Almacenamiento en Android

Contenidos

SharedPreferences

- Almacena pares clave / valor en archivos asociados a la aplicación.
- Limitado a tipos primitivos y objetos String

```
// Obtención del objeto por defecto
SharedPreferences myPreferences
    = PreferenceManager.getDefaultSharedPreferences(MyActivity.this)

// Objeto con fichero personalizado
SharedPreferences datos = getSharedPreferences("MisDatos", Context.MODE_PRIVATE);

// Inserción
SharedPreferences.Editor myEditor = myPreferences.edit();
myEditor.putString("NAME", "Alice");
myEditor.putInt("AGE", 25);
myEditor.putBoolean("SINGLE?", true);
myEditor.commit();

// Lectura
String name = myPreferences.getString("NAME", "unknown");
int age = myPreferences.getInt("AGE", 0);
boolean isSingle = myPreferences.getBoolean("SINGLE?", false);
```

Almacenamiento Interno

- Directorio de almacenamiento privado a la app.
- Puede almacenar texto y archivos binarios.
- No accesibles directamente por el usuario u otras aplicaciones.
- La información se elimina de forma automática cuando el usuario desinstala la aplicación.

```
// Acceso
File internalStorageDir = getFilesDir();

// Escritura
File alice = new File(internalStorageDir, "alice.csv");
fos = new FileOutputStream(alice);
fos.write("Alice,25,1".getBytes());
fos.close();

// Lectura con métodos estándar java
```

Almacenamiento Externo

- Para almacenar archivos de gran tamaño
- No siempre disponible
- Accesible por el usuario y otras apps

```
// Comprobación de disponibilidad
if(Environment.getExternalStorageState()
               .equals(Environment.MEDIA_MOUNTED)) {

    // Acceso a fichero en la carpeta raíz
    File bob = new File(getExternalFilesDir(null), "bob.jpg");

    // Acceso a fichero en carpeta de fotos
    File bobInPictures = new File(
      Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES),
      "bob.jpg"
    );
} else {
    // No tenemos acceso al almacenamiento externo
```

```
}
```

Base de Datos SQLite

- Para almacenar datos estructurados
- No necesita servidor
- Ligera y fácil de utilizar

```
// Creación o acceso
SQLiteDatabase myDB =
    openOrCreateDatabase("my.db", MODE_PRIVATE, null);

// Ejecución de sentencias
myDB.execSQL(
    "CREATE TABLE IF NOT EXISTS user (name VARCHAR(200), age INT, is_single INT)"
);

// Comandos especializados
ContentValues row1 = new ContentValues();
row1.put("name", "Alice");
row1.put("age", 25);
row1.put("is_single", 1);

ContentValues row2 = new ContentValues();
row2.put("name", "Bob");
row2.put("age", 20);
row2.put("is_single", 0);

myDB.insert("user", null, row1);
myDB.insert("user", null, row2);

// Consultas
Cursor myCursor =
    myDB.rawQuery("select name, age, is_single from user", null);

while(myCursor.moveToNext()) {
    String name = myCursor.getString(0);
    int age = myCursor.getInt(1);
    boolean isSingle = (myCursor.getInt(2)) == 1 ? true:false;
}

myCursor.close();

// Shutdown de la bd
myDB.close();
```

Room

- Capa de abstracción sobre SQLite
- Validación de sentencias sql en tiempo de compilación
- Uso de anotaciones

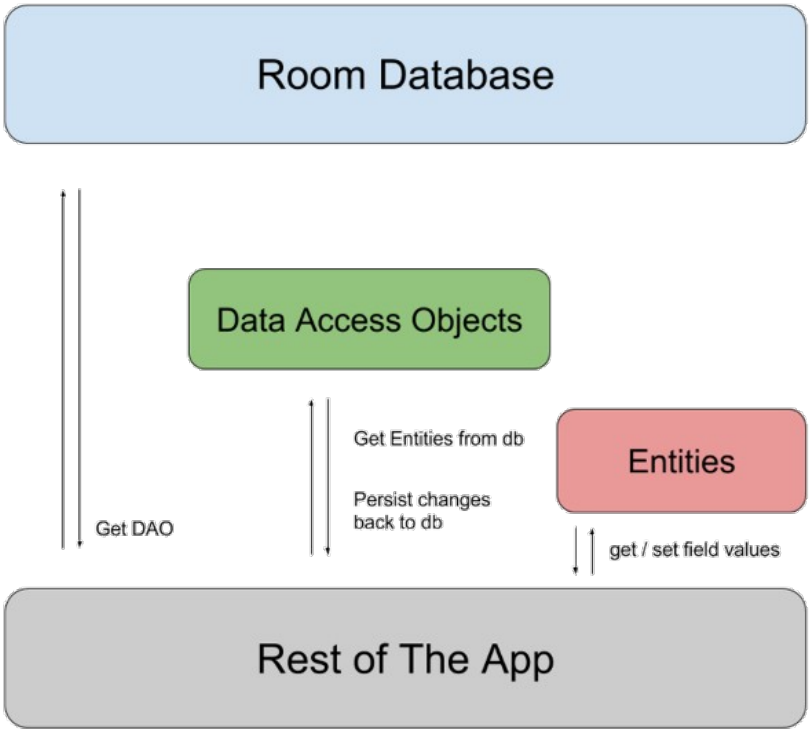
Para que todo esto funcione necesitamos declarar algunas dependencias Gradle:

```
dependencies {
    def room_version = "2.6.1"

    implementation "androidx.room:room-runtime:$room_version"

    // If this project only uses Java source, use the Java annotationProcessor
    // No additional plugins are necessary
    annotationProcessor "androidx.room:room-compiler:$room_version"

    // optional - RxJava2 support for Room
    implementation "androidx.room:room-rxjava2:$room_version"

    // optional - RxJava3 support for Room
```

```
    implementation "androidx.room:room-rxjava3:$room_version"

    // optional - Guava support for Room, including Optional and ListenableFuture
    implementation "androidx.room:room-guava:$room_version"

    // optional - Test helpers
    testImplementation "androidx.room:room-testing:$room_version"

    // optional - Paging 3 Integration
    implementation "androidx.room:room-paging:$room_version"
}
```

Componentes Principales

- Clase Base de Datos
- Data Entities
- DAOS



Ejemplo

Data Entity

```
@Entity
public class User {
    @PrimaryKey
    public int uid;

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

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

DAO

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

    @Query("SELECT * FROM user WHERE uid IN (:userIds)")
    List<User> loadAllByIds(int[] userIds);

    @Query("SELECT * FROM user WHERE first_name LIKE :first AND " +
           "last_name LIKE :last LIMIT 1")
    User findByName(String first, String last);

    @Insert
    void insertAll(User... users);

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

Clase de Base de Datos

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

Uso

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

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