

**INSTITUTO POLITÉCNICO NACIONAL**

**ESCUELA SUPERIOR DE CÓMPUTO**

**ALUMNO:**

**ALDAVERA GALLAGA IVÁN**

**APPLICATION DEVELOPMENT FOR MOBILE DEVICES**

**PROFESOR:**

**ALEJANDRO SIGFRIDO CIFUENTES ÁLVAREZ**

**TÍTULO DEL REPORTE:**

**SQLITE**

**FECHA DE REALIZACIÓN:**

**03 – JUNIO – 2020**

**VERSIÓN DEL REPORTE:**

**V1**

**ÍNDICE**

OBJETIVO………………………………………………………………..………….1

CONCEPTOS……………………………………………………………..…………1

DESARROLLO……………………………………………………….….…………..2

Ejemplo 1………………………………………………..…….……………..3

Software Utilizado………………………………………….………………..3

Hardware Utilizado………………………………………….……………….3

PRUEBAS…………………………………………………….………….…………..4

CONCLUSIONES……………………………………………..…………………….4

BIBLIOGRAFÍA………………………………………………..……………………..4

**OBJETIVO**

Entender el concepto de una base de datos SQL propia de Android conocida como SQLite para almacenar datos y archivos en un dispositivo móvil que cuente con el sistema operativo Android.

**CONCEPTOS**

Concepto:

* **SQLite**: Base de datos SQL de código abierto. Almacena datos en un archivo de texto del dispositivo.

Algunas de sus características:

* Android posee la implantación de la base de datos SQLite.
* SQLite permite todas las características de una base de datos relacional.
* Para tener acceso a la base de datos no se necesita establecer alguna conexión.
* El paquete android.database.sqlite contiene las clases para administrar las bases de datos.
* Se crea una base de datos invocando al método openOrCreateDatabase() con los parámetros del nombre de la base de datos y el modo.
* Se regresa una instancia de la base de datos SQLite la cual se debe recibir en un objeto propio.

Los métodos principales de SQLite son:

|  |  |
| --- | --- |
| **MÉTODO** | **DESCRIPCIÓN** |
| *openDatabase(String ruta, SQLiteDatabase.CursorFactory cf, int flags)* | Abre la base de datos actual en el modo. El modo puede ser OPEN\_READWRITE OPEN\_READONLY. |
| *openDatabase(String ruta, SQLiteDatabase.CursorFactory cf, int flags)* | Abre la base de datos actual pero no define algún manejo de errores en la base de datos. |
| *openOrCreateDatabase(String ruta, SQLiteDatabase.CursorFactory cf)* | Abre la base de datos o la crea si no existe. Es equivalente al método openDatabase. |
| *openOrCreateDatabase(File file, SQLiteDatabase.CursorFactory factory)* | Toma el objeto File como la ruta y no como cadena. Es equivalente a file.getPath(). |
| *execSQL(String sql, Object[] bindArgs)* | Solamente inserta datos, pero también puede actualizar o modificar datos existentes utilizando los adecuados argumentos con: |
| *mydatabase.execSQL("CREATE TABLE IF NOT EXISTS TutorialsPoint(User VARCHAR, Pwd VARCHAR);");* | Similar a execSQL. |
| *mydatabase.execSQL("INSERT INTO TutorialsPoint VALUES('admin', 'admin');");* | Similar a execSQL |

Otros de sus métodos:

|  |  |
| --- | --- |
| **MÉTODO** | **DESCRIPCIÓN** |
| *getColumnCount()* | Regresa el total de columnas de la tabla. |
| *getColumnIndex(String colName)* | Regresa el índice de la columna al especificar el nombre de la columna. |
| *getColumnName(int columnIndex)* | Regresa el nombre de la columna al especificar el índice de la columna. |
| *getColumnNames()* | Regresa el arreglo de todos los nombres de las columnas de la tabla. |
| *getCount()* | Regresa el 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 de otra forma. |

**DESARROLLO**

Para el desarrollo de esta práctica se tuvo que realizar un ejercicio consistente en ingresar el ID y nombre de una persona, de manera que mediante 2 EditText se escribieran ambos datos, además de un Button para ingresarlos y otro para listarlos, así también varios TextView dinámicos para conformar la lista.

Ejemplo1:

Lo primero que se hizo fue editar el archivo MainActivity.java colocando su respectivo código del ejemplo, posteriormente, también se le cambió el nombre del archivo a DbmsSQLite.java para más comodidad.

En la misma carpeta del archivo anterior se creó otro llamado DbmsSQLiteHelper.java el cual sirve para crear la base de datos y actualizarla.

En la carpeta res/values se modificó el archivo activity\_main.xml y strings.xml para colocar lo referente a la interfaz gráfica y de usuario, así como para las cadenas de texto que más se utilizarían respectivamente.

Con respecto al código cabe destacar que la mayor parte de la lógica recae en el archivo DbmsSQLite.java y en DbmsSQLiteHelper.java pues son los que mantienen la base de datos y su lógica.

En el archivo DbmsSQLite.java vemos la manera de crear la base de datos:

DbmsSQLiteHelper dsqlh = **new** DbmsSQLiteHelper(**this**, "DBContactos", null, 1); sqld = dsqlh.getWritableDatabase();

También creamos un objeto ContentValues para almacenar temporalmente los datos a ingresar (ID, nombre)

ContentValues cv = **new** ContentValues(); cv.put("id", id); cv.put("nombre", nombre); sqld.insert("Contactos", null, cv);

Así como la manera de seleccionar todos los elementos de la base de datos y listarlos:

Cursor c = sqld.rawQuery("SELECT id,nombre FROM Contactos", null);

**if** (c.moveToFirst()) { **do** { id = c.getString(0); nombre = c.getString(1); jtvL.append(" " + id + "\t" + nombre + "\n"); } **while**(c.moveToNext()); }

Mientras que en el archivo DbmsSQLiteHelper.java vemos otras funciones como agregar la tabla principal o borrarla si existe previamente

1. import android.content.Context;
2. import android.database.sqlite.\*;
3. import android.database.sqlite.SQLiteDatabase.CursorFactory;
4. **public** **class** DbmsSQLiteHelper extends SQLiteOpenHelper {
5. String sqlCreate = "CREATE TABLE Contactos (id INTEGER, nombre TEXT)";
6. **public** DbmsSQLiteHelper(Context c, String s, CursorFactory cf, **int** v){
7. super(c, s, cf, v);
8. }
9. @Override
10. **public** **void** onCreate(SQLiteDatabase db){
11. db.execSQL(sqlCreate);
12. }
13. @Override
14. **public** **void** onUpgrade(SQLiteDatabase sqld, **int** ov, **int** nv) {
15. sqld.execSQL("DROP TABLE IF EXISTS Contactos");
16. sqld.execSQL(sqlCreate);
17. }
18. }

LISTA DE SOFTWARE UTILIZADO:

* Java Development Kit 8: Necesario para el desarrollo de aplicaciones Android.
* Android Studio 4.0: IDE para el desarrollo de aplicaciones Android
* Windows 10: Sistema operativo sobre el que corren los anteriores programas.
* Android 10: Sistema Operativo del dispositivo móvil utilizado.

LISTA DE HARDWARE UTILIZADO:

* Laptop HP con GB RAM DDR4 y procesador Intel Core i3.
* Dispositivo Móvil Motorola One con 4GB de RAM y procesador SoC Qualcomm Snapdragon 625.
* Cable tipo C para la transferencia de datos.

**PRUEBAS**

En la realización de esta práctica no se encontró ningún error al llevarla a cabo debido a que el código proporcionado corrió sin ningún problema.

Aún así, se probó en el dispositivo móvil su funcionamiento realizando las siguientes acciones:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

Como se pudo observar la aplicación cumplió con su función.

**CONCLUSIONES**

La base de datos de SQLite nos permite persistir datos en una aplicación que corra en un dispositivo móvil de Android con el tradicional modelo SQL que se utiliza en otras aplicaciones como las aplicaciones web o de escritorio.

Es muy versátil y tiene muchas funciones pues se adapta a cualquier tipo de situación que requiera una base de datos fija y de tamaño pequeño o mediano.

**BIBLIOGRAFÍA**

* Archivo Android CEC SQLite, Cifuentes, A. IPN-ESCOM.