

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC MỞ TP. HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN**



Báo Cáo Giữa Kỳ

Đề Tài: Ứng dụng đặt lịch cắt tóc

Môn học: Lập trình trên thiết bị di động
Giảng viên hướng dẫn: Dương Thái Bảo
Lớp: DH21CS01
Nhóm thực hiện: Nhóm 19
Thành viên: 2151010290 – Đặng Minh Phúc
2151010005 -Phạm Dương An
2151013093 – Hà Phúc Thịnh

Thành phố Hồ Chí Minh, 2024

MỤC LỤC

Chương 1: TỔNG QUAN.....	4
1. Giới thiệu đề tài:.....	4
2. Lý do chọn đề tài:.....	4
Chương 2: CƠ SỞ LÝ THUYẾT.....	5
2.1 Android App Architecture & Components	5
2.1.1 App Architecture & Components là gì?	5
2.1.2 Lợi ích của Android Architecture Components.....	5
2.2 Mô hình MVVM (Model-View-ViewModel)	6
2.2.1 Mô Hình MVVM là gì?	6
2.2.2 Giới thiệu về kiến trúc MVC	8
2.3 LyfeCycle.....	9
2.4 Singleton pattern	10
2.4.1 Singleton pattern là gì ?	10
2.4.2 Nguyên lý cấu tạo lớp singleton:	10
2.5 FireBase	11
2.5.1 FireBase là gì	11
2.5.2 Cách thức hoạt động của FireBase.....	12
2.5.3 Nhóm công cụ Firebase Develop & Test Your App.....	12
2.5.4 Ưu Điểm.....	13
2.5.5 Hạn Chế.....	13
2.6 Giao Diện JetPack Compose.....	14
2.6.1 JetPack Compose là gì?	14
2.6.2 Các thành phần của Android Jetpack.....	14
2.7 Clean Architecture	14
2.7.1 Clean Architecture là gì?	14
2.7.2 Lợi ích của việc sử dụng Clean Architecture trong Jetpack Compose .	15
CHƯƠNG 3: CÁC CHỨC NĂNG CỦA ĐỀ TÀI.....	17
3.1 Đăng ký, đăng nhập	17
3.1.1 Đăng nhập	18

3.1.2 Đăng ký	19
3.2 Bên User.....	20
3.2.1 Trang Home	20
3.2.3 Trang Baber.....	21
3.2.4 Trang Boking	22
3.2.5 Trang History	23
3.2.6 Trang Profile	24
3.3.1 Trang Home Admin	25
3.2.2 Trang quản lý người dùng.....	26
3.3.3 Trang quản lý đơn hàng	27
3.3.4 Trang quản lý thợ cắt tóc	28
Chương4 : KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....	29
1. Kết quả đạt được	29
1.1 Ưu điểm.....	29
1.2 Nhược điểm.....	29
2. Hướng phát triển	29
3. Tài liệu tham khảo.....	29

MỤC LỤC ẢNH

Hình 1 Android App Architecture & Components	5
Hình 2 Mô hình MVVM	7
Hình 3 Mô hình MVC	8
Hình 4 Hoạt động và kết hợp với nhau của LifeCycle, State và Event	10
Hình 5 Nguyên lý cấu tạo lớp singleton	11
Hình 6 The clean Architecture	15
Hình 7 UI Domain Data	16
Hình 8 Màn hình đăng ký đăng nhập.....	17
Hình 9 Màn hình đăng nhập.....	18
Hình 10 Màn hình Đăng ký tài khoản mới	19
Hình 11 Màn hình trang Home	20
Hình 12 Màn hình trang danh sách thợ cắt tóc	21

Hình 13	Màn hình Booking	22
Hình 14	Màn hình lịch sử đặt lịch.....	23
Hình 15	Màn hình profile.....	24
Hình 16	Màn hình trang HomeAdmin	25
Hình 17	Màn hình trang quản lý người dùng.....	26
Hình 18	Màn hình quản lý đơn hàng	27
Hình 19	Màn hình quản lý nhân viên.....	28

Chương 1: TỔNG QUAN

1. Giới thiệu đề tài:

Đề tài xây dựng phần mềm đặt lịch cắt tóc. Có các chức năng như: đăng nhập, tạo tài khoản, phân quyền, đặt lịch, quản lý khách hàng, thống kê đơn hàng.

Chương trình được xây dựng trên ngôn ngữ lập trình Kotlin và sử dụng Firebase để quản lý cơ sở dữ liệu trong chương trình.

2. Lý do chọn đề tài:

Thời đại 4.0 nở rộ tạo cho con người nhiều thành tựu tiên tiến.

Ngày nay cuộc sống của con người đòi hỏi những nhu cầu một tầng cao. Từ công nghệ thông tin góp phần quan trọng làm cho các ngành này phát triển nhanh hơn.

Với sự phát triển mạnh mẽ của Công nghệ thông tin, ngày nay rất nhiều ứng dụng ra đời với mục tiêu quảng bá thông tin, thương hiệu, thông tin cho công ty, cho lĩnh vực nào đó. Nhờ vậy mà đơn giản và tối ưu hóa mọi việc từ những việc nhỏ nhất nhằm giúp ích cho mọi ngành hiện nay. Dựa vào khảo sát thực tế trên mạng xã hội và internet thì em nhận ra rằng nhu cầu tiện lợi của các dịch vụ ngày càng được chú trọng trong việc sắp xếp thời gian. Nhờ vào lời gợi ý của bạn bè và những người có kinh nghiệm đi trước thì em thấy những quán cắt tóc hiện nay vẫn còn khá chậm so với thực tế. Họ chưa chú trọng tới việc quảng bá hình ảnh.

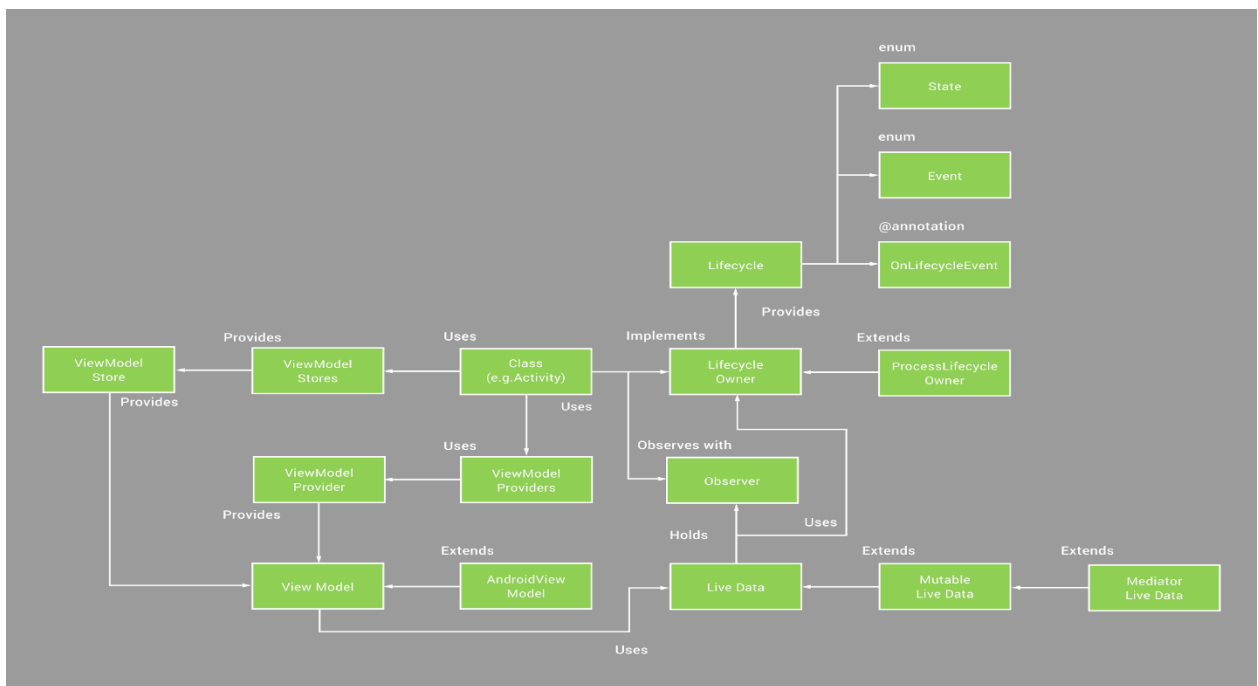
Để cụ thể hóa thì em xin thực hiện đề tài: “Xây dựng app Đặt Lịch Cắt Tóc”

Chương 2: CƠ SỞ LÝ THUYẾT

2.1 Android App Architecture & Components

2.1.1 App Architecture & Components là gì?

Android App Architecture & Components là một bộ thư viện giúp xây dựng và phát triển ứng dụng Android một cách hiệu quả và dễ dàng bảo trì. Kiến trúc ứng dụng Android thường được dùng theo mô hình **MVVM (Model-View-ViewModel)** hoặc **MVP (Model-View-Presenter)**



Hình 1 Android App Architecture & Components

2.1.2 Lợi ích của Android Architecture Components

Quản lý vòng đời ứng dụng của bạn dễ dàng hơn.

Giúp giữ lại trạng thái cấu hình, tránh bị lack bộ nhớ và dễ dàng hiển thị dữ liệu lên giao diện.

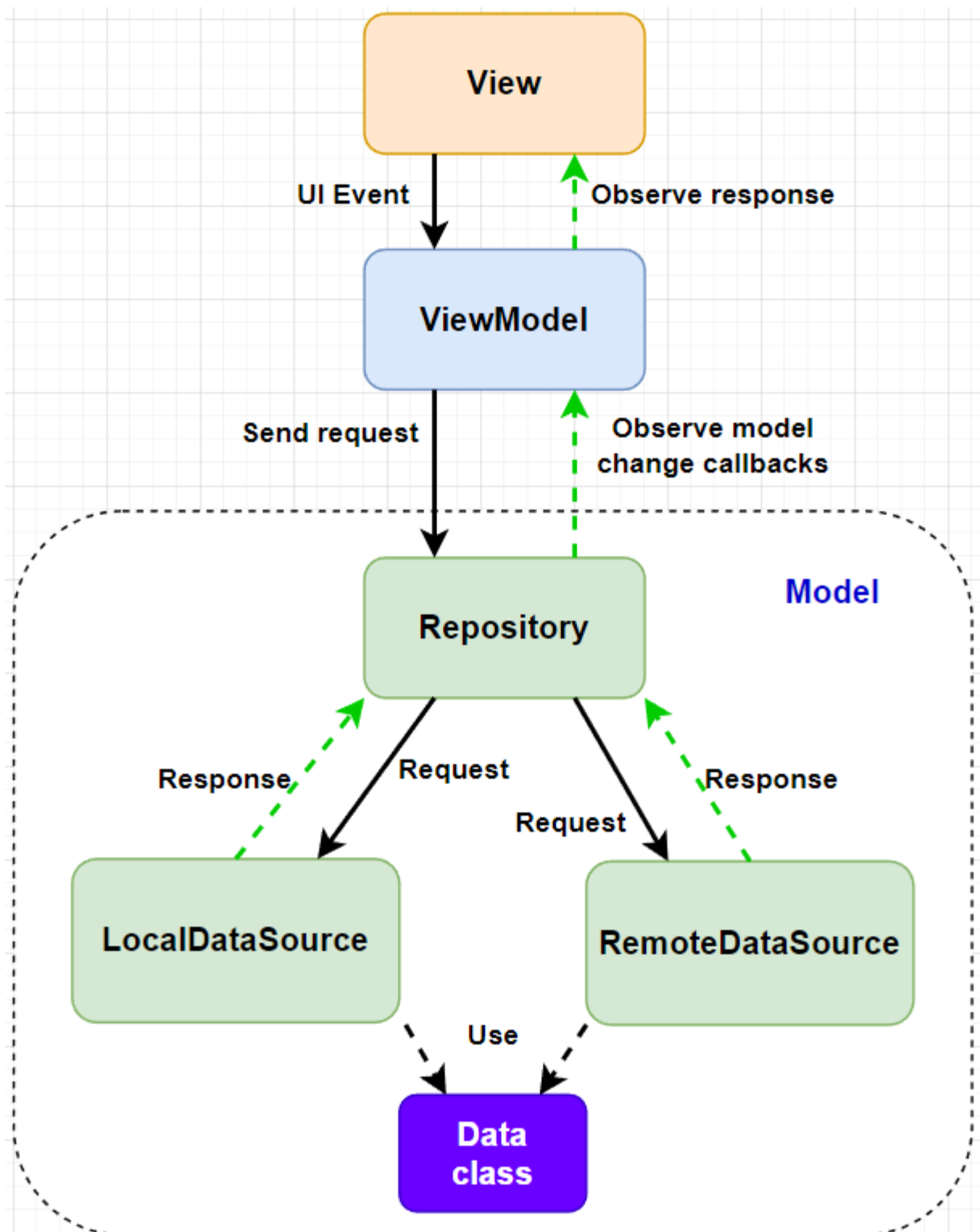
Sử dụng LiveData để xây dựng object có thể thông báo cho UI khi database thay đổi.

ViewModel để lưu giữ UI khi app quay ngang/dọc màn hình

2.2 Mô hình MVVM (Model-View-ViewModel)

2.2.1 Mô Hình MVVM là gì?

MVVM là viết tắt của **Model – View – ViewModel**. Đây là các lớp nòng cốt trong kiến trúc ở đó view (tức giao diện người dùng) sẽ được cập nhật bởi ViewModel và việc xử lý Logic hoặc trình bày dữ liệu sẽ do Model đảm nhận..



Hình 2 Mô hình MVVM

Model:

- Đại diện cho phần dữ liệu và logic nghiệp vụ của ứng dụng Android. Trong đó bao gồm các lớp model(data class trong Kotlin), các repository, các lớp local data source và remote data source.

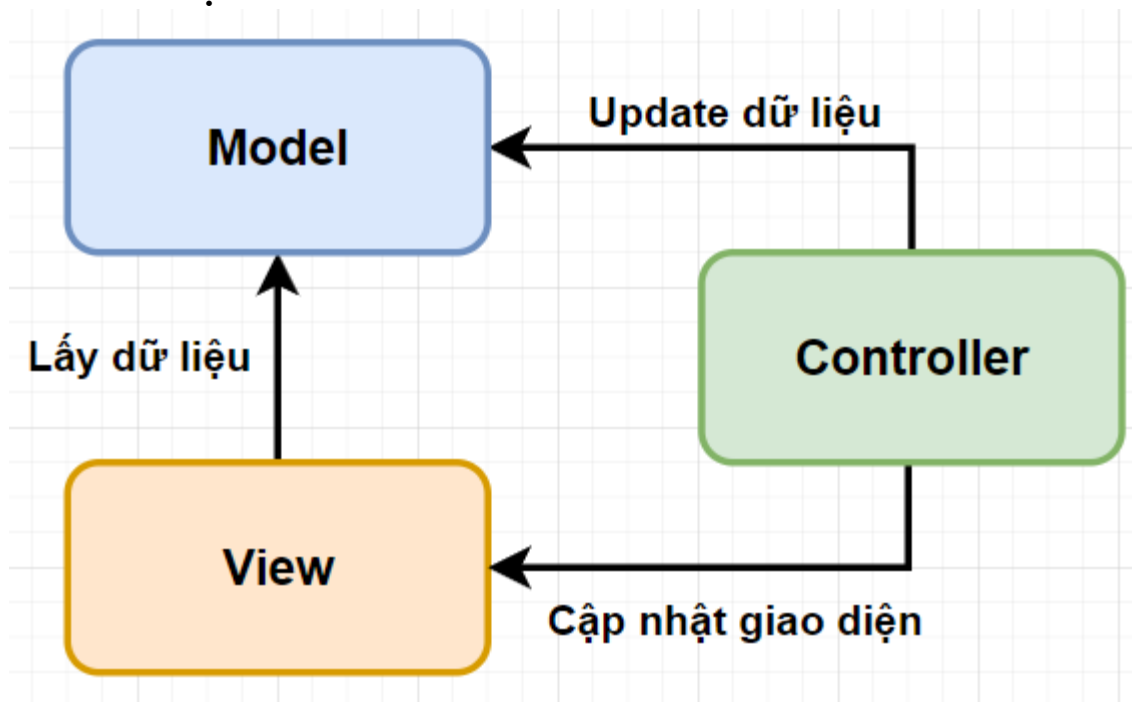
View:

- Là các lớp quản lý UI như các activity, fragment, xml. Các lớp này có nhiệm vụ hiển thị dữ liệu lên màn hình và nhận các tương tác từ người dùng gửi tới ViewModel nhưng không nhận phản hồi trực tiếp từ ViewModel. Các lớp View sẽ đăng ký nhận thông tin thay đổi trạng thái từ ViewModel cung cấp.

ViewModel:

- Là cầu nối giữa các View(UI) và Model(data). ViewModel không chứa tham chiếu trực tiếp tới bất kỳ View nào cả, do đó nó không nhận biết được là đang làm việc, tương tác với View nào. ViewModel tương tác với các lớp Model và cung cấp các dữ liệu có thể quan sát được cho các lớp View.

2.2.2 Giới thiệu về kiến trúc MVC



Hình 3 Mô hình MVC

MVC chia các lớp trong ứng dụng của ta thành 3 tầng là Model – View – Controller.

Model: tầng này lưu trữ dữ liệu của ứng dụng. Nó không có thông tin gì về phần giao diện người dùng của ứng dụng. Lớp model cũng chịu trách nhiệm xử lý các logic nghiệp vụ và giao tiếp với database, kết nối mạng...

View: là tầng giao diện người dùng chứa các thành phần hiển thị trên màn hình và nhận tương tác từ người dùng. Trong Android, tầng view là nơi chứa các view như activity, fragment, các view tùy chỉnh.

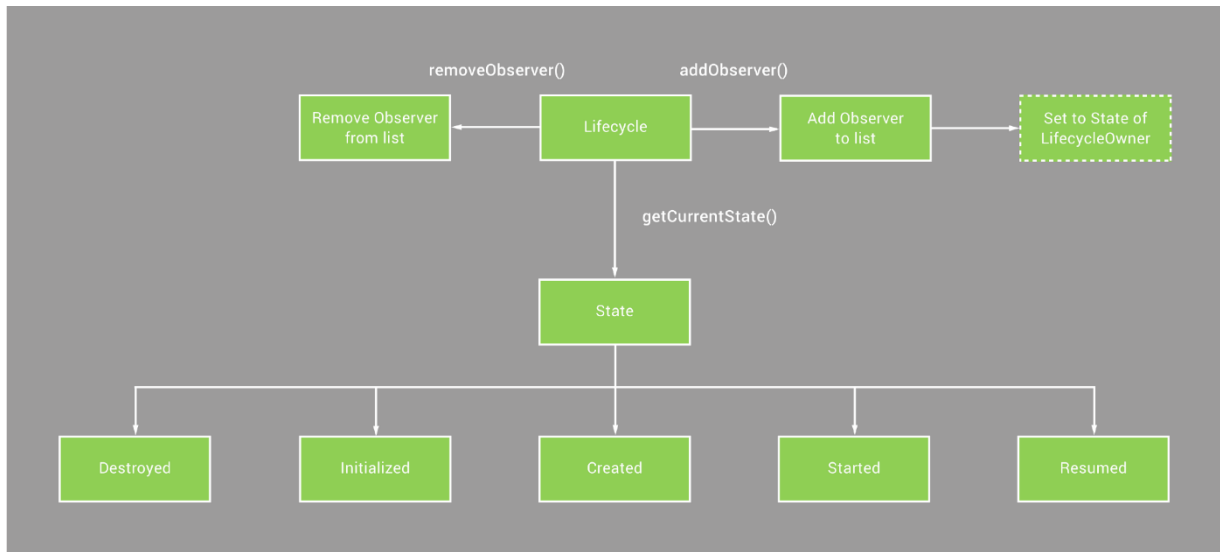
Controller: tầng này chứa các lớp dùng để thiết lập mối quan hệ giữa View và Model. Nó chứa các logic lõi của ứng dụng, phản hồi tương tác của người dùng và cập nhật dữ liệu trong tầng model nếu cần.

Ý tưởng của mô hình MVC là cung cấp một thiết kế mô đun hóa cho ứng dụng. Các tầng luôn phụ thuộc vào nhau, cụ thể, tầng View và Controller phụ thuộc vào dữ liệu được cung cấp bởi tầng Model.

2.3 LyfeCycle

LyfeCycle là 1 abstract class dùng để quản lý vòng đời của ứng dụng Android; Lifecycle có thể theo dõi trạng thái vòng đời của ứng dụng và đưa ra những event phù hợp. Để theo dõi được trạng thái vòng đời ta có hai biểu diễn kiểu Enum : Event và State

Event: Khi có bất kỳ vòng đời nào của ứng dụng có thay đổi thì sự kiện của Lifecycle sẽ được gọi đến; thông qua khai báo dạng Annotation của method
State: là thời điểm giữa 2 sự kiện của vòng đời ứng dụng



Hình 4 Hoạt động và kết hợp với nhau của Lifecycle, State và Event

2.4 Singleton pattern

2.4.1 Singleton pattern là gì ?

Mẫu thiết kế singleton là một trong các mẫu thiết kế đơn giản nhất thường sử dụng trong dự án phát triển phần mềm. Mẫu thiết kế này sử dụng vào các trường hợp ta cần có 1 đối tượng duy nhất chia sẻ chung giữa các đối tượng nhằm giảm tải, tránh dư thừa.

Ví dụ: đối tượng Live Data dùng chung, đối tượng ViewModel dùng chung, đối tượng kết nối database, thiết lập cài đặt chung toàn ứng dụng...

Về khái niệm, singleton pattern là mẫu thiết kế trong đó chỉ cho phép tạo duy nhất 1 đối tượng từ một lớp cụ thể. Thường sử dụng các tính chất và đặc trưng của các keyword hướng đối tượng như private, static để tạo ra mẫu thiết kế này.

2.4.2 Nguyên lý cấu tạo lớp singleton:

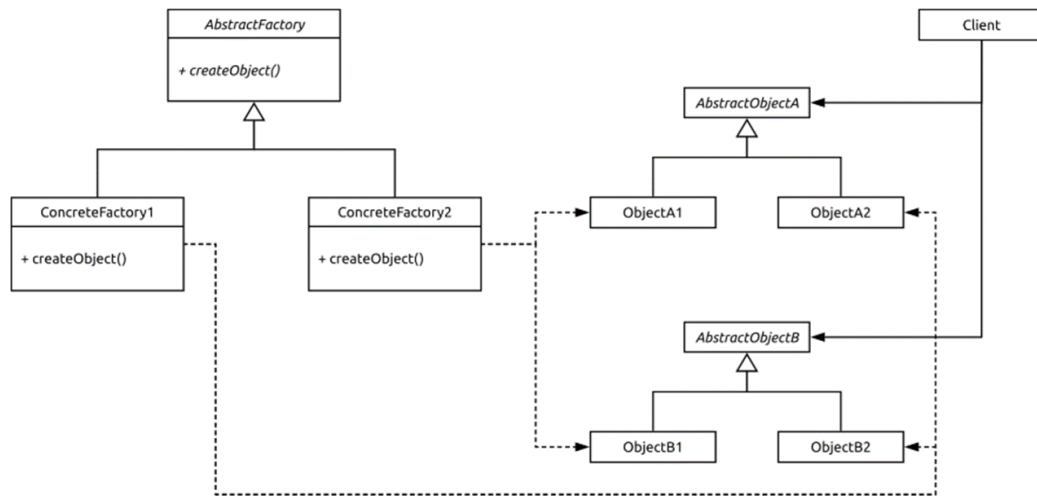
Tạo một biến static có kiểu là kiểu của lớp cần tạo singleton.

Tạo các private constructor để khởi tạo nội bộ và cấm tạo lập đối tượng của lớp từ bên ngoài lớp.

Tạo phương thức getInstance() để trả về tham chiếu tới đối tượng duy nhất của lớp.

Nguyên lý khởi tạo: ta có thể tạo ngay đối tượng của lớp khi khai báo biến static tham chiếu tới đối tượng của lớp hoặc khởi tạo lười biếng, tức khởi tạo đối tượng khi lần đầu tiên phương thức getInstance() được gọi. Để đảm bảo thread safe (an toàn luồng) khi tạo đối tượng từ đa luồng ta kiểm tra khóa 2 lần trước khi tạo đối

tượng.



Hình 5 Nguyên lý cấu tạo lớp singleton

2.5 FireBase

2.5.1 FireBase là gì

Firestore là một nền tảng để phát triển ứng dụng di động và trang web, bao gồm các API đơn giản và mạnh mẽ mà không cần backend hay server.

Firestore là dịch vụ cơ sở dữ liệu hoạt động trên nền tảng đám mây – cloud. Kèm theo đó là hệ thống máy chủ cực kỳ mạnh mẽ của Google. Chức năng chính là giúp người dùng lập trình ứng dụng bằng cách đơn giản hóa các thao tác với cơ sở dữ liệu.

Cụ thể là những giao diện lập trình ứng dụng API đơn giản. Mục đích nhằm tăng số lượng người dùng và thu lại nhiều lợi nhuận hơn.

Đặc biệt, còn là dịch vụ đa năng và bảo mật cực tốt. Firestore hỗ trợ cả hai nền tảng Android và IOS

Hiện tại Firestore có nhiều loại hình dịch vụ khác nhau, trong đó có 2 dịch vụ liên quan đến lưu trữ bao gồm Realtime database và Cloud Firestore.

Cả hai loại này đều dựa trên nền tảng cloud của Google. Dữ liệu được lưu trữ trên đây không phải dạng bảng gồm các bản ghi, hàng cột mà là NoSQL.

Ta có thể sử dụng các cách thức lưu trữ trực tuyến này cho các ứng dụng Android, iOS hoặc Web...

2.5.2 Cách thức hoạt động của Firebase

Khi đăng ký một tài khoản trên **Firebase** để tạo ứng dụng, bạn đã có một cơ sở dữ liệu thời gian thực. Dữ liệu bạn nhận được dưới dạng JSON. Đồng thời nó cũng luôn được đồng bộ thời gian thực đến mọi kết nối client.

Đối với các ứng dụng đa nền tảng, tất cả các client đều sử dụng cùng một cơ sở dữ liệu. Nó được tự động cập nhật dữ liệu mới nhất bất cứ khi nào các lập trình viên phát triển ứng dụng. Cuối cùng, tất cả các dữ liệu này được truyền qua kết nối an toàn SSL có bảo mật với chứng nhận 2048 bit.

Trong trường hợp bị mất mạng, dữ liệu được lưu lại ở local. Vì thế khi có mọi sự thay đổi nào đều được tự động cập nhật lên Server của **Firebase**. Bên cạnh đó, đối với các dữ liệu ở local cũ hơn với Server thì cũng tự động cập nhật để được dữ liệu mới nhất.

Hoạt động nổi bật của Firebase là xây dựng các bước xác thực người dùng bằng Email, Facebook, Twitter, GitHub, Google. Đồng thời cũng xác thực nặc danh cho các ứng dụng. Hoạt động xác thực có thể giúp thông tin cá nhân của người sử dụng được an toàn và đảm bảo không bị đánh cắp tài khoản.

2.5.3 Nhóm công cụ Firebase Develop & Test Your App

Nhóm công cụ này – hay còn gọi là công cụ phát triển và kiểm thử các ứng dụng được thiết kế, bao gồm các dịch vụ nổi bật sau:

Realtime Database: là dịch vụ lưu trữ và đồng bộ dữ liệu người dùng thời gian thực. Có hỗ trợ cho Android, IOS, Web, C++, Unity và Xamarin. Người dùng có thể lưu trữ và lấy dữ liệu từ máy chủ rất dễ dàng.

Crashlytics: là hệ thống theo dõi và lưu trữ thông tin lỗi của ứng dụng. Các thông tin lỗi sẽ được thu thập triệt để và trình bày hợp lý. Từ mỗi chu trình hoạt động đến khi xảy ra lỗi.

Cloud Firestore: là dịch vụ lưu trữ và đồng bộ dữ liệu giữa người dùng và thiết bị quy mô toàn cầu. Dịch vụ sử dụng NoSQL được lưu trữ trên hạ tầng cloud.

Authentication: là dịch vụ quản lý người dùng đơn giản và an toàn.

Authentication cung cấp nhiều phương pháp xác thực email và mật khẩu Google, Facebook.

Cloud Functions: là dịch vụ mở rộng ứng dụng bằng mã phụ trợ tùy chỉnh mà không cần quản lý và quy mô các máy chủ riêng.

Cloud Storage: là dịch vụ có khả năng lưu trữ và chia sẻ nội dung do người dùng tạo ra như hình ảnh, âm thanh và video với bộ nhớ mạnh, đơn giản và tiết kiệm chi phí được xây dựng cho quy mô của Google.

Hosting: Dịch Vụ Thuê Hosting giúp đơn giản hóa lưu trữ web với các công cụ thực hiện cụ thể có tính năng cao dành cho các trang web hiện đại.

Test Lab for Android; là công cụ tự động chạy thử và tùy chỉnh cho ứng dụng trên các thiết bị ảo và vật lý của Google cung cấp

Performance Monitoring: là dịch vụ có khả năng chẩn đoán các vấn đề xảy ra với hiệu suất ứng dụng.

2.5.4 Ưu Điểm

Tạo tài khoản và sử dụng dễ dàng

Tốc độ phát triển nhanh

Nhiều dịch vụ trong một nền tảng

Được cung cấp bởi Google

Tập trung vào phát triển giao diện người dùng

Firebase không có máy chủ

Học máy (Machine Learning)

Tạo lưu lượng truy cập

Theo dõi lỗi

Sao lưu

2.5.5 Hạn Chế

Không phải là mã nguồn mở

Người dùng không có quyền truy cập mã nguồn

Firebase không hoạt động ở nhiều quốc gia

Chỉ hoạt động với Cơ sở dữ liệu NoSQL

Truy vấn chậm

Không phải tất cả các dịch vụ Firebase đều miễn phí

Firebase khá đắt và giá không ổn định

Chỉ chạy trên Google Cloud

Thiếu Dedicated Servers và hợp đồng doanh nghiệp

Không cung cấp các API GraphQL

2.6 Giao Diện JetPack Compose

2.6.1 JetPack Compose là gì?

Android Jetpack là một tập hợp các components, tools giúp bạn nhanh chóng tạo ra các ứng dụng Android tuyệt vời. Các components này kết hợp giữa Support Library và Architecture Components.

2.6.2 Các thành phần của Android Jetpack

Có thể phân loại Android Jetpack thành 4 thành phần chính:

Foundation components (Ví dụ: ktx, appcompat, multidex, test)

Architecture components (Ví dụ: Data Binding, Lifecycles, ViewModel, LiveData, Room, Paging, Navigation, WorkManager)

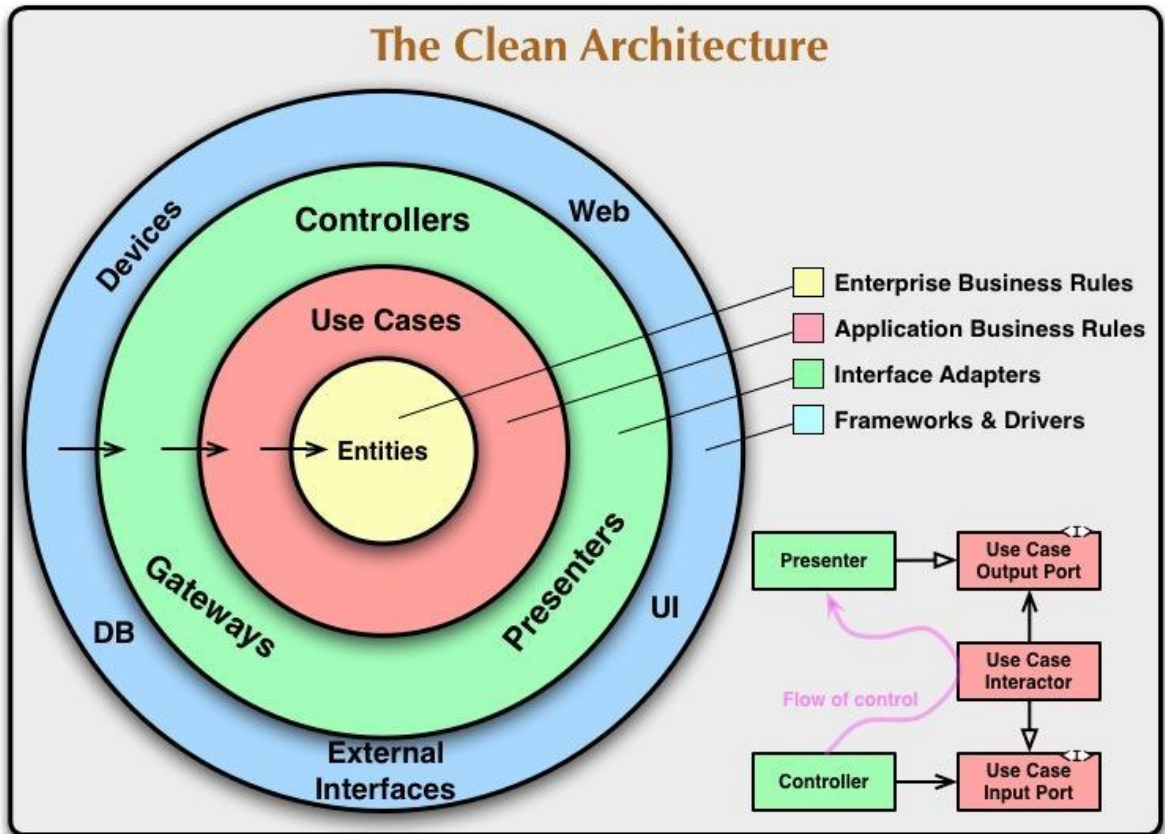
Behavior components (Ví dụ: Download manager, Media, Notifications, Permissions, Sharing, Slices)

UI components (Ví dụ: Animations, Auto, Emoji, Fragment, Layout, Palette, TV, Wear OS)

2.7 Clean Architecture

2.7.1 Clean Architecture là gì?

Clean Architecture Là một mẫu thiết kế phần mềm phân chia các mối quan tâm của ứng dụng thành các lớp khác nhau. Mục tiêu chính của kiến trúc sạch là làm cho mã nguồn dễ bảo trì, kiểm thử, và mở rộng.



Hình 6 The clean Architecture

2.7.2 Lợi ích của việc sử dụng Clean Architecture trong Jetpack Compose

Để bảo trì: Phân chia ứng dụng thành các lớp riêng biệt giúp hiểu và làm việc với mã nguồn dễ hơn. Điều này giúp dễ dàng thay đổi và sửa lỗi mà không ảnh hưởng đến toàn bộ mã nguồn.

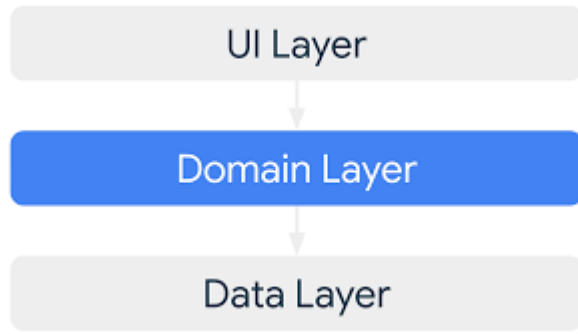
Tăng khả năng kiểm thử: Mỗi lớp có thể được kiểm thử độc lập, giúp đảm bảo mã nguồn hoạt động như mong đợi và giảm rủi ro lỗi.

Mở rộng dễ dàng: Kiến trúc sạch giúp dễ dàng thêm tính năng và chức năng mới mà không làm ảnh hưởng đến mã nguồn hiện tại.

Tính linh hoạt cao: Kiến trúc sạch cho phép thay đổi các triển khai mà không ảnh hưởng đến phần còn lại của mã nguồn.

Tái sử dụng mã nguồn: Kiến trúc sạch làm cho mã nguồn trở nên mô-đun, có thể tái sử dụng trong các dự án khác hoặc các phần khác của dự án.

Tách biệt các mối quan tâm: Dễ dàng phân tách các mối quan tâm của ứng dụng thành các lớp khác nhau, giúp dễ hiểu và làm việc với mã nguồn hơn.



Hình 7 UI Domain Data

CHƯƠNG 3: CÁC CHỨC NĂNG CỦA ĐỀ TÀI

3.1 Đăng ký, đăng nhập



Hình 8 Màn hình đăng ký đăng nhập

-Khi chạy máy ảo trang đăng ký, đăng nhập xuất hiện kèm với slider chạy tự động. Người dùng có thể chọn đăng ký nếu chưa có tài khoản hoặc tiến hành đăng nhập. Khi đăng ký mặc định phân quyền role là khách hàng.

3.1.1 Đăng nhập

10:13

BarBerShopApp

Đăng Nhập

Email

Mật khẩu

[quên mật khẩu](#)

Đăng Nhập

hoặc

Bạn chưa có tài khoản? [Đăng Ký](#)

Hình 9 Màn hình đăng nhập

- Cho phép người dùng đăng nhập tài khoản của mình nếu đã có tài khoản.

3.1.2 Đăng ký

10:24

BarBerShop

Create An Account

Họ và tên

Email

Số điện thoại

Ngày Sinh

Mật khẩu

☐ Bằng cách tiếp tục, bạn chấp nhận [Chính sách Bảo mật](#) và [Điều khoản Sử dụng](#) của chúng tôi

Đăng Ký

hoặc

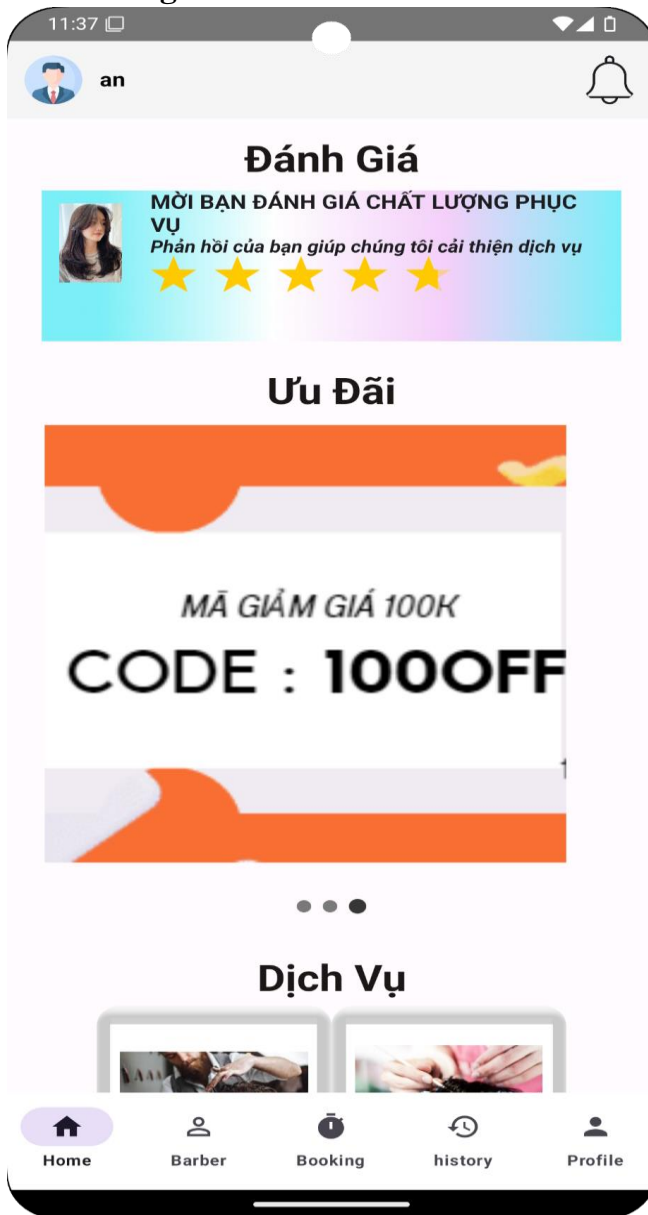
Bạn đã có tài khoản? [Đăng Nhập](#)

Hình 10 Màn hình Đăng ký tài khoản mới

- Nếu chưa có tài khoản có thể đăng ký một tài khoản mới. Tại đây sẽ tiến hành validation kiểm tra từng trường dữ liệu không cho phép nhập dữ liệu không đúng định dạng

3.2 Bên User

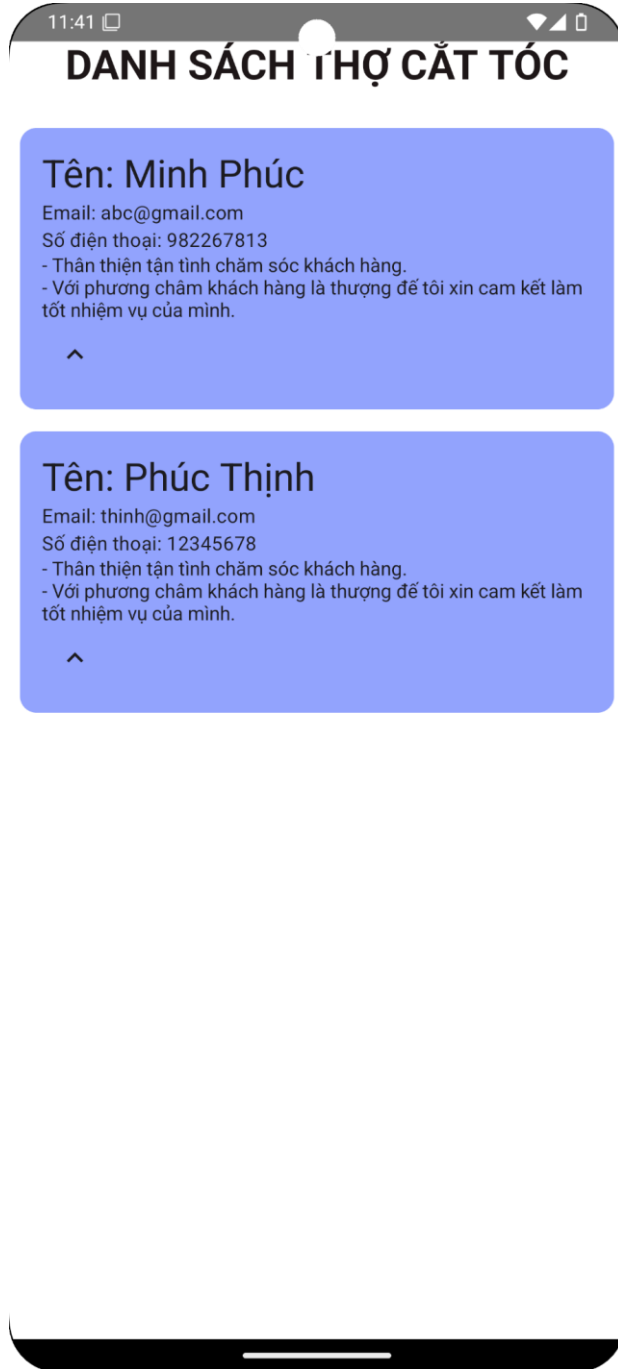
3.2.1 Trang Home



Hình 11 Màn hình trang Home

-Trang Home sẽ hiển thị các thông tin nổi bật như mã voucher, dịch vụ, đánh giá

3.2.3 Trang Barber



Hình 12 Màn hình trang danh sách thợ cắt tóc

-Trang Stylist sẽ hiển thị danh sách thợ hớt tóc mà tiệm đang có. Khách hàng có thể tham khảo để tiến hành đặt lịch và lựa chọn thợ phù hợp

3.2.4 Trang Booking

Hình 13 Màn hình Booking

-Khách hàng tiến hành lựa chọn các trường thông tin phù hợp sau đó tiến hành đặt lịch

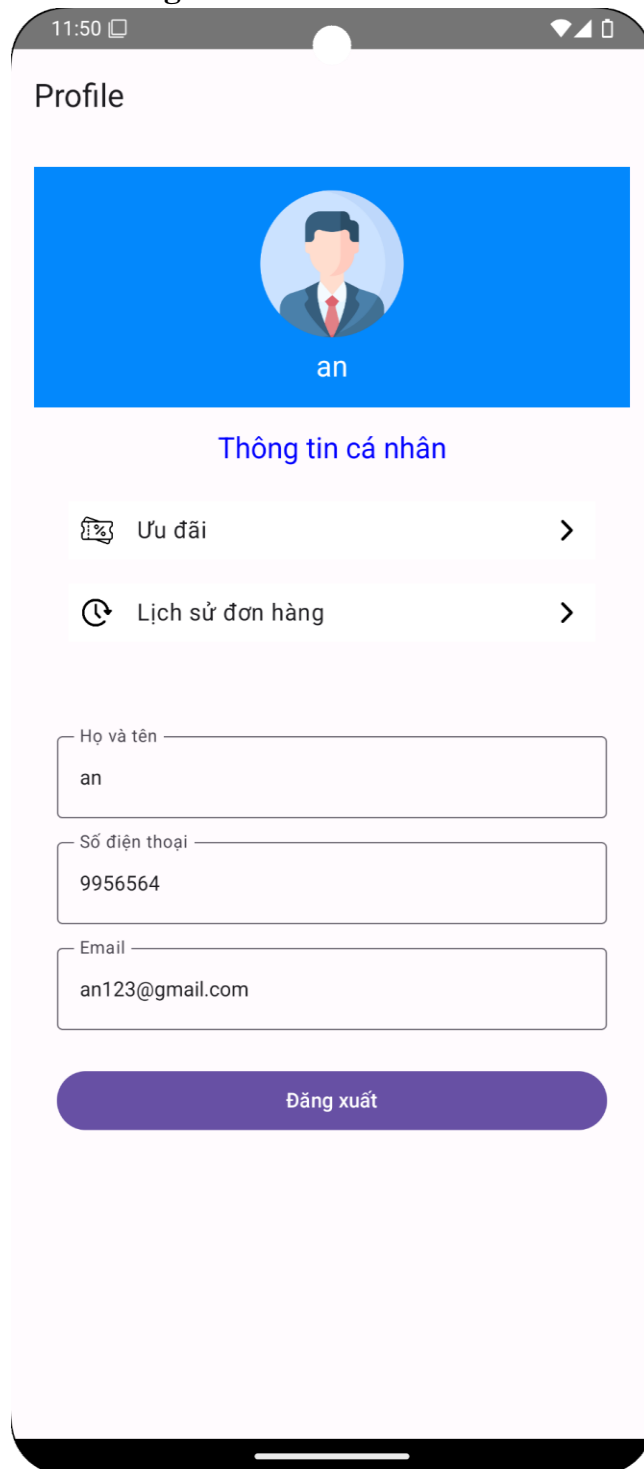
3.2.5 Trang History



Hình 14 Màn hình lịch sử đặt lịch

-Trang lịch sử đặt lịch sẽ cho biết khách hàng đã đặt lịch ngày hôm nào và có thể hủy lịch nếu ngày hiện tại muốn hủy lớn hơn 1 ngày so với ngày đặt.

3.2.6 Trang Profile



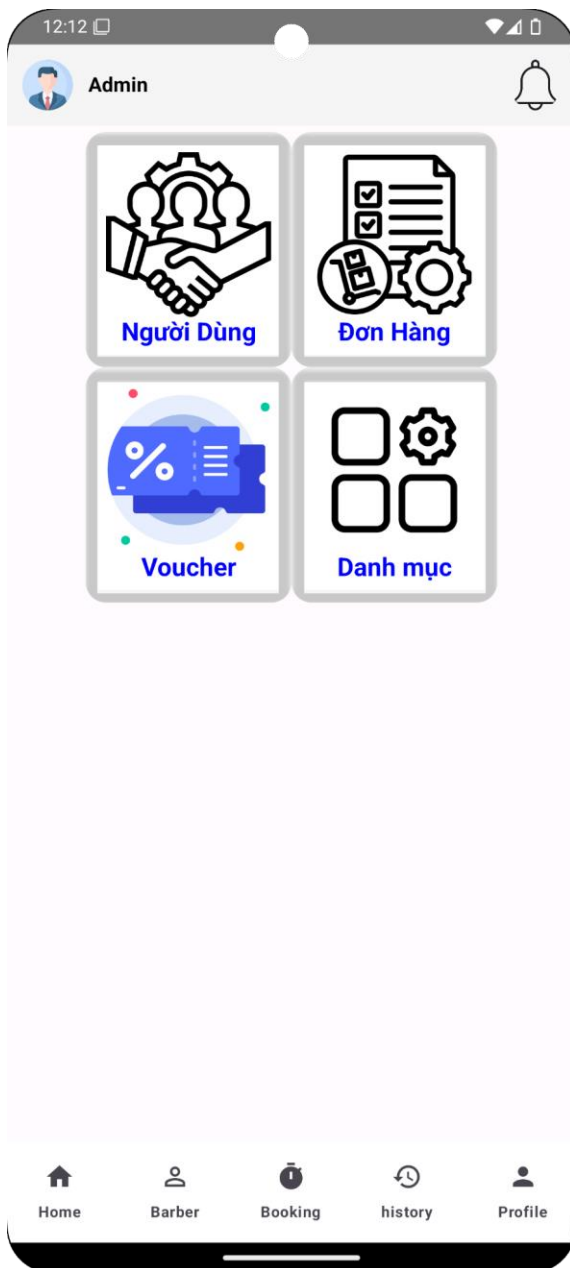
The image shows a mobile application interface for a user profile. At the top, there's a status bar with the time 11:50 and battery level. Below it, the title 'Profile' is displayed. A large blue rectangular area contains a circular profile picture of a man in a suit, with the text 'an' centered below it. Underneath the profile picture, the text 'Thông tin cá nhân' (Personal Information) is shown in blue. Below this, there are two menu items: 'Ưu đãi' (Offers) with a tag icon and 'Lịch sử đơn hàng' (Order History) with a clock icon, both followed by right-pointing chevrons. Further down, there are three input fields: 'Họ và tên' (Last name and first name) containing 'an', 'Số điện thoại' (Phone number) containing '9956564', and 'Email' containing 'an123@gmail.com'. At the bottom, there is a purple button labeled 'Đăng xuất' (Logout).

Hình 15 Màn hình profile

-Trang Profile cập nhập thông tin của khách hàng cũng như đăng xuất tài khoản

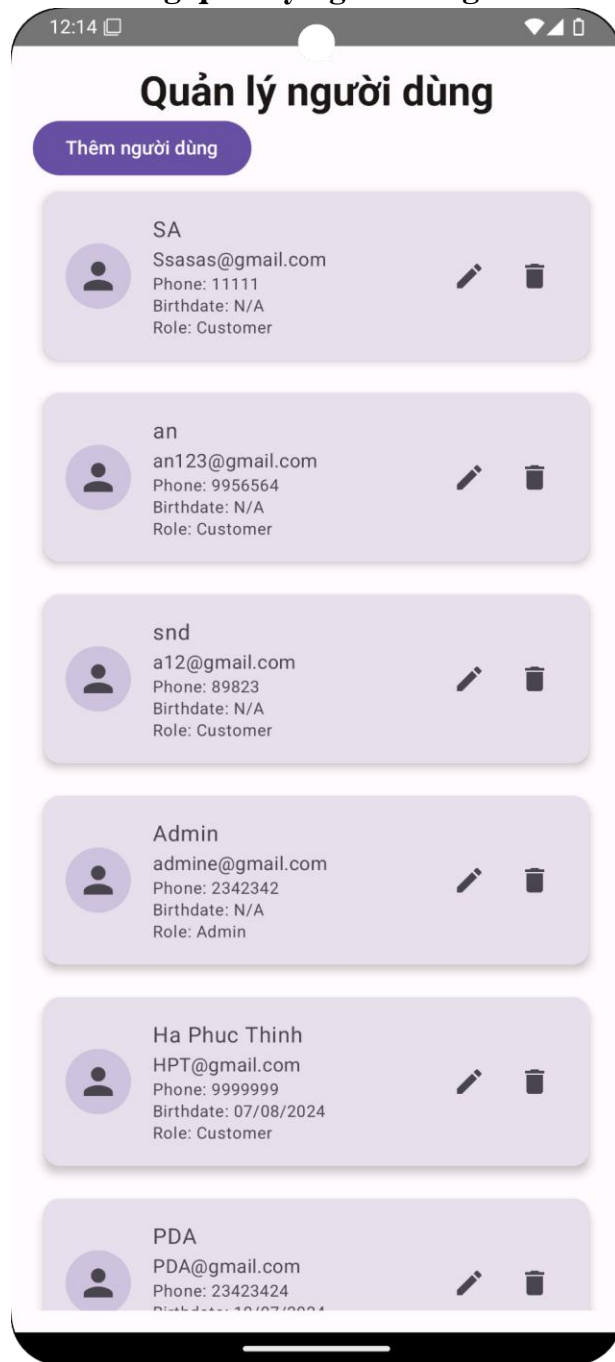
3.3 Bên Admin

3.3.1 Trang Home Admin



Hình 16 Màn hình trang HomeAdmin

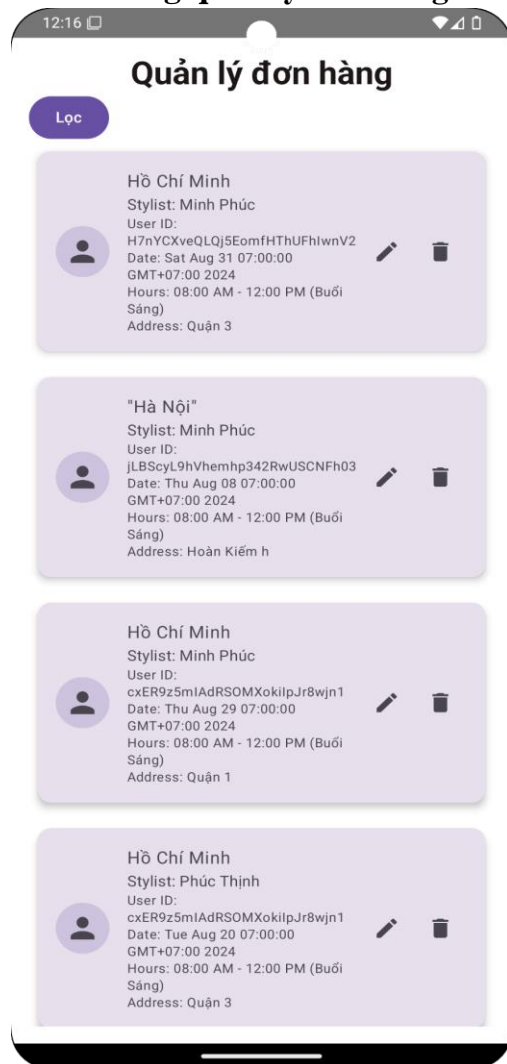
3.2.2 Trang quản lý người dùng



Hình 17 Màn hình trang quản lý người dùng

-Trang chủ admin có thể thêm sửa xóa nhân viên

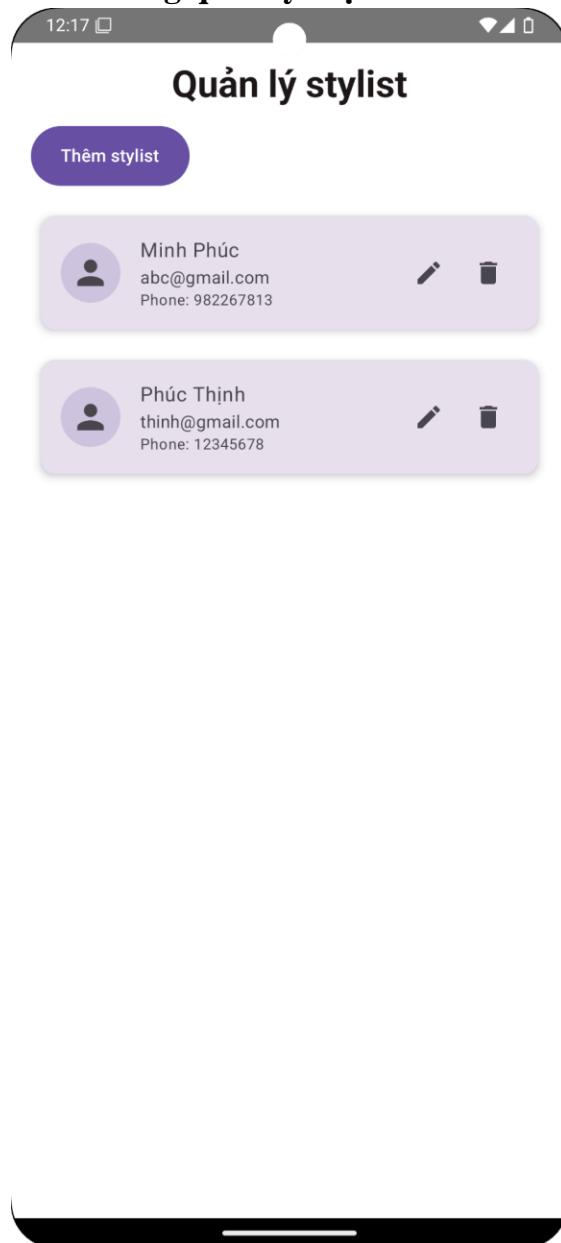
3.3.3 Trang quản lý đơn hàng



Hình 18 Màn hình quản lý đơn hàng

-Trang admin có thể thêm sửa xóa, tìm kiếm đơn hàng

3.3.4 Trang quản lý thợ cắt tóc



Hình 19 Màn hình quản lý nhân viên

-Trang admin có thể thêm sửa xóa nhân viên

Chương 4 : KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết quả đạt được

1.1 Ưu điểm

- Thể hiện đầy đủ chức năng cơ bản của một app đặt lịch hớt tóc
- Giao diện dễ nhìn sinh động

1.2 Nhược điểm

- Về dữ liệu: Dữ liệu chưa được đầy đủ.
- Về chức năng:
 - + Chưa có thanh toán online
 - + chưa có dịch vụ chat giữa user và admin

2. Hướng phát triển

Đề tài cần phát triển thêm các hướng sau:

- + Trang thanh toán online cho khách hàng đặt lịch
- + Dịch vụ trao đổi trực tiếp giữa user và admin
- + upload lên nền tảng thứ 3 để

3. Tài liệu tham khảo

[Công cụ dành cho nhà phát triển ứng dụng Android – Nhà phát triển Android | Android Developers](#)

[Firebase | Google's Mobile and Web App Development Platform](#)

<https://firebase.google.com/docs>

<https://dashwave.io/blog/android-architecture-patterns/>

<https://developer.android.com/compose>

BẢNG PHÂN CÔNG CÔNG VIỆC

HỌ VÀ TÊN	CÁC CÔNG VIỆC THỰC HIỆN	ĐÁNH GIÁ (10)
Đặng Minh Phúc	Đăng ký đăng nhập, thiết kế giao diện app	10
Phạm Dương An	Quản lý admin	10
Hà Phúc Thịnh	Quản lý User	10