Nombre: Nayelis B. Velásquez Cruz

Carnet: 2011298

Curso: Aplicación Movil

Carrera: Ingeniería en Computación

Informe de Investigación: Room en Android

Introducción

Room es una biblioteca de persistencia de datos que forma parte del conjunto de bibliotecas de Jetpack de Android. Proporciona una capa de abstracción sobre SQLite, permitiendo a los desarrolladores interactuar con la base de datos de manera más sencilla y segura, asegurando al mismo tiempo el cumplimiento de las mejores prácticas de diseño de la arquitectura de la aplicación. Este informe explora las características, beneficios, implementación y mejores prácticas de Room.

Características Principales de Room

1. **Mapeo de Entidades**: Room utiliza anotaciones para definir las tablas de la base de datos y las relaciones entre ellas. Cada entidad se representa como una clase de datos y se anota con `@Entity`.
2. **Consultas Simplificadas**: Las consultas SQL se pueden realizar mediante anotaciones en métodos de interfaces, utilizando `@Query`, `@Insert`, `@Update` y `@Delete`.
3. **Verificación en Tiempo de Compilación**: Room verifica las consultas SQL durante la compilación para asegurar que no haya errores sintácticos y que las consultas sean válidas respecto a las entidades de la base de datos.
4. **Integración con LiveData y Flow**: Room se integra de manera eficiente con LiveData y Flow, permitiendo a los desarrolladores observar los cambios en la base de datos en tiempo real.
5. **Manejo Automático de Migraciones**: Room facilita el manejo de migraciones de la base de datos cuando el esquema cambia entre versiones de la aplicación.

Beneficios de Usar Room

1. **Simplicidad y Legibilidad**: La API de Room es más simple y legible en comparación con el uso directo de SQLite. Esto reduce el tiempo y esfuerzo necesarios para gestionar bases de datos en Android.
2. **Seguridad y Fiabilidad**: Room ofrece una mayor seguridad al proporcionar verificación en tiempo de compilación, lo que reduce la posibilidad de errores en las consultas SQL.
3. **Integración con la Arquitectura de la Aplicación**: Room se integra perfectamente con otros componentes de la arquitectura de Android, como ViewModel y LiveData, facilitando la implementación de un diseño arquitectónico limpio y mantenible.

4. **Mejor Rendimiento**: La integración con LiveData y Flow permite que las aplicaciones respondan de manera más eficiente a los cambios en la base de datos sin necesidad de realizar consultas repetidas.

Implementación de Room

Configuración Inicial:

- Añade la dependencia de Room en el archivo `build.gradle`

```
implementation "androidx.room:room-runtime:2.4.3"
kapt "androidx.room:room-compiler:2.4.3"
```

Definición de Entidades:

```
@Entity(tableName = "users")
data class User(
    @PrimaryKey(autoGenerate = true) val id: Int,
    val name: String,
    val age: Int
)
```

Creación de DAO (Data Access Object):

```
@Dao
interface UserDao {
    @Query("SELECT * FROM users")
    fun getAllUsers(): LiveData<List<User>>

    @Insert
    suspend fun insertUser(user: User)

    @Update
    suspend fun updateUser(user: User)

    @Delete
    suspend fun deleteUser(user: User)
}
```

Creación de la Base de Datos:

```
@Database(entities = [User::class], version = 1)
abstract class AppDatabase : RoomDatabase() {
    abstract fun userDao(): UserDao

    companion object {
        @Volatile private var INSTANCE: AppDatabase? = null

        fun getDatabase(context: Context): AppDatabase {
            return INSTANCE ?: synchronized(this) {
```

```
                val instance = Room.databaseBuilder(
                    context.applicationContext,
                    AppDatabase::class.java,
                    "app_database"
                ).build()
                INSTANCE = instance
                instance
            }
        }
    }
}
```

Mejores Prácticas

1. **Usar el Patrón Repository**: Encapsula el acceso a los datos en una clase repository para separar la lógica de negocio de la capa de datos.
2. **Manejo de Migraciones**: Implementa correctamente las migraciones cuando se realizan cambios en el esquema de la base de datos para evitar la pérdida de datos.
3. **Operaciones en Segundo Plano**: Realiza operaciones de base de datos en hilos secundarios para no bloquear el hilo principal y mantener la interfaz de usuario responsiva.
4. **Pruebas**: Es importante escribir pruebas unitarias para las consultas de Room utilizando el framework de pruebas de Android.

Conclusión

Room es una herramienta poderosa y eficiente para manejar la persistencia de datos en aplicaciones Android. Su simplicidad, integración con componentes de la arquitectura de Android y características avanzadas como la verificación en tiempo de compilación y las migraciones automáticas hacen de Room una elección excelente para la gestión de bases de datos en aplicaciones modernas.