LAB 7: SQLite

I. MỤC TIÊU

- Tạo mới một CSDL SQLite
- Kết nối ứng dụng android với một CSDL SQLite
- Khai thác CSDL SQLite bằng android (đọc, thêm, sửa, xóa dữ liệu)

II. TÓM TẮT LÝ THUYẾT

1. SQLite

SQLite là phiên bản rút gọn của CSDL SQL, được dùng để lưu trữ dữ liệu dưới dạng bảng quan hệ, với mỗi cột tương ứng với một trường hoặc thuộc tính của dữ liệu, mỗi dòng tương ứng với một thể hiện cụ thể của dữ liệu. Tuy chỉ là bản rút gọn nhưng SQLite vẫn đáp ứng được hầu hết các yêu cầu về quản lý, lưu trữ và khai thác dữ liệu. Vì tính chất nhỏ và nhẹ của ứng dụng nên trên hầu hết các hệ điều hành di động hiện nay đều hỗ trợ SQLite thay vì các loại CSDL đầy đủ như SQL hay Oracle. Android đã tích hợp sẵn thư viện cho phép quản lý và khai thác CSDL SQLite.

Để tạo và quản lý cơ sở dữ liệu SQLite ta có thể dùng các chương trình bên thứ 3 như DB Browser for SQLite, SQLiteStudio, SQLite Expert…. Ngoài ra, ta có thể trực tiếp khởi tạo và sử dụng CSDL SQLite trong Android.

2. Nơi lưu trữ CSDL SQLite cho ứng dụng

Để một CSDL SQLite có thể được sử dụng bằng ứng dụng android, CSDL đó phải được lưu trữ tại thư mục

DATA/data/<tên_gói>/databases/<tên_file_SQLite>

3. Thao tác với SQLite sử dụng Room Persistence Library

Room là một Persistence Library được Google giới thiệu trong sự kiện Google I/O 2017, nó là một abstract layer cung cấp cách thức truy cập thao tác với dữ liệu trong cơ sở dữ liệu SQLite. Bình thường để tạo được database ta cần viết các câu lệnh rất dài, mà viết sai một chút thôi là lại phải ngồi rà soát lại ngay.

Đặc điểm của Room database

Framework chính (Sqlite Database) cung cấp các built-in support cho các trường hợp làm việc với các nội dung SQL thô. Mặc dù các API này khá mạnh mẽ nhưng chúng lại tương đối low-level và yêu cầu khá nhiều thời gian và nỗ lực để sử dụng:

- Không có xác thực các câu truy vấn SQL ở thời điểm compile-time. Khi dữ liệu thay đổi thì ta sẽ phải cập nhật lại các câu truy vấn SQL thủ công. Việc này khá mất thời gian và xác suất gặp lỗi trong quá trình khá lớn.
- Sẽ phải dùng nhiều code khung để chuyển đổi giữa truy vấn SQL với các Java data object.

Room sẽ giải quyết cả hai vấn đề này.

Cách import Room

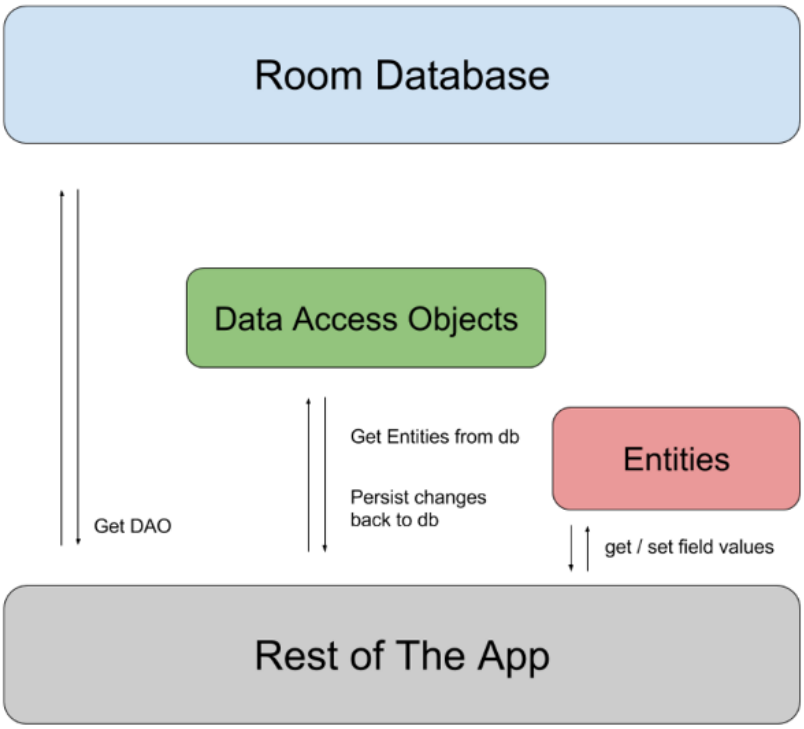
Mở build.gradle (app) và thêm 2 dòng lệnh sau trong dependencies

```
implementation 'android.arch.persistence.room:runtime:1.1.1'
annotationProcessor "android.arch.persistence.room:compiler:1.1.1"
```

Chú ý: nên sử dụng phiên bản Room mới nhất. Phiên bản hiện tại là 1.1.1. Đối với ứng dụng sử dụng androidx.* artifacts thì dùng 2 dòng lệnh sau

```
implementation 'androidx.room:room-runtime:2.1.0'
annotationProcessor 'androidx.room:room-compiler:2.1.0'
```

Các thành phần chính trong Room



- Entity: Component này đại diện cho một class chứa một row của database. Với mỗi một entity thì một database table sẽ được tạo để giữ các items tương ứng. Nên tham chiếu lớp enity thông qua mảng entities trong class Database. Mỗi một trường của enitty sẽ được pesist trong database trừ trường hợp bị chú thích là @Ignore.
  Lưu ý: Các entity có thể hoặc là có hàm khởi tạo rỗng (trường hợp lớp DAO có thể truy cập từng field đã persist) hoặc là hàm khởi tạo với các đối số là các kiểu dữ liệu và tên khớp với một trong các field của entity. Room còn có thể sử dụng hàm khởi tạo đầy đủ hoặc một phần, ví dụ như hàm khởi tạo chỉ nhận một trong các field.
  Ví dụ: Lưu lại toàn bộ các user và add vào database, đối tượng user gồm các thuộc tính name, password và place

User.java

```
@Entity(tableName = "users")
public class User {
    private static final String DEFAULT_PW = "12345678";
    @NonNull
    @PrimaryKey(autoGenerate = true)
    @ColumnInfo(name = "id")
    private int mId;
    @ColumnInfo(name = "first_name")
    private String mFirstName;
    @ColumnInfo(name = "last_name")
    private String mLastName;
    @ColumnInfo(name = "password")
    private String mPassword;
    @Embedded
    private Place mPlace;

    public User() {
    }

    @Ignore
```

```
    public User(String firstName, String lastName) {
        mFirstName = firstName;
        mLastName = lastName;
        mPassword = DEFAULT_PW;
    }

    // Getter and Setter

    @Override
    public String toString() {
        if (mPlace != null) {
            return mFirstName + " " + mLastName + "\n" +
mPlace.getName();
        }
        return mFirstName + " " + mLastName;
    }
}
```

Place.java

```
@Entity(tableName = "place")
public class Place {
    @PrimaryKey(autoGenerate = true)
    private int mId;
    @ColumnInfo(name = "lat")
    private double mLat;
    @ColumnInfo(name = "lng")
    private double mLng;
    @ColumnInfo(name = "name")
    private String mName;

    public Place() {
    }

    public int getId() {
        return mId;
    }

    public void setId(int id) {
        mId = id;
    }

    public double getLat() {
        return mLat;
    }

    public void setLat(double lat) {
        mLat = lat;
    }

    public double getLng() {
        return mLng;
    }

    public void setLng(double lng) {
```

```
        mLng = lng;
    }

    public String getName() {
        return mName;
    }

    public void setName(String name) {
        mName = name;
    }
}
```

+ Primary key: Mỗi Object phải xác định ít nhất 1 trường làm khóa chính. Ngay cả khi chỉ có 1 trường, ta vẫn cần chú thích trường này bằng anotation @PrimaryKey. Ngoài ra, nếu ta muốn Room gán ID tự động cho các thực thể, ta có thể đặt thuộc tính autoGenerate của @ PrimaryKey (trường hợp thuộc tính là int, long)

```
@PrimaryKey(autoGenerate = true)
private int mId;
```

+ Indices and uniqueness: Trường hợp muốn đánh index cho một số trường trong database để tăng tốc độ truy vấn ta có thể sử dụng như sau

```
@Entity(indices = {@Index(value = {"first_name", "last_name"})})
```

Một số trường hợp ta có thể muốn một số trường là duy nhất trong db ví dụ first_name và last_name không thể có bản ghi nào trùng nhau ta có thể thêm unique như sau

```
@Entity(indices = {@Index(value = {"first_name", "last_name"}, unique = true)})
```

+ Nested objects: Trong một số trường hợp ta tạo ra object với các nested object mà không có nhu cầu lưu chúng thành 1 bảng riêng mà đơn giản chỉ giống như 1 column bình thường thì có thể sử dụng anotaion @Embedded cho chúng giống như đã làm cho Place trong object User

```
@Embedded
private Place mPlace;
```

+ Relationships between objects: Định nghĩa foreignKeys. Ví dụ ta có đối tượng khác là Pet.java và ta có thể định nghĩa relationship tới đối tượng User.java thông qua @ForeignKey annotation như sau

```
@Entity(foreignKeys = @ForeignKey(entity = User.class, parentColumns = "id",
                childColumns = "user_id"))
class Pet {
   @PrimaryKey
   public int petId;
   public String name;
   @ColumnInfo(name = "user_id")
   public int userId;
}
```

- DAO (Data Access Objects): Đây là component đại diện cho lớp hoặc interface như một đối tượng truy cập dữ liệu (DAO). DAO là thành phần chính của Room là chịu trách nhiệm trong việc định nghĩa các phương thức truy cập CSDL. Các lớp được chú thích với

@Database phải chứa một phương thức trừu tượng có số lượng đối số truyền vào là 0 và đối tượng trả về là đối tượng của lớp được chú thích bởi @Dao. Khi code được sinh ra ở thời điểm biên dịch thì Room sẽ tạo một implementation của class này.

Lưu ý: Bằng cách truy cập database sử dụng lớp DAO thay vì query builder hoặc queries trực tiếp thì ta có thể cô lập các thành phần khác nhau của kiến trúc database.

```
@Dao
public interface UserDAO {
    @Query("SELECT * FROM users WHERE id = :userId")
    User getUserByUserId(int userId);

    @Query("SELECT * FROM users WHERE first_name LIKE
:userName OR last_name LIKE :userName")
    List<User> getUserByName(String userName);

    @Query("SELECT * FROM users")
    List<User> getALlUser();

    @Insert
    void insertUser(User... users);

    @Delete
    void deleteUser(User user);

    @Query("DELETE FROM users")
    void deleteAllUser();

    @Update
    void updateUser(User... users);
}
```

- Database: Có thể dùng componenet này để tạo database holder. Annotation sẽ cung cấp danh sách các thực thể và nội dung class sẽ định nghĩa danh sách các DAO (đối tượng truy cập CSDL) của CSDL. Nó cũng là điểm truy cập chính cho các kết nối phía dưới. Annotated class nên để là lớp abstract extends RoomDatabase. Tại thời điểm runtime thì ta có thể nhận được một instance của nó bằng cách gọi Room.databaseBuilder() hoặc Room.inMemoryDatabaseBuilder().

```
@Database(entities = {User.class}, version = DATABASE_VERSION)
public abstract class UserDatabase extends RoomDatabase {
    private static UserDatabase sUserDatabase;

    public static final int DATABASE_VERSION = 1;
    public static final String DATABASE_NAME = "user_db";

    public abstract UserDAO userDAO();

    public static UserDatabase getInstance(Context context) {
        if (sUserDatabase == null) {
            sUserDatabase = Room.databaseBuilder(context,
UserDatabase.class, DATABASE_NAME)
                    .fallbackToDestructiveMigration()
                    .build();
        }
```

```
            return sUserDatabase;
        }
    }
```

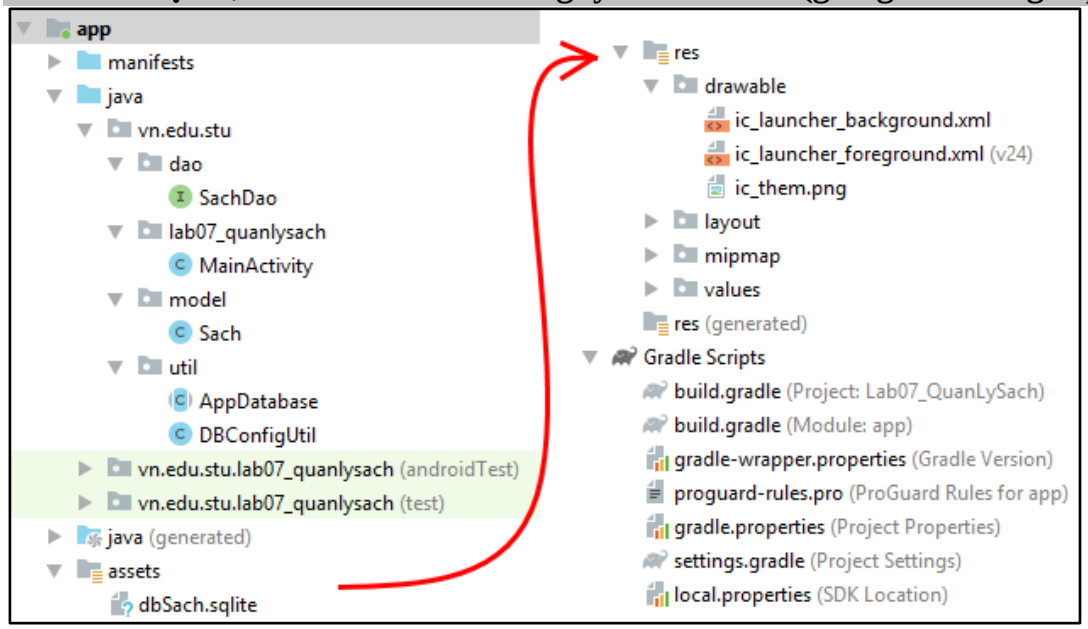
III. NỘI DUNG THỰC HÀNH

1. Quản lý sách

Xây dựng ứng dụng Quản lý sách. Thông tin của sách gồm mã, tên, tác giả và năm xuất bản. Ứng dụng gồm 2 activity:

- MainActivity hiển thị danh sách các sách: khi nhấn lên một item trong danh sách thì mở EditActivity lên, truyền thông tin mã sách qua để chỉnh sửa; khi nhấn giữ 1 item trong danh sách thì xóa cuốn sách đó.
- EditActivity dùng để thêm/chỉnh sửa thông tin sách.

Cấu trúc dự án, database và các tài nguyên drawable (giảng viên cung cấp)



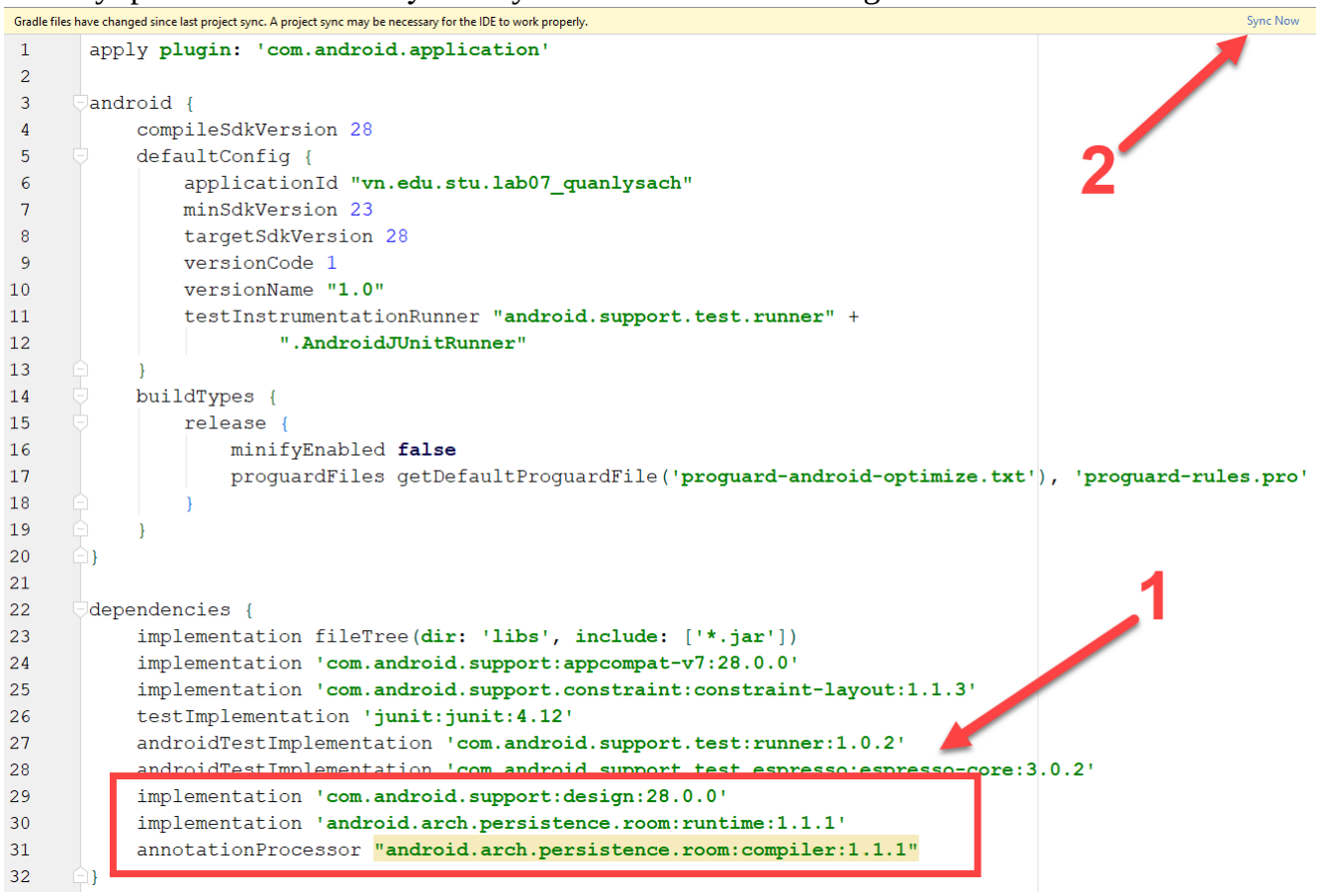
ic_them.png

Gradle Scripts/build.gradle (Module:app)

Thêm 3 dòng sau vào cuối mục dependencies:

```
implementation 'com.android.support:design:28.0.0'
implementation 'android.arch.persistence.room:runtime:1.1.1'
annotationProcessor "android.arch.persistence.room:compiler:1.1.1"
```

* Chú ý: phiên bản có thể thay đổi tùy vào thời điểm viết mã nguồn



model/Sach.java

```java
package vn.edu.stu.model;

import android.arch.persistence.room.ColumnInfo;
import android.arch.persistence.room.Entity;
import android.arch.persistence.room.PrimaryKey;

@Entity(tableName = "Sach_TBL")
public class Sach {
    @PrimaryKey(autoGenerate = true)
    private int ma;

    private String ten;

    private String tacgia;

    @ColumnInfo(name = "namxuatban")
    private int namXuatban;

    public Sach() {
    }

    public int getMa() { return ma; }

    public void setMa(int ma) { this.ma = ma; }

    public String getTen() { return ten; }

    public void setTen(String ten) { this.ten = ten; }

    public String getTacgia() { return tacgia; }

    public void setTacgia(String tacgia) { this.tacgia = tacgia; }

    public int getNamXuatban() { return namXuatban; }

    public void setNamXuatban(int namXuatban) { this.namXuatban = namXuatban; }

    @Override
    public String toString() {
        return "Mã: " + ma + "\n"
                + "Tên: " + ten + "\n"
                + "Tác giả: " + tacgia + "\n"
                + "Năm XB: " + namXuatban;
    }
}
```

dao/SachDao.java

```java
package vn.edu.stu.dao;

import android.arch.persistence.room.Dao;
import android.arch.persistence.room.Delete;
import android.arch.persistence.room.Query;

import java.util.List;

```

```
8
9   import vn.edu.stu.model.Sach;
10
11  @Dao
12  public interface SachDao {
13      @Query("SELECT * FROM Sach_TBL")
14      List<Sach> getAll();
15
16      @Delete
17      int delete(Sach sach);
18  }
```

util/AppDatabase.java

```
1   package vn.edu.stu.util;
2
3   import android.arch.persistence.room.Database;
4   import android.arch.persistence.room.Room;
5   import android.arch.persistence.room.RoomDatabase;
6   import android.content.Context;
7
8   import vn.edu.stu.dao.SachDao;
9   import vn.edu.stu.model.Sach;
10
11  @Database(entities = {Sach.class}, version = 1, exportSchema = false)
12  public abstract class AppDatabase extends RoomDatabase {
13      private static AppDatabase INSTANCE;
14
15      public abstract SachDao sachDao();
16
17      public static AppDatabase getAppDatabase(Context context) {
18          if (INSTANCE == null) {
19              INSTANCE =
20                      Room.databaseBuilder(context.getApplicationContext(),
21                              AppDatabase.class, DBConfigUtil.DATABASE_NAME)
22                              // Dòng allowMainThreadQueries() cho phép truy vấn
23                              // trên MainThread. Nếu dữ liệu nhỏ thì được, nhưng
24                              // nếu dữ liệu lớn có thể làm cho ứng dụng bị treo
25                              // lúc truy vấn. Vì vậy chỉ nên allow trong các
26                              // ứng dụng demo, còn ứng dụng thực tế với
27                              // dữ liệu lớn thì không nên allow, lúc này các
28                              // truy vấn nên thực hiện bằng AsyncTask
29                              .allowMainThreadQueries()
30                              .build();
31          }
32          return INSTANCE;
33      }
34
35      public static void destroyInstance() {
36          INSTANCE = null;
37      }
38  }
```

util/DBConfigUtil.java (giảng viên cung cấp)

```
1   package vn.edu.stu.util;
2
3   import android.content.Context;
4   import android.widget.Toast;
```

```
5
6   import java.io.File;
7   import java.io.FileOutputStream;
8   import java.io.IOException;
9   import java.io.InputStream;
10  import java.io.OutputStream;
11
12  public class DBConfigUtil {
13      final static String DATABASE_NAME = "dbSach.sqlite";
14      final static String DB_PATH_SUFFIX = "/databases/";
15
16  @      public static void copyDatabaseFromAssets(Context context) {
17          File dbFile = context.getDatabasePath(DATABASE_NAME);
18          if (!dbFile.exists()) {
19              // Tạo thư mục chứa CSDL nếu chưa có
20              File dbDir = new File(context.getApplicationInfo().dataDir
21                      + DB_PATH_SUFFIX);
22              if (!dbDir.exists())
23                  dbDir.mkdir();
24
25              InputStream is = null;
26              OutputStream os = null;
27              try {
28                  // Tạo file mới
29                  is = context.getAssets().open(DATABASE_NAME);
30                  String outputFilePath = context.getApplicationInfo().dataDir
31                          + DB_PATH_SUFFIX + DATABASE_NAME;
32                  os = new FileOutputStream(outputFilePath);
33
34                  // Chép nội dung từ file CSDL trong assets vào thư mục CSDL
35                  byte[] buffer = new byte[1024];
36                  int length;
37                  while ((length = is.read(buffer)) > 0) {
38                      os.write(buffer, 0, length);
39                  }
40                  os.flush();
41                  Toast.makeText(
42                          context,
43                          "Đã chép CSDL xong",
44                          Toast.LENGTH_LONG
45                  ).show();
46              } catch (Exception e) {
47                  Toast.makeText(context, e.toString(), Toast.LENGTH_LONG).show();
48              } finally {
49                  try {
50                      os.close();
51                  } catch (IOException e) {
52                  }
53                  try {
54                      is.close();
55                  } catch (IOException e) {
56                  }
57              }
58          }
59      }
60  }
```

activity_main.xml (giảng viên cung cấp)

```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:background="#ffff00"
        android:gravity="center"
        android:text="Quản lý Sách"
        android:textSize="24sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ListView
        android:id="@+id/lvSach"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/textView" />

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fabThem"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="end|bottom"
        android:layout_margin="16dp"
        android:src="@drawable/ic_them"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toRightOf="parent" />
</android.support.constraint.ConstraintLayout>
```

MainActivity.java

```java
package vn.edu.stu.lab07_quanlysach;

import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;

import java.util.ArrayList;

```

```
13  import vn.edu.stu.model.Sach;
14  import vn.edu.stu.util.AppDatabase;
15  import vn.edu.stu.util.DBConfigUtil;
16
17  public class MainActivity extends AppCompatActivity {
18      AppDatabase db;
19      ArrayList<Sach> dsSach;
20      ArrayAdapter<Sach> adapter;
21      ListView lvSach;
22      FloatingActionButton fabThem;
23
24      @Override
25      protected void onCreate(Bundle savedInstanceState) {
26          super.onCreate(savedInstanceState);
27          setContentView(R.layout.activity_main);
28          DBConfigUtil.copyDatabaseFromAssets(MainActivity.this);
29          addControls();
30          hienthiDanhsachSach();
31          addEvents();
32      }
33
34      private void addControls() {
35          db = AppDatabase.getAppDatabase(this);
36          dsSach = new ArrayList<>();
37          adapter = new ArrayAdapter<>(
38                  MainActivity.this,
39                  android.R.layout.simple_list_item_1,
40                  dsSach
41          );
42          lvSach = findViewById(R.id.lvSach);
43          lvSach.setAdapter(adapter);
44          fabThem = findViewById(R.id.fabThem);
45      }
46
47      private void hienthiDanhsachSach() {
48          // Xóa danh sách cũ, phòng trường hợp gọi hàm nhiều lần
49          dsSach.clear();
50          dsSach.addAll(db.sachDao().getAll());
51          adapter.notifyDataSetChanged();
52      }
53
54      private void addEvents() {
55          lvSach.setOnItemLongClickListener(
56                  new AdapterView.OnItemLongClickListener() {
57                      @Override
58                      public boolean onItemLongClick(AdapterView<?> parent,
59                                                     View view, int position,
60                                                     long id) {
61                          if (position >= 0 && position < dsSach.size()) {
62                              Sach sach = dsSach.get(position);
63                              int deletedRowCount = db.sachDao().delete(sach);
64                              if (deletedRowCount > 0) {
65                                  dsSach.remove(position);
66                                  adapter.notifyDataSetChanged();
67                              }
68                          }
```

```
69                          return true;
70                      }
71                  });
72          }
73      }
```

Yêu cầu sinh viên:

- Tiến hành cài đặt lại và chạy thử
- Viết code cho chức năng thêm và sửa thông tin sách

Hướng dẫn:

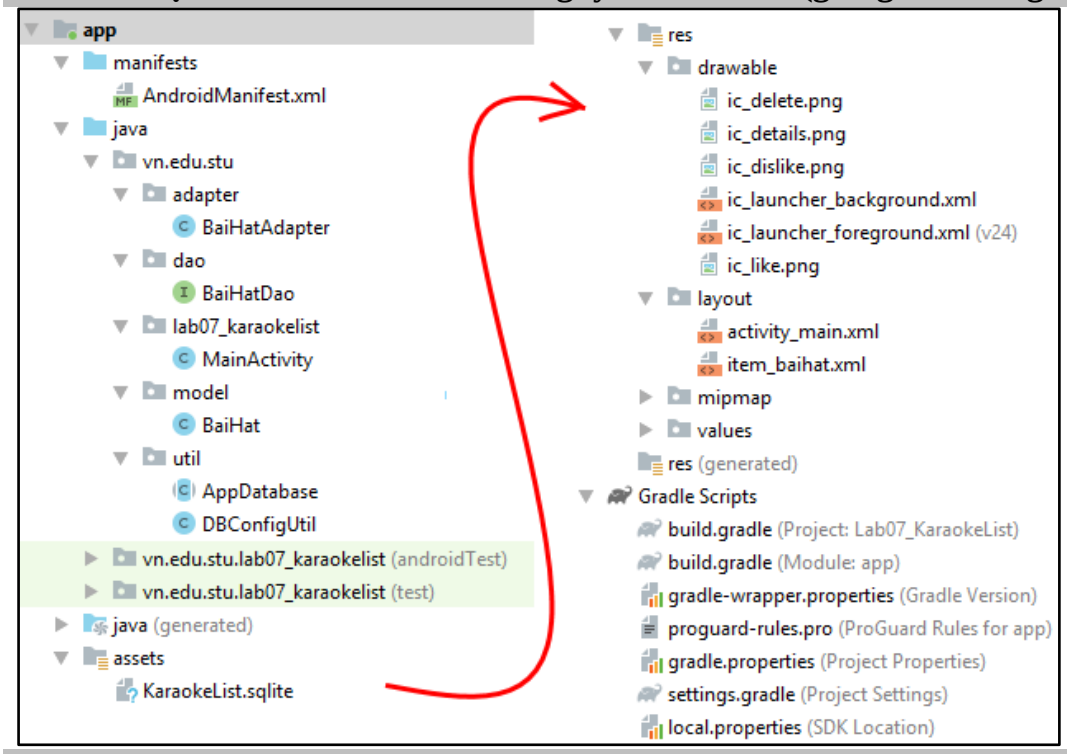
+ Tạo EditActivity, thiết kế các View cần thiết cho việc thêm/sửa sách
+ Xử lý FloatingActionButton của MainActivity để gọi sang EditActivity để thêm mới
+ Xử lý sự kiện nhấn 1 item trên danh sách Sách, truyền mã của sách được nhấn sang EditActivity để chỉnh sửa

2. KaraokeList

Xây dựng ứng dụng tra cứu mã số bài hát karaoke. Thông tin của bài hát gồm mã, tên, lời, tác giả, thể loại và yêu thích. Ứng dụng gồm 2 activity:

- MainActivity chứa ListView hiển thị danh sách các bài hát và 1 EditText để tìm kiếm thông tin bài hát. Các dòng hiển thị thông tin bài hát được tùy biến như hình dưới: có chứa các ImageButton để Thích/Bỏ thích, Xem thông tin bài hát, Xóa bài hát. Khi nhấn Thích/Bỏ thích thì ứng dụng sẽ cập nhật lại cột Yêu thích trong CSDL. Khi nhấn Chi tiết thì ứng dụng mở một BaiHatActivity lên để hiển thị thông tin chi tiết của bài hát. Khi nhấn Xóa thì ứng dụng sẽ xóa bài hát đó trong CSDL.
- BaiHatActivity dùng để hiển thị thông tin bài hát (sinh viên tự làm ở nhà).

Cấu trúc dự án, database và các tài nguyên drawable (giảng viên cung cấp)



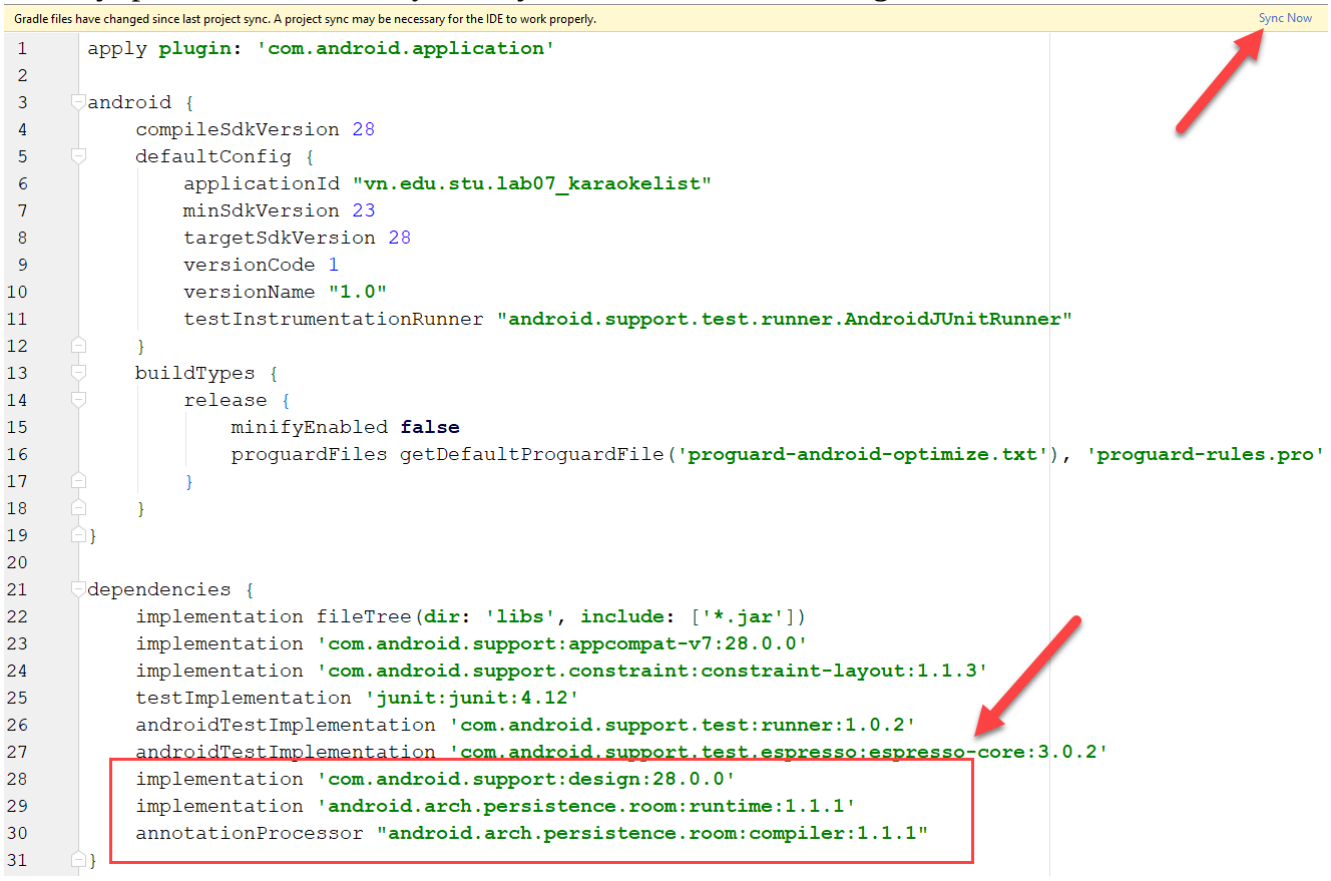
ic_details.png
ic_delete.png
ic_like.png
ic_dislike.png

Gradle Scripts/build.gradle (Module:app)

Thêm 3 dòng sau vào cuối mục dependencies:

```
implementation 'com.android.support:design:28.0.0'
implementation 'android.arch.persistence.room:runtime:1.1.1'
annotationProcessor "android.arch.persistence.room:compiler:1.1.1"
```

* Chú ý: phiên bản có thể thay đổi tùy vào thời điểm viết mã nguồn



model/BaiHat.java

```
1   package vn.edu.stu.model;
2
3   import android.arch.persistence.room.ColumnInfo;
4   import android.arch.persistence.room.Entity;
5   import android.arch.persistence.room.PrimaryKey;
6   import android.support.annotation.NonNull;
7
8   @Entity(tableName = "BaiHat_TBL")
9   public class BaiHat {
10      @PrimaryKey
11      @NonNull
12      @ColumnInfo(name = "MABH")
13      private String maBH;
14
15      @ColumnInfo(name = "TENBH")
16      private String tenBH;
17
18      @ColumnInfo(name = "LOIBH")
19      private String loiBH;
20
21      @ColumnInfo(name = "TACGIA")
22      private String tacgia;
23
24      @ColumnInfo(name = "THELOAI")
25      private String theloai;
26
27      @ColumnInfo(name = "YEUTHICH")
28      private boolean yeuThich;
29
30  @   public BaiHat() {
31      }
32
33      public String getMaBH() { return maBH; }
36
37      public void setMaBH(String maBH) { this.maBH = maBH; }
40
41      public String getTenBH() { return tenBH; }
44
45      public void setTenBH(String tenBH) { this.tenBH = tenBH; }
48
49      public String getLoiBH() { return loiBH; }
52
53      public void setLoiBH(String loiBH) { this.loiBH = loiBH; }
56
57      public String getTacgia() { return tacgia; }
60
61      public void setTacgia(String tacgia) { this.tacgia = tacgia; }
64
65      public String getTheloai() { return theloai; }
68
69      public void setTheloai(String theloai) { this.theloai = theloai; }
72
73      public boolean isYeuThich() { return yeuThich; }
76
77      public void setYeuThich(boolean yeuThich) { this.yeuThich = yeuThich; }
80  }
```

dao/BaiHatDao.java

```java
package vn.edu.stu.dao;

import android.arch.persistence.room.Dao;
import android.arch.persistence.room.Query;
import android.arch.persistence.room.Update;

import java.util.List;

import vn.edu.stu.model.BaiHat;

@Dao
public interface BaiHatDao {
    @Query("SELECT * FROM BaiHat_TBL")
    List<BaiHat> getAll();

    @Update
    int updateBaiHat(BaiHat baiHat);

    @Query("SELECT * FROM BaiHat_TBL where MABH LIKE :dieukien" +
            " OR TENBH LIKE :dieukien OR LOIBH LIKE :dieukien" +
            " OR TACGIA LIKE :dieukien OR THELOAI LIKE :dieukien")
    List<BaiHat> findBaiHat(String dieukien);
}
```

util/AppDatabase.java

```java
package vn.edu.stu.util;

import android.arch.persistence.room.Database;
import android.arch.persistence.room.Room;
import android.arch.persistence.room.RoomDatabase;
import android.content.Context;

import vn.edu.stu.dao.BaiHatDao;
import vn.edu.stu.model.BaiHat;

@Database(entities = {BaiHat.class}, version = 1, exportSchema = false)
public abstract class AppDatabase extends RoomDatabase {
    private static AppDatabase INSTANCE;

    public abstract BaiHatDao baiHatDao();

    public static AppDatabase getAppDatabase(Context context) {
        if (INSTANCE == null) {
            INSTANCE =
                    Room.databaseBuilder(context.getApplicationContext(),
                            AppDatabase.class, DBConfigUtil.DATABASE_NAME)
                            // Dòng allowMainThreadQueries() cho phép truy vấn
                            // trên MainThread. Nếu dữ liệu nhỏ thì được, nhưng
                            // nếu dữ liệu lớn có thể làm cho ứng dụng bị treo
                            // lúc truy vấn. Vì vậy chỉ nên allow trong các
                            // ứng dụng demo, còn ứng dụng thực tế với
                            // dữ liệu lớn thì không nên allow, lúc này các
                            // truy vấn nên thực hiện bằng AsyncTask
                            .allowMainThreadQueries()
                            .build();
```

```
        }
        return INSTANCE;
    }

    public static void destroyInstance() {
        INSTANCE = null;
    }
}
```

util/DBConfigUtil.java (giảng viên cung cấp)

```
package vn.edu.stu.util;

import android.content.Context;
import android.widget.Toast;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

public class DBConfigUtil {
    final static String DATABASE_NAME = "KaraokeList.sqlite";
    final static String DB_PATH_SUFFIX = "/databases/";

    public static void copyDatabaseFromAssets(Context context) {
        File dbFile = context.getDatabasePath(DATABASE_NAME);
        if (!dbFile.exists()) {
            // Tạo thư mục chứa CSDL nếu chưa có
            File dbDir = new File(context.getApplicationInfo().dataDir
                    + DB_PATH_SUFFIX);
            if (!dbDir.exists())
                dbDir.mkdir();

            InputStream is = null;
            OutputStream os = null;
            try {
                // Tạo file mới
                is = context.getAssets().open(DATABASE_NAME);
                String outputFilePath = context.getApplicationInfo().dataDir
                        + DB_PATH_SUFFIX + DATABASE_NAME;
                os = new FileOutputStream(outputFilePath);

                // Chép nội dung từ file CSDL trong assets vào thư mục CSDL
                byte[] buffer = new byte[1024];
                int length;
                while ((length = is.read(buffer)) > 0) {
                    os.write(buffer, 0, length);
                }
                os.flush();
                Toast.makeText(
                        context,
                        "Đã chép CSDL xong",
                        Toast.LENGTH_LONG
                ).show();
            } catch (Exception e) {
```

```
                Toast.makeText(context, e.toString(), Toast.LENGTH_LONG).show();
            } finally {
                try {
                    os.close();
                } catch (IOException e) {
                }
                try {
                    is.close();
                } catch (IOException e) {
                }
            }
        }
    }
}
```

item_baihat.xml (giảng viên cung cấp)

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/txtMaBH"
        android:layout_width="110dp"
        android:layout_height="0dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:background="#00FF00"
        android:gravity="center"
        android:text="66666"
        android:textColor="#FF0000"
        android:textSize="35sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="@+id/btnThich"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/txtTenBH"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginEnd="8dp"
        android:text="Tên bài hát"
        android:textColor="#0d34ab"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/txtMaBH"
        app:layout_constraintTop_toTopOf="@+id/txtMaBH" />

    <TextView
        android:id="@+id/txtTacGia"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
```

```
42          android:text="Tác giả"
43          android:textColor="#6c0bcd"
44          android:textStyle="italic"
45          app:layout_constraintEnd_toEndOf="parent"
46          app:layout_constraintStart_toStartOf="@+id/txtTenBH"
47          app:layout_constraintTop_toBottomOf="@+id/txtTenBH" />
48
49      <ImageButton
50          android:id="@+id/btnThich"
51          android:layout_width="wrap_content"
52          android:layout_height="wrap_content"
53          app:layout_constraintEnd_toStartOf="@+id/btnBoThich"
54          app:layout_constraintTop_toBottomOf="@+id/txtTacGia"
55          app:srcCompat="@drawable/ic_like" />
56
57      <ImageButton
58          android:id="@+id/btnBoThich"
59          android:layout_width="wrap_content"
60          android:layout_height="wrap_content"
61          app:layout_constraintEnd_toStartOf="@+id/btnChitiet"
62          app:layout_constraintTop_toBottomOf="@+id/txtTacGia"
63          app:srcCompat="@drawable/ic_dislike" />
64
65      <ImageButton
66          android:id="@+id/btnChitiet"
67          android:layout_width="wrap_content"
68          android:layout_height="wrap_content"
69          app:layout_constraintEnd_toStartOf="@+id/btnXoa"
70          app:layout_constraintTop_toBottomOf="@+id/txtTacGia"
71          app:srcCompat="@drawable/ic_details" />
72
73      <ImageButton
74          android:id="@+id/btnXoa"
75          android:layout_width="wrap_content"
76          android:layout_height="wrap_content"
77          android:layout_marginEnd="8dp"
78          app:layout_constraintEnd_toEndOf="parent"
79          app:layout_constraintTop_toBottomOf="@+id/txtTacGia"
80          app:srcCompat="@drawable/ic_delete" />
81
82      <TextView
83          android:id="@+id/textView5"
84          android:layout_width="0dp"
85          android:layout_height="1dp"
86          android:layout_marginStart="8dp"
87          android:layout_marginTop="4dp"
88          android:layout_marginEnd="8dp"
89          android:background="#ab0ffe03"
90          android:text="TextView"
91          app:layout_constraintEnd_toEndOf="parent"
92          app:layout_constraintStart_toStartOf="parent"
93          app:layout_constraintTop_toBottomOf="@+id/btnXoa" />
94  </android.support.constraint.ConstraintLayout>
```

adapter/BaiHatAdapter.java

```java
package vn.edu.stu.adapter;

import android.app.Activity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageButton;
import android.widget.TextView;
import android.widget.Toast;

import java.util.List;

import vn.edu.stu.lab07_karaokelist.R;
import vn.edu.stu.model.BaiHat;
import vn.edu.stu.util.AppDatabase;

public class BaiHatAdapter extends ArrayAdapter<BaiHat> {
    Activity context;
    int resource;
    List<BaiHat> objects;

    public BaiHatAdapter(Activity context, int resource, List<BaiHat> objects) {
        super(context, resource, objects);
        this.context = context;
        this.resource = resource;
        this.objects = objects;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        LayoutInflater inflater = this.context.getLayoutInflater();
        View item = inflater.inflate(this.resource, null);

        TextView txtMaBH = item.findViewById(R.id.txtMaBH);
        TextView txtTenBH = item.findViewById(R.id.txtTenBH);
        TextView txtTacgia = item.findViewById(R.id.txtTacGia);
        final ImageButton btnThich = item.findViewById(R.id.btnThich);
        final ImageButton btnBoThich = item.findViewById(R.id.btnBoThich);
        final ImageButton btnChitiet = item.findViewById(R.id.btnChitiet);
        final ImageButton btnXoa = item.findViewById(R.id.btnXoa);

        final BaiHat baiHat = objects.get(position);
        txtMaBH.setText(baiHat.getMaBH());
        txtTenBH.setText(baiHat.getTenBH());
        txtTacgia.setText(baiHat.getTacgia());

        if (baiHat.isYeuThich()) {
            btnThich.setVisibility(View.INVISIBLE);
            btnBoThich.setVisibility(View.VISIBLE);
        } else {
            btnThich.setVisibility(View.VISIBLE);
            btnBoThich.setVisibility(View.INVISIBLE);
        }

```

```
56          btnThich.setOnClickListener(new View.OnClickListener() {
57              @Override
58              public void onClick(View v) {
59                  xuliThich(baiHat, btnThich, btnBoThich);
60              }
61          });
62
63          btnBoThich.setOnClickListener(new View.OnClickListener() {
64              @Override
65              public void onClick(View v) {
66                  xuliBoThich(baiHat, btnThich, btnBoThich);
67              }
68          });
69
70          btnChitiet.setOnClickListener(new View.OnClickListener() {
71              @Override
72              public void onClick(View v) {
73                  xuliChitiet(baiHat);
74              }
75          });
76
77          btnXoa.setOnClickListener(new View.OnClickListener() {
78              @Override
79              public void onClick(View v) {
80                  xuliXoa(baiHat);
81              }
82          });
83          return item;
84      }
85
86  @      private void xuliThich(BaiHat baiHat, ImageButton btnThich,
87                                ImageButton btnBoThich) {
88          AppDatabase db = AppDatabase.getAppDatabase(this.context);
89          baiHat.setYeuThich(true);
90          int ret = db.baiHatDao().updateBaiHat(baiHat);
91          if (ret > 0) {
92              baiHat.setYeuThich(true);
93              Toast.makeText(
94                      context,
95                      "Gán yêu thích thành công",
96                      Toast.LENGTH_SHORT
97              ).show();
98          } else {
99              baiHat.setYeuThich(false);
100             Toast.makeText(
101                     context,
102                     "Gán yêu thích thất bại",
103                     Toast.LENGTH_SHORT
104             ).show();
105         }
106         btnThich.setVisibility(View.INVISIBLE);
107         btnBoThich.setVisibility(View.VISIBLE);
108     }
109
110 @      private void xuliBoThich(BaiHat baiHat, ImageButton btnThich,
111                                 ImageButton btnBoThich) {
112         AppDatabase db = AppDatabase.getAppDatabase(this.context);
```

```
113             baiHat.setYeuThich(false);
114             int ret = db.baiHatDao().updateBaiHat(baiHat);
115             if (ret > 0) {
116                 Toast.makeText(
117                         context,
118                         "Bỏ yêu thích thành công",
119                         Toast.LENGTH_SHORT
120                 ).show();
121             } else {
122                 baiHat.setYeuThich(true);
123                 Toast.makeText(
124                         context,
125                         "Bỏ yêu thích thất bại",
126                         Toast.LENGTH_SHORT
127                 ).show();
128             }
129             btnThich.setVisibility(View.VISIBLE);
130             btnBoThich.setVisibility(View.INVISIBLE);
131         }
132
133         private void xuliChitiet(BaiHat baiHat) {
134             //Sinh viên tự viết
135         }
136
137         private void xuliXoa(BaiHat baiHat) {
138             //Sinh viên tự viết
139         }
140     }
```

activity_main.xml (giảng viên cung cấp)

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <android.support.constraint.ConstraintLayout
3       xmlns:android="http://schemas.android.com/apk/res/android"
4       xmlns:app="http://schemas.android.com/apk/res-auto"
5       xmlns:tools="http://schemas.android.com/tools"
6       android:layout_width="match_parent"
7       android:layout_height="match_parent"
8       tools:context=".MainActivity">
9
10      <EditText
11          android:id="@+id/txtTimKiem"
12          android:layout_width="0dp"
13          android:layout_height="wrap_content"
14          android:hint="Tìm ở đây"
15          app:layout_constraintLeft_toLeftOf="parent"
16          app:layout_constraintRight_toRightOf="parent"
17          app:layout_constraintTop_toTopOf="parent" />
18
19      <ListView
20          android:id="@+id/lvBaiHat"
21          android:layout_width="match_parent"
22          android:layout_height="0dp"
23          app:layout_constraintBottom_toBottomOf="parent"
24          app:layout_constraintTop_toBottomOf="@+id/txtTimKiem" />
25  </android.support.constraint.ConstraintLayout>
```

MainActivity.java

```java
package vn.edu.stu.lab07_karaokelist;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.text.Editable;
import android.text.TextWatcher;
import android.widget.EditText;
import android.widget.ListView;

import java.util.ArrayList;

import vn.edu.stu.adapter.BaiHatAdapter;
import vn.edu.stu.model.BaiHat;
import vn.edu.stu.util.AppDatabase;
import vn.edu.stu.util.DBConfigUtil;

public class MainActivity extends AppCompatActivity {
    AppDatabase db;
    EditText txtTimKiem;
    ListView lvBaiHat;
    ArrayList<BaiHat> dsBaiHat;
    BaiHatAdapter adapter;
    TextWatcher textWatcher;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        DBConfigUtil.copyDatabaseFromAssets(MainActivity.this);
        addControls();
        hienthiDanhsachBaiHat();
    }

    private void addControls() {
        db = AppDatabase.getAppDatabase(this);
        txtTimKiem = findViewById(R.id.txtTimKiem);
        textWatcher = new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count,
                                          int after) {
            }

            @Override
            public void onTextChanged(CharSequence s, int start, int before,
                                      int count) {
                //Sinh viên tự viết code xử lý tìm kiếm
            }

            @Override
            public void afterTextChanged(Editable s) {
            }
        };
        txtTimKiem.addTextChangedListener(textWatcher);

        lvBaiHat = findViewById(R.id.lvBaiHat);
```

```
56          dsBaiHat = new ArrayList<>();
57          adapter = new BaiHatAdapter(
58                  MainActivity.this,
59                  R.layout.item_baihat,
60                  dsBaiHat
61          );
62          lvBaiHat.setAdapter(adapter);
63      }
64
65      private void hienthiDanhsachBaiHat() {
66          // Xóa danh sách cũ, phòng trường hợp gọi hàm nhiều lần
67          dsBaiHat.clear();
68          // Gọi tới BaiHatDao để lấy dữ liệu từ Database lên
69          dsBaiHat.addAll(db.baiHatDao().getAll());
70          adapter.notifyDataSetChanged();
71      }
72  }
```

Yêu cầu sinh viên:

- Tiến hành cài đặt lại và chạy thử
- Viết code cho chức năng xóa bài hát
- Viết code cho chức năng tìm kiếm bài hát: Tìm theo cả mã, tên, lời, tác giả và thể loại; gõ chữ tới đâu tìm kiếm tới đó, nếu chuỗi rỗng thì liệt kê toàn bộ bài hát. **Gợi ý:** chỉnh sửa hàm ***hienthiDanhsachBaiHat*** để nhận thêm chuỗi cần tìm, chỉnh code truy vấn để tìm theo điều kiện nhận được.

IV. BÀI TẬP LÀM THÊM

1. Hoàn thiện KaraokeList

- Chỉnh lại mã nguồn cho nút xóa: xác nhận có muốn xóa hay không trước khi xóa trong CSDL
- Cài đặt mã nguồn cho nút "Chi tiết": mở một Activity để xem và sửa thông tin bài hát (không cho sửa mã bài hát)

2. Quản lý nhân viên

Xây dựng ứng dụng quản lý toàn bộ nhân viên của công ty: liệt kê, thêm, sửa, xóa nhân viên. Thông tin nhân viên gồm: Mã nhân viên, Tên, Ngày sinh, Địa chỉ, Số điện thoại, Mã phòng ban. Thông tin phòng ban gồm: Mã phòng ban, Tên phòng ban. Ứng dụng gồm 4 Activity: MainActivity liệt kê toàn bộ danh sách nhân viên, có thêm Button để chuyển sang Activity hiển thị phòng ban; UpdateNhanVienActivity để nhập hoặc chỉnh sửa thông tin nhân viên; PhongBanActivity để hiển thị danh sách các phòng ban, có thêm Button để chuyển sang MainActivity; UpdatePhongBanActivity để nhập hoặc chỉnh sửa phòng ban. Phòng ban của nhân viên được nhập bằng Spinner. Thông tin nhân viên và phòng ban được lưu trữ trong CSDL SQLite do sinh viên tự tạo.