Technical Project Report - Flutter Module

Project X - Community collaboration platform for bicycle users

Subject:	Mobile Computing
Date:	Aveiro, 9th of January 2023
Students:	103154: João Fonseca 103183: Diogo Paiva
Project abstract:	This Flutter based application provides a community collaboration platform for bicycle users. It provides a map using CyclOSM (a custom OpenStreetMap layer), allows the creation of points of interest using the camera, recording of routes using the location and getting directions from any location to a destination by calling GraphHopper, an external routing engine. As a community, users can rate points of interest and level up in the ranking system. The backend is deployed on AWS and was developed using FastAPI. The authentication mechanism is provided by AWS Cognito, the database to store the information is provided by Amazon RDS and we used Amazon S3 to store all the images.

Report contents:

1 Application concept

This Flutter app was made for people that like to ride a bike, for fun or as a sport. It acts as a community collaboration platform for bicycle users. It provides a map using CyclOSM (a custom OpenStreetMap layer), and allows the creation of points of interest using the camera, recording of routes using the location and getting directions from any location to a destination (it can have multiple intermediate points) by calling GraphHopper, an external routing engine. As a community, users can rate points of interest and level up in the ranking system. This encourages the users to create more POIs and rate them, gaining experience points and climbing levels in return.

2 Implemented solution

Architecture overview (technical design)

In this application, we used cubits for the connection between the logic and the UI. We have 8 different cubits:

- Geocoding: to translate between location and coordinates (and vice-versa).
- GraphHopper: to fetch route information from a set of points (lines on map, navigation instructions).
- Map: to allow for interaction with the map.
- POI: to manage the points of interest that are visible in the map and also their filtering.
- Position: to keep the user's last known position on the map.
- Profile: to create, update or fetch user's profile.
- Route: to manage displayed routes, created routes (with points on the map), or to track a route (using the user's location).
- Token: to get the user's authentication token for using the API endpoints.

Our UI is constructed using the Material module of Flutter, and is organized in different screens. These screens include the map view (which is the main screen), the user details page, the list of routes, the list of POIs, and the forms to sign-up a user, to confirm the code that is sent to the email, to sign-in the user, add user information after registration, change user profile details and to add a new point of interest. It is possible to navigate between pages, thanks to Navigator from Flutter.

The backend is deployed on AWS. We have an API developed in FastAPI, and each endpoint is protected, requiring the user's authentication token to be able to access the endpoints. We also have a custom configured instance of the GraphHopper routing engine, optimized for generating bicycle routes. Both the API and GraphHopper instances are running on Amazon ECS Containers. Authentication is done using AWS Cognito and was configured on the Flutter application through Amplify[1]. The database runs on Amazon RDS and stores all the information (except the images) about points of interest, recorded and created routes, and users. We used Amazon S3 to store all the images, from points of interest and the users. Geocoding and reverse geocoding services used to search for locations on the map and to get the names of the start and destination points of a route are provided by the HERE API.

In the database we store points of interest (POI), users, POI status, routes, and points. The relation between them can be seen in the diagram below.

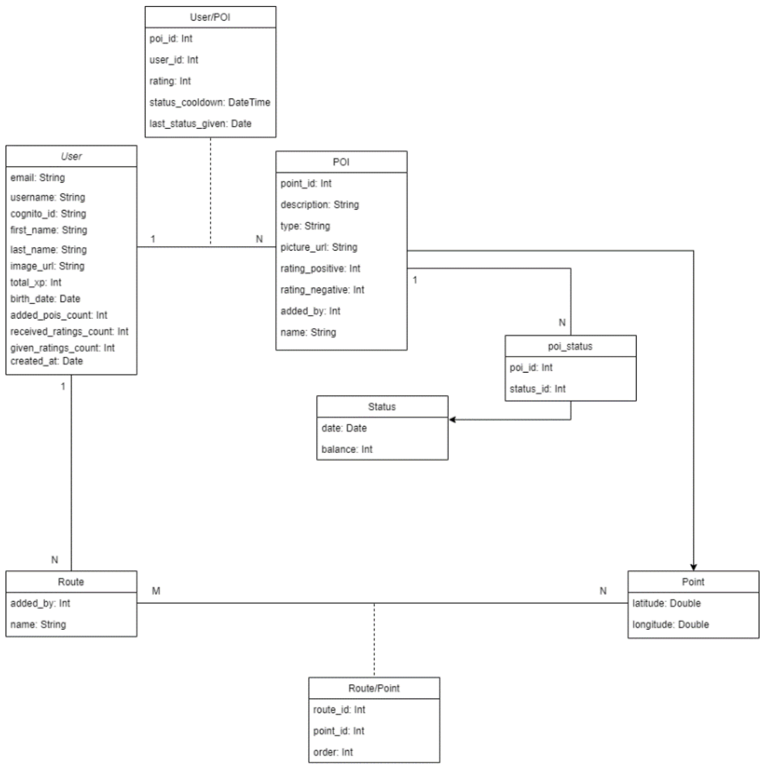


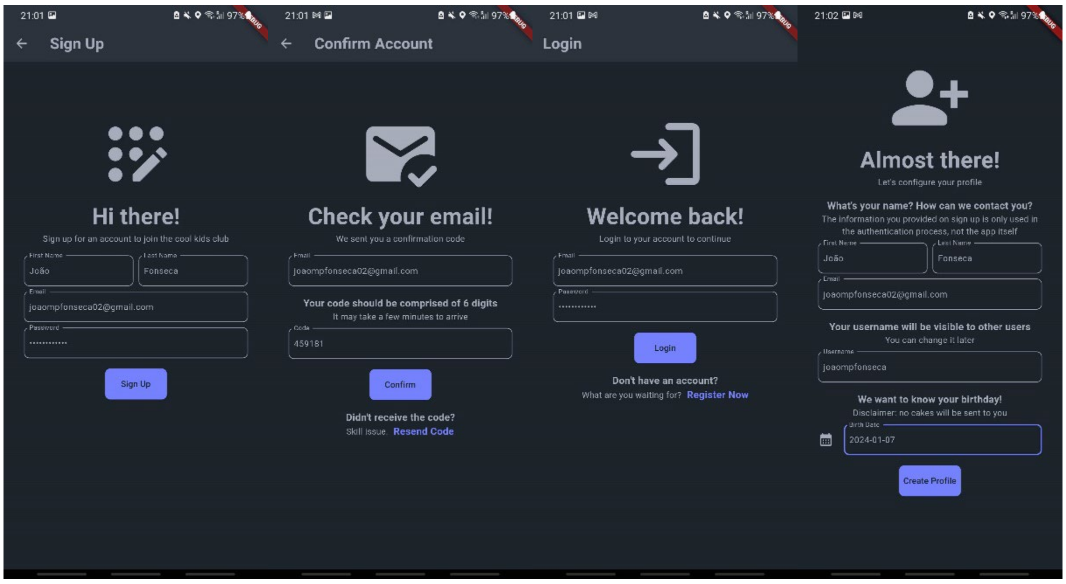
Fig. 1 – Database diagram.

To have access to the user's current location, either to display on the map, record a route or add a new point of interest, we had to include the right access permissions the Android Manifest. We used the geolocator Flutter package. We receive location updates through a callback and each time there is a change, the position cubit retains that valued. The user's location in the position cubit is used by the map and route cubits.

To add a profile picture and a new point of interest, the user needs to provide a picture from either the device's storage or the camera. In the case of a locally stored image, the user is able to access the device's file system and select the preferred image. In the case of the camera, the app opens the camera and the user can take a photo. Until the user submits the form, the picture is stored locally. Only after submission, it is sent to the API and the API takes care of uploading the photo to AWS S3.

For the API to know which user is logged in, if there is one, we have to pass in each call a JWT Token that is available in the Amplify module.

Implemented interactions


Sign Up

Hi there!

Sign up for an account to join the cool kids club

First Name: João
Last Name: Fonseca
Email: joaompfonseca02@gmail.com
Password: ••••••••••••

Sign Up

Confirm Account

Check your email!

We sent you a confirmation code

Email: joaompfonseca02@gmail.com

Your code should be comprised of 6 digits
It may take a few minutes to arrive

Code: 459181

Confirm

Didn't receive the code?
Skill issue. Resend Code

Login

Welcome back!

Login to your account to continue

Email: joaompfonseca02@gmail.com
Password: ••••••••••••

Login

Don't have an account?
What are you waiting for? Register Now

Almost there!

Let's configure your profile

What's your name? How can we contact you?
The information you provided on sign up is only used in the authentication process, not the app itself

First Name: João
Last Name: Fonseca
Email: joaompfonseca02@gmail.com

Your username will be visible to other users
You can change it later

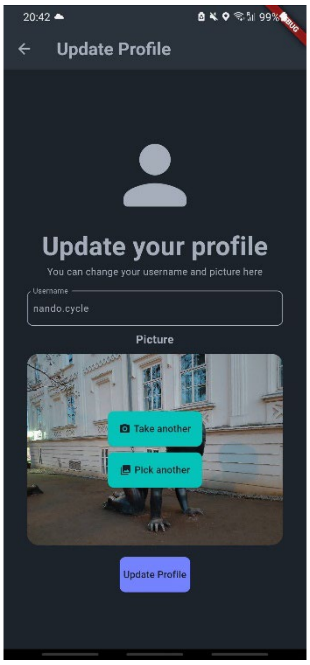
Username: joaompfonseca

We want to know your birthday!
Disclaimer: no cakes will be sent to you

Birth Date: 2024-01-07

Create Profile

Fig. 2 – Sign up and login.


Update Profile

Update your profile

You can change your username and picture here

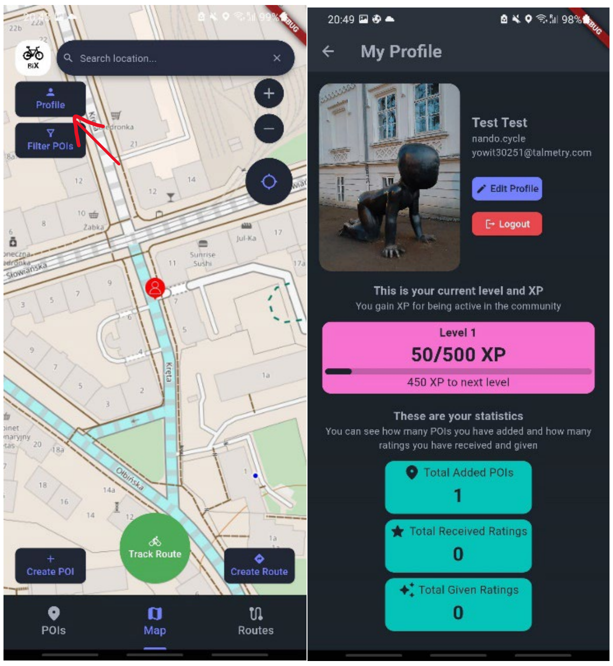
Username: nando.cycle

Picture

Take another
Pick another

Update Profile

Fig. 3 – Update user profile.



20:49

My Profile

Test Test
nando.cycle
yowit30251@talmetry.com

Edit Profile
Logout

This is your current level and XP
You gain XP for being active in the community

Level 1
50/500 XP
450 XP to next level

These are your statistics
You can see how many POIs you have added and how many ratings you have received and given

Total Added POIs	1
Total Received Ratings	0
Total Given Ratings	0

Fig. 4 – Check user profile.

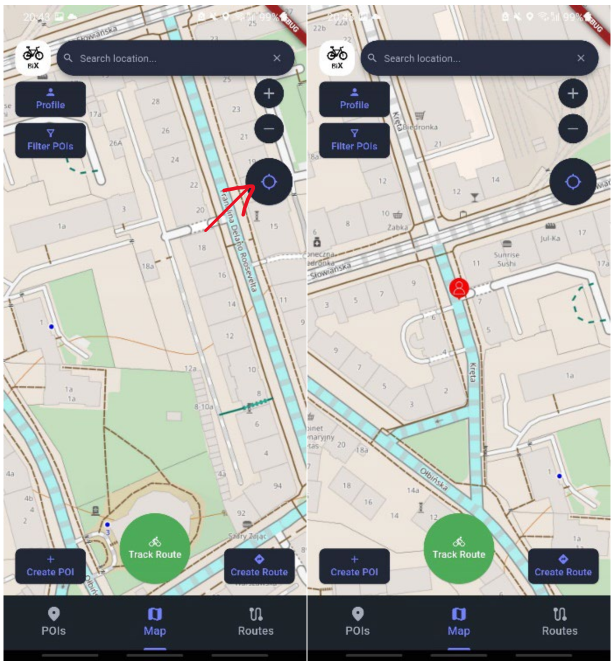


Fig. 5 – Go to user’s location.

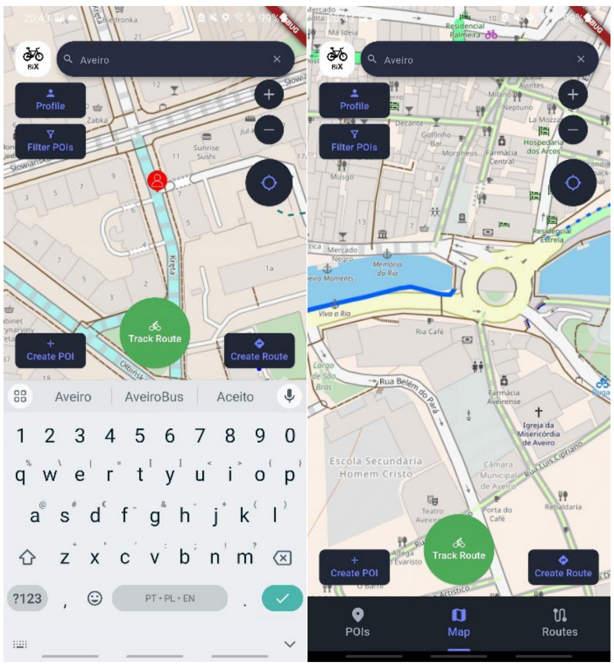
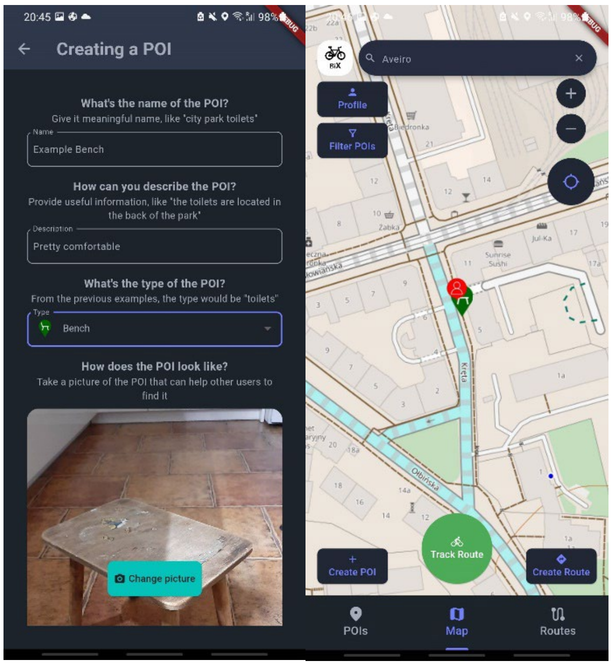


Fig. 6 – Search for a location.



20:45 98%

Creating a POI

What's the name of the POI?
Give it meaningful name, like 'city park toilets'

Name: Example Bench

How can you describe the POI?
Provide useful information, like 'the toilets are located in the back of the park'

Description: Pretty comfortable

What's the type of the POI?
From the previous examples, the type would be "toilets"

Type: Bench

How does the POI look like?
Take a picture of the POI that can help other users to find it

Fig. 7 – Add a point of interest.

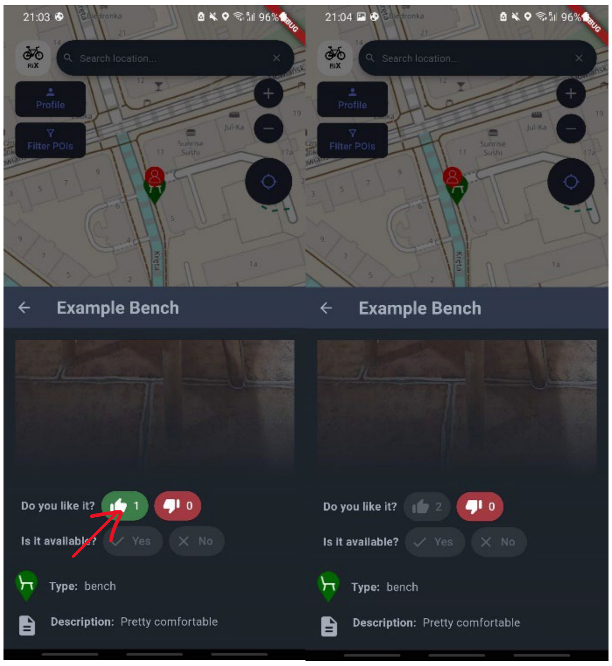


Fig. 8 – Rate a point of interest (with a different account).

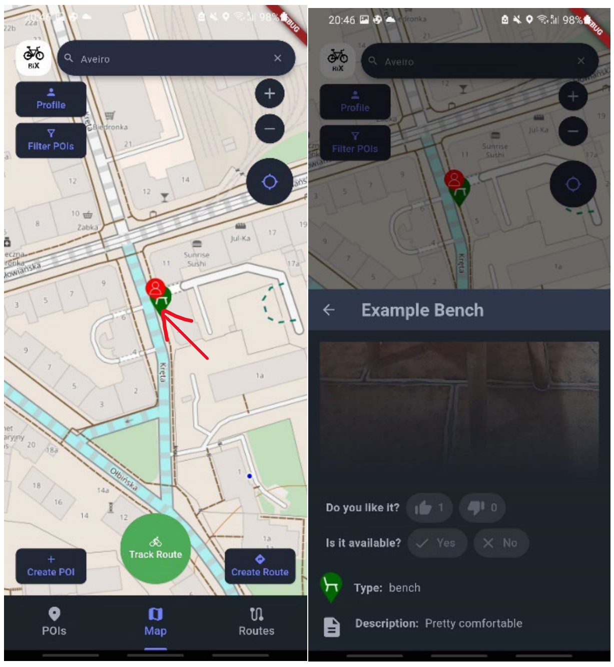


Fig. 9 – Check point of interest details.

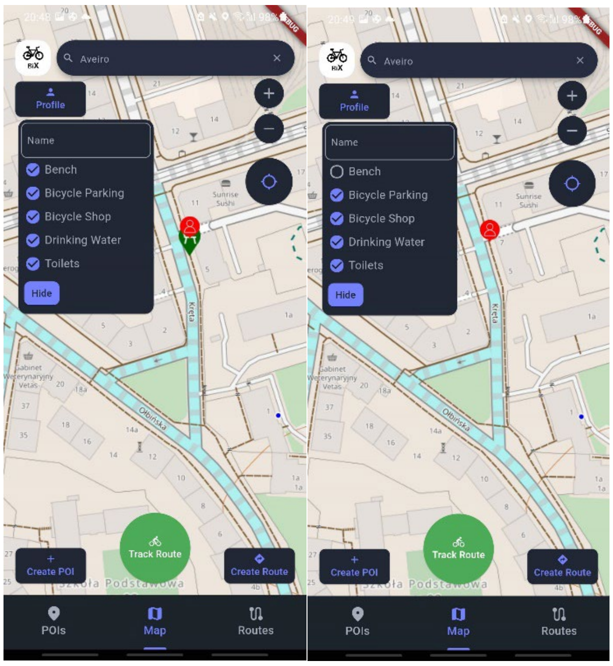


Fig. 10 – Filter points of interest.

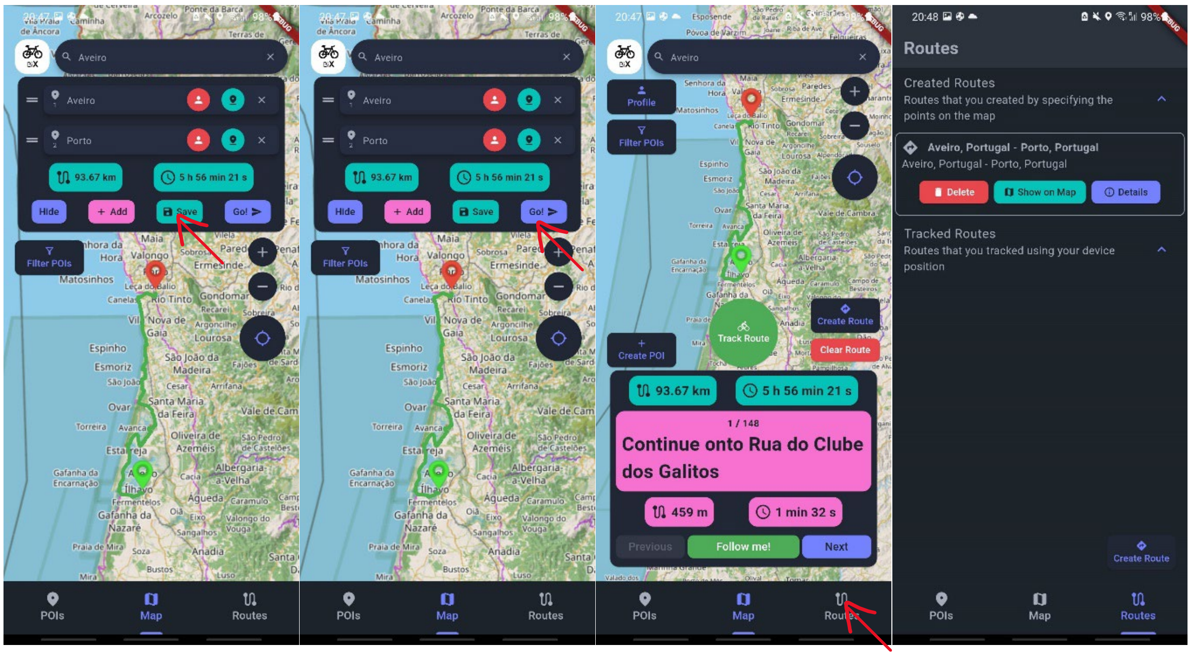


Fig. 11 – Create a route, save a route and see navigation instructions.

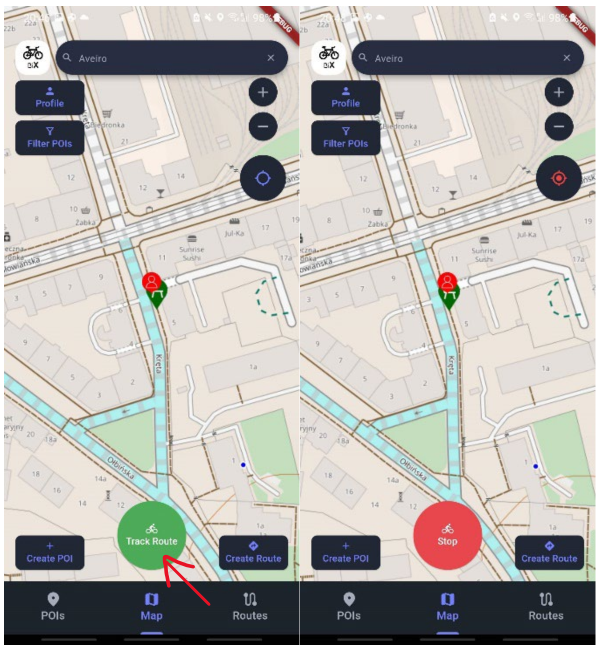


Fig. 12 – Track a route.

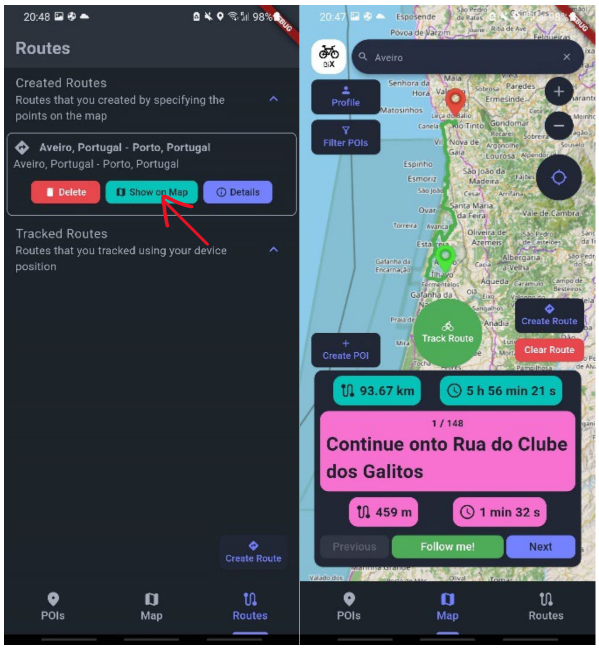


Fig. 13 – Display a route on the map.

Project Limitations

It is not possible to modify information of the points of interest and of the routes. The app doesn't work if not connected to the internet. The directions only appear by writing.

3 Conclusions and supporting resources

Lessons learned

We had problems with the JWT Token, because in the documentation of Amplify there is no reference on how to obtain the token so we had to research is various websites and finally we were able to obtain the token from a deprecated, but still working, function of Amplify.

The use of Cubits allowed for an easy centralization of the business logic of the application. They are referenced in the root widget, and are passed down the widget tree so that every widget can easily access them. This allowed for a better code organization and decoupling of the logic and the UI.

The use of AWS was challenging and was something we had never used before. It allowed for the deployment of the backend on the cloud, and it is done in a highly scalable manner using ECS instances.

Work distribution within the team

Taking into consideration the overall development of the project, the contribution of each team member was distributed as follow: João Fonseca did 50% of the work, and Diogo Paiva contributed with 50%.

Project resources

Resource:	**Available at:**
Code repository:	https://github.com/joaompfonseca/cm-flutter-project
Ready-to-deploy APK:	https://github.com/joaompfonseca/cm-flutter-project/blob/main/release/v0.0.3/android/app-release.apk

Reference materials

[1] Amplify Documentation for Flutter
https://docs.amplify.aws/flutter/