Descripción del Proyecto

El proyecto tiene como objetivo gestionar clientes y sus registros asociados en una base de datos local utilizando **Room**, una biblioteca de Android para el acceso a bases de datos SQLite. Además de la gestión de clientes, también incluye un sistema de logs para registrar acciones importantes, como la creación, modificación o eliminación de clientes.

Tecnologías Utilizadas

- **Room Database**: Para la persistencia de datos.
- **Kotlin**: Lenguaje de programación utilizado para el desarrollo de la aplicación.
- **Jetpack Compose**: Para la interfaz de usuario (UI) de la aplicación.
- **Coroutines (suspend)**: Para operaciones asíncronas, asegurando que las operaciones de base de datos no bloqueen el hilo principal de la aplicación.

Estructura del Proyecto

1. Entidades (Entities)

El proyecto consta de dos entidades principales que representan tablas en la base de datos: **ClienteEntity** y **LogEntity**.

- **ClienteEntity**:
    - **Tabla**: clientes
    - **Atributos**:
        - id: Identificador único autogenerado para cada cliente.
        - documento: Número de documento del cliente.
        - nombre: Nombre completo del cliente.
        - telefonos: Lista de números de teléfono del cliente, almacenada como una cadena separada por comas usando el **TypeConverter** TelefonosConverter.
- **LogEntity**:
    - **Tabla**: logs
    - **Atributos**:

        - id: Identificador único autogenerado del log.
        - comentario: Comentario que describe la acción registrada (por ejemplo, creación, edición o eliminación de clientes).

2. DAOs (Data Access Objects)

El proyecto incluye dos DAOs: **ClienteDao** y **LogDao**. Estos objetos proporcionan métodos para interactuar con la base de datos.

- **ClienteDao**:
    - **Métodos**:
        - insertCliente(cliente: ClienteEntity): Inserta un nuevo cliente en la base de datos.
        - getAllClientes(): Recupera todos los clientes almacenados.
        - update(cliente: ClienteEntity): Actualiza los datos de un cliente existente.
        - deleteByDocumento(documento: Int): Elimina un cliente usando su número de documento.
        - getClienteById(clienteId: Int): Recupera un cliente específico por su número de documento.
- **LogDao**:
    - **Métodos**:
        - getAllLogs(): Recupera todos los registros de logs.
        - insertLog(logEntity: LogEntity): Inserta un nuevo registro de log.

3. Base de Datos (RoomDatabase)

La base de datos está representada por la clase UserDatabase, que extiende de RoomDatabase y proporciona acceso a los DAOs.

- **Tabla**: clientes y logs.
- **Versión**: 3.
- **Métodos**:
    - getDatabase(context: Context): Método para obtener una instancia de la base de datos utilizando el patrón **Singleton** para asegurar que solo se cree una instancia.

4. Repositorio (Repository)

El **Repository** actúa como una capa intermedia entre el ViewModel y los DAOs. Gestiona las operaciones de los datos y permite realizar acciones como la inserción, actualización, eliminación y obtención de clientes y logs.

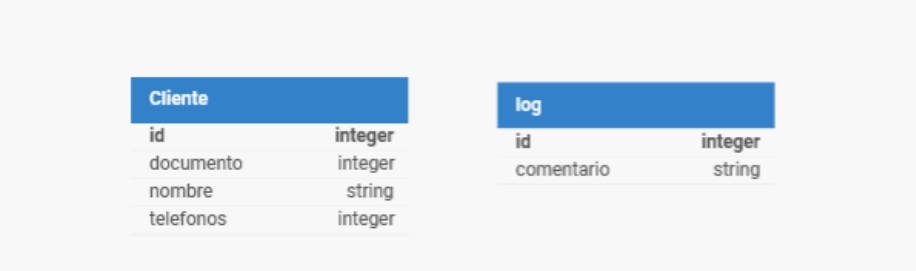
- **Métodos**:
    - insertCliente(cliente: ClienteEntity): Inserta un cliente en la base de datos.
    - getAllClientes(): Obtiene la lista de todos los clientes.
    - updateCliente(cliente: ClienteEntity): Actualiza un cliente.
    - deleteCliente(documento: Int): Elimina un cliente.
    - getClienteById(clienteId: Int): Obtiene un cliente específico por su número de documento.
    - insertLog(logEntity: LogEntity): Inserta un registro de log en la base de datos.

5. Convertidores (TypeConverters)

Se utiliza un **TypeConverter** para manejar la conversión de tipos complejos, como listas de teléfonos, entre la base de datos y la aplicación.

- **TelefonosConverter**:
    - Convierte una lista de teléfonos (List<String>) a una cadena de texto separada por comas y viceversa.

Diagrama Entidad Relacional



Funcionalidades Principales

1. **Gestión de Clientes**:
    - Los usuarios pueden agregar, editar y eliminar clientes.
    - Se guarda información del cliente como su documento, nombre y teléfono en la base de datos.
2. **Gestión de Logs**:
    - Cada acción importante (como la creación, edición o eliminación de un cliente) se registra en la base de datos de logs.
3. **Persistencia**:
    - Los datos de los clientes y los logs se almacenan en una base de datos local mediante Room.
4. **Interfaz de Usuario**:
    - La UI está construida con Jetpack Compose y permite visualizar y gestionar la lista de clientes y logs.
    - Los usuarios pueden actualizar la lista de clientes y acceder a formularios para crear y editar registros.