LỜI MỞ ĐẦU

Sự phát triển như vũ bão của công nghệ tông tin (CNTT) đã tác động mạnh mẽ và to lớn đến mọi mặt đời sống kinh tế xã hội. Ngày nay, CNTT đã trở thành một trong những động lực quan trọng nhất của sự phát triển.

Đối với quá trình hoàn thiện sản phẩn CNTT trong quá trình học tập và nghiên cứu, cụ thể là phần mềm "Ứng dụng bán hàng" đã giúp em có những khái niệm cơ bản về cách hoạt động bán hàng tại một cửa hàng. Kết hợp xây dựng phần mềm với việc bổ sung những kiến thức đã có và tìm hiểu thêm những kiến thức mới thông qua lập trình Android môn Lập trình trên thiết bị di động.

Hiện nay, thương mại điện tử(TMĐT) được nhiều quốc gia quan tâm, coi đây là thúc đẩy phát triển kinh tế. TMĐT đem lại lợi ích cho doanh nghiệp giảm chi phí tiếp thị và thời gian, tạo dựng và cũng cố quan hệ bán hàng.

Việc mua hàng qua mạng với thủ tục đăng ký đơn giản và nhanh chống. Vì vậy việc mua hàng qua mạng rất được nhiều người quan tâm.

Do thời gian còn hạn chế nên đồ án có những phần chưa tối ưu mong thầy và các bạn có thể đóng góp ý kiến để khắc phục những điểm hạn chế đó.

Em xin gửi lời cảm ơn chân thành đến Thầy Giang Hào Côn đã tận tình giảng dạy, hướng dẫn chúng em trong suốt thời gian học vừa qua và các bạn trong lớp đã góp ý và giúp đỡ em trong quá trình thực hiện đồ án này.

NHẬN XÉT CỦA GIÁO VIÊN

MỤC LỤC

[PHÂN 1. LÝ THUYẾT 1](#_Toc7977958)

[CHƯƠNG 1. TỔNG QUAN VỀ LẬP TRÌNH ANDROID 1](#_Toc7977959)

[1.KHÁI NIỆM 1](#_Toc7977960)

[2. XU THẾ PHÁT TRIỄN CÔNG NGHỆ DI ĐỘNG 3](#_Toc7977961)

[3. THỊ TRƯỜNG THIẾT BỊ DI ĐỘNG 5](#_Toc7977962)

[4.TÍNH NĂNG CỦA ANDROID 6](#_Toc7977963)

[CHƯƠNG 2. CÁC THÀNH PHẦN ANDROID 7](#_Toc7977964)

[1.Activity 7](#_Toc7977965)

[2. Service 7](#_Toc7977966)

[3. Content Provider 8](#_Toc7977967)

[4. Intent 8](#_Toc7977968)

[5. Broadcast Receiver 8](#_Toc7977969)

[6. Notification 8](#_Toc7977970)

[7. Context ứng dụng 8](#_Toc7977971)

[8.Resource 9](#_Toc7977972)

[9. Manifest 9](#_Toc7977973)

[CHƯƠNG 3. GIAO DIỆN NGƯỜI DÙNG VÀ XỬ LÝ SỰ KIỆN 9](#_Toc7977974)

[1. GIAO DIỆN NGƯỜI DÙNG 9](#_Toc7977975)

[2.XỬ LÝ SỰ KIỆN 14](#_Toc7977976)

[CHƯƠNG 4. TÀI NGUYÊN ỨNG DỤNG TRONG ANDROID VÀ INDEX 15](#_Toc7977977)

[1.ĐỊNH NGHĨA 15](#_Toc7977978)

[2.CUNG CẤP TÀI NGUYÊN THAY THẾ 17](#_Toc7977979)

[3.CÁC LOẠI TÀI NGUYÊN 17](#_Toc7977980)

[CHƯƠNG 5. MULTIMEDIA 18](#_Toc7977981)

[1.TỔNG QUAN 18](#_Toc7977982)

[CHƯƠNG 6. BỘ QUẢN LÝ ASSET, SHAREPREFERENCE VÀ SQL 23](#_Toc7977983)

[1.QUẢN LÝASSET 23](#_Toc7977984)

[2.SharedPreference 25](#_Toc7977985)

[3.SQL 26](#_Toc7977986)

[CHƯƠNG 7. BỘ NHỚ THIẾT BỊ VÀ TOOLBAR , ACTIONBAR 27](#_Toc7977987)

[1. BỘ NHỚ THIẾT BỊ 27](#_Toc7977988)

[Là nơi lưu trữ các dữ liệu cá nhân của từng ứng dụng, mà các dữ liệu này được lưu trữ và sử dụng cho riêng ứng dụng đó. Các ứng dụng khác không thể truy cập vào được. Thông thường khi ứng dụng bị gỡ bỏ khỏi thiết bị. 27](#_Toc7977989)

[2.TOOLBAR 29](#_Toc7977990)

[3.ACTIONBAR 30](#_Toc7977991)

[PHẦN 2. THỰC HÀNH 31](#_Toc7977992)

[1.Giới thiệu đề tài 31](#_Toc7977993)

[2.Mô hình quan hệ dữ liệu 32](#_Toc7977994)

[3.Cài đặt 32](#_Toc7977995)

[3.1. Giao diện trên Android studio 32](#_Toc7977996)

[3.2.Giao diện và tác vụ trên điện thoại 33](#_Toc7977997)

[TÀI LIỆU THAM KHẢO 50](#_Toc7977998)

# PHÂN 1. LÝ THUYẾT

## CHƯƠNG 1. TỔNG QUAN VỀ LẬP TRÌNH ANDROID

### 1.KHÁI NIỆM

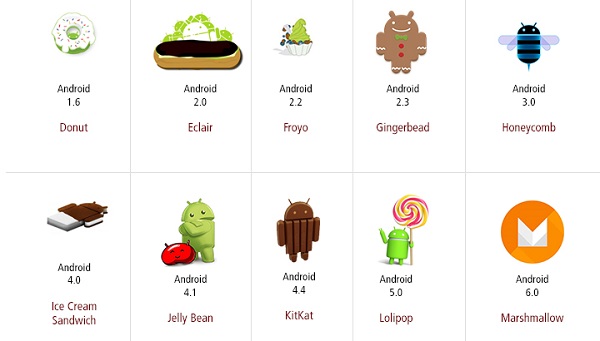
Android là hệ điều hành chạy trên điện thoại di động được phát triển từ nhân Linux. Mang nhiều đặc tính đặc trưng của một hệ điều hành di động nhưng vẫn mang những tính chất chung của các hệ điều hành. Đầu tiên được ra đời bởi công ty liên hợp Android, sau đó được Google mua lại và phát triển từ nằm 2005 và trở thành một hệ điều hành di động mã nguồn mở, miễn phí, mạnh mẽ và được ưa chuộng cao trên thế giới.

Android liên tục được phát triển, mỗi bản cập nhật từ Google là mỗi lần Android được tối ưu hóa để hoạt động tốt hơn, nhanh và ổn định hơn, hỗ trợ thêm công nghệ mới. Chẳng hạn như theo một đánh giá thì Android phiên bản Froyo 2.2 hoạt động nhanh hơn bản 2.1 tới 450%.

Vào năm 2008, hệ điều hành Android đã chính thức mở toàn bộ mã nguồn, đã mang lại một luồng sinh khí mới trong việc phát triển hệ điều hành dành cho di động. Quan trọng hơn hết nó mang lại cho học sinh sinh viên chúng ta một cơ hội tuyệt vời để khám phá, học hỏi, và sáng tạo phần mềm trên hệ điều hành mới mẻ này.

Hiện nay, phiên bản Nougat 7.0-7.1.2 đang là phiên bản đang được sử dụng ở hầu hết các model smartphone Android, phát hành ngày 22/8/2016 và hiện tại phiên bản mới nhất được công bố là phiên bản Android Oreo 8.0-8.1, phát hành ngày 21/8/2017.

-  Lịch sử của hệ điều hành Android theo trình tự:

**

*Hình 1. Các phiên bản hệ điều hành Android.*

*-* API là một giá trị số nguyên xác định duy nhất bản sửa đổi API khung được cung cấp bởi một phiên bản của nền tảng Android.

|  |  |  |  |
| --- | --- | --- | --- |
| **Phiên bản** | **API** | **Mã phiên bản** |  |
| Android 6.0 | 23 | MARSHMALLOW |  |
| Android 5.1 | 22 | LOLLIPOP\_MR1 |  |
| Android 5.0 | 21 | LOLLIPOP |  |
| Android 4.4W | 20 | KITKAT\_WATCH | KitKat for Wearables Only |
| Android 4.4 | 19 | KITKAT |  |
| Android 4.3 | 18 | JELLY\_BEAN\_MR2 |  |
| Android 4.2, 4.2.2 | 17 | JELLY\_BEAN\_MR1 |  |
| Android 4.1, 4.1.1 | 16 | JELLY\_BEAN |  |
| Android 4.0.3, 4.0.4 | 15 | ICE\_CREAM\_SANDWICH\_MR1 |  |
| Android 4.0, 4.0.1, 4.0.2 | 14 | ICE\_CREAM\_SANDWICH |  |
| Android 3.2 | 13 | HONEYCOMB\_MR2 |  |
| Android 3.1.x | 12 | HONEYCOMB\_MR1 |  |
| Android 3.0.x | 11 | HONEYCOMB |  |
| Android 2.3.4  Android 2.3.3 | 10 | GINGERBREAD\_MR1 |  |
| Android 2.3.2  Android 2.3.1  Android 2.3 | 9 | GINGERBREAD |  |
| Android 2.2.x | 8 | FROYO |  |
| Android 2.1.x | 7 | ECLAIR\_MR1 |  |
| Android 2.0.1 | 6 | ECLAIR\_0\_1 |  |
| Android 2.0 | 5 | ECLAIR |  |
| Android 1.6 | 4 | DONUT |  |
| Android 1.5 | 3 | CUPCAKE |  |
| Android 1.1 | 2 | BASE\_1\_1 |  |
| Android 1.0 | 1 | BASE |  |

*Bảng 1. Các phiên bản hệ điều hành Android*

### 2. XU THẾ PHÁT TRIỄN CÔNG NGHỆ DI ĐỘNG

Ngành lập trình ứng dụng di động đang thay đổi hàng năm, nhiều công nghệ mới nổi lên thay thế cái cũ, làm cho quá trình phát triển trở nên hiệu quả, dễ dàng, mạnh mẽ và nhanh chóng hơn.

#### 2.1. Internet of Things (IOT)

  IOT là một viễn cảnh của thế giới, khi mà mỗi đồ vật, con vật, con người được cung cấp một định danh của riêng mình và tất cả có khả năng truyền tải, trao đổi thông tin, dữ liệu qua một mạng duy nhất mà không cần đến sự tương tác trực tiếp giữa người với người, hay người với máy tính.

  IOT đang là **xu hướng phát triển lập trình ứng dụng di động** **phổ biến nhất** trong năm nay. Hầu hết các nhà phát triển đều đang tham gia vào dự án này. IOT giúp cho cuộc sống con người trở nên dễ dàng, thông minh và tự chủ hơn.

#### 2.2. Phát triển công nghệ đa nền tảng trên ứng dụng di động

Đa nền tảng, như tên gọi của nó, bạn không cần phát triển ứng dụng cho một nền tảng cụ thể. Với các công cụ phát triển đa nền tảng, bạn có thể phát triển, lập trình các ứng dụng di động trên nền tảng độc lập hay thông qua nhiều nền tảng khác.

#### 2.3. Phát triển ứng dụng di động doanh nghiệp

Các doanh nghiệp luôn tìm cách nâng cao năng suất, tạo các sản phẩm tốt nhất nhằm tối ưu hoá hiệu quả sử dụng thời gian. Nếu bạn có giải pháp tốt hơn cho doanh nghiệp, hãy để giải pháp đó giúp nâng cao năng suất và lợi nhuận. Khi đó doanh nghiệp sẽ trả công bạn nhiều hơn những gì bạn có thể nhận được bằng các ứng dụng cho người dùng. Đó là lý do vì sao việc lập trình ứng dụng di động trong doanh nghiệp đang phát triển nhanh chóng – Giúp doanh nghiệp phát triển tốt hơn và luôn tự chủ.

#### 2.4. Phát triển điện toán “đám mây” trong ứng dụng di động

Việc áp dụng điện toán “đám mây” trong ứng dụng di động giúp quá trình phát triển trở nên dễ dàng và hiệu quả hơn. Bằng cách kết hợp điện toán “đám mây” với phát triển ứng dụng di động, bạn sẽ không cần cài đặt SDK’s ( Công cụ phát triển ứng dụng ), không cần mô phỏng hay kiểm định với các công cụ khác.

#### 2.5. Mobile App Marketing

  Tiếp thị ứng dụng di động thực sự phát triển vào khoảng giữa năm 2019. Chắc bạn cũng biết có rất nhiều ứng dụng di động có sẵn trong cửa hàng của iOs và Android. Nhưng chúng không hề phổ biến, đa phần mọi người không biết sự tồn tại của các ứng dụng này. Một trong những xu hướng phát triển ngành lập trình di động hiện nay chính là khiến người dùng biết đến sự tồn tại đó - Không có việc gì được phổ biến trong thời gian gần đây mà không cần đến “marketing”.

* Trên thế giới:

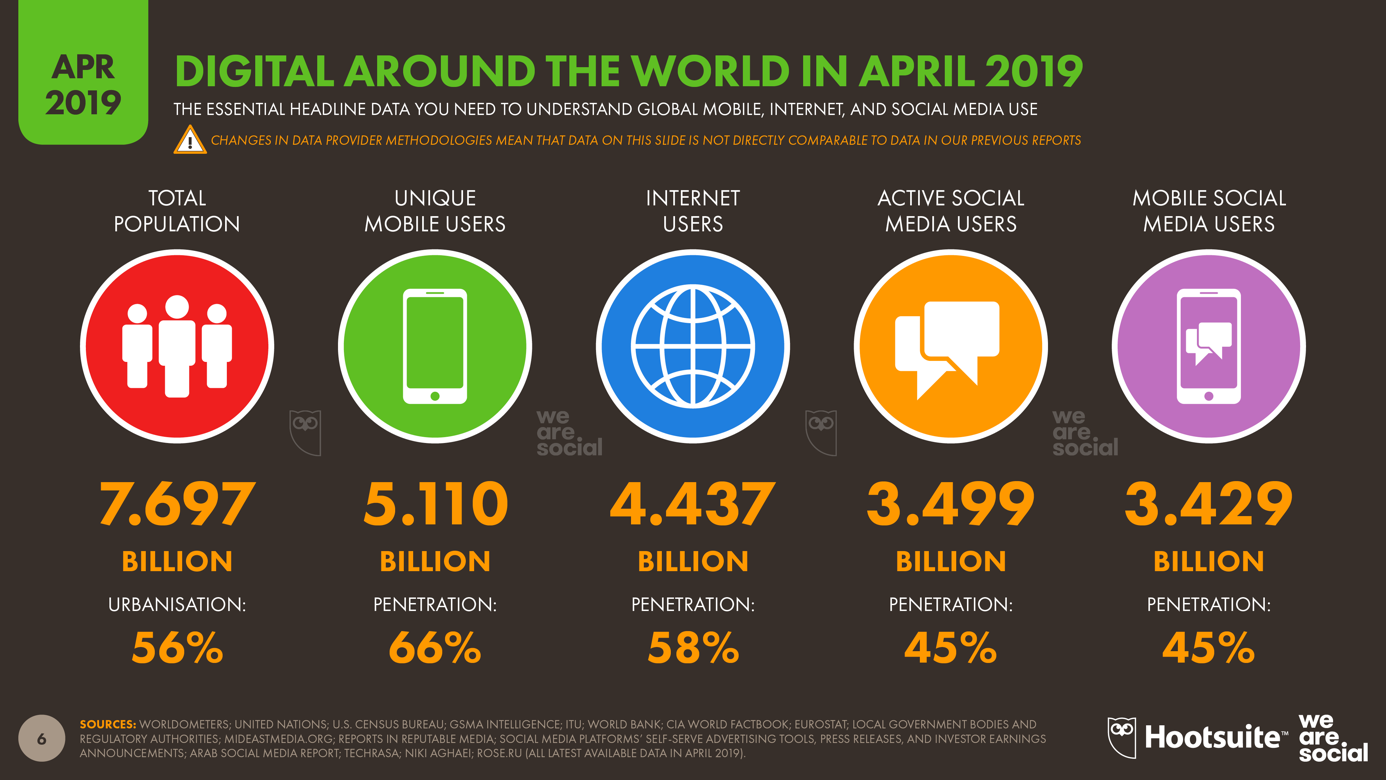
o Tháng 04/2019, trang WeAreSocial đã đưa ra báo cáo với những chỉ số phát triển rất đang kinh ngạc của Thế Giới Số. Cụ thể số liệu thống kê của WeAreSocial cho thấy:

   Hiện tại có hơn 5,1 tỷ người trên toàn thế giới sử dụng điện thoại di động - tăng 2,7% mỗi năm - với điện thoại thông minh chiếm hơn hai phần ba tổng số thiết bị đang sử dụng hiện nay.

 Người dùng internet toàn cầu đã tăng 8,6% trong 12 tháng qua, với 350 triệu người dùng mới đóng góp vào tổng số 4,437 tỷ vào đầu tháng 4 năm 2019.

   Số người dùng phương tiện truyền thông xã hội cũng đã đăng ký tăng trưởng vững chắc, tăng hơn 200 triệu kể từ thời điểm này năm ngoái để đạt gần 3,5 tỷ vào thời điểm xuất bản.

     Khoảng 98% người dùng phương tiện truyền thông xã hội trên toàn thế giới - hơn 3,4 tỷ người - truy cập các nền tảng xã hội thông qua các thiết bị di động.



*Hình 2. Phân tích bối cảnh dữ liệu- Nguồn: WeAreSocial*

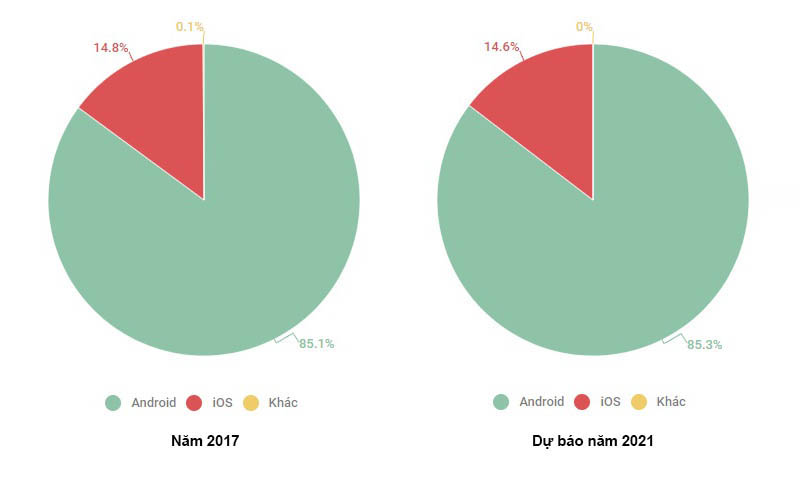
### 3. THỊ TRƯỜNG THIẾT BỊ DI ĐỘNG

-Hiện nay, khi nói về hệ điều hành smartphone, chúng ta thường chỉ biết đến Android và iOS. Dựa trên dữ liệu từ Newzoo (được eMarketer dẫn lại), 95% điện thoại thông minh trên toàn cầu chạy 1 trong 2 nền tảng này. Tuy nhiên, tầm ảnh hưởng của chúng rất khác nhau tùy theo khu vực địa lý.

Smartphone Android chiếm 3/4 thế giới. Android vẫn là hệ điều hành nắm lợi thế ở quy mô toàn cầu.

\_Cụ thể, đến tháng 11/2017, 75% smartphone trên toàn thế giới cài đặt Android, tỷ lệ cũng được cung cấp bởi Newzoo.

-Android vẫn tiếp tục chiếm ưu thế về thị phần trong tương lai.



*Hình 3. Thị phần smartphone theo hệ điều hành*

-Với xu thế phát triển công nghệ di động nhanh và mạnh như hiện nay, thị trường thiết

bị Anroid chiếm vị trí cao nhất không chỉ ở Việt Nam mà trên toàn thế giới, thì nhu cầu sử dụng các ứng dụng cho các thiết bị Android là rất lớn. Vì vậy, nhu cầu tuyển dụng lập trình viên Android cũng rất lớn và sẽ tăng nhanh.

### 4.TÍNH NĂNG CỦA ANDROID

### 

-Android là một hệ điều hành mạnh mẽ cạnh tranh với Apple 4GS và hỗ trợ các tính năng tuyệt vời. Một vài trong số chúng được liệt kê dưới đây:

|  |  |
| --- | --- |
| STT | Tính năng và mô tả |
| 1 | Giao diện người dùng đẹp Màn hình cơ bản của HĐH Android cung cấp giao diện người dùng đẹp và trực quan. |
| 2 | Kết nối GSM / EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC và WiMAX. |
| 3 | Lưu trữ SQLite, một cơ sở dữ liệu quan hệ nhẹ, được sử dụng cho mục đích lưu trữ dữ liệu. |
| 4 | Hỗ trợ truyền thông H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF và BMP. |
| 5 | Nhắn tin SMS và MMS |
| 6 | Trình duyệt web Dựa trên công cụ bố cục WebKit nguồn mở, kết hợp với công cụ JavaScript V8 của Chrome hỗ trợ HTML5 và CSS3. |
| 7 | Cảm ưng đa điểm Android có hỗ trợ riêng cho cảm ứng đa điểm, ban đầu được cung cấp trong các thiết bị cầm tay như HTC Hero. |
| 8 | Đa tác vụ Người dùng có thể nhảy từ nhiệm vụ này sang nhiệm vụ khác và đồng thời nhiều ứng dụng khác nhau có thể chạy cùng lúc. |
| 9 | Vật dụng thay đổi kích thước Các widget có thể thay đổi kích thước, vì vậy người dùng có thể mở rộng chúng để hiển thị nhiều nội dung hơn hoặc thu nhỏ chúng để tiết kiệm dung lượng. |
| 10 | Wi-Fi Direct Một công nghệ cho phép các ứng dụng khám phá và ghép nối trực tiếp, qua kết nối ngang hàng băng thông cao. |

*Bảng 2. Các tính năng Android*

## 

## CHƯƠNG 2. CÁC THÀNH PHẦN ANDROID

### 1.Activity

-Hiểu một cách nôm na thì Activity tương tự như khái niệm Window (cửa sổ hay Form) trong hệ điều hành Windows. Khi khởi động một ứng dụng, bao giờ cũng có một Activity được gọi, hiển thị màn hình giao diện của ứng dụng cho phép người dùng tương tác.

### 2. Service

-Thành phần chạy ẩn của Android. Service sử dụng để update dữ liệu, đưa ra các cảnh báo (notification) và không bao giờ hiển thị cho người dùng thấy.

– Ví dụ: khi bạn yêu cầu phát một bài nhạc, sẽ có một Service chạy bên dưới để phát i nhạc. Khi bạn download một tập tin, sẽ có một Serive thực hiện tác vụ download.

### 3. Content Provider

-Kho dữ liệu chia sẻ, được dùng để quản lý và chia sẽ dữ liệu giữa các ứng dụng.

–Ví dụ: thông tin người dùng lưu trong contact, dữ liệu lưu trữ trên SQL Lite, dữ liệu lưu trữ trong các tận tin

### 4. Intent

-Được sử dụng để truyền các thông báo nhằm khởi tạo một Activity hoặc Service để thực hiện công việc bạn mong muốn.

– Ví dụ: khi mở một trang web, bạn sẽ gửi một intent đi để tạo một Activity mới hiển thị trang web đó.

### 5. Broadcast Receiver

-Thành phần thu nhận các intent từ bên ngoài gửi tới.

– Ví dụ: bạn viết một chương trình thay thế cho phần gọi điện thoại mặc định của Android, khi đó, bạn cần 1 Broadcast Receiver để nhận các intent là các cuộc gọi đến.

### 6. Notification

- Đưa ra các cảnh báo mà không làm cho các Activity phải ngừng hoạt động

Activity, Service, Broadcast Receiver và Content Provider là những thành phần chính cấu thành nên ứng dụng Android, bắt buộc phải khai báo trong AndroidManifest.

### 7. Context ứng dụng

-Khi ứng dụng biên dịch, một lớp có tên là **R** được tự động tạo ra, nó chứa các tham khảo trỏ đến tài nguyên của ứng dụng. Các file manifest và tài nguyên sẽ được kết hợp lại với nhau được hiểu là Context ứng dụng. Context trong ứng dụng được biểu diễn bằng lớp Context, được sử dụng trong ứng dụng để thông qua nó truy cập tới các loại tài nguyên khi ứng dụng đang chạy.

-Ngoài ra, có nhiều phương thức dựa vào context để lấy các thông tin về môi trường ứng dụng hoạt động khi chạy.

### 8.Resource

-Ứng dụng Android được đóng gói ngoài các loại file byte code, nó còn chứa tập hợp các file tài nguyên (Resource) như chuỗi ký tự, hình ảnh, font chữ, màu sắc ... những thành phần xuất hiện trong giao diện người dùng và được trình bày với file XML.

-Mặc định những file này lưu trữ bên trong thư mục / res.

### 9. Manifest

-Manifest của ứng dụng định nghĩa trong file định dạng XML, nó mô tả để hệ thống Android hiểu khái quát về ứng dụng như các Activity, các Service, Broadcast Receiver, Content Provider, các quyền truy cập.

-Những thông tin này cần thiết để hệ thống Android chạy được ứng dụng như mong muốn.

## 

## CHƯƠNG 3. GIAO DIỆN NGƯỜI DÙNG VÀ XỬ LÝ SỰ KIỆN

### 1. GIAO DIỆN NGƯỜI DÙNG

Giao diện người dùng là một trong những yếu tố quan trọng, quyết định sự thành công của một ứng dụng Android. Ứng dụng Android muốn thành công thì phải có giao diện trực quan, dễ hiểu và dễ sử dụng. Ở bài này, chúng ta sẽ tìm hiểu cấu trúc giao diện, các thành phần trên giao diện và các thuộc tính của chúng.

Giao diện được tạo bởi nhiều tài nguyên như: layout, các diều khiển (control), tài nguyên hình ảnh, màu sắc, v.v... Và tất cả các tài nguyên dùng để thiết kế giao diện sẽ được lưu trong thư mục res/.

#### 1.1. Layout

-  Các tài nguyên layout được lưu trữ trong thư mục res/layout.

-  Có thể có nhiều thư mục layout theo từ hạn định khác nhau:

-  Ví dụ: layout-land, layout-xhdpi...

-  Truy xuất: bao gồm 2 cách thức:

o Java: R.layout.<tên tài nguyên>.

o XML: @[pakage:]layout/<tên tài nguyên>. - Các định dạng Layout

* Layout bao gồm những lớp được mở rộng từ lớp ViewGroup mà ở đó ta có thể sắp xếp và bố trí các điều khiển cho một giao diện. Tùy thuộc vào cách bố trí của mỗi người mà sẽ có các giao diện khác nhau và tất nhiên cần cân bằng giữa tính tiện dụng và tính thẩm mỹ. Android cũng cấp một số các lớp Layout cho phép chúng ta có thể sử dụng hoặc tùy chỉnh tùy thuộc vào từng ứng dụng.
* Bao gồm các lớp kế thừa từ ViewGroup: AbsoluteLayout (Deprecated), AdapView (ListView, Gridview...), DrawerLayout, FragmentBreadCrumbs, FrameLayout, GridLayout, LinearLayout, PagerTitleStrip, RelativeLayout, SlidingDrawer, SlidingPaneLayout, SwipeRefreshLayout, ViewPager.

##### 1.1.1. Frame Layout

- Sử dụng trong các trường hợp xây dựng bố cục tổ chức hiển thị một đối tượng duy nhất.

-  Đối tượng mặc định vị trí top-left trên FrameLayout, có thể sử dụng thuộc tính Gravity để thiết lập lại vị trí.

-  Frame Layout có thể chứa nhiều View và các View này có thể sắp chồng lên nhau. Và các View nằm dưới có thể bị các View nằm trên che khuất. Vì vậy, thường Frame Layout chỉ chứa một View.

-  Ví dụ khai báo:

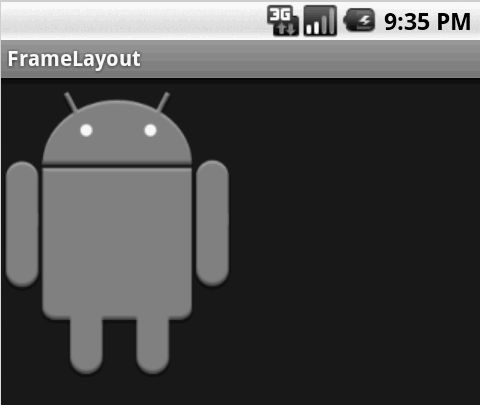
<?xml version="1.0" encoding="utf-8"?> <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent”" > <ImageView

android:id="@+id/imageView1" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:src="@drawable/android" />

</FrameLayout>



*Hình 4. FrameLayout*.

- Các đối tượng kế thừa phổ biến của FrameLayout:

o ViewFlipper: đối tượng cho phép thực hiện hiển thị các đối tượng ở chế độ phân trang, chỉ hiển thị một đối tượng ở một thời điểm.

Ví dụ khai báo:

<ViewFlipper android:layout\_width="match\_parent"

android:layout\_height="match\_parent”" > </ViewFlipper> 

Các phương thức sử dụng:

startFlipping

setAutoStart

showNext

showPrevious

o ScrollView: đối tượng cho phép thực hiện hiển thị các đối tượng ở chế độ cuộn màn hình, chỉ cho phép chứa một đối tượng ở một thời điểm.

Ví dụ khai báo:

- Các phương thức sử dụng:

o setFillViewPort

o scrollBy

o scrollTo

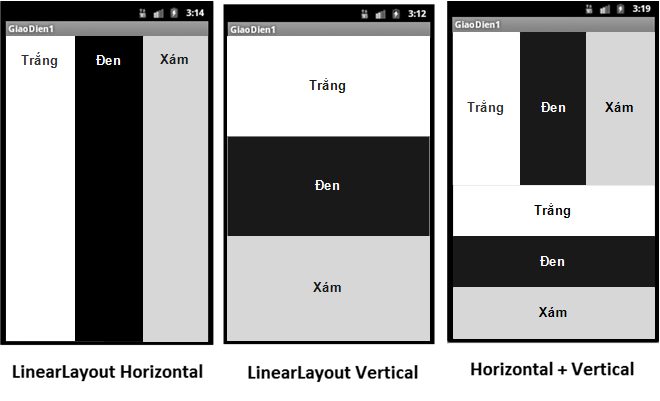
o smoothScrollBy

o smoothScrollTo

##### 1.1.2. Linear Layout

-  Giao diện một hướng, có nghĩa là các View sẽ được sắp xếp lên Layout theo một hướng  nhất định hoặc theo chiều ngang (từ trái qua phải) hoặc theo chiều dọc (từ trên xuống dưới). Thuộc tính weight của Layout sẽ cho phép điều chỉnh kích thước của các View dựa trên mối quan hệ của chúng (cùng một hàng chẳng hạn).

-  Được xem dạng là Layout dễ sử dụng nhất, chúng cho phép chúng ta tạo ra các giao diện đơn giản bằng cách sắp xếp các View theo một chiều duy nhất. Điều này gây khó khăn khi chúng ta cần thiết kế các giao diện phức tạp, do đó LinearLayout thường được sử dụng trong một Layout khác. Ví dụ, ở đây ta thiết kế khác LinearLayout lồng nhau, một Layout dạng Horizontal để chứa hai Button và một Layout dạng Vertical cho phép hai Button nằm trên ListView.



*Hình 5. Linear Layout*

- Ví dụ khai báo:

Hình 3.2. Các dạng LinearLayout.

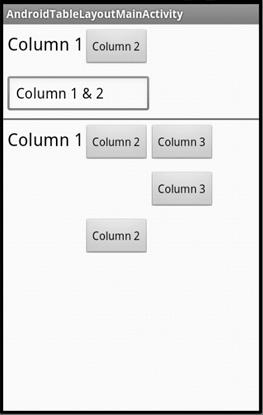
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android" android:layout\_width="match\_parent" android:layout\_height="match\_parent” android:orientation=“vertical” >

</LinearLayout>

##### 1.1.3. Table Layout

-  Table Layout là đối tượng layout kế thừa từ LinearLayout, cho phép hiển thị các đối  tượng theo nhiều dòng (TableRow).

-  Mỗi dòng có thể chứa nhiều View, mỗi View được xem là một cột.



*Hình 6. Table layout*

- Ví dụ khai báo:

<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent" android:layout\_height="match\_parent” >

<Tablerow>

<Button/>

</Tablerow>

<Tablerow>

<Button/>

</Tablerow>

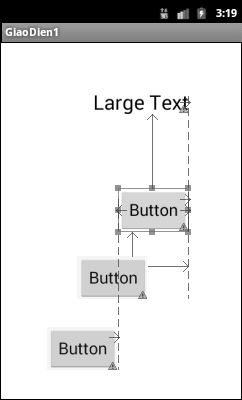
</TableLayout>

- Có thể kéo dãn độ rộng một cột với thuộc tính: android: layout\_span.

##### 1.1.4. Relative Layout

-  Các View khi được đặt lên Layout sẽ có vị trí phụ thuộc vào View đã đặt vào trước nó,  do đó khi thay đổi vị trí của một View sẽ làm thay đổi vị trí của các View còn lại.

-  Đối với các giao diện phức tạp hơn thì việc dùng Relative Layout sẽ dễ dàng hơn. Các View khi được đặt lên Layout sẽ có vị trí phụ thuộc vào vị trí các View đã đặt vào trước  đó và đối tượng đang chứa nó.

-  Ví dụ ở đây ta có thể xây dựng Layout cho TextView nằm phía trên ba Button 

*Hình 7. RelativeLayout.*

* Chúng ta có thể kết hợp các loại layout lại với nhau để thiết kế giao diện. Tuy nhiên, việc tổ chức nhiều layout lồng nhau trên một giao diện sẽ tạo ra độ phức tạp và có thể làm giảm tốc độ xử lý. Vì vậy, khi tạo giao diện, chúng ta nên chú ý và tránh sử dụng nhiều layout chồng nhau.

### 2.XỬ LÝ SỰ KIỆN

|  |  |
| --- | --- |
| Xử lý sự kiện | **Tên xữ lý sự kiện** |
| onClick() | **OnClickListener()**Đăng ký sự kiện khi người dùng hoặc click hoặc chạm (touche) hoặc focuse trên bất ký widget như button, text, image vv. Chúng ta sẽ sử dụng phương onClick() để xử lý sự kiện. |
| onLongClick() | **OnLongClickListener()**Đăng ký sự kiện khi người dùng hoặc click hoặc chạm (touche) hoặc focus trên bất ký widget như button, text, image vv. trong một hoặc nhiều giây. Chúng ta sẽ sử dụng phương onLongClick() để xử lý sự kiện. |
| onFocusChange() | **OnFocusChangeListener()**Sự kiện phát sinh khi widget mất focus. |
| onKey() | **OnFocusChangeListener()**Sự kiện phát sinh khi người dùng focus trên widget và nhấn (presse) hoặc thả (release) một phím trên thiết bị. |
| onTouch() | **OnTouchListener()**Sự kiện phát sinh khi người dùng nhấn phím, nhả phím, hoặc bất kỳ cử chỉ chuyển động trên màn hình. |
| onMenuItemClick() | **OnMenuItemClickListener()**Sự kiện phát sinh khi người dùng chọn một mục trong menu. |
| onCreateContextMenu() | **OnCreateContextMenuItemListener()**Sự kiện phát sinh khi người dùng chọn một mục trong menu ngữ cảnh (Context Menu) |

*Bảng 3. Bảng xữ lý sự kiện Android*

- Ví dụ: bắt lại các sự kiện nhấn xảy ra trên điều khiển:

|  |
| --- |
| view.setOnCLickListener(new OnClickLisntener() { @Override  public void onClick(View v) {  Log.i(“View”, “onClick”);  } }); |

## 

## CHƯƠNG 4. TÀI NGUYÊN ỨNG DỤNG TRONG ANDROID VÀ INDEX

### 1.ĐỊNH NGHĨA

-Tài nguyên là một dạng dữ liệu được xây dựng nhằm đáp ứng các yêu cầu về hiển thị  bao gồm hình ảnh, âm thanh, văn bản, các bố cục...tương thích cho từng thiết bị riêng biệt.

-Sau khi ngoại hiện hóa các tài nguyên ứng dụng của mình, bạn có thể truy cập chúng bằng cách sử dụng các ID tài nguyên được khởi tạo trong lớp R của dự án của bạn.

-Bạn nên đặt từng loại tài nguyên vào một thư mục con cụ thể trong thư mục res/ dự án của mình. Ví dụ, sau đây là phân cấp tệp của một dự án đơn giản:

MyProject/

src/

MyActivity.java

res/

drawable/

graphic.png

layout/

main.xml

info.xml

mipmap/

icon.png

values/

strings.xml

*Bảng 4. Tổ chức tài nguyên Android*

Thư mục res/ chứa tất cả tài nguyên (trong các thư mục con): một tài nguyên hình ảnh, hai tài nguyên bố trí, các thư mục mipmap/ cho biểu tượng của trình khởi chạy, và một tệp tài nguyên xâu.

* Các thư mục tài nguyên được hỗ trợ bên trong thư mục res/ của dự án.

o Animator: Các dạng tài nguyên chuyển hoạt. (property animation)

o Anim: Các dạng tài nguyên chuyển hoạt. (property animation)

o Color: Các dạng tài nguyên màu sắc.

o Drawable: Các dạng tài nguyên hình ảnh.

o Layout: Các dạng tài nguyên giao diện.

o Menu: Các dạng tài nguyên chỉ mục.

o Raw: Các dạng tài nguyên không biên dịch.

o Values: Các dạng tài nguyên cơ bản. Ví dụ: String, Integer, Boolean…

### 2.CUNG CẤP TÀI NGUYÊN THAY THẾ

-Hầu như mọi ứng dụng đều nên cung cấp các tài nguyên thay thế để hỗ trợ những cấu hình thiết bị cụ thể. Ví dụ, bạn nên bao gồm các tài nguyên vẽ được thay thế cho các mật độ màn hình khác nhau và tài nguyên xâu thay thế cho các ngôn ngữ khác nhau.

-Để quy định các phương án thay thế theo cấu hình cụ thể cho một tập hợp tài nguyên:

1.Tạo một thư mục mới trong res/ có tên theo dạng <em>&lt; resources\_name&gt;</em>-<em>&lt;config\_qualifier&gt;</em>.

&lt;resources\_name&gt;

&lt;qualifier&gt;  là tên quy định một cấu hình riêng mà những tài nguyên này sẽ được sử dụng cho nó (được định nghĩa trong bảng 2).

Bạn có thể nối nhiều hơn một &lt;qualifier&gt;. Tách riêng từng cái bằng một nét gạch.

2.Lưu các tài nguyên thay thế tương ứng vào thư mục mới này. Tệp tài nguyên phải được đặt tên đúng như các tệp tài nguyên mặc định.

* Ví dụ, sau đây là một số tài nguyên mặc định và thay thế:

res/

drawable/

icon.png

background.png

drawable-hdpi/

icon.png

background.png

### 3.CÁC LOẠI TÀI NGUYÊN

-  Các tài nguyên cơ bản được lưu trữ trong thư mục res/values.

-  Một số tên tập tin đề xuất trong values: string.xml, arrays.xml, colors.xml, dimens.xml,  ...

Để tạo một bí danh cho một xâu hiện có, chỉ cần sử dụng ID tài nguyên của xâu mong muốn làm giá trị cho xâu mới. Ví dụ:

<?xmlversion="1.0"encoding="utf-8"?>  
<resources>  
<string name="hello">Hello</string>  
    <string name="hi">@string/hello</string>  
</resources>

Tài nguyên R.string.hi lúc này là một bí danh cho R.string.hello các giá trị đơn giản cũng hoạt động tương tự. Ví dụ, màu sắc:

<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <color name="yellow">#f00</color>  
    <color name="highlight">@color/red</color>  
</resources>

## CHƯƠNG 5. MULTIMEDIA

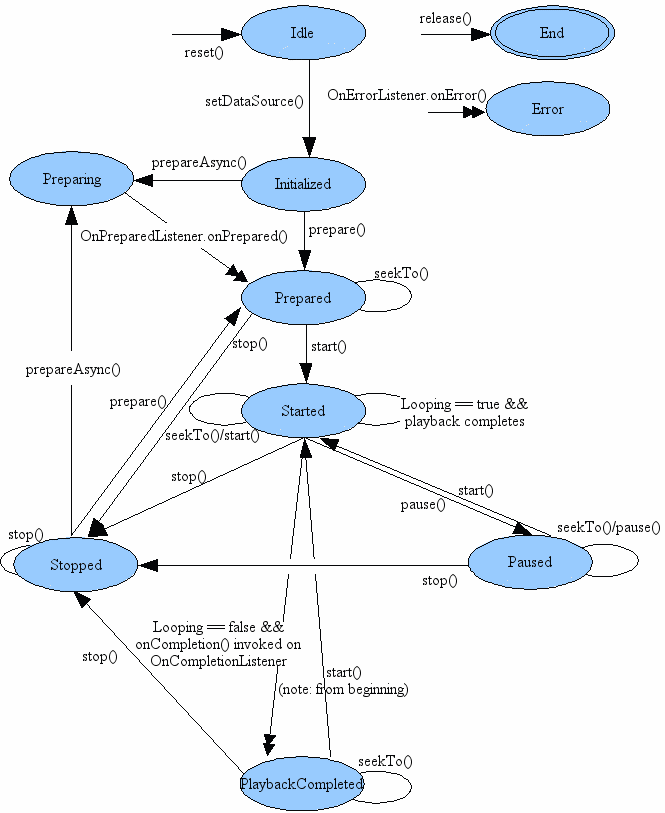
### 1.TỔNG QUAN

-Media Player trong Android là một framework hỗ trợ để play những định dạng như video, nhạc, ảnh trong ứng dụng của mình.

-Ứng dụng di động của các bạn hầu hết đều có thể phát 1 đoạn video, chơi một bài nhạc hoặc 1 bản audio từ trong bộ nhớ máy, trong ứng dụng hay phát trực tiếp trên mạng. Tất cả đều sử dụng MediaPlayer APIs.

#### 1.1. Vòng đời và các trạng thái của MediaPlayer

Vòng đời của MediaPlayer trông khá rắc rối và loằng ngoằng, trong qua trình mình giải thích ở dưới có thể gặp nhiều lỗi sai, vậy nên nếu có mong các bạn thông cảm và coment xuông dươi giúp mình. Dưới đây là hình ảnh minh họa vòng đời và các trạng thái của MediaPlayer



*Hình 8. Vòng đời Mediaplayer*

#### Phân tích trạng thái Multimedia

##### 1.2.1 Idle State

Khi mà MediaPlayer được khởi tạo bằng từ khóa new hoặc sau khi gọi method reset(), nó sẽ rơi vào trạng thái **Idle**(trạng thái chờ).

Có sự khác biệt giữa 2 cách này là khi khởi tạo bằng từ khóa new, ngay sau đó các bạn gọi các phương thức trong MediaPlayer như pause(), seekTo(), stop(), start(),…thì callback OnErrorListener.onError() sẽ không được cung cấp bởi hệ thống và trạng thái của MediaPayer vẫn không thay đổi. Còn nếu dung reset() mà sau đó gọi các phương thức trên thì call back OnErrorListener.onError() sẽ được cung cấp bởi hệ thống và mediaplayer sẽ được chuyển sang trạng thái **Error**.

Khi ở trạng thái này, chúng ta có thể chuyển qua trạng thái **End** nếu gọi method release(), hay sang trạng thái **Initialized** nếu sử dụng setDataSource().

##### 1.2.2. End state

MediaPlayer sẽ rơi vào trạng thái này khi gọi method release(), lúc đó MediaPlayer sẽ không được sử dụng nữa. Chúng ta có thể gọi phương thức release() ở bất kì trạng thái nào của MediaPlayer.

Khi MediaPlayer ở trạng thái này quá lâu thì sẽ không có cách nào đưa về bất kì trạng thái nào khác.

##### 1.2.3. Initialized State

Khi gọi phương thức setDataSource() và truyền vào nó các đối số phù hợp thì MediaPlayer sẽ được chuyển từ trạng thái **Idle** sang trạng thái **Initialized.**

Các phương thức setDataSource() có thể kể đến như: setDataSource(FileDescriptor), setDataSource(String), setDataSource(Context, Uri), setDataSource(FileDescriptor, long, long), hoặc setDataSource(MediaDataSource). Sẽ có 1 IllegalStateException được ném ra nếu setDataSource() được gọi ở các trạng thái khác, vậy nên chỉ setDataSource() khi MediaPlayer ở trạng thái **Idle**.

Ở trạng thái này có thể chuyển đến trạng thái **Prepared** nếu gọi method prepare(), còn khi gọi method prpareAsynce() sẽ chuyển sang trạng thái **Preparing**.

##### 1.2.4. Preparing state

khi chúng ta gọi prepareAsync(), đây là 1 phương thức không đồng bộ, mediaPlayer sẽ rơi vaò trạng thái này.

Khi sử dụng method này thì sẽ được chạy trong 1 other thread, do đó giao diện người dùng không bị chặn vì nó trả về gần như ngay lập tức, trong trạng thái này, sẽ check trong local đã có file nhạc hay file audio chưa, nếu có rồi thì lấy ra và chuyển luôn vào trạng thái **Prepared**, còn nếu chưa có thì sẽ thực hiện các thao tác connect đến server và buffer. Khi mọi thao tác chuẩn bị trên network được hoàn thành thì người dung sẽ được hệ thống cung cáp callback OnPreparedListener.onPrepared () khi mà trước đó đã setOnPreparedListener(), sau đó sẽ chuyển sang trạng thái **Prepared**.

##### 1.2.5. Prepared State

MediaPlayer ở trong trạng thái này khi gọi phương thức prepare() và phương thức prepareAsync () khi đã được cung cấp callback OnPreparedListener.onPrepared ().

Ở trạng thái này, có thể start được MediaPlayer bằng cách gọi start(), MediaPlayer sẽ chuyển sang trạng thái **Started**, hoặc có thể chuyển sang trạng thái **Stop** bằng cách gọi method stop().

Sử dụng method seekTo() trong trạng thái này không làm ảnh hưởng đến trạng thái của MediaPlayer.

##### 1.2.6. Started State

Trạng thái khi mà media đang được chạy, chuyển đến trạng thái này bằng cách gọi start() có thể từ các trạng thái **Prepare**, **Pause**, **PlaybackComplete**.

Lúc này có thể sử dụng pause() để tạm dừng nhạc và chuyển sang trạng thái **Pause** hoặc stop() để dừng hẳn nhạc và chuyển sang trạng thái **Stop** của MediaPlayer. Gọi isPlaying() để biết được media có đang được chạy không.

Khi đang ở trong trạng thái này, khi gọi phương thức seekTo() hay start(), MediaPlayer vẫn giữ nguyên trạng thái. Trong trạng thái này khi mà trước đó dã setOnBufferingUpdateListener(OnBufferingUpdateListener), chúng ta sẽ được hệ thống cung cấp callback OnBufferingUpdateListener.onBufferingUpdate() để theo dõi trạng thái buffer trong suốt quá trình stream audio/video.

Khi mà audio/video được phát hết, mà được setLooping(true) thì MediaPlayer sẽ được duy trì ở trạng thái **Started**.

##### 1.2.7. Pause state

Khi media đang chạy, gọi hàm pause() để dừng media khi đang phát nhạc. Lúc này MediaPlayer sẽ chuyển sang trạng thái **Pause**.

Tại trạng thái này, có thể chuyển sang trạng thái **Started** để play lại media bằng cách gọi start() hoặc có thể chuyển sang trạng thái **Stop** bằng cách gọi method stop(). Thường dùng cho nút play/pause gọi hàm này để thực hiện việc quản lý media.

Chúng ta có thể gọi seekto()/pause() khi đang ở trong **Pause State** mà không làm thay đổi trạng thái của MediaPlayer.

##### 1.2.8. Stop State

Rơi vào trạng thái này khi gọi phương thức stop(), phương thức stop() có thể được gọi từ các trạng thái như **Prepared**, **Pause**, **PlaybackComplete** khi ở trong trạng thái này. MediaPlayer phải được đưa về trạng thái **Prepared**hoặc **Preparing** trước khi muốn quay lại trạng thái **Started** bằng cách sử dụng prepared() hoặc prepareAsync().

Khi MediaPlayer đã ở trong trạng thái này rồi thì khi gọi stop() không làm thay đổi trạng thái của MediaPlayer.

##### 1.2.9. PlaybackComplete State

Trạng thái mà chạy xong media sẽ nhảy vào đây, có thể lắng nghe sự kiện OnCompletionListener của nó để có thể thực hiện chuyển bài hoặc là vẫn play bài đó với các trường hợp của loop media. Tại đây có thể gọi start() để play lại audio/media

Tại trạng thái này có thể gọi phương thức stop() để chuyển sang trạng thái **Stop** hay start() để quay lại trạng thái **Started**.

Khi ở trạng thái này, nếu gọi method seekTo() không làm thay đổi trạng thái.

Khi setLooping(false) thì sẽ được cung cấp callback OnCompletion.onCompletion() nếu OnCompletionListener đã được đăng kí trước đó, MediaPlayer sẽ rơi vào trạng thái **PlaybackComplete**.

##### 1.2.10. Error State

Trong quá trình thao tác với MediaPlayer nếu xảy ra lỗi, có thể là khi setDataSource() sai, định dạng audio, video không đúng, kích thước ảnh quá lớn,.. sẽ rơi vào trong trạng thái này.

Khi rơi vào trạng thái **Error**, chúng ta sẽ được hệ thống cung cấp callback OnErrorListener.onError() nếu OnErrorListener đã được đăng kí từ trước.

Ở trạng thái này chúng ta có thể quay lại trạng thái **Idle** khi gọi method reset()

#### 1.3. Các Callback trong MediaPlayer.

Những callback trả về cho chúng ta những trạng thái và giá trị của Media Player:

setOnCompletionListener(OnCompletionListener): Lắng nghe sự kiện khi play xong một media, sử dụng để có thể quy định việc tiếp theo nó sẽ làm là play lại hay play bài hát khác.

* setOnVideoSizeChangedListener(OnVideoSizeChangedListener): Lắng nghe sự kiện thay đổi size của media trong lúc play video.

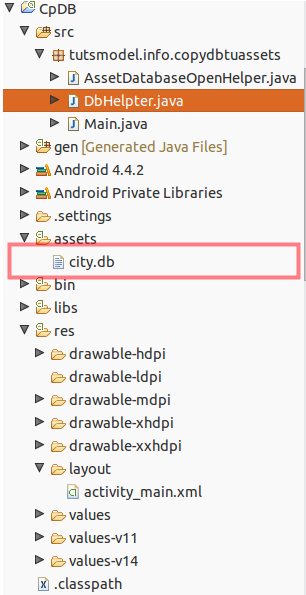
setOnPreparedListener(OnPreparedListener): Lắng nghe sự kiện Prepared xong của media để chuẩn bị cho việc play.

* setOnBufferingUpdateListener(OnBufferingUpdateListener): Lắng nghe sự kiện thay đổi của bộ đệm khi play media online
* setOnSeekCompleteListener(OnSeekCompleteListener): Lắng nghe sự kiện khi di chuyển seekbar hỗ trợ bởi MediaPlayer.
* setOnInfoListener(OnInfoListener): Lắng nghe sự kiện khi có thông tin hoặc cảnh báo.
* setOnErrorListener(OnErrorListener): Lắng nghe sự kiện nếu xảy ra lỗi khi play media.

## CHƯƠNG 6. BỘ QUẢN LÝ ASSET, SHAREPREFERENCE VÀ SQL

### 1.QUẢN LÝASSET

-  Asset là thư mục chứa dữ liệu đầu vào. Ví dụ: âm thanh, hình ảnh, tập tin CSDL hoặc  tập tin có đuôi khác ... Những tập tin này sẽ không được biên dịch khi ứng dụng được đóng gói.



*Hình 9. Thư mục asset*

-  Truy xuất đối tượng quản lý: getAssets() – AssetManager

-  Các phương thức xử lý chính:

* list(String path) – String[]
* open(String fileName) – InputStream
* open(String fileName, int mode) – InputStream
* close()

-  Các chế độ mở Asset

* ACCESS\_BUFFER
* ACCESS\_RANDOM
* ACCESS\_STREAMING
* ACCESS\_UNKNOWN

-Cú pháp khởi tạo AssetManager:

AssetManager assetManager=Context.getAssets();

### 2.SharedPreference

#### 1.Giới thiệu

-SharedPreference là một lớp cho phép lưu trữ và nhận dữ liệu **key value**

với các kiểu dữ liệu nguyên thuỷ như integer, string, boolean, float, long.

-Preference gần giống với Bundle nhưng nó có thể lưu trữ dữ liệu còn Bundle thì không.

-  Shared Preferences là một dạng lưu trữ nhanh bằng file XML.

-  Dữ liệu vẫn được bảo toàn ngay cả khi ứng dụng bị đóng hoàn toàn.

-  Các dữ liệu được lưu trữ và truy xuất qua từng phiên (Session) tương tác của ứng dụng  với người dùng.

-  Lưu trữ và truy xuất trạng thái ứng dụng

#### 2. Lưu trữ

-  Sử dụng đối tượng của lớp SharedPreferences thông qua phương thức:  GetSharedPreferences(String, int);

-  Tạo đối tượng SharedPreferences: sử dụng phương thức getSharedPreferences().

o Trong đó:

PREFS\_NAME: tên của sharedPreference cần tạo

Mode: kiểu ghi dữ liệu

MODE\_PRIVATE: mode ghi mặc định, chỉ có duy nhất ứng dụng tạo ra  tập tin được phép truy cập vào.

MODE\_WORLD\_READABLE: cho phép các ứng dụng khác truy cập  vào.

MODE\_WORLD\_WRITEABLE: cho phép các ứng dụng khác truy cập  và sửa đổi.

MODE\_MULTI\_PROCESS: cho phép nhiều tiến trình trên ứng dụng  cùng truy xuất vào cùng một thời điểm.

  -  Các giá trị lưu trữ được truy xuất thông qua phương thức get<KDL>:

o getBoolean(Key, DefValue)

o getFloat(Key, DefValue)

o getInt(Key, DefValue)

o getLong(Key, DefValue)

o getString(Key, DefValue)

#### 3.Lấy một SharePreference

Lấy một SharedPreference mặc định trong ứng dụng bằng cách gọi hàm

SharedPreference sharedPreference = PreferenceManager.getDefaultSharedPreference(this)

* Tạo một SharedPreference

SharedPreference shared = getSharedPreference(“MySharedPreference”, MODE\_PRIVATE);

* Lấy một SharedPreference mặc định của một Activity

|  |
| --- |
| SharedPreference shared = getPreferences(MODE\_PRIVATE); |

* Các chế độ SharedPreference:
* ***MODE\_PRIVATE***: chỉ ứng dụng có thể truy nhập vào.
* ***MODE\_WORLD\_READABLE***: các ứng dụng khác có thể đọc được nội dung nhưng không thể sửa hoặc thêm.
* ***MODE\_WORLD\_WRITABLE***: Ứng dụng khác có thể đọc/ghi nội dung.
* ***MODE\_MULTI\_PROCESS***: Nhiều process có thể cùng chỉnh sửa SharedPreference.

### 3.SQL

#### 1.Khái niệm

SQLITE là một loại CSDL quan hệ, nhỏ gọn, nó là hệ CSDL được sử dụng rất rộng rãi trên nhiều nền tảng (Mobile, Destop, Webserver ...), nó có các ưu điểm có thể kế ra như: Là hệ CSDL SQL nhúng, sử dụng ngay mà không cần cấu hình, không cần có một Server SQL riêng (khác với MySQL, MS SQL Server ...) ...

Mặc định mỗi ứng dụng sẽ được cấp phát một thư mục cho việc lưu trữ cơ sở dữ liệu và nó chỉ có thể được dùng bởi ứng dụng đó. Nếu muốn chia sẻ dữ liệu dùng chung giữa các ứng dụng ta có thể sử dụng Content Provider.

#### 2.XÂY DỰNG CƠ SỞ DỮ LIỆU VỚI SQLITE

 Tạo cơ sở dữ liệu thông qua lớp SQLiteOpenHelper.

o SQLiteOpenHelper là một lớp ảo, SQLiteOpenHelper giúp tạo các cơ sở dữ liệu dùng SQLite (vì SQLite không hỗ trợ các phương thức khởi tạo cơ sở dữ liệu). Vậy làm sao để sử dụng được SQLiteOpenHelper? Vì SQLiteOpenHelper là một lớp ảo nên ta cần khai báo một lớp khác kế thừa lớp này.

o SQLiteOpenHelper thực hiện các phương thức cần thiết cho phép khởi tạo, nâng cấp cơ sở dữ liệu. Tạo đối tượng để truy cấp cơ sở dữ liệu (Read và Write).

o SQLiteDatabase cung cấp phương thức insert(), update(), delete(), hoặc execSQL() cho phép thực hiện truy xuất dữ liệu.

Override phương thức onCreate() để tạo cơ sở dữ liệu.

Override phương thức onUpgrade() để nâng cấp cơ sở dữ liệu.

\_ Lớp SQLiteOpenHelper cung cấp 2 phương thức getReadableDatabase() và getWriteableDatabase() để trả về đối tượng SQLiteDatabase.

## CHƯƠNG 7. BỘ NHỚ THIẾT BỊ VÀ TOOLBAR , ACTIONBAR

### 1. BỘ NHỚ THIẾT BỊ

#### 1.1. Bộ nhớ trong

Là nơi lưu trữ các dữ liệu cá nhân của từng ứng dụng, mà các dữ liệu này được lưu trữ và sử dụng cho riêng ứng dụng đó. Các ứng dụng khác không thể truy cập vào được. Thông thường khi ứng dụng bị gỡ bỏ khỏi thiết bị.

Đường dẫn các tập tin được lưu trữ:

|  |
| --- |
| data/data/<pakage\_name\_app>/files |

Phương thức truy xuất:

|  |
| --- |
| getFilesDir - File |

* Tạo và lưu trữ file vào bộ lưu trữ trong
* Gọi openFileOutput() với tên file và chế độ truy cập file.
* Ghi lên file sử dụng phương thức write().
* Đóng stream với phương thức close().
* Đọc file từ bộ lưu trữ trong.
* Gọi openFileInput() với tên file cần đọc.
* Đọc nội dụng từ file sử dụng phương thức read().
* Đóng stream với phương thức close().

#### 1.2. Bộ nhớ ngoài

Là nơi lưu trữ dữ liệu ngoài của **Android**, các file dữ liệu lưu trữ mà bạn lưu trữ tại đây không được hệ thống áp dụng bảo mật.

Thông thường có 2 loại lưu trữ ngoài (external storage).

1.Lưu trữ ngoài cố định: Thường được hiểu là ổ cứng của điện thoại.

2.Lưu trữ ngoài lưu động (Removeable Storage): Chẳng hạn SD Card

* Để đọc ghi dữ liệu trên bộ lưu trữ ngoài yêu cầu bạn phải cấu hình vào file AndroidManifest.xml

|  |
| --- |
| <uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>  <uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE"/> |

* Phương thức truy xuất:

o getExternalFilesDir(String) – File

o getExternalFilesDirs(String) – File[] (API 19 - KITKAT)

o ContextCompat.getExternalFilesDirs(String) – File[] (Support Library)

* Các phương thức để lưu trữ dữ liệu thông qua phương thức getExternalStorageState trong đối tượng Environment.

1. ***getExternalStorageState* ()**

Nên luôn kiểm tra tính sẵn sàng cho phép truy xuất của bộ nhớ ngoài thông qua hàm *getExternalStorageState()* – trả về trạng thái hiện tại của bộ nhớ ngoài.

1. **getExternalFileDir(String type)**

Đối tượng File có đường dẫn thư mục nội bộ bên trong thư mục dữ liệu của ứng dụng đặt trong bộ nhớ ngoài. Nếu bạn thông số truyền vào là Null, thì kết quả trả về là thư mục gốc của thư mục dữ liệu của ứng dụng trong bộ nhớ ngoài.

1. **Environment.getExternalStoragePublicDirectory()**

Đối tượng File có đường dẫn thư mục trong bộ nhớ ngoài dùng chung, vì mục đích chia sẻ, các tập tin này sẽ không bị xóa đi khi gỡ ứng dụng ra khỏi thiết bị. Ví dụ như, ứng dụng bạn dùng camera để chụp ảnh, và những file ảnh này nên được giữ lại cho người dùng dù ứng dụng của bạn đã được gỡ ra khỏi thiết bị rồi.

* Các thông số được trả về bao gồm:
* **DIRECTORY\_ALARMS** — Các file âm thanh được dung làm báo thức
* **DIRECTORY\_DCIM**— Hình ảnh và video được chụp hoặc quay bằng thiết bị
* **DIRECTORY\_DOWNLOADS** — chứa các file được tải về.
* **DIRECTORY\_MOVIES**— chứa các file video.
* **DIRECTORY\_MUSIC** — Chứa các file âm thanh cho chương trình nghe nhạc.
* **DIRECTORY\_NOTIFICATIONS** — Các file âm thanh cho các thông báo tin nhắn.
* **DIRECTORY\_PICTURES** — Chứa các file hình ảnh.
* **DIRECTORY\_RINGTONES** — Các file âm thanh được dung làm nhạc chuông.

### 2.TOOLBAR

Toolbar được giới thiểu từ API 21, nó như là mở rộng của ActionBar, giúp cho việc tùy biến dễ dàng hơn ActionBar. Trong thư viện AppCompat.Toolbar được triển khai ở lớp android.support.v7.widget.Toolbar.

Toolbar được xem là bản cải tiến của ActionBar và được kết thừa từ ViewGroup, vẫn có thể dùng một số thuộc tính hay phương thức của ActionBar như thêm Logo, Labels, Navigation items...

* Toolbar cùng lúc có thể chứa các yếu tố sau:

-  A Navigation Button. Nó có thể là Up arrow, navigation menu toggle, close, collapse, done hay một số hình ảnh mà bạn tự thiết kế.

-  A branded logo image. Có thể thiết lập chiều cao cho logo, thiết lập khoảng cách tùy ý.

-  A title and subtitle. Thiết lập tiêu đề cho Toolbar, nhưng có thể nó sẽ bị che lấp khi ta  thiết lập logo quá lớn.

-  An action menu. Cũng giống như thanh ActionBar, “menu” sẽ mằm ở vị trí cuối của Toolbar.

### 3.ACTIONBAR

Được hiểu là App Bar theo thiết kế Meterial Design, nó chứa các thành phần với mục đích điều hướng ứng dụng như:

* Icon ứng dụng
* Nút mũi tên để quay lại cửa sổ cha
* Tiêu đề
* Những nút bấm chính cho Activity

- Các thành phần điều hướng

- Action Bar có chức năng:

o Điều hướng giao diện ứng dụng.

o Hiển thị các thao tác trên toàn bộ hệ thống ứng dụng hoặc thao tác tại màn hình

hiển thị (tìm kiếm, chỉnh sửa, xóa...).

o Thao tác chuyển đến các giao diện khác nhau trong cùng một màn hình hiển thị (tabhost, list navigation...).

* Ví dụ đổi tiêu đề ActionBar:

ActionBar actionbar=getSupportActionBar();

Actionbar.setTitle(“Tiêu đề Activity”);

String title=actionbar.getTitle().toString();

Actionbar.hide();

# PHẦN 2. THỰC HÀNH

## 1.Giới thiệu đề tài

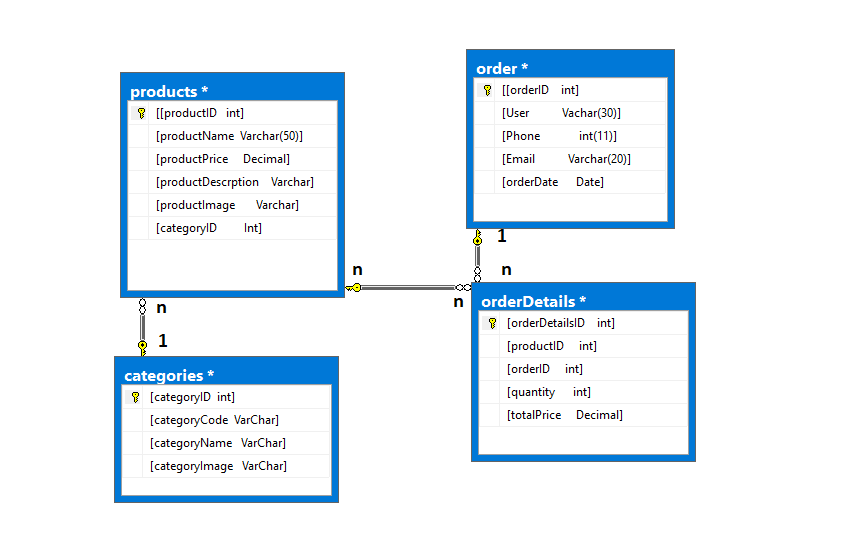
* Đề tài shop bán hàng (**Shop thể thao**) là một ứng dụng chạy và sử dụng trên hệ điều hành Android.

- Phần mềm chạy trên điện thoại được phát triển trên nền tảng SDK.

- Phần mềm cho phép người sử dụng có thể mua và đặt hàng các sản phẩm trên ứng dụng.

* App shop thể thao chạy trên nền Android nên khi người dùng tải ứng dụng phải hỗ trợ Android 5.0 trở lên. Người dùng có thể chọn xem thông tin sản phẩm và đăng ký thông tin khách hàng dữ liệu sẽ được lưu trong Sqlite.
* Ứng dụng có Navigation Drawer có các doanh mục:
* Trang chủ
* Sản phẩm
* Thông tin đặt hàng
* Địa chỉ
* Shop thể thao có 3 loại sản phẩm là: Quần, Áo, Giày.

## 2.Mô hình quan hệ dữ liệu

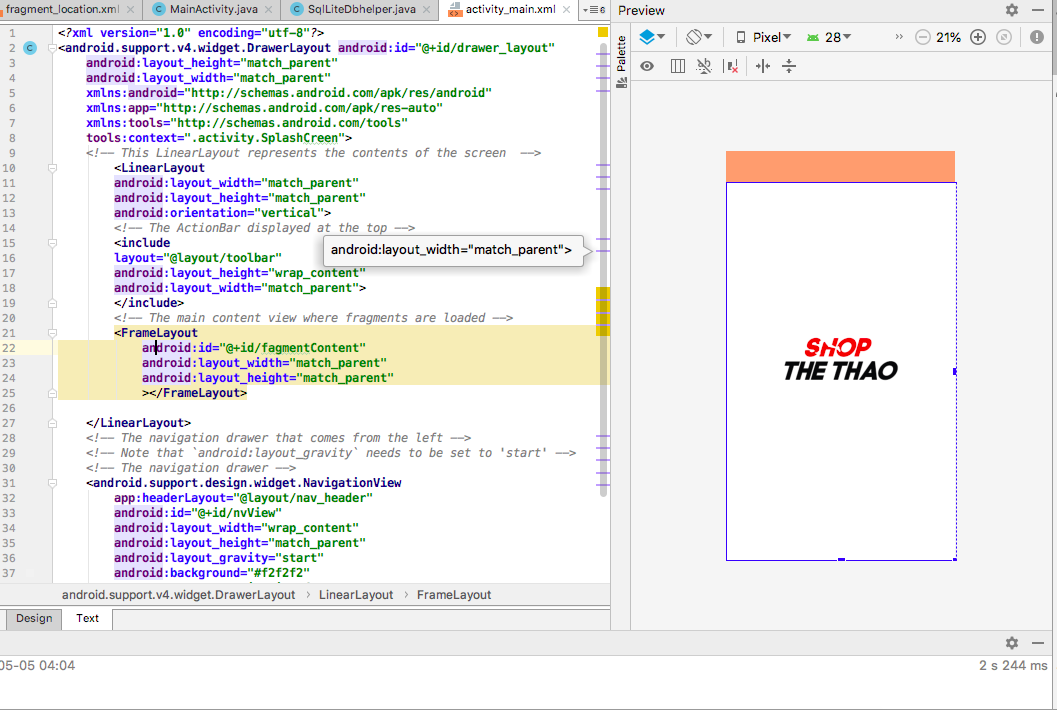


*Hình 10. Mô hình quan hệ dữ liệu*

## 3.Cài đặt

### 3.1. Giao diện trên Android studio

Thiết kế giao diện bằng cách tạo file .xml



*Hình 11. Tạo layout trên Android Studio*

* Sau khi tạo giao diện xong ta code trong file **java**.

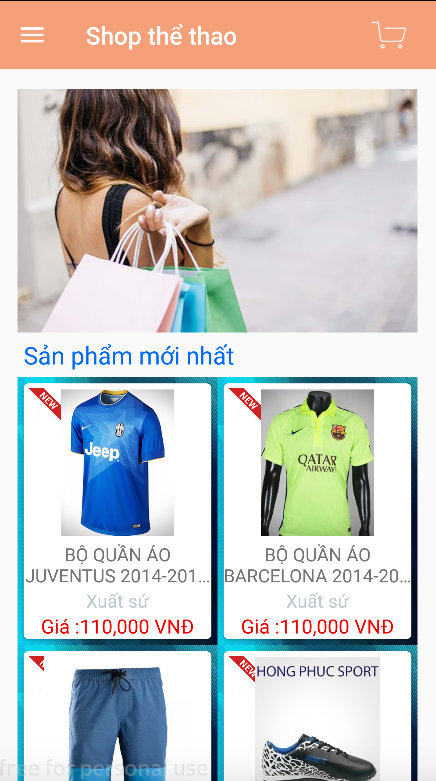
### 3.2.Giao diện và tác vụ trên điện thoại

Khi ứng dụng được khởi động thì màn hình SplashCreen sẽ hiện ra



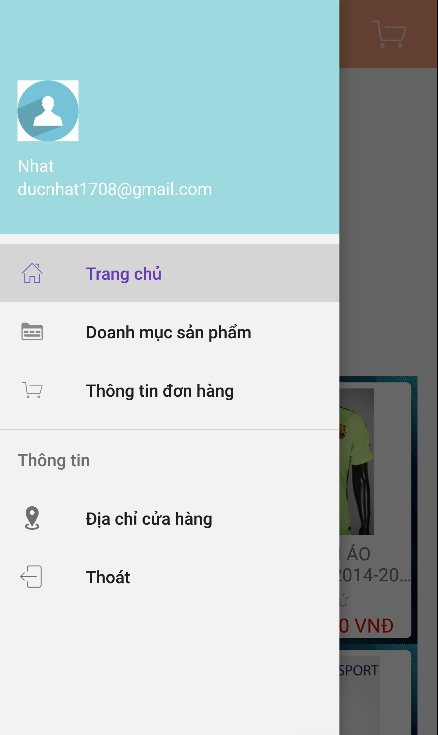
*Hình 12. Giao diện SplashCreen*

* Sau đó khoảng 3 giây app sẽ vô màn hình chính



*Hình 12. Giao diện trang chủ*

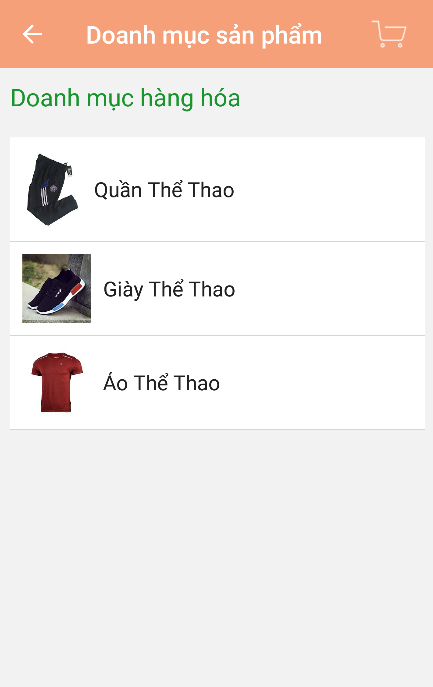
\_Nếu người dùng chọn vào menu thì giao diện Navigation Drawer sẽ hiện ra chèn vào trang chủ



*Hình 13. Giao diện menu*

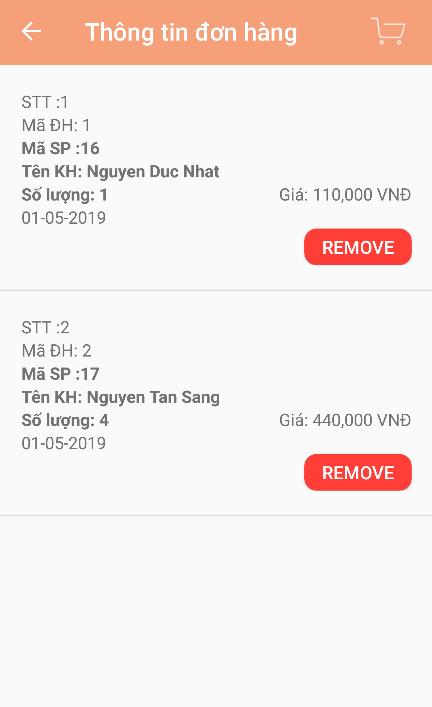
\_Các giao diện trong menu

1.Giao diện Doanh mục sản phẩm



*Hình 14. Giao diện doanh mục sản phẩm*

2.Giao diện thông tin đơn hàng



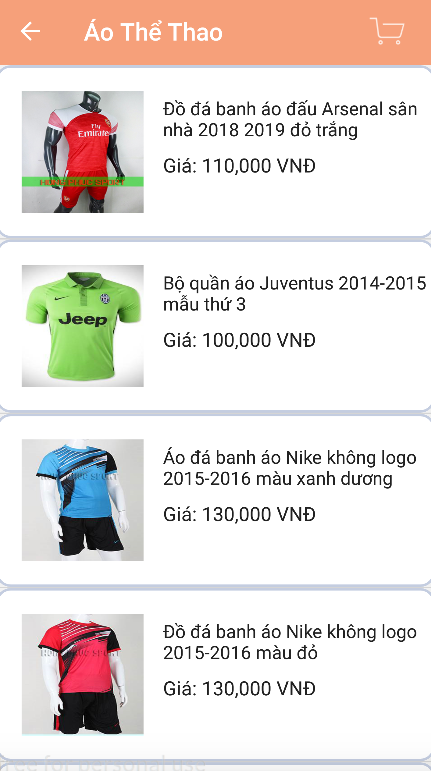
*Hình 15. Giao diện thông tin đơn hàng*

3.Giao diện Địa chỉ cửa hàng



*Hình 16. Màn hình địa chỉ*

\_Khi người dùng chọn doanh mục thì list sản phẩm sẽ show ra



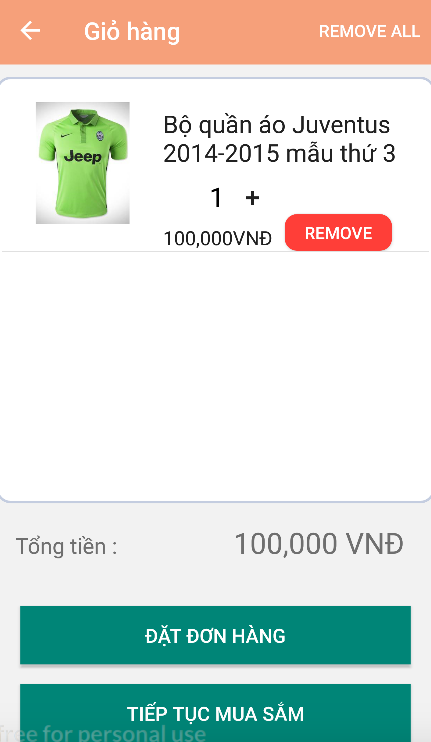
*Hình 17. Giao diện sản phẩm*

\_Sau khi đã chọn sản phẩm người dùng có thể xem chi tiết sản phẩm



*Hình 18. Giao diện chi tiết sản phẩm*

\_Khi người dùng nhấn vào button add to cart thì sản phẩm sẽ vào giỏ hàng với số lượng là một



*Hình 19. Màn hình giỏ hàng*

\_Tại đây người dùng có thể thêm số lượng (số lượng phải >1 và <= 10) nếu người dùng nhấn quá hạn số lượng thì buton cộng, trừ sẽ ẩn đi và có thể **Remove** sản phẩm

\_Khi người dùng quay về màn hình chí mà không thanh toán thì toolbar giỏ hàng sẽ cập nhật theo số lượng sản phẩm vd: Đơn hàng là 2 thì icon sẽ hiện tương ứng



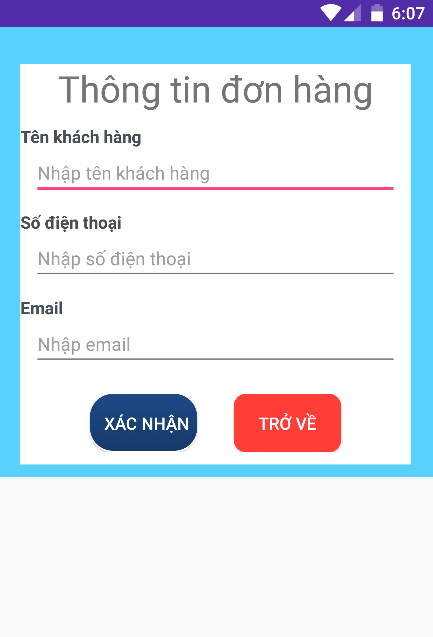
*Hình 20. Update giỏ hàng*

\_Nếu thanh toán mà thông tin giỏ hàng rỗng thì sẽ hiện thông báo “Giõ hàng rỗng không thể thanh toán”



*Hình 21. Giỏ hàng rỗng*

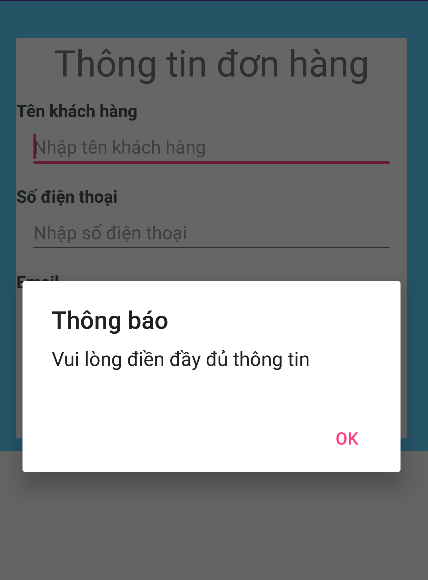
\_Ngược lại, màn hình thông tin khách hàng sẽ hiên ra



*Hình 22. Thông tin khách hàng*

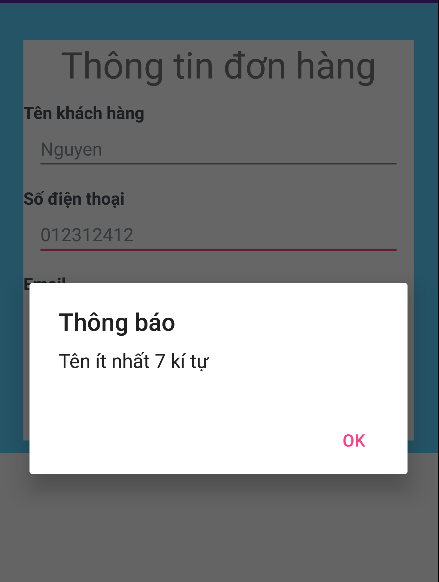
* Xử lý bắt lỗi editText trong nhập thông tin đơn hàng

1.Chưa điền thông tin AlertAdiolog sẽ thông báo



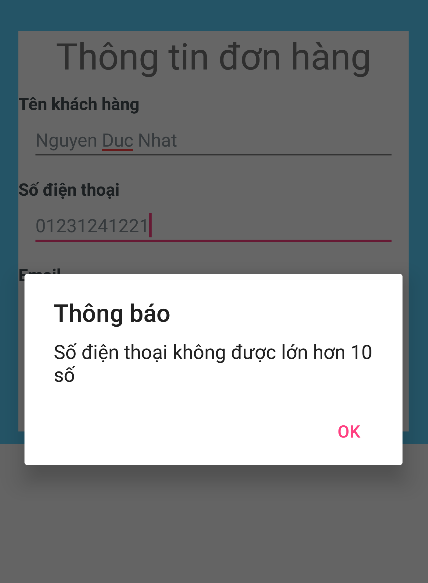
*Hình 23. AlertAdiolog Chưa điền thông tin*

2.Tên Tên khách ít nhất phải 7 kí tự



*Hình 24. AlertAdiolog tên khách hàng ít nhất phải 7 kí tự*

3.Số điện thoại không vượt quá 10 số



*Hình 25. AlertAdiolog kiểm tra số điện thoại*

\_Nếu hợp lệ thì sẽ hiên FancyToast “**Thêm đơn hàng thành công**” và trờ về màn hình chính



*Hình 26. Thêm đơn hàng thành công*

# TÀI LIỆU THAM KHẢO

1. *Bài giảng môn Android của thầy Giang Hào Côn* Khoa CNTT, Trường ĐH. Nguyễn Tất Thành.

2.Tải liệu Khoa Phạm

3.Andoid document: <https://developer.android.com/reference/org/w3c/dom/Document>