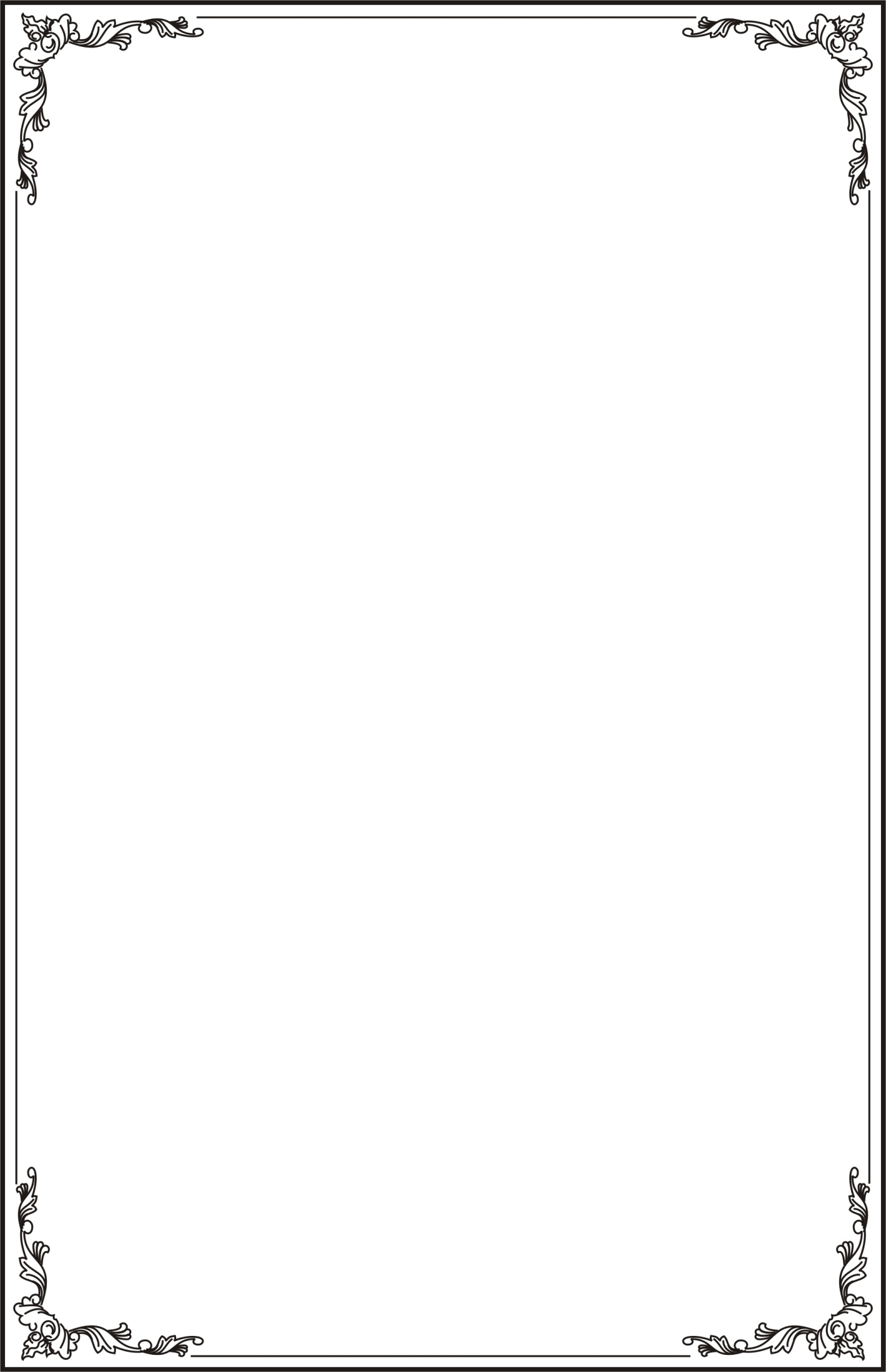
**HỌC VIỆN QUẢN LÝ GIÁO DỤC**

**KHOA CÔNG NGHỆ THÔNG TIN**

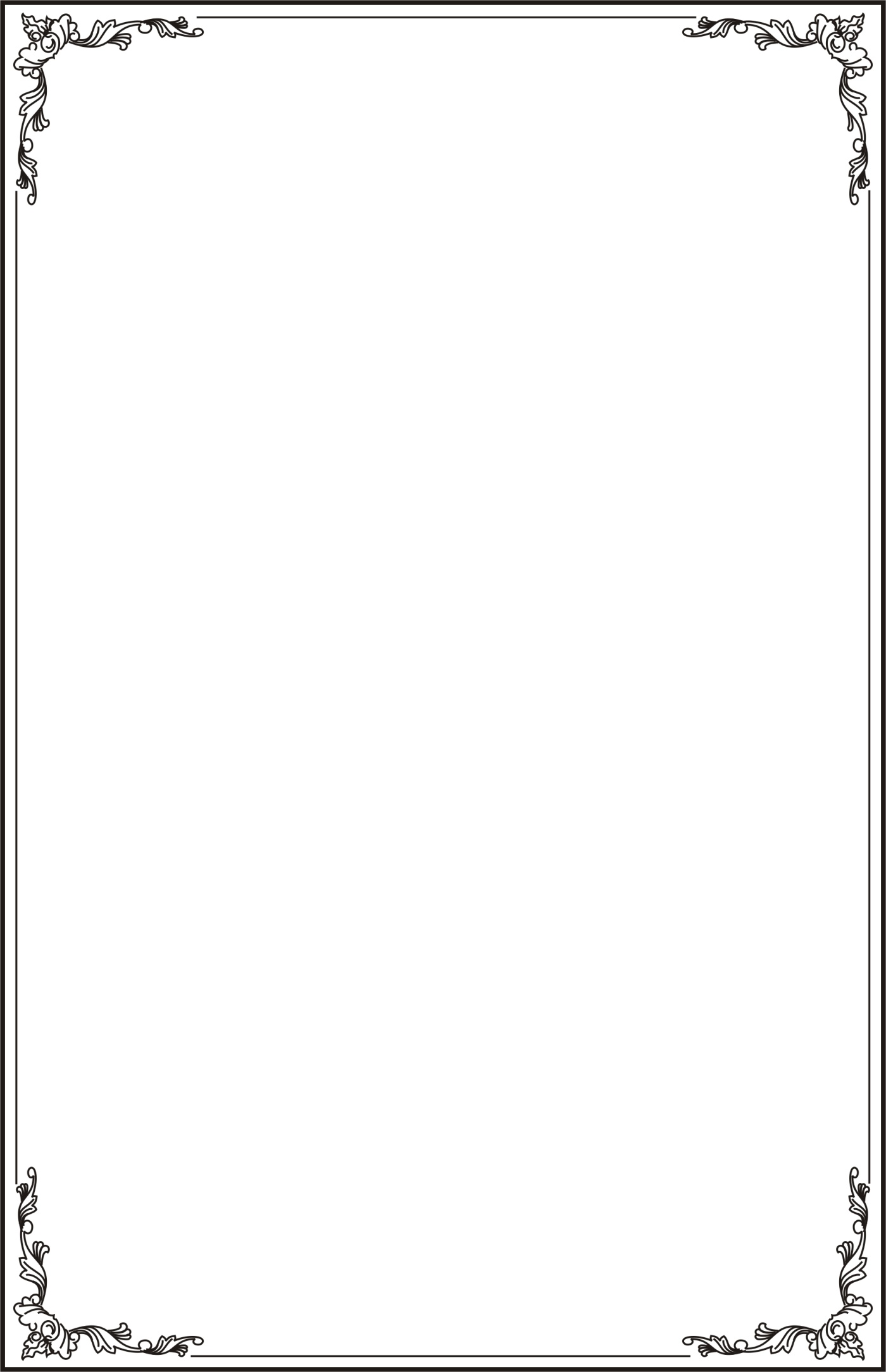
-------\*\*\*-------

**ĐỒ ÁN TỐT NGHIỆP**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**“Ứng dụng nghe nhạc TPlayer”**

**HÀ NỘI - 2017**

**HỌC VIỆN QUẢN LÝ GIÁO DỤC**

**KHOA CÔNG NGHỆ THÔNG TIN**

-------\*\*\*-------

**ĐỒ ÁN TỐT NGHIỆP**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**“Ứng dụng nghe nhạc TPlayer”**

|  |  |  |
| --- | --- | --- |
| Sinh viên thực hiên | : | NGUYỄN HỮU THỌ  Lớp K7B - CNTT |
| Mã số | : | 15 |

**HÀ NỘI - 2017**

# 

# *LỜI CẢM ƠN*

*Lời đầu tiên tôi xin chân thành cảm ơn các thầy cô giáo Trường Học Viện Quản Lý Giáo Dục đã nhiệt tình dạy bảo và truyền đạt những kiến thức kinh nghiệm quý báu cho tôi trong suốt thời gian học tập tại trường.*

*Trong thời gian làm đồ án tôi xin cảm ơn Tiến sĩ Nguyễn Mạnh Hùng đã tận tình chỉ dẫn tôi. Tôi cũng xin chân thành cảm ơn tới các thầy cô trong khoa Công nghệ thông tin – đã trang bị cho tôi những kiến thức chuyên môn quý giá, giúp đỡ và tạo điều kiện để tôi có thể học tập và thực hành tại khoa.*

*Sau cùng tôi xin gửi lời cảm ơn đến gia đình, bạn bè – những người đã luôn ủng hộ, động viên, củng cố niềm tin và tạo điều kiện tốt nhất cho tôi trong suốt quá trình hoàn thành bài luận này.*

*Vì thời gian làm đồ án có hạn nên trong quá trình làm không tránh khỏi những thiếu sót, tôi rất mong sự đóng góp ý kiến từ thầy cô và các bạn.*

*Tôi xin chân thành cảm ơn!*

Hà Nội, ngày 25 tháng 4 năm 2017

Sinh viên

Nguyễn Hữu Thọ

MỤC LỤC

Trang

Trang bìa

PHỤ LỤC

**CHƯƠNG 1 : GIỚI THIỆU CÔNG NGHỆ**

1. **Hệ điều hành Android**
2. **Lịch sử**

[Android](https://vi.wiktionary.org/wiki/android) là một [hệ điều hành](https://vi.wikipedia.org/w/index.php?title=H%E1%BB%87_%C4%91i%E1%BB%81u_h%C3%A0nh_di_%C4%91%E1%BB%99ng&action=edit&redlink=1) dựa trên nền tảng [Linux](https://vi.wikipedia.org/wiki/Linux) được thiết kế dành cho các thiết bị di động có [màn hình cảm ứng](https://vi.wikipedia.org/wiki/M%C3%A0n_h%C3%ACnh_c%E1%BA%A3m_%E1%BB%A9ng) như [điện thoại thông minh](https://vi.wikipedia.org/wiki/%C4%90i%E1%BB%87n_tho%E1%BA%A1i_th%C3%B4ng_minh) và [máy tính bảng](https://vi.wikipedia.org/wiki/M%C3%A1y_t%C3%ADnh_b%E1%BA%A3ng). Ban đầu, Android được phát triển bởi Tổng công ty Android, với sự hỗ trợ tài chính từ [Google](https://vi.wikipedia.org/wiki/Google) và sau này được chính Google mua lại vào năm 2005. Android ra mắt vào năm 2007 cùng với tuyên bố thành lập [Liên minh thiết bị cầm tay mở](https://vi.wikipedia.org/wiki/Li%C3%AAn_minh_thi%E1%BA%BFt_b%E1%BB%8B_c%E1%BA%A7m_tay_m%E1%BB%9F): một hiệp hội gồm các công ty [phần cứng](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_c%E1%BB%A9ng), [phần mềm](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m), và [viễn thông](https://vi.wikipedia.org/wiki/Vi%E1%BB%85n_th%C3%B4ng) với mục tiêu đẩy mạnh các [tiêu chuẩn mở](https://vi.wikipedia.org/w/index.php?title=Ti%C3%AAu_chu%E1%BA%A9n_m%E1%BB%9F&action=edit&redlink=1) cho các thiết bị di động.Chiếc điện thoại đầu tiên chạy Android được bán vào tháng 10 năm 2008 Android có [mã nguồn mở](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m_ngu%E1%BB%93n_m%E1%BB%9F) và Google phát hành mã nguồn theo [Giấy phép Apache](https://vi.wikipedia.org/wiki/Gi%E1%BA%A5y_ph%C3%A9p_Apache). Chính mã nguồn mở cùng với một giấy phép không có nhiều ràng buộc đã cho phép các nhà phát triển thiết bị, mạng di động và các lập trình viên nhiệt huyết 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, bằng một loại ngôn ngữ lập trình [Java](https://vi.wikipedia.org/wiki/Java_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)) có sửa đổi.Vào tháng 10 năm 2012, có khoảng 700.000 ứng dụng trên Android, và số lượt tải ứng dụng từ [Google Play](https://vi.wikipedia.org/wiki/Google_Play), cửa hàng ứng dụng chính của Android, ước tính khoảng 25 tỷ lượt.

1. **Phát triển**

Android có một [hạt nhân](https://vi.wikipedia.org/wiki/Nh%C3%A2n_h%E1%BB%87_%C4%91i%E1%BB%81u_h%C3%A0nh) dựa trên [nhân Linux](https://vi.wikipedia.org/wiki/H%E1%BA%A1t_nh%C3%A2n_Linux) phiên bản 2.6, kể từ Android 4.0 Ice Cream Sandwich (bánh ngọt kẹp kem) trở về sau, là phiên bản 3.x, với [middleware](https://vi.wikipedia.org/wiki/Middleware), [thư viện](https://vi.wikipedia.org/w/index.php?title=Th%C6%B0_vi%E1%BB%87n_ph%E1%BA%A7n_m%E1%BB%81m&action=edit&redlink=1) và [API](https://vi.wikipedia.org/wiki/Giao_di%E1%BB%87n_l%E1%BA%ADp_tr%C3%ACnh_%E1%BB%A9ng_d%E1%BB%A5ng) viết bằng [C](https://vi.wikipedia.org/wiki/C_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)), còn [phần mềm ứng dụng](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m_%E1%BB%A9ng_d%E1%BB%A5ng) chạy trên một [nền tảng ứng dụng](https://vi.wikipedia.org/w/index.php?title=N%E1%BB%81n_t%E1%BA%A3ng_%E1%BB%A9ng_d%E1%BB%A5ng&action=edit&redlink=1) gồm các thư viện tương thích với Java dựa trên [Apache Harmony](https://vi.wikipedia.org/w/index.php?title=Apache_Harmony&action=edit&redlink=1). Android sử dụng [máy ảo Dalvik](https://vi.wikipedia.org/w/index.php?title=Dalvik_(ph%E1%BA%A7n_m%E1%BB%81m)&action=edit&redlink=1) với một [trình biên dịch động](https://vi.wikipedia.org/w/index.php?title=Tr%C3%ACnh_bi%C3%AAn_d%E1%BB%8Bch_JIT&action=edit&redlink=1) để chạy 'mã dex' (Dalvik Executable) của Dalvik, thường được biên dịch sang [Java bytecode](https://vi.wikipedia.org/w/index.php?title=Java_bytecode&action=edit&redlink=1). Nền tảng phần cứng chính của Android là [kiến trúc ARM](https://vi.wikipedia.org/wiki/C%E1%BA%A5u_tr%C3%BAc_ARM). Người ta cũng hỗ trợ [x86](https://vi.wikipedia.org/wiki/X86) thông qua dự án [Android x86](https://vi.wikipedia.org/w/index.php?title=Android_x86&action=edit&redlink=1), và [Google TV](https://vi.wikipedia.org/wiki/Google_TV) cũng sử dụng một phiên bản x86 đặc biệt của Android.Nhân Linux dùng cho Android đã được Google thực hiện nhiều thay đổi về kiến trúc so với nhân Linux gốc. Android không có sẵn [X Window System](https://vi.wikipedia.org/w/index.php?title=X_Window_System&action=edit&redlink=1) cũng không hỗ trợ các thư viện [GNU](https://vi.wikipedia.org/wiki/D%E1%BB%B1_%C3%A1n_GNU) chuẩn, nên việc chuyển các ứng dụng hoặc thư viện Linux có sẵn sang Android rất khó khăn. Các ứng dụng C đơn giản và [SDL](https://vi.wikipedia.org/wiki/Simple_DirectMedia_Layer) cũng được hỗ trợ bằng cách chèn những đoạn [shim](https://vi.wikipedia.org/w/index.php?title=Shim&action=edit&redlink=1) Java và sử dụng tương tự [JNI](https://vi.wikipedia.org/w/index.php?title=JNI&action=edit&redlink=1), như khi người ta chuyển [Jagged Alliance 2](https://vi.wikipedia.org/wiki/Jagged_Alliance_2) sang Android.[Bộ lưu trữ flash](https://vi.wikipedia.org/wiki/B%E1%BB%99_nh%E1%BB%9B_flash) trên các thiết bị Android được chia thành nhiều phân vùng, như "/system" dành cho hệ điều hành và "/data" dành cho dữ liệu người dùng và cài đặt ứng dụng. Khác với các bản phân phối Linux cho máy tính để bàn, người sở hữu thiết bị Android không được trao quyền truy cập [root](https://vi.wikipedia.org/w/index.php?title=Superuser&action=edit&redlink=1) vào hệ điều hành và các phân vùng nhạy cảm như /system được thiết lập [chỉ đọc](https://vi.wikipedia.org/wiki/B%E1%BB%99_nh%E1%BB%9B_ch%E1%BB%89_%C4%91%E1%BB%8Dc). Tuy nhiên, quyền truy cập root có thể chiếm được bằng cách tận dụng những [lỗ hổng bảo mật](https://vi.wikipedia.org/w/index.php?title=L%E1%BB%97_h%E1%BB%95ng_(m%C3%A1y_t%C3%ADnh)&action=edit&redlink=1) trong Android, điều mà [cộng đồng mã nguồn mở](https://vi.wikipedia.org/wiki/Android_(h%E1%BB%87_%C4%91i%E1%BB%81u_h%C3%A0nh)#C.E1.BB.99ng_.C4.91.E1.BB.93ng_m.C3.A3_ngu.E1.BB.93n_m.E1.BB.9F) thường xuyên sử dụng để nâng cao tính năng thiết bị của họ, kể cả bị những người ác ý sử dụng để cài [virus](https://vi.wikipedia.org/wiki/Virus_(m%C3%A1y_t%C3%ADnh)) và [phần mềm ác ý](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m_%C3%A1c_%C3%BD).Việc Android có được xem là một bản phân phối Linux hay không vẫn còn là vấn đề gây tranh cãi, tuy được [Linux Foundation](https://vi.wikipedia.org/w/index.php?title=Linux_Foundation&action=edit&redlink=1) và [Chris DiBona](https://vi.wikipedia.org/w/index.php?title=Chris_DiBona&action=edit&redlink=1), trưởng nhóm mã nguồn mở Google, ủng hộ. Một số khác, như *linux-magazine.com* thì không đồng ý, do Android không không hỗ trợ nhiều công cụ GNU, trong đó có [glibc](https://vi.wikipedia.org/w/index.php?title=Glibc&action=edit&redlink=1).

1. **Cộng đồng mã nguồn mở**

Android có một cộng đồng các lập trình viên và những người đam mê rất năng động. Họ sử dụng mã nguồn Android để phát triển và phân phối những phiên bản chỉnh sửa của hệ điều hành. Các bản Android do cộng đồng phát triển thường đem những tính năng và cập nhật mới vào nhanh hơn các kênh chính thức của nhà sản xuất/nhà mạng, tuy không được kiểm thử kỹ lưỡng cũng như không có đảm bảo chất lượng; cung cấp sự hỗ trợ liên tục cho các thiết bị cũ không còn nhận được bản cập nhật chính thức; hoặc mang Android vào những thiết bị ban đầu chạy một hệ điều hành khác, như [HP Touchpad](https://vi.wikipedia.org/w/index.php?title=HP_Touchpad&action=edit&redlink=1). Các bản Android của cộng đồng thường được [root](https://vi.wikipedia.org/wiki/Root_(Android)) sẵn và có những điều chỉnh không phù hợp với những người dùng không rành rẽ, như khả năng [ép xung](https://vi.wikipedia.org/wiki/T%C4%83ng_t%E1%BB%91c_ph%E1%BA%A7n_c%E1%BB%A9ng_m%C3%A1y_t%C3%ADnh) hoặc [tăng/giảm áp](https://vi.wikipedia.org/w/index.php?title=Thay_%C4%91%E1%BB%95i_%C4%91i%E1%BB%87n_%C3%A1p_%C4%91%E1%BB%99ng&action=edit&redlink=1) bộ xử lý của thiết bị. [CyanogenMod](https://vi.wikipedia.org/wiki/CyanogenMod) là firmware của cộng đồng được sử dụng phổ biến nhất, và hoạt động như một tổ chức của số đông khác.Trước đây, nhà sản xuất thiết bị và nhà mạng tỏ ra thiếu thiện chí với việc phát triển [firmware](https://vi.wikipedia.org/wiki/Firmware) của bên thứ ba. Những nhà sản xuất còn thể hiện lo ngại rằng các thiết bị chạy phần mềm không chính thức sẽ hoạt động không tốt và dẫn đến tốn tiền hỗ trợ. Hơn nữa, các firmware đã thay đổi như CyanogenMod đôi khi còn cung cấp những tính năng, như [truyền tải mạng](https://vi.wikipedia.org/w/index.php?title=Truy%E1%BB%81n_t%E1%BA%A3i_m%E1%BA%A1ng&action=edit&redlink=1) (tethering), mà người dùng bình thường phải trả tiền nhà mạng mới được sử dụng. Kết quả là nhiều thiết bị bắt đầu đặt ra hàng rào kỹ thuật như khóa [bootloader](https://vi.wikipedia.org/w/index.php?title=Bootloader&action=edit&redlink=1) hay hạn chế quyền truy cập root. Tuy nhiên, khi phần mềm do cộng đồng phát triển ngày càng trở nên phổ biến, và sau một thông cáo của Thư viện Quốc hội Hoa Kỳ cho phép "[jailbreak](https://vi.wikipedia.org/w/index.php?title=IOS_jailbreaking&action=edit&redlink=1)" (vượt ngục) thiết bị di động, các nhà sản xuất và nhà mạng đã tỏ ra mềm mỏng hơn với các nhà phát triển thứ ba, thậm chí một số hãng như [HTC](https://vi.wikipedia.org/wiki/HTC), [Motorola](https://vi.wikipedia.org/wiki/Motorola), [Samsung](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_%C4%91o%C3%A0n_Samsung) và [Sony](https://vi.wikipedia.org/wiki/Sony_Mobile_Communications), còn hỗ trợ và khuyến khích phát triển. Kết quả của việc này là dần dần nhu cầu tìm ra các hạn chế phần cứng để cài đặt được firmware không chính thức đã bớt đi do ngày càng nhiều thiết bị được phát hành với [bootloader](https://vi.wikipedia.org/w/index.php?title=Bootloader&action=edit&redlink=1) đã mở khóa sẵn hoặc có thể mở khóa, tương tự như điện thoại dòng [Nexus](https://vi.wikipedia.org/wiki/Google_Nexus), tuy rằng thông thường họ sẽ yêu cầu người dùng từ bỏ chế độ bảo hành nếu họ làm như vậy. Tuy nhiên, tuy được sự chấp thuận của nhà sản xuất, một số nhà mạng tại Mỹ vẫn bắt buộc điện thoại phải bị khóa. Việc mở khóa và "hack" điện thoại thông minh và máy tính bảng vẫn còn là tác nhân gây căng thẳng giữa cộng đồng và công nghiệp. Cộng đồng luôn biện hộ rằng sự hỗ trợ không chính thức ngày càng trở nên quan trọng trước việc nền công nghiệp không cung cấp các bản cập nhật thường xuyên và/hoặc ngưng hỗ trợ cho chính các thiết bị của họ.

1. **Tỷ lệ sử dụng các phiên bản Android**

Tỷ lệ sử dụng các phiên bản khác nhau tính đến tháng 4 năm 2014. Phần lớn các thiết bị Android cho tới nay vẫn chạy hệ điều hành phiên bản 4.1.x Jelly Bean được phát hành ngày 9 tháng 7 năm 2012 nhờ tính ổn định và hỗ trợ tốt các máy có cấu hình thấp.

****

****

1. **Công cụ phát triển**
2. **IDE Android Studio**

Trước năm 2015, để lập trình ứng dụng Android bạn cần cài đặt Eclipse (Vốn là một công cụ lập trình ứng dụng Java), sau đó bạn cần phải cài đặt thêm plugin vào Eclipse, nó sẽ cho phép bạn lập trình các ứng dụng Android trên Eclipse.Năm 2015 Google cho ra mắt **Android Studio**, một công cụ lập trình dành riêng cho ứng dụng **Android**, và chính thức không hỗ trợ Plugin cho Eclipse nữa. Về mặt logic những công cụ gì được tạo ra dành cho một nhiệm vụ chuyên biệt bao giờ cũng tốt hơn.  Điều đó có nghĩa là để lập trình **Android** bạn  nên chọn **Android Studio**.

1. **Genymotion**

Genymotion là một máy ảo chạy nhanh rất nhiều so với Android Emulator và giúp tiết kiệm thời gian test khi chạy ứng dụng khi bạn [lập trình android](https://hocthietkeweb.net.vn/khoa-hoc-lap-trinh-game-android-co-ban.html).

**Tính năng đặc trưng của Genymotion :**

* Genymotion shell cho phép tương tác với máy ảo bằng cách sử dụng dòng lệnh.
* Hỗ trợ ADB.
* Eclipse và Android Studio plugin.
* Hỗ trợ Linux, Windows và Mac.

1. **Sqlite manager**

Bất kì một ứng dụng nào dù lớn hay nhỏ chúng ta hầu như đều phải sử dụng cơ sở dữ liệu. Khi sử dụng SQLite các bạn có thể tạo cơ sở dữ liệu trực tiếp trong code nhưng việc làm này khá tốn thời gian, và để giải quyết điều này chúng ta có thể sử dụng Sqlite Manager được tích hợp như là plug-in của Firefox Sqlite Manager cho phép chúng ta Thêm, Sửa, Xóa các trường, các hàng, các cột thậm chí là các bảng một cách trực quan trên giao diện, điều này giúp lập trình viên dễ mường tượng và thao tác dễ dàng hơn với cơ sở dữ liệu

1. **Các thư viện, công nghệ sử dụng**
2. **ButterKnife**

ButterKnife là một sản phẩm của Jake Wharton, một người sau khi đã phát chán với việc phải tự tay viết các câu findViewById khi lập trình Android và anh đã nảy ra ý tưởng về một API giúp giảm bớt thời gian và sự ức chế đó mỗi khi lập trình giao diện trên Android. Nói cơ bản thì khi sử dụng API này, các lập trình viên Android có thể quên luôn cái method findViewById đi, không cần sử dụng đến nó nữa

Việc sử dụng Butter Knife cực kì đơn giản, gồm 3 bước :

**Bước 1: Thêm thư viện vào file build.gradle**

Rất đơn giản, chỉ cần thêm 2 dòng sau vào phần dependencies trong file build.gradle của module

****

### Bước 2: Bind target

### Việc bind target là để butter knife được xem bạn định sử dụng layout nào và trong layout đó có các thành phần nào để từ đó tự động gán các field mà bạn khai báo ở bước 3 với các thành phần tương ứng trong layout. Lưu ý là bước này các bạn chỉ cần thực hiện 1 lần đối với mỗi 1 activity, fragment hay ViewHolder thôi

### Bước 3: Gán view hoặc resouce

### *Đối với view :*

### Butter Knife hỗ trợ bạn 1 cách rất đơn giản để có thể gán thuộc tính (field) tương ứng với view trên layout, điều này sẽ giúp bạn giảm đi được n lần lặp đi lặp lại để viết findViewById

### Để làm điều này, đơn giản chỉ thêm tiền tố(annotation) [@BindView](https://kipalog.com/users/BindView/mypage)(id\_view) vào trước khai báo của thuộc tính tương ứng với view đó.

### C:\Users\hnc\Desktop\2017-04-26_085028.jpg

### *Đối với resource khác như string, drawable, color…*

### Rất đơn giản, tương tự như với view, các bạn cũng có thể sử dụng các tiền tố (annotation) [@BindBool](https://kipalog.com/users/BindBool/mypage), [@BindColor](https://kipalog.com/users/BindColor/mypage), [@BindDimen](https://kipalog.com/users/BindDimen/mypage), [@BindDrawable](https://kipalog.com/users/BindDrawable/mypage), [@BindInt](https://kipalog.com/users/BindInt/mypage), [@BindString](https://kipalog.com/users/BindString/mypage), … để gán resource với field tương ứng

### 

### Bước 4: Gán listener cho các view

### Không chỉ hỗ trợ các bạn trong việc gán (bind) view hay resource mà Butter Knife còn hỗ trợ các bạn trong việc gán (bind) các listener cho các view tương ứng

### 

### Các bạn có thể để ý thấy là ta chỉ việc thêm [@OnClick](https://kipalog.com/users/OnClick/mypage)(view\_id) sau đó viết hàm handler sự kiện onClick là xong chứ không cần thiết phải qua từng bước getViewById, implement OnClickListener rồi setOnClickListener như trước nữa, rất tiện.

### Jsoup

Như đã nói **Jsoup** là một thư viện của **Java** để cung câp một bộ các API cho việc trích xuất và thao tác các file **HTML** . Có một vài phương thức cho việc đọc và phân tích một trang **HTML** , trong trường hợp của chúng ta, chúng ta muốn lấy một vài thông tin từ một trang HTML trên server ở xa và chúng ta phải cung cấp một **URL**. Nếu chúng ta muốn phân tích trang như là DOM, chúng ta có:

Document doc = Jsoup.connect(URL).get();

**doc** là một biến thuộc class **Document** để lưu trữ những tài liệu đã được lấy về. Bây giờ chúng ta đã có **Document** và chúng ta có thể tự do trích xuất thông tin. Chúng ta có thể lấy title và những thông tin khác đang được sử dụng trong các\*\* Tag HTML\*\*  
Cho ví dụ, nếu chúng ta muốn lấy tất cả các tag có tên là\*\* meta \*\*:

Elements metaElems = doc.select("meta");

Phương thức **select** được sử dụng khi chúng ta muốn lấy về những tag đang sử dụng truy vấn **CSS** . Cho ví dụ, nếu chúng ta muốn lấy giá trị thuộc tính từ một tag :

String name = metaElem.attr("name");

**name** ở đây là tên thuộc tính. Hơn nữa, Chúng ta có thể lựa chọn tất cả các thành phần trong một trang HTML có một giá trị CSS xác định. Cho ví dụ, trong website có một vài thành phần có một class CSS tên là \*\*“topic" \*\*, vậy chúng tao có.

Elements topicList = doc.select("h2.topic");

1. **Retrofit**

[Retrofit](https://square.github.io/retrofit/) là một HTTP client type-safe cho Android và Java. Retrofit giúp dễ dàng kết nối đến một dịch vụ REST trên web bằng cách chyển đổi API thành Java Interface. Retrofit là một trong những thư viện HTTP phổ biến nhất và được khuyến dùng nhất cho Android.

Thư viện mạnh mẽ này giúp bạn dễ dàng xử lý dữ liệu JSON hoặc XML sau đó phân tích cú pháp thành Plain Old Java Objects (POJOs). Tất cả các yêu cầu GET, POST, PUT, PATCH, và DELETE đều có thể được thực thi.

Giống như hầu hết các phần mềm mã nguồn mở khác, Retrofit được xây dựng dựa trên một số thư viện mạnh mẽ và công cụ khác. Đằng sau nó, Retrofit làm cho việc sử dụng [OkHttp](http://square.github.io/okhttp/) (từ cùng một nhà phát triển) để xử lý các yêu cầu trên mạng. Ngoài ra, Retrofit không tích hợp bất kỳ một bộ chuyển đổi JSON nào để phân tích từ JSON thành các đối tượng Java. Thay vào đó nó đi kèm với các thư viện chuyển đổi JSON sau đây để xử lý điều đó:

* Gson: com.squareup.retrofit:converter-gson
* Jackson: com.squareup.retrofit:converter-jackson
* Moshi: com.squareup.retrofit:converter-moshi
* Protobuf: com.squareup.retrofit2:converter-protobuf
* Wire: com.squareup.retrofit2:converter-wire

Và đối với XML, Retrofit hỗ trợ:

* SimpleFramework:com.squareup.retrofit2:converter-simpleframework

Việc phát triển thư viện HTTP type-safe của chính bạn để giao tiếp với REST API có thể thật sự là một điều khó khăn: bạn phải xử lý nhiều chức năng chẳng hạn như tạo các kết nối, lưu trữ cach, thử lại các yêu cầu thất bại, phân tích luồng phản hồi, xử lý lỗi, và nhiều hơn nữa. Retrofit, mặt khác, được hoạch định rất tốt, có tài liệu hướng dẫn và kiểm đã được kiểm thử—một thư viện đã được thử nghiệm rất nhiều trong thực tế sẽ giúp bạn tiết kiệm rất nhiều thời gian quý báu và giảm bớt căng thẳng.

1. **Gson**

Gson là một thư viện java cho phép người sử dụng có thể chuyển đổi từ một đối tượng Java sang JSON và cũng có thể chuyển đổi từ một đối tượng JSON sang java.Gson có thể làm việc với đối tượng java tùy ý bao gồm các đối tượng tồn tại sẵn mà bạn không có source-code của chúng.

1. **Picasso**

Khi phát triển các ứng dụng có nhiều ảnh hoặc yêu cầu phải load và hiển thị ảnh từ internet thì rất nhiều bạn sẽ băn khoăn về cách làm sao cho vừa không bị “đơ” UI (không gây khó chịu cho người dùng) lại vừa dễ dàng trong việc viết code và quản lý code. Trước đây mình cũng đã từng rất băn khoăn về vấn đề này bởi nếu dùng Loader hay AsyncTask thì code sẽ khá phức tạp, còn nếu cứ load thẳng ảnh trong Main Thread thì người dùng sẽ khó chịu vì không thể tương tác với UI, nhưng từ khi chuyển sang sử dụng Picasso thì các vấn đề này đã không còn nữa bởi vì chính Picasso đã lo hết những phần này rồi. Chính vì vậy, trong bài viết này mình muốn giới thiệu tới các bạn về Picasso, một thư viện chuyên về download và hiển thị ảnh từ internet rất tuyệt vời cho Android.

Picasso là một thư viện chuyên về load ảnh với các ưu điểm sau:

* **Đơn giản hóa code** chính là ưu điểm hàng đầu của Picasso. Thay vì phải viết hàng chục thậm chí là cả trăm dòng code thì với thư viện này bạn sẽ chỉ mất vài dòng thậm chí là chỉ 1 dòng code là bạn đã hoàn thành việc load ảnh tử internet vào ImageView của bạn. Đây chính là điểm tuyệt vời nhất khi bạn sử dụng thư viện load ảnh này.
* **Tối ưu hóa bộ nhớ** sử dụng của ứng dụng
* **Tự động sử dụng RAM cache và Disk cache** để tối ưu hóa lưu lượng mạng và tăng tốc độ load ảnh
* **Tự động recycle bitmap và cancel download theo ngữ cảnh**. Nhờ ưu điểm này mà bạn sẽ không còn phải để ý xem mình đã recycler bitmap hay chưa, hay có cần bắt exception khi quá trình download bị cancel hay không… Tất tần các công việc này hãy để thư viện lo và bạn chỉ cần viết code để xử lý cho luồng chính mà thôi.

Không chỉ dừng lại ở việc load ảnh, Picasso còn cung cấp cho chúng ta 1 số tùy biến khác để việc load ảnh và handle các tình huống được đơn giản và dễ dàng hơn. Các tùy biến đó bao gồm:

* Placeholder() : Tùy chọn placeholder() cho phép chúng ta hiển thị 1 ảnh nào đó trên imageView trong quá trình mà Picasso đang thực hiện load bức ảnh mà ta cần load vào ImageView, ví dụ như bạn có thể hiển thị hình ảnh loading… hoặc biểu tượng bất kì nào đó để thể hiện cho người dùng biết là ảnh đang trong quá trình loading.
* Error() : Tùy chọn error() cho phép bạn hiển thị một bức ảnh nào đó nếu như việc load bức ảnh vào imageView không thể hoàn thành được. Với tùy chọn này, các bạn có thể dễ dàng thể hiện cho người dùng việc load ảnh hiện đang gặp vấn đề và cần phải tải lại.

Load ảnh từ Resource, Assets, file … : Không chỉ cho phép load ảnh từ internet, Picasso còn cho phép chúng ta load ảnh từ Resources, Assets của application hoặc từ files nằm trên bộ nhớ trong hoặc thẻ nhớ của máy.

1. **Material Design**

Đội ngũ thiết kế của Google đã tạo nên một loạt nguyên tắc về diện mạo của các ứng dụng. Nó bao gồm việc đơn giản hóa và tiêu chuẩn hóa các thành phần đồ họa của Android, từ việc tạo các công cụ chọn bảng màu cho tới nâng cấp font chữ Roboto mà Google đã áp dụng trên Android trong vòng 1 năm qua. Tất cả đã rõ ràng hơn, xuyên suốt trên nhiều thiết bị khác nhau.

Không chỉ về giao diện ứng dụng mà còn là tạo nên một tiêu chuẩn chung cho tất cả, điều này rất quan trọng cho Google trong tương lai. Bởi vì Material Design không chỉ áp dụng cho không gian chữ nhật trên điện thoại hay tablet, nó sẽ được vận dụng cho màn hình dạng tròn trên đồng hồ đeo tay thông minh. Hay thậm chí là trên kính mắt (Google Glass). Nó đơn giản hơn, sáng hơn và rõ ràng hơn. Nó rất quan trọng bởi Material Design sẽ xuất hiện trên nhiều thiết bị hơn bất kỳ giao diện người dùng (UI) nào khác trong lịch sử.

Như Mattias Duarte trưởng bộ phận thiết kế của Google đã giải thích, Material Design là tính ba chiều của mọi sự vật: hãy nghĩ tới các điểm ảnh (pixel) không chỉ là điểm 2 chiều của màu sắc, nó là các khối hình 3 chiều với chiều sâu và chiều cao. Lấy ví dụ với một thẻ Google Now. Khi nó xuất hiện trên màn hình điện thoại, cửa sổ đó không chỉ là tập hợp của những khung hình vuông màu trắng mà nó có những đặc tính giống như một thẻ thật ngoài đời khi di chuyển trên màn hình của bạn. Điều tương tự với các hiệu ứng như đổ bóng hay phối cảnh. Material Design sẽ biến thẻ Google Now như thể nó đang nổi trên màn hình đi kèm với hiệu ứng đổ bóng vật lý và độ sâu.

Như vậy có thể thấy Material Design rất khác so với Flat Design mà chúng ta đã nghe nhiều về nó. Thiết kế phẳng đề cao sự đơn giản với ít tính ba chiều nhưng với Material Design thì Google đã mang rất nhiều tính ba chiều vào Android. Và nó cũng có nguyên nhân cả.

#### Một số đặc điểm có thể dùng để nhận biết những ứng dụng Material Design như sau:

* Sử dụng các màu nổi bật, thường có một mảng màu chủ đạo nằm ở cạnh trên ứng dụng
* Các biểu tượng phẳng, đơn giản nhưng dễ hiểu
* Một số ứng dụng sẽ có một nút tròn to nằm ở góc dưới bên phải, thường có chức năng tạo mới
* Giao diện phẳng, ít hoặc không có hiệu ứng chuyển màu, có hoặc không có hiệu ứng đổ bóng đen
* Menu, nút nhấn, chữ viết… có nhiều khoảng cách trắng nên trông thoáng đãng
* Có các hiệu ứng chuyển động tự nhiên, dễ hiểu, có thể gợi ý cho một tính năng nào đó

#### ****Lợi ích của Material Design là gì?****

Như đã nói ở trên, Material Design là một ngôn ngữ được Google tạo ra nhằm “phát triển một hệ thống nền duy nhất cho phép xây dựng các trải nghiệm đồng bộ giữa nhiều loại và kích thước thiết bị khác nhau. Việc thiết kế cho thiết bị di động được đặt lên hàng đầu, nó còn phải chú trọng đến việc nhập liệu bằng cảm ứng, giọng nói, chuột, bàn phím”.

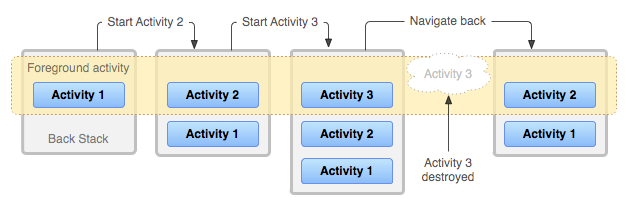
Thông qua lời nói trên, chúng ta có thể thấy là Google muốn mang lại trải nghiệm thống nhất cho người dùng trong hệ sinh thái của hãng. Người dùng cuối như chúng ta sẽ có cảm giác quen thuộc hơn, dành ít thời gian để học hỏi hơn khi sử dụng cùng một app nhưng trên nhiều máy khác nhau. Các lập trình viên thì có thể đảm bảo rằng cách người dùng trải nghiệm app của họ là như nhau không quan trọng thiết bị đang chạy là gì.

Theo thiết kế của Google, nội dung trên trang web và [ứng dụng android](http://devpro.edu.vn/khoa-hoc-lap-trinh-android/) sẽ đồng nhất về hình ảnh và giao diện, trong đó có tùy chỉnh về mặt hiển thị trên các thiết bị khác nhau. Material Design đã cung cấp một giải pháp dễ dàng để thực hiện điều này, thiết kế ưu việt và tối ưu hóa tốt.

1. **Các kiến thức core áp dụng** 
   1. **Activity**

**Activity là gì ?**

Activity trong Android được xem là một **màn hình tương tác** trong ứng dụng Android. Activity chịu trách nhiệm chuyển giao sự kiện cho các view trong nó và quản lý vòng đời (LifeCycle) của nó. Một ứng dụng Android có thể có một hoặc nhiều Activity. Và những Activity này có vòng đời độc lập với nhau và được duy trì trong Activity BackStack



* Bạn đang ở Activity 1 và từ Activity này bạn chuyển sang Activity 2. Khi đó Activty 1 sẽ được đẩy xuống dưới stack và Activity 2 nằm trên cùng của Stack.
* Từ Activity 2 bạn lại chuyển sang Activity 3 thì tiếp tục Activty 2 lại được đẩy xuống, Activity 3 nằm ở trên stack.
* Ở Activty 3 bạn nhất nút back thì activity 3 được destroy và đẩy ra khỏi stack. Khi đó Activity thứ 2 sẽ được đưa lên top của stack và tương tác với người dùng.

**Vòng đời của Activity**

Dưới dây là Diagram mô tả vòng đời của ứng dụng Android.



### onCreate()

Phương thức callback này được gọi khi ứng dụng mới start, nhưng chưa có thể tương tác với người dùng. Ở trong phương thức này chúng ta thường sử dụng để setContentView(), khởi tạo cái view, inflate view từ xml đăng kí event cho view, khởi tạo Adapter cho ListView, RecyclerView…

### onStart()

Phương thức callback này được gọi ngay sau khi onCreate()b được gọi. Ở phương thức này chúng ta vẫn chưa tương tác được với người dùng trên UI.

### onResume()

Phương thức này được gọi ngay sau khi onStart() được gọi. Ở phương thức này chúng ta có thể tương tác được với UI.

### onPause()

Phương thức callback này được sử dụng khá phổ biến trong vòng đời của Activity. Chúng ta thường lưu lại trạng thái của các biến trong method này.

### onStop()

Phương thức callback này có thể được gọi ngay sau khi phương thức onPause được gọi. Phương thức này được gọi khi chúng ta nhấn nút home app.

### onRestart()

Phương thức callback này gọi khi activity đã stoped, gọi trước khi bắt đầu start lại Activity.

### onDestroy()

Phương thức này đợc gọi khi chúng ta nhấn back từ activity, hoặc call method finish() của activity.

## Chuyển đổi Activity trong kèm theo dữ liệu

Việc chuyển đổi Activity mà không gửi kèm dữ liệu cực kì đơn giản. Chúng ta chỉ cần tạo một đối tượng Intent sử dụng Constructor dưới đây

public Intent(Context packageContext, Class<?> cls) {

Với đối số thứ nhất bạn truyền vào là Activity hiện tại.

Và đối số thứ hai là Activity mà bạn muốn chuyển hướng.

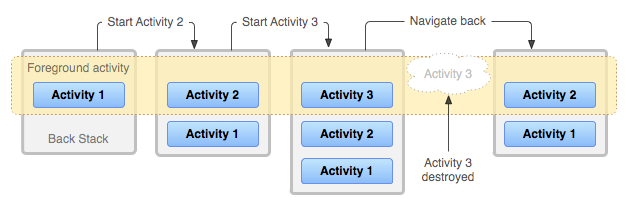
Ví dụ tôi muốn chuyển hướng từ MainActivity sang AboutAcitivity thì tôi sẽ tạo Intent như sau:

Intent intent = Intent(MainActivity.this,AboutActivity.class);

Và cuối cùng để thực hiện chuyển Activity chúng ta sử dụng method của lớp Context.

public void startActivity(Intent intent);

Khi chúng ta chuyển trang như vậy thì MainActivity sẽ được đẩy xuống dưới **BackStack** và **AboutActivity** được add vào đầu của **BackStack**. Vì vậy khi mà chúng ta nhấn back từ **AboutActivity** thì **MainActivity** được đẩy lên và vào lại **onResume**.



Chuyển Activity trong Android rất dễ truy nhiên bạn phải nhớ được Life Cycle (onCreate, onStart, onResume, onPause, onStop, onStart, onDestroy…) của từng Activity khi chúng ta gọi **startActivity**.

***Lưu ý:*** Mọi Activity mà bạn tạo ra phải được khai báo trong file **AndroidManifest.xm**l. Nếu bạn không khai báo mà cố tình sử dụng Activity này thì hệ thống sẽ ném ra ngoại lệ nên các bạn hết sức lưu ý. Điều này cũng được áp dụng khi các bạn sử dụng Service trong Android.

### Gửi dữ liệu đi

#### Gửi dữ liệu đi riêng lẻ

Intent intent = new Intent(MainActivity.this, AboutActivity.class);

                intent.putExtra("int\_key", 4);

                intent.putExtra("char\_key", 'r');

                intent.putExtra("boolean\_key", true);

                intent.putExtra("long\_key", 323L);

                intent.putExtra("float\_key", 3.2f);

                intent.putExtra("string\_key", "Chuyen Activity trong Android");

                intent.putExtra("double", 32D);

                intent.putExtra("int\_array\_key", new int[]{1, 2, 9});

                intent.putExtra("char\_array\_Key", new char[] {'e', 'i', 't', 'g', 'u', 'i', 'd', 'e'});

                intent.putExtra("rect\_key", new Rect(0,0, 200, 200));

                startActivity(intent);

#### Gửi dữ liệu đi sử dụng Bundle

Thay vì cách sử dụng là put vào intent thì chúng ta sẽ sử dụng một đối tượng gọi là Bundle. Chúng ta sẽ put các dữ liệu vào đối tượng này và sử dụng phương thức **putExtras** để put “cả cục” này vào intent.

Bundle bundle = new Bundle();

                bundle.putChar("char", 'e');

                bundle.putInt("int", 3);

                bundle.putFloat("float", 5.2f);

                bundle.putParcelable("parcelable", new Rect(0, 0, 300, 300));

                intent.putExtras(bundle);

                startActivity(intent);

### Nhận dữ liệu

Sau khi startActivity thì ở Activity bạn muốn chuyển sẽ có được dữ liệu mà activity kia gửi đi. Và để lấy được dữ liêu chúng ta phải làm như sau:

Lấy ra đối tượng Intent mà bên kia gửi qua bằng phương thức getIntent();

Từ đối tượng đối tượng intent có được từ getIntent() chúng ta bắt đầu lấy ra những dữ liệu trong đối tượng này bằng phương thức **getXExtra**. Với X là kiêu dữ liệu mà bạn muốn lấy ra. Key lấy ra phải đúng với key bạn truyền vào.

int intValue = intent.getIntExtra("int\_key", 0);

        char charValue = intent.getCharExtra("char\_key", 'a');

        float floatValue = intent.getFloatExtra("float\_key", 0f);

        boolean booleanValue = intent.getBooleanExtra("boolean\_key", false);

        String stringValue = intent.getStringExtra("string\_key");

        int[] intArrayValue = intent.getIntArrayExtra("int\_array\_key");

        char[] charArrayValue = intent.getCharArrayExtra("char\_array\_key");

        Rect rect = intent.getParcelableExtra("rect\_key");

* 1. **Fragment**

Fragment cho phép phân chia Activity thành các thành phần được đóng gói đầy đủ và tái sử dụng được. Các Fragment có vòng đời và giao diện UI riêng của mình.

Ưu điểm chính của Fragment là dễ dàng tạo ra thiết kế giao diện người dùng (UI) động và linh hoạt, phù hợp với các kích cỡ màn hình khác nhau từ màn hình nhỏ trên điện thoại thông minh đến kích thước màn hình máy tính bảng.

Mỗi Fragment là một module độc lập có ràng buộc chặt chẽ vào Activity mà nó gắn vào. Fragment có thể được tái sử dụng trong nhiều Activity, và trong một activity có thể được gắn vào nhiều Fragment.

Fragment cung cấp cách để trình bày giao diện người dùng (UI) phù hợp và tối ưu cho các loại thiết bị [Android](https://viblo.asia/tuanhv/posts/zb7vD8gQvjKd) có kích thước màn hình và mật độ điểm ảnh khác nhau.

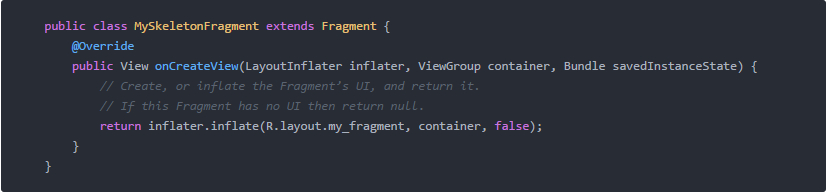
Fragment được giới thiệu trong phiên bản [Android](https://viblo.asia/tuanhv/posts/zb7vD8gQvjKd) 3.0 Honeycomb (API level 11).

**Tạo một Fragment**

Extend lớp Fragment để tạo một fragment mới, có thể định nghĩa UI và implement các chức năng đóng gói của nó.

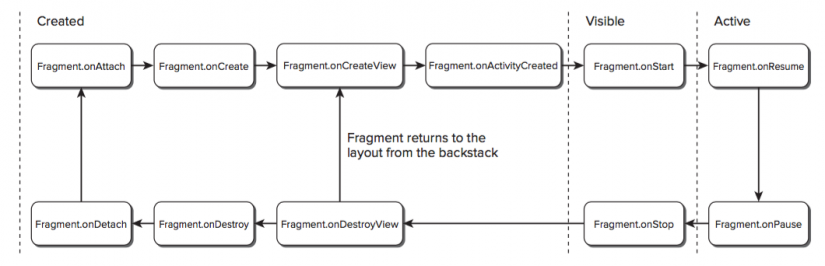
Trong hầu hết các trường hợp, bạn sẽ muốn chỉ định UI cho fragment. Tuy nhiên, bạn cũng có thể sử dụng một fragment để cung cấp một hành vi chạy ngầm cho hoạt động mà không cần đưa UI bổ sung.

Để cung cấp một layout cho một fragment, phải triển khai phương thức gọi lại onCreateView(), phương thức này được hệ thống [Android](https://viblo.asia/tuanhv/posts/zb7vD8gQvjKd) gọi khi đến khi fragment vẽ layout của nó. Việc triển khai phương thức này phải trả về một View là phần gốc cho layout của fragment.



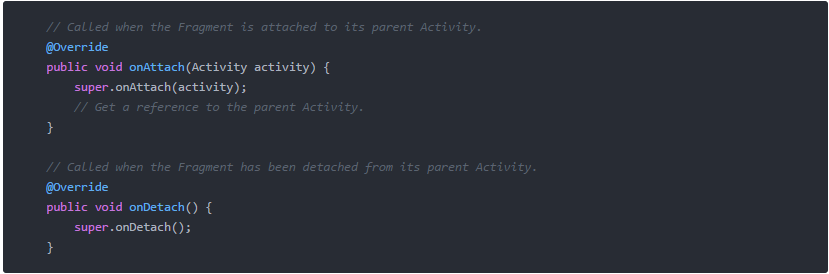
**Vòng đời Fragment**

Việc quản lý vòng đời của một fragment rất giống với quản lý vòng đời của một activity. Tuy nhiên, vòng đời của fragment chịu ảnh hưởng bởi vòng đời của activity mà nó gắn vào. Chỉ khi activity ở trạng thái resumed thì vòng đời của fragment mới có thể thay đổi độc lập.



**Attach và Detach fragment từ Activity**

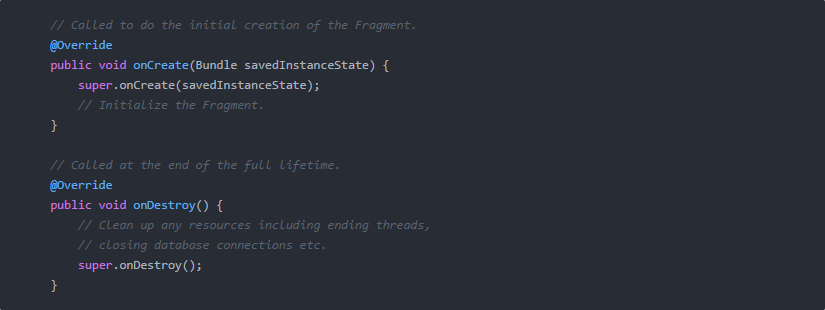
Vòng đời đầy đủ của fragment được bắt đầu khi nó được gắn với Activity cha và kết thúc khi nó được tách ra. Những sự kiện này tương ứng với những phương thức gọi lại onAttach() và onDetach().



**Create và Destroy Fragment**

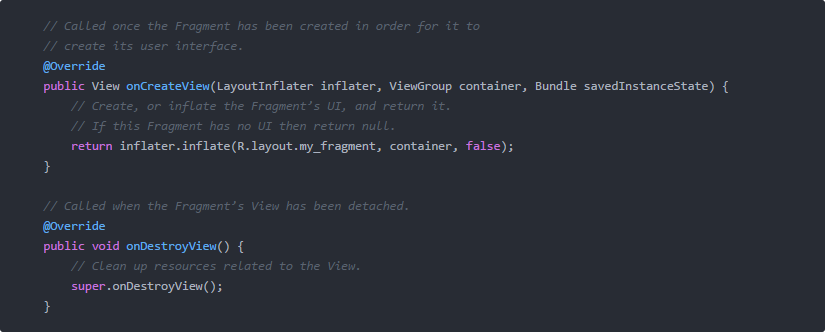
Cũng như Activity, bạn có thể sử dụng phương thức onCreate() để khởi tạo fragment. Đây cũng là vị trí tốt để tạo các đối tượng trong phạm vi class để đảm bảo chúng chỉ được tạo ra chỉ một lần trong vòng đời fragment.

Chú ý: Không giống như Activity, giao diện UI của fragment không được khởi tạo bên trong*onCreate()*



**Create và Destroy giao diện ngươi dùng (UI)**

Giao diện người dùng(UI) của fragment được khởi tạo (và huỷ) tương ứng với với việc xử lý sự kiện trong phương thức onCreateView() và onDestroyView().



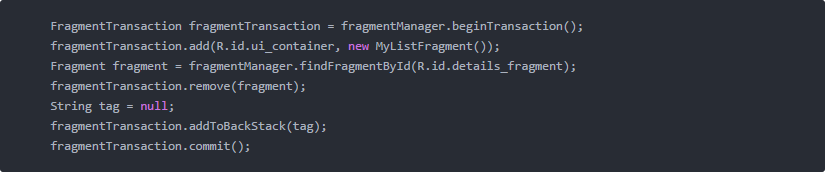
**Quản lý fragment**

Mỗi Activity có một Fragment Manager để quản lý các fragment bên trong nó. Để có nó, sử dụng phương thức getFragmentManager()



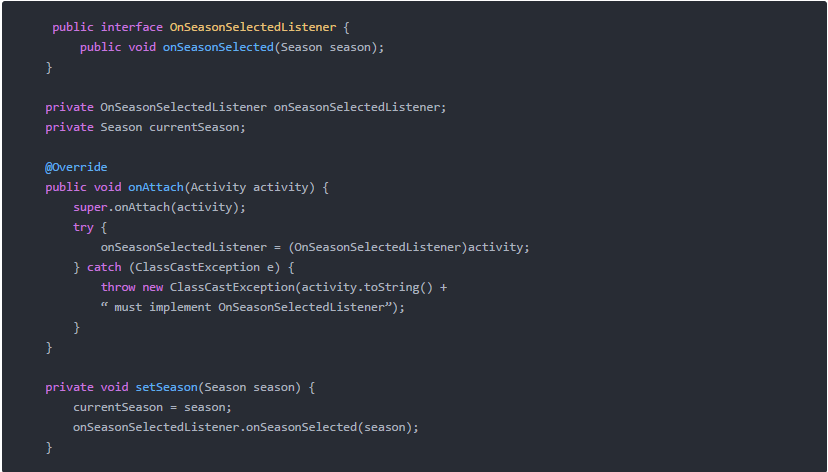
Mỗi giao tác là một tập hợp những thay đổi mà bạn muốn thực hiện tại cùng thời điểm. Bạn có thể thiết lập tất cả thay đổi mà mình muốn thực hiện đối với một giao tác cho trước bằng cách sử dụng các phương pháp như add(), remove(), và replace(). Sau đó, để áp dụng giao tác cho Activity, bạn phải gọi commit()

Trước khi bạn gọi commit(), bạn có thể muốn gọi addToBackStack(), để thêm giao tác vào một ngăn xếp của các giao tác fragment. Ngăn xếp này được quản lý bởi Activity và cho phép người dùng trở về trạng thái fragment trước đó, bằng cách nhấp nút Quay lại



**Sự kiện giữa Activity và Fragment**

Trong một số trường hợp, bạn có thể cần một fragment để chia sẻ sự kiện với Activity. Một cách hay để làm điều này đó là định nghĩa một giao diện gọi lại bên trong fragment và yêu cầu Activity cha triển khai nó. Khi Activity nhận được một lệnh gọi lại thông qua interface, nó có thể chia sẻ thông tin với các fragment khác nếu cần.



**Sub class Fragment**

[Android](https://viblo.asia/tuanhv/posts/zb7vD8gQvjKd) SDK cung cấp một số lớp con của Fragment:

* DialogFragment: Hiển thị một hộp thoại trôi nổi. Sử dụng lớp này để tạo một hộp thoại là một phương án hay cho việc sử dụng các phương pháp trình trợ giúp hộp thoại trong lớp Activity, vì bạn có thể kết hợp một hộp thoại fragment vào ngăn xếp của các fragment được quản lý bởi Activity, cho phép người dùng trả về một fragment bị bỏ.
* ListFragment: Hiển thị một danh sách các mục được quản lý bởi một trình điều hợp (chẳng hạn như một SimpleCursorAdapter), tương tự như ListActivity. Nó cung cấp một vài phương pháp để quản lý một dạng xem danh sách, chẳng hạn như phương pháp gọi lại onListItemClick() để xử lý các sự kiện nhấp.
* PreferenceFragment: Hiển thị một phân cấp các đối tượng Preference dưới dạng một danh sách, tương tự như PreferenceActivity. Điều này hữu ích khi tạo một Activity "setting" cho ứng dụng của bạn.
  1. **Service**

Service là một thành phần rất quan trọng trong ứng dụng Android. Nó đáp ứng các nhu cầu để thực hiện một hoạt động lâu trong một ứng dụng, hoặc nó cung cấp một số chức năng cho các ứng dụng khác.

**Service là gì ?**

Service là một thành phần đơn giản khởi tạo bởi hệ thống để làm một số tác vụ lâu dài mà không nhất thiết phải phụ thuộc vào tương tác người dùng. Nó có thể độc lập với vòng đời của Activity và có thể chạy ở 1 tiến trình hoàn toàn khác biệt.

Trước khi thảo luận sâu hơn về Service, chúng ta cần hiểu rõ rằng mặc dù Service thường được sử dụng cho các tác vụ chạy ở background và thực hiện trên các tiến trình khác nhau, nhưng Service không hề đại diện cho một Thread hoặc một tiến trình.

Một Service có hai tính năng chính sau :

* Một cơ sở cho ứng dụng để báo cho hệ thống về những việc nó muốn được thực hiện trong background.
* Một cơ sở cho ứng dụng để cung cấp một số chức năng của nó cho các ứng dụng khác sử dụng.

**Service và Thread**

Có rất nhiều nhầm lẫn giữa Service và Thread. Khi một Service được khai báo, nó không chứa một Thread. Thực tế , theo mặc định nó chạy trực tiếp trên main thread và bất kì công việc nào hoàn thành trong nó có khả năng tác động trực tiếp đến ứng dụng.

Vì vậy, làm thế nào các Service cung cấp một giải pháp đồng thời? Vâng, một Service không chứa một Thread theo mặc định, nhưng nó có thể dễ dàng cấu hình để làm việc với Thread của riêng mình hoặc với nhiều Thread khác nhau.

Bất chấp việc thiếu một Thread dựng sẵn, một Service là một giải pháp tuyệt vời cho vấn đề đồng bộ trong các tình huống nhất định. Những lý do chính để lựa chọn một Service so với các giải pháp đồng bộ khác như AsyncTask hoặc Hamer framework là:

* Một Service có thể độc lập với vòng đời của Activity.
* Một Service thích hợp với các tác vụ chạy lâu dài.
* Service không cần phụ thuộc vaò tương tác người dùng.
* Khi chạy trên các tiến trình khác nhau, Android có thể giữ cho các service luôn hoạt động khi hệ thống ko cung cấp đủ tài nguyên.
* Một Service có thể tái khởi động để tiếp tục công việc của nó.

**Các loại Service**

Có 2 loại Service : start và bound.

Một start service được khởi chạy thông qua Context.startService(). Nói chung nó thực hiện chỉ một tác vụ và nó sẽ chạy vô thời hạn cho đến khi tác vụ kết thúc, sau đó nó sẽ tự tắt. Thông thường, nó không trả lại bất kỳ kết quả với giao diện người dùng.

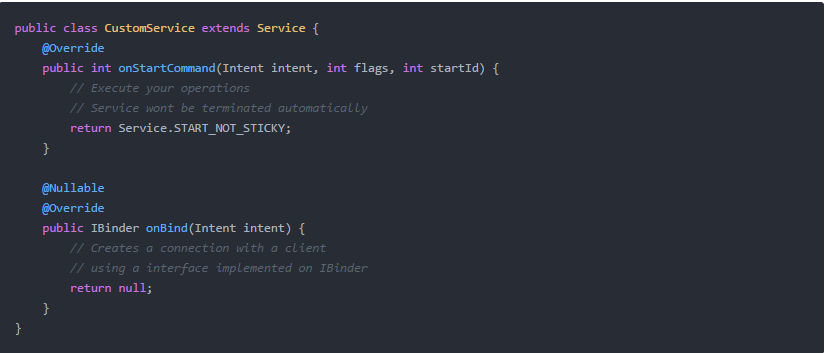
Một bound service được khởi chạy thông qua Context.bindService(), và nó cho phép 2 cách giao tiếp giữa client và Service. Nó cũng có thể kết nối với nhiều client. Nó tự tắt khi ko có client nào kết nối tới nó.

Để chọn giữa hai loại service này, Service phải implement một số hàm callback : onStartCommand() để chạy một started service, và onBind() để chạy một bound service. Một Service có thể chọn một trong hai loại kể trên, nhưng nó cũng có thể áp dụng cả hai cùng một lúc mà không gặp bất kỳ vấn đề gì.

Trong đề tài này tôi có sử dụng Bind service và bind một cách đặc biệt đó là bind service qua một ứng dụng (proccess) khác, vấn đề này tôi xin được trình bày sau.

**Service Implement**

Để sử dụng một Service, extend từ class Service và override các phương thức callback của nó tùy theo loại Service. Như đã nói ở trên, đối với started services thì implement onStartCommand(), còn đối với bound service thì implement onBind(). Trên thực tế, onBind() phải được khai báo trong cả 2 loại Service kể trên, nhưng nó có thể trả về null cho start service.



* onStartCommand() : khởi chạy bởi Context.startService(), thường gọi từ một Activity. Khi gọi xong, nó sẽ chạy vô hạn cho đến khi bạn dừng nó lại bằng cách gọi stopSelf() hoặc stopService().
* onBind() : gọi khi một thành phần muốn kết nối tới service. Được gọi trên hệ thống bởi Context.bindService(). Nó trả về một IBinder cung cấp một interface để giao tiếp với client.

Vòng đời của service cũng rất quan trọng. Phương thức onCreate() và onDestroy() nên được implement để khởi tạo và giair phóng các tài nguyên của Service.

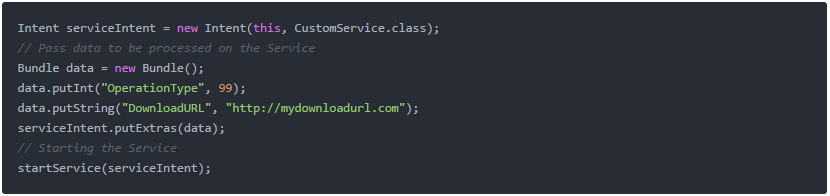
**Khai báo trong Manifest.xml**

Service được khai báo trong Manifest với thẻ :

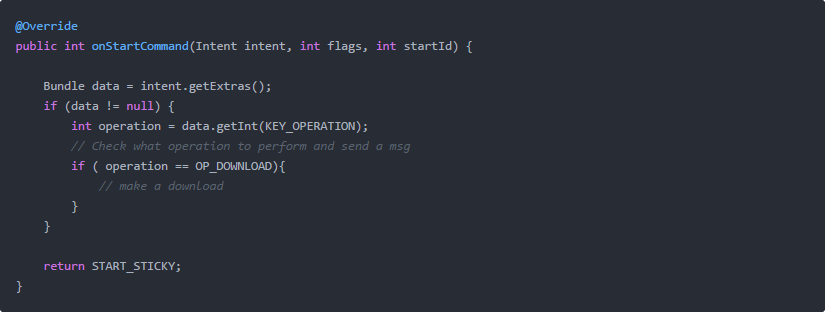


**Started Service**

Để khởi tạo một started service bạn phải gọi phương thức Context.startService(). Intent phải được tạo với Context và Service. Bất kì thông tin hoặc dữ liệu liên quan nào phải được truyền vào Intent này



Trong Service của bạn, phương thức bạn cần quan tâm là onStartCommand(). Trong phương thức này bạn sẽ gọi bất kì hoạt động nào muốn thực hiện trong started service. Bạn sẽ dùng Intent để lưu lại thông tin gửi bởi client. startId đại diện cho một ID duy nhất



onStartCommand() trả về một hằng số kiểu int điều khiển các hành vi như sau :

* Service.START\_STICKY : Service được khởi động lại nếu nó bị tắt đi.
* Service.START\_NOT\_STICKY : Service không được khởi động lại.
* Service.START\_REDELIVER\_INTENT : Service được khởi động lại sau khi bị crash và Intent sau khi xử lí sẽ được gửi đi.

Nhuư đã đề cập ở trên, một started service cần phải được dừng lại nếu không nói sẽ chạy vô thời hạn. Để thực hiện được điều này ta gọi stopSelf() từ chính Service hoặc goị stopService() từ phía client :



* 1. **Media player**

Bạn có thể dễ dàng tích hợp audio, video và hình ảnh trong ứng dụng với sự hỗ trợ của Android multimedia framework. Bạn có thể chạy audio và video từ các file media lưu trữ trong resource của ứng dụng, từ các file độc lập trong filesystem, hoặc từ 1 dữ liệu stream qua kết nối mạng, tất cả sử dụng MediaPlayer APIs.

Khai báo trong Manifest

Trước khi bắt tay vào code, chúng ta cần khai báo các features liên quan của media player trong manifest.

* Internet permission:



Wakelock permission: nếu ứng dụng của bạn cần giữ màn hình từ sleeping, hoặc sử dụng các hàm MediaPlayer.setScreenOnWhilePlaying(), MediaPlayer.setWakeMode(), bạn phải yêu cầu permission này:



**Sử dụng MediaPlayer**

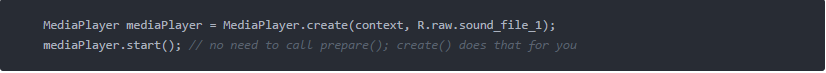
Một trong những thành phần quan trọng nhất của media framework là class MediaPlayer. Một đối tượng của class này có thể fetch, mã hoá, play cả audio và video chỉ với tối thiểu cài đặt. Nó hỗ trợ một vài media resource khác nhau, ví dụ:

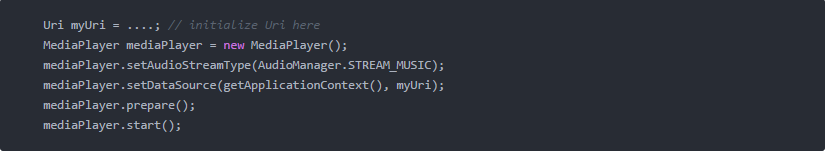
* Local resource

Internal URI, ví dụ bạn có thể nhận được từ một Content Resolver

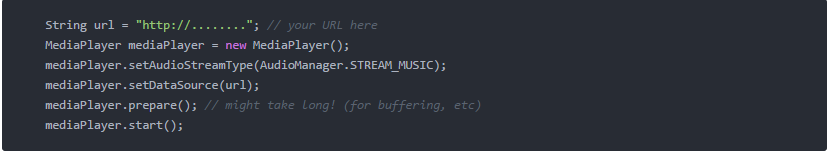
External URL (streaming)

Đây là một ví dụ của cách play audio có trong local raw resource ( lưu trong thư mục res/raw của ứng dụng)





Playing từ một remote URL qua HTTP stream sẽ giống như thế này:



Chú ý: bạn phải bắt hoặc pas cả 2 IllegalArgumentException và IOException khi sử dụng setDataSource(), vì file bạn đang tham khảo có thể không tồn tại.

**Sự chuẩn bị không đồng bộ**

Sử dụng MediaPlayer có thể đơn giản về nguyên tắc. Mặc dù nó rất quan trọng để nhớ một vài thứ cần thiết để tích hợp nó tương ứng với một ứng dụng Android điển hình. Ví dụ, để gọi tới prepare() có thể tốn một thời gian dài để thực hiện, vì nó có thể liên quan đến fetching và decoding dữ liệu media. Vì thế, giống như trường hợp với bất kỳ một phương thúc tốn nhiều thời gian để thực hiện, bạn không bao giờ nên gọi nó từ UI thread của ứng dụng. Làm thế sẽ là nguyên nhân của việc UI phải chờ cho tới khi phương thức thực hiện xong, một trải nghiệm rất tệ của người dùng và có thể là nguyên nhân của ANR ( Application Not Responding) error. Kể cả bạn mong muốn resource load nhanh, nhớ rằng bất cứ thứ gì tốn thời gian hơn 10 giây để phản hồi trong UI sẽ là nguyên nhân của một sự pause đáng chú ý và sẽ gây cho người dùng ấn tượng rằng ứng dụng của bạn chậm.

Để tránh sự trễ trong UI thread của bạn, tạo một thread khác để chuẩn bị cho MediaPlayer và chú ý main thread khi xong. Mặc dù, trong khi bạn có thể tự viết một threadinng logic, một mẫu chung rất thuận lợi khi sử dụng MediaPlayer với sự hỗ trợ của framework là phương thức prepareAsync(). Phương thức này bắt đầu để chuản bị media trong background và trả về ngay lập tức. Khi media chuẩn bị xong, hàm onPrepared() của MediaPlayer.OnPreparedListener, cài đặt qua setOnPreparedListener() sẽ được gọi.

**Quản lí trạng thái**

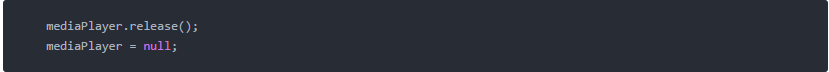
Một khía cạnh khác của MediaPlayer mà bạn cần nhớ là state-based. MediaPlayer có một internal state mà bạn luôn phải chú ý khi viết code, vì một số hoạt động chỉ có hiệu lực khi người chơi ở trong một trang thái đặc biệt. Nếu bạn thực hiện một tiến trình trong trạng thái sai, hệ thống có thể sẽ ném vào một exception hoặc gây ra một hành vi không đoán trước được.

Ví dụ, khi tạo một MediaPlayer, nó ở trạng thấy Idle, bạn nên khởi tạo nó với setDataSource() khi ở trạng thái khởi tạo. Sau đó, bạn phải chuẩn bị cho nó với cả 2 hàm prepare() và prepareAsync(). Khi MediaPlayer đã chuẩn bị xong, nó sẽ vào trạng thái Prepared, tức là bạn có thể gọi start() để play media. Lúc này, bạn có thể chuyển giữa các trạng thái Started, Paused, và PlaybackCompleted bởi các hàm start(), pause(), seekTo() ở trong chúng. Khi gọi stop(), chú ý rằng bạn không thể gọi start() lần nữa cho đến khi chuẩn bị xong MediaPlayer.

**Giải phóng MediaPlayer**

Một MediaPlayer có thể tiêu thụ tài nguyên quý giá của hệ thống, vì thế khi bạn thực hiện xong với nó, cần luôn nhớ gọi hàm release() để đảm bảo bất kỳ tài nguyên hệ thống nào đã được phân phát sẽ được giải phóng.

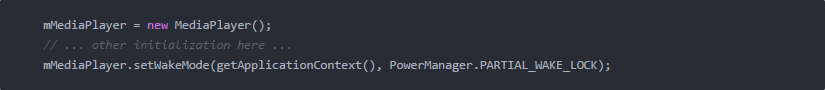
Đây là cách bạn nên giải phóng MediaPlayer:



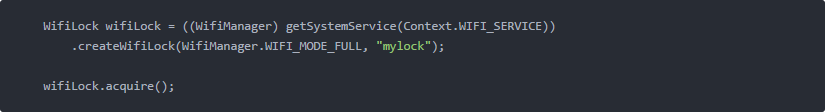
**Sử dụng service với MediPlayer**

Nếu bạn muốn media play trong background ngay cả khi ứng dụng không ở trên màn hình, là khi bạn muốn media tiếp tục chạy khi bạn sử dụng 1 ứng dụng khác, bạn cần bắt đầu 1 Service và điều khiển MediaPlayer từ đó.

**Sử dụng wakelock**  
Khi thiết kế một ứng dụng chạy nhạc trên background, thiết bị có thể sleep trong khi service vẫn đang chạy. Vì hệ thống Android cố gắng tiết kiệm pin khi thiết bị sleeping, hệ thống sẽ cố để tắt bất kì feature nào không cần thiết, kể cả CPU và phần cứng WiFi. Mặc dù, nếu bạn đang chạy hoặc stream nhạc, bạn muốn tránh hệ thống gây trở ngại cho playback của bạn.  
Để chắc chắn rằng service của bạn tiếp tục chạy dưới những điều kiện này, bạn phải sử dụng wakelock. Một wakelock là một cách để báo hiệu với hệ thống rằng ứng dụng của bạn đang sử dụng một số feature nên có sẵn kể cả nếu điện thoại của bạn ở chế độ nghỉ.  
Để chắc chắn rằng CPU tiếp tục chạy trong khi MediPlayer đang chạy, gọi hàm setWakeMode() khi khởi tạo MediaPlayer.



Nếu bạn đang stream media qua network và bạn đang dùng WiFi, có lẽ bạn muốn giữ WifiLock, bạn phải yêu cầu và giải phóng. Vì thế, khi bạn bắt đầu chuẩn bị MediaPlayer với chế độ URL, bạn nên tạo và mua một Wifilock. Ví dụ:



* 1. **Intent**

**Intent là gì ?**

**Intent** (ý định) là những tin nhắn không đồng bộ cho phép các thành phần ứng dụng yêu cầu chức năng từ các thành phần Android khác. Intents cho phép bạn tương tác với các thành phần từ các ứng dụng như nhau cũng như với các thành phần đóng gói trong một ứng dụng khác. Ví dụ, một Activity có thể gọi một Activity bên ngoài để chụp ảnh.   
  
**Intent** là đối tượng của class **android.content.Intent**. Mã của bạn có thể gửi **Intent** vào hệ thống Android với chỉ định thành phần mục tiêu gửi đến.   
  
Ví dụ thông qua phương thức ***startActivity()***, bạn có thể định nghĩa một **Intent** sử dụng để gọi chạy một **Activity** khác. Tại **Activity** mục tiêu, thông qua phương thức ***startActivity()*** bạn có thể xác định được ý định của người gửi đến để bắt đầu (start) **Activity** này.   
  
Một **Intent** có thể chứa dữ liệu thông qua một **Bundle**. Dữ liệu này có thể được sử dụng bởi các thành phần tiếp nhận

### Intent có thể sử dụng để:

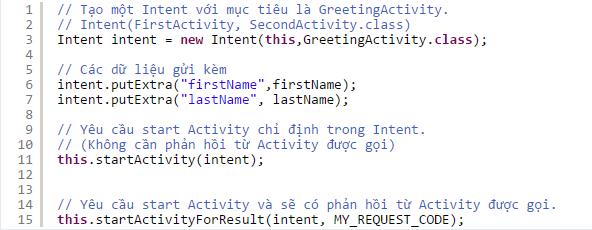
* + Bắt đầu một Activity
  + Bắt đầu một Activity con.
  + Bắt đầu một dịch vụ (Service).

**Các loại Intent**

**Android** hỗ trợ 2 loại **Intent** là **Intent** tường minh (explicit) và **Intent** không tường minh (implicit).   
  
Một ứng dụng có thể xác định thành phần mục tiêu một cách trực tiếp vào **Intent** (mục tiêu được yêu cầu là rõ ràng) hoặc yêu cầu hệ thống**Android** đánh giá các thành phần đã đăng ký trên dữ liệu đích để chọn ra một cái để gửi yêu cầu đến ( **Intent** không tường minh).

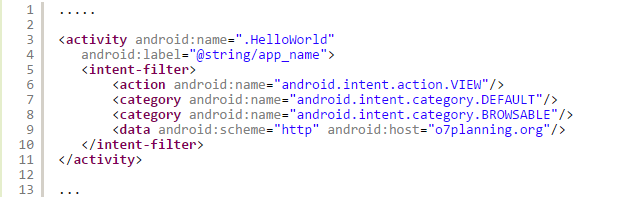
**Intent tường mình**

**Intent tường minh (Explicit intents):** Là những ý định (intent) chỉ định rõ ràng tên của các thành phần mục tiêu để xử lý; trong đó, trường mục tiêu (tùy chọn) được sét một giá trị cụ thể thông qua các phương thức ***setComponent()*** hoặc ***setClass()***.



**Bộ lọc Filter**

Khi bạn tạo ra một ý định ngầm (Implicit Intent), hệ thống **Android** sẽ tìm kiếm các thành phần thích hợp để start bằng cách so sánh nội dung của ý định (Intent) với các bộ lọc ý định khai báo trong file manifest của  ứng dụng khác trên thiết bị. Nếu mục đích (Intent) phù hợp với một bộ lọc ý định, hệ thống start thành phần đó và truyền vào đối tượng Intent.



**Intent không tường minh**

**Intent không tường minh (Implicit Intents):** Là những ý định (intent) không chỉ định rõ một mục tiêu thành phần, nhưng bao gồm đầy đủ thông tin cho hệ thống để xác định các thành phần có sẵn là tốt nhất để chạy cho mục đích đó. Hãy xem xét một ứng dụng liệt kê các nhà hàng có sẵn ở gần bạn. Khi bạn bấm vào một tùy chọn nhà hàng cụ thể, ứng dụng sẽ hỏi một ứng dụng khác để hiển thị các tuyến đường đến nhà hàng đó. Để đạt được điều này, nó hoặc có thể gửi một ý định rõ ràng trực tiếp đến các ứng dụng **Google Maps**, hoặc gửi ý định ngầm, ý định sẽ được chuyển giao cho bất kỳ ứng dụng nào cung cấp các tính năng bản đồ (map)  (chẳng hạn, **Yahoo Maps**).

* 1. **Broadcast receiver**

Android BroadcastReceiver là một thành phần nơi bạn có thể đăng ký sự kiện của hệ thống (system) hay ứng dụng (application). Bạn sẽ nhận được thông báo về các sự kiện một khi đã đăng ký. Việc phát tin (Broadcast) bắt nguồn từ hệ thống cũng như từ các ứng dụng (applications). Ví dụ về việc phát tin bắt nguồn từ hệ thống có thể kể ra như thông báo pin yếu . Với cấp độ ứng dụng có thể kể ra như khi bạn download một file nhạc. Ứng dụng nhạc này sẽ nhận được thông báo về việc download, sau khi kết thúc sẽ đưa bài hát này vào danh sách để bật. Để thực hiện được việc này, ứng dụng nhạc cần thiết phải đăng ký cho sự kiện. Một ví dụ đơn giản khác, người dùng sẽ nhận thông báo khi pin của máy mình đã xuống dưới một mức độ nào đấy

Có 2 cách để đăng ký Android broadcast receiver

* Một là phương thức tĩnh khi đăng ký broadcast receiver trong ứng dụng thông qua file AndroidManifest.xml.
* Hai là phương thức động để đăng ký broadcast receiver thông qua việc sử dụng method Context.registerReceiver(). Các broadcast receiver được đăng ký động có thể được gỡ bởi method Context.unregisterReceiver().

Chúng ta phải hết sức cẩn thận khi tạo ra các broadcast receiver động. Nếu quên việc gỡ đăng ký broadcast receiver, hệ thống sẽ thông báo lỗi lộ (leaked) broadcast receiver

**Tạo broadcast receiver và sử dụng method onReceive()**

Một broadcast receiver sẽ mở rộng (extends) class trừu tượng (abstract) BroadcastReceiver. Điều này có ý nghĩa bạn phải sử dụng method onReceive() của class chính (base) này. Mỗi khi có sự kiện xảy ra, Android sẽ gọi method onReceive()ở broadcast receiver đã được đăng ký. Ví dụ, nếu bạn đăng ký cho sự kiện ACTION\_POWER\_CONNECTED , mỗi khi di động được kết nối với nguồn điện, method onReceive() broadcast receiver của bạn sẽ được gọi

**Method onReceive()**

Method onReceive() có 2 đối số (argument)

onReceive(Context context, Intent intent)

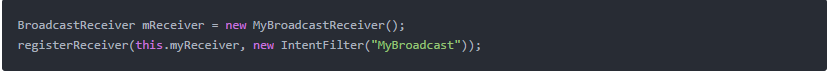
context (phạm vi) có thể được sử dụng để chạy service hoặc activity và intent là object (đối tượng) cùng với action (hành động) bạn đã sử dụng để đăng ký receiver của bạn. Nó có thể chứa các thông tin thêm mà bạn có thể dùng để implementation (thi hành).

**Đăng ký Broadcast Receiver tĩnh**

Giống như đã nói ở trên, broadcast receiver có thể đăng ký tĩnh thông qua file AndroidManifest.xml

Element (yếu tố) được sử dụng để định rõ sự kiện (event) mà receiver nên phản ứng lại. Chính vì thế mỗi lần sự kiện này (MyBroadcast ) xảy ra, method onReceive()của MyBroadcastReceiver sẽ được gọi.

**Đăng ký broadcast receiver động**



* 1. **File (write, read)**
  2. **Permission**

Mọi ứng dụng [Android](https://viblo.asia/nguyen.dinh.duc.0605/posts/MVpvKBLgkKd) đều chạy với các quyền truy cập nhất định. Nếu 1 ứng dụng cần sử dụng tài nguyên hay thông tin nào ngoài các quyền cơ bản của nó thì đều phải gửi yêu cầu các quyền thích hợp. Bạn phải khai báo ứng dụng của bạn cần những quyền nào bằng việc liệt kê các quyền đó trong file Manifest.

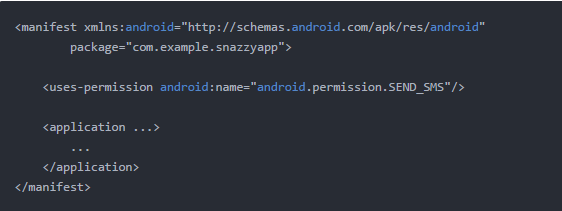
Tùy thuộc vào các quyền đó có truy cập vào các thông tin hay tài nguyên nhạy cảm (ví dụ thông tin cá nhân, danh bạ,...) hay không mà hệ thống sẽ tự động cấp quyền, hoặc phải chờ sự cho phép của người sử dụng thiết bị. Đồng thời tùy thuộc vào phiên bản của thiết bị người dùng sẽ cấp quyền khi họ cài đặt ứng dụng (từ [Android](https://viblo.asia/nguyen.dinh.duc.0605/posts/MVpvKBLgkKd) 5.1 trở xuống) hoặc khi đang chạy ứng dụng (từ [Android](https://viblo.asia/nguyen.dinh.duc.0605/posts/MVpvKBLgkKd) 6.0 trở lên).

**Xác định ứng dụng của bạn cần những quyền nào**

Khi bạn phát triển ứng dụng của mình, bạn nên chú ý khi nào ứng dụng của bạn cần được cấp quyền. Đặc biệt, 1 ứng dụng cần cấp quyền bất cứ khi nào nó sử dụng thông tin hay các tài nguyên mà bản thân ứng dụng đó không tạo ra, hoặc hành động của ứng dụng ảnh hưởng tới thiết bị hoặc các ứng dụng khác. Ví dụ, 1 ứng dụng cần truy cập vào internet, sử dụng camera, hay bật tắt wifi, ứng dụng đó cần yêu cầu các quyền phù hợp. Để xem đầy đủ danh sác các quyền của hệ thống, bạn truy cập vào [Danh sách các quyền thông thường và nguy hiểm](https://developer.android.com/training/permissions/best-practices.html#perms-vs-intents).

**Thêm các quyền vào file Manifest**

Để khai báo ứng dụng của bạn cần 1 quyền, đặt 1 thẻ <use - permission> trong file Manifest, như là phần tử con của thẻ gốc <manifest>. Ví dụ, 1 ứng dụng cần gửi tin nhắn SMS thì cần thêm như sau:



Phản hồi của hệ thống sau khi bạn khai báo 1 quyền dựa trên việc quyền đó có nhạy cảm hay không. Nếu như không ảnh hưởng tới sự riêng tư của người sử dụng, hệ thống sẽ cấp quyền tự động. Ngược lại nếu liên quan tới các thông tin nhạy cảm của người dụng, hệ thống sẽ yêu cầu người dùng đồng ý cấp quyền đó

**Permission Android 6.0**

Bắt đầu từ Androdi 6.0 (API 23), người dùng cần cấp quyền cho ứng dụng khi ứng dụng đang chạy, không phải khi cài đặt. Với cách tiếp cận này, người dùng không cần phải cấp quyền khi mà họ cài đặt hay cập nhật ứng dụng. Điều này cũng giúp cho người dùng kiểm soát tốt hơn các chức năng của ứng dụng. Ví dụ, 1 người dùng có thể cho phép ứng dụng truy cập vào camera nhưng không thể lấy vị trí của thiết bị. Người dùng có thể thay đổ các quyền bắt cứ lúc nào, bằng các vào mục cài đặt của thiết bị.

Các quyền hệ thống được chia làm 2 loại: bình thường và nguy hiểm:

* Các quyền bình thường là không ảnh hưởng đến thông tin cá nhân của người dùng. Hệ thống sẽ tự động cấp các quyền này mà không cần hỏi ý kiến người dùng
* Các quyền nguy hiểm là cho phép truy cập vào các thông tin cá nhân của người dùng. Hệ thống sẽ yêu cầu sự đồng ý của người dùng sau đó mới cấp quyền cho ứng dụng

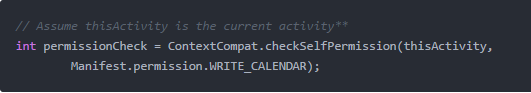
Dù ở phiên bản Android nào, ứng dụng của bạn đều phải khai báo các quyền bình thường và nguy hiểm trong file Manifest. Tuy nhiên, tùy từng phiên bản hệ thống sẽ xử lý khác nhau:

* Thiết bị chạy Android 5.1 hay thấp hơn, hoặc ứng dụng của bạn nhắm đến SDK 22 hay thấp hơn thì người dùng sẽ cho cấp các quyền khi họ cái đặt ứng dụng; nếu họ không cấp quyền, thì hệ thống sẽ không cài đặt ứng dụng
* Thiết bị chạy Android 6.0 hay cao hơn, và ứng dụng của bạn nhắm đến SDK 23 hay cao hơn thì nó sẽ yêu cầu người dùng cấp các quyên nguy hiểm khi ứng dụng đang chạy. Người dùng có thể đồng ý hoặc từ chối từng quyền, và ứng dụng có thể tiếp tục chạy với khả năng bị giới hạn thậm chí khi người dùng từ chối các yêu cầu cấp quyền

**Kiểm tra các quyền**

Nếu ứng dụng của bạn cần 1 quyền nguy hiểm, bạn phải kiểm tra liệu quyền đó đã được cấp chưa mỗi lần bạn thực hiện 1 thao tác mà yêu cầu quyền đó. Người dùng luôn luôn có thể gọi lại các quyền, thậm chí nếu ứng dụng đã sử dụng camera hôm qua, không thể đảm bảo rằng hôm nay nó vẫn có quyền đó.

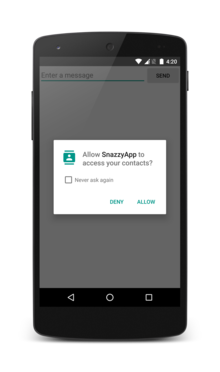
Để kiểm tra liệu bạn đang có 1 quyền hay không, gọi phương thức [ContextCompat.checkSelfPermission()](https://developer.android.com/training/permissions/requesting.html) . Ví dụ, đoạn code sau cho thấy làm thế nào để kiểm tra 1 activity có quyền viết vào lịch hay không:



Nếu ứng dụng có quyền này, phương thức sẽ trả về [PackageManager.PERMISSION\_GRANTED](https://developer.android.com/reference/android/content/pm/PackageManager.html#PERMISSION_GRANTED), và ứng dụng có thể thực hiện thao tác. Nếu ứng dụng không có quyền này, phương thức trả về [PERMISSION\_DENIED](https://developer.android.com/reference/android/content/pm/PackageManager.html#PERMISSION_DENIED), và ứng dụng phải hỏi lại người dùng để được cấp quyền.

**Yêu cầu để được cấp quyền**

Nếu ứng dụng cảu bạn cần 1 quyền nguy hiểm đã được liệt kê trong file Manifest, nó phải yêu cầu người dùng cấp quyền đó. Android cung cấp 1 số phương thức bạn có thể sử dụng để yêu cầu được cấp quyền. Gọi các phương thức này sẽ hiển thị 1 hộp hội thoại trên màn hình.



**Giải thích tại sao ứng dụng cần được cấp quyền**

Trong vài trường hợp, bạn có thể muốn giúp người dùng hiểu rõ hơn tại sao ứng dụng của bạn là cần 1 quyền nào đó. Ví dụ, nếu nguwoif dùng bật ứng dụng ảnh, người dùng có thể không bất ngờ khi ứng dụng yêu cầu cần quyền sử dụng camera, nhưng người dùng sẽ không hiểu tại sao ứng dụng lại cần truy cập vào danh bạ. Trước khi bạn yêu cầu 1 quyền, bạn nên cân nhắc việc giải thích lý do với người dùng. Nhưng luôn nhớ đừng làm người dùng khó chịu với quá nhiều giải thích.

1 cách tiếp cận mà bạn có hể sử dụng là chỉ cung cấp giải thích khi người dùng đã từng từ chối cấp quyền trước đây. Nếu người dùng tiếp tục muốn sử dụng chức năng này, chứng tỏ người dùng có thể không hiểu tại sao lại cần phải cấp quyền đó. Trong trường hợp đó, ta nên cung cấp lời giải thích.

Để giúp tìm các trường hợp khi mà người dùng cần 1 lời giải thích, Android cung câp 1 phương thức[shouldShowRequestPermissionRationale()](https://developer.android.com/reference/android/support/v4/app/ActivityCompat.html). Phương thức này trả về true nếu ứng dụng đã từng yêu cầu cấp quyền trước đây nhưng bị từ chối.

**Yêu cầu các quyền mà bạn cần**

Nếu ứng dụng của bạn không có quyền mà nó cần, ứng dụng cần phải gọi 1 trong các phương thức [requestPermissions()](https://developer.android.com/reference/android/support/v4/app/ActivityCompat.html) để yêu cầu các quyền phù hợp. Ứng dụng của bạn truyền các quyền mà nó muốn, cùng với 1 số request code để xác định yêu cầu này. Phương thức này không đồng bộ: nó trả về ngay lập tức, và sau đó người dùng thao tác với hộp hội thoại, hệ thống sẽ gọi các phương thức callback với kết quả, trả về cùng 1 request code mà ứng dụng đã truyền từ trước.

**Xử lý kết quả trả về của các yêu cầu cấp quyền**

Khi ứng dụng của bạn yêu cầu cấp quyền, hệ thống hiển thị 1 hộp hội thoại cho người dùng. Khi người dùng thao tác xong với hộp hội thoại đó, hệ thông sẽ gọi phương thức [onRequestPermissionsResult()](https://developer.android.com/reference/android/support/v4/app/ActivityCompat.OnRequestPermissionsResultCallback.html), truyền cho nó kết quả mà nguời dùng đã chọn. Ứng dụng phải viết lại phương thức để tìm ra liệu quyền đó có được cấp hay không. Callback được truyền cùng 1 request code mà bạn đã truyền từ trước đó.

Hộp hội thoại được hiển thị bởi hệ thống mô tả tập các quyền ứng dụng cần truy cập, nó không liệt kê từng quyền chi tiết. Người dùng chỉ cần cấp quyền 1 lần cho từng tập quyền. Nếu ứng dụng yêu cầu bât kỳ các quyền nào khác trong tập quyền đó, hệ thống sẽ từ cấp các quyền đó.

Nếu người dùng từ chối 1 yêu cầu cấp quyền, ứng dụng nên chọn hành động thích hợp. Ví dụ, ứng dụng có thể hiển thị 1 hộp hội thoại giải thích tại sao nó không thể thực hiện thao tác người dùng cần.

Khi hệ thống yêu cầu người dùng cấp quyền, người dùng có thể chọn không hiển thị yêu cầu này bao giờ nữa. Trong trường hợp đó, bất cứ khi nào ứng dụng sử dụng [requestPermissions()](https://viblo.asia/nguyen.dinh.duc.0605/posts/roavrJWpvRM) để yêu cầu quyền đó lần nữa, hệ thống tự động từ chối. Hệ thống gọi phương thức callback [onRequestPermissionsResult()](https://viblo.asia/nguyen.dinh.duc.0605/posts/roavrJWpvRM) và truyền [PERMISSION\_DENIED](https://viblo.asia/nguyen.dinh.duc.0605/posts/roavrJWpvRM), tương tự như người dùng chọn không cấp quyền lần nữa.

* 1. **Asyntask**

Trong ứng dụng Android có nhiều tác vụ (task) chạy song song với nhau để đảm bảo ứng dụng của chúng ta tận dụng được hết khả năng của CPU và mang  lại performence cho ứng dụng. Vậy để lập trình được nhiều tác vụ chạy song song trong ứng dụng Android như thế nào thì bài viết này sẽ cùng các bạn tìm hiểu chi tiết.

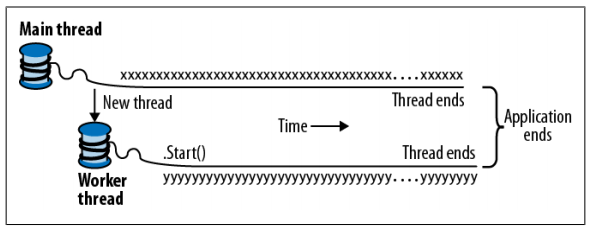
**MultiThreading là gì**

MultiThreading là khái niệm dùng để nhắc đến những tác vụ chạy song song với nhau và giúp quá trình tương tác của người dùng với ứng dụng không bị gián đoạn. Khi một ứng dụng được start lên sẽ có một Thread được gọi là Main Thread hay UI Thread được chạy. Từ Thread này chúng ta có thể tạo ra những Thread khác để làm những tác vụ chiếm nhiều thời gian. Nếu chúng ta cố tình xử lý những tác vụ này trên UI Thread mà không tạo ra Thread khác có thể dẫn tới làm đơ UI và crash ứng dụng.

Những trường hợp nên sử dụng MultiThreading.

* Thao tác với hệ thống mạng (network).
* Đọc, ghi file.

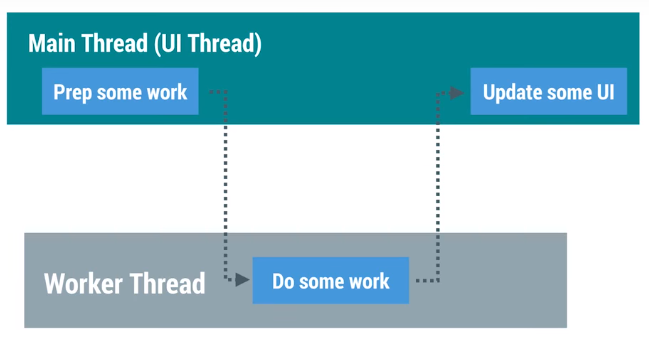
Hình ảnh mô phòng MultiThreading.



**MultiThreading trong Android**

Trong Android, khi chúng ta tạo một Thread để xử lý một tác vụ nào đó trong phương thức run() và gọi phương thức start() để chạy tác vụ này. Thì trong phương thức run chúng ta không được update các giá trị trực tiếp lên UI mà phải gửi message về UI Thread để UI Thread update các giá trị đó lên UI. Nếu bạn cố tình thì sẽ application sẽ ném ra một ngoại lệ.

Mô hình làm việc MultiThreading trong Android.



Để hổ trợ người dùng lập trình multithreading dễ dàng và nhanh gọn. Google giới thiệu cho chúng ta một class dùng để đáp ứng công việc này đó là **AsyncTask**.

**Sử dụng Asynctask**

Class AsyncTask có các phương thức chính như sau:

**onPreExecute**

Là phương thức được chạy đầu tiên khi task được bắt đầu. Phương thức này làm việc trên UI Thread.

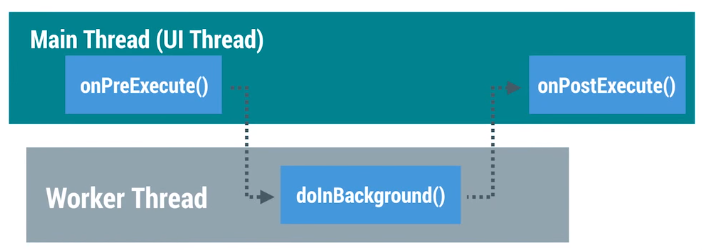
**doInBackground**

Là phương thức chạy background thực hiện tác vụ nào đó. Phương thức này chạy trên Background Thread vì thế chúng ta không được update giá trị lên UI trong phương thức này.

**onPostExecute**

Là phương thức được gọi ngay sau khi phương thức doInBackground kết thúc. Đối số của phương thức này chính là kết quả return từ phương thức doInBackground. Phương thức này làm việc trên UI Thread.

Mô hình làm việc.



Ngoài 3 phương thức chính trên còn có 2 phương thức nữa đó là:

**publishProgress**

Được sử dụng trong phương thức doInBackground để cập nhật quá trình làm việc lúc runtime và gọi phương thức onProgressUpdate trên UI Thread để update giá trị trong khi doInBackground vẫn làm việc.

**onProgressUpdate**

Update giá trị lên UI, được gọi nếu phương thức **publishProgress** được gọi trong phương thức **doInBackground**. Phương thức này làm việc trên UI Thread.

Ba đối số mà khi chúng ta tạo class extends từ AsyncTask là:

AsyncTask<Params, Progress, Result>

**Param**

Là kiểu của đối số được truyền vào phương thức doInBackground.

**Progress**

Là kiểu biến dùng để update giá trị trong quá tình **doInBackground** chạy. Giá trị này được truyền vào phương thức **onProgressUpdate.**

**Result**

Là kiểu trả về của phương thức **doInBackground** và là kiểu của đối số của phương thức **onPostExcute**.

* 1. **Sqlite**
  2. **Animation**

**CHƯƠNG 2 : PHÂN TÍCH**

1. **Mô tả yêu cầu người dùng**

* Người dùng cần gì ở một ứng dụng Nghe nhạc
* Giao diện đẹp mắt, thân thiện mà dễ sử dụng.
* Đa dạng các thể loại nhạc.
* Ứng dụng nhẹ mà lại nhanh và mượt.
* Nhiều hình ảnh của ca sĩ, bài hát..
* Hiển thị lời bài hát theo từng câu hát.
* Đồng bộ, lưu trữ nhạc trên cloud hoặc driver.
* Cập nhật những bài hát hit, những album mới hàng ngày.
* Equalizer : bộ chỉnh âm sắc cho âm thanh.
* Và nhiều tiện ích khác nữa.

1. **Phân tích cơ sở dữ liệu**

Cơ sở dữ liệu ở đây là csdl offline để lưu trữ những bài hát khi tải online về và lưu giữ những bài hát có trong bộ nhớ máy

Csdl sẽ import các bài hát trong bộ nhớ máy vào và lưu trữ nó, và ứng dụng sẽ đọc csdl này chứ không đọc csdl của hệ điều hành Android.

CSDL này có cấu trúc gần tương tự như CSDL của hệ điều hành Android đang sử dụng, tuy nhiên do mục đích sử dụng cho riêng app Nghe nhạc nên nó đã được tối ưu thêm một số cột cần thiết cũng như loại bỏ một số cột mà không sử dụng đến.

1. **Giải thích các thành phần**
2. **Giải thích các mối quan hệ của các bảng trong csdl**
3. Bảng Media :

* MID : khóa chính của bảng Media, tự động tăng.
* \_ID : Cột ID, dữ liệu được lấy từ csdl của HĐH Android.
* \_DATA : Đường dẫn của file bài hát trong bộ nhớ máy, được lấy từ csdl của hệ điều hành Android.
* \_DISPLAY\_NAME : Tên hiển thị của bài hát (bao gồm cả đuôi).
* \_SIZE : Dung lượng tính bằng KB của bài hát.
* \_MIME\_TYPE : kiểu của file.
* \_DATE\_ADDED : ngày file được thêm vào trong bộ nhớ máy.
* \_ TITLE : Tên bài hát (không có đuôi).
* \_DURATION : Độ dài bài hát (milisecond).
* \_ARTIST\_ID : Id của Artist tham chiếu đến bảng ARTIST.
* \_ARTIST : Tên hiển thị của Artist.
* \_ALBUM\_ID : Id của Album tham chiếu đến bảng ALBUM.

1. Bảng Artist

* **MID** : Id của từng row trong bảng, tự động tăng.
* **\_ID :** Id của Artist, được lấy từ CSDL của hệ điều hành.
* **\_ARTIST :** Tên hiển thị của Artist.
* \_NUMBER\_OF\_ALBUM : số Album của Artist.
* \_NUMBER\_OF\_TRACK : số Track của Artist.

1. Bảng Album

* **MID** : Id của từng row trong bảng, tự động tăng.
* \_ID : Id của Album, được lấy từ CSDL của hệ điều hành.
* \_ALBUM : Tên hiển thị của Album.
* \_ARTIST : Tên hiển thị Artist thực hiện Album.
* \_ARTIST\_ID : Id của Artist thể hiện Album, tham chiếu đến bảng Artist.
* \_NUMBER\_OF\_SONG : số bài hát trong Album.
* \_ALBUM\_ART : Ảnh của Album.

1. Bảng Playlist

* MID : Id của từng row trong bảng, tự động tăng.
* \_ID : Id của Playlist.
* \_DATA: Đường dẫn đến thư mục chứa các bài hát trong Playlist.
* \_NAME : Tên hiển thị Playlist.
* \_DATA\_ADDED : Ngày tạo Playlist.

1. Bảng PlaylistMember

* MID : Id của từng row trong bảng, tự động tăng.
* \_ID : Id của PlaylistMember.
* \_AUDIO\_ID : Id của bài hát, tham chiếu đến bảng Media.
* \_PLAYLIST\_ID : Id của Playlist, tham chiếu đến bảng Playlist.
* \_ARTIST\_ID : Id của Artist, tham chiếu đến bảng Artist.
* \_ARTIST : Tên Artist.
* \_ALBUM\_ID : Id của Album, tham chiếu đến bảng Album.
* \_ALBUM : Tên của album.
* \_DATA : Đường dẫn đến Playlist.
* \_DISPLAY\_NAME : Tên hiển thị của bài hát.
* \_SIZE : Dung lượng tính bằng KB của bài hát.
* \_MIME\_TYPE : kiểu của file.
* \_DATE\_ADDED : ngày file được thêm vào trong bộ nhớ máy.
* \_ TITLE : Tên bài hát (không có đuôi).

1. **Phân tích chức năng**

Quản lý Data trong máy

Download (option)

Quản lý PlayLists

Tìm kiếm

Chia sẻ

Chơi nhạc

***Ứng dụng nghe nhạc TPlayer***

Quản lý nhạc Online

Quản lý nhạc Offline

Quản lý Data ZingMp3

Lưu cache

Tìm kiếm

Chia sẻ

Download (bài hát)

Chơi nhạc

***Biểu đồ phân cấp chức năng của ứng dụng Tplayer***

Đáp ứng lại

Yêu cầu cập nhật thông tin

Quản trị

Người dùng

Yêu cầu chức năng

Hiển thị yêu cầu chức năng

Control bài hát

Yêu cầu cấp quyền

Đáp ứng lại yêu cầu

Đáp ứng yêu cầu

Y/c tìm kiếm

K/Q tìm kiếm

***Biểu đồ luồng dữ liệu mức khung cảnh***

K/Q download

Zing Mp3

Người dùng

Y/c xem nhạc offline

Kết quả

Y/c Playlist

Đáp ứng PlayLists

Y/c Tìm kiếm

K/Q tìm kiếm

Download (option)

K/Q download

Yêu cầu xem nhạc online (song, album,thể loại …)

Kết quả

Y/c tìm kiếm

Thông tin tìm kiếm

Download

Yêu cầu tự hủy khi thiếu tài nguyên

Quản trị

Cấp quyền online

Xác nhận đã cấp quyền

Cấp quyền online, offline

Xác nhận cấp quyền

Yêu cầu tự hủy khi thiếu tài nguyên

Kho chứa Offline

***Biểu đồ luồng dữ liệu mức đỉnh***

K/Q hủy

Read cache

Quản trị

Write cache

Đáp ứng

Permission

Yêu cầu hủy

Người dùng

Hiển thị music type

Control player

Đáp ứng Control

Donwload

Đáp ứngDownload

Chia sẻ

Phản hồi chia sẻ

Lựa chọn music type

Hiển thị music tương ứng

Zing mp3

***Biểu đồ luồng dữ liệu mức dưới đỉnh chức năng Online***

K/Q hủy

Read cache

Quản trị

Write cache

Đáp ứng

Permission

Yêu cầu hủy

Người dùng

Hiển thị music type

Control player

Đáp ứng Control

Donwload

Đáp ứngDownload

Chia sẻ

Phản hồi chia sẻ

Lựa chọn music type

Hiển thị music tương ứng

Yêu cầu music type

IUD playlist

Đáp ứng UID

***Biểu đồ luồng dữ liệu mức dưới đỉnh chức năng Offline***

1. **Nghe nhạc Online**

* Chức năng nghe nhạc Online lấy dữ liệu chính từ trang nghe nhạc trực tuyến <http://mp3.zing.vn>
* Khi vào màn hình Nghe nhạc Online, người dùng sẽ thấy có 4 Tab phía trên Top của màn hình
* Tab “TopMusic” : tab này chịu trách nhiệm hiển thị cho người dùng các thể loại nhạc đang đứng Top40 theo như mp3.zing.vn thông kê. Ở đây có rất nhiều các thể loại nhạc, nhưng tôi đã lược bỏ bớt một số thể loại nhạc kén người nghe, như “Nhạc không lời, Nhạc phim TQ. ..”. Khi click chọn xem từng thể loại nhạc, thì màn hình tiếp theo sẽ show ra một list các danh sách các bài hát thuộc thể loại đó và sắp xếp theo thứ tự từ cao xuống thấp. Từ đây người dùng có thể chọn từng bài để chơi nhạc, hoặc cũng có thể nhấn nút Play All để chơi tất cả các bài hát trên list đó. Ngoài ra còn nhiều tùy chọn trong list các bài hát đó, người dùng có thể trải nghiệm sau khi vào app sau.
* Tab “HotMusic” : Tab HotMusic chịu trách nhiệm hiển thị cho người dung.
* Các Album mới nhất vừa phát hành.
* Các bài hát mới nhất vừa được phát hành.
* Các bài hát đang Hot trên thị trường âm nhạc.

Do trong tab này hiển thị khác nhiều thông tin của các loại nhạc khác nhau, nên để không gây nhàm chán, tôi đã cho hiển thị các list danh sách loại nhạc theo nhiều cách khác nhau, tạo nên trải nghiệm người dùng tốt hơn.

Tùy theo mỗi loại nhạc sẽ có animation kèm theo tương ứng giúp người dùng trải nghiệm ứng dụng tốt hơn và muốn sử dụng nhiều hơn

Với Album thì người dùng k thể PlayAll album được, vì trong album có chứa nhiều bài hát rồi vì vậy tôi k sử dụng nút PlayAll cho list Album

Với list bài hát mới nhất và đang hot thì sẽ có nút PlayAll cho phép người dùng chơi tất cả các bài hát đó.

* Tab “ChartsMusic” : Tab này hiển thị Bảng xếp hạng âm nhạc của các Quốc gia. Ở đây trong khuôn khổ ứng dụng, tôi cho phép người dùng cập nhất 2 bảng xếp hạng âm nhạc đó là Bảng xếp hạng Âm nhạc Việt nam và Bảng xếp hạng Âm nhạc Âu-mỹ. Để cập nhật được 2 bảng xếp hạng này, tôi có thiết kế 2 Tab bên phía dưới màn hình, 1 tab hiển thị BXH Vn, tab còn lại sẽ hiển thị BXH Âu-mỹ. Và nút PlayAll cũng xuất hiện để người dùng có thể chơi tất cả các bài hát trên bảng xếp hạng.
* Tab “Album” : Giống như Tab TopMusic thì ở Tab Album cũng có rất nhiều thể loại cho từng Album như : Album nhạc rap, Album nhạc trữ tình, Album nhạc trẻ, K-pop, EDM ……Khi click vào từng thể loại Album thì ứng dụng sẽ tiếp tục hiển thị cho người dùng các Album theo từng thể loại đó. Những album này cực kì đa dạng. Và dữ liệu sẽ được phân trang, sẽ load trang tiếp theo khi người dùng cuộn xuống dưới dùng, điều này giúp tiết kiệm tối đa dung lượng mạng của người dùng.

Do đắc điểm của ứng dụng online nên đa số các ứng dụng kết nối tới môi trường mạng đều sẽ lưu Cache Data để tối ưu hóa trải nghiệm người dùng.

Do hệ điều hành Android chưa cho phép lưu List Object vào file Cache nên tôi đã sử dụng cách : chuyển List Object thành định dạng Json và rồi lưu cache chuỗi Json này xuống File. Sau khi lưu cache xong, khi muốn lấy List Object thì tôi sẽ đọc File cache lấy chuỗi Json và chuyển lại thành List Object để sử dụng. Để thực hiện việc này tôi có sử dụng thư viện Gson để thuận tiện trong việc chuyển đổi List Object thành chuỗi Json cũng như ngược lại.

1. **Nghe nhạc Offline**

* Nhạc Offline sẽ xử lý nhiều vấn đề hơn là Nhạc Online do phải làm việc với local rất nhiều.
* Do ứng dụng đọc dữ liệu từ Database riêng thiết kế riêng cho ứng dụng, nên khi cài đặt app, mặc định ứng dụng sẽ không hiển thị gì cả. Để có được dữ liệu, người dùng phải import data từ hệ điều hành vào, thao tác này gọi là Scan Data. Hầu hết các ứng dụng nghe nhạc hiện này đều làm theo phương pháp này, với cách này ứng dụng của ta sẽ chạy ổn định hơn, nhanh hơn, và tối ưu hơn.
* Để trỏ được tới Database của hệ điều hành ứng dụng phải được cấp quyền READ\_EXTERNAL\_STORAGE. Quyền này cho phép ứng dụng của bạn được quyền truy cập vào bộ nhớ ngoài của thiết bị. Để cấp quyền thì ứng dụng phải khai báo quyền này trong file AndroidManifest.xml. Phần này tôi xin được trình bày thêm sau.
* Sau khi trỏ được tới Database của hệ điều hành, ta tiếp tục xin cấp quyền WRITE\_EXTERNAL\_STORAGE, quyền này cho phép chúng ta ghi dữ liệu lên bộ nhớ ngoài. Bắt đầu từ bảng Media. Ở đây, ta sẽ lọc từng row dữ liệu của bảng, lấy những thông tin cần thiết rồi Insert vào Database của ứng dụng.
* Các bảng tiếp theo làm tương tự
* Vấn đề Download nhạc :
* Download một bài mới : data của bài hát này là data mới hoàn toàn, nó sẽ được hệ điều hành lưu các thông tin về nó, sau khi tải về, ứng dụng sẽ tự động query đến bài hát đó để lấy dữ liệu chèn vào database của ứng dụng.
* Download option : download này sẽ download lyric và hình ảnh cho bài hát. Giả sử như trong máy có một bài hát là “abcd.mp3” và bài hát này chưa co lyric cũng như hình ảnh, khi người dùng yêu cầu ứng dụng tải lyric và image thì ứng dụng sẽ mò lên Zing.Mp3 đề tìm lyric và image, nếu tìm được ứng dụng sẽ tải lyric và image và lưu trữ vào trong database.
* Khi download ứng dụng sẽ mở một IntentService để download, intentService sẽ chạy trên một Thread khác với UI Thread điều này tránh Block Ui của người dung.
* Về thiết kế giao UX của Offline, thì các ứng dụng cũng sẽ phân chia database ra và hiển thị theo các mục như : “Bài hát” , “Album” , “Artist”, “Favorite”, “Playlist”.
* Tạo Playlist mới : Khi người dùng khởi tạo một Playlist mới, điều đó tương ứng là tạo một row trong bảng Playlist đồng thời cũng add bài hát mà người dùng chọn vào trong bảng PlaylistMember với \_ID\_PLAYLIST trỏ thẳng vào vị trí của Playlist vừa tạo trước đó.

1. **Android Interface Define Langugae (AIDL)**
   1. **Service**

Một Service là một thành phần mà chạy trong Background để thực hiện các hoạt động mà không cần tương tác với người dùng và nó làm việc ngay cả khi nếu ứng dụng bị hủy.

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.

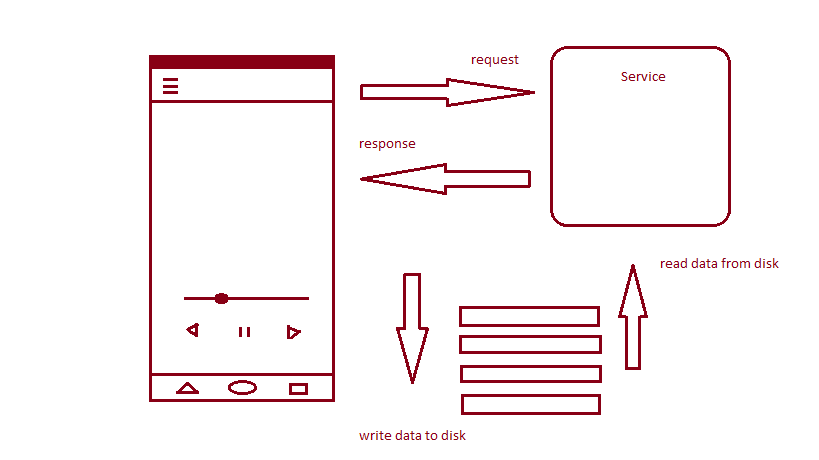
Để tạo một **service** bạn có thể tạo một **class** kế thừa **class Service**. **Class Service** có nhiều phương thức **callback**, bạn không cần phải **implement** hết các phương thức này, tuy nhiên việc hiểu rõ và áp dụng chúng trong code là rất quan trọng trong việc đáp ứng yêu cầu của người dùng.

Service có thể bị Hệ điều hành shutdown khi mà hệ điều hành cần tài nguyên cho việc khác. Giả sử service bị kill khi mà hệ thống đang thiếu Ram trầm trọng, sau khi bộ nhớ ram được giải phóng, service có thể tự động khởi chạy lại nếu ta đặt một cờ (Flag) thích hợp vào trong phương thức onStartCommand (). Nếu muốn service sống sót trong trường hợp tài nguyên hệ thống còn ít, ta cũng có thể set độ ưu tiên cao hơn bình thường cho service bằng thuộc tính setPiority trong AndroidManifets.xml. Khi set như vậy, ứng dụng sẽ thông báo cho hệ điều hành biết là Service này có độ ưu tiên cao cần được duy trì, hãy shutdown các thành phần khác trước đi, nếu shutdown hết rồi mà vẫn thiếu bộ nhớ thì lúc này Service mới bị shutdown

* Chú ý : Để Service được ứng dụng sử dụng, phải khai báo tên của Service trong file AndroidManifest.xml

Về cơ bản, một Service nhận hai trạng thái:

* Started (ii) : Dịch vụ có dạng "được bắt đầu" khi một thành phần ứng dụng (chẳng hạn như một hoạt động) bắt đầu nó bằng cách gọi [startService()](https://developer.android.com/reference/android/content/Context.html#startService(android.content.Intent)). Sau khi được bắt đầu, dịch vụ có thể chạy ngầm vô thời hạn, ngay cả khi thành phần bắt đầu nó bị hủy. Thông thường, dịch vụ được bắt đầu sẽ thực hiện một thao tác đơn lẻ và không trả về kết quả cho hàm gọi. Ví dụ, nó có thể tải xuống hoặc tải lên một tệp thông qua mạng. Khi thao tác được hoàn thành, dịch vụ tự nó sẽ dừng lại.
* Bound (iii) : Dịch vụ có dạng "gắn kết" khi một thành phần ứng dụng gắn kết với nó bằng cách gọi [bindService()](https://developer.android.com/reference/android/content/Context.html#bindService(android.content.Intent, android.content.ServiceConnection, int)). Dịch vụ gắn kết sẽ đưa ra một giao diện máy khách-máy chủ cho phép các thành phần tương tác với dịch vụ, gửi yêu cầu, nhận kết quả, và thậm chí làm vậy thông qua truyền thông liên tiến trình (IPC). Dịch vụ gắn kết chỉ chạy trong khi một thành phần ứng dụng khác được gắn kết với nó. Nhiều thành phần có thể gắn kết cùng lúc với dịch vụ, nhưng khi tất cả bị bỏ gắn kết thì dịch vụ sẽ bị hủy.
  1. **IntentService**
* IntentService là một class mở rộng (extend) từ Service.
* IntentService sẽ tạo ra một Thread riêng chạy độc lập với UiThread để tránh block Ui của người dùng. Vì vậy nó thường được dùng để làm các công việc nặng nhọc mất thời gian như là download .
* IntenterService nó tạo ra một hàng đợi (queue) để nhét các công việc vào đó rồi thực hiện tuần tự, vì vậy chúng ta cũng không cần quan tâm đến vấn đề đa luồng.
* IntentService tự biết khi nào nó cần kết liễu chính mình vì vậy chúng ta cũng k cần quan tâm đến nó làm chậm hoặc chiếm nhiều tài nguyên hệ thống. Khi nó làm xong công việc được giao nó sẽ tự kết liễu chính nó.
  1. **RemoteService**
* RemoteService chính là trạng thái khi một Component từ App này bound tới một Service ở một App khác để control Service đó
* RemoteService cho phép thành phần Bind (gắn kết) đến nó có thể giao tiếp và điều khiển nó. Giả sử khi đang download người dùng có thể bind tới ServiceDownload và tạm dừng công việc download lại.
* Tại sao lại dùng RemoteService trong ứng dụng nghe nhạc này ? Service sẽ là thành phần nắm giữ MediaPlayer để nó phát nhạc, vì Nhạc cần được play ở dưới nền để không bị ảnh hưởng khi người dùng điều hướng các màn hình hoặc điều hướng sang ứng dụng khác. Nếu sử dụng một Service mà nằm trong cùng 1 ứng dụng thì khi đóng hẳn ứng dụng (bật taskmanager, vuốt app lên trên hoặc sang ngang) thì Service sẽ bị đóng theo mặc dù sau đó nó có thể khởi chạy lại nhưng hãy tưởng tượng rằng : Bài hát đang hát ở 1’45s và service bị đóng lại, thì lúc này bài hát sẽ bị ngắt quãng ở 1’45s, sau đó service khởi chạy lại nó sẽ phát lại bài hát đó từ giây đầu tiên. Điều này gây khó chịu cho người dùng. Vì lý do đó, ta tạo một ứng dụng khác chứa thằng Service và từ app chính ta Bind (gắn) tới thằng Service đó để điều khiển theo ý muốn của ta, khi app chính bị đóng thì cũng k ảnh hưởng gì tới thằng Service kia.
  1. **AIDL**
* AIDL viết tắt của Android Interface Define Language, dịch ra là “Ngôn ngữ giao diện tự định nghĩa”
* Vì App chính và App chứa Service chạy độc lập với nhau, nên ta cần một cái gì đấy để dùng làm công cụ giao tiếp với 2 thằng đó. Và công cụ đó chính là AIDL. Để giao tiếp giữa App và Service chạy độc lập, tôi phải tự định nghĩa ra những interface, tự định nghĩa ra các func, các chức năng, tham số đầu vào, và cái tôi định nghĩa ra cũng được gọi à language, language này chỉ có App chính và Service có thể đọc được, và chúng sẽ dùng language này để giao tiếp với nhau.
* Hạn chế của của AIDL là không thể truyền thẳng một Object hay một List Object qua nhau được mà phải viết cái dữ liệu đó xuống disk, sau khi nhận được tín hiệu bên service sẽ đọc dữ liệu đã viết đó để sử dụng.

****

1. **Player**

* Để control Player do thằng Service đang cầm, tôi tạo một giao diện với đầy đủ các chức năng như : Pause, Play, Forward, Backward, Repeat , Volume, Seekbar với xu hướng nghe nhạc hiện đại tôi đã bỏ control Shuffle (trộn) ra khỏi giao diện
* Không chỉ có những control mà ứng dụng còn có thể hiển thị ảnh của Artist và ảnh của bài hát lên giao diện, gây thích thú và đổi mới hình ảnh liên tục cho người dùng, để người dùng k cảm thấy nhàm chán.
* Từ trải nghiệm của bản thân cũng như khảo sát tôi quyết định tạo thêm một View nữa để hiển thị lời bài hát, sau đó viết Thuật toán để Ứng dụng tính tính chính xác là bài hát đang hát đến câu chữ nào từ đó hiển thị lên cho người dùng. Lyric này cực kì hữu ích và sẽ giữ người dùng ở lại màn hình Player lâu hơn, nghĩa là người dùng sẽ sử dụng ứng dụng của tôi nhiều hơn.

1. **Equalizer**

* Chán nản với cái loa ngoài và headphone quá "cùi" hoặc nghe không "đã"?
* Buộc phải xóa bỏ một chương trình nghe nhạc đẹp và hiện đại đúng "gu" của mình vì không có chức năng chỉnh [equalizer](https://tinhte.vn/tags/equalizer/)?

Trên đây là 2 lý do phổ biến cho thấy tầm quan trọng của Equalizer.

Equalizer là bộ chỉnh âm sắc, nó đóng vai trò như một chiếc Ampli cho phép âm thanh phát qua nó có thể trầm, bổng bằng cách thay đổi tần số âm thanh của đầu ra Media.

Hiện nay trên thị trường các ứng dụng nghe nhạc nổi tiếng đã và đang áp dụng bộ Equalizer này cho ứng dụng của họ, nhằm nâng cao chất lượng trải nghiệm âm nhạc của người dùng. Và ở đây tôi cũng sẽ áp dụng bộ EQ này vào trong ứng dụng nghe nhạc của tôi.

1. **Android permission 6**

Khi một ứng dụng nào đó cần truy cập đến một tính năng đặc biệt trên thiết bị Android của bạn, nó đang hỏi hệ điều hành có cho phép xài hay không, và quyền truy cập đó chính là Permission. Ví dụ, khi app camera của bạn muốn đính kèm dữ liệu GPS vào ảnh, nó phải hỏi Android xem nó có được vào GPS để định vị hay không. Tương tự, khi một app tin nhắn muốn gửi SMS, nó phải liên hệ với Android để xem có được quyền làm chuyện này không.

Mục đích của Permission là để đảm bảo sự an toàn và [quyền riêng tư](https://tinhte.vn/tags/quyen-rieng-tu/) cho người dùng. Điện thoại của chúng ta giờ là một "kho tàng" dữ liệu cá nhân trong đó nên việc bảo vệ như thế này là cần thiết. Trong điện thoại có đủ hết mọi loại dữ liệu nhạy cảm, từ tin nhắn, hình ảnh, video cho đến dữ liệu duyệt web hay tài khoản ngân hàng. Thậm chí bộ định vị GPS, dữ liệu di động camera hay micro của smartphone cũng là thứ nhạy cảm vì nó có thể ghi lại hình ảnh của bạn hay những gì bạn nói, cho người khác biết bạn đang ở đâu, bạn đang nói chuyện với ai, nội dung bạn nói là gì...

Bạn có muốn một app theo dõi nào đó lén định vị bạn bằng GPS hay không? Bạn có muốn một app hack tiền nào đó lén gửi SMS đến tổng đài để trừ tiền của bạn trong khi bạn không biết gì? Bạn có muốn một ứng dụng nào đó tưởng chừng vô hại nhưng lại ghi âm các cuộc nói chuyện giữa bạn với đối tác hay không? Permission xuất hiện để ngăn ngừa những tình trạng như vậy, hay nói rộng hơn, là để ngăn cản các lập trình viên xấu khai thác, trục lợi và lừa đảo người dùng khi nói không đúng về chức năng app do họ viết ra.

* Với những thiết bị chạy Android 5.0 trở xuống, app sẽ hỏi trực tiếp ngay khi bạn chuẩn bị nhấn nút download trên Play Store, kể cả bản nền web hay bản Store có trong Android. Danh sách các permission mà app cần sẽ được liệt kê ra chi tiết kèm theo mô tả ngắn gọn cho bạn biết permission đó có tác dụng gì.
* Với những thiết bị chạy Android 6.0 trở lên, app sẽ chỉ hỏi permission của bạn khi nào nó cần. Đây là cơ chế mới của Android 6.0 nhằm hạn chế việc người dùng lỡ cài một app nào đó mà không để ý tới permission mà nó đang hỏi để rồi bị lợi dụng. Một điểm mới nữa của Android 6.0 đó là bạn có thể tự mình tắt mở từng permission của từng ứng dụng một, không phải đồng ý hoặc không đồng ý hàng loạt như hồi Android 5.0 trở về trước.

Google chia tất cả permission mà Android cấp cho app thành 2 loại: Permission thường và Permission nguy hiểm.

* **Permission thường**: là những permission có rất ít nguy cơ ảnh hưởng đến sự riêng tư của người dùng hay ảnh hưởng tới các ứng dụng khác. Ví dụ, permission để cài đặt múi giờ là permission thường. Nếu một app khai báo rằng nó cần permission thường, Android sẽ tự động cấp phép cho app mà không cần bạn cho phép, cũng như không thông báo trên Play Store lúc cài đặt.
* **Permission nguy hiểm**: là những permission liên quan đến sự riêng tư và [bảo mật](https://tinhte.vn/tags/bao-mat/), hoặc có khả năng ảnh hưởng đến dữ liệu đang lưu trên điện thoại hay việc hoạt động của những app khác. Ví dụ, quyền đọc danh bạ của người dùng là một permission nguy hiểm. Nếu một app đòi permission nguy hiểm, người dùng sẽ phải chấp thuận thì nó mới được sử dụng. Một nhóm permission mà mình liệt kê bên dưới có thể có nhiều permission nhỏ hơn, lập trình viên có thể đòi 1, 2 hoặc tất cả permission nhỏ trong một nhóm (ví dụ: có app chỉ đòi đọc danh bạ, app khác đòi đọc và thay đổi danh bạ luôn). Tất cả đều sẽ được hỏi kĩ càng lúc bạn xài tính năng của app.

Người dùng nên hết sức cẩn thận khi cài đặt app với những permission vô lý chẳng hạn như : một ứng dụng xem ảnh thì tại sao lại đòi quyền permission là đọc danh bạ. Hoặc ứng dụng nghe nhạc tại sao lại đòi quyền truy cập tin nhắn

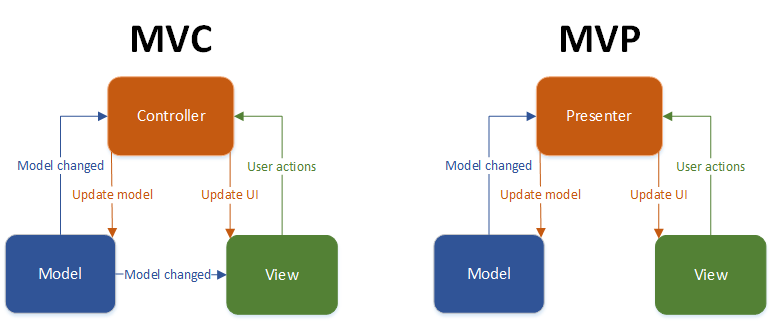
**CHƯƠNG 3 : THIẾT KẾ**

1. **Thiết kế giao diện**

* Material deign đang là xu hướng thiết kế của mọi ứng dụng từ mobie cho đến ứng dụng web, và không ngoại lệ tôi cũng đã áp dụng xu hướng này vào ứng dụng TPlayer.
* Material Design là ngôn ngữ thiết kế mới được Google dựng lên chủ yếu cho ứng dụng Android cũng như các nền tảng khác. Ngôn ngữ thiết kế này được giới thiệu trong Android 5.0 Lollipop
* Material Design ra đời với nhiệm vụ cực kì quan trọng: hợp nhất thiết kế giao diện phần mềm xuyên suốt nhiều loại thiết bị khác nhau – điện thoại, máy tính bảng, máy tính, thiết bị đeo được, TV, tất cả mọi thứ. Nó áp dụng không chỉ với các sản phẩm của Google mà còn cho các lập trình viên bên thứ ba lẫn các đối tác phần cứng đang tham gia vào hệ sinh thái Google. Nếu chúng ta xem Android là một hành tinh, và các ứng dụng chính là những quốc gia trên đó, thì Material Design chính là một “bộ luật” mang tính quốc tế.
* Material design có nhiều thành phần giao diện bắt mắt như là “Floating Action Button”. Tuy nhiên các thành phần giao diện Material vốn chỉ có trong Android 5.0 trở lên. Do đó sẽ phải tự viết lại các thành phần đó, hoặc đơn giản hơn là tìm một thư viện ngoài nếu như muốn đưa Material design trên các thiết bị chạy Android 4.4 trở về trước
* Một trong những khía cạnh quan trọng nhất của Material đó là việc sử dụng nhiều lớp đồ họa cũng như các chiều không gian. Theo một góc nhìn nào đó thì ngôn ngữ thiết kế này cũng giống như là những tờ giấy, những tờ giấy ma thuật. Nó có dạng phẳng, nó trôi nổi trên bề mặt nền và có xu hướng mang nhiều màu trắng. Ngoài ra còn có những chỗ đổ bóng đồng nhất, và một nguồn sáng cố định chiếu sáng các vật thể trên màn hình. Khi bạn di chuyển các đối tượng, chúng không chỉ đơn biến mất đi mà trượt đi từ chỗ này sang chỗ khác một cách tự nhiên. Chúng cũng không tự mình di chuyển mà chỉ có người dùng mới có thể can thiệp.
* Một khía cạnh quan trọng khác của Material Design đó là việc sử dụng các hiệu ứng chuyển động. Khi bạn chạm vào một nút nào đó thì không chỉ có các menu bung ra, mà kèm theo đó là hiệu ứng menu xổ xuống từ nơi bạn đã nhấn vào. Một ít vòng tròn chuyển động nhẹ nhàng cũng được hiển thị, giống như khi bạn chạm ngón tay xuống một mặt hồ phẳng lặng. Duarte nói: “Chúng tôi muốn tất cả năng lượng của giao diện phải đến từ những cái chạm của các bạn. Ngay cả việc điều hướng cũng phải tuân theo quy tắc này”.
* Qua những đoạn mô tả nói trên, chúng ta có thể cảm nhận rằng Material Design sẽ góp phần tạo ra những giao diện vô cùng đẹp mắt và sinh động. Vậy còn trong thực tế thì sao? Ứng dụng Inbox mới ra mắt gần đây chính là một thứ được Google để trình diễn về Material Design. Nó có rất nhiều lớp đối tượng đồ họa, chọn lựa màu sắc vô cùng táo bạo, đậm và hiệu ứng cũng rất trực quan.

1. **Thiết kế ứng dụng**
2. **Mô hình code**

* Mô hình code là một chuẩn để viết code trong lập trình
* Mô hình code thường được các chuyên gia, hoặc các lập trình viên có nhiều kinh nghiệm sáng tạo ra khi họ gặp nhiều vấn đề khi viết code
* Hiện nay có 2 mô hình code cực kì phổ biến áp dụng trong mobie cũng như trên nền web đó là mô hình MVC (Model – View - Controller) và mô hình MVP (Model – View - Presenter).
* MVC là viết tắt của Model – View – Controller. Là một kiến trúc phần mềm hay mô hình thiết kế được sử dụng trong kỹ thuật phần mềm. Nói cho dễ hiểu, nó là mô hình phân bố source code thành 3 phần, mỗi thành phần có một nhiệm vụ riêng biệt và độc lập với các thành phần khác.
* Controller : Giữ nhiệm vụ nhận điều hướng các yêu cầu từ người dùng và gọi đúng những phương thức xử lý chúng… Chẳng hạn thành phần này sẽ nhận request từ url và form để thao tác trực tiếp với Model.
* Model : Đây là thành phần chứa tất cả các nghiệp vụ logic, phương thức xử lý, truy xuất database, đối tượng mô tả dữ liệu như các Class, hàm xử lý…
* View : Đảm nhận việc hiển thị thông tin, tương tác với người dùng, nơi chứa tất cả các đối tượng GUI như textbox, images… Hiểu một cách đơn giản, nó là tập hợp các form hoặc các file HTML.
* MVP đại diện cho Model – View – Presenter.
* M – Model: lớp xử lí dữ liệu. Lớp này sẽ chịu trách nhiệm lấy dữ liệu từ database hoặc network một cách bất đồng bộ. Sau đó sẽ trả về dữ liệu cho Presenter thông qua các hàm callback.
* V- View: lớp xử lí view. Lớp này chịu trách nhiệm tìm view (bind view), đưa dữ liệu vào view, animation, kiểm soát các input event của user và gửi cho present các event
* P – Presenter: lớp này sẽ là lớp xử lí các bussiness logic. Đây là sẽ lớp trung gian có thể giao tiếp được với hai lớp M & V (chú ý M – V ko  giao tiếp trực tiếp với nhau). Khi lớp View nhận một input event sẽ gửi sự kiện xuống lớp P, lớp P lấy sẽ liệu từ lớp Model và gửi lại cho lớp V và hướng dẫn lớp V cách hiển thị.
* Nếu làm đúng, trong lớp P sẽ chỉ có code Java. Mọi code liên quan đến Android sdk sẽ được thực hiện trong lớp View hoặc lớp Model. Như vậy, sẽ dễ để viết unittest (Local test cho lớp P chỉ toàn Java và instrument test cho lớp View). Đồng thời code của chúng ta sẽ rõ ràng rành mạch, xử lí riêng biệt từng phần giữa business logic, view. Chương trình sẽ được chia nhỏ ra, các task lớn được chia thành task nhỏ.



* Khoảng 2 năm trở lại đây thì mô hình MVP đang là mô hình phổ biến hơn vì những lợi ích nó mang lại, code theo mô hình MVP giúp maintain phần mềm dễ dàng và mạch lạc hơn so với code của mô hình MVC
* Trong khuôn khổ của ứng dụng này tôi cũng sử dụng mô hình code MVP. Tuy cách code này có dài hơn, phân chia thành nhiều class nhỏ hơn nhưng việc maintain hoặc thêm những feature mới sẽ đơn giản và dễ dàng hơn sau này

1. **Cấu trúc code**

* Cấu trúc code được tổ chức rõ rang phân chia từng Package riêng rẽ với các chức năng tương ứng.
* Code theo mô hình MVP giúp tập trung những Ui một nơi và đọc lập với code Logic giúp code trông clean hơn và dễ hiểu hơn

**CHƯƠNG 4 : CÀI ĐẶT VÀ KIỂM THỬ ỨNG DỤNG**

**Cài đặt ứng dụng**

**Kiểm thử ứng dụng**