

**UNIVERSIDAD
DE ANTIOQUIA**
1803

Sesión 7. SQLite

Ing. Edwin Andrés Cubillos Vega Msc.



¿Qué es SQL?

- ❖ **SQL = Structured Query Language (Lenguaje de Consulta Estructurado).**
- ❖ **Lenguaje** vinculado con la gestión de **bases de datos de carácter relacional**
- ❖ Permite la especificación de distintas clases de operaciones entre éstas.
- ❖ SQL brinda la posibilidad de realizar consultas con el objetivo de recuperar información de las bases de datos de manera sencilla.

¿Qué es SQLite?

- ❖ SQLite es una herramienta de software libre,
- ❖ Permite almacenar información en dispositivos embebidos de una forma sencilla, eficaz, potente, rápida
- ❖ equipos con pocas capacidades de hardware, como puede ser una PDA o un teléfono celular.



Características principales

- ❖ Sin configuración
- ❖ Sin servidor
- ❖ Un solo archivo
- ❖ Tipos de datos
- ❖ Registros de longitud variable
- ❖ Código fuente de dominio público

No se recomienda para

- ❖ Aplicaciones Cliente/Servidor.
- ❖ Sitios Web con un alto número de datos.
- ❖ Aplicaciones con alta concurrencia.

- ❖ Crear una nueva aplicación
- ❖ Agenda Sencilla
- ❖ Definir el siguiente layout



Creamos una clase que herede de SQLiteOpenHelper la cual contendrá un constructor y dos métodos: onCreate() y onUpgrade().

```
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

/**...*/
public class ContactosSQLiteHelper extends SQLiteOpenHelper{

    public ContactosSQLiteHelper(Context context, String name, SQLiteDatabase.CursorFactory factory,
                                int version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {

    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {

    }
}
```

- ❖ Creamos dos variables un String para el nombre de la base de datos y un Int para la versión

```
private String DATA_BASE_NAME = "AgendaBD";  
private int DATA_VERSION = 1;
```

- ❖ Luego creamos un String con una sentencia SQL para crear una tabla

```
String sqlCreate = "CREATE TABLE Contactos (" +  
    "id            INTEGER PRIMARY KEY AUTOINCREMENT, " +  
    "nombre        TEXT," +  
    "telefono      INTEGER," +  
    "correo        TEXT)";
```


- ❖ Ejecutamos la sentencia SQL para crear la tabla, esto se hace en el método onCreate

```
@Override
public void onCreate(SQLiteDatabase sqLiteDatabase) {
    sqLiteDatabase.execSQL(sqlCreate);
}
```

Utilizamos el método onUpgrade para actualizar o crear la base de datos sino existe

```
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS Contactos");
    db.execSQL(sqlCreate);
}
```

```
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class ContactosSQLiteHelper extends SQLiteOpenHelper {

    private String DATA_BASE_NAME = "AgendaBD";
    private int DATA_VERSION=1;

    String sqlCreate = "CREATE TABLE Contactos (" +
        "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
        "nombre TEXT, " +
        "telefono INTEGER, " +
        "correo TEXT) ";

    public ContactosSQLiteHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) { db.execSQL(sqlCreate); }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS Contactos");
        db.execSQL(sqlCreate);
    }
}
```

- ❖ Se instancia contactos para el Helper de la base de datos y dbContactos que es la base de datos a trabajar

```
ContactosSQLiteHelper contactos;  
SQLiteDatabase dbContactos;
```

- ❖ Se crea el objeto del helper y se configura la versión editable de la base de datos

```
contactos = new ContactosSQLiteHelper(this, "ContactosBD", null, 1);  
dbContactos = contactos.getWritableDatabase();
```

- ❖ Creamos los objetos EditText y Button que necesitamos

```
EditText eNombre, eTelefono, eCorreo;  
Button bInsertar, bActualizar, bBorrar, bBuscar;
```

- ❖ Y tres variables para almacenar los datos de los EditText

```
String nombre, correo;  
int telefono;
```

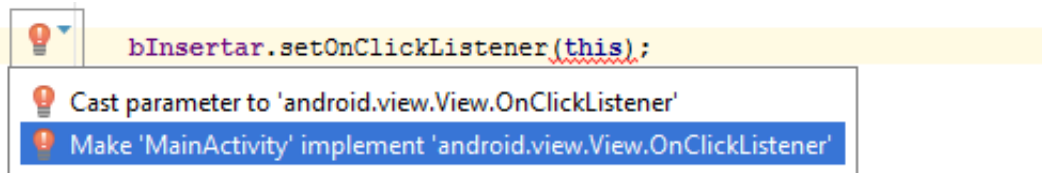
- ❖ Configuramos los EditText y Buttons para tener acceso desde el Main Activity

```
eNombre = (EditText) findViewById (R.id.eNombre);  
eTelefono = (EditText) findViewById (R.id.eTelefono);  
eCorreo = (EditText) findViewById (R.id.eMail);  
bInsertar = (Button) findViewById(R.id.bInsertar);  
bActualizar = (Button) findViewById(R.id.bActualizar);  
bBorrar = (Button) findViewById(R.id.bBorrar);  
bBuscar = (Button) findViewById(R.id.bBuscar);
```

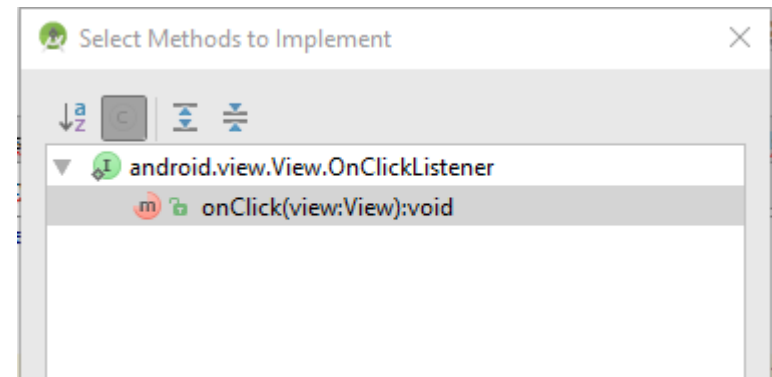
- ❖ Configuramos para que el listener de los botones sea el mismo para los 4

```
bInsertar.setOnClickListener(this);  
bActualizar.setOnClickListener(this);  
bBorrar.setOnClickListener(this);  
bBuscar.setOnClickListener(this);
```

- ❖ Y hacemos que el Main Activity implemente el Listener dando click sobre (this) que esta de color rojo



```
@Override
public void onClick(View view) {
}
}
```

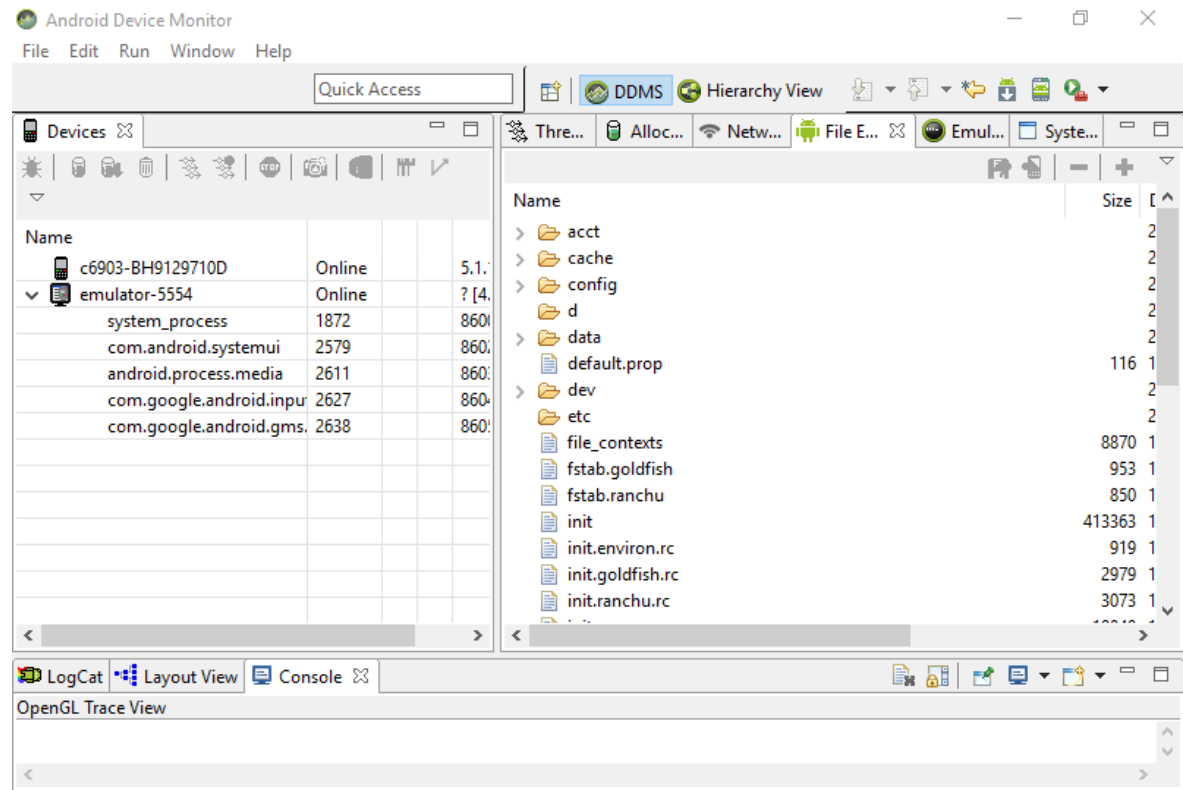


```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {
```

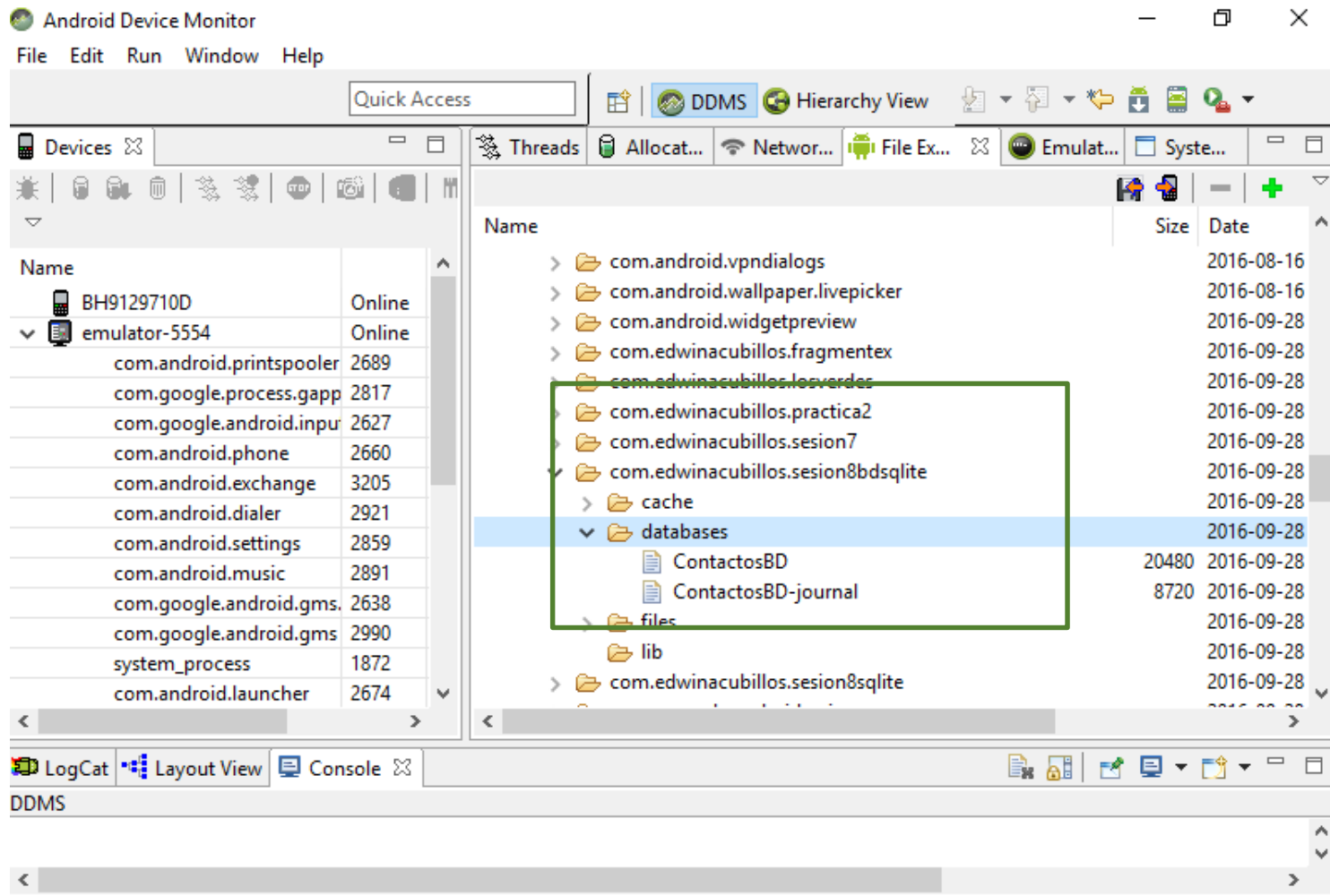
- ❖ Creamos un switch dentro del método onClick para cada uno de los botones:

```
@Override
public void onClick(View view) {
    switch (view.getId()) {
        case R.id.bInsertar:
            break;
        case R.id.bActualizar:
            break;
        case R.id.bBuscar:
            break;
        case R.id.bBorrar:
            break;
    }
}
```

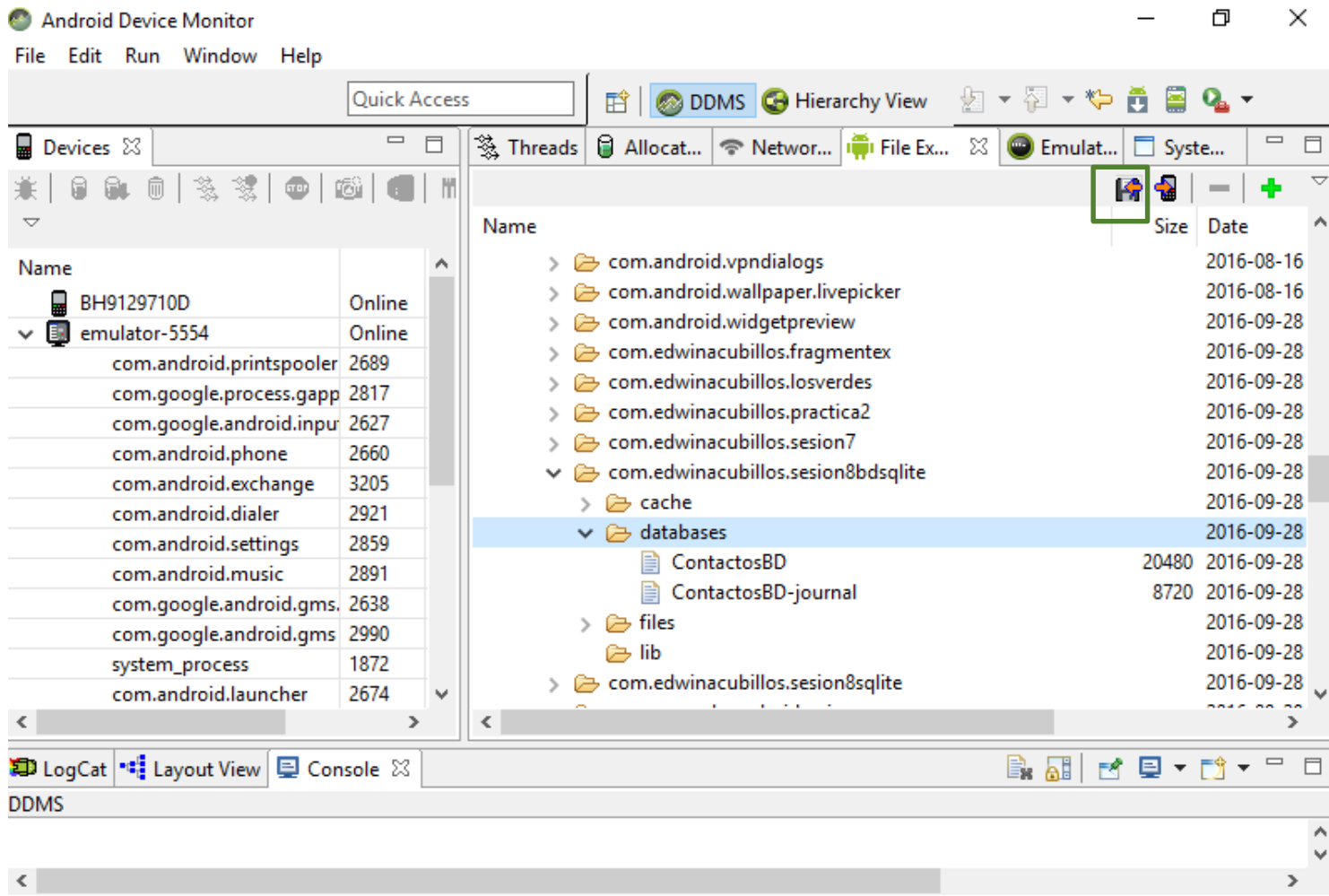
- ❖ Vamos a probar el programa como esta para verificar que si se esta creando la base de datos y la tabla
- ❖ Vamos a Toos - > Android -> Android Device Monitor



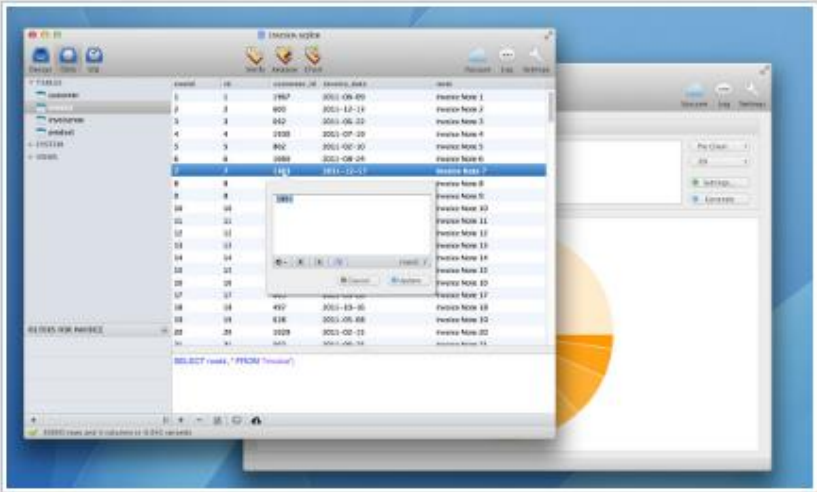
Ir al directorio data -> data - > "paquete"



❖ Exportamos la base de datos



❖ Entrar a la página SQLABS y descargar el SQLiteManager



SQLiteManager is a powerful database management system for sqlite databases, it combines an easy to use interface with blazing speed and [advanced features](#). SQLiteManager allows you to work with a wide range of sqlite 3 databases (like plain text, encrypted databases, SQLCipher or databases).

www.sqlabs.com/download/SQLiteManager4Setup.exe

Details

- Overview
- Features
- Encryption
- Upgrade
- Older Versions
- Version History

Download

Current version: 4.6.6
Release date: September 23, 2016

[MacOS X Version](#)
[Windows Version 4.6.5](#)

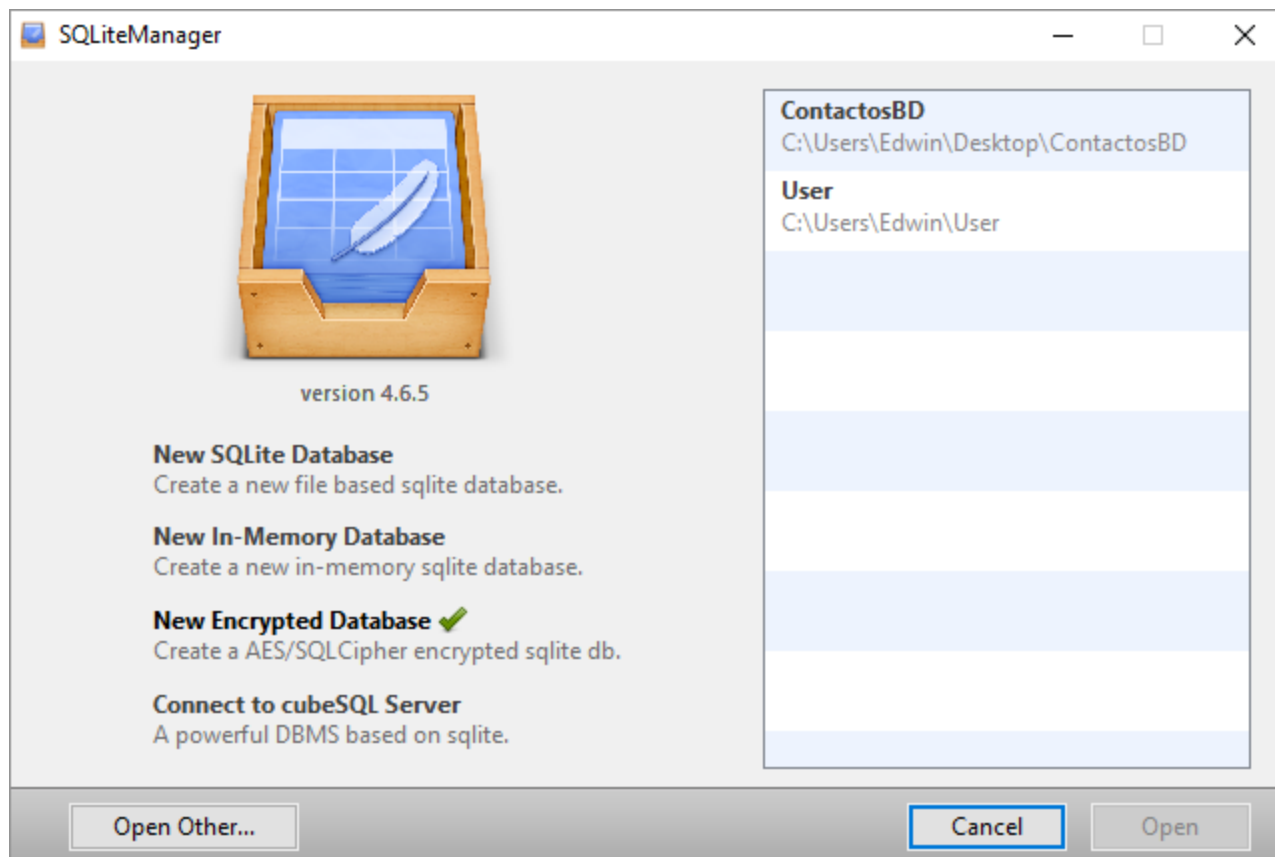
Windows download link

Requirements

❖ Abrir el programa SQLiteManager



- ❖ Damos click en "Open Other" y seleccionamos la base de datos que exportamos de Android



❖ Verificamos que si esta la tabla creada

The screenshot shows the SQLite Designer application window titled 'ContactosBD'. The interface includes a menu bar (File, Edit, Database, GoTo, Script, Window, Help) and a toolbar with icons for Design, Data, SQL, Verify, Analyze, Chart, Vacuum, Log, and Settings. The 'Data' tab is selected, displaying a table named 'Contactos' with the following columns: rowid, id, nombre, telefono, and correo. The table is currently empty. Below the table, the 'FILTERS FOR CONTACTOS' section is visible. The SQL editor at the bottom shows the query: `SELECT rowid, * FROM "Contactos";`. The status bar at the bottom indicates: `✓ 0 rows and 5 columns in 0,031 seconds`.



- ❖ Cargamos el contenido de los EditText en las variables creadas

```
@Override  
public void onClick(View view) {  
    int id = view.getId();  
  
    nombre = eNombre.getText().toString();  
    correo = eCorreo.getText().toString();  
    telefono = eTelefono.getText().toString();  
}
```

- ❖ Y empezamos a programar el CRUD

❖ Creamos un objeto tipo ContentValues

```
ContentValues dataBD;
```

❖ Agregamos los datos a almacenar con put

```
case R.id.bInsertar:

    dataBD = new ContentValues();
    dataBD.put("nombre", nombre);
    dataBD.put("telefono", telefono);
    dataBD.put("correo", correo);

    dbContactos.insert("Contactos", null, dataBD);
    dbContactos.execSQL("INSERT INTO Contactos VALUES(null, '"+nombre+"', '"+telefono+"', " +
        " '"+correo+"')");

    break;
```

Sentencia SQL para insertar

❖ Para Actualizar

```
case R.id.bActualizar:

    dataBD = new ContentValues();
    dataBD.put("telefono", telefono);
    dataBD.put("correo", correo);

    // dbContactos.update("Contactos", dataBD, "nombre='"+nombre+"'", null);
    dbContactos.execSQL("UPDATE Contactos SET Telefono='"+telefono+"', correo='"+correo+"' -
        "WHERE nombre = '"+nombre+"'");

    break;
```

❖ Para Buscar

```
case R.id.bBuscar:

    Cursor c = dbContactos.rawQuery("select * from Contactos where nombre='" + nombre + "'", null);

    if (c.moveToFirst()) {
        eTelefono.setText(c.getString(2));
        eCorreo.setText(c.getString(3));
    }

    break;
```

❖ Para Eliminar

```
case R.id.bBorrar:  
    dbContactos.delete("Contactos", "nombre='"+nombre+"'", null);  
  
    break;
```