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

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
* Par of event handle và activity state

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

Với SharedPreference là nơi lưu trữ nội tại trong nơi chưa ứng dụng, và có thể dùng để chia sẻ thông tin lưu trữ với ứng dụng khác, về bản chất là hệ điều hành sẽ tạo ra 1 file xml để lưu trữ

Với cách lưu trữ này, thì chỉ sử dụng để lưu trữ các kiểu dữ liệu cơ bản như: số nguyên, số thực, kiểu giá trị logic đúng/sai

Các lưu trữ này cũng được dùng để lữu trữ thông tin session của người dùng hiện thời, trạng thái của ứng dụng hiện tại hay là các cấu hình cài đặt ứng dụng để chia sẻ với các ứng dụng khác và các thành phần trong ứng dụng hiện tại

Điểm lưu ý là đối với mỗi activity thì hàm onSaveInstanceState là hàm để sử dụng lưu trữ thông tin trạng thái của 1 màn hình hiện thời

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 Một dapters  
Adapters được dùng kết nối các lớp ràng buộc dữ liệu để sử dụng trên giao diện Views. Các bộ chuyển đổi có trách nhiệm tạo ra kết nối chuyển giao từ dữ liệu sang các thành phần view trên giao diện  
các lớp con sử dụng để đại diện cho từng hạng mục và cung cấp quyền truy cập vào các dữ liệu cơ bản.  
Điều khiển giao diện người dùng hỗ trợ adapter ràng buộc phải dẫn xuất từ ​​lớp trừu tượng AdapterView. Nó là  
có thể tạo ra các điều khiển AdapterView nguồn gốc của riêng bạn và tạo các lớp Adapter để ràng buộc họ.  
Giới thiệu Một số Adapters Android-Cung cấp  
Trong nhiều trường hợp, bạn sẽ không phải để tạo ra adapter riêng bạn từ đầu. Android cung cấp một tập hợp các  
Adapters mà bơm dữ liệu vào các vật dụng sử dụng giao diện bản địa.  
Bởi vì Adapters chịu trách nhiệm cả hai để cung cấp các dữ liệu và lựa chọn đại diện cho mỗi lượt xem  
mục, Adaptors hoàn toàn có thể thay đổi sự xuất hiện và chức năng của các điều khiển họ đang bị ràng buộc để.  
Danh sách sau đây nhấn mạnh hai của bộ điều hợp nguồn gốc hữu dụng và đa năng:  
❑ ArrayAdapter Các ArrayAdapter là một lớp chung mà gắn bó với adapter đọc tới một mảng của  
đối tượng. Theo mặc định, các ArrayAdapter liên kết với các giá trị toString của từng đối tượng để một TextView  
Defi Kiểm định nghĩa trong một bố cục. Nhà xây dựng thay thế cho phép bạn sử dụng bố trí phức tạp hơn,  
hoặc bạn có thể mở rộng các lớp để sử dụng thay thế cho các văn bản Xem (như Populating một  
ImageView hoặc bố trí lồng nhau) bằng cách ghi đè các phương pháp getView.  
❑ SimpleCursorAdapter Các SimpleCursorAdapter gắn Views để con trỏ trở về từ  
Content Provider câu truy vấn. Bạn chỉ rõ một layout XML Defi Định nghĩa và sau đó ràng buộc giá trị trong vòng lặp  
mỗi cột trong kết quả thiết lập, đến một Xem trong cách bố trí đó.  
Các phần sau sẽ đi sâu vào các lớp adapter chi tiết hơn. Các ví dụ được cung cấp  
dữ liệu ràng buộc vào Danh mục Views, mặc dù cùng một logic sẽ làm việc tốt cho các lớp học AdapterView khác  
như Spinners và Phòng Trưng Bày.  
Sử dụng adapter cho Data Binding  
Để áp dụng một Adapter để một lớp AdapterView nguồn gốc, bạn gọi phương thức của Xem setAdapter, đi qua  
trong một thể Adapter, như thể hiện trong đoạn 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,)

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

1. Phần sử dụng hệ quản trị cơ sở dữ liệu đơn giản SQLite cho Android , kết quả nghiên cứu của học viên Trần Như Hoa
2. 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

# Thiết kế ứng dụng

1. **Chức năng**

Đăng nhập

Màn hình chính hiển thị các ghi chú đã có theo khối, theo danh sách

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

Sửa ghi chú

Xóa ghi chú

Tìm kiếm ghi chú

Nhắc lịch khi ghi chú đến hạn

Nhắc lịch khi đến hạn và trong phạm vi địa điểm

1. **Thiết kế giao diện**

* Ứng dụng sử dụng 4 activites chính và các dialog

1. Màn hình đăng nhập

|  |
| --- |
|  |
| LoginActivity |
| Đoạn mã dưới đây miêu tả chức năng kiểm tra đăng nhập, dữ liệu ghi thông tin mật khẩu sẽ được lấy ở shared prefrences |
| @Override  **protected** Boolean doInBackground(Void... params) {  **try** {  Thread.*sleep*(2000);  } **catch** (InterruptedException e) {  **return** **false**;  }  SharedPreferences settings = getSharedPreferences(***PREFS\_NAME***, Context.***MODE\_PRIVATE***);  String savedPWD = settings.getString(***PREFS\_KEY***, **null**);  **return** savedPWD.equals(mPassword);  } |

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

|  |
| --- |
|  |
| HomeActivity |
| Đoạn code ví dụ dưới đây sẽ load dữ liệu từ cơ sở dữ liệu vào trang home sử dụng Adapter |
| list = (ListView) findViewById(R.id.***listView***);  list.setAdapter(**new** NoteAdapter(**this**, getData())); |

1. Màn hình chính hiện danh sách theo danh mục

|  |
| --- |
|  |
| HomeActivity |
| Đoạn code dưới đây mô tả việc thay đổi view từ khối sang danh sách |
| **case** R.id.***btnHswitch***: {  Toast.*makeText*(**this**, "Change view", Toast.***LENGTH\_SHORT***).show();  //setContentView(R.layout.activity\_home);  list = (ListView) findViewById(R.id.***listView***);  **if** (list.getAdapter() **instanceof** NoteAdapter) {  list.setAdapter(**new** NoteListAdapter(**this**, getData()));    } **else** {  list.setAdapter(**new** NoteAdapter(**this**, getData()));  }    }  **break**; |

1. Màn hình hiện chi tiết 1 ghi chú

|  |
| --- |
|  |
| NoteDetail activity |
| Đoạn code dưới đây minh họa việc gán giá trị vào các trường trên giao diện xem chi tiết ghi chú |
| what=(TextView) findViewById(R.id.***txtTitle***);  when=(TextView) findViewById(R.id.***textwhen***);  reminder = (RadioButton) findViewById(R.id.***remind***);  where=(TextView) findViewById(R.id.***textWhere***);  when.setText(n\_.getFormatedDate());  what.setText(n\_.title);  where.setText(n\_.getAddress());  reminder.setChecked(Boolean.*parseBoolean*(n\_.remind)); |

1. Màn hình chỉnh sửa/tạo mới ghi chú

|  |
| --- |
|  |
| NoteEdit activity |
| Đoạn code dưới đây miêu tả việc load form edit/tạo mới khi click vào các button tương ứng |
| } **case** R.id.***btnEdit***: {  Intent i = **new** Intent(v.getContext(), NoteEdit.**class**);  i.putExtra("note", n\_);  i.putExtra("type", NoteEdit.*ACTION\_TYPE\_ADDNEW*);  **this**.fileList();  v.getContext().startActivity(i);  }  **break**;  **case** R.id.***btnAddnew***: {  Intent i = **new** Intent(v.getContext(), NoteEdit.**class**);  i.putExtra("type", NoteEdit.*ACTION\_TYPE\_ADDNEW*);  **this**.fileList();  v.getContext().startActivity(i);  isEdit = **false**;  current\_action = *ACTION\_TYPE\_ADDNEW*;  }  **break**; |

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

|  |
| --- |
|  |
| NoteEdit activity |
| Đoạn code dưới đây miêu tả việc gắn sự kiện và xử lý phần chọn ngày tháng |
| **switch** (v.getId()) {  **case** R.id.***date***: {  DatePickerDialog dialog = **new** DatePickerDialog(v.getContext(), **this**, 2015, 11,27);  dialog.show();  }  **break**; |
| Đoạn code dưới đây miêu tả việc gắn sự khiển và xử lý chọn địa điểm dùng bản đổ |
| **case** R.id.***loc***: {  String uri= "geo:47.6,-122.3";  Intent intent = **new** Intent(android.content.Intent.***ACTION\_VIEW***, Uri.*parse*(uri));  **if** (isAppInstalled("com.google.android.apps.maps")) {  intent.setClassName("com.google.android.apps.maps", "com.google.android.maps.MapsActivity");  startActivity(intent);  }  }  **break**; |

Phần lưu trữ db

|  |
| --- |
| Đoạn code dưới đây mô tả khởi tạo cơ sở dữ liệu SQL Lite |
|  |
| Đoạn code dưới đây mô tả cập nhật cơ sở dữ liệu SQL Lite |
|  |

# 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  Vân đọc/ dịch phần shared preferences nhé |
| Trần Thị Như Hoa | Màn hình chính, quản lý danh sách ghi chú dùng truy vấn SQL Lite  HomeActivity.java  Activity\_home.xml  Activity\_display\_block.xml  Activity\_display\_list.xml | Sách tham khảo  2009, Reto Meier, Professional\_android\_application\_development, Wiley  Trang 176 -> 189  Chị hoa đọc và dịch phần SQL lite |
| Phan Bá Tiệp | Màn hình sửa chữa/thêm mới ghi chú, insert update SQL Lite  NoteEdit.java  NoteDetail.java  Activity\_display\_detail.xml  Activity\_display\_edit.xml | Sách tham khảo  2009, Reto Meier, Professional\_android\_application\_development, Wiley  Trang 174 -> 175  Tiệp check phần này nhé |
| Phạm Tuân | Thiết kế ứng dụng, xử lý lưu trữ ảnh chụp | Code dự án |

# Thực hiện chi tiết

1. Phạm Thị Hải Vân

|  |  |  |
| --- | --- | --- |
| Tên file | Mục đích sử dụng | Nội dung chi tiết |
| activity\_login.xml | Thiết kế giao diện của màn hình login | Có 2 phần giao diện  1 phần là để tạo mật khẩu  1 phần là để đăng nhập |
| LoginActivity.java | Xử lý phần login của người dùng | Kiểm tra shared preferences nếu chưa lưu mật khẩu sẽ hiện giao diện tạo mật khẩu  Nếu lưu mật khẩu rồi hiện login  Gồm có phần kiểm tra đã lưu mật khẩu chưa  Lưu mật khẩu và chuyển sang đăng nhập  Chọn đăng nhập, kiểm tra mật khẩu  Đúng thì load home\_activity  Không đúng nhập lại mật khẩu |
|  | | |
| **// Class cài đặt login chạy ngầm AsyncTask**  **public** **class** UserLoginTask **extends** AsyncTask<Void, Void, Boolean> {…..} | | |
| **// Thực hiện login chạy ngầm**  @Override  **protected** Boolean doInBackground(Void... params) {  **try** {  Thread.*sleep*(2000);  } **catch** (InterruptedException e) {  **return** **false**;  }  SharedPreferences settings = getSharedPreferences(***PREFS\_NAME***, Context.***MODE\_PRIVATE***);  String savedPWD = settings.getString(***PREFS\_KEY***, **null**);  **return** savedPWD.equals(mPassword);  }  @Override  **protected** **void** onCancelled() {  mAuthTask = **null**;  showProgress(**false**);  } | | |
| **// Thực hiện chuyển giao diện nếu login thành công**  @Override  **protected** **void** onPostExecute(**final** Boolean success) {  mAuthTask = **null**;  showProgress(**false**);  **if** (success) {  Intent i = **new** Intent(getBaseContext(), HomeActivity.**class**);  startActivity(i);  } **else** {  mPasswordView.setError(getString(R.string.***error\_incorrect\_password***));  mPasswordView.requestFocus();  }  }  } | | |
| **// Thực hiện kiểm tra mật khẩu**  **if** (!TextUtils.*isEmpty*(password) && !isPasswordValid(password)) {  mPasswordView.setError(getString(R.string.***error\_invalid\_password***));  focusView = mPasswordView;  cancel = **true**;  } | | |
| setContentView(R.layout.***activity\_login***);  **// Lấy thông tin sharedprefrence để kiểm tra xem người dùng đã lưu mật khẩu chưa**  SharedPreferences settings = getSharedPreferences(***PREFS\_NAME***, Context.***MODE\_PRIVATE***);  **// lấy form login**  mLoginFormView = findViewById(R.id.***login\_form***);  // lấy form có ảnh progess  mProgressView = findViewById(R.id.***login\_progress***);  createPass = findViewById(R.id.***createPass***);  **// Kiểm tra mật khẩu đã lưu nếu chưa lưu mật khẩu hiện form lưu mật khẩu**  **if** (settings.getString(***PREFS\_KEY***, **null**) == **null**) {….} | | |
| @Override  **public** **void** onClick(View v) {  SharedPreferences settings = getSharedPreferences(***PREFS\_NAME***, Context.***MODE\_PRIVATE***);  SharedPreferences.Editor editor = settings.edit();  String p1 = e1.getText().toString();  **// Mật khẩu không được để trống**  **if** (!TextUtils.*isEmpty*(p1) && !isPasswordValid(p1)) {  e1.setError(getString(R.string.***error\_invalid\_password***));  e1.requestFocus();  **return**;  }  String p2 = e2.getText().toString();  **// Ghõ lại mật khẩu không được để trống**  **if** (!TextUtils.*isEmpty*(p2) && !isPasswordValid(p2)) {  e2.setError(getString(R.string.***error\_invalid\_password***));  e2.requestFocus();  **return**;  } **else** **if** (!p1.equals(p2)) {  e2.setError(getString(R.string.***error\_invalid\_confirm***));  e2.requestFocus();  } **else** {  editor.putString(***PREFS\_KEY***, p1);  editor.commit();  Toast.*makeText*(v.getContext(), "password saved!", Toast.***LENGTH\_SHORT***).show();  Intent i = **new** Intent(v.getContext(), LoginActivity.**class**);  v.getContext().startActivity(i);  }  }  }); | | |
| } **else** {  **// Nếu đã lưu mật khẩu rồi thì hiện form đăng nhập**  createPass.setVisibility(View.***GONE***);  mLoginFormView.setVisibility(View.***VISIBLE***);  mPasswordView = (EditText) findViewById(R.id.***password***);  mPasswordView.setOnEditorActionListener(**new** TextView.OnEditorActionListener() {  @Override  **public** **boolean** onEditorAction(TextView textView, **int** id, KeyEvent keyEvent) {  **if** (id == R.id.***login*** || id == EditorInfo.***IME\_NULL***) {  **// Gọi hàm login**  attemptLogin();  **return** **true**;  }  **return** **false**;  }  });  Button mEmailSignInButton = (Button) findViewById(R.id.***sign\_in\_button***);  mEmailSignInButton.setOnClickListener(**new** OnClickListener() {  @Override  **public** **void** onClick(View view) {  attemptLogin();  }  });  } | | |
| **// Hàm thực hiện việc login và load ảnh quay (progressbar)**  @TargetApi(Build.VERSION\_CODES.***HONEYCOMB\_MR2***)  **public** **void** showProgress(**final** **boolean** show) {  **if** (Build.VERSION.***SDK\_INT*** >= Build.VERSION\_CODES.***HONEYCOMB\_MR2***) {  **int** shortAnimTime = getResources().getInteger(android.R.integer.***config\_shortAnimTime***);  mLoginFormView.setVisibility(show ? View.***GONE*** : View.***VISIBLE***);  mLoginFormView.animate().setDuration(shortAnimTime).alpha(show ? 0 : 1)  .setListener(**new** AnimatorListenerAdapter() {  @Override  **public** **void** onAnimationEnd(Animator animation) {  mLoginFormView.setVisibility(show ? View.***GONE*** : View.***VISIBLE***);  }  });  mProgressView.setVisibility(show ? View.***VISIBLE*** : View.***GONE***);  mProgressView.animate().setDuration(shortAnimTime).alpha(show ? 1 : 0)  .setListener(**new** AnimatorListenerAdapter() {  @Override  **public** **void** onAnimationEnd(Animator animation) {  mProgressView.setVisibility(show ? View.***VISIBLE*** : View.***GONE***);  }  });  } **else** {  mProgressView.setVisibility(show ? View.***VISIBLE*** : View.***GONE***);  mLoginFormView.setVisibility(show ? View.***GONE*** : View.***VISIBLE***);  }  } | | |

1. Trần Thị Như Hoa

|  |  |  |
| --- | --- | --- |
| Tên file | Mục đích sử dụng | Nội dung chi tiết |
| activity\_home.xml | Thiết kế giao diện của màn hình chính | Có giao diện trang chủ với 1 ô tìm kiếm  1 list view để hiện danh sách ghi chú  1 menu có các nút sự kiện ở đáy màn hình  Nút tạo mới, nút thay đổi view dạng list hoặc dạng block  Nút khóa máy (quay về màn hình đăng nhập) |
| HomeActivity.java | Xử lý màn hình chính với listview | Kiểm tra dữ liệu sql lite, nếu chưa có data thì tạo data  Kiểm tra dữ liệu sql lite nếu chưa có table thì tạo talble  Query hết dữ liệu đã có trong table và hiện lên list |
| **// Hàm load dữ liệu đã lưu trong cơ sở dữ liệu SQLite**  **private** List<Note> getData() {  SQLiteDatabase db = openOrCreateDatabase("note8db", ***MODE\_PRIVATE***, **null**);  db.execSQL(NoteEdit.*CREATE\_TABLE*);  Cursor cs = db.rawQuery("SELECT \* FROM note8tb", **null**);  *data\_* = **new** ArrayList<Note>();  **if** (cs.moveToFirst()) {  **do** {  **// Duyệt qua toàn bộ doanh sách, tạo đối tượng để hiện dữ liệu trong list view**  Note n = **new** Note(String.*valueOf*(cs.getInt(0)), cs.getString(1), cs.getString(2), cs.getString(3),  cs.getString(4), cs.getString(5), cs.getBlob(6), cs.getString(7));  *data\_*.add(n);  } **while** (cs.moveToNext());  }  db.close();  **return** *data\_*;  } | | |
| **// Xử lý sự kiện khi có touch vào các nút**  @Override  **public** **void** onClick(View v) {  **switch** (v.getId()) {  **case** R.id.***btnAddnew***: {  Intent i = **new** Intent(**this**, NoteEdit.**class**);  i.putExtra("type", NoteEdit.*ACTION\_TYPE\_ADDNEW*);  v.getContext().startActivity(i);  }  **break**;  **// Cập nhật sự kiện chuyển dạng view**  **case** R.id.***btnSwitch***: {  Toast.*makeText*(**this**, "Change view", Toast.***LENGTH\_SHORT***).show();  list = (ListView) findViewById(R.id.***listView***);  SharedPreferences.Editor editor = settings.edit();  **if** (list.getAdapter() **instanceof** NoteAdapter) {  list.setAdapter(**new** NoteListAdapter(**this**, getData()));  **// Lưu dạng view vào sharedpreference**  editor.putString(LoginActivity.***PREFS\_VIEW***, "list");  } **else** {  list.setAdapter(**new** NoteAdapter(**this**, getData()));  **// Lưu dạng view vào sharedpreference**  editor.putString(LoginActivity.***PREFS\_VIEW***, "block");  }  **// save dữ liệu vào sharedpreference**  editor.commit();  }  **break**;  **case** R.id.***btnLock***: {  Intent i = **new** Intent(**this**, LoginActivity.**class**);  v.getContext().startActivity(i);  }  **break**;  **default**:  **break**;  }  } | | |
| list = (ListView) findViewById(R.id.***listView***);  **// Kiểm tra kiểu view đã lưu lần trước**  settings = getSharedPreferences(LoginActivity.***PREFS\_NAME***, Context.***MODE\_PRIVATE***);  String viewType = settings.getString(LoginActivity.***PREFS\_VIEW***, **null**);  **// Cài đặt filter để search trên list view**  list.setTextFilterEnabled(**true**);  **if** (viewType == **null**) {  **// Nếu chưa lưu kiểu view lần nào thì chọn view theo kiểu khối**  list.setAdapter(**new** NoteAdapter(**this**, getData()));  } **else** {  **// Nếu đã lưu kiểu view lần nào thì chọn view tương ứng kiểu view đã lưu**  **if** ("list".equals(viewType)) {  **// view theo dạng list**  list.setAdapter(**new** NoteListAdapter(**this**, getData()));  } **else** {  **// view theo dạng khối**  list.setAdapter(**new** NoteAdapter(**this**, getData()));  }  } | | |
| **// Hàm filter để thực hiện search trên list view đã có**  @Override  **public** **boolean** onQueryTextChange(String key) {  **if** (TextUtils.*isEmpty*(key)) {  list.clearTextFilter();  } **else** {  list.setFilterText(key);  }  **return** **true**;  } | | |

1. Phan Bá Tiệp

|  |  |  |
| --- | --- | --- |
| Tên file | Mục đích sử dụng | Nội dung chi tiết |
| activity\_display\_block.xml |  | Giao điện khi hiển thị dạng khối |
| NoteAdapter.java |  |  |
|  |  |  |
| Activity\_display\_list.xml |  | Giao diện khi hiển thị dạng danh sách |
| NoteListAdapter.java |  |  |
|  |  |  |

1. Phạm Tuân

|  |  |  |
| --- | --- | --- |
| Tên file | Mục đích sử dụng | Nội dung chi tiết |
| Activity\_display\_edit.xml | Thiết kế giao diện của màn hình chính | Giao diện edit form  1 menu có các nút sự kiện ở đáy màn hình  Nút save, Nút home quay về màn hình chính  Nút khóa máy (quay về màn hình đăng nhập |
| NoteEdit.java | Xử lý màn hình chính với listview | Xử lý phần edit form |
| Activity\_display\_detail.xml |  | Giao diện show chi tiết 1 ghi chú  1 menu có các nút sự kiện ở đáy màn hình  Nút edit, Nút tạo mới, Nút home quay về màn hình chính  Nút khóa máy (quay về màn hình đăng nhập |
| NoteDetail.java |  | Xử lý phần display cài đặt các giá trị hiện ra trên display form |