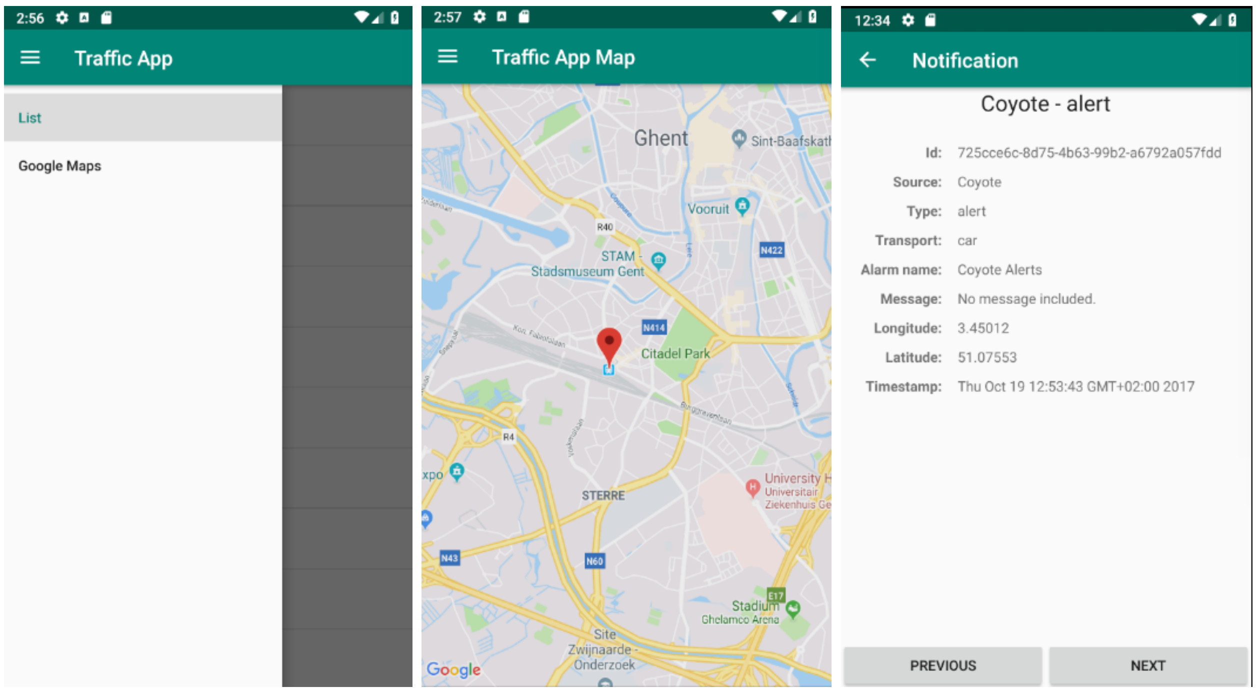
Lab 6: Google Maps and Navigation Drawer

Design and Development of Mobile Applications
E. De Coninck, S. Leroux, P. Simoens
2019-2020

In this lab session we will expand the functionality of the traffic alert app we have built in the previous lab sessions. The updated app will provide two view types on the open data available for the city of Ghent: a list view and a Google Maps view, as shown in Figure 1. We will add a navigation drawer (sometimes called a hamburger menu) that allows the user to switch between both views. The start code on Minerva is an alternative solution to the previous session.

You will learn the following concepts:

- Using API and Training documentation on `https://developer.android.com`
- Building Navigation Drawer with default Action Bar (see `https://developer.android.com/training/implementing-navigation/nav-drawer#java` and specifically for Navigation component `https://developer.android.com/topic/libraries/architecture/navigation/navigation-ui#java`)
- Building layout with Google Map: SupportMapFragment (`https://developers.google.com/maps/documentation/android-sdk/intro#java`)
- Handling Google map marker clicks



Coyote - alert

Id: 725cce6c-8d75-4b63-99b2-a6792a057fdd
Source: Coyote
Type: alert
Transport: car
Alarm name: Coyote Alerts
Message: No message included.
Longitude: 3.45012
Latitude: 51.07553
Timestamp: Thu Oct 19 12:53:43 GMT+02:00 2017

Figure 1: Example layout of the Traffic Notifications Application with a navigation drawer, a Google map view with markers for items.

1 Project set-up

- The start code project can be downloaded from Ufora. This start code is a solution of the last session with some important edits.

1.1 Start code updates

For this lab exercise we changed the solution of the last session to better accomodate for this session. These changes do not alter the function of the application but simplify this next itteration of the application.

- Change the remote fetch url to a static remote file as the datasource got deprecated and returned an empty list. New url: `https://users.ugent.be/~ejodconi/verkeersmeldingen.json`.
- Handle response of Volley request in `TrafficRepository` in an AsyncTask as the response is handled on the UI Thread. A Toast msg returns the number of new items inserted.
- Updated JsonUtils to correctly return same UUID's for same objects. Objects without UUID default to same ID.
- Removed landscape layout file for the `MainActivity` and added a landscape layout file for the `MainFragment`. This allows other view types to decide if they show extra content in landscape or not. This is needed so we can use a `NavHostFragment` in the `MainActivity`.
- Check for detail view when item is selected to decide if a new fragment needs to be opened (see dualPaneMode variable).
- Updated `MainActivity` to create an `AppBarConfiguration` with multiple top destinations. This allows us to have multiple fragments which show the hamburger icon of the Navigation Drawer.

1.2 Setup for Google Maps

- Open the start code and install missing sdks if any.
  - Enable the ability to fetch remote data with the correct permissions. (`https://developer.android.com/training/basics/network-ops/connecting`)
- To build the application and be allowed to use Google Play services (maps) we need an API key and the maps library as a dependency:
  - Add the Google Maps dependency to the module's build.gradle file (only emulators image with Google APIs supports this version of the application as we need the Play Store to be installed).

    ```
    implementation 'com.google.android.gms:play-services-maps:17.0.0'
    ```
  - In order to use a Google map in our application need Google Play Services installed on the device. Google Play delivers service updates for users on Android 4.0 and higher through the Google Play Store app. However, updates might not reach all users immediately, so your app should verify the version available before attempting to perform API transactions. Specify the Google Play services version number in the **AndroidManifest** file (see `https://developers.google.com/maps/documentation/android-sdk/config`).
  - Follow this guide (`https://developers.google.com/maps/documentation/android-sdk/signup`) to create a new API key that you can add to your app manifest. For testing purposes we will not restrict this key to a specific Android application.

2 Traffic alerts shown on Google Maps

- Create a new destination Fragment in the navigation **nav_graph.xml** file and link it with an action to the detail page. Update the code in the map Fragment so the ViewModel is initialised correctly and add a static SupportMapFragment to this Fragment's layout file. Add this Fragment as a top-level destination in MainActivity to setup up navigation. If you manually change the start destination of the navigation graph you should be able to run the application, see the world map and interact with it.
- The SupportMapFragment is not the map view itself. The necessary View object can only be acquired through an asynchronous callback method. Implement the OnMapReadyCallback interface in the map Fragment's class and override the necessary method. Get a reference of the fragment through the FragmentManager of this fragment (use getChildFragmentManager()). Set the MapFragment as the listener for the SupportMapFragment with getMapAsync(...). Save the GoogleMap object in a private field variable (mMap) and call updateMap(...) method provided below, each time the ViewModel's notification list is updated. This method adds a marker for each traffic alert and recalibrates the map's camera so the markers are within view.

```
private void updateMap(List<TrafficNotification> trafficNotifications) {
    if (mMap == null || trafficNotifications == null)
        return;

    mMap.clear();
    if (!trafficNotifications.isEmpty()) {
        LatLngBounds.Builder builder = new LatLngBounds.Builder();
        for (TrafficNotification item : trafficNotifications) {
            if (item.hasValidPosition()) {
                Marker marker = mMap.addMarker(new MarkerOptions()
                                .position(new LatLng(item.getLatitude(), item.getLongitude()))
                                .title(item.toString()));
                marker.setTag(item);
                builder.include(marker.getPosition());
            }
        }
        LatLngBounds bounds = builder.build();
        final CameraUpdate cameraUpdate = CameraUpdateFactory.newLatLngBounds(bounds, 100);
        // try catch to fix a bug: updating camera position if the map is not yet fully loaded.
        try {
            mMap.animateCamera(cameraUpdate);
        } catch (IllegalStateException ise) {
            mMap.setOnMapLoadedCallback(new GoogleMap.OnMapLoadedCallback() {
                @Override
                public void onMapLoaded() { mMap.animateCamera(cameraUpdate); }
            });
        }
    }
}
```

- Add a marker listener when you receive the GoogleMap and open the details of the selected marker using the navigation component action. The markers's tag (marker.getTag()) is set to the traffic notification in the updateMap method.

3 Support multiple view types with Navigation Drawer

A Navigation Drawer is mostly used as a panel swiped in from the left to select certain view types or switch between top level navigation elements. Using the Navigation Architecture Component enables automatically updating the fragment based on the selected menu item by matching the menu item ids to the destination ids.

- For the drawer we are going to use a NavigationView. Replace the root view group with a 'androidx.drawerlayout.widget.DrawerLayout' (note the androidx part) in the MainActivity's layout file. The first child of this view group is the content and the second the drawer which can be pulled in. Add a `com.google.android.material.navigation.NavigationView` (not the version from support libraries) as the second child which will be used as the slide-in view. Populate this list by using an menu resource file directly from the xml layout file (attribute `app:menu`). Add a menu item for each view type and set the menu item id to the `Fragment` destination's id. Setup the `NavigationView` to automatically open the selected `Fragment` by setting up the `NavController` with `NavigationUI.setupWithNavController(...)` (see `https://developer.android.com/topic/libraries/architecture/navigation/navigation-ui#Tie-navdrawer` and bottom navigation). At this time the navigation drawer can be tested by swiping in from the left side.
- Left navigation drawers typically have a toggle button in the upper left corner (action bar) to open the drawer. To enable this functionality we still need to update the `AppBarConfiguration` with a reference to the `DrawerLayout` (see `https://developer.android.com/topic/libraries/architecture/navigation/navigation-ui#appbarconfiguration`).