Mobile Computing 2023/2024 - Flutter Module

Project Report - Run Route

Group: g05
Students: Pompeu Costa (103294), Inês Santos (102484)
Repository: https://github.com/pompeucosta/projeto_cm_flutter

08/01/2024

Contents:

What is Run Route

Run Route is a running companion application, meaning it is an application for people that like to go running. The application is useful for people that want statistics about their running session. Run Route tracks duration, distance, steps taken, average speed, top speed and calories burned. It also tracks the user path in a map so the user can see the path travelled.
Run Route also saves a history of sessions, which the users can use to identify where to improve, for example,if they can run a higher distance in less time, or lose more calories in less time.
Users can create presets which are similar to clock alarms, where the user defines a name for the preset, the desired duration and distance and if he wants *Two Way* or not. Run Route uses this information during a session to warn users, ie. send notifications, when the desired duration/distance is reached. If the preset has *Two Way* enabled then Run Route will warn the user when he reaches half of the desired duration/distance. The point of *Two Way* is to start and end in the same location. Let's imagine the user wants to run for 30 minutes then after running for 15 minutes the user should go back, thus taking another 15 minutes and thus taking in total 30 minutes and ending at the same location he started.

Solution

In this section we will talk about the solution, including the requirements and features implemented.
This application was developed in flutter dart and uses some third party libraries, listed in Table I.

Library	Its use	Source
flex_seed_scheme	Dark mode and light mode generator based on three seed colours (primary, secondary, tertiary)	https://pub.dev/packages/flex_seed_scheme
hive	Persistence library to store data locally. We use it to store presets and sessions.	https://pub.dev/packages/hive
bloc	We use it to create a single source of information for the different screens in our application.	https://pub.dev/packages/flutter_bloc
flutter_background_service	We use it to create a foreground service when the user has a session going on. This way the user can minimise the app without interrupting the session.	https://pub.dev/packages/flutter_background_service

permission_handler	As the name suggests, we use it to handle permissions.	https://pub.dev/packages/permission_handler
flutter_local_notifications	Library to send notifications.	https://pub.dev/packages/flutter_local_notifications
flutter_map	Used to display and control the map and all the information displayed in it.	https://pub.dev/packages/flutter_map
latlong2	Provides common latitude and longitude calculations.	https://pub.dev/packages/latlong2
url_launcher	Used by the RichAttributionWidget of the map.	https://pub.dev/packages/url_launcher
camera	Used to access camera functionalities.	https://pub.dev/packages/camera
path	Library for path manipulation.	https://pub.dev/packages/path
path_provider	Library used for finding a storage location for the photos.	https://pub.dev/packages/path_provider
geolocator	Provides access to the device's location service.	https://pub.dev/packages/geolocator
geocode	Used to obtain the name of the city from the current location.	https://pub.dev/packages/geocode
pedometer	Provides access to the device's step counter sensor.	https://pub.dev/packages/pedometer
intl	Used for number formatting.	https://pub.dev/packages/intl

Table I: Third party libraries that we use in our application

Requirements

Table II shows a list of requirements for the application. The requirements are in no specific order.

Requirements	Description
Req #1	The application shall run in the background when the user has an on going session
Req #2	The session screen shall show a map with the user's path drawn in it. The user's path is the path the user travelled during the session.
Req #3	The session screen shall show the following statistics: current duration of the session, distance travelled, steps taken, calories burned,top speed and average speed
Req #4	The user shall be able to create a preset and save it for future use
Req #5	The user must select a preset to start a session. The session then uses the information of that preset to determine when to send notifications.
Req #6	The session shall send notifications, if it has permission to, when the user reaches the desired duration and another notification when the user reaches the desired distance.
Req #7	If the preset has *Two way* active then the application shall send a notification when the user reaches half of the desired duration and another when the user reaches half of the desired distance, so that he knows when to go back.
Req #8	The homepage shall show the last session the user made. If there isn't one then it shall show a greetings message.
Req #9	A preset shall contain a desired duration, a desired distance, a name and a *Two way* boolean option.
Req #10	The app shall have a history screen that shows a list of sessions the user made.
Req #11	The history screen shall have the sessions sorted by most recent to oldest

Req #12	The history screen shall show the date and the duration of each session
Req #13	Upon clicking in a session in the history screen, the app shall redirect the user to the session details screen
Req #14	The session details screen shall show the map with the path drawn in it and the statistics defined in Req #3
Req #15	During the session, the user should be able to take and save photos.
Req #16	The session details screen should allow the user to view the photos taken during the session.

Table II: Application Requirements

Features Implemented

All requirements have been fully implemented.

Architecture

RunRoute is composed of many widgets, there are many provided by flutter like Text, some Buttons (TextButton, ElevatedButton,...), input fields, etc… Other widgets are created by us, to represent a screen/page for example. While there are too many widgets to talk about and we can't talk about all of them, we're going to talk about some more important widgets like providers and BLoCs. Diagram I show a widget tree where it's possible to see where BLoCs are being created and used. We create the BLoCs at the top of the tree so that all widgets below have access to them. We create them in the Providers widget which contains a MultiBlocProvider, however those BLoCs require repositories which are created at the MyApp widget that contains a MultiRepositoryProvider. Since MyApp is created above the Providers widget, the BLoCs have access to those repositories.
On the bottom part of the tree is where we use the BLoCs mentioned previously. We use them through BlocBuilders and BlocConsumer. Although both are very similar, BlocConsumer allows to listen for event changes, so when the BLoC sends a new event, instead of building the widget again, we can simply listen to the event and change whatever we wish based on the event. We may not even change at all if it isn't necessary, which happens for a lot of events. For example, in the RunningSessionPage widget, we listen for the event that says that the session is over, so that we can tell SessionsBloc to update its list, since there is a new session.

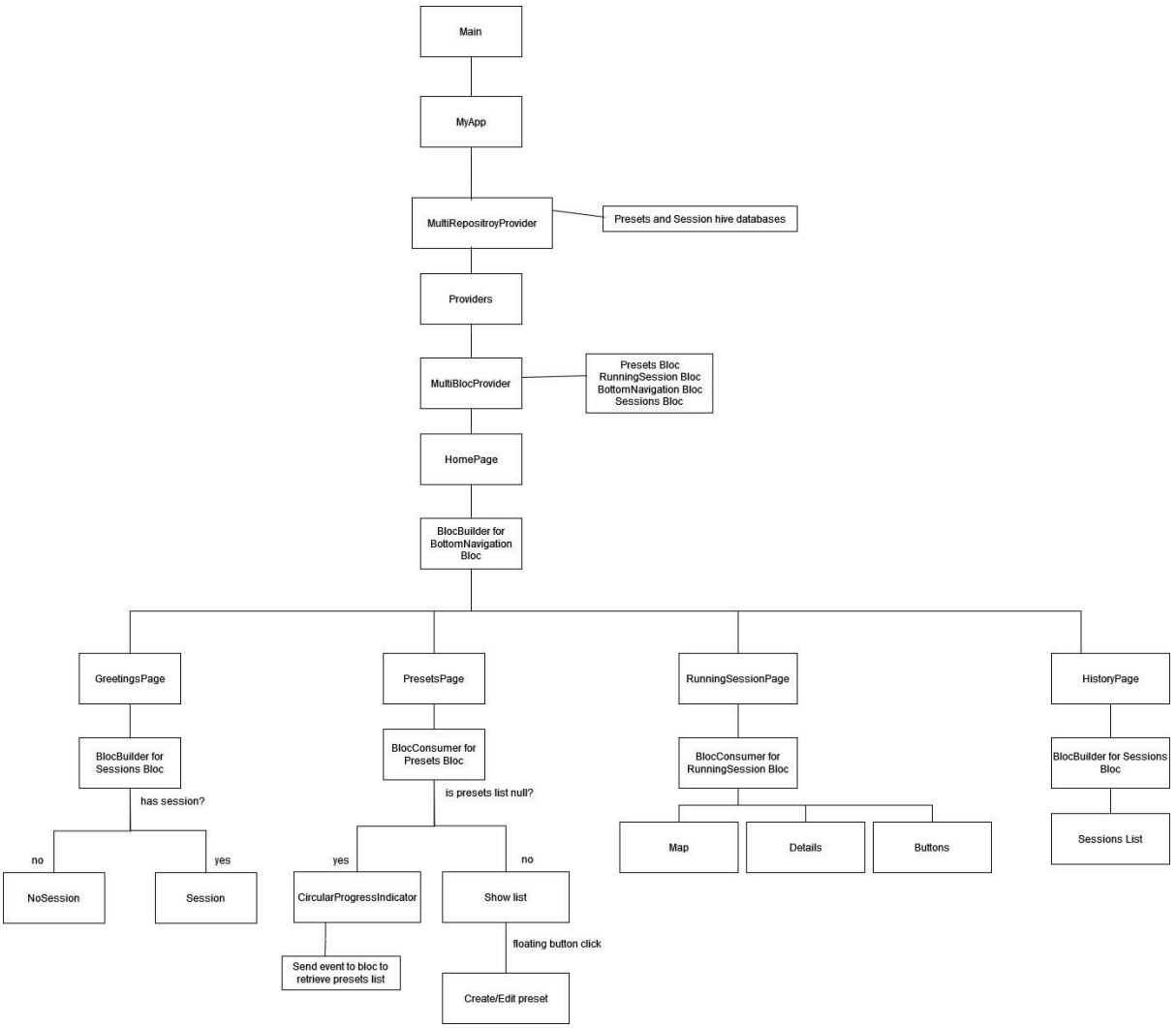

Diagram I: Widget tree showcasing where we create and use BLoCs and Providers

Assessment

Although we achieved all of our objectives, we wanted to be able to keep the app running in the background when closed by the user if the user has a session going. However we did not manage to get that done but it works if minimised which is enough for us. Another issue found is in the geocode functionality, which provides the nearest city's name. This functionality is limited and doesn't always work due to the API's limit of 1 request per second for all free users combined.

Pompeu Costa: 50%
Work done:

- Presets and Session hive databases
- BottomNavigation, Presets, RunningSession and Sessions BLoCs
- Homepage, GreetingsPage, PresetsPage, RunningSessionPage, HistoryPage
- BLoC builders and consumers
- Notifications
- Foreground service
- MultiBloc and MultiRepository providers

Inês Santos: 50%
Work done:

- Camera and photo storage functionalities
- Homepage, RunningSessionPage and SessionDetails
- Location tracking service
- Map functionalities
- Session details calculation (distance, top/average speeds and calories spent) and access to sensors (step counting)

Usage Manual

The main flow of the app starts by selecting a session preset with the desired parameters or creating a new one if it doesn't exist (fig.1).

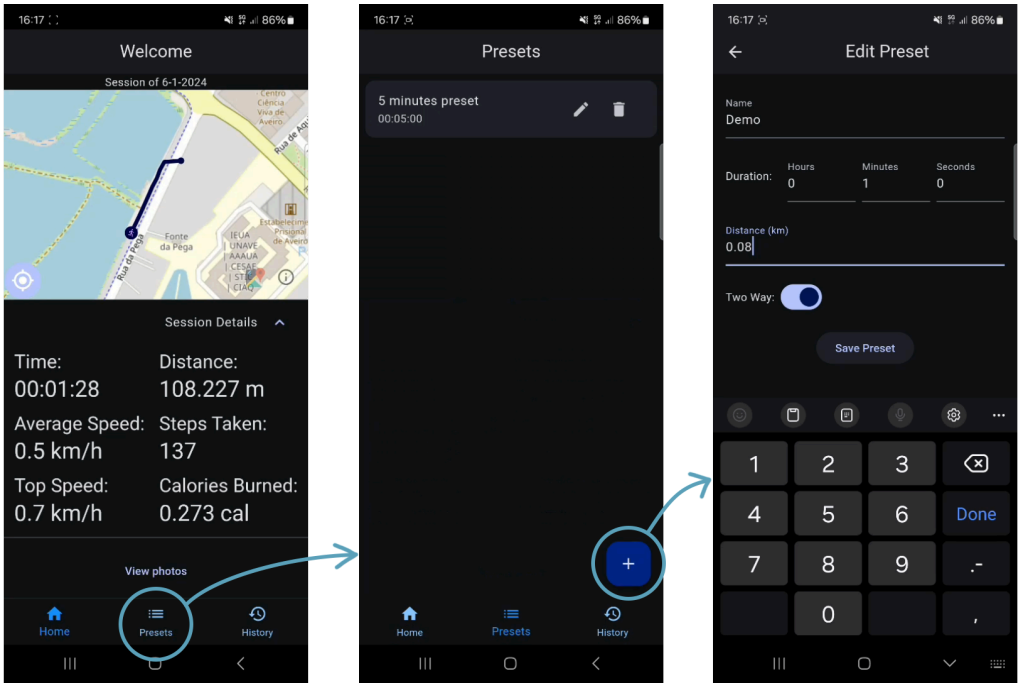

Fig.1 - Steps to create a new preset

After the preset is created it can be selected to immediately start the session, during which the statistics will be tracked and displayed in real time.

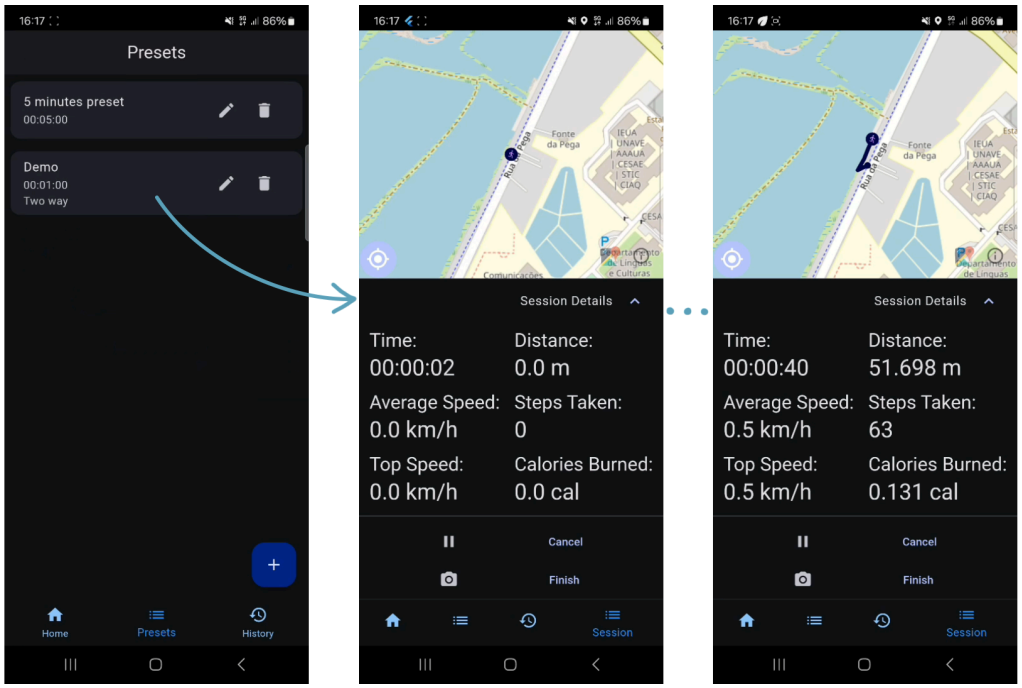

Fig.2 - Session in progress screen.

During the session, the user can take and save photos, which will be accessible from the session details screen after the session is concluded (Fig.3).

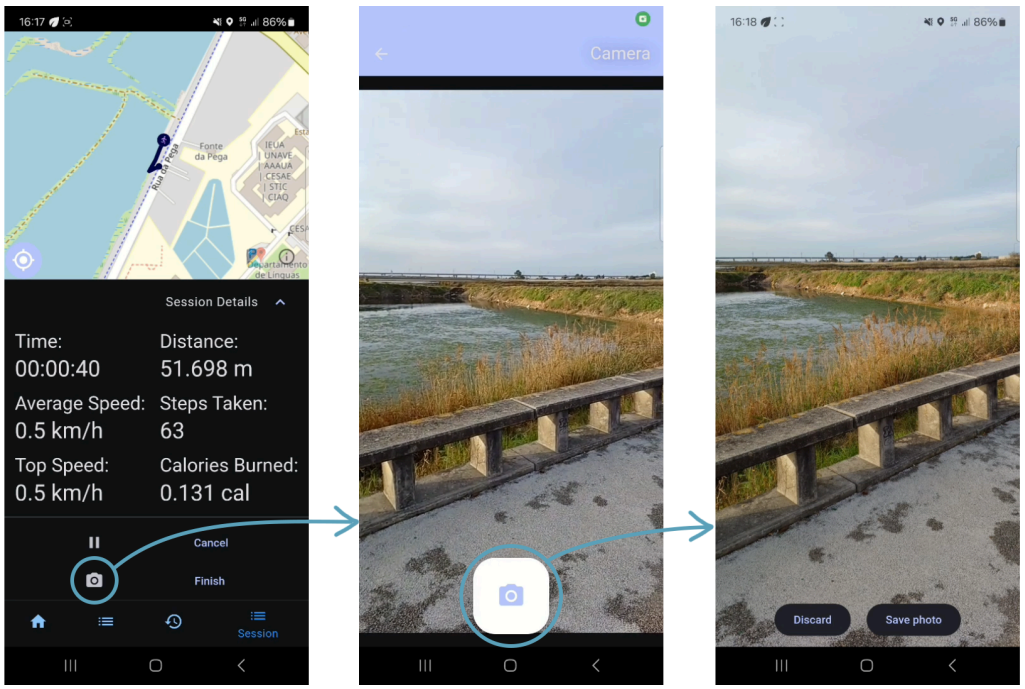

Fig.3 - Steps to take a photo during the session.

Once the selected distance or time objectives have been achieved, the user is notified. If *Two Way* was selected, a notification will also be sent when the distance and/or the time values reach half of the objective (Fig.4).

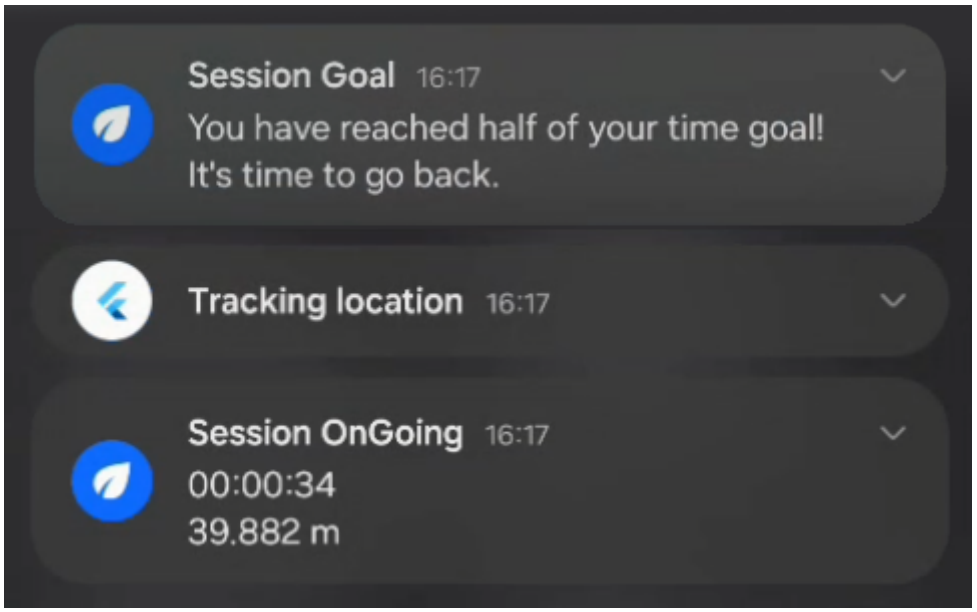

Fig.4 - Notifications during a session.

Once the user finishes their run, they can finish the session and check their results, as well as view any photos they took during it (Fig.5).

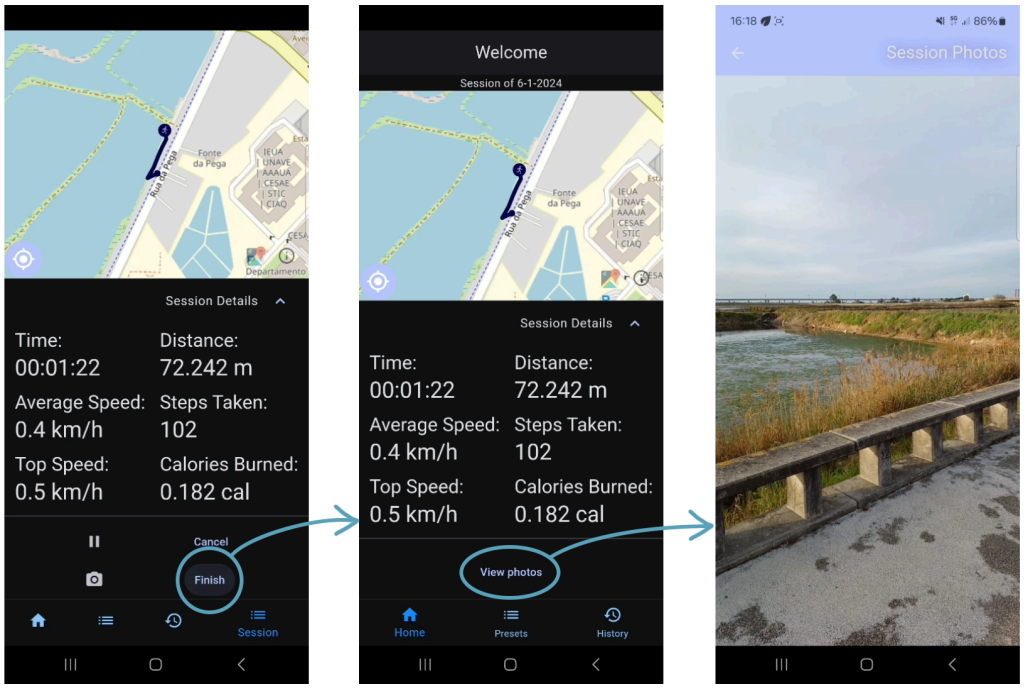

Fig.5 - Steps to finish and view a session's photos

Users can always check their previous runs in the history page (Fig.6).

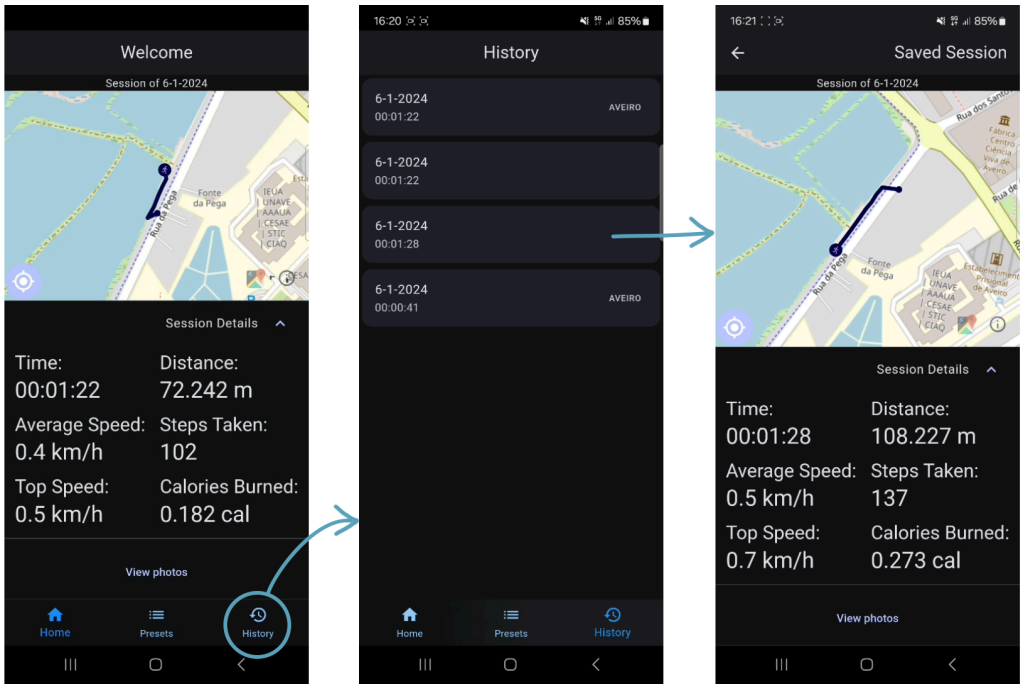

Fig.6 - Steps to view a previous run's details.

On both the “Session in Progress” and the “Session Details” screens the user can hide the details to see and interact with the map. The button in the bottom left corner of the map resets the center and zoom of the map to display the whole path.

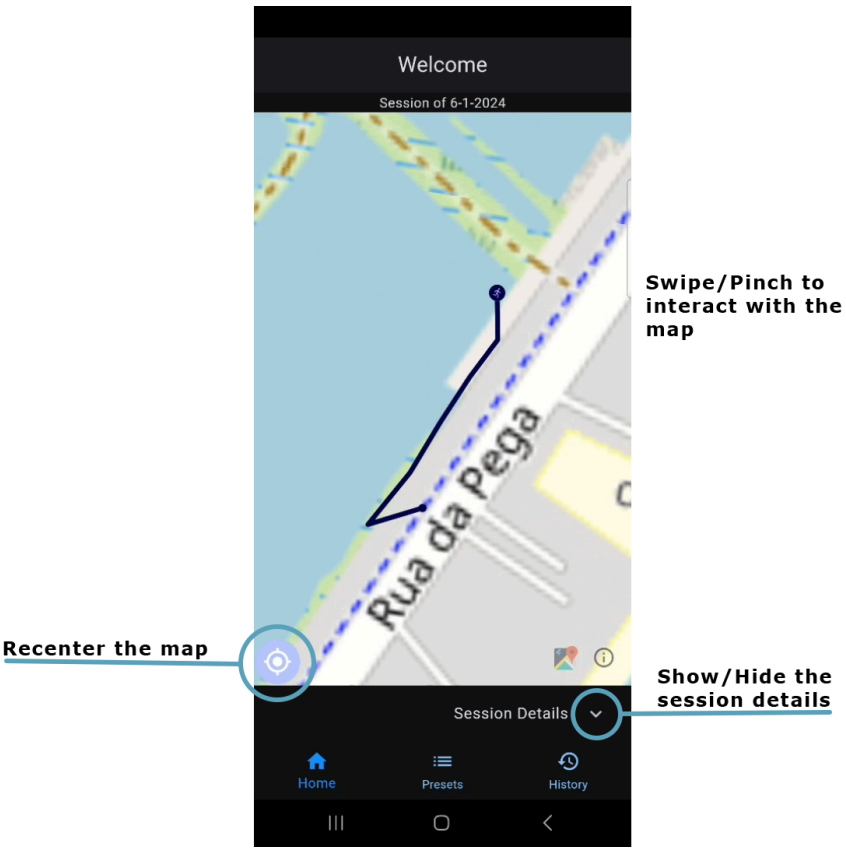

Fig.7 - Map controls.