L'API Google Maps

Utiliser les cartes de Google

- Un des grands avantages d'Android est de pouvoir bénéficier des principales applications déjà développées par Google. L'une d'elles sont les Google maps
- Toute une API permet d'utiliser ces cartes Google

La bibliothèque Google Maps

- On utilise les Google Maps Android API v2. La page d'accueil de cette technologie est à l'URL `https://developers.google.com/maps/documentation/android-api/`
- Cette API permet de manipuler des cartes terrestres. Ces classes se trouvent dans le package `com.google.android.gms.maps`
- Pour afficher une carte, on utilise les fragments
- Cette API gère les entrées clavier, le zoom, le toucher sur une carte affichée
- On peut ajouter des dessins, des images sur la carte
- Pour utiliser cette API, on doit être enregistré auprès du service Google Maps et avoir obtenu une clé Maps API v2
- On peut commencer à étudier cette API à partir de l'URL `https://developers.google.com/maps/documentation/android/intro`

Les cartes de Google : avant tout

- Il faut avoir un compte Google : un gmail suffit
- Si on veut faire afficher le résultat dans un AVD, c'est possible ! Mais cet AVD doit avoir les APIs Google

Les étapes pour utiliser les cartes de Google

- Pour utiliser les cartes Google avec Android Studio, il suffit de créer un projet dans le cloud Google qui gère les Google maps et d'obtenir une Maps API v2 Key
- Toute la procédure est indiquée à
  `https://developers.google.com/maps/documentation/android/start`
  voire à
  `https://developers.google.com/maps/documentation/android-api/signup`

Obtenir une Maps API v2 Key (1/7)

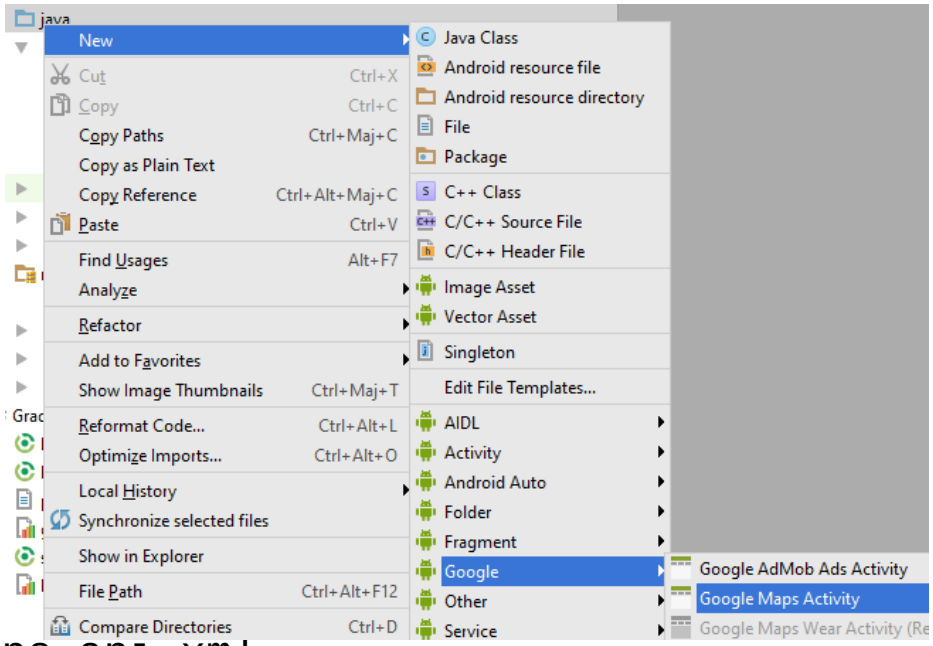
- Une Maps API key est obtenue en envoyant le couple (réduit du certificat de la clé publique de signature de l'application, nom du paquetage de l'application)
- Comme toute application Android est signée, les clés Google API sont liées à ce couple (clé publique de signature de l'app, paquetage de l'application). Elles ne dépendent pas des utilisateurs (et de leur nombre)
- Rappel : les app utilisent un algorithme de cryptage asymétrique (clé privée, clé publique cf. RSA par exemple). Les app sont signées par une clé privée
- En mode debug, sur une machine donnée, toutes les apps ont le même couple de clés. La clé publique est appelée la debug key
- En mode release, on peut construire un couple de clés propre à chaque app. Il est conseillé de garder ce couple de clés pour toutes les versions d'une app donnée

Obtenir une Maps API v2 Key (2/7)

- L'exposé ci dessous est fait avec le "Debug Certificate". C'est similaire pour le "Release Certificate" (= certificate de la clé de publication de l'application)
- Il y a 3 étapes :
  - a) obtenir l'empreinte (réduit , digest, hashcode, ...) SHA1 du certificate de la clé publique (de debug)
  - b) créer (ou utiliser) un projet Google qui gère les cartes Google
  - c) obtenir une clé "API key" de Google de ce projet pour votre app
- En fait, Android Studio permet de faire ces trois étapes très rapidement

Obtenir une Maps API v2 Key (3/7)

- Créer une activité qui va afficher une carte Google par New | Google | Google Maps Activity



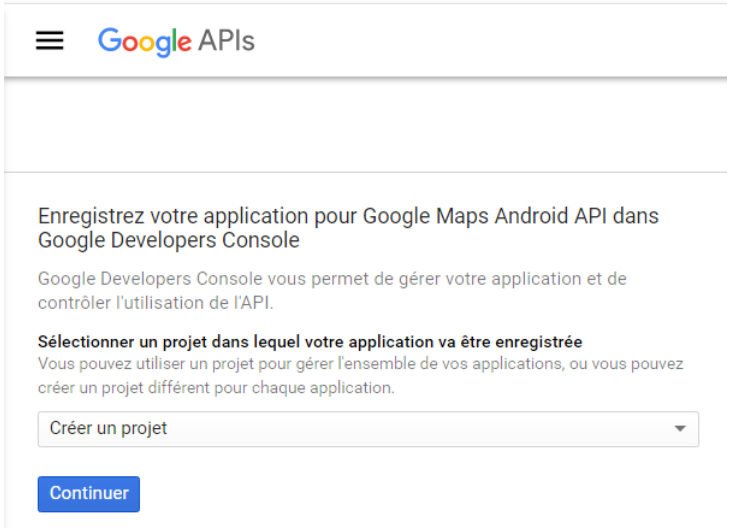
- Il est alors créé :
  - une activité adaptée
  - un fichier `google_maps_api.xml`
- De plus l'`AndroidManifest.xml` est enrichi

Obtenir une Maps API v2 Key (4/7) : `google_maps_api.xml`

- Dans le fichier `google_maps_api.xml` généré, les commentaires sont très intéressants
- Entre autre une URL qui va faire (presque) tout le travail !
- On reconnait dans ce fichier le couple *réduit SHA-1 du certificat* ;*paquetage de l'app* (sans gribouillis ici)
- De plus l'emplacement est prévu pour mettre la clé Maps API v2 qu'on va récupérer

Obtenir une Maps API v2 Key (5/7) : projet cloud Google

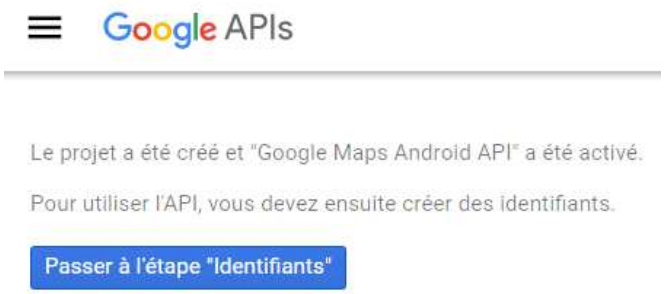
- Se connecter à l'URL indiquée
- On a alors :



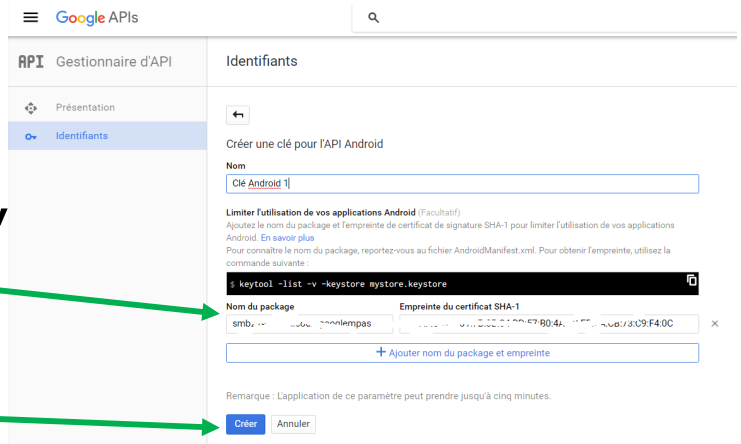
- On peut :
  - soit créer un projet en cliquant sur le bouton Continuer,
  - soit voir tous les projets déjà créés dans le cloud Google en cliquant le menu "Créer un projet"
  - (bravo l'ergonomie de l'IHM ;-))
- Après le choix ou la création d'un projet, cliquer le bouton Continuer

Obtenir une Maps API v2 Key (6/7) : projet cloud Google

- On passe ensuite à l'écran :



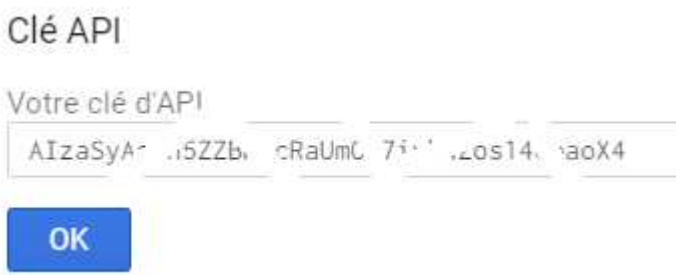
- Il indique qu'un projet Google configuré pour traiter (envoyer) des Google maps a bien été créé
- En cliquant sur le bouton Identifiants on a :



- Le couple (réduit du certificat de la clé d'encryptage de l'application, nom du paquetage de l'application) est déjà écrit !
- Cliquer sur le bouton Créer

Obtenir une Maps API v2 Key (7/7) : projet cloud Google

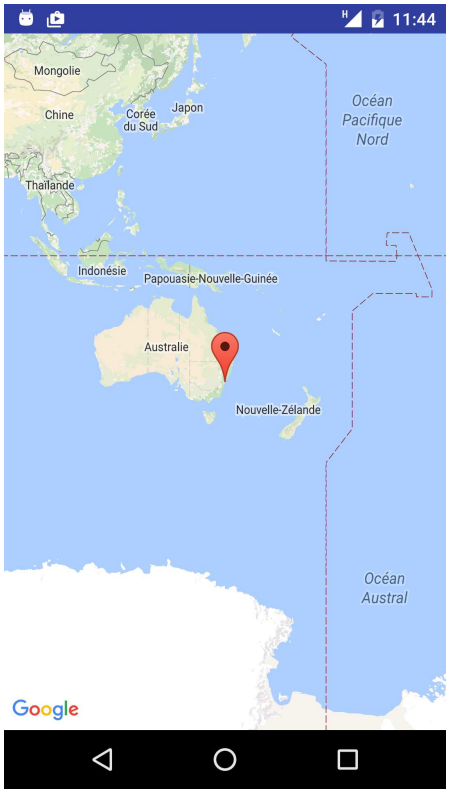
- On obtient la maps API Key de valeur commençant par `AIza...` :



- L'insérer alors dans le fichier `google_maps_api.xml` en remplacement de `YOUR_KEY_HERE`. Ne pas ajouter de retour chariot, d'intentation, de caractère espace, etc. ;-)

Exécution du programme sur un smartphone

- En lançant l'activité qui gère la google map on obtient :
- Remarque : tout a été bien configuré par Android Studio à savoir
  - la création du projet Google cloud pour les google maps
  - la configuration de l'`AndroidManifest.xml`
  - le code de l'activité et son fichier d'IHM XML associé
- Vérifier ces parties dans votre app : elles ne sont pas banales



L'activité fournie par Android studio

- Android studio fournit l'activité :

```
public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {
    private GoogleMap mMap;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        SupportMapFragment mapFragment =
          (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }

    @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));
    }
}
```

Le fichier XML qui affiche une carte Google

- Un `activity_maps.xml` qui affiche la carte est :

```
<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="classeFournissantLeContexte" />
```

- La classe fournissant le contexte est souvent l'activité qui charge ce fragment

Explication du code (1/3)

- Pour modéliser les couples (latitude, longitude) on dispose de la classe `com.google.android.gms.maps.model.LatLng`.
  Pour accéder à la latitude et la longitude d'un objet de cette classe on appele directement les champs publics :
  `public final double latitude` (la latitude en degrés)
  `public final double longitude` (la longitude en degrés)
  (beurk)
- Les valeurs pour latitude et longitude comportent souvent des décimales (obligatoire si on utilise une échelle plus petite)

Explication du code (2/3)

- Le chargement de la carte est fait dans une thread autre que la UI thread. Au `mapFragment` obtenu avec :

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

  on associe un objet d'une classe implémentant l'interface `OnMapReadyCallback` par `mapFragment.getMapAsync(...);`

- Lorsque la carte est chargée, cet objet est averti et lance sa méthode `public void onMapReady(GoogleMap map)`

Explication du code (3/3)

- La méthode `public void onMapReady(GoogleMap map)` possède la carte (de type `GoogleMap`) en paramètre. On peut centrer cette carte par :

```
mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
```

avec

```
LatLng sydney = new LatLng(-34, 151);
```

- On peut centrer et préciser le zoom par

```
mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(Sydney, 2.0f));
```

Position d'un point terrestre

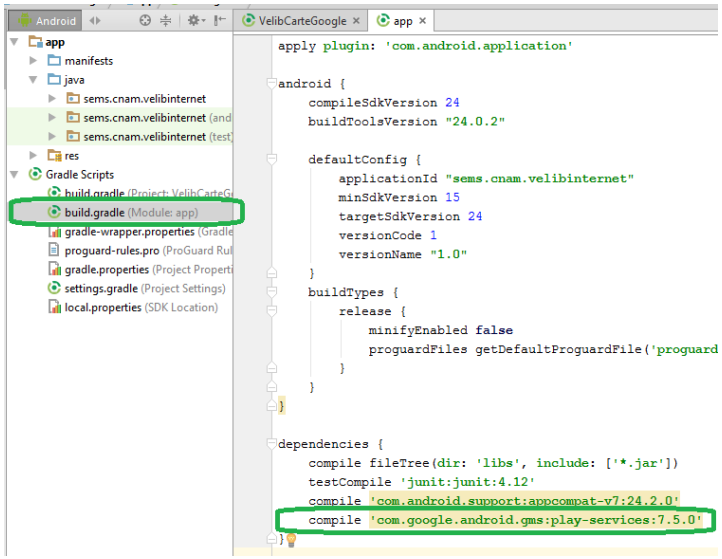
- Un objet de la classe `CameraUpdateFactory` modélise un point de vue
- Les méthodes (toutes statiques) de cette classe modifie le point de vue (zoom, déplacement, animation de déplacement, ...)
- La méthode `public static CameraUpdate newLatLngZoom (LatLng latLng, float zoom)` "returns a `CameraUpdate` that moves the center of the screen to a latitude and longitude specified by a `LatLng` object, and moves to the given zoom level."
- La valeur de zoom est ajustée de `2.0f` à `21.0f` (zoom maximal) : c'est un `float`
- La méthode `public final void moveCamera (CameraUpdate update)` de la classe `GoogleMap` positionne le point de vue indiqué par l'argument. Le mouvement est instantaté

Bidouille sur la version de la bibliothèque Google Play services

- Anciennement Android Studio proposait une version 9.8.0. Cette version fonctionne ... mal ! On avait des erreurs de multidex
- Il fallait alors mieux utiliser la version des Google Play 7.5.0 (merci Tarik)
- On l'indique dans le fichier `build.gradle` du module par :

```
compile 'com.google.android.gms:play-services:7.5.0'
```

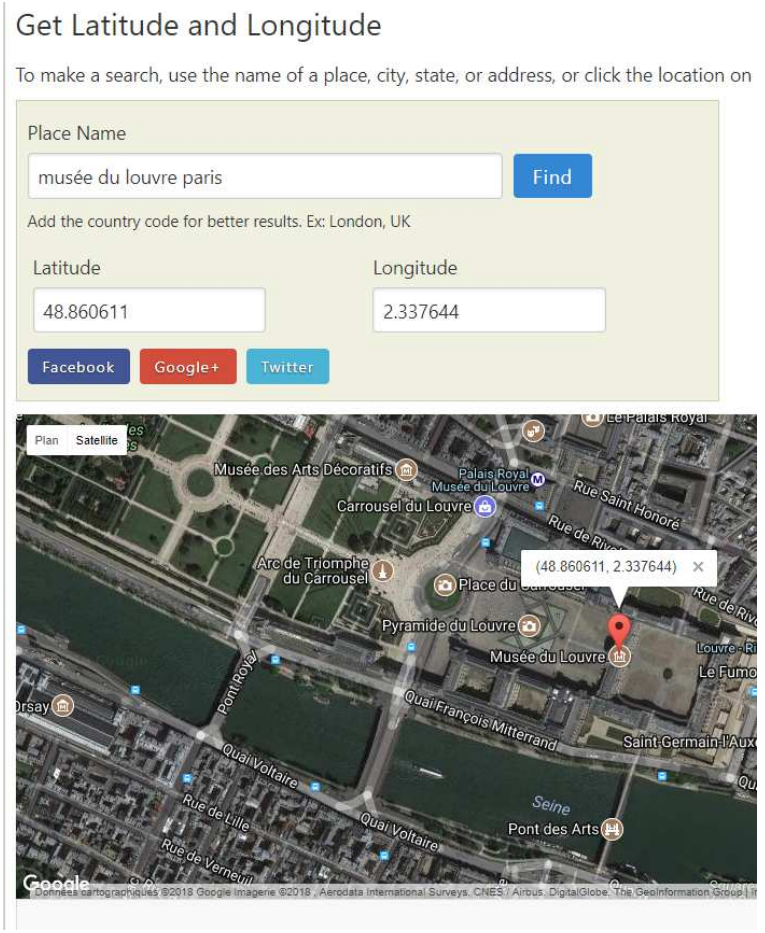
- C'est à dire :



- Mais cela c'était avant ! Désormais on a la version 10.2.1 des play-services-maps qui fonctionne bien

Conversion adresse d'un point terrestre en (longitude, latitude)

- Utiliser le site `https://www.latlong.net/`
- Indiquez l'adresse
- On obtient la latitude et la longitude
- Et un belle carte pour vérifier !



Bulle d'aide sur un marqueur

- Pour ajouter un marqueur sur une carte on écrit :

```
mMap.addMarker(new MarkerOptions()
     .position(sydney)
     .title("Marker in Sydney"));
```

- Avec une bulle d'aide contenant un titre, un commentaire et une image associée, il suffit d'écrire :

```
mMap.addMarker(new MarkerOptions()
     .position(GAUMONT_CHAMPS_ELYSEES)
     .title("cinéma gaumont")
     .snippet("super le cinéma")
     .icon(BitmapDescriptorFactory.fromResource(R.drawable.cinema)));
```

- Les méthodes `position(...)`, `title(...)`, `snippet(...)`, `icon(...)` retourne le `MarkerOptions` sur lequel elles ont été lancées. D'où l'enchainement des appels

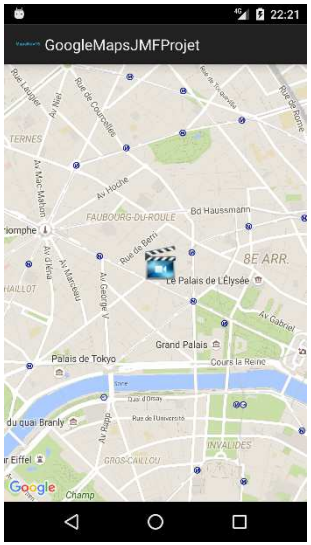
Afficher une carte Google centrée et avec marqueurs

- Le code de `onMapReady()` :

```
private static final LatLng PARIS_CHAMPS_ELYSEES =
     new LatLng(48.870209, 2.306268);
private static final LatLng GAUMONT_CHAMPS_ELYSEES =
     new LatLng(48.870386, 2.306793);

public void onMapReady(GoogleMap map) {
     mMap.addMarker(new MarkerOptions().position(GAUMONT_CHAMPS_ELYSEES)
               .icon(BitmapDescriptorFactory.fromResource(R.drawable.cinema)));
     mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(PARIS_CHAMPS_ELYSEES, 14));
}
```

fait afficher :



Marqueur "cliquable"

- Pour lancer du code quelconque lorsqu'un marqueur est cliqué, il suffir d'ajouter un `OnMarkerClickListener` à la `GoogleMap`
- Par exemple :

```
mMap.setOnMarkerClickListener(new OnMarkerClickListener() {
  @Override
  public boolean onMarkerClick(Marker marker) {
     Toast leToast = Toast.makeText(getApplicationContext(), "message", Toast.LENGTH_LONG);
     leToast.show();
     return false;
  }
});
```

- L'argument de `onMarkerClick(...)` est le marqueur utilisé (qui a été retourné par la méthode `addMarker()`)
- tutorial sur les marqueurs à `https://developers.google.com/maps/documentation/android/marker`

Bibliographie pour ce chapitre

- La page d'accueil de Maps Android API :
  `https://developers.google.com/maps/documentation/android-api/`
- Le tutorial de Google sur les Maps Android API v2 :
  `https://developers.google.com/maps/documentation/android/start`
- Le tutorial de Lars Vogel :
  `http://www.vogella.com/tutorials/AndroidGoogleMaps/article.html`
- La documentation des classes Google Maps Android :
  `http://developer.android.com/reference/com/google/android/gms/maps/package-summary.html`

Fin