BASE DE DATOS ANDROID

SQLite

- Es un ligero motor de base de datos, de código abierto, que se caracteriza por mantener el almacenamiento de la información persistente **de forma sencilla.**

SQLite

- Ventajas
  - No requiere el soporte de un servidor.
  - No necesita configuración.
  - Usa un archivo para el esquema.
  - Es de código abierto.
- Es por esto que SQLite es una tecnología cómoda para los dispositivos móviles.

SQLite

- Limitaciones
  - No se pueden implementar clausulas como FULL OUTER JOIN y RIGHT JOIN.

SQLite

- Limitaciones
  - No se pueden implementar cláusulas como FULL OUTER JOIN y RIGHT JOIN.

SQLite

- Comandos

```
sqlite3 transportes.db
```

SQLite

- Comandos

```
PRAGMA foreign_keys = ON;
```

```
create table bus(
  id_bus char (6) primary key not null,-- bus001 hasta bus015
  descripcion varchar(25) not null,-- describe el bus
  fecha date not null,--fecha de adquisición o compra
  capacidad integer not null);--cantidad máxima de pasajeros
```

```
create table pasajero(
  id_bus_pas char (6) not null,
  dni int not null,
  nombres varchar(25) not null,
  apellidos varchar(25) not null,
  FOREIGN KEY(id_bus_pas) REFERENCES bus(id_bus)
);
```

SQLite

- Comandos

```
PRAGMA foreign_keys = ON;
```

```
create table bus(
  id_bus char (6) primary key not null,-- bus001 hasta bus015
  descripcion varchar(25) not null,-- describe el bus
  fecha date not null,--fecha de adquisición o compra
  capacidad integer not null);--cantidad máxima de pasajeros
```

```
create table pasajero(
  id_bus_pas char (6) not null,
  dni int not null,
  nombres varchar(25) not null,
  apellidos varchar(25) not null,
  FOREIGN KEY(id_bus_pas) REFERENCES bus(id_bus)
);
```

Android y SQLite

- Android SDK nos provee una serie de clases para administrar nuestro archivo de datos SQLite.
  - **SQLiteOpenHelper**
    - Es una clase abstracta que provee mecanismos básicos para la relación entre la aplicación Android y la información.

Android y SQLite

- Ejemplo de Aplicación Helper

```
public class BaseHelper extends SQLiteOpenHelper {
    public static final String DATABASE_NAME="baseDemo.db";
    public static final int DATABASE_VERSION=1;

    public BaseHelper(Context context){
        super(context,DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        //crear la tabla
        db.execSQL("CREATE TABLE GASTO(CODIGO INTEGER PRIMARY KEY AUTOINCREMENT, NOMBRE TEXT, MONTO INTEGER)");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE GASTO");
        db.execSQL("CREATE TABLE(CODIGO INTEGER PRIMARY KEY AUTOINCREMENT, NOMBRE TEXT, GASTO INTEGER)");
    }
}
```

Android y SQLite

- Ejemplo de Aplicación Clase Gestión I

```
private BaseHelper helper;

public GestionBase(Context context){
    this.helper = new BaseHelper(context);
}
```

Android y SQLite

- Ejemplo de Aplicación Clase Gestión II

```
public String insertar(String nombre, int monto){
    SQLiteDatabase db = helper.getWritableDatabase();
    try {
        ContentValues c = new ContentValues();
        c.put("NOMBRE", nombre);
        c.put("MONTO", monto);
        db.insert("GASTO",null,c);
        //OTRA MANERA
        //db.execSQL(INSERT INTO GASTO VALUES(NULL,X,X));
        return "Transaction OK";
    }catch (Exception e){
        return "Error "+e.getMessage();
    }finally {
        db.close();
    }
}
```

Android y SQLite

- Ejemplo de Aplicación Clase Gestión III

```
public String eliminar(int codigo){
    SQLiteDatabase db = helper.getWritableDatabase();
    db.delete("GASTO", "codigo=?", new String[]{String.valueOf(codigo)});
    db.close();
    return "Gasto eliminado";
}
```

```
public String actualizar(int codigo, String nombre, int monto){
    SQLiteDatabase db = helper.getWritableDatabase();
    ContentValues cv = new ContentValues();
    cv.put("nombre", nombre);
    cv.put("monto",monto);
    db.update("GASTO",cv, "codigo=?", new String[]{String.valueOf(codigo)});
    db.close();
    return "Gasto Actualizado";
}
```

Android y SQLite

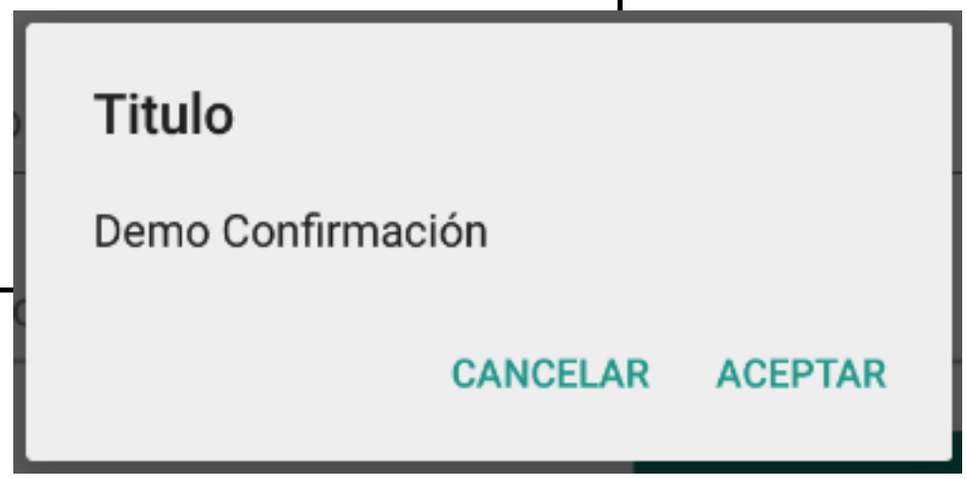
- Ejemplo de Aplicación Clase Gestión IV

```
public ArrayList<String> getGastos(){
    ArrayList<String> lista = new ArrayList<>();
    SQLiteDatabase db = helper.getReadableDatabase();

    Cursor c = db.rawQuery("SELECT * FROM GASTO",null);
    while (c.moveToNext()){
        lista.add(c.getInt(0)+" "+c.getString(1)+" "+c.getString(2));
    }
    db.close();

    return lista;
}
```

Alertas

- Mensajes de Confirmación

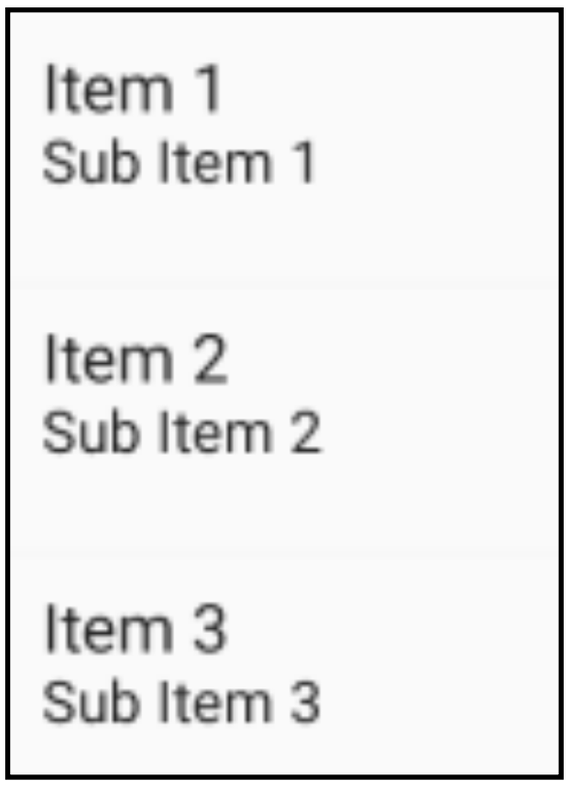
```
AlertDialog.Builder builder=new AlertDialog.Builder(this);
builder.setMessage("Demo Confirmación");
builder.setTitle("Titulo");
builder.setPositiveButton("Aceptar", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {

    }
});

builder.setNegativeButton("Cancelar", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        dialog.cancel();
    }
});
AlertDialog dialog = builder.create();
dialog.show();
```



ListView

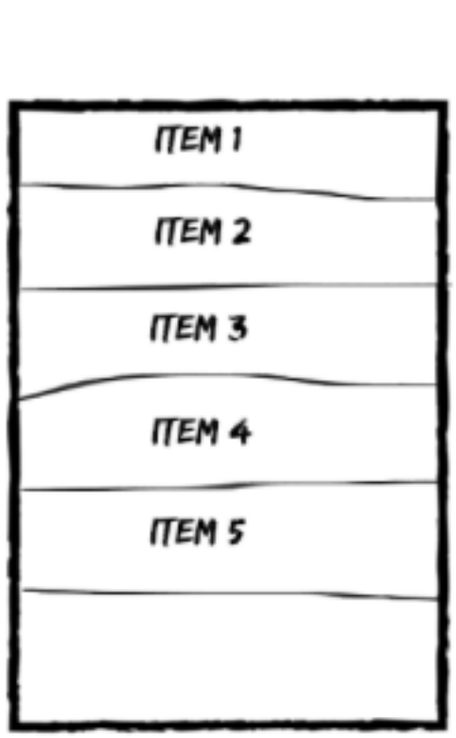
- Control utilizado para cargar un conjunto de datos

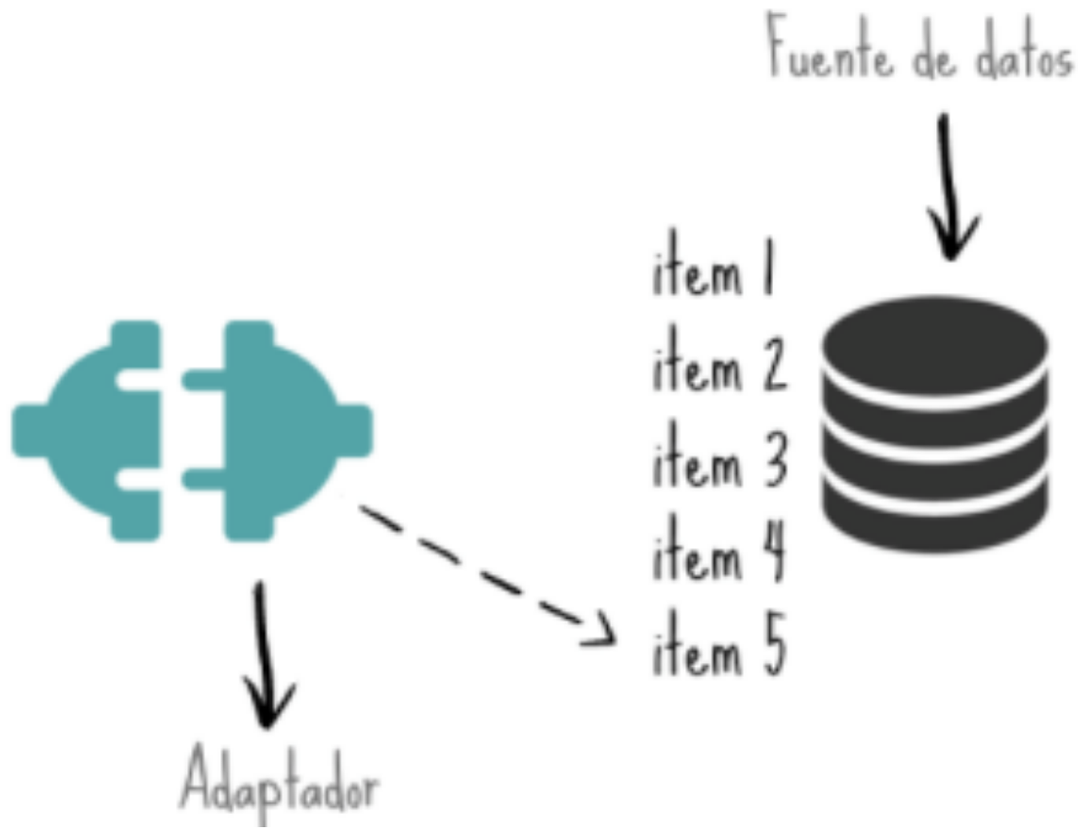
```
<ListView
    android:id="@+id/listaGastos"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

</ListView>
```



Adaptador ListView

- Un adaptador conecta el ListView con una fuente de información.





Adaptador ListView

- Cargar un ListView

```
public void cargarListView(){
    ArrayList<String> lista = gestionBase.getGastos();
    ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                                                            android.R.layout.simple_list_item_1,
                                                            lista);
    listaGastos.setAdapter(adapter);
}
```

Context context: Representa el contexto de la aplicación. Usamos this para indicar que será la actividad que se basa en la creación de la lista.

int resource: Es el recurso de diseño o layout que representará cada fila de la lista. En este caso usamos un recurso del sistema llamado simple_list_item_1.xml. Este layout contiene un solo TextView que contendrá el texto de cada fila.

T[] objects: Es la referencia del array de objetos de tipo T con los cuales crearemos la lista. Si deseas puedes variar el constructor y usar una lista dinámica List<T> en lugar del array.

Evento ListView

- Click sobre un item del ListView

```
listaGastos.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

    }
});
```

El primero es el View que usa al adaptador, en este caso la lista.
El segundo es el View del ítem que ha sido presionado.
El tercero hace referencia a la posición del ítem en la fuente de datos que maneja el adaptador y el cuarto es un identificador del elemento.

PRACTICO