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.

Propuesta: MVVM (Model-View-ViewModel)

Ventajas:

- Separación de responsabilidades: MVVM separa claramente la lógica de negocio (ViewModel) de la interfaz de usuario (View), lo que facilita la colaboración entre el desarrollador y el diseñador.
- Facilidad de mantenimiento: Permite una fácil gestión y actualización de la aplicación debido a la separación de componentes.
- Escalabilidad: Es adecuada para futuras expansiones, lo que es importante si se prevé un crecimiento de la aplicación.
- Rendimiento: Si se optimiza correctamente, puede lograr un buen rendimiento.

Desventajas:

- Curva de aprendizaje: Si el desarrollador no tiene experiencia en implementaciones de dicha arquitectura puede requerir tiempo adicional para que se familiarice con la arquitectura MVVM.

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.

Propuesta: Arquitectura Clean Architecture

Ventajas:

- Modularidad: Clean Architecture divide la aplicación en capas claramente definidas, lo que facilita la gestión de un equipo de desarrollo diverso, y funciones claramente delimitadas para cada rol dentro del equipo.
- Facilita el testing: La separación de las capas permite realizar pruebas unitarias y de integración de manera efectiva, lo que es esencial para una aplicación interactiva de alta calidad.
- Robustez: Transmisiones en vivo exige que se procese y transfiera una gran cantidad de datos con un ancho de banda limitado de los canales de comunicación, por lo tanto la arquitectura debe tener capacidad para resistir cambios y desafíos a lo largo del ciclo de vida de la aplicación.
- Escalabilidad: Puedes escalar componentes individualmente para gestionar el alto tráfico y la interacción en tiempo real.
- Seguridad: Facilita la implementación de medidas de seguridad en las capas necesarias.

Desventajas:

- Mayor complejidad inicial: Puede requerir más tiempo para establecer la estructura inicial de la aplicación debido a su modularidad.

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.

Propuesta: Arquitectura Hexagonal:

Ventajas:

- Independencia de la interfaz de usuario: La arquitectura hexagonal separa claramente la lógica de la aplicación de la interfaz de usuario, lo que es importante en aplicaciones financieras. En este caso la lógica de la aplicación se encuentra en el núcleo de la aplicación y no depende de ninguna tecnología de interfaz de usuario específica. Puedes tener adaptadores diferentes para interactuar con distintas interfaces de usuario (por ejemplo, web, móvil, consola) sin afectar la lógica de negocio central.

  Clean Architecture también enfatiza la independencia de la interfaz de usuario, pero no necesariamente de la manera tan clara como la Arquitectura Hexagonal.

- Seguridad: Facilita la implementación de medidas de seguridad sólidas al permitir la segregación de las capas de seguridad. Por ejemplo, las capas de seguridad específicas se pueden colocar en los adaptadores que interactúan con sistemas externos o interfaces de usuario.

- Escalabilidad: Puedes escalar componentes de manera independiente para adaptarte al crecimiento de la aplicación. Ejemplo: En temporadas picos de pagos (primeros y últimos días del mes) el componente de pagos puede desplegarse múltiples veces para asegurar el procesamiento de un volumen alto de transacciones.
- Mantenimiento y pruebas: La separación de capas simplifica el mantenimiento y la realización de pruebas.

Desventajas:

- Mayor complejidad inicial: La implementación de esta arquitectura puede ser más compleja que enfoques más tradicionales, y requerirá que el equipo realmente tenga conocimiento sólido en la implementación de este tipo de arquitectura porque el tiempo del proyecto no contempla los tiempos que el equipo requiere para tener el conocimiento requerido.

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.

Propuesta: Arquitectura Hexagonal (Puertos y Adaptadores)

Ventajas:

- Independencia de la interfaz de usuario: Separa claramente la lógica de la aplicación de la interfaz de usuario, lo que facilita la adaptación a diferentes tipos de dispositivos y sistemas. Ejemplo: Un adaptador de interfaz de usuario web mostrará los datos de los pacientes en un navegador, mientras que un adaptador de aplicación móvil lo hará en una aplicación para dispositivos móviles.

  Por otra parte permite a cada miembro del equipo derrallorar sus habilidades específicas tanto desde el análisis y diseño de la arquitectura como en el desarrollo del proyecto, los roles son claramente asociados a las funciones que tiene cada componente dentro de la arquitectura.
- Máxima seguridad de datos: Permite la implementación de medidas de seguridad sólidas al permitir la segregación de las capas de seguridad, lo que es fundamental para la protección de datos

médicos sensibles. Distintos niveles de seguridad se pueden aplicar a distintos componentes en las distintas capas de la arquitectura.
- Escalabilidad extrema: Puede escalar componentes específicos para manejar el tráfico constante y alto de manera más flexible.

Desventajas:

- Mayor complejidad inicial: La implementación de esta arquitectura puede ser más compleja que enfoques más tradicionales, y requerirá que el equipo realmente tenga conocimiento sólido en la implementación de este tipo de arquitectura porque el tiempo del proyecto no contempla los tiempos que el equipo requiere para tener el conocimiento requerido.

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.

Propuesta: Modelo Vista Controlador (MVC)

Ventajas:

- Sencillez y rapidez: MVC es una arquitectura simple y rápida de implementar, ideal para un proyecto de hackathon con plazos ajustados.
- Separación de preocupaciones: Es casi similar a una arquitectura monolito, sin embargo, la interfaz de usuario + lógica de negocio está separada del almacenamiento de datos. Ya que Activity sirve como Controlador y Vista, y el modelo está separado.
- Fácil acceso a recursos gratuitos: Dado que los estudiantes tienen un presupuesto mínimo y el tiempo es limitado, MVC puede integrarse con herramientas y recursos gratuitos disponibles en línea.

Desventajas:

- Menos escalable: MVC tiende a ser menos escalable y modular en comparación con otras arquitecturas