Lab 6 - Geofencing

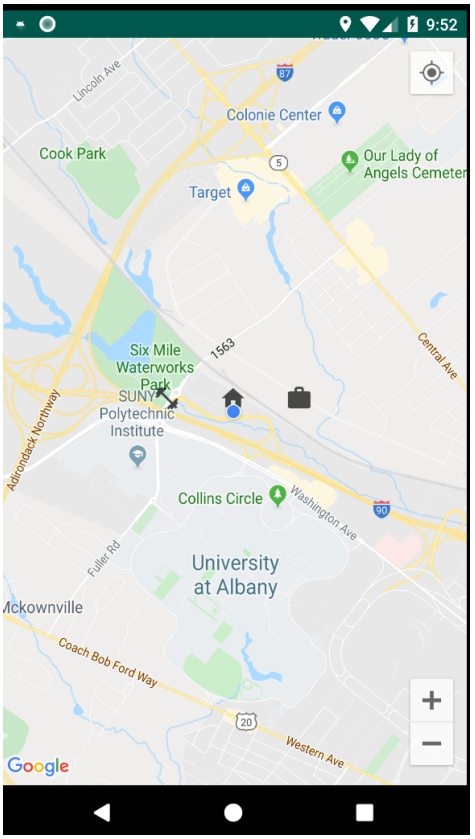
Geofence is a virtual area, corresponding to the actual geographic area. Detecting a user's entrance, exit or dwelling within a geofence enables a number of interesting location-based mobile applications. For example, geofences can be used to issue location-based reminders, we can use geofences to post location-based information, and we can also set geofences dynamically, thus detect when a person is close to one of her friends.

In Android, geofences are supported directly by Google Play Services - you simply have to define where the center of the monitored area is, the radius of the circular area that you want to monitor, and the transition (enter, exit, dwell) that you want to monitor. This lab we are going to build an app with location-based reminders. The app will remind its users to:

This lab we are going to build an app with location-based reminders. The app will remind its users to:

1. go to the gym after work;
2. once in the gym for some time (dwell), synchronize their smart wristbands;
3. remotely turn the heating on, as they are getting closer to home. Three geofences, at work, gym, and home, will be used.

Application Scaffolding



We are going to implement a rather advanced application in a short period of time, thus, we won't start from zero. In Android Studio open a new project from Version Control and type in https://github.com/vpejovic/GeofencingApp/

Our app is going to rely on Google Map tiles. Using this requires a Google Maps API key. Follow the instructions described here and set `google_maps_key` parameter in `google_maps_api.xml` file in `res` directory of your project to the value of your key.

Ensure that the project compiles and runs on your emulator or phone. You should see a map (the actual location will depend on what you set in the emulator) similar to the one on the right. Three movable icons, for work, home, and gym, should be on the map. Test whether moving the icons works.

To test our app we are not going to run around the city to trigger geofences, but will use the emulator and mock locations provided through a pre-collected trace. First, download the gpx trace from here. Then, open the extended controls of the emulator (three dots) and put `46.05178` for latitude and `14.49968` for longitude and click Send. Then, open Google Maps on the emulator and click on the My Location icon. This should set the emulator's location to Ljubljana. Next, back in the extended controls load the gpx trace you previously downloaded. Finally, open our Geofencing App and using the extended controls play the gpx trace. You should see the user's location moving through Ljubljana.

Brief Tutorial on Geofencing in Android

Geofences are set through **GeofencingClient**. This class has a method `addGeofences` that takes the following arguments:

- An instance of **GeofencingRequest** - representing the definition of the geofence (e.g. it's location, transition type monitored, etc.);
- An instance of **PendingIntent** - referring to the **Intent** that will be called when the right transition to/from a geofence happens;

We want our app to send notifications when a user is entering/exiting a geofence defined by the location of the icons on the map. Thus, setting the **GeofenceRequest** should happen when a user moves an icon on the map. Further, the notification should be fired when the right event happens. We don't know whether the user will have our application open at the time when this happens, thus, we should be prepared to send the notifications from the background. We can do that via **JobIntentService**. This UI-less class enables us to perform short actions from the background. However, we will not call this class directly, but have a **BroadcastReceiver** that will be triggered by GooglePlayServices when the geofence-related transition happens. Then, this receiver is going to forward the request to **JobIntentService**, which will then issue a notification to the user. In a nutshell:

MapsActivity:
- Instantiate **GeofencingClient**;
- Set **GeofencingRequest** to define the geofences
- Set **PendingIntent** to call **BroadcastReceiver** when a geofencing-related transition happens;

BroadcastReceiver:
- Call **JobIntentService** when GooglePlayServices detect geofencing-related events;

JobIntentService:
- Fire a notification with the appropriate text, depending on which geofence was triggered;

Make sure you understand the above before you proceed with programming.

Programming the App

Open **MapsActivity** and instantiate `mGeofencingClient` in `onCreate`:

```
mGeofencingClient = LocationServices.getGeofencingClient(this)
```

Create geofencing request

Implement a private function `getGeofencingRequest` that takes a marker type (home, work, fitness) and the coordinates (latitude and longitude) of a marker, and returns a request for a specific geofence. The function should check which marker triggered the method and if the type equals to:

- `R.string.map_marker_home`: set a geofence with a circular region of 200m, with a transition type `Geofence.GEOFENCE_TRANSITION_ENTER`
- `R.string.map_marker_work`: set a geofence with a circular region of 300m, with a transition type `Geofence.GEOFENCE_TRANSITION_EXIT`
- `R.string.map_marker_fitness`: set a geofence with a circular region of 300m, with a transition type `Geofence.GEOFENCE_TRANSITION_DWELL`

To build the **Geofences** you should use something like (example for **home**):

```
geofence = with(Geofence.Builder()){
                setRequestId(type)
                setCircularRegion(lat, lon, 200F)
                setExpirationDuration(Geofence.NEVER_EXPIRE)
                setTransitionTypes(Geofence.GEOFENCE_TRANSITION_ENTER)
                build()
            }
```

Exceptionally, for the dwell transition you should also set the delay before the action is triggered. This ensures that the notification is not triggered if a user simply passes by the gym. Use: `setLoiteringDelay(1000);`

To build and return the **GeofenceRequest**, you just need to put the **Geofence** in a list and call the request builder:

```
return with(GeofencingRequest.Builder()){
            setInitialTrigger(GeofencingRequest.INITIAL_TRIGGER_ENTER)
            addGeofences(listOf(geofence))
            build()
```

Defining PendingIntent to be activated when geofence-transition happens

We will now create **mGeofencePendingIntent** property that refers to a **PendingIntent** set it to call a broadcast to **GeofenceBroadcastReceiver**. This property is initialized lazily, i.e. only once we first time need it:

```
private val mGeofencePendingIntent: PendingIntent by lazy {
    val intent = Intent(this,
    GeofenceBroadcastReceiver::class.java)
    PendingIntent.getBroadcast(this, 0, intent,
    PendingIntent.FLAG_UPDATE_CURRENT)
}
```

Adding the geofence through the client

With the **Intent** prepared and geofences specified, we can tell our **GeofencingClient** to start observing the location and firing our **Intent** in case the geofence conditions are satisfied. In **addGeofence** function add:

```
mGeofencingClient.addGeofences(request, mGeofencePendingIntent)
                .addOnCompleteListener(this)
```

Finally, uncomment the code in `setOnMarkerDragListener` to enable geofence setting when a marker is moved.

Sending notifications

Open `GeofenceJobIntentService`. This is a service that will be called when a geofencing-related event is detected. Check its `onHandleWork` method. Here we handle Intents that have triggered the service. We should check which geofence was triggered and what was the event. We should then send a

different notification depending on what has happened. The whole class is already written, but make sure you understand what each of the lines is for.

What is missing, though, is a means to fire the `GeofenceJobIntentService` one the geofence transition is detected. Remember that `GeofenceBroadcastReceiver` is called using the `Intent` we supplied in `MapsActivity`. Open `GeofenceBroadcastReceiver` and in `onReceive` add the following line to make sure that `GeofenceJobIntentService` is called:

```
GeofenceJobIntentService.enqueueWork(context, intent)
```

Testing

Your app is now completed. Testing it will require you to move the markers around the map of Ljubljana and run the trace. If you set the work icon somewhere close to the Faculty of Computer and Information Science on Večna pot, the home icon close to Kolodvor, and the fitness icon on Bleiweisova cesta you should see all three notifications, for leaving work, dwelling at the gym, and arriving home, triggered by the app.

If you did not attend the lab slot in-person at FRI, you should commit your solution to a private Bitbucket repository named **FRIMS2021-LAB-6** and a user **pbdfrita (pbdfrita@gmail.com)** should be added as a read-only member. The solutions will be pulled from your repository on **Sunday, April 17, 23:59**.

Happy coding!