Technical Project Report - Android Module

TqsIcmProject

Subject:	Computação Móvel
Date:	Aveiro, 22/06/2022
Students:	97880: Camila Fonseca 98475: Rodrigo Lima
Project abstract:	Mobile application to allow users to create, choose a delivery, and track the delivery process for delivering packages for a delivery company. The application has GPS positioning detection, QR code generation and reading, push notifications, persistence and communication with an external API with an integrated DB.

Report contents:

1 Application concept

The app is for Riders (Usually independent contractors) that work for a Delivery Company, which works with multiple stores or locations. They would use the app to select from the available deliveries to carry out, pick one, and update their status on its progress as they work on it. The app enables the whole business model to work.

The app can also be used to create orders for the riders to deliver.

2 Implemented solution

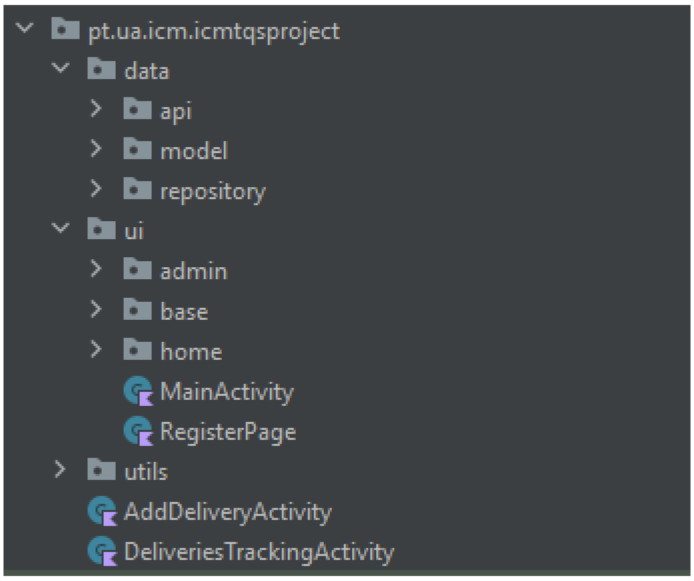
Architecture overview

Login/Register Pages:

Utilizes an active **internet** connection to call an **external API**, in order to authenticate login attempts and register new users.

Until a successful login attempt takes place, no other areas of the app can be accessed.

There's **persistence** on this step, since a previous login/registration will mean the user is sent straight to the Admin/Home pages instead of being shown the login page again.



```
pt.ua.icm.icmtqsproject
  data
    api
    model
    repository
  ui
    admin
    base
    home
    MainActivity
    RegisterPage
  utils
  AddDeliveryActivity
  DeliveriesTrackingActivity
```

Fig 1 - The project's file organization

Admin Page

In case the user logs in with an Admin account, they'll be redirected to this page.

Here the user can view all previously created Deliveries, and create new ones.

When creating a new delivery, a library called **Locus**[1] is used to provide GPS coordinates if the user chooses to set the delivery's drop-off point to their current coordinates, automatically.

If the user clicks a previously created Delivery, its respective **QR code** is generated, and displayed as a pop-up. This code is used to turn in the delivery on the Riders' side - It would be printed on the package in a real-world scenario.

The admin page is built with a Adapter-View-Viewmodel structure. Data is retrieved from, and sent to, an **external API**, by calling a **Service,** which abstracts the call with a **Repository**.

Home Page

The home page is also built with an Adapter-View-Viewmodel structure. The Adapter binds the data fetched from the API in the ViewModel (From a Repository, which fetches the data from an **External API.**) to the View, which deals with actually presenting it to the user.

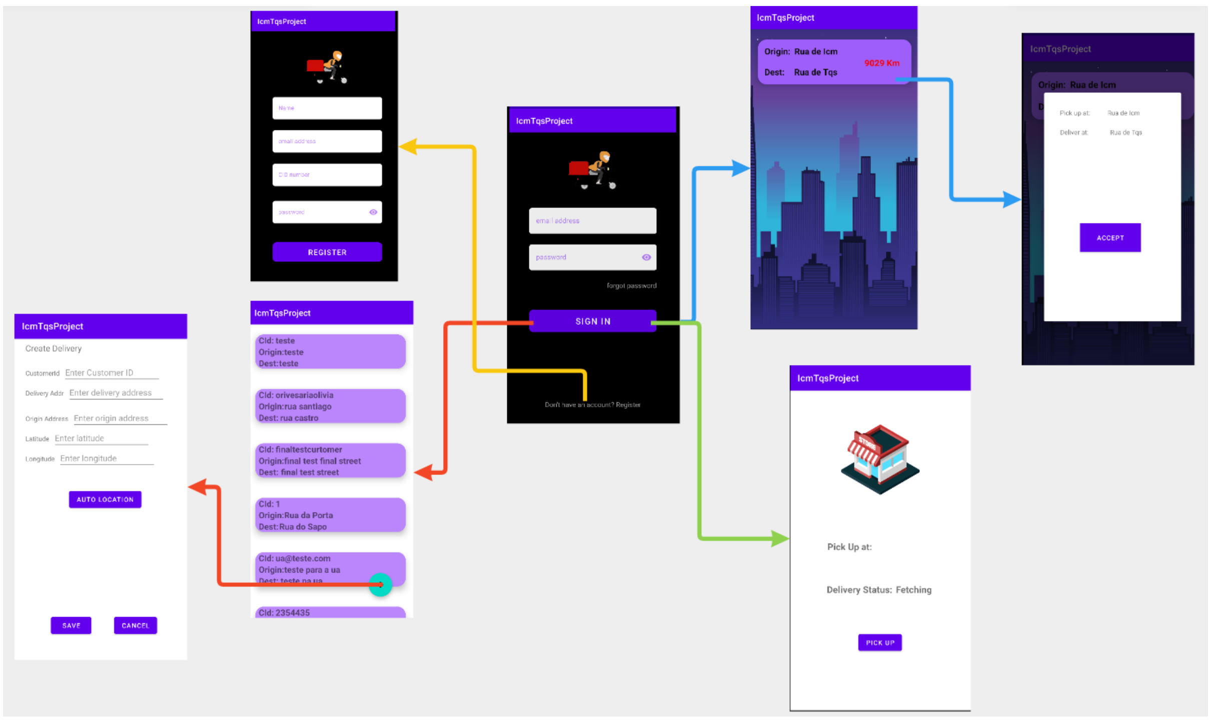
This page contains the deliveries that are still available (Not assigned and not expired.) for the Rider to pick from. Opening one will display a dialog with information about it, which allows the Rider to “Accept” and bid on it. After being created, the delivery has a **10min Auction** where it accepts Bids from Riders. After this time, the closest rider to the destination is picked and a **push notification** is sent to them (Generated using the Karn/notify library[2]). They’ll be directed to the **Tracking Page** whenever they log in, until the delivery is finished.

This page uses a Recycler view to display the deliveries fetched from the API, via a Repository, either when the app is opened or when the user logs in, depending on if the user was already logged in prior to opening the app. The Rider’s distance to the delivery drop-off point is automatically calculated and displayed here.

Tracking Page

Once the user has bid and won a delivery, they are directed to this page while the delivery isn’t complete. They can update the delivery status via a button press that simultaneously updates the View and requests the **External API** to update its status there as well. In the last Stage, the user needs to **Scan the QR code corresponding to the delivery** to proceed, using the **camera.**

Implemented interactions



Project Limitations

There are issues retrieving the data associated with the delivery once the Rider 'wins' the bid - it is fetched from the server but something's going wrong while fetching it to display.

3 Conclusions and supporting resources

Lessons learned

The major problems we had were getting the separate components we learned about to work together, as well as implementing some features that should have been simple but ended up taking up a lot of time.

About the course itself, we thought that splitting the work between Flutter and Android leads to a lot of excess workload, having to learn the absolute basics twice, and starting over just as we're getting comfortable is a bit jarring. Also, for an introductory course, teaching the basics and then asking for a final project with such a large gap in complexity is a very large (And time-consuming) jump.

Work distribution within the team

Taking into consideration the overall development of the project, the contribution of each team member is distributed as follows: Rodrigo Lima did 50% of the work, and Camila Fonseca did 50%.

Project resources

Resource:	**Available from:**
Code repository:	https://github.com/Pengrey/ICM
Ready-to-deploy APK:	https://github.com/Pengrey/ICM/blob/main/Android/Project/Deliverables/app-debug.apk

Reference materials

[1] GitHub - BirjuVachhani/locus-android: An Awesome Kotlin Location library to retrieve location merely in 3 lines of code

[2] GitHub - Karn/notify: Simplified notification construction and delivery for Android.