­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­

Contents

[LỜI CẢM ƠN 2](#_Toc450466320)

[LỜI NÓI ĐẦU 3](#_Toc450466321)

[CHƯƠNG 1:TỔNG QUAN VỀ ANDROID 4](#_Toc450466322)

[1.1 Lịch sử phát triển của Android 4](#_Toc450466323)

[1.2 Kiến trúc Android 6](#_Toc450466324)

[1.2.1 Linux Kernel 7](#_Toc450466325)

[1.2.2 Library và Android Runtime 8](#_Toc450466326)

[1.2.3 Application Framework 9](#_Toc450466327)

[1.2.4 Application 10](#_Toc450466328)

[1.3 Các thành phần của Android 10](#_Toc450466329)

[1.3.1 Thành phần của một chương trình Android 10](#_Toc450466330)

[1.3.2 Các thành phần giao diện Widget 17](#_Toc450466331)

[1.3.3 Bắt sự kiện trong Android 19](#_Toc450466332)

[1.3.4 Menu 21](#_Toc450466333)

[1.3.5 Cở sở dữ liệu trong Android 22](#_Toc450466334)

[1.3.6 Broadcast receivers 24](#_Toc450466335)

[1.3.7 Thread,Asynctask,Handle 26](#_Toc450466336)

[1.3.8 Service 28](#_Toc450466337)

[CHƯƠNG 2:PHÂN TÍCH THIẾT KẾ CHƯƠNG TRÌNH ANDROID ĐỌC TRỘM TIN NHẮN NGƯỜI DÙNG 30](#_Toc450466338)

[2.1 Phân tích 30](#_Toc450466339)

[2.2 Thiết Kế 32](#_Toc450466340)

[2.2.1 Biểu đồ phân cấp chức năng BPC 32](#_Toc450466341)

[2.2.2 Thiết kế cơ sở dữ liệu 32](#_Toc450466342)

[2.2.3 Thiết kế giao diện 33](#_Toc450466343)

[2.2.4 Kết quả chạy chương trình 34](#_Toc450466344)

[2.2.5 Các hướng cài đặt và phát tán chương trình 35](#_Toc450466345)

[KẾT LUẬN 36](#_Toc450466346)

[1. Kết quả thu được 36](#_Toc450466347)

[2. Hướng phát triển đề tài 37](#_Toc450466348)

[TÀI LIỆU THAM KHẢO 37](#_Toc450466349)

[PHỤ LỤC 37](#_Toc450466350)

# 

# DANH MỤC HÌNH ẢNH

[Hình 1.1:Sơ đồ kiến trúc Android 9](#_Toc450689437)

[Hình 1.2:Quan hệ giữa Activity và Layout 13](#_Toc450689438)

[Hình 1.3:Gửi Intent từ Activity1 đến Activity2 14](#_Toc450689439)

[Hình 1.4:Những app độc lập trên thiết bị 15](#_Toc450689440)

[Hình 1.5:Truyền thông Intent giữa hai app 15](#_Toc450689441)

[Hình 1.6:Minh họa Action 16](#_Toc450689442)

[Hình 1.7:Gửi Intent đến Android 16](#_Toc450689443)

[Hình 1.8:Android sẽ nhận Intent và tìm những activity có ACTION\_SEND 17](#_Toc450689444)

[Hình 1.9:Android gửi Intent đến activity có ACTION\_SEND 17](#_Toc450689445)

[Hình 1.10:Android đưa ra lựa chọn nếu có nhiều Activity có ACTION\_SEND 18](#_Toc450689446)

[Hình 1.11:Android gửi Intent đến Activity 18](#_Toc450689447)

[Hình 1.12:Vòng đời của một Activity 19](#_Toc450689448)

[Hình 1.13:Vị trí của Manifest file 20](#_Toc450689449)

[Hình 1.14:Adapter và ListView 20](#_Toc450689450)

[Hình 1.15:Mô hình của ViewGroup trong Android 21](#_Toc450689451)

[Hình 1.16:Option Menu 24](#_Toc450689452)

[Hình 1.17:Context Menu 25](#_Toc450689453)

[Hình 1.18:Popup Menu 25](#_Toc450689454)

[Hình 1.19:Vị trí của database trong ứng dụng 26](#_Toc450689455)

[Hình 1.20:Các phương thức của lớp SQLiteOpenHelper 27](#_Toc450689456)

[Hình 1.21:Các phương thức của lớp SQLiteDatabase 27](#_Toc450689457)

[Hình 1.22:Boadcast Receiver nhận tín hiệu từ Android và gửi lệnh tới Activity 28](#_Toc450689458)

[Hình 1.23:Mô hình BoadcastReceive 29](#_Toc450689459)

[Hình 1.24:Luồng chính và các luồng phụ trong Android 30](#_Toc450689460)

[Hình 1.25:Vòng đời của Service 32](#_Toc450689461)

[Hình 2.1:Hệ thống gián điệp của công ty Việt Hồng 34](#_Toc450689462)

[Hình 2.2:Biểu đồ phân cấp chức năng 35](#_Toc450689463)

[Hình 2.3:Giao diện nhập thông tin 37](#_Toc450689464)

[Hình 2.4:Icon của chương trình 37](#_Toc450689465)

[Hình 2.5:Icon chương trình trên thiết bị 38](#_Toc450689466)

[Hình 2.6:Thông tin gián điệp được gửi về 38](#_Toc450689467)

[Hình 2.7:Nội dung thông tin gián điệp thu được 39](#_Toc450689468)

[Hình 2.8:File .apk của chương trình được upload lên Drive 39](#_Toc450689469)

[Hình 2.9:Sử dụng trình duyệt để tải file cài đặt về 40](#_Toc450689470)

# LỜI CẢM ƠN

Sau khoảng thời gian hai tháng tìm hiểu về lập trình ứng dụng trên những thiết bị chạy hệ điều hành Android thì em cũng đã xây dựng thành công “ứng dụng bí mật theo dõi hoạt động người dùng trên điện thoại chạy Android “,qua đó có những hiểu biết và kinh nghiệm ban đầu để đi vào thế giới của những lập trình viên phát triển ứng dụng Android.

Để đạt được những thành công ban đầu này thì chắc chắn không thể một mình em có thể làm được,mà cũng phải có rất nhiều sự giúp đỡ chỉ bảo của các thầy cô giáo trong khoa CNTT đại học Vinh. Em xin chân thành cảm ơn cô Ths Phạm Trà My là người đã trực tiếp hướng dẫn em làm đồ án này,em cũng xin chân thành cảm ơn các thầy cô ở khoa CNTT đại học Vinh đã nhiệt tình giảng dạy và truyền đạt kiến thức quý báu,bổ ích trong suốt quá trình 3 năm em học.

Do thời gian cũng như năng lực vẫn còn hạn chế nên mặc dù đã rất cố gắng nhưng đồ án do em thực hiện vẫn có thể tồn tại một số thiếu sót,em rất mong nhận được sự góp ý từ phía các thầy cô cũng như quý độc giả qua địa chỉ Email : [hieupn89@gmail.com](mailto:hieupn89@gmail.com), em xin chân thành cảm ơn.

# 

# LỜI NÓI ĐẦU

Trong những năm gần đây chúng ta có thể thấy rằng ngành công nghiệp thiết bị di động đang phát triển rực rỡ hơn bao giờ hết.Sỡ dĩ nó phát triển được như vậy là do những lợi ích mà nó đem lại cho cuộc sống của con người là vô cùng to lớn,chẳng hạn như chỉ với một thiết bị nhỏ gọn là chiếc smarphone chúng ta đã có thể mang trong tay hàng ngàn cuốn sách đã được số hóa và có thể đọc mọi lúc mọi nơi một cách vô cùng tiện lợi mà không cần phải đến thư viện hay mang theo những cuốn sách cồng kềnh.Rồi cũng chỉ với một chiếc Smartphone chúng ta có thể nghe nhạc, xem phim,nghe đài FM ,ghi chú công việc , truy cập internet và làm hàng nghàn tác vụ khác nữa một cách vô cùng nhanh chóng và tiện lợi.Rõ ràng sự phát triển của ngành công nghiệp thiết bị di động là điều tất yếu và chắc chắn trong tương lai sự phát triển đó còn mãnh mẽ và sâu rộng vào đời sống của con người nhiều hơn nữa.

Đi song song với sự phát triển của ngành công nghiệp thiết bị di động thì chúng ta có thể thấy trong ngành phát triển phần mềm thì lĩnh vực phát triển phần mềm cho các thiết bị di động cũng theo đó mà đang phát triển rất mạnh,đặc biệt ở một quốc gia chú trọng đến gia công phần mềm như Việt Nam thì lĩnh vực này đang trở nên khát nguồn nhân lực hơn bao giờ hết.Vì vậy cho nên với sinh viên ngành công nghệ thông tin thì đây cũng là một hướng đi để phát triển sự nghiệp,đáp ứng nhu cầu xã hội,tạo công ăn việc làm cho bản thân.

Chính vì những nhận định trên nên em đã chọn đề tài để thực hiện đồ án là “ tìm hiểu phát triển ứng dụng android ,xây dựng ứng dụng bí mật theo dõi điện thoại người dùng “ với một mục đích là tìm hiểu cách xây dựng một ứng dụng trên thiết bị di động chạy android,và thực hành xây dựng một ứng dụng thực tế.

# CHƯƠNG 1:TỔNG QUAN VỀ ANDROID

## Lịch sử phát triển của Android

Android là một hệ điều hành dựa trên nền tảng Linux,được thiết kế dành cho các thiết bị di động có màn hình cảm ứng như điện thoại thông minh và máy tính bảng.Ban đầu Android được phát triển bởi tổng công ty Android,với sự hỗ trợ tài chính từ Google và sau này được chính Google mua lại vào năm 2005.

Chính mã nguồn mở của Android cùng với tính không ràng buộc nhiều đã cho phép các nhà phát triển thiết bị di động và các lập trình viên được điều chỉnh và phân phối Android một cách tự do. Ngoài ra, Android còn có một cộng đồng lập trình viên đông đảo chuyên viết các ứng dụng để mở rộng chức năng của thiết bị.

Cho đến hiện tại, Google đã bán ra 200.000 thiết bị Android mỗi ngày và Play Store của họ vượt mức 90.000 ứng dụng. Nhờ yếu tố mở, dễ dàng tinh chỉnh cùng sự phát triển nhanh chóng đã khiến hệ điều hành này dần trở nên phổ biến, kết quả là mặc dù được thiết kế để chạy trên điện thoại và máy tính bảng nhưng giờ đây Android đã xuất hiện trên các smart TV, máy chơi game và một số thiết bị điện tử khác.

Android bắt đầu với bản beta đầu tiên vào tháng 11 năm 2007 và phiên bản thương mại đầu tiên, Android 1.0, được phát hành vào tháng 9 năm 2008. Kể từ tháng 4 năm 2009, phiên bản Android được phát triển, đặt tên theo chủ đề bánh kẹo và phát hành theo thứ tự bảng chữ cái: Cupcake (bánh cupcake), Donut (bánh vòng), Eclair (bánh Eclair), Froyo (ya-ua lạnh), Gingerbread (bánh gừng), Honeycomb (mật ong), Ice Cream Sandwich (bánh kẹp kem), Jelly Bean (kẹo dẻo), Kitkat (bánh Kitkat) và phiên bản hiện tại Lollipop (kẹo mút).

Kỷ nguyên của Android chính thức bắt đầu vào ngày 22 tháng 10 năm 2008, khi chiếc điện thoại T-Mobile G1 bắt đầu được bán ra tại Mỹ. Vào thời gian đầu, rất nhiều tính năng cơ bản bị thiếu sót như: bàn phím ảo, cảm ứng đa điểm và tính năng mua ứng dụng vẫn chưa xuất hiện. Tuy nhiên, một số tính năng cũng như giao diện đặc sản của hệ điều hành này đã khởi nguồn từ chiếc G1 và trở thành những yếu tố không thể thiếu trên Android sau này:

***Thanh thông báo vuốt từ trên xuống (Notification bar):*** Ngay từ những ngày đầu tiên của Android, thanh thông báo này đã đánh dấu một bước quan trọng mà trước đây chưa hề có hệ điều hành nào làm được - đưa tất cả thông tin tin nhắn, tin thoại hoặc các cuộc gọi nhỡ chỉ với thao tác vuốt xuống. Apple cũng đã học tập tính năng này và đưa vào hệ điều hành iOS của mình sau 3 năm kể từ khi phiên bản iOS đầu tiên ra mắt. Thanh notification này vẫn được Android sử dụng đến nay và đã cải tiến rất nhiều so với phiên bản đầu tiên.

***Màn hình chính (Home Screen) và các widget:*** Một điểm khác biệt giữa Android so với các hệ điều hành khác là phần màn hình chính của mình. Bên cạnh việc thay đổi được hình nền, Android còn cho phép người dùng tùy biến màn hình chính của mình với nhiều widgets kèm theo, chẳng hạn như đồng hồ, lịch, trình nghe nhạc, đưa các icon ứng dụng ra ngoài hoặc thậm chí có thể can thiệp sâu hơn để thay đổi toàn bộ giao diện màn hình Home Screen này.

***Đồng bộ và tích hợp chặt chẽ với Gmail:*** Vào thời điểm điện thoại G1 được bán ra, Gmail đã hỗ trợ giao thức POP và IMAP để tích hợp với các trình email trên di động. Tuy nhiên, lúc bấy giờ không có bất kì sản phẩm nào có thể hỗ trợ được hoàn toàn những tính năng ưu việt này của Gmail. Mãi cho đến khi Android 1.0 xuất hiện, vấn đề này đã được khắc phục và G1 trở thành chiếc điện thoại mang lại trải nghiệm Gmail tốt nhất trên thị trường lúc bấy giờ.

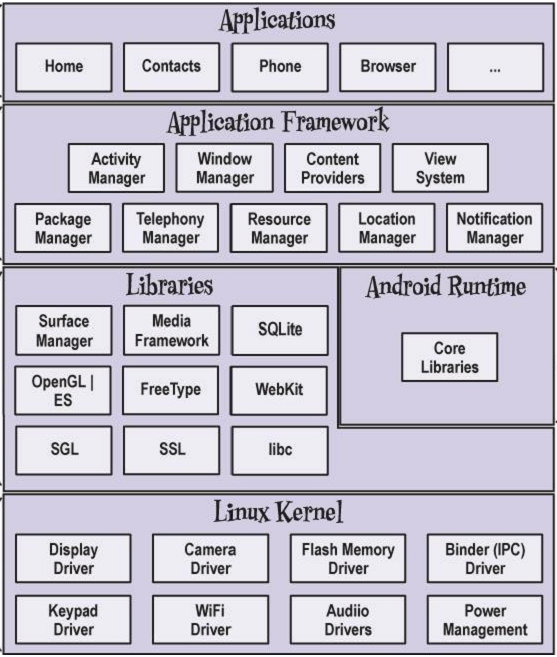
**Kho ứng dụng Android Market:** Thật khó có thể tưởng tượng một chiếc smartphone mà không hề có kho ứng dụng, nhưng vào thời điểm Android mới ra mắt, gần như không có bất kì điện thoại nào có kho ứng dụng nào được tích hợp và chính Android đã mở đầu cuộc cách mạng ứng dụng di động này. Android Market trên G1 thời bấy giờ có rất ít ứng dụng và giao diện cực kỳ đơn giản, hơn nữa tính năng mua ứng dụng trên phiên bản này vẫn chưa được xuất hiện mãi cho đến năm sau - những vấn đề này dễ hiểu vì thời điểm này Android chỉ mới được khai sinh nên mọi thứ còn khá thô sơ.

**Giao diện:** Google đã phát triển giao diện Android phiên bản 1.0 với sự hỗ trợ từ TAT, viết tắt từ The Astonishing Tribe, một công ty thiết kế tương tác của Thụy Điển. Dấu ấn rõ ràng nhất mà TAT để lại trên phiên bản Android từ phiên bản 1.0 cho đến 2.2 chính là widget đồng hồ kim nằm ngoài Home Screen tuy đơn giản nhưng rất đẹp mắt. Công ty này sau đó ngừng hợp tác với Google và bị RIM mua lại để tập trung phát triển sản phẩm Blackberry cũng như nền tảng BBX sau này.

Vào cuối năm 2008,Google cho phát hành một thiết bị cầm tay được gọi là Android Dev Phone 1 có thể chạy được 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 4 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.

## Kiến trúc Android

Sau đây là mô hình tổng quát các thành phần của hệ điều hành Android:



Hình 1.1:Sơ đồ kiến trúc Android

Chúng ta sẽ đi sâu hơn một chút về các thành phần này như sau , theo thứ tự từ dưới lên :

### Linux Kernel

Android là hệ điều hành sử dụng nhân Linux, nhưng không phải chính là nhân để chạy hệ điều hành Linux của các thiết bị chạy Linux. Có rất nhiều mã lập trình của Android sử dụng rất riêng so với hệ thống Linux, ví dụ như kernel của Google’s Android sử dụng có chế độ làm việc riêng của nó. Hệ thống lập trình OEMs là ngôn ngữ chủ đạo để viết nên hệ thống cho Kernel, bởi vì nó là phương tiện dùng để lập trình các hệ thống driver cho các phần cứng khác cho các phiên bản khác nhau của Android Kernel. Điều quan trọng của Kernel là kiểm soát và điều khiển phần cứng, hiểu một cách đơn giản, Kernel chính là một mối nối và là “thông dịch viên” giữa phần cứng và phần mềm.

Khi phần mềm cần phần cứng làm một việc gì đó, nó sẽ gửi yêu cần tới bộ vi xử lý và nó sẽ nói “các bạn nói bất cứ cái gì, chúng tôi sẽ đáp ứng cái đó”. Từ trình điều khiển độ sáng màn hình, mức âm lượng, hiệu suất tốc độ của thiết bị, điều khiển xung nhịp CPU. Ngay cả việc di chuyển con trỏ chuột hay ngón tay lướt trên màn hình cảm ứng… tất cả đều được kiểm soát bởi kernel.

Đôi khi hệ thống và Kernel không chỉ nhận được những thông tin từ những những dòng lệnh được đưa vào từ màn hình cảm ứng, nó cũng có thể nhận tín hiệu từ một thiết bị khác bằng cách này hay cách khác, Bluetooth chẳng hạn.

Nghe có vẻ phức tạp, nên chúng ta có thể tóm gọn như sau:

* Kernel được gọi là hạt nhân của hệ điều hành Android
* Đáp ứng lại các hoạt động của bạn trên smartphone, là cầu nối giữa phần mềm và phần cứng(ví dụ: tăng âm lượng bằng phím cứng, khóa màn hình bằng nút cứng, chạm vào màn hình)
* Giúp Android hiểu các kết nối vật lý bên ngoài ( ví dụ : bạn cắm cáp kết nối với máy tính ,cắm chuột, usb, hay bàn phím). Do đó có nhiều smartphone không sử dụng được USB OTG là do kernel chưa hỗ trợ bạn làm việc đó.

Các thành phần của nhân Linux:

* Display Driver :điều khiển việc hiện thị lên màn hình cũng như thu nhận những điều khiển của người dùng lên màn hình
* Camera Driver : Điều khiển hoạt động của camera,nhận luồng dữ liệu từ camera trả về
* Bluetooth Diver : Điều khiển thiết bị thu và phát sóng Bluetooth.
* USB Driver : Quản lý hoạt động của các cổng giao tiếp USB
* Keypad Driver : Điều khiển bàn phím
* Wifi Driver : Chịu trách nhiệm về việc thu phát sóng wifi
* Audio Diver : Điều khiển bộ thu phát âm thanh,giải mã các tín hiệu dạng audio thành tín hiệu số và ngược lại
* Power Management : Giám sát việc tiêu thụ điện năng
* Flash Memory Driver : Giám sát việc đọc ghi lên các thiết bị nhớ như thẻ SD,Flash
* Binder IPC Driver : Chịu trách nhiệm việc kết nối và liên lạc với mạng vô tuyến như CDMA,GSM,3G,4G để đảm bảo chức năng truyền thông được thực hiện.

### Library và Android Runtime

Phần đầu của nhân Linux có một tập hợp các thư viện bao gồm cả mã nguồn mở trình duyệt web WebKit, các thư viện phổ biến, cơ sở dữ liệu SQLite có thể được coi như một kho lưu trữ hữu ích cho việc lưu trữ và chia sẻ dữ liệu ứng dụng, thư viện để chạy các ứng dụng như ghi âm và video, SSL thư viện chịu trách nhiệm về an ninh Internet, vv...,nói chung thì các thư viện được tập hợp thành một số nhóm như :

* Thư viện hệ thống libc : thư viện dựa trên chuẩn C,được sử dụng chỉ bởi hệ điều hành.
* Thư viện Media : có nhiều code hỗ trỡ việc phát và ghi các loại định dạng âm thanh,hình ảnh,video động.
* Thư viện web :đây là thành phần để xem nội dung trên web,được sử dụng để xây dựng phần mềm duyệt web cũng như để các ứng dụng khác có thể nhúng vào.Nó cực kỳ mạnh,hỗ trợ được nhiều công nghệ mạnh mẽ như HTML 5,JavaScript,CSS,DOM,AJAX…
* Surface Manage : quản lý hiện thị 2D và 3D.
* SGL :công nghệ 2D.
* SQLite : quản lý database của ứng dụng.
* Free Type : dùng cho hiện thị ảnh,font.
* SSL :tầng thiết lập giao dịch an toàn trên 2 ứng dụng trên 1 cổng (socket)
* Open GL/EL : hỗ trợ thư viện tùy chọn 3D tĩnh và động

Adroid Runtime là phần thứ ba của kiến ​​trúc và có sẵn trên lớp thứ hai từ dưới lên. Phần này cung cấp một thành phần quan trọng được gọi là *Dalvik* (có thể đã thay đổi theo phiên bản) máy ảo là một loại máy Java được thiết kế đặc biệt và tối ưu hóa cho Android.  
  
Máy ảo Dalvik sử dụng các tính năng cốt lõi Linux như quản lý bộ nhớ và đa luồng, mà là nội tại trong ngôn ngữ Java . Máy ảo Dalvik cho phép tất cả các ứng dụng Android chạy trong tiến trình riêng của nó, với trường hợp riêng của các máy ảo Dalvik .  
  
Android Runtime cũng cung cấp một tập hợp các thư viện lõi cho phép các nhà phát triển ứng dụng Android sử dụng ngôn ngữ lập trình Java để viết các ứng dụng của mình.

### Application Framework

* Activity Manager - quản lý vòng đời của các ứng dụng.​
* Windows Manager - quản lý form của các ứng dụng.​
* Content Providers - cho phép các ứng dụng truy cập dữ liệu từ các ứng dụng khác hoặc để chia sẻ dữ liệu của riêng ứng dụng​
* View UI - để xây dựng layout của ứng dụng bao gồm: list view, text field, button, dialog, form …​
* Resource Manager - cung cấp cách thức truy cập đến non-code resources như các asset, graphic, image, music, video …​
* Notification Manager - cho phép tất cả các ứng dụng hiển thị thông báo của mình trên hệ điều hành.​

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,họ có thể truy cập vào các API trong cùng một khuôn khổ được sử dụng bởi các ứng dụng lõi.

### Application

Đây là phần ứng dụng người dùng mà chúng ta có thể thấy được,chẳng hạn như các ứng dụng lõi sẵn có như Calendar,SMS, trình duyệt web (Browser),quản lý danh bạ (Contacts),đọc e-mail (Email-Client),bản đồ (Map),quay phim chụp ảnh (camera)….Ngoài ra nó còn có thêm những ứng dụng do người dùng cài đặt thêm.Đây chính là tâng mà những người phát triển ứng dụng di động sẽ làm việc trực tiếp trên đó để tạo ra các ứng dụng.

## Các thành phần của Android

### Thành phần của một chương trình Android

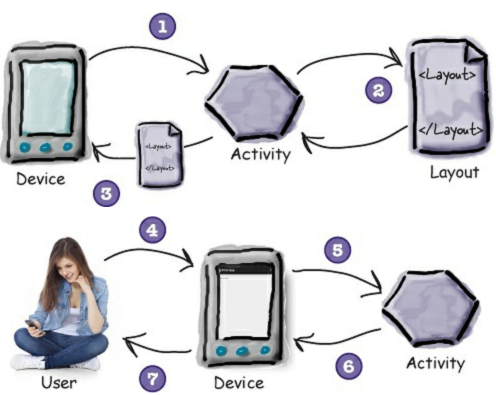
1. *Activities và Layouts*

Bất cứ một ứng dụng Android nào cũng chính là một tập hợp của những màn hình khác nhau (Screen),và mỗi một màn hình bao gồm một Activity và một Layout.

Activity định nghĩa những thứ mà người dùng có thể làm.Chúng ta có thể có một activity định nghĩa những chức năng như gửi mail,chụp ảnh hoặc tìm một thông tin liên lạc.Activity là luôn luôn tương tác với một màn hình (Screen) và chúng được viết bằng ngôn ngữ Java.

Layout chính là giao diện của màn hình, một ứng dụng có thể có nhiều giao diện tương ứng với những trạng thái khác nhau.Layout được viết ở file XML,căn cứ vào file .XML này android có thể biết được cách sắp xếp các thành phần trên màn hình như thế nào.

Sau đây chúng ta sẽ nhìn một cách trực quan hơn về mối quan hệ giữa Activity và Layout trong ứng dụng Android.



Hình 1.2:Quan hệ giữa Activity và Layout

Nhìn vào hình trên chúng ta sẽ có những bức thực thi như sau(số thứ tự được đánh trên mỗi dấu mũi tên) :

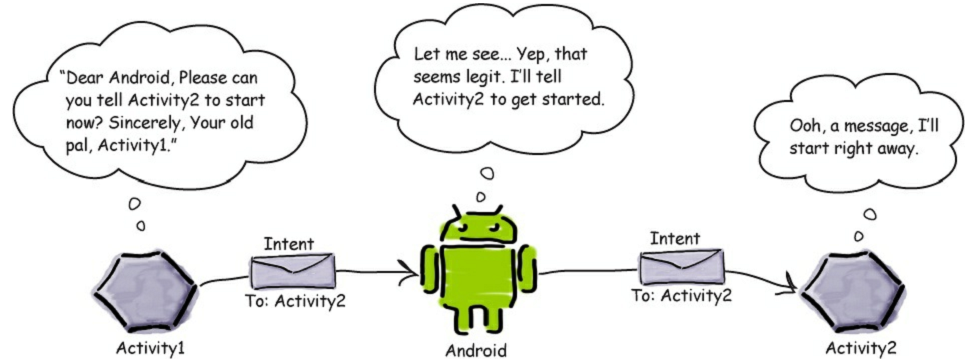
* Đầu tiên (1) thiết bị chạy ứng dụng và tạo một đối tượng Activity
* Sau đó (2) đối tượng Activity sẽ chỉ định một Layout xuất hiện
* Tiếp theo (3) activity nói Android hiện thị Layout ra màn hình
* ở bước (4) này thì người sử dụng tương tác với Layout qua màn hình của thiết bị
* bước (5) Activity sẽ hồi đáp những thao tác của người dùng bằng cách chạy những đoạn mã được viết bằng ngôn ngữ Java
* bước (6) Activity cập nhật những dữ liệu mới trên màn hình hiện thị
* bước (7) người sử dụng thấy kết quả hiện thị trên màn hình

Chú ý rằng Activity và Layout là hai khái niệm quan trọng trong lập trình ứng dụng Android.

1. *Intent*

Trong lập trình ứng dụng cho Android có một khái niệm rất quan trọng đó là Intent.Vậy thì Intent là gì ?

Intent nó giống như một người đưa thư để từ một Activity này chúng ta có thể gọi một Activity khác hoạt động.Nên nhớ rằng trên một thiết bị Android thì lại có nhiều ứng dụng, mà mỗi ứng dụng lại có thể có nhiều Activity khác nhau và cái hay là ở chỗ các Activity này có thể triệu gọi lẫn nhau để hoạt động.Nghĩa là trong một cùng một ứng dụng Activity này có thể gọi Activity kia và Activity từ ứng dụng này có thể gọi Activity của ứng dụng kia.Trong android cung cấp cho chúng một khái niệm mang tên là Intent dùng để làm những việc này.



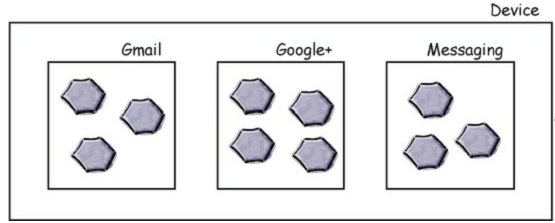
Hình 1.3:Gửi Intent từ Activity1 đến Activity2

Nhìn vào sơ đồ trên chúng ta có thể thấy là để Activity 1 gọi Activity 2 hoạt động thì đầu tiên Activity 1 sẽ khởi tạo một “bức thư “ Intent trên đó có ghi “nơi nhận “ là Acivity 2,sau đó nó đưa “ bức thư “ này đến “ bưu điện “ – chính là Android để gửi.Sau đó thì “bưu điện “ – Android sẽ xem “ bức thư “ đó và chuyển đến cho “ người nhận “ là Activity 2.Sau khi nhận được thì Activity 2 sẽ được kích hoạt.

Có một chú ý nữa là ngoài việc gửi lệnh kích hoạt thì Intent còn có thể mang theo dữ liệu,điều đó có nghĩa là Activity này hoàn toàn có thể gửi dữ liệu đến Activity kia.

Có thể có một số người sẽ hỏi rằng tại sao lại phải cần đến Intent trong khi chúng ta hoàn toàn có thể triệu gọi Activity 2 từ Activity 1 chỉ bằng cách tạo một đối tượng của lớp Activity 2 ngay bên trong Activity 1.Thực ra thì chúng ta cần phải hiểu rằng đó không phải là cách làm việc của Android,sở dĩ nó phải cần đến Intent là bởi vì Android cần phải biết được chuỗi hành động (Activity) diễn tra của người dùng,để khi người dùng nhấn nút Back trên thiết bị thì Android biết được Activity trước đó để quay lại.

Vậy còn việc hai Activity của hai ứng dụng khác nhau thì gọi nhau như thế nào ? chúng ta xem hình sau :



Hình 1.4:Những app độc lập trên thiết bị

Hình ảnh trên cho chúng ta một hình tượng về một thiết bị với nhiều ứng dụng trên đó (ví dụ cụ thể ở đây là ứng dụng Gmail,Google+ và ứng dụng nhắn tin Messaging) và trong mỗi ứng dụng này đều có nhiều những Activity.

Còn sau đây là hình ảnh mô hình về việc hai Activity của hai ứng dụng khác nhau giao tiếp với nhau:



Hình 1.5:Truyền thông Intent giữa hai app

với trường hợp hai Activity của hai ứng dụng khác nhau truyền thông cho nhau như hình trên thì có một vài câu hỏi phổ biến như sau :

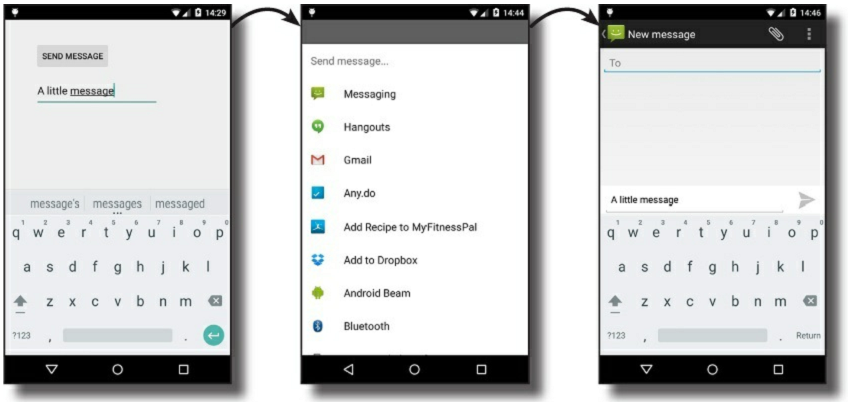
* Làm thế nào để chúng ta biết được những Activity nào là Available trên thiết bị của người sử dụng ?
* Làm thế nào để chúng ta biết được những Activity nào là có những chức năng mà ta cần ?
* Làm thế nào để biết được cách sử dụng những Activity mà Available

Câu trả lời cho những câu hỏi lớn trên đó là sử dụng actions

1. *Actions*

Actions là một cách để “nói” với Android những tác vụ cơ bản mà Activity có thể thực hiện.Chẳng hạn như Android biết rằng tất cả những Activity đăng ký send action là có khả năng của việc gửi tin nhắn.Để hiểu rõ hơn vấn đề giả sử chúng ta tạo ra một ứng dụng gửi tin nhắn đơn giản như hình sau :

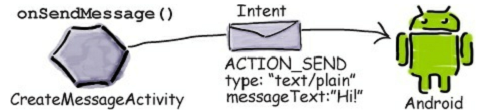
Ở Layout đầu tiên của ứng dụng thì khi chúng ta nhập text vào và nhấn nút Send Message thì ngay sau đó một danh sách các ứng dụng có khả năng gửi tin nhắn hiện lên để chúng ta chọn.Giả sử nếu chúng ta chọn ứng dụng Messaging thì ngay lập tức ứng dụng nhắn tin này được hiện lên với nội dung được điền sẵn trong ô nhập tin nhắn.



Hình 1.6:Minh họa Action

Mô hình hoạt động của ứng dụng này sẽ được mô tả trong những hình ảnh sau:

* Đầu tiên khi nhấn nút SEND MESSAGE thì một intent được khởi tạo đính kèm với dòng text mà ta nhập vào ô nhập liệu , intent này đồng thời phải chỉ rõ action nào cần được sử dụng , mà trong trường hợp này là ACTION\_SEND.Sau đó thì Intent sẽ được gửi đến Android



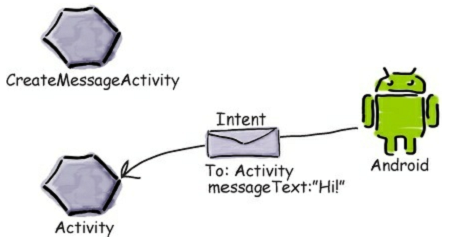
Hình 1.7:Gửi Intent đến Android

* Sau khi nhận được yêu cầu từ Activity thì Android sẽ kiểm tra xem trong tất cả các Activity ở tất cả các ứng dụng đang có trên thiết bị thì có Activity nào có chức năng gửi tin nhắn,nếu có thì đưa danh sách những ứng dụng đó ra,còn nếu không có thì ném ra một ngoại lệ ActivityNotFoundException



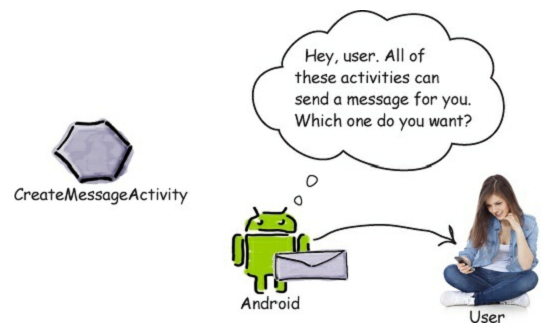
Hình 1.8:Android sẽ nhận Intent và tìm những activity có ACTION\_SEND

* Nếu chỉ có duy nhất một Activity chấp thuận nhận Intent thì Android sẽ gửi Intent đến cho Activity đó



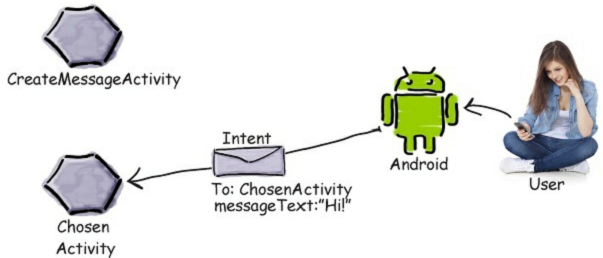
Hình 1.9:Android gửi Intent đến activity có ACTION\_SEND

* Nhưng nếu có nhiều Activity của nhiều ứng khác có khả năng nhận Intent thì ngay lập tức một Dialog hiện thị lên để người dùng chọn ứng dụng mà mình muốn dùng



Hình 1.10:Android đưa ra lựa chọn nếu có nhiều Activity có ACTION\_SEND

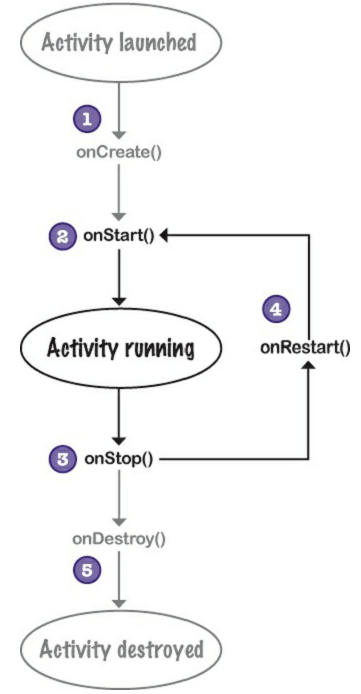
* Khi người sử dụng chọn ứng dụng mình muốn dùng xong thì Android sẽ gửi Intent đến ứng dụng đó



Hình 1.11:Android gửi Intent đến Activity

1. *Vòng đời của một Activity*

Như đã nói thì Activity là một khái niệm rất quan trọng trong việc phát triển ứng dụng Android, và mọi Activity trong ứng dụng Android đều được kế thừa từ lớp Activity đã được xây dựng sẵn.Vòng đời của một Activity được miêu tả trong hình ảnh sau:

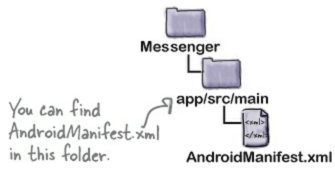


Hình 1.12:Vòng đời của một Activity

* Đầu tiên Activity khởi chạy và phương thức onCreate() được thực thi,chú ý rằng lúc này Activity vẫn chưa được visible.
* Sau đó phương thức onStart() khởi chạy lúc này Activity mới được visible,lúc này người dùng mới thấy được giao diện của ứng dụng hiện thị lên màn hình.
* Phương thức onStop() khởi chạy khi activity dừng visible tới người sử dụng
* Nếu sau khi activity dừng visible mà nó lại visible lại một lần nữa thì lúc này phương thức onRestart() sẽ được khởi chạy.
* Cuối cùng thì activity được destroyed,có một điều hết sức chú ý là trước khi phương thức onDestroy() được thực thì thì phương thức onStop() luôn được khởi chạy.Tất nhiên có một ngoại lệ là trường hợp thiết bị bị quá tải và chạy quá chậm thì Android sẽ bỏ qua onStop() .

1. *Manifest file*

Bất cứ một ứng Android nào cũng phải có một file được gọi là *AndroidManifest.xml* .Chúng ta có thể tìm nó ở đường dẫn như sau : *app/src/main* bên trong project mà chúng ta đang xây dựng.

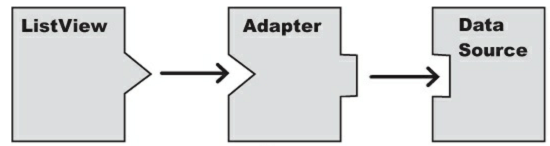


Hình 1.13:Vị trí của Manifest file

*AndroidManifest.xml* file chứa những thông tin cần thiết về ứng dụng mà chúng ta đang xây dựng, như là những Activity , những thư viện cần sử dụng và những khai báo khác.Android tạo file này cho chúng ta một cách tự động khi chúng ta tạo ứng dụng,tất nhiên nếu chúng ta tạo ứng dụng mà không sử dụng bất cứ một IDE nào thì *AndroidManifest.xml* chúng ta phải tạo bằng tay.

1. *Adapter*

Trong Adroid có đối tượng ListView để hiện thị thông tin dưới dạng danh sách.Và nếu như chúng ta muốn hiện thị dữ liệu trên ListView từ một nguồn không tĩnh như database hay từ một mảng Java thì bắt buộc chúng ta phải dùng một Adapter.Một Adapter như một cái cầu nối giữa nguồn dữ liệu và ListView:



Hình 1.14:Adapter và ListView

Có một vài loại Adapter khác nhau, và một trong số đó là Array Adapter,đây là một loại Adapter được sử dụng để đưa dữ liệu từ mảng Array vào Views.Chúng ta có thể sử dụng nó với bất cứ một lớp con nào của class AdapterView,điều đó có nghĩa là chúng ta hoàn toàn có thể áp dụng với ListView và Spinner.

### Các thành phần giao diện Widget

Trong Android, giao diện người dùng được xây dựng bởi 2 đối tượng là View và ViewGroup. Có nhiều dạng của View và ViewGroup và tất cả chúng đều là lớp con của lớp View. Đối tượng View là đơn vị cơ bản nhất của giao diện người dùng trên nền tảng Android. Chúng ta hãy đi từ khái niệm cơ bản về View và ViewGroup rồi tiến đến các kiến thức hoàn chỉnh về việc xây dựng giao diện của một ứng dụng.

*a)View*

* Một ***View*** là một đối tượng của lớp android.view.View.
* Một View chiếm giữ một không gian hình chữ nhật trên màn hình thiết bị và đảm nhiệm trọng trách vẽ lên không gian chữ nhật đó, đồng thời nhận các sự kiện tương tác lên nó.
* Như vậy nói một cách đầy đủ hơn thì một đối tượng View nắm giữ các thông tin về giới hạn của mình trong không gian hình chữ nhật trên màn hình thiết bị, đồng thời còn đảm nhiệm việc xếp đặt, vẽ, nhận biết sự thay đổi focus, tạo hiệu ứng scrolling (cuộn), và nhận các sự kiện khác lên vùng hiển thị đó.
* Những lớp con của lớp View được gọi là các ***Widget*** (nên phân biệt giữa App Widget là các ứng dụng dạng đơn giản có khả năng sắp đặt lên màn hình home của thiết bị, còn đây đang nói đến các Widget là các lớp con của lớp View). Widget là các thành phần giao diện cơ bản cho người dùng như Textview, Button, Edittext,… mà ta sẽ khảo sát ngay sau đây.

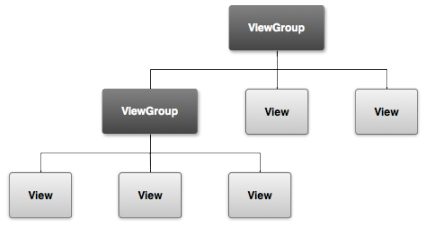
*b)ViewGroup*

***ViewGroup*** là một đối tượng của lớp android.view.ViewGroup.

ViewGroup là một dạng đặc biệt của đối tượng View mà chức năng của nó là chứa và quán lý các View hoặc là các ViewGroup khác.

Những lớp con của lớp ViewGroup được gọi là Layout (hay còn được gọi là Container).Layout hỗ trợ xây dựng kiến trúc tổng thể của giao diện như LinearLayout, RelaytiveLayout,…

Cũng như những ai đã từng làm quen với việc thiết kế giao diện (hay ta còn gọi là thiết kế layout) cho các ứng dụng Java trước kia, việc kết hơp lồng ghép nhau giữa View và ViewGroup với nhau sẽ tạo nên một cây các View và ViewGroup, cây này được bố trí và sắp xếp theo ý đồ của chúng ta để tạo nên một giao diện người dùng theo đúng thiết kế. Việc mô tả cây các View và ViewGroup như hình sau.



Hình 1.15:Mô hình của ViewGroup trong Android

Chúng ta có thể khai báo nên một cây các *View* bằng cách sử dụng Java code hoặc bằng một hoặc nhiều file XML layout. Đối với những người đã từng làm quen với Java hay J2ME trước đây có lẽ không quên kiến thức về việc tạo giao diện cho ứng dụng bằng Java code, nhưng đối với Android, việc hỗ trợ thêm cho chúng ta cách thức tạo giao diện bằng file XML layout là một cách tiếp cận mới có thể nói là dễ dàng hơn so với cách tiếp cận kiểu Java code cũ.

### Bắt sự kiện trong Android

Trong Android có 6 kiểu lập trình sự kiện,bao gồm :

* OnClick XML
* Inline anonymous listener
* Activity is listener
* Listener in variable
* Explicit listener class

*a) OnClick XML*

Cách dùng này khá phổ biến nhưng không phải tối ưu nhất khi ta bắt sự kiện cho nhiều control.Trong file XML chúng ta khai báo phương thức sẽ xảy ra khi sự kiện Click diễn ra trên button.Sau đó trong Mail Activity chúng ta cài đặt phương thức này là được.

*b)Inline anonymous listener*

Không giống như phương pháp OnCLick XML ,chúng ta không cần đặt thuộc tính android:onClick.Thay vào đó để xử lý được sự kiện click vào button thì trong file MainActivity,trong phương thức onCreate ta cần đăng ký sự kiện click cho button như trong ví dụ sau:

**protected** **void** onCreate(Bundle savedInstanceState) {

           Button btn\_clickme = (Button) findViewById(R.id.button1);

           btn\_clickme.setOnClickListener(**new** View.OnClickListener() {

                @Override

**public** **void** onClick(View arg0) {

                                     }

           });

     }

*c) Activity is listener*

Đây có lẽ là cách mà nhiều lập trình viên dùng nhất, nó khá tiện cho việc quản lí code. **Activity** sẽ **implements interface** của sự kiện, có rất nhiều loại sự kiện, tùy từng control mà các bạn chọn loại sự kiện cho phù hợp. Ở đây chúng ta dùng View.OnClickListener dùng cho button. Các bạn có thể dùng chung hàm onClick cho nhiều button khác nhau.

Khi implements interface View.OnClickListener, sẽ Override hàm onClick, tại đây các bạn viết hàm mình muốn thực hiện.

*d) Listener in variable*

Tương tự như Activity is Listener,nhưng khác ở chỗ thay vì implement interface cho Activity thì nó lại được lưu trữ vào một biến có kiểu Listener trong activity.Làm cách này thì ta có thể chia sẻ chung một biến sự kiện cho các control khác nhau.

Để hiểu rõ hơn chúng ta có thể tham khảo ví dụ sau:

**protected** **void** onCreate(Bundle savedInstanceState) {

           OnClickListener myListener = **new** OnClickListener() {

                @Override

**public** **void** onClick(View arg0) {

                }

           };

           Button btn\_clickme = (Button) findViewById(R.id.button1);

           btn\_clickme.setOnClickListener(myListener);

  }

*e) Explicit listener class*

Trường hợp này ta tách riêng một class đóng vai trò là class sự kiện riêng,khi nào mà lượng coding trong ứng dụng khổng lồ và phức tạp thì ta nên tách class sự kiện riêng để dễ quản lý.Hãy xem ví dụ sau :

**public** **class** MainActivity **extends** Activity {

   @Override

**protected** **void** onCreate(Bundle savedInstanceState) {

        Button btn\_clickme = (Button) findViewById(R.id.button1);

        btn\_clickme.setOnClickListener(**new** myListener());

   }

**private** **class** myListener **implements** OnClickListener {

        @Override

**public** **void** onClick(View arg0) {

        }

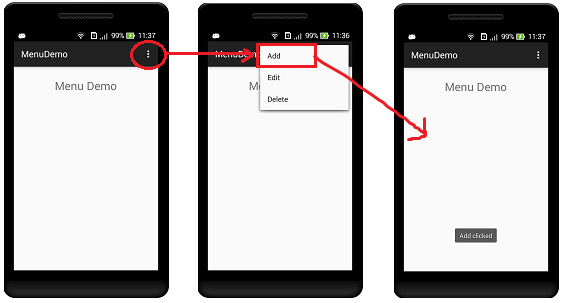
   }

### Menu

Menu là Control rất hay được sử dụng trong Android.Có 3 loại menu đó là OptionMenu,ContextMenu và PopupMenu.

*a) OptionMenu trong Android*

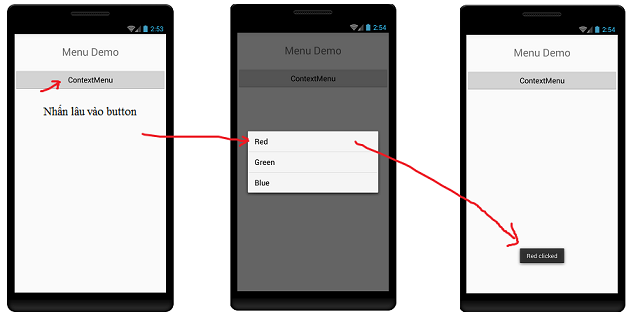
1 ứng dụng có thể mở ra 1 menu để hiển thị các hành động thông qua 1 popup. Các Option menu chỉ tồn tại nếu thiết bị có hỗ trợ phím “option”. Nhưng mặc dù có sẵn phím này, Android 4.0 trở lên khuyến khích người lập trình sử dụng ActionBar thay cho menu.



Hình 1.16:Option Menu

*b) ContextMenu trong Android*

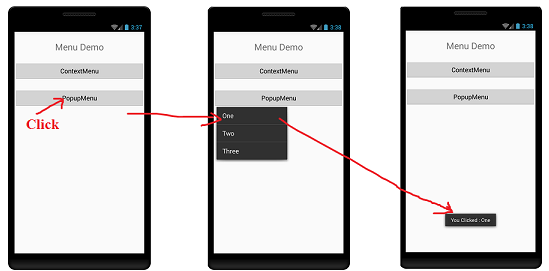
Chúng ta cũng có thể chỉ định một ContextMenu(Menu ngữ cảnh) cho một View.Nó được kích hoạt thông qua hành động long-click lên View.Một context menu cho một View được đăng ký thông qua hàm registerForContextMenu(View).Hàm này sẽ được gọi mỗi khi context menu được kích hoạt.



Hình 1.17:Context Menu

*c) Popup menu trong Android*

Đây là loại menu mà nó hiện thị bên dưới văn bản neo nếu không gian có sẵn hoặc ngay phía bên trên văn bản neo.Nó biến mất nếu bạn nhấp vào bên ngoài popup menu.Các *android.widget.PopupMenu* là lớp con trực tiếp của lớp *java.lang.Object*.



Hình 1.18:Popup Menu

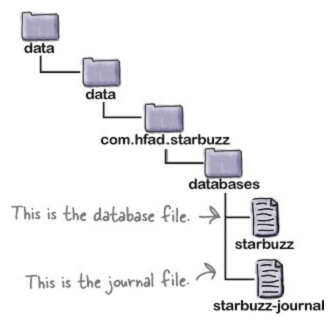
### Cở sở dữ liệu trong Android

Chương trình ứng dụng Android sử dụng SQLLite database để lưu giữ dữ liệu.Lý do mà những nhà phát triển chọn SQLite là bởi vì đặc điểm của nó là gọn,nhẹ và đơn giản.Chương trình gồm một file duy nhất vọn 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.

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

1. *Database được lưu trữ ở đâu ?*

Android tự động tạo một folder cho mỗi một ứng dụng có CSDL,ở folder đó database sẽ được lưu trữ.Chúng ta có thể tham khảo cấu trúc thư mục chứa CSDL bên trong một ứng dụng như sau :



Hình 1.19:Vị trí của database trong ứng dụng

ở đây ứng dụng của chúng ta có tên là starbzz và nó có sử dụng CSDL để lưu trữ dữ liệu.

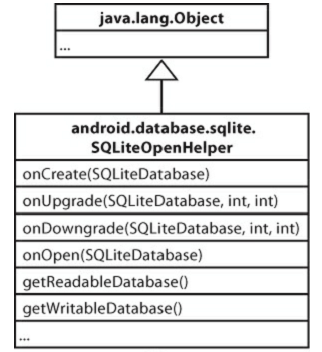
Ứng dụng có thể chứa nhiều hơn một database trong folder,mỗi một database bao gồm hai file:

File đầu tiên là file CSDL ,nó có tên tương tự như tên của database,ví dụ như ở trên đó là file starbuzz,tất cả dữ liệu được chứa trong file này.File thứ hai là file journal,chúng ta có thể thấy như ví dụ trên đó là file starbuzz-journal,nó có tên tương tự như tên của database chỉ có điều là thêm hậu tố “-journal”,file này chứa tất cả những sự thay đổi bên trong CSDL,nếu có vấn đề gì xảy ra thì Android sẽ sử dụng file này để phục hồi lại trạng thái CSDL trước đó.

1. *Các class làm việc với SQLite trong Android*

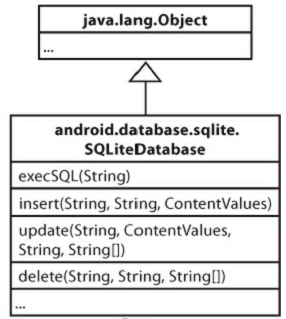
Những Class sau được sử dụng để làm việc với CSDL trong Android.

* SQLite Helper : chúng ta tạo một SQLite helper bằng cách là kế thừa lớp SQLiteOpenHelper,nó cho phép tạo và quản lý database.



Hình 1.20:Các phương thức của lớp SQLiteOpenHelper

* Cursor : một Cursor cho phép bạn đọc và ghi database,nó giống như ResultSet trong JDBC.
* SQLiteDatabase : lớp này cho phép chúng ta truy cập CSDL,nó tương tự như SQLConnection trong JDBC



Hình 1.21:Các phương thức của lớp SQLiteDatabase

Có một câu hỏi đặt ra là với SQLite không có username và password trên CSDL thì như thế nào để đảm bảo được tính bảo mật.Câu trả lời là thư mục chứa CSDL chỉ được đọc bởi ứng dụng chứa nó và việc bảo mật được thực hiện bên dưới tầng hệ điều hành.

### Broadcast receivers

Broadcast receivers là một khái niệm rất quan trọng trong lập trình Android,chúng ta có thể hiểu nó được dùng để lắng nghe các sự kiện xảy ra của hệ thống,ví dụ chúng ta tạo ra một ứng dụng nghe nhạc mà trong khi bạn đang nghe nhạc qua tai phone mà bạn rút tai phone ra thì ngay lập tức chương trình nghe nhạc rơi vào trạng thái tạm dừng.Như vậy sự kiện hệ thống ở đây chính là việc tai phone được rút ra khỏi điện thoại,chúng ta sẽ dùng Broadcast recevers để lắng nghe sự kiện đó và lập trình để tạm dừng trong phần mềm của chúng ta.Các sự kiện hệ thống ở đây có thể kể thêm như là :nhận tin nhắn,cuộc gọi đến,máy đang cắm sạc hay không…v…v.

Chúng ta có hai cách để đăng ký sử dụng BroadcastReceiver đó là :

* Đăng ký ngay trong coding:cách này ít được dùng vì chỉ khi ứng dụng được mở thì nó mới nhận các thông tin của hệ thống,khi ứng dụng tắt đi thì nó không nhận được nữa.
* Đăng ký trong AndroidMainifest.xml : cách này được sử dụng rất nhiều,với cách đăng ký này ngay cả khi ứng dụng của bạn đang tắt thì nó vẫn có thể nhận được thông tin và ngay lập tức khởi động ứng dụng.



Hình 1.22:Boadcast Receiver nhận tín hiệu từ Android và gửi lệnh tới Activity

Để có thể hiểu về BoardcastReceiver một cách hình tượng hơn chúng ta có thể hình dung theo cách như thế này :

Như đã biết ở Việt Nam chúng ta có rất nhiều đài phát thanh truyền hình. Và tất nhiên nó có 1 cột phát sóng, phát đi một loạt các hình ảnh liên tục với nhiều tầng số khác nhau, mỗi tầng số ở đây chính là 1 kênh truyền hình... thì ở đây chúng ta cũng có 1 bộ phát sóng dữ liệu, dữ liệu được gửi đi có 1 key riêng kèm theo dữ liệu

Xung quanh đó có rất nhiều nhà sử dụng tivi, chiếc tivi này sẽ thu lại sóng đó và chọn xem một kênh hiện tại. Và trong ứng dụng chúng ta cũng vậy, lúc này chúng ta có rất nhiều Activity chồng lên nhau, và trạng thái của nó cũng khác nhau ... Activity này đang hoạt động, Activity kia đang onPause() "bị đè lên"...

Tuy nhiên dù có thế nào đi nữa nếu các Activity này có bộ thu sóng thì chúng đều lấy được dữ liệu,và tất nhiên để lấy chính xác dữ liệu muốn lấy ta phải đưa vào đúng giá trị key ban đầu mà bộ phát đã phát đi.Không chỉ vậy mỗi Activity này cũng lại gồm nhiều bộ phát sóng khác và phát dữ liệu đi,và có bộ thu nào thu giá trị key tương ứng đều thu được.



Hình 1.23:Mô hình BoadcastReceive

### Thread,Asynctask,Handle

Trong Android khái niệm đa luồng (Multithread) là một khái niệm rất quan trọng.Chúng ta có thể hiểu rằng điện thoại chỉ có một bộ vi xử lý,điều đó có nghĩa là trong cùng một thời điểm chỉ có duy nhất một tác vụ được thực thi,khi tác vụ này thực hiện xong thì tác vụ khác mới được thực hiện.Chúng ta có thể hình dung khi các bạn vừa chơi game vừa nghe nhạc,nếu chạy theo kiểu thông thường thì nếu bài nhạc của bạn chạy mất 5 phút thì các bạn phải chờ nhạc chạy xong thì mới chơi game được.Điều này gây ra nhiều bất tiện.

Đó là lý do người ta sinh ra khái niệm đa luồng để có thể thực hiện nhiều tác vụ đồng thời cùng một lúc.

*a)Thread*

Thread là một tiến trình đơn vị xử lý của máy tính có thể thực hiện một công việc riêng biệt.

Multi-thread là khái niệm cho nhiều tiến trình chạy đồng thời. Một ứng dụng Java ngoài luồng chính có thể có các luồng khác thực thi đồng thời làm ứng dụng chạy nhanh và hiệu quả hơn.

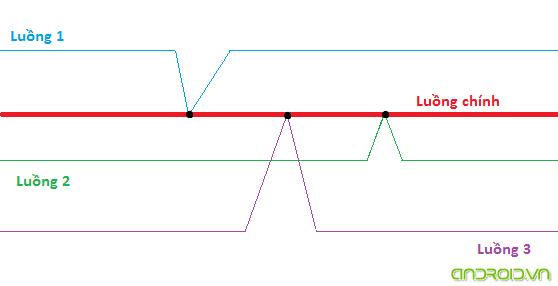
Ưu điểm của đa luồng

Mỗi luồng có thể dùng chung và chia sẻ nguồn tài nguyên trong quá trình chạy, nhưng có thể thực hiện một cách độc lập.

Ứng dụng Android có thể được tách

+ Luồng chính chạy giao diện người dung,được phép can thiệp vào các thành phần UI.

+ Các luồng phụ gửi đến luồng chính,luồng phụ không được phép can thiệp vào các thành phần UI.



Hình 1.24:Luồng chính và các luồng phụ trong Android

Nhược điểm của đa luồng:

Càng nhiều luồng thì xử lý càng phức tạp

Cần phát hiện tránh các luồng chết, luồng chạy mà không làm gì trong ứng dụng cả

*b)AsyncTask*

Một chương trình chạy trên Android có thể sẽ có cấu trúc phức tạp. Yêu cầu kết nối đến Server, CSDL, tải file … Nếu chúng ta xử lý các công việc đó trên Main Thread sẽ làm ứng dụng có vẻ chạy chậm hay treo vì chúng làm gián đoạn việc cập nhật, xử lý trên GUI. Có nhiều cách để giải quyết vấn đề này: sử dụng Service, Thread hay đơn giản hơn là dùng Async Task (Asynchronous Task).

Cấu trúc

*AsyncTask<Params, Progress, Result>* có 3 đối số là các Generic Type:

Params: Là giá trị ((biến) được truyền vào khi gọi thực thi tiến trình và nó sẽ được truyền vào *doInBackground*

Progress: Là giá trị (biến) dùng để update giao diện diện lúc tiến trình thực thi, biến này sẽ được truyền vào hàm *onProgressUpdate*.

Result: Là biến dùng để lưu trữ kết quả trả về sau khi tiến trình thực hiện xong.

Những đối số nào không sử dụng trong quá trình thực thi tiến trình thì ta thay bằng Void.

Thông thường trong 1 AsyncTask sẽ chứa 4 hàm

*onPreExecute()* :Tự động được gọi đầu tiên khi tiến trình được kích hoạt.

*doInBackground()*: Được thực thi trong quá trình tiến trình chạy nền, thông qua hàm này để ta gọi hàm *onProgressUpdate* để cập nhật giao diện (gọi lệnh publishProgress). Ta không thể cập nhật giao diện trong hàm *doInBackground().*

*onProgressUpdate ()*: Dùng để cập nhật giao diện lúc runtime

*onPostExecute()*: Sau khi tiến trình kết thúc thì hàm này sẽ tự động xảy ra. Ta có thể lấy được kết quả trả về sau khi thực hiện tiến trình kết thúc ở đây.

Trong 4 hàm trên thì hàm *doInBackground()* bắt buộc phải tồn tại, còn các hàm khác có thể khuyết.

Đối với AsyncTask thì ta cần tạo một lớp kế thừa từ AsyncTask, sau đó từ MainActivity ta gọi hàm *execute()* của tiến trình này là được.

*c)Handler*

Handler() để giao tiếp giữa 2 hay nhiều thread,ngoài ra Handler() còn được sử dụng để xử lý 1 yêu cầu nào đó của người dùng sau một khoảng thời gian nhất định.

### Service

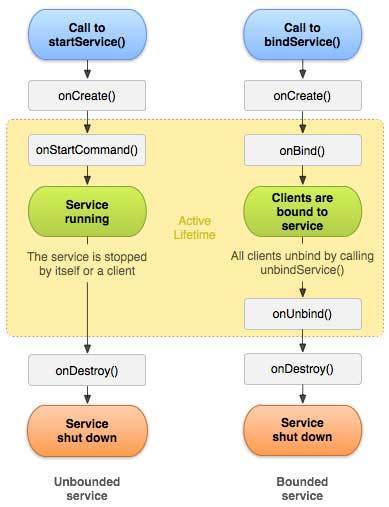
Một **Service** là một thành phần được chạy bên trong nền để xử lý các công việc trong thời gian dài. Một ứng dụng nghe nhạc có thể phát nhạc, trong khi đó người dùng đang ở giao diện của ứng dụng khác. Hoặc ứng dụng download có thể tải dữ liệu trên mạng về máy mà không ngăn chặn người dùng tương tác với các ứng dụng khác. Một **Service** gồm hai trạng thái cơ bản:

**Started**: Một **service** được gọi là **started** khi một thành phần của ứng dụng, chẳng hạn như là **activity**, start nó bằng cách gọi phương thức **startService()**. Mỗi lần được**started**, **service** chạy bên dưới vô thời hạn, thậm chí ngay cả khi thành phần đã **started**nó bị hủy.

**Bound**: Một **service** được gọi là **bound** khi một thành phần ứng dụng liên kết với nó bằng cách gọi phương thức **bindService()**. Một dịch vụ ràng buộc cung cấp một giao diện client-server cho phép các thành phần tương tác với service, gửi yêu cầu, nhận kết quả, thậm chí tương tự trong việc giao tiếp với interprocess (IPC).

Mỗi **service** có tập các phương thức và bạn có thể **implement** để giám sát việc thay đổi các trạng thái của **service**, và thậm chí bạn có thể thực hiện các công việc ở mỗi trạng thái thích hợp.

Trong sơ đồ sau phía bên trái mô tả vòng đời của service khi được tạo với startService(), và phía bên phải mô tả vòng đời của service khi được tạo với bindService().



Hình 1.25:Vòng đời của Service

Để tạo một service chúng ta có thể tạo một class kế thừa class Service. Class Service có nhiều phương thức callback, không cần phải implement hết các phương thức này, tuy nhiên việc hiểu rõ và áp dụng chúng trong code là rất quan trọng trong việc đáp ứng yêu cầu của người dùng.

* **onStartCommand()**: Hệ thống gọi phương thức này khi có một thành phần khác, chẳng hạn như activity, yêu cầu service đã được started, bởi gọi phương thúc startService(). Nếu bạn implement phương thức này thì bạn phải có trách nhiệm stop service khi công việc của nó đã hoàn thành, bằng cách gọi phương thức stopSelf() hoặc stopService().
* **onBind()**: Hệ thống gọi phương thức này khi có một thành phần khác một liên kết với service bằng cách gọi phương thức bindService(). Nếu bạn implement phương thức này, thì bạn phải cung cấp một giao diện để client sử dụng giao tiếp với service, bằng cách trả về đối tượng IBinder. Bạn phải luôn implement phương thức này, tuy nhiên nếu bạn không muốn các liên kết, thì có thể trả vể null.
* **onUnbind()**: Hệ thống gọi phương thức này khi tất cả các client đã bị ngắt kết nối từ một giao diện đặt biệt được tạo bởi các service.
* **onRebind()**: Hệ thống gọi phương thức này khi có một client mới được kết nối với service, mà trước đó đã có thông báo rằng tất cả đã bị ngắt kết nối bởi onUnbind().
* **onCreate()**: Hệ thống gọi phương thức này khi service được tạo lần đầu tiên bởi gọi phương thức onStartCommand() hoặc onBind().
* **onDestroy()**: Hệ thống gọi phương thức này khi service đã không còn được sử dụng trong thời gian dài. Bạn nên gọi phương phức này để giải phóng các tài nguyên hệ thống.

Hệ thống sẽ bắt buộc phải tắt service trong trường hợp bộ nhớ xuống thấp hoặc nó phải phục hồi lại tài nguyên cho các thao tác mà người dùng đang sử dụng.nếu Service ràng buộc bởi activity mà người dùng đang sử dụng thì sẽ ít bị giải phóng hơn hoặc Service chạy dưới foreground sẽ không bao giờ bị giải phóng.Ngoài ra Service bắt đầu trong một khoảng thời gian dài cũng có nguy cơ cao bị hệ thống giải phóng.

# CHƯƠNG 2:PHÂN TÍCH THIẾT KẾ CHƯƠNG TRÌNH ANDROID ĐỌC TRỘM TIN NHẮN NGƯỜI DÙNG

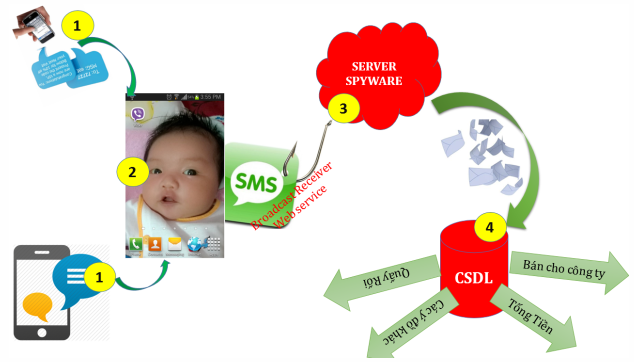
## Phân tích

Để thực hiện demo những tìm hiểu về xây dựng ứng dụng Android ,chúng ta sẽ đi thử xây dựng một ứng dụng về nó.Loại ứng dụng mà ta sẽ xây dựng là loại phần mềm gián điệp ăn cắp thông tin người dùng trên điện thoại android.Thực ra thì trên thiết bị android người dùng có rất nhiều thông tin có thể khai thác,chẳng hạn như những tin nhắn đến và đi, những cuộc gọi đến và đi của thiết bị,rồi còn có danh bạ,các bức ảnh ,các đoạn video lưu trong máy người dùng…Vậy làm cách nào để chúng ta có thể xây dựng được một ứng dụng gián điệp chạy trong thiết bị và tự động ăn cắp những thông tin này ? Đó chính là mục tiêu chúng ta sẽ hướng tới trong việc xây dựng demo phần mềm này.

Có một điều hết sức đáng lưu ý ở đây đó là việc chúng ta xây dựng phần mềm đọc trộm tin nhắn không phải là để thực hiện những ý đồ không lành mạnh, mà thay vào đó việc xây dựng phần mềm này cũng là một cách để chúng ta tìm hiểu xây dựng một ứng dụng Android,đồng thời thông qua việc đóng giả là một tin tặc đang tìm cách thâm nhập và ăn cắp thông tin người dùng thì chúng ta càng nhận thức rõ hơn về mối nguy hiểm của những phần mềm gián điệp trên Android,qua đó với tư cách là một người có hiểu biết ít nhiều về công nghệ thông tin chúng ta sẽ có những hành động,biện pháp cảnh báo đến cộng đồng những người sử dụng thiết bị cá nhân cần phải cảnh giác hơn với những phần mềm độc hại có thể ăn cắp thông tin của mình.

Về khía cạnh kỹ thuật thì Android là một hệ thống tương đối “ mở, mặc dù việc truy cập trực tiếp vào CSDL của từng ứng dụng cụ thể là điều bất khả thi , nhưng những nhà thiết kế Android cũng cho chúng ta những “ cổng “ để các nhà phát triển phần mềm có thể truy xuất những dữ liệu cần thiết khi có nhu cầu.Đây chính là cơ sở để chúng ta có thể lấy được một cách tự động những thông tin có trong thiết bị chạy Android như tin nhắn , cuộc gọi….

Thực thế thì hiện nay có rất nhiều phần mềm gián điệp với đủ thể loại khác nhau,chẳng hạn như tháng 5 năm 2015 công an thành phố Hà Nội đã điều tra ra vụ án phần mềm gián điệp Ptracker của công ty TNHH Việt Hồng với thủ đoạn cung cấp dịch vụ nghe lén người dùng để thu lời bất chính.Mô hình hoạt động phần mềm Ptracker của công ty Việt Hồng có thể hình dung theo sơ đồ sau :



Hình 2.1:Hệ thống gián điệp của công ty Việt Hồng

Ptracker với một loạt các chức năng mãnh mẽ như thu thập tin nhắn, danh bạ người dùng, tự động bật chức năng ghi âm cuộc gọi, ghi âm môi trường xung quanh hay định vị vị trí người dùng ,sau đó sẽ đưa lên CSDL của công ty bán cho người có nhu cầu.Tới khi bị cơ quan công an phát hiện, đã có tổng số 14.140 tài khoản từng bị cài phần mềm giám sát Ptracker . Trong đó, có 7.447 tài khoản mà dữ liệu vẫn còn lưu trong máy chủ của Công ty Việt Hồng, 6.693 tài khoản đã bị xóa dữ liệu khỏi máy chủ, 670 tài khoản vẫn đang còn trong thời gian giám sát.

Với tư cách giả định đang là một tin tặc muốn xây dựng một phần mềm ăn cắp dữ liệu người dùng thì chúng ta có nhận xét về phần mềm Ptracker của công ty Việt Hồng là mặc dù rất mạnh mẽ với nhiều tính năng nhưng nó vẫn có một nhược điểm ở chỗ toàn bộ dữ liệu thu thập được điều phải chuyển lên và lưu tại server của công ty.Vì vậy khi cơ quan công an phát hiện họ chỉ cần chặn ở phía server chứa CSDL là coi như toàn bộ hệ thống ăn cắp thông tin bị vô hiệu hóa.Vì vậy chúng ta sẽ xây dựng một hệ thống mà có thể tính năng không mạnh mẽ bằng Ptracker nhưng bù lại hoạt động độc lập server,thay vào đó những thông tin ăn cắp được sẽ trực tiếp gửi đến Email người dùng.Giải pháp này có ưu điểm là ngay cả khi cơ quan công an có phát hiện ra đi chăng nữa thì cũng rất khó có thể ngăn chặn được một khi phần mềm đã được phát tán một cách miễn phí trên mạng (xin lưu ý là chúng ta đang đóng vai của một tin tặc).

## Thiết Kế

### Biểu đồ phân cấp chức năng BPC

Trước khi đi vào lập trình thì bước đầu tiên vẫn là bước phân tích và thiết kế hệ thống,chúng ta sẽ sử dụng phương pháp phân tích có cấu trúc đề phân tích hệ thống của phần mềm.Biểu đồ phân cấp chức năng (BPC) là biểu đồ mô tả hệ thống về mặt chức năng, còn biểu đồ luồng dữ liệu (BLD) là biểu đồ mô tả mối liên kết giữa các chức năng trong hệ thống.Với ứng dụng của chúng ta đây chưa phải là ứng dụng lớn nên chỉ cần sử dụng biểu đồ phân cấp chức năng là đủ vì tính đơn giản,thông dụng và dễ hiểu của nó.

Hoạt Động Gián Điệp

Nhận Lệnh

## 

Gửi thông tin

Thu thập cuộc gọi

Thu thập danh bạ

Thu thập tin nhắn

Hình 2.2:Biểu đồ phân cấp chức năng

### Thiết kế cơ sở dữ liệu

Cơ sở dữ liệu của chúng ta phải đủ để lưu dữ những thông tin thu thập được từ máy điện thoại của nạn nhân.

Bảng STORE lưu dữ những tin nhắn đến :

* Trường PHONENUMBER định dạng dữ liệu TEXT dùng để lưu giữ số điện thoại người nhắn tin đến.
* Trường TIME định dạng dữ liệu TEXT dùng để lưu lại thời gian mà tin nhắn đến.
* Trường CONTENT định dạng dữ liệu TEXT dùng để lưu lại nội dung của tin nhắn đến.

Bảng STORESEND lưu dữ những tin nhắn gửi đi:

* Trường PHONENUMBER định dạng dữ liệu TEXT dùng để lưu giữ số điện mà nhắn tin đi.
* Trường TIME định dạng dữ liệu TEXT dùng để lưu lại thời gian mà gửi tin nhắn đi.
* Trường CONTENT định dạng dữ liệu TEXT dùng để lưu lại nội dung của tin nhắn gửi đi.

Bảng CALLINCOME lưu dữ những cuộc gọi đến:

* Trường PHONENUMBER định dạng dữ liệu TEXT dùng để lưu lại số điện thoại cuộc gọi đến.
* Trường TIME định dạng dữ liệu TEXT dùng để lưu lại thời gian mà cuộc gọi đến.

Bảng CALLOUTCOME lưu dữ những cuộc gọi đến:

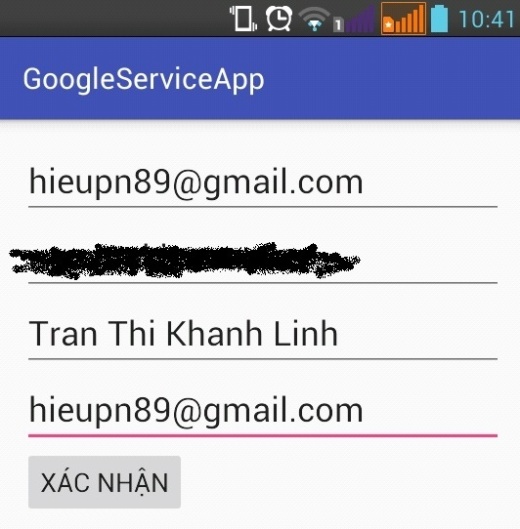
* Trường PHONENUMBER định dạng dữ liệu TEXT dùng để lưu lại số điện thoại cuộc gọi đi.
* Trường TIME định dạng dữ liệu TEXT dùng để lưu lại thời gian mà gọi đi.

Bảng CONTACT lưu dữ danh bạn trên thiết bị nạn nhân:

* Trường TIME định dạng dữ liệu TEXT dùng để lưu lại thời gian mà danh bạ được thu thập.
* Trường COTENT định dạng dữ liệu TEXT dùng để lưu lại danh bạn trên thiết bị.

### Thiết kế giao diện

Do đây là một chương trình gián điệp nên nó chỉ có duy nhất một giao diện hiện lên lúc cài đặt để người dùng nhập liệu như sau:



Hình 2.3:Giao diện nhập thông tin

Dòng đầu tiên chính là Email Sender(phải dùng Gmail), dòng thứ hai là mật khẩu của Email Sender,dòng thứ ba là tên của nạn nhân hoặc bạn viết gì cũng được ,dòng thứ tư chính là Email sẽ nhận những thông tin gián điệp thu được từ Email Sender.Có một điều lưu ý là do ở đây có bước cung cấp mật khẩu của Email Sender – là một thông tin khá nhạy cảm ,bởi vậy tốt hơn hết chúng ta nên dùng một Email phụ được lập ra để làm việc này thay vì sử dụng Email chính của mình, thêm vào đó Email Sender phải được cài đặt ở chế độ cho phép tự động gửi mail bằng phần mềm,hãy vào cài đặt lại thông số này của Email Sender.Như hình ảnh ở trên chúng ta đã dùng Email Sender và Email nhận tin là cùng một Email , như vậy cũng được , nhưng tốt hơn hết là dùng hai Email khác nhau.

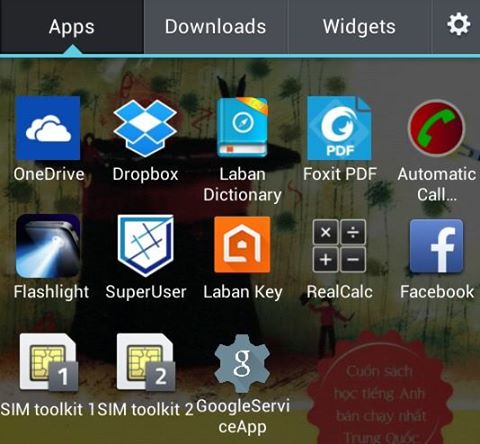
Sau khi các thông tin đã được nhập người dùng nhấn nút ” Xác Nhận” toàn bộ thông tin sẽ được lưu lại, đồng thời phần mềm của chúng ta sẽ tự động thoát về màn hình chủ của điện thoại.Thêm một điều nữa đó là nếu có mở ứng dụng lại thì những ô nhập liệu và button không xuất hiện nữa.

Icon của chương trình được lựa chọn một cách khéo léo như hình sau :



Hình 2.4:Icon của chương trình

Còn dưới đây là Icon chương trình xuất hiện trên thiết bị :

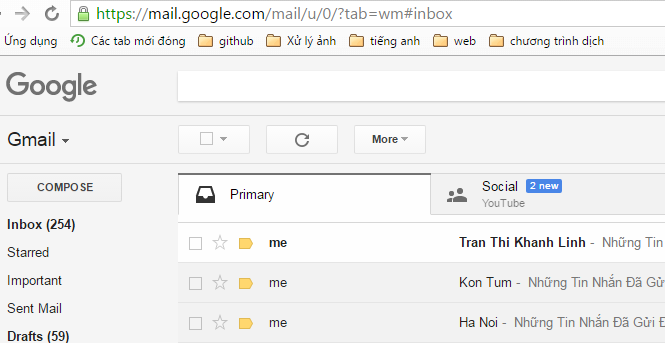


Hình 2.5:Icon chương trình trên thiết bị

Nó được đặt một cái tên rất mỹ miều là “GoogleServiceApp” - tất cả cũng chỉ để qua mắt người dùng.

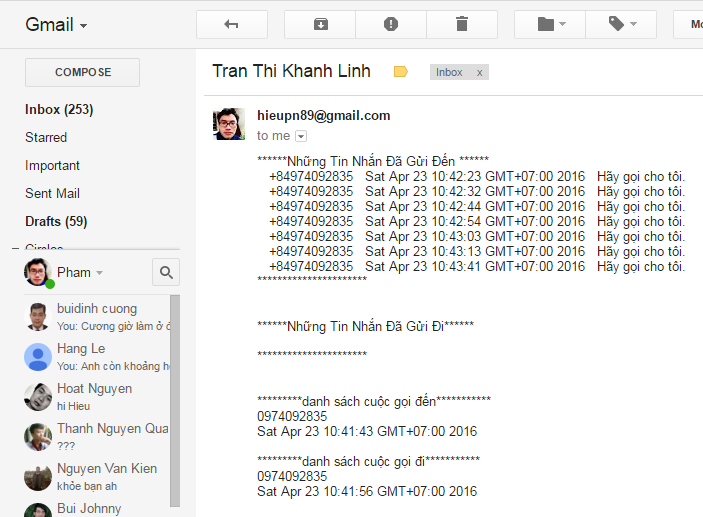
### Kết quả chạy chương trình

Sau khi được cài đặt vào máy người dùng,phần mềm sẽ âm thầm lưu lại các tin nhắn đến và đi, những cuộc gọi đến và đi,nếu muốn có danh bạ thì chúng ta chỉ cần nhắn tin vào máy nạn nhân một tin nhắn với mã lệnh : PRETTY\_BOY, ngay lập tức chương trình sẽ thu thập danh bạ, lưu vào CSDL và chờ khi có mạng để gửi về Email người dùng. Dưới đây là kết quả thử nghiệm chương trình :



Hình 2.6:Thông tin gián điệp được gửi về

Nội dung mà phần mềm gián điệp thu được :



Hình 2.7:Nội dung thông tin gián điệp thu được

### Các hướng cài đặt và phát tán chương trình

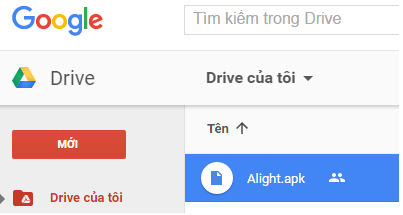
Do đây là phần mềm gián điệp nên việc cài đặt chương trình cần phải thực hiện nhanh chóng và bí mật.

Những yêu cầu cần phải có trước khi tiến hành cài đặt :

* Phần mềm gián điệp ở dạng file chạy .apk.
* Thiết bị của nạn nhân chạy Android từ 4.1 trở lên.

Thực ra thì có thể tiến hành cài đặt chương trình theo nhiều cách khác nhau,sau đây là phương pháp mà tác giả đóng vai trò là một tin tặc đang có ý định tấn công thiết bị của nạn nhân.

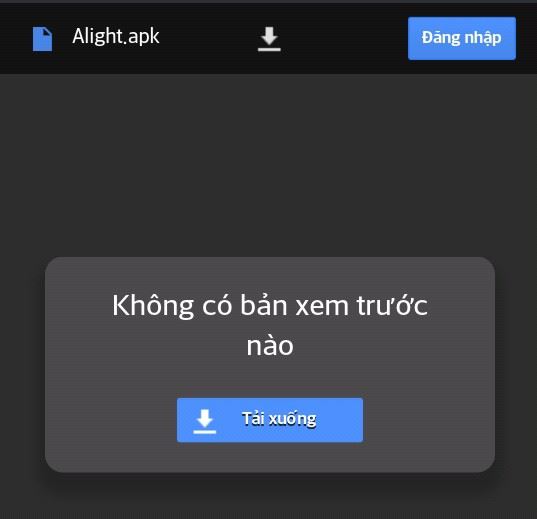
Đầu tiên file .apk của phần mềm gián điệp sẽ được tải lên một trang chia sẻ trực tuyến,trong trường hợp này tin tặc sử dụng trang drive.goole.com của google,sau khi upload lên thì nó sẽ cho chúng ta một đường link để chia sẻ file.



Hình 2.8:File .apk của chương trình được upload lên Drive

Như hình trên chúng ta có thể thấy phần mềm gián điệp Alight.apk đã được đưa lên trang chia sẻ trực tuyến Drive của google.

Tiếp theo , đây là bước khó nhất , phải tiếp cận được với thiết bị của nạn nhân, sử dụng các trình duyệt như Chrome hay Firefox truy cập vào đường link chia sẻ file phần mềm gián điệp mà tin tặc đã tải lên Drive.google.com.Trình duyệt hiện thị giao diện như sau :



Hình 2.9:Sử dụng trình duyệt để tải file cài đặt về

Click vào “Tải Xuống” và sau đó cài đặt bình thường.Toàn bộ quá trình này chỉ mất khoảng 20 – 30 giây vẫn đảm bảo được tính bí mất và không bị nạn nhân nghi ngờ.

# KẾT LUẬN

## 1. Kết quả thu được

Qua quá trình tìm hiểu và thực hiện đồ án,bản thân người thực hiện đã đạt được những kết quả như sau:

* Tìm hiểu được cách thức phát triển một chương trình ứng dụng trên thiết bị chạy hệ điều hành Android.
* Xây dựng thành công phần mềm gián điệp có khả năng theo dõi tin nhắn, cuộc gọi và danh bạ người dùng.
* Nhận thức được rõ hơn về vấn đề bảo mật thông tin trên các thiết bị di động chạy hệ điều hành Android.

## 2. Hướng phát triển đề tài

Phần mềm gián điệp trên đây hoàn toàn có thể phát triển thêm để trở nên nguy hiểm hơn với các tính năng :

* Phát hiện và nhận diện ảnh nhạy cảm,qua đó đánh cắp những ảnh nhạy cảm của nạn nhân.
* Lấy thông tin về vị trí của nạn nhân.
* Lấy thông tin lịch sử truy cập web của nạn nhân
* Thực hiện điều khiển việc chặn cuộc gọi của nạn nhân
* Thực hiện việc điều khiển chặn truy cập web của nạn nhân.

# TÀI LIỆU THAM KHẢO

* **Giáo trình:**
* [1] Nguyễn Văn Ba, *Phân tích và thiết kế hệ thống thông tin*, Nhà xuất bản Đại Học Quốc Gia Hà Nội, Năm xuất bản 2009.
* [2] Phạm Minh Tuấn, *Phân tích và thiết kế hệ thống thông tin*, Nhà xuất bản Hà Nội, Năm xuất bản 2005.
* [3] [Dawn Griffiths, David Griffiths](http://shop.oreilly.com/product/0636920029045.do#tab_04_2), *Head First Android Development*, Nhà xuất bản O'Reilly,Năm xuất bản 2015
* **Địa chỉ websites:**
* [1] duythanhcse.wordpress.com
* [2] www.tutorialspoint.com
* [3] developer.android.com
* [4] www.[stackoverflow.com](http://stackoverflow.com)