Explicación del proyecto

Mario de Domingo y Diego Novillo
1DAW

Explicación del Proyecto Cine:

Filosofías:

- Todos los repositorios hacen uso de las funciones CRUD y devuelven, o el objeto que se espera (o una lista) o null.
- Todos los servicios y validadores devuelven Result que puede tener, o un error o el objeto que se espera (o una lista)

Clases:

- Las clases más importantes del proyecto son:
    - Butacas: están compuestas de:
        - Id: está compuesto por la fila y la columna en la que está.
        - Estado: define el estado en el que se encuentra. Puede ser:
            - Activa.
            - En mantenimiento.
            - Fuera de servicio.
        - Ocupación: define si está ocupada, libre o reservada. Puede ser:
            - Ocupada.
            - Libre.
            - Reservada.
        - Tipo: define el tipo de butaca que es. Dependiendo del tipo de butaca también dependerá su precio.
        - IsDeleted: muestra si el producto está borrado o no.
        - CreatedAt: cuando se creó la butaca en la base de datos.
        - UpdatedAt: cuando se actualizó por última vez.
    - Producto: están compuestos de:
        - Id: un UID que sirve para identificarlo.
        - Tipo: define el tipo de artículo que es. Puede ser:
            - Comida.
            - Bebida.
            - Otros.
        - Stock: el stock que queda en el inventario.

        - Precio: el precio del producto.
        - IsDeleted: muestra si el producto está borrado o no.
        - CreatedAt: cuando se creó el producto en la base de datos.
        - UpdatedAt: cuando se actualizó por última vez.
    - Cuenta:
        - Id: consiste en una cadena con un formato: lllnnn.
        - IsDeleted: muestra si el producto está borrado o no.
        - CreatedAt: cuando se creó la cuenta en la base de datos.
        - UpdatedAt: cuando se actualizó por última vez.
    - Venta:
        - Id: un UID que sirve para identificarlo.
        - Cliente: el cliente al que pertenece la venta.
        - Líneas: las líneas de venta de las que se compone la venta. Puede estar vacía si el cliente no ha comprado nada más que la entrada.
        - Butaca: la butaca que ha comprado el cliente.
        - IsDeleted: muestra si la venta está borrada o devuelta o no.
        - CreatedAt: cuando se creó la venta en la base de datos.
        - UpdatedAt: cuando se actualizó por última vez.
    - Líneas de venta:
        - Id: un UID que sirve para identificarlo.
        - Producto: el producto en el que se basa la línea de venta.
        - Cantidad: la cantidad que se ha comprado del producto.
        - Precio: el precio del producto que forma parte de la línea de venta.
        - IsDeleted: muestra si la línea de venta está borrada o no.
        - CreatedAt: cuando se creó la línea de venta en la base de datos.
        - UpdatedAt: cuando se actualizó por última vez.

Nuestro programa también hace uso de diferentes clases como:

- Config: guarda datos dentro de ella para que las clases que necesiten información que esté guardada en un archivo externo puedan acceder a ella sin dificultad, como por ejemplo la URL de la base de datos que queramos utilizar.

- DatabaseManager: es una clase que nos permite gestionar la conexión con la base de datos, implementa la interfaz AutoCloseable de Java lo que nos permite cerrar la conexión fácilmente.
- SqlDelightManager: sirve para guardar una variable de tipo _Queries que es generada por SqlDelight dependiendo del nombre del archivo (.sq) el cual nos permite ejecutar las queries que hemos metido en él. La ruta de este archivo se especifica en el archivo build.gradle.kts. También sirve para inicializar las tablas si queremos crearlas, borrar todos los datos, etc.
- CineApp: se encarga de mostrar una interfaz visual con la que el usuario puede interactuar con el programa. Le pasa las instrucciones a los servicios para poder almacenar y exportar la información en la base de datos.
- Algunas clases tienen su propio validador para verificar que los datos que se están introduciendo en la base de datos son válidos.
- También hay clases que tienen una interfaz _Storage para poder intercambiar el tipo de archivo que queremos leer o exportar dependiendo de la implementación que implemente su servicio. El nombre del archivo que se quiere leer está en config.properties.
- Algunas clases tienen su propio DTO, para poder serializarlo y escribirlo en archivos en diferentes formatos como JSON.

Inyección de dependencias:

- La mayoría del proyecto hace uso de anotaciones para crear un módulo que darle a Koin (llamado defaultModule), pero para asegurarse de que los servicios (y cualquier otra clase que implemente un_Storage) coja el correcto, hemos creado otro (llamado storageModule) en el que se especifica qué implementación debe implementar cada clase.

Base de datos:

- Las tablas que están presentes en la base de datos son prácticamente iguales a las clases, excepto por:
    - Debido a la cantidad limitada de tipos que ofrece SQLite, la mayoría de los campos son pasados a una cadena para que los pueda almacenar. Los números a INTEGER o REAL, etc.
    - La tabla de venta no tiene campo de líneas de venta, en su lugar para relacionarlas la tabla de Líneas de venta tiene id_venta.
    - Las tablas se llaman "_Entity" para evitar que SqlDelight cree clases con el mismo nombre que las nuestras y que sean fáciles de identificar.
- Como tuvimos que crear un sistema con el que es posible buscar el estado de un objeto en una fecha específica, las claves de todas las tablas son updatedAt y su clave para poder guardar varios registros del mismo objeto con diferentes fechas.

- Nuestra base de datos no permite borrar nada físicamente, en su lugar guarda un nuevo registro con isDeleted como 1. Tampoco se actualiza ningún registro para no perder datos, sino que cada vez que se modifica, se almacena un nuevo registro de ese mismo objeto con una fecha nueva y la nueva información.