Technical Project Report - Android Module

V Tracker

Subject: Computação Móvel

Date: Aveiro, 02/06/2020

Authors: 100720: Fabio Cecchinato
85135: Rui Silva
85134: Tiago Feitor

Project abstract: Our application is intended to give the users a way to know the locations of COVID-19 infected people, by tracking the position and showing on the map where there are such users.

Table of contents:

1 Application concept

The V Tracker main purpose is to allow users to see the location of other infected users, so they can be aware of the zones they might have to pass by. The user creates an account, and the app tracks his positions over time, working even in the background. The user can change his status if he is infected, so the other users can be made aware about him. It is also possible to see some statistics about the worldwide spreading of disease and to monitor the user's last known locations.

2 Implemented solution

Architecture overview

The main component of our application is the MainActivity, with a layout defined by a DrawerLayout containing the application bar and the navigation view. This is also divided in two blocks:

- NavigationHeader, with the icon of the app and the names of the creators
- MenuDrawer, with the button to navigate to the fragments where we implemented the functionalities of the app

We started from the template app with a drawer which is available in Android Studio.

Our project has also other activities: for the login/register operations, for scanning a QR code and for the foreground service.

We used SharedPreferences to check if the user is logged in or scanning the QR code. We had to do this because the foreground service starts whenever the onStop() method of MainActivity is called, so also when from MainActivity we go to one of the other activities. Using SharedPreferences, we could add conditions in order to avoid the call to the service when it is not needed. The user needs to be logged in to use the app. By closing the app while logged in, the app will close its window but start the background service (it stays in the notifications). By closing the app while not logged, the app will close normally.

We used the GPS to get the current location, the accelerometer to send the location to the database only if the device is moving enough (to prevent overload of information) and the camera to read the QR codes.

While the app is in foreground, the current location is sent to the database every 30 seconds without other checks, instead during the background phase this happens every 5 minutes, only if the conditions on the accelerometer are satisfied (there is a threshold here that probably is not perfectly tuned).

We used string values for every piece of text which has to be displayed to the user: in this way, our app can work in three languages (Italian, Portuguese and English) depending on the language settings in the device.

To get the information from the used API, we utilized the Retrofit library, that uses the http method GET to retrieve the information from the API and parses it to the Post class

created to receive the information of the API.

In the AppInfo fragment the coordinates the user has been is shown in the list. We used an external library to give the street name based on the location (Reverse GeoCoding) only for the few first coordinates in the list due to it taking a long time to process those names.

Project structure and files

1. **Activities and fragments**

   a. **Login and Register Activities**

      LoginActivity.java

      RegisterActivity.java

   b. **Main Activity**

      MainActivity.java - activity that shows the menu and different fragments.

      i. **Fragments**

         ui/appinfo/**AppinfoFragment.java** - some info about the app

         ui/map/**MapFragment.java** - main fragment of the app. Shows your location and other infected users.

         ui/report/**ReportFragment.java** - see current user status and change it with QR code.

         ui/signout/**SignoutFragment.java**

         ui/userinfo/**UserinfoFragment.java** - shows the list of positions the user has been.

         ui/covid/**CovidFragment.java** - shows the information about the global info about Covid-19 retrieved from API

         ui/covid/**CountriesFragment.java** - shows the list of countries the API has info on. (uses ui/covid/**CountryAdapter.java** as an adapter for RecyclerView)

         ui/covid/**CountryInfo.java** - shows the info on Covid-19 about the selected country

   c. **ScannedBarcodeActivity.java** - Activity to read a QR code. Returns to MainActivity and triggers a message to confirm the user status change.

2. **Background**

   LocationForegroundService.java - Service that works on background to save the user's locations

3. **Database**

   Database/**Database.java** - Methods on how to read/write info to the database

   models/**ListAdapter.java** - Adapter for RecyclerView used in the list of user's positions

   models/**ListHolder.java** - Holder for RecyclerView used in the list of user's positions

   models/**Position.java** - Model class for a GPS position

   models/**User.java** - Model class for user's information

   models/**UserList.java** - Model class to retrieve the list of user's positions

4. **Remote API** (ui/covid/api_models)

   Country.java - class that has info about 1 country

   Countries.java - list of countries (for Post)

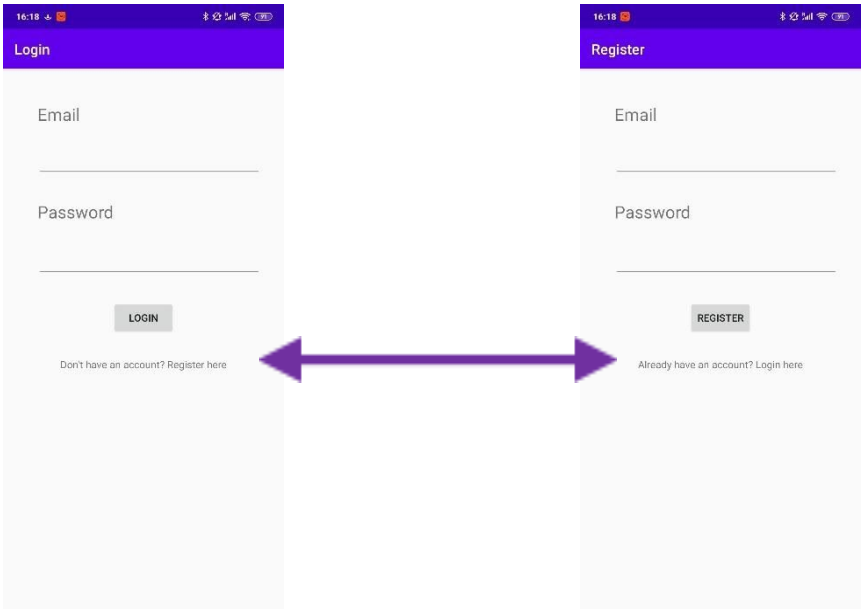
   Global.java - info about the global view on Covi-19

   Post.java - Countries, Global and a Date. Class returned while using Retrofit to access the API

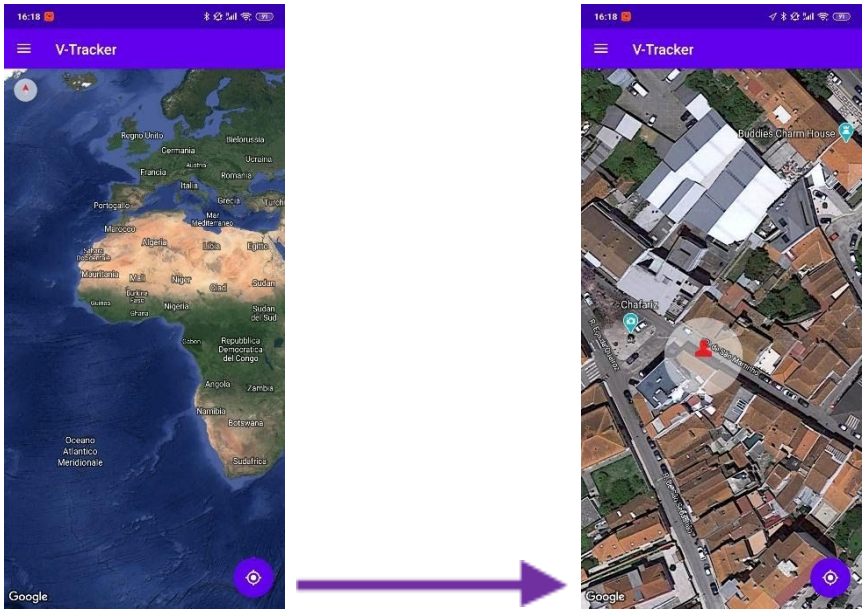
   JsonPlaceHolderAPI.java - interface for the API.

Implemented interactions

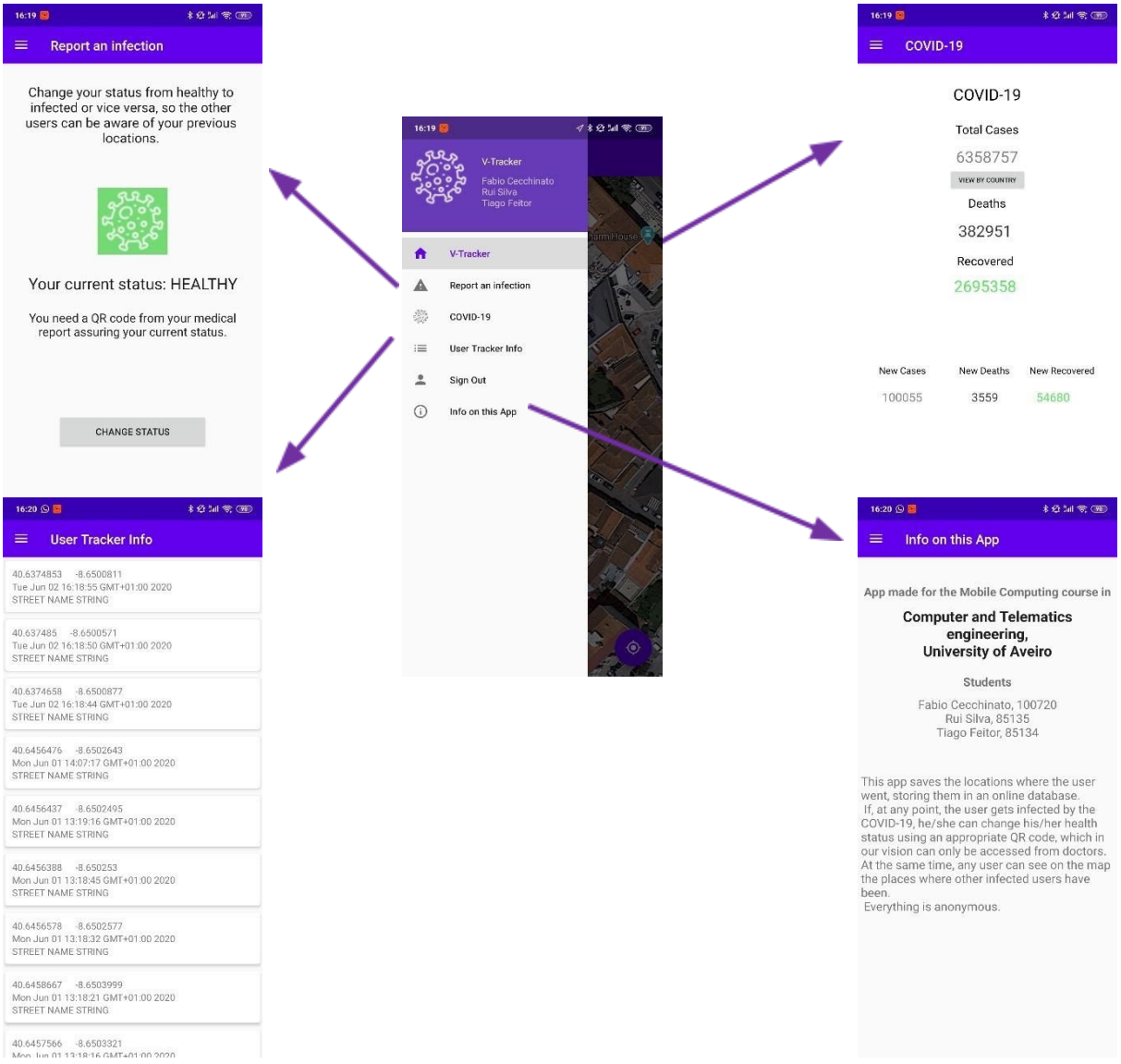
When the app is launched for the first time, the login form appears: here, the user can choose whether to log in (if he already has some credentials) or to register (creating new credentials.



After the log in, the user can see the homepage of the app with the map: he must click on the floating action button in the bottom-right corner to center the map on his current location and start the service. Also, if there are any infected users, they are visible on the map, which is fully explorable.



On the top-left corner, by clicking on the hamburger icon, a side menu will appear. Here we can see the other functionalities of the app: V-Tracker (home page with the map), Report an infection (to change the user status from healthy to infected and vice-versa), COVID-19 (to check statistical information about the disease), User Tracker Info (to monitor the user's latest locations), Sign Out and Info on this App.



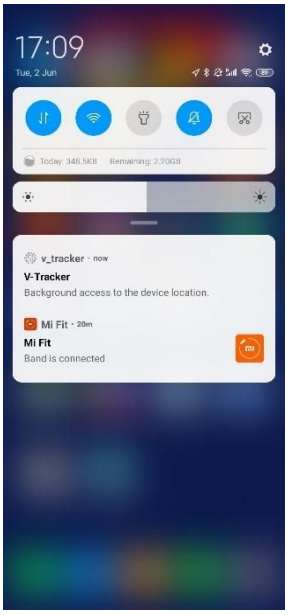
We already talked about the home page. Let us see more in detail the other functionalities:

- Report an infection: here the user can change his status from healthy to infected and vice-versa. By clicking on the button, he can scan the appropriate QR code to set his new status. In our vision, these QR codes should be provided by the doctor who

states the infection/recovery of that user. After scanning the QR Code the user is redirected to the MainActivity (map) where a message appears to confirm the status change.

- COVID-19: here we can see some information about the worldwide spreading of the virus. Clicking on the button, we can select for which country we want to see those pieces of information.
- User Tracker Info: here the user has access to the list of his positions since the moment he created the account.
- Sign Out: button that allows the users to log out from the system, going back to the login screen.
- Info on this App: informative screen with a description of what the application does and the list of the authors.

One important feature of this project is the fact that the location updates work even if the app is not currently in the foreground, using a foreground service together with a persistent notification: whenever the user (being logged in) closes the app, the service starts and the notification tells him that the tracking is happening in background.



Project Limitations

Probably the major limitation of this project is a matter of privacy, because tracking the location of the users is not the best privacy policy practice. In fact, even if everything is anonymous, every user can see on the map the location of the other users if they are infected (and maybe they are inside their houses…). This is the main reason why in real applications of this kind of apps they tend to track not the users but the contact between them, using for example Bluetooth. We decided not to implement this kind of solution, because it is still a quite unexplored field and also because we wanted to stay on the same wavelength of our first Flutter version of this project.

While coding we had a limitation while using operations with the Firebase database. Even though the functions for write operations are in the **Database.java** file, we have to replicate the code in the other files when using those functions because for the operation to be synchronized with the database the code we want to run needs to be in the

OnSuccess() method when accessing the database, like the function **updateState()** in the **MainActivity.java**, that updates the status of the user from infected to healthy or vice versa.

```
public void updateState(boolean isInfected) {
    List<Position> list;
    FirebaseAuth mAuth = FirebaseAuth.getInstance();
    FirebaseFirestore db = FirebaseFirestore.getInstance();

    db.collection("users").document(mAuth.getCurrentUser().getUid()).get().addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>
        @Override
        public void onSuccess(DocumentSnapshot documentSnapshot) {
            User user = documentSnapshot.toObject(User.class);
            user.setInfected(isInfected);
            db.collection("users").document(mAuth.getCurrentUser().getUid()).set(user);
        }
    });
}
```

3 Conclusions and Resources

Lessons learned

The implementation of the background access to the location was definitely easier and feasible when comparing with Flutter, where we were not able to do it. We used a foreground service, to have also a persistent notification showing to the user that the app is running in background.

Key project resources

- Code repository: https://github.com/cecchifabi/v_tracker
- Ready-to-deploy APK: https://github.com/cecchifabi/v_tracker/blob/master/app-debug.apk

Reference materials

Libraries, web services and references that we used:

[1] Reverse Geocoding Library
https://www.daniel-braun.com/technik/reverse-geocoding-library-for-java/

[2] Firebase
https://firebase.google.com/docs/android

[3] Shared Preferences

https://developer.android.com/training/data-storage/shared-preferences

[4] Get current position

https://developers.google.com/android/reference/com/google/android/gms/location/FusedLocationProviderClient

https://medium.com/@droidbyme/get-current-location-using-fusedlocationproviderclient-in-android-cb7ebf5ab88e

https://developer.android.com/training/location/retrieve-current

[5] Foreground service

https://androidwave.com/foreground-service-android-example/