

Curso Android V2.0



ANDROID

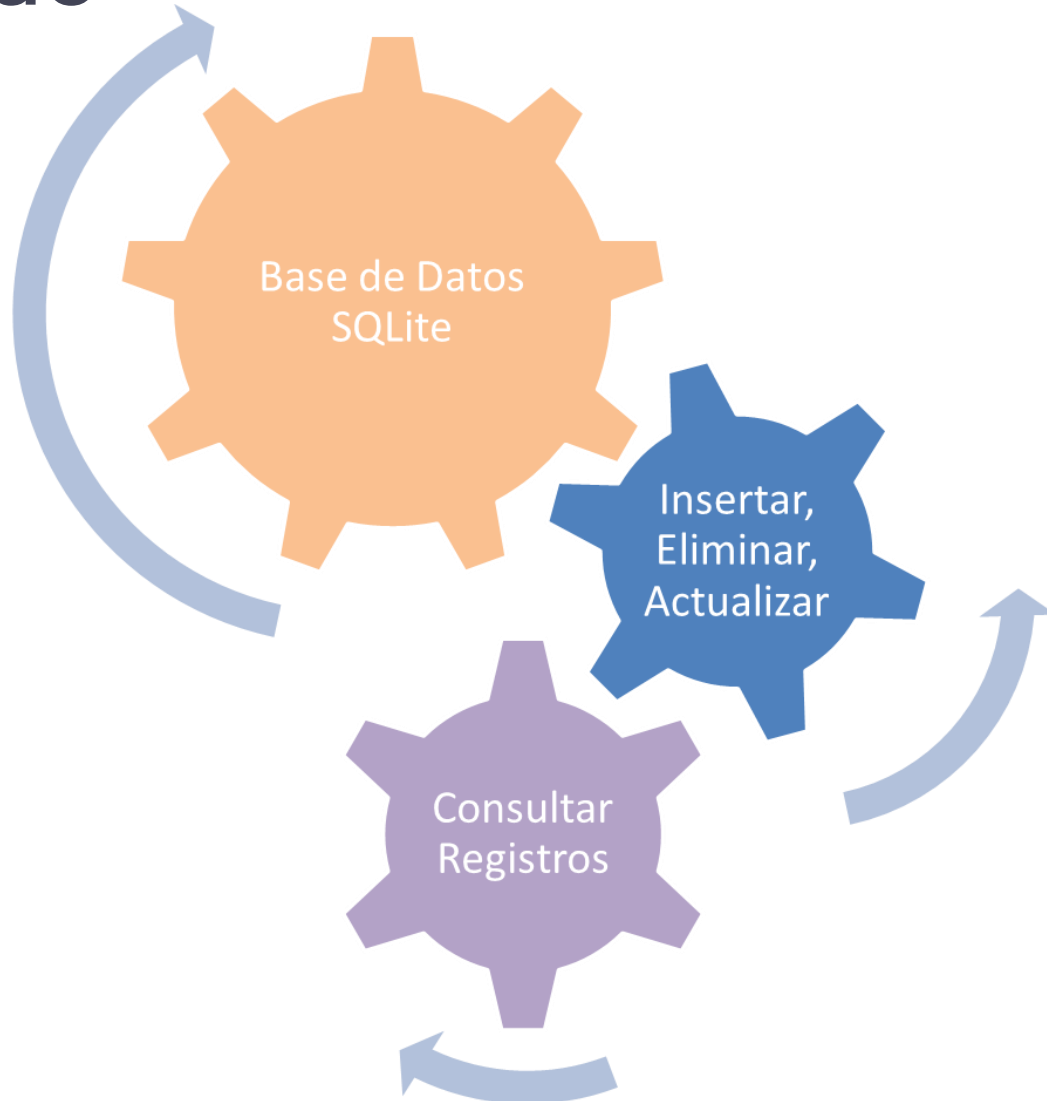
Base de Datos SQLite

Emiliano Gonzalez

Pedro Coronel

2013

Contenido



Base de Datos SQLite

- SQLite es un motor de bases de datos muy popular en la actualidad por ofrecer características tan interesantes como:
 - Tamaño pequeño
 - No necesita servidor
 - Precisa poca configuración
 - Es transaccional
 - Código libre

Base de Datos SQLite

- En Android, la forma típica para crear, actualizar, y conectar con una base de datos SQLite será a través de una clase auxiliar llamada *SQLiteOpenHelper*, o para ser más exactos, de una clase propia que derive de ella y que debemos personalizar para adaptarnos a las necesidades concretas de nuestra aplicación.

Base de Datos SQLite

- La clase SQLiteOpenHelper tiene tan sólo un constructor, que normalmente no necesitaremos sobrescribir, y dos métodos abstractos, onCreate() y onUpgrade(), que deberemos personalizar con el código necesario para crear nuestra base de datos y para actualizar su estructura respectivamente.

Ejemplo

- Crear una base de datos llamada BDUsuarios, con una sola tabla llamada Usuarios que contenga sólo dos campos: nombre y email.

```

1 package py.com.cursoandroid;
2
3 import android.content.Context;
4 import android.database.sqlite.SQLiteDatabase;
5 import android.database.sqlite.SQLiteDatabase.CursorFactory;
6 import android.database.sqlite.SQLiteOpenHelper;
7
8 public class UsuariosSQLiteHelper extends SQLiteOpenHelper {
9
10     //Sentencia SQL para crear la tabla de Usuarios
11     String sqlCreate = "CREATE TABLE Usuarios (codigo INTEGER, nombre TEXT)";
12
13     public UsuariosSQLiteHelper(Context contexto, String nombre,
14                               CursorFactory factory, int version) {
15         super(contexto, nombre, factory, version);
16     }
17
18     @Override
19     public void onCreate(SQLiteDatabase db) {
20         //Se ejecuta la sentencia SQL de creación de la tabla
21         db.execSQL(sqlCreate);
22     }
23
24     @Override
25     public void onUpgrade(SQLiteDatabase db, int versionAnterior, int versionNueva) {
26         //NOTA: Por simplicidad del ejemplo aquí utilizamos directamente la opción de
27         //      eliminar la tabla anterior y crearla de nuevo vacía con el nuevo formato.
28         //      Sin embargo lo normal será que haya que migrar datos de la tabla antigua
29         //      a la nueva, por lo que este método debería ser más elaborado.
30
31         //Se elimina la versión anterior de la tabla
32         db.execSQL("DROP TABLE IF EXISTS Usuarios");
33
34         //Se crea la nueva versión de la tabla
35         db.execSQL(sqlCreate);
36     }
37 }

```

¿Cómo vemos la Base de Datos?

- Todas las bases de datos SQLite creadas por aplicaciones Android se almacenan en la memoria del teléfono en un fichero con el mismo nombre de la base de datos situado en una ruta que sigue el siguiente patrón:
- `/data/data/paquete.java.de.la.aplicacion/databases/nombre_base_datos`
- En nuestro Ejemplo
- `/data/data/py.com.cursoandroid/databases/DBUsuarios`

¿Cómo vemos la Base de Datos?

- Para ello podemos recurrir a dos posibles métodos:
 - Transferir la base de datos a nuestra PC y consultarla con cualquier administrador de bases de datos SQLite.
 - Acceder directamente a la consola de comandos del emulador de Android y utilizar los comandos existentes para acceder y consultar la base de datos SQLite.

Mediante un IDE SQLite

- El fichero de la base de datos podemos transferirlo a nuestra PC utilizando el botón de descarga situado en la esquina superior derecha del explorador de archivos
- Una vez descargado el fichero a nuestro sistema local, podemos utilizar cualquier administrador de SQLite para abrir y consultar la base de datos, por ejemplo SQLite Administrator (freeware).

SQLite Administrator - DBUsuarios2

Database Table Index View Trigger Query Data Help

DBUSUARIOS2

- Tables
 - Usuarios
 - android_metadata
- Indexes
- Views
- Triggers
- Queries

SQL Query Result Edit Data

nombre	email
Usuario1	usuario1@gmail.com
Usuario2	usuario2@gmail.com
Usuario3	usuario3@gmail.com
Usuario4	usuario4@gmail.com
Usuario5	usuario5@gmail.com
Usuario1	usuario1@gmail.com
Usuario2	usuario2@gmail.com
Usuario3	usuario3@gmail.com
Usuario4	usuario4@gmail.com
Usuario5	usuario5@gmail.com
Usuario1	usuario1@gmail.com
Usuario2	usuario2@gmail.com
Usuario3	usuario3@gmail.com
Usuario4	usuario4@gmail.com
Usuario5	usuario5@gmail.com

Query executed within 0ms SQLite: 3.5.1 C:\Users\coronelp\Desktop\DBUsuarios2

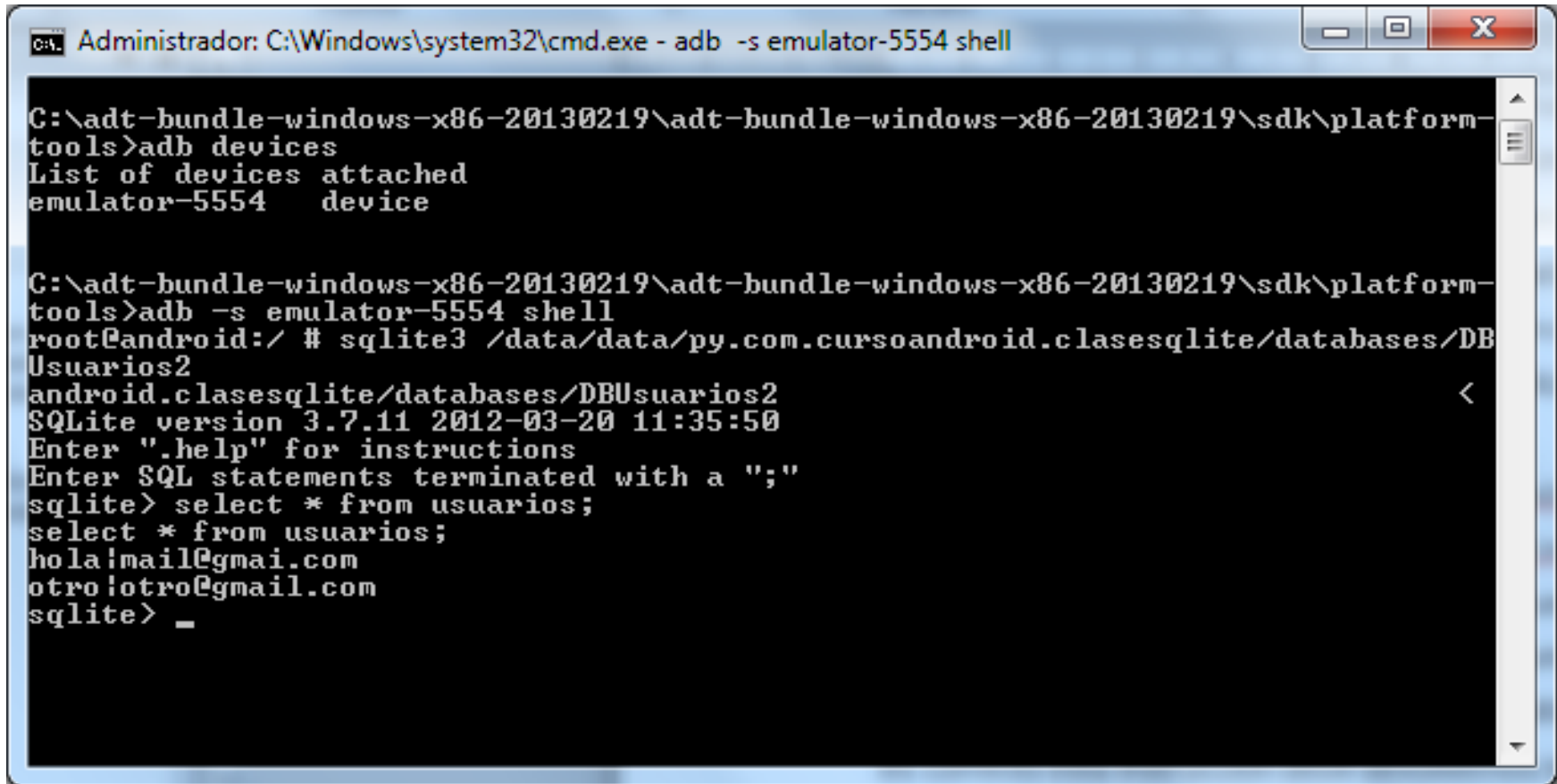
Mediante Consola de Comandos

- Abrir una consola de MS-DOS y utilizar la utilidad adb.exe (*Android Debug Bridge*) situada en la carpeta platform-tools del SDK de Android
 - `c:\android-sdk-windows\platform-tools\`
- Consultar los identificadores de todos los emuladores en ejecución mediante el comando “adb devices”.
- Tras conocer el identificador de nuestro emulador, vamos a acceder a su shell mediante el comando
 - `adb -s identificador-del-emulador shell`
- Acceder a nuestra base de datos utilizando el comando sqlite3 pasándole la ruta del fichero
 - `sqlite3 /data/data/py.com.cursoandroid/databases/DBUsuarios`

Mediante Consola de Comandos

- Debe aparecernos el *prompt* de SQLite lo que nos indicará que ya podemos escribir las consultas SQL necesarias sobre nuestra base de datos
 - `sqlite>`
- Haremos la siguiente consulta, y si todo es correcto esta instrucción debe devolvernos los cinco usuarios existentes en la tabla
 - `SELECT * FROM Usuarios;`

Vista desde línea de comandos



```
C:\Windows\system32\cmd.exe - adb -s emulator-5554 shell

C:\adt-bundle-windows-x86-20130219\adt-bundle-windows-x86-20130219\sdk\platform-tools>adb devices
List of devices attached
emulator-5554    device

C:\adt-bundle-windows-x86-20130219\adt-bundle-windows-x86-20130219\sdk\platform-tools>adb -s emulator-5554 shell
root@android:/ # sqlite3 /data/data/py.com.cursoandroid.clasesqlite/databases/DBUsuarios2
android.clasesqlite/databases/DBUsuarios2
SQLite version 3.7.11 2012-03-20 11:35:50
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> select * from usuarios;
select * from usuarios;
hola!mail@gmail.com
otro!otro@gmail.com
sqlite> _
```

Insertar/Actualizar/Eliminar

- La API de SQLite de Android proporciona dos alternativas para realizar operaciones sobre la base de datos que no devuelven resultados (entre ellas la inserción/actualización/eliminación de registros, pero también la creación de tablas, de índices, etc).

Primera Alternativa

- El método `execSQL()` de la clase `SQLiteDatabase`. Este método permite ejecutar cualquier sentencia SQL sobre la base de datos, siempre que ésta no devuelva resultados

```
1 //Insertar un registro
2 db.execSQL("INSERT INTO Usuarios (usuario,email) VALUES ('usu1','usu1@email.com') ");
3
4 //Eliminar un registro
5 db.execSQL("DELETE FROM Usuarios WHERE usuario='usu1' ");
6
7 //Actualizar un registro
8 db.execSQL("UPDATE Usuarios SET email='nuevo@email.com' WHERE usuario='usu1' ");
```

Segunda Alternativa

- Utilizar los métodos `insert()`, `update()` y `delete()` proporcionados también con la clase `SQLiteDatabase`
- Estos métodos permiten realizar las tareas de inserción, actualización y eliminación de registros de una forma algo más paramétrica que `execSQL()`, separando tablas, valores y condiciones en parámetros independientes de estos métodos.

Insert

```
1 //Creamos el registro a insertar como objeto ContentValues
2 ContentValues nuevoRegistro = new ContentValues();
3 nuevoRegistro.put("usuario", "usu10");
4 nuevoRegistro.put("email", "usu10@email.com");
5
6 //Insertamos el registro en la base de datos
7 db.insert("Usuarios", null, nuevoRegistro);
```

Update

```
1 //Establecemos los campos-valores a actualizar
2 ContentValues valores = new ContentValues();
3 valores.put("email","usu1_nuevo@email.com");
4
5 //Actualizamos el registro en la base de datos
6 db.update("Usuarios", valores, "usuario='usu1'");
```

Delete

```
1 //Eliminamos el registro del usuario 'usu2'
2 db.delete("Usuarios", "usuario='usu2'");
```

Consultar / Recuperar Registros

- Hay dos alternativas principales para recuperar registros de una base de datos SQLite en Android.
 - La primera de ellas utilizando directamente un comando de selección SQL,
 - La segunda utilizando un método específico donde *parametrizaremos* la consulta a la base de datos.

Primera Alternativa

- Utilizaremos el método `rawQuery()` de la clase `SQLiteDatabase`. Este método recibe directamente como parámetro un comando SQL completo, donde indicamos los campos a recuperar y los criterios de selección. El resultado de la consulta lo obtendremos en forma de cursor, que posteriormente podremos recorrer para procesar los registros recuperados

```
1  Cursor c = db.rawQuery(" SELECT usuario,email FROM Usuarios  
2  WHERE usuario='usul' ");
```

Segunda Alternativa

- Utilizar el método `query()` de la clase `SQLiteDatabase`. Este método recibe varios parámetros: el nombre de la tabla, un array con los nombre de campos a recuperar, la cláusula *WHERE*, un array con los argumentos variables incluidos en el *WHERE* (si los hay, null en caso contrario), la cláusula *GROUP BY* si existe, la cláusula *HAVING* si existe, y por último la cláusula *ORDER BY* si existe. Opcionalmente, se puede incluir un parámetro al final más indicando el número máximo de registros que queremos que nos devuelva la consulta.

```
1 String[] campos = new String[] {"usuario", "email"};
2 String[] args = new String[] {"usu1"};
3
4 Cursor c = db.query("Usuarios", campos, "usuario=?", args, null, null, null);
```

Manipulación del Cursor

- Para recorrer y manipular el cursor devuelto por cualquiera de los dos métodos mencionados tenemos a nuestra disposición varios métodos de la clase Cursor, entre los que destacamos dos de los dedicados a recorrer el cursor de forma secuencial y en orden natural:
 - `moveToFirst()`: mueve el puntero del cursor al primer registro devuelto.
 - `moveToNext()`: mueve el puntero del cursor al siguiente registro devuelto.

Manipulación del Cursor

- Una vez posicionados en cada registro podremos utilizar cualquiera de los métodos getXXX(índice_columna) existentes para cada tipo de dato para recuperar el dato de cada campo del registro actual del cursor.

```
1  String[] campos = new String[] {"usuario", "email"};
2  String[] args = new String[] {"usu1"};
3
4  Cursor c = db.query("Usuarios", campos, "usuario=?", args, null, null, null);
5
6  //Nos aseguramos de que existe al menos un registro
7  if (c.moveToFirst()) {
8      //Recorremos el cursor hasta que no haya más registros
9      do {
10         String usuario = c.getString(0);
11         String email = c.getString(1);
12     } while(c.moveToNext());
13 }
```

Ejercicio

- Construir el proyecto anteriormente descrito y crear una pequeña interfaz para el mismo. La interfaz deberá tener las siguientes funcionalidades:
 - Actualizar 1 elemento de la Base de Datos
 - Insertar un elemento nuevo
 - Borrar un elemento existente
 - Listar todos los datos de la tabla Usuarios

Layout

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    tools:context=".MainActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Clase SQLite" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Nombre:" />

    <EditText
        android:id="@+id/numero1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text" >
    </EditText>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Email:" />

    <EditText
        android:id="@+id/numero2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text" >
    </EditText>
```

```
<LinearLayout
    android:id="@+id/linearLayout1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

<Button android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Ins"
    android:id="@+id/Insertar">
</Button>

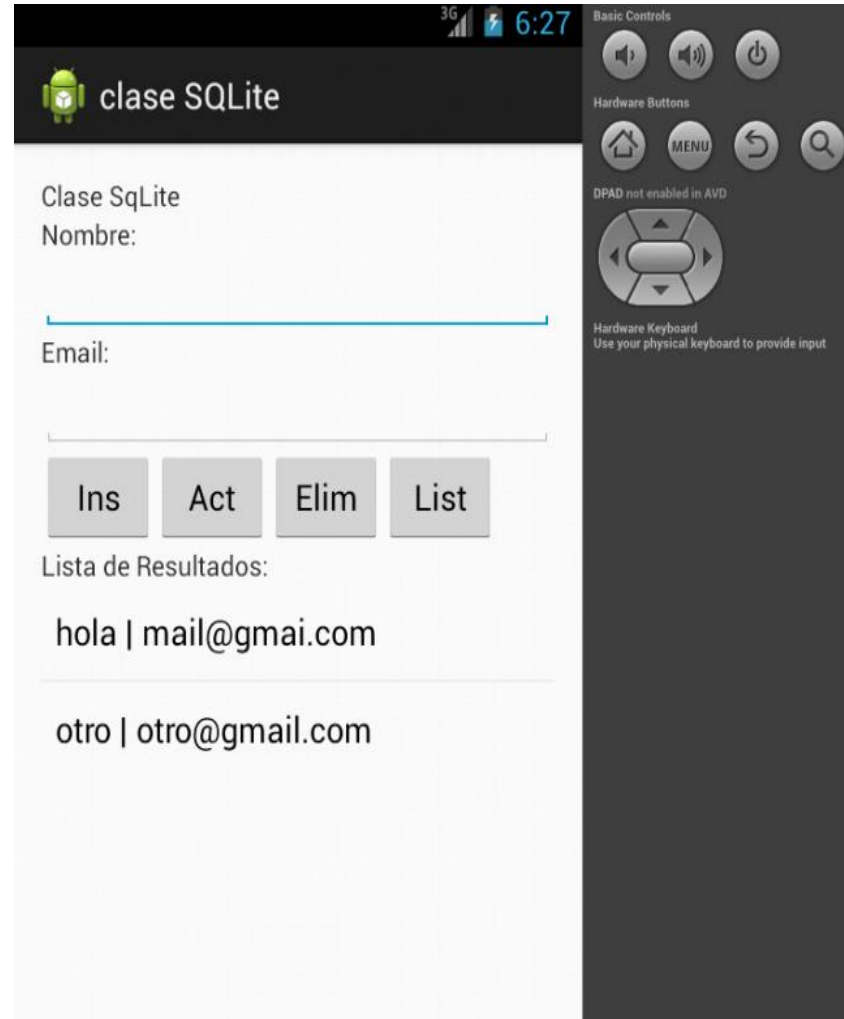
<Button android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Act"
    android:id="@+id/Actualizar">
</Button>

<Button android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Elim"
    android:id="@+id/Borrar">
</Button>

<Button android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="List"
    android:id="@+id/Listar">
</Button>
</LinearLayout>
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Lista de Resultados:" />

<ListView android:id="@+id/LstValores"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
</LinearLayout>
```

Como se vería la Aplicación



Source - DatabaseHelper

```
package py.com.cursoandroid.clasesqlite;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

public class UsuariosSQLiteHelper extends SQLiteOpenHelper {

    //Sentencia SQL para crear la tabla de Usuarios
    String sqlCreate = "CREATE TABLE Usuarios (nombre TEXT, email TEXT)";

    public UsuariosSQLiteHelper(Context contexto, String nombre,
                                CursorFactory factory, int version) {
        super(contexto, nombre, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        //Se ejecuta la sentencia SQL de creación de la tabla
        try{
            db.execSQL(sqlCreate);
        }
        catch(Exception ex)
        {
            Log.e("SQLite", "Error!", ex);
        }
    }
}
```

```
@Override
public void onUpgrade(SQLiteDatabase db, int versionAnterior, int versionNueva) {
    //NOTA: Por simplicidad del ejemplo aquí utilizamos directamente la opción de
    //      eliminar la tabla anterior y crearla de nuevo vacía con el nuevo formato.
    //      Sin embargo lo normal será que haya que migrar datos de la tabla antigua
    //      a la nueva, por lo que este método debería ser más elaborado.

    //Se elimina la versión anterior de la tabla
    db.execSQL("DROP TABLE IF EXISTS Usuarios");

    //Se crea la nueva versión de la tabla
    db.execSQL(sqlCreate);
}
}
```

Main Activity

```
package py.com.cursoandroid.clasesqlite;
import android.app.Activity;
import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.TextView;
```

```
public class MainActivity extends Activity {

    private Button btnInsertar;
    private Button btnActualizar;
    private Button btnEliminar;
    private Button btnListar;

    private EditText txtNombre;
    private EditText txtEmail;

    private ListView lstUsuarios;
    SQLiteDatabase db;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //Abrimos la base de datos 'DBUsuarios' en modo escritura
        UsuariosSQLiteHelper usdbh =
            new UsuariosSQLiteHelper(this, "DBUsuarios", null, 1);

        db = usdbh.getWritableDatabase();
        CrearEventosOnClick();

        //Si hemos abierto correctamente la base de datos
        if(db != null)
        {
            //Cerramos la base de datos
            // db.close();
        }

    }
}
```


Manejador de Eventos

```
private void CrearEventosOnClick()
{
    btnInsertar = (Button)findViewById(R.id.Insertar);
    btnActualizar = (Button)findViewById(R.id.Actualizar);
    btnEliminar = (Button)findViewById(R.id.Borrar);
    btnListar = (Button)findViewById(R.id.Listar);

    txtNombre = (EditText)findViewById(R.id.TxtNombre);
    txtEmail = (EditText)findViewById(R.id.TxtEmail);

    lstUsuarios = (ListView)findViewById(R.id.LstValores);

    btnInsertar.setOnClickListener(new OnClickListener()
    {
        @Override
        public void onClick(View v) {
            insertarDatos(txtNombre.getText().toString(),txtEmail.getText().toString());
        }
    });

    btnEliminar.setOnClickListener(new OnClickListener()
    {
```

```
@Override
public void onClick(View v) {
    String[] nombre = new String[1];
    nombre[0] = txtNombre.getText().toString();
    eliminarDatos (nombre);

}
});

    btnListar.setOnClickListener(new OnClickListener()
    {
        @Override
        public void onClick(View v) {
            lstUsuarios.setAdapter(listarDatos());
        }
    });

    btnActualizar.setOnClickListener(new OnClickListener()
    {
        @Override
        public void onClick(View v) {
            String[] nombre = new String[1];
            nombre[0] = txtNombre.getText().toString();
            actualizarDatos(nombre,txtEmail.getText().toString());

        }
    });

}
```

Metodos de Base de Datos

```
Private void insertarDatos (String nombre, String email){  
    //Creamos el registro a insertar como objeto ContentValues  
    ContentValues nuevoRegistro = new ContentValues();  
    nuevoRegistro.put("nombre", nombre);  
    nuevoRegistro.put("email",email);  
  
    //Insertamos el registro en la base de datos  
    try  
    {  
        db.insert("Usuarios", null, nuevoRegistro);  
    }  
    catch(Exception ex)  
    {  
        Log.e("SQLite", "Error!", ex);  
    }  
}  
private void actualizarDatos (String[] ArgNombres, String email){  
  
    //OJO: la funcion esta preparada para recibir un solo nombre como argumento.  
    //Actualizar dos registros con update(), utilizando argumentos  
    ContentValues valores = new ContentValues();  
    valores.put("email",email);  
  
    String[] args = new String[1];  
    args = ArgNombres;  
    try{  
        db.update("Usuarios", valores, "nombre=?", args);  
    }  
    catch(Exception ex)  
    {  
        Log.e("SQLite", "Error!", ex);  
    }  
}
```

```
private void eliminarDatos (String[] ArgNombres){

String[] args = new String[1];
args = ArgNombres;
try{
db.delete("Usuarios", "nombre=?", args);
}
catch(Exception ex)
{
    Log.e("SQLite", "Error!", ex);
}
}

private ArrayAdapter<String> listarDatos()
{

String[] campos = new String[] {"nombre", "email"};
String[] Usuarios;
//String[] args = new String[] {"usu1"}; --si necesitamos filtrar por algun valor aqui deberian ir los valores

Cursor c = db.query("Usuarios", campos,null, null, null, null, null);

Usuarios = new String[c.getCount()];
//Nos aseguramos de que existe al menos un registro
if (c.moveToFirst()) {
    //Recorremos el cursor hasta que no haya más registros
    int i =0;
    do {
        String usuario = c.getString(0);
        String email = c.getString(1);
        Usuarios[i] = "" + usuario + " | " + email;
        i++;
    } while(c.moveToNext());
}

ArrayAdapter<String> adaptador =
    new ArrayAdapter<String>(MainActivity.this,
        android.R.layout.simple_list_item_1, Usuarios);
return adaptador;

}
```

Tarea:

- Realizar el metodo de busqueda de un usuario en particular dado por el nombre de usuario y mostrar los datos del usuario en el listado de Resultados de la interfaz
- Agregar otros campos a la base de datos (nro_celular, usuario_twitter, direccion)