SQLite [JAVA]

Android tiene un pequeño gestor de base de datos que forma parte de su sistema operativo. Esto nos permite almacenar información como cualquier otra base de datos **relacional**. Acceder a SQLite es relativamente sencillo: simplemente necesita una clase gestora y unos cuantos métodos.

Para trabajar con SQLite, la norma general es crear una clase **DataManager** que contenga las consultas que necesitemos. Esta clase debe heredar de **SQLiteOpenHelper**, la cual nos obligará a implementar los métodos **onCreate ()** y **onUpgrade ()**.

Lo primero que hacemos en **DataManager** es indicar la información relacionada con el SQLite. Entre otros, vamos a necesitar el nombre de las tablas, columnas, y las sentencias de SQL indispensables para crear dichas tablas. A modo de ejemplo:

```
// Database Information
private static final String DB_NAME = "alumnos.db";

// database version
private static final int DB_VERSION = 1;

// Table Name
public static final String TABLE_NAME = "Alumno";

// Table columns
private static final String ID = "id";
private static final String LOGIN = "login";
private static final String PASS = "pass";

// Creating table query
private static final String CREATE_TABLE = "create table " + TABLE_NAME + "(" +
        ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
        LOGIN + " TEXT NOT NULL, " +
        PASS + " TEXT " +
        ");";
```

No es habitual que una app haga uso intensivo de las bases de datos, pero nada te impide diseñar apps con varias tablas, y gran cantidad de entradas. En ese caso, añade en **DataManager** toda la información necesaria para todas las tablas que quieras usar.

Ahora toca definir los constructores e implementar **onCreate ()** y **onUpgrade ()**. El primer método se lanza automáticamente cuando intentemos acceder a la Base de Datos y ésta no existe. Si ya existe, este método nunca se lanzará. De esta manera no tenemos que preocuparnos de crear las tablas de forma manual. El último método indica lo que hay que hacer cuando se actualiza la estructura de las tablas. Lo más normal será eliminar las tablas anteriores, si existen.

```
private final Context context;

public DataManager(Context context) {
    super(context, DB_NAME, factory: null, DB_VERSION);
    this.context = context;
}

@Override
public void onCreate(SQLiteDatabase sQLiteDatabase) {
    sQLiteDatabase.execSQL(CREATE_TABLE);
}

@Override
public void onUpgrade(SQLiteDatabase sQLiteDatabase, int oldVersion, int newVersion) {
    sQLiteDatabase.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
    onCreate(sQLiteDatabase);
}
```

Una vez configurado todo esto, debemos implementar los métodos que necesitemos para trabajar con la base de datos. Esto ya es 'Java normal', no depende de las peculiaridades de Android para acceder a la información de las tablas. Están disponibles todas las operaciones de Insert, Update, Select y Delete, de la forma habitual. Es recomendable definir POJOs para mover la información de las **Activity** a la **DataManager**, pero por lo demás, el código ya os es conocido.

```
public List<User> selectAllUsers () {
    List<User> ret = new ArrayList<>();
    String query = "SELECT * FROM " + TABLE_NAME;
    SQLiteDatabase sQLiteDatabase = this.getWritableDatabase();
    Cursor cursor = sQLiteDatabase.rawQuery(query, selectionArgs: null);
    User user;
    while (cursor.moveToNext()) {
        user = new User();
        user.setId(cursor.getInt( i: 0));
        user.setLogin(cursor.getString( i: 1));
        user.setPass(cursor.getString( i: 2));
    }
    cursor.close();
    sQLiteDatabase.close();
    return ret;
}
```

Para hacer uso del DataManager desde una *actividad* bastaría con añadir:

```
DataManager dbManager = new DataManager( context: this);
dbManager.getWritableDatabase();
List<User> alumnos = dbManager.selectAllUsers();
```

Un **detalle final**. La base de datos que creas se guarda dentro del propio paquete de tu app. Esto significa que el peso de tu app puede dispararse fácilmente una vez instalada. A demás, esto significa que, si borras y reinstalas la app en el emulador, la base de datos se borrará. No ocurre lo mismo si simplemente le das a reiniciar la app.

Práctica 35

[Java] Realiza una app que sea capaz de almacenar sus canciones favoritas. Se guardará el título de la canción, el autor, y la URL de YouTube donde se puede escuchar. La app debe permitir listar todas las canciones, añadir una canción, borrar una canción y modificar sus datos. Las canciones deberán aparecer en formato lista.

Práctica 36

[Java] Lo mismo que antes, pero al pulsar una canción de la lista se hará un *intent* y se reproducirá la canción en el navegador.

Práctica 37

[Java] Lo mismo que antes, pero al pulsar una canción de la lista se hará un *intent* y se reproducirá la canción en otro activity que tendrá un componente **Media Player**. Este ejercicio propuesto es para que investigues un rato si ya has terminado otras tareas. Esto lo veremos más adelante. Que te diviertas.