Nhóm 8

# Cơ sở lý luận

Nghiên cứu cơ sở dữ liệu sử dụng trong android

* Vì sao lại làm sản phẩm?

+ hầu hết các điện thoại thông minh dùng android đã có phần ghi chú có sẵn, nhưng tính năng hạn chế ví dụ:

Không nhúng ảnh hoặc video vào ghi chú

Có ghi chú nhưng không có nhắc lịch

Không có phần chọn địa điểm ghi chú

Không có nhắc lịch theo địa điểm, ví dụ chỉ nhắc lịch khi người dùng vào vùng địa điểm định sẵn

Không có phẩn bảo mật ghi chú nên ai sử dụng điện thoại cũng có thể dùng được

+ với ý tưởng viết một ứng dụng ghi chú thay thế và có những tính năng đáp ứng những hạn chế nêu trên, cũng như đáp ứng yêu cầu môn học là tìm hiểu về hệ quản trị cơ sở dữ liệu SQL Lite trong Android nên nhóm đã chọn

Xây dựng sản phẩm ghi chú dành cho điện thoại sử dụng SQL Lite trong android

# Cơ sở lý thuyết và kỹ thuật

Theo đề tài nhóm đã chọn, chúng em nghiên cứu các cơ sở lý thuyết và kỹ thuật để đáp ứng được yêu cầu chức năng của ứng dụng

1. Phần sử dụng sharedprefrences để lưu trữ mật khẩu bảo mật App. Phần này nghiên cứu của học viên Phạm Thị Hải Vân

**Khái niệm sharedpreferences**

Như chúng ta biết android là một nền tảng dùng cho thiết bị di động mà nhân của nó là hệ điều hành linux, sử dụng máy ảo java kvm, vì được thiết kế để chạy các ứng dụng di động nên luôn luôn phải có phần lưu trữ dữ liệu người dùng khi thực thi ứng dụng cụ thể

Để lưu trữ dữ liệu đơn giản, kiến trúc của 1 ứng dụng android có 2 kỹ thuật đơn giản nhất để lưu trữ thông tin đó là

Shared Preference

Cặp giá trị giữa giá trị và trạng thái activity

Về nguyên tắc chung, việc lưu trữ này là lưu trữ các cặp key/value giống như cơ chế của Map trong java

Để tạo hoặc thay đổi một Shared Preference, gọi getSharedPreferences trên ứng dụng ngữ cảnh, chuyển sang tên của Shared Preference để thay đổi. Shared Preference được chia sẻ qua các thành phần của một ứng dụng nhưng không thể dùng được cho các ứng dụng khác.

Để thay đổi một Shared Preference, sử dụng lớp SharedPreferences.Editor. Lấy đối tượng Editor bằng cách gọi edit trên đối tượng SharedPreferences của đối tượng muốn thay đổi. Để lưu các edit, gọi commit trên Editor, giống như mã nguồn bên dưới.

|  |
| --- |
| public static final String MYPREFS = “mySharedPreferences”;  protected void savePreferences(){  // Create or retrieve the shared preference object.  // Tạo hoặc nhận đối tượng shared preference  int mode = Activity.MODE\_PRIVATE;  SharedPreferences mySharedPreferences = getSharedPreferences(MYPREFS, mode);  // Retrieve an editor to modify the shared preferences.  // Nhận một editor để thay đổi shared preferences  SharedPreferences.Editor editor = mySharedPreferences.edit();  // Store new primitive types in the shared preferences object.  // Lưu các kiểu nguyên thủy trong đối tượng shared preferences.  editor.putBoolean(“isTrue”, true);  editor.putFloat(“lastFloat”, 1f); editor.putInt(“wholeNumber”, 2); editor.putLong(“aNumber”, 3l); editor.putString(“textEntryValue”, “Not Empty”);  // Commit the changes.  // Đưa ra các thay đổi.  editor.commit();  } |

**Lấy ra Shared Preferences**

Truy cập Shared Preferences đã lưu cũng có thể được làm với phương thức getSharedPreferences. Qua tên của Shared Preference ta muốn truy cập, và sử dụng các loại phương pháp an toàn get <type> để trích xuất các giá trị lưu.

Mỗi getter lấy một khóa và một giá trị mặc định (dùng khi không có giá trị nào có thể dùng được cho khóa đó), như trong mã nguồn dưới đây:

|  |
| --- |
| public void loadPreferences(){  // Get the stored preferences  // Lấy các sở thích đã được lưu trữ.  int mode = Activity.MODE\_PRIVATE;  SharedPreferences mySharedPreferences = getSharedPreferences(MYPREFS, mode);  // Retrieve the saved values.  // Lấy các giá trị đã lưu  boolean isTrue = mySharedPreferences.getBoolean(“isTrue”, false); float lastFloat = mySharedPreferences.getFloat(“lastFloat”, 0f); int wholeNumber = mySharedPreferences.getInt(“wholeNumber”, 1); long aNumber = mySharedPreferences.getLong(“aNumber”, 0);  String stringPreference;  stringPreference = mySharedPreferences.getString(“textEntryValue”, “”);  } |

**Mội ứng dụng nữa của sharedpreferences là Lưu trạng thái hoạt động**

Nếu chúng ta muốn lưu thông tin hoạt động mà không cần phải chia sẻ với các thành phần khác (ví dụ: ví dụ lớp các biến), ta có thể gọi Activity.getPreferences() mà không cần đặc tả tên một preferences. Truy cập tới bản đồ Shared Preferences trả về bị hạn chế việc gọi hoạt động; mỗi hoạt động hỗ trợ một đối tượng SharedPreferences không tên riêng lẻ.

Mã nguồn dưới đây thể hiện làm cách nào để sử dụng Shared Preferences riêng của hoạt động.

|  |
| --- |
| protected void saveActivityPreferences(){  // Create or retrieve the activity preferences object.  // Tạo hay nhận đối tượng hoạt động preferences  SharedPreferences activityPreferences = getPreferences(Activity.MODE\_PRIVATE);  // Retrieve an editor to modify the shared preferences.  // Nhận một editor để thay đổi shared preferences  SharedPreferences.Editor editor = activityPreferences.edit();  // Retrieve the View  // Nhận View  TextView myTextView = (TextView)findViewById(R.id.myTextView);  // Store new primitive types in the shared preferences object.  // Lưu trữ các loại sơ khai trong đối tượng shared preferences  editor.putString(“currentTextValue”,  myTextView.getText().toString());  // Commit changes.  // Đưa ra các thay đổi.  editor.commit();  } |

**Lưu và trả lại trường hợp trạng thái**

Để lưu các biến trường hợp hoạt động, Android đưa ra một thay thế chuyên dụng cho Shared Preferences.

Bằng cách ghi đè xử lý sự kiện onSaveInstanceState của một hoạt động, ta có thể sử dụng tham số Bundle của nó để lưu các giá trị ví dụ. Lưu trữ các giá trị sử dụng cùng các phương thức get và put như đã trình bày cho Shared Preferences, trước khi đi qua các Bundle được chỉnh sửa vào xử lý của lớp cha, như thể hiện trong đoạn mã sau

|  |
| --- |
| private static final String TEXTVIEW\_STATE\_KEY = “TEXTVIEW\_STATE\_KEY”;  [@Override](mailto:@Override)  public void onSaveInstanceState(Bundle outState) {  // Retrieve the View  // Nhận View  TextView myTextView = (TextView)findViewById(R.id.myTextView);  // Save its state  // Lưu trữ trạng thái của nó  outState.putString(TEXTVIEW\_STATE\_KEY, myTextView.getText().toString());  super.onSaveInstanceState(outState);  } |

Xử lý này sẽ được kích hoạt bất cứ khi nào một hoạt động hoàn thành vòng đời của hoạt động của nó, nhưng chỉ khi nó không được hoàn thành một cách rõ ràng. Kết quả là, nó được sử dụng để đảm bảo trạng thái Hoạt động nhất quán giữa các vòng đời hoạt động của một phiên làm việc duy nhất.

Bundle lưu được thông qua trong các phương pháp onRestoreInstanceState và onCreate nếu ứng dụng bắt buộc phải khởi động lại trong một phiên làm việc. Đoạn mã sau đây cho thấy làm thế nào để trích xuất các giá trị từ các Bundle và sử dụng chúng để cập nhật trạng thái Hoạt động:

|  |
| --- |
| [@Override](mailto:@Override)  public void onCreate(Bundle icicle) { super.onCreate(icicle); setContentView(R.layout.main);  TextView myTextView = (TextView)findViewById(R.id.myTextView); String text = “”;  if (icicle != null && icicle.containsKey(TEXTVIEW\_STATE\_KEY))  text = icicle.getString(TEXTVIEW\_STATE\_KEY);  myTextView.setText(text);  } |

Điều quan trọng là phải nhớ rằng onSaveInstanceState chỉ được gọi khi một hoạt động trở thành không hoạt động, nhưng không phải khi nó đang được đóng lại bằng việc gọi kết thúc hoặc do người sử dụng nhấn vào nút Back.

**Ví dụ 1 ứng dụng sử dụng Lưu trạng thái hoạt động To-Do List dùng sharedpreferences**

Hiện nay, mỗi khi ứng dụng mẫu To-Do List được khởi động lại, tất cả các mục phải làm bị mất và bất kỳ văn bản nhập vào ô nhập văn bản sẽ bị xóa. Trong ví dụ này, ta sẽ bắt đầu để lưu các trạng thái ứng dụng của ứng dụng To-Do list qua các phiên làm việc.

Trạng thái trong Hoạt động ToDoList bao gồm ba biến:

- Có phải một mục mới được thêm vào?

- Văn bản gì tồn tại trong textbox mới nhập hàng?

- Mục đang được chọn là gì?

Sử dụng Shared Preference mặc định của Hoạt động, ta có thể lưu trữ mỗi giá trị và cập nhật giao diện người dùng khi Hoạt động được khởi động lại.

Sau đó trong chương này, ta cũng sẽ tìm hiểu làm thế nào để sử dụng cơ sở dữ liệu SQLite để tiếp tục các mục To-Do. Ví dụ này là bước đầu cho thấy làm thế nào để đảm bảo một trải nghiệm liền mạch bằng cách lưu những chi tiết mẫu của hoạt động.

Bắt đầu bằng việc thêm các biến tĩnh String để sử dụng như là các khóa preference.

|  |
| --- |
| private static final String TEXT\_ENTRY\_KEY = “TEXT\_ENTRY\_KEY”;  private static final String ADDING\_ITEM\_KEY = “ADDING\_ITEM\_KEY”;  private static final String SELECTED\_INDEX\_KEY = “SELECTED\_INDEX\_KEY”; |

Tiếp theo, chép đè phương thức onPause. Nhận tin đối tượng Sở thích chung của hoạt động, và có được đối tượng biên tập viên của nó.

Sử dụng các phím mà bạn đã tạo ở bước một, lưu trữ các giá trị mẫu dựa trên việc một mục mới đang được bổ sung và bất kỳ văn bản trong "mục mới" Edit Box được hay không.

|  |
| --- |
| [@Override](mailto:@Override)  protected void onPause(){  super.onPause();  // Get the activity preferences object.  // Lấy đối tượng hoạt động preferences  SharedPreferences uiState = getPreferences(0);  // Get the preferences editor.  // Lấy editor preferences  SharedPreferences.Editor editor = uiState.edit();  // Add the UI state preference values.  // Thêm giá trị trạng thái preference giao diện người dùng  editor.putString(TEXT\_ENTRY\_KEY, myEditText.getText().toString());  editor.putBoolean(ADDING\_ITEM\_KEY, addingNew);  // Commit the preferences  // Đưa ra các preferences.  editor.commit();  } |

Viết một phương thức restoreUIState áp dụng các ví dụ giá trị mà bạn ghi lại ở bước 2 khi khởi động lại ứng dụng.

Sửa đổi các phương thức onCreate để thêm một cuộc gọi đến các phương pháp restoreUIState ở cuối cùng.

|  |
| --- |
| [@Override](mailto:@Override)  public void onCreate(Bundle icicle) {  [ ... existing onCreate logic ... ]  restoreUIState();  }  private void restoreUIState() {  // Get the activity preferences object.  // Lấy đối tượng hoạt động preferences  SharedPreferences settings = getPreferences(Activity.MODE\_PRIVATE);  // Read the UI state values, specifying default values.  // Đọc giá trị trạng thái giao diện người dùng, đặc tả các giá trị mặc định  String text = settings.getString(TEXT\_ENTRY\_KEY, “”);  Boolean adding = settings.getBoolean(ADDING\_ITEM\_KEY, false);  // Restore the UI to the previous state.  // Khôi phục giao diện người dùng về trạng thái trước đó  if (adding) {  addNewItem();  myEditText.setText(text);  }  } |

Ghi chỉ số của các mục đã chọn sử dụng kỹ thuật onSaveInstanceState / onRestore InstanceState. Sau đó chỉ lưu và áp dụng nếu như ứng dụng bị hủy mà không có sự cho phép của người sử dụng.

|  |
| --- |
| [@Override](mailto:@Override)  public void onSaveInstanceState(Bundle outState) {  outState.putInt(SELECTED\_INDEX\_KEY, myListView.getSelectedItemPosition());  super.onSaveInstanceState(outState);  }  [@Override](mailto:@Override)  public void onRestoreInstanceState(Bundle savedInstanceState) {  int pos = -1;  if (savedInstanceState != null)  if (savedInstanceState.containsKey(SELECTED\_INDEX\_KEY))  pos = savedInstanceState.getInt(SELECTED\_INDEX\_KEY, -1);  myListView.setSelection(pos);  } |

Khi ta chạy các ứng dụng To-Do List, chúng ta sẽ thấy tình trạng giao diện người dùng liên tục theo phiên làm việc. Điều đó nói rằng, nó vẫn sẽ không tồn tại các mục trong danh sách công việc phải làm - ta sẽ thêm phần cần thiết của chức năng này ở phần sau của chương.

**Ví dụ 2 Tạo một trang Preferences cho người quan sát động đất**

Trong chương 5, ta tạo ra một màn hình động đất cho thấy một danh sách các trận động đất gần đây dựa trên nguồn dữ liệu từ Internet.

Trong ví dụ sau đây, ta sẽ tạo ra một trang Preferences cho người quan sát động đất, cho phép người dùng cấu hình cài đặt ứng dụng cho nhiều kinh nghiệm cá nhân hơn. Ta sẽ cung cấp tùy chọn để chuyển đổi tự động cập nhật, kiểm soát tần số cập nhật, và lọc các trận động đất cường độ tối thiểu hiển thị.

Sau đó trong chương này, ta sẽ mở rộng ví dụ này hơn nữa bằng cách tạo ra một nhà cung cấp nội dung để lưu và chia sẻ dữ liệu động đất với các ứng dụng khác.

1. Mở dự án Earthquake đã tạo ra trong Chương 5.

Thêm nguồn String mới cho các nhãn hiển thị trong màn hình "Preferences". Ngoài ra, thêm một chuỗi cho các Menu Item mới sẽ cho phép người dùng truy cập vào màn hình Preferences.

|  |
| --- |
| <?xml version=”1.0” encoding=”utf-8”?>  <resources>  <string name=”app\_name”>Earthquake</string>  <string name=”quake\_feed”>  <http://earthquake.usgs.gov/eqcenter/catalogs/1day-M2.5.xml>  </string>  <string name=”menu\_update”>Refresh Earthquakes</string>  <string name=”auto\_update\_prompt”>Auto Update?</string>  <string name=”update\_freq\_prompt”>Update Frequency</string>  <string name=”min\_quake\_mag\_prompt”>Minimum Quake Magnitude</string>  <string name=”menu\_preferences”>Preferences</string>  </resources> |

2. Tạo một tài nguyên layout preferences.xml mới, đưa ra các giao diện người dùng cho các Hoạt động Preferences. Bao gồm một hộp kiểm cho thấy sự "cập nhật tự động" chuyển đổi, và quay để chọn tốc độ cập nhật và bộ lọc độ lớn.

|  |
| --- |
| <LinearLayout [xmlns:android=”http://schemas.android.com/apk/res/android”](http://schemas.android.com/apk/res/android) android:orientation=”vertical” android:layout\_width=”fill\_parent” android:layout\_height=”fill\_parent”>  <TextView android:layout\_width=”fill\_parent” android:layout\_height=”wrap\_content” [android:text=”@string/auto\_update\_prompt”](mailto:@string)  />  <CheckBox [android:id=”@+id/checkbox\_auto\_update”](mailto:@+id) android:layout\_width=”fill\_parent” android:layout\_height=”wrap\_content”  />  <TextView android:layout\_width=”fill\_parent” android:layout\_height=”wrap\_content”android:text=”@string/update\_freq\_prompt”  />  <Spinner [android:id=”@+id/spinner\_update\_freq”](mailto:@+id) android:layout\_width=”fill\_parent” android:layout\_height=”wrap\_content” android:drawSelectorOnTop=”true”  />  <TextView android:layout\_width=”fill\_parent” android:layout\_height=”wrap\_content” [android:text=”@string/min\_quake\_mag\_prompt”](mailto:@string)  />  <Spinner [android:id=”@+id/spinner\_quake\_mag”](mailto:@+id) android:layout\_width=”fill\_parent” android:layout\_height=”wrap\_content” android:drawSelectorOnTop=”true”  />  <LinearLayout android:orientation=”horizontal” android:layout\_width=”fill\_parent” android:layout\_height=”wrap\_content”>  <Button  [android:id=”@+id/okButton”](mailto:@+id) android:layout\_width=”wrap\_content” android:layout\_height=”wrap\_content” [android:text=”@android:string/ok”](mailto:@android:string)  />  <Button [android:id=”@+id/cancelButton”](mailto:@+id) android:layout\_width=”wrap\_content” android:layout\_height=”wrap\_content” [android:text=”@android:string/cancel”](mailto:@android:string)  />  </LinearLayout>  </LinearLayout> |

3. Tạo bốn nguồn mảng mới trong tệp res/values/arrays.xml mới. Chúng sẽ cung cấp các giá trị sử dụng cho các bản cập nhật tần suất và cường độ lọc tối thiểu.

|  |
| --- |
| <?xml version=”1.0” encoding=”utf-8”?>  <resources>  <string-array name=”update\_freq\_options”>  <item>Every Minute</item>  <item>5 minutes</item>  <item>10 minutes</item>  <item>15 minutes</item>  <item>Every Hour</item>  </string-array>  <array name=”magnitude”>  <item>3</item>  <item>5</item>  <item>6</item>  <item>7</item>  <item>8</item>  </array>  <string-array name=”magnitude\_options”>  <item>3</item>  <item>5</item>  <item>6</item>  <item>7</item>  <item>8</item>  </string-array>  <array name=”update\_freq\_values”>  <item>1</item>  <item>5</item>  <item>10</item>  <item>15</item>  <item>60</item>  </array>  </resources> |

4. Tạo các Hoạt động Preferences. Nó sẽ được sử dụng để hiển thị các ứng dụng preferences.

Ghi đè onCreate để đưa bố trí mà bạn đã tạo ở bước 2, và nhận được tài liệu tham khảo để các

Checkbox và cả hai điều khiển Spinner. Sau đó thực hiện gọi đến các populateSpinners còn sơ khai.

|  |
| --- |
| package com.paad.earthquake;  import android.app.Activity;  import android.content.SharedPreferences;  import android.content.SharedPreferences.Editor;  import android.os.Bundle;  import android.view.View;  import android.widget.ArrayAdapter;  import android.widget.Button; import android.widget.CheckBox; import android.widget.Spinner;  public class Preferences extends Activity { CheckBox autoUpdate;  Spinner updateFreqSpinner;  Spinner magnitudeSpinner;  [@Override](mailto:@Override)  public void onCreate(Bundle icicle) { super.onCreate(icicle); setContentView(R.layout.preferences);  updateFreqSpinner = (Spinner)findViewById(R.id.spinner\_update\_freq);  magnitudeSpinner = (Spinner)findViewById(R.id.spinner\_quake\_mag);  autoUpdate = (CheckBox)findViewById(R.id.checkbox\_auto\_update);  populateSpinners();  }  private void populateSpinners() {  }  } |

5. Điền vào các phương thức populateSpinners, sử dụng adapter mảng để ràng buộc mỗi Spinner đến mảng tương ứng của nó

|  |
| --- |
| private void populateSpinners() {  // Populate the update frequency spinner  // Gắn vào bản cập nhật tần số spinner  ArrayAdapter<CharSequence> fAdapter;  fAdapter = ArrayAdapter.createFromResource(this, R.array.update\_freq\_options, android.R.layout.simple\_spinner\_item);  fAdapter.setDropDownViewResource(  android.R.layout.simple\_spinner\_dropdown\_item);  updateFreqSpinner.setAdapter(fAdapter);  // Populate the minimum magnitude spinner  //Gắn vào spinner độ lớn nhỏ nhất  ArrayAdapter<CharSequence> mAdapter;  mAdapter = ArrayAdapter.createFromResource(this, R.array.magnitude\_options, android.R.layout.simple\_spinner\_item);  mAdapter.setDropDownViewResource(  android.R.layout.simple\_spinner\_dropdown\_item);  magnitudeSpinner.setAdapter(mAdapter);  } |

1. Thêm giá trị toàn cục chuỗi tĩnh để sử dụng để xác định các Shared Preference ta sắp tạo ra, và các khóa sẽ sử dụng để lưu trữ giá trị của từng preference. Cập nhật các phương thức onCreate để lấy lại preference được đặt tên và gọi updateUIFromPreferences. Phương thức updateUIFromPreferences sử dụng các phương thức get <type> trên đối tượng Shared Preference để lấy giá trị của từng preference và áp dụng nó vào giao diện người dùng hiện tại.

|  |
| --- |
| public static final String USER\_PREFERENCE = “USER\_PREFERENCES”;  public static final String PREF\_AUTO\_UPDATE = “PREF\_AUTO\_UPDATE”;  public static final String PREF\_MIN\_MAG = “PREF\_MIN\_MAG”;  public static final String PREF\_UPDATE\_FREQ = “PREF\_UPDATE\_FREQ”; SharedPreferences prefs;  [@Override](mailto:@Override)  public void onCreate(Bundle icicle) { super.onCreate(icicle); setContentView(R.layout.preferences);  updateFreqSpinner = (Spinner)findViewById(R.id.spinner\_update\_freq); magnitudeSpinner = (Spinner)findViewById(R.id.spinner\_quake\_mag); autoUpdate = (CheckBox)findViewById(R.id.checkbox\_auto\_update);  populateSpinners();  prefs = getSharedPreferences(USER\_PREFERENCE, Activity.MODE\_PRIVATE);  updateUIFromPreferences();  }  private void updateUIFromPreferences() {  boolean autoUpChecked = prefs.getBoolean(PREF\_AUTO\_UPDATE, false);  int updateFreqIndex = prefs.getInt(PREF\_UPDATE\_FREQ, 2);  int minMagIndex = prefs.getInt(PREF\_MIN\_MAG, 0);  updateFreqSpinner.setSelection(updateFreqIndex);  magnitudeSpinner.setSelection(minMagIndex);  autoUpdate.setChecked(autoUpChecked);  } |

1. Vẫn còn trong phương thức onCreate, thêm xử lý sự kiện cho nút OK và Cancel. Cancel nên đóng Hoạt động, trong khi OK nên gọi savePreferences trước.

|  |
| --- |
| [@Override](mailto:@Override)  public void onCreate(Bundle icicle) { super.onCreate(icicle); setContentView(R.layout.preferences);  updateFreqSpinner = (Spinner)findViewById(R.id.spinner\_update\_freq); magnitudeSpinner = (Spinner)findViewById(R.id.spinner\_quake\_mag); autoUpdate = (CheckBox)findViewById(R.id.checkbox\_auto\_update);  populateSpinners();  prefs = getSharedPreferences(USER\_PREFERENCE, Activity.MODE\_PRIVATE);  updateUIFromPreferences();  Button okButton = (Button) findViewById(R.id.okButton);  okButton.setOnClickListener(new View.OnClickListener() {  public void onClick(View view) { savePreferences(); Preferences.this.setResult(RESULT\_OK); finish();  }  });  Button cancelButton = (Button) findViewById(R.id.cancelButton);  cancelButton.setOnClickListener(new View.OnClickListener() {  public void onClick(View view) { Preferences.this.setResult(RESULT\_CANCELED); finish();  }  });  }  private void savePreferences() {  } |

8. Điền vào các phương thức savePreferences để ghi lại các preferences hiện hành, dựa trên chọn lọc giao diện người dùng, đến đối tượng Shared Preference.

|  |
| --- |
| private void savePreferences() {  int updateIndex = updateFreqSpinner.getSelectedItemPosition(); int minMagIndex = magnitudeSpinner.getSelectedItemPosition(); boolean autoUpdateChecked = autoUpdate.isChecked();  Editor editor = prefs.edit();  editor.putBoolean(PREF\_AUTO\_UPDATE, autoUpdateChecked); editor.putInt(PREF\_UPDATE\_FREQ, updateIndex); editor.putInt(PREF\_MIN\_MAG, minMagIndex); editor.commit();  } |

1. Hoàn thành Hoạt động Preferences. Làm cho nó có thể truy cập vào các ứng dụng bằng cách thêm vào các biểu hiện ứng dụng.

|  |
| --- |
| <activity android:name=”.Preferences”  android:label=”Earthquake Preferences”>  </activity> |

10. Bây giờ quay trở lại các hoạt động động đất, và thêm hỗ trợ cho các tập tin Shared Preferences mới và một mục trình đơn để hiển thị các Hoạt động Preferences.

Bắt đầu bằng cách thêm mới mục trình đơn. Mở rộng các phương pháp onCreateOptionsMenu để bao gồm một mục mới để mở Hoạt động Preferences.

|  |
| --- |
| static final private int MENU\_PREFERENCES = Menu.FIRST+1;  [@Override](mailto:@Override)  public boolean onCreateOptionsMenu(Menu menu) {  super.onCreateOptionsMenu(menu);  menu.add(0, MENU\_UPDATE, Menu.NONE, R.string.menu\_update);  menu.add(0, MENU\_PREFERENCES, Menu.NONE, R.string.menu\_preferences);  return true;  } |

11. Sửa đổi các phương pháp onOptionsItemSelected để hiển thị các Hoạt động Preferences khi mục trình đơn mới được chọn. Tạo một ý định rõ ràng, và vượt qua nó để phương pháp startActivityForResult. Điều này sẽ khởi động màn hình Preferences và cảnh báo các lớp Earthquake khi đi những preferences được thông qua các xử lý onActivityResult.

|  |
| --- |
| private static final int SHOW\_PREFERENCES = 1;  public boolean onOptionsItemSelected(MenuItem item) {  super.onOptionsItemSelected(item);  switch (item.getItemId()) {  case (MENU\_UPDATE): { refreshEarthquakes(); return true;  }  case (MENU\_PREFERENCES): {  Intent i = new Intent(this, Preferences.class); startActivityForResult(i, SHOW\_PREFERENCES); return true;  }  }  return false;  } |

12. Chạy ứng dụng, và chọn Preference từ bảng chọn hoạt động. Hoạt động Preference được hiển thị giống như Hình 6.1

|  |
| --- |
|  |
| Hình 6.1 |

13. Tất cả những gì còn lại là áp dụng preferences cho các chức năng Earthquake.

Thực hiện cập nhật tự động sẽ được để lại cho đến Chương 8, lúc đó ta sẽ tìm hiểu làm thế nào để sử dụng dịch vụ và chủ đề nền. Bây giờ, ta có thể đặt khung tại chỗ và áp dụng bộ lọc độ lớn.

14. Bắt đầu bằng cách tạo ra một phương pháp mới updateFromPreferences mà đọc các giá trị Shared Preference và tạo ra các biến ví dụ cho mỗi biến trong số chúng.

|  |
| --- |
| int minimumMagnitude = 0; boolean autoUpdate = false; int updateFreq = 0;  private void updateFromPreferences() { SharedPreferences prefs =  getSharedPreferences(Preferences.USER\_PREFERENCE, Activity.MODE\_PRIVATE);  int minMagIndex = prefs.getInt(Preferences.PREF\_MIN\_MAG, 0);  if (minMagIndex < 0)  minMagIndex = 0;  int freqIndex = prefs.getInt(Preferences.PREF\_UPDATE\_FREQ, 0);  if (freqIndex < 0)  freqIndex = 0;  autoUpdate = prefs.getBoolean(Preferences.PREF\_AUTO\_UPDATE, false); Resources r = getResources();  // Get the option values from the arrays.  // Lấy giá trị lựa chọn từ các mảng  int[] minMagValues = r.getIntArray(R.array.magnitude);  int[] freqValues = r.getIntArray(R.array.update\_freq\_values);  // Convert the values to ints.  // Chuyển các giá trị sang số nguyên  minimumMagnitude = minMagValues[minMagIndex];  updateFreq = freqValues[freqIndex];  } |

15. Áp dụng các bộ lọc độ lớn bằng cách cập nhật các phương thức addNewQuake để kiểm tra độ mạnh của một trận động đất mới trước khi thêm nó vào danh sách.

|  |
| --- |
| private void addNewQuake(Quake \_quake) {  if (\_quake.getMagnitude() > minimumMagnitude) {  // Add the new quake to our list of earthquakes.  // Thêm rung động mới vào danh sách các trận động đất  earthquakes.add(\_quake);  // Notify the array adapter of a change.  // Thông báo mảng những sự thay đổi  aa.notifyDataSetChanged();  }  } |

16. Ghi đè xử lý onActivityResult để gọi updateFromPreferences và làm mới các trận động đất bất cứ khi nào các Hoạt động Preferences lưu những thay đổi.

|  |
| --- |
| [@Override](mailto:@Override)  public void onActivityResult(int requestCode, int resultCode, Intent data) {  super.onActivityResult(requestCode, resultCode, data);  if (requestCode == SHOW\_PREFERENCES)  if (resultCode == Activity.RESULT\_OK) { updateFromPreferences(); refreshEarthquakes();  }  } |

17. Cuối cùng, gọi updateFromPreferences trong onCreate (trước khi gọi đến refreshEarthquakes) để đảm bảo rằng các preferences được áp dụng khi các Hoạt động đầu tiên bắt đầu.

|  |
| --- |
| [@Override](mailto:@Override)  public void onCreate(Bundle icicle) { super.onCreate(icicle); setContentView(R.layout.main);  earthquakeListView = (ListView)this.findViewById(R.id.earthquakeListView);  earthquakeListView.setOnItemClickListener(new OnItemClickListener() {  public void onItemClick(AdapterView \_av, View \_v, int \_index, long arg3) {  selectedQuake = earthquakes.get(\_index);  showDialog(QUAKE\_DIALOG);  }  });  int layoutID = android.R.layout.simple\_list\_item\_1;  aa = new ArrayAdapter<Quake>(this, layoutID, earthquakes);  earthquakeListView.setAdapter(aa);  updateFromPreferences();  refreshEarthquakes();  } |

**Sử dụng SharedPreferences trong ứng dụng**

Trong bài tập này chúng em chỉ sử dụng sharedpreferences để lưu trữ mật khẩu ban đầu khi người dùng chạy ứng dụng cũng như lưu trữ việc người dùng chọn view theo list hay block khi dùng ứng dụng.

1. Phần sử dụng database adapter để kết nối cơ sở dữ liệu cũng như lấy dữ liệu lên thể hiện trên ứng dụng phần này là kết quả nghiên cứu của học viên Phan Bá Tiệp

**Giới thiệu Adapter**  
Adapters bản chất có thể hiểu là bộ chuyển đổi được dùng kết nối các lớp ràng buộc dữ liệu với thành phần giao diện Views.

Các bộ chuyển đổi có trách nhiệm tạo ra kết nối và chuyển giao từ dữ liệu sang các thành phần view trên giao diện ứng dụng.

Các thành phần điều khiển giao diện người dùng nế sử dụng nhờ sự hỗ trợ của adapter từ ​​lớp trừu tượng AdapterView.

Ngoài ra người dùng có thể tạo ra các điều khiển AdapterView của riêng mình để sử dụng cho các mục đích trình diễn dữ liệu khác.

Trong phần này sẽ giới thiệu một số Adapters trong Android cung cấp sẵn:

**ArrayAdapter**

Các ArrayAdapter là một lớp chung và sử dụng một mảng dữ liệu trả về.

Theo mặc định, các ArrayAdapter liên kết với các giá trị String của từng đối tượng phần thể hiện ví dụ một TextView, TextEdit, CheckBox …

Các nhà phát triển của android đã dựng sẵn những công cụ cho phép bạn sử dụng cho một giao diện phức tạp hơn chứa nhiều thành phần hiển thị dữ liệu hơn, hoặc bạn có thể mở rộng các lớp để sử dụng thay thế ImageView hoặc bố trí lồng nhau bằng cách ghi đè các phương thức getView trong adapter view  
**SimpleCursorAdapter**

Các SimpleCursorAdapter gắn Views với việc sử dụng con trỏ trả về từ Content Provider trong câu truy vấn. Bạn cần phải chỉ rõ một layout XML định nghĩa và sau đó gán giá trị trong vòng lặp cho mỗi cột trong kết quả trả về đó để bố trí thể hiện các thành phần giao diện

Để áp dụng một Adapter để một lớp AdapterView, bạn gọi phương thức của setAdapter, như thể hiện trong đoạn code dưới đây:

|  |
| --- |
| ArrayList <String> myStringArray = new ArrayList <String> (); ArrayAdapter <String> myAdapterInstance; int layoutID = android.R.layout.simple\_list\_item\_1; myAdapterInstance = new ArrayAdapter <String> (this, layoutID,) |

**Customizer Adapter**   
Bằng kỹ thuật sử dụng adaptor trong ứng dụng ghi chú của nhóm, chúng em xây dựng 2 adapter để thực hiện việc lấy dữ liệu và view theo 2 kiểu khác nhau đó là dạng list và dạng block, cả 2 adaptors này là customize adaptor kế thừa từ base adaptor có sẵn trong android

Đó là NoteAdaptor và NoteListAdaptor

1. Phần sử dụng hệ quản trị cơ sở dữ liệu đơn giản SQLite cho Android , kết quả nghiên cứu của học viên TrầnThị Như Hoa

**Giới thiệu SQLite**

SQLite là một hệ thống quản lý cơ sở dữ liệu quan hệ (RDBMS). Nó cũng được biết đến bởi:  
Nguồn mở

Tuân thủ các tiêu chuẩn về CSDL

Nhẹ, tiện dụng

Đơn tầng, dễ quản lý  
 Nó đã được viết như một thư viện bằng ngôn ngữ C nhỏ gọn. Và được bao gồm như là một phần của bộ phần mềm Android.  
Bằng cách cung cấp chức năng thông qua thư viện, mỗi cơ sở dữ liệu sẽ trở thành một phần tích hợp của các ứng dụng đã tạo khởi tạo và sử dụng cơ sở dữ liệu đó. Điều này làm giảm phụ thuộc bên ngoài, giảm thiểu độ trễ, giảm các khóa trong phiên giao dịch và đơn giản hóa vấn đề đồng bộ.  
SQLite có tiếng là cực kỳ đáng tin cậy và là hệ thống cơ sở dữ liệu của nhiều thiết bị điện tử như máy nghe nhạc MP3, iPhone, và iPod Touch.  
**Nhẹ và mạnh mẽ**

SQLite khác với nhiều cơ sở dữ liệu thông thường, bằng cách sử dụng một cách tiếp cận đơn giản linh hoạt. Thay vì đòi hỏi các giá trị trong cột dữa liệu để phù hợp với một loại dữ liệu duy nhất, cũng như các giá trị trong mỗi hàng cho mỗi cột được đánh chỉ số riêng.

Kết quả là, không cần có sự kiểm tra nghiêm ngặt khi nhập/xuất giá trị từ mỗi cột trong khi truy vấn.  
Để có được thông tin toàn diện hơn về SQLite, bao gồm cả những điểm mạnh và hạn chế bạn có thể tham khảo tại www.sqlite.org/.  
**Các thành phần chính của sqlite**

Các thành phần chính của SQLite hay được sử dụng là Cursors và ContentValues

Cursor, các kết quả truy vấn sqlite được trả về đối tượng cusor. Thay vì giải nén và trả về một bản sao của các giá trị kết quả, Cursors hoạt động như con trỏ đến một tập hợp con của các dữ liệu cơ bản.

Con trỏ là một cách quản lý, kiểm soát vị trí hiện thời theo dòng dữ liệu tại thời điểm thực thi câu truy vấn.

Lớp Cursor thiết kế sẵn bao gồm một số chức năng để điều hướng kết quả truy vấn và thông qua những hàm sau đây:  
- moveToFirst: Di chuyển con trỏ đến dòng đầu tiên trong kết quả truy vấn.

- moveToNext: Di chuyển con trỏ đến dòng kế tiếp.

- moveToPrevious: Di chuyển con trỏ đến dòng trước đó.

- getCount: Trả về số hàng trong tập kết quả.

- getColumnIndexOrThrow: Trả về một số cho các cột với tên xác định (có tung ra một ngoại lệ nếu không cột không tồn tại).

- getColumnName:Trả về tên của chỉ số cột xác định.

- getColumnNames: Trả về một mảng String của tất cả các tên cột trong con trỏ hiện hành.

- moveToPosition: Di chuyển con trỏ đến dòng xác định.

- getPosition: Trả về vị trí con trỏ hiện hành.  
Android cung cấp một cơ chế để quản lý tài nguyên Cursor trong mọi activity. Phương pháp tích hợp startManagingCursor trong vòng đời của Cursor

Khi bạn đã kết thúc với các Cursor, gọi hàm stopManagingCursor kết thúc

**Làm việc với cơ sở dữ liệu Android** **Sử dụng SQLiteOpenHelper**

Trong android chúng ta cung cũng có thể sử dụng class helper để đơn giản hóa các tương tác cơ sở dữ liệu.

SQLiteOpenHelper là một lớp trừu tượng mà phục vụ tốt cho việc thao tác trên cơ sở dữ liệu Khi sử dụng SQLiteOpenHelper, bạn không cần quan tâm đến các logic được sử dụng bên trong để kiểm soát việc đóng mở kết nối cũng như khởi tạo cơ sở dữ liệu.  
Để mở rộng các lớp SQLiteOpenHelper chúng ta có thể ghi đè các hàm dựng sẵn như, onCreate, và onUpgrade để xử lý việc tạo ra một cơ sở dữ liệu mới và cập nhật dữ liệu.

Bạn cũng có thể sử dụng CursorFactory và gọi hàm getReadableDatabase hoặc getWriteableDatabase để mở và trả về một kết nối cơ sở dữ liệu có thể thể đọc / ghi được

Trong trường hợp sử dụng getWriteableDatabase lư ý có thể có lỗi xẩy ra vì không gian lưu trữ hoặc vấn đề phân quyền.

Ta có thể sử dụng như đoạn code ví dụ dưới đây:

|  |
| --- |
| dbHelper = new myDbHelper (context, database\_name, null, DATABASE\_VERSION); SQLiteDatabase db; try {db = dbHelper.getWritableDatabase (); } Catch (SQLiteException ex) {db = dbHelper.getReadableDatabase (); } |

Trong trường hợp nếu các cơ sở dữ liệu không tồn tại, dbHelper sẽ thực hiện xử lý onCreate để tạo ra cơ sở dữ liệu

Thiết lập một cơ sở dữ liệu là một quá trình hai bước.

Đầu tiên, gọi openOrCreateDatabase để tạo ra cơ sở dữ liệu mới.

Sau đó, gọi execSQL trên dụ cơ sở dữ liệu kết quả để chạy các lệnh SQL mà sẽ tạo ra các bảng mối quan hệ dữ liệu giữa các bảng.

Quá trình này được thể hiện trong đoạn code dưới đây:

|  |
| --- |
| private static string database\_name thức = "myDatabase.db"; private static String cuối cùng DATABASE\_TABLE = "mainTable"; private static thức chuỗi DATABASE\_CREATE = "tạo bảng" + DATABASE\_TABLE + "(\_id số nguyên autoincrement khóa chính," + "column\_one văn bản không null);"; SQLiteDatabase myDatabase; private void CreateDatabase () {myDatabase = openOrCreateDatabase (database\_name, Context.MODE\_PRIVATE, null); myDatabase.execSQL (DATABASE\_CREATE); } |

**Truy vấn cơ sở dữ liệu**: Tất cả các truy vấn cơ sở dữ liệu được trả về một đối tượng con trỏ đến một tập kết quả. Điều này cho phép Android quản lý tài nguyên tốt hơn bằng cách lấy và giải phóng bộ nhớ theo yêu cầu.  
Để thực hiện một truy vấn trên một cơ sở dữ liệu, sử dụng các phương pháp truy vấn vào đối tượng cơ sở dữ liệu sử dụng đoạn code mẫu dưới đây:

|  |
| --- |
| // Return all rows for columns one and three, no duplicates  String[] result\_columns = new String[] {KEY\_ID, KEY\_COL1, KEY\_COL3};  Cursor allRows = myDatabase.query(true, DATABASE\_TABLE, result\_columns,  null, null, null, null, null, null);  // Return all columns for rows where column 3 equals a set value  // and the rows are ordered by column 5.  String where = KEY\_COL3 + “=” + requiredValue;  String order = KEY\_COL5;  Cursor myResult = myDatabase.query(DATABASE\_TABLE, null, where,  null, null, null, order); |

**Kết quả trả về từ Cursor**

Để lấy các giá trị thực tế từ một Cursor kết quả, đầu tiên sử dụng các phương pháp moveTo <location> mô tả ở phần trước để định vị con trỏ ở chính xác hàng của của kết quả trả về.  
Khi Cursor đang ở hàng mong muốn, sử dụng các phương thức get với kiểu dữ liệu tương ứng trong các cột để trả về giá trị được lưu trữ ta sử dụng đoạn mã sau:

|  |
| --- |
| String columnValue = myResult.getString (columnIndex); |

**Chèn dòng mới**

Để tạo một hàng mới, ta cần xây dựng một đối tượng ContentValues, và sử dụng phương thức set của nó để cung cấp giá trị cho mỗi cột thể hiện trong đoạn dưới đây:

|  |
| --- |
| // Create a new row of values to insert.  ContentValues newValues = new ContentValues();  // Assign values for each row.  newValues.put(COLUMN\_NAME, newValue);  [ ... Repeat for each column ... ]  // Insert the row into your table  myDatabase.insert(DATABASE\_TABLE, null, newValues); |

**Cập nhật một Row trên hàng cơ sở dữ liệu**

cập nhật dữ liệu cũng được thực hiện bằng cách sử dụng ContentValues trong đoạn mã dưới đây:

|  |
| --- |
| // Define the updated row content.  ContentValues updatedValues = new ContentValues();  // Assign values for each row.  updatedValues.put(COLUMN\_NAME, newValue);  [ ... Repeat for each column ... ]  String where = KEY\_ID + “=” + rowId;  // Update the row with the specified index with the new values.  myDatabase.update(DATABASE\_TABLE, updatedValues, where, null); |

**Xóa Rows**

Để xóa một hàng, chỉ cần gọi xóa trên đối tượng cơ sở dữ liệu của bạn, xác định tên bảng và điểu kiện tìm ra hàng bạn muốn xóa, thể hiện trong đoạn code dưới đây:

|  |
| --- |
| myDatabase.delete(DATABASE\_TABLE, KEY\_ID + “=” + rowId, null); |

**Sử dụng sqlite trong ứng dụng**

Trong phần bài tập này chúng em sử dụng sqlite ở mức đơn giản thực thi các câu query và đối tượng Cursor cũng như sử dụng ContentValuies để update, delete row dữ liệu

1. Phần xử lý nhắc lịch và truy vấn địa điểm thể hiện lên trên ứng dụng, là kết quả nghiên cứu của học viên Phạm Tuân

Đối với mọi ứng dụng cho tiết bị Android, việc sử dụng dịch vụ google play service có sẵn có thể giúp lập trình viên truy cập vào được các dịch vụ địa điểm có sẵn trên máy ngoài ra với lớp Geocoder, bạn có thể chuyển đổi qua lại giữa vĩ độ / kinh độ giá trị và địa chỉ thực tế.  
   
Dưới đây là các ứng dụng thực tế mà bạn có thể thực hiện được khi sử dụng dịch vụ địa điểm của thiết bị kết hợp với dịch vụ bản đồ của google:  
- Thiết lập giả lập của bạn để kiểm tra các dịch vụ dựa trên địa điểm.  
- Tìm và theo dõi vị trí thiết bị.  
- Tạo cảnh báo gần.  
- Rẽ vị trí địa lý vào các địa chỉ đường phố và ngược lại.  
- Tạo và tùy chỉnh bản đồ dựa trên hoạt động bằng cách sử dụng MapView và MapActivity.  
- Thêm Overlays để bản đồ của bạn.  
**Sử dụng dịch vụ địa điểm**  
Dịch vụ dựa trên địa điểm (LBS) là một thuật ngữ chung dùng để mô tả các công nghệ khác nhau được sử dụng để tìm vị trí hiện tại của thiết bị.

Hai yếu tố LBS chính là:  
- LocationManager: Cung cấp móc để các dịch vụ dựa trên địa điểm.  
- LocationProviders: Mỗi thiết bị khác nhau, công nghệ khác nhau được sử dụng để  
xác định vị trí hiện tại của thiết bị.

Với LocationManager bạn có thể:  
- Có được vị trí hiện tại của bạn.  
- Theo dõi di chuyển

- Thiết lập cảnh báo khoảng cách để phát hiện chuyển động vào và ra khỏi khu vực đã định nghĩa

Tùy thuộc vào các thiết bị, có thể có một số công nghệ mà Android có thể sử dụng để xác định vị trí hiện tại.

Mỗi công nghệ, hoặc nhà cung cấp dịch vụ địa điểm, sẽ cung cấp các khả năng khác nhau cũng như thông tin điện năng tiêu thụ, chi phí, độ chính xác của thông tin địa điểm trả về.

Để có được thông tin của một nhà cung cấp dịch vụ địa điểm ta gọi hàm getProvider như code bên dưới:

|  |
| --- |
| String providerName = LocationManager.GPS\_PROVIDER;  LocationProvider gpsProvider;  gpsProvider = locationManager.getProvider(providerName); |

**Việc tìm kiếm các nhà cung cấp có sẵn trên thiết bị**  
Lớp LocationManager chứa các hằng chuỗi tĩnh mà trả lại thông tin tên nhà cung cấp dịch vụ địa điểm thường là:  
- LocationManager.GPS\_PROVIDER  
- LocationManager.NETWORK\_PROVIDER  
Để có được một danh sách các tên cho tất cả các nhà cung cấp có sẵn trên thiết bị, chúng ta có thể gọi hàm getProviders theo code mẫu dưới đây:

|  |
| --- |
| boolean enabledOnly = true;  List<String> providers = locationManager.getProviders(enabledOnly); |

**Việc tìm kiếm các nhà cung cấp dựa trên tiêu chí yêu cầu**  
Trong hầu hết các tình huống, bạn không chắc rằng bạn sẽ muốn chọn một cách rõ ràng các nhà cung cấp địa điểm để sử dụng. Ta có thể tìm các nhà cung cấp được phép theo một tiêu chí liên quan.

|  |
| --- |
| Criteria criteria = new Criteria();  criteria.setAccuracy(Criteria.ACCURACY\_COARSE);  criteria.setPowerRequirement(Criteria.POWER\_LOW);  criteria.setAltitudeRequired(false);  criteria.setBearingRequired(false);  criteria.setSpeedRequired(false);  criteria.setCostAllowed(true); |

**Bạn có thể sử dụng getBestProvider để trả lại kết hợp tốt nhất**  
Đoạn code sau đây gọi hàm getBestProvider để trả lại nhà cung cấp tốt nhất cho các tiêu chí yêu cầu của bạn:

|  |
| --- |
| String bestProvider = locationManager.getBestProvider(criteria, true); |

Để xem danh sách các tên cho tất cả các nhà cung cấp phù hợp với tiêu chí của bạn, bạn có thể sử dụng hàm getProviders và duyệt qua các nhà cung cấp dịch vụ địa điểm để lấy ra.

Nếu không tìm thấy, cuộc gọi này trả về null.

|  |
| --- |
| List<String> matchingProviders = locationManager.getProviders(criteria,  false); |

**Quản lý địa điểm**,

Yêu cầu một dịch vụ địa điểm sử dụng đối tượng LOCATION\_SERVICE ta gọi hàm getSystemService thể hiện trong đoạn code sau đây:

|  |
| --- |
| String serviceString = Context.LOCATION\_SERVICE;  LocationManager locationManager;  locationManager = (LocationManager)getSystemService(serviceString); |

Trước khi bạn có thể sử dụng quản lý địa điểm, bạn cần phải thêm một hoặc nhiều thẻ cấp phép của bạn trong file manifest để hỗ trợ truy cập vào các phần cứng LBS.

|  |
| --- |
| <uses-permission android:name=”android.permission.ACCESS\_FINE\_LOCATION”/>  <uses-permission android:name=”android.permission.ACCESS\_COARSE\_LOCATION”/> |

Bạn có thể tìm vị trí cuối cùng được xác định bởi một nhà cung cấp vị trí cụ thể bằng cách sử dụng hàm getLastKnownLocation, như đoạn code sau đây:

|  |
| --- |
| String provider = LocationManager.GPS\_PROVIDER;  Location location = locationManager.getLastKnownLocation(provider); |

Lưu ý rằng getLastKnownLocation không yêu cầu các nhà cung cấp địa điểm cập nhật vị trí hiện tại.

Nếu thiết bị đã không cập nhật vị trí hiện tại thì giá trị này có là giá trị cũ của lần cập nhật gần nhất.

**Sử dụng quản lý địa điểm trong ứng dụng**

Trong phần bài tập thực hành chúng em có sử dụng đến phần lấy địa điểm được biết cuối cùng thông qua dịch vụ GPS và kết hợp với dữ liệu của bản đồ Google Map để trả về tên địa điểm

# Thiết kế ứng dụng

1. **Cấu hình thiết bị yêu cầu**

* Android 4.0 trở lên
* Có Internet và GPS để sử dụng dịch vụ định vị
* Chi phép truy cập file, internet, gps, camera,alarm

1. **Chức năng**
2. Đăng nhập
3. Màn hình chính hiển thị các ghi chú đã có theo khối, theo danh sách
4. Thêm mới ghi chú

* Thêm ảnh vào ghi chú
* Chọn ngày, giờ để ghi chú
* Thêm địa điểm dùng định vị
* Nhắc lịch khi ghi chú sắp đến hạn

1. Sửa ghi chú
2. Xóa ghi chú
3. Tìm kiếm ghi chú
4. **Thiết kế giao diện**

* Ứng dụng sử dụng 4 activites chính và các dialog

1. Màn hình đăng nhập

|  |
| --- |
|  |
| LoginActivity |
| Màn hình form đăng nhập khi người dùng đã lưu mật khẩu sẽ hiện login form |

1. Màn hình chính view danh sách ghi chú theo khối

|  |
| --- |
|  |
| HomeActivity |
| Màn hình danh sách ghi chú theo dạng khối hiện lên sau khi người dùng đăng nhập thành công |

1. Màn hình chính hiện danh sách theo danh mục

|  |
| --- |
|  |
| HomeActivity |
| Màn hình danh sách ghi chú theo dạng danh sách |

1. Màn hình hiện chi tiết 1 ghi chú

|  |
| --- |
|  |
| NoteDetail activity |
| Chi tiết 1 ghi chú có các trường thông tin và ảnh ghi chú đã lưu |

1. Màn hình chỉnh sửa/tạo mới ghi chú

|  |
| --- |
|  |
| NoteEdit activity |
| Màn hình chỉnh sửa hoặc tạo mới 1 ghi chú, có các nút tương tác, chụp ảnh, chọn ngày, chọn địa điểm và chọn thời gian nhắc lịch |

1. Màn hình chỉnh sửa ghi chú chọn ngày và địa điểm

|  |
| --- |
|  |
| NoteEdit activity |
| Màn hình chỉnh sửa ghi chú vơi lựa chọn nhắc lịch và chọn địa điểm |

# Phân công công việc

|  |  |  |
| --- | --- | --- |
| Phạm Thị Hải Vân | Màn hình login, quản lý tài khoản trên SharePreferences  Các file sẽ xem  LoginActivity.java  Activity\_login.xml | Sách tham khảo  2009, Reto Meier, Professional\_android\_application\_development, Wiley  Trang 160 -> 173 |
| Trần Thị Như Hoa | Màn hình chính, quản lý danh sách ghi chú dùng truy vấn SQL Lite lấy ra các dữ liệu đã ghi thể hiện lên trên màn hình chính  HomeActivity.java  Activity\_home.xml | Sách tham khảo  2009, Reto Meier, Professional\_android\_application\_development, Wiley  Trang 176 -> 189 |
| Phan Bá Tiệp | Màn hình giao diện chính sử dụng adapters tự tạo để thiết kế danh sách ghi chú NoteAdapter.java  NoteListAdapter.java  Activity\_display\_block.xml  Activity\_display\_list.xml | Sách tham khảo  2009, Reto Meier, Professional\_android\_application\_development, Wiley  Trang 174 -> 175 |
| Phạm Tuân | Màn hình hình sửa chữa/thêm mới ghi chú, insert update SQL Lite xử lý lưu trữ ảnh chụp, lấy thông tin địa điểm  NoteEdit.java  NoteDetail.java  Activity\_display\_detail.xml  Activity\_display\_edit.xml | Sách tham khảo  2009, Reto Meier, Professional\_android\_application\_development, Wiley  Trang 207 -> 215 |