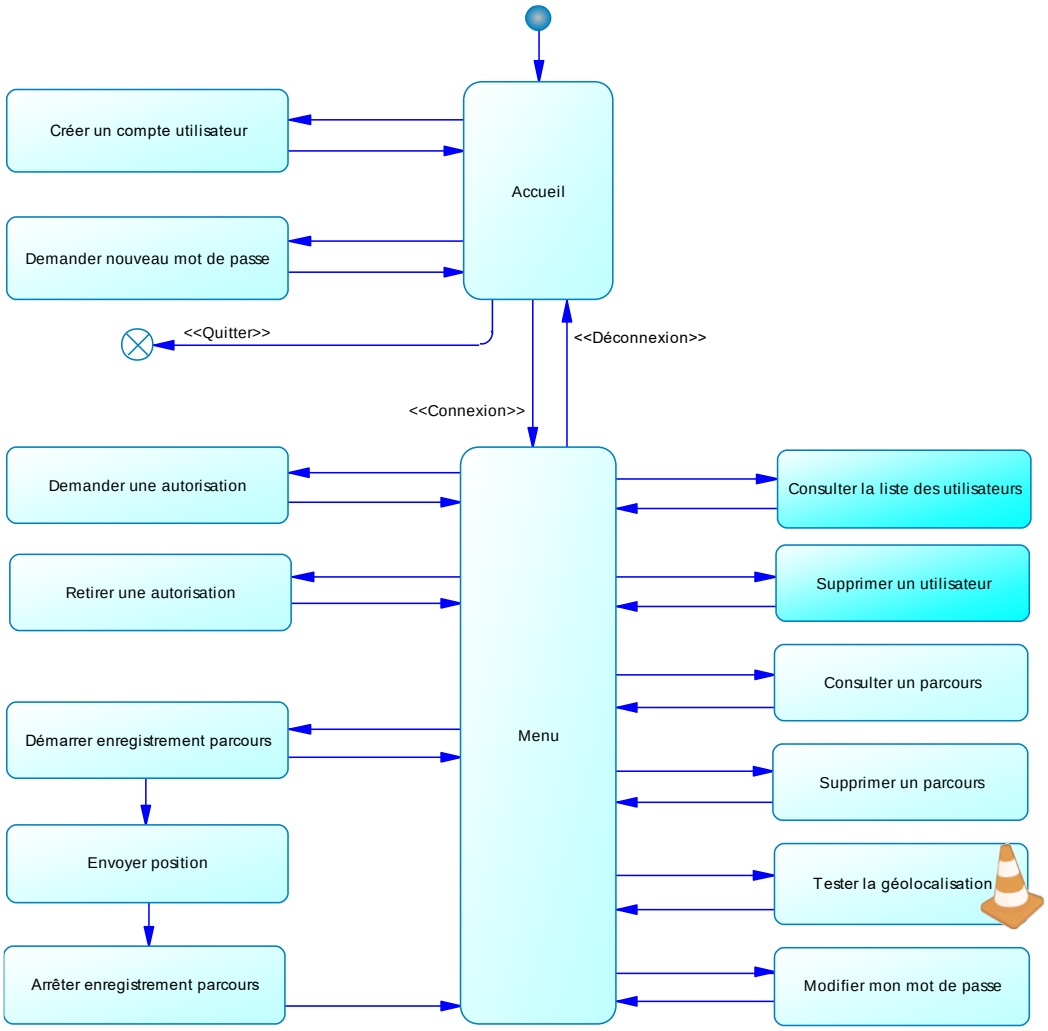
AP

5- Développement de l'application mobile Android

5-9 TesterGeolocalisation (page de test de géolocalisation)

1- Situation de l'activité dans la structure de l'application
2- Obtenir une empreinte SHA1 sur le poste de développement
3- Mise à jour du SDK avec les services Google Play
4- Modification du fichier strings.xml
5- Création d'une activité Google Maps
6- Obtention d'une clé API Google Maps
7- Modification de la programmation Java de MenuGeneral.java
8- Création de l'interface graphique
9- Programmation Java de l'activité TesterGeolocalisation.java

1- Situation de l'activité dans la structure de l'application



2- Obtenir une empreinte SHA1 sur le poste de développement

> ⚠ Cette étape peut être contournée si vous avez déjà récupéré l'empreinte **SHA1** de votre poste.
> Attention si vous changez de poste de développement, l'empreinte **SHA1** change.

La récupération de l'empreinte **SHA1** se fait en mode commande.
Celà nécessite de repérer l'emplacement de 2 fichiers :

- le fichier **keytool.exe** (un programme à exécuter en mode commande)
  Ce fichier se trouve dans le dossier **bin** du jdk.

  Ce chemin ressemble aux exemples suivants :
  C:\Program Files (x86)\Java\jdk1.7.0_01\bin
  C:\Program Files\Java\jdk1.8.0_112\bin

- le **fichier des clés** (keystore) créé lors de la génération des fichiers apk (voir dossier portant sur la création et la signature des fichiers apk) :

  Dans mon cas personnel, ce chemin est (sur 2 postes différents) :
  D:\ApplicationsAndroid\ jim.android.cles.jks
  C:\JM\Java\applisAndroid\AndroidStudio\jim.android.cles.jks

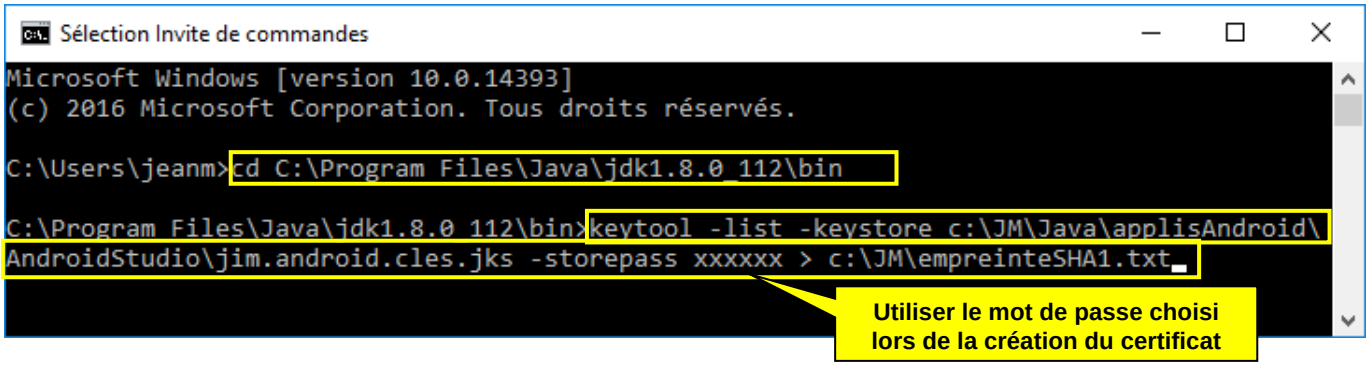
Passer en mode commande et taper les 2 commandes suivantes :

CD *[emplacement de l'outil keytool.exe]*

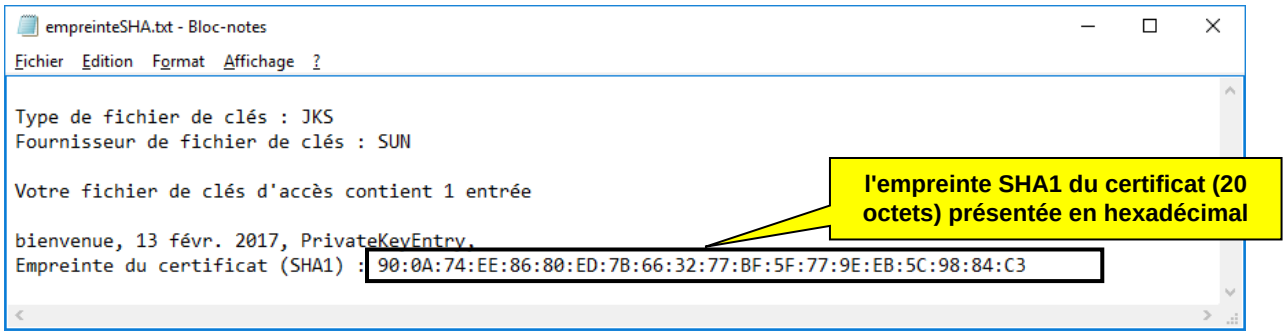
keytool -list -keystore *[emplacement du certificat]* **-storepass** *[mot de passe]* > d:\empreinteSHA1.txt

- la première commande (**CD** pour **Change Directory**) active le dossier contenant **keytool.exe**
- la deuxième commande utilise le programme **keytool.exe** pour rechercher la signature SHA1 et redirige l'affichage vers un fichier texte (**d:\empreinteSHA1.txt**) que l'on peut bien sûr nommer et sauvegarder ailleurs.

Exemple :



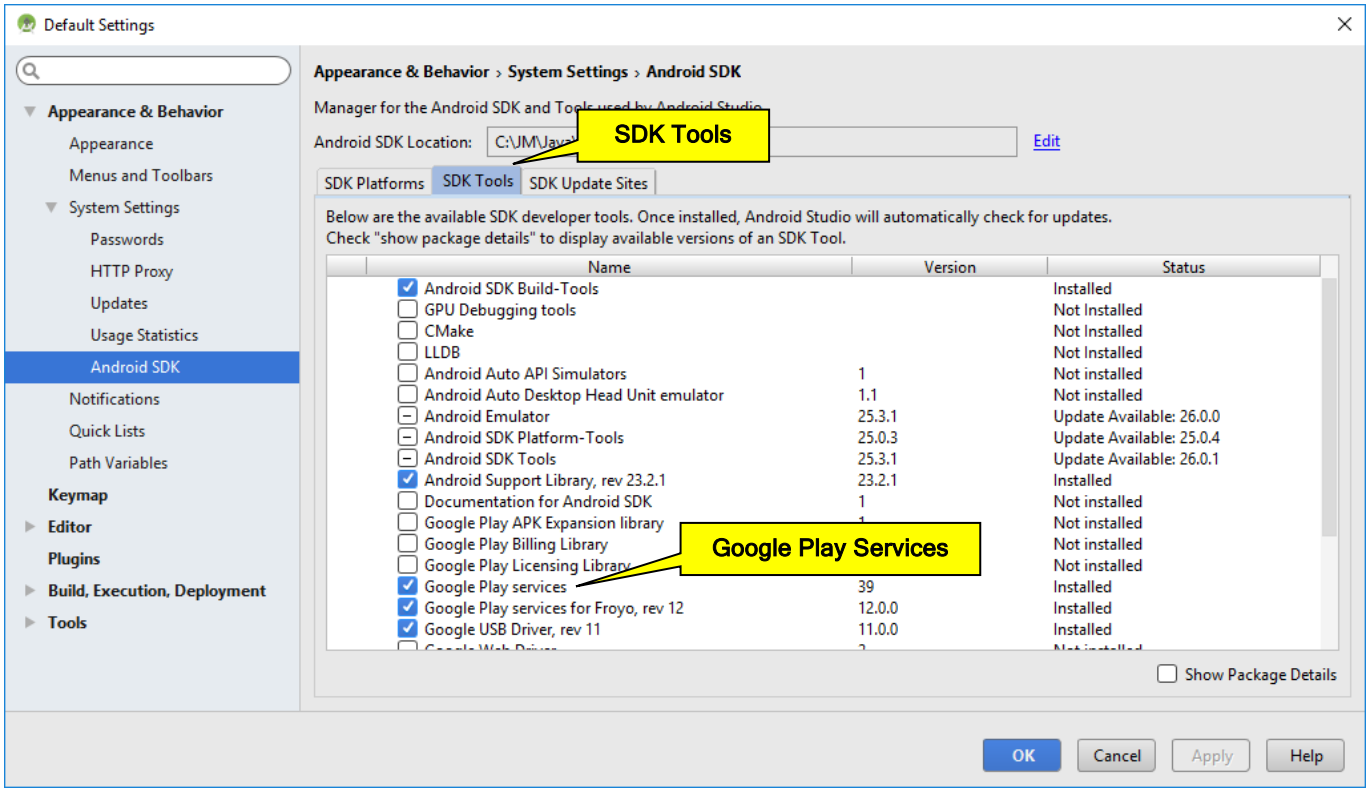
Pour récupérer l'empreinte SHA1, il suffit d'ouvrir le fichier texte créé par la dernière commande :



3- Mise à jour du SDK avec les services Google Play

Démarrez **Android Studio** et lancez le **SDK Manager**.

Assurez-vous que les **services Google Play** sont bien installés (si ce n'est pas le cas, installez-les car l'affichage de cartes a besoin de ces services) :



4- Modification du fichier strings.xml

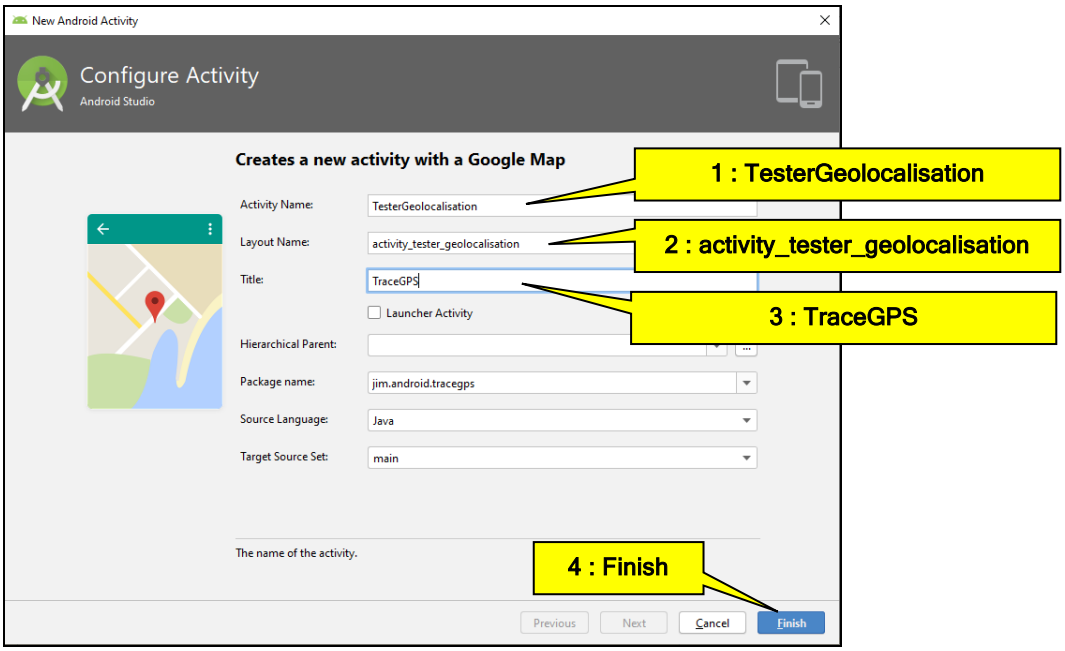
L'interface graphique à créer :

Dans le dossier **res/values**, complétez le fichier **strings.xml** avec le code suivant :

```
<!-- Les textes de la page de test de la géolocalisation -->
<string name="tester_geolocalisation_titre1">Tester la géolocalisation</string>
<string name="tester_geolocalisation_bouton_retourner">Retourner au menu</string>
```

5- Création d'une activité Google Maps

Créer une nouvelle activité en faisant un clic droit sur la racine **app** du projet et en choisissant la commande **New / Google / Google Maps Activity** :



Une fois l'activité créee, Android Studio ouvre les fichiers **TesterGeolocalisation.java** et **google_maps_api.xml**.

Le fichier **res/values/google_maps_api.xml** contient des instructions sur le moyen d'obtenir une clé **d'API Google Maps** afin de pouvoir exécuter l'application (**en mode Debug**) :

```
<resources>
    <!--
    TODO: Before you run your application, you need a Google Maps API key.

    To get one, follow this link, follow the directions and press "Create" at the end:
    https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID
    &r=7C:F1:B8:3D:B0:45:ED:84:55:4A:F0:3C:AC:AD:24:5A:E5:C2:3D:8C%3Bjim.android.tracegps

    You can also add your credentials to an existing key, using these values:

    Package name:
    7C:F1:B8:3D:B0:45:ED:84:55:4A:F0:3C:AC:AD:24:5A:E5:C2:3D:8C

    SHA-1 certificate fingerprint:
    7C:F1:B8:3D:B0:45:ED:84:55:4A:F0:3C:AC:AD:24:5A:E5:C2:3D:8C

    Alternatively, follow the directions here:
    https://developers.google.com/maps/documentation/android/start#get-key

    Once you have your key (it starts with "AIza"), replace the "google_maps_key"
    string in this file.
    -->
    <string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">
        YOUR_KEY_HERE
    </string>
</resources>
```

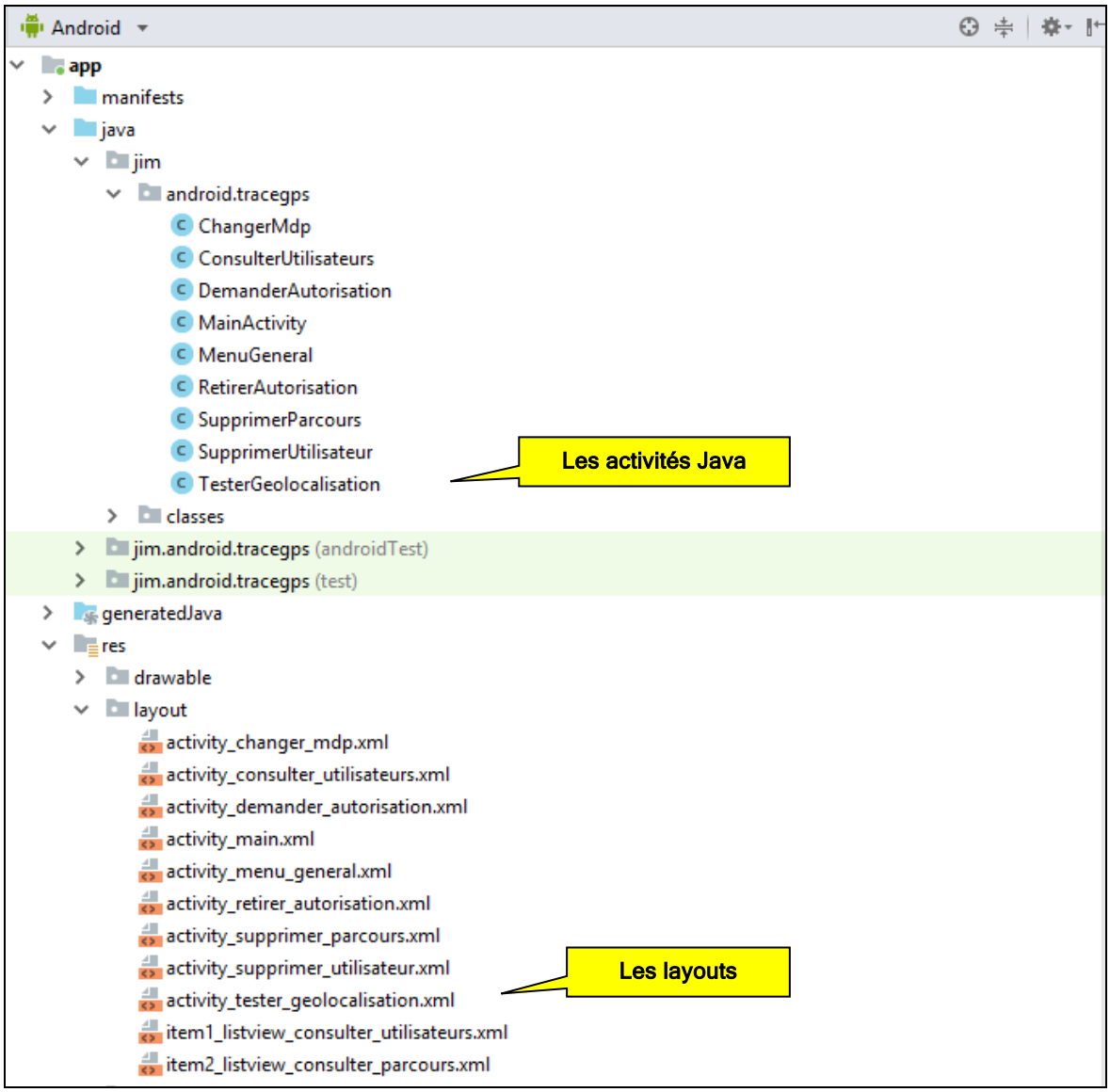
Etape 1 : ce lien permet de créer une clé API Google

Etape 2 : la clé doit être placée ici

Remarquez que le lien contient 2 informations :

- la **clé SHA1** (20 octets en hexadécimal) utilisée pour identifier le poste **en mode Debug**
- le **nom du package** (**jim.android.tracegps** dans cet exemple)

L'activité **TesterGeolocalisation.java** et le layout **activity_tester_geolocalisation.xml** ont été ajoutés :



On peut constater que le fichier **AndroidManifest.xml** a été automatiquement complété :

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="jim.android.tracegps">

    <!--
         The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
         Google Maps Android API v2, but you must specify either coarse or fine
         location permissions for the 'MyLocation' functionality.
    -->
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        android:usesCleartextTraffic="true">

        <!--
             The API key for Google Maps-based APIs is defined as a string resource.
             (See the file "res/values/google_maps_api.xml").
             Note that the API key is linked to the encryption key used to sign the APK.
             You need a different API key for each encryption key, including the release key that is used to
             sign the APK for publishing.
             You can define the keys for the debug and release targets in src/debug/ and src/release/.
        -->
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="@string/google_maps_key" />

        <activity
            android:name=".TesterGeolocalisation"
            android:label="@string/title_activity_tester_geolocalisation"></activity>
        <activity android:name=".SupprimerParcours" />
        <activity android:name=".SupprimerUtilisateur" />
        <activity android:name=".RetirerAutorisation" />
        <activity android:name=".DemanderAutorisation" />
        <activity android:name=".ConsulterUtilisateurs" />
        <activity android:name=".ChangerMdp" />
        <activity android:name=".MenuGeneral" />
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

L'autorisation de géolocalisation a été ajoutée

La clé API Google Maps

L'activité ajoutée

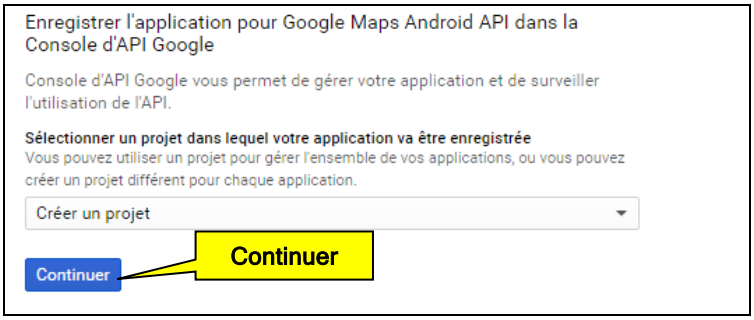
6- Obtention d'une clé API Google Maps

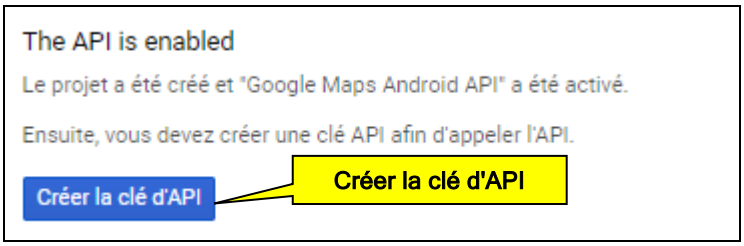
Une application Android doit disposer d'une **clé d'API** pour pouvoir accéder aux serveurs **Google Maps**. Le type de clé dont elle a besoin est une clé d'API avec une restriction pour les **applications Android**. Cette clé est gratuite. On peut l'utiliser avec n'importe quelle application qui appelle **Google Maps Android API** ; elle prend en charge un nombre illimité d'utilisateurs.

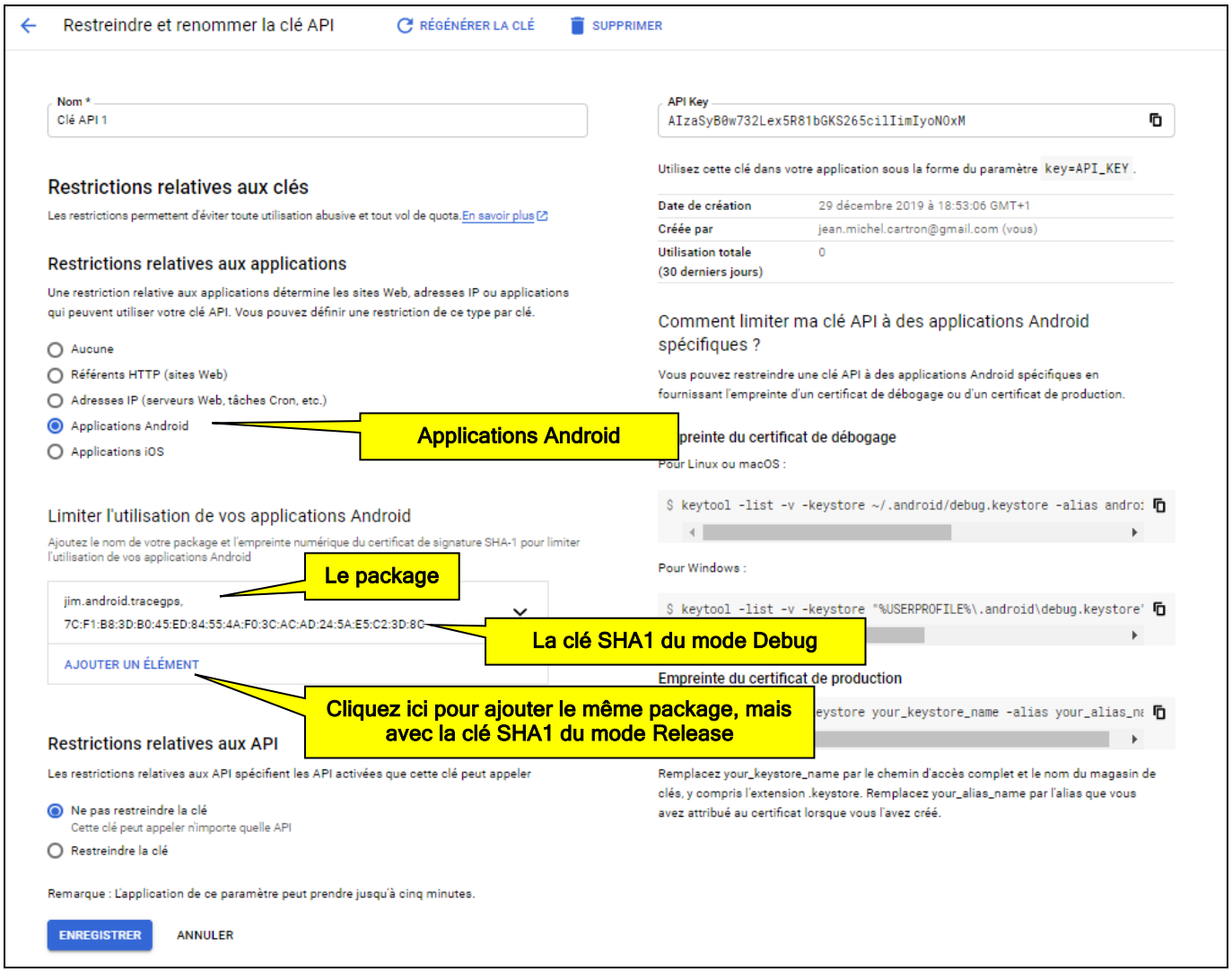
La méthode la plus simple et la plus rapide pour obtenir une clé d'API consiste à utiliser le lien fourni dans le fichier **google_maps_api.xml** :

1. Copier le lien fourni et le coller dans un navigateur. Ce lien dirige vers la **Google API Console** et fournit les informations requises via des paramètres d'URL, réduisant ainsi la saisie manuelle.
2. Suivre les instructions pour créer un **nouveau projet** sur la **Google API Console** ou sélectionner un **projet existant**.
3. Créer une clé d'API restreinte à Android pour le projet.
4. Copier la clé d'API obtenue, retourner dans Android Studio et coller la clé d'API dans l'élément **<string>** du fichier **google_maps_api.xml**.

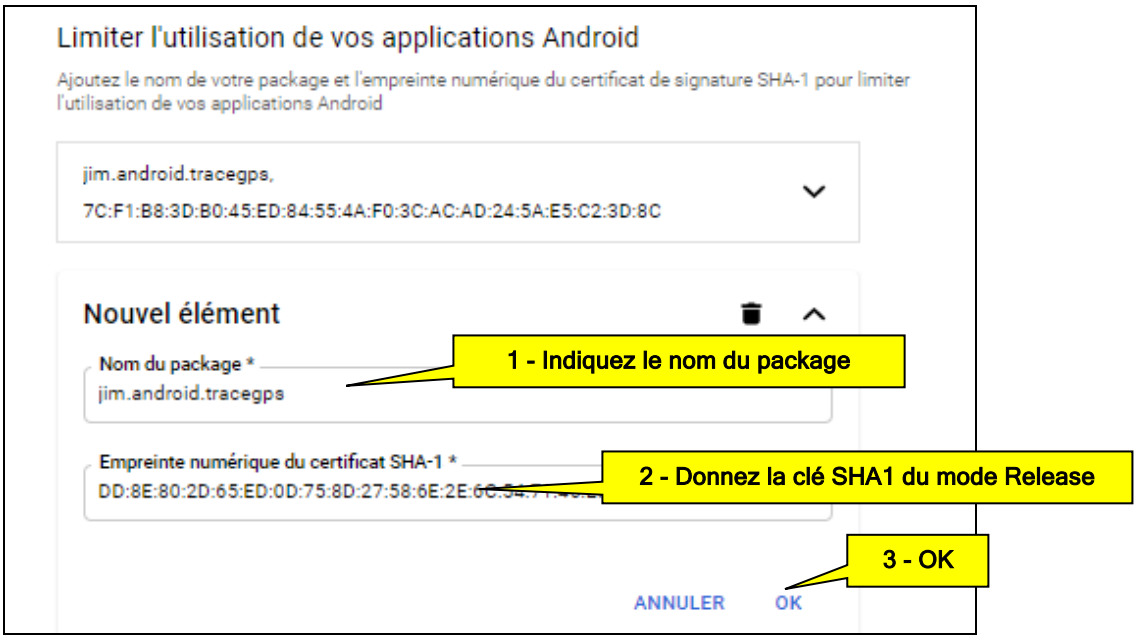
Ce qui donne les extraits d'écrans suivants :

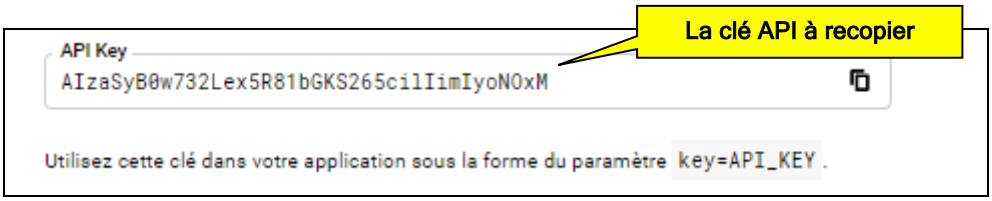






On souhaite que la clé API fonctionne également **en mode Release** :



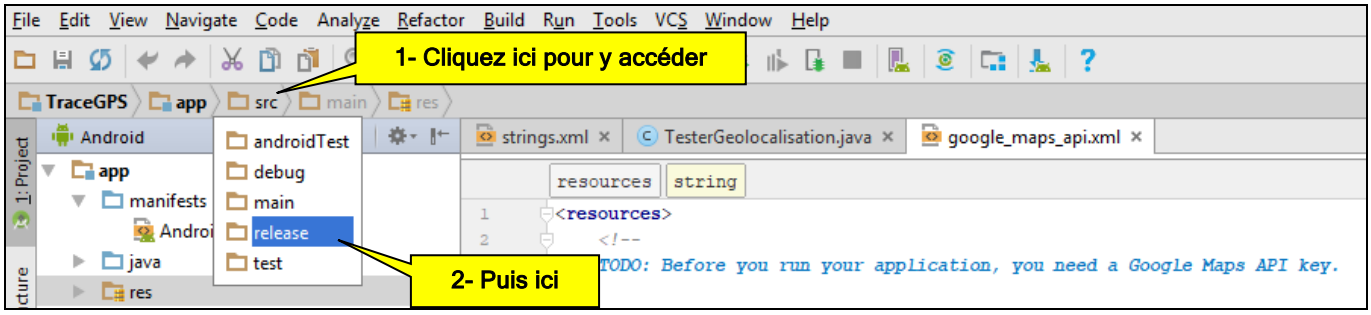


Enregistrez l'opération.

Puis collez la clé dans le fichier **google_maps_api.xml** du mode **Debug** (et enregistrez le fichier) :

```
<string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">
    AIzaSyB0w732Lex5R81bGKS265cillimIyoNOxM
</string>
```

Le fichier que vous venez de modifier correspond au mode **Debug**. Il faut également modifier le fichier **google_maps_api.xml** du mode **Release** accessible de la façon suivante :



Puis collez la clé dans le fichier **google_maps_api.xml** du mode **Release** (et enregistrez le fichier) :

```
<resources>
    <!--
    TODO: Before you release your application, you need a Google Maps API key.

    To do this, you can either add your release key credentials to your existing
    key, or create a new key.

    Note that this file specifies the API key for the release build target.
    If you have previously set up a key for the debug target with the debug signing certificate,
    you will also need to set up a key for your release certificate.

    Follow the directions here:

    https://developers.google.com/maps/documentation/android/signup

    Once you have your key (it starts with "AIza"), replace the "google_maps_key"
    string in this file.
    -->
    <string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">
        AIzaSyBcg0n1MDoH7yAanGuvVC2X1aT6cpQIw-w
    </string>
</resources>
```

Copiez la clé ici

Vous pouvez maintenant refermer ces 2 fichiers de configuration.

7- Modification de la programmation Java de MenuGeneral.java

Dans l'activité MenuGeneral.java, complétez l'écouteur d'événement associé au bouton **buttonTesterGeolocalisation** :

```
/** classe interne pour gérer le clic sur le bouton buttonTesterGeolocalisation. */
private class buttonTesterGeolocalisationClickListener implements View.OnClickListener {
    public void onClick(View v) {
        // crée une Intent pour lancer l'activité
        Intent uneIntent = new Intent(MenuGeneral.this, TesterGeolocalisation.class);
        // passe nom, mdp et typeUtilisateur à l'Intent
        uneIntent.putExtra(EXTRA_PSEUDO, pseudo);
        uneIntent.putExtra(EXTRA_MDP, mdp);
        uneIntent.putExtra(EXTRA_TYPE_UTILISATEUR, typeUtilisateur);
        // démarre l'activité à partir de l'Intent
        startActivity(uneIntent);
    }
}
```

Testez cette étape sur un mobile réel et corrigez les erreurs si besoin.

Le bouton Tester la géolocalisation doit activer l'activité **TesterGeolocalisation** ; vous obtiendrez une carte centrée sur la ville de **Sydney** en Australie (Google n'a pas choisi la ville de Rennes...) :

8- Création de l'interface graphique

Nous allons ajouter un Button pour permettre de revenir au menu général et un TextView pour afficher des consignes.

8-1 L'interface de base proposée par Android Studio

Le fichier **res / layout / activity_tester_geolocalisation.xml** contient la description de l'interface graphique :

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="jim.android.tracegps.TesterGeolocalisation" />
```

Il comporte un élément **<fragment>** occupant tout l'espace et dont l'identifiant est **map**.

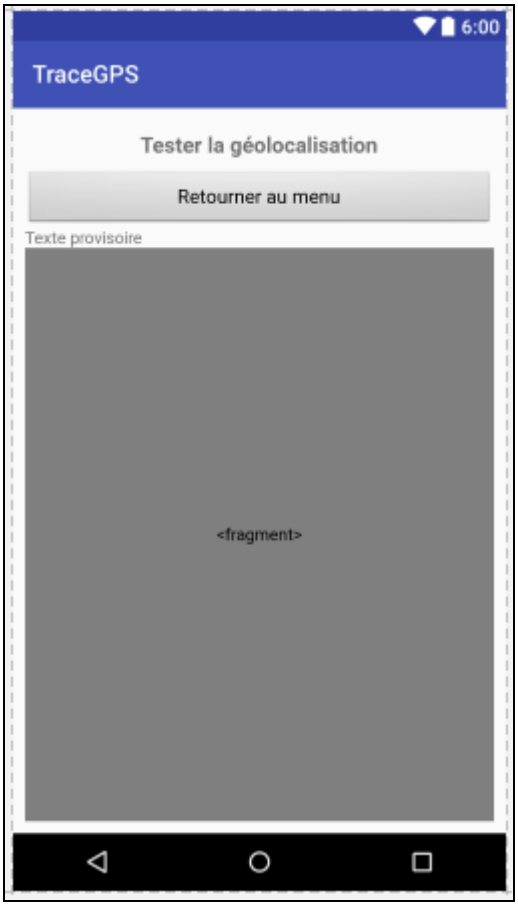
8-2 Modification de l'interface graphique

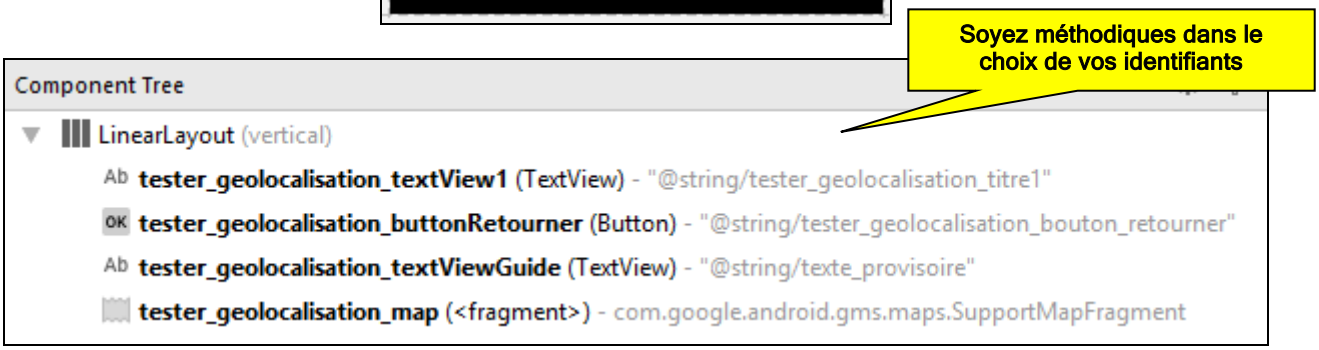
Il faut commencer par ajouter un **LinearLayout vertical** qui contiendra le bouton et le **Fragment**.
Cette modification ne peut pas se faire en mode **Design**. Passez en mode **Text** et modifiez le code :

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
tools:context="jim.android.tracegps.TesterGeolocalisation">

    <fragment xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/tester_geolocalisation_map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />

</LinearLayout>
```

Modifier l'id

Revenez maintenant en mode **Design** et placez les différents composants en suivant la structure suivante et en utilisant bien sûr les chaines du fichier **strings.xml** :





Le code XML de l'activité :

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context="jim.android.tracegps.TesterGeolocalisation">

    <TextView
        android:id="@+id/tester_geolocalisation_textView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingTop="10dp"
        android:paddingBottom="10dp"
        android:text="@string/tester_geolocalisation_titre1"
        android:textAlignment="center"
        android:textSize="18sp"
        android:textStyle="bold" />

    <Button
        android:id="@+id/tester_geolocalisation_buttonRetourner"
        style="@android:style/Widget.Button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingBottom="10dp"
        android:text="@string/tester_geolocalisation_bouton_retourner"
        android:textSize="16sp" />

    <TextView
        android:id="@+id/tester_geolocalisation_textViewGuide"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/texte_provisoire"
        android:textSize="14sp" />

    <fragment xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/tester_geolocalisation_map"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

9- Programmation Java de l'activité TesterGeolocalisation.java

9-1 Description du code Java initial

Le code initial du fichier **TesterGeolocalisation.java** est le suivant :

```
package jim.android.tracegps;

import android.support.v4.app.FragmentActivity;
import android.os.Bundle;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

public class TesterGeolocalisation extends FragmentActivity implements OnMapReadyCallback {

    private GoogleMap mMap;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_tester_geolocalisation);
        // Obtain the SupportMapFragment and get notified when the map is ready
        SupportMapFragment mapFragment = (SupportMapFragment)
                                         getSupportFragmentManager().findFragmentById(R.id.tester_geolocalisation_map);
        mapFragment.getMapAsync(this);
    }

    /**
     * Manipulates the map once available.
     * This callback is triggered when the map is ready to be used.
     * This is where we can add markers or lines, add listeners or move the camera. In this case,
     * we just add a marker near Sydney, Australia.
     * If Google Play services is not installed on the device, the user will be prompted to install
     * it inside the SupportMapFragment. This method will only be triggered once the user has
     * installed Google Play services and returned to the app.
     */
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

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

L'objet GoogleMap (qui sera renommé)

onCreate s'exécute lors de la création de l'activité et appelle le service Google Maps

Modifier l'identifiant du fragment

L'appel du service Google Maps en mode asynchrone (correspond à l'appel d'une AsyncTask)

onMapReady s'exécute dès que la carte est prête (correspond au onPostExecute d'une AsyncTask)

Remarquez que le **traitement asynchrone des tâches** est déjà intégré dans **l'API Google Maps**, et évite au développeur d'applications de mettre en oeuvre des objets **AsyncTask** :

- l'instruction **mapFragment.getMapAsync(this)** est équivalente au déclenchement d'une tâche asynchrone
- l'événement **onMapReady** est équivalent à l'événement **onPostExecute** de la classe **AsyncTask**

9-2 Modification du code Java

Dans le code initialement généré par Android Studio, l'objet **GoogleMap** représentant la carte est nommé **mMap** ; pour améliorer la lisibilité du code, vous allez le renommer **laCarte.**

Supprimez également les lignes barrées :

```
package jim.android.tracegps;

import android.support.v4.app.FragmentActivity;
import android.os.Bundle;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

public class TesterGeolocalisation extends FragmentActivity implements OnMapReadyCallback {

    private GoogleMap mMap;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout. activity_tester_geolocalisation);
        // Obtain the SupportMapFragment and get notified when the map is ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment)
                                                getSupportFragmentManager().findFragmentById(R.id.tester_geolocalisation_map);
        mapFragment.getMapAsync(this);
    }

    /**
     * Manipulates the map once available.
     * This callback is triggered when the map is ready to be used.
     * This is where we can add markers or lines, add listeners or move the camera. In this case,
     * we just add a marker near Sydney, Australia.
     * If Google Play services is not installed on the device, the user will be prompted to install
     * it inside the SupportMapFragment. This method will only be triggered once the user has
     * installed Google Play services and returned to the app.
     */
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

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

Supprimez les lignes barrées

Changez mMap en laCarte

Changez mMap en laCarte

9-3 Affichage des consignes et gestion du bouton Retourner au menu

Dans le fichier **TesterGeolocalisation.java**, ajoutez le code indiqué en gras :

```
package jim.android.tracegps;

import android.support.v4.app.FragmentActivity;
import android.os.Bundle;

import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;

import android.widget.Button;
import android.widget.TextView;
import android.view.View;

public class TesterGeolocalisation extends FragmentActivity implements OnMapReadyCallback {

    private GoogleMap laCarte;
    private Button buttonRetourner;
    private TextView textViewGuide;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_tester_geolocalisation);
        // Obtain the SupportMapFragment and get notified when the map is ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
                .findFragmentById(R.id.tester_geolocalisation_map);
        mapFragment.getMapAsync(this);

        // récupération des objets grâce à leur ID
        buttonRetourner = (Button) findViewById(R.id.tester_geolocalisation_buttonRetourner);
        textViewGuide = (TextView) findViewById(R.id.tester_geolocalisation_textViewGuide);

        // affichage des consignes
        String msg ="Appuyez sur le bouton localisation (en haut et à droite de la carte) pour tester la géolocalisation.\n";
        msg += "Si ce bouton est absent, autorisez la localisation pour cette application.\n";
        textViewGuide.setText(msg);

        // association d'un écouteur d'évenement au bouton
        buttonRetourner.setOnClickListener ( new buttonRetournerClickListener());
    }

    @Override
    public void onMapReady(GoogleMap googleMap) {
        laCarte = googleMap;
    }

    /** classe interne pour gérer le clic sur le bouton buttonRetourner. */
    private class buttonRetournerClickListener implements View.OnClickListener {
        public void onClick(View v) {
            finish();
        }
    }

} // fin de l'activité
```

Testez et vérifiez le bon fonctionnement du bouton **Retourner au menu**.

9-4 Ajout de la géolocalisation

La fonction de géolocalisation permet d'indiquer sur la carte la position de l'utilisateur au moyen d'un petit cercle bleu.

Un bouton **My position** (en haut et à droite) permet de centrer la carte sur ce point.

Commencez par ajouter les **import** suivants :

```
import android.Manifest;
import android.content.pm.PackageManager;
import androidx.core.content.ContextCompat;
```

Puis complétez la fonction **onMapReady** :

```
public void onMapReady(GoogleMap googleMap) {
    laCarte = googleMap;

    // active le calque Ma position
    if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) ==
            PackageManager.PERMISSION_GRANTED) {
        laCarte.setMyLocationEnabled(true);
    }
}
```

Le calque est activé seulement si le manifeste autorise la géolocalisation.

Exécutez l'application sur un mobile réel, et vérifiez que la géolocalisation et le bouton **My position** fonctionnent :