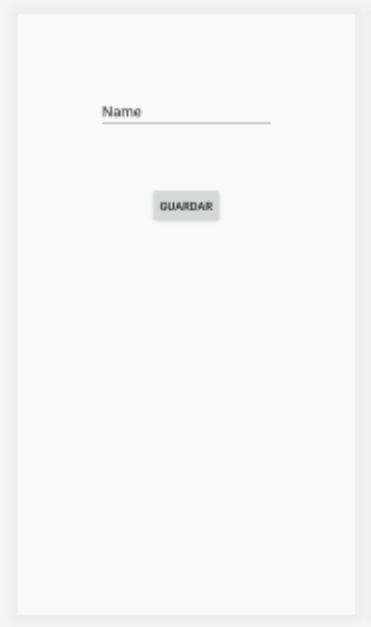
Tutorial.
Persistència amb SQLite

Objectiu

Guardar les dades de l'aplicació a la base de dades SQLite.
Recomanació abans de començar: guarda una còpia de la pràctica tal i com la tens ara amb ArrayList ja que avui modificarem el funcionament de gran part de l'aplicació.

En el següent tutorial realitzarem un petit exemple de com guardar dades a una base de dades.
La pantalla utilitzada és la següent:



Demo

Descarrega la demo aquí

Pas a pas.

main_activity.xml. La pantalla té un camp de text (EditText) i un botó de guardar (Button)

Creem la classe ContactsDBHelper

1. Creem un package (new → package i l'anomenarem DB)
2. Dins d'aquest package crearem una classe java que es dirà ContactsDBHelper que haurà d'extendre de la classe SQLiteOpenHelper

   ```
   public class ContactsDBHelper extends SQLiteOpenHelper
   ```

3. Farem Alt Intro i implementarem els mètodes onCreate i onUpgrade
4. Definirem les següents variables globals

   ```
   public static final int DATABASE_VERSION = 1;
   public static final String DATABASE_NAME = "contacts.db";
   ```

5. Veurem que ens demana un constructor per a la classe, el crearem fent Alt Intro i analitzarem els paràmetres, modificarem el constructor perquè ens creiï una base de dades amb el nom definit i la versió

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

   Els paràmetres de constructor tenen la següent finalitat:

   - Context context: Context d'acció per al helper.
   - String name: Nom de l'arxiu amb extensió .db, on s'emmagatzemarà la base de dades, que al seu torn correspon a el nom de la base de dades.
   - CursorFactory factory: Assignem null, per ara no cal comprendre el funcionament d'aquest paràmetre.
   - int versió: Enter que representa la versió de la base de dades. El seu valor inicial per defecte és 1. Si en algun moment la versió és més gran es diu al mètode onUpgrade () per actualitzar la base de dades a la nova versió. Si és menor, es diu a downUpgrade () per tornar a una versió prèvia

ContactsContract.java

La forma en què una base de dades està estructurada (quantitat de taules, registres, índexs, etc.) i el conjunt de convencions per anomenar-ne els objectes se'ls anomena esquema. En general l'esquema inicial es guarda en un Script que ens permeti recuperar les condicions prèvies en qualsevol moment.

Amb SQLite no és diferent, de manera que has de crear un esquema predefinit per implementar-la a l'hora de crear la teva base de dades.

La documentació d'Android ens recomana crear una classe anomenada Contract Class, la qual guarda com constants totes les característiques de la base de dades.

```
public class ContactsContract  {
   private ContactsContract(){}
   public static class ContactsEntry implements BaseColumns {
       public static final String TABLE_NAME ="contacts";
```

```
        public static final String ID = "id";
        public static final String COLUMN_NAME_TITLE = "name";
    }
}
```

ContactsDBHelper.java

6. En el mètode on create necessitem crear la taula, per a fer-ho definim un string amb la query que haurem d'executar, utilitzarem les variables definides anteriorment (ID, TABLE_NAME i COLUMN_NAME_TITLE)

```
private static final String SQL_CREATE_ENTRIES = "CREATE TABLE " +
ContactsEntry.TABLE_NAME + "(" + ContactsEntry.ID + " INTEGER PRIMARY KEY
AUTOINCREMENT, " + ContactsEntry.COLUMN_NAME_TITLE + " TEXT)";
```

Per fer ús de les variables definides a ContactsContract.java necessitarem importar la classe

```
import com.example.sqlite.DB.ContactsContract.*;
```

Per executar la query definida al String utilitzarem el mètode execSQL

```
public void onCreate(SQLiteDatabase db) {
    db.execSQL(SQL_CREATE_ENTRIES);
}
```

7. Ara ja tenim la taula creada. Necessitem crear un mètode per insertar els Contactes.

```
public void insertContact(SQLiteDatabase db, Contact c){
    //Check the bd is open
    if (db.isOpen()){
            //Creation of the register for insert object with the content
values
        ContentValues values = new ContentValues();

        //Insert the contacts getting all values
        values.put(ContactsEntry.COLUMN_NAME_TITLE, c.getNom());

        db.insert(ContactsEntry.TABLE_NAME, null, values);
    }else{
        Log.i("sql","Database is closed");
    }
}
```

MainActivity.java

1. Des del main activity farem crides a la classe ContactsDBHelper, per això necessitem

declarar-la i crear-la.

```
//Create the instance of dbHelper
private ContactsDBHelper dbHelper;
private SQLiteDatabase db;

//Creation of the dbHelper
dbHelper = new ContactsDBHelper(getApplicationContext());
db = dbHelper.getWritableDatabase();
```

2. En fer click al botó haurem d'agafar el text, crear un contacte i afegir-lo a la base de dades.

   En principi ja hauríeu de crear l'objecte amb les dades del formulari i per tant l'únic que heu de modificar és com afegiu les dades

   ```
   dbHelper.insertContact(db, c);
   ```

3. Quan tanquem l'activitat és important que tanquem també la base de dades per tant haurem de sobreescriure el mètode

   ```
   //Close the db when the activity onDestroy
   @Override
   protected void onDestroy() {
       dbHelper.close();
       db.close();
       super.onDestroy();
   }
   ```

Exercicis:

1. Afegeix un botó que es digui LLISTAR que mostri totes les dades de la taula creada mitjançant un Log
2. Afegeix un botó que es digui ELIMINAR que elimini un