Technical Project Report - Android Module

FoodDelivery

Subject:	Computação Móvel
Date:	Aveiro, 3/01/2023
Students:	95278: Diogo Cunha
Project abstract:	This report presents the development process of an Android application. The project consisted of a food delivery service. The application will have an interface for the courier and another for the client. The clients will be able to order food from different restaurants, and the couriers will make sure the client orders are received at their respective addresses.

Report contents:

1 Application concept

The application developed is called FoodDelivery and its target is exactly that, to deliver food. There are 2 groups of target users, the clients, and the couriers.

The clients are people that want to order food from a variety of restaurants in the application. When ordered the client can track the delivery with the application map. This brings more comfort to the users when ordering food from home.

The couriers want to make money delivering, so they receive a list of orders and can accept an order if they choose to. Then they can obtain directions to complete the order, by visualizing the directions on the application map.

Both groups use the same application, and the login is also the same but once they log in they will be redirected to different pages of the application.

The clients and couriers can have a “profile” that contains a name and a profile image.

2 Implemented solution

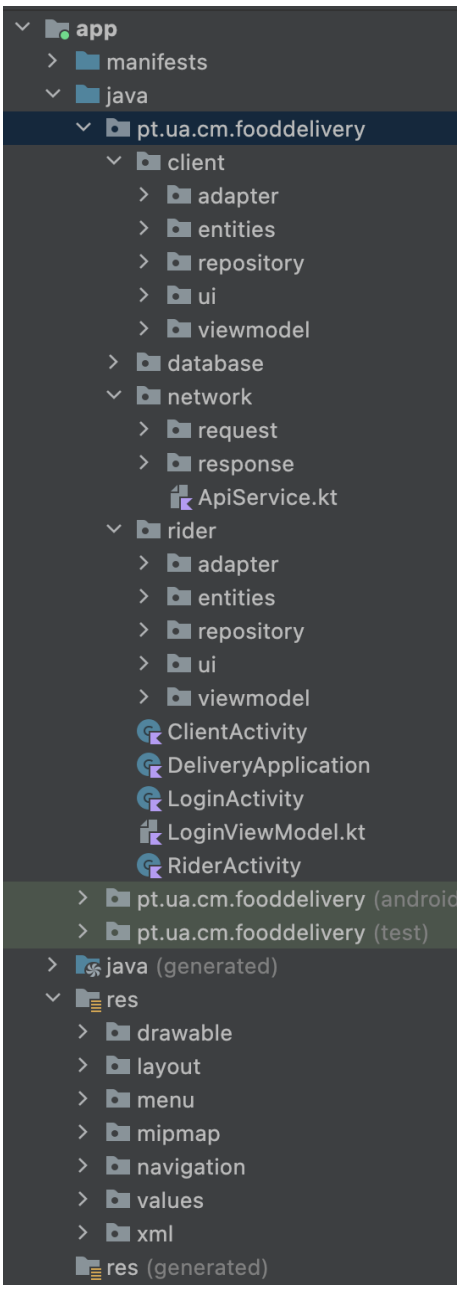
Architecture overview (technical design)

Regarding the application architecture, the pattern used was the MVVM. The local database used was the Room DB, and to make external API requests was used the Retrofit with the Moshi converter. To update the UI I used view binding. Used the device location and camera and Coroutines were used to write values to the Room DB asynchronously.

The application is divided into three activities (Login, Client, and Rider Activities). The login activity is responsible for identifying the user and starting the next activity accordingly. The client activity will contain its own navigation and will deal with client functionalities. The rider activity will also have its navigation component and be responsible for the rider functionalities.

Also developed an external API that the application makes use of. That API was implemented with NodeJS and contains a MySQL database. That API helps with almost all the functionalities of the application (login, get client orders, accept orders, etc).

The folder structure is represented in the next image, where I have three activities, a package for the client and another for the rider, one package for the Room DB, and another package for the network (external API calls).



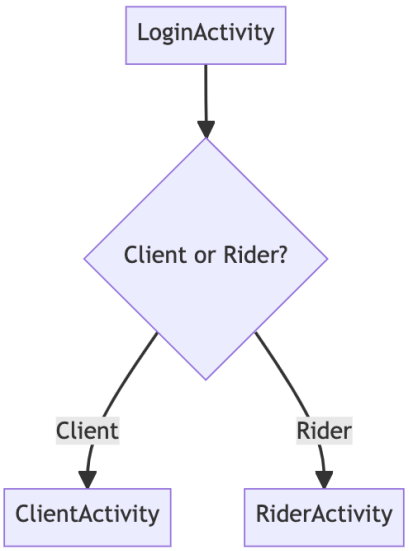
The client and rider packages are divided into more packages, adapters, entities, repositories, UI, and ViewModel. The adapter was used to build the recycler views present in the application, the entities packages contain the entities used and the needed annotations in order to build the DB Schema, the UI packages contain the fragments used and it gets updated when the view model changes the live data variables. The view model can access the repository for local data or can make external requests using the network packages and then post values for the live data.

The network packages are where the requests and response models are defined, where the Retrofit and Moshi are initiliazed, and where the endpoints are defined.

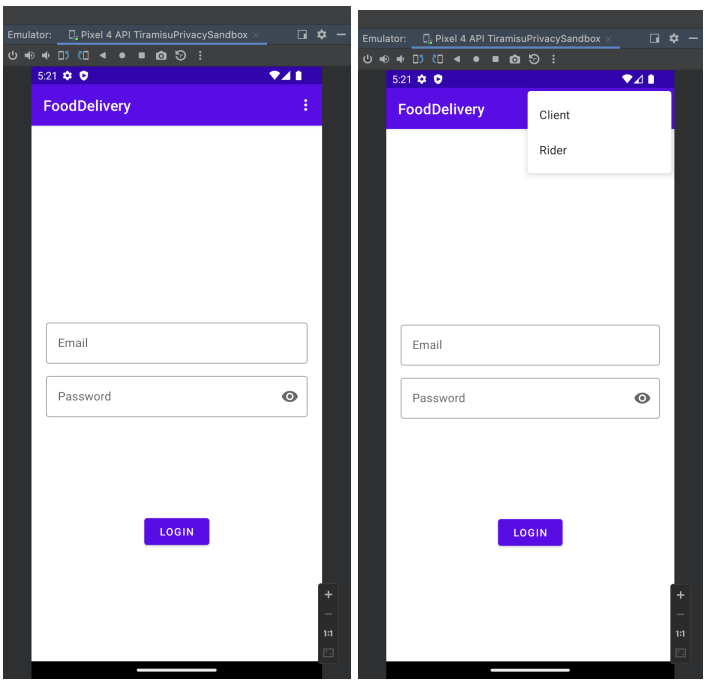
For last, the database contains the class that instantiates the DB and populates it with static data.

Implemented interactions

The application is divided into 3 activities, the “LoginActivity” is the first one to appear to the user. Once the user logs into his account a new activity will be created depending on the type of account the user has. In case the user has logged into the device once, the next time the application is opened on the same device it will log in automatically.



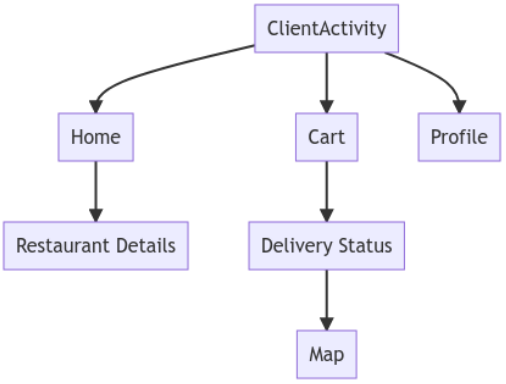
Let's first look into the login activity, it simply contains a form so that the user can log into his account. The user simply needs to select on the top menu which type of account he wants to log in at.



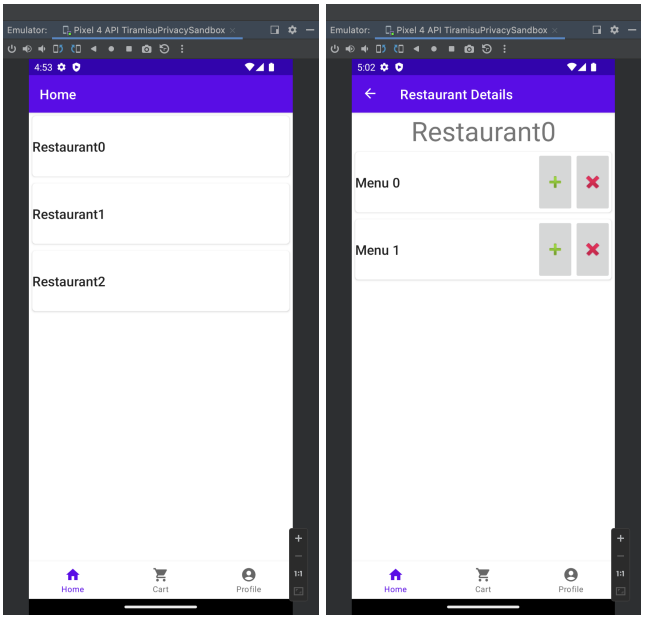
Once the user logs another activity will be created. After the first login, the client/rider information is stored in the Room DB and the login will be done automatically.

Let's now have a look at ClientActivity. The ClientActivity contains a bottom navigation bar that the user can use to switch between the three main fragments (Home,

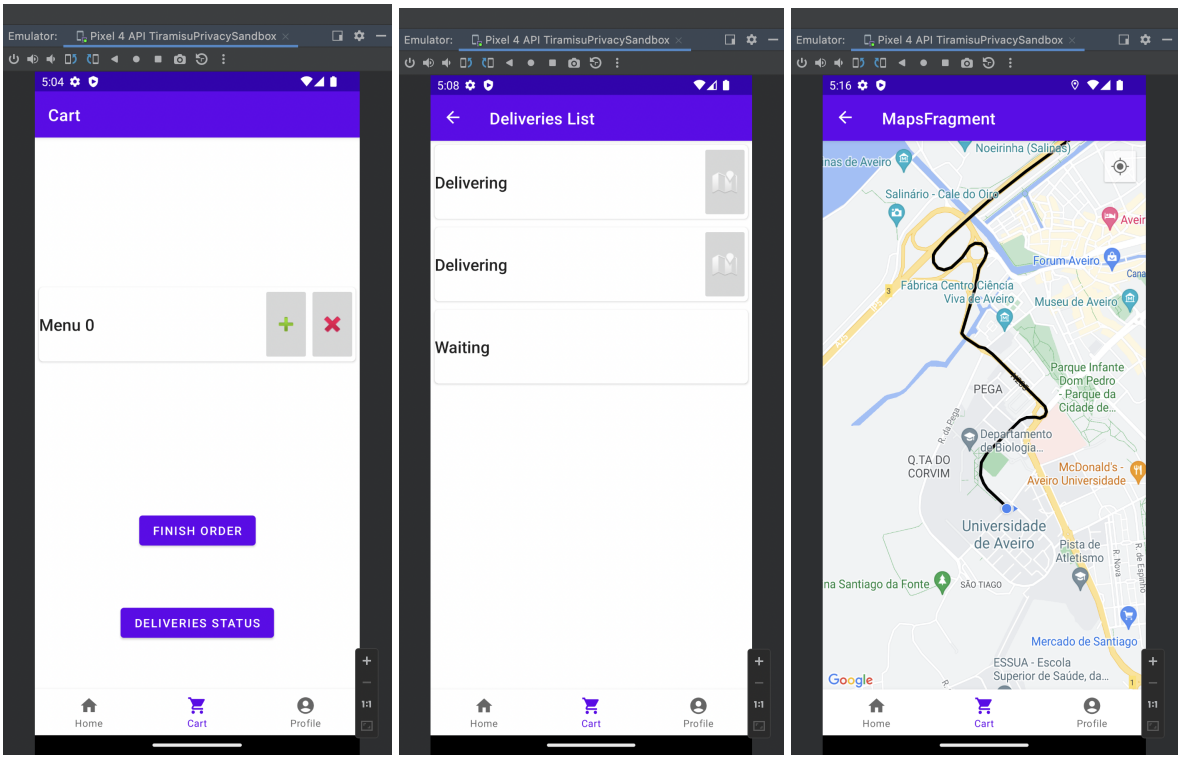
Cart, and Profile).



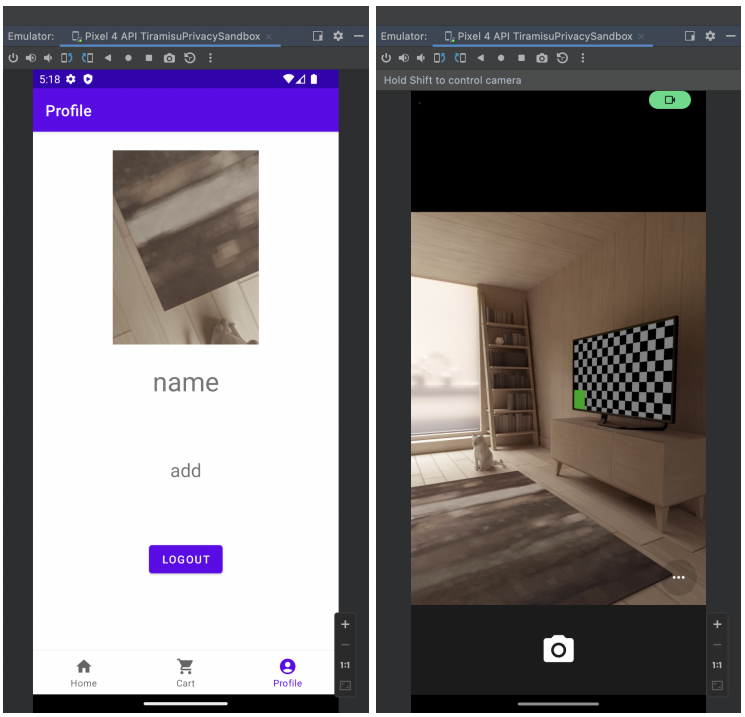
The next two images are the home page and then when the user clicks on a restaurant in the recycler view a details fragment of that restaurant appears, where the user can add or remove menus to the cart.



Next, we have the cart page. On that page, we can see the menus that were added to the cart and finish that order. By clicking on the “Deliveries Status” we can see the status of current active orders. The image in the middle shows exactly that, we can see 2 orders in the status “Delivering” and one “Waiting” for a rider to accept. If we click on the map button in the delivering orders we can see the location of the rider, as shown in the image on the right side.

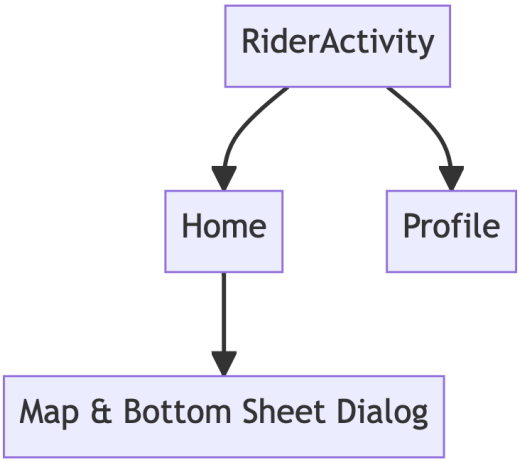


For last, the following image is the profile page. Where is presented the profile image, name, and address of the client. The user can click on the image to change the profile page and logout by clicking the “Logout” button.

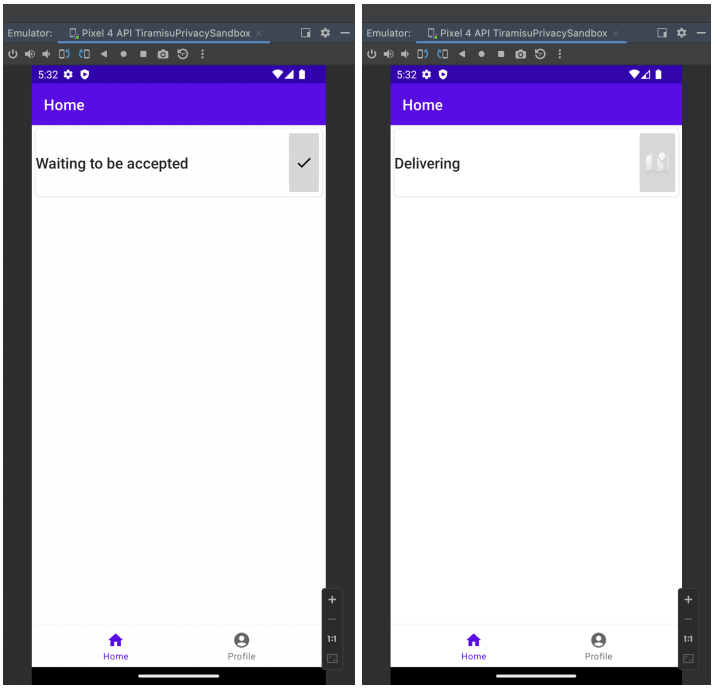


Now let's look at the Rider Activity. And as we will see it's very similar to the client,

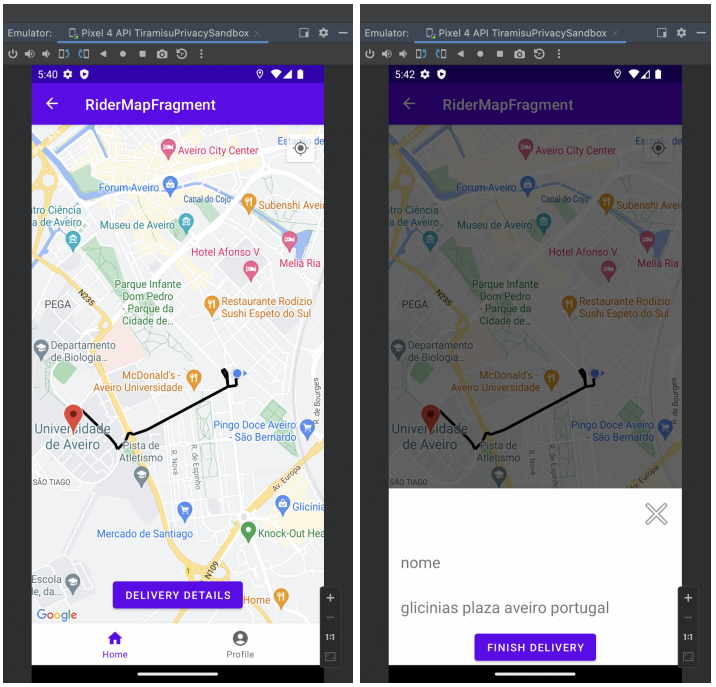
containing a bottom navigation bar to switch from the home page to the profile page.



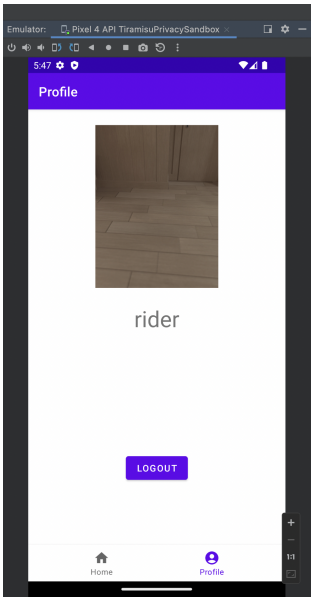
So the home page looks like the following images. The one on the left is an order that is waiting to be accepted by the rider, and once the rider accepts view gets updated to look like the image on the right.



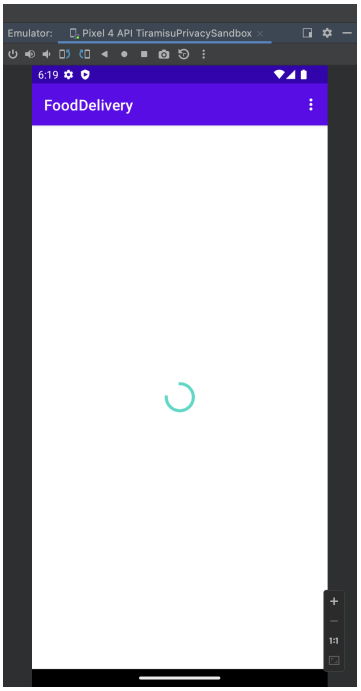
Once an order is accepted, the only delivery service that will be present to the rider on his home page is the delivery he just accepted. And by clicking on the map on the delivery list, we will get the next map fragment along with a bottom sheet dialog where the rider can get more info about the delivery, like the name of the client and the address and where he can finish the order.



The profile page is similar to the Client Profile page.



All the activities when making external requests with the API have a progress bar appear in the display. Example:



Project Limitations

The project contains some limitations and things that weren't implemented the right way. The biggest reason for that was that I had to do it alone.

So exists limitations on the impossibility of making the registration with the application. The profile picture only gets updated locally and isn't possible to make the request for the external API to persist that change.

Another limitation is the fact that the delivery act is reduced to only the distance from where the driver is to the location of the client, that is, the location of the restaurant is irrelevant, I always assume that the delivery service starts from where the rider is located.

Unfortunately, there exist some bugs in the maps fragment that I didn't have the time to try and fix it. Those bugs are when requesting location permission and the directions are not always accordingly to the current location.

Also planned to make the delivery confirmation by reading a QR code and improving the UI style, but wasn’t possible due to time constraints.

3 Conclusions and supporting resources

Lessons learned

I for sure learned a lot about android development using kotlin. During the project realized some things I was doing wrong, like for example, the separation of concerns, which I refactored at the beginning of the project, and the unidirectional data flow.

Having to develop the same application earlier, but using flutter, I found the development of android apps using kotlin more difficult to understand and implement.

Project resources

Resource:	**Available at:**
Code repository:	https://github.com/dgCunhaUA/Food-Delivery-Kotlin
Ready-to-deploy APK:	https://github.com/dgCunhaUA/Food-Delivery-Kotlin/blob/master/app-debug.apk
Demo video:	https://www.youtube.com/watch?v=OpZe-dQvrgk

Reference materials

https://developer.android.com/courses/android-development-with-kotlin/course

https://chris-ribetti.medium.com/android-viewmodel-livedata-repository-and-di-complete-and-super-quick-5a7d78fa7946

https://stackoverflow.com/questions/64009427/how-can-i-get-continuous-location-updates-in-android