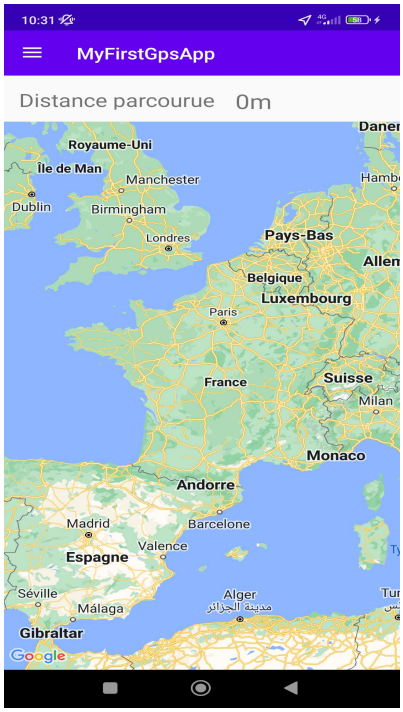
Application de gestion des données GPS

I. Présentation générale

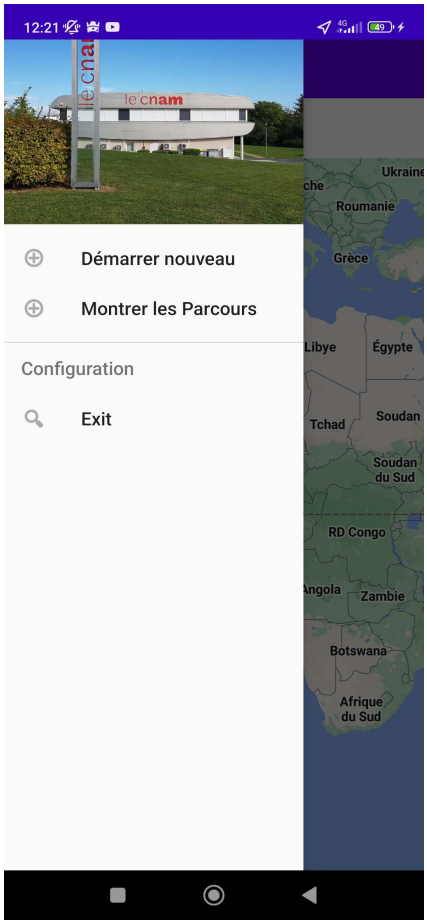
Ce TP va nous permettre de comprendre la gestion des données GPS et l'affichage d'une carte sur la plateforme android. Nous métrons en place une base de données afin de sauvegarder un parcours et revoir le parcours ultérieurement

L'application s'articulera sur 2 activités :

La première activité nous permet de visualiser la carte mondiale en utilisant l'API Google. Il est donc conseillé d'utiliser un device android plutôt qu'un émulateur, ainsi qu'une connexion 4G. Nous pourrons également voir un parcours sauvegardé sur cette carte.



Cette activité dispose d'un menu principal que nous allons mettre en place :



- Etude de la gestion de menu :

Afin d'étudier sa mise en place, nous allons créer un projet temporaire :
File → New Project → Navigation Drawer Activity

Vous pouvez observer que le layout principale s'articule selon le pattern suivant :

```
<androidx.drawerlayout.widget.DrawerLayout>

        <androidx.appcompat.widget.Toolbar />

        // Contenu de votre activité

        <com.google.android.material.navigation.NavigationView
            app:headerLayout="@layout/activity_main_nav_header"
            app:menu="@menu/activity_main_menu_idle_drawer" />

</androidx.drawerlayout.widget.DrawerLayout>
```

Pour lier les différents éléments graphiques entre eux, vous devrez opérer comme suit dans l'activité :

1. lié la ToolBar à l'activité :

```
 private void configureToolBar(){
    this.toolbar = findViewById(R.id.activity_main_toolbar);
    setSupportActionBar(toolbar);
 }
```

2. lié le menu à la ToolBar

```
 private void configureDrawerLayout(){
    this.drawerLayout = findViewById(R.id.activity_main_drawer_layout);
    ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(this,
drawerLayout, toolbar, R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
    drawerLayout.addDrawerListener(toggle);
    toggle.syncState();
 }
```

3. lié les actions sur le menu à l'activité

```
 private void configureNavigationView(){
    this.navigationView = findViewById(R.id.activity_main_nav_view);
    navigationView.setNavigationItemSelectedListener(this);
 }
```

Ajouter dans le layout le NavigationDrawer comme suit :

```
<com.google.android.material.navigation.NavigationView
   android:id="@+id/activity_main_nav_view"
   android:layout_width="wrap_content"
   android:layout_height="match_parent"
   android:layout_gravity="start"
   android:fitsSystemWindows="true"
   app:headerLayout="@layout/activity_main_nav_header"
   app:menu="@menu/activity_main_menu_idle_drawer" />
```

Pour mettre en place le menu, il vous faudra créer les 2 fichier XML suivants :

- activity_main_nav_header.xml : (contenu supérieur du menu)

```
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:background="#B2B9E1"
   android:gravity="center"
   android:orientation="vertical">

   <ImageView
      android:id="@+id/imageView"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:adjustViewBounds="true"
      app:srcCompat="@drawable/cnamgrandest" />

</LinearLayout>
```

- activity_main_menu_idle_drawer.xml :

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   tools:showIn="navigation_view">

   <group>
      <item
         android:id="@+id/activity_main_start_new_path"
         android:icon="@android:drawable/ic_menu_add"
         android:title="Démarrer nouveau Parcours" />
      <item
         android:id="@+id/activity_main_show_all_path"
         android:icon="@android:drawable/ic_menu_add"
         android:title="Montrer les Parcours" />
   </group>

   <item android:title="Configuration">
      <menu>
         <item
            android:id="@+id/activity_main_exit"
            android:icon="@android:drawable/ic_menu_search"
            android:title="Exit" />
      </menu>
   </item>

</menu>
```

Enfin le click sur une entrée du menu se fera dans la methode surchargée onNavigationItemSelected :

```
    @Override
    public boolean onNavigationItemSelected(MenuItem item) {
        int id = item.getItemId();

        switch (id){
            case R.id.activity_main_start_new_path :
                ...
                break;
…...
            case R.id.activity_main_exit:
                finish();
                break;
            default:
                break;
        }

        this.drawerLayout.closeDrawer(GravityCompat.START);

        return true;
    }
```

Gestion de la Map Google

Tout d »abord vous allez devoir récupérer une API Key Google, en suivant le lien :
https://developers.google.com/maps/documentation/javascript/get-api-key

Une fois cette clé récupéré depuis votre compte Google ajouter la ligne suivante dans votre fichier AndroidManifest :

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

Dans le layout de votre activité principale, ajouter un Fragment qui servira de conteneur pour la Map Google

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

Dans l'activité, ajouter la ligne suivante afin de charger la Map Google :

```
SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
      .findFragmentById(R.id.map);
mapFragment.getMapAsync(this);
```

Ceci nous permettra d'etre notifié lorsque la Map sera chargée par la méthode :

```
@Override
public void onMapReady(GoogleMap googleMap) {
   Log.v(LOG_TAG, ">onMapReady");
   m_googleMap = googleMap;

   // Add a marker in Sydney and move the camera
//    LatLng sydney = new LatLng(-34, 151);
//    m_googleMap.addMarker(new MarkerOptions()
//          .position(sydney)
//          .title("Marker in Sydney"));
//    m_googleMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
}
```

NB : vous pouvez activer la partie commentée pour placer un marker sur Sydney et voir le comportement.

Récupération de la localisation en temps réel pour affichage sur la carte :

Tout d'abord mettre la permission android.permission.ACCESS_FINE_LOCATION dans le fichier manifest.

Récupérer le locationManager dans la méthode onCreate de l'activité, puis une fois la permission ACCESS_FINE_LOCATION accordée s'enregistrer sur les notifications de localisation.

```
m_locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
2000, 5, this);
```

Pour se faire, votre activité devra implémenter l'interface LocationListener et donc implémenter la méthode onLocationChanged comme suit :

```
@Override
public void onLocationChanged(Location location) {
    if (!m_newPathStarted) return;

    // Add a marker in current position and move the camera
    LatLng myPosition = new LatLng(location.getLatitude(), location.getLongitude());

    if (m_currentPath != null) {
        GpsPoint currentPoint = new GpsPoint(location.getLatitude(), location.getLongitude());
        m_currentPath.addGpsPoint(currentPoint);
        m_gpsPathDbHelper.addPathGpsPoint(m_currentPath.getGpsPointTableName(), currentPoint);
    }

    MarkerOptions markerOptions = new MarkerOptions();
    markerOptions.position(myPosition);

    if (m_myLastLocation == null) {
        markerOptions.title("Starting Point");
    } else {
         m_distanceParcourue += m_myLastLocation.distanceTo(location);
        m_distanceParcourueTv.setText(String.valueOf(m_distanceParcourue));
    }
    m_myLastLocation = location;

    m_googleMap.addMarker(markerOptions);

    m_googleMap.moveCamera(CameraUpdateFactory.newLatLng(myPosition));
}
```

onStatusChanged doit également être surchargée pour respecter l'interface LocationListener mais elle ne servira pas .

Enregistrement de parcours dans une base de données

Dans la méthode onNavigationItemSelected, gérer le clique sur le start new Path en appelant la méthode askPathNameBeforeCreating. Cette mothde

La méthode askPathNameBeforeCreating permet de créer une AlertDialog afin de demander le nom du path de la facon suivante :

```
    private void askPathNameBeforeCreating() {
        // TODO 16) create an AlertDialog
        AlertDialog.Builder alertDialog = new AlertDialog.Builder(MainActivity.this);
        alertDialog.setTitle("Path name");
        alertDialog.setMessage("Enter Path Name");

        final EditText input = new EditText(MainActivity.this);
        LinearLayout.LayoutParams lp = new LinearLayout.LayoutParams(
                LinearLayout.LayoutParams.MATCH_PARENT,
                LinearLayout.LayoutParams.MATCH_PARENT);
        input.setLayoutParams(lp);
        alertDialog.setView(input);
        alertDialog.setIcon(android.R.drawable.ic_menu_save);

        alertDialog.setPositiveButton("YES",
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                        String name = input.getText().toString();
                        Log.i(LOG_TAG, "alertDialog >answer YES" + name);

                        long id = m_gpsPathDbHelper.addPathEntry(name);
                        m_currentPath = new GpsPath(id, name);
                        m_gpsPathDbHelper.createTablePoint(m_currentPath.getGpsPointTableName());
                        m_newPathStarted = true;

                        // Update Menu
                        MainActivity.this.navigationView.getMenu().clear();

MainActivity.this.navigationView.inflateMenu(R.menu.activity_main_menu_recording_drawer);
                    }
                });

        alertDialog.setNegativeButton("NO",
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                        dialog.cancel();
                    }
                });

        alertDialog.show();
```

NB : Une fois que vous lancez un Path GPS,nous créons ces données dans une base de données grace au DbHelper MyGpsPathDbHelper. Un boolean m_newPathStarted est positionné à true afin de se mettre en mode affichage des points courants sur la carte Google.

Il vous faudra compléter les quelques méthode du DBHelper afin de poursuivre le developpement.

Une fois l'étape de l'alertDialog passé vous arrivez normalement dans la seconde activité qui vous listera les PATH GPS sauvegardés dans la database MySql.

Pour se faire instancier MyGpsPathDbHelper dans la methode onCreate de l'activité GpsPathListActivity, et récupérez tous les Path de la dB en utilisant la méthode getAllPath().
Le résultat est sauvegardé dans m_gpsPathList.

Puis il faudra donner cette liste à MyCustomRecyclerAdapter comme suit :

```
m_listAdapter = new MyCustomRecyclerAdapter(this, m_gpsPathList);
this.recyclerView.setAdapter(m_listAdapter);
this.recyclerView.setLayoutManager(new LinearLayoutManager(this));
```

Une classe SwipeHelper est fournie pour gérer le swipe sur chaque élément de liste, celui-ci propose 2 boutons cachés ; Delete et Show.
Observez le fonctionnement de ce Helper et fournissez le comportement adéquat à chaque action.

Pour l'action Show il faudra fermer l'activité courante et renvoyer à l'activité principale une information lui permettant de retrouver dans la dB le Path en question.

Une fois que vous aurez fini la gestion des 2 boutons, il faudra gérer le retour dans MainActivity dans la méthode onActivityResult.

Pour visualise un path, la méthode addFakeCnamPath() est disponible dans MainActivity. Vous pouvez ajouter une entrée de menu afin de rajouter à la main ce path et verifier ainsi l'ensemble du comportement.