Google Maps (Localisation, itinéraire)

KACI Sofiane
06/12/2018

Résumé

La carte Google Map est une carte qui permet à un utilisateur notamment d'indiquer sa localisation sur la carte, de trouver les différentes adresses saisies par un utilisateur et enfin d'indiquer un itinéraire selon les critères de l'utilisateur entre deux adresses (ou même deux points sur la carte).

Dans ce TP, nous verrons tout d'abord comment créer une Google Map et générer une clé API Google. Ensuite, nous allons voir comment retrouver sa position sur la carte. Enfin, nous verrons comment créer un itinéraire entre deux adresses.

Prérequis

- *Savoir programmer une application Android avec plusieurs activités qui s'appellent entre elles*
- *Avoir une clé API Google (API direction, géolocalisation, API maps static) activé*

Code source

Code source **initial** disponible à

https://github.com/sofianepichichi/TP-Google-maps-initial

Code source **final** disponible à

https://github.com/sofianepichichi/TP-GoogleMaps-final

Explications du TP

Une Google Map est un élément important à utiliser dans les applications Android dans lesquels un utilisateur a besoin d'être localisé ou dans lesquels il est nécessaire d'afficher des périmètres ou des zones de déplacement.

Dans ce tutoriel simple, nous allons voir comment créer un une Google Map, l'utilisation des fragments, comment gérer les permissions des API Google comme Localisation et Direction API et faire des configurations au niveau d'Android studio afin d'exécuter une activité maps.

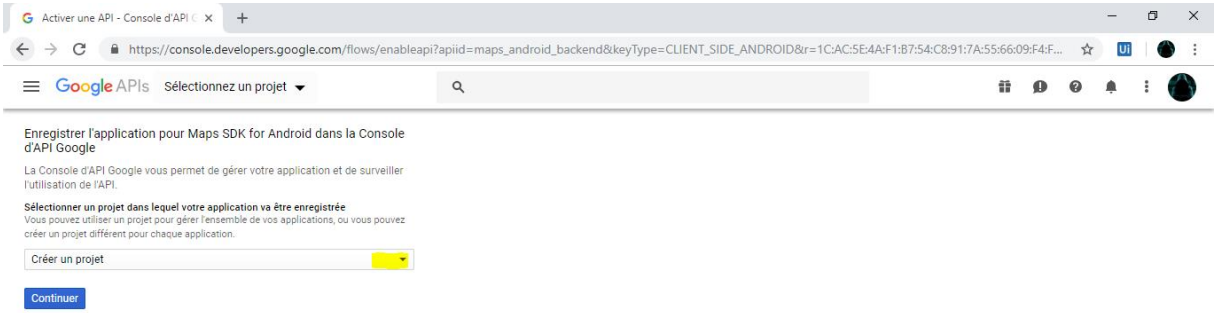
Etape 1 : Créer une Clé Google API pour votre projet.

Dans cette étape, nous allons créer une clé Google API, car la carte map est récupérée depuis une API Google. Il faut avoir un compte Gmail pour pouvoir se connecter sur le site.

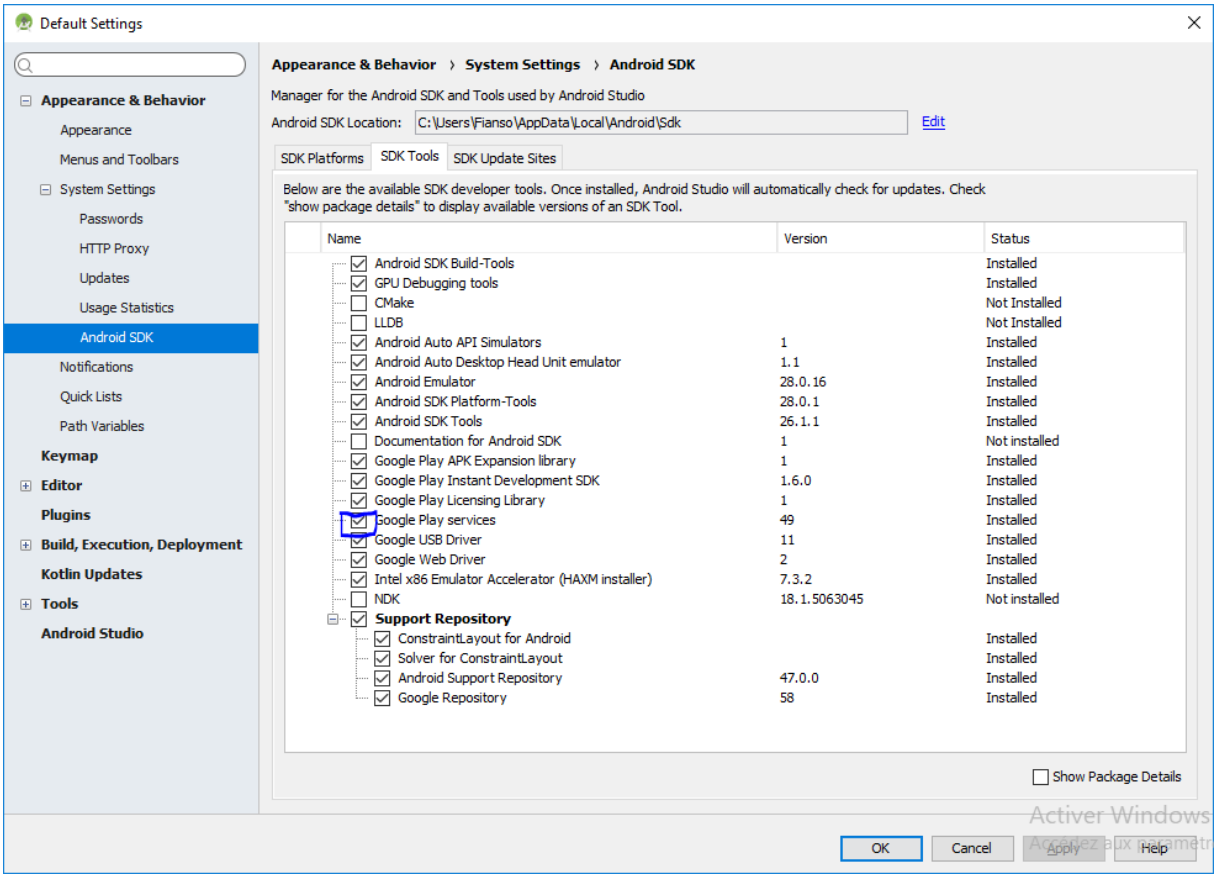
Voici le lien pour pouvoir créer votre clé Google API.

https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=1C:AC:5E:4A:F1:B7:54:C8:91:7A:55:66:09:F4:F9:59:73:26:F0:20%3Bcom.example.fianso.gpsdestination

Une fois sur le site suivez les étapes suivantes :



1. Cocher la case GooglePlayServices au niveau de SDKManager comme ci-dessous :



Etape 2 : Insertion de la Clé Google Api sur le projet.

Dans cette étape, nous allons faire un peu de configuration

*On commence par le Package **res** : il faut se rendre dans le sous package Values et ouvrir le fichier « Google_maps_api.xml »*

*//********** ETAPE 1 Insérer votre clé ici**********

```
<string name="google_maps_key" templateMergeStrategy="preserve" translatable="false"> insérez votre clé ici </string>
</resources>
```

*//********** ETAPE 1 Insérer votre clé ici**********

Ensuite, on passe au dossier **Manifests :** Ouvrez le fichier « AndroidManifest.xml » et insérez une fois de plus la clé dans la partie suivante :

```
<meta-data
  android:name="com.google.android.geo.API_KEY"
  android:value="Inserez votre clé ici " />
```

Etape 3 : Modifier les fichiers XML « activity_maps.xml » dans le répertoire Layout

Dans cette étape, nous allons créer la vue de notre application en utilisant un fragment. Nous allons y afficher la carte Google Maps, deux champs de texte pour respectivement et lieu de départ et la destination et un bouton pour valider la recherche d'un itinéraire.

*//********** ETAPE 3 implémenter le layout **********

Voir le fichier **activity_maps.xml**

Etape 4 : Afficher la carte GoogleMap

Dans cette partie, nous allons passer à la partie Code de notre application dans laquelle nous allons afficher la carte Google Map et ajouter une fonctionnalité qui va nous permettre de repérer notre position sur la carte Map.

*Une fois l'application configuré, il va falloir ajouter les implémentations et les services nécessaires pour afficher la carte Google Map dans l'activity **MapService.java.***

1. *Ajoutons les lignes suivantes dans **MapService.java** sous le commentaire suivant :*

```
//*** ETAPE 4 initialisation de la map ********
```

```
  // Initialisation de la map
private void initMap(){
      Log.d(TAG, "initialisation de la map");
      SupportMapFragment mapFragment = (SupportMapFragment)
      getSupportFragmentManager().findFragmentById(R.id.map);
      mapFragment.getMapAsync(MapsActivity.this);
}
```

*La méthode initMap sert à initialiser la carte. U*ne Google Map doit être acquise en utilisant getMapAsync(MapsActivity .this). Cette classe initialise automatiquement le système de cartes et la vue, et SupportMapFragement crée un fragment de carte en utilisant les options par défaut.

Etape 5 : Afficher ma Position sur la carteMap :

1. *Ajoutons les lignes suivantes dans **MapService.java** sous le commentaire suivant dans la méthode **OnMapReady()***

```
// * * * ETAPE  5 appeler le service google map lors d'un clic maintenu * *
* * * *
```

```
mMap.setOnMapLongClickListener(new GoogleMap.OnMapLongClickListener() {
    @Override
    public void onMapLongClick(LatLng latLng) {
        mMap.clear();
        mMap.addMarker(new MarkerOptions()
                .position(latLng)
                .title("Problem Location"));
        problemLocation = latLng;
    }
});

updateLocationUI();
getDeviceLocation();
```

*Nous allons faire appel au service GoogleMap lors d'un clic sur un bouton de la carte map. Dans le cas on l'on clique sur un button pour définir la géolocalisation de l'utilisateur, on récupère la latitude et la longtitude, et on ajoute un marker sur la position repéré. La méthode **updateLocationUI ()*** sert à mettre à jour la position de l'utilisateur à chaque déplacement, et la méthode **getDeviceLocation()** permet d'obtenir l'emplacement de l'appareil utilisateur.

2. *Ajoutons les lignes suivantes dans **MapService.java** sous le commentaire suivant dans la méthode getLocationPermission()*

```
// * * * ETAPE  6 récupérer les permissions pour la localisation * * * * *
*


if (ContextCompat.checkSelfPermission(this.context.getApplicationContext(),
        android.Manifest.permission.ACCESS_FINE_LOCATION)
        == PackageManager.PERMISSION_GRANTED) {
    mLocationPermissionGranted = true;
} else {
    ActivityCompat.requestPermissions(this.context,
            new String[]{android.Manifest.permission.ACCESS_FINE_LOCATION},
            PERMISSIONS_REQUEST_ACCESS_FINE_LOCATION);
}
```

3. *Ajoutons la méthode **OnrequestPermission( )***

```
// * * * ETAPE  7 recevoir la réponse des permissions obtenues* * * * * *


@Override
public void OnrequestPermission(int requestCode,
                                @NonNull String permissions[],
                                @NonNull int[] grantResults) {
    mLocationPermissionGranted = false;
    switch (requestCode) {
        case PERMISSIONS_REQUEST_ACCESS_FINE_LOCATION: {
            if (grantResults.length > 0
                    && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                mLocationPermissionGranted = true;
            }
        }
    }
    updateLocationUI();
}
```

*La méthode **OnrequestPermission** va demander si les permissions ont été obtenues par la méthode **getLocationPermission** et non plus et à chaque réception. La méthode updateLocationUI sera appelé pour mettre à jour la localisation de l'utilisateur, **OnrequestPermission** demande les permissions à l'utilisateur donc la demande pourrait être refusé ou annulé.*

4. *Ajoutons la méthode updateLocationUI(), pour cela, ajoutez la ligne de code suivante sous le commentaire dans la méthode updateLocationUI()*

```
// * * * ETAPE  8 mettre à jour la posistion de l'utilisateur* * * * * *


if (mMap == null) {
    return;
}
try {
    if (mLocationPermissionGranted) {
//on autorise l'api à afficher le bouton pour accéder à notre position
courante
        mMap.setMyLocationEnabled(true);
        mMap.getUiSettings().setMyLocationButtonEnabled(true);
    } else {

        mMap.setMyLocationEnabled(false);
        mMap.getUiSettings().setMyLocationButtonEnabled(false);
    }
} catch (SecurityException e) {
    Log.e("Exception: %s", e.getMessage());
}
```

Lorsque la couche Ma position est activée et les permissions de localisation sont obtenus, le bouton Ma position apparaît dans le coin supérieur droit de la carte. Lorsqu'un utilisateur clique sur le bouton, la caméra centre la carte sur l'emplacement actuel du périphérique s'il est connu. L'emplacement est indiqué sur la carte par un petit point bleu si l'appareil est immobile ou par un chevron si l'appareil est en mouvement.
La méthode setMyLocationEnabled() va autoriser l'application à afficher le bouton pour accéder à la position courante.

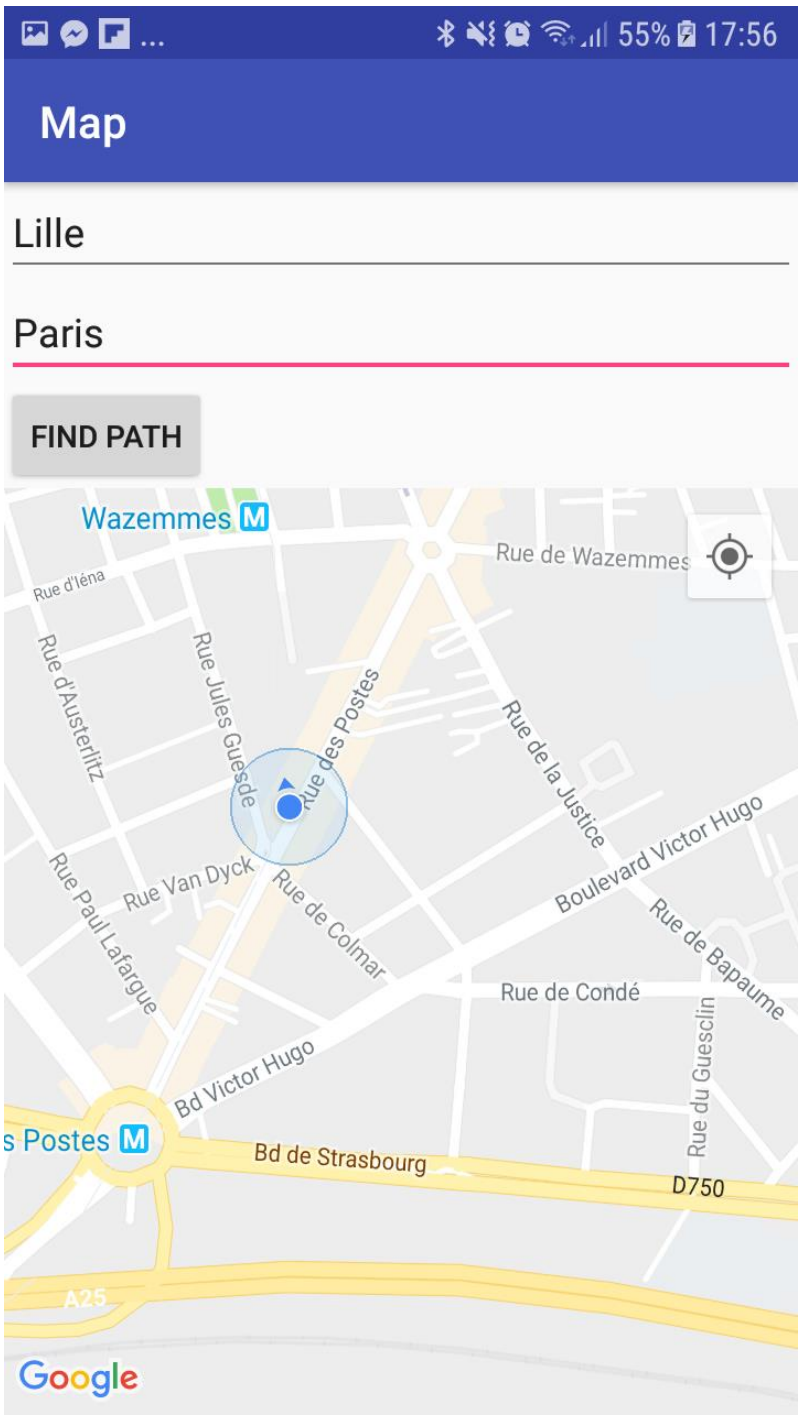
5. *Ajoutons la méthode **getDeviceLocation()**. Pour cela, ajoutez les lignes de code suivantes sous le commentaire dans la **getDeviceLocation().***

```
// * * * ETAPE  9 récupération de la position de l'appareil utilisateur* *
* * * *


try {
    if (mLocationPermissionGranted) {
        Task locationResult =
mFusedLocationProviderClient.getLastLocation();
        locationResult.addOnCompleteListener(this.context, new
OnCompleteListener() {
            @Override
            public void onComplete(@NonNull Task task) {
                if (task.isSuccessful()) {
                    mLastKnownLocation = (Location) task.getResult();
                        mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(
                                new
LatLng(mLastKnownLocation.getLatitude(),
                                        mLastKnownLocation.getLongitude()),
DEFAULT_ZOOM));
                    problemLocation = new
LatLng(mLastKnownLocation.getLatitude(),
                            mLastKnownLocation.getLongitude());
                } else {
                    Log.d("TAG", "Current location is null. Using
```

```
defaults.");
                    Log.e("TAG", "Exception: %s", task.getException());

mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(mDefaultLocation,
DEFAULT_ZOOM));
                    mMap.getUiSettings().setMyLocationButtonEnabled(false);
                }
            }
        });
    }
} catch (SecurityException e) {
    Log.e("Exception: %s", e.getMessage());
}
```

Dans cette méthode le service API Google va repérer la position de l'appareil de l'utilisateur après avoir donnée accès aux permissions nécessaire comme la localisation. La classe **mFusedLocationProviderClient** *permet d'obtenir l'emplacement actuel à l'aide du GPS et les fournisseurs réseaux. C'est le point d'entrée principal pour l'interaction avec le fournisseur d'emplacement fusionné.*
La méthode CleanMap Location permet de faire un clean sur la map à chaque changement de donnée si l'objet qui définit la map n'est pas vide.

Au lancement de l'application vous allez obtenir une interface comme ci-dessous :



Etape 7 : Activer L'API Google Direction

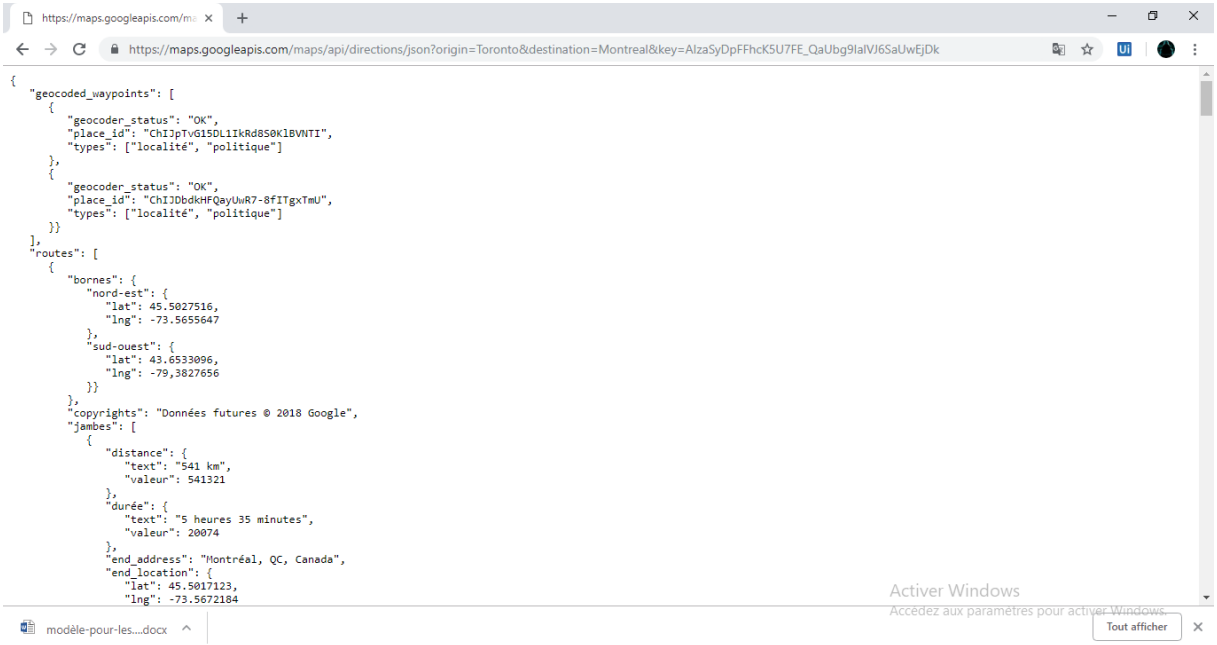
On va passer maintenant à la partie trouver un itinéraire entre deux adresses saisies. On a donc besoin de tracer des Polylignes pour spécifier l'itinéraire entre ces deux adresses.

Pour tracer des polylignes à mapper à travers des coordonnées données, utilisez la méthode : addPolyline ().

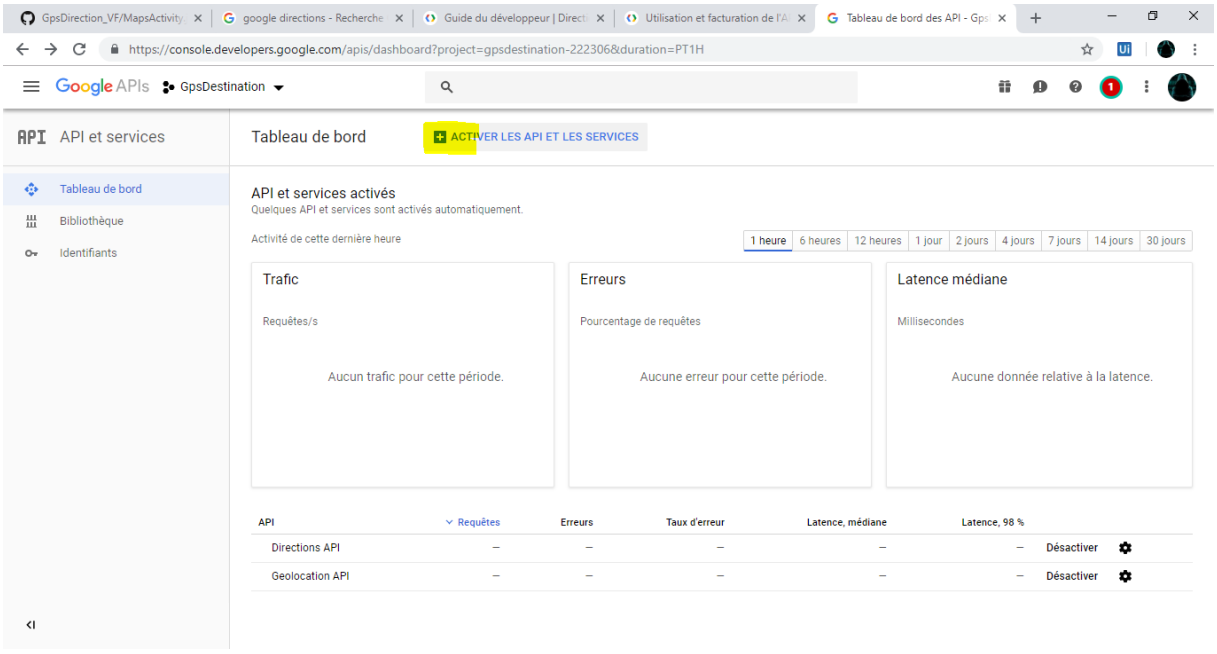
Avant de commencer, tester cette URL et ajouter votre clé API comme indiqué :

https://maps.googleapis.com/maps/api/directions/json?origin=Toronto&destination=Montreal&key=YOUR_API_KEY

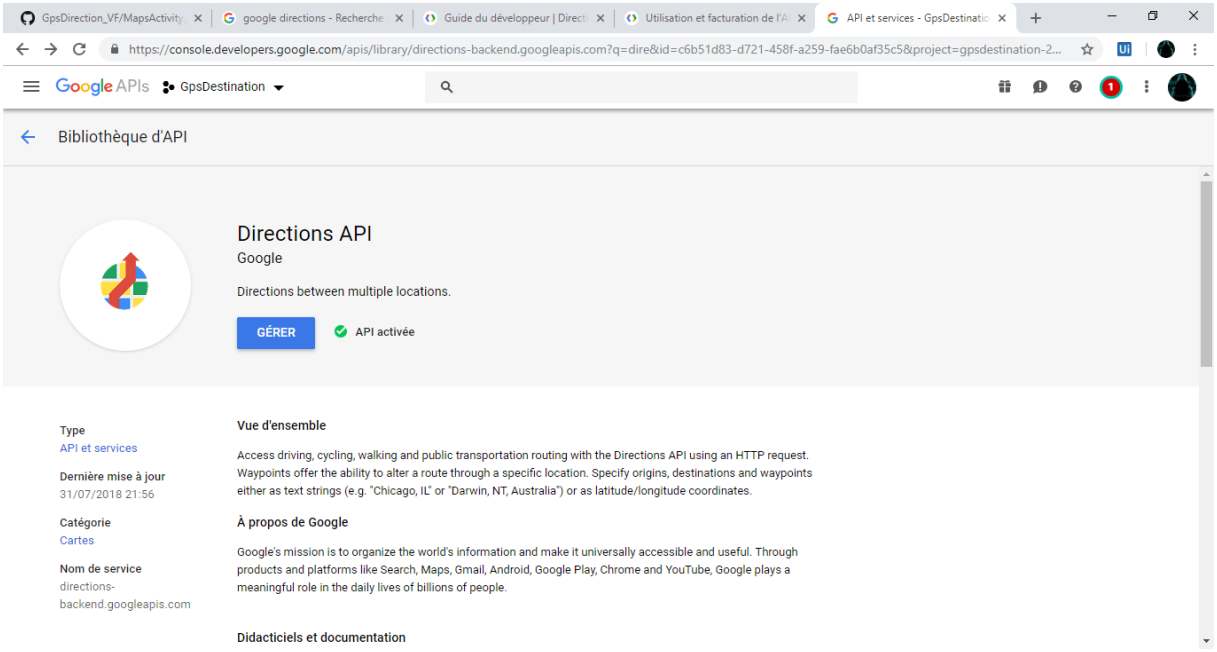
La demande suivante renvoie les itinéraires routiers de Toronto (Ontario) à Montréal (Québec). La sortie sera donc au format JSON. Google recommande d'utiliser le format JSON au lieu de XML, vous allez obtenir les résultats suivants dans la page web :



Pour afficher un itinéraire sur la carte avec votre Clé API, vous devez activer L'API Google Map Direction dans la console où vous avez créé auparavant votre clé. Sélectionner votre projet :



Cliquer sur « activer les Api et les services » et recherchez L'API Direction.



Activer votre API Direction (dans le but de pouvoir interagir avec votre API Direction via des requêtes pour obtenir les informations nécessaire (distance et Polyligne etc..)).

Etape 8 : Trouver le chemin entre deux endroits

Dans cette partie, je vais vous expliquer l'implémentation des méthodes nécessaire pour trouver le chemin entre deux endroits. Avant tout, nous allons implémenter la méthode :

```
// * * * ETAPE  10 Trouver le chemin entre deux adresse sur la map* * * * * *


// Trouver une direction entre deux adresses
@Override
public void onDirectionFinderStart() {
    progressDialog = ProgressDialog.show(this, "Please wait.",
            "Finding direction..!", true);

    if (originMarkers != null) {
        for (Marker marker : originMarkers) {
            marker.remove();
        }
    }

    if (destinationMarkers != null) {
        for (Marker marker : destinationMarkers) {
            marker.remove();
        }
    }

    if (polylinePaths != null) {
        for (Polyline polyline:polylinePaths ) {
            polyline.remove();
        }
```

```
    }
}
```

à cette étape penser à faire l'implémentation de l'interface ***DirectionFinderListener dans l'activity MapService.***

class MapService implements OnMapReadyCallback , MapServiceInter,DirectionFinderListener { les méthodes}

Pareil au niveau de l'activity ***MapsActivity***

public class MapsActivity extends AppCompatActivity implements DirectionFinderListener { les méthodes }

```
      // * * * ETAPE  11 Récupérer les chemins obtenus* * * * * *
@Override
public void onDirectionFinderSuccess(List<Route> routes) {
    progressDialog.dismiss();
    polylinePaths = new ArrayList<>();
    originMarkers = new ArrayList<>();
    destinationMarkers = new ArrayList<>();


    for (Route route : routes) {

        originMarkers.add(mMap.addMarker(new MarkerOptions()
                .title(route.startAddress)
                .position(route.startLocation)));
        destinationMarkers.add(mMap.addMarker(new MarkerOptions()
                .title(route.endAddress)
                .position(route.endLocation)));

        PolylineOptions polylineOptions = new PolylineOptions().
                geodesic(true).
                color(Color.BLUE).
                width(10);

        for (int i = 0; i < route.points.size(); i++)
            polylineOptions.add(route.points.get(i));

        polylinePaths.add(mMap.addPolyline(polylineOptions));
    }
}
```

Nous avons besoin d'un lien à coder sous forme d'une URL que l'on a vu au début.

Nous avons besoin d'une fonction CreateUrl () pour créer un lien à partir de l'adresse origine saisi et la destination et de la clé. Après le téléchargement, une méthode qui s'appelle OnPostExecute() sera traité. Puis, ParseJSON sera appelé à son tour pour obtenir les données nécessaires.

JSONObject jsonData = new JSONObject(data) va créer un JSONObject à partir d'une chaine de données. Ensuite, nous allons l'appeler pour chaque objet du tableau, on va récupérer chaque objet et le traiter comme Polyline, durée etc.

Pour obtenir les routes JSONArray à partir de jsonData. Après le traitement complet de JSON, il retournera les routes de la liste<Routes>, et la fonction d'interface « onDirectionFindersSuccess » de Listener pour afficher le résultat comme ceci : routes.add(route).

Lorsque L'API Direction renvoie des résultats, elle les places dans un routes tableau JSON, même si le service ne renvoie aucun résultat, si l'adresse originale ou destination est vide, il renvoie néanmoins un ***routes tableau*** *vide.*

L'encodage polyline est un algorithme de compression avec perte qui vous permet de stocker une série de coordonnées sous la forme d'une chaîne unique

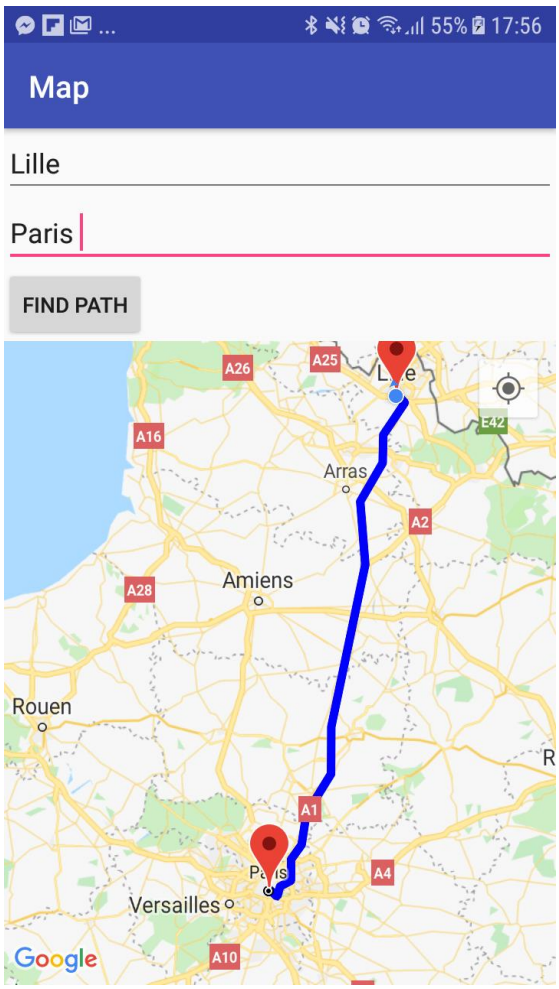
Nous avons également Besoin de ***PolylinePath*** *pour stocker les polylines renvoyées par mMap.addPolyline, que nous pourrons supprimer ultérieurement si nous n'en avons plus besoin.*

```
      // * * * ETAPE  12 ajouter votre clé pour récupérer les données du chemin* * * * * *
```

Rajoutez votre clé et stocker la dans la variable GOOGLE_API_KEY dans l'activity ***DirectionFinder***

```
private static final String DIRECTION_URL_API = "https://maps.googleapis.com/maps/api/directions/json?";

private static final String GOOGLE_API_KEY = "Votre clé API ici ";
```

Vous pouvez lancer votre application.



Informations complémentaires

https://developers.google.com/maps/documentation/geocoding/usage-and-billing: *permet à un développeur de comprendre comment fonctionne les quotas pour les requête API*

NB: un seul quota de requête quotidien pour cette API. Sinon facturation obligatoire.

https://developers.google.com/maps/documentation/directions/intro : *permet à un débutant de s'auto-former et de comprendre le fonctionnement des API Google (Direction, Localisation)*

https://developers.google.com/maps/documentation/geocoding/intro : *ce lien vous permet de convertir une adresse saisi en texte en Lagtng (latitude, longitude) et d'autres informations utiles.*

https://developers.google.com/android/guides/permissions