Tarea 9: La Base de Datos SQLite.

CONCEPTOS

SQLite es una base de datos SQL de código abierto, que almacena los datos a un archivo de texto en un dispositivo. Android viene interconstruido en la aplicación de base de datos SQLite. SQLite es compatible con todas las características de bases de datos relacionales.

Para poder acceder a esta base de datos, no es necesario establecer ningún tipo de conexiones con ella. El paquete principal es `android.database.sqlite` que contiene las clases para gestionar bases de datos propias.

Se necesita crear un método para invocar a la base de datos `openOrCreateDatabase`, con el nombre de base de datos y el modo como parámetros, y que devuelve una instancia de la base de datos SQLite, que tiene que recibir el objeto.

Método	Descripción
`openDatabase(String ruta, SQLiteDatabase.CursorFactory cf, int flags, DatabaseErrorHandler deh)`	Abre la base de datos actual con el modo `flag`, que puede ser, por ejemplo `OPEN_READWRITE OPEN_READONLY`.
`openDatabase(String ruta, SQLiteDatabase.CursorFactory cf, int flags)`	Abre la base de datos actual pero no define un control de errores para la base de datos.
`openOrCreateDatabase(String ruta, SQLiteDatabase.CursorFactory cf)`	Abre pero crea la base de datos si ésta no existe. Es similar al método `openDatabase`.
`openOrCreateDatabase(File file, SQLiteDatabase.CursorFactory factory)`	Toma el objeto `File` como una ruta en lugar de una cadena. Es similar a `file.getPath`().
`execSQL(String sql, Object[] bindArgs)`	Sólo inserta datos, pero también se usa para actualizar o modificar datos ya existentes usando argumentos de enlace.
`getColumnCount()`	Regresa el número total de columnas de la tabla.
`getColumnIndex(String colName)`	Regresa el índice de la columna, especificando el nombre de la columna.
`getColumnName(int columnIndex)`	Regresa el nombre de la columna, especificando el índice de la columna.
`getColumnNames()`	Regresa un arreglo con todos los nombre de las columnas de la tablas.
`getCount()`	Regresa el número total de filas en el cursor.
`getPosition()`	Regresa la posición actual del cursor en la tabla.
`isClosed()`	Regresa true si el cursor está cerrado y regresa `false` en cualquier otro estado.

Otros métodos útiles
`mydatabase.execSQL("CREATE TABLE IF NOT EXISTS usuarios(User VARCHAR,Pwd VARCHAR);");`
`mydatabase.execSQL("INSERT INTO usuarios VALUES('admin','admin');");`
`Cursor resultSet = mydatbase.rawQuery("Select * from MiTabla",null);` `resultSet.moveToFirst();`
`String username = resultSet.getString(1); String password = resultSet.getString(2);`

Ejemplo del uso de la clase `SQLiteOpenHelper`:

```
public class DBHelper extends SQLiteOpenHelper {
      public DBHelper(){
          super(context,DATABASE_NAME,null,1);
          public void onCreate(SQLiteDatabase db) {}
          public void onUpgrade(SQLiteDatabase database, int oldVersion, int newVersion) {}
       }
}
```

Pasos para ver Archivo de Base de Datos en Android Studio:

1. Seleccionar View -> Tool Windows -> Device File Explorer, en el menú principal de Android Studio.
2. Seleccionar data -> data ->.
   a. En esta carpeta buscar el paquete, por ejemplo com.example.x.x, según se haya nombrado al paquete en la configuración del proyecto. Dentro de esta carpeta se encuentran todos los archivos del proyecto completos, incluyendo los archivos de los datos almacenados, por ejemplo los txt, xml y sql.

EJEMPLO 1.

Paso 1. Crear un nuevo proyecto en Android Studio. En la carpeta java/com.example.mipaquete, abrir y modificar el archivo `MainActivity.java` con el siguiente código:

```
import android.os.Bundle;
import android.app.Activity;
import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.*;
public class MainActivity extends Activity {
      EditText    jetI, jetN;
      Button      jbnA, jbnL;
      TextView jtvL; SQLiteDatabase sqld; @Override
      protected void onCreate(Bundle b) {
            super.onCreate(b);
            setContentView(R.layout.activity_main);
            jetI = (EditText) findViewById(R.id.xetI);
            jetN = (EditText) findViewById(R.id.xetN);
            jbnA = (Button) findViewById(R.id.xbnA);
            jbnL = (Button) findViewById(R.id.xbnL);
            jtvL = (TextView) findViewById(R.id.xtvL);
            DbmsSQLiteHelper dsqlh = new DbmsSQLiteHelper(this, "DBContactos", null, 1);
            sqld = dsqlh.getWritableDatabase();
            jbnA.setOnClickListener(new OnClickListener() {
                  public void onClick(View v) {
                        String      id = jetI.getText().toString();
                        String      nombre = jetN.getText().toString();
                        ContentValues cv = new ContentValues();
                        cv.put("id", id);
                        cv.put("nombre", nombre);
                        sqld.insert("Contactos", null, cv);
                        jetI.setText("");
                        jetN.setText("");
                  }
            });
            jbnL.setOnClickListener(new OnClickListener() {
                  public void onClick(View v) { String id, nombre;
                        Cursor c = sqld.rawQuery("SELECT id,nombre FROM Contactos", null);
                        jtvL.setText("");
                        if (c.moveToFirst()) {
                              do {
                                    id = c.getString(0); nombre = c.getString(1);
                                    jtvL.append(" " + id + "\t" + nombre + "\n");
                              } while(c.moveToNext());
                        }
                  }
            });
      }
```

```
}
```

Paso 2. En la carpeta java/com.examples.mipaquete, crear el archivo `DbmsSQLiteHelper.java` y agregar el siguiente código:

```
import android.content.Context;
import android.database.sqlite.*;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
public class DbmsSQLiteHelper extends SQLiteOpenHelper {
	String sqlCreate = "CREATE TABLE Contactos (id INTEGER, nombre TEXT)";
	public DbmsSQLiteHelper(Context c, String s, CursorFactory cf, int v){
		super(c, s, cf, v);
	}
	@Override
	public void onCreate(SQLiteDatabase db){ db.execSQL(sqlCreate);
	}
	@Override
	public void onUpgrade(SQLiteDatabase sqld, int ov, int nv) {
		sqld.execSQL("DROP TABLE IF EXISTS Contactos"); sqld.execSQL(sqlCreate);
	}
}
```

Paso 3. En la carpeta res/values, abrir el archivo activity_main.xml para modificarlo con el siguiente código:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
	xmlns:android="http://schemas.android.com/apk/res/android"
	android:id="@+id/xll1"
	android:layout_width="match_parent"
	android:layout_height="match_parent"
	android:orientation="vertical" >
	<TextView
		android:id="@+id/xtvI"
		android:layout_width="wrap_content"
		android:layout_height="wrap_content"
		android:text="@string/sid" />
	<EditText
		android:id="@+id/xetI"
		android:layout_width="wrap_content"
		android:layout_height="wrap_content"
		android:inputType="number" />
	<TextView
		android:id="@+id/xtvN"
		android:layout_width="match_parent"
		android:layout_height="wrap_content"
		android:text="@string/snom" />
	<EditText
		android:id="@+id/xetN"
		android:layout_width="match_parent"
		android:layout_height="wrap_content"
		android:inputType="text"/>
	<Button
		android:id="@+id/xbnA"
		android:layout_width="match_parent"
		android:layout_height="wrap_content"
		android:text="@string/ins" />
	<Button
		android:id="@+id/xbnL"
		android:layout_width="match_parent"
		android:layout_height="wrap_content"
```

```
            android:text="@string/cons" />
      <TextView
            android:id="@+id/xtvL"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:text="" />
</LinearLayout>
```

Paso 4. En la carpeta res/values, abrir el archivo strings.xml para modificarlo con el siguiente código:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
      <string name="app_name">Android Base Datos</string>
      <string name="action_settings">Settings</string>
      <string name="hello_world">Hello world!</string>
      <string name="snom">Nombre:</string>
      <string name="sid">ID:</string>
      <string name="ins">Alta</string>
      <string name="cons">Lista</string>
</resources>
```

Paso 5. Por último, ejecutar la aplicación. Al inicio, se muestra solamente el contenido vacío del ID y del Nombre. Agregar un ID y un nombre y enseguida digitar el botón de Alta. Después se pueden agregar más datos para posteriormente, o en cualquier momento, mostrar la lista de elementos digitando el botón Lista, como se muestra en las siguientes imágenes:

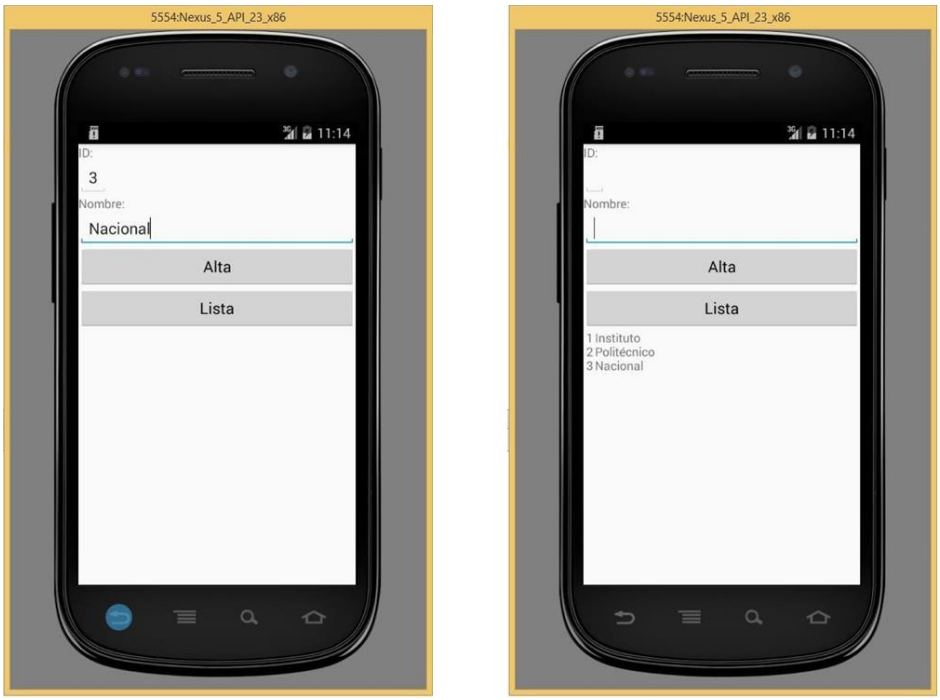

Figura 1. Imágenes que muestran los datos almacenados.

EJERCICIO.
Completar el ejemplo anterior, añadiendo los necesarios que permitan los cambios y las eliminaciones de elementos de la base de datos. Las instrucciones poseen la misma sintaxis que las utilizadas en una base de datos relacional distribuida, las cuales consisten en alta, baja, cambio y selección de elementos con las instrucciones `insert`, `delete`, `update` y `select`, respectivamente.

Nota. Generar un reporte con la inclusión de las imágenes de la ejecución de las aplicaciones. Guardar el documento con la sintaxis AlumnoTarea9Grupo.pdf y enviarlo al sitio indicado por el profesor