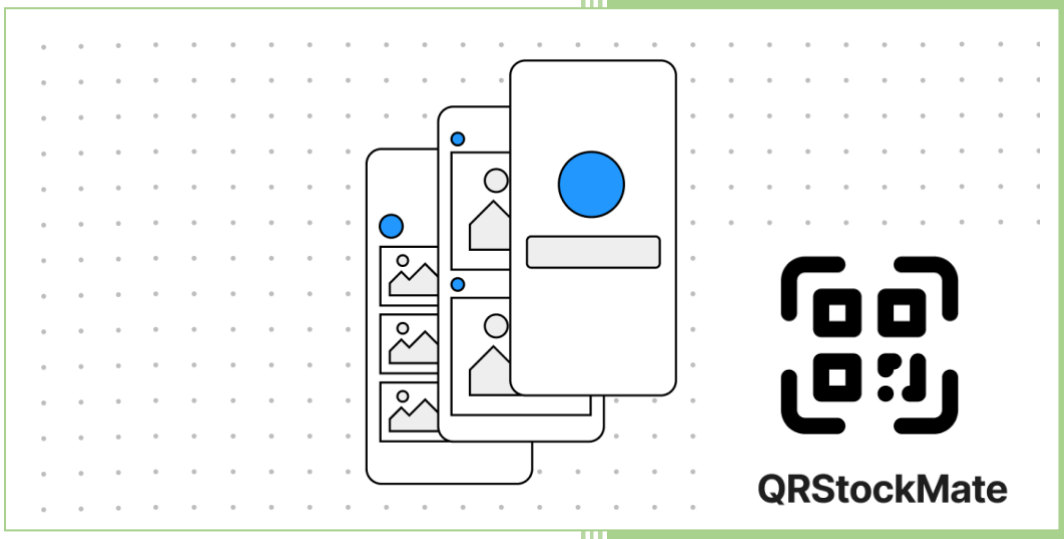
01/10/2023

Práctica 3: Diseño Arquitectónico



Integrantes Grupo 05:

Acorán González Moray

Santiago Flores Reina

Javier Ocón Barreiro

Universidad de las Palmas de Gran Canaria

Programación de Aplicaciones Móviles

Contenido

Definición del Dominio:

La aplicación "QRStockMate" se enfoca en la gestión de inventario y almacenes a través de la tecnología de códigos QR, con el propósito de mejorar la eficiencia y la precisión en el control de inventario y simplificar las tareas de gestión de productos.

Definir las principales funcionalidades

La funcionalidad clave de esta aplicación se basa en el uso de códigos QR, que actúan como una interfaz rápida y precisa para añadir y rastrear productos en almacenes.

Registro de Productos:

- Permite registrar nuevos productos en el inventario, incluyendo información como nombre, descripción, localización, fecha de entrada, stock.

CRUD de Almacenes:

- Permite gestionar almacenes relacionados con la empresa, incluyendo información como nombre, localización, administradores, organización.

Lectura de Códigos QR:

- Permite escanear códigos QR de productos para obtener información instantánea sobre el producto y gestionarlos.

Entrada y Salida de Productos:

- Permite registrar la entrada y salida de productos en el inventario, actualizando la cantidad disponible.

Historial de Transacciones:

- Permite generar informes de todas las operaciones realizadas en la aplicación para facilitar la auditoría.

Búsqueda y Filtros:

- Permite buscar productos por nombre, código de barras o cualquier otro criterio relevante.

Información Detallada del Producto:

- Proporciona información detallada sobre cada producto incluyendo nombre, descripción, localización, fecha de entrada.

Seguridad y Acceso Controlado:

- Proporciona acceso seguro a la aplicación a través de autenticación.

- Administra diferentes niveles de permisos de usuario para garantizar un acceso adecuado a las funcionalidades.

Gestión de Almacenes Centralizada:

- Proporciona una visión general de todos los almacenes gestionados, permitiendo a los usuarios seleccionar un almacén específico para acceder a sus detalles y funciones.

Identificación de los principales elementos de la arquitectura elegida:

La arquitectura limpia (Clean Architecture) es un enfoque para organizar el código de un software de una manera que promueva la escalabilidad, mantenibilidad y testeabilidad, al mismo tiempo que se centra en la separación de preocupaciones y la claridad en el diseño.

En este caso nuestro proyecto consta de **2 partes**:

Frontend (Cliente): **Aplicación nativa android** cuya función principal es el consumo de la API creada por nosotros.

Backend: **API** creada desde 0, aquí es donde reside toda la lógica del proyecto y **donde se va a aplicar la arquitectura limpia** para la organización del proyecto, este backend va a disponer de los siguientes elementos:

1.API, interfaz que permite que la aplicación Android se comunique con el backend básicamente es un conjunto de endpoints, controladores o rutas HTTP que permiten realizar diversas operaciones.

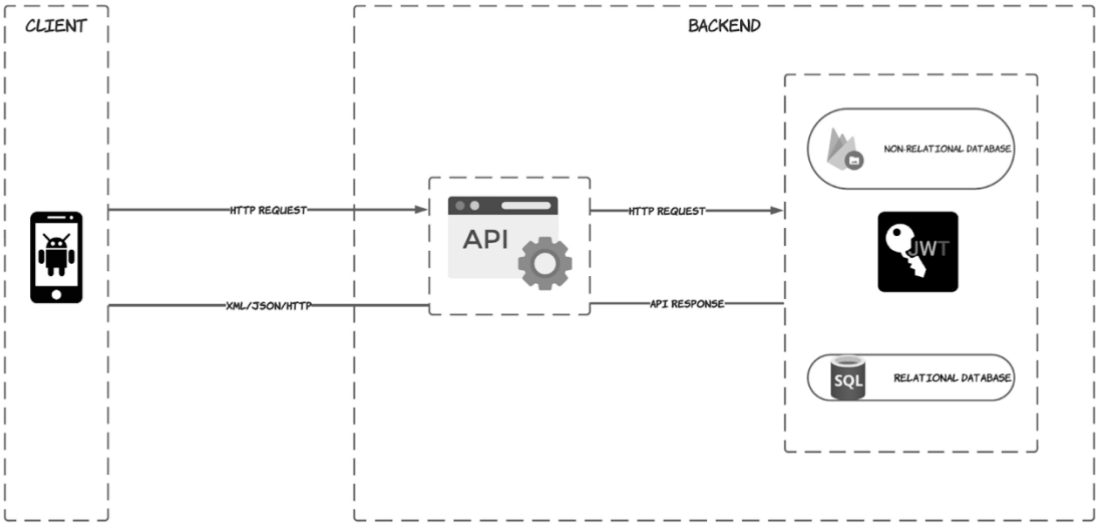
2.INFRASTRUCTURE, capa tecnológica que soporta nuestro backend. Incluye principalmente los siguientes componentes:

- Base de datos relacional (sql server)
- Base de datos no relacional (firebase-storage)
- Estándar de autenticación (JWT)

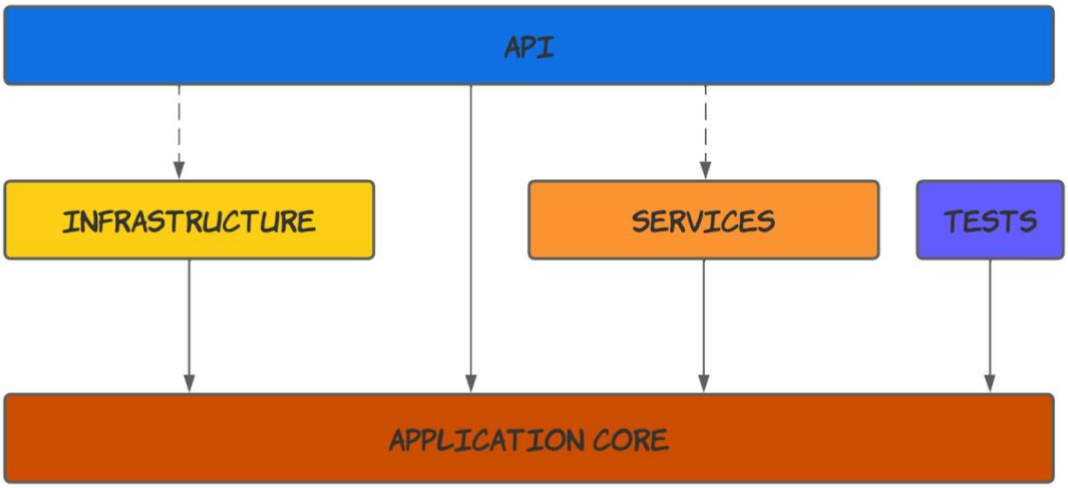
3.SERVICES, se utilizan para mantener la modularidad y la reutilización del código en el backend. Cada servicio se encarga de una tarea particular y es invocado desde la API para no llamar directamente a la infraestructura y hacer que esta sea independiente.

4.APPLICATION CORE, es el corazón del backend donde reside la lógica de negocio principal. Aquí se definen las reglas de negocio que gobiernan la funcionalidad de tu aplicación.

Para hacernos una idea del proyecto en su conjunto se ha creado el siguiente diseño:



Diseño de la Arquitectura:



Cada uno de los niveles aquí representados podría considerarse dentro de la estructura de nuestra aplicación como paquetes independientes, diferentes módulos o librerías que se incluyen como dependencias del nivel principal.

Caso de Uso:

Caso de uso (CU01): **Autenticación de Usuarios**

Autenticación de Usuarios:

- El usuario inicia la aplicación móvil e ingresa sus credenciales (correo y contraseña) en la interfaz de inicio de sesión.
- La aplicación móvil envía una solicitud de autenticación a la API backend a través de una llamada HTTP POST al endpoint de autenticación.
- En el backend, la capa de API recibe la solicitud de autenticación y llama al servicio correspondiente de autenticación que realiza llamadas a la capa de infraestructura para verificar las credenciales del usuario en la base de datos.
- Si las credenciales son correctas, el servicio de autenticación genera un token de acceso y lo devuelve a la API.
- La API responde al cliente móvil con el token de acceso y los datos del usuario encontrados.
- El cliente móvil almacena el token de acceso de manera segura (por ejemplo, en el almacenamiento seguro del dispositivo) para usarlo en las solicitudes posteriores.

Conclusión:

Ventajas de la Arquitectura Limpia (Clean Architecture) en Comparación con Otros Patrones:

Mantenibilidad y Evolución: Clean Architecture fomenta la independencia de las capas de la aplicación, lo que facilita el mantenimiento y la evolución a lo largo del tiempo. Puedes realizar cambios en una capa sin afectar a las demás, lo que es crucial para proyectos a largo plazo.

Flexibilidad y Adaptabilidad: La arquitectura limpia permite que la lógica de negocio sea independiente de los detalles de la interfaz de usuario o la infraestructura. Esto significa que puedes cambiar la interfaz de usuario o la base de datos sin cambiar la lógica subyacente.

Pruebas Unitarias Eficientes: Debido a la separación de responsabilidades y la independencia de las capas, es más fácil escribir pruebas unitarias efectivas para la lógica de negocio. Esto mejora la calidad del código y ayuda a identificar problemas más rápidamente.

Independencia de Plataforma: Clean Architecture permite la independencia de la plataforma. Puedes reutilizar la lógica de negocio en diferentes interfaces de usuario, como aplicaciones móviles, aplicaciones web o incluso interfaces de línea de comandos.

Claridad de Responsabilidades: La arquitectura limpia establece claramente dónde reside la lógica de negocio, qué pertenece a la interfaz de usuario y qué está relacionado con la infraestructura. Esto facilita la comprensión del código y la colaboración en equipos.

Reducción de Acoplamiento: Al separar las capas de la aplicación y utilizar interfaces, Clean Architecture reduce el acoplamiento entre los componentes. Esto hace que sea más fácil realizar cambios sin afectar otras partes del sistema.

Mejora la Escalabilidad: Puedes escalar componentes específicos según sea necesario sin afectar la totalidad del sistema. Esto es especialmente útil para aplicaciones con requerimientos de escalabilidad variables.

Seguridad y Control: La arquitectura limpia permite implementar medidas de seguridad en la capa de lógica de negocio, asegurando que las reglas críticas de seguridad se mantengan de manera consistente en toda la aplicación.

Facilita la Colaboración: La claridad en la estructura de Clean Architecture hace que sea más fácil para los equipos de desarrollo y diseño colaborar de manera efectiva, ya que todos entienden dónde encaja cada componente.

Documentación Eficaz: Debido a la separación de responsabilidades y a la estructura organizada, es más sencillo documentar y comunicar el diseño de la aplicación a otros miembros del equipo o a partes interesadas.

La arquitectura limpia ofrece ventajas sustanciales en términos de organización, mantenibilidad, pruebas y flexibilidad en comparación con otros patrones arquitectónicos. Sin embargo, su elección debe basarse en los requisitos específicos de tu proyecto y en la complejidad de la lógica de negocio que debes gestionar.

Bibliografía

https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures

https://www.lucidchart.com/pages/es