Objective

A data access object (DAO) is used for managing the database connection, accessing data, and modifying data. It will also convert database objects to Java objects**(data model)**. The relational schema is : comments(**_id**,comment)

1. **Practical Tasks**

   (a) Create a new Android Project in Android Studio, named as **DbSQLiteExampleGXX** with a *Empty Activity*, named as **TestDatabaseActivity** which extend class **ListActivity**.

   (b) Open the UI associated with the created activity and (see 1) :

      i. Modify the layout type to **"LinearLayout"** with the attribute value as **orientation ="verticale**".

      ii. Add **LinearLayout** with **orientation ="hrizontal"**.

      iii. Drag 02 *"Button"* view from the palette, with **text attribute values : "Add New", "Delete First" respectively**.

      iv. Put the value **"onClick"** in the attribute **android :onClick** of both buttons to manage the click event.

      v. Drag the following view from the palette(legacy) : *"ListView"* inside the root layout element with **id attribute value : "@android :id/list"**.

   (c) Create the assistance class **CommentDbHelper**, which extend **SQLiteOpenHelper** in order to create the DB **comments.db** (see 2).

   (d) Create the class **Comment**, which represent **data model** (see 3).

   (e) Create the class **CommentsDataSource**, which represent **DAO** (see 4).

   (f) Open *TestDatabaseActivity.java* and (see 5) :

      i. Examine the **onCreate() method** for this activity and modify it to perform the view of data extracted from the data source (DAO) in the UI using "ListActivity".

      ii. Implement **onClick() method** for this activity to handle the event click of 02 buttons views.

   (g) Compile and run the app (see 6).

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <LinearLayout
        android:id="@+id/group"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >

        <Button
            android:id="@+id/add"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Add New"
            android:onClick="onClick"/>

        <Button
            android:id="@+id/delete"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Delete First"
            android:onClick="onClick"/>

    </LinearLayout>

    <ListView
        android:id="@android:id/list"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />

</LinearLayout>
```

```java
public class CommentDbHelper extends SQLiteOpenHelper {
    public static final String TABLE__NAME = "comments";
    public static final String COLUMN_ID = "_id";
    public static final String COLUMN_COMMENT = "comment";

    private static final String DATABASE_NAME = "commments.db";
    private static final int DATABASE_VERSION = 1;

    // Commande sql ( sql statement) pour la création de la table de la base de
données
    private static final String DATABASE_CREATE = "create table "
            + TABLE__NAME + "(" + COLUMN_ID
            + " integer primary key autoincrement, " + COLUMN_COMMENT
            + " text not null);";
    // Commande sql ( sql statement) pour la supprsseion de la table de la base de
données
    private static final String SQL_DELETE_ENTRIES =
            "DROP TABLE IF EXISTS " + TABLE__NAME;

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

    @Override
    public void onCreate(SQLiteDatabase database) {
        database.execSQL(DATABASE_CREATE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        Log.w(CommentDbHelper.class.getName(),
                "Upgrading database from version " + oldVersion + " to "
                        + newVersion + ", which will destroy all old data");
        db.execSQL(SQL_DELETE_ENTRIES);
        onCreate(db);
    }
}
```

```java
package com.example.databasesqliteexample;

public class Comment {
    private long id;
    private String comment;

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getComment() {
        return comment;
    }

    public void setComment(String comment) {
        this.comment = comment;
    }

    // Sera utilisée par ArrayAdapter dans la ListView
    @Override
    public String toString() {
        return comment;
    }
}
```

```
public class CommentsDataSource {
    // Champs de la base de données
    private SQLiteDatabase database;
    private CommentDbHelper dbHelper;
    private String[] allColumns = { CommentDbHelper.COLUMN_ID,
            CommentDbHelper.COLUMN_COMMENT };

    public CommentsDataSource(Context context) {
        dbHelper = new CommentDbHelper(context);
    }

    public void open() throws SQLException {
        database = dbHelper.getWritableDatabase();
    }

    public void close() {
        dbHelper.close();
    }

    public Comment createComment(String comment) {
        ContentValues values = new ContentValues();
        values.put(CommentDbHelper.COLUMN_COMMENT, comment);
        long insertId = database.insert(CommentDbHelper.TABLE__NAME, null,
                values);
        Cursor cursor = database.query(CommentDbHelper.TABLE__NAME,
                allColumns, CommentDbHelper.COLUMN_ID + " = " + insertId, null,
                null, null, null);
        cursor.moveToFirst();
        Comment newComment = cursorToComment(cursor);
        cursor.close();
        return newComment;
    }

    public void deleteComment(Comment comment) {
        long id = comment.getId();
        System.out.println("Comment deleted with id: " + id);
        database.delete(CommentDbHelper.TABLE__NAME, CommentDbHelper.COLUMN_ID
                + " = " + id, null);
    }

    public List<Comment> getAllComments() {
        List<Comment> comments = new ArrayList<Comment>();

        Cursor cursor = database.query(CommentDbHelper.TABLE__NAME,
                allColumns, null, null, null, null, null);

        cursor.moveToFirst();
        while (!cursor.isAfterLast()) {
            Comment comment = cursorToComment(cursor);
            comments.add(comment);
            cursor.moveToNext();
        }
        // assurez-vous de la fermeture du curseur
        cursor.close();    return comments;
    }

    private Comment cursorToComment(Cursor cursor) {
        Comment comment = new Comment(); comment.setId(cursor.getLong(0));
comment.setComment(cursor.getString(1));  return comment; }  }
```

```
public class TestDatabaseActivity extends ListActivity {
    private CommentsDataSource datasource;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        datasource = new CommentsDataSource(this);
        datasource.open();

        List<Comment> values = datasource.getAllComments();

        // utilisez SimpleCursorAdapter pour afficher les
        // éléments dans une ListView
        ArrayAdapter<Comment> adapter = new ArrayAdapter<Comment>(this,
                android.R.layout.simple_list_item_1, values);
        setListAdapter(adapter);
    }
    // Sera appelée par l'attribut onClick
    // des boutons déclarés dans main.xml
    public void onClick(View view) {
        @SuppressWarnings("unchecked")
        ArrayAdapter<Comment> adapter = (ArrayAdapter<Comment>) getListAdapter();
        Comment comment = null;
        switch (view.getId()) {
            case R.id.add:
                String[] comments = new String[] { "Cool", "Very nice", "Hate it"
};
                int nextInt = new Random().nextInt(3);
                // enregistrer le nouveau commentaire dans la base de données
                comment = datasource.createComment(comments[nextInt]);
                adapter.add(comment);
                break;
            case R.id.delete:
                if (getListAdapter().getCount() > 0) {
                    comment = (Comment) getListAdapter().getItem(0);
                    datasource.deleteComment(comment);
                    adapter.remove(comment);
                }
                break;
        }
        adapter.notifyDataSetChanged();
    }

    @Override
    protected void onResume() {
        datasource.open();
        super.onResume();
    }

    @Override
    protected void onPause() {
        datasource.close();
        super.onPause();
    }
}
```

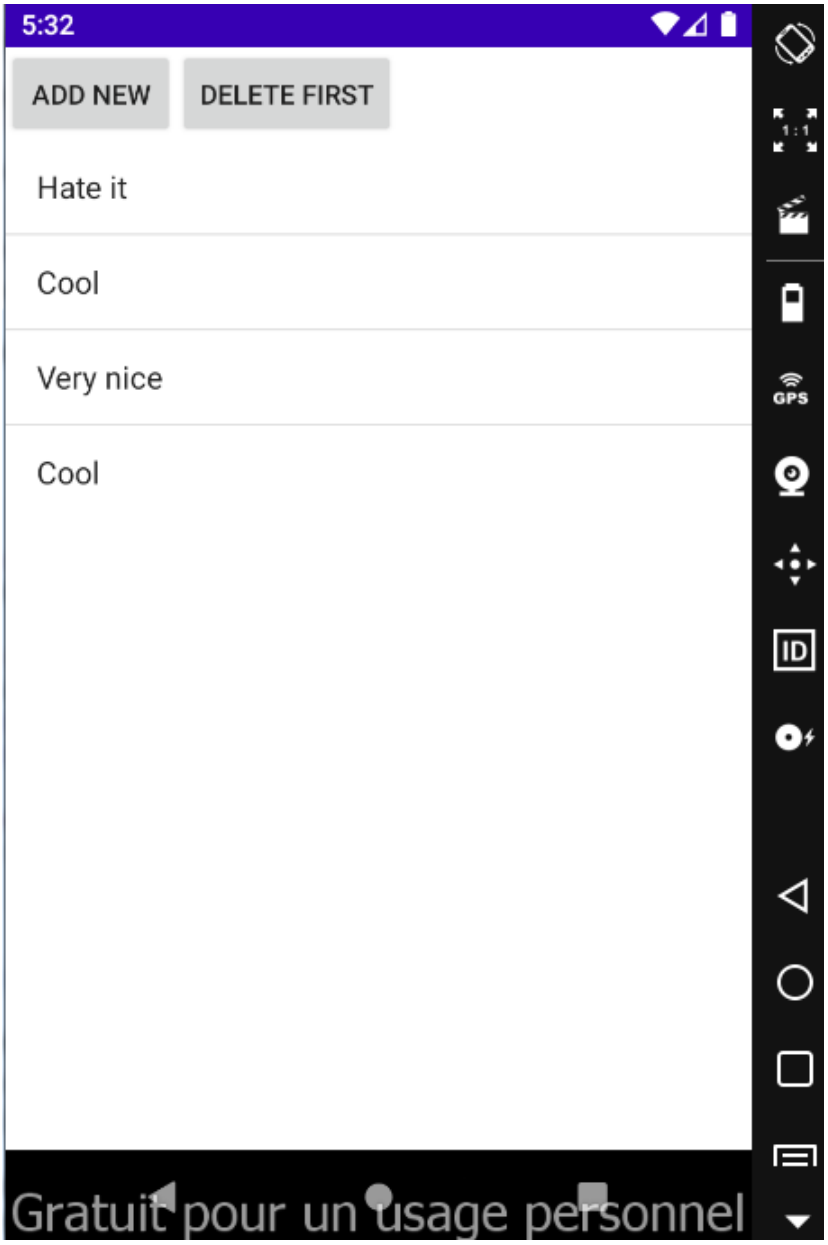

FIGURE 6: Figure of obtained UI