Ứng dụng quản lý booking dành cho các barber/hairdresser shop

vấn đề

- Các salon thường gặp khó khăn trong việc tránh tình trạng lịch trùng, nhầm lẫn và thiếu sự tương tác với khách hàng.

- Để khách hàng đợi lâu vì phải dành thời gian phục vụ khách đến trước đó

các giải pháp hiện có:

- khách hàng nhắn trước với shop trên các nền tảng mạng xã hội(facebook) để đặt lịch trước -> thuận tiện cho khách hàng, khó khăn trong việc quản lý cho shop

- khách hàng tải app đặc thù của shop và chỉ sử dụng để đặt lịch cho shop đó -> hao phí tài nguyên

giải pháp của app:

Khách hàng nhắn tin trên facebook, app sẽ liên kết với facebook page nhận và gửi tin nhắn và xử lý đặt lịch, cung cấp 1 giao diện chung cho chủ shop

shop sẽ cập nhật lại lịch trên app, từ đó thông qua facebook api sẽ gửi các option đến cho khách hàng lựa chọn

giới hạn:

- app chỉ sử dụng để quản lý đặt lịch , không có chấm công cho các thợ, chỉ có 1 tài khoản của chủ quán (tài khoản page liên kết với app )

- không có app riêng cho khách hàng để đặt lịch

hướng mở rộng:

- tích hợp thêm nhiều nền tảng mạng xã hội khác

- thêm chấm công

- xử lý văn bản

dự kiến cách hoạt động của app:

để nhận tin nhắn từ client:  
 webhook + firebase fcm

hoặc

request thủ công mỗi 2-5 phút 1 lần

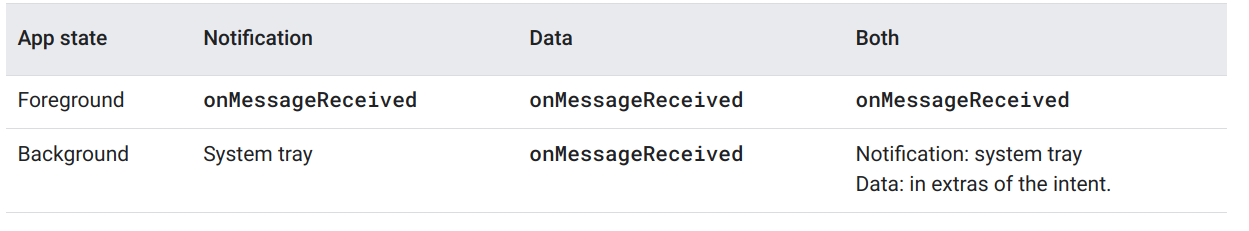
facebook api  
 GET /{page-id}/conversations?access\_token={page-access-token}

nên lưu data đặt lịch ở local hay remote?

lấy Page Access Token của facebook page? facebook sdk hoặc hướng dẫn shop lấy trực tiếp

# fcm: 2 types (https://firebase.google.com/docs/cloud-messaging/android/receive#handling\_messages

)



để gửi notification payload: firebase notification compose whatever trên console.firebase

để gửi data payload: tạo server riêng, có firebase sdk hoặc dùng http v1 api ( POST [https://fcm.googleapis.com/v1/{parent=projects/\*}/messages:send](https://fcm.googleapis.com/v1/%7Bparent=projects/*%7D/messages:send))

see more: https://firebase.google.com/docs/reference/fcm/rest/v1/projects.messages/send

→~~taoj server, dungf httpv1api,~~ dùng firebase sdk trên server

todo: login bangwf facebook sdk

<https://developers.facebook.com/docs/facebook-login/facebook-login-for-business/>

<https://developers.facebook.com/docs/permissions/#pages_manage_metadata>

VỀ LUỒNG XÁC THỰC người dùng để liên kết đến facebok CỦA APP:

1.user đăng nhập bằng button của facebook sdk, chuyển hướng đến facebook permission request

2 user cấp quyền cho app, “page\_show\_list”,”page\_messaging” và lấy được USER\_ACCESS\_TOKEN

2.1 user hoàn thành luồng oauth, quay trở về app, sau luồng này app có dduocj USER\_ACESS\_TOKEN sẽ đưa FirebaseAuth (import com.google.firebase.auth.FirebaseAuth;

) để quản lý đăng nhập cho app

3 Sử dụng graph api với USER\_ACCESS\_TOKEN lấy danh sách các page user có quyền quản trị, user chọn page-> lấy được PAGE\_ACCESS\_TOKEN

4 đăng ký đến messenger webhook cho page bằng PAGE\_ACCESS\_TOKEN thông qua [subscribed\_apps](https://developers.facebook.com/docs/graph-api/reference/page/subscribed_apps) endpoint từ đó liên kết được page với app, server của app qua đó sẽ nhận mọi tin nhắn gửi đến page

ref:

<https://developers.facebook.com/docs/messenger-platform/webhooks>

https://developers.facebook.com/docs/messenger-platform/reference/webhook-events/messages

About Thống kê functionality::

//*Firestore is really not a great solution for data analysis because of how expensive the reads are. -* anon

* also talk abit about firebase rule and security, scenario:

hiện tại quyền của tken firebase sdk được set allow \* => query được tất cả các shop document trong app collection

* talk about setting up firebase sdk? facebook sdk in an app? how about go the raw dogging api call way?=> reduce the size of the app, talk about android manifest, userpermission, gradle,...

Todo:

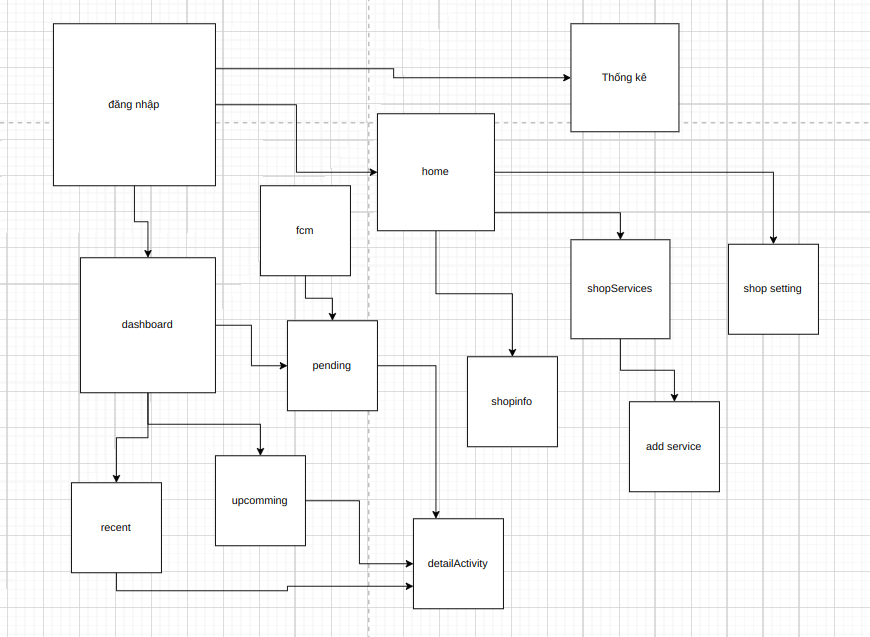
* Khi khách đến quán:

khách gửi id appointment, lookup by hand in app, xác nhận

* idea: Gửi khách id qr generated, -> khách đến quán, scan qr khách đưa, jump into appointment detail activity

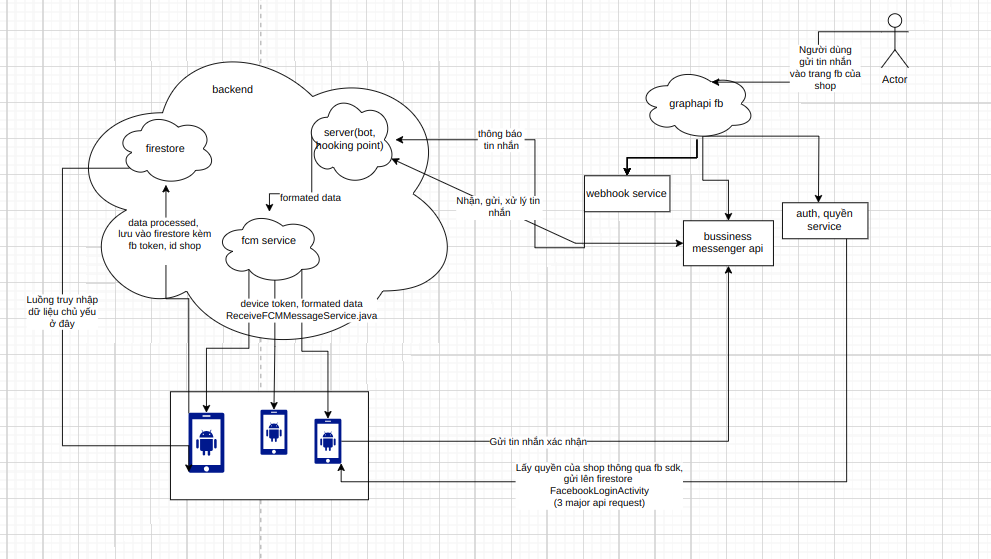
Main activity: lấy template default android, 3 fragments: home, dashboard, thống kê/lich bieu.

Roadmap luồng hoạt động app:



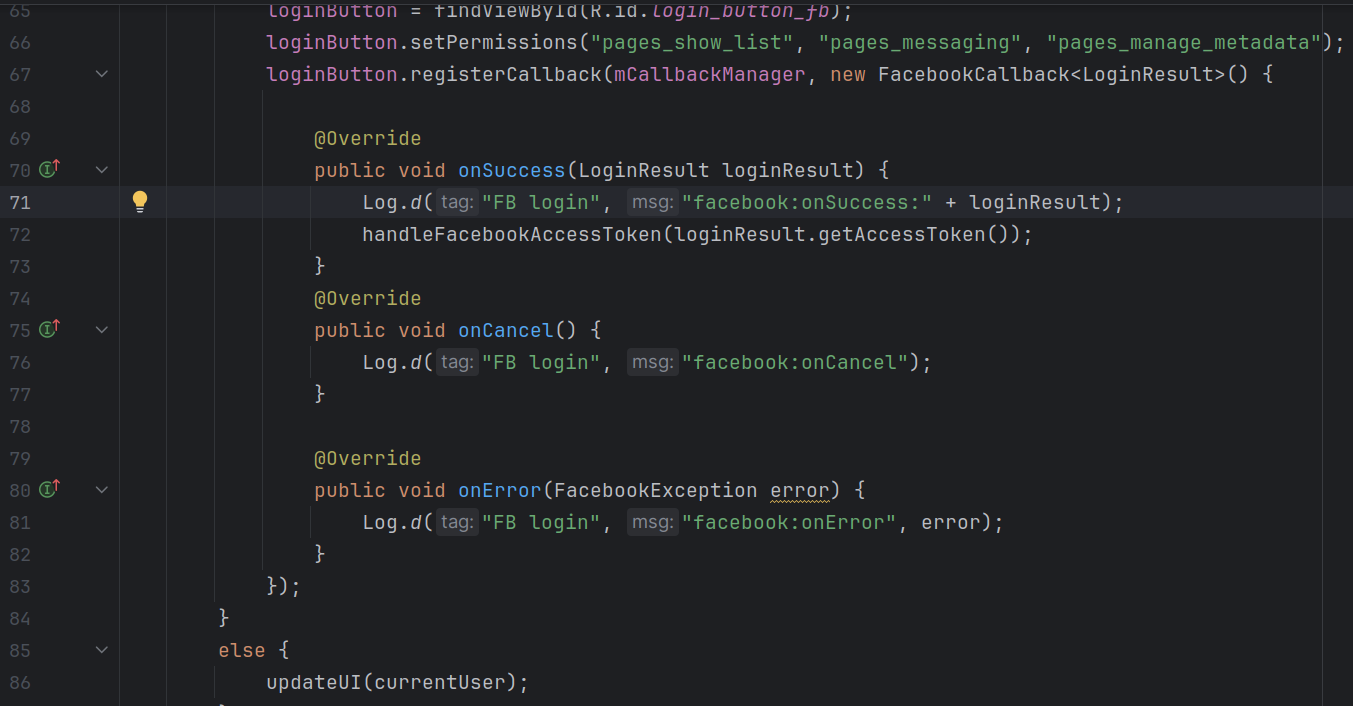
Tổng quan kiến trúc app:

<https://drive.google.com/drive/folders/0ABun9N53JsQRUk9PVA>

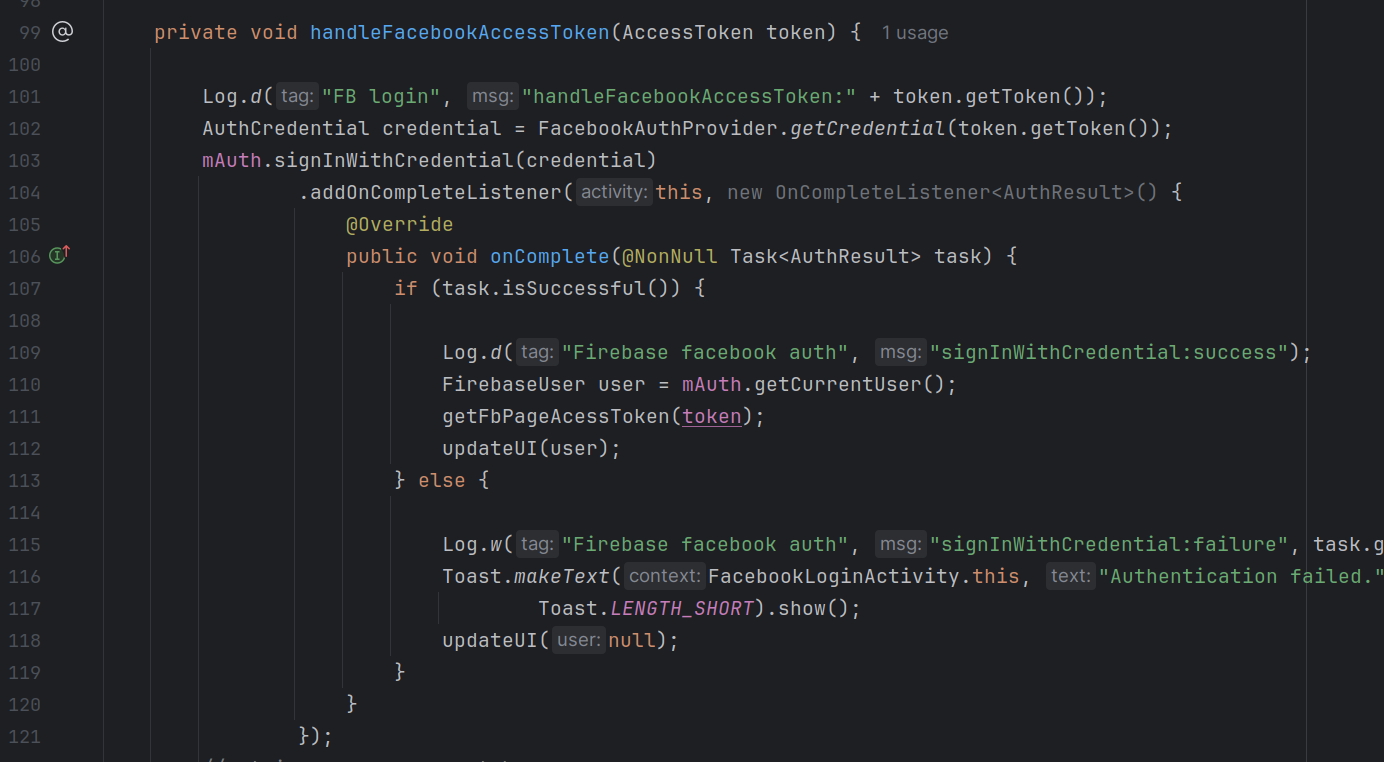


code for LUỒNG XÁC THỰC:

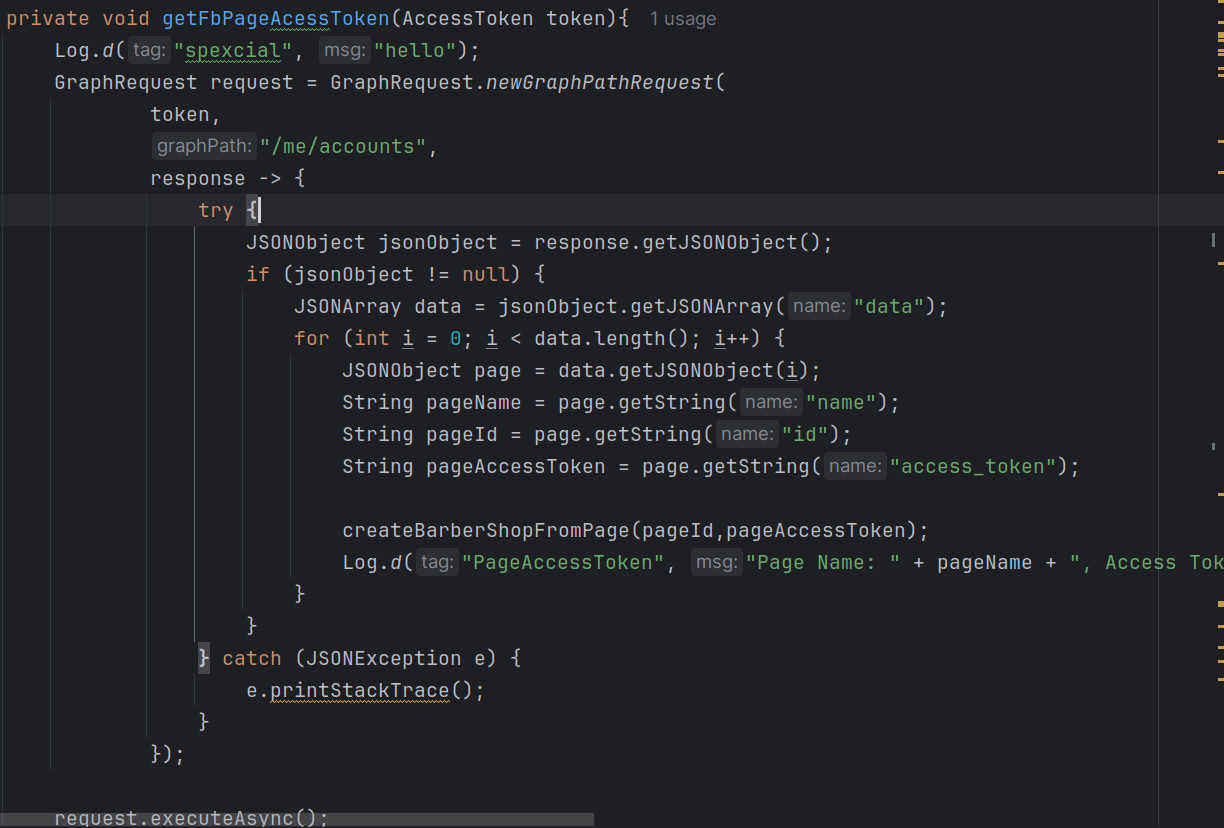
FacebookLoginActivity.java



=> use default login button from fb sdk, if sucess process to next stage handleFacebookAccessToken() with the retrived user access token



=> another nested callback, lưu token vào firestore(mauth firebase) trước sau đó call api 1 lần nữa lấy các page mà token (user ) này quản lý trong hàm getFbPageAcessToken();

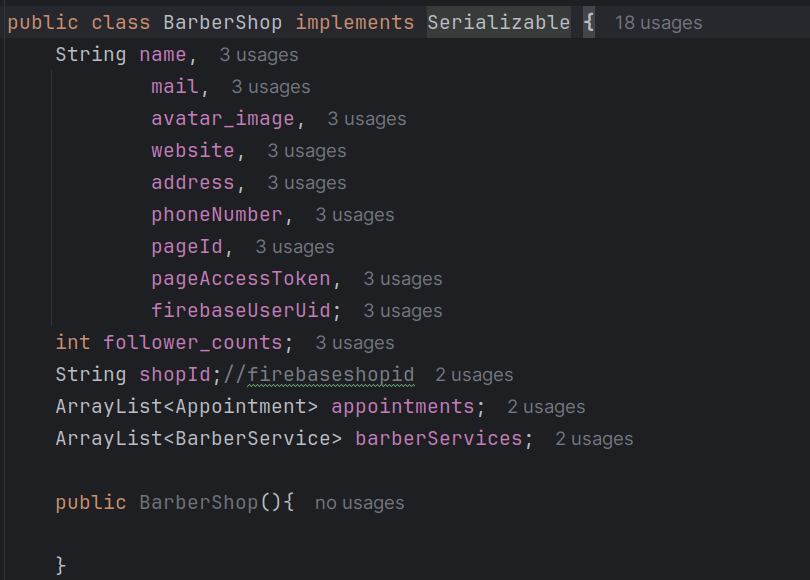


=> dùng graph api thực hiện 1 request đến endpoint /me/accounts, parse the return data to get page id and page token ⇒ finally got the page token to function the app.

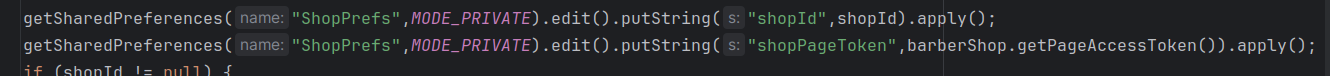
reference:<https://developers.facebook.com/docs/graph-api/reference/user/accounts/>

* createBarberShopFromPage() method request laays thông tin mới từ fb, tạo 1 object BarbẻrShop mới từ json trả về và lưu thông tin lên firestore





Ngoài ra có sử dụng SharedPreferences để lưu token shop locally=> some service require it



Khiếm khuyết: 4 request, 3 nested callback hell

giao diện đăng nhập:

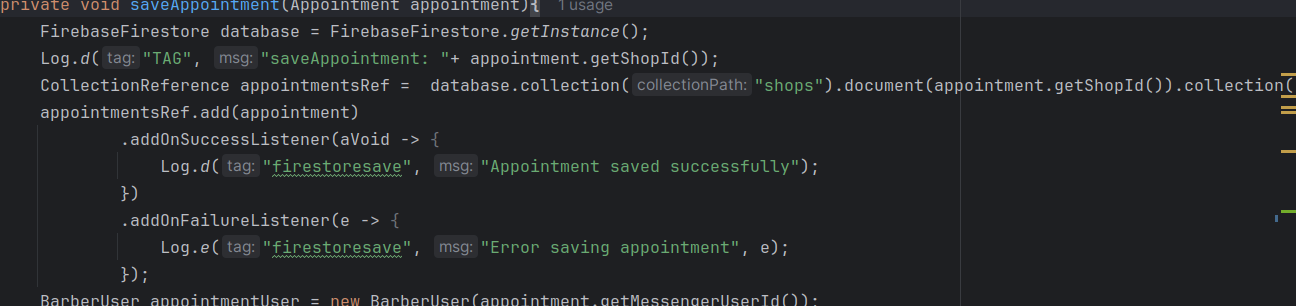


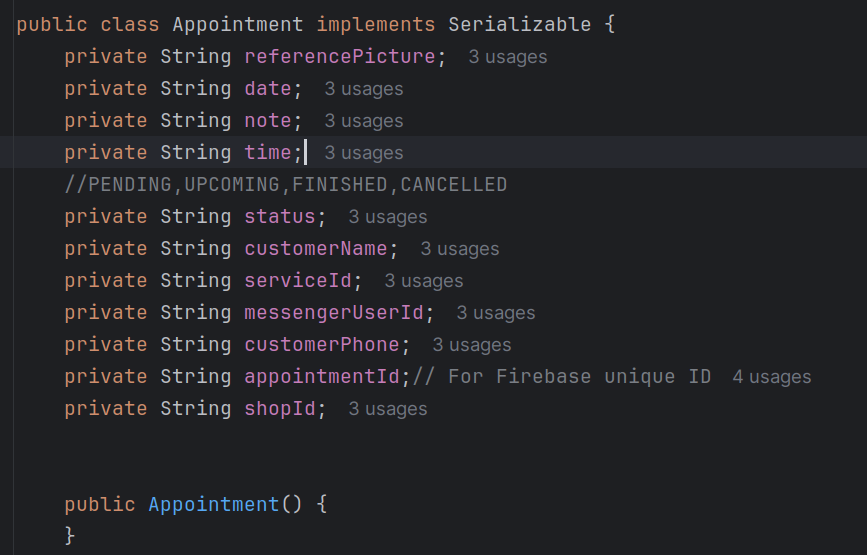
fcm:

1 service class extends FirebaseMessagingService để nhận tb từ fcm[,2 type of message](#_fc74x7kfisba)

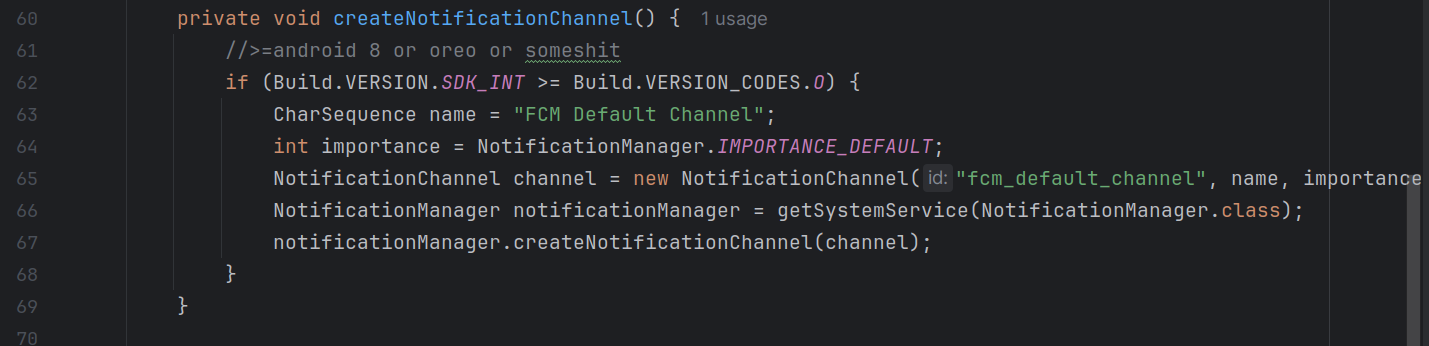


cấu trúc data payload fcm được gửi từ server(show code nếu cần thiết,), sau khi nhận được data, tạo appointment mới và chuyển vào phương thức saveApointment() to store in firebase với status appointment mặc định là PENDING





about the notification: từ android 8.0 trở đi các app bắt buộc phải có notification channel to work in background so that the hacky way to get over it is literally create notification at the first time open the app

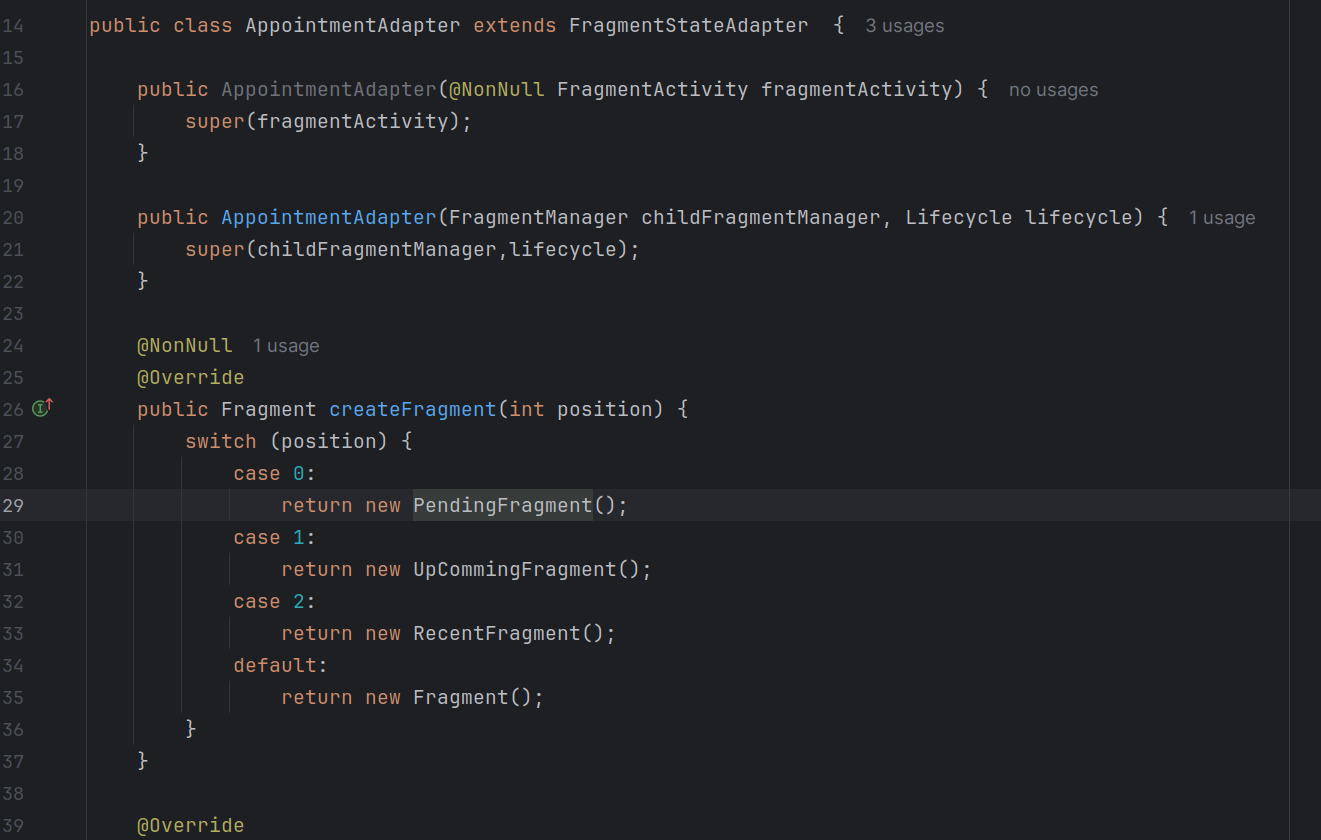


### **DASHBOARD:**

dashboard sử dụng tablayout và viewpager2 để chia các fragment appointment. Initialize dashboard and inflate view:



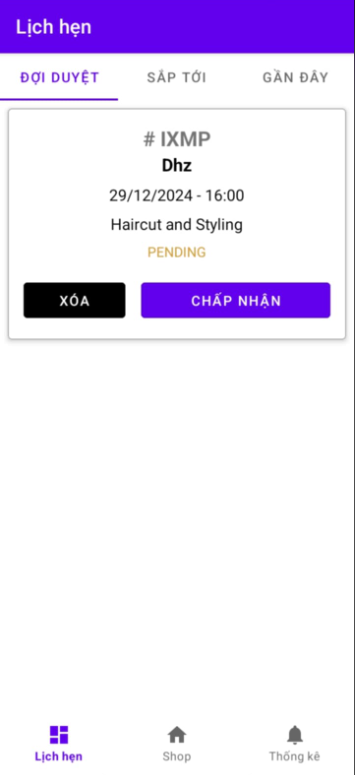
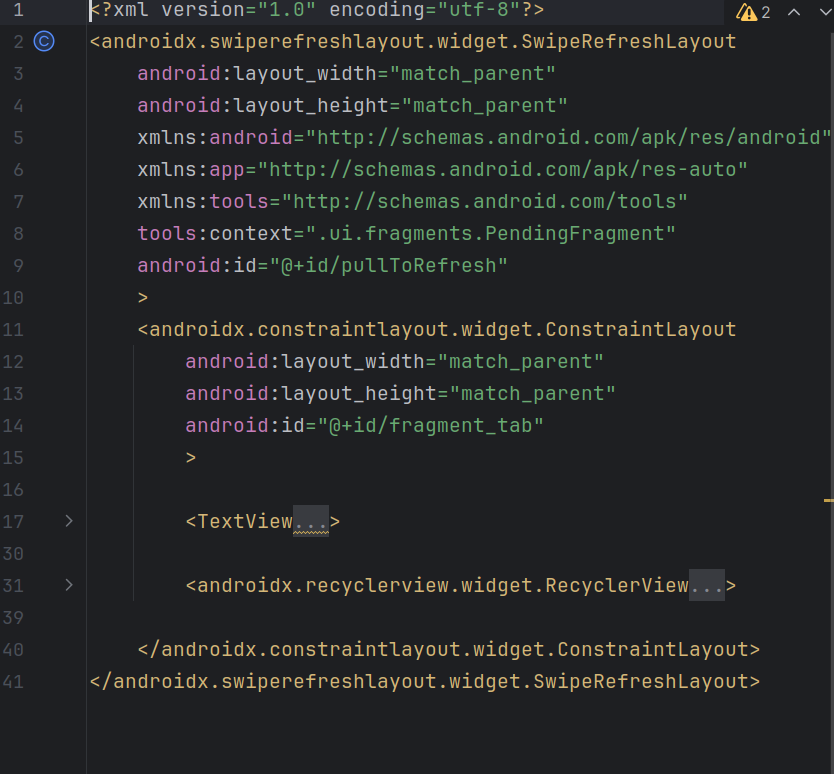
ở đây dashboard fragment host 1 fragment con khác nên phải sử dụng getChildFragmentManager() to access the FragmentManager con của *this* để sự dụng trong adapter



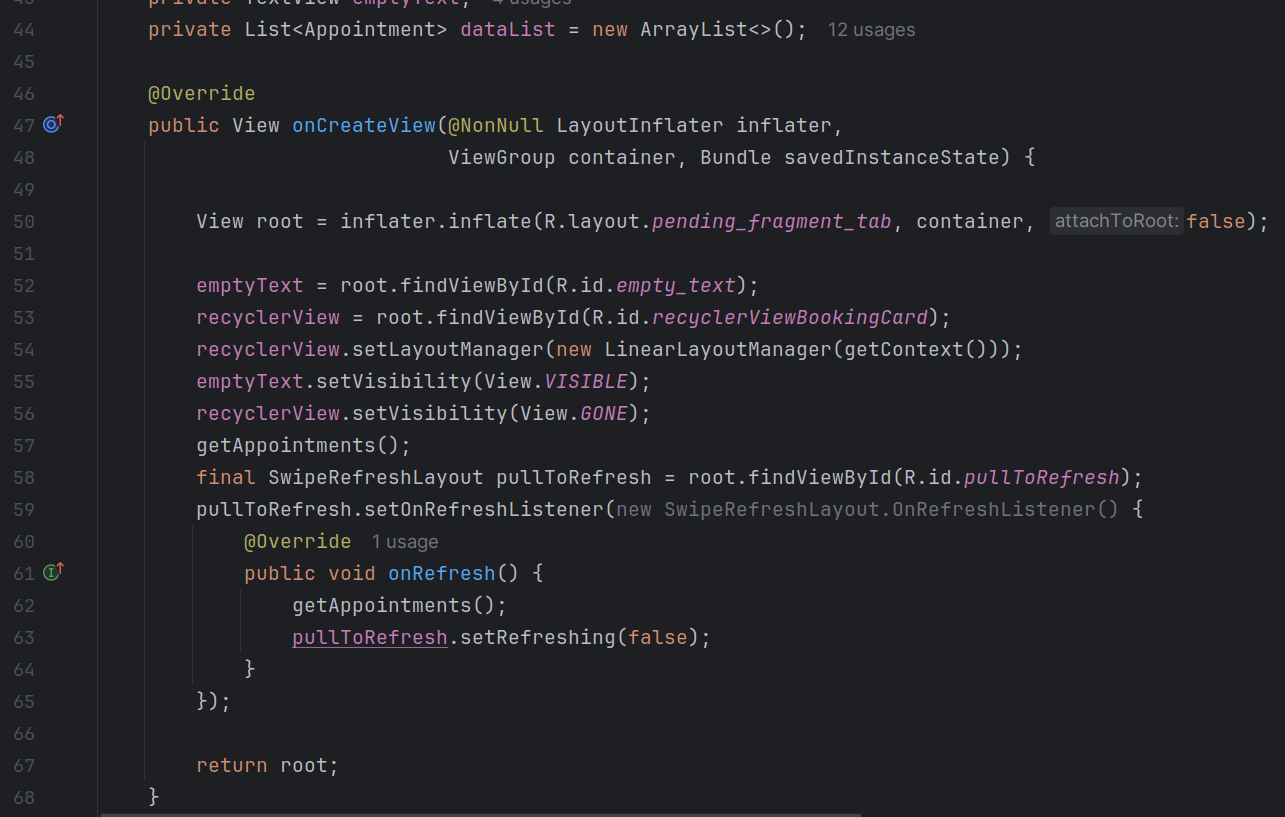
1 adapter đơn giản trả về các fragment khác tùy thuộc vào position trên tablayout

#### **Pending fragment:**

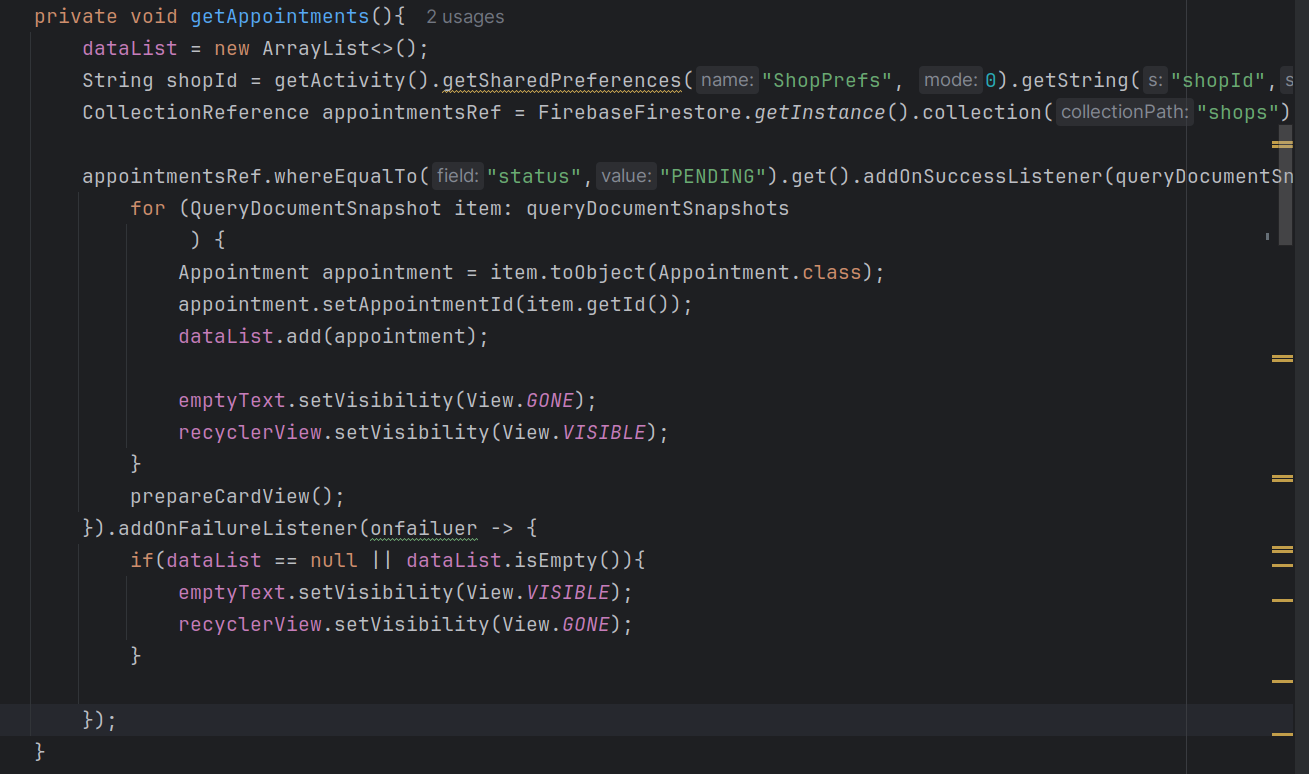
* layout pending fragment sử dụng SwipeRefreshLayout và RecyclerView:
* các event trong fragment: user lướt trái, phải, button từ card của recylerview, kéo xuống để refresh, gửi tin nhắn đến fb sau khi chấp nhận appoiment



pendingFragment onCreateView():



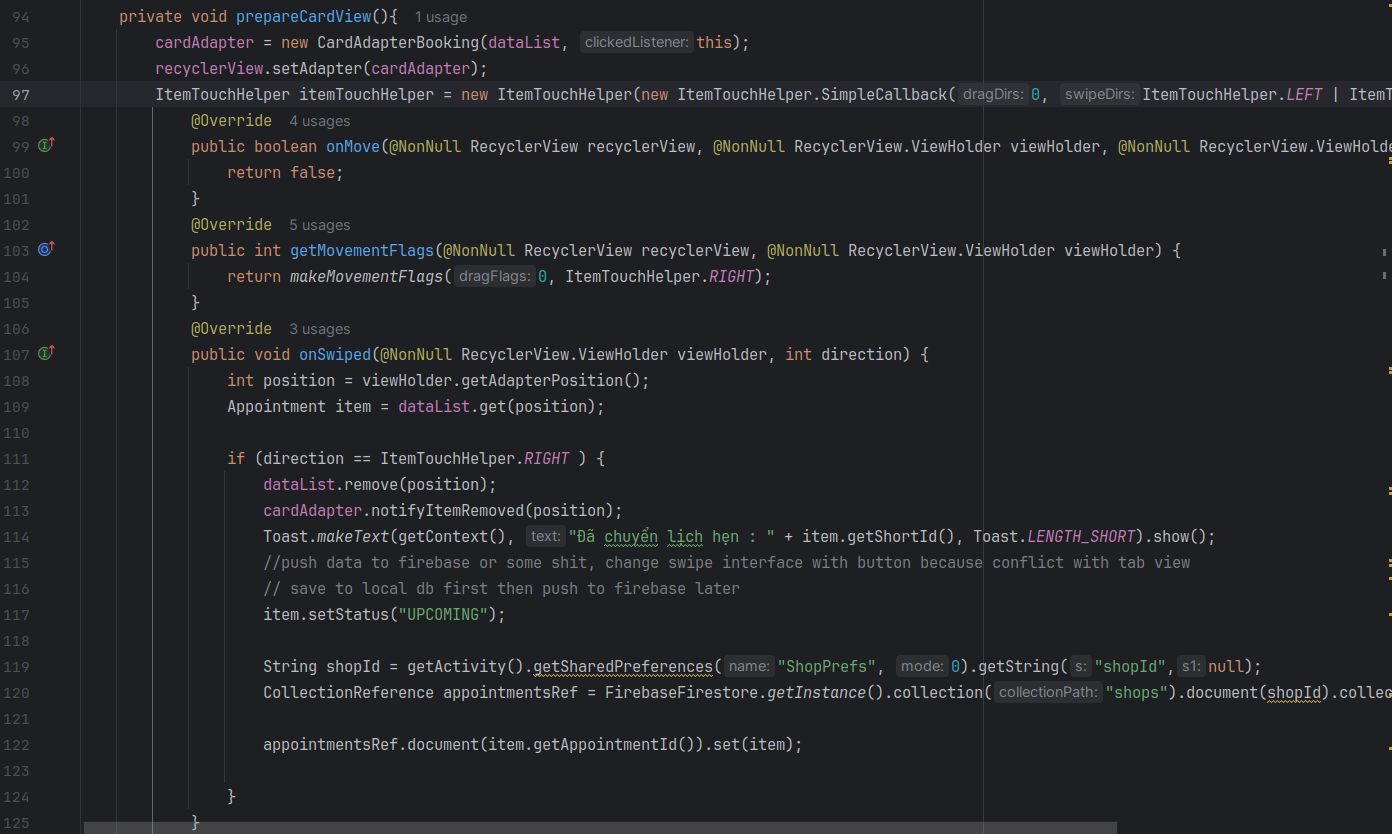
getAppointment():



retrive alll appointment that has PENDING status to from firestore and add it to datalist field in global class,

the path to collection is firestore được lấy từ getSharedPreference trong lần đăng nhập đầu tiên => đảm bảo chỉ được retrieve collection từ chính shop đó , prepareCardView() được gọi trong callback khi và chỉ khi lấy data từ firebase thành công và fill in dataList, must refactor this lines because possible causing dataList to be null, or should just pass the dataList to prepareCardView()=> this is the matter of code design choice

prepareCardView():



Trong prepareCardView set up ItemTouchHelper và taọ adapter mới cho recycler view in xml file

constructor cardAdapterBooking nhận 2 tham số: List<Appointment> và OnItemClickListener interface với 3 method:

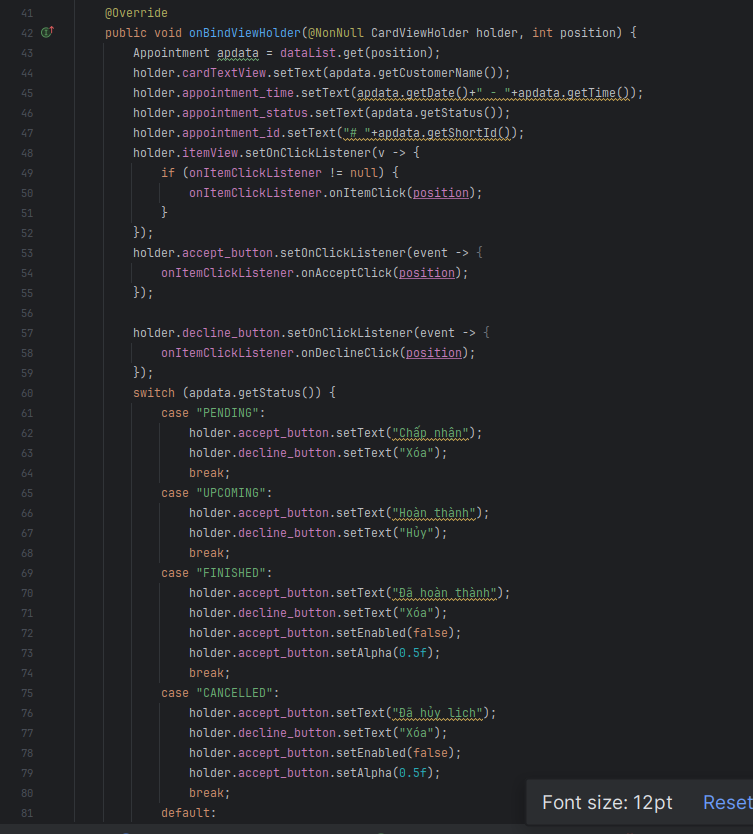
void onItemClick(int position);

void onDeclineClick(int position);

void onAcceptClick(int position);

funny things is the onDeclineClick(int position) and onAcceptClick(int position); is passed in the button in the cardCallback, so it’s callback of the callback of the callback được handle in onBindViewHolder():

trong bindviewholder cungx xử lý tùy vào item mà nó bind mà thay đổi text của button như trong hình:



trở về pendingFragment implement OnItemClickListener của cardviewholder

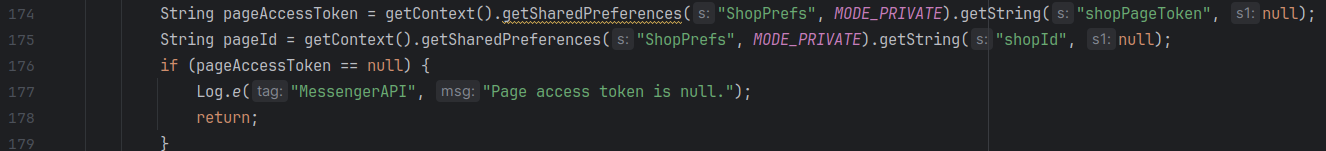
void onItemClick(int position): mở detail activity;

void onDeclineClick(int position): set Status thành cancelled và cập nhật firestore

void onAcceptClick(int position): set Status thành UPCOMING và cập nhật firestore, gửi tin nhắn đến khách hàng thông qua fbsdk(ở đây có thể handle event by calling to our own server and let the bot handle it but 1/it’s the android course, do in android as much as possible,2/ minimize complexity, since we already have fb sdk, we will just use it anyway instead of setup to make a single https call to our server, that will be for future development).

để gửi tin nhắn:

lấy page access token:



tạo message json object:



{

"messaging\_type": "RESPONSE",

"recipient": {

"id": "{user\_id}"

},

"message": {

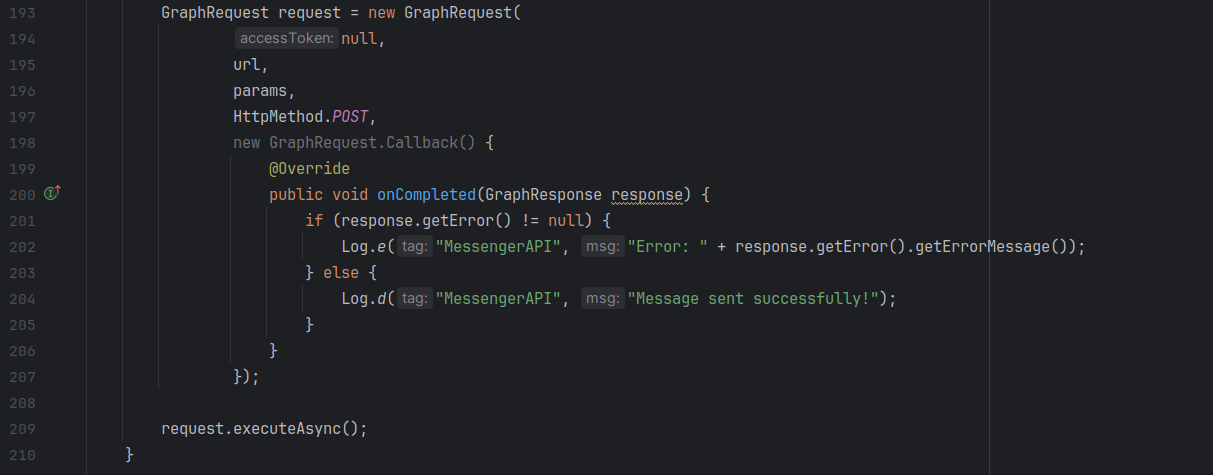
"text": "Hello, this is a message from the page!"

}

}

references:<https://developers.facebook.com/docs/messenger-platform/reference/send-api/>

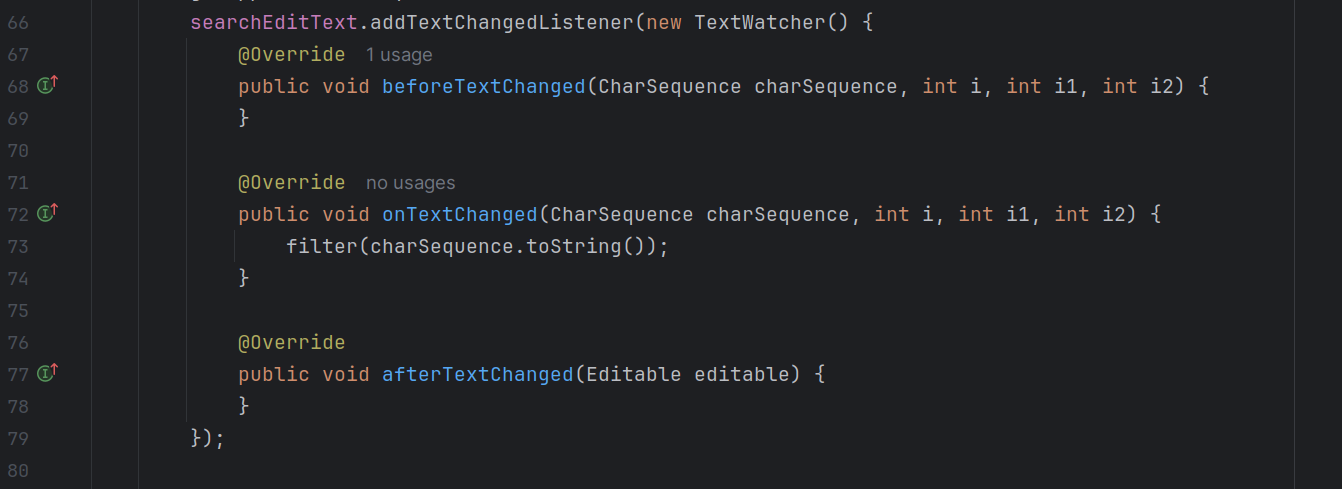
send request with fb sdk:



#### **Upcoming fragment:**

Tương tự như pendingfragment nhưng có search filter lọc ra appointment bằng tên, sdt, appoiment id, sử dụng edittext và add event listener to it. để filter data:

sử dụng 2 List<Appointment> dataList và RawDatalist, sau mỗi lần fetch từ database copy và RawDatalist lưu trữ tất cả data được fetch, filter sử dụng để tìm kiếm trong RawDatalist và thêm vào dataList, cardadapter được set với dataList





#### **Recent fragment:**

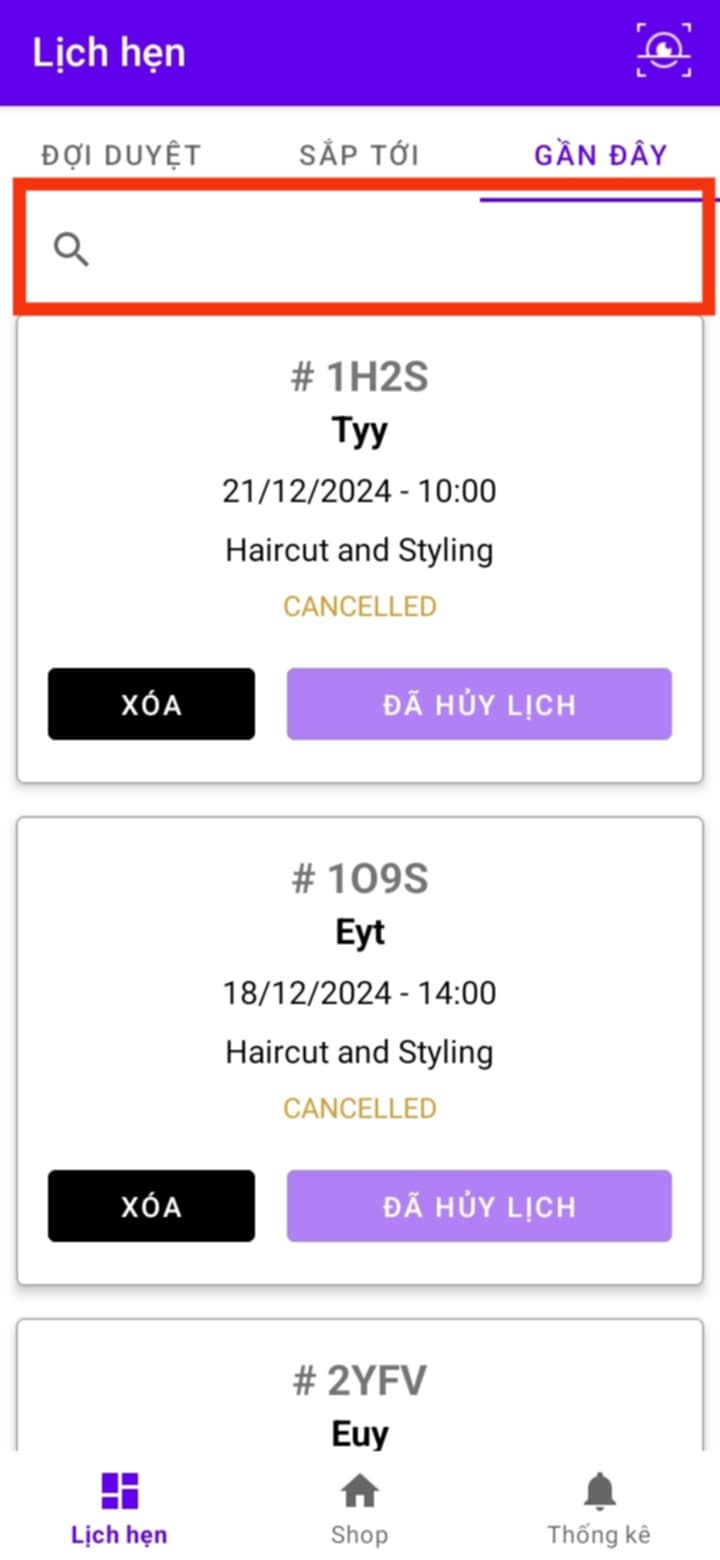
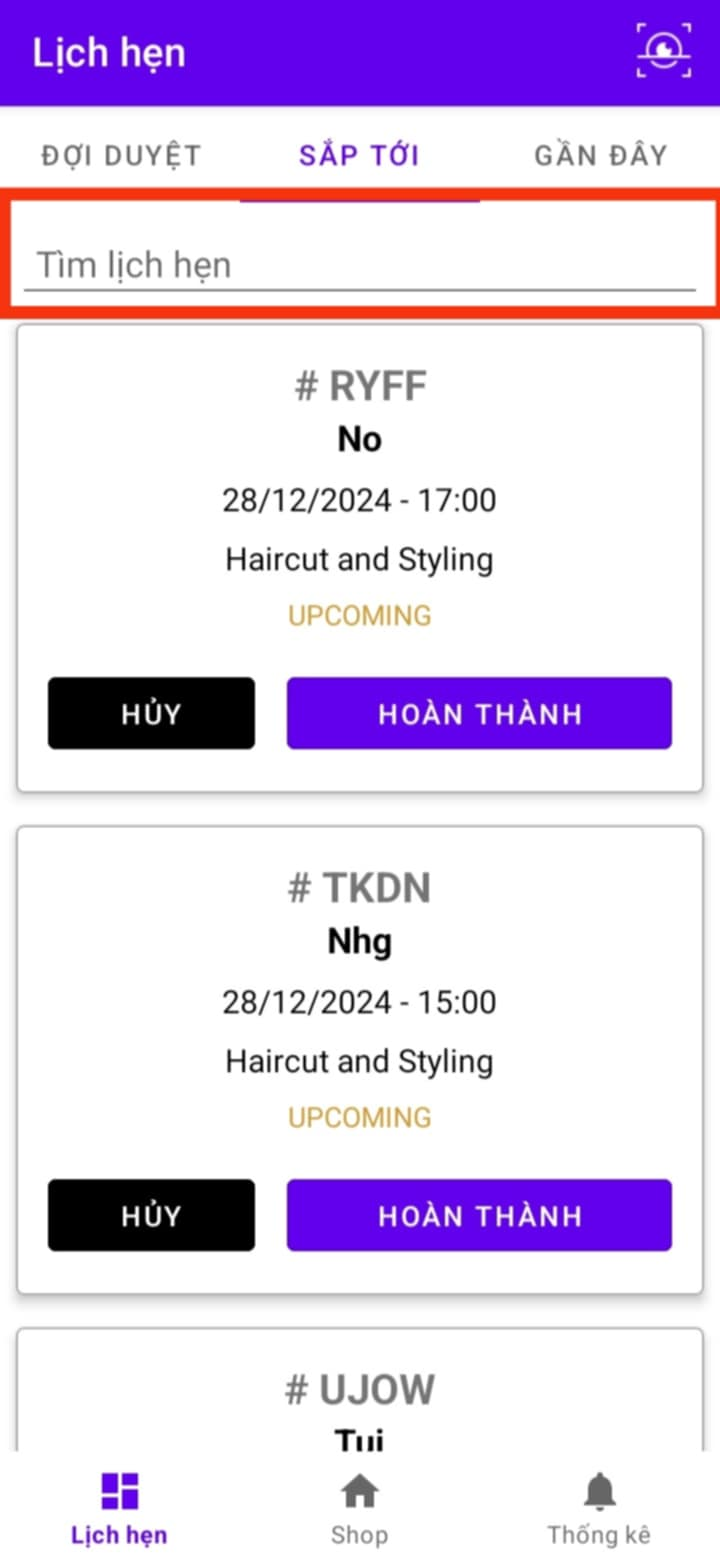
tương tự như upcomming fragment nhưng filter được cài đặt trong cardAdapter riêng(RecentCardAdapter.java)

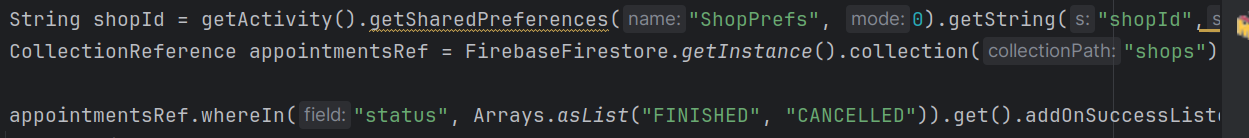
-sử dụng searchview thay vì editext

-recent fragement lấy những appointment that has status finsished and cancelled from firestore and listen to setOnQueryTextListener for fast search query event

why do it in cardAdapter instead of outside like upcomming fragment?

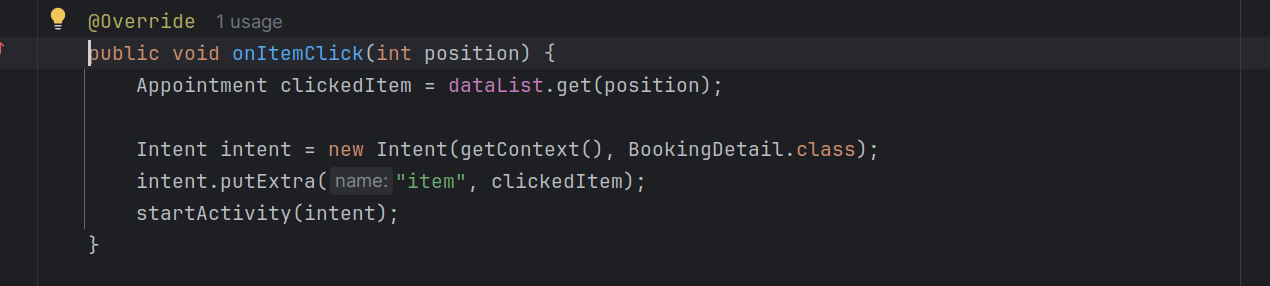
because the search filter is implemented in this fragment first thus have to remodify the xml from constraint layout to linear layout and make a new separate adapter for the card, therefore later when implement search filter in upcomming fragment take an easy approach is filter outside the adapter and editext for quick adapt and too lazy to change this fragment again(see image):





### **DetailActivity**

Activity luôn được mở bằng 1 Appointment class truyền vào mỗi khi init intent:



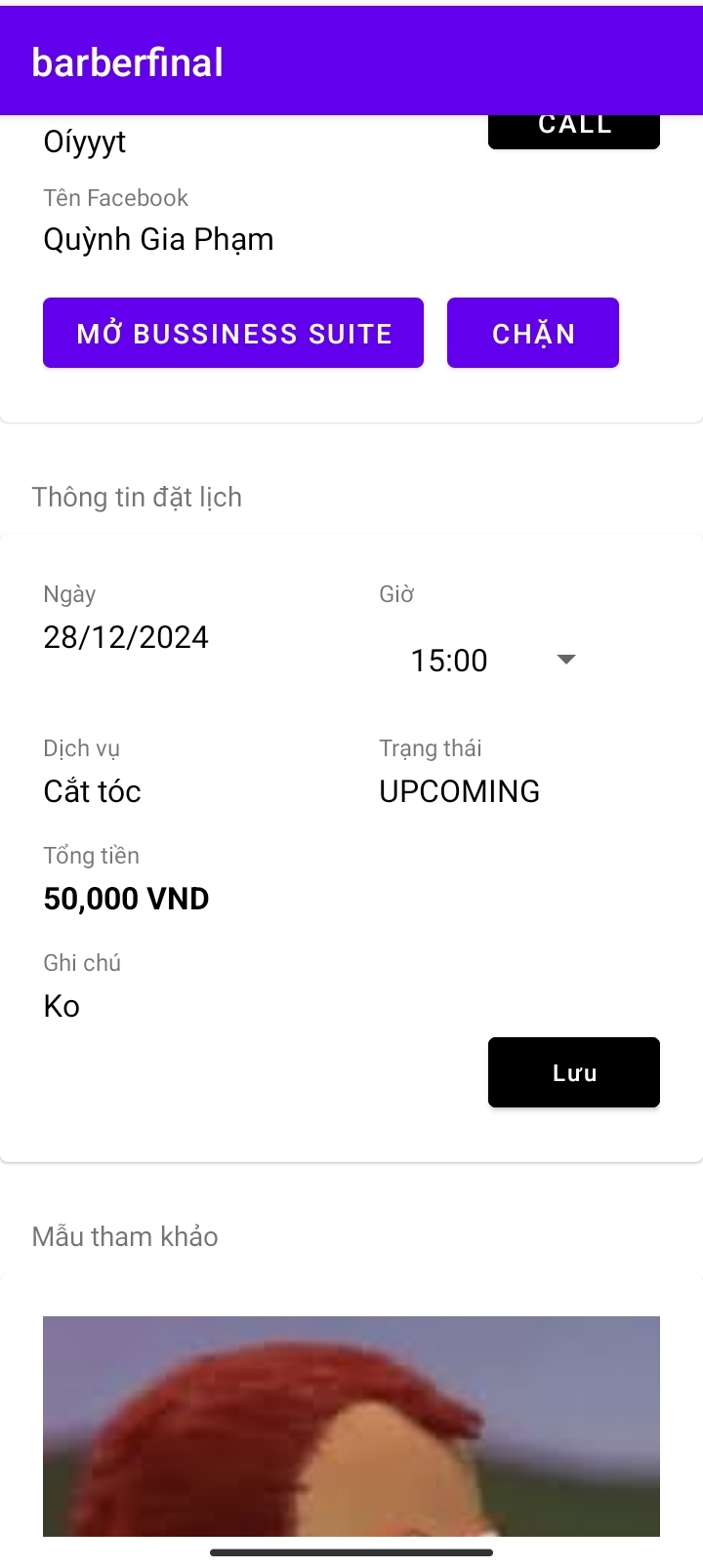
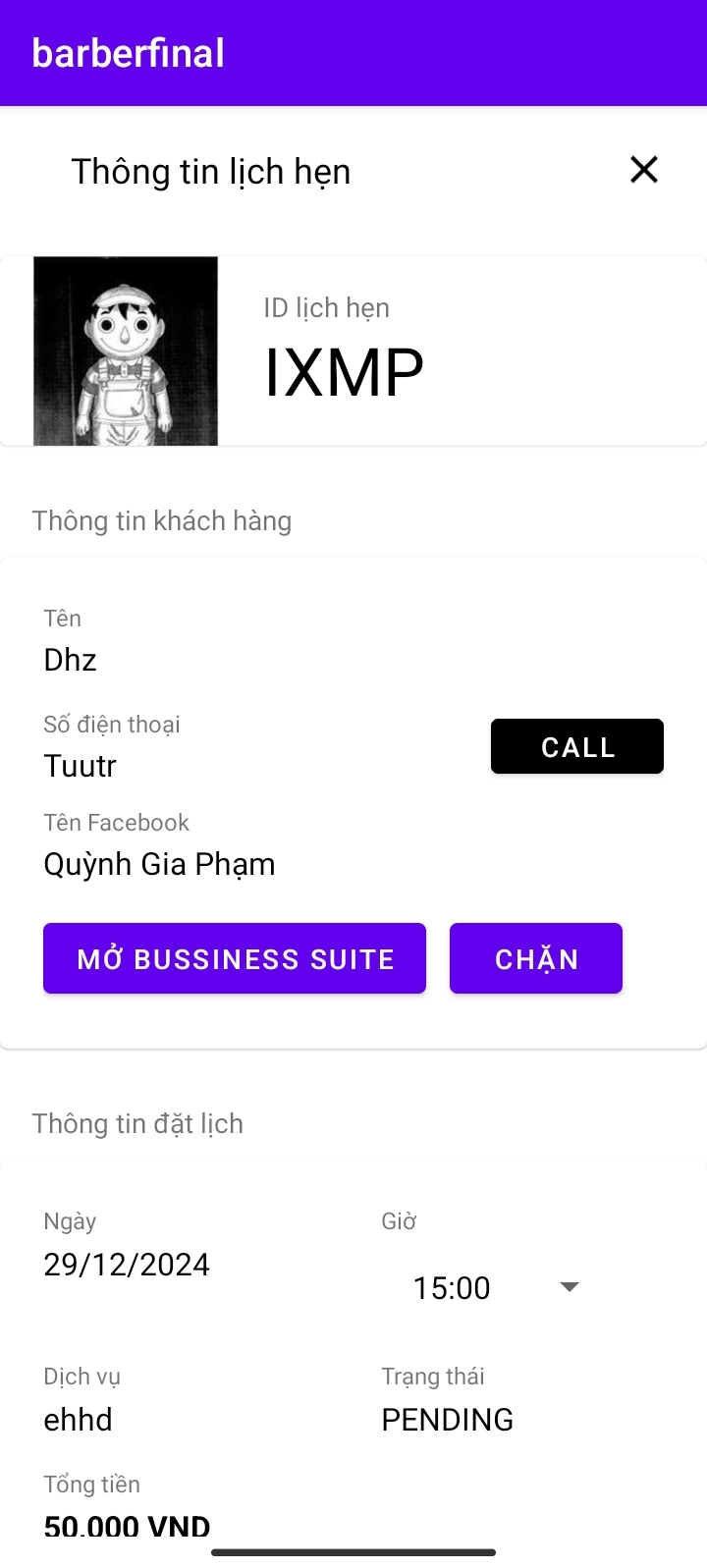
handle the item object:



activity này sử dụng các view:



giao diện:



VÌ Appointment chỉ lưu psid của người dùng nên thông tin người dùng được lấy thêm từ Graph Api facebook.

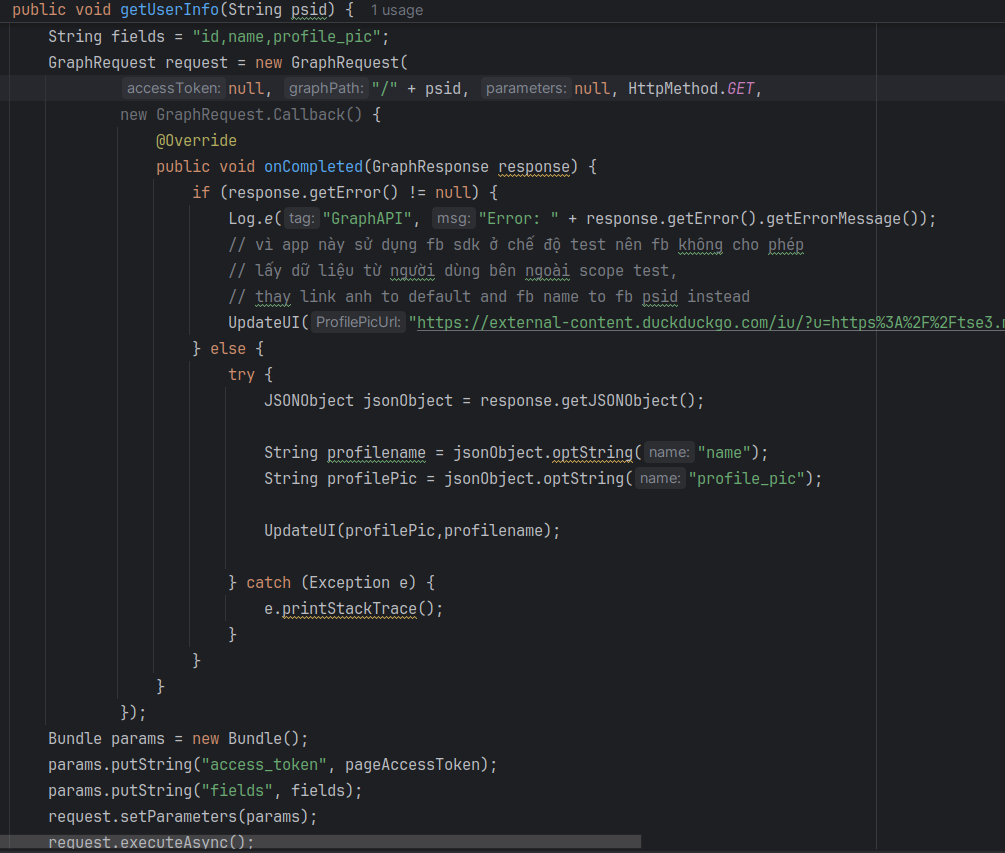
Các trường thông tin cần lấy từ Graph API.

String fields = "id,name,profile\_pic";

vì app này sử dụng fb sdk ở chế độ test nên fb không cho phép lấy dữ liệu từ người dùng bên ngoài scope test, mặc định psid chỉ cho phép lấy các thông tin cơ bản tối thiểu, more on that:

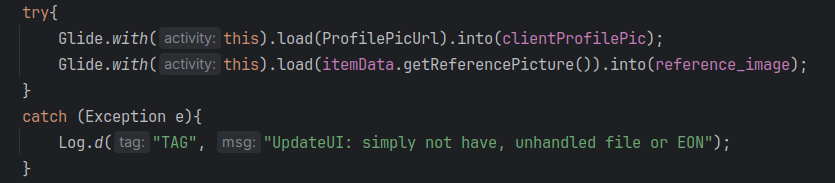
ref: <https://developers.facebook.com/docs/messenger-platform/identity/user-profile/> ,

kết quả nhận được là String name và profile\_pic là url được truyền vào UpdateUI() để xử lý



* để hiển thị image bằng URL thay vì resource: xử dụng thư viện glide

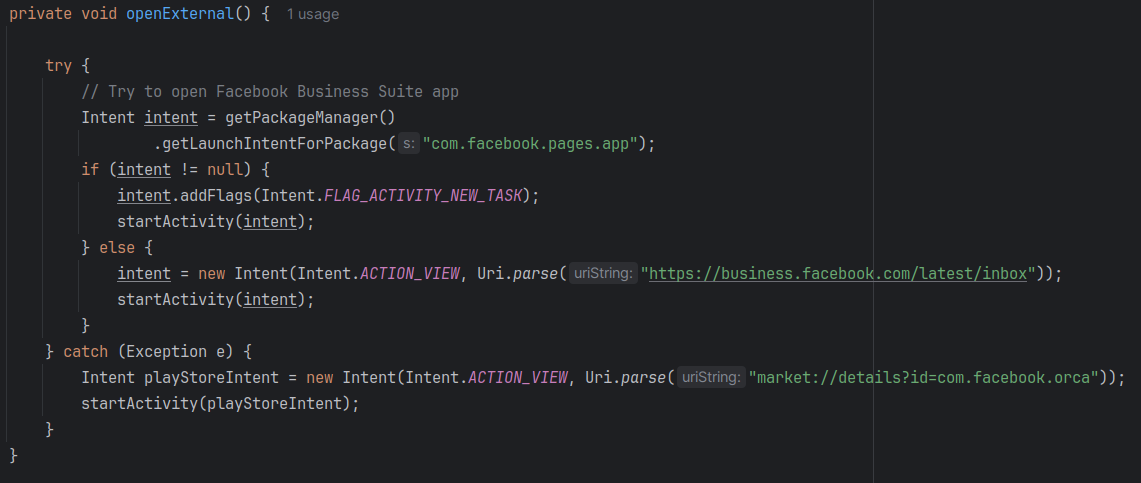




call button sử dụng intent ACTION\_DIAL để mở call intent



open bussiness suite button kiểm tra app bussiness suite hoặc mở url web bussiness suite

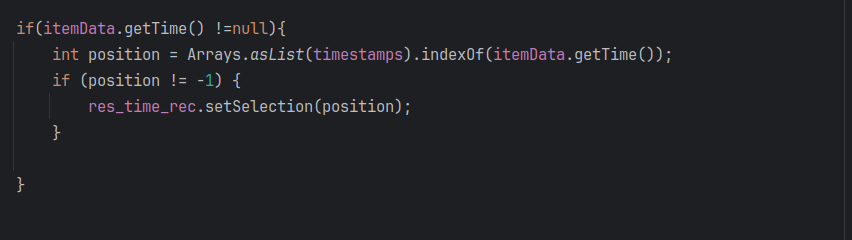


saveapp button update the appointment



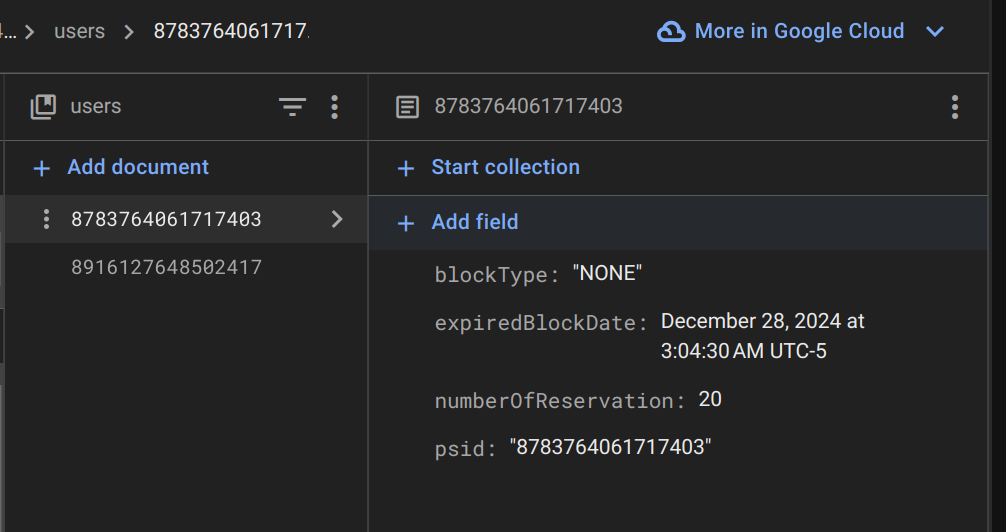
on inital view, the time spinner use default value from the appointment so when the save button is pressed it has the value of the appointmnet, value of the time spinner is a list string represent different point of time in a working hours:





để xử lý chặn người dùng:

trong firestore, mỗi shop đêù có 1 collection lưu trữ thông tin người dùng với id là psid fb, với mỗi conversation, the app server check if the user psid exist in the db or not then add it to the db with those basic field:



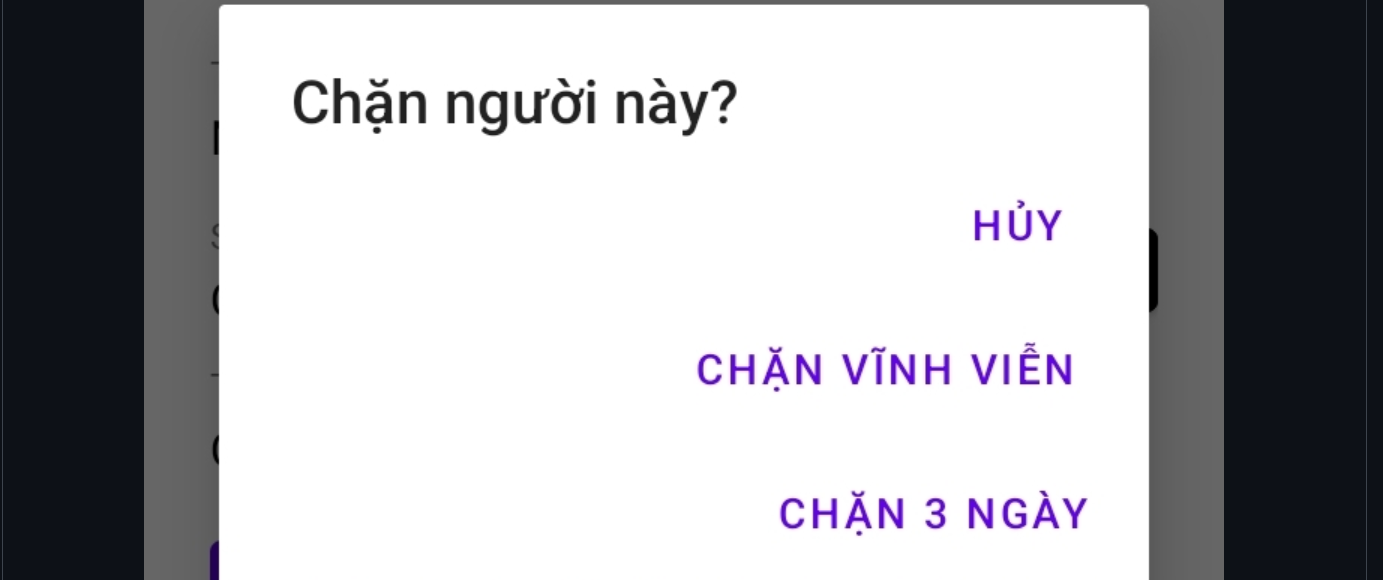
this open the oportunity for 2 type of blocking, 3 days block and permanent block. Because facebook dont allow to get user profile via psid or fill a block report as a page to the user so this type of blocking only handle message block. as user engage in conversation with the bot when blocking, the bot simply refuse to answer, this mechanic is implemented in server side, show code if needed.

back to android, the button action fetch data from firestore then decide what to do next, khi click, button taoj 1 diaglog với 3 option: chặn vĩnh viễn, chặn 3 ngày, hủy với các value tương ứng “FOREVER”, “DAY3” và truyền giá trị vào phương thức handle block button





handleBlockButton ghi và cập nhật lại giá trị của block type trên firestore,đối với type DAY3, expiredBlockDate được set với giá trị của 3 ngày sau tính từ ngày hôm nay.( expiredBlockDate khi khởi tạo từ server có giá trị luôn luôn nhỏ hơn ngày hôm nay). Server will use this field to determine whether or not to reply to a particular convesation

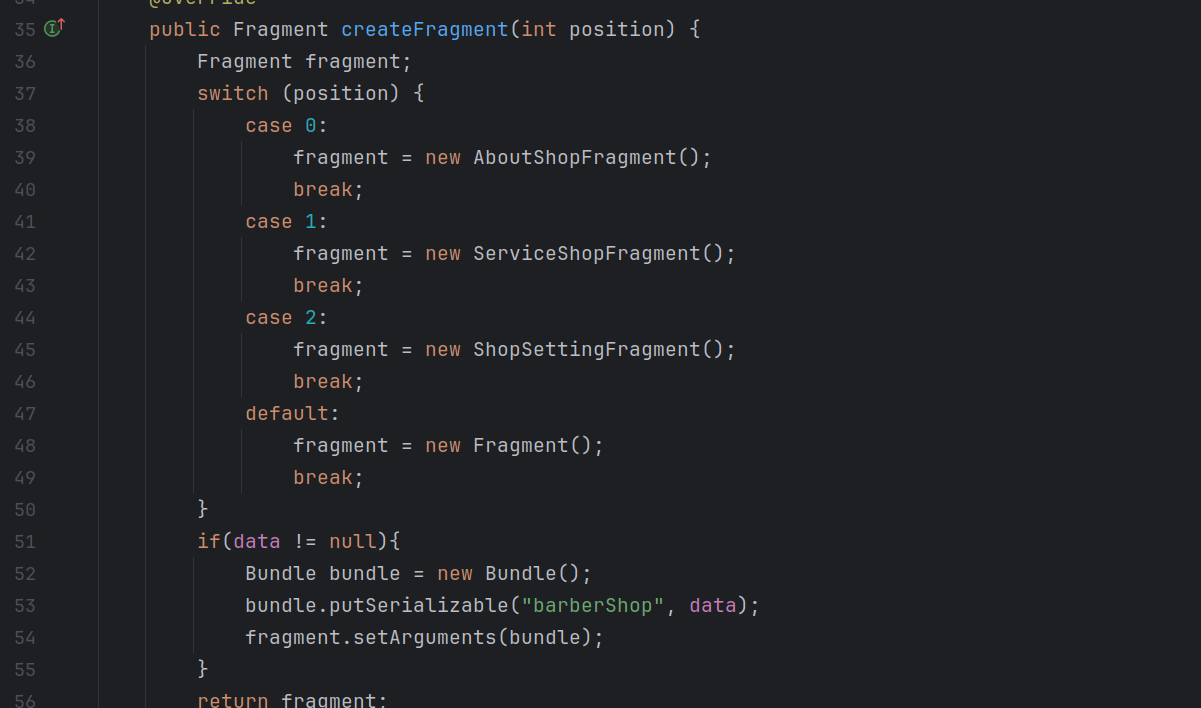


### **HOME:**

Home sử dụng tablayout và viewpager2 để chia các fragment appointment. Initialize dashboard and inflate view, avatar shop sử dụng glide để hiện thị from url



Khởi tạo view với shop data retrieve từ firestore, sau đó truyền data vào Update UI để cài đặt adapter. Với mỗi fragment tương ứng, adapter đều truyền data shop



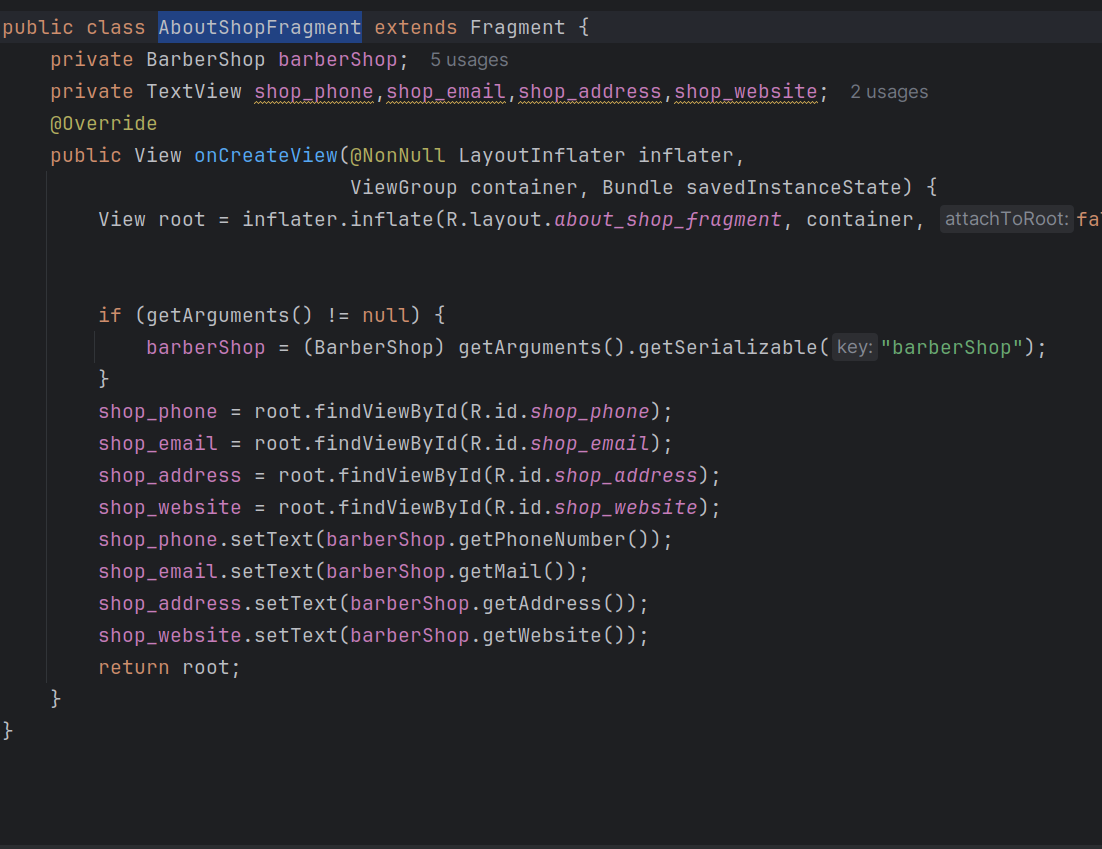


giao diện:

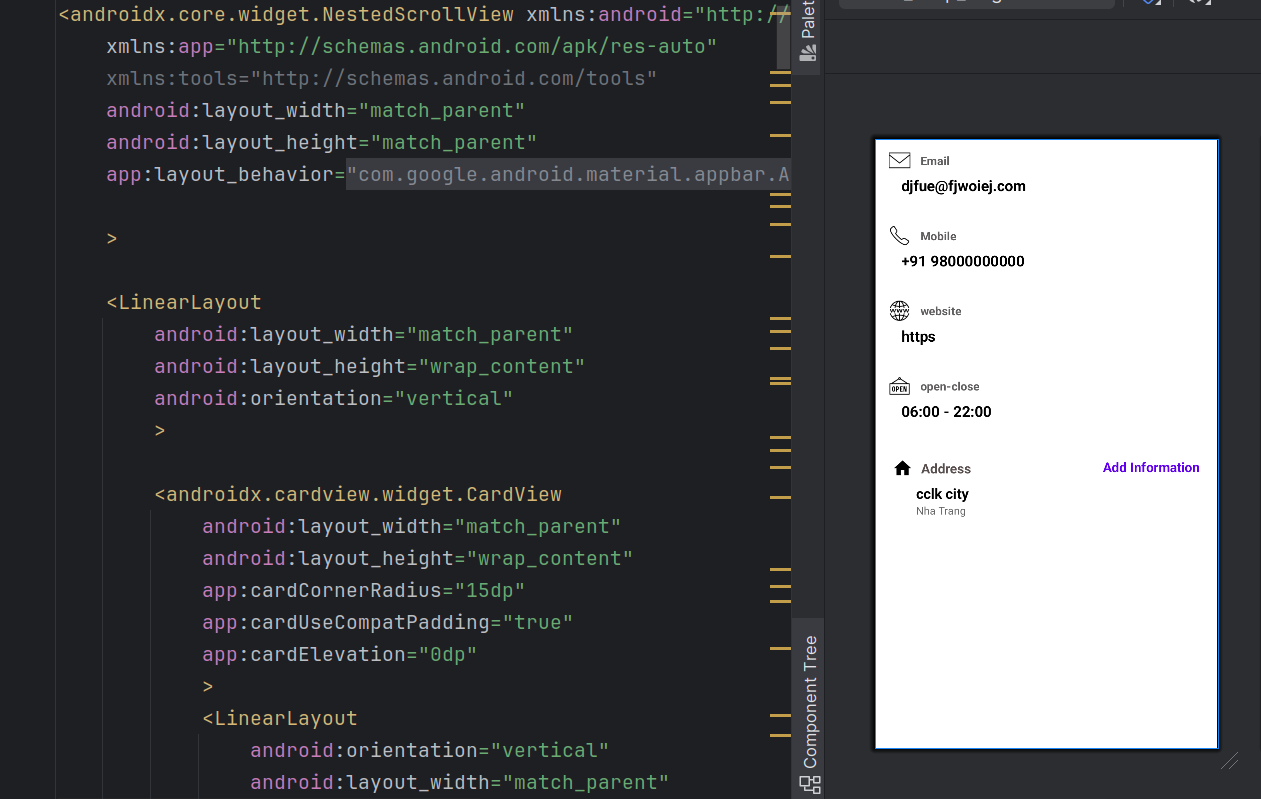


#### Shop infor(AboutShopFrgment):

get data from adapter và hiển thị cơ bản:



giao diện: xử dụng linear layout là chủ yêu:

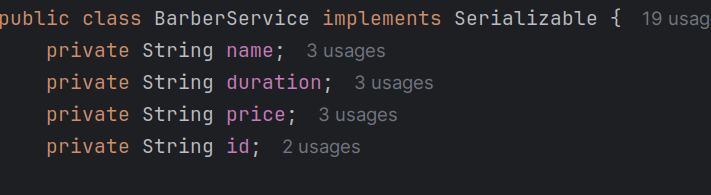


TODO:

có thể tùy chỉnh được thông tin và cập nhật lại trên trang fb

#### Service fragment:

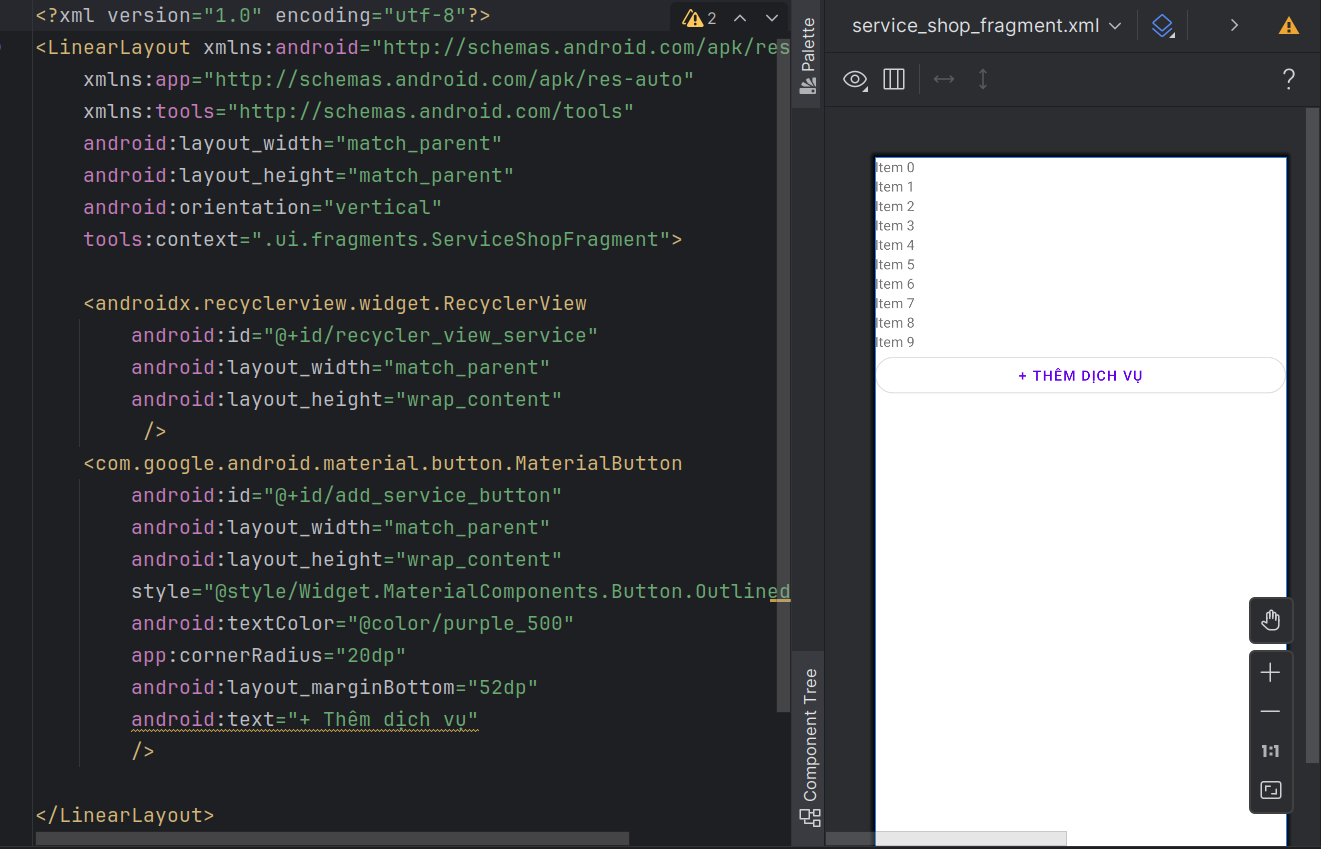
fragment này lấy danh sách các dịch vụ của shop, hiển thị và thêm ,xóa các dịch vụ , các dịch vụ bao gồm tên , số tiền và thời gian, các dịch vụ này sẽ đucợ app server hiển thị cho người dùng chọn trong messenger



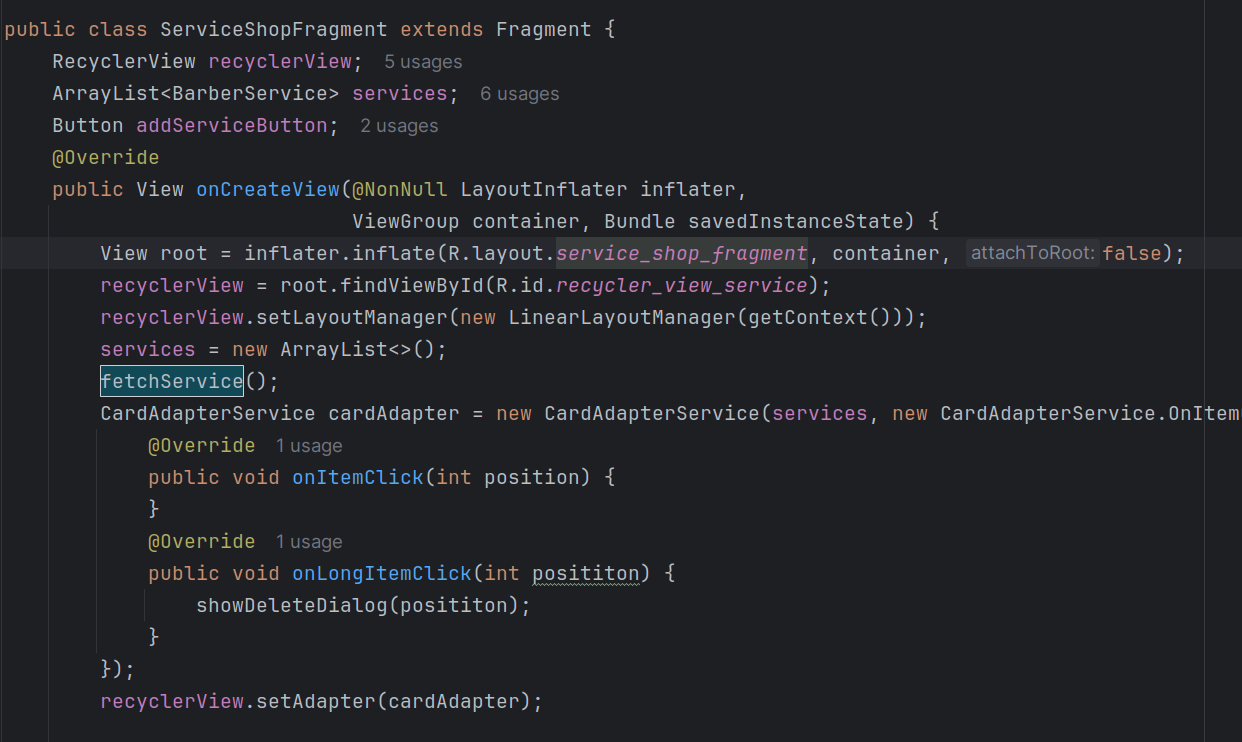
TODO:

implement stricter data type instead of string

Giao diện: sử dụng recylerview và materialbutton

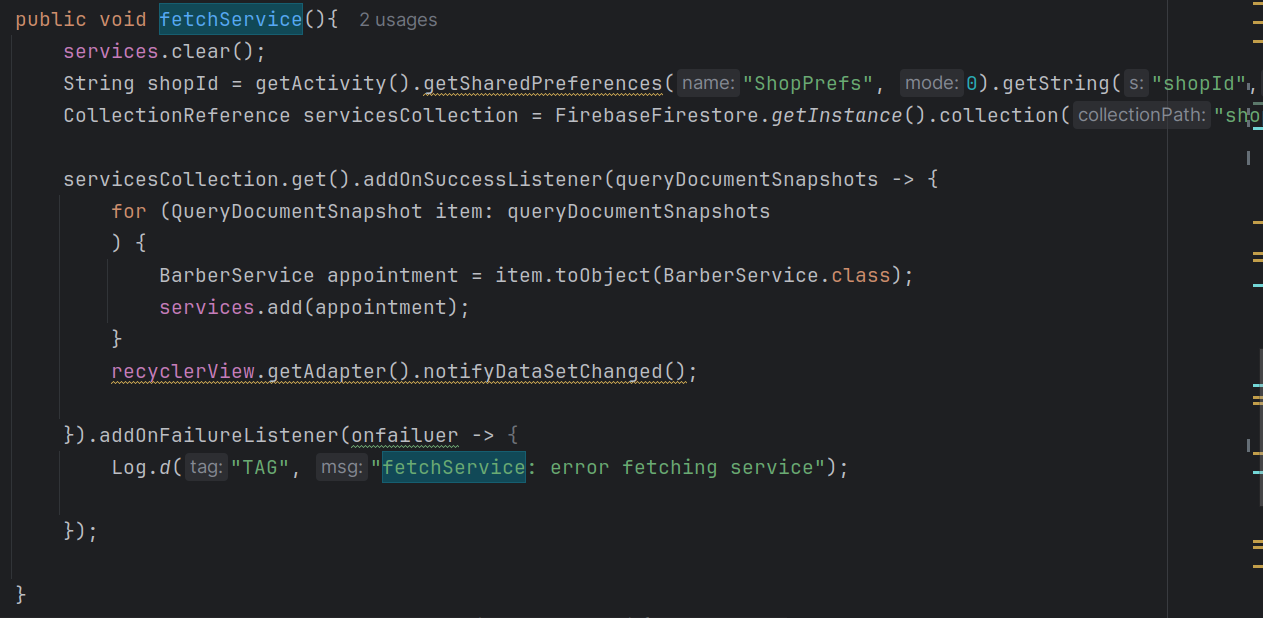


fragment initial:

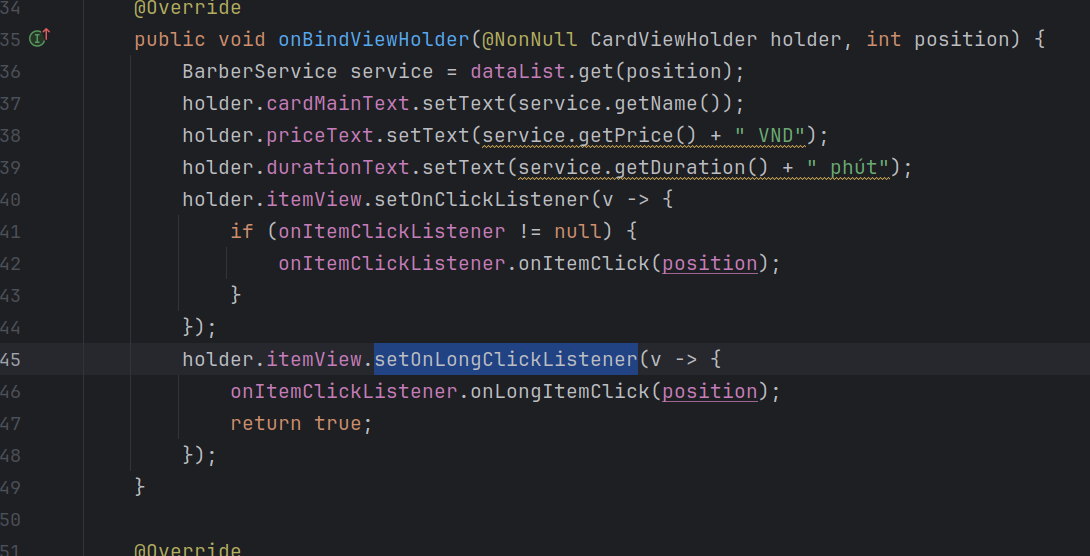


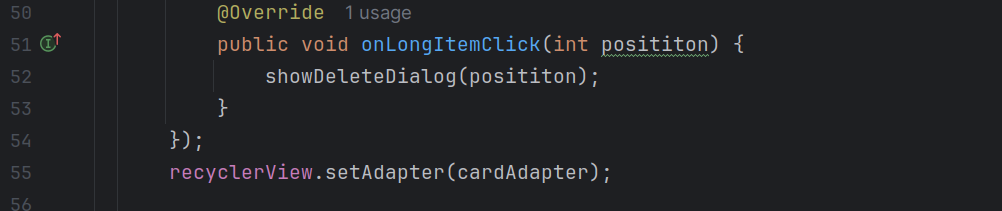
fetchService():

fetchservice đucợ call và reset services list, để notify adapter, notifyDataChanged is used in callback function after firebase call

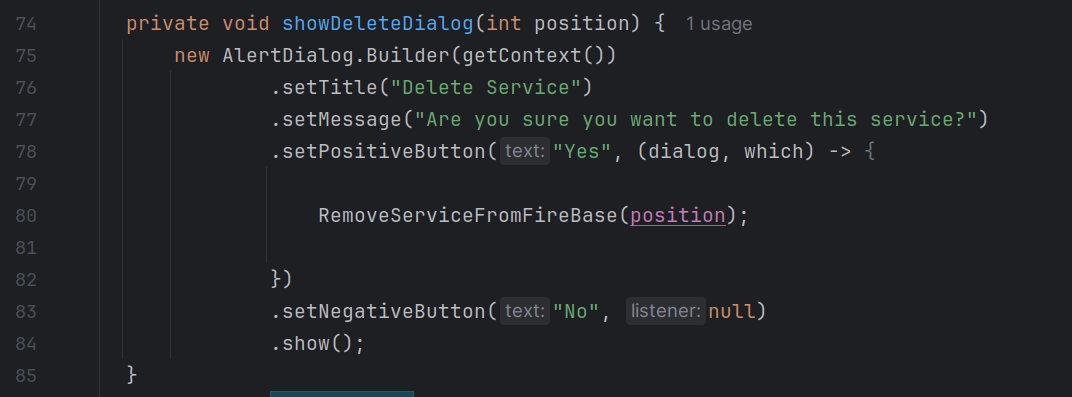


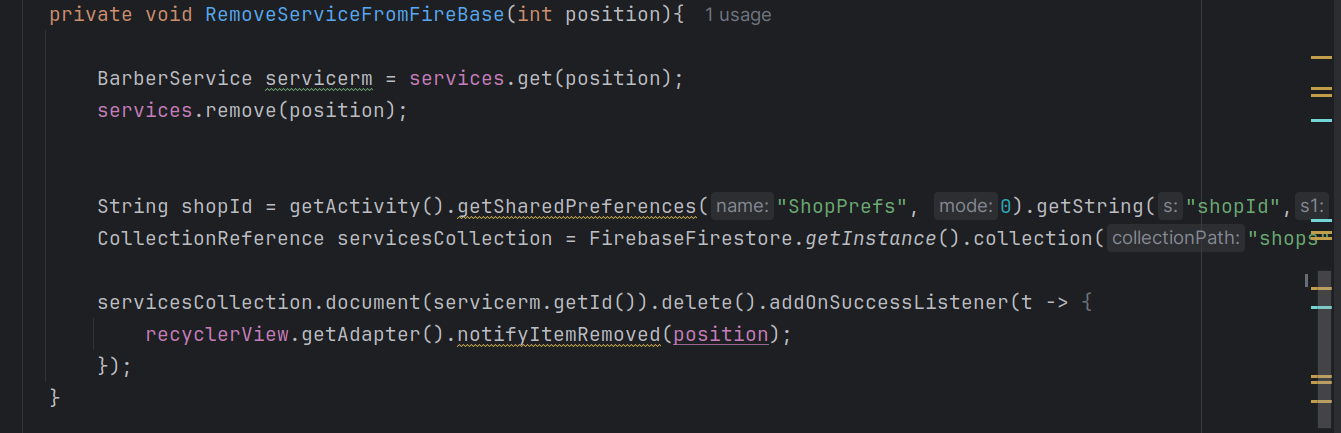
to handle service delete: use setOnLongClickListener call back trong adapter và truyền vào posittion của card:



and then showDeleteDialog is called:  


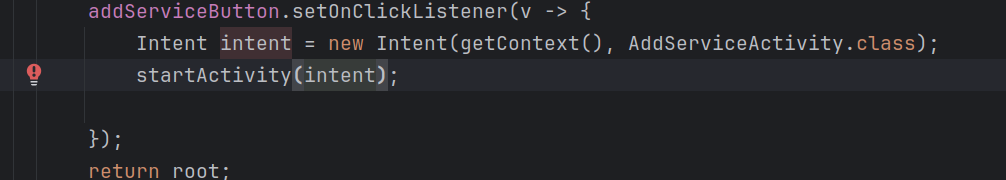
show delete diaglog then handle the actual deletion in RemoveServiceFromFirebase method:



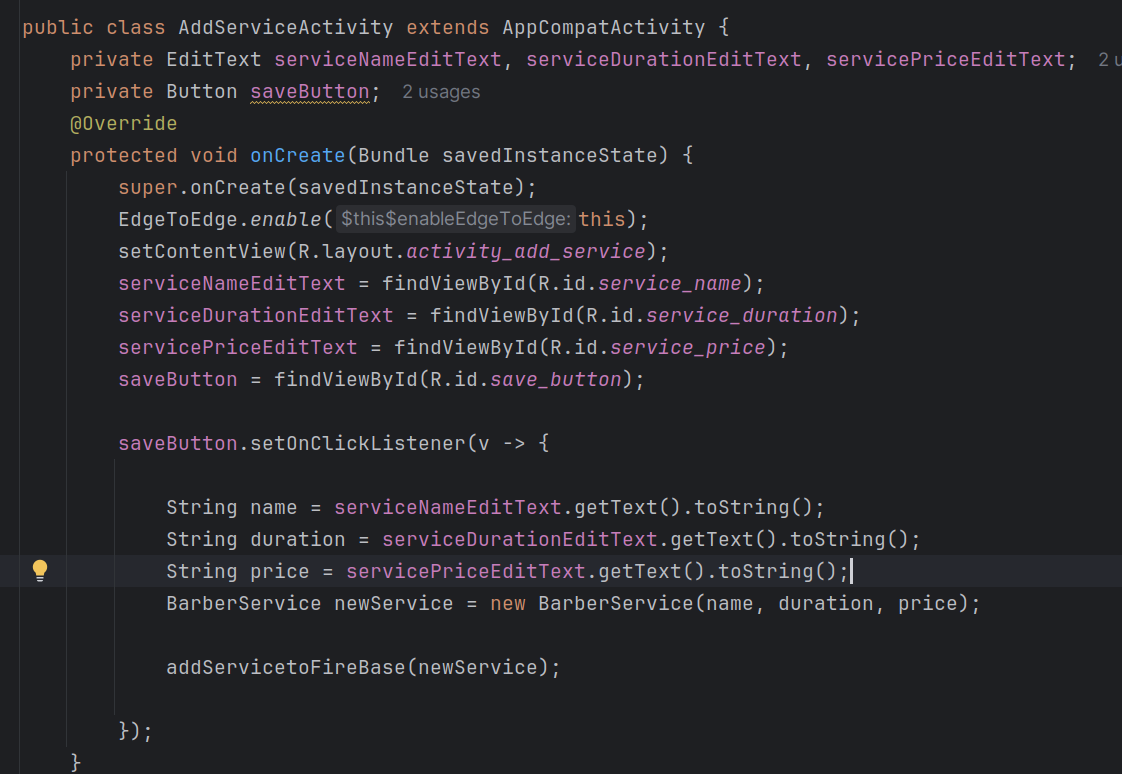


to add a service:

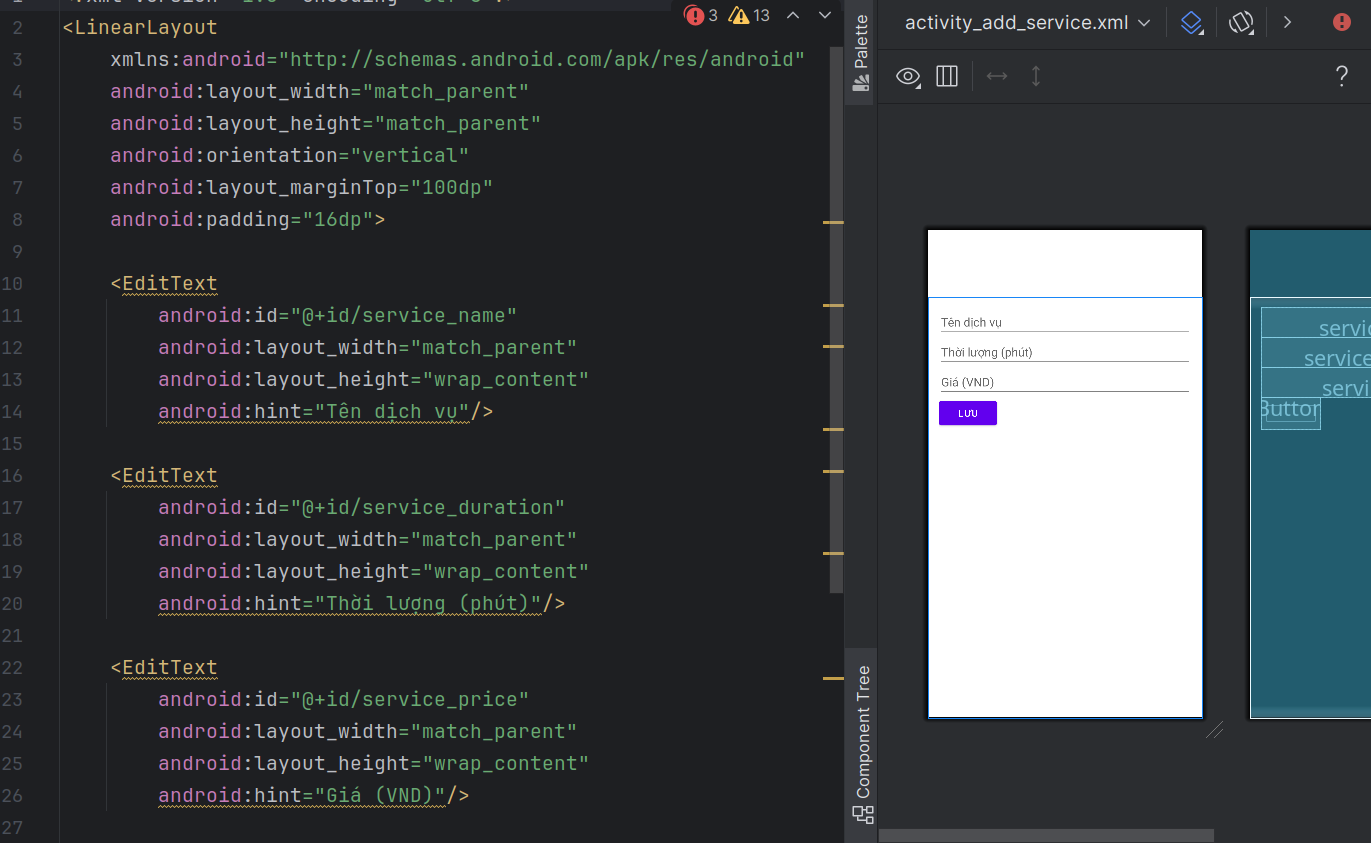
bằng cách mở activity mới mỗi lần thêm service button is clicked



AddServiceActivity is just a simple acitvy to handle data filling

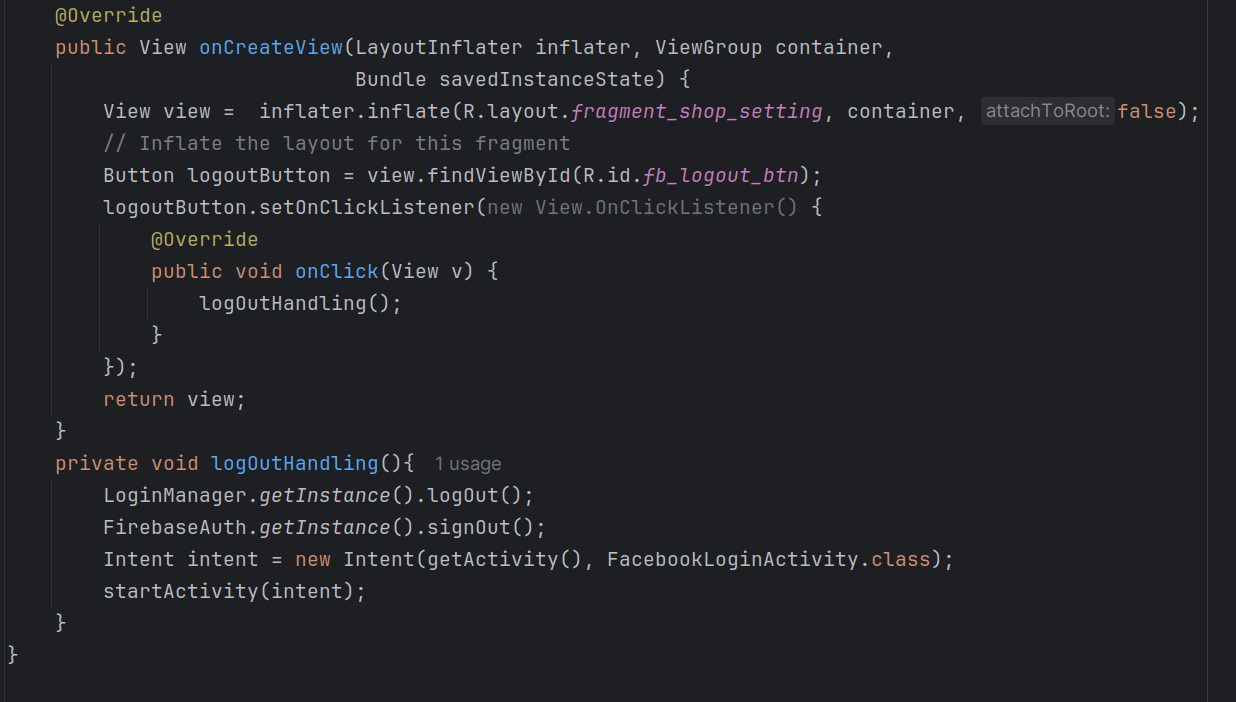


giao diện:

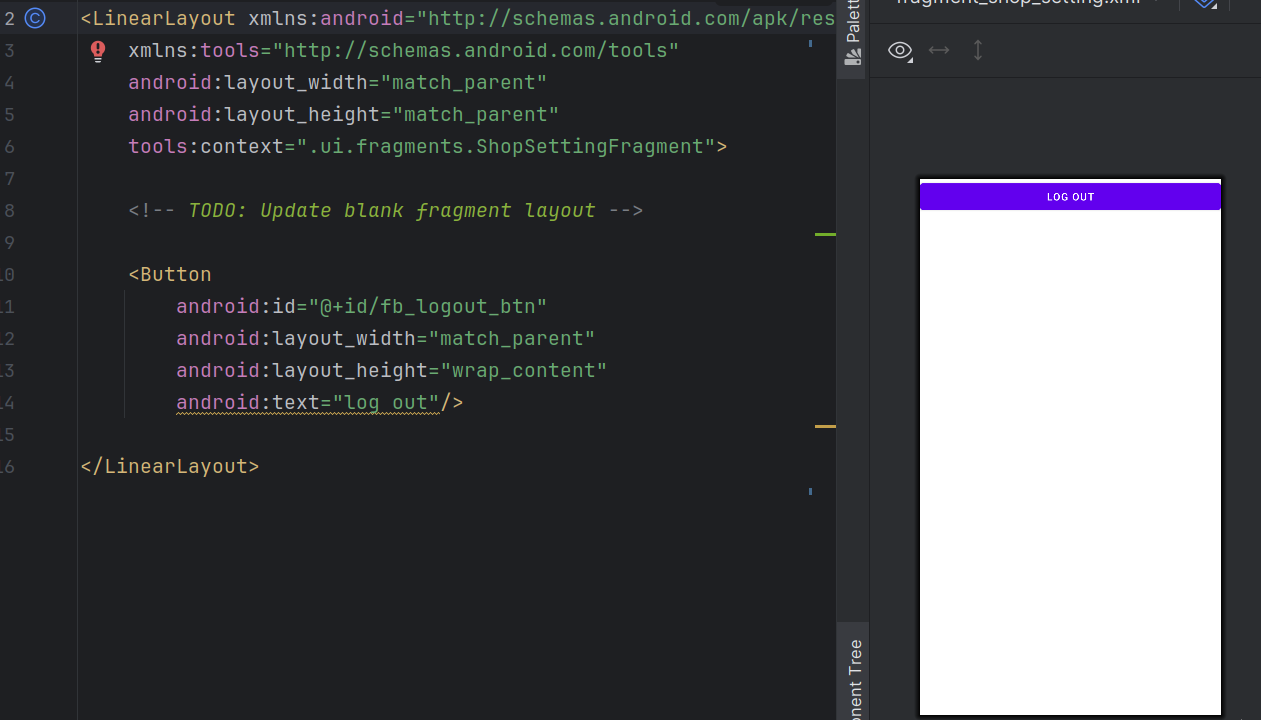


#### Shop setting fragment:

just log the fuck out of facebook and return to loging activity



UI:



TODO:

add more shop setting reference option

### **Thống kê:**

chỉ có thống kê tổng ngày giờ, lịch hẹn, các dịch vụ căn bản, có thể làm thêm thống kê chuyên sâu nhưng vì bản chất nosql nên việc liên kết giữa cacs thực thể không chặt chẽ, truy xuất dữ liệu liên kết trong nosql là một thao tác phức tạp, ví dụ:

một cấu trúc trong firestore được tổ chức như sau:

shops

shops/appointments

shops/appoiments.serviceid

shops/users

shops/service

shop/service.price

để thực hiện việc thống kê tổng tiền thu được trong ngày hôm nay ta thực hiện 2 lần gọi api:

1 lần gọi đến shops/appointments để lấy danh sách các lịch hẹn và lọc ra những lịch hẹn đã hoàn thành

1 lần gọi đến shops/service để truy xuaat dịch vụ theo id

thực hiện liên kết thủ công giữa 2 danh sách appoinments và services:

foreach appoiment in appointments:

sum+= getPriceByServicesId(appointments.serviceid);

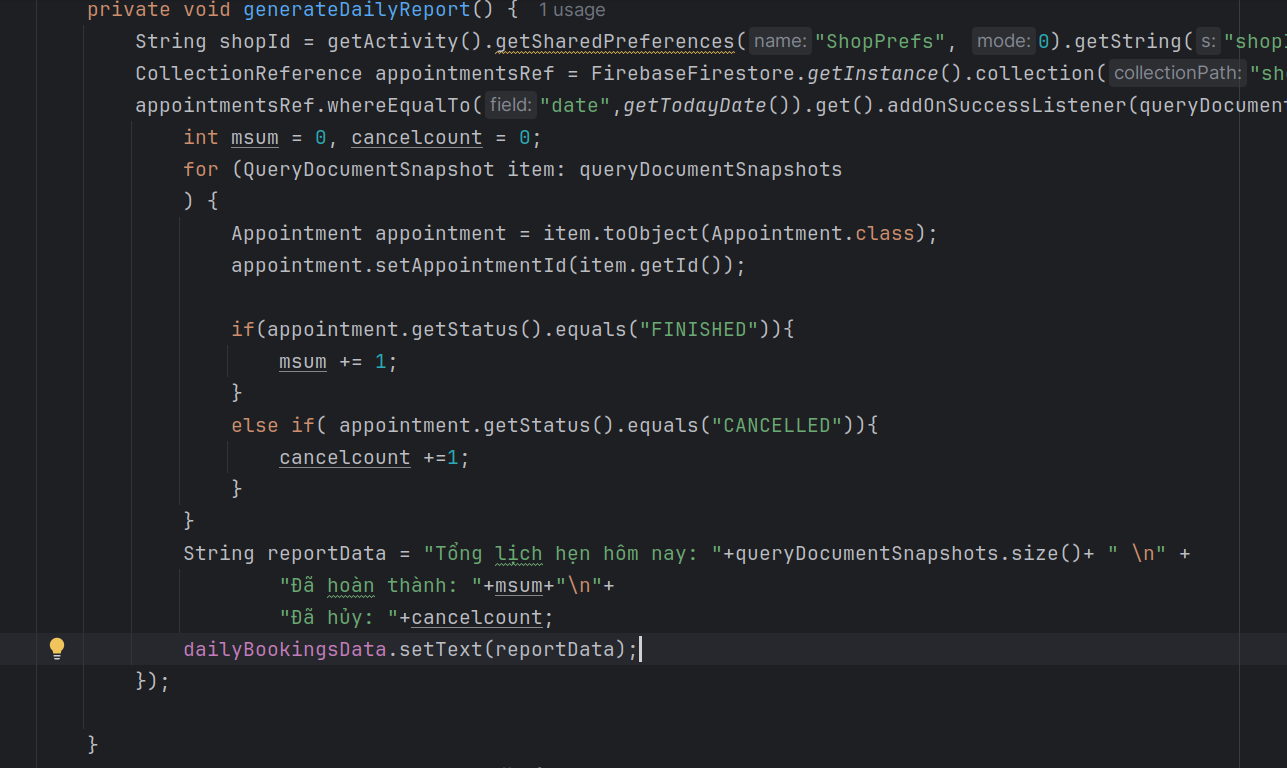
getPriceByServicesId(x) {

foreach service in services:

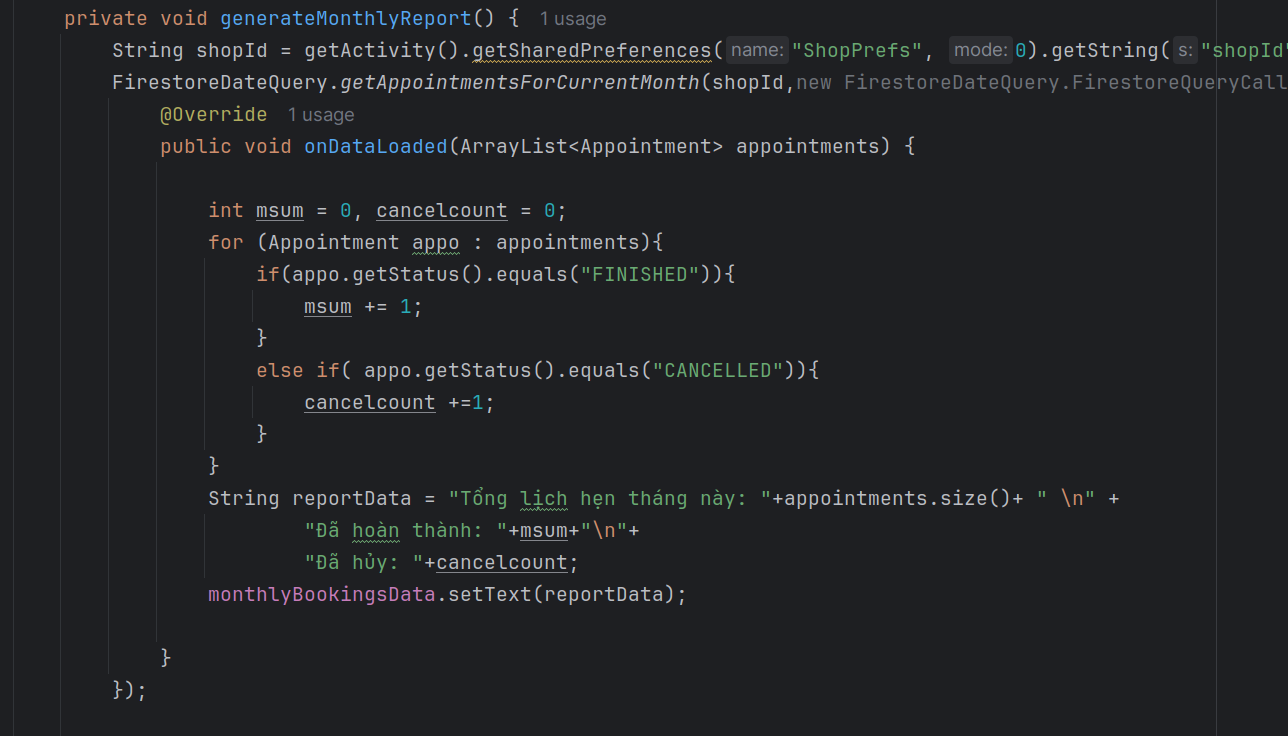
if(service.serviceId == x) return service.price

}

Để lấy thôngs kê hôm nay:



Thống kê tháng:



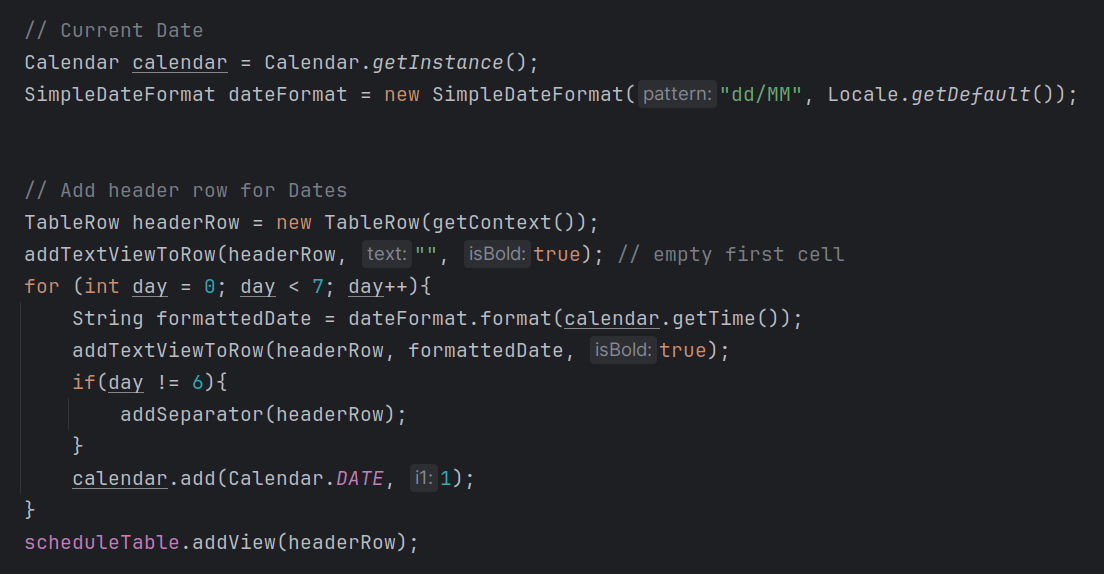
Xử dụng tableLayout để dựng lịch biểu cho quán:

idea:

1 bảng với các cột là ngày, tính từ hôm nay đến 7 ngày sau, các hàng là các ca làm trong ngày, từ 7:00 đến 16:00

Sử dụng calendar để generate and track date

to generate column:

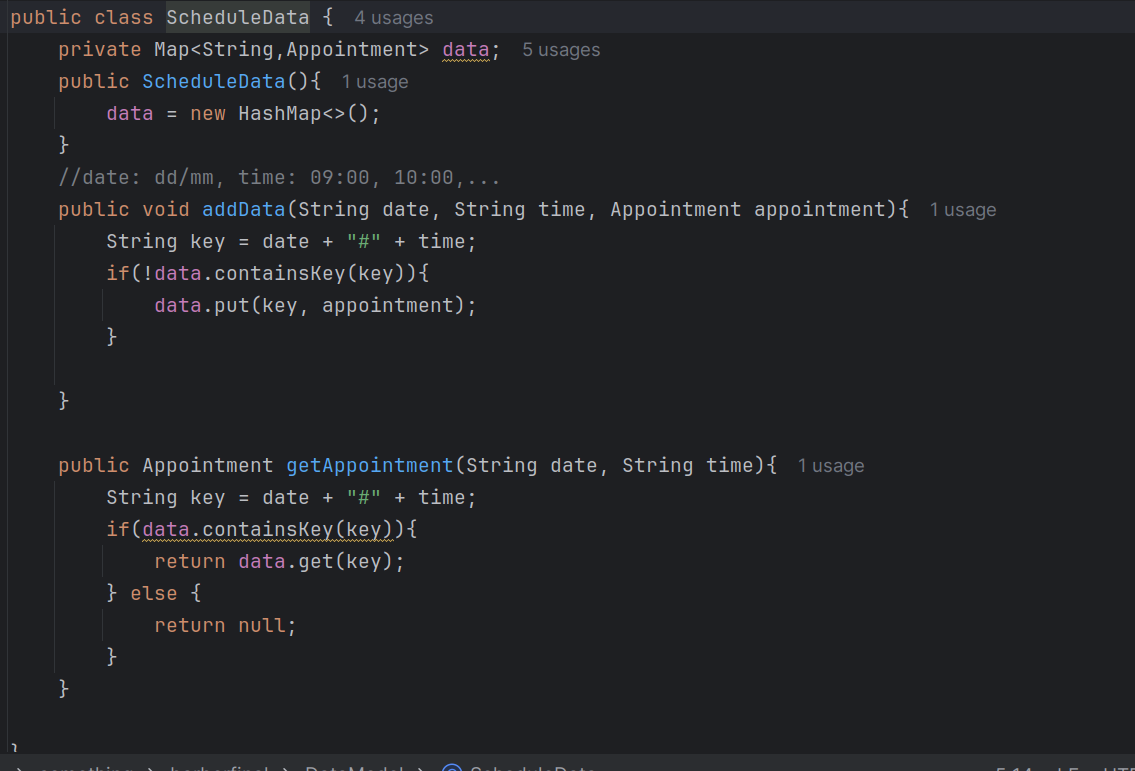


iterate through each row to fill the column:

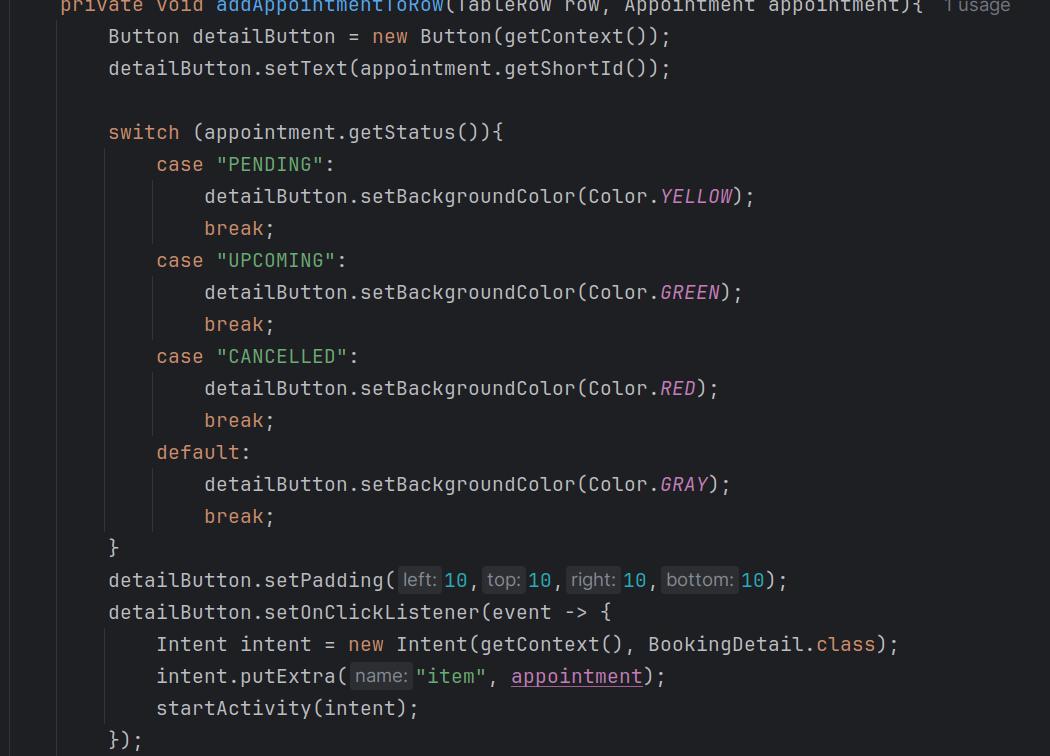
for each cell, look up in a hashmap table with key is date(column)+ row(time) to retrive any appointment matchés that time, then add data



code for the hasmap date:



if match then generate button to open detail of the appointment:



also set the color based on the appointment status