Shape, square

Description automatically generated

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

Logo

Description automatically generated

**BÁO CÁO MÔN THỰC TẬP CƠ SỞ**

**ĐỀ TÀI: APP NGHE NHẠC (MUZIKO)**

|  |  |
| --- | --- |
| **Giảng viên:** | **Ths. Nguyễn Hoàng Anh** |
| **Họ tên SV:** | **Kiều Đức Long** |
| **MSV:** | **B19DCCN389** |
| **Lớp:** | **D19CNMP6** |

Hà Nội, tháng 5 năm 2022

**MỤC LỤC**

[**LỜI CẢM ƠN** 4](#_Toc104240317)

[**I.** **Đặt vấn đề** 5](#_Toc104240318)

[**II.** **Giải quyết vấn đề** 6](#_Toc104240319)

[**1.** **Mục đích – phạm vi** 6](#_Toc104240320)

[**2.** **Sơ đồ hệ thống** 6](#_Toc104240321)

[**3.** **Sơ đồ CSDL** 6](#_Toc104240322)

[**4.** **Thực hiện dự án** 7](#_Toc104240323)

[*4.1.1.* *Tổ chức dự án* 7](#_Toc104240324)

[*4.1.2.* *Front End* 7](#_Toc104240325)

[*4.1.3.* *Back End* 7](#_Toc104240326)

[**5.** **Giới thiệu mô hình MVVM** 7](#_Toc104240327)

[**6.** **Giới thiệu Git** 9](#_Toc104240328)

[**7.** **Giới thiệu Android Studio** 10](#_Toc104240329)

[**8.** **Giới thiệu Kotlin** 21](#_Toc104240330)

[**9.** **Giới thiệu Room Database** 23](#_Toc104240331)

[**10.** **Bắt đầu xây dựng ứng dụng** 23](#_Toc104240332)

[- Thiết kế giao diện ứng dụng: 23](#_Toc104240333)

[- Tạo ứng dụng bằng Android Studio. 23](#_Toc104240334)

[- Xây dựng database theo sơ đồ csdl đã nêu trên: 23](#_Toc104240335)

[- Lấy file âm thanh từ thiết bị: 26](#_Toc104240336)

[- Tạo các class helper, utils: 29](#_Toc104240337)

[- Tạo BottomNavigation: 30](#_Toc104240338)

[- Tạo các ViewModel, Repository theo MVVM 32](#_Toc104240339)

[- Tạo các fragment (view) để lấy data từ viewmodel và hiển thị lên: 33](#_Toc104240340)

[- Tạo navigation để chuyển giữa các fragment: 33](#_Toc104240341)

[- Tạo Service, BroadcastReceive để điều khiển nhạc bằng thông báo: 34](#_Toc104240342)

[**III.** **Kết quả** 38](#_Toc104240343)

[**1.** **SplashScreen** 38](#_Toc104240344)

[**2.** **Màn hình chính** 39](#_Toc104240345)

[**3.** **SongFragment** 40](#_Toc104240346)

[**4.** **SearchFragment** 41](#_Toc104240347)

[**5.** **FavoriteFragment** 42](#_Toc104240348)

[**6.** **Playlist** 43](#_Toc104240349)

[**7.** **Tạo mới playlist** 44](#_Toc104240350)

[**8.** **Panel nghe nhạc** 45](#_Toc104240351)

[**TỔNG KẾT** 47](#_Toc104240352)

[**TÀI LIỆU THAM KHẢO** 48](#_Toc104240353)

# **LỜI CẢM ƠN**

Em xin gửi lòng biết ơn sâu sắc tới thầy Nguyễn Hoàng Anh – người đã tận tình hướng dẫn, hỗ trợ về chuyên môn và theo dõi sát sao em trong cả quá trình. Sau khi kết thúc môn học này, em đã có nhiều hiểu biết hơn cũng như đã làm được một sản phẩm nhỏ nhưng vẫn không thể tránh khỏi các sai sót. Em rất mong sẽ tiếp tục nhận được những ý kiến đóng góp quý báu của thầy để em có thể hoàn thiện hơn trong tương lai.

Sau cùng, em xin kính chúc thầy luôn dồi dào sức khỏe và công tác tốt!

1. **Đặt vấn đề**

Hiện nay, với sự phát triển mạnh mẽ của internet và những ứng dụng của nó trong đời sống. Máy tính, điện thoại thông minh không còn là một phương tiện lạ lẫm đối với mọi người mà nó đã dần trở thành một công cụ làm việc và giải trí thông dụng và hữu ích của chúng ta ở mọi nơi nhờ có kết nối internet. Do đó việc xây dựng các ứng dụng cho điện thoại di động đang là một nghành công nghiệp mới đầy tiềm năng và hứa hẹn sự phát triển vượt bậc.

Phần mềm, ứng dụng cho điện thoại di động hiện nay rất đa dạng và phong phú trên các hệ điều hành di động. Các hệ điều hành như Android, IOS, Hybrid,… đã rất phát triển trên thị trường truyền thông di động.

Trong nhiều năm trở lại đây, hệ điều hành android ra đời với sự kế thừa những ưu việt của các hệ điều hành trước và sự kết hợp của nhiều công nghệ tiên tiến nhất hiện nay. Android đã nhanh chóng trở thành đối thủ cạnh tranh mạnh mẽ với các hệ điều hành trước đó và đang là hệ điều hành được ưa chuông nhất.

Với sự phát triển nhanh chóng của xã hội nhu cầu giải trí thông qua điện thoại di động ngày càng phổ biến, đặc biệt là việc nghe nhạc thì nhu cầu có 1 ứng dụng quản lý và nghe nhạc trở nên rất cần thiết.

1. **Giải quyết vấn đề**

* Từ vấn đề trên, app nghe nhạc Muziko ra đời.

1. **Mục đích – phạm vi**

* **Mục đích:**
* Ứng dụng cho phép nghe nhạc trên nền tảng android, tạo playlist, nghe theo album, ca sĩ, yêu thích.
* **Phạm vi:**
* Là ứng dụng chạy trên điện thoại cá nhân, sử dụng độc lập.

1. **Diagram

   Description automatically generatedSơ đồ hệ thống**
2. **Sơ đồ CSDL**
3. **Table

   Description automatically generatedThực hiện dự án**
   * 1. *Tổ chức dự án*

* Tổ chức dự án theo mô hình MVVM
* Quản lý dự án bằng Git
  + 1. *Front End*
* Phát triển ứng dụng trên nền tảng Android
* Công cụ phát triển Android Studio
* Ngôn ngữ: Kotlin
  + 1. *Back End*
* Room Database

1. **Giới thiệu mô hình MVVM**

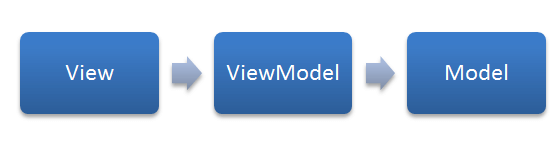
*Mô hình MVVM là gì ?*

MVVM không phải là framework hay thư viện, api… nó chỉ đơn thuần là hướng dẫn bạn định nghĩa cấu trúc ứng dụng của bạn. MVVM được phát triển dựa trên kiến trúc MVP.

Mô hình mvvm cho phép tách biệt dữ liệu (Model), mã thực thi (logic hay ViewModel) và giao diện người dùng (View).

Trong các mô hình truyền thống, chúng ta thường xử lý sự kiện Click và viết mã thực thi trực tiếp ở trên một Button nhưng với mô hình MVVM không cho phép làm điều này.

Trong mô hình MVVM, các điều khiển(control) như Button, ListView, SearchBar, v.v. không thể kết buộc trực tiếp đến dữ liệu mà phải thông qua thuộc tính Command – là một thuộc tính kiểu ICommand.



*MVVM được hiểu như thế nào ?*

***View:***

* Thành phần giao diện của ứng dụng. Tương tự như ở trong mô hình MVC, View là thành phần duy nhất mà người dùng có thể tương tác được trong chương trình, nó chính là thành phần mô tả dữ liệu.
* Một điểm khác biệt so với các mô hình khác là View trong mô hình này tích cực hơn. Nó có khả năng thực hiện các hành vi và phản hồi lại người dùng thông qua tính năng là: binding, command.

***Model:***

* Cũng tương tự như trong mô hình MVC. Model là các đối tượng giúp truy xuất và thao tác trên dữ liệu thực sự.

***ViewModel:***

* Lớp trung gian giữa View và Model. ViewModel có thể được xem là thành phần thay thế cho Controller trong mô hình MVC. Nó chứa các mã lệnh cần thiết để thực hiện data binding, command.

*Ưu nhược điểm của MVVM: Model – View – ViewModel?*

***Ưu điểm:***

* Thực hiện Unit testing bây giờ sẽ rất dễ dàng, vì bạn thực sự không phụ thuộc vào view.
* MVVM sẽ tạo sự tương tác hiệu quả giữa designer và developer.
* Tăng khả năng sử dụng lại các thành phần hay việc thay đổi giao diện chương trình mà không cần phải viết lại code quá nhiều.
* Phát triển ứng dụng nhanh, đơn giản, dễ nâng cấp, bảo trì…

***Nhược điểm :***

* Khả năng duy trì khi view có thể gán cả biến và biểu thức, các logic không liên quan sẽ tăng dần theo thời gian, ảnh hưởng đến việc thêm code vào XML.
* Đối với dự án nhỏ việc áp dụng mô hình MVVM gây cồng kềnh, tốn thời gian trong quá trình phát triển. Tốn thời gian trung chuyển dữ liệu của các thành phần.
* Đối với dự án lớn hơn, nó gây khó khăn và mất thời gian để thiết kế các ViewModel.
* Việc liên kết dữ liệu cho tất cả các thành phần gây khó khăn trong việc debug khi cơ sở dữ liệu phức tạp.

1. **Giới thiệu Git**

****

Git là một trong những hệ thống kiểm soát phiên bản dùng để theo dõi các thay đổi trong tập tin, nó được dùng để thay đổi trong các tập tin bất kỳ và nó được sử dụng chính trong các lĩnh vực phát triển kỹ thuật phần mềm.

Git được hiểu là hệ thống điều khiển về mặt cơ bản nó là một trình theo dõi nội dung vì thế Git có thể sử dụng được để lưu trữ nội dung chủ yếu sử dụng để lưu trữ các mã do tính năng khác mà nó cung cấp.

Git được coi là một hệ thống điều khiển phân tán có tốc độ xử lý nhanh những vẫn đảm bảo được toàn vẹn dữ liệu và hỗ trợ hiệu quả cao cho các workflow phân tán. Giống như các hệ thống quản lý phiên bản phân tán khác, mọi thư mục Git trên máy tính đều là một kho chứa lịch sử . Gồm có kho lưu trữ từ xa trong máy chủ và một kho lưu trữ cục bộ được lưu trữ trong máy tính của nhà phát triển.

*Hệ thống kiểm soát Git hoạt động như thế nào?*

Trong thực tế làm việc thì các nhà phát triển thường thực hiện song song các dự án. Để tránh sự xung đột giữa các mã giữa các nhà phát triển ta cần một hệ thống kiểm soát. Để làm được điều này hầu hết các nhà phát triển chon Git là hệ thống kiểm soát các mã để tránh sự xung đột các mã của nhà phát triển.

Bên cạnh đó, các yêu cầu trong các dự án thường xuyên được thay đổi. Có những khi bạn đã xóa một đoạn code nhưng lúc sau phát hiện ra đoạn code đó vẫn cần dùng. Thế nên cần có một kiểm soát để nhà phát triển có thể quay lại phiên bản cũ của mã. Chính vì vậy Git là một trong những lựa chọn để sử dụng. Vì Git giúp người dùng lưu lại những phiên bản của những lần thay đổi mã nguồn.

Hầu như các hệ thống quản lý phiên bản tập chung tất cả mã nguồn và lịch sử thay đổi ở một chỗ máy chủ, thì với Git nó lại là một hệ thống quản lý phiên bản phân tán. Điều này giúp ích nhiều trong khi các dự án đang được chạy song song liên quan đến cùng một cơ sở mã.

1. **Giới thiệu Android Studio**

*Android Studio là gì ?*

Android Studio là môi trường phát triển tích hợp (IDE) chính thức để phát triển ứng dụng Android, dựa trên IntelliJ IDEA. Dựa trên các trình soạn thảo mã và công cụ phát triển mạnh mẽ của IntelliJ, Android Studio còn cung cấp thêm nhiều tính năng giúp bạn nâng cao năng suất khi xây dựng ứng dụng Android, chẳng hạn như:

* Một hệ thống xây dựng linh hoạt dựa trên Gradle
* Một trình mô phỏng nhanh và nhiều tính năng
* Một môi trường hợp nhất nơi bạn có thể phát triển cho mọi thiết bị Android
* Áp dụng thay đổi để đẩy mã và tài nguyên cần thay đổi vào ứng dụng đang chạy mà không cần khởi động lại ứng dụng
* Mã mẫu và tích hợp GitHub để giúp bạn xây dựng các tính năng ứng dụng phổ biến cũng như nhập mã mẫu
* Đa dạng khung và công cụ thử nghiệm
* Công cụ tìm lỗi mã nguồn (lint) để nắm bắt hiệu suất, khả năng hữu dụng, khả năng tương thích với phiên bản và các vấn đề khác
* Hỗ trợ C++ và NDK
* Tích hợp sẵn tính năng hỗ trợ Google Cloud Platform, giúp dễ dàng tích hợp Google Cloud Messaging và App Engine

*Lịch sử hình thành Android Studio là gì?*

Android Studio đã được công bố vào năm 2013 tại hội nghị Google I/O và được phát hành vào năm 2014 sau nhiều phiên bản khác nhau.

Trước đó, thì các nhà phát triển của Android thường sử dụng các công cụ như Eclipse IDE hoặc một IDE Java chung để hỗ trợ cho nhiều ngôn ngữ lập trình khác.

Android Studio giúp cho việc tạo ứng dụng dễ dàng hơn so với các phần mềm chuyên dụng. Với người mới, sẽ có rất nhiều thứ phải học và nhiều thông tin có sẵn. Thậm chí, chúng còn thông qua nhiều kênh chính thức hoặc có thể có lỗi khiến người dùng hoang mang.

*Cấu trúc dự án*

Mỗi dự án trong Android Studio chứa một hoặc nhiều mô-đun có tệp mã nguồn và tệp tài nguyên. Có các loại mô-đun sau:

* Mô-đun ứng dụng Android
* Mô-đun thư viện
* Mô-đun Google App Engine

Theo mặc định, Android Studio thể hiện các tệp dự án của bạn trong chế độ xem dự án Android, như trong hình 1. Chế độ xem này được sắp xếp theo mô-đun để bạn có thể truy cập nhanh vào các tệp nguồn chính của dự án.

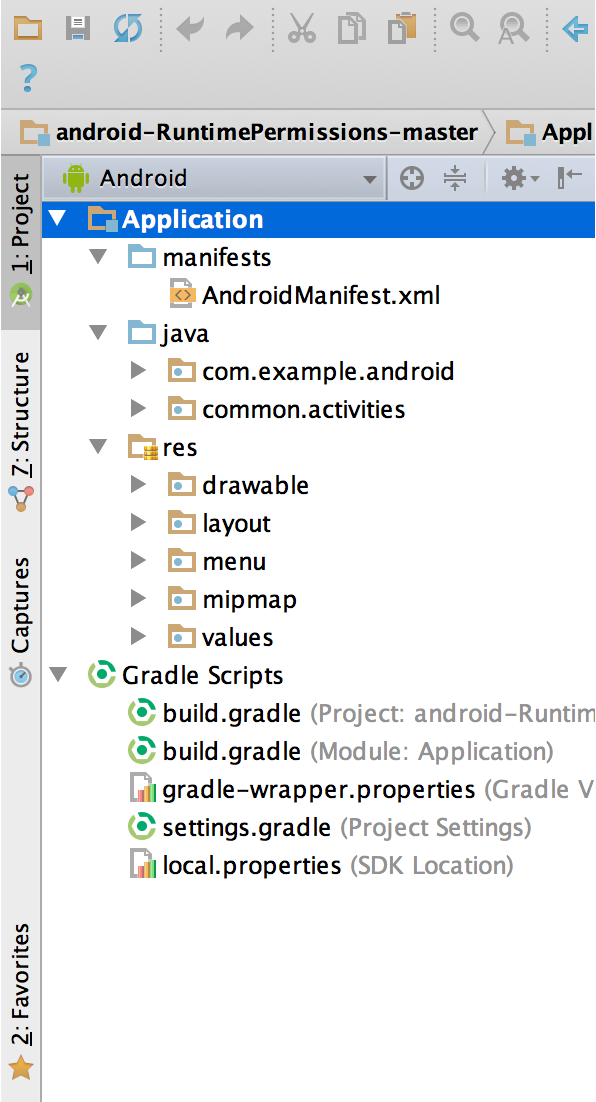
Bạn có thể thấy tất cả tệp bản dựng ở cấp cao nhất trong Gradle Scripts (Tập lệnh Gradle) và mỗi mô-đun ứng dụng chứa các thư mục sau:

* **manifests**: Chứa tệp AndroidManifest.xml.
* **java**: Chứa các tệp mã nguồn Java, bao gồm cả mã kiểm thử JUnit.
* **res**: Chứa mọi tài nguyên không phải mã, chẳng hạn như bố cục XML, chuỗi giao diện người dùng và hình ảnh bitmap.

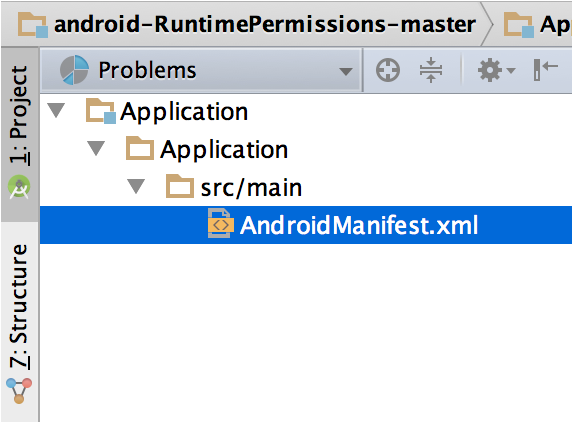
Cấu trúc dự án Android trên ổ đĩa khác với cách trình bày ở đây. Để xem cấu trúc tệp thực tế của dự án, hãy chọn Project (Dự án) trong trình đơn thả xuống Project (Dự án) (trong hình 1, mục này có tên Android).

Bạn cũng có thể tùy chỉnh chế độ xem các tệp dự án để có thể tập trung vào một số khía cạnh cụ thể trong việc phát triển ứng dụng. Ví dụ: khi chọn chế độ xem Problems (Vấn đề) cho dự án, bạn sẽ thấy đường liên kết đến các tệp nguồn chứa mọi lỗi cú pháp và mã hoá đã tìm thấy, chẳng hạn như thẻ đóng phần tử XML bị thiếu trong tệp bố cục.

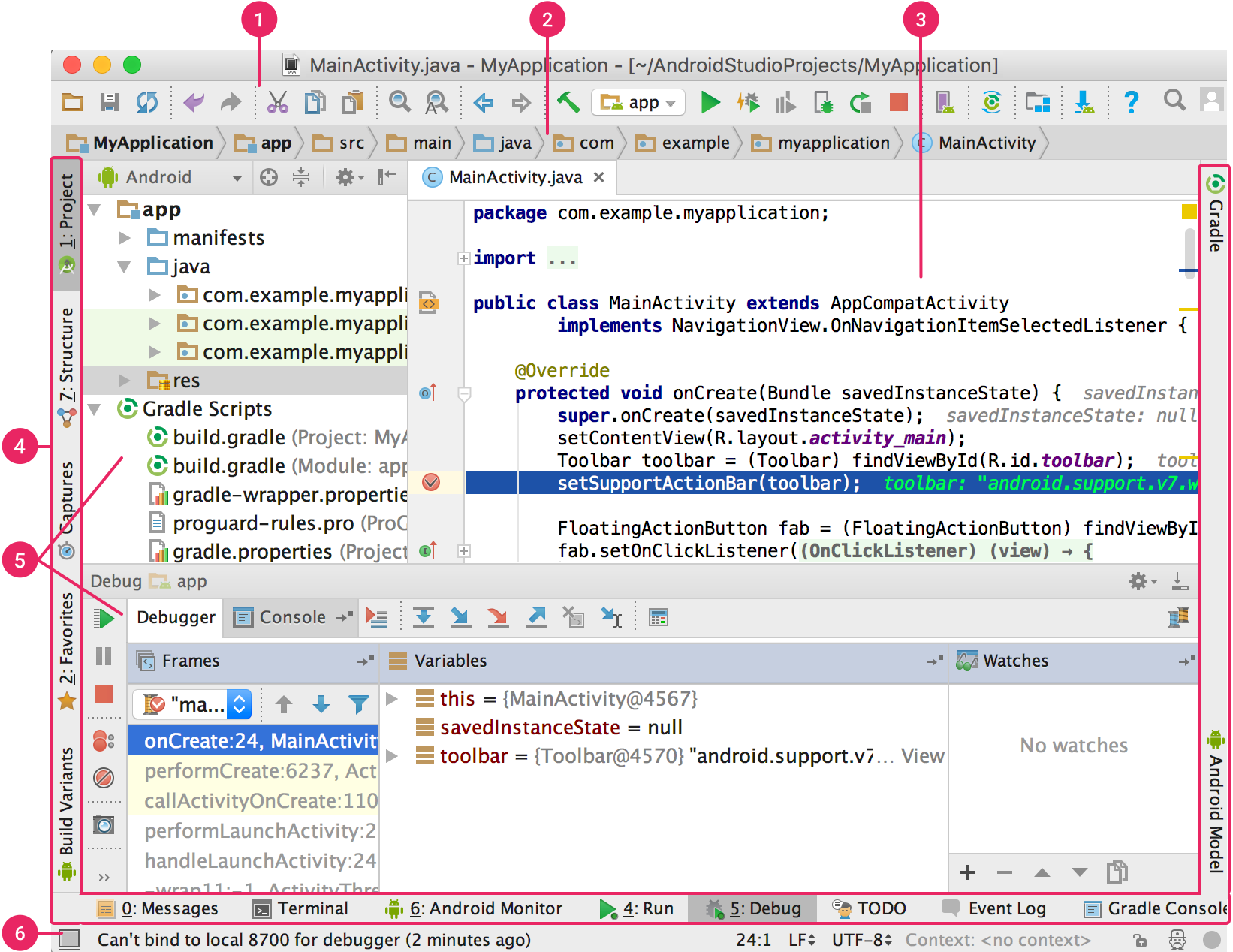
Hình 1: Các tệp dự án trong chế độ xem Android:



Hình 2: Các tệp dự án trong chế độ xem Problems(vấn đề) cho thấy một tệp về bố cục có lỗi:



*Giao diện người dùng*



1: Toolbar (thanh công cụ) cho phép bạn thực hiện nhiều thao tác, bao gồm cả việc chạy ứng dụng của bạn và chạy các công cụ Android.

2: Navigation bar (thanh điều hướng) giúp bạn điều hướng trong dự án và mở các tệp để chỉnh sửa. Công cụ này mang một cấu trúc nhỏ gọn hơn cấu trúc bạn thấy trong cửa sổ Project (Dự án).

3: Editor window (Cửa sổ trình chỉnh sửa) là nơi bạn tạo và sửa đổi mã. Tuỳ thuộc vào loại tệp hiện tại, trình chỉnh sửa có thể thay đổi. Ví dụ: khi bạn xem tệp bố cục, trình chỉnh sửa sẽ xuất hiện dưới dạng Layout Editor (Trình chỉnh sửa bố cục).

4: Tool window bar (Thanh cửa sổ công cụ) nằm xung quanh bên ngoài cửa sổ IDE và chứa các nút cho phép bạn mở rộng hoặc thu gọn từng cửa sổ công cụ riêng lẻ.

5: Tool windows (Cửa sổ công cụ) cho phép bạn truy cập vào các công việc cụ thể như quản lý dự án, tìm kiếm, kiểm soát phiên bản, v.v. Bạn có thể mở rộng và thu gọn chúng.

6: Status bar (Thanh trạng thái) cho thấy trạng thái của dự án và chính IDE cũng như mọi cảnh báo hoặc thông báo.

Bạn có thể sắp xếp cửa sổ chính để có thêm không gian trên màn hình bằng cách ẩn hoặc di chuyển thanh công cụ và cửa sổ công cụ. Bạn cũng có thể sử dụng phím tắt để truy cập vào hầu hết các tính năng của IDE.

Lúc nào bạn cũng có thể tìm kiếm trong mã nguồn, cơ sở dữ liệu, thao tác, phần tử của giao diện người dùng, v.v. bằng cách nhấn đúp vào phím Shift hoặc nhấp vào kính lúp ở phía trên bên phải cửa sổ Android Studio. Tính năng này có thể rất hữu ích, chẳng hạn như khi bạn đang cố gắng tìm một thao tác IDE cụ thể mà bạn đã quên cách kích hoạt.

*Hệ thống xây dựng Gradle*

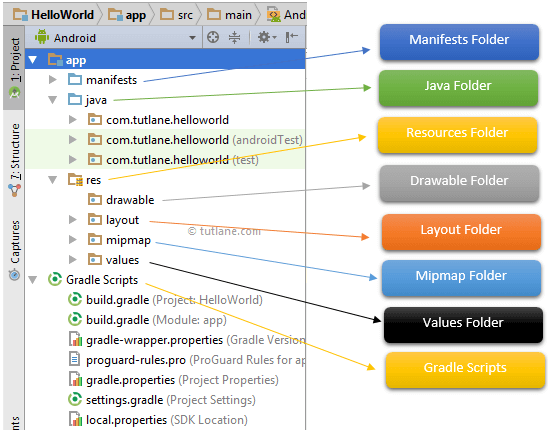
Android Studio sử dụng Gradle làm nền tảng cho hệ thống xây dựng với nhiều tính năng dành riêng cho Android do Trình bổ trợ Android dành cho Gradle cung cấp. Hệ thống xây dựng này hoạt động như một công cụ tích hợp trên trình đơn Android Studio và độc lập với dòng lệnh. Bạn có thể sử dụng các tính năng của hệ thống xây dựng để làm những việc sau:

* Tuỳ chỉnh, định cấu hình và mở rộng quy trình xây dựng.
* Tạo nhiều tệp APK cho ứng dụng với đa dạng tính năng trong khi sử dụng cùng một dự án và mô-đun.
* Sử dụng lại mã và tài nguyên trên các tập hợp nguồn (sourceset).

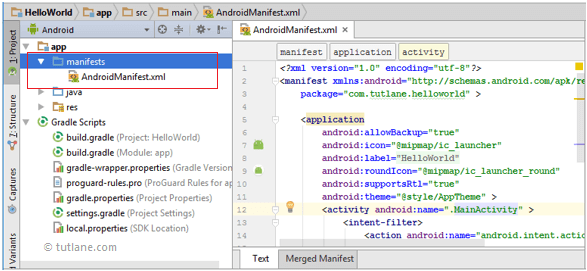
Nhờ vận dụng tính linh hoạt của Gradle, bạn có thể làm được những việc này mà không cần sửa đổi các tệp nguồn cốt lõi của ứng dụng. Tệp xây dựng trên Android Studio có tên là build.gradle. Đây là các tệp văn bản thuần tuý sử dụng cú pháp Groovy để định cấu hình bản dựng với các phần tử do Trình bổ trợ Android cho Gradle cung cấp. Mỗi dự án có một tệp xây dựng cấp cao nhất cho toàn bộ dự án và các tệp xây dựng cấp mô-đun riêng cho từng mô-đun. Khi bạn nhập một dự án hiện có, Android Studio sẽ tự động tạo ra các tệp xây dựng cần thiết.

*Cấu trúc thư mục*

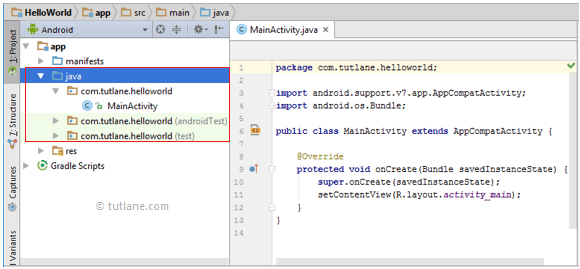
Khi thiết lập môi trường phát triển Android và tạo một ứng dụng bằngAndroid Studio thì bạn sẽ nhận được cấu trúc thư mục project sẽ giống như hình dưới đây:



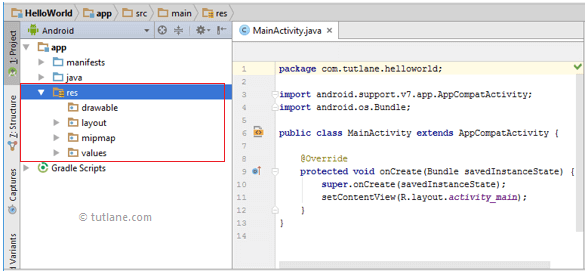
* Thư mục Manifest: Thư mục này sẽ chứa một file Manifest ( AndroidManifest.xml ) cho ứng dụng Android. File manifest này sẽ chứa thông tin về ứng dụng, chẳng hạn như android version, access permissions, metadata, v.v. và các component của ứng dụng. File manifest sẽ hoạt động như một trung gian giữa hệ điều hành Android và ứng dụng. Sau đây là cấu trúc của thư mục manifest:



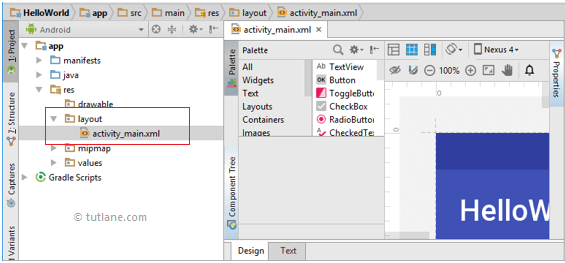
* Thư mục java: Thư mục này sẽ chứa tất cả các file mã nguồn java ( .java ) mà chúng ta sẽ tạo trong quá trình phát triển ứng dụng, bao gồm cả mã JUnit test code. Bất cứ khi nào tạo bất kỳ project / ứng dụng mới nào, file lớp MainActivity.java sẽ tự động tạo trong package là com.tutlane.helloworld, giống như dưới đây.



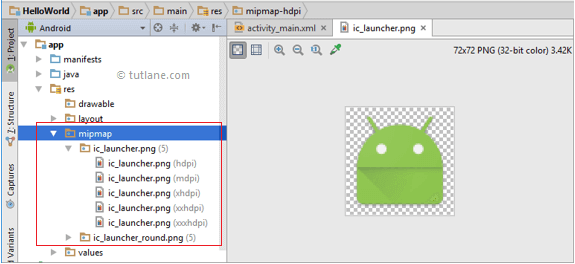
* Thư mục res(Resources): Đây là một thư mục quan trọng sẽ chứa tất cả các resource không phải code, chẳng hạn như ảnh bitmap, UI strings, XML layouts như hiển thị bên dưới.



* Thư mục drawable (res/drawable): Nó sẽ chứa các dạng ảnh khác nhau. Tốt nhất là nên thêm tất cả các hình ảnh vào thư mục drawable ngoại trừ các biểu tượng ứng dụng / laucher.
* Thư mục layout (res/layout): Thư mục này sẽ chứa tất cả các file XML layout đã sử dụng để xác định Giao diện người dùng của ứng dụng. Sau đây là cấu trúc của thư mục layout.



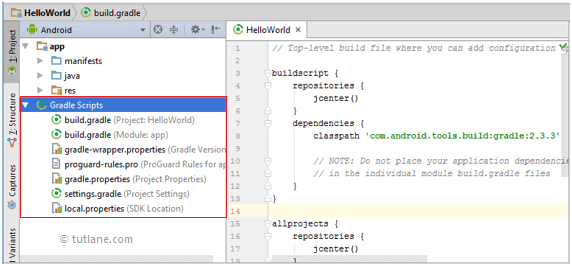
* Thư mục mipmap (res/ mipmap): Thư mục này sẽ chứa các biểu tượng ứng dụng / laucher được sử dụng để hiển thị trên màn hình chính. Các loại biểu tượng sẽ có tỷ trọng khác nhau như hdpi, mdpi, xhdpi, xxhdpi, xxxhdpi, để sử dụng dựa trên kích thước của thiết bị. Sau đây là cấu trúc của thư mục mipmap:



* Thư mục values (res/values): Thư mục này sẽ chứa một số file XML khác nhau, chẳng hạn như strings, colors, styles . Sau đây là cấu trúc của thư mục values.



* Gradle Scripts: Trong Android, Gradle là công cụ build hệ thống và Gradle được tích hợp sẵn vào Android Studio, và được điều khiển một cách tự động thông qua Android Studio. Trong gradle có build.gradle (Project) và build.gradle (Module) được sử dụng để build các cấu hình áp dụng cho tất cả các module ứng dụng hoặc dành riêng cho một mô-đun ứng dụng.



1. **Giới thiệu Kotlin**

*Kotlin là gì ?*

Kotlin là ngôn ngữ lập trình dành cho các ứng dụng đa nền tảng hiện đại, tương thích hoàn toàn với Android và được phát triển bởi JetBrains, công ty đứng sau IntelliJ Java IDE (bộ công cụ và ứng dụng dành cho nhà phát triển) mà Android Studio dựa trên. Tuy nhiên, đây không phải là ngôn ngữ lập trình riêng biệt mà được xem là một phương pháp mới để sử dụng và phát triển code của Java. Kotlin hoạt động liền mạch với Java, nhờ đó giúp các nhà phát triển yêu thích Java dễ dàng sử dụng thêm một ngôn ngữ lập trình tuyệt vời là Kotlin cũng như tận dụng thêm các thư viện hữu ích từ ngôn ngữ này.



Cũng giống như nhiều ngôn ngữ lập trình khác, Kotlin chạy trên Java Virtual Machine (JVM) và sử dụng các công cụ và thư viện sẵn có trên Java.

*Kotlin có gì vượt trội hơn Java?*

Java có thể được xem là ngôn ngữ lập trình xây dựng các ứng dụng chạy trên hầu hết mọi hệ điều hành, với mọi phần cứng. Java vô cùng mạnh mẽ, bạn có thể sử dụng Java để viết các chương trình thực hiện nhiều thứ khác nhau, như trò chơi điện tử, các tiện ích công nghệ, điều khiển phần cứng hoặc bất cứ thứ gì khác. Đó cũng là ngôn ngữ lập trình chính để phát triển các ứng dụng trên hệ điều hành Android.

Java nhìn chung là rất phức tạp. Và Kotlin ra đời để “loại bỏ” một số thành phần rườm rà, giúp chương trình trở nên ngắn gọn và rõ ràng hơn.

Mục đích chính của Kotlin là kế thừa những tốt đẹp và giải quyết những vấn đề mà Java gặp phải. Vì vậy câu hỏi "Kotlin hay Java?" đã xuất hiện rất nhiều trong cộng đồng phát triển Android kể từ sự kiện Google I/O khi mà Kotlin được gọi tên trong phần "IDE Android", trở thành ngôn ngữ được hỗ trợ số 1 trên nền tảng hệ điều hành này. Câu trả lời là code của Kotlin ngắn gọn và rõ ràng hơn Java, và các tệp của cả hai ngôn ngữ này có thể cùng tồn tại trong các ứng dụng Android. Do đó, Kotlin không chỉ hữu ích cho các ứng dụng mới mà còn mở rộng các ứng dụng Java hiện có.

Một số ưu điểm của Kotlin có thể kể đến như:

* Code ngắn gọn, giảm thiểu các đoạn không cần thiết, cấu trúc ngắn, trực quan, dễ học, dễ làm, dễ sử dụng, tiết kiệm thời gian. Theo ước tính của Ovechkin thì với việc sử dụng Kotlin, code base có thể được thu nhỏ hơn từ 15-25%.
* An toàn, tránh lỗi NullPointerException. NullPointerException xảy ra khi code tham chiếu đến một biến, nhưng biến lại không có giá trị khi ứng dụng đang chạy. Kết quả là ứng dụng sẽ không có bất kỳ tùy chọn nào ngoại trừ việc báo lỗi. Và tất nhiên, với Kotlin, đảm bảo các nhà phát triển sẽ không gặp phải vấn đề này.
* Khả năng tương tác cao, tương thích 100% và đẩy mạnh sử dụng các thư viện hiện có trong JVM, bao gồm cả SAM support.
* Tool làm việc thân thiện, có thể lựa chọn các Java IDE để làm việc hoặc làm việc với command line.

Tất cả những điều trên chính là lý do tại sao nhà phát triển yêu rất thích Kotlin. Code nhanh hơn, ít quy tắc hơn và đơn giản hơn so với Java. Bạn có thể đọc code mà chính bạn hoặc người khác viết một cách dễ chịu hơn hơn. Vẫn là code, nhưng không còn quá phức tạp đau đầu.

1. **Giới thiệu Room Database**

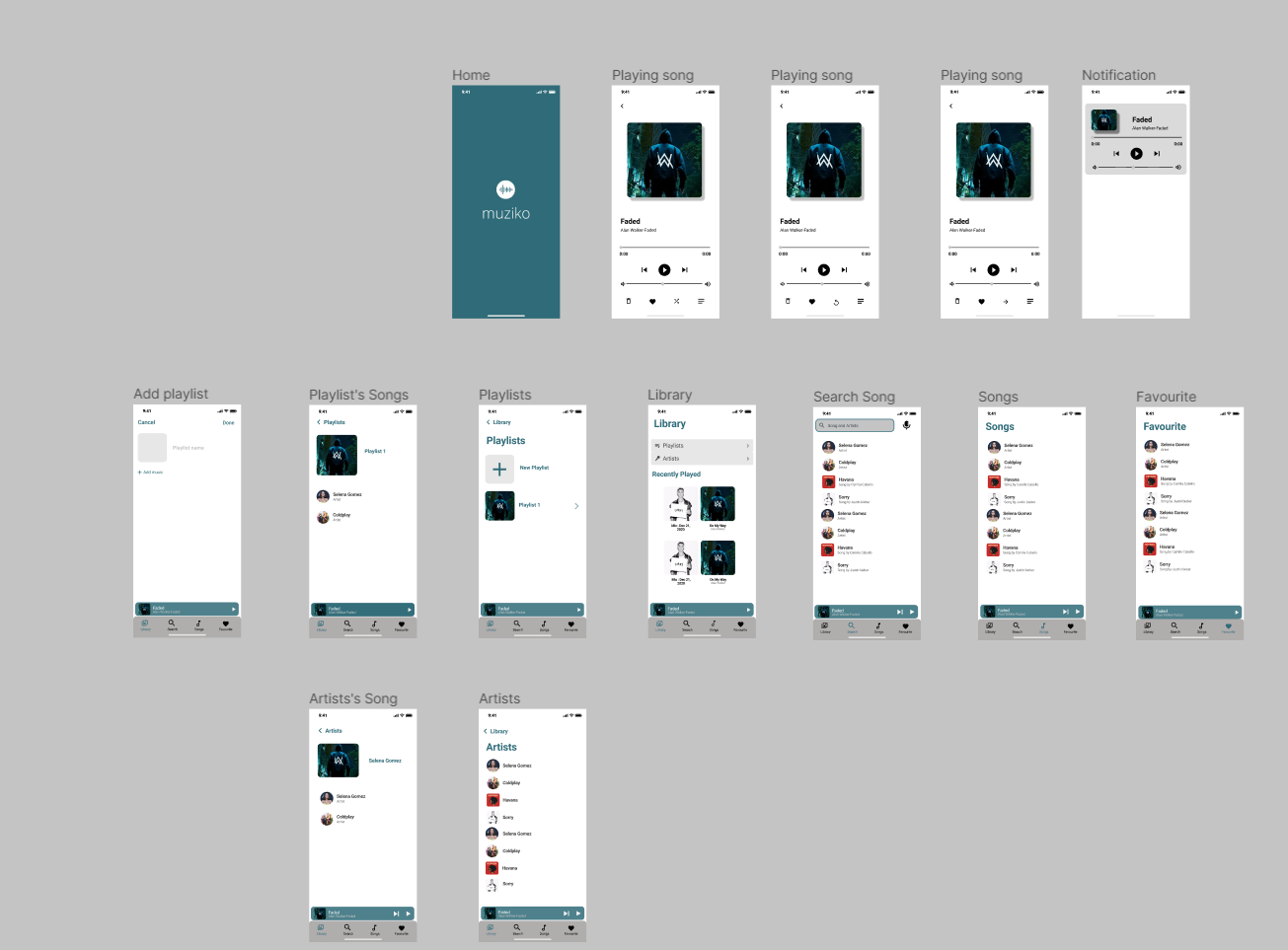
Room database được phát triển và cải tiến từ sqlite. Room database giúp đơn giản hoá việc code,và giảm thiểu các công đoạn liên quan đến cơ sở dữ liệu.

Bản chất Room database là abstract layer gồm cơ sở dữ liệu chuẩn SQLite được Android thông qua.

Với 3 thành phần chính là: Database, DAO(Data Access Object) và entity. Mỗi thành phần đều có nhiệm vụ và chức năng riêng.

1. **Bắt đầu xây dựng ứng dụng**

* Thiết kế giao diện ứng dụng:

****

* Tạo ứng dụng bằng Android Studio.
* Xây dựng database theo sơ đồ csdl đã nêu trên:
* Bảng Recently:

@Entity(tableName = "recently\_table")  
data class Recently(  
 @PrimaryKey(autoGenerate = true)  
 @ColumnInfo val songId: Long,  
 @ColumnInfo val time: Long  
)

* Bảng Favorite:

@Entity(tableName = "favorites\_table")  
data class Favorites(  
 @PrimaryKey(autoGenerate = true)  
*// @ColumnInfo var fId : Long,* @ColumnInfo val songId: Long  
)

* Bảng Playlist:

@Entity(tableName = "playlist\_table")  
@Parcelize  
data class Playlist(  
 @PrimaryKey var id: String = "",  
 @ColumnInfo var name: String = "",  
 @ColumnInfo var countOfSongs: Int = 0,  
 @ColumnInfo var songs: String  
): *Parcelable*

* Thiết lập database:

@Database(  
 entities = [Playlist::class, Favorites::class, Recently::class],  
 version = 1  
)  
abstract class MyDatabase : RoomDatabase() {  
 abstract fun playlistDAO(): *PlaylistDAO* abstract fun favoriteDao(): *FavouriteDao* abstract fun recentlyDao(): *RecentlyDao* private class MyDatabaseCallback(private val scope: *CoroutineScope*) : RoomDatabase.Callback() {  
 override fun onCreate(*db*: *SupportSQLiteDatabase*) {  
 super.onCreate(*db*)  
 INSTANCE?.*let* **{** *database* **->** scope.*launch* **{** populatePlaylistDatabase(database.playlistDAO())  
 populateFavoriteDatabase(database.favoriteDao())  
 populateRecentlyDatabase(database.recentlyDao())  
 **}  
 }** }  
  
 suspend fun populatePlaylistDatabase(*playlistDAO*: *PlaylistDAO*) {  
 *// Delete all content here.  
 playlistDAO*.deleteAll()  
  
 }  
 suspend fun populateFavoriteDatabase(*favoriteDao*: *FavouriteDao*) {  
 *// Delete all content here.  
 favoriteDao*.deleteAll()  
  
 }  
  
 suspend fun populateRecentlyDatabase(*recentlyDao*: *RecentlyDao*){  
 *recentlyDao*.deleteAll()  
 }  
 }  
  
 companion object {  
 @Volatile  
 private var INSTANCE: MyDatabase? = null  
  
 fun getDatabase(*context*: Context, *scope*: *CoroutineScope*): MyDatabase {  
 return INSTANCE ?: *synchronized*(MyDatabase::class.*java*)  
 **{** val instance = Room.databaseBuilder(  
 *context*.*applicationContext*,  
 MyDatabase::class.*java*, "\_database"  
 )  
 .addCallback(MyDatabaseCallback(*scope*))  
 .allowMainThreadQueries()  
 .build()  
 INSTANCE = instance  
  
*// return instance* instance  
 **}** }  
  
  
 }  
}

* Ta có 3 DAO chịu trách nhiệm cho việc thao tác với database như sau:

@Dao  
interface *PlaylistDAO* {  
  
 @Insert(onConflict = OnConflictStrategy.*IGNORE*)  
 suspend fun addPlaylist(*playlist*: Playlist)  
  
  
 @Query("DELETE FROM playlist\_table WHERE id = :playlistId")  
 suspend fun deletePlaylist(*playlistId*: String)  
  
  
 @Query("DELETE FROM playlist\_table")  
 suspend fun deleteAll()  
  
  
 @Query("SELECT \* FROM playlist\_table")  
 fun getPlaylists(): *List*<Playlist>  
  
  
 @Query("UPDATE playlist\_table SET songs=:songs WHERE id = :id")  
 suspend fun addSongToPlaylist(*id*: String, *songs*: String)  
  
 @Query("SELECT songs FROM playlist\_table WHERE id = :id")  
 suspend fun getSongsOfPlaylist(*id*: String): String  
  
 @Query("SELECT countOfSongs FROM playlist\_table WHERE id = :id")  
 fun getCountOfSongsInPlaylist(*id*: String): Int  
  
 @Query("UPDATE playlist\_table SET countOfSongs = :count WHERE id = :id")  
 fun setCountOfSongs(*id*: String, *count*: Int)  
  
 @Query("UPDATE playlist\_table SET songs = :songs WHERE id = :id")  
 fun updateSongs(*id*: String, *songs*: String)  
  
}

@Dao  
interface *FavouriteDao* {  
 @Insert(onConflict = OnConflictStrategy.*IGNORE*)  
 suspend fun addSong(*song*: Favorites)  
  
 @Delete  
 suspend fun deleteSong(*song*: Favorites)  
  
 @Query("DELETE FROM playlist\_table")  
 suspend fun deleteAll()  
  
 @Query("SELECT \* FROM favorites\_table")  
 fun getFavs(): *List*<Favorites>  
}

@Dao  
interface *RecentlyDao* {  
 @Insert(onConflict = OnConflictStrategy.*REPLACE*)  
 suspend fun addSong(*song*: Recently)  
  
 @Delete  
 suspend fun deleteSong(*song*: Recently)  
  
 @Query("DELETE FROM recently\_table")  
 suspend fun deleteAll()  
   
 @Query("SELECT \* FROM recently\_table ORDER BY time DESC")  
 fun getRecentlyTime(): *List*<Recently>  
}

* Lấy file âm thanh từ thiết bị:
* Khai báo lớp đối tượng:

data class Song(  
 val id : Long? = -1,  
 var title: String? = "",  
 val duration: Long? = -1,  
 val data: String? = "",  
 val dateAdded : String? = "",  
 val artist : String? = "",  
 val uri : Uri? = null,  
 val albumId : Long? = -1, *//TODO(change to long)* val size: String? = "",  
 val bitrate : String? ="",  
 val image : Bitmap? = null,  
 val trackNumber: String? = "",  
 val year: Int? = -1,  
 val dateModified: Long? = -1,  
 val artistId: Long? = -1,  
 val artistName: String? = "",  
 val composer: String? = "",  
 val albumArtist: String?= ""  
)

* Yêu cầu các quyền truy cập:

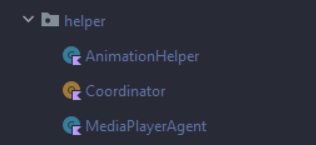
xmlns:tools="http://schemas.android.com/tools"  
package="com.android.music">  
<uses-permission  
 android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"  
 android:maxSdkVersion="31" />  
<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />  
<uses-permission android:name="android.permission.FOREGROUND\_SERVICE" />  
<uses-permission android:name="android.permission.MANAGE\_MEDIA"  
 tools:ignore="ProtectedPermissions" />  
<uses-permission android:name="android.permission.ACCESS\_MEDIA\_LOCATION" />  
<uses-permission android:name="android.permission.READ\_PHONE\_STATE" />  
  
<permission android:name="android.permission.MEDIA\_CONTENT\_CONTROL" />  
  
<uses-permission android:name="android.permission.WAKE\_LOCK" />  
<uses-permission android:name="android.permission.MODIFY\_AUDIO\_SETTINGS" />

* Đọc file âm thanh từ thiết bị:

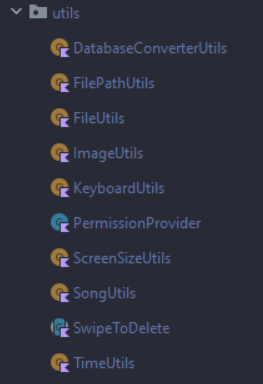
@SuppressLint("Range")  
fun createSongFromCursor(*cursor*: *Cursor*): Song {  
 val title = *cursor*.*getString*(MediaStore.Audio.*AudioColumns*.*TITLE*)  
 val duration = *cursor*.*getLong*(MediaStore.Audio.*AudioColumns*.*DURATION*)  
 val data = *cursor*.*getString*(MediaStore.Audio.*AudioColumns*.*DATA*)  
 val id = *cursor*.*getLong*(MediaStore.Audio.*AudioColumns*.*\_ID*)  
 val dateAdded = *cursor*.*getString*(MediaStore.Audio.*AudioColumns*.*DATE\_ADDED*)  
 val artist = *cursor*.*getString*(MediaStore.Audio.*AudioColumns*.*ARTIST*)  
 val year = *cursor*.*getInt*(MediaStore.Audio.*AudioColumns*.*YEAR*)  
 val dateModified = *cursor*.*getLong*(MediaStore.Audio.*AudioColumns*.*DATE\_MODIFIED*)  
 val artistId = *cursor*.*getLong*(MediaStore.Audio.*AudioColumns*.*ARTIST\_ID*)  
 val artistName = *cursor*.*getString*(MediaStore.Audio.*AudioColumns*.*ARTIST*)  
 val uri = ContentUris  
 .withAppendedId(  
 MediaStore.Audio.Media.*EXTERNAL\_CONTENT\_URI*,  
 *cursor*.getLong(*cursor*.getColumnIndex(MediaStore.Images.*ImageColumns*.*\_ID*))  
 )  
 val albumId = *cursor*.*getLong*(MediaStore.Audio.*AudioColumns*.*ALBUM\_ID*)  
 val size = *cursor*.*getString*(MediaStore.Audio.*AudioColumns*.*SIZE*)  
  
 val image = ImageUtils.albumArtUriToBitmap(context, albumId)  
 ?: BitmapFactory.decodeResource(  
 context.*resources*, R.mipmap.*icon* )  
  
  
 var bitrate = ""  
 if (data != "") {  
 val metadata = MediaMetadataRetriever()  
 metadata.setDataSource(data)  
 bitrate =  
 metadata.extractMetadata(MediaMetadataRetriever.*METADATA\_KEY\_BITRATE*).*toString*()  
  
 }  
  
 return Song(  
 title = title,  
 duration = duration,  
 data = data,  
 dateAdded = dateAdded,  
 artist = artist,  
 id = id,  
 uri = uri,  
 albumId = albumId,  
 size = size,  
 bitrate = bitrate,  
 image = image,  
 trackNumber = "",  
 year = year,  
 dateModified = dateModified,  
 artistId = artistId,  
 artistName = artistName,  
 composer = "",  
 albumArtist = ""  
 )  
}

private fun getSongsFromStorage(): ArrayList<Song> {  
 val songsAreInStorage = ArrayList<Song>()  
 val cursor = FileUtils.fetchFiles(  
 fileType = FileUtils.*FILE\_TYPES*.MUSIC,  
 context = context  
 )  
 if (cursor != null && cursor.*count* != 0) {  
 do {  
 cursor.moveToNext()  
 cursor.*getLong*(MediaStore.Audio.*AudioColumns*.*DURATION*)  
 if (cursor.*getLong*(MediaStore.Audio.*AudioColumns*.*DURATION*)!! > 60000)  
 songsAreInStorage.add(createSongFromCursor(cursor))  
 } while (!cursor.*isLast*)  
 } else {  
 }  
 cursor?.close()  
 return songsAreInStorage  
}

* Tạo các class helper, utils:

****

* AnimationHelper: Hiệu ứng cho các nút bấm, chuyển trang,…
* Coordinator, MediaPlayerAgent: Điều khiển chơi nhạc, next, play, pause, nghe nhạc dưới dạng thông báo.

****

* DatabaseConverterUtils:Phục vụ cho việc sửa đổi playlist bài hát.
* FilePathUtils: getMusic, Playlist, Album, …
* FileUtils: Chia sẻ file, chuyển bài hát thành file.
* ImageUtils: load ảnh vào các view, lấy ảnh từ file âm thanh.
* KeyBoardUtils: Ẩn bàn phím
* PermissionProvider: Yêu cầu các quyền truy cập.
* ScreenSizeUtils: Lấy kích thước của màn hình đang hiển thị.
* SongUtils: Xoá bài hát, lấy bài hát bằng id.
* SwipeToDelete: Call back để xoá.
* TimeUtils: Hàm xử lý thời gian.
* Tạo BottomNavigation:
* Tạo menu gồm các item như hình:

**Table

Description automatically generated**

* Tạo BottomNavigationView trong activity\_main.xml:

<com.google.android.material.bottomnavigation.BottomNavigationView  
 android:id="@+id/nav\_view"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:background="?android:attr/windowBackground"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:menu="@menu/nav\_menu" />

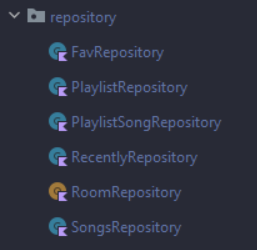
* Hoàn thành setUp ta được:

val navView: BottomNavigationView = binding.navView  
val navHostFragment = *supportFragmentManager*.findFragmentById(R.id.*nav\_host\_fragment\_activity\_main*) as NavHostFragment  
val navController = navHostFragment.*navController*navView.*setupWithNavController*(navController)

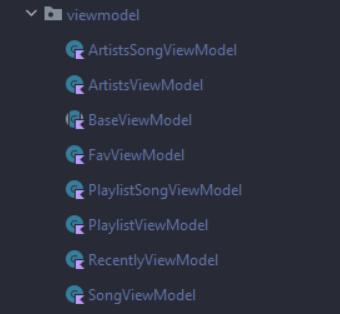
**Text

Description automatically generated**

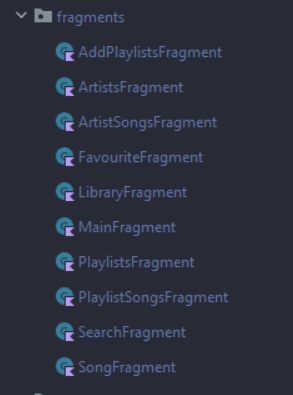
* Tạo các ViewModel, Repository theo MVVM

****

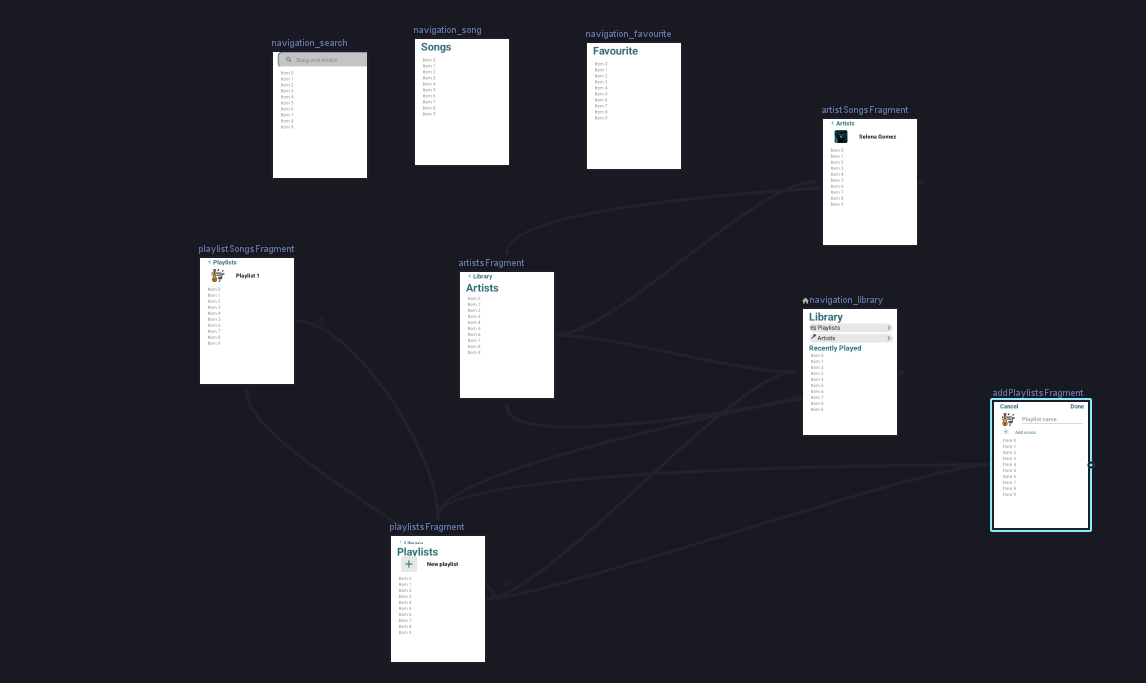
* Ứng với mỗi Repository có 1 ViewModel (1 số ViewModel như Artist thì sẽ lấy thẳng ở SongsRepository.

****

* Tạo các fragment (view) để lấy data từ viewmodel và hiển thị lên:

****

* Tạo navigation để chuyển giữa các fragment:

****

* Tạo Service, BroadcastReceive để điều khiển nhạc bằng thông báo:
* Thêm vào file Manifest.xml:

<service android:name="com.android.music.muziko.service.NotificationPlayerService" />  
<receiver android:name="com.android.music.muziko.service.NotificationBroadcastReceiver" />

* Tạo service + notification:

class NotificationPlayerService : Service() {  
  
  
 lateinit var notification: Notification  
  
 companion object {  
  
 fun startNotification(*context*: Context, *message*: String) {  
 Log.e("service", "start noti")  
 val intent = Intent(*context*, NotificationPlayerService::class.*java*)  
 intent.putExtra("inputExtra", *message*)  
 ContextCompat.startForegroundService(*context*, intent)  
 }  
  
 fun stopNotification(*context*: Context) {  
 Log.e("service", "stop noti")  
 val intent = Intent(*context*, NotificationPlayerService::class.*java*)  
 *context*.stopService(intent)  
 }  
 }  
  
 @SuppressLint("ResourceAsColor", "UnspecifiedImmutableFlag")  
 @RequiresApi(Build.VERSION\_CODES.*O*)  
 override fun onStartCommand(*intent*: Intent?, *flags*: Int, *startId*: Int): Int {  
 super.onStartCommand(*intent*, *flags*, *startId*)  
  
 createNotificationChannel()  
  
 try {  
 registerReceiver(broadcastNotificationReceiver, IntentFilter("Songs"))  
 } catch (e: Exception) {  
 }  
  
  
 val notificationIntent = Intent(this, MainActivity::class.*java*)  
 PendingIntent.getActivity(  
 this,  
 0, notificationIntent, 0  
 )  
  
 val mediaSession = android.support.v4.media.session.MediaSessionCompat(  
 this,  
 "notif"  
 )  
  
  
 val style = androidx.media.app.NotificationCompat.MediaStyle()  
 .setMediaSession(mediaSession.*sessionToken*)  
 .setShowCancelButton(false)  
  
  
 val intentNext = Intent(this, NotificationBroadcastReceiver::class.*java*)  
 .setAction(getString(R.string.*notification\_action\_next*))  
 val nextPendingIntent =  
 PendingIntent.getBroadcast(this, 0, intentNext, PendingIntent.*FLAG\_UPDATE\_CURRENT*)  
  
 val intentPlay = Intent(this, NotificationBroadcastReceiver::class.*java*)  
 .setAction(getString(R.string.*notification\_action\_play*))  
 val playPendingIntent =  
 PendingIntent.getBroadcast(this, 0, intentPlay, PendingIntent.*FLAG\_UPDATE\_CURRENT*)  
  
 val intentPause = Intent(this, NotificationBroadcastReceiver::class.*java*)  
 .setAction(getString(R.string.*notification\_action\_pause*))  
 val pausePendingIntent =  
 PendingIntent.getBroadcast(this, 0, intentPause, PendingIntent.*FLAG\_UPDATE\_CURRENT*)  
  
 val intentPrev = Intent(this, NotificationBroadcastReceiver::class.*java*)  
 .setAction(getString(R.string.*notification\_action\_previous*))  
 val prevPendingIntent =  
 PendingIntent.getBroadcast(this, 0, intentPrev, PendingIntent.*FLAG\_UPDATE\_CURRENT*)  
  
  
 notification = NotificationCompat.Builder(this, getString(R.string.*notification\_channel\_id*))  
 .setContentTitle(Coordinator.currentPlayingSong?.title)  
 .setNotificationSilent()  
 .setContentText(Coordinator.currentPlayingSong?.artistName ?: "")  
 .setStyle(style)  
 .setAutoCancel(true)  
 .setColor(Color.*BLUE*)  
 .setStyle(  
 androidx.media.app.NotificationCompat.MediaStyle().setShowActionsInCompactView(  
 0,  
 1,  
 2,  
 3,  
 4  
 )  
 )  
 .setPriority(NotificationCompat.*PRIORITY\_DEFAULT*)  
 .setSmallIcon(R.drawable.*ic\_song*)  
 .setLargeIcon(Coordinator.currentPlayingSong?.image)  
 .addAction(R.drawable.*ic\_notification\_previous*, "previous", prevPendingIntent)  
 .addAction(R.drawable.*ic\_notification\_play*, "play", playPendingIntent)  
 .addAction(R.drawable.*ic\_notification\_pause*, "pause", pausePendingIntent)  
 .addAction(R.drawable.*ic\_notification\_next*, "next", nextPendingIntent)  
 .build()  
*// .setContentIntent(pendingIntent)* startForeground(1, notification)  
  
 return *START\_NOT\_STICKY* }  
  
  
 private fun createNotificationChannel() {  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*) {  
 val serviceChannel = NotificationChannel(  
 getString(R.string.*notification\_channel\_id*), "Foreground Service Channel",  
 NotificationManager.*IMPORTANCE\_DEFAULT* )  
 serviceChannel.*description* =  
 getString(R.string.*notification\_description*)  
 val manager = getSystemService(NotificationManager::class.*java*)  
 manager!!.createNotificationChannel(serviceChannel)  
 }  
 }  
  
 override fun onTaskRemoved(*rootIntent*: Intent?) {  
 stopSelf()  
 unregisterReceiver()  
 onDestroy()  
 }  
  
 private fun unregisterReceiver() {  
 unregisterReceiver(broadcastNotificationReceiver)  
 }  
  
  
 override fun onBind(*intent*: Intent?): *IBinder*? {  
 return null  
 }  
  
 private val broadcastNotificationReceiver: BroadcastReceiver = object : BroadcastReceiver() {  
 @RequiresApi(Build.VERSION\_CODES.*O*)  
 override fun onReceive(*context*: Context, *intent*: Intent) {  
 when (*intent*.*extras*!!.getString(*context*.getString(R.string.*extra\_key*))) {  
 getString(R.string.*notification\_action\_next*) -> {  
 Log.e("service", "received")  
 Coordinator.playNextSong()  
 Coordinator.currentPlayingSong?.*let* **{** MainActivity.activity.updateVisibility(**it**) **}** }  
 getString(R.string.*notification\_action\_play*) -> {  
 Coordinator.resume()  
 Coordinator.currentPlayingSong?.*let* **{** MainActivity.activity.updateVisibility(**it**) **}** }  
 getString(R.string.*notification\_action\_pause*) -> {  
 Coordinator.pause()  
 Coordinator.currentPlayingSong?.*let* **{** MainActivity.activity.updateVisibility(**it**) **}** }  
 getString(R.string.*notification\_action\_previous*) -> {  
 Coordinator.playPrevSong()  
 Coordinator.currentPlayingSong?.*let* **{** MainActivity.activity.updateVisibility(**it**) **}** }  
 }  
  
 }  
 }  
}

* Tạo BroadCastReceiver:

class NotificationBroadcastReceiver: BroadcastReceiver() {  
 override fun onReceive(*p0*: Context?, *p1*: Intent?) {  
 *p0*?.sendBroadcast(  
 Intent("Songs")  
 .putExtra("actionname", *p1*?.*action*)  
 )  
 }  
}

Thành quả:

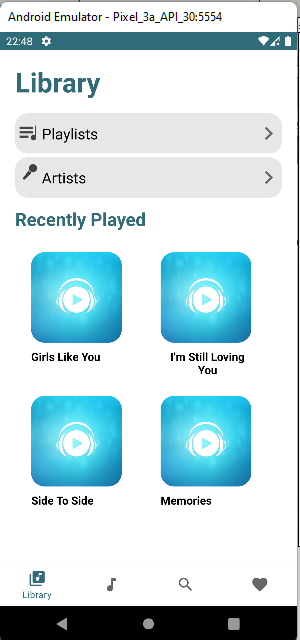
Graphical user interface, application, website

Description automatically generated

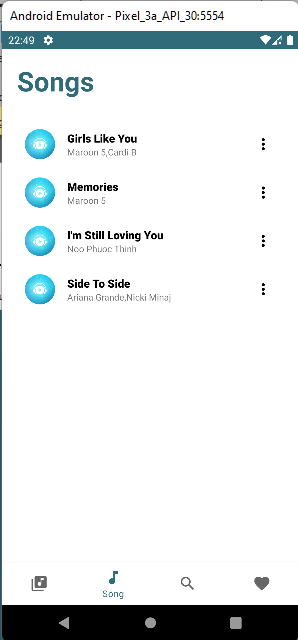
1. **Kết quả**
2. **SplashScreen**

****

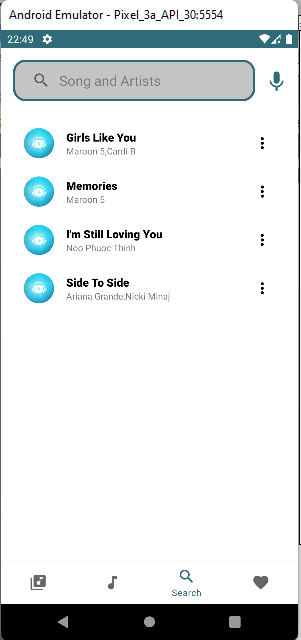
1. **Màn hình chính**

****

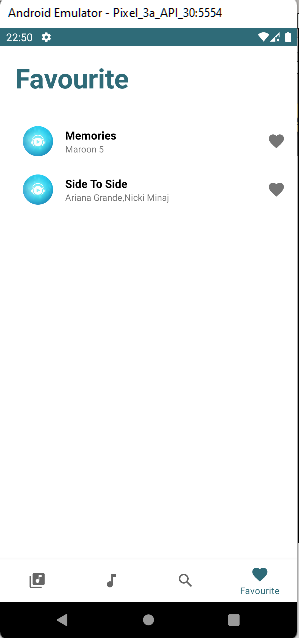
1. **SongFragment**

****

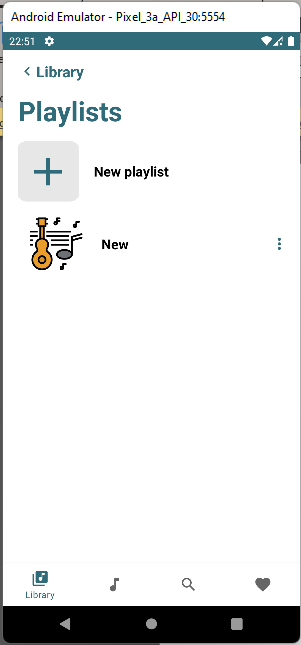
1. **SearchFragment**

****

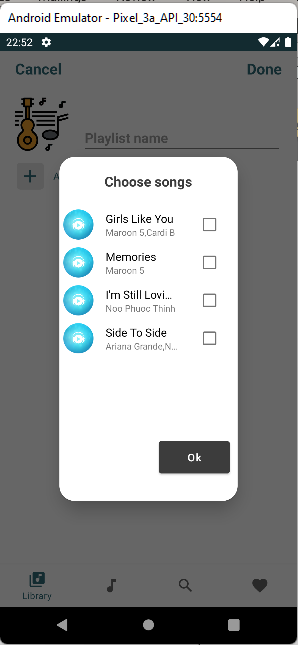
1. **FavoriteFragment**

****

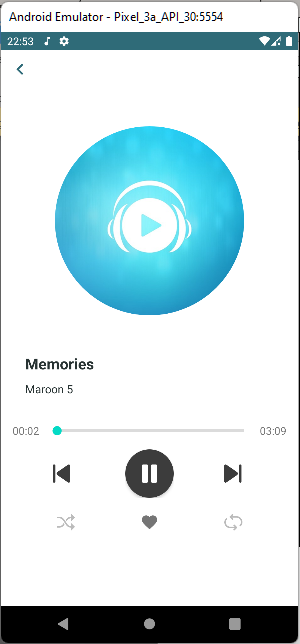
1. **Playlist**

****

1. **Tạo mới playlist**

****

1. **Panel nghe nhạc**

****

**TỔNG KẾT**

Trong project này, em đã nghiên cứu, tìm hiểu về ứng dụng phát nhạc trên Android.

Project đã thực hiện các nhiệm vụ sau:

* Tìm hiểu Android.
* Tìm hiểu IDE Android Studio.
* Tìm hiểu Git.
* Tìm hiểu ngôn ngữ Kotlin.
* Tìm hiểu Room Database.
* Tìm hiểu MVVM.
* Nắm bắt quy trình làm một phần mềm Android.
* Kỹ thuật phát nhạc trên Android.
* Các kỹ thuật liên quan đến Service, MediaPlayer, Navigation, BroadCastReceiver,…
* Bước đầu xây dựng thành công ứng dụng Android nghe nhạc offline.

Tuy nhiên, do thời gian và khả năng có hạn, nên em chưa đi sâu tìm hiểu được thêm về ứng dụng, vì vậy giao diện ứng dụng vẫn còn sơ sài, xử lý code còn chưa tốt, chương trình vẫn còn nhiều thiếu xót, mong thầy tận tình giúp đỡ để ứng dụng ngày một hoàn thiện hơn.

Trong tương lại, em mong muốn ứng dụng phát triển về mọi mặt (giao diện, chức năng, …) thêm nghe nhạc online, cài đặt,… Và sẽ trở thành một trình nghe nhạc phố biến.

**TÀI LIỆU THAM KHẢO**

[1] <https://chiasekinang.com/mo-hinh-mvvm/>

[2] [Introduction to Git and GitHub (w3schools.com)](https://www.w3schools.com/git/git_intro.asp?remote=github#:~:text=Git%20is%20a%20popular%20version,Tracking%20code%20changes)

[3] [MVVM (Model View ViewModel) Architecture Pattern in Android - GeeksforGeeks](https://www.geeksforgeeks.org/mvvm-model-view-viewmodel-architecture-pattern-in-android/)

[4] [Android Developers](https://developer.android.com/)

[5] [Philipp Lackner - YouTube](https://www.youtube.com/c/PhilippLackner)