

MỤC LỤC

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT.....	4
1. LỊCH SỬ ANDROID	4
2. DELVING VỚI MÁY ẢO DALVIK.....	5
3. KIẾN TRÚC CỦA ANDROID.....	5
3.1. Tầng ứng dụng	6
3.2. Application framework.....	6
3.3. Library	7
3.4. Android Runtime.....	7
3.5. Linux kernel.....	8
4. ANDROID EMULATOR	8
5. CÁC THÀNH PHẦN TRONG MỘT ANDROID PROJECT.....	9
5.1. AndroidManifest.xml	9
5.2. File R.java.....	10
6. CHU KỲ ỨNG DỤNG ANDROID	11
6.1. Chu kỳ sống thành phần	12
6.2. Activity Stack.....	12
6.3. Các trạng thái của chu kỳ sống	13
6.4. Chu kỳ sống của ứng dụng	13
6.5. Các sự kiện trong chu kỳ sống của ứng dụng.....	14
6.6. Thời gian sống của ứng dụng.....	14
6.7. Thời gian hiển thị của Activity	14
6.8. Các phương thức của chu kỳ sống	14
7. CÁC THÀNH PHẦN GIAO DIỆN TRONG ANDROID	16
7.1. View.....	16
7.2. ViewGroup.....	16
7.2.1. LinearLayout	16
7.2.2. FrameLayout	17
7.2.3. AbsoluteLayout	17
7.2.4. RetaliveLayout	18
7.2.5. TableLayout	18
7.3. Button	18
7.4. ImageButton.....	19
7.5. ImageView	20
7.6. ListView	20
7.7. TextView	21

7.8.	EditText	21
7.9.	CheckBox.....	22
7.10.	MenuOptions.....	22
7.11.	ContextMenu.....	24
7.12.	Quick Search Box.....	24
7.13.	Activity & Intend.....	25
7.13.1.	Activity.....	25
7.13.2.	Intent	26
8.	CONTENT PROVIDER VÀ URI.....	28
9.	BACKGROUND SERVICE	29
10.	TELEPHONY	32
11.	SQLITE	32
12.	ANDROID & WEBSERVICE	33
12.1.	Khái niệm Web service và SOAP	33
12.2.	Giới thiệu về XStream	34
12.3.	Thao tác với web service trong Android	36
CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG.....		38
13.	ĐẶC TẢ BÀI TOÁN.....	38
14.	ĐẶC TẢ CHỨC NĂNG.....	38
14.1.	Upload phone book.....	38
14.2.	Revert phone book.....	39
14.3.	Send contact	40
14.4.	Reveive contact	40
14.5.	Export contact tới SD Card.....	40
14.6.	Import contact từ SD Card.....	41
14.7.	Account Manager	41
14.7.1.	Login.....	41
14.7.2.	Change password.....	41
14.7.3.	Restore account	41
14.7.4.	Create account	42
15.	CÁC BIỂU ĐỒ HỆ THỐNG	42
15.1.	Biểu đồ Usecase	42
15.1.1.	Restore account	43
15.1.2.	Export Contact vào SD Card.....	43
15.1.3.	Import Contact từ SD Card	43
15.1.4.	Login	44
15.2.	Biểu đồ Class.....	44

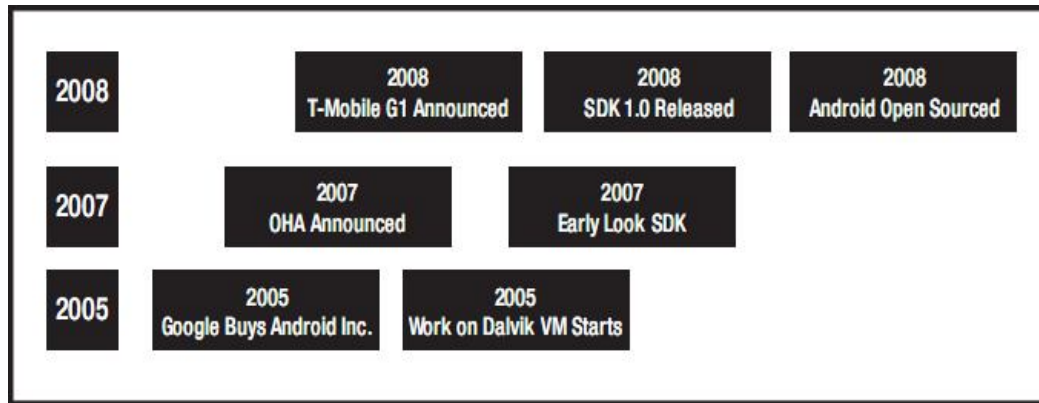
15.2.1.	Biểu đồ Class giữa các thành phần giao diện	44
15.3.	Biểu đồ Class Contact.....	44
15.4.	Biểu đồ Class ContactProvider	44
15.5.	Class Webservice	45
15.6.	Biểu đồ Sequence	45
15.6.1.	Login.....	45
15.6.2.	Create new account.....	46
15.6.3.	Restore account	47
15.6.4.	Upload phone book.....	48
15.6.5.	Revert phone book.....	48
15.6.6.	Export Contact to SD Card	49
15.6.7.	Import Contact from SD Card.....	49
15.6.8.	Send Contact	49
15.7.	Biểu đồ Activity	51
15.7.1.	Change password.....	51
15.7.2.	Export Contact to SD Card	52
15.7.3.	Upload phone book.....	52
15.7.4.	Send Contact	53
15.7.5.	Revert phone book.....	53
15.7.6.	Import Contact from SD Card.....	54
16.	CƠ SỞ DỮ LIỆU WEB SERVICE	54
CHƯƠNG 3: DEMO ỨNG DỤNG.....		57
17.	MỘT VÀI HÌNH ẢNH ỨNG DỤNG	57
18.	SƠ ĐỒ LIÊN KẾT CÁC MÀN HÌNH VÀ DIALOG	58
18.1.	Danh sách các liên kết màn hình mức 1&2	58
18.2.	Danh sách liên kết các màn hình mức 3	58
18.2.1.	Màn hình View Contact.....	58
18.2.2.	Màn hình danh sách các Contact mới.....	58
18.2.3.	Màn hình danh sách phone book đã upload.....	58
19.	ĐÁNH GIÁ ỨNG DỤNG	59
20.	HƯỚNG PHÁT TRIỂN TRONG TƯƠNG LAI.....	59
21.	KẾT LUẬN.....	60

ĐÁNH GIÁ CỦA GIÁO VIÊN
TÀI LIỆU THAM KHẢO

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

1. LỊCH SỬ ANDROID

Ban đầu, Android là hệ điều hành cho các thiết bị cầm tay dựa trên lõi Linux do công ty Android Inc. (California, Mỹ) thiết kế. Công ty này sau đó được Google mua lại vào năm 2005 và bắt đầu xây dựng Android Platform. Các thành viên chủ chốt tại Android Inc. gồm có: Andy Rubin, Rich Miner, Nick Sears, and Chris White.



Hình 1-1 Android timeline

Và sau tiếp, vào cuối năm 2007, thuộc về Liên minh Thiết bị Cầm tay Mã Nguồn mở (Open Handset Alliance) gồm các thành viên nổi bật trong ngành viễn thông và thiết bị cầm tay như:

Texas Instruments, Broadcom Corporation, Google, HTC, Intel, LG, Marvell Technology Group, Motorola, Nvidia, Qualcomm, Samsung Electronics, Sprint Nextel, T-Mobile, ARM Holdings, Atheros Communications, Asustek Computer Inc, Garmin Ltd, Softbank, Sony Ericsson, Toshiba Corp, and Vodafone Group,...

Mục tiêu của Liên minh này là nhanh chóng đổi mới để đáp ứng tốt hơn cho nhu cầu người tiêu dùng và kết quả đầu tiên của nó chính là nền tảng Android. Android được thiết kế để phục vụ nhu cầu của các nhà sản xuất thiết, các nhà khai thác và các lập trình viên thiết bị cầm tay.

Phiên bản SDK lần đầu tiên phát hành vào tháng 11 năm 2007, hãng T-Mobile cũng công bố chiếc điện thoại Android đầu tiên đó là chiếc T-Mobile G1, chiếc smartphone đầu tiên dựa trên nền tảng Android. Một vài ngày sau đó, Google lại tiếp tục công bố sự ra mắt phiên bản Android SDK release Candidate 1.0. Trong tháng 10 năm 2008, Google được cấp giấy phép mã nguồn mở cho Android Platform.

Khi Android được phát hành thì một trong số các mục tiêu trong kiến trúc của nó là cho phép các ứng dụng có thể tương tác được với nhau và có thể sử dụng lại các thành phần từ những ứng dụng khác. Việc tái sử dụng không chỉ được áp dụng cho

cho các dịch vụ mà nó còn được áp dụng cho cả các thành phần dữ liệu và giao diện người dùng.

Vào cuối năm 2008, Google cho phát hành một thiết bị cầm tay được gọi là Android Dev Phone 1 có thể chạy được các ứng dụng Android mà không bị ràng buộc vào các nhà cung cấp mạng điện thoại di động. Mục tiêu của thiết bị này là cho phép các nhà phát triển thực hiện các cuộc thí nghiệm trên một thiết bị thực có thể chạy hệ điều hành Android mà không phải ký một bản hợp đồng nào. Vào khoảng cùng thời gian đó thì Google cũng cho phát hành một phiên bản vá lỗi 1.1 của hệ điều hành này. Ở cả hai phiên bản 1.0 và 1.1 Android chưa hỗ trợ soft-keyboard mà đòi hỏi các thiết bị phải sử dụng bàn phím vật lý. Android cố định vấn đề này bằng cách phát hành SDK 1.5 vào tháng Tư năm 2009, cùng với một số tính năng khác. Chẳng hạn như nâng cao khả năng ghi âm truyền thông, vật dụng, và các live folder.

2. DELVING VỚI MÁY ẢO DALVIK

Dalvik là máy ảo giúp các ứng dụng java chạy được trên các thiết bị động Android. Nó chạy các ứng dụng đã được chuyển đổi thành một file thực thi Dalvik (dex). Định dạng phù hợp cho các hệ thống mà thường bị hạn chế về bộ nhớ và tốc độ xử lý. Dalvik đã được thiết kế và viết bởi Dan Bornstein, người đã đặt tên cho nó sau khi đến thăm một ngôi làng đánh cá nhỏ có tên là Dalvík ở đảo Eyjafjörður, nơi mà một số tổ tiên của ông sinh sống.

Từ góc nhìn của một nhà phát triển, Dalvik trông giống như máy ảo Java (Java Virtual Machine) nhưng thực tế thì hoàn toàn khác. Khi nhà phát triển viết một ứng dụng dành cho Android, anh ta thực hiện các đoạn mã trong môi trường Java. Sau đó, nó sẽ được biên dịch sang các bytecode của Java, tuy nhiên để thực thi được ứng dụng này trên Android thì nhà phát triển phải thực thi một công cụ có tên là dx. Đây là công cụ dùng để chuyển đổi bytecode sang một dạng gọi là dex bytecode. "Dex" là từ viết tắt của "Dalvik executable" đóng vai trò như cơ chế ảo thực thi các ứng dụng Android.

3. KIẾN TRÚC CỦA ANDROID

Mô hình sau thể hiện một cách tổng quát các thành phần của hệ điều hành Android. Mỗi một phần sẽ được đặc tả một cách chi tiết dưới đây.



Hình 1-2 Cấu trúc stack hệ thống Android

3.1. Tầng ứng dụng

Android được tích hợp sẵn một số ứng dụng cần thiết cơ bản như: contacts, browser, camera, Phone,... Tất cả các ứng dụng chạy trên hệ điều hành Android đều được viết bằng Java.

3.2. Application framework

Bằng cách cung cấp một nền tảng phát triển mở, Android cung cấp cho các nhà phát triển khả năng xây dựng các ứng dụng cực kỳ phong phú và sáng tạo. Nhà phát triển được tự do tận dụng các thiết bị phần cứng, thông tin địa điểm truy cập, các dịch vụ chạy nền, thiết lập hệ thống báo động, thêm các thông báo để các thanh trạng thái, và nhiều, nhiều hơn nữa.

Nhà phát triển có thể truy cập vào các API cùng một khuôn khổ được sử dụng bởi các ứng dụng lõi. Các kiến trúc ứng dụng được thiết kế để đơn giản hóa việc sử dụng lại các thành phần; bất kỳ ứng dụng có thể xuất bản khả năng của mình và ứng dụng nào khác sau đó có thể sử dụng những khả năng (có thể hạn chế bảo mật được thực thi bởi khuôn khổ). Cơ chế này cho phép các thành phần tương tự sẽ được thay thế bởi người sử dụng.

Cơ bản tất cả các ứng dụng là một bộ các dịch vụ và các hệ thống, bao gồm:

- Một tập hợp rất nhiều các View có khả năng kế thừa lẫn nhau dùng để thiết kế phần giao diện ứng dụng như: gridview, tableview, linearlayout,...

- Một “Content Provider” cho phép các ứng dụng có thể truy xuất dữ liệu từ các ứng dụng khác (chẳng hạn như Contacts) hoặc là chia sẻ dữ liệu giữa các ứng dụng đó.
- Một “Resource Manager” cung cấp truy xuất tới các tài nguyên không phải là mã nguồn, chẳng hạn như: localized strings, graphics, and layout files.
- Một “Notification Manager” cho phép tất cả các ứng dụng hiển thị các custom alerts trong status bar.

Activity Manager được dùng để quản lý chu trình sống của ứng dụng và điều hướng các activity.

3.3. Library

Android bao gồm một tập hợp các thư viện C/C++ được sử dụng bởi nhiều thành phần khác nhau trong hệ thống Android. Điều này được thể hiện thông qua nền tảng ứng dụng Android. Một số các thư viện cơ bản được liệt kê dưới đây:

- **System C library:** a BSD-derived implementation of the standard C system library (libc), tuned for embedded Linux-based devices.
- **Media Libraries** - based on PacketVideo's OpenCORE; the libraries support playback and recording of many popular audio and video formats, as well as static image files, including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG
- **Surface Manager** – Quản lý việc truy xuất vào hệ thống hiển thị
- **LibWebCore** - a modern web browser engine which powers both the Android browser and an embeddable web view.
- **SGL** - the underlying 2D graphics engine.
- **3D libraries** - an implementation based on OpenGL ES 1.0 APIs; the libraries use either hardware 3D acceleration (where available) or the included, highly optimized 3D software rasterizer.
- **FreeType** - bitmap and vector font rendering.

SQLite - a powerful and lightweight relational database engine available to all applications.

3.4. Android Runtime

Android bao gồm một tập hợp các thư viện cơ bản mà cung cấp hầu hết các chức năng có sẵn trong các thư viện lõi của ngôn ngữ lập trình Java. Tất cả các ứng dụng Android đều chạy trong tiến trình riêng. Máy ảo Dalvik đã được viết để cho một thiết bị có thể chạy nhiều máy ảo hiệu quả. Các VM Dalvik thực thi các tập tin thực thi

Dalvik (dex). Định dạng được tối ưu hóa cho bộ nhớ tối thiểu. VM là dựa trên register-based, và chạy các lớp đã được biên dịch bởi một trình biên dịch Java để chuyển đổi thành các định dạng dex. Các VM Dalvik dựa vào nhân Linux cho các chức năng cơ bản như luồng và quản lý bộ nhớ thấp.

3.5. Linux kernel

Android dựa trên Linux phiên bản 2.6 cho hệ thống dịch vụ cốt lõi như security, memory management, process management, network stack, and driver model. Kernel Linux hoạt động như một lớp trừu tượng hóa giữa phần cứng và phần còn lại của phần mềm stack.

4. ANDROID EMULATOR

Android SDK và Plugin Eclipse được gọi là một Android Developer Tool (ADT). Các Android coder sẽ cần phải sử dụng công cụ IDE (Integrated Development Environment) này để phát triển, debugging và testing cho ứng dụng. Tuy nhiên, các coder cũng có thể không cần phải sử dụng IDE mà thay vào đó là sử dụng command line để biên dịch và tất nhiên là vẫn có Emulator như thường.

Android Emulator được trang bị đầy đủ hầu hết các tính năng của một thiết bị thật. Tuy nhiên, một số đã bị giới hạn như là kết nối qua cổng USB, camera và video, nghe phone, nguồn điện giả lập và bluetooth.

Android Emulator thực hiện các công việc thông qua một bộ xử lý mã nguồn mở, công nghệ này được gọi là QEMU (<http://bellard.org/qemu/>) được phát triển bởi Fabrice Bellard.



Hình 1-3 *Android emulator*

5. CÁC THÀNH PHẦN TRONG MỘT ANDROID PROJECT

5.1. AndroidManifest.xml

Trong bất kỳ một project Android nào khi tạo ra đều có một file AndroidManifest.xml, file này được dùng để định nghĩa các screen sử dụng, các permission cũng như các theme cho ứng dụng. Đồng thời nó cũng chứa thông tin về phiên bản SDK cũng như main activity sẽ chạy đầu tiên.

File này được tự động sinh ra khi tạo một Android project. Trong file manifest bao giờ cũng có 3 thành phần chính đó là: application, permission và version.

Dưới đây là nội dung của một file AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="dtu.k12tpm.pbs.activity"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon"
        android:label="@string/app_name">
        <activity android:name=".LoginActivity"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".MainContactListActivity" />
        <activity android:name=".RestoreContactActivity" />
    </application>
    <uses-sdk android:minSdkVersion="7" />
    <uses-sdk android:minSdkVersion="7" />
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.CALL_PHONE"/>
</manifest>
```

Application

Thẻ <application>, bên trong thẻ này chứa các thuộc tính được định nghĩa cho ứng dụng Android như:

- **android:icon** = “drawable resource” → Ở đây đặt đường dẫn đến file icon của ứng dụng khi cài đặt. VD: android:icon = “@drawable/icon”.
- **android:name** = “string” → thuộc tính này để đặt tên cho ứng dụng Android. Tên này sẽ được hiển thị lên màn hình sau khi cài đặt ứng dụng.
- **android:theme** = “drawable theme” → thuộc tính này để đặt theme cho ứng dụng. Các theme là các cách để hiển thị giao diện ứng dụng.

Ngoài ra còn nhiều thuộc tính khác...

Permission

Bao gồm các thuộc tính chỉ định quyền truy xuất và sử dụng tài nguyên của ứng dụng. Khi cần sử dụng một loại tài nguyên nào đó thì trong file manifest của ứng dụng cần phải khai báo các quyền truy xuất như sau:

```
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.ACCOUNT_MANAGER" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.CALL_PHONE" />
```

SDK version

Thẻ xác định phiên bản SDK được khai báo như sau:

```
<uses-sdk android:minSdkVersion="7" />.
```

Ở đây chỉ ra phiên bản SDK nhỏ nhất mà ứng dụng hiện đang sử dụng.

5.2. File R.java

File R.java là một file tự động sinh ra ngay khi tạo ứng dụng, file này được sử dụng để quản lý các thuộc tính được khai báo trong file XML của ứng dụng và các tài nguyên hình ảnh.

Mã nguồn của file R.java được tự động sinh khi có bất kỳ một sự kiện nào xảy ra làm thay đổi các thuộc tính trong ứng dụng. Chẳng hạn như, bạn kéo và thả một file hình ảnh từ bên ngoài vào project thì ngay lập tức thuộc tính đường dẫn đến file đó cũng sẽ được hình thành trong file R.java hoặc xóa một file hình ảnh thì đường dẫn tương ứng đến hình ảnh đó cũng tự động bị xóa.

Có thể nói file R.java hoàn toàn không cần phải đụng chạm gì đến trong cả quá trình xây dựng ứng dụng.

Dưới đây là nội dung của một file R.java:

```
/* AUTO-GENERATED FILE. DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found. It
 * should not be modified by hand.
```

```
*/
package dtu.kl2tpm.pbs.activity;
public final class R {
    public static final class array {
        public static final int array_timeout=0x7f050000;
    }
    public static final class attr {
    }
    public static final class drawable {
        public static final int add=0x7f020000;
        public static final int backup_icon=0x7f020001;
        public static final int checkall=0x7f020002;
    }
    public static final class id {
        public static final int Button01=0x7f070006;
        public static final int Button02=0x7f070007;
        public static final int CheckBox01=0x7f070017;
    }
    public static final class layout {
        public static final int contact_list=0x7f030000;
        public static final int content_sender=0x7f030001;
        public static final int friend_list=0x7f030002;
    }
    public static final class menu {
        public static final int context_menu=0x7f060000;
        public static final int menu_options=0x7f060001;
        public static final int options_menu=0x7f060002;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int context_menu_item_delete=0x7f04000b;
        public static final int context_menu_item_edit=0x7f04000a;
    }
}
```

6. CHU KỲ ỨNG DỤNG ANDROID

Một tiến trình Linux gói gọn một ứng dụng Android đã được tạo ra cho ứng dụng khi codes cần được run và sẽ còn chạy cho đến khi:

- Nó không phụ thuộc.
- Hệ thống cần lấy lại bộ nhớ mà nó chiếm giữ cho các ứng dụng khác

Một sự khác thường và đặc tính cơ bản của Android là thời gian sống của tiến trình ứng dụng không được điều khiển trực tiếp bởi chính nó. Thay vào đó, nó được xác định bởi hệ thống qua một kết hợp của:

- Những phần của ứng dụng mà hệ thống biết đang chạy
- Những phần quan trọng như thế nào đối với người dùng

Bao nhiêu vùng nhớ chiếm lĩnh trong hệ thống.

6.1. Chu kỳ sống thành phần

Các thành phần ứng dụng có một chu kỳ sống, tức là mỗi thành phần từ lúc bắt đầu khởi tạo và đến thời điểm kết thúc. Giữa đó, đôi lúc chúng có thể là active hoặc inactive, hoặc là trong trường hợp activities nó có thể visible hoặc invisible

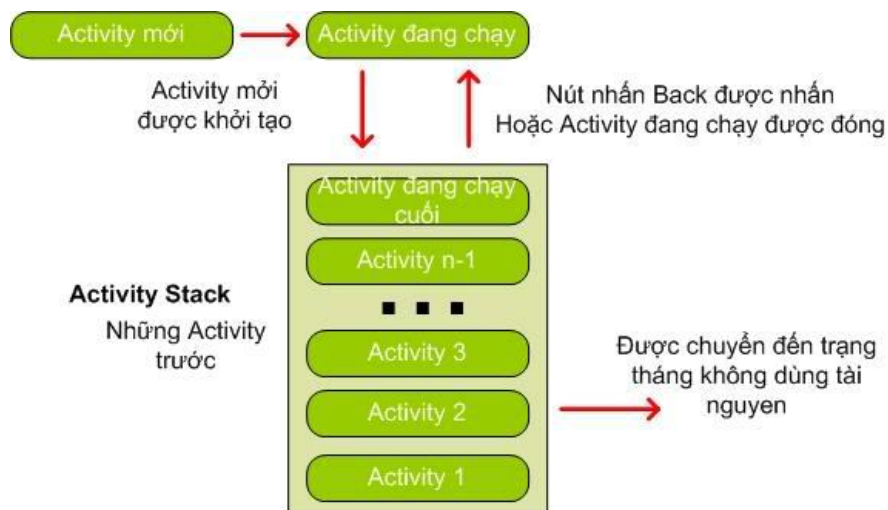


Hình 1-4

6.2. Activity Stack

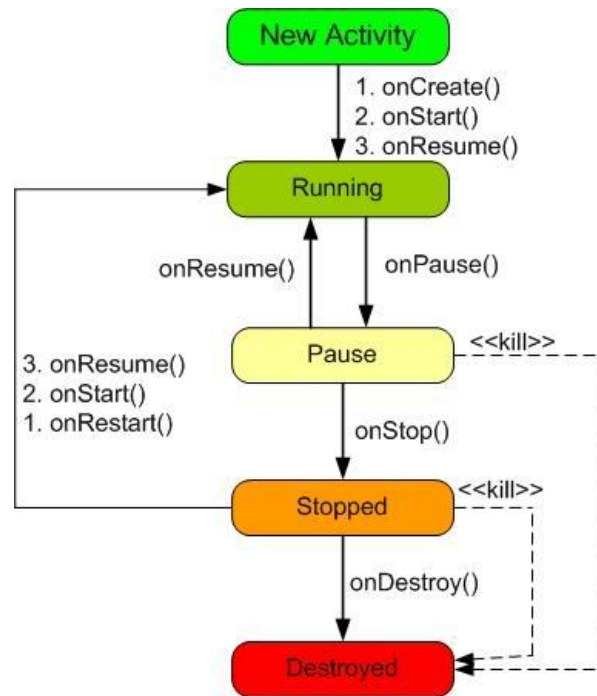
Bên trong hệ thống các activity được quản lý như một activity stack. Khi một Activity mới được start, nó được đặt ở đỉnh của stack và trở thành activity đang chạy activity trước sẽ ở bên dưới activity mới và sẽ không thấy trong suốt quá trình activity mới tồn tại.

Nếu người dùng nhấn nút Back thì activity kết tiếp của stack sẽ di chuyển lên và trở thành active.



Hình 1-5 Activity stack

6.3. Các trạng thái của chu kỳ sống



Hình 1-6 Chu kỳ sống của Activity

Một Activity chủ yếu có 3 chu kỳ chính sau:

- Active hoặc running: Khi Activity là được chạy trên màn hình. Activity này tập trung vào những thao tác của người dùng trên ứng dụng.
- Paused: Activity là được tạm dừng (paused) khi mất focus nhưng người dùng vẫn trông thấy. Có nghĩa là một Activity mới ở trên nó nhưng không bao phủ đầy màn hình. Một Activity tạm dừng là còn sống nhưng có thể bị kết thúc bởi hệ thống trong trường hợp thiếu vùng nhớ.

Stopped: Nếu nó hoàn toàn bao phủ bởi Activity khác. Nó vẫn còn trạng thái và thông tin thành viên trong nó. Người dùng không thấy nó và thường bị loại bỏ trong trường hợp hệ thống cần vùng nhớ cho tác vụ khác.

6.4. Chu kỳ sống của ứng dụng

Trong một ứng dụng Android có chứa nhiều thành phần và mỗi thành phần đều có một chu trình sống riêng. Và ứng dụng chỉ được gọi là kết thúc khi tất cả các thành phần trong ứng dụng kết thúc. Activity là một thành phần cho phép người dùng giao tiếp với ứng dụng. Tuy nhiên, khi tất cả các Activity kết thúc và người dùng không còn giao tiếp được với ứng dụng nữa nhưng không có nghĩa là ứng dụng đã kết thúc. Bởi vì ngoài Activity là thành phần có khả năng tương tác người dùng thì còn có các thành phần không có khả năng tương tác với người dùng như là Service, Broadcast

receiver. Có nghĩa là những thành phần không tương tác người dùng có thể chạy background dưới sự giám sát của hệ điều hành cho đến khi người dùng tự tắt chúng.

6.5. Các sự kiện trong chu kỳ sống của ứng dụng

Nếu một Activity được tạm dừng hoặc dừng hẳn, hệ thống có thể bỏ thông tin khác của nó từ vùng nhớ bởi việc finish() (gọi hàm finish() của nó), hoặc đơn giản giết tiến trình của nó. Khi nó được hiển thị lần nữa với người dùng, nó phải được hoàn toàn restart và phục hồi lại trạng thái trước. Khi một Activity chuyển qua chuyển lại giữa các trạng thái, nó phải báo việc chuyển của nó bằng việc gọi hàm transition.

<code>void onCreate(Bundle savedInstanceState)</code>	<code>Void onPause()</code>
<code>void onStart()</code>	<code>Void onStop()</code>
<code>void onRestart()</code>	<code>Void onDestroy()</code>
<code>void onResume()</code>	

Hình 1-7

Tất cả các phương thức là những móc nối mà bạn có thể override để làm tương thích công việc trong ứng dụng khi thay đổi trạng thái. Tất cả các Activity bắt buộc phải có onCreate() để khởi tạo ứng dụng. Nhiều Activity sẽ cũng hiện thực onPause() để xác nhận việc thay đổi dữ liệu và mặt khác chuẩn bị dừng hoạt động với người dùng.

6.6. Thời gian sống của ứng dụng

Bảy phương thức chuyển tiếp định nghĩa trong chu kỳ sống của một Activity. Thời gian sống của một Activity diễn ra giữa lần đầu tiên gọi onCreate() đến trạng thái cuối cùng gọi onDestroy(). Một Activity khởi tạo toàn bộ trạng thái toàn cục trong onCreate(), và giải phóng các tài nguyên đang tồn tại trong onDestroy().

6.7. Thời gian hiển thị của Activity

Visible lifetime của một activity diễn ra giữa lần gọi một onStart() cho đến khi gọi onStop(). Trong suốt khoảng thời gian này người dùng có thể thấy activity trên màn hình, có nghĩa là nó không bị foreground hoặc đang tương tác với người dùng. Giữa 2 phương thức người dùng có thể duy trì tài nguyên để hiển thị activity đến người dùng.

6.8. Các phương thức của chu kỳ sống

Phương thức: onCreate()

- ❖ Được gọi khi activity lần đầu tiên được tạo
- ❖ Ở đây bạn làm tất cả các cài đặt tĩnh -- tạo các view, kết nối dữ liệu đến list và .v.v
- ❖ Phương thức này gửi qua một đối tượng *Bundle* chứa đựng từ trạng thái trước của Activity

- ❖ Luôn theo sau bởi onStart()

Phương thức: onRestart()

- ❖ Được gọi sau khi activity đã được dừng, chỉ một khoảng đang khởi động lần nữa (started again)
- ❖ Luôn theo sau bởi onStart()

Phương thức: onStart()

- ❖ Được gọi trước khi một activity visible với người dùng.
- ❖ Theo sau bởi *onResume()* nếu activity đến trạng thái foreground hoặc *onStop()* nếu nó trở nên ẩn.

Phương thức: onResume()

- ❖ Được gọi trước khi activity bắt đầu tương tác với người dùng
- ❖ Tại thời điểm này activity ở trên đỉnh của stack activity.
- ❖ Luôn theo sau bởi *onPause()*

Phương thức: onPause()

- ❖ Được gọi khi hệ thống đang resuming activity khác.
- ❖ Phương thức này là diễn hình việc giữ lại không đổi dữ liệu.
- ❖ Nó nên được diễn ra một cách nhanh chóng bởi vì activity kế tiếp sẽ không được resumed ngay cho đến khi nó trở lại.
- ❖ Theo sau bởi *onResume* nếu activity trở về từ ở trước, hoặc bởi *onStop* nếu nó trở nên visible với người dùng.
- ❖ Trạng thái của activity có thể bị giết bởi hệ thống.

Phương thức: onStop()

- ❖ Được gọi khi activity không thuộc tầm nhìn của người dùng.
- ❖ Nó có thể diễn ra bởi vì nó đang bị hủy, hoặc bởi vì activity khác vừa được resumed và bao phủ nó.
- ❖ Được theo sau bởi *onRestart()* nếu activity đang chờ lại để tương tác với người dùng, hoặc *onDestroy()* nếu activity đang bỏ.
- ❖ Trạng thái của activity có thể bị giết bởi hệ thống.

Phương thức: onDestroy()

- ❖ Được gọi trước khi activity bị hủy.
- ❖ Đó là lần gọi cuối cùng mà activity này được nhận.
- ❖ Nó được gọi khác bởi vì activity đang hoàn thành, hoặc bởi vì hệ thống tạm thời bị hủy diệt để tiết kiệm vùng nhớ.
- ❖ Bạn có thể phân biệt giữa 2 kịch bản với phương *isFinishing()*.

Trạng thái của activity có thể được giết bởi hệ thống.

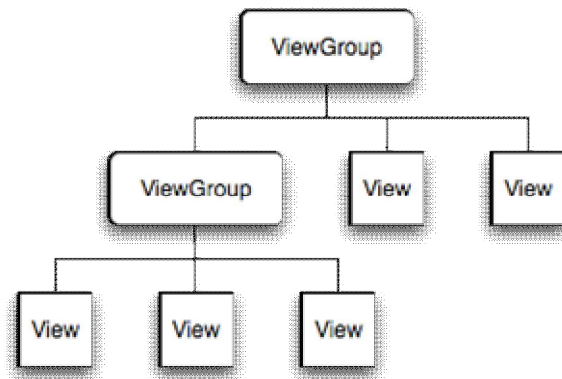
7. CÁC THÀNH PHẦN GIAO DIỆN TRONG ANDROID

7.1. View

Trong một ứng dụng Android, giao diện người dùng được xây dựng từ các đối tượng View và ViewGroup. Có nhiều kiểu View và ViewGroup. Mỗi một kiểu là một hậu duệ của class View và tất cả các kiểu đó được gọi là các Widget.

Tất cả mọi widget đều có chung các thuộc tính cơ bản như là cách trình bày vị trí, background, kích thước, lề,... Tất cả những thuộc tính chung này được thể hiện hết ở trong đối tượng View.

Trong Android Platform, các screen luôn được bố trí theo một kiểu cấu trúc phân cấp như hình dưới. Một screen là một tập hợp các Layout và các widget được bố trí có thứ tự. Để thể hiện một screen thì trong hàm onCreate của mỗi Activity cần phải được gọi một hàm là setContentView(R.layout.main); hàm này sẽ load giao diện từ file



XML lên để phân tích thành mã bytecode.

Hình 1-8 Cấu trúc một giao diện ứng dụng Android

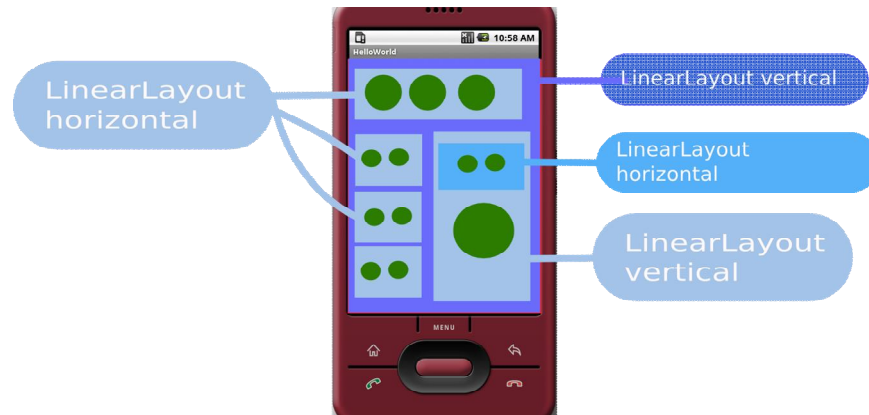
7.2. ViewGroup

ViewGroup thực ra chính là View hay nói đúng hơn thì ViewGroup chính là các widget Layout được dùng để bố trí các đối tượng khác trong một screen. Có một số loại ViewGroup như sau:

7.2.1. LinearLayout

LinearLayout được dùng để bố trí các thành phần giao diện theo chiều ngang hoặc chiều dọc nhưng trên một line duy nhất mà không có xuống dòng.

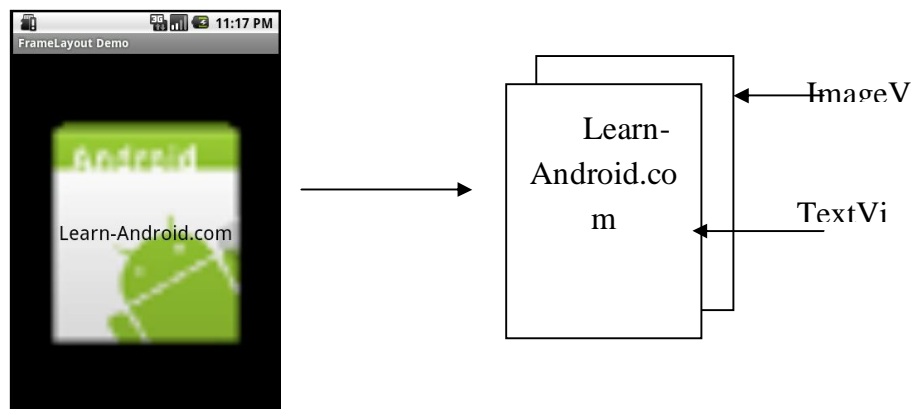
LinearLayout làm cho các thành phần trong nó không bị phụ thuộc vào kích thước của màn hình. Các thành phần trong LinearLayout được dàn theo những tỷ lệ cân xứng dựa vào các ràng buộc giữa các thành phần.



Hình 1-9 Bố trí các widget sử dụng *LinearLayout*

7.2.2. **FrameLayout**

FrameLayout được dùng để bố trí các đối tượng theo kiểu giống như là các Layer trong Photoshop. Những đối tượng nào thuộc Layer bên dưới thì sẽ bị che khuất bởi các đối tượng thuộc Layer nằm trên. FrameLayer thường được sử dụng khi muốn tạo ra các đối tượng có khung hình bên ngoài chẳng hạn như contact image button.



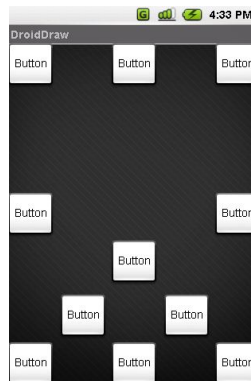
Hình 1-10 Bố trí các widget trong *FrameLayout*

7.2.3. **AbsoluteLayout**

Layout này được sử dụng để bố trí các widget vào một vị trí bất kỳ trong layout dựa vào 2 thuộc tính tọa độ x, y. Tuy nhiên, kiểu layout này rất ít khi được dùng bởi vì tọa độ của các đối tượng luôn cố định và sẽ không tự điều chỉnh được tỷ lệ khoảng cách giữa các đối tượng. Khi chuyển ứng dụng sang một màn hình có kích thước với màn hình thiết kế ban đầu thì vị trí của các đối tượng sẽ không còn được chính xác như ban đầu.

7.2.4. RelativeLayout

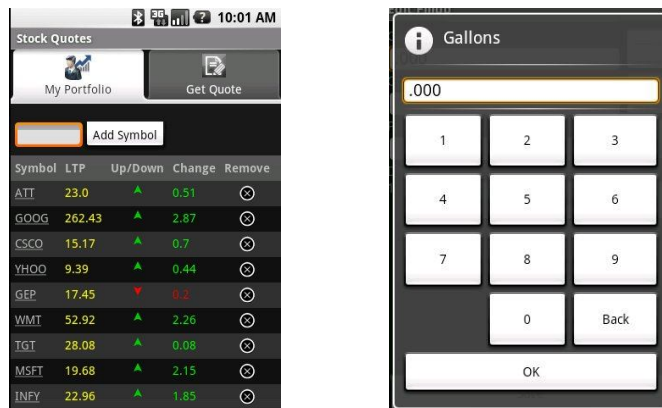
Layout này cho phép bố trí các widget theo một trục đối xứng ngang hoặc dọc. Để đặt được đúng vị trí thì các widget cần được xác định một mối ràng buộc nào đó với các widget khác. Các ràng buộc này là các ràng buộc trái, phải, trên, dưới so với một widget hoặc so với layout parent. Dựa vào những mối ràng buộc đó mà RelativeLayout cũng không phụ thuộc vào kích thước của screen thiết bị. Ngoài ra, nó còn có ưu điểm là giúp tiết kiệm layout sử dụng nhằm mục đích giảm lượng tài nguyên sử dụng khi load đồng thời đẩy nhanh quá trình xử lý.



Hình 1-11 Bố trí widget trong RelativeLayout

7.2.5. TableLayout

Layout này được sử dụng khi cần thiết kế một table chứa dữ liệu hoặc cần bố trí các widget theo các row và column. Chẳng hạn như, giao diện của một chiếc máy tính đơn giản hoặc một danh sách dữ liệu.



Hình 1-12 Bố trí widget trong TableLayout

7.3. Button

Sở dĩ widget button được giới thiệu đầu tiên trong số các widget khác là vì đây là đối tượng có thể nói là được dùng nhiều nhất trong hầu hết các ứng dụng Android.

Để thiết kế giao diện với một button ta có 2 cách như sau:

Thiết kế bằng XML

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/cmdButton1"
    android:text="Touch me!"
    android:onClick="touchMe" />
```

Thuộc tính `android:onClick="touchMe"` được dùng để nắm bắt sự kiện click vào button. Khi sự kiện click button xảy ra thì phương thức “touchMe” được khai báo trong thẻ thuộc tính sẽ được gọi. Nếu trường hợp phương thức “touchMe” chưa được khai báo trong file mã nguồn tương ứng thì sẽ phát sinh một exception. Ngược lại, phương thức “touchMe” sẽ nhận được một đối tượng tham biến là đối tượng View nơi đã phát sinh ra sự kiện. Đối tượng View này có thể ép kiểu trực tiếp sang kiểu Button vì thực chất nó là một button.

VD: trong file mã nguồn khai báo một hàm như sau:

```
public void touchMe(View v){
    Button me = (Button) v;
    Me.setText("Touched");
}
```

Thiết kế bằng code

Thực ra mà nói thì nếu không phải đòi hỏi phải custom lại một widget thì không cần phải sử dụng tới code. Trong một số trường hợp bắt buộc chúng ta phải custom các widget để cho phù hợp với hoàn cảnh. Chẳng hạn như trong game, các menu hay các nút điều khiển,...

Để khai báo một Button trong code ta làm như sau:

```
Button cmdButton = new Button(this);
cmdButton.setText("Touch Me!");
cmdButon.setOnClickListener(...);
```

Để custom một widget nào đó ta phải tạo một class kế thừa từ class Widget muốn custom, sau đó sử dụng hàm draw để vẽ lại widget đó như một Canvas.

VD: `canvas.drawPicture(Picture.createFromStream(...));`

7.4. ImageButton

Cũng tương tự như Button, ImageButton chỉ có thêm một thuộc tính

android:src = “@drawable/icon” để thêm hình ảnh vào và không có thẻ text

```
<ImageButton  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:id="@+id/cmdButton1"  
    android:src="@drawable/icon"  
    android:onClick="touchMe" />
```



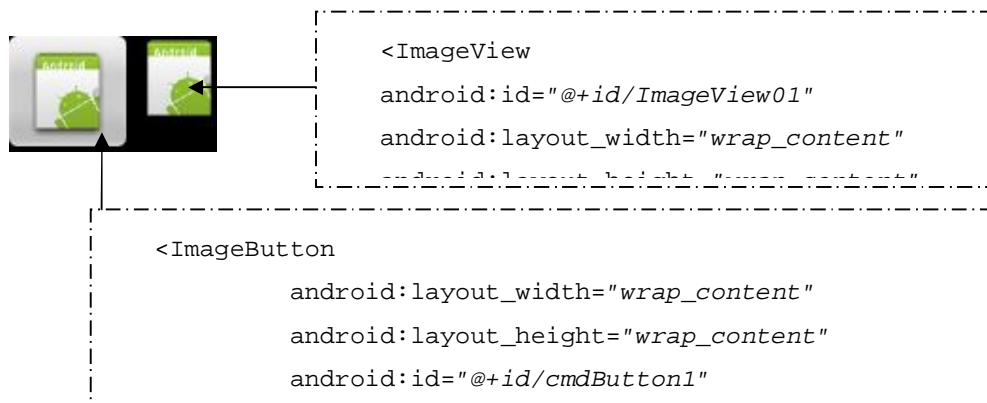
Hình 1-13 ImageButton

7.5. ImageView

Được dùng để thể hiện một hình ảnh. Nó cũng giống như ImageButton, chỉ khác là không có hình dáng của một cái button.

Code:

```
ImageView iv = new ImageView(this);  
iv.setImageResource(R.drawable.icon);
```



Hình 1-14 ImageView và ImageButton

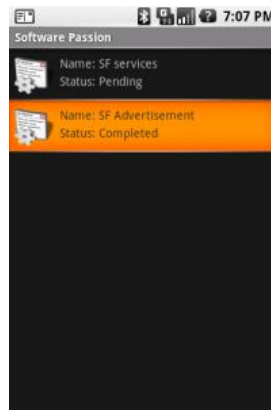
7.6. ListView

Được sử dụng để thể hiện một danh sách các thông tin theo từng cell. Mỗi cell thông thường được load lên từ một file XML đã được cố định trên đó số lượng thông tin và loại thông tin cần được thể hiện.

Để thể hiện được một list thông tin lên một screen thì cần phải có 3 yếu tố chính:

- ✚ **Data Source:** Data Source có thể là một ArrayList, HashMap hoặc bất kỳ một cấu trúc dữ liệu kiểu danh sách nào.

- ✚ **Adapter:** Adapter là một class trung gian giúp ánh xạ dữ liệu trong Data Source vào đúng vị trí hiển thị trong ListView. Chẳng hạn, trong Data Source có một trường name và trong ListView cũng có một TextView để thể hiện trường name này. Tuy nhiên, ListView sẽ không thể hiển thị dữ liệu trong Data Source lên được nếu như Adapter không gán dữ liệu vào cho đối tượng hiển thị.
- ✚ **ListView:** ListView là đối tượng để hiển thị các thông tin trong Data Source ra một cách trực quan và người dùng có thể thao tác trực tiếp trên đó.



Hình 1-15 Minh họa cho một ListView

7.7. TextView

TextView ngoài tác dụng là để hiển thị văn bản thì nó còn cho phép định dạng nội dung bằng thẻ html.

VD:

```
TextView textView = (TextView)findViewById(R.id.textView);
CharSequence styledText =
Html.fromHtml("<i>This</i> is some <b>styled</b> <s>text</s>");
textView.setText(styledText);
```

Nội dung TextView cũng có thể được định dạng bằng thẻ html ngay trong XML.


7.8. EditText

Trong Android đối tượng EditText được sử dụng như một TextField hoặc một TextBox.

```
<EditText
    android:id="@+id/EditText01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textStyle="bold"
```

```
android:textSize="20dip"
android:textColor="#000000"
android:text="Hello Android!"
android:singleLine="true"
android:inputType="textCapWords"/>
```

Các thuộc tính cần chú ý sử dụng EditText đó là:

 **android:inputType** = “...” sử dụng để xác định phương thức nhập cho EditText. Chẳng hạn như khi bạn muốn một ô để nhập password hay một ô để nhập Email thì thuộc tính này sẽ làm điều đó.

android:singleLine = “true” EditText của bạn sẽ trở thành một TextField, ngược lại sẽ là TextBox.

7.9. CheckBox

Nhận 2 giá trị true hoặc false. Đối tượng CheckBox cho phép chọn nhiều item cùng một lúc.


Khai báo: `CheckBox cb = new CheckBox(Context ...);`

XML:

```
<CheckBox
android:id="@+id/CheckBox01"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Check me"
android:checked="true"/>
```

7.10. MenuOptions

Có 2 cách tạo một MenuOptions:

 Tạo bằng code:

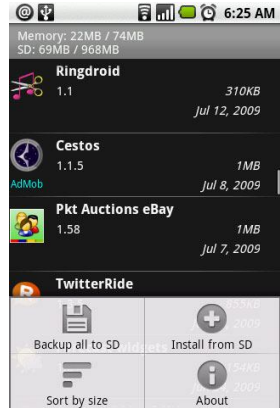
```
public class Main extends Activity {

    private int searchBtnId = Menu.FIRST;
    private int scheduleBtnId = Menu.FIRST + 1;
    private int playBtnId = Menu.FIRST + 2;
    private int stopBtnId = Menu.FIRST + 3;
    private int group1Id = 1;
    private int group2Id = 2;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this setContentView(R.layout.main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
```



```
menu.add(group1Id,searchBtnId ,searchBtnId,"Search");
menu.add(group2Id,scheduleBtnId,scheduleBtnId,R.string.schedule);
menu.add(group2Id,playBtnId ,playBtnId,"Play");
menu.add(group2Id,stopBtnId ,stopBtnId,R.string.stop);
// the following line will hide search
// when we turn the 2nd parameter to false
menu.setGroupVisible(1, false);
return super.onCreateOptionsMenu(menu);
}
}
```



Hình 1-16 Minh hoạ option menu

Tạo bằng XML

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:icon="@drawable/icon" android:title="Item1"
        android:orderInCategory="1" android:id="@+id/item1">
        <item android:title="Item 2" android:id="@+id/item2"
            android:orderInCategory="2">
            <menu>
                <item android:id="@+id/item01" android:title="Sub item 1"
                    android:orderInCategory="1"/>
                <item android:title="Sub item 2" android:id="@+id/item02"
                    android:orderInCategory="2" />
            </menu>
        </item>
    </menu>
</menu>
```

```
public boolean onCreateOptionsMenu(Menu menu) {
    new MenuInflater(
        getApplication()).inflate(R.menu.menu_options, menu);
    return super.onCreateOptionsMenu(menu);
}
```

7.11. ContextMenu

ContextMenu được sử dụng để hiển thị các tùy chọn khi người dùng nhấn dài vào một cell nào đó trong ListView. Để tạo một ContextMenu ta cũng có 2 cách giống như tạo MenuOptions ở trên chỉ khác tên phương thức.

Khi nhấn dài vào một cell trong ListView thì phương thức:

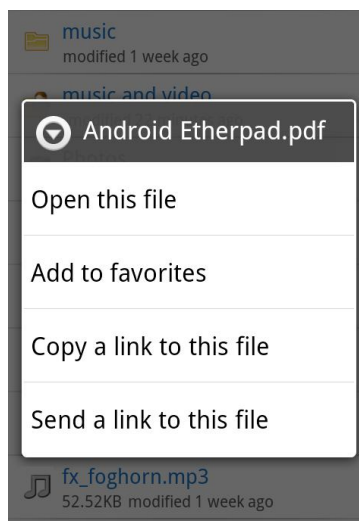
```
public void onCreateContextMenu(ContextMenu menu, View v,
ContextMenuItem menuInfo)
```

sẽ được gọi và truyền vào 3 tham số là:

- ❖ ContextMenu: đối tượng để add các context menu item
- ❖ View: Đối tượng nơi mà xảy ra sự kiện

ContextMenuInfo: Cho biết vị trí xảy ra sự kiện trong ListView. Để biết được vị trí xảy ra sự kiện ta làm như sau:

```
AdapterView.AdapterContextMenuInfo info;
try {
    info = (AdapterView.AdapterContextMenuInfo) menuInfo;
} catch (ClassCastException e) {
    return;
}
info.position
```

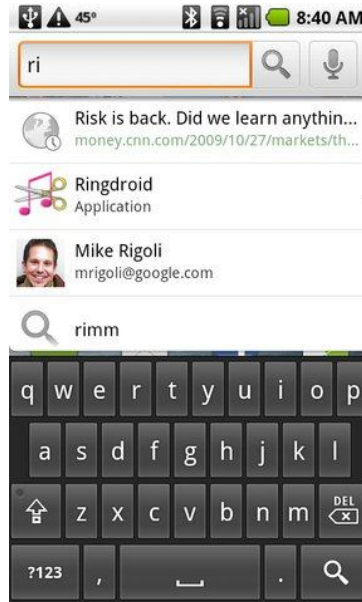


Hình 1-17 Minh họa context menu

7.12. Quick Search Box

Một trong những tính năng mới trong phiên bản Android 1.6 đó là Quick Search Box. Đây là khuôn khổ tìm kiếm mới trên toàn hệ thống Android, điều này làm cho người dùng có thể nhanh chóng tìm kiếm bất cứ thứ gì có trên chiếc điện thoại Android của họ và cả các tài nguyên trên web khi họ đang online. Nó tìm kiếm và

hiển thị kết quả tìm kiếm ngay khi bạn đang gõ. Nó cũng cung cấp các kết quả từ các gợi ý tìm kiếm web, danh sách doanh nghiệp địa phương, và thông tin khác từ Google, chẳng hạn như báo giá cổ phiếu, thời tiết, và tình trạng chuyến bay. Tất cả điều này có sẵn ngay từ màn hình chủ, bằng cách khai thác trên Quick Search Box (QSB).



Hình 1-18 Minh họa Quick Search Box

7.13. Activity & Intend

7.13.1. Activity

Activity là một thành chính của một ứng dụng Android, được dùng để hiển thị một màn hình và nắm bắt các hoạt động xảy ra trên màn hình đó. Khi làm việc với Activity cần nắm bắt được một số kiến thức cơ bản như sau:

❖ Chu kỳ sống của một Activity

(Xem chu kỳ ứng dụng của Android mục 1.6)

❖ Tạo menu và dialog

❖ Khởi động một Activity

Để khởi động một Activity ta sử dụng Intend sẽ tìm hiểu kỹ hơn ở phần b. Tuy nhiên, trong phần này tôi sẽ hướng dẫn cách chuyển giữa các Intend theo 2 loại:

🚦 Khai báo không tường minh:

Cung cấp chính xác thông tin của activity cần gọi bằng cách truyền vào tên class của Activity đó

VD: Từ Activity A muốn chuyển qua Activity B ta khai báo một Intend trong Activity A:

```
Intent intend = new Intent(this, B.class);  
startActivity(intend);
```

✚ Khai báo không tường minh

Cung cấp các thao tác cần làm gì với loại dữ liệu nào, hệ thống sẽ tìm đến activity tương ứng để khởi động.

VD: Để xem thông tin một contact nào đó trong Activity của ứng dụng Contact trong Android ta chỉ đến dữ liệu contact và chỉ đến Activity View contact như sau:

```
Intent i = new Intent();  
i.setAction(Intent.ACTION_VIEW);  
i.setData(Uri.withAppendedPath(  
    android.provider.Contacts.People.CONTENT_URI, "1"));  
startActivity(i);
```

❖ **Tính liên lạc giữa 2 activity**

Khi chuyển sang một Activity khác ta có thể gửi kèm dữ liệu trong intend đó như sau:

```
intend.putExtra("key1", "value1");  
intend.putExtra("key2", 23);
```

Bên phía Activity được khởi động hay được chuyển đến, có thể lấy dữ liệu được gửi như sau:

```
getIntent().getStringExtra().getString("key1");  
getIntent().getStringExtra().getInt("key2");
```

❖ **Task**

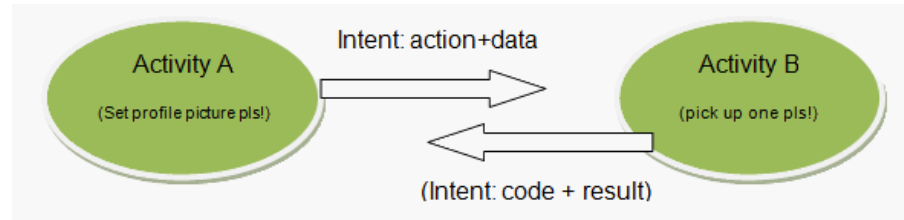
Android là một hệ điều hành đa tiến trình. Khi lập trình trên nền tảng Android thì tiến trình là một vấn đề cần phải được chú ý nhiều nhất. Mặc dù Android hỗ trợ đa tiến trình nhưng trên một thiết bị di động với cấu hình thấp mà chúng ta quá lạm dụng tiến trình thì sẽ rất tốn bộ xử lý điều này cũng đồng nghĩa với việc bạn đang biến ứng dụng của bạn trở thành một thứ phần mềm tiêu thụ điện năng.

7.13.2.Intent

❖ Khái niệm Intent:

- ✓ Là một cấu trúc dữ liệu mô tả cách thức, đối tượng thực hiện của một Activity
- ✓ Là cầu nối giữa các Activity: ứng dụng Android thường bao gồm nhiều Activity, mỗi Activity hoạt động độc lập với nhau và thực hiện những công việc khác nhau. Intent chính là người đưa thư, giúp các Activity có thể triệu gọi cũng như truyền các dữ liệu cần thiết tới một Activity khác. Điều này cũng

giống như việc di chuyển qua lại giữa các Forms trong lập trình Windows Form.



Hình 1-19 Truyền dữ liệu giữa 2 Activity

❖ Dữ liệu của Intent:

- ✓ Intent về cơ bản là một cấu trúc dữ liệu, được mô tả trong lớp android.content.Intent
- ✓ Các thuộc tính của một đối tượng Intent:

Thuộc tính chính	Thuộc tính phụ
action -tên (string) của action mà Intent sẽ yêu cầu thực hiện -có thể là action được Android định nghĩa sẵn (built-in standard action) hoặc do người lập trình tự định nghĩa	category -thông tin về nhóm của action
data -dữ liệu mà Activity được gọi sẽ xử lý -định dạng Uri (thông qua hàm Uri.parse(data))	type -định dạng kiểu dữ liệu (chuẩn MIME) -thường được tự động xác định
	component -chỉ định cụ thể lớp sẽ thực thi Activity -khi được xác định, các thuộc tính khác trở thành không bắt buộc (optional)
	extras -chứa tất cả các cặp (key,value) do ứng dụng thêm vào để truyền qua Intent (cấu trúc Bundle)
http://developer.android.com/reference/android/content/Intent.html	

Hình 1-20 Các thuộc tính của Intent

➤ Các Action được định nghĩa sẵn:

Dưới đây là những hằng String đã được định nghĩa sẵn trong lớp Intent. Đi kèm với nó là các Activity hay Application được xây dựng sẵn sẽ được triệu gọi mỗi khi Intent tương ứng được gửi (tất nhiên khi được cung cấp đúng data). VD: Gọi tới một số điện thoại:

```
Intent dialIntent =  
    new Intent(Intent.ACTION_DIAL, Uri.parse("tel:123456"));  
startActivity(dialIntent);
```

Built-in Standard Actions	
ACTION_MAIN	ACTION_ANSWER
ACTION_VIEW	ACTION_INSERT
ACTION_ATTACH_DATA	ACTION_DELETE
ACTION_EDIT	ACTION_RUN
ACTION_PICK	ACTION_SYNC
ACTION_CHOOSER	ACTION_PICK_ACTIVITY
ACTION_GET_CONTENT	ACTION_SEARCH
ACTION_DIAL	ACTION_WEB_SEARCH
ACTION_CALL	ACTION_FACTORY_TEST
ACTION_SEND	ACTION_SENDTO
Built-in Standard Broadcast Actions	
ACTION_TIME_TICK	ACTION_PACKAGE_RESTARTED
ACTION_TIME_CHANGED	ACTION_PACKAGE_DATA_CLEARED
ACTION_TIMEZONE_CHANGED	ACTION_UID_REMOVED
ACTION_BOOT_COMPLETED	ACTION_BATTERY_CHANGED
ACTION_PACKAGE_ADDED	ACTION_POWER_CONNECTED
ACTION_PACKAGE_CHANGED	ACTION_POWER_DISCONNECTED
ACTION_PACKAGE_REMOVED	ACTION_SHUTDOWN

Hình 1-21 Các Action đã được định nghĩa sẵn trong Intent

8. CONTENT PROVIDER VÀ URI

Trong hệ thống Android tất cả các tài nguyên như Contact, SMS,... đều được lưu trữ vào CSDL SQLite của hệ thống. Cũng như các CSDL khác, CSDL mà hệ thống Android sử dụng để lưu trữ thông tin cũng cho phép chúng ta truy vấn dữ liệu như một CSDL MSSQL thông thường. Tuy nhiên, trong hệ thống đó chúng ta không cần phải thao tác bằng lệnh SQL nhiều để truy xuất dữ liệu mà thay vào đó Android đã được trang bị một API cho phép người lập trình có thể dễ dàng truy xuất dữ liệu. Đó gọi là ContentProvider. ContentProvider cung cấp cho chúng ta một đối tượng con trỏ giúp chúng ta có thể dễ dàng lấy được bất cứ dữ liệu lưu trữ nào chỉ cần cung cấp một đường dẫn đúng đến dữ liệu đó. Đường dẫn này còn được gọi là Uri.

✚ Tạo một Uri:

`content://com.example.transportationprovider/trains/122`

The diagram shows the URI `content://com.example.transportationprovider/trains/122` with brackets underneath identifying its parts: A is `content`, B is `com.example.transportationprovider`, C is `trains`, and D is `122`.

Uri uri = Uri.parse("content://com.android.contacts/contacts");

✚ Cấu trúc gồm có 4 phần chính như sau:

Phần A: Đây là tiền tố chỉ ra dữ liệu được điều khiển bởi Content Provider và nó không bao giờ thay đổi.

Phần B: Phần này chỉ đến nơi lưu trữ dữ liệu. Cũng giống như cấu trúc của một số điện thoại thì cái này có thể hình dung nó như là mã quốc gia hoặc cũng có thể coi nó như là tên của CSDL.

Phần C: Phần này chỉ ra loại dữ liệu. Chẳng hạn như, dữ liệu contact, dữ liệu SMS,... Phần này có thể coi nó như là tên của một table

Phần D: Phần này chỉ đến đúng vị trí của dữ liệu, có thể coi phần này như là ID của row trong table hoặc một dữ liệu nào đó dùng để truy vấn.

VD: Uri chỉ đến contact thứ 0 trong CSDL là

`content://contacts/people/0`

Để có thể thực hiện truy vấn đến vùng dữ liệu được chỉ ra bởi một Uri ta cần có 2 đối tượng con trỏ được cung cấp bởi Activity đó là: Cursor và ContentResolver.

Để lấy được 2 đối tượng này thì trong Activity sử dụng hàm

`getContentResolver()` trả về đối tượng ContentResolver.

`getContentResolver().query(Uri uri);` trả về đối tượng Cursor.

9. BACKGROUND SERVICE

Service là 1 trong 4 thành phần chính trong 1 ứng dụng Android (Activity, Service, BroadcastReceiver, ContentProvider) thành phần này chạy trong hậu trường và làm những công việc không cần tới giao diện như chơi nhạc, download, xử lý tính toán...

Một Service có thể được sử dụng theo 2 cách:

- 🚩 Nó có thể được bắt đầu và được cho phép hoạt động cho đến khi một người nào đó dừng nó lại hoặc nó tự ngắt. Ở chế độ này, nó được bắt đầu bằng cách gọi `Context.startService()` và dừng bằng lệnh `Context.stopService()`. Nó có thể tự ngắt bằng lệnh `Service.stopSelf()` hoặc `Service.stopSelfResult()`. Chỉ cần một lệnh `stopService()` để ngừng Service lại cho dù lệnh `startService()` được gọi ra bao nhiêu lần.
- 🚩 Service có thể được vận hành theo như đã được lập trình việc sử dụng một Interface mà nó định nghĩa. Các người dùng thiết lập một đường truyền tới đối tượng Service và sử dụng đường kết nối đó để thâm nhập vào Service. Kết nối này được thiết lập bằng cách gọi lệnh `Context.bindService()` và được đóng lại bằng cách gọi lệnh `Context.unbindService()`. Nhiều người dùng có thể kết nối tới cùng một thiết bị. Nếu Service vẫn chưa được khởi chạy, lệnh `bindService()` có thể tùy ý khởi chạy nó. Hai chế độ này thì không tách biệt toàn bộ. Bạn có thể kết nối với một Service mà nó đã được bắt đầu với lệnh `startService()`. Ví dụ, một Service nghe nhạc ở chế độ nền có thể được bắt đầu bằng cách gọi lệnh `startService()` cùng với một đối tượng Intent mà định dạng được âm nhạc để chơi. Chỉ sau đó, có thể là khi người sử dụng muốn kiểm soát trình chơi nhạc hoặc biết thêm thông tin về bài hát hiện tại đang chơi, thì sẽ có một Activity tạo lập một đường truyền tới Service bằng cách gọi `bindService()`. Trong trường hợp như thế này, `stopService()` sẽ không thực sự ngừng Service cho đến khi liên kết cuối cùng được đóng lại.

Giống như một Activity, một Service cũng có các phương thức chu kỳ thời gian mà bạn có thể cài đặt để kiểm soát những sự thay đổi trong trạng thái của nó. Service chỉ có 3 phương thức được gọi đến trong chu trình sống là:

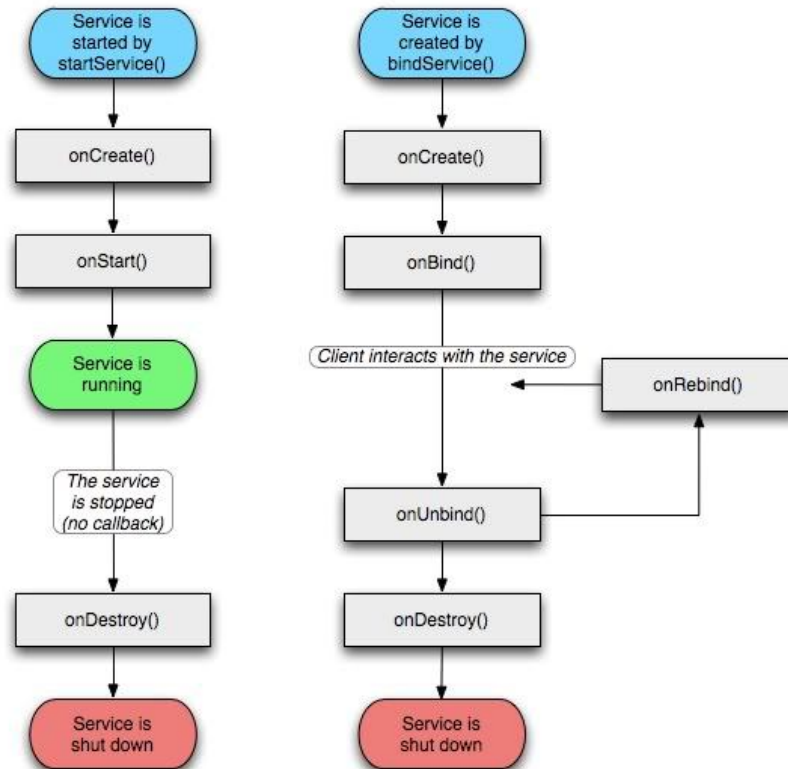
```
void onCreate()  
void onStart(Intent intent)  
void onDestroy()
```

Bằng việc thực hiện những phương thức này, bạn có thể giám sát 2 vòng lặp của chu kỳ thời gian của mỗi Service Entire lifetime của một Service diễn ra giữa thời gian onCreate() được gọi ra và thời gian mà onDestroy() trả lại. Giống như một Activity, một Service lại tiết hành cài đặt ban đầu ở onCreate(), và giải phóng tất cả các tài nguyên còn lại ở onDestroy() Ví dụ, một Service phát lại nhạc có thể tạo ra một luồng và bắt đầu chơi nhạc onCreate(), và sau đó luồng chơi nhạc sẽ dừng lại ở onDestroy(), Active lifetime của một Service bắt đầu bằng một lệnh tới onStart(). Đây là phương thức được chuyển giao đối tượng Intent mà đã được thông qua để tới startService() Service âm nhạc sẽ mở đối tượng Intent để quyết định xem sẽ chơi loại nhạc nào và bắt đầu phát nhạc. Không có callback tương đương nào cho thời điểm Service ngừng lại – không có phương thức onStop(). Các phương thức onCreate() và onDestroy() được gọi cho tất cả các Service dù chúng có được bắt đầu bằng Context.startService() hoặc Context.bindService() hay không. Tuy nhiên, onStart() chỉ được gọi ra đối với các Service bắt đầu bằng startService(). Nếu một Service cho phép những Service khác kết nối với nó thì sẽ có thêm các phương thức callback dành cho Service đó để thực hiện.

```
IBinder onBind(Intent intent)  
boolean onUnbind(Intent intent)  
void onRebind(Intent intent)
```

Hàm callback onBind() thông qua đối tượng Intent đã được truyền đến bindService và onUnbind() được chuyển giao đối tượng mà đã được chuyển đến. Nếu Service đang được chỉ định (binding), onBind() quay trở lại kênh thông tin mà người dùng sử dụng để tương tác với Service. Phương thức onUnbind() có thể yêu cầu onRebind() được gọi nếu một người dùng kết nối với Service.

Biểu đồ dưới đây minh họa cho các phương thức callback giành cho một Service.



Hình 1-22 Chu trình sống của một Service

Mặc dù, nó phân tách các Service được tạo ra thông qua startService với các Service mà được tạo ra bằng bindService(). Hãy nhớ rằng bất kì Service nào, cho dù nó được khởi tạo như thế nào thì nó vẫn có thể cho phép các người dùng kết nối tới nó một cách hiệu quả nhất, cho nên bất kì Service nào cũng có thể được chỉ định thông qua các các phương thức onBind() và onUnbind().

Để hiểu hơn về Service chúng ta hãy làm một ví dụ nhỏ sau:

Đầu tiên, mở file AndroidManifest.xml và tạo một tham chiếu đến class Service

```
<service android:name=".myservice.MyService"/>
```

Tiếp theo, tạo một file MyService.java kế thừa từ class Service:

Trong file MyService.java bắt buộc phải override phương thức:

```
public IBinder onBind(Intent intent);
```

Để có thể start và stop Service thì cũng cần override 2 phương thức là:

```
protected void onCreate();
protected void onDestroy();
```

Thêm một biến toàn cục: `private Timer timer = new Timer();`

Timer thực chất cũng là một Thread. Việc bạn sử dụng Timer và Thread hoàn toàn không có sự khác biệt gì. Biến Timer này sẽ được cài đặt vào bên trong hàm `onCreate` như sau:

```
timer.scheduleAtFixedRate(  
    new TimerTask() {  
        public void run() {  
            //Do something  
        }  
    }, 0, 5000);
```

Khi muốn dừng Service lại thì chỉ cần hủy Thread Timer bằng hàm `timer.cancel();`

Cuối cùng là khởi động Service từ Activity:

```
Intent svc = new Intent(this, MyService.class);
```

```
startService(svc, Bundle.EMPTY);
```

10. TELEPHONY

Telephony là một trong 4 thành phần chính của một hệ thống Android. Nó cho phép người lập trình có thể lấy các thông tin của hệ thống như thông tin SIM, thông tin thiết bị, thông tin mạng,... Ngoài ra, chúng ta cũng có thể cài đặt các thông số cho thiết bị nếu các thông số đó có thể thay đổi được. Tất cả những điều đó được quản lý bởi một class `TelephonyManager` trong Android.

```
TelephonyManager telMan =  
(TelephonyManager) getSystemService(Context.TELEPHONY_SERVICE);
```

Vd:

❖ Lấy thông tin ID thiết bị

```
telMan.getDeviceId();
```

❖ Lấy thông tin số serial SIM

```
telMan.getSimSerialNumber();
```

11. SQLITE

SQLite là một dạng CSDL tương tự như Mysql, PostgreSQL... Đặc điểm của SQLite là **gọn, nhẹ, đơn giản**. Chương trình gồm 1 file duy nhất vốn vẹn chưa đến 500kB, không cần cài đặt, không cần cấu hình hay khởi động mà có thể sử dụng ngay. Dữ liệu database cũng được lưu ở một file duy nhất. Không có khái niệm user, password hay quyền hạn trong SQLite database.

SQLite không thích hợp với những hệ thống lớn nhưng ở quy mô vừa tầm thì SQLite phát huy uy lực và không hề yếu kém về mặt chức năng hay tốc độ. Với các đặc điểm trên SQLite được sử dụng nhiều trong việc phát triển, thử nghiệm v.v.. và là

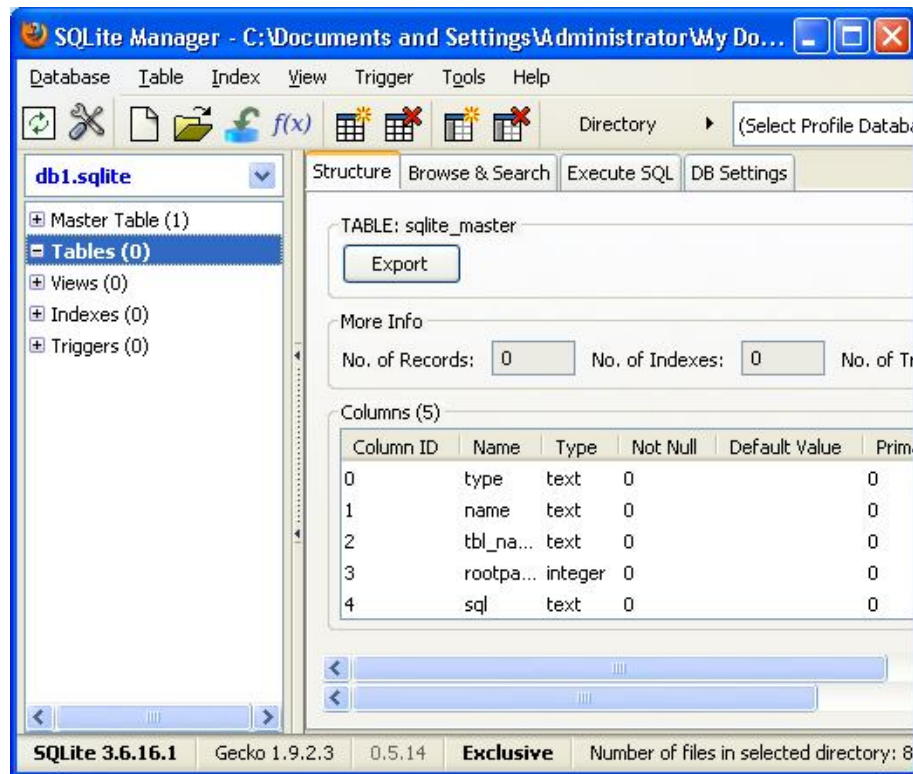
sự lựa chọn phù hợp cho những người bắt đầu học database. Hiện nay thì SQLite đã được ứng dụng vào smartphone như iPhone và Android để lưu trữ dữ liệu.

Để có thể dễ dàng thao tác với SQLite chúng ta có thể sử dụng trình duyệt FireFox và tải về plugin SQLite tại link sau:

<http://code.google.com/p/sqlite-manager/>

Sau khi tải về file xpi, kéo file này vào cửa sổ firefox để cài đặt plugin.

Sau khi cài đặt plugin xong thì vào Menu_tools trong firefox sẽ có chức năng SQLite Manager. Giao diện của SQLite manager trong firefox như sau:



Hình 1-23 SQLite Manager

12. ANDROID & WEBSERVICE

12.1. Khái niệm Web service và SOAP

Webservice là một dịch vụ cung cấp cơ chế triệu gọi các đối tượng từ xa thông qua giao thức HTTP cùng với cơ chế truyền tải định dạng đối tượng theo công nghệ XML. Chính vì sử dụng giao thức HTTP của Web nên giờ đây các lời gọi trở nên đơn giản và thông qua được các rào cản về tường lửa. Để đảm bảo điều này, một giao thức mới là SOAP (Simple Object Access Protocol) ra đời để hỗ trợ cho Web services. SOAP được định nghĩa dựa trên giao thức chuẩn HTTP, SOAP cho phép dữ liệu chuyển đi bằng HTTP và định dạng theo chuẩn XML. Các lời gọi hàm tham số truyền hàm, dữ

liệu trả về từ hàm, tất cả đều được chuyển sang dạng XML và có thể dễ dàng xử lý bởi tất cả các ngôn ngữ. Một thế mạnh khác đó là nếu các đối tượng phân tán xây dựng trên mô hình Web services sẽ có thể triệu gọi lẫn nhau, bất chấp đối tượng đó được viết trên ngôn ngữ Java của Sun hay .NET của Microsoft. Hiện tại, SOAP được coi là một sự thay đổi lớn kể từ khi COM, RMI, CORBA ra đời.

12.2. Giới thiệu về XStream

XStream là một công cụ giúp chuyển các đối tượng hay những thể hiện của những lớp Java qua dạng XML hay ngược lại. Nó là một mã nguồn mở, được thiết lập từ tháng giêng năm 2004.

Trong một đề án IT đôi khi bạn cần phải chuyển các đối tượng của các lớp Java có chứa thông tin và đưa nó qua dạng XML. Việc làm này để giúp mang thông tin từ hệ thống này qua hệ thống khác bằng những gói hay tập tin XML (giả sử các hệ thống này viết bằng ngôn ngữ Java). Nó cũng giúp bạn tránh được nhiều phiền toái như cách sắp đặt chuyển kiểu cho hai dữ liệu giữa hai hệ thống. Do đó dùng dạng XML như là phương tiện trao đổi dữ liệu giữa hai hệ thống là cách hữu hiệu nhất. Sau khi hệ thống đã nhận được dữ liệu nằm ở dạng XML rồi, thì việc kế tiếp là người lập trình chỉ chuyển chúng về các đối tượng Java để phù hợp với ngôn ngữ mà hệ thống đó đang dùng. Công cụ XStream giúp bạn thực hiện được giải pháp vừa nói ở trên. Nếu bạn không dùng XML như là phương tiện trao đổi dữ liệu, thì trong Java cũng có cách đưa đối tượng Java từ nơi này sang nơi khác là dùng Serialize. Bài này không nói đến Serialize, mà chỉ nói đến công cụ XStream. Tất nhiên, ngoài XStream ra cũng có một công cụ nữa có chức năng tương còn được biết đến với cái tên Castor.

Thư viện XStream có thể tải tại <http://xstream.codehaus.org/index.html>

Cách sử dụng thư viện XStream:

❖ Tạo class PhoneNumber:

```
public class PhoneNumber {
    private int code;
    private String number;
    PhoneNumber(int code, String number){
        this.code = code;
        this.number = number;
    }
    public int getCode() {return code;}
    public void setCode(int code) {this.code = code;}
    public String getNumber() {return number;}
    public void setNumber(String number) {this.number = number;}
}
```

❖ Tạo class Person:

```
public class Person {
    private String firstName;
```

```
private String lastName;
private PhoneNumber phone;
private PhoneNumber fax;
Person(String firstName, String lastName){
    this.firstName = firstName;
    this.lastName = lastName;
}
public String getFirstName() {return firstName;}
public void setFirstName(String firstName) {
    this.firstName = firstName;
}
public String getLastName() {return lastName;}
public void setLastName(String lastName) {
    this.lastName = lastName;
}
public PhoneNumber getPhone() {return phone;}
public void setPhone(PhoneNumber phone) {
    this.phone = phone;
}
public PhoneNumber getFax() {return fax;}
public void setFax(PhoneNumber fax) {this.fax = fax;}
}
```

❖ Tạo class TestXStream:

```
import com.thoughtworks.xstream.XStream;
public class TestXStream {
    public static void main(String[] args) {
        XStream xstream = new XStream();
        Person joe = new Person("Joe", "Walnes");
        joe.setPhone(new PhoneNumber(110, "111-111-1111"));
        joe.setFax(
            new PhoneNumber(220, "222-221-2222"));
        String xml = xstream.toXML(joe);
        System.out.println("xml output:\n"+ xml);
        Person newJoe = (Person)xstream.fromXML(xml);
        //Xem thuộc tính trong biến số newJoe thuộc lớp Person
        System.out.println("\nIn ra thông tin của biến newJoe
với First Name, Last Name and Phone:\n ");
        System.out.println("First Name: " + newJoe.getFirstName());
        System.out.println("Last Name: " + newJoe.getLastName());
        System.out.println("Phone: " + newJoe.getPhone().getNumber());
    }
}
```

Kết Quả:

Khi ta cho chạy thử chương trình TestXStream.java, kết quả thu được là một dạng XML được tạo ra mà trong đó nó có cấu trúc chỉ sự liên hệ giữa Person và PhoneNumber như sau:

Cho ra dạng xml là:

```
<Person>
  <firstName>Joe</firstName>
  <lastName>Walnes</lastName>
  <phone>
    <code>110</code>
    <number>111-111-1111</number>
  </phone>
  <fax>
    <code>220</code>
    <number>222-221-2222</number>
  </fax>
</Person>
```

12.3. Thao tác với web service trong Android

Cách gọi hàm từ webservice dotNet trong Android như sau:

Input: các tham số kiểu String

Output: giá trị kiểu String

```
public static Result addUser(int from, String username, String
                                display_name) throws Exception {
    String SOAP_ACTION = "http://tempuri.org/AddUsername";
    String METHOD_NAME = "AddUsername";
    String NAMESPACE = "http://tempuri.org/";
    String URL = "http://10.0.2.2:1217/Service1.asmx?op=AddUsername";
    SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME);

    request.addProperty("from", from);
    request.addProperty("username", username);
    request.addProperty("display_name", display_name);

    SoapSerializationEnvelope envelope =
        new SoapSerializationEnvelope(SoapEnvelope.VERSION1);
    envelope.dotNet = true;
    envelope.setOutputSoapObject(request);
    Trans trans = new Trans(URL);
    trans.call(SOAP_ACTION, envelope);
    SoapPrimitive result = (SoapPrimitive) envelope.getResponse();
    if (result.equals("-1")) {
        return Result.EXCEPTION;
    } else if (result.equals("0")) {
        return Result.FAILED;
    } else {
        return Result.SUCCEEDED;
    }
}
```

Các biến SOAP_ACTION, METHOD_NAME, NAMESPACE, URL để xác định tên phương thức, port mà webservice đang sử dụng,...

Phương thức `addProperty(String var_name, String value)` có 2 tham đối. Tham đối thứ nhất là tên biến cần truyền tham trị vào và tham đối thứ 2 là giá trị của tham biến.

Sau khi thực hiện lệnh gọi hàm `trans.call(SOAP_ACTION, envelope);` thì các giá trị truyền vào sẽ được chuyển đổi thành XML và truyền lên webservice.

Kết quả trả về thông qua đối tượng `SoapPrimitive` hoặc `SoapObject`. Đối với giá trị trả về là một kiểu chuỗi thì có thể thực hiện ép kiểu trực tiếp nhưng còn đối với giá trị trả về là một kiểu danh sách thì đối tượng `SoapObject` cho phép ta có thể duyệt tới từng phần tử trong danh sách. Thực chất đó là một quá trình mã hoá và giải mã một nội dung XML mà đã được SOAP hỗ trợ.

Phương thức dưới đây sẽ mô tả cách nhận về dữ liệu kiểu danh sách từ webservice:

```
public static ArrayList<UserInfo> getListUsername(int userid)throws
Exception {
    String SOAP_ACTION = "http://tempuri.org/GetListUsernameOf";
    String METHOD_NAME = "GetListUsernameOf";
    String NAMESPACE = "http://tempuri.org/";
    String URL =
"http://10.0.2.2:1217/Service1.asmx?op=GetListUsernameOf";
    SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME);
    request.addProperty("userid", userid);
    SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(
        SoapEnvelope.VER11);
    envelope.dotNet = true;
    envelope.setOutputSoapObject(request);
    Trans trans = new Trans(URL);
    trans.call(SOAP_ACTION, envelope);
    SoapObject resultsRequestSOAP = (SoapObject) envelope.bodyIn;
    SoapObject a = (SoapObject) resultsRequestSOAP.getProperty(0);
    int count = a.getPropertyCount();
    ArrayList<UserInfo> bki = new ArrayList<UserInfo>();
    for (int i = 0; i < count; i++) {
        SoapObject so = (SoapObject) a.getProperty(i);
        String us = so.getProperty("display_name").toString();
        String rname = so.getProperty("username").toString();
        String id = so.getProperty("id").toString();
        bki.add(new UserInfo(id, us, rname));
    }
    return bki;
}
```

CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

13. ĐẶC TẢ BÀI TOÁN

Đối với mọi chiếc điện thoại di động Contact là ứng dụng không thể thiếu, dùng để quản lý danh bạ điện thoại, thông tin của những cá nhân cần liên lạc trong danh bạ. Trong Android trên phiên bản mới nhất là 2.1 hiện nay cũng đã có hỗ trợ ứng dụng Contact với nhiều chức năng cơ bản giống các ứng dụng thông thường khác trên các điện thoại di động. Ngoài ra, còn có một số chức năng nổi trội như:

- *Hỗ trợ kết nối với ứng dụng Map, có thể định vị người cần liên lạc thông qua địa chỉ được lưu trong Contact.*
- *Cho phép lưu trữ nhiều số điện thoại, email, address của người dùng trong Contact.*
- *Cho phép đồng bộ hoá giữa các contact trong điện thoại và trong tài khoản Gmail và tài khoản trên mạng xã hội facebook.*

Mặc dù, ứng dụng contact của Android đã cung cấp khá nhiều chức năng nhưng ứng dụng này vẫn còn nhiều hạn chế, đặc biệt thiếu nhiều chức năng cơ bản như quản lý Group, xác lập lại số điện thoại chính hay Email chính. Ngoài ra, với một số người sử dụng thì thông tin lưu trong Contact cũng rất quan trọng. Đề phòng mất điện thoại hay tránh để người khác truy cập vào những thông tin khác được lưu trong điện thoại hoặc khi bị mất các contact do các tác động ngoại cảnh hoặc khi thay số thuê bao. Do đó, ứng dụng PhoneBookSafeManager phát triển trên nền tảng ứng dụng Contact sẵn có trong Android, sẽ đáp ứng nhu cầu bảo mật thông tin cá nhân được lưu trong Contact, đồng thời cải tiến một vài chức năng khác còn hạn chế của ứng dụng Contact trong Android nhằm giúp cho người dùng có thể yên tâm contact của họ được bảo vệ an toàn.

Kết luận: bài toán đặt ra cho chúng ta là: Xây dựng một ứng dụng cho phép người sử dụng điện thoại Android quản lý các contact của họ một cách an toàn, hiệu quả, dễ dàng chia sẻ, dễ dàng quản lý và tìm kiếm.

14. ĐẶC TẢ CHỨC NĂNG

14.1. Upload phone book

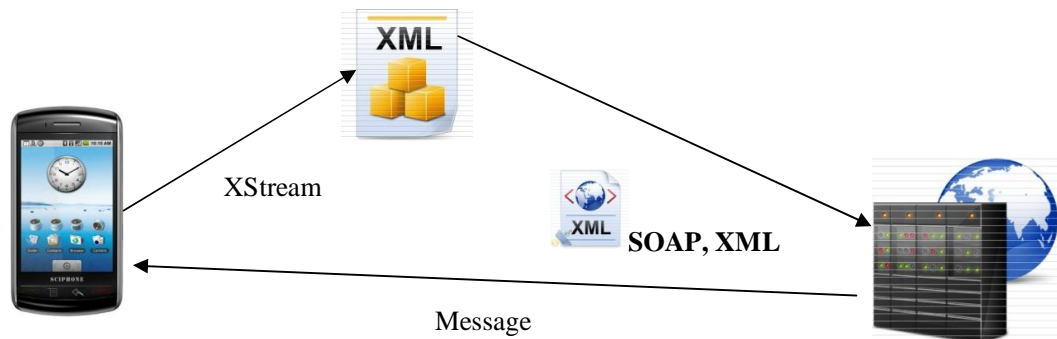
❖ *Mô tả chức năng:*

Chức năng upload phone book cho phép người dùng lưu trữ toàn bộ số contact của họ vào tài khoản online. Tuy nhiên, một số thông tin ít sử dụng sẽ được lược bỏ bớt nhằm giảm kích thước của nội dung XML khi đóng gói.

❖ *Mô tả hoạt động:*

1. Đọc tất cả các thông tin trong phone book và lưu vào một list các contacts.

2. Phân tích list các contact đó thành nội dung XML dựa vào kỹ thuật XStream.
3. Kết nối tới webservice và gửi nội dung XML kèm theo các thông tin tài khoản người dùng cần thiết.
4. Webservice lưu nội dung XML vào đúng tài khoản người dùng.
5. Webservice trả về một thông điệp thông báo là đã lưu hoặc nếu không lưu được.



Hình 2-1. Quá trình upload nội dung XML tới Webserver bằng giao thức SOAP.

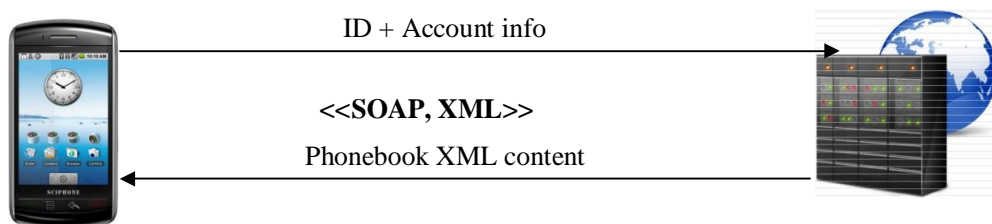
14.2. Revert phone book

❖ *Mô tả chức năng:*

Cho phép người dùng có thể lấy lại các bản phone book mà đã từng upload trước đây.

❖ *Mô tả hoạt động:*

1. Người dùng chọn phiên bản cần upload.
2. Ứng dụng gửi ID của phiên bản và thông tin về tài khoản người dùng tới webservice.
3. Webservice chịu trách nhiệm tìm kiếm dữ liệu tương ứng với ID mà người dùng đã gửi.
4. Ứng dụng nhận dữ liệu trả về từ webservice và phân tích nội dung XML thành một list contact.
5. Cảnh báo người dùng trước khi thay thế phone book.
6. Ghi các dữ liệu trong phiên bản lấy về vào điện thoại.



Hình 2-2. Mô tả quá trình revert phone book

14.3. Send contact

❖ Mô tả chức năng:

Cho phép người dùng gửi một hoặc nhiều contact tới một hoặc nhiều người khác cũng có tài khoản của ứng dụng.

❖ Mô tả hoạt động:

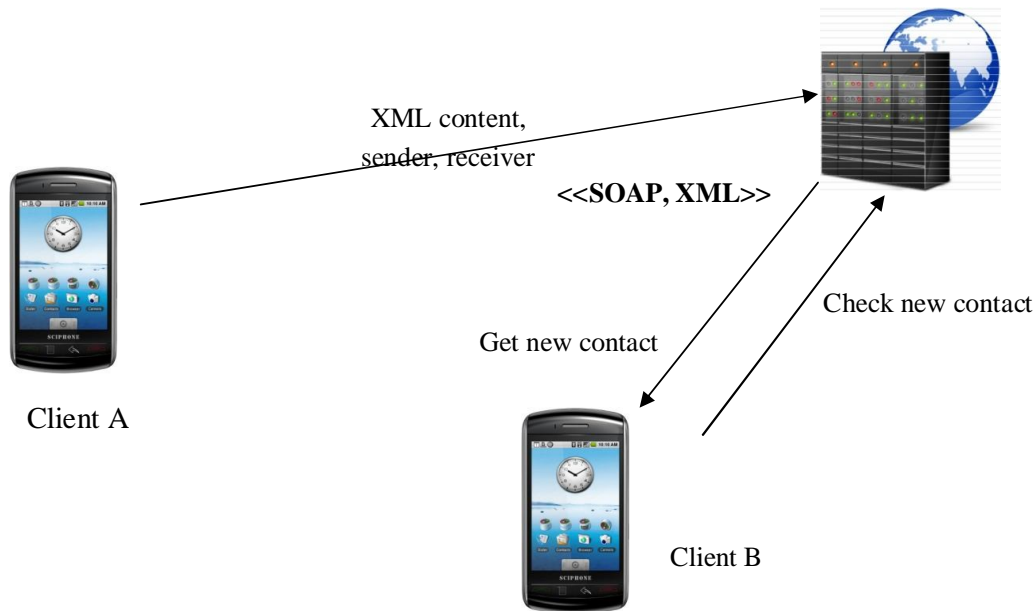
1. Chuyển đổi các contact cần gửi thành nội dung XML.
2. Gửi yêu cầu tới webservice kèm theo nội dung XML, thông tin tài khoản và các thông tin cần thiết khác.

Webservice gửi nội dung XML vào contact box của người nhận và trả về thông điệp nếu gửi thành công.

14.4. Receive contact

🚦 Mô tả chức năng:

Thông báo tới người dùng nếu có ai đó gửi contact tới tài khoản của họ bằng một thông điệp. Đồng thời, người dùng cũng có thể xem, lưu các contact hoặc gỡ bỏ chúng.

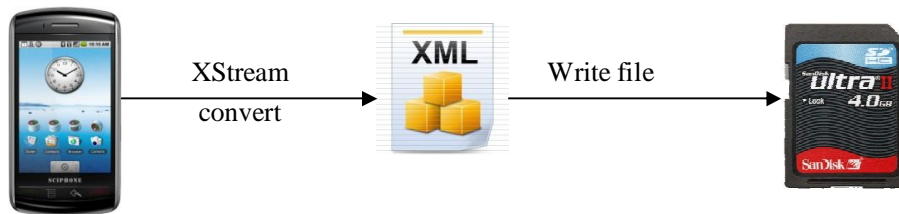


Hình 2-3 Quá trình gửi và nhận contact

14.5. Export contact tới SD Card

🚦 Mô tả chức năng:

Chức năng export to an SD Card cho phép người dùng chia sẻ hoặc backup các contact của họ vào thẻ nhớ dưới dạng XML. Khi người dùng cần chia sẻ họ có thể trao đổi thông qua thẻ nhớ hoặc có thể gửi file XML này đi bất kỳ đâu và người nhận có thể phân tích file XML này thành các contact như cũ.

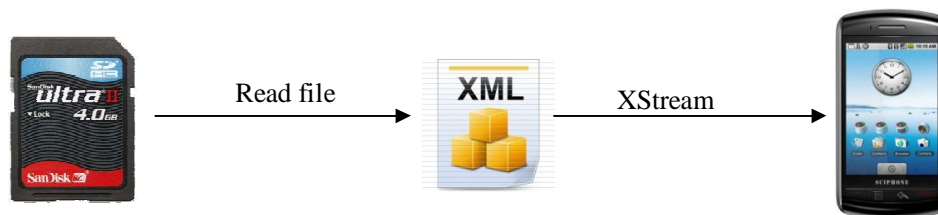


Hình 2-4 Quá trình lưu contact xuống SD Card

14.6. Import contact từ SD Card

✚ Mô tả chức năng:

Ngược lại với chức năng export to SD Card thì chức năng này được dùng để phân tích ngược lại từ một file XML thành các contact.



Hình 2-5 Quá trình đọc contact từ SD Card

14.7. Account Manager

14.7.1.Login

✚ Mô tả chức năng:

Chức năng login không phải được thiết kế để bảo vệ hệ thống mà chủ yếu là sử dụng để quản lý người dùng và dữ liệu người dùng trên server. Khi người dùng login thì username and password sẽ được lưu lại để làm chìa khoá liên lạc với server, đảm bảo rằng dữ liệu được an toàn và dễ dàng truy xuất. Hay nói cách khác mỗi username là một định danh giúp server quản lý được dữ liệu của họ.

14.7.2.Change password

✚ Mô tả chức năng:

Chức năng này giúp người dùng quản lý an toàn hơn tài khoản của họ. Để sử dụng được chức năng này bắt buộc người dùng phải đăng nhập vào tài khoản cũ.

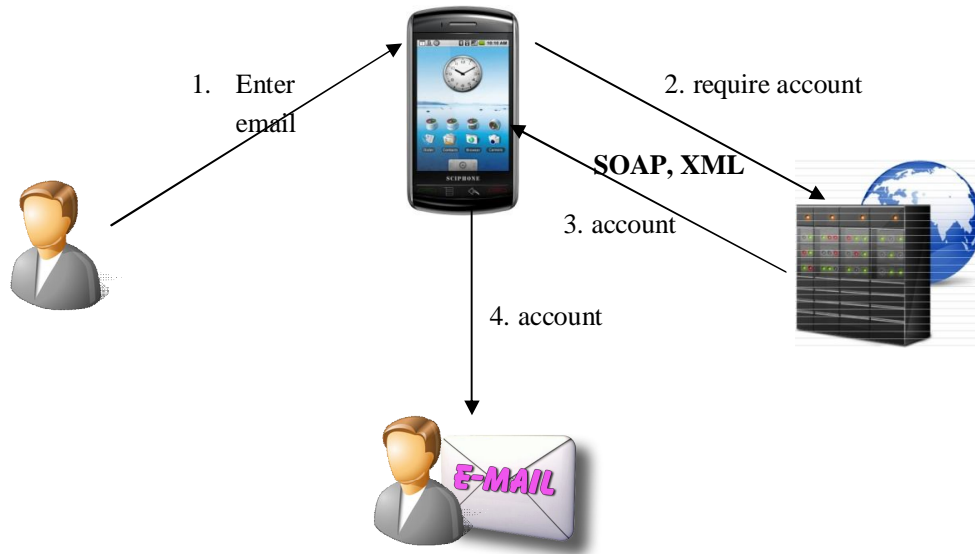
Sau người dùng touch chọn password mới thì ứng dụng sẽ gửi password mới và các thông tin người dùng lên web service và web service sẽ thực hiện việc thay đổi password. Chức năng này sẽ được mô tả kỹ hơn ở chương 2.

14.7.3.Restore account

✚ Mô tả chức năng:

Khi quên username hoặc password thì người dùng có thể sử dụng chức năng này để khôi phục lại tài khoản của họ. Ứng dụng sẽ bắt buộc người dùng phải nhập một email dự phòng trong khi tạo tài khoản để ứng dụng có thể gửi tài khoản của họ vào email đó.

Dưới đây là qui trình khôi phục tài khoản của người dùng:



Hình 2-6 Quá trình khôi phục tài khoản người dùng

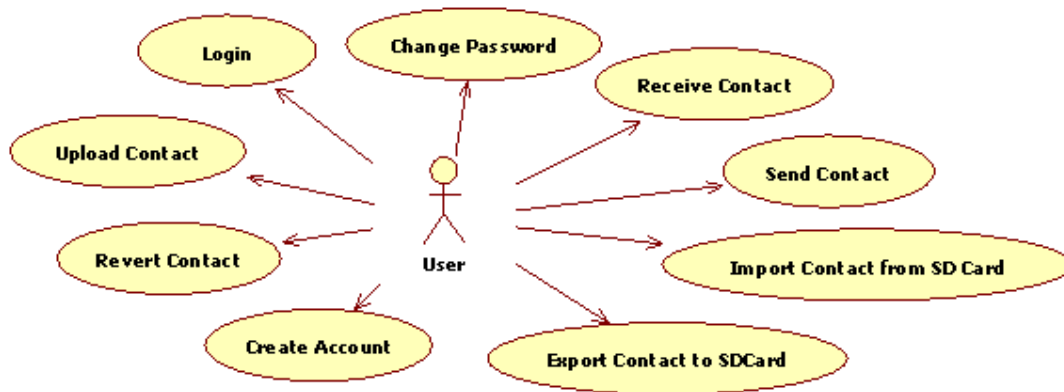
14.7.4.Create account

Thay vì phải tạo tài khoản trên một trang web, điều này sẽ không đảm bảo an toàn và thực sự là một phiên phức không cần thiết. Do đó, ứng dụng cung cấp chức năng tạo tài khoản ngay trên màn hình đăng nhập của ứng dụng nhằm tạo điều kiện thuận lợi nhất đối với người dùng. Để tạo tài khoản người dùng cần nhập các thông tin: Username, Password, Confirm password, Email.

15. CÁC BIỂU ĐỒ HỆ THỐNG

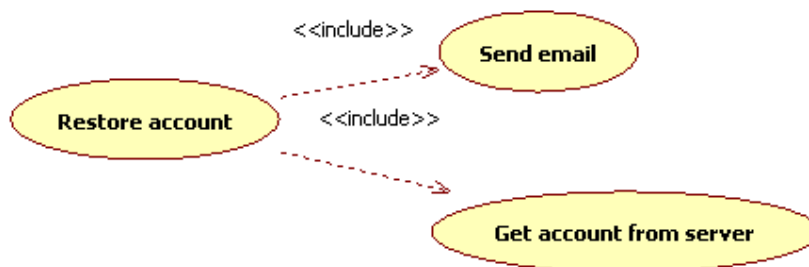
15.1. Biểu đồ Usecase

Mô hình usecase dưới đây thể hiện cho số lượng chức của hệ thống mà người dùng có thể tương tác được trực tiếp.



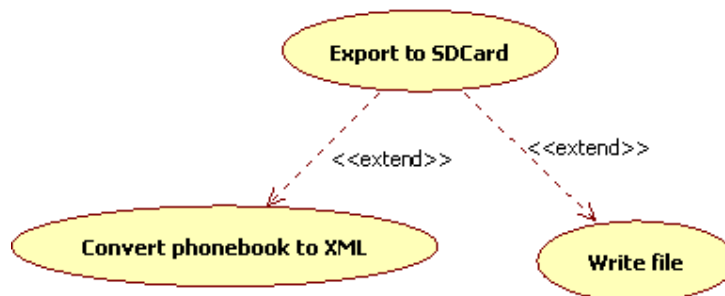
Hình 2-7 Sơ đồ Usecase tổng quát

15.1.1.Restore account



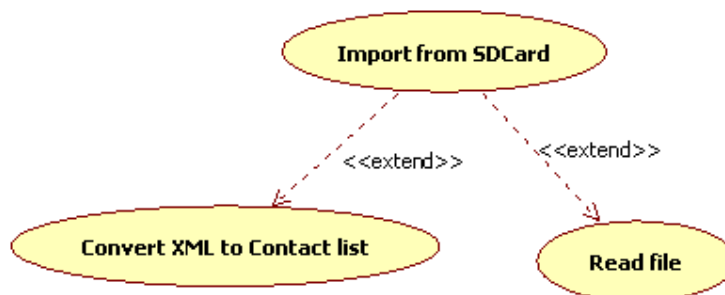
Hình 2-8 Usecase khôi phục tài khoản

15.1.2.Export Contact vào SD Card



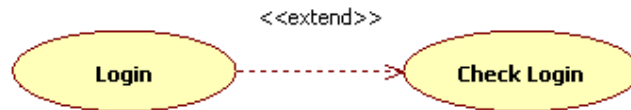
Hình 2-9 Usecase Import contact từ sd card

15.1.3.Import Contact từ SD Card



Hình 2-10 Usecase Import contact từ sd card

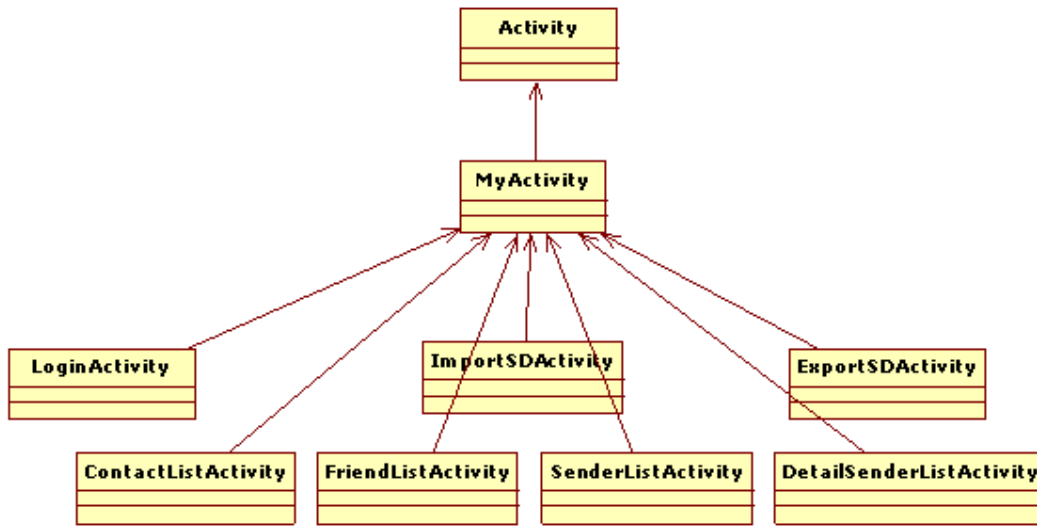
15.1.4.Login



Hình 2-11 Usecase chức năng Login

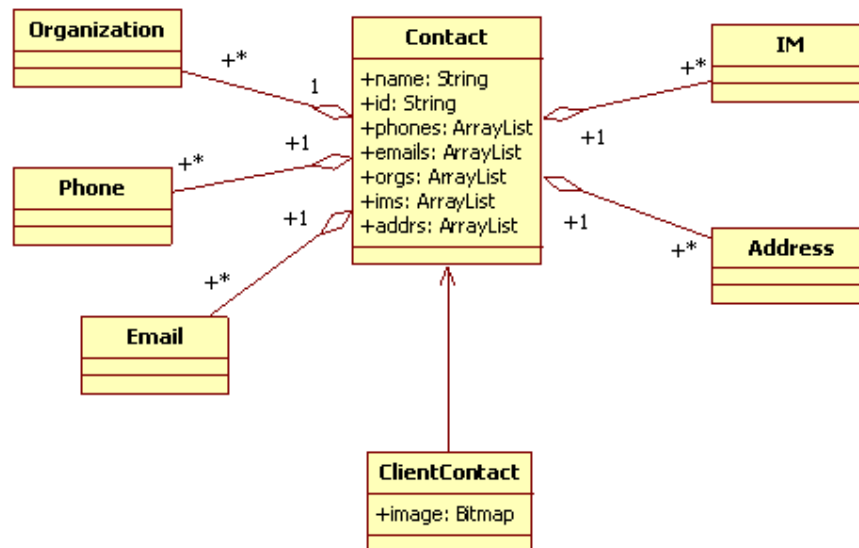
15.2. Biểu đồ Class

15.2.1.Biểu đồ Class giữa các thành phần giao diện



Hình 2-12 Biểu đồ class Contact

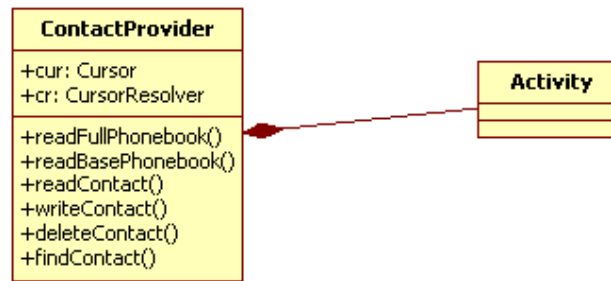
15.3. Biểu đồ Class Contact



Hình 2-13 Biểu đồ class Contact

15.4. Biểu đồ Class ContactProvider

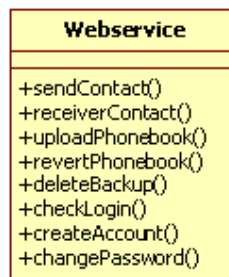
ContactProvider là một Class dùng để đọc/ghi trên phone book.



Hình 2-14 Biểu đồ class ContactProvider

15.5. Class Webservice

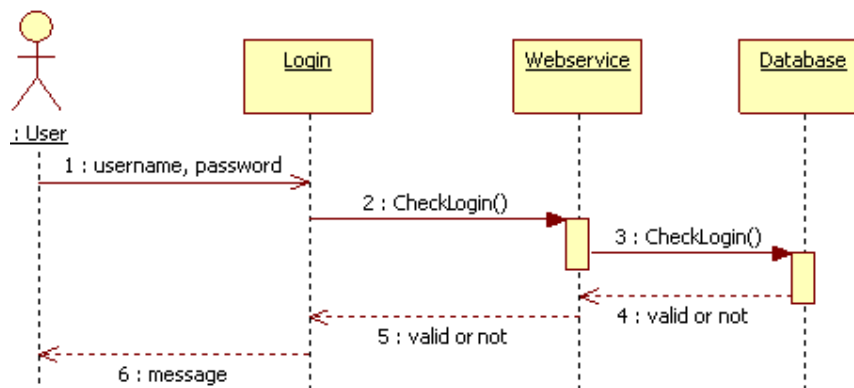
Class Webservice được dùng để kết nối và trao đổi dữ liệu với web service.



Hình 2-15

15.6. Biểu đồ Sequence

15.6.1. Login



Hình 2-16 Biểu đồ sequence chức năng Login

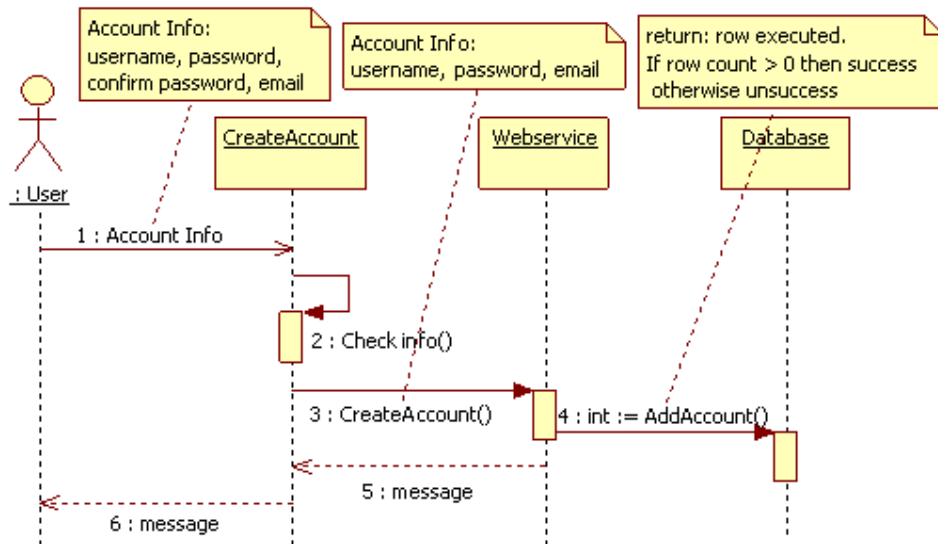
Bước 1: Người dùng nhập username, password và nhấn nút login

Bước 2: Chức năng Login thực hiện triệu gọi hàm `CheckLogin` trên webservice và truyền các tham số username, password vào.

Bước 3: Hàm `CheckLogin` truy vấn vào CSDL để kiểm tra username và password có tồn tại hay không.

Bước 4, 5, 6: Trả về kết quả valid, invalid hoặc exception do lỗi kết nối.

15.6.2. Create new account



Hình 2-17. Biểu đồ tuần tự chức năng tạo tài khoản mới

Bước 1: Người dùng nhập username, password, confirm password, email.

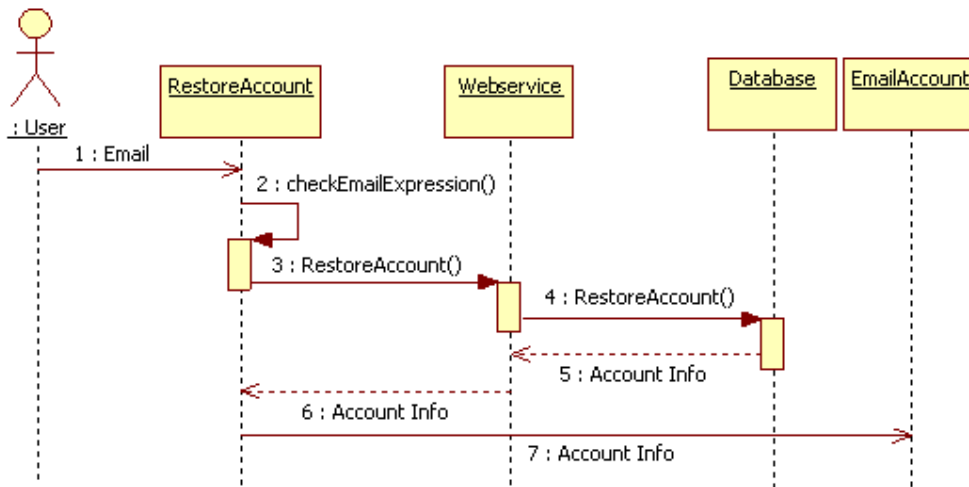
Bước 2: Kiểm tra thông tin nhập vào có hợp lệ hay không.

Bước 3: Triệu gọi hàm tạo tài khoản trên web service. Đồng thời, gửi lên các tham số tương ứng.

Bước 4: Hàm tạo tài khoản trên web service sẽ thực hiện lệnh SQL để insert thông tin tài khoản mới vào.

Bước 5, 6: Trả về kết quả của quá trình tạo tài khoản cho người dùng bằng một message.

15.6.3.Restore account



Hình 2-18. Biểu đồ tuần tự chức năng khôi phục tài khoản

Bước 1: Người dùng nhập địa chỉ email dự phòng đã được nhập trong quá trình tạo tài khoản trước đây.

Bước 2: Kiểm tra biểu thức email có chính xác hay không. Bước này nhằm mục đích giảm thời gian kiểm tra trên server nếu như email không chính xác.

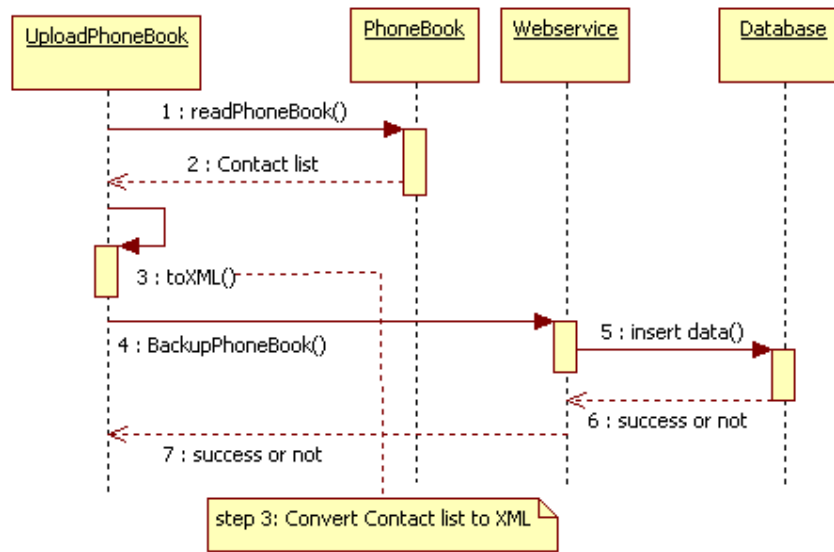
Bước 3: Triệu gọi từ xa hàm RestoreAccount trên web service.

Bước 4: Thực hiện truy vấn để lấy về thông tin tài khoản trong CSDL.

Bước 5, 6: Trả thông tin tài khoản về Client.

Bước 7: Client sẽ thực hiện gửi thông tin tài khoản vào email dự phòng của người dùng. Người dùng sẽ truy cập email dự phòng của họ để xem lại thông tin tài khoản và có thể đăng nhập lại để đổi thông tin tài khoản.

15.6.4.Upload phone book



Hình 2-19. Biểu đồ tuần tự chức năng upload phone book

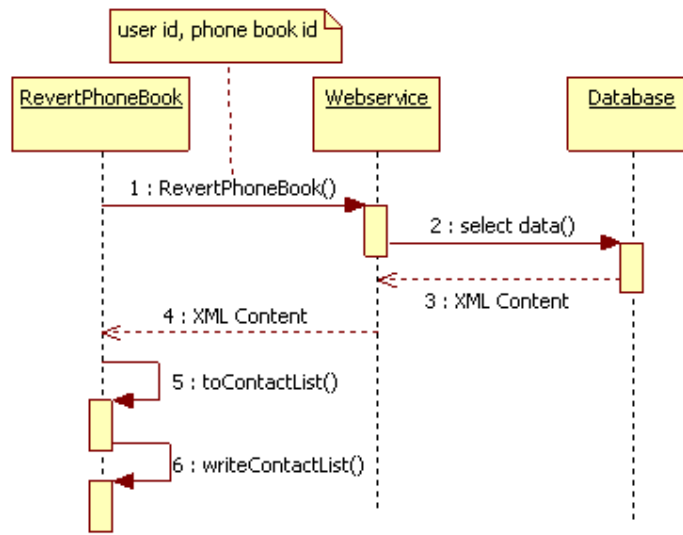
Bước 1,2 : Đọc tất cả các Contact trong phone book và lưu vào một list

Bước 3: Chuyển đổi Contact list thành XML

Bước 4, 5: Triệu gọi hàm BackupConatct trên Webservice và truyền vào các tham số là các thông tin upload, UserID, hàm này sẽ thực hiện chèn dữ liệu vào CSDL tương ứng với tài khoản của người dùng.

Bước 6, 7: Trả về kết quả là đã upload thành công hay không.

15.6.5.Revert phone book



Hình 2-20. Biểu đồ tuần tự chức năng khôi phục phone book

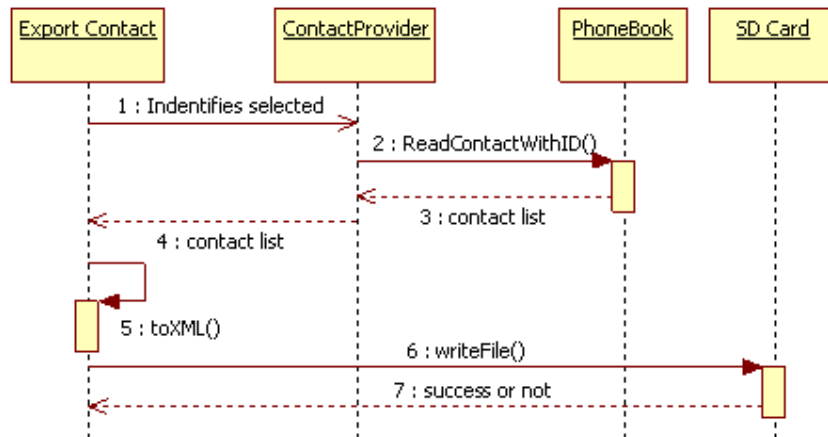
Bước 1,2 : Đọc tất cả các Contact trong phone book và lưu vào một list

Bước 3: Chuyển đổi Contact list thành XML

Bước 4, 5: Triệu gọi hàm BackupConatct trên Webservice và truyền vào các tham số là các thông tin upload, UserID, hàm này sẽ thực hiện chèn dữ liệu vào CSDL tương ứng với tài khoản của người dùng.

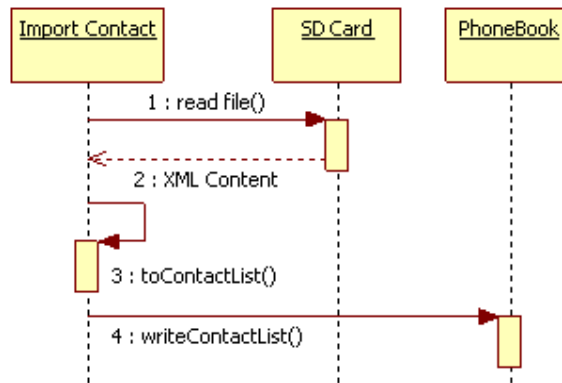
Bước 6, 7: Trả về kết quả là đã upload thành công hay không.

15.6.6.Export Contact to SD Card



Hình 2-21. Biểu đồ tuần tự chức năng Export contact vào sd card

15.6.7.Import Contact from SD Card



Hình 2-22. Biểu đồ tuần tự chức năng import contact từ sd card

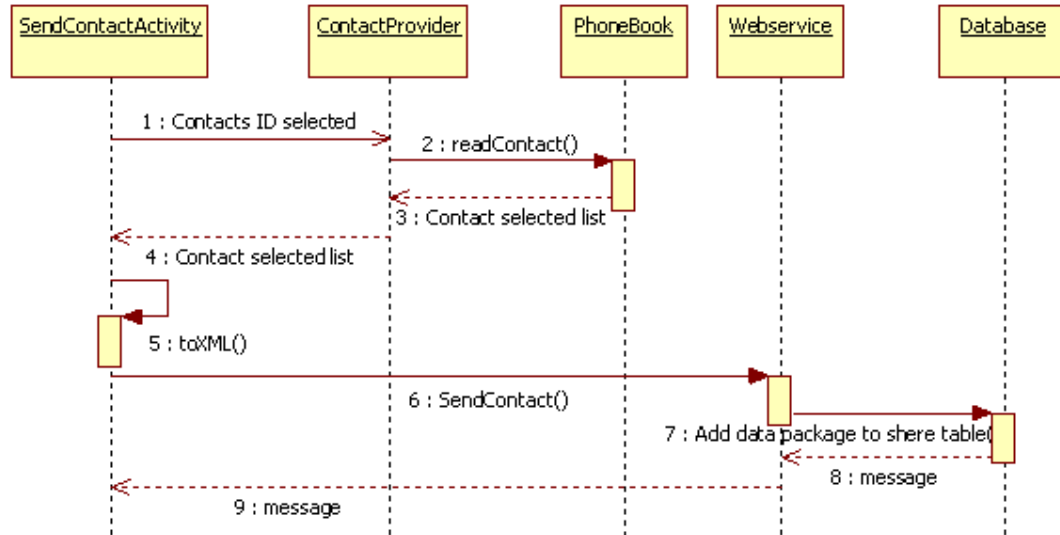
Bước 1, 2: Đọc file từ SD Card.

Bước 3, 4: Convert nội dung XML đã được đọc từ file thành Contact list và ghi các Contact đó vào phone book trong điện thoại Android.

15.6.8.Send Contact

Biểu đồ sequence send Contact thể hiện ở mức hệ thống quá trình gửi các Contact tới một tài khoản nào đó cũng được quản lý bởi hệ thống.

Để tiết kiệm bộ nhớ, khi hiển thị một Contact list nào đó thì chỉ những thông tin cần hiển thị được load vào bộ nhớ còn những thông tin khác thì không được load. Vì vậy, để gửi các Contact tới một người dùng nào đó ứng dụng sẽ thực hiện qua các bước như biểu đồ được mô tả chi tiết sau:



Hình 2-23. Biểu đồ tuần tự chức năng send contact

Bước 1,2,3,4: Đọc các thông tin đầy đủ của các Contact đã được chọn và trả về một Contact list.

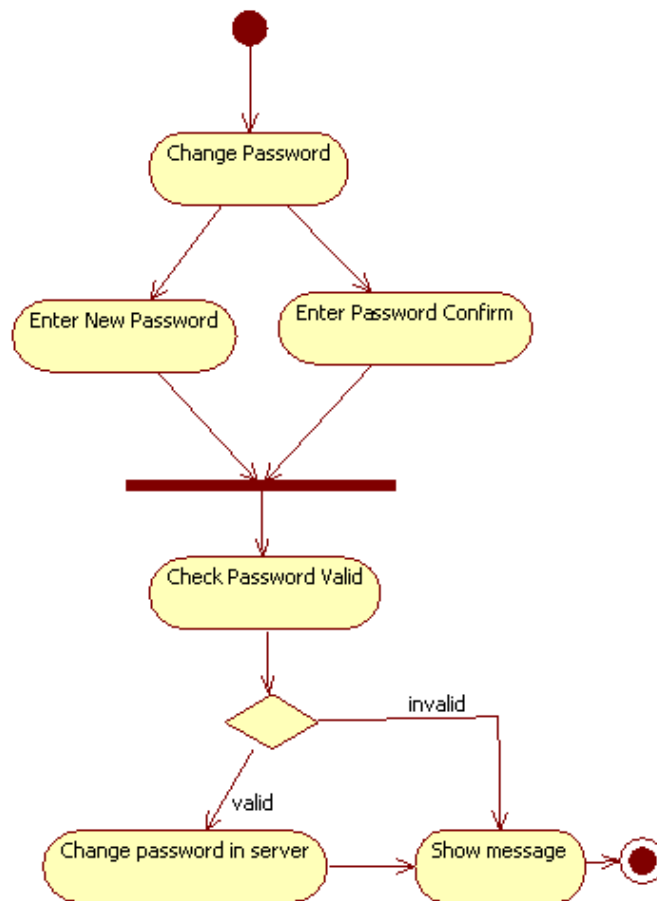
Bước 5,6: Convert Contact list thành nội dung XML và triệu gọi từ xa hàm SendContact từ web service. Đồng thời truyền vào các tham số là nội dung cần gửi và các thông tin cần thiết như người gửi, người nhận, ngày gửi,...

Bước 7: Add nội dung gửi và các thông tin gửi vào một table. Từ table này tất cả các người dùng sẽ truy cập vào và lấy các dữ liệu của họ.

Bước 8,9: Trả về một thông điệp thông báo cho người dùng biết là đã gửi được hay chưa cũng như các trường hợp lỗi.

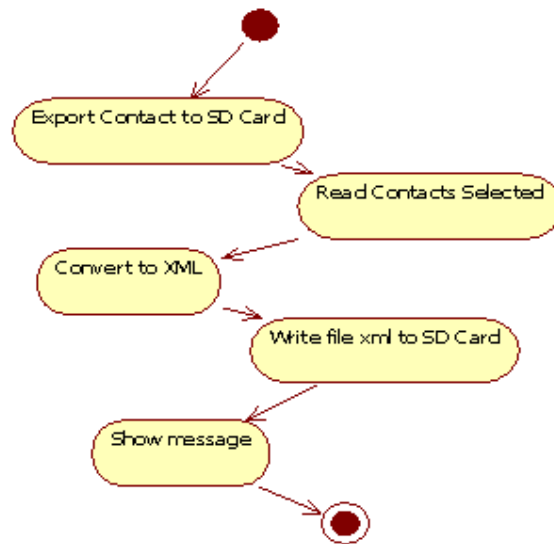
15.7. Biểu đồ Activity

15.7.1.Change password



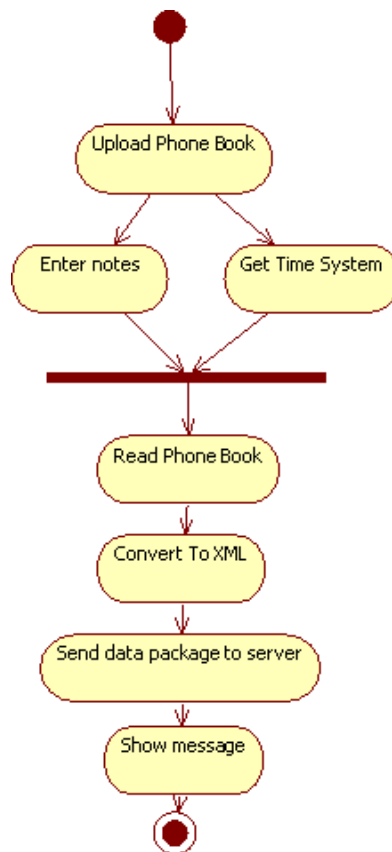
Hình 2-24

15.7.2.Export Contact to SD Card



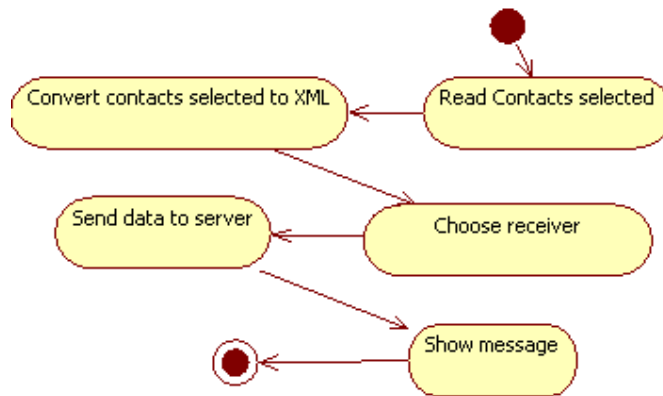
Hình 2-25

15.7.3.Upload phone book



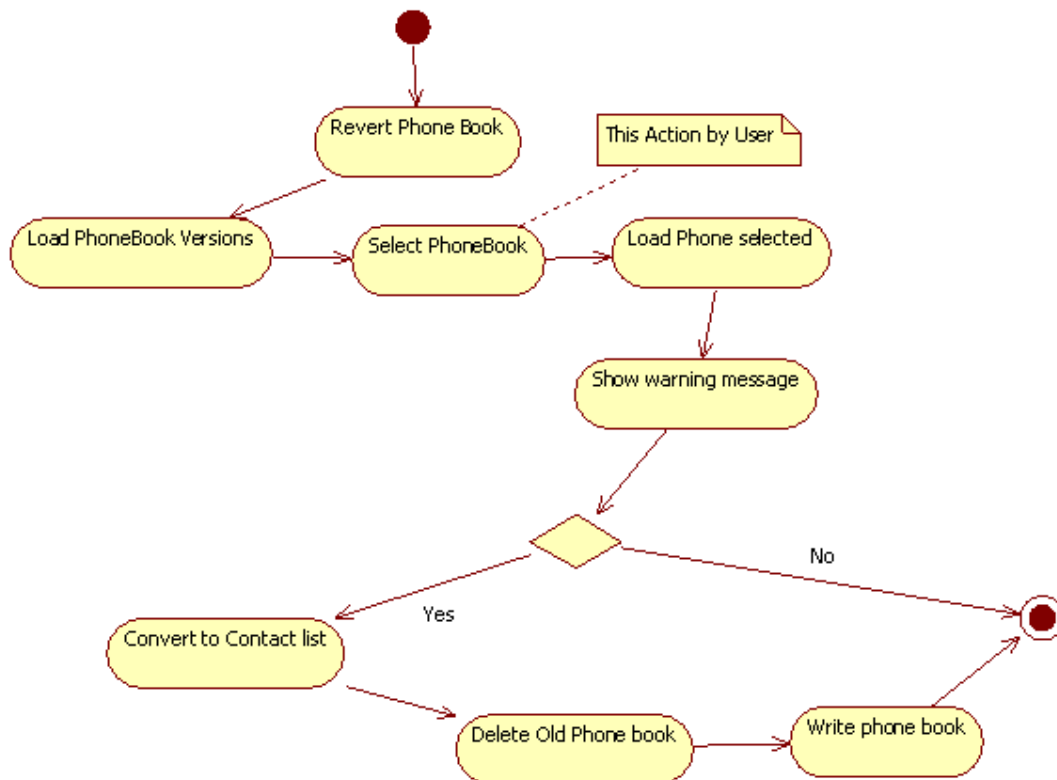
Hình 2-26

15.7.4.Send Contact



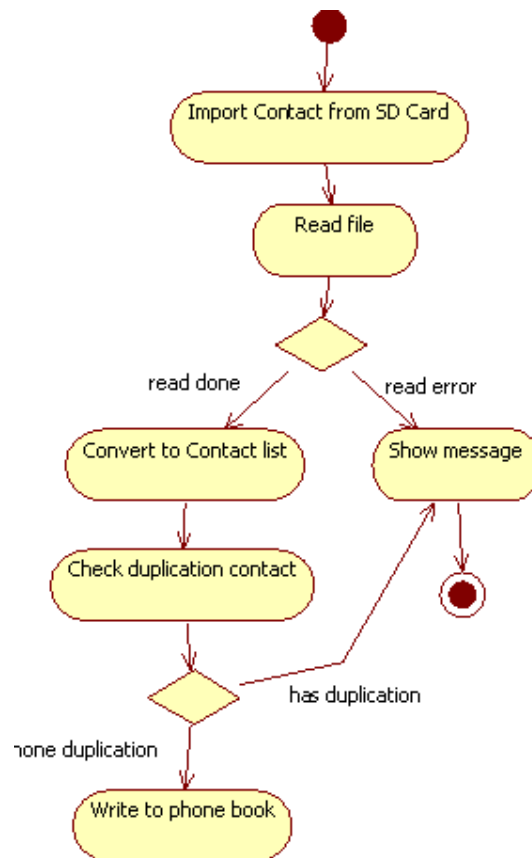
Hình 2-27

15.7.5.Revert phone book



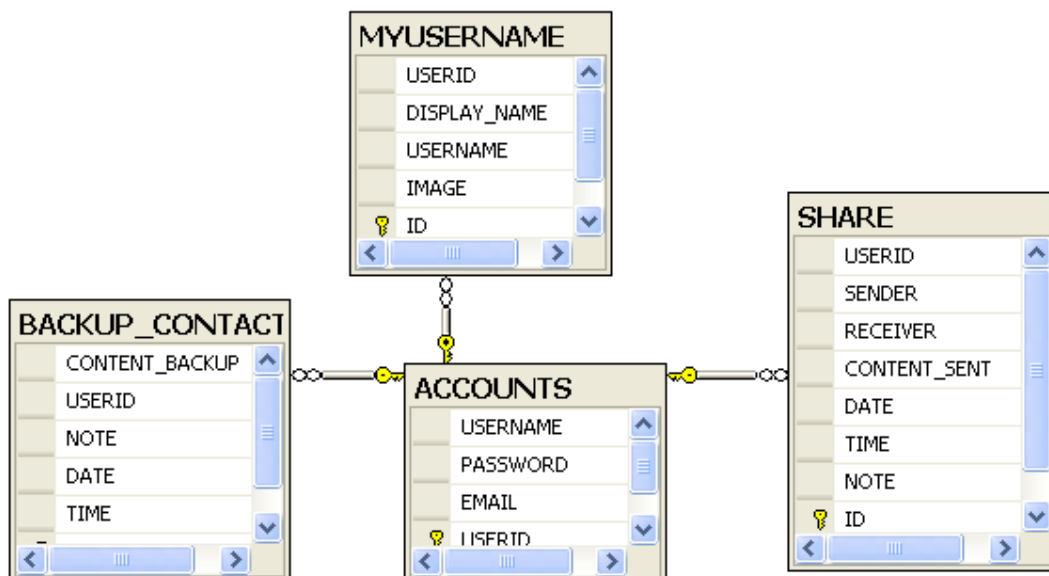
Hình 2-28

15.7.6.Import Contact from SD Card



Hình 2-29

16. CƠ SỞ DỮ LIỆU WEB SERVICE



Hình 2-30

❖ **Bảng ACCOUNTS:**

Được dùng để lưu trữ thông tin tài khoản người dùng.

Thuộc tính của các field như sau:

Field Name	Type
USERNAME	nvarchar(50)
PASSWORD	nvarchar(50)
EMAIL	nvarchar(100)
USERID	bigint, primary key

Bảng 2-1

❖ **Bảng BACKUP_CONTACT**

Được dùng để lưu trữ các phiên bản phone book của người dùng

Thuộc tính:

Field Name	Type
CONTENT_BACKUP	xml
USERID	bigint
NOTE	nvarchar(MAX), allow null
DATE	nvarchar(30)
TIME	nvarchar(30)
VERSION_ID	bigint, primary key

Bảng 2-2

Bảng MYUSERNAME

Được dùng để lưu trữ tên tài khoản của những người bạn có sử dụng hệ thống phone book safe. Cũng giống như khi chat yahoo hoặc skype thì đều có một contact list nhằm giúp bạn dễ dàng quản lý và tìm kiếm khi cần.

Thuộc tính:

Field Name	Type
USERID	bigint
DISPLAY_NAME	nvarchar(100), allow null
USERNAME	nvarchar(50)

IMAGE	image, allow null
ID	bigint

Bảng 2-3

❖ **Bảng SHARE**

Khi một user nào đó gửi dữ liệu tới một người khác thì dữ liệu đó kèm các thông tin liên quan sẽ được lưu vào bảng này. Các ứng dụng sẽ tự động truy cập vào bảng này để lấy dữ liệu của user tương ứng.

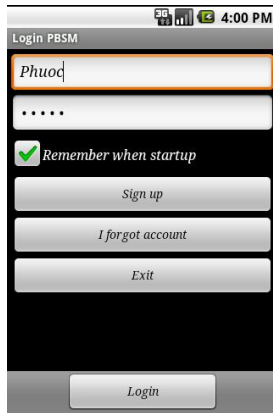
Thuộc tính:

Field Name	Type
USERID	bigint
SENDER	nvarchar(50)
RECEIVER	nvarchar(50)
CONTENT_SENT	xml
DATE	nvarchar(30)
TIME	nvarchar(30)
NOTE	nvarchar(MAX), allow null
ID	bigint

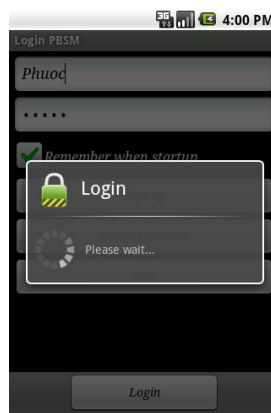
Bảng 2-3

CHƯƠNG 3: DEMO ỨNG DỤNG

17. MỘT VÀI HÌNH ẢNH ỨNG DỤNG



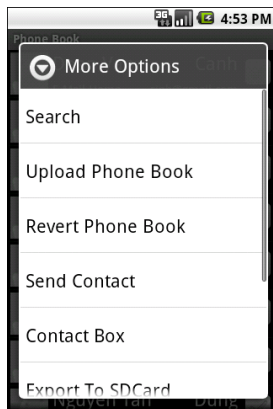
Hình 3-1
Login



Hình 3-2
Đang Login



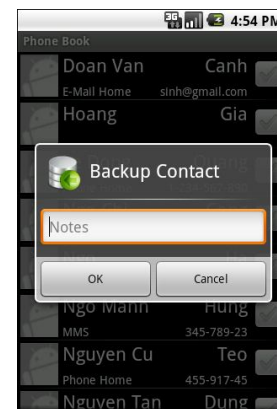
Hình 3-3
Sau Login



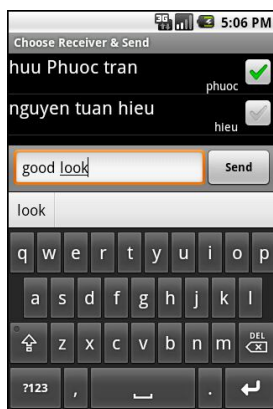
Hình 3-4
Các chức năng



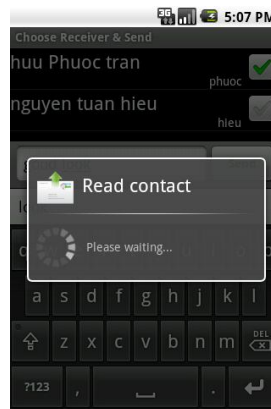
Hình 3-5
Revert phonebook



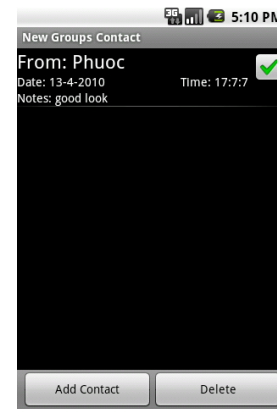
Hình 3-6
Upload phonebook



Hình 3-7
Send Contact



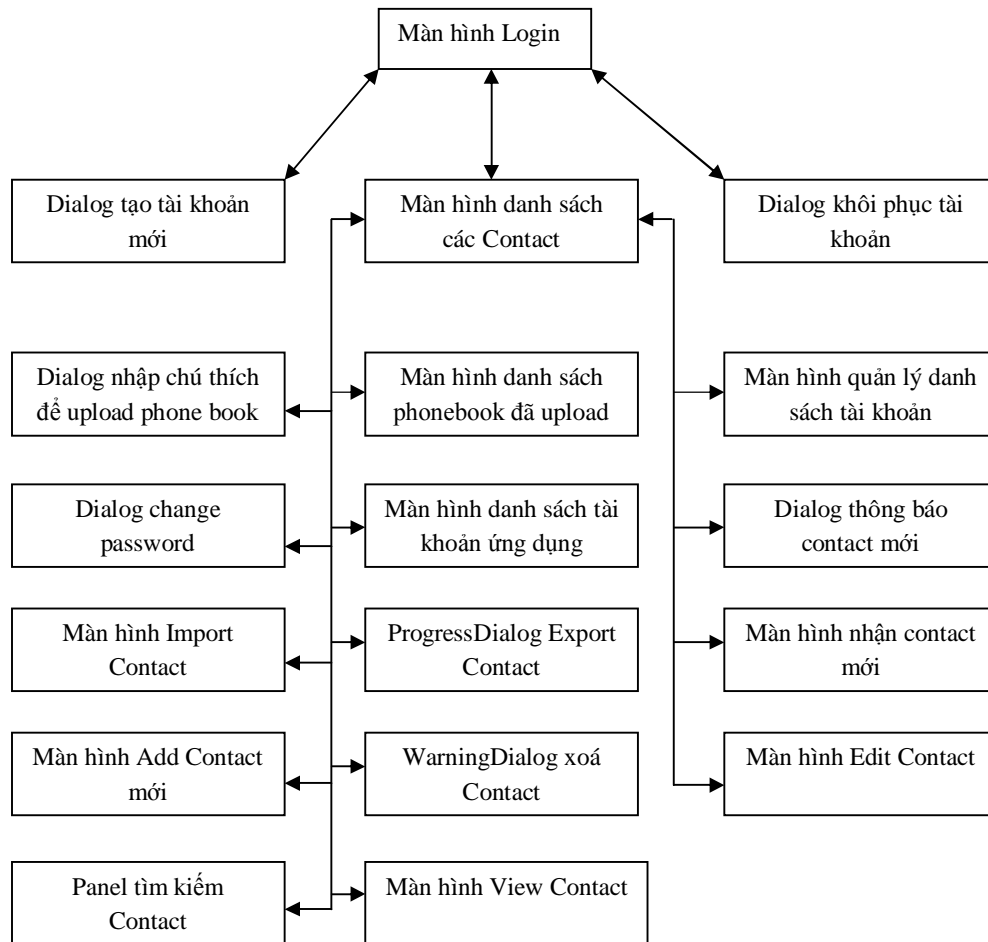
Hình 3-8
Đọc Contact để gửi



Hình 3-9
Nhận Contact

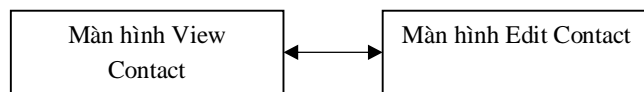
18. SƠ ĐỒ LIÊN KẾT CÁC MÀN HÌNH VÀ DIALOG

18.1. Danh sách các liên kết màn hình mức 1&2

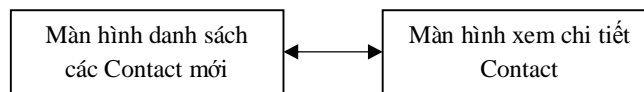


18.2. Danh sách liên kết các màn hình mức 3

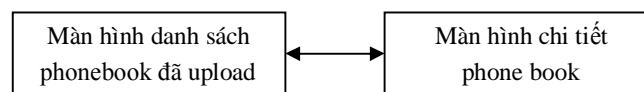
18.2.1. Màn hình View Contact



18.2.2. Màn hình danh sách các Contact mới



18.2.3. Màn hình danh sách phone bool đã upload



19. ĐÁNH GIÁ ỨNG DỤNG

Mục tiêu đã đạt được

Các chức năng chính	Hoàn thành
Upload Contact	100%
Revert Contact	100%
Export Contact to SDCard	100%
Import Contact from SDCard	100%
Send Contact	100%
Receive Contact	100%
Change Password	100%
Create new account	100%
Friends Android manager	100%

Các issue và bug chưa giải quyết

- ✓ Chưa upload được hình ảnh lên server (issue)

Lý do: Trong Android tất cả các hình ảnh khi đọc vào bộ nhớ đều được mã hoá thành đối tượng Bitmap. Đối tượng này không thể tiếp tục mã hoá để lưu vào file XML. Mặt khác, webservice là một kỹ thuật cho phép triệu gọi các hàm từ xa, liên lạc giữa client và server đều thông qua XML. Do đó, các đối số phức tạp hoặc kiểu Stream đều không được chấp nhận.

Hướng giải quyết: Sử dụng kỹ thuật JSON. Đây cũng là một kỹ thuật giúp liên lạc giữa client với web service. Tuy nhiên, kỹ thuật này cho phép chúng ta duy trì được kết nối với server thông qua giao thức http. Vì vậy, dữ liệu có thể được truyền lên server theo Stream.

- ✓ ListView cuộn không mượt (issue)

Lý do: Mỗi lần xảy ra sự kiện cuộn thì Adapter sẽ refresh liên tục và đó cũng là lúc ứng dụng chiếm nhiều tài nguyên CPU nhất. Điều này làm cho ListView bị giật mỗi lần cuộn.

Hướng giải quyết: chưa có.

20. HƯỚNG PHÁT TRIỂN TRONG TƯƠNG LAI

- ❖ Chức năng gửi và nhận tin nhắn thông qua wifi

Chức năng này sẽ được phát triển trong tương lai nhằm mục đích giúp những người thường xuyên có kết nối wifi có thể gửi tin nhắn mà không mất tiền. Tin nhắn đến sẽ được thông báo bằng dialog ngay trên màn hình desktop và kể cả khi đang sử dụng ứng dụng khác.

❖ Chức năng gửi thư thoại thông qua wifi

Chức năng này sẽ tận dụng chức năng thu âm được trang bị trong các máy Android. Dữ liệu sẽ được đóng gói và truyền lên tài khoản người nhận.

❖ Chức năng huỷ dữ liệu

Khi bị mất điện thoại thì điều mà người dùng lo lắng nhất chính là những dữ liệu cá nhân trong đó. Chức năng này sẽ tự động huỷ các dữ liệu được chỉ định sẵn trong thẻ nhớ hoặc bộ nhớ máy khi nhận được yêu cầu thông qua tin nhắn hoặc wifi. Ngoài ra người dùng còn có thể thiết lập cơ chế huỷ dữ liệu tự động khi không được thực hiện một sự kiện nào đó trong một khoảng thời gian nào đó.

21. KẾT LUẬN

Sau khi thực hiện xong project này em đã nắm bắt được qui trình làm một dự án phần mềm trên mobile nói chung đồng thời hiểu thêm được nhiều kiến thức về Android và các công nghệ liên quan như là XML, XStream, Web service và SOAP,...

TÀI LIỆU THAM KHẢO

- [1] **Copyright © 2008 by Chris Haseman.** *Android Essentials.* Appres[®]
- [2] **Copyright © 2009 by Mark L. Murphy.** *Beginning Android.* Appress[®]
- [3] **Copyright © 2009 by Sayed Y. Hashimi and Satya Komatineni.** *Pro Android.* Appress[®]
- [4] **Copyright © 2008 by The McGraw-Hill Companies.** *Android™ A Programmer's Guide*[®]
- [5] **Copyright © 2008 by Ed Burnette.** *Hello, Android. (Introducing Google's Mobile Development Platform)*
- [6] **Copyright © 2009 by W. FRANK ABLESON, CHARLIE COLLINS, ROBI SEN.** *Unlocking Android*

<http://www.google.com.vn>