Programación Móvil

Nombre: ***Adrian Eduardo Reynosa Flores***

#2010455

¿Qué es Room?

Room es parte de la arquitectura de Android Jetpack, diseñado para simplificar el acceso a la base de datos SQLite en aplicaciones Android. Proporciona una capa de abstracción sobre SQLite para permitir operaciones de base de datos más limpias y menos propensas a errores.

Características Principales de Room

1. **Annotation-based**: Room utiliza anotaciones en Java o Kotlin para definir la estructura de la base de datos y las consultas SQL. Esto facilita la definición de entidades, relaciones y consultas sin la necesidad de escribir código SQL directamente.
2. **Components**:
    - **Entity**: Representa una tabla en la base de datos.
    - **DAO (Data Access Object)**: Define métodos para acceder a la base de datos y realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar).
    - **Database**: Contiene la base de datos y sirve como punto de acceso principal para la recuperación de datos. Se define como una clase abstracta que extiende `RoomDatabase`.
3. **Relaciones entre entidades**: Room permite definir relaciones entre entidades mediante anotaciones como `@Relation`, simplificando así consultas que involucren múltiples tablas.
4. **Soporte para LiveData y Coroutines**: Room es compatible con LiveData y Kotlin Coroutines para ejecutar operaciones de base de datos de manera asíncrona y gestionar automáticamente los cambios en los datos.
5. **Migraciones automáticas**: Room facilita la gestión de cambios en la estructura de la base de datos mediante la definición de clases de migración. También proporciona migraciones automáticas para cambios simples de esquema.
6. **Soporte para consultas complejas**: A través del uso de SQLite, Room permite la ejecución de consultas SQL complejas y optimizadas directamente cuando es necesario.

Uso Básico de Room

1. **Definición de la entidad** (`@Entity`): Define la estructura de una tabla en la base de datos.

```
@Entity(tableName = "users")
public class User {
    @PrimaryKey(autoGenerate = true)
    public int id;

    @ColumnInfo(name = "first_name")
    public String firstName;

    @ColumnInfo(name = "last_name")
    public String lastName;
}
```

2. **Definición del DAO** (`@Dao`): Define métodos para acceder a la base de datos y realizar operaciones CRUD.

```
@Dao
public interface UserDao {
    @Query("SELECT * FROM users")
    List<User> getAll();

    @Insert
    void insert(User user);

    @Delete
    void delete(User user);
}
```

3. **Configuración de la base de datos** (`@Database`): Define la base de datos y especifica las entidades asociadas.

```
@Database(entities = {User.class}, version = 1)
public abstract class AppDatabase extends RoomDatabase {
    public abstract UserDao userDao();
}
```

4. **Obtención de una instancia de la base de datos**:

```
AppDatabase db = Room.databaseBuilder(getApplicationContext(),
        AppDatabase.class, "database-name").build();
```

5. **Uso de la base de datos**:

```
UserDao userDao = db.userDao();
List<User> users = userDao.getAll();
```

Ventajas de usar Room

- **Menos código boilerplate**: Reduce la cantidad de código necesario para interactuar con la base de datos en comparación con el uso directo de SQLite.
- **Compilación temprana**: Room realiza verificaciones en tiempo de compilación de SQL y evita errores comunes de sintaxis.

- **Mejor soporte para pruebas unitarias**: Facilita la escritura de pruebas unitarias al permitir la inyección de dependencias en los DAOs.

Conclusión

Room es una herramienta poderosa y eficiente para la persistencia de datos en aplicaciones Android. Proporciona una capa de abstracción sobre SQLite que simplifica el acceso a la base de datos y mejora la seguridad y la fiabilidad de las operaciones de persistencia de datos. Su integración con otros componentes de Jetpack y su soporte para características modernas como LiveData y Coroutines lo convierten en una opción ideal para desarrolladores Android que buscan una solución robusta y fácil de usar para la gestión de datos.