

Lập trình trên thiết bị di động

# SQLITE & SHARED PREFERENCES

---

GV: Nguyễn Huy Cường

Email: [nh.cuong@hutech.edu.vn](mailto:nh.cuong@hutech.edu.vn)

# Nội dung

## 1. SQLITE

- 📖 Giới thiệu về SQLite
- 📖 Các kiểu dữ liệu trong SQLite

## 2. Sử dụng SQLite

- 📖 SQLiteHelper: Tạo Database và Table
- 📖 SQLiteDatabase: Xử lý CRUD
- 📖 Tổ chức cấu trúc ứng dụng

## 3. Shared Preferences



# SQLITE

---

# Tổng quan

- SQLite là phần mềm quản lý CSDL nhưng không có máy chủ riêng biệt để xử lý.
- Đặc điểm:
  - ❑ Đơn giản & gọn nhẹ: Dữ liệu database được lưu vào một file duy nhất, không cần cài đặt, không cần cấu hình.
  - ❑ Không có user, password hay quyền hạn trong Sqlite
  - ❑ Dễ Sử Dụng: cung cấp một giao diện lập trình ổn định và dễ sử dụng thông qua các lớp như **SQLiteOpenHelper**, giúp quản lý việc tạo và nâng cấp cơ sở dữ liệu



## Tại sao nên dùng SQLite ?

- SQLite không cần thiết phải cài đặt, cấu hình, không cần mô hình client-server để hoạt động.
- Một SQLite Database đầy đủ được lưu giữ trong một disk file đơn. SQLite là rất nhỏ gọn, nhỏ hơn 400kB đã được cấu hình đầy đủ hoặc nhỏ hơn 250kB khi đã bỏ qua các tính năng tùy ý.
- Các Transaction trong SQLite là tuân theo đầy đủ chuẩn ACID, đảm bảo truy cập an toàn từ nhiều tiến trình hoặc thread.
- SQLite hỗ trợ hầu hết các tính năng của một ngôn ngữ truy vấn trong chuẩn SQL92.

# Kiểu dữ liệu SQLite

## ❖ **Lớp lưu trữ** (Storage Class)

Mỗi giá trị được lưu giữ trong SQLite có một trong các lớp lưu trữ (Storage Class) sau:

Lớp lưu trữ	Miêu tả
NULL	Giá trị là một giá trị NULL
INTEGER	Giá trị là một số nguyên có dấu, được lưu giữ trong 1, 2, 3, 4, 6, hoặc 8 byte tùy thuộc vào độ lớn của giá trị
REAL	Giá trị số thực dấu chấm động, được lưu giữ như là một số thực dấu chấm động 8-byte IEEE
TEXT	Giá trị là một text string, được lưu trữ bởi sử dụng Encoding của cơ sở dữ liệu (UTF-8, UTF-16BE hoặc UTF-16LE)
BLOB	Giá trị là một blob của dữ liệu, nhập vào như thế nào thì lưu giữ chính xác như thế

# Kiểu dữ liệu SQLite

## ❖ Kiểu lưu trữ (Affinity Type)

Affinity Type trên các cột. Bất cứ cột nào có thể vẫn lưu giữ bất kỳ kiểu dữ liệu nào nhưng lớp lưu trữ ưu tiên cho một **cột** được gọi là **Affinity** của nó.

Affinity	Miêu tả
TEXT	Cột này lưu giữ tất cả dữ liệu sử dụng các lớp lưu trữ NULL, TEXT hoặc BLOB
NUMERIC	Cột này có thể chứa các giá trị sử dụng tất cả 5 lớp lưu trữ
INTEGER	Vận hành giống như một cột với NUMERIC affinity với một ngoại lệ trong một biểu thức CAST
REAL	Vận hành giống như một cột với NUMERIC affinity, ngoại trừ rằng nó ép các giá trị nguyên thành dạng biểu diễn số thực dấu chấm động
NONE	Một cột với NONE affinity không ưu tiên một lớp lưu trữ nào khi so với lớp khác và không ép dữ liệu từ một lớp lưu trữ này sang dạng một lớp lưu trữ khác

# Kiểu dữ liệu SQLite

## ❖ Kiểu dữ liệu

Kiểu dữ liệu	Affinity
INT INTEGER TINYINT SMALLINT MEDIUMINT BIGINT UNSIGNED BIG INT INT2 , INT8	INTEGER
CHARACTER(20) VARCHAR(255) VARYING CHARACTER(255) NCHAR(55) NATIVE CHARACTER(70) NVARCHAR(100) TEXT	TEXT

Kiểu dữ liệu	Affinity
BLOB •Không có kiểu dữ liệu nào được xác định	NONE
REAL DOUBLE DOUBLE PRECISION FLOAT	REAL
NUMERIC DECIMAL(10,5) BOOLEAN DATE DATETIME	NUMERIC



# Kiểu dữ liệu SQLite

## ✓ Kiểu dữ liệu Boolean trong SQLite

- ❑ SQLite không hỗ trợ lớp lưu trữ Boolean.
- ❑ Giá trị Boolean (false/ true) được lưu trữ dưới dạng các số nguyên: 0/1.

## ✓ Kiểu dữ liệu Date và Time trong SQLite

- ❑ SQLite không hỗ trợ lớp lưu trữ date/time.
- ❑ Giá trị Date/ Time được lưu trữ dưới dạng giá trị TEXT, REAL hoặc INTEGER.

# Các hàm quan trọng trong SQLite

Method	Description
<code>getColumnNames()</code>	This method is used to get the Array of column names of our SQLite table.
<code>getCount()</code>	This method will return the number of rows in the cursor.
<code>isClosed()</code>	This method returns a Boolean value when our cursor is closed.
<code>getColumnCount()</code>	This method returns the total number of columns present in our table.
<code>getColumnName(int columnIndex)</code>	This method will return the name of the column when we passed the index of our column in it.
<code>getColumnIndex(String columnName)</code>	This method will return the index of our column from the name of the column.
<code>getPosition()</code>	This method will return the current position of our cursor in our table.

# SỬ DỤNG SQLITE

---

SQLiteOpenHelper  
SQLiteDatabase

# CLASS hỗ trợ của SQLite

- Android cung cấp 2 Class hỗ trợ cho việc tạo và quản lý Database:

## 1. SQLiteOpenHelper

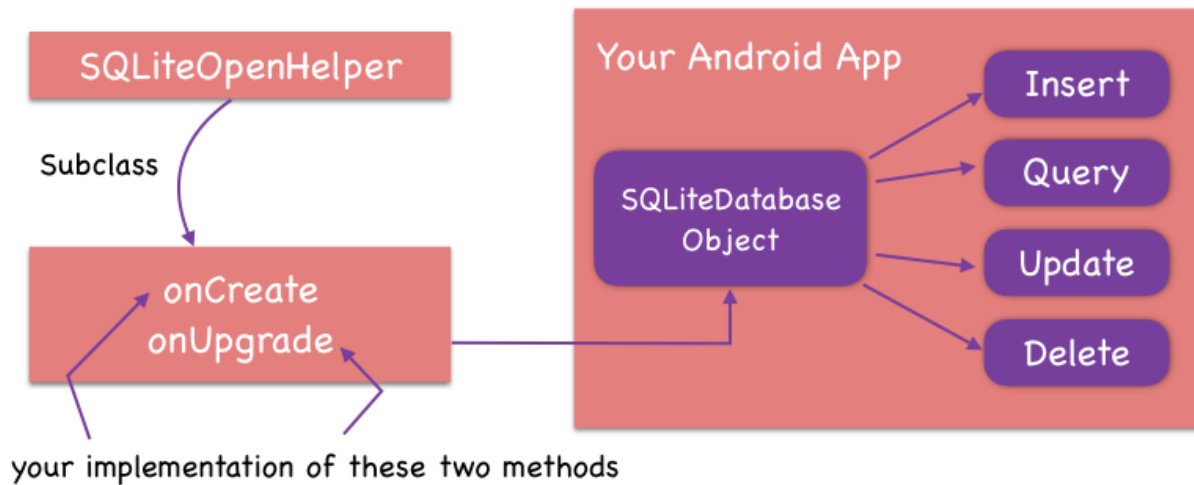
- ❑ Là lớp **abstract** chịu trách nhiệm tạo/nâng cấp CSDL SQLite và trả về phiên bản đối tượng SQLiteDatabase.
- ❑ Phải tạo lớp kế thừa của lớp **abstract** này.

## 2. SQLiteDatabase

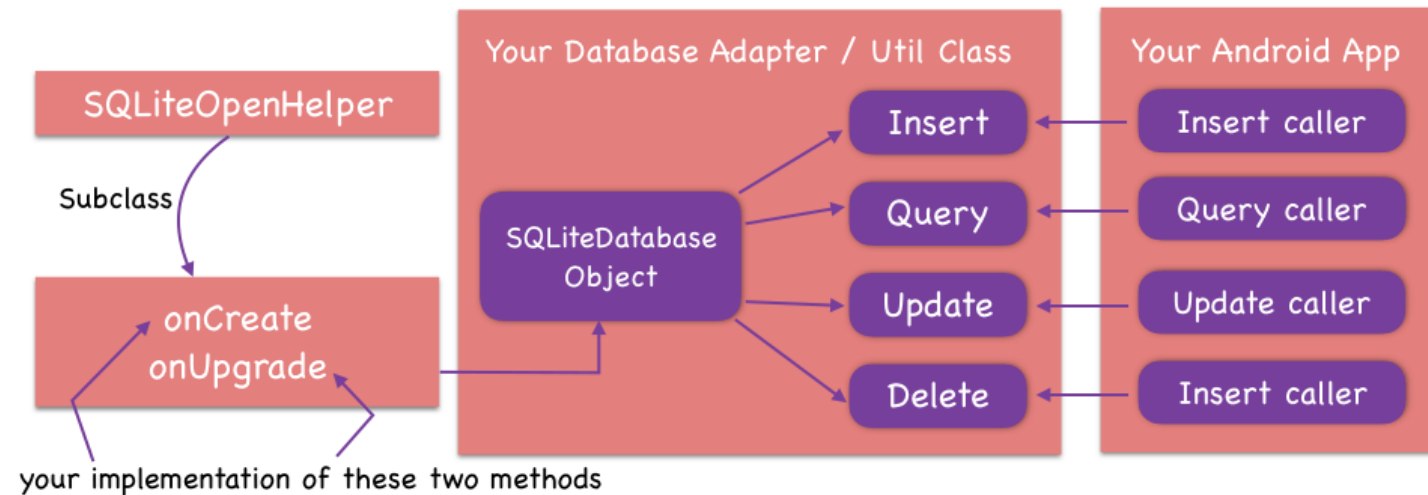
- ❑ Là một lớp có API để xử lý các thao tác CSDL như: Create, Read, Update, Delete (CRUD)

# Kiến trúc Android với SQLite

## Cách 1: Gọi trực tiếp API từ ứng dụng



## Cách 2: Tạo một Adapter/ Util (DAO..) để đóng gọi các phương thức



# Sử dụng SQLite: (SQLiteOpenHelper & SQLiteDatabase)

Các bước thực hiện:

**Bước 0:** Thêm quyền truy cập cho ứng dụng

<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />

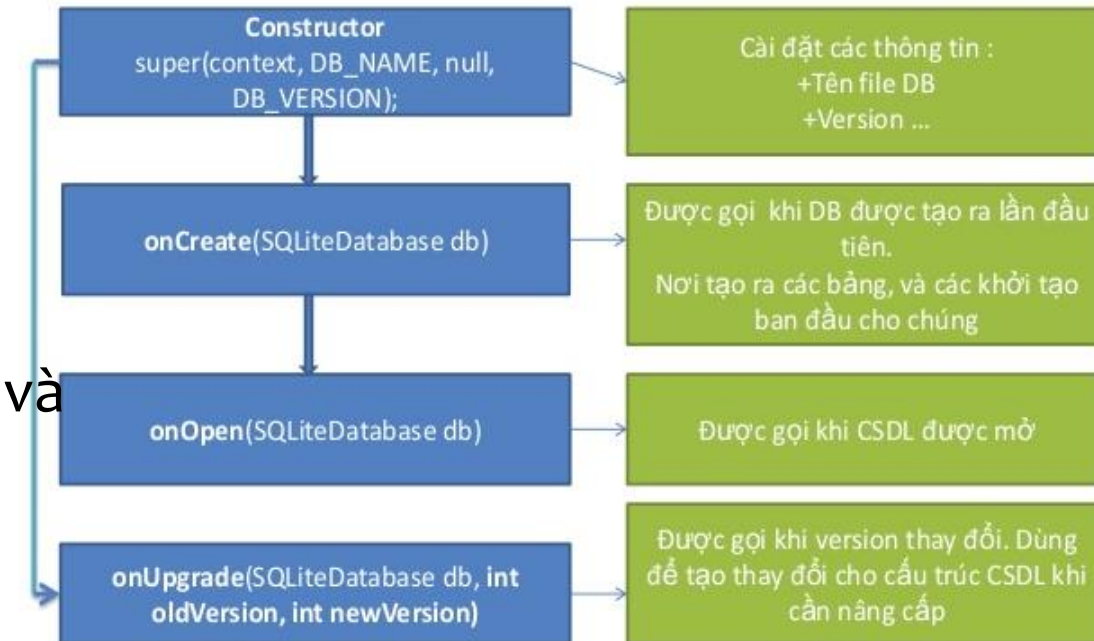
**Bước 1:** Tạo Class kế thừa **SQLiteOpenHelper** và override 2 phương thức **onCreate** và **onUpgrade**

❑ **onCreate:** Được gọi khi CSDL chưa tồn tại

Nếu CSDL không tồn tại: Được gọi tự động.

❑ **onUpgrade:** Thực hiện khi có phiên bản CSDL mới.

Khi ta nâng version mới, cần xoá các table cũ và gọi lại onCreate



# Sử dụng SQLite: SQLiteOpenHelper (B1)

VD: Tạo Database “QLSP”

Với bảng “Product”

Giải thích code:

- ✓ Hàm khởi tạo **SQLiteHelper**:  
Tạo database “QLSP” với version 1
- ✓ Hàm **onCreate**:  
Thực hiện viết code tạo table “Product”
- ✓ Hàm **onUpgrade**:  
Thực hiện xóa và gọi lại **onCreate** nếu có version mới.

```
public class DBHelper extends SQLiteOpenHelper {
    public DBHelper(Context context) {
        super(context, "QLSP", null, 1);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        String query = String.format("CREATE TABLE %s (" +
            "%s INTEGER PRIMARY KEY AUTOINCREMENT, " +
            "%s TEXT, " +
            "%s REAL, " +
            "%s REAL)",
            "Product", "id", "name", "price", "image");
        db.execSQL(query);
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        if(oldVersion != newVersion)
        {
            //drop
            String query = "DROP TABLE Product";
            db.execSQL(query);
            //create again
            onCreate(db);
        }
    }
}
```

# Sử dụng SQLite: SQLiteOpenHelper (B2)

## Bước 2: Viết các hàm CRUD thực hiện

### (Sử dụng SQLiteDatabase)

❑ Đối với Insert, update, delete: sử dụng các câu query và gọi **execSQL(query)**;

❑ Đối với hàm lấy dữ liệu: Thông qua **Cursor**

### VD: Viết CRUD cho “Product”

```
public class ProductDAO {
    DBHelper dbHelper;
    public ProductDAO(Context context) {
        dbHelper = new DBHelper(context);
    }
    public List<Product> GetAll()
    {
        SQLiteDatabase db = dbHelper.getWritableDatabase();
        List<Product> listProduct = new ArrayList<>();
        String query = "SELECT * FROM product";
        Cursor c = db.rawQuery(query, null);
        while (c.moveToNext())
        {
            Product temp = new Product();
            temp.setId(c.getInt(0));
            temp.setName(c.getString(1));
            temp.setPrice(c.getFloat(2));
            temp.setImage(c.getString(3));
            listProduct.add(temp);
        }
        return listProduct;
    }
    public void Insert(Product p) {
        SQLiteDatabase db = dbHelper.getWritableDatabase();
        //c1: sử dụng contentValues
        ContentValues values = new ContentValues();
        // values.put("id", p.getId());
        values.put("name", p.getName());
        values.put("price", p.getPrice());
        values.put("image", p.getImage());
        db.insert("product", null, values);
        //c2: sử dụng câu query thuần
    }
    public void Delete(int productId) {
        SQLiteDatabase db = dbHelper.getWritableDatabase();
        //c1: sử dụng delete
        db.delete("product", "id=?", new String[] { String.valueOf(productId) });
    }
}
```

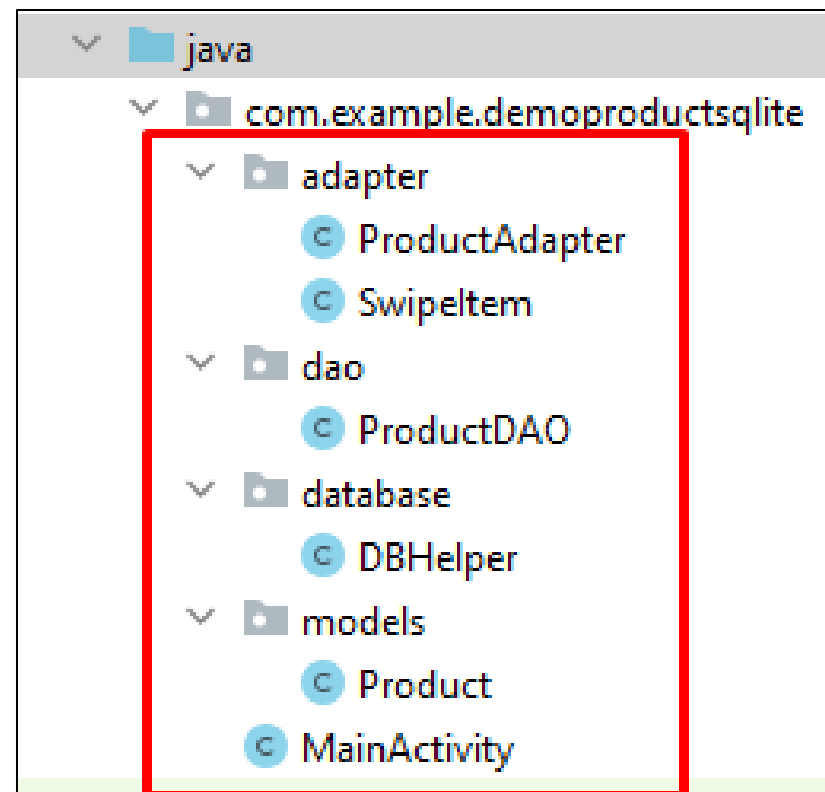


## Tổ chức cấu trúc ứng dụng

Để dễ viết code và bảo trì chúng ta cần tổ chức cấu trúc ứng dụng theo nhiều **package**

### Ví dụ cấu trúc 1:

- Adapter:
- Model:
- SQLite:
- Dao





## Quản lý sản phẩm

[Cấp phép quyền]

B0: Khai báo quyền lưu trữ và truy xuất

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

B1: Tạo data models VD: Lớp **Product**

[Sử dụng SQLite]

B2: Tạo lớp DBHelper extends SQLiteOpenHelper để tạo kết nối/update với CSDL

- ☐ Override onCreate
- ☐ Override onUpgrade
- ☐ Tạo Constructor

B3: Viết hàm CRUD sử dụng trong ứng dụng

- ☐ Tách ra trên DAO hoặc để trong Models

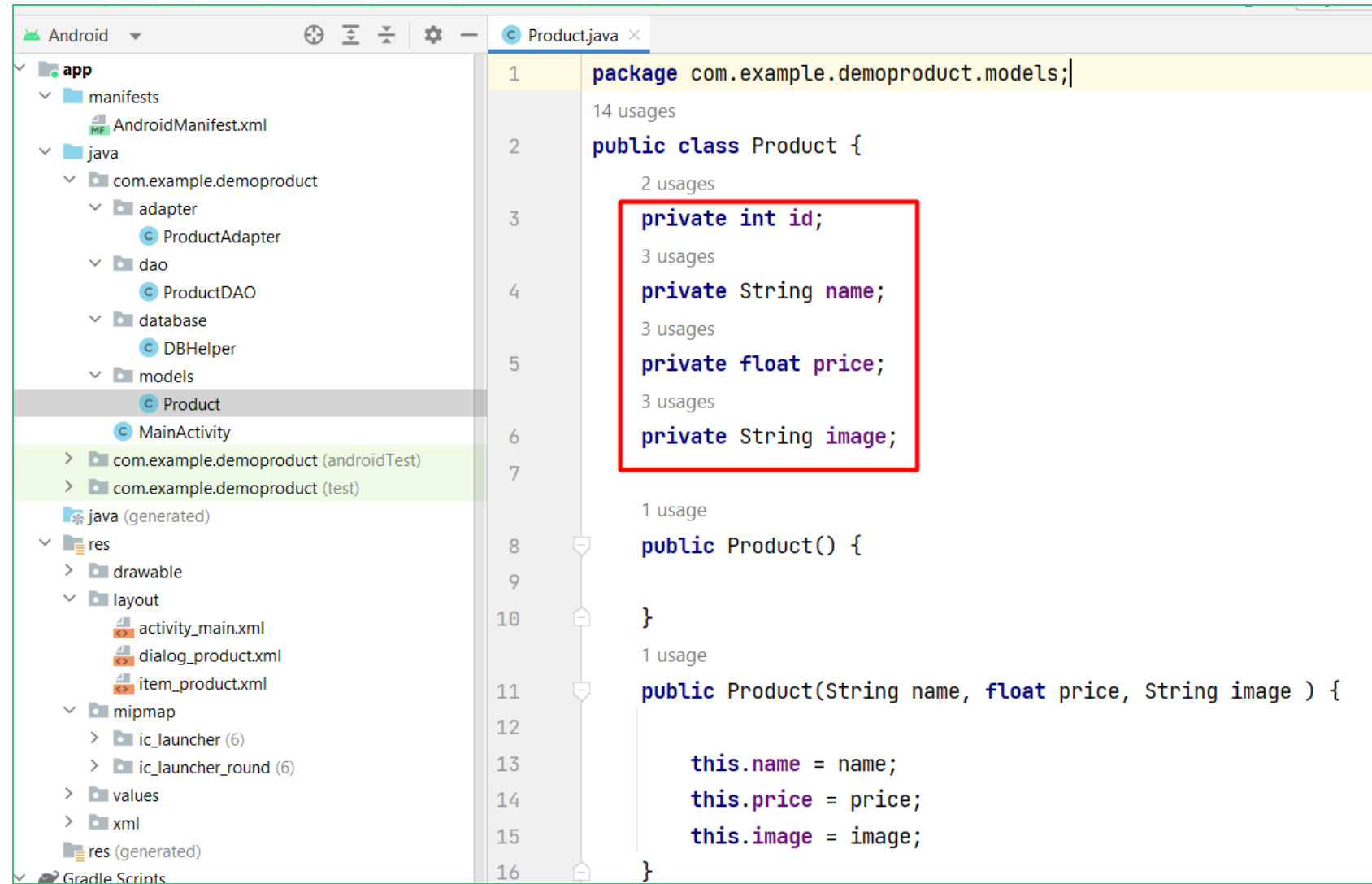
[Thực hiện giao diện]

B4: Thiết kế giao diện và thực hiện

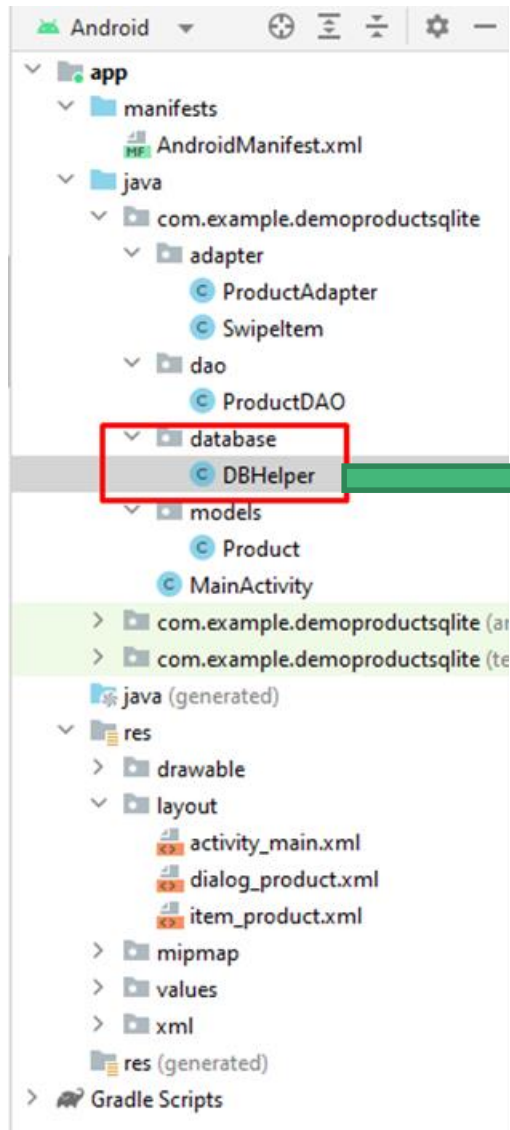
- ☐ Xây dựng ứng dụng có **RecyclerView** cho danh sách các Sản phẩm
- ☐ Cho phép thêm, xóa, sửa trên RecyclerView
- ☐ ...

## B1 – Tạo data model

### Bước 1: Tạo data models Product



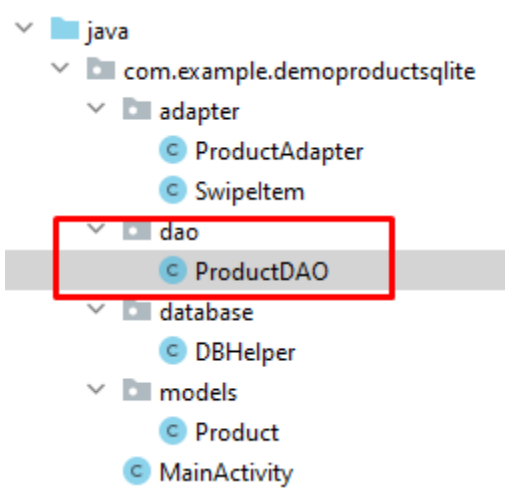
## B2 - Tạo lớp DBHelper extends SQLiteOpenHelper



```
public class DBHelper extends SQLiteOpenHelper {
    public DBHelper(Context context) {
        super(context, "QLSP", null, 1);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        String query = String.format("CREATE TABLE %s (" +
            "%s INTEGER PRIMARY KEY AUTOINCREMENT, " +
            "%s TEXT, " +
            "%s REAL, " +
            "%s REAL)",
            "Product", "id", "name", "price", "image");
        db.execSQL(query);
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        if(oldVersion != newVersion)
        {
            //drop
            String query = "DROP TABLE Product";
            db.execSQL(query);
            //create again
            onCreate(db);
        }
    }
}
```

## B3 - Viết hàm CRUD sử dụng SQLiteDatabase

### Bước 3: Tạo ProductDAO sẽ thực hiện việc CRUD trên Product



```
public class ProductDAO {
    DBHelper dbHelper;
    public ProductDAO(Context context) {
        dbHelper = new DBHelper(context);
    }
    public List<Product> GetAll()
    {
        SQLiteDatabase db = dbHelper.getWritableDatabase();
        List<Product> listProduct = new ArrayList<>();
        String query = "SELECT * FROM product";
        Cursor c = db.rawQuery(query, null);
        while (c.moveToNext())
        {
            Product temp = new Product();
            temp.setId(c.getInt(0));
            temp.setName(c.getString(1));
            temp.setPrice(c.getFloat(2));
            temp.setImage(c.getString(3));
            listProduct.add(temp);
        }
        return listProduct;
    }
}
```

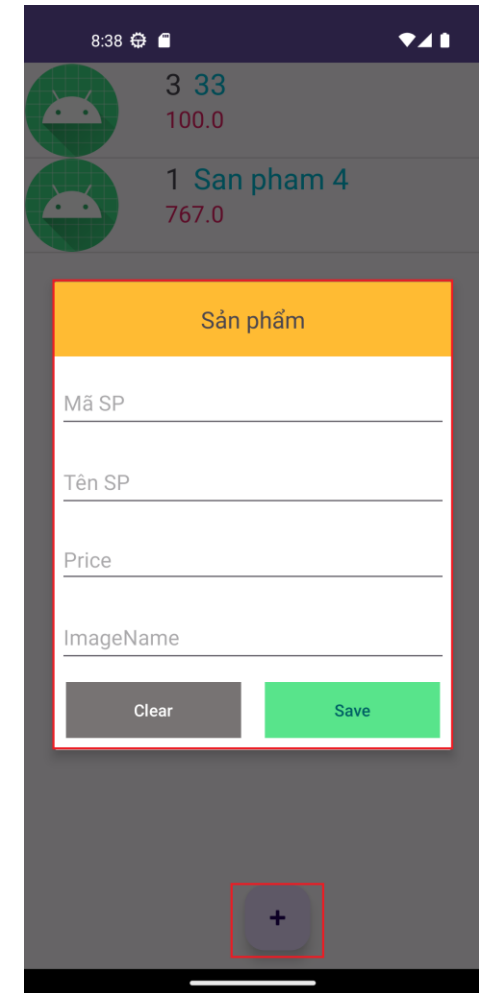
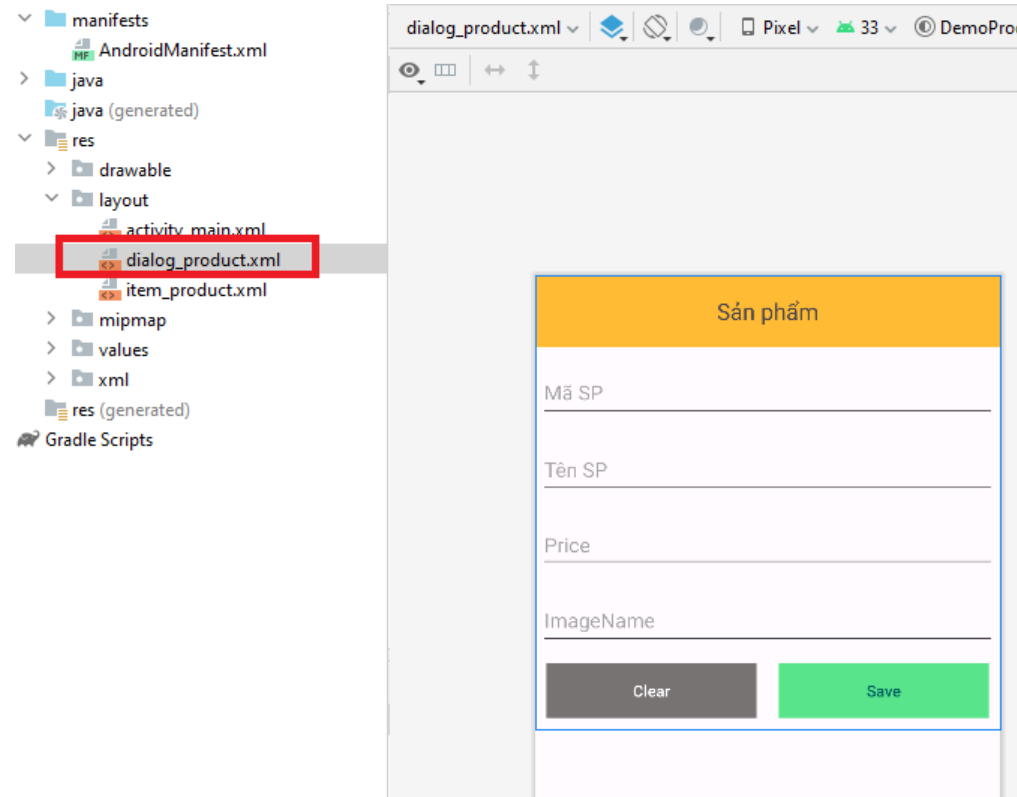
```
public void Insert(Product p) {
    SQLiteDatabase db = dbHelper.getWritableDatabase();
    //c1: sử dụng contentValues
    ContentValues values = new ContentValues();
    // values.put("id", p.getId());
    values.put("name", p.getName());
    values.put("price", p.getPrice());
    values.put("image", p.getImage());
    db.insert("product", null, values);
    //c2: sử dụng câu query thuần
    //String query = String.format("INSERT INTO %s
VALUES('%s','%s',%f,'%s')", "product", p.getId(),p.getName(),
p.getPrice(), p.getImage());
    //db.execSQL(query);
}

public void Delete(int productId) {
    SQLiteDatabase db = dbHelper.getWritableDatabase();
    //c1: sử dụng delete
    db.delete("product", "id=?", new String[] {
String.valueOf(productId) });
    //c2: sử dụng câu query thuần
    //String query = String.format("delete * from %s where id =
%d", "product", productId);
    //db.execSQL(query);
}
```

## B4 – Thực hiện CustomAlertDialog

### 4.1 Add: thêm mới sản phẩm

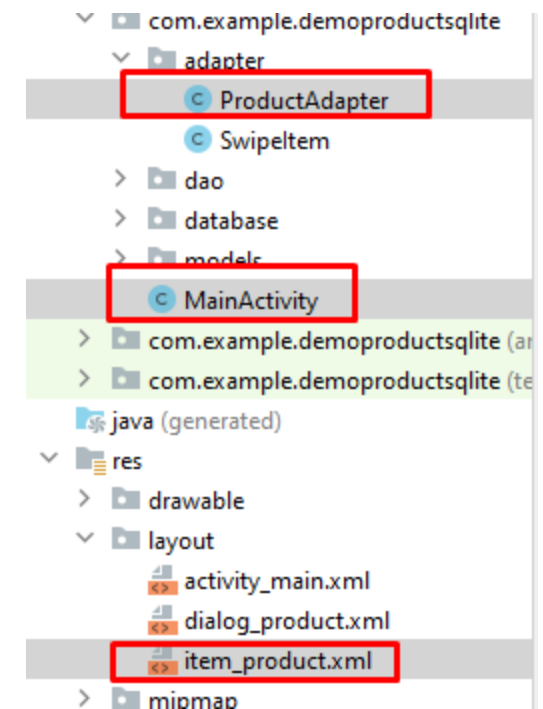
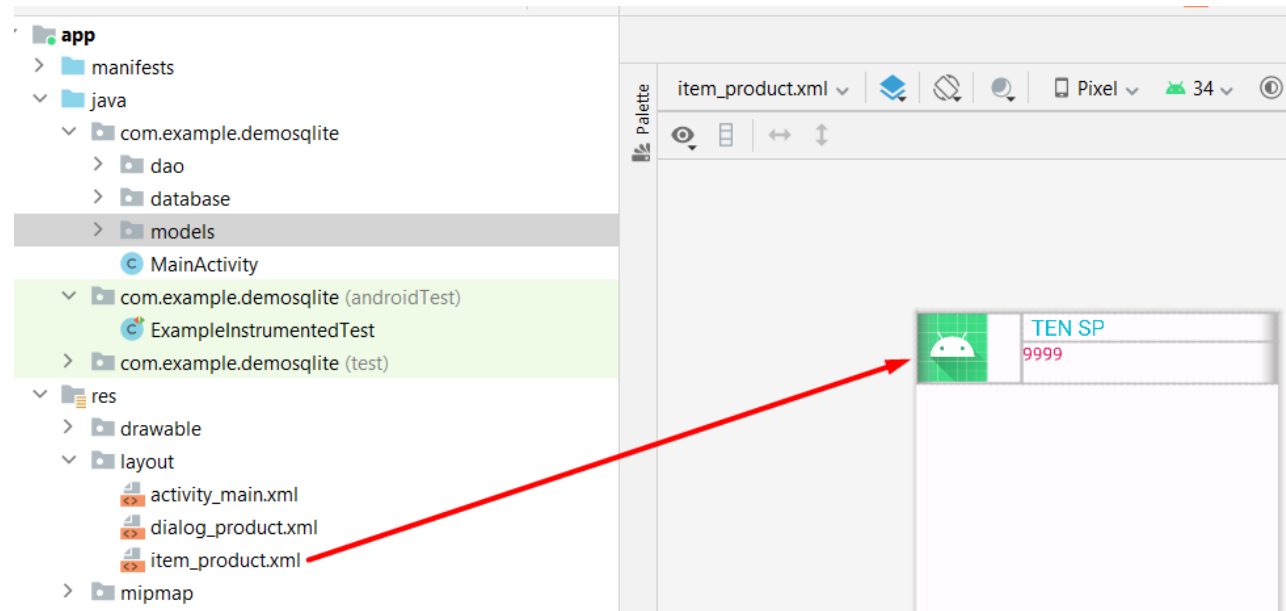
- Thiết kế CustomAlertDialog bằng cách tạo mới xml
- Add gọi customAlertDialog
- Viết sự kiện Save thêm dữ liệu vào bảng Product
- Viết sự kiện Clear



## B4 – Thực hiện RecyclerView

### 4.2 Thực hiện RecyclerView để lấy danh sách sản phẩm từ SQLite

- Thiết kế RecyclerView vào Activity
- Thiết kế giao diện **item\_product** cho recyclerView
- Thực hiện Adapter
  - ❑ **public class** ProductAdapter **extends** RecyclerView.Adapter<ProductAdapter.ViewHolder>
  - ❑ **public class** ViewHolder **extends** RecyclerView.ViewHolder
- Đổ dữ liệu List<product> được lấy từ Sqlite vào Adapter
- Kết hợp với Add



## B4 – Thực hiện RecyclerView

### 4.3 Thực hiện **Swipe** trong **Recyleview** để lấy xóa sản phẩm

- Thực hiện SwipeItem **extends** ItemTouchHelper.SimpleCallback
- Override onSwiped:
  - + Tìm vị trí product trong recyclerView
  - + Xóa product ở vị trí tìm và trả về mã sản phẩm cần xóa
  - + Thực hiện xóa sản phẩm qua DAO
  - + Cập nhật thay đổi ở recyclerView

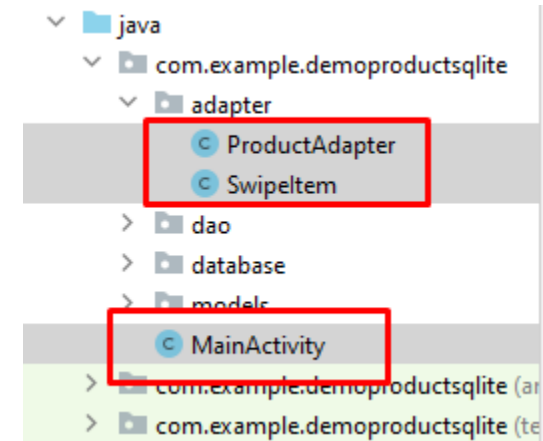
*@Override*

```
public void onSwiped(@NonNull RecyclerView.ViewHolder viewHolder, int direction) {
    int pos = viewHolder.getAdapterPosition();
    int id = this.mAdapter.deleteItem(pos);
    new ProductDAO(context).Delete(id);
    mAdapter.notifyItemRemoved(pos);
    mAdapter.notifyItemRangeChanged(pos, mAdapter.getItemCount());
}
```

- Đưa swipe vào recyclerView

*//Animators*

```
ItemTouchHelper itemTouchHelper = new ItemTouchHelper(new SwipeItem(adapter, this));
itemTouchHelper.attachToRecyclerView(rcvProduct);
```





# SHARED PREFERENCES

---

# Shared Preferences trong android

- **Shared Preferences:**

- ❑ Lưu trữ các thông tin dưới dạng **key-value** tại DATA/data/[application package name]/**shared\_prefs**/[shared\_preferences\_name].xml
- ❑ Làm việc qua 2 class: **SharedPreferences** và **SharePreferences.Editor**.

- Sử dụng **SharePreferences**:

- ❑ Tạo tên và MODE (Loại tương ứng)

**SharedPreferences** sharedpreference\_name = **getSharedPreferences**(String name, MODE);

- ❑ Một số Mode:













MODE\_PRIVATE = 0

MODE\_APPEND = 32768

MODE\_WORLD\_READABLE, MODE\_WORLD\_WRITEABLE, MODE\_MULTI\_PROCESS

# Một số hàm trong Shared Preferences

## ❑ getString/ getInt / getFloat/ getLong/ getBoolean...

 <code>getString(String s, String s1)</code>	String
 <code>edit()</code>	Editor
 <code>contains(String s)</code>	boolean
 <code>getAll()</code>	Map<String, ?>
 <code>getBoolean(String s, boolean b)</code>	boolean
 <code>getFloat(String s, float v)</code>	float
 <code>getInt(String s, int i)</code>	int
 <code>getLong(String s, long l)</code>	long
 <code>getStringSet(String s, Set&lt;String&gt; set)</code>	Set<String>
 <code>registerOnSharedPreferenceChangeListener(OnSharedPreferenceChangeListe...</code>	void
 <code>unregisterOnSharedPreferenceChangeListener(OnSharedPreferenceChangeLis...</code>	void
 <code>equals(Object obj)</code>	boolean

Press Ctrl+. to choose the selected (or first) suggestion and insert a dot afterwards [Next Tip](#)

❑ **edit()**: Lấy Editor để cho phép thay đổi dữ liệu và commit/ push tự động vào **Shared Preferences**.

❑ **contains( String key)**: Kiểm tra có chứa key

❑ **getAll()**: Lấy tất cả giá trị đã lưu ở shared Preferences.

❑ **getStringSet(String key, Set defValue)**: ...



# Lưu lại thông tin đăng nhập sau khi đăng nhập thành công

//1 - Khai báo sPref là SharedPreferences

SharedPreferences **sPref**;

//2- tìm tên "Registration\_setting" đã có trong ứng dụng chưa?

**sPref** = getApplicationContext().getSharedPreferences("Registration\_setting",  
Context.MODE\_PRIVATE);

if(**sPref** != null) {

String LOGIN\_NAME = **sPref**.getString("KEY\_LOGIN\_NAME", "");

String PASS = **sPref**.getString("KEY\_LOGIN\_PASS", "");

if(LOGIN\_NAME != "") {

**txtUser**.setText(LOGIN\_NAME);

**txtPass**.setText(PASS);

}

}

findViewById(R.id.**btnlogin**).setOnClickListener(new View.OnClickListener() {

@Override

**public void** onClick(View view) {

//3 - Lưu giá trị Key – value vào "Registration\_setting" đã có

SharedPreferences.Editor editor = **sPref**.edit();

editor.putString("KEY\_LOGIN\_NAME", **txtUser**.getText().toString() ).apply();

editor.putString("KEY\_LOGIN\_PASS", **txtPass**.getText().toString() ).apply();

}

});



## Exercise 6.1: SQLite & custom DialogView cho QLSV

### Ứng dụng quản lý Sinh viên và Khoa

#### ❑ Quản lý Khoa:

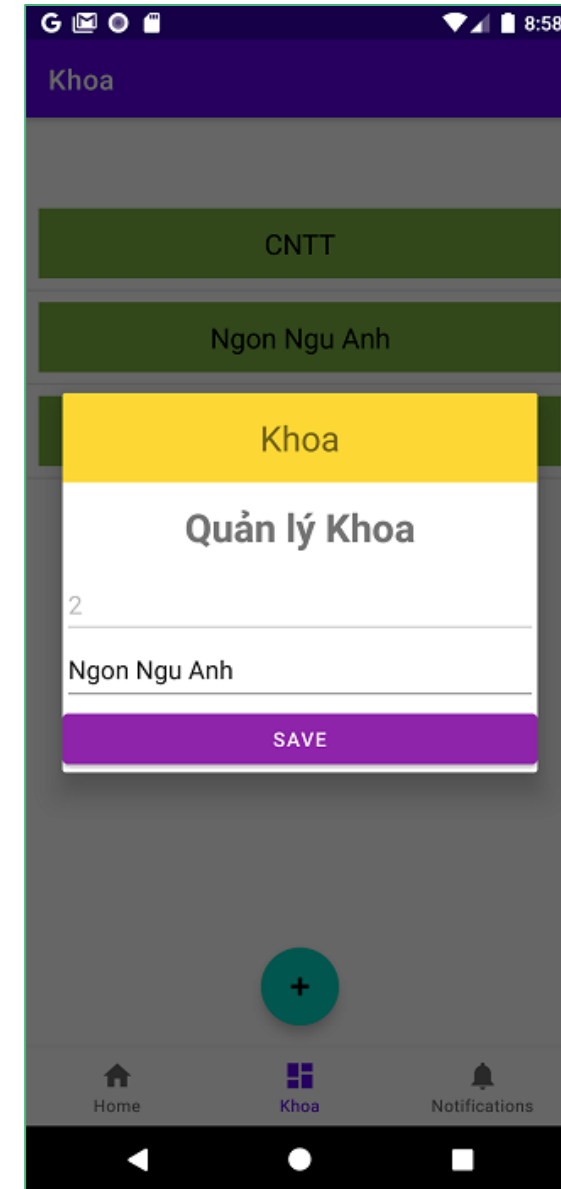
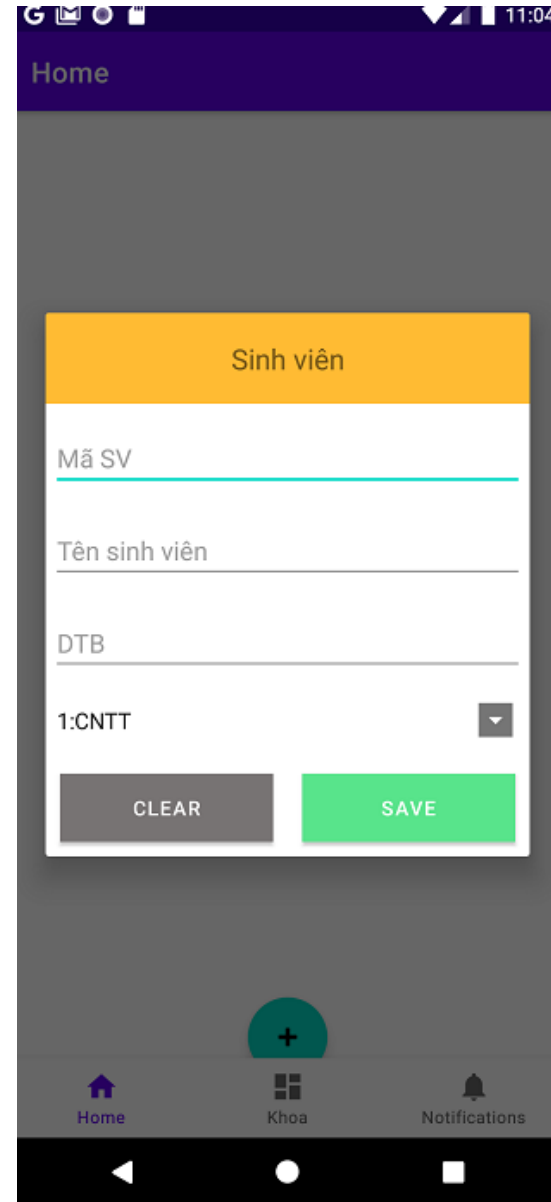
Hiển thị tất cả các khoa ở RecyclerView trong Khoa fragment

Thêm, sửa: Show CustomDialogView

Xóa: Swipe trên item

#### ❑ Quản lý sinh viên (Home)

Tương tự Khoa, nhưng mỗi sinh viên được chọn thuộc về 1 khoa bất kì (khoa lấy từ CSDL) và đưa lên **Spinner**



## Exercise 6.2 Ứng dụng hiển thị tin tức

### Chương trình ứng dụng xem tin tức

1. Sử dụng **Recycler View** Thiết kế chương trình xem tin tức. Mỗi tin có: Hình ảnh, Title , Url
2. Khi vào **FloatActionButton** sẽ cho phép thêm mới 1 tin tức (các hình ảnh được lấy từ internet)  
Sử dụng Custom DialogView thay vì Activity
3. Khi OnClick trên item sẽ mở “DetailActivity” có **WebView** để hiển thị nội dung tin tức và có thể **back** về.
4. Xóa tin trên RecyclerView, Lấy hình ảnh từ CSDL

