1 Location API and Google MAPS

1.1 Get the current position

The current position can be obtained through methods: GPS/A-GPS, Cell triangulation, Wi-Fi positions, and IP locations, which offers different accuracy.

1.2 Android Location API

Most Android devices today have a device which allows determining where the device is. This is based on a GPS (Global Positioning System) device. Android provides location API in the package "*android.location*" which allows determining the current position. You can register a listener to the location manager and receive periodic updates about the current location. In addition, it is possible to receive the information if the device enters an area given by a longitude, latitude and radius (proximity alert).

Important Classes:

1. The class "LocationManager" provides access to the location service. This allows to access location provider to register for a location update or setting a proximity alert.
2. The class "LocationListener" can be registered with the "LocationManager" and will receive periodic updates about the location.
3. The class "LocationProvider" is the superclass of the different location providers which deliver the information about the current location.
    a. Network: Wi-Fi-based or IP-based
    b. GPS: GPS or A-GPS

1.3 Google Maps

Google provides also a library in the package "com.google.android.maps" for using Google Maps in Android. Google Maps support is not part of the standard Open Source Platform Android and you require an additional key to use them.

1. The class "MapActivity" is an activity which you normally need to extend. "MapActivity" provides the basic functionality to get a "MapView" which can be used to display the map and takes care of the network communication required for the map.
2. The class "MapController" can be used to interact with the "MapView", e.g. by moving it. A "Geopoint" is a position described via latitude and longitude and the class "Overlay" can be used to drawn on the map, for example position markers.

1.4 Android Emulator

In case you want to use Google Maps or location APIs in your emulator make sure you have created a new device which supports the Google API's. This should be the case if you use the emulator created for the previous lab. During emulator creation select the target Google API's in the version of your SDK

1.4.1 Set the location on the emulator

1.4.1.1 DDMS to set the location:

- After running your virtual device (you can do it from the virtual device manager tools -> android -> AVD manager)
- Click on extended controls, the three dots in the emulator
- Select the tab “Location”
- Choose a new location
- Click save location and set location

1.4.1.2 Terminal to set the location

You can use legacy Linux terminal or the one embedded in android studio

- Get the port number of your device (e.g. 5554, usually shown in the top bar of the emulator)
- First you need to find your emulator_ console_auth_token, usually located in your home directory.
- Copy the token.
- telnet localhost port_number
- You need to authenticate, run auth “token you copied”
- Then you can run geo fix <longitude> <latitude> (e.g. geo fix 7.04 14.8)

2 Activating the GPS Module [15’]

Important NOTE: To display the location you may need to run Google Maps on the emulator, this will activate the GPS provider.

3 Creating the API key to be able to use google APIs

Since we are going to deal with Google APIs during this lab, it is better to create the APIs key before starting, you will use the debug key and not the release.

1. Go to https://console.developers.google.com/home/projects (authenticate with your google account if necessary and select Create Project, give to it the name you prefer
2. Wait until your project is created
3. Be sure to select it – Go to your home page and at the top you will see all your projects, just click on the one you’ve just created
4. After that, you will click on **APIs & Services** on the left sidebar and then onto **Credentials**
5. Click on **Create Credentials > API KEY**

6. Automatically an API Key is created but still no specific Google Service is enabled
7. We need to enable Google Maps Services now
8. Click on close and then go to **Library**
9. **Here you will find all google services**
10. You will need to enable 3 services, so look for them and then just click on **Enable**:
    a. **Maps SDK for Android**
    b. **Places API**
    c. **Maps Embed API**
11. Now go back to **Credentials** and just copy your API Key
12. You will need to paste it into your project's manifest

4 Show Location [1h]

4.1 Alternative Building of Project (option) Google Maps Activity (Option 1)

What you can do in order to avoid next steps is basically building a new "Google Maps Activity":

Follow these steps to create a new app project including a map activity:

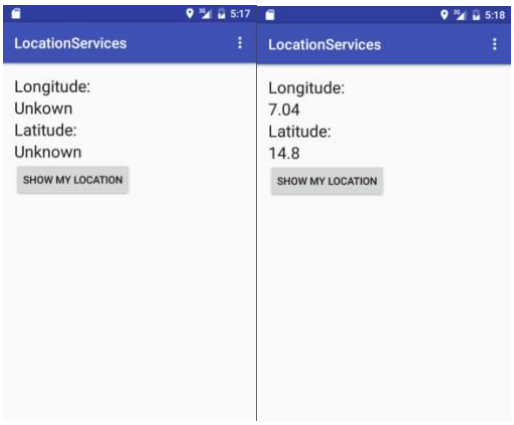
1. Start Android Studio.
2. Create a new project as follows:
    - If you see the **Welcome to Android Studio** dialog, choose **Start a new Android Studio project**, available under 'Quick Start' on the right of the dialog.
    - Otherwise, click **File** in the Android Studio menu bar, then **New**, **New Project**.
3. In the **Choose your project** dialog, select the tab that corresponds to the platform you intended to develop for. Most users will want to keep the default **Phone and Tablet**.
4. Select **Google Maps Activity**, then click **Next**.
5. Enter your app name, package name, and project location, programming language (Java or Kotlin), and the minimum Android API level supported by your app, then click **Finish**.

Android Studio starts Gradle and builds your project. This may take a few seconds. For more information about creating a project in Android Studio, see the Android Studio documentation.

When the build is finished, Android Studio opens the google_maps_api.xml and the MapsActivity.java files in the editor. (Note that your activity may have a different name, but it will be the one you configured during setup.) Notice that the google_maps_api.xml file contains instructions on getting a Google Maps API key before you try to run the application. The next section describes getting the API key in more detail.

After these steps you can simply go on with the creation of your API key and you can both restrict it or not.

4.2 Create Project (option 2)



Application name: LocationServices
Package name : fr.eurecom.locationservices
Activity Type: **Blank** or GoogleMaps Activity (as you prefer)
Activity Name: ShowLocation
Min SDK Version: 23
Use the SDK target to Google APIs 30 SDK. (or automatic)
The activity Showlocation **implements LocationListener**.

Just to be very clear here, there are 2 ways of creating this project, either you create a new project with an empty activity and you personalize it however you think it could be more useful, or you create an already existing google maps activity (point 4.1). Both options have merit but the the GoogleMapsActivity could save some time, as some settings are already set up for you.

4.3 Layout and Permissions

Add the following to your main.xml or activity_maps.xml depending on whether you chose blank or maps activity.

1) Add 4 “TextView” with “android:id = “@id/TextView01”, “@id/TextView02”, ...)
   a) Set the android to “Longitude, unknown, Latitude, unknown”, respectively.
2) Add a button with “android:id = “@id/Button01”, and android:onClick=”showLocation”
3) Add a “ScrollView”
   Note: Make sure to choose “0dp (match constraints)” for layout_width and layout_height
   a) Add a “TextView” inside the ScrollView and set the android:id= “@id/output”
4) Add the following permission to your manifest.xml
   a) INTERNET, ACCESS_FINE_LOCATION , ACCESS_COARSE_LOCATION

4.4 Code the Activity

4.4.1 Define class vars

```
protected LocationManager locationManager = null;
private String provider;
Location location;
TextView latitudeField;
TextView longitudeField;
public static final int MY_PERMISSIONS_LOCATION = 0;
```

4.4.2 Check if GPS is enabled at onStart, if not, ask user to enable it

```
@Override
protected void onStart() {
   super.onStart();
   locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
   boolean gpsEnabled =
           locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);
   if (!gpsEnabled) {
       Log.i("GPS", "not enabled");
       // Build an alert dialog here that requests the user
       // to enable location services when he clicks over "ok"
       enableLocationSettings();
   } else {
       Log.i("GPS", "enabled");
   }

}

private void enableLocationSettings() {
   Intent settingsIntent = new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
   startActivity(settingsIntent);
}
```

4.4.3 Run and check if the alert is working

To check if your alert is properly working simply disable the location service on the emulator you can do this from the settings or simply swiping down from the top of the screen toward the bottom and then disabling the location icon. Once the location is off, the app should ask the user if he is willing to enable it and then correctly start the settings intent.

4.4.4 Start the location manager onCreate

We now start the location manager. When accessing to contacts, camera, and location, it is important to request permissions and ask for the user authorization. Here, we also need to ask the user if it allows to access its own position before using the “getLastKnownLocation. Add the following to the “onCreate” lifecycle method.

```
Criteria criteria = new Criteria();
locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
provider = locationManager.getBestProvider(criteria, false);
if (ContextCompat.checkSelfPermission(this.getApplicationContext(),
       Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
       ContextCompat.checkSelfPermission(this.getApplicationContext(),
               Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
   Log.i("Permission: ", "To be checked");
   ActivityCompat.requestPermissions(this,
           new String[]{Manifest.permission.ACCESS_FINE_LOCATION,
                   Manifest.permission.ACCESS_COARSE_LOCATION},
           MY_PERMISSIONS_LOCATION);
```

```
    //return;
} else
    Log.i("Permission: ", "GRANTED");


latitudeField = (TextView) findViewById(R.id.TextView04);
longitudeField = (TextView) findViewById(R.id.TextView03);
location = locationManager.getLastKnownLocation(provider);
if (locationManager == null) {
    locationManager =
            (LocationManager) getSystemService(Context.LOCATION_SERVICE);
}
provider = locationManager.getBestProvider(criteria, false);
location = locationManager.getLastKnownLocation(provider);
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0,
this);
```

We now require checking the result of the "permission request". For this purpose, we simply need to override `onRequestPermissionsResult`, the variable `MY_PERMISSIONS_LOCATION` is used in order to distinguish the different permissions we might have asked to the user, e.g. location from access to the contact list, access to SMS service, camera etc.

```
@Override
public void onRequestPermissionsResult(int requestCode,
                                       String permissions[], int[] grantResults)
{
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    switch (requestCode) {
        case MY_PERMISSIONS_LOCATION: {
            // If request is cancelled, the result arrays are empty.
            if (grantResults.length > 0
                    && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                Log.i("Access:", "Now permissions are granted");
                // permission was granted, yay!
            } else {
                Log.i("Access:", " permissions are denied");
                //disable the functionality that depends on this permission.
            }
            break;
        }
         //other 'case' lines to check for other permissions this app might
request
    }
}
```

4.4.4 Remove update onPause

```
@Override
protected void onPause() {
    super.onPause();
    locationManager.removeUpdates(this);
}
```

4.4.5 Show the user location

This activity will query the location manager and display the queried values in the activity.

```
public void showLocation(View view) {
   Log.i("showLocation", "Entered");
   switch (view.getId()) {
       case R.id.Button01:
           updateLocationView();
       }
}
```

```
public void updateLocationView() {
   if (location != null){
        double lat = location.getLatitude();
        double lng = location.getLongitude();
        latitudeField.setText(String.valueOf(lat));
        longitudeField.setText(String.valueOf(lng));

   } else{
        Log.i("showLocation","NULL");
   }

}
```

4.4.6 Implement the listener

Now you need to implement the listener so that when the user changes location, its position is updated.

```
@Override
public void onLocationChanged(Location location) {
   Log.i("Location","LOCATION CHANGED!!!");
   updateLocationView();
}

@Override
public void onStatusChanged(String provider, int status, Bundle extras) {}

@Override
public void onProviderEnabled(String provider) {
   Toast.makeText(this, "Enabled new provider " + provider,
           Toast.LENGTH_SHORT).show();
}

@Override
public void onProviderDisabled(String provider) {
   Toast.makeText(this, "Disabled provider " + provider,
           Toast.LENGTH_SHORT).show();
}
```

4.4.7 Run the application

Run the application and through geo fix command or DDMS try to change the user position.

- Is the position changing? If not, what is the problem? Fix it.

4.5 Add Google Map View

This lab uses Google Maps API, you should in this phase use the key created before in the introduction section. Add the following information to your AndroidManifest.xml file just before closing the application tag (i.e., before `</application>`)

```
<meta-data
   android:name="com.google.android.geo.API_KEY"
   android:value="THE_PREVIOUSLY_CREATED_KEY" />
```

This key will be used to access the API correctly while the application is running.

You will see a different piece of code in the manifest where there will be specified where to put the API_KEY. In the end the functionality is the same, so don't worry.

4.5.1 Compiling google Libraries

We need to compile additional libraries, namely Google Services libraries, to use the map object. In the Grandle file add the highlighted line to the dependencies:

```
dependencies {

   implementation 'androidx.appcompat:appcompat:1.5.1'
   implementation 'com.google.android.material:material:1.7.0'
   implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
   implementation 'com.google.android.gms:play-services-maps:18.1.0'
```

After adding the line you need to sync the Grandle file to the project, this will basically allow to use all those library that before were not available.

4.5.2 Adding the map fragment

The **MapFragment** class extends the Fragment class. Remember that a fragment has its own lifecycle, and thus the MapFragement. Add the following element to the layout file.

```
<fragment
   android:name="com.google.android.gms.maps.MapFragment"
   android:id="@+id/map"
   android:layout_width="match_parent"
   android:layout_height="match_parent"/>
```

This fragment would be not recognized without importing the play services in the Gradle file.

Add the fragment to the previously created scrollview.

4.5.3 Implementing the OnMapReadyCallback

Add the following global variables to the ShowLocation activity.

```
private GoogleMap googleMap;
static final LatLng NICE= new LatLng(43.7031,7.02661);
static final LatLng EURECOM = new LatLng(43.614376,7.070450);
```

Now we have all the elements to use a Google maps inside our application, we can start by implement the MapReady Callback in order to be able to handle it. Declare that the ShowLocation activity implements also OnMapReadyCallback and, as suggested add the method implementation:

At the end of the onCreate add the following lines:

```
MapFragment mapFragment = (MapFragment) getFragmentManager()
        .findFragmentById(R.id.map); mapFragment.getMapAsync((OnMapReadyCallback)
this);
```

4.5.4 Fill/add onMapReady

We know fill the onMapReady method in order to center the map on Eurecom

```
@Override
public void onMapReady(GoogleMap Map) {

   googleMap = Map;
   CameraPosition cameraPosition = new CameraPosition.Builder()
           .target(EURECOM)
           .zoom(17)
           .bearing(90)
           .tilt(30)
           .build(); googleMap.addMarker(new MarkerOptions()
           .position(NICE)
           .title("Nice")
           .snippet("Enjoy French Riviera")); googleMap.addMarker(new
MarkerOptions()

           .position(EURECOM)
           .title("EURECOM")
           .snippet("ENJOY STUDY!"));

   googleMap.animateCamera(CameraUpdateFactory.newCameraPosition(cameraPosition)
```

4.5.5 Run the application

4.5.6 Questions

- Explain what a camera is in google maps?
- Update automatically the position both in terms of longitude and latitude labels and in the map
- Now that the position is automatically updated, the button “show my location” is useless: change it and use it to change the type of map from normal to hybrid.
- Detect when a user is clicking on the map and show it through a toast.
- Identify 5 category of applications that require different location information granularity from 10cm, 10m, 100m, 1000m, and 10000m ?
- Describe the Google MAP API pricing model when used in a mobile app.

5 Proximity Alert

Proximity Alert Mobile Application allowing users to set and receive alerts based on location proximity, similar to setting and receiving traditional time-based event reminders [5-7].
Clicking on the map the user has the possibility to add proximity alert in specific positions.

Import in your project the class ProximityIntentReceiver.java that can be found at

https://my.eurecom.fr/jcms/p0_2046751/fr/proximityintentreceiver.

Add in the ShowLocation activity file the following global variables:

```
PendingIntent pendingIntent;
public SharedPreferences sharedPreferences;
private static final String PROX_ALERT_INTENT =
       "fr.eurecom.locationservices.android.lbs.ProximityAlert";
private static final String POINT_LATITUDE_KEY = "POINT_LATITUDE_KEY";
private static final String POINT_LONGITUDE_KEY = "POINT_LONGITUDE_KEY";
private int count = 0;
```

In onCreate() initialize sharedPreferences:

```
sharedPreferences = getSharedPreferences("location", 0);
```

Through the last question of the previous section you should have correctly added the onMapClickListener and implemented the onMapClick method, in this method you need to insert the code which will add the proximity alert.

```
Intent intent = new Intent(PROX_ALERT_INTENT);
PendingIntent proximityIntent =
       PendingIntent.getBroadcast(getApplicationContext(), 0, intent, 0);

if (ContextCompat.checkSelfPermission(getApplicationContext(),
       Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
       ContextCompat.checkSelfPermission(getApplicationContext(),
              Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED)
{
   Log.i("Permission: ", "To be checked");


   return;
} else {
   Log.i("Permission: ", "GRANTED");
}

saveCoordinatesInPreferences((float) point.latitude, (float) point.longitude);

locationManager.addProximityAlert(point.latitude, point.longitude, 20, -1,
proximityIntent);
IntentFilter filter = new IntentFilter(PROX_ALERT_INTENT);
```

```
registerReceiver(new ProximityIntentReceiver(), filter); Log.i("Registred",
"proximity");

Toast.makeText(getBaseContext(), "Added a proximity Alert",
Toast.LENGTH_LONG).show();
++count;
```

You need also to add the method saveCoordinatesInPreferences(…)

```
private void saveCoordinatesInPreferences(float latitude, float longitude) {
   SharedPreferences prefs =
          this.getSharedPreferences(getClass().getSimpleName(),
Context.MODE_PRIVATE);
   SharedPreferences.Editor prefsEditor = prefs.edit();
   prefsEditor.putFloat(POINT_LATITUDE_KEY, latitude);
   prefsEditor.putFloat(POINT_LONGITUDE_KEY, longitude);
   prefsEditor.commit();
}
```

5.1.1 Run the code

Test the proximity by changing your simulator position

5.1.2 Questions

- What is a broadcast receiver? And a Filter?
- Add a marker where the proximity alert is added
- You have the possibility to add several Proximity Alerts, but, if n is added, you will notice that you will receive n notifications, can you solve this problem?

Upload the application to the next cloud with the following name,
"Android_lab3_name1_name2.zip"
https://nextcloud.eurecom.fr/s/rPwXWeexqgCcQxQ

6 Reference

[1] https://developers.google.com/maps/documentation/android/
[2] http://developer.android.com/guide/topics/fundamentals.html
[3] http://www.vogella.de/google.html
[4] http://developer.android.com/resources/index.html
[5] http://code.google.com/p/android-for-gods/w/list
[6] http://blog.brianbuikema.com/2010/07/part-1-developing-proximity-alerts-for-mobile-applications-using-the-android-platform/
[7] http://marakana.com/training/android/ and http://marakana.com/forums/android/examples/
[8] https://github.com/gauntface/Android-Proximity-Alerts-Example
[9] http://code.google.com/p/demo-android-proximity-alert/source/checkout