

Cas EDF : Développement Android - Concepts avancés – Partie 2

Cette publication comporte cinq parties dont l'ordre est dicté par la logique du développement. Les parties 2 et 3 sont facultatives.

- Partie 1 : Gestion des clients
- **Partie 2 : Géolocalisation de l'agent et géocodage du client sélectionné**
- Partie 3 : Signature Client
- Partie 4 : Communication avec le serveur
- Partie 5 : Identification, import et export des données.

Description du thème

Propriétés	Description
Intitulé long	Cas EDF : Développement Android - Concepts avancés - Partie 2 : Géolocalisation de l'agent et géocodage du client sélectionné
Formation concernée	BTS Services Informatiques aux Organisations
Matière	SLAM 4
Présentation	Développement permettant d'aborder des concepts de la programmation Android d'une application embarquée, communiquant avec un serveur. Il aborde les notions : <ul style="list-style-type: none">➤ d'affichage de liste / d'adapter,➤ de GEOLOCALISATION / GEOCODER,➤ de graphisme (canvas) et d'encodage JPG,➤ d'échange avec un serveur WEB (THREAD / JSON / GSON),➤ d'utilisation d'un SGBDO DB4o.
Notions	Savoirs <ul style="list-style-type: none">• D4.1 - Conception et réalisation d'une solution applicative• D4.2 - Maintenance d'une solution applicative Savoir-faire <ul style="list-style-type: none">• Programmer un composant logiciel• Exploiter une bibliothèque de composants• Adapter un composant logiciel• Valider et documenter un composant logiciel• Programmer au sein d'un framework
Transversalité	SLAM5
Pré-requis	Développement d'une application Android sous un environnement Eclipse. (Exemple : Cas AMAP Jean-Philippe PUJOL)
Outils	Eclipse, DB4o, OME, Gson, Google play services, Apache, Mysql
Mots-clés	Application mobile, Android, SGBDO, DB4o, Géolocalisation, Géocodage, Thread, json, Gson, MVC, canvas, encodage JPG
Durée	24 heures (8,4,4,4,4) (Temps divisé par 2 si utilisation du squelette application)
Auteur	Pierre François ROMEUF avec la relecture et les judicieux conseils de l'équipe CERTA
Version	v 1.0
Date de publication	Juin 2014

Géolocalisation de l'agent et géocodage du client sélectionné

Activity Geolocalisation

Exemple d'écran d'affichage de position géolocalisée de l'agent et de l'adresse géocodée du client que nous souhaitons créer :



Attention pour géocoder, les tablettes en 4.1.1 ne sont pas compatibles. **Mettre à jour le micro logiciel.**

Si vous n'avez pas votre propre STA, téléchargez le simulateur Android Genymotion (avec le Galaxy Nexus 4.2.2 Api 17) puis téléchargez l'application Google Apps 20130812 à l'adresse http://wiki.rootzwiki.com/Google_Apps#20130812 et suivez les indications de <http://blog.zeezonline.com/2013/11/install-google-play-on-genymotion-2-0/> afin d'installer l'application. Puis, une fois l'application installée, ajoutez votre compte Google.

Vous pouvez modifier vos coordonnées de géolocalisation en cliquant sur GPS.

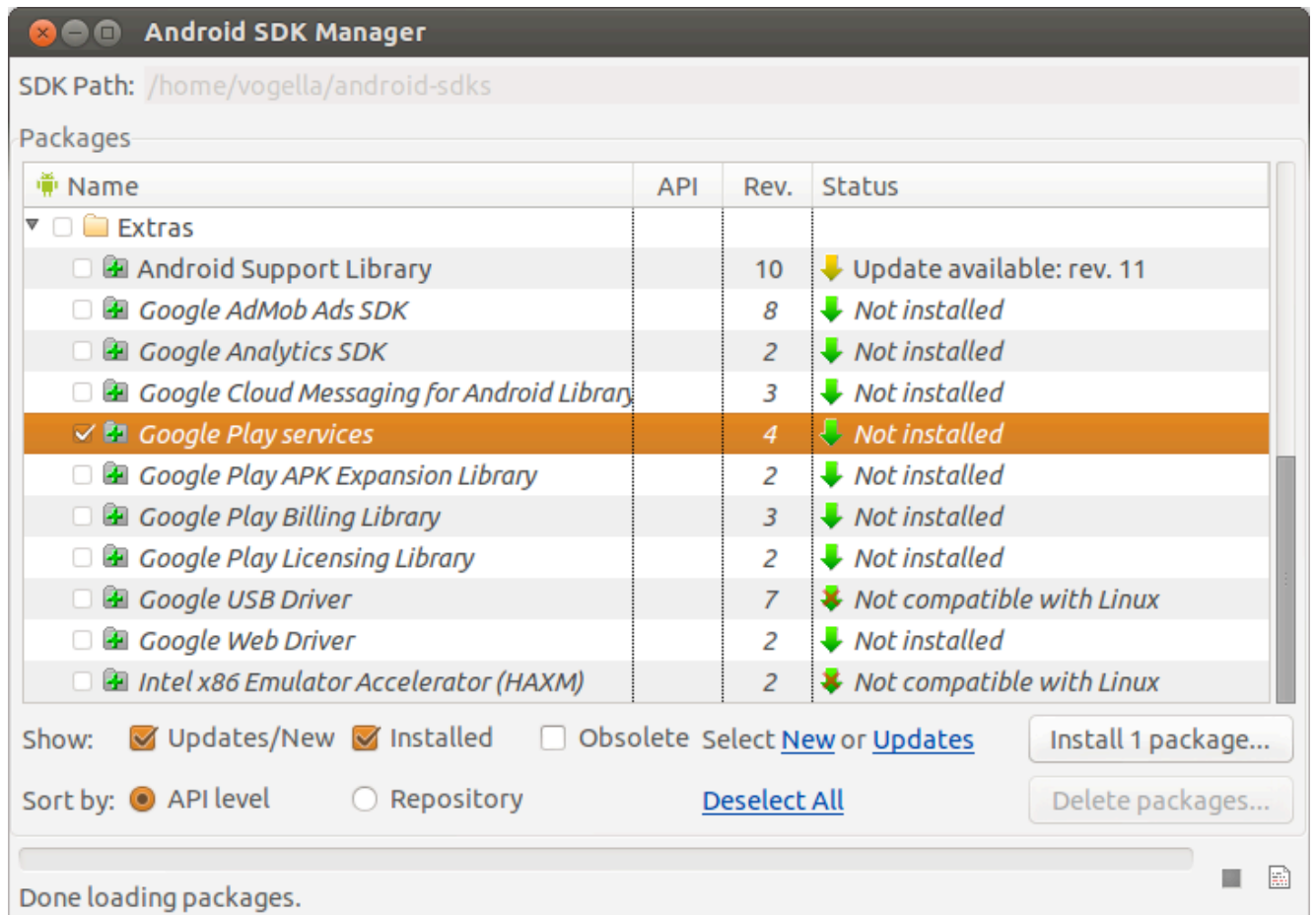
➤ Créez une nouvelle *Activity Geolocalisation*.
Cette *Activity* implémente *LocationListener*

➤ Associez l'activity Geolocalisation au clic du bouton 'Geoloc' de l'Activity ModificationClient

A partir de votre l'Activity ModificationClient faites appel à Geolocalisation sur le clic du bouton 'Geoloc' en lui passant l'identifiant du client.

Installation de Google Play services

➤ Ouvrez l' *Android SDK Manager* et installez *Extras* → *Google Play services*



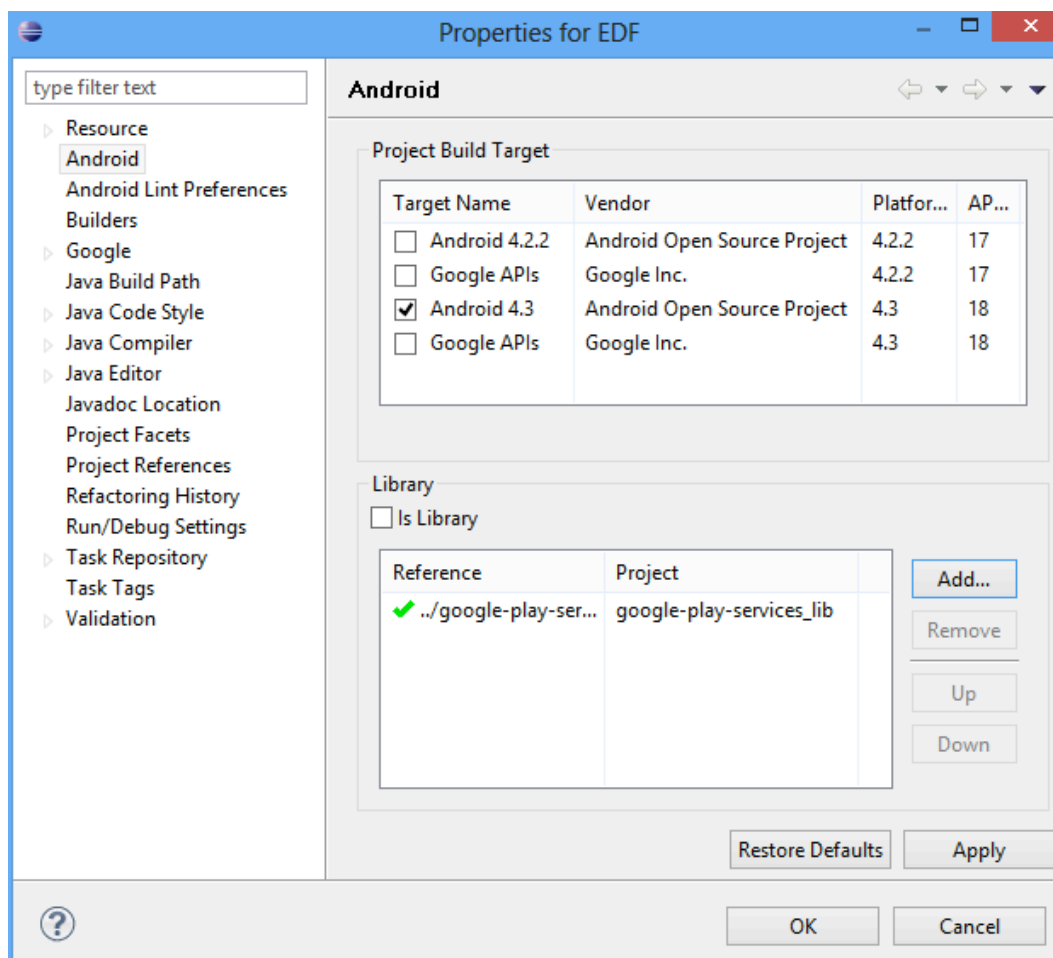
➤ Importez la librairie téléchargée `google-play-services_lib`
File → *Import* → *Android* → *Existing Android Code into Workspace*.

`google-play-services_lib` se trouve à l'emplacement `sdk\extras\google\google_play_services\libproject\google-play-services_lib`
sdk étant l'emplacement de votre sdk que vous pouvez retrouver via *Windows / Preferences / Android*.

Ne pas oublier **Copy projects into workspace**

➤ Faites un "build" du projet `google-play-services_lib`

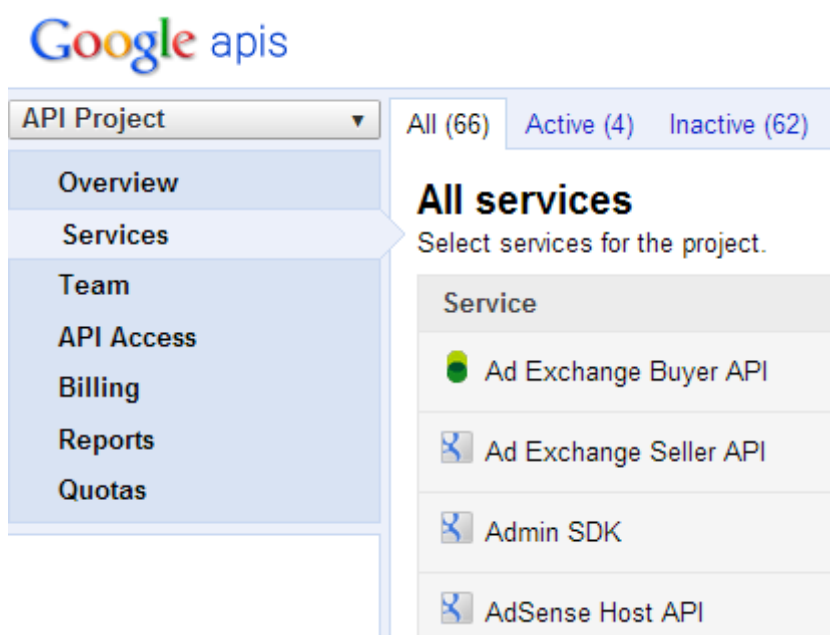
- Ajoutez dans les dépendances de votre projet EDF, le projet *google_play_services_lib* via

















Obtention de la clé Google Maps API

- Connectez vous sur votre compte Google <https://code.google.com/apis/console/>
Clic sur 'Old console'

A partir de services



➤ Autorisez :

 Google Contacts CardDAV API		<input type="checkbox"/> OFF
 Google Maps Android API v2		<input checked="" type="checkbox"/> ON
 Google Maps API v3		<input checked="" type="checkbox"/> ON
 Google Maps Coordinate API		<input type="checkbox"/> OFF
 Google Maps Engine API		<input type="checkbox"/> OFF
 Google Maps Geolocation API		<input checked="" type="checkbox"/> ON
 Google Maps SDK for iOS		<input type="checkbox"/> OFF

➤ Générez votre clé GOOGLE

La clé va nous permettre d'accéder au service de géocodage de GOOGLE. Elle est aussi utile pour une facturation des services selon le degré d'utilisation.



My Project

- Overview
- Services
- Team
- API Access**
- Reports
- Quotas

API Access

To prevent abuse, Google places limits on API requests. Using a valid OAuth token or API key a

Authorized API Access

OAuth 2.0 allows users to share specific data with you (for example, contact lists) while keeping their usernames, passwords, and other information private. A single project may contain up to 20 client IDs. [Learn more](#)

[Create an OAuth 2.0 client ID...](#)

Simple API Access

Use API keys to identify your project when you do not need to access user data. [Learn more](#)

Key for browser apps (with referers)

API key: AIzaSyACP8z708Z-4c4Bk_xmDhG61HLexi0BPWo
Referers: Any referer allowed
Activated on: Jan 11, 2013 1:23 AM
Activated by: lars.vogel@gmail.com – you

[Create new Server key...](#) [Create new Browser key...](#) [Create new Android key...](#)

➤ Copiez cette clé dans un bloc note :

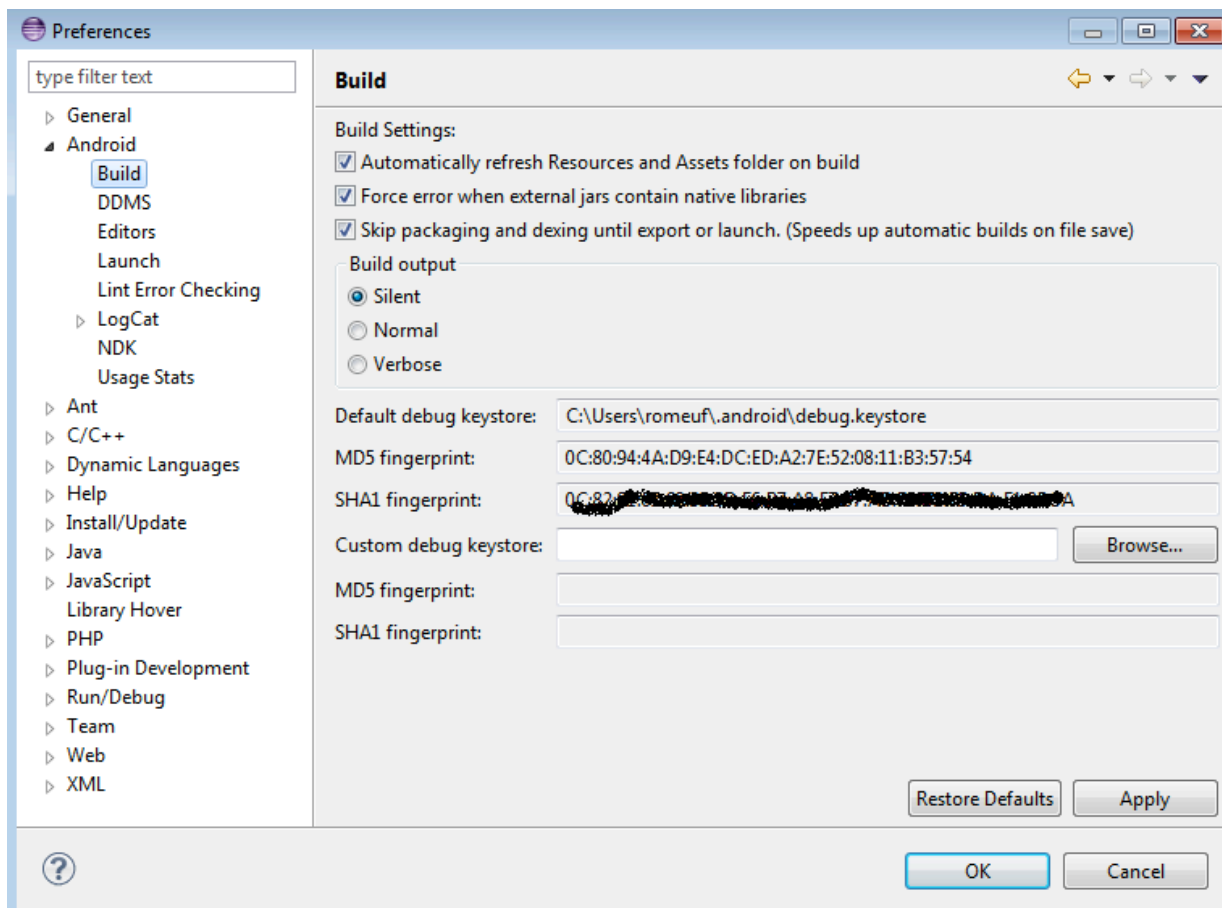
Simple API Access

Use API keys to identify your project when you do not need to access user data. [Learn more](#)

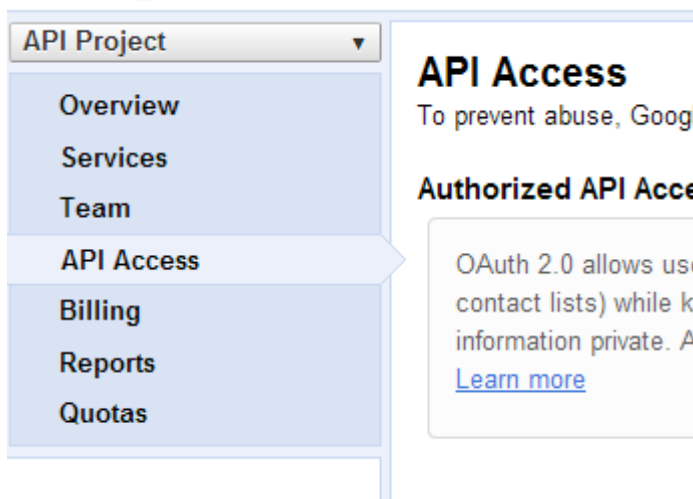
Key for Android apps (with certificates)

API key: AIzaSyACP8z708Z-4c4Bk_xmDhG61HLexi0BPWo
Android apps: Any app allowed
Activated on: Oct 3, 2013 3:49 AM
Activated by: pfromeuf@gmail.com – you

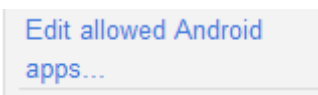
- Récupérez la clé SHA1 de votre Eclipse :



- Autorisez votre application à utiliser la géolocalisation :

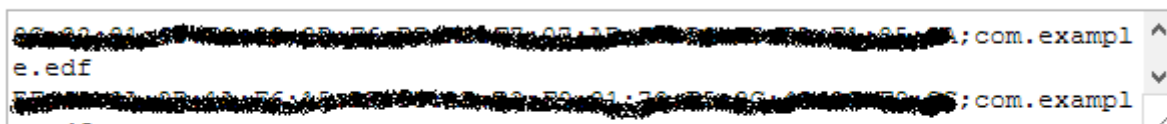


Clic sur (à droite) :



Ajoutez tous les couples Eclipses / applications qui auront le droit d'utiliser la géolocalisation.

Accept requests from an Android application with one of the certificate fingerprints and package names listed below:



One SHA1 certificate fingerprint and package name (separated by a semicolon) per line. Example:
45:B5:E4:6F:36:AD:0A:98:94:B4:02:66:2B:12:17:F2:56:26:A0:E0;com.example



Modification du manifest pour autoriser la géolocalisation

Juste en dessous de

```
<uses-sdk  
    android:minSdkVersion="??"  
    android:targetSdkVersion="??" />
```

- Ajoutez en remplaçant `com.example.edf` par votre package

```
<permission  
    android:name="com.example.edf.permission.MAPS_RECEIVE"  
    android:protectionLevel="signature" />  
  
<uses-permission android:name="com.example.edf.permission.MAPS_RECEIVE" />  
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />  
<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission  
    android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />  
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />  
  
<!-- Required to show current location -->  
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />  
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />  
  
<!-- Required OpenGL ES 2.0. for Maps V2 -->  
<uses-feature  
    android:glEsVersion="0x00020000"  
    android:required="true" />
```

Juste en dessous de

```
<activity  
    android:name="com.example.edf.Geoloc"  
    android:label="@string/title_activity_geoloc"  
    android:parentActivityName="com.example.edf.MainActivity" >  
</activity>
```

- Ajoutez

```
<meta-data  
    android:name="com.google.android.maps.v2.API_KEY"  
    android:value="Votre clé" />  
  
<meta-data  
    android:name="com.google.android.gms.version"  
    android:value="@integer/google_play_services_version" />
```

Layout de Geolocalisation

Il contiendra, de plus, un bouton permettant de rafraichir la carte pour empêcher un rafraichissement automatique à chaque déplacement de 5 mètres de la position du contrôleur.

Il faut rajouter un fragment qui va permettre d'afficher la Google Map (widget GoogleMap)

```
<fragment
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.MapFragment"
    android:apiKey="Votre clé"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

Test

Sur le clic du bouton "Geoloc" de l'Activity ModificationClient appel de l'Activity Geolocalisation, si vos autorisations / clés sont opérationnelles, voici le type d'affichage que vous devriez obtenir :



Vue d'ensemble Android Location API

La plupart des appareils Android permettent de déterminer la géolocalisation actuelle.

Cela peut se faire via 3 fournisseurs d'accès de localisation (*LocationProvider*) :

- un module GPS (*Global Positioning System*) (utilise le récepteur GPS dans le dispositif Android pour déterminer le meilleur emplacement via satellites. Habituellement meilleure précision que réseau).
- une triangulation via relais cellulaires (permet d'économiser l'énergie) ;
- une position donnée par le réseau mobile ou le WiFi (pourrait avoir une plus grande précision dans des locaux fermés que le GPS).

Android.location fournit l'API permettant de déterminer la position géographique actuelle.

La classe *LocationManager* donne accès aux services de géolocalisation d'Android. Ces services permettent d'avoir accès aux fournisseurs d'accès de localisation, de s'inscrire aux *Listener* de modification de la position ou d'alertes de proximité (*geofencing*)...

La classe *LocationProvider* est la superclasse des différents fournisseurs d'accès de localisation.

L'objet *criteria* permet de définir la façon dont le fournisseur doit être sélectionné (souvent le meilleur).

Vous pouvez savoir si un *LocationManager* est activé via la méthode *isProviderEnabled()* (utile pour GPS et WiFi).

La classe *Geocoder* permet de déterminer la géo-coordonnée (longitude, latitude) pour une adresse donnée. La classe *Geocoder* utilise un service en ligne de Google.

Vous pouvez utiliser "*DDMS*" *Perspective* d'Eclipse afin de modifier votre position *Geoloc* pour des tests et l'envoyer à votre STA connecté.

Window → *Open Perspective* → *Other...* → *DDMS*.

Google offre, via Google Play, une bibliothèque pour l'utilisation de Google Maps dans votre application. Le code est basé sur le Google Maps API Android v2 (actuelle v3).

La classe *MapFragment* étend l'interface *Fragment* et permet d'afficher un widget *GoogleMap*. *GoogleMap* est la classe qui représente la carte. Le *MapFragment* possède une méthode *getMap()* pour accéder à la carte.

La classe *LatLng* (latitude, longitude) peut être utilisée pour interagir avec la classe *GoogleView* afin de positionner, par exemple, des marqueurs sur la carte via la classe *Marker*. Cette classe *Marker* peut être hautement personnalisée.

NB : Pour la géolocalisation vous pouvez aussi utiliser la classe *LocationClient* (cf méthode *getLastLocation()*) de `com.google.android.gms.location.LocationClient` de l'api V2 cf

<http://developer.android.com/reference/com/google/android/gms/location/LocationClient.html>

Code de Geolocalisation

- Déclarez les attributs nécessaires.

```
private GoogleMap googleMap;
private LocationManager locationManager;
private String provider, adresseClient;
private LatLng positionClient, positionAgent;
private boolean reussiGeolocalisationAgent = false,
                reussiGeolocalisationClient = false;
private LatLngBounds.Builder builder = new LatLngBounds.Builder();
```
- Créez les 3 méthodes suivantes en vous aidant des annexes :

public void recupPositionAgent() qui récupère la position géolocalisée, et met à jour `reussiGeolocalisationAgent`

public void recupPositionClient() qui détermine une position à partir d'une adresse et met à jour `reussiGeolocalisationClient`

public void afficheCarte() qui affiche une carte en positionnant les marqueurs pour la position géolocalisée et la position du client, puis, en faisant un zoom sur les marqueurs (Centre la carte sur les marqueurs positionnés).

Un test sur le boolean `reussiGeolocalisationClient`, `reussiGeolocalisationClient` est nécessaire pour ajouter les marqueurs.

➤ Modifiez la méthode `onCreate` :

- Récupération de l'identifiant du client via le Bundle et création de l'adresse client (adresse "+","+cp+","+ville+ " France")
- Appel aux méthodes `recupPositionAgent();recupPositionClient();afficheCarte();`
- Sur le clic du bouton, rafraichir l'appel des méthodes nécessaires avec une réinitialisation de la carte via `googleMap.clear();`

ANNEXE : Exemple de code

GÉOLOCALISATION

Classes utiles

- GoogleMap
- LocationManager
- LatLng
- LatLngBounds.Builder

Géolocalisation possible

```
locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
Criteria criteria = new Criteria();
/*
 * Vous pouvez modifier les critères exemple
 * criteria.setPowerRequirement(Criteria.POWER_LOW);
 * criteria.setAccuracy(Criteria.ACCURACY_FINE);
 * criteria.setCostAllowed(true);
 * criteria.setSpeedRequired(false);
 * Puis trouver le meilleur provider en fonction des critères
 */
provider = locationManager.getBestProvider(criteria, false);
if (provider == null || provider.equals("")) {
    //géoloc impossible
}
```

Géofencing possible

```
if (!Geocoder.isPresent()) {
    //géofencing impossible
}
```

Position géolocalisation

```
locationManager.requestLocationUpdates(provider, 20000, 0, this);
Location location = locationManager.getLastKnownLocation(provider);
if (location != null) {
    positionAgent = new LatLng(location.getLatitude(),
        location.getLongitude());
    reussiGeolocalisationAgent = true;
}
else {
    //"Erreur dans la géolocalisation"
}
```

Position géofencing

```
Geocoder fwdGeocoder = new Geocoder(this, Locale.FRANCE);
List<Address> locations = null;
try {
    locations = fwdGeocoder.getFromLocationName(adresseClient,
10);
} catch (IOException e) {
    // "Pbs geocoder adresse client"
}
if ((locations == null) || (locations.isEmpty())) {
    // "Adresse client inconnu !"
} else {
    positionClient = new LatLng(locations.get(0).getLatitude(),
        locations.get(0).getLongitude());
    reussiGeolocalisationClient = true;
}
}
```

Affichage de carte Google map et intégration de marqueur

```
googleMap = ((MapFragment) getFragmentManager().findFragmentById(
    R.id.map)).getMap();
if (reussiGeolocalisationClient) {
    googleMap.addMarker(new MarkerOptions()
        .position(positionClient)
        .title("Client")
        .snippet("Point de rendez vous prochain client")
        .icon(BitmapDescriptorFactory
            .fromResource(R.drawable.grnpushpin)));
    builder.include(positionClient);
}
if (reussiGeolocalisationAgent) {
    googleMap.addMarker(new MarkerOptions().position(positionAgent)
        .title("Ma position"));
    builder.include(positionAgent);
}
if (reussiGeolocalisationClient && reussiGeolocalisationAgent) {
    googleMap.moveCamera(CameraUpdateFactory.newLatLngBounds(builder
        .build(),
        this.getResources().getDisplayMetrics().widthPixels,
this
        .getResources().getDisplayMetrics().heightPixels,
        100));
}
...
...
//NB : si un seul marqueur exemple pas de geocodage ou pas de geolocalisation on
centrera la carte sur un marqueur exemple pas de geocodage
//googleMap.animateCamera(CameraUpdateFactory.newLatLngZoom(
    positionClient, 15));
```

Nb : Le géocodage via l'appel de `new Geocoder` n'est pas très opérationnel sur certaines STA. Si cet appel n'a pas abouti on privilégiera alors (après avoir fait la partie 4) un appel via `thread` à l'API Google via internet en lui passant comme URL :

```
String vad=adresseclient;
try {
    vad=URLEncoder.encode(vad, "UTF-8");
} catch (UnsupportedEncodingException e1) {
}
String vurl="http://maps.google.com/maps/api/geocode/json?address="
+vad+",france&sensor=false";
```

Puis, sur un retour valide de cet appel, l'appel à une méthode qui permet de traiter le `json` renvoyé par cet appel :

```
JSONObject jsonObject = new JSONObject();
try {
    jsonObject = new JSONObject(sb.toString());
} catch (JSONException e) {
    alertMessage("Pbs JSON builder adresse", sb.toString());
    return;
}
Double lon = new Double(0);
Double lat = new Double(0);

try {

    lon = ((JSONArray) jsonObject.get("results")).getJSONObject(0)
        .getJSONObject("geometry").getJSONObject("location")
        .getDouble("lng");

    lat = ((JSONArray) jsonObject.get("results")).getJSONObject(0)
        .getJSONObject("geometry").getJSONObject("location")
        .getDouble("lat");

} catch (JSONException e) {
    alertMessage("Pbs JSON latt lon", jsonObject.toString());
    return;
}
positionClient = new LatLng(lat, lon);
reussiGeolocalisationClient = true;
afficheCarte();
```