MỤC LỤC

LỜI CẢM ƠN

Em xin bày tỏ lòng biết ơn sâu sắc nhất tới thầy giáo ThS.Trần Ngọc Thái, thầy đã tận tình hướng dẫn và giúp đỡ em trong suốt quá trình làm tốt nghiệp. Với sự chỉ bảo của thầy, em đã có những định hướng tốt trong việc triển khai và thực hiện các yêu cầu trong quá trình làm đồ án tốt nghiệp.

Em xin chân thành cảm ơn sự dạy bảo và giúp đỡ của các thầy giáo, cô giáo Khoa Công Nghệ Thông Tin – Trường Đại Học Dân Lập Hải Phòng đã trang bị cho em những kiến thức cơ bản nhất để em có thể hoàn thành tốt báo cáo tốt nghiệp này.

Xin cảm ơn tới những người thân trong gia đình đã quan tâm,động viên trong suốt quá trình học tập và làm tốt nghiệp.

Xin gửi lời cảm ơn tới tất cả bạn bè, đặc biệt là các bạn trong lớp CT1102 đã giúp đỡ và đóng góp ý kiến để tôi hoàn thành chương trình.

Em xin trân thành cảm ơn!

Hải Phòng,ngày 24 tháng 12 năm 2012

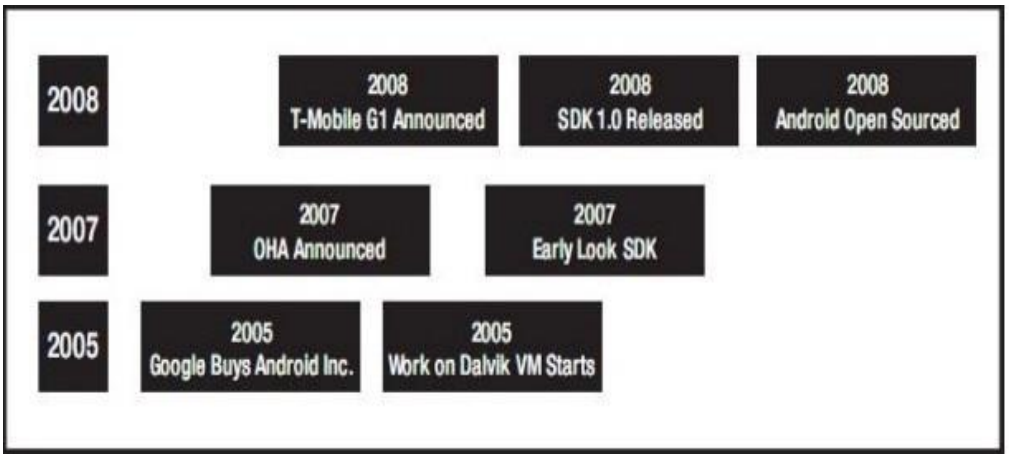
Sinh viên

Phạm Hùng Nam

CHƯƠNG 1. GIỚI THIỆU HỆ ĐIỀU HÀNH ANDROID

1.1. Lịch sử Hệ điều hành 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 thong và thiết bị cầm tay như:

TexasInstruments,BroadcomCorporation,Google,HTC,Intel,LG,Marvell,TechnologyGroup,Motorola,Nvidia,Qualcomm,SamsungElectronics,Sprint Nextel,T-Mobile,ARM Holdings,Atheros Communications,Asustek Computer Inc,Garmin Ltd,Softbank,Sony Ericsson,ToshibaCorp,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.

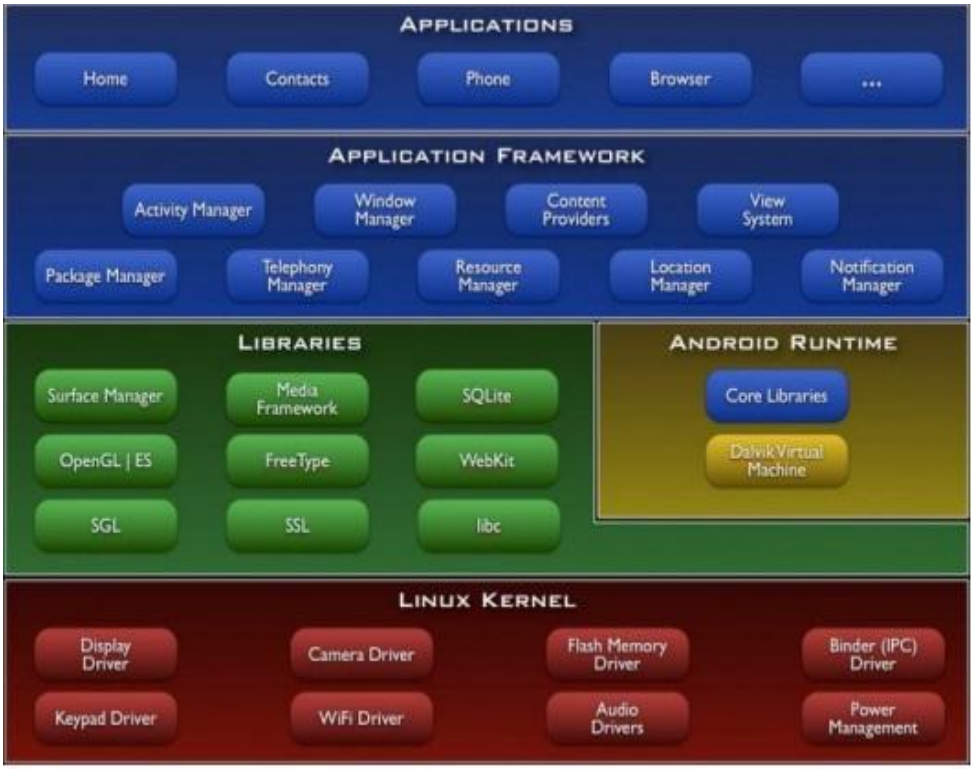
1.2. DELVING với máy ảo DALVIK

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.

1.3. 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

1.3.1. 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.

1.3.2. 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 "Notifycation Manager" cho phép tất cả các ứng dụng hiển thị các custom alerts trong status bar. Activity Maanager được dùng để quản lý chu trình sống của ứng dụng và điều hướng các activity.

1.3.3. Library

Android bao gồm một tập hợp các thư viên 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:

- Hệ thống thư viện C: một BSD có nguồn gốc từ hệ thống thư viện tiêu chuẩn C (libc), điều chỉnh để nhúng vào các thiết bị dựa trên Linux
- Thư viện Media – dựa trên PacketVideo's OpenCORE;các thư viện hỗ trợ phát lại và ghi âm của âm thanh phổ biến và các định dạng video, cũng như các tập tin hình ảnh tĩnh,bao gồm cả MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG
- Bề mặt quản lý– Quản lý việc truy xuất vào hệ thống hiển thị
- LibWebCore- một công cụ trình duyệt web hiện đại mà quyền hạn cả hai trình duyệt web Android và xem web nhúng.
- SGL- Đồ họa 2D cơ bản của máy.
- Thư viện 3D – một thực hiện dựa vào OpenGL ES 1.0 APIs; các thư viện sử dụng phần cứng tăng tốc 3D (nếu có), tối ưu hóa cao rasterizer phần mềm 3D.
- FreeType- vẽ phông chữ bitmap và vector**.**

SQLite một công cụ cơ sở dữ liệu quan hệ mạnh mẽ và nhẹ có sẵn cho tất cả các ứng dụng.

1.3.4. 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.

1.3.5. 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.

1.4. ANDROID EMULATOR

Android SDK và Plugin Eclipse được gọi là một Android Deverloper Tool (ADT).Các Android coder sẽ cần phải sử dụng công cụ IDE (Integrated Development Enveronment) 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

CHƯƠNG 2. PHÁT TRIỂN ỨNG DỤNG TRÊN ANDROID

2.1. Các thành phần trong một ANDROID PROJECT

2.1.1. AndroidManifest.xml

Trong bất kì 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.

2.1.2. 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 kì một sự kiện nào xảy xa 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 xoá một file hình ảnh thì đường dẫn tương ứng đến hình ảnh đó cũng tự động bị xoá.

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.k12tpm.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;
}
}
```

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

Bao nhiêu vùng nhớ chiếm lĩnh trong hệ thống.

2.2.1. Chu kỳ sống thành phần

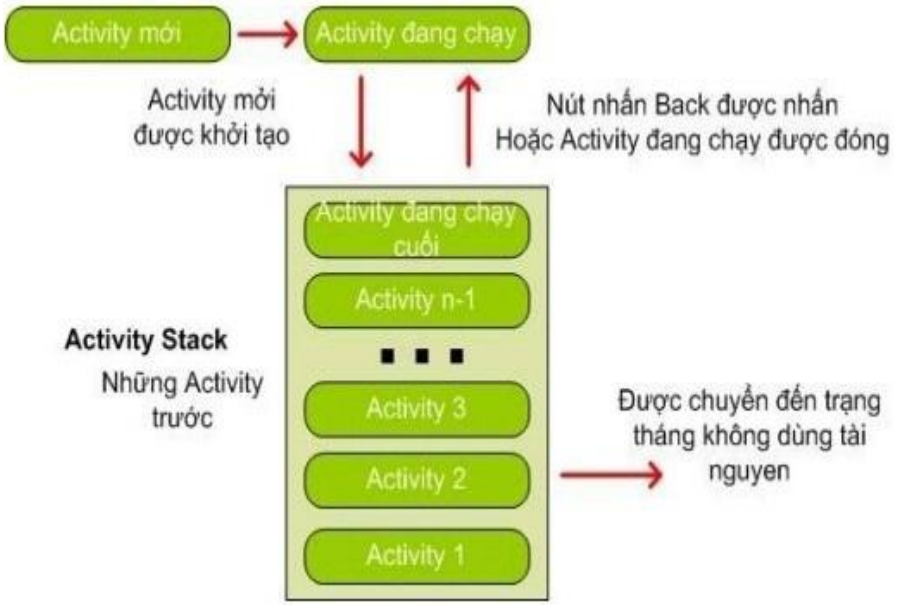
Các thành phần ứng dụng có một chu kỳ sống, tức là mỗi thành phần từ lúc bắt đầu khởi tạo và đến thời điểm kết thúc. Giữa đó, đôi lúc chúng có thể là active hoặc inactive, hoặc là trong trường hợp activies nó có thể visible hoặc invisible

Hình 2.1

2.2.2. Activity Stack

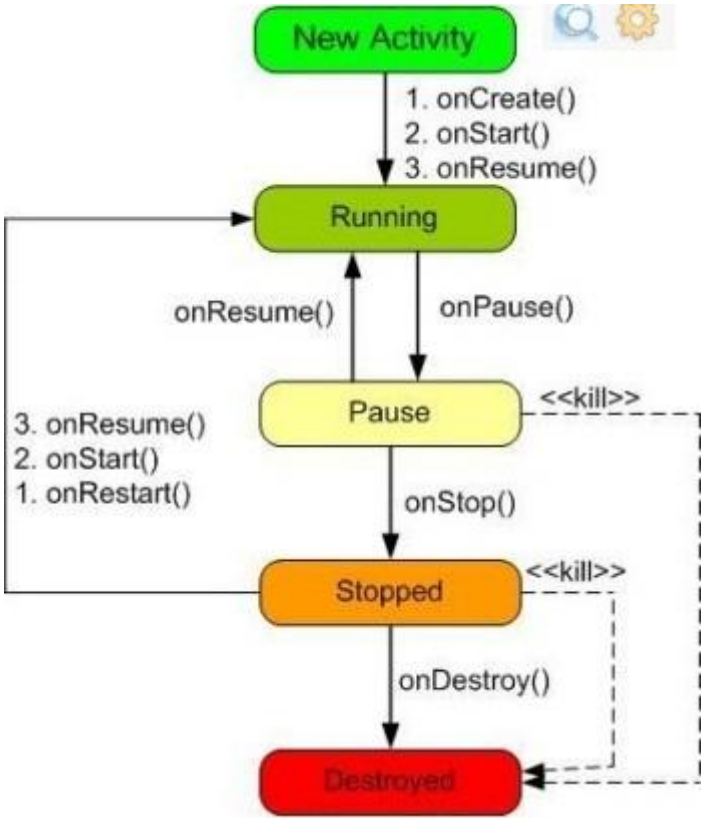
Bên trong hệ thống các activity được quản lý như một activity stack.Khi một Activity mới được start, nó được đặt ở đỉnh của stack và trở thành activity đang chạy activity trước sẽ ở bên dưới activity mới và sẽ không thấy trong suốt quá trình activity mới tồn tại.

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



Hình 2.2 Activity stack

2.2.3. Các trạng thái của chu kỳ sống



Hình 2.3 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.

2.2.4. 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 cảcác 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 thểchạy background dưới sự giám sát của hệ điều hành cho đến khi người dùng tự tắt chúng.

2.2.5. 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.

Hình 2.4

Tất cả các phương thức là những móc nối mà bạn có thể override để làm tương thich công việc trong ứng dụng khi thay đổi trạng thái. Tất cảcác Activity bắt buộc phải có onCreate() đểkhởi 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.

2.2.6. 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().

2.2.7. 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.

2.2.8. 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 tượngBundle chứa đựng từ trạng thái trược 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 (stared 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()nên 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 dỉnh của stack activity.
- Luôn theo sau bởi onPause()

Phương thức:onPause()

- Được gọi khi hệ thống đang resuming activity khác.
- Phương thức này là điển 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 onResumenếu 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 vữa được resumed và bao phủ nó.

- Được theo sau bởi onRestart()nếu activity đang đở 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 thởi bị hủy diệt để tiết kiệm vùng nhớ.
- Bạn có thể phân biệt giữa 2 kịch bản với phương isFinshing().

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

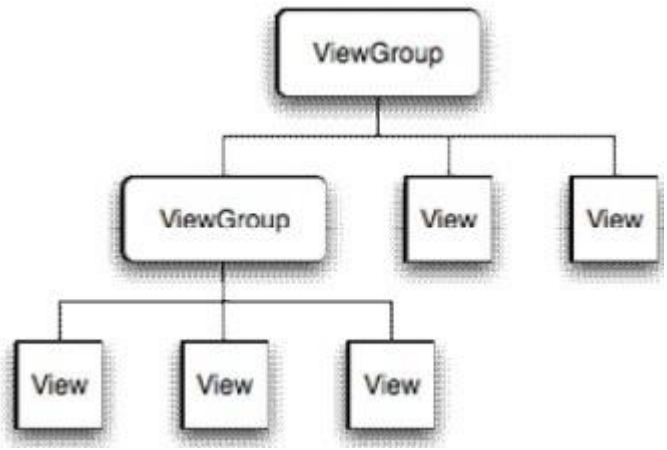
2.3.Các thành phần giao diện trong ANDROID

2.3.1. 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 thứtự. Để thể hiện một screen thì trong hàm onCreate của mỗi activity cần phải được gọi một hàm setContentView(R.layout.main); hàm này sẽ load giao diện từ file XML lên để phân tích thành mã bytecode.



Hình 2.5 Cấu trúc một giao diện ứng dụng android

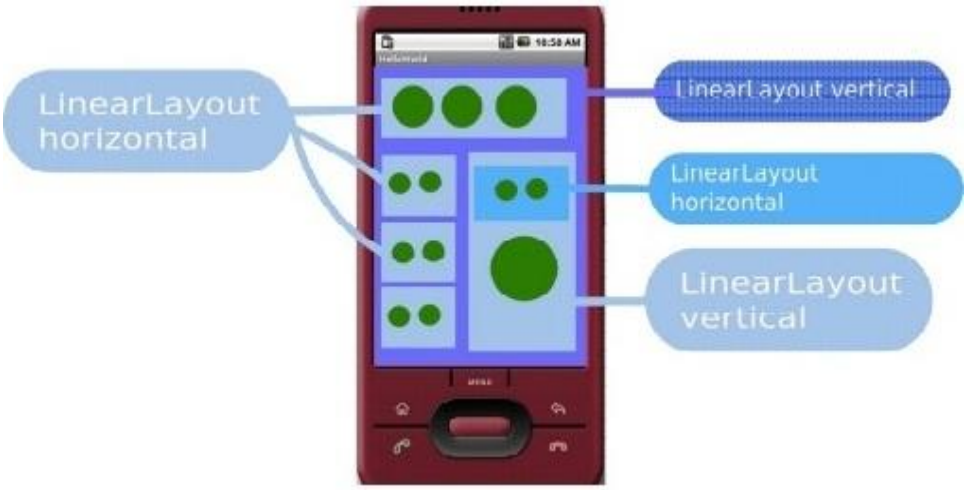
2.3.2. ViewGroup

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:

2.3.3. 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.

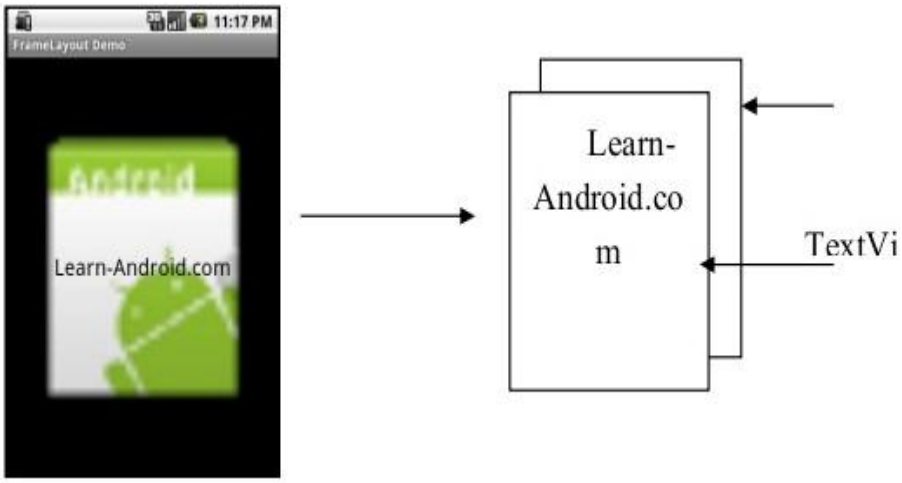
LinearLayout làm cho các thành phần trong nó không bị phụ thuộc vào kích thước của màn hình.Các thành phần trong LinearLayout được dàn theo những tỷ lệ cân xứng dựa vào các ràng buộc giữa các thành phần.



Hình 2.6 Bố trí các widget sử dụng LinearLayout

2.3.3.1. 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 2.7 Bố trí các widget trong FrameLayout

2.3.3.2. AbsoluteLayout

Layout này được sử dụng để bố trí các widget vào một vị trí bất kì trong layout dựa vào 2 thuộc tính toạ độ x, y. Tuy nhiên, kiểu layout này rất ít khi được dùng bởi vì toạ 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 vịtrí của các đối tượng sẽ không còn được chính xác như ban đầu.

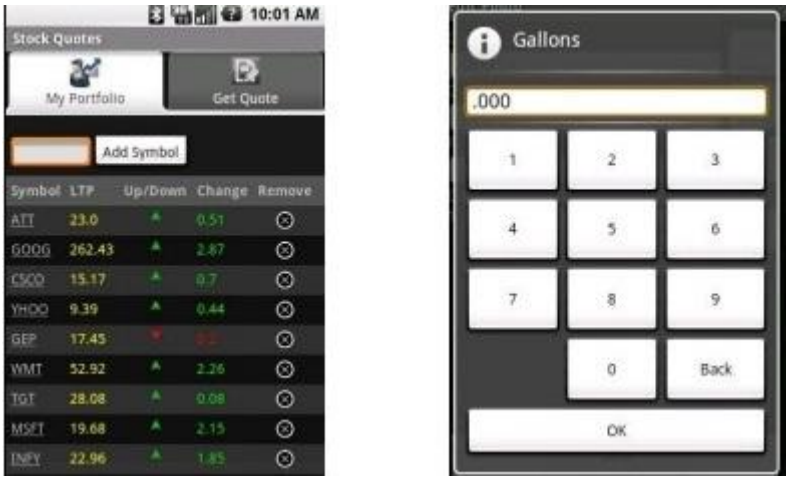
2.3.3.3. RetaliveLayout

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 RetaliveLayout 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 2.8 Bố trí widget trong RetaliveLayout

2.3.3.4. 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 2.9 Bố trí widget trong TableLayout

2.3.4. 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 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(...));

2.3.5. ImageButton

Cũng tương tự như Button,ImageButton chỉcó 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 2.10 Image Button

2.3.6. 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 = newImageView(this);
iv.setImageResource(R.drawable.icon);
```

Hình 2.11 ImageView và ImageButton

2.3.7 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 2.12 Minh hoạ cho một ListView

2.3.8. 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 thẻhtml ngay trong XML.

2.3.9. 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.

2.3.10. 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"/>
```

2.3.11. 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 2.13 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>
public booleanonCreateOptionsMenu(Menu menu) {
newMenuInflater(
getApplication()).inflate(R.menu.menu_options, menu);
return super.onCreateOptionsMenu(menu);
}
```

2.3.12. ContextMenu

ContextMenu được sử dụng để hiển thị các tuỳ chọn khi người dùng nhấn một cell nào đó trong ListView. Để tạo một ContextMenu ta cũng có 2 cách 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, ContextMenuInfo menuInfo)

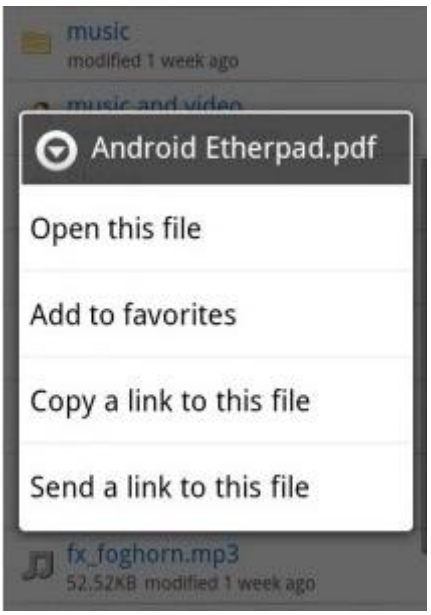
sẽ được gọi và truyền vào 3 tham số là:

- ContextMenu: đối tượng đểadd 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 xảy ra sự kiện ta làm như sau:

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

return;
}
info.position



Hình 2.14 Minh hoạ context menu

2.3.13. 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 2.15 Minh hoạ Quick Search Box

2.3.14. Activity & Intend

2.3.14.1. 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
- 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: TừActivity A muốn chuyển qua Activity B ta khai báo một Intend trong Act

ivity A:

```
Intend intend = new Intend(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 = newIntent();
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:

```
getIntend().getExtra().getString(“key1”);
getIntend().getExtra().getInt(“key2”);
```

-Task

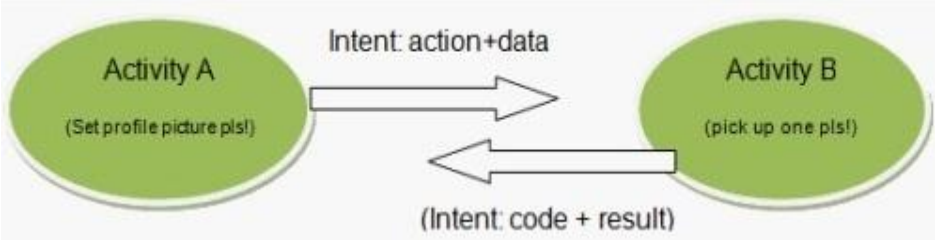
Android là một hệ điều hành đa tiến trình. Khi lập trình trên nền tảng Android thì tiến trình là một vấn đề cần phải được chú ý nhiều nhất. Mặc dù Android hỗ trợ đa tiến trình nhưng trên một thiết bị di động với cấu hình thấp mà chúng ta quá lạm dụng tiến trình thì sẽ rất tốn bộ xử lý điều này cũng đồng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

2.3.14.2. Intent

-Khái niệm Intend:

- ✓ 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 2.16 Truyền dữ liệu giữa 2 Activity

-Dữliệu của Intend:

- ✓ 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 Intend:

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
	type -định dạng kiểu dữ liệu (chuẩn MIME) -thường được tự động xác định
data -dữ liệu mà Activity được gọi sẽ xử lý -định dạng Uri (thông qua hàm Uri.parse(data))	**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)

Hình 2.17 Các thuộc tính của Intend

- ✓ 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);
```

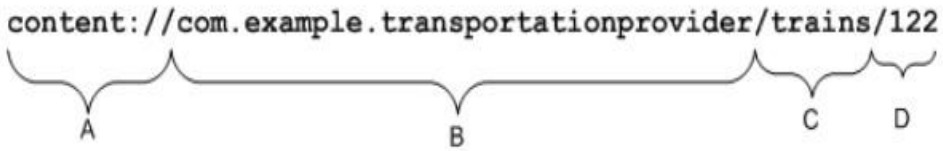
Built-in Standard Actions	
ACTION_MAIN	ACTION_ANSWER
ACTION_VIEW	ACTION_INSERT
ACTION_ATTACH_DATA	ACTION_DELETE
ACTION_EDIT	ACTION_RUN
ACTION_PICK	ACTION_SYNC
ACTION_CHOOSER	ACTION_PICK_ACTIVITY
ACTION_GET_CONTENT	ACTION_SEARCH
ACTION_DIAL	ACTION_WEB_SEARCH
ACTION_CALL	ACTION_FACTORY_TEST
ACTION_SEND	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 2.18 Các Action đã được định nghĩa sẵn trong Intend

2.4.CONTENT PROVIDER và URI

Trong hệ thống Android tất cả các tài nguyên ngư 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:



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 chỉra loại dữ liệu. Chẳng hạn như, dữliệu contact, dữliệu 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 thứ0 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() trảvề đối tượng ContentResolver.

getContentResolver().query(Uri uri); trảvề đối tượng Cursor.

2.5. 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 lí 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 nói 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 sẽcó 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(Intentintent)

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 onCreate(), 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.

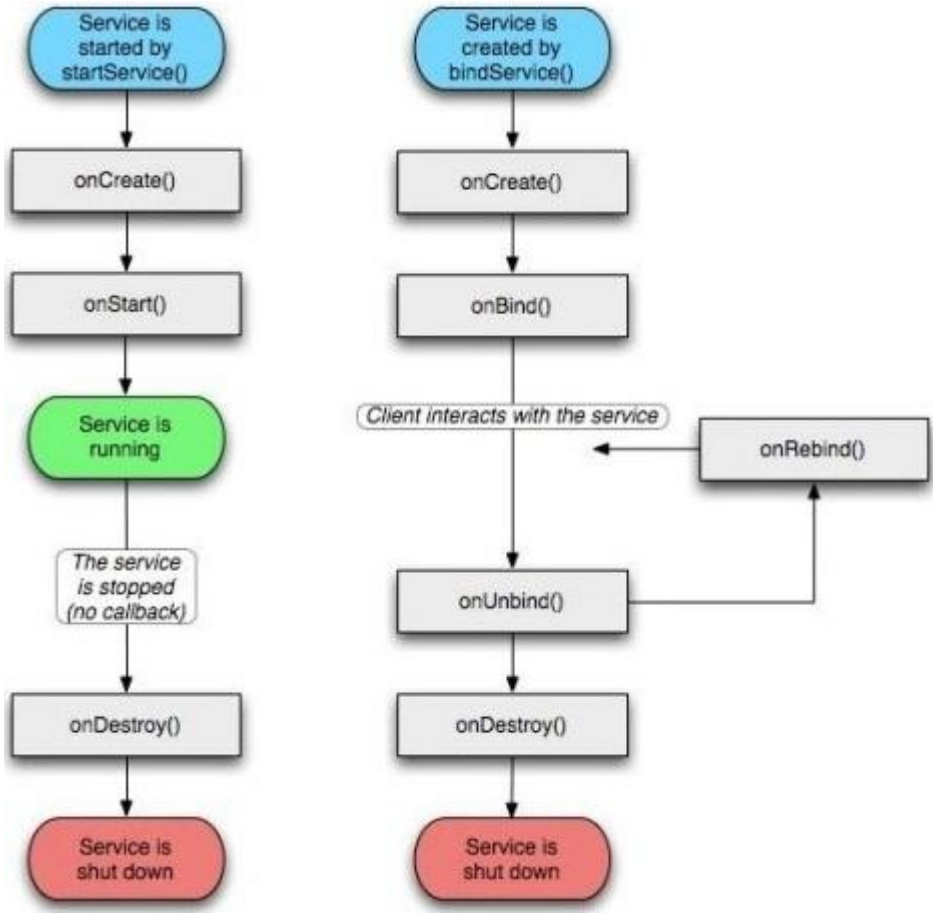
IBinderonBind(Intentintent)

booleanonUnbind(Intentintent)

void onRebind(Intentintent)

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 2.19 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 onBinder(Intend intent);
```

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

```
protected voidonCreate();
protected voidonDestroy();
```

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 somthing
}
}, 0, 5000);
```

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

Cuối cùng là khởi động Service từActivity:

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

2.6. 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_SEVICE);
```

Vd:

-Lấy thông tin ID thiết bị

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

-Lấy thông tin số serial SIM

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

2.7.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 vỏn 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.

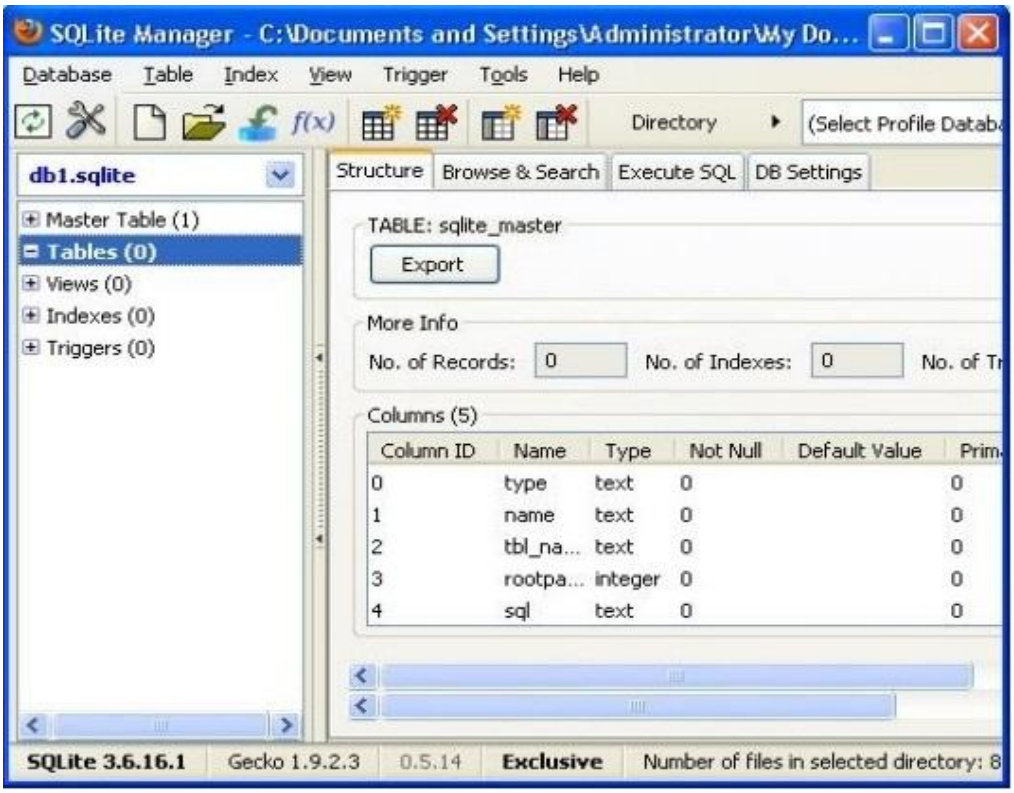
SQLite không thích hợp với những hệ thống lớn nhưng ở quy mô vừa tầm thì SQLite phát huy uy lực và không hề yếu kém về mặt chức năng hay tốc độ. Với các đặc điểm trên SQLite được sử dụng nhiều trong việc phát triển, thử nghiệm v..v.. và là sự lựa chọn phù hợp cho những người bắt đầu học database. Hiện 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 2.20 SQLite Manager

2.8. ANDROID & WEBSERVICE

2.8.1. 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 cảcác ngôn ngữ. Một thếmạnh 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.

2.8.2. 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 classPhoneNumber {
private int code;
privateString number;
PhoneNumber(intcode, String number){
this.code = code;
this.number = number;
}
public intgetCode() {return code;}
public voidsetCode(intcode) {this.code = code;}
publicString getNumber() {return number;}
public voidsetNumber(String number) {this.number = number;}
}
```

-Tạo class Person:

```
public classPerson {
privateString firstName;
privateString lastName;
privatePhoneNumber phone;
privatePhoneNumber fax;
Person(String firstName, String lastName){
this.firstName = firstName;
this.lastName = lastName;
}
publicString getFirstName() {return firstName;}
public voidsetFirstName(String firstName) {
this.firstName = firstName;
}
publicString getLastName() {return lastName;}
public voidsetLastName(String lastName) {
this.lastName = lastName;
}
publicPhoneNumber getPhone() {return phone;}
public voidsetPhone(PhoneNumber phone) {
this.phone = phone;
}
```

```
publicPhoneNumber getFax() {return fax;}
public voidsetFax(PhoneNumber fax) {this.fax = fax;}
}
```

-Tạo class TestXStream:

```
importcom.thoughtworks.xstream.XStream;
public classTestXStream {
public static voidmain(String[] args) {
XStream xstream = newXStream();
Person joe = newPerson("Joe", "Walnes");
joe.setPhone(newPhoneNumber(110, "111-111-1111"));
joe.setFax(
newPhoneNumber(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 trongbiến 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>
```

2.8.3. 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 staticResult addUser(intfrom, String username, String
display_name)throwsException {
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 = newSoapObject(NAMESPACE, METHOD_NAME);
request.addProperty("from", from);
request.addProperty("username", username);
request.addProperty("display_name", display_name);
SoapSerializationEnvelope envelope =
newSoapSerializationEnvelope(SoapEnvelope.VER11);
envelope.dotNet = true;
envelope.setOutputSoapObject(request);
Trans trans = newTrans(URL);
trans.call(SOAP_ACTION, envelope);
SoapPrimitive result = (SoapPrimitive) envelope.getResponse();
if(result.equals("-1")) {
returnResult.EXCEPTION;
} else if(result.equals("0")) {
returnResult.FAILED;
} else{
returnResult.SUCCESSED;
}
}
```

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 SoapObjectcho 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 staticArrayList<UserInfo> getListUsername(intuserid)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 = newSoapObject(NAMESPACE, METHOD_NAME);
request.addProperty("userid", userid);
SoapSerializationEnvelope envelope = newSoapSerializationEnvelope(
SoapEnvelope.VER11);
envelope.dotNet = true;
envelope.setOutputSoapObject(request);
Trans trans = newTrans(URL);
trans.call(SOAP_ACTION, envelope);
SoapObject resultsRequestSOAP = (SoapObject) envelope.bodyIn;
SoapObject a = (SoapObject) resultsRequestSOAP.getProperty(0);
intcount = a.getPropertyCount();
ArrayList<UserInfo> bki = newArrayList<UserInfo>();
for(inti = 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(newUserInfo(id, us, rname));
}
returnbki;
}
```

CHƯƠNG 3. PHÁT TRIỂN ỨNG DỤNG TỪ ĐIỂN ANH -VIỆT

3.1. Mô tả ứng dụng từ điển.

Từ điển là một ứng dụng hữu hiệu cho chiếc điện thoại thông minh. Nhu cầu tra cứu từ điển đối với những người sử dụng smartphone là rất lớn. Ứng dụng từ điển rất tiện dụng cho người dùng với khả năng tra từ nhanh và số lượng từ lớn. Người dùng tiện lợi hơn rất nhiều trong việc cài đặt và sử dụng từ điển mà không mất công cài đặt nhiều lần. một trong những từ điển phổ biến thường được sử dụng trên smartphone là từ điển Anh - Việt. Đã có rất nhiều các ứng dụng từ điển được phát triển trên các thiết bị Android, trong đồ án này trình bày quá trình phát triển ứng dụng từ điển Anh – Việt cơ bản đáp ứng các tính năng chính của từ điển là xây dựng dữ liệu từ và tra từ.

3.2. Các lớp xử lý chính

3.2.1. Lớp database từ điển.

- Đầu tiên add 1 class DBAdapter để xử lý tất cả các thao tác liên quan đến CSDL.
- Tạo 1 lớp bên trong DBAdapter được extend từ lớp SQLiteOpenHelper, override 2 phương thức onCreate() và onUpgrade() để quản lý việc tạo CSDL và version của CSDL đó.
- Mở file dữ liệu.
- Thêm giá trị vào CSDL
- Truy vấn: có thể get toàn bộ data hoặc có thể get data theo ID (tiện cho việc chỉnh sửa hay cập nhật thông tin của từng bản ghi).Còn rất nhiều các thao tác như sửa, xóa, update.... bản ghi.Tất cả các chức năng đó đều được cung cấp bởi lớp SQLiteDatabase,chỉ cần cụ thể hóa bằng các câu truy vấn là được.
- Đóng file CSDL.
- Sử dụng CSDL

3.2.2. Lớp kiểm soát tra cứu.

- Cung cấp quyền truy cập vào cơ sở dữ liệu từ điển.
- Xây dựng một UriMatcher đề xuất tìm kiếm và phím tắt làm mới truy vấn.
- Xử lý tất cả các tìm kiếm từ điển và các truy vấn gợi ý từ trình quản lý tìm kiếm.
- Khi yêu cầu một từ cụ thể, uri một mình là cần thiết.

- Khi tìm kiếm tất cả các từ điển , các đối số selectionArgs phải thực hiện các truy vấn tìm kiếm như là các yếu tố đầu tiên.Tất cả các đối số khác được bỏ qua.

3.2.3. Lớp hiển thị kết quả

- Hiển thị kết quả tìm kiếm được kích hoạt bởi hộp thoại tìm kiếm và xử lý hành động từ gợi ý tìm kiếm
- Tìm kiếm các kết quả từ điển và hiển thị cho các truy vấn nhất định.

3.2.4. Lớp tra từ.

- Hiển thị một từ và định nghĩa của nó.

3.3. Đặc tả lớp thư viện chính

3.3.1. Lớp DictionaryDatabase

```
public class DictionaryDatabase {
   private static final String TAG = "DictionaryDatabase";
   //The columns we'll include in the dictionary table
   public static final String KEY_WORD =
SearchManager.SUGGEST_COLUMN_TEXT_1;
   public static final String KEY_DEFINITION =
SearchManager.SUGGEST_COLUMN_TEXT_2;\
   private static final String DATABASE_NAME = "dictionary";
   private static final String FTS_VIRTUAL_TABLE = "FTSdictionary";
   private static final int DATABASE_VERSION = 2;
   private final DictionaryOpenHelper mDatabaseOpenHelper;
   private static final HashMap<String,String> mColumnMap =
buildColumnMap();

   public DictionaryDatabase(Context context) {
      mDatabaseOpenHelper = new DictionaryOpenHelper(context);
   }
      private static HashMap<String,String> buildColumnMap() {
      HashMap<String,String> map = new HashMap<String,String>();
      map.put(KEY_WORD, KEY_WORD);
      map.put(KEY_DEFINITION, KEY_DEFINITION);
      map.put(BaseColumns._ID, "rowid AS " +
            BaseColumns._ID);
```

```
        map.put(SearchManager.SUGGEST_COLUMN_INTENT_DATA_ID, "rowid
AS " +
                SearchManager.SUGGEST_COLUMN_INTENT_DATA_ID);
        map.put(SearchManager.SUGGEST_COLUMN_SHORTCUT_ID, "rowid AS
" +
                SearchManager.SUGGEST_COLUMN_SHORTCUT_ID);
        return map;
    }

    public Cursor getWord(String rowId, String[] columns) {
        String selection = "rowid = ?";
        String[] selectionArgs = new String[] {rowId};
        return query(selection, selectionArgs, columns);
    }
    public Cursor getWordMatches(String query, String[] columns) {
        String selection = KEY_WORD + " MATCH ?";
        String[] selectionArgs = new String[] {query+"*"};
        return query(selection, selectionArgs, columns);

    }

    private Cursor query(String selection, String[] selectionArgs, String[] columns) {
        SQLiteQueryBuilder builder = new SQLiteQueryBuilder();
        builder.setTables(FTS_VIRTUAL_TABLE);
        builder.setProjectionMap(mColumnMap)
        Cursor cursor = builder.query(mDatabaseOpenHelper.getReadableDatabase(),
                columns, selection, selectionArgs, null, null, null);
        if (cursor == null) {
            return null;
        } else if (!cursor.moveToFirst()) {
            cursor.close();
            return null;
        }
        return cursor;
    }
```

```
    private static class DictionaryOpenHelper extends SQLiteOpenHelper {
        private final Context mHelperContext;
        private SQLiteDatabase mDatabase;
        private static final String FTS_TABLE_CREATE =
                "CREATE VIRTUAL TABLE " + FTS_VIRTUAL_TABLE +
                " USING fts3 (" +
                KEY_WORD + ", " +
                KEY_DEFINITION + ");";
        DictionaryOpenHelper(Context context) {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);
            mHelperContext = context;
        }
        @Override
        public void onCreate(SQLiteDatabase db) {
            mDatabase = db;
            mDatabase.execSQL(FTS_TABLE_CREATE);
            loadDictionary();
        }
        private void loadDictionary() {
            new Thread(new Runnable() {
                public void run() {
                    try {
                        loadWords();
                    } catch (IOException e) {
                        throw new RuntimeException(e);
                    }
                }
            }).start();
        }
        private void loadWords() throws IOException {
            Log.d(TAG, "Loading words...");
            final Resources resources = mHelperContext.getResources();
            InputStream inputStream = resources.openRawResource(R.raw.definitions);
            BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream));
            try {
```

```
            String line;
            while ((line = reader.readLine()) != null) {
               String[] strings = TextUtils.split(line, "-");
               if (strings.length < 2) continue;
               long id = addWord(strings[0].trim(), strings[1].trim());
               if (id < 0) {
                  Log.e(TAG, "unable to add word: " + strings[0].trim());
               }
            }
         } finally {
            reader.close();
         }
         Log.d(TAG, "DONE loading words.");
      }
      public long addWord(String word, String definition) {
         ContentValues initialValues = new ContentValues();
         initialValues.put(KEY_WORD, word);
         initialValues.put(KEY_DEFINITION, definition);
         return mDatabase.insert(FTS_VIRTUAL_TABLE, null, initialValues);
      }
      @Override
      public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
         Log.w(TAG, "Upgrading database from version " + oldVersion + " to "
               + newVersion + ", which will destroy all old data");
         db.execSQL("DROP TABLE IF EXISTS " + FTS_VIRTUAL_TABLE);
         onCreate(db);
      }
   }
}
```

3.3.2. Lớp DictionaryProvider

```
public class DictionaryProvider extends ContentProvider {
   String TAG = "DictionaryProvider";
   public static String AUTHORITY =
"com.example.android.searchabledict.DictionaryProvider";
```

```
    public static final Uri CONTENT_URI = Uri.parse("content://" + AUTHORITY
+ "/dictionary");

    public static final String WORDS_MIME_TYPE =
ContentResolver.CURSOR_DIR_BASE_TYPE +
                                    "/vnd.example.android.searchabledict";
    public static final String DEFINITION_MIME_TYPE =
ContentResolver.CURSOR_ITEM_BASE_TYPE +
                                        "/vnd.example.android.searchabledict";
    private DictionaryDatabase mDictionary;

    private static final int SEARCH_WORDS = 0;
    private static final int GET_WORD = 1;
    private static final int SEARCH_SUGGEST = 2;
    private static final int REFRESH_SHORTCUT = 3;
    private static final UriMatcher sURIMatcher = buildUriMatcher();



    private static UriMatcher buildUriMatcher() {
        UriMatcher matcher =  new UriMatcher(UriMatcher.NO_MATCH);
        matcher.addURI(AUTHORITY, "dictionary", SEARCH_WORDS);
        matcher.addURI(AUTHORITY, "dictionary/#", GET_WORD);
        matcher.addURI(AUTHORITY,
SearchManager.SUGGEST_URI_PATH_QUERY, SEARCH_SUGGEST);
        matcher.addURI(AUTHORITY,
SearchManager.SUGGEST_URI_PATH_QUERY + "/*", SEARCH_SUGGEST);
        matcher.addURI(AUTHORITY,
SearchManager.SUGGEST_URI_PATH_SHORTCUT, REFRESH_SHORTCUT);
        matcher.addURI(AUTHORITY,
SearchManager.SUGGEST_URI_PATH_SHORTCUT + "/*",
REFRESH_SHORTCUT);
        return matcher;
    }
    @Override
    public boolean onCreate() {
        mDictionary = new DictionaryDatabase(getContext());
```

```
        return true;
    }

    @Override
    public Cursor query(Uri uri, String[] projection, String selection, String[]
selectionArgs,
                        String sortOrder) {
        switch (sURIMatcher.match(uri)) {
            case SEARCH_SUGGEST:
                if (selectionArgs == null) {
                  throw new IllegalArgumentException(
                      "selectionArgs must be provided for the Uri: " + uri);
                }
                return getSuggestions(selectionArgs[0]);
            case SEARCH_WORDS:
                if (selectionArgs == null) {
                  throw new IllegalArgumentException(
                      "selectionArgs must be provided for the Uri: " + uri);
                }
                return search(selectionArgs[0]);
            case GET_WORD:
                return getWord(uri);
            case REFRESH_SHORTCUT:
                return refreshShortcut(uri);
            default:
                throw new IllegalArgumentException("Unknown Uri: " + uri);
        }
    }
    private Cursor getSuggestions(String query) {
      query = query.toLowerCase();
      String[] columns = new String[] {
          BaseColumns._ID,
          DictionaryDatabase.KEY_WORD,
          DictionaryDatabase.KEY_DEFINITION,
          SearchManager.SUGGEST_COLUMN_INTENT_DATA_ID};
      return mDictionary.getWordMatches(query, columns);
```

```
}
private Cursor search(String query) {
  query = query.toLowerCase();
  String[] columns = new String[] {
      BaseColumns._ID,
      DictionaryDatabase.KEY_WORD,
      DictionaryDatabase.KEY_DEFINITION};
  return mDictionary.getWordMatches(query, columns);
}
private Cursor getWord(Uri uri) {
  String rowId = uri.getLastPathSegment();
  String[] columns = new String[] {
      DictionaryDatabase.KEY_WORD,
      DictionaryDatabase.KEY_DEFINITION};
  return mDictionary.getWord(rowId, columns);
}
private Cursor refreshShortcut(Uri uri) {
  String rowId = uri.getLastPathSegment();
  String[] columns = new String[] {
      BaseColumns._ID,
      DictionaryDatabase.KEY_WORD,
      DictionaryDatabase.KEY_DEFINITION,
      SearchManager.SUGGEST_COLUMN_SHORTCUT_ID,
      SearchManager.SUGGEST_COLUMN_INTENT_DATA_ID};
  return mDictionary.getWord(rowId, columns);
}
@Override
public String getType(Uri uri) {
    switch (sURIMatcher.match(uri)) {
        case SEARCH_WORDS:
            return WORDS_MIME_TYPE;
        case GET_WORD:
            return DEFINITION_MIME_TYPE;
        case SEARCH_SUGGEST:
            return SearchManager.SUGGEST_MIME_TYPE;
        case REFRESH_SHORTCUT:
```

```
            return SearchManager.SHORTCUT_MIME_TYPE;
        default:
            throw new IllegalArgumentException("Unknown URL " + uri);
      }
   }
   @Override
   public Uri insert(Uri uri, ContentValues values) {
      throw new UnsupportedOperationException();
   }
   @Override
   public int delete(Uri uri, String selection, String[] selectionArgs) {
      throw new UnsupportedOperationException();
   }
   @Override
   public int update(Uri uri, ContentValues values, String selection, String[]
selectionArgs) {
      throw new UnsupportedOperationException();
   }
}
```

3.3.3. Lớp SearchableDictionary

```
public class SearchableDictionary extends Activity {
   private TextView mTextView;
   private ListView mListView;
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      mTextView = (TextView) findViewById(R.id.text);
      mListView = (ListView) findViewById(R.id.list);
      handleIntent(getIntent());
   }
   @Override
   protected void onNewIntent(Intent intent) {
      handleIntent(intent);
   }
```

```
    private void handleIntent(Intent intent) {
        if (Intent.ACTION_VIEW.equals(intent.getAction())) {
            Intent wordIntent = new Intent(this, WordActivity.class);
            wordIntent.setData(intent.getData());
            startActivity(wordIntent);
        } else if (Intent.ACTION_SEARCH.equals(intent.getAction())) {
            String query = intent.getStringExtra(SearchManager.QUERY);
            showResults(query);
        }
    }
Searches the dictionary and displays results for the given query.
@param query The search query
    private void showResults(String query) {
        Cursor cursor = managedQuery(DictionaryProvider.CONTENT_URI, null,
null,
                            new String[] {query}, null);
        if (cursor == null) {
            mTextView.setText(getString(R.string.no_results, new Object[] {query}));
        } else {
            int count = cursor.getCount();
            String countString =
getResources().getQuantityString(R.plurals.search_results,
                                count, new Object[] {count, query});
            mTextView.setText(countString);
            String[] from = new String[] { DictionaryDatabase.KEY_WORD,
                                        DictionaryDatabase.KEY_DEFINITION };
            int[] to = new int[] { R.id.word,
                                R.id.definition };
            SimpleCursorAdapter words = new SimpleCursorAdapter(this,
                                        R.layout.result, cursor, from, to);
            mListView.setAdapter(words);
            mListView.setOnItemClickListener(new OnItemClickListener() {
                @Override
                public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
```

```
                Intent wordIntent = new Intent(getApplicationContext(),
WordActivity.class);
                Uri data =
Uri.withAppendedPath(DictionaryProvider.CONTENT_URI,
                                        String.valueOf(id));
                wordIntent.setData(data);
                startActivity(wordIntent);
            }
        });
    }
  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.options_menu, menu);
    if(Build.VERSION.SDK_INT >=
Build.VERSION_CODES.HONEYCOMB){
        SearchManager searchManager = (SearchManager)
getSystemService(Context.SEARCH_SERVICE);
        SearchView searchView = (SearchView)
menu.findItem(R.id.search).getActionView();
searchView.setSearchableInfo(searchManager.getSearchableInfo(getComponentNa
me()));
        searchView.setIconifiedByDefault(false);
    }
    return true;
  }
  @Override
  public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
      case R.id.search:
        onSearchRequested();
        return true;
      default:
        return false;
```

```
        }
    }
}
```

3.3.4. Lớp WordActivity

Displays a word and its definition.

```
public class WordActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.word);
        if(Build.VERSION.SDK_INT >=
Build.VERSION_CODES.HONEYCOMB){
            ActionBar actionBar = getActionBar();
            actionBar.setDisplayHomeAsUpEnabled(true);
        }
        Uri uri = getIntent().getData();
        Cursor cursor = managedQuery(uri, null, null, null, null);
        if (cursor == null) {
            finish();
        } else {
            cursor.moveToFirst();
            TextView word = (TextView) findViewById(R.id.word);
            TextView definition = (TextView) findViewById(R.id.definition);
            int wIndex =
cursor.getColumnIndexOrThrow(DictionaryDatabase.KEY_WORD);
            int dIndex =
cursor.getColumnIndexOrThrow(DictionaryDatabase.KEY_DEFINITION);
            word.setText(cursor.getString(wIndex));
            definition.setText(cursor.getString(dIndex));
        }
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.options_menu, menu);
```

```
        if(Build.VERSION.SDK_INT >=
Build.VERSION_CODES.HONEYCOMB){
            SearchManager searchManager = (SearchManager)
getSystemService(Context.SEARCH_SERVICE);
            SearchView searchView = (SearchView)
menu.findItem(R.id.search).getActionView();
searchView.setSearchableInfo(searchManager.getSearchableInfo(getComponentNa
me()));
            searchView.setIconifiedByDefault(false);
        }
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.search:
                onSearchRequested();
                return true;
            case android.R.id.home:
                Intent intent = new Intent(this, SearchableDictionary.class);
                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                startActivity(intent);
                return true;
            default:
                return false;
        }
    }
}
```

3.4. Đặc tả các lớp giao diện ứng dụng

3.4.1 Giao diện chính

- Giao diện:

Hình 1.giao diện chính khi Chạy ứng dụng

Hình 2.Giao diện chính khi thiết kế

- Mã nguồn:

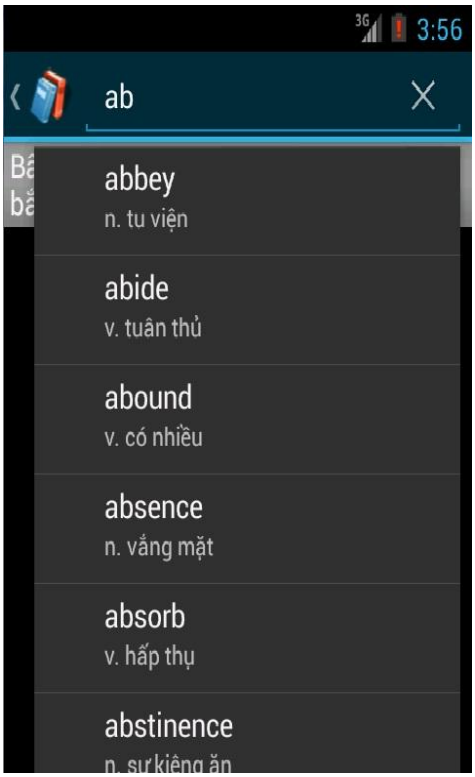
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
  <TextView
      android:id="@+id/text"
      android:textColor="?android:textColorPrimary"
      android:textSize="17sp"
      android:text="@string/search_instructions"
      android:background="@android:drawable/title_bar"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content" />
  <ListView
      android:id="@+id/list"
```

```
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="1" />
</LinearLayout>
```

3.4.2 Giao diện tra từ

- Giao diện:

Hình 3.Giao diện sau khi bấm Vào nút search



Hình 4.Giao diện khi gõ từ cần tra

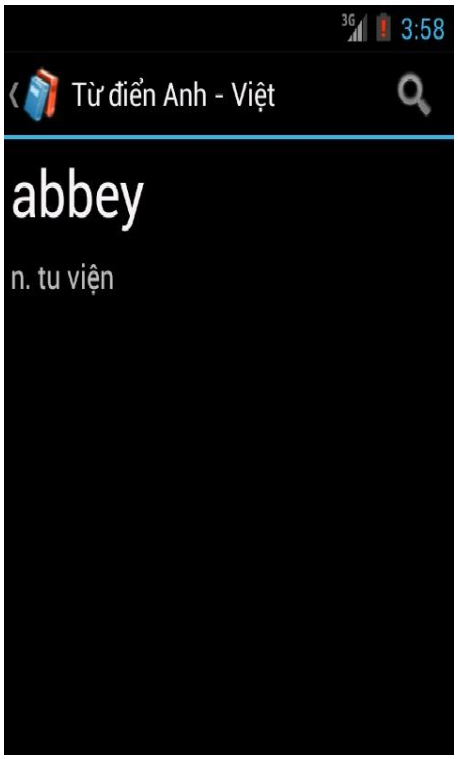
- Mã nguồn:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="5dp">
  <TextView
      android:id="@+id/word"
```

```
        style="@android:style/TextAppearance.Large"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TextView
        android:id="@+id/definition"
        style="@android:style/TextAppearance.Small"
        android:singleLine="true"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
```

3.4.3 Giao diện kết quả

- Giao diện:



Hình 5.Giao diện kết quả tìm được

Hình 6.Giao diện kết quả khi không tìm được

- Mã nguồn:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
```

```
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:padding="5dp">
    <TextView
            android:id="@+id/word"
            android:textSize="35sp"
            android:textColor="?android:textColorPrimary"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
    <TextView
            android:id="@+id/definition"
            android:textSize="18sp"
            android:textColor="?android:textColorSecondary"
            android:paddingTop="10dp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
</LinearLayout>
```

KẾT LUẬN

Trong đồ án này em đã nghiên cứu và tìm hiểu cách phát triển ứng dụng trên Hệ điều hành Android. Đồ án đã thực hiện các nhiệm vụ sau:

- Nắm bắt được qui trình làm một phần mềm trên mobile

- Tìm hiểu lịch sử và kiến trúc của Hệ điều hành Android

- Các bước phát triển một ứng dụng trên Hệ Điều Hành Android

- Phát triển ứng dụng từ điển Anh – Việt

- Hiểu thêm được nhiều kiến thức về Android và các công nghệ liên quan như là XML, XStream, Web service và SOAP…

Trong thời gian nghiên cứu, xây dựng chương trình, em đã hết sức cố gắng làm việc với sự giúp đỡ tận tình của thầy giáo hướng dẫn. Chương trình đã đạt được kết quả nhất định. Tuy nhiên với thời gian ngắn, trình độ và kinh nghiệm còn hạn chế chương trình vẫn còn nhiều thiếu sót. Rất mong các thầy cô giáo và các bạn tận tình giúp đỡ để chương trình ngày càng được hoàn thiện hơn.

Nếu được phát triển tiếp đề tài này, em sẽ tiếp tục bổ sung thêm nhiều ngôn ngữ khác và phát triển thêm giao diện chương trình để giúp người dùng sử dụng tiện ích hơn.

TÀI LIỆU THAM KHẢO

[1] The Complete Android Guide - Kevin purdy

[2] Lập trình Android - ThS Trương thị ngọc Phượng,NXB Thời Đại

[3] Hướng dẫn lập trình Android - Trần vũ tất Bình

[4] Tìm hiểu ngôn ngữ XML - Nguyễn trung Hiếu