

Programmation Android

Travaux pratiques - ENSG

Yann Caron

ENSG
Géomatique

Sommaire - TP

- Cahier des charges
- Création du projet et IHM
- Google Map Activity
- Spatialite
- Web services
- Boni
- Pour aller plus loin !



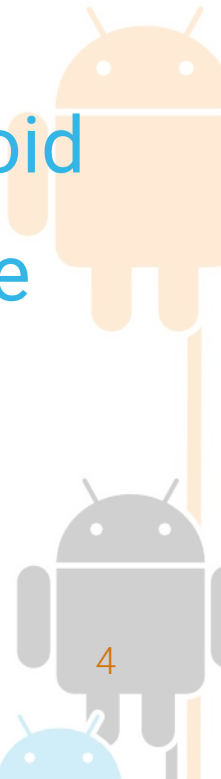
IN01 – TP

Cahier des charges



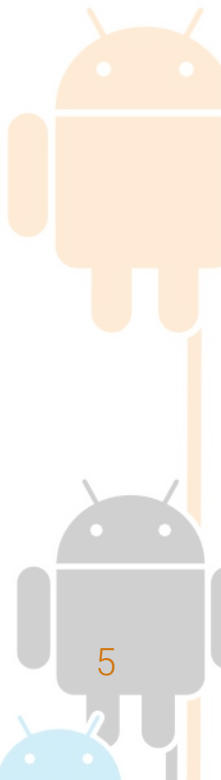
Contexte

- Vous êtes en délégation pour l'Office National des forêts
- On vous commande une application pour gérer les gardes forestiers et leurs “circonscriptions territoriales”
- Les gardes possèdent tous un smartphone Android
- Qui pourra se synchroniser à l'application centrale via Wifi



Cahier des charges

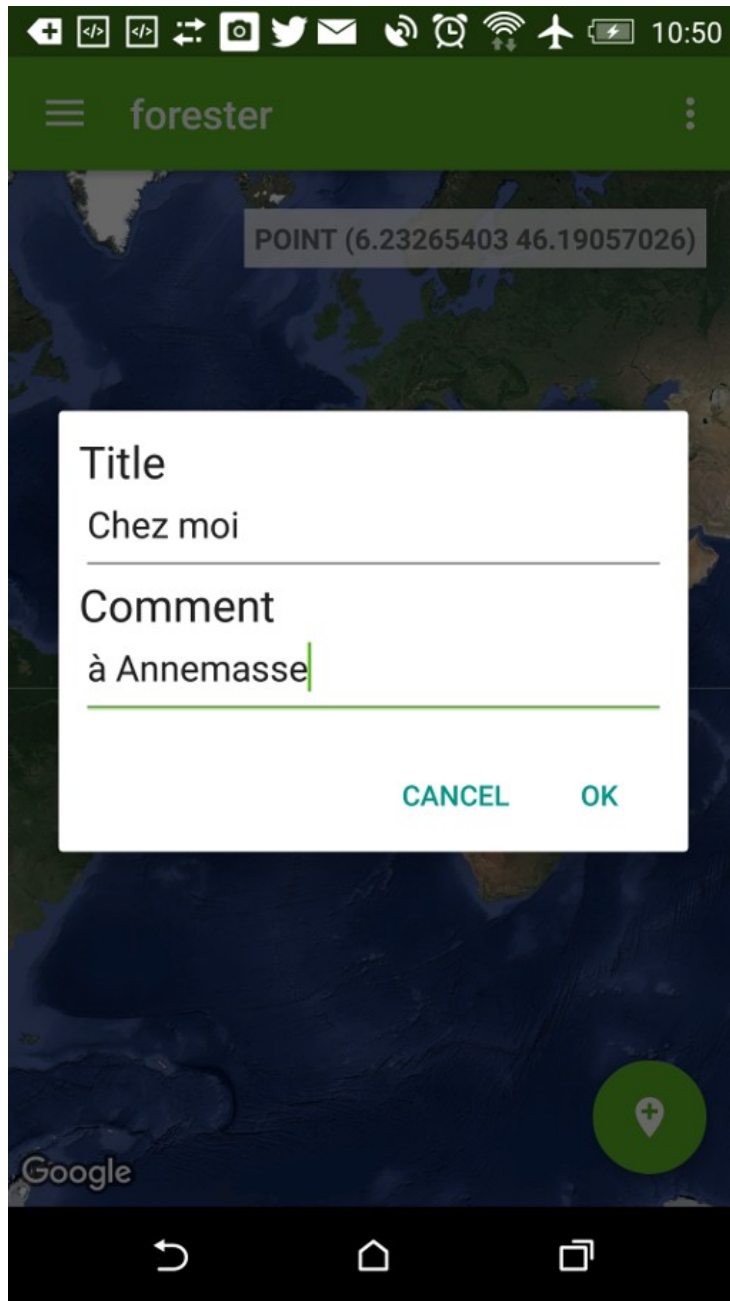
- Nous nous concentrerons sur l'application embarquée Android qui devra :
 - Afficher la carte en vue satellite
 - Donner accès aux coordonnées GPS de l'appareil
 - Donner la possibilité de stocker des points d'intérêts (libellé de leur adresse éventuelle)
 - Donner la possibilité de stocker des procès verbaux (optionel)
 - Délimiter les circonscriptions
 - Consulter les données stokées
 - Afficher les données météorologiques (optionel)
 - Synchroniser les données sur le serveur central (réflexion)



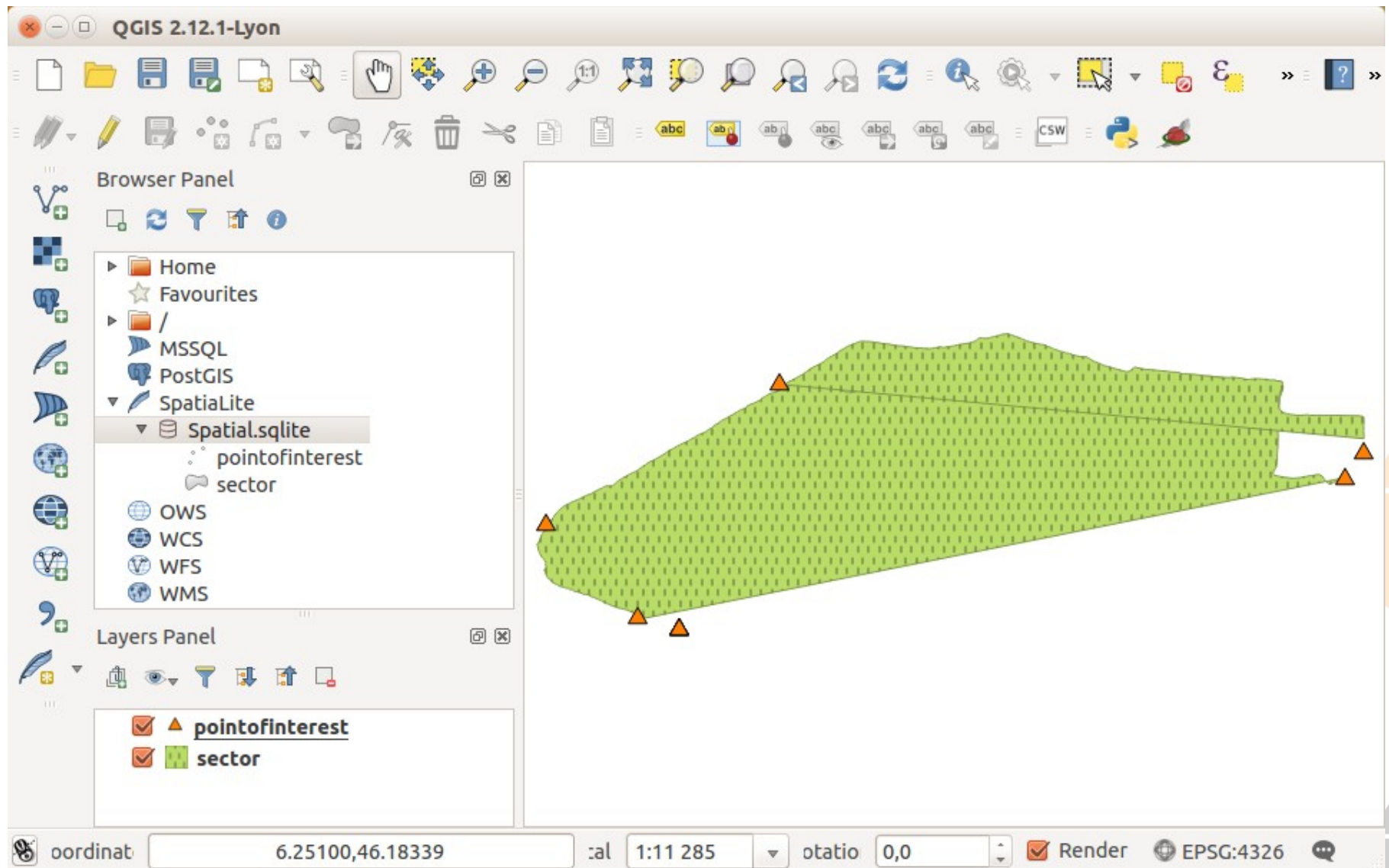
En image / demo

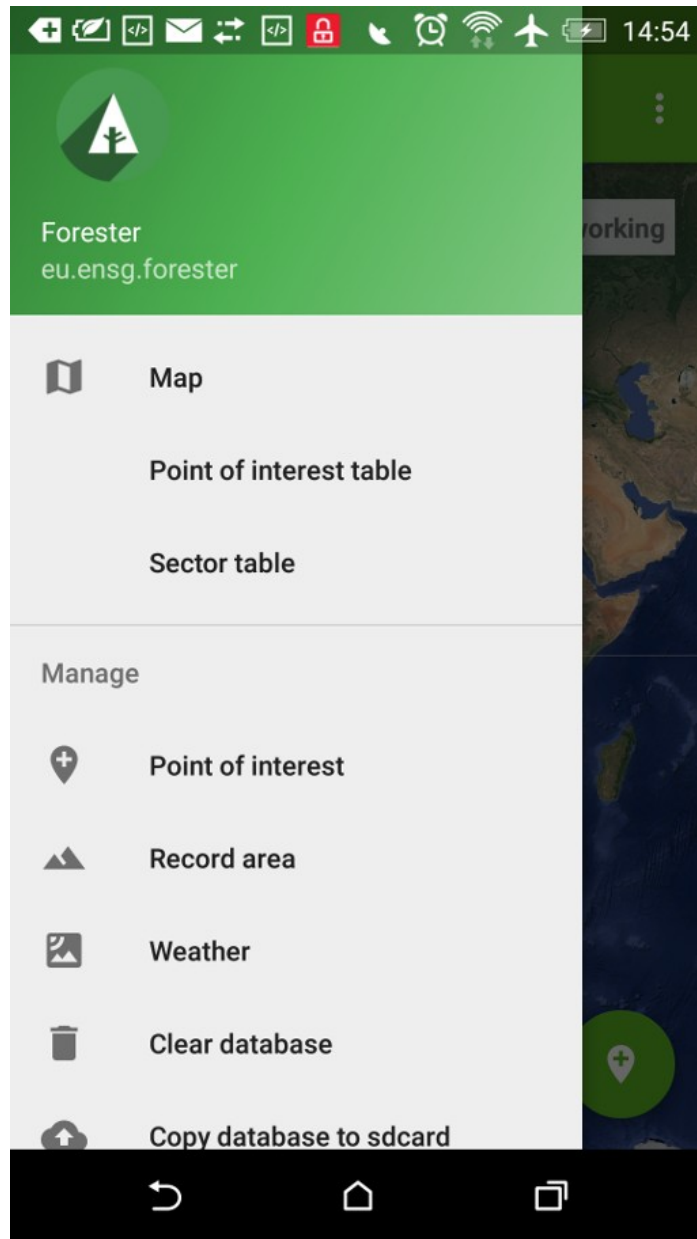


En image / demo



Spatialite vers QGIS

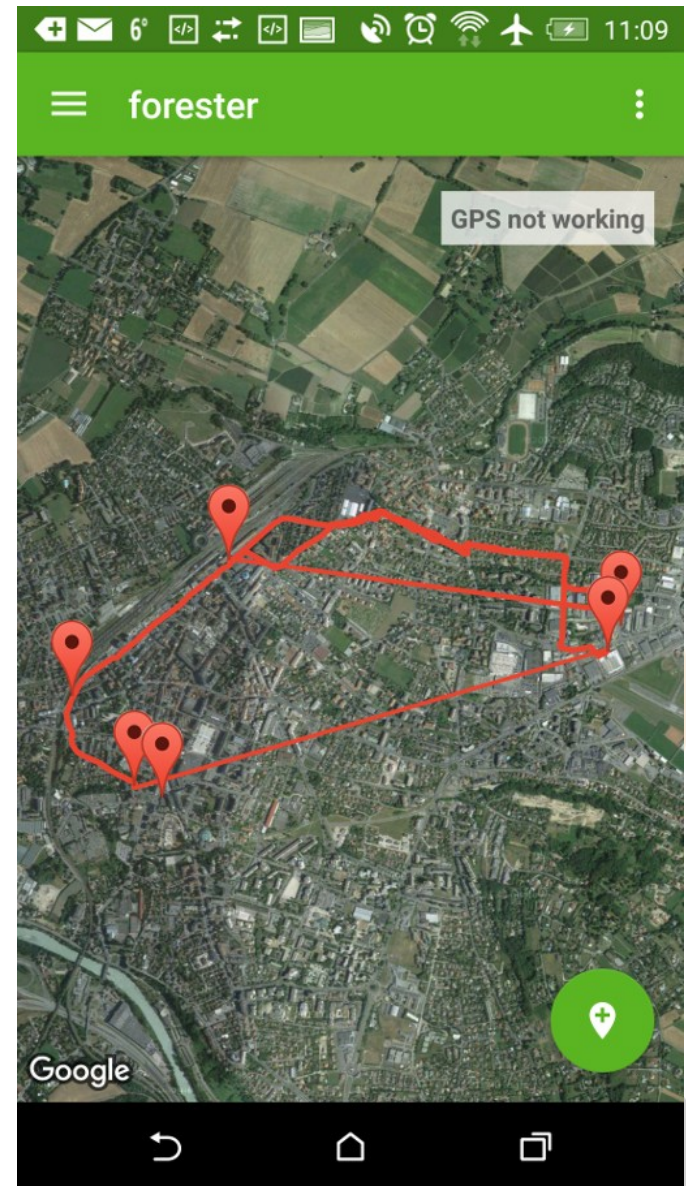




session fev 2016

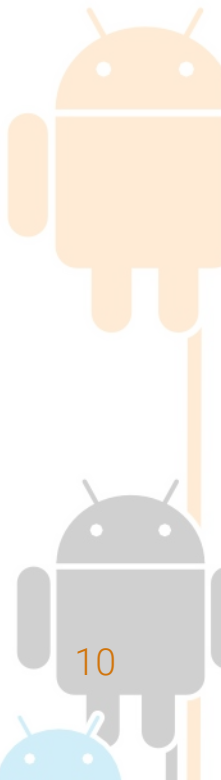
Yann Caron (c) 2014

En image



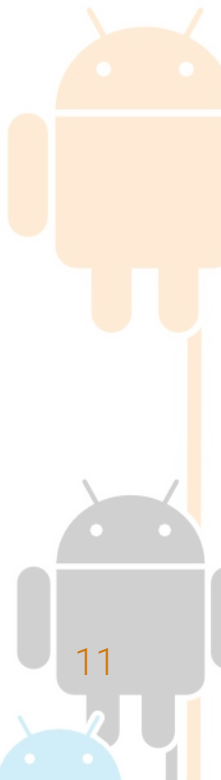
Les outils en place

- Spatialite : Spatialite-Database-Driver
- `eu.ensg.spatialite`, outils pour faciliter l'utilisation de spatialite
 - ➔ Deux composants principaux
 - SpatialiteOpenHelper
 - Objets géométriques



Les outils - bis

- `eu.ensg.common`
- Des outils divers comme `FileSystem` et `enums.FailSafeValueOf`
- Une bonne pratique :
 - Réutilisabilité
 - Subdiviser en bibliothèques logiciel
 - Diviser pour régner
 - TDD : Test driven development



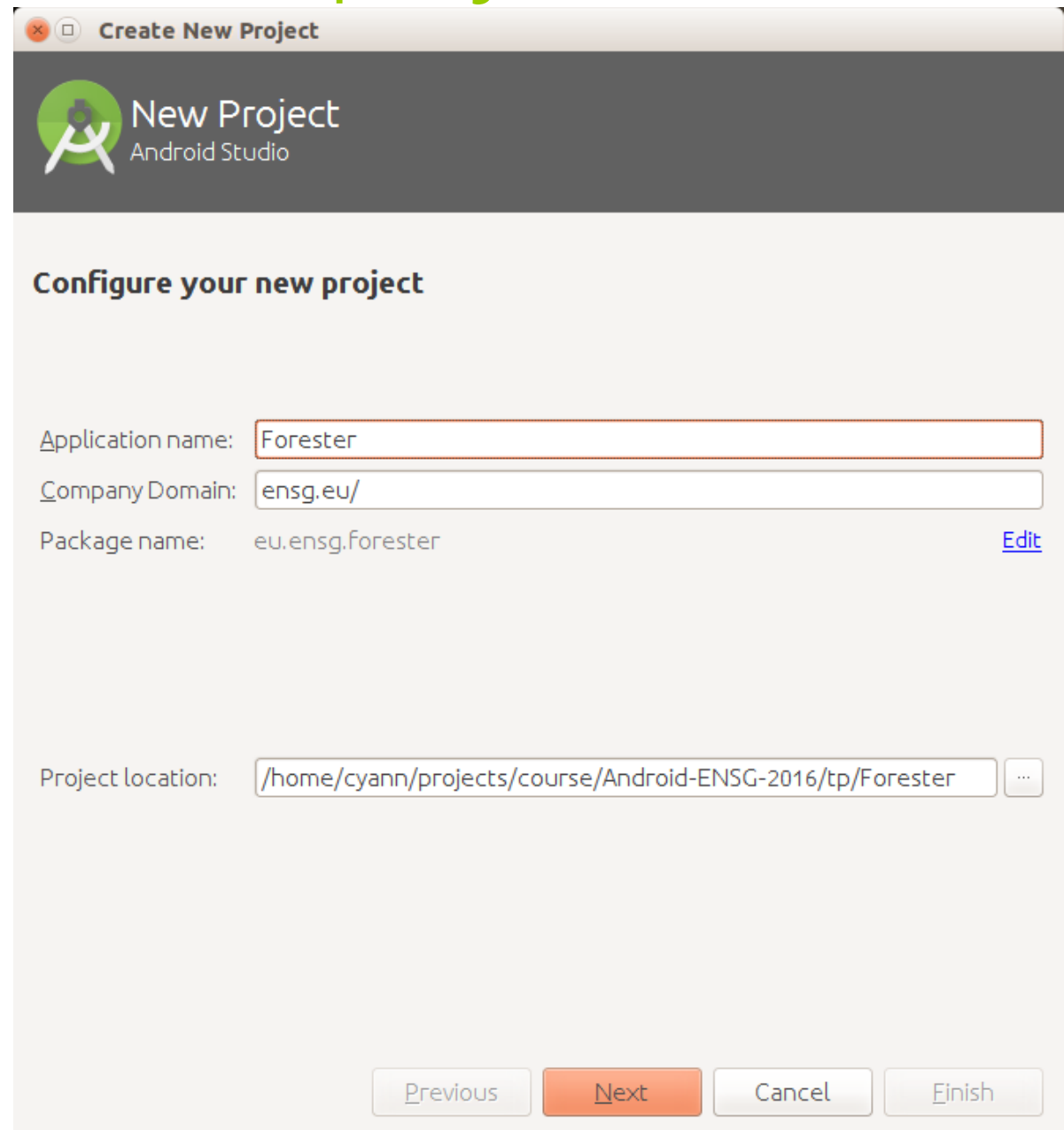
IN01 – TP

Création du projet et IHM



Création du projet

- Choisissons son nom
- Son package
- Par convention
eu.ensg.forester



Create New Project

New Project
Android Studio

Configure your new project

Application name:

Company Domain:

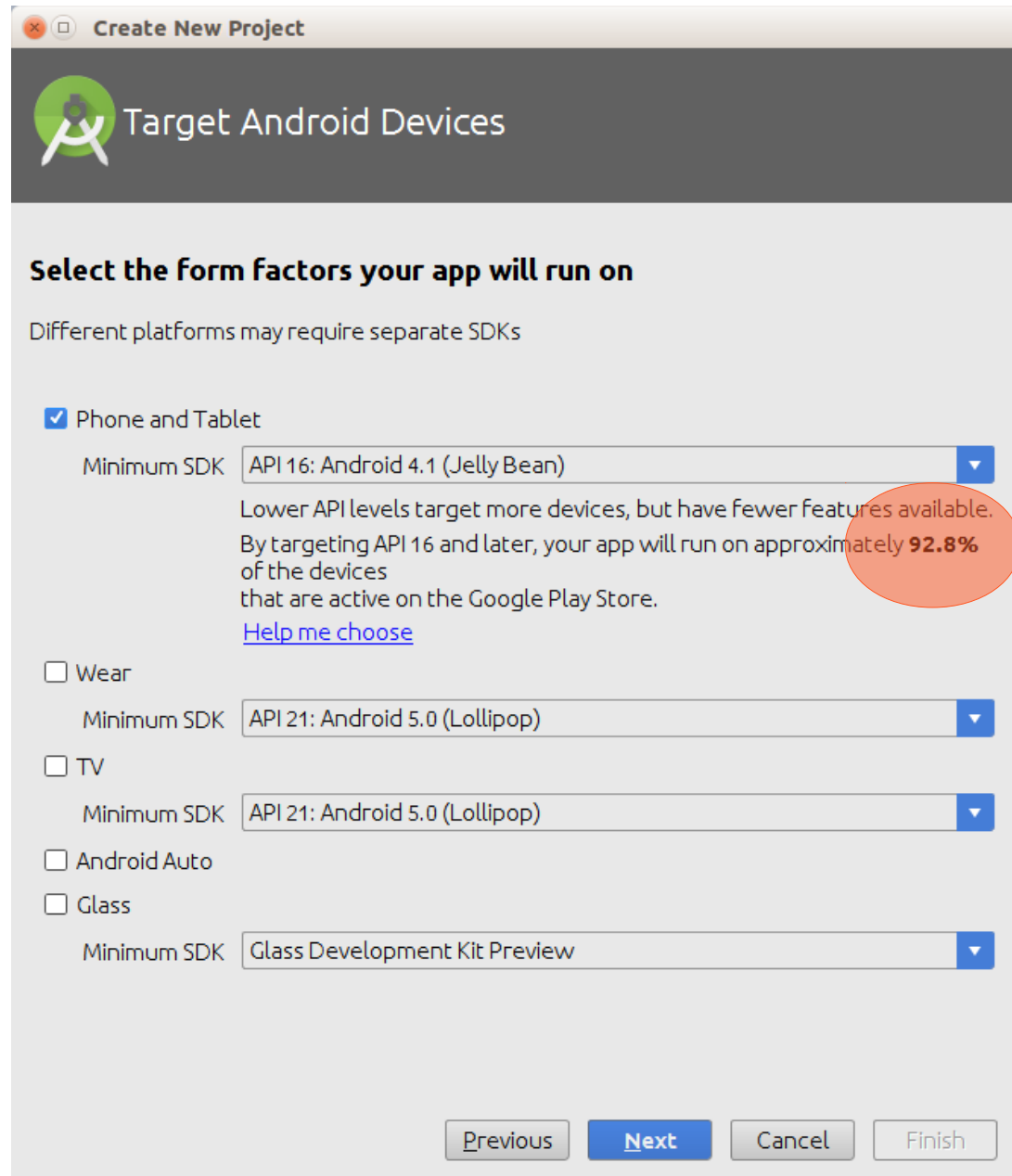
Package name: [Edit](#)

Project location: ...

[Previous](#) [Next](#) [Cancel](#) [Finish](#)

Création du projet

- Choisissons l'API



Create New Project

Target Android Devices

Select the form factors your app will run on

Different platforms may require separate SDKs

☒ Phone and Tablet

Minimum SDK: API 16: Android 4.1 (Jelly Bean)

Lower API levels target more devices, but have fewer features available. By targeting API 16 and later, your app will run on approximately **92.8%** of the devices that are active on the Google Play Store. [Help me choose](#)

☐ Wear

Minimum SDK: API 21: Android 5.0 (Lollipop)

☐ TV

Minimum SDK: API 21: Android 5.0 (Lollipop)

☐ Android Auto

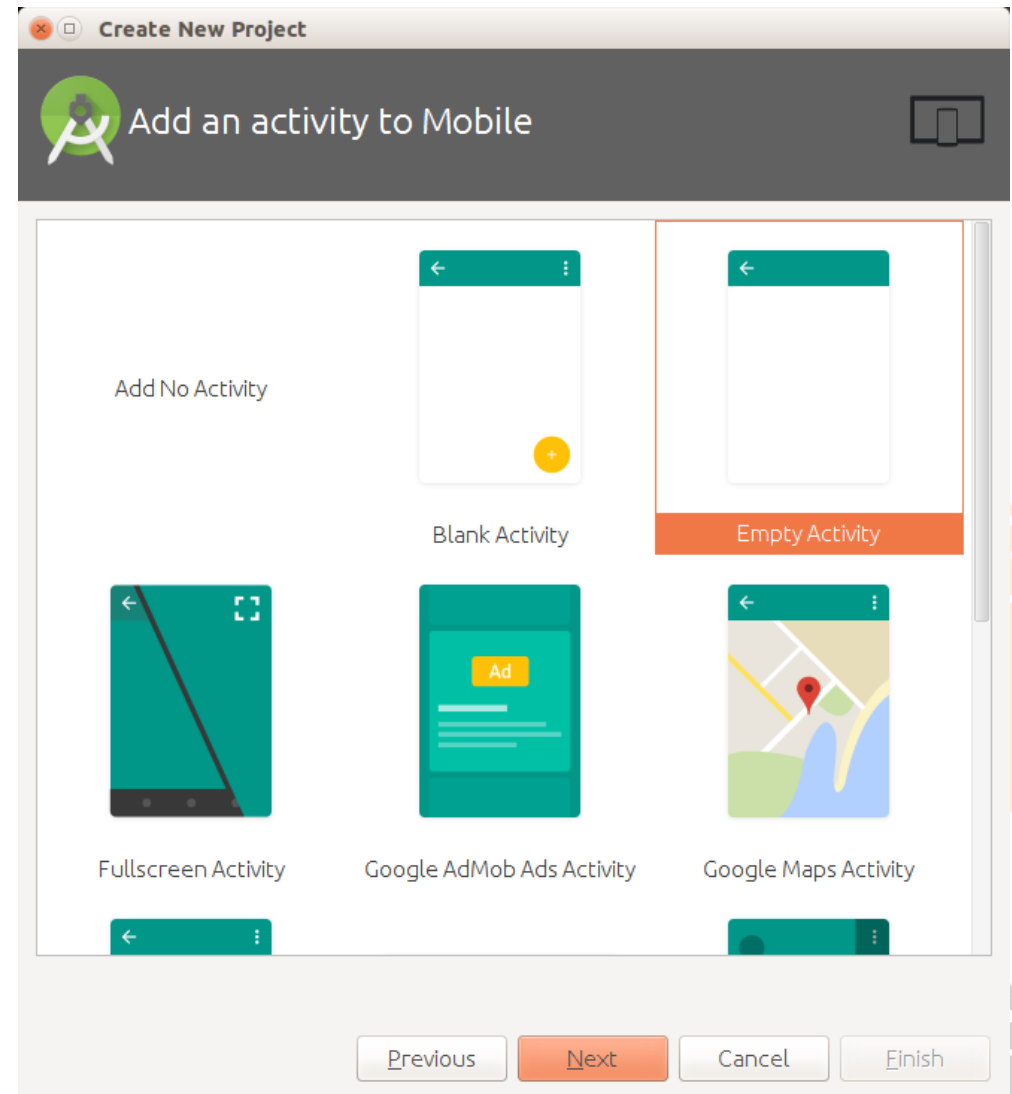
☐ Glass

Minimum SDK: Glass Development Kit Preview

[Previous](#) [Next](#) [Cancel](#) [Finish](#)

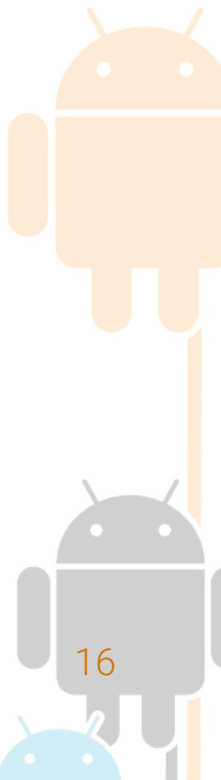
Création du projet

- Choisissons le type de projet
- Empty Activity
- Activity Name : LoginActivity
- Layout Name : activity_login

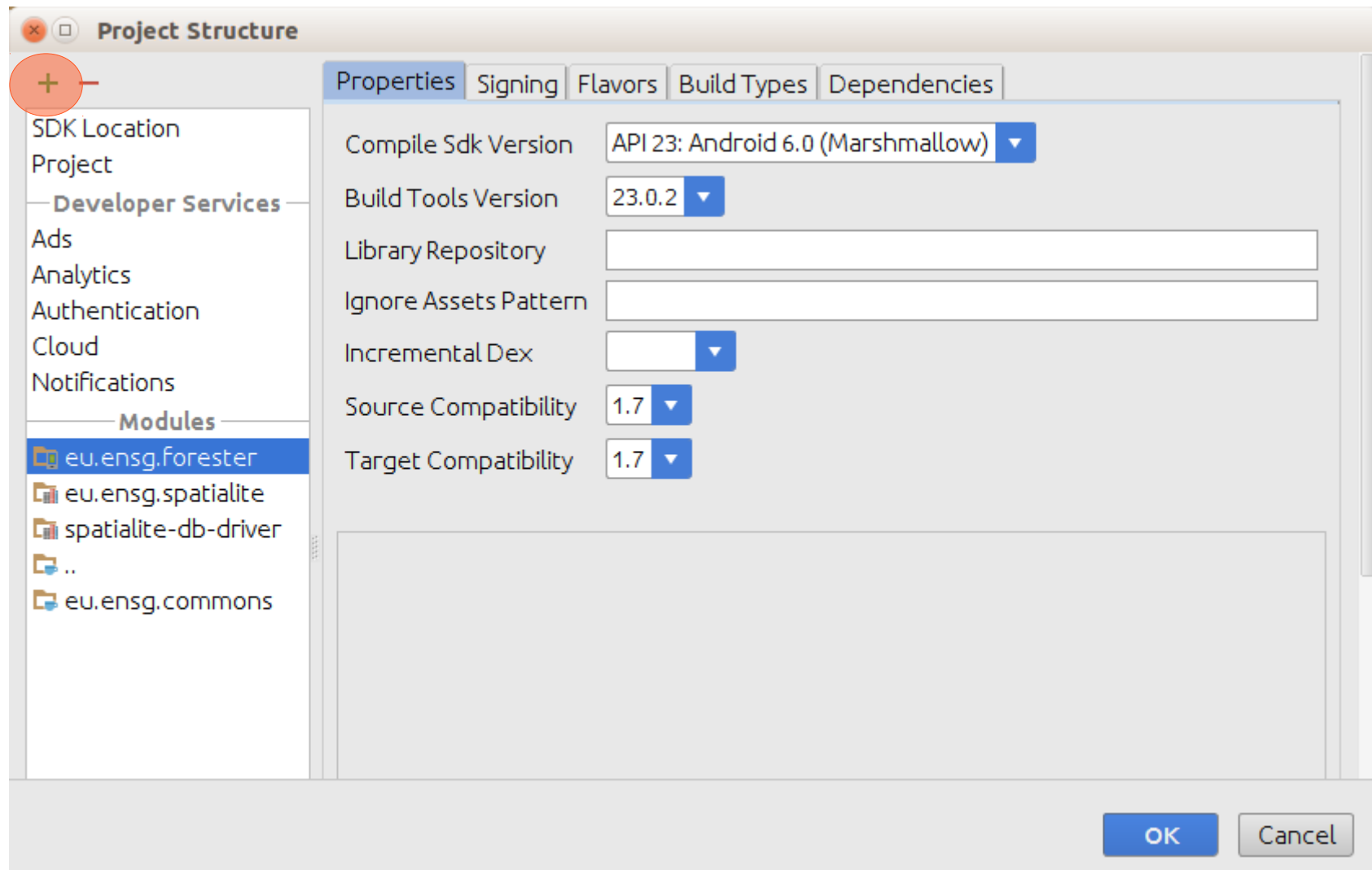


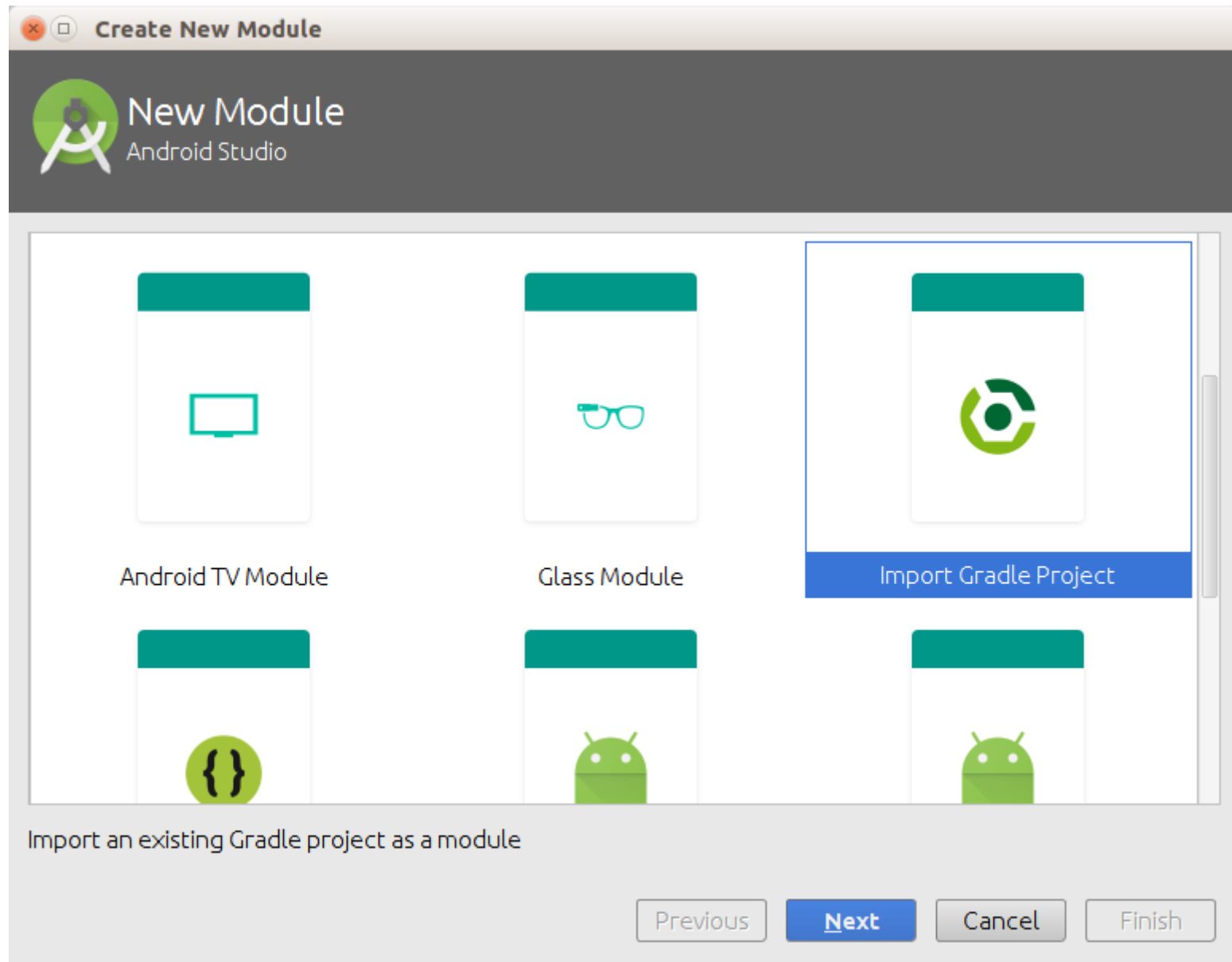
Importer / créer une bibliothèque

- Bonne pratique, séparer les responsabilités en découpant par librairie
- Principe : Diviser pour régner
- Facilite les tests unitaires !!!! TDD
- IntelliJ prend en charge la gestion des dépendances Gradle



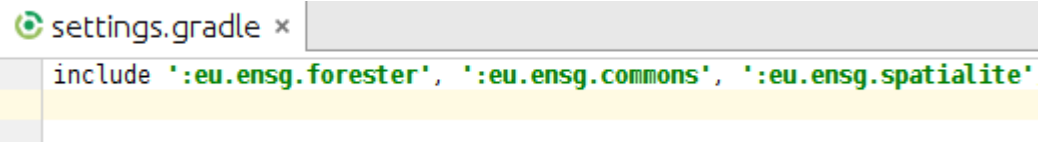
Ajouter une bibliothèques





Dans les fichiers

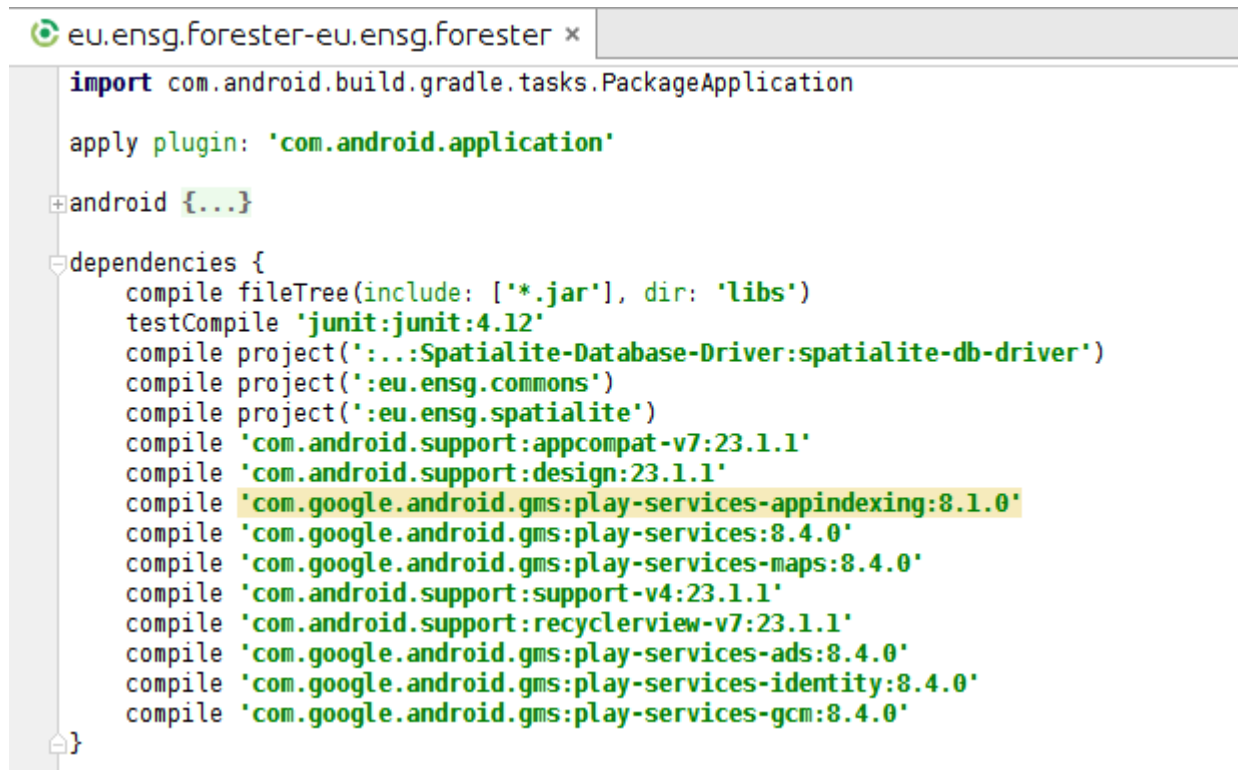
- settings.gradle



settings.gradle x

```
include ':eu.ensg.forester', ':eu.ensg.common', ':eu.ensg.spatialite'
```

- build.gradle
du projet



eu.ensg.forester-eu.ensg.forester x

```
import com.android.build.gradle.tasks.PackageApplication

apply plugin: 'com.android.application'

android { ... }

dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    testCompile 'junit:junit:4.12'
    compile project('...:Spatialite-Database-Driver:spatialite-db-driver')
    compile project(':eu.ensg.common')
    compile project(':eu.ensg.spatialite')
    compile 'com.android.support:appcompat-v7:23.1.1'
    compile 'com.android.support:design:23.1.1'
    compile 'com.google.android.gms:play-services-appindexing:8.1.0'
    compile 'com.google.android.gms:play-services:8.4.0'
    compile 'com.google.android.gms:play-services-maps:8.4.0'
    compile 'com.android.support:support-v4:23.1.1'
    compile 'com.android.support:recyclerview-v7:23.1.1'
    compile 'com.google.android.gms:play-services-ads:8.4.0'
    compile 'com.google.android.gms:play-services-identity:8.4.0'
    compile 'com.google.android.gms:play-services-gcm:8.4.0'
}
```

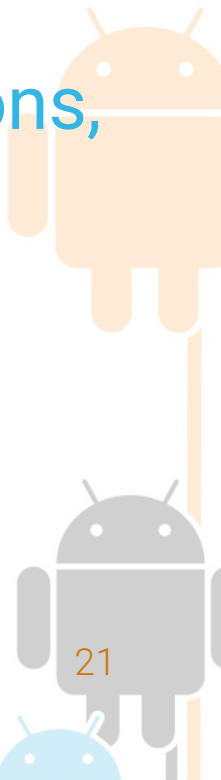
Canevas

- J'ai préparé un Caneva de TP pour vous !
- Les dépendances sont gérées
- Particulièrement celle pour Spatialite-Database-Driver qui est spéciale
 - ➔ Gérer les librairies JNI (downgrade de gradle)
 - ➔ Pour plus d'informations :
<https://github.com/kristinahager/Spatialite-Database-Driver>
- Copier le projet Forester.tp et le renommer en Forester



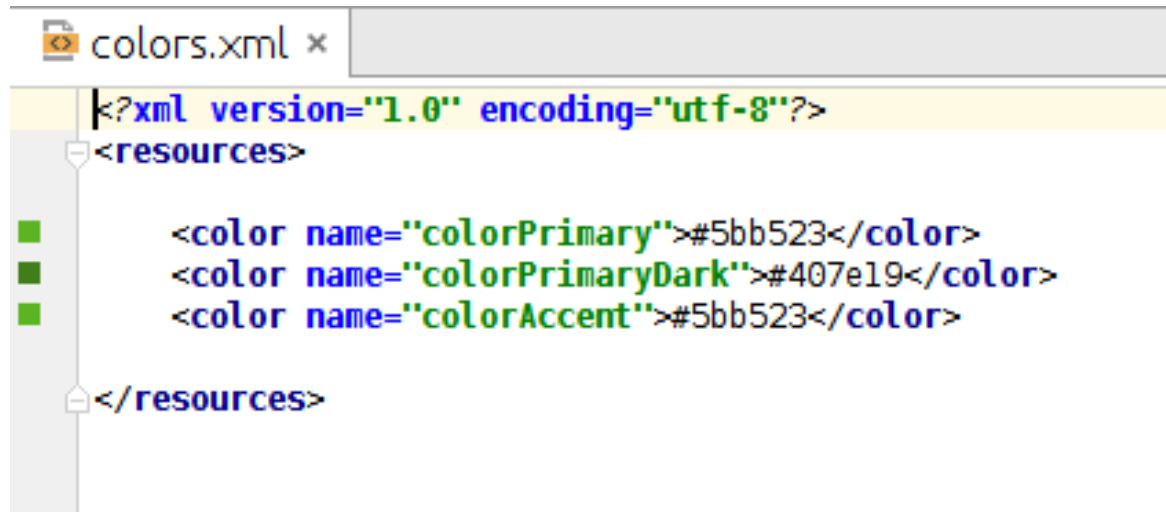
Personnalisation

- Rappel :
 - ➔ /res/mipmap et /res/drawable pour les images
 - ➔ /res/layout pour les IHM
 - ➔ /res/menu pour les menus
 - ➔ /res/values pour les couleurs, texts, dimensions, styles, etc....
- Le tout sous-catégorisé par résolution, compatibilité, etc....



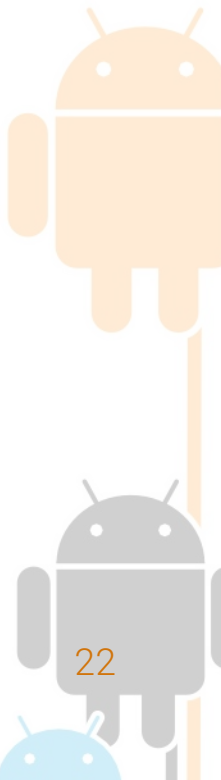
Couleurs

- Changer les couleurs dans
res/values/colors.xml

A screenshot of an IDE window titled 'colors.xml x'. The code is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#5bb523</color>
    <color name="colorPrimaryDark">#407e19</color>
    <color name="colorAccent">#5bb523</color>
</resources>
```

The code is color-coded: XML tags are blue, attribute names are green, and values are black. The first line is highlighted in yellow. On the left, there is a vertical toolbar with icons for file operations and a list of three green squares.



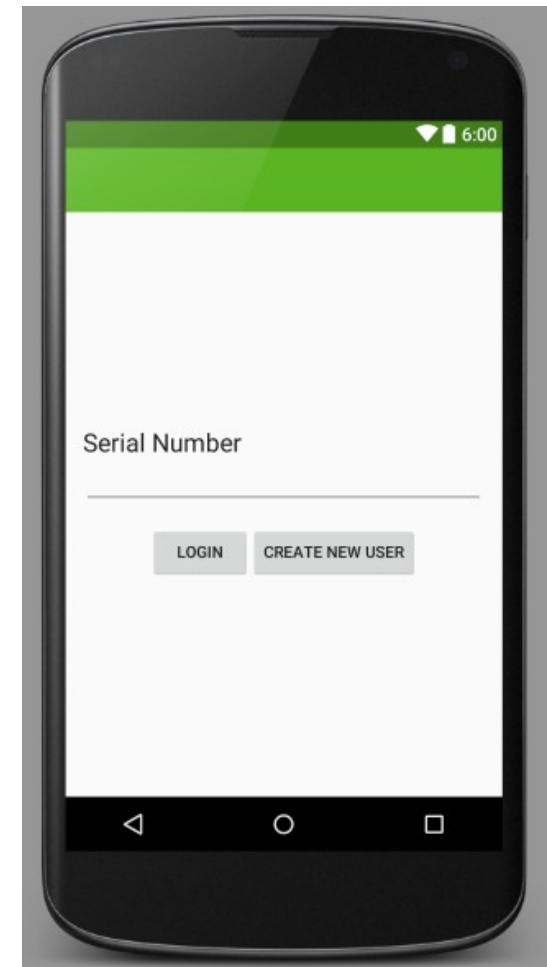
Changer l'Icon

- Une site d'icon : <https://www.iconfinder.com/>
- Que l'on place dans :
 - ➔ `forester/app/src/main/res/mipmap`
 - ➔ Notons les différentes résolutions



IHM Login

- Éditions le fichier `res/layout/activity_login.xml`
- `LinearLayout`, orientation vertical
- Une text view “Serial number”
- Texte large
- Texte dans String
- Un Edit text (`@+id=serial`)
- Un autre `LinearLayout`
 - `Gravity=Center`
 - `PaddingTop`
- Deux boutons



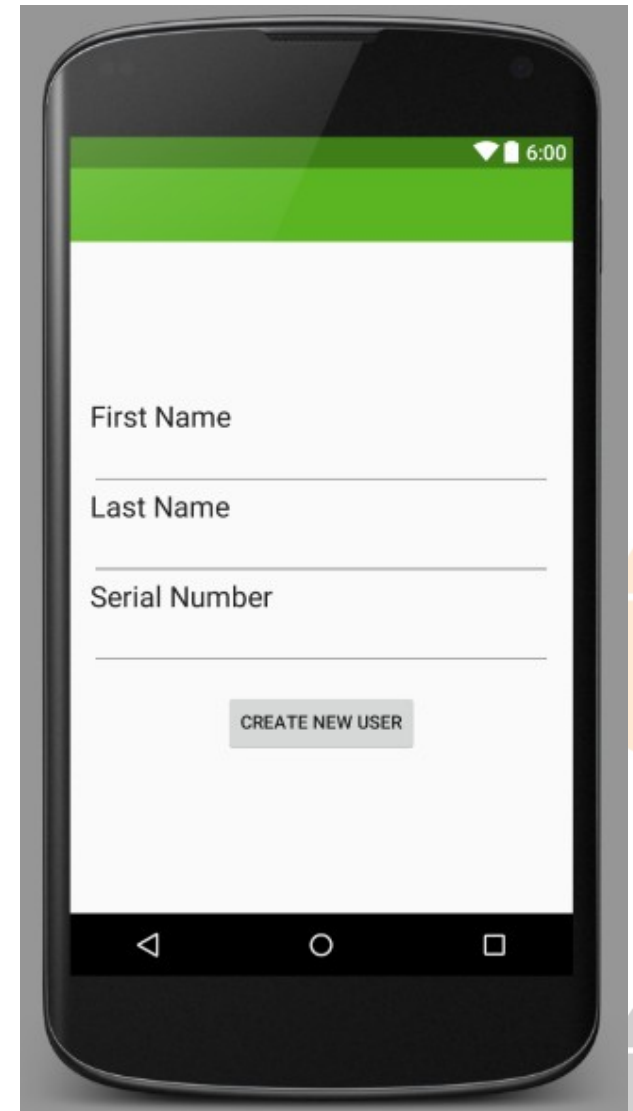
IHM gestion des événements

- Dans la classe LoginActivity
- Notons AppCompatActivity
- Récupérons les instances des composants
 - ➔ `findViewById(R.id.[ID])`
- Gérons les événements
 - ➔ `private void login_onClick(View view)`
 - ➔ `private void create_onClick(View view)`



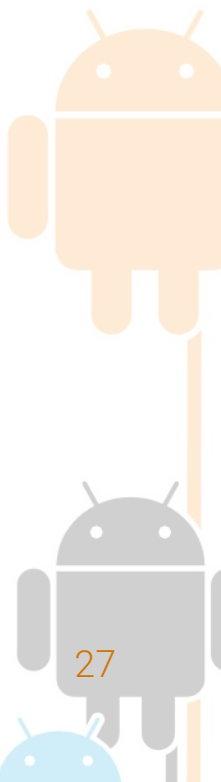
IHM Create New User

- On crée une nouvelle Activity “CreateUser”
- Créer un nouvel utilisateur
- Un wizard Android studio
- On gère les événements



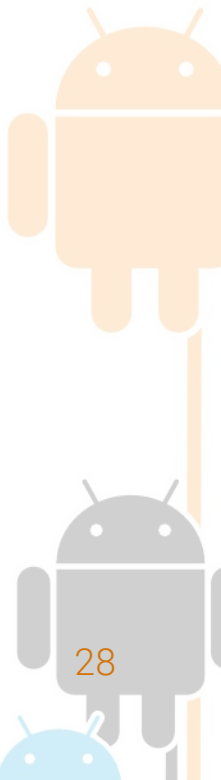
Navigation

- Depuis LoginActivity :
 - ➔ Quand on s'authentifie on appelle une MapActivity (plus tard)
 - Appel à la base de données (plus tard)
 - Quand on touche “create new user” on appelle CreateUserActivity
- Depuis CreateUserActivity :
 - ➔ Quand on touche “create new user” :
 - on crée le user dans la base de données (plus tard)
 - on appelle de nouveau LoginActivity avec le numéro de série comme paramètre



Optionnel

- On peut vérifier que les données soient bien complétées
- Notifier le cas échéant
- Surligner en rouge (ajouter une couleur)
- Retenir le login (SharedPreferences)
- Qu'on peut voir avec l'ADB



Persister des données

■ Sauver

```
SharedPreferences preferences =  
    getSharedPreferences("MyPreferenceFile", Context.MODE_PRIVATE);  
  
SharedPreferences.Editor editor = preferences.edit();  
editor.putString("MyKey", value);  
editor.commit();  
editor.apply();
```

■ Récupérer

```
SharedPreferences preferences =  
    getSharedPreferences("MyPreferenceFile", Context.MODE_PRIVATE);  
String value = preferences.getString("MyKey", "DefaultValue");
```

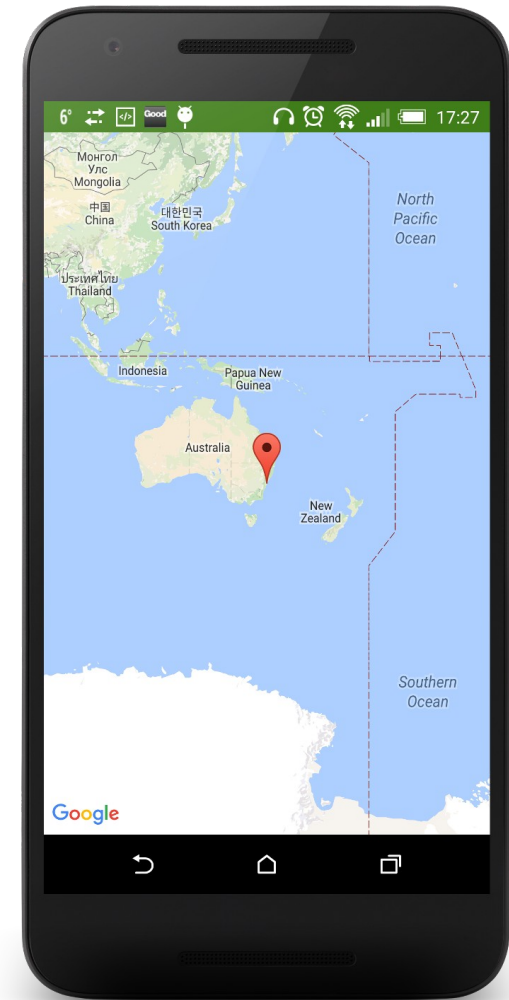
IN01 – TP

Google Map Activity



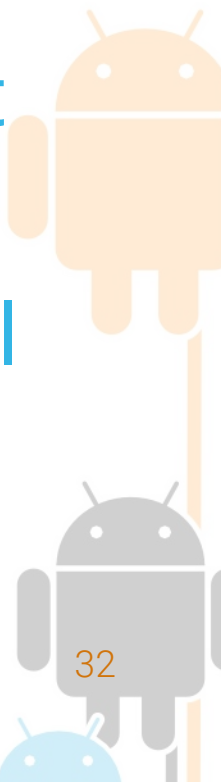
Créer une MapActivity

- Comme décrit dans le cours
- Créer la mapactivity
- Ajouter les dépendances
- Obtenir une clé
- Vérifier le manifest



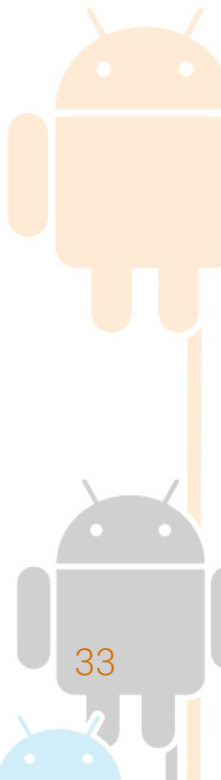
Gérer le chargement de la carte

- Le processus de chargement de la carte est asynchrone
- Android Studio génère un code qui gère le callback à travers la méthode `getMapAsync`
- Attention à l'attribut `mMap`, qui sera nul avant cela (null pointer exception / Asynchrone)
- C'est donc dans le callback `onMapReady` qu'il faudra commencer à utiliser la carte.



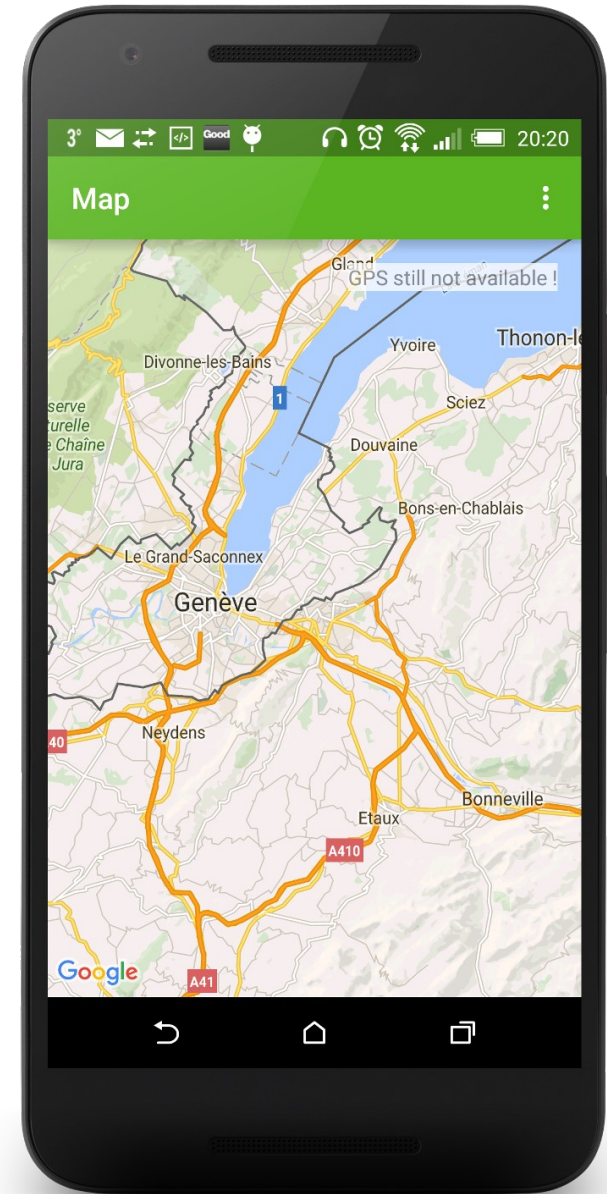
Gérer le GPS

- Au chargement de l'activity :
 - ➔ Créer un attribut :
 - `eu.ensg.spatialite.geom.Point currentPosition`
 - ➔ Au chargement, une position par défaut (ou la dernière chargée)
 - ➔ Gérer le changement de coordonnées GPS
 - ➔ Stocker les données GPS dans `currentPosition`
 - ➔ La stocker dans une `SharedPreferences`
 - ➔ Zoomer la carte sur cette position



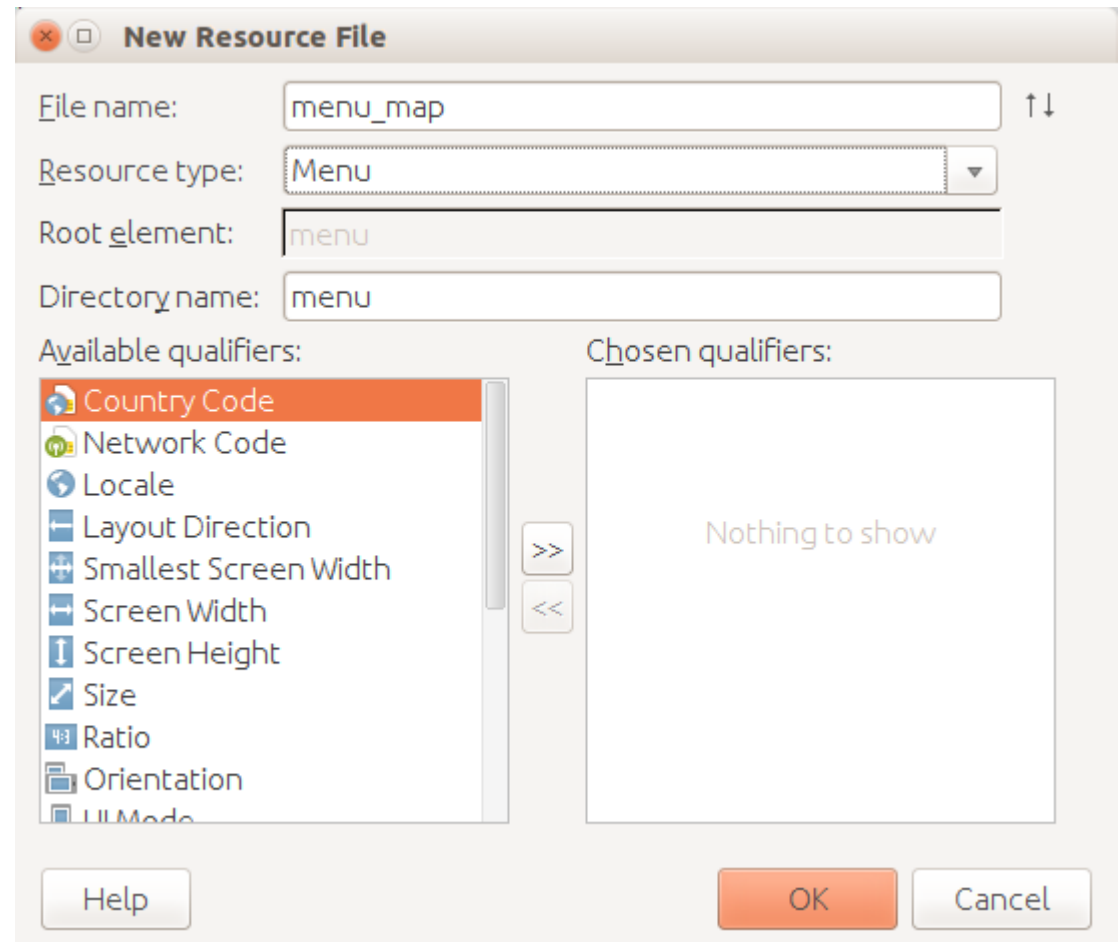
Ajouter un label

- En-capsuler le fragment `googleMap` dans un `FrameLayout`
- Ajouter un label `lblPosition`
- Au changement de position, mettre à jour le label



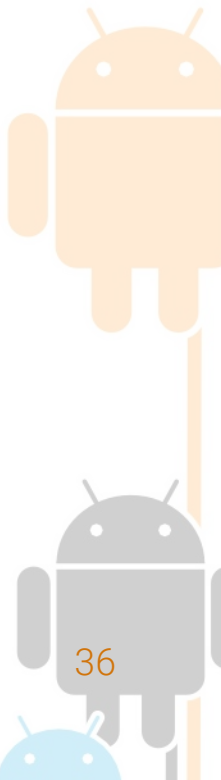
Le menu

- Créer un menu pour ajouter des actions à l'activity GoogleMap
- Attention ! `mapActivity` doit hériter de `AppCompatActivity`



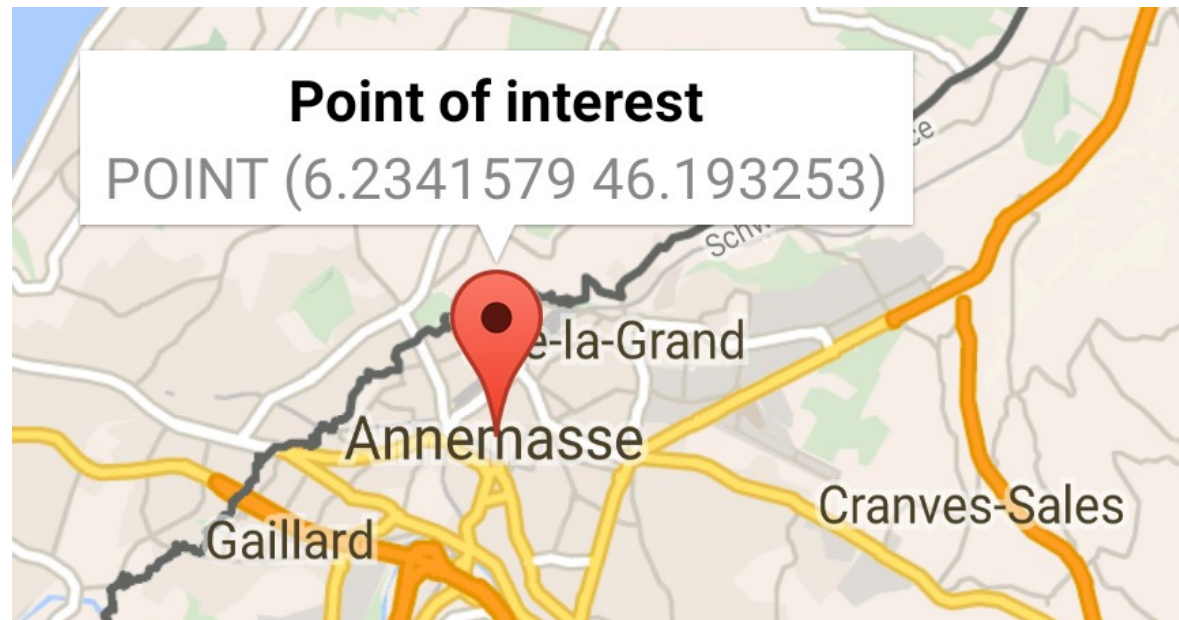
Le menu

- Seulement deux entrées dans le menu pour le moment :
 - ➔ Add point of interest
 - ➔ Add sector
- Le menu est rattaché à l'activity



Evénements

- Gérer les événements pour créer les points d'intérêts et les secteurs
- Point d'intérêt doit ajouter un marqueur sur la map

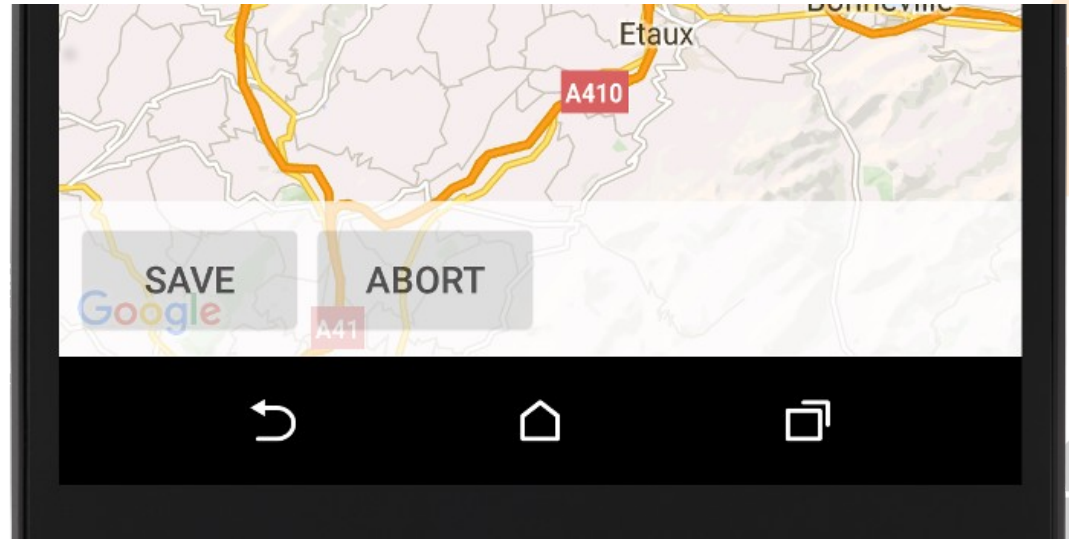


Secteur

- Créer un attribut :
 - ➔ `eu.ensg.spatialite.geom.Polygon currentDistrict`
- Et un attribut : `bool isRecording = false`
- Lorsque l'utilisateur touche le menu “add district”, l'attribut est passé à `true`
- Et le `currentDistrict` est instancié avec un nouvel objet
- Et lorsque celui-ci est à `true`, les coordonnées GPS sont ajoutées à l'attribut `currentDistrict`

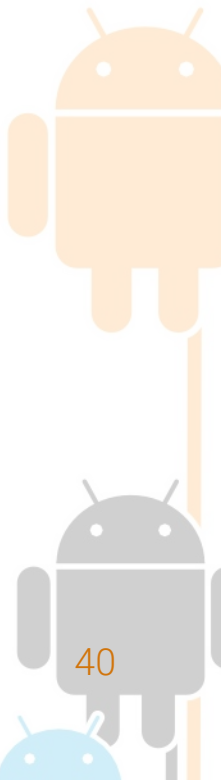
Secteur

- Il faut créer un layout (avec un id) par-dessus la carte
- Qui contient des boutons (save, abort)
- Visible uniquement en mode “recording”
- Puis gérer les évènements de ces boutons



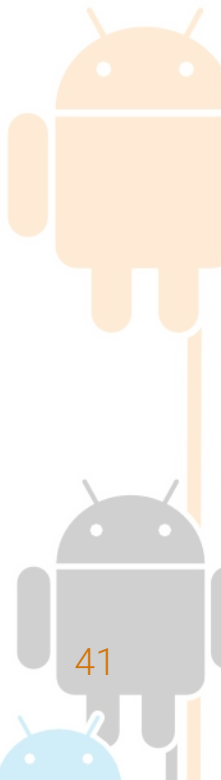
Pour finaliser

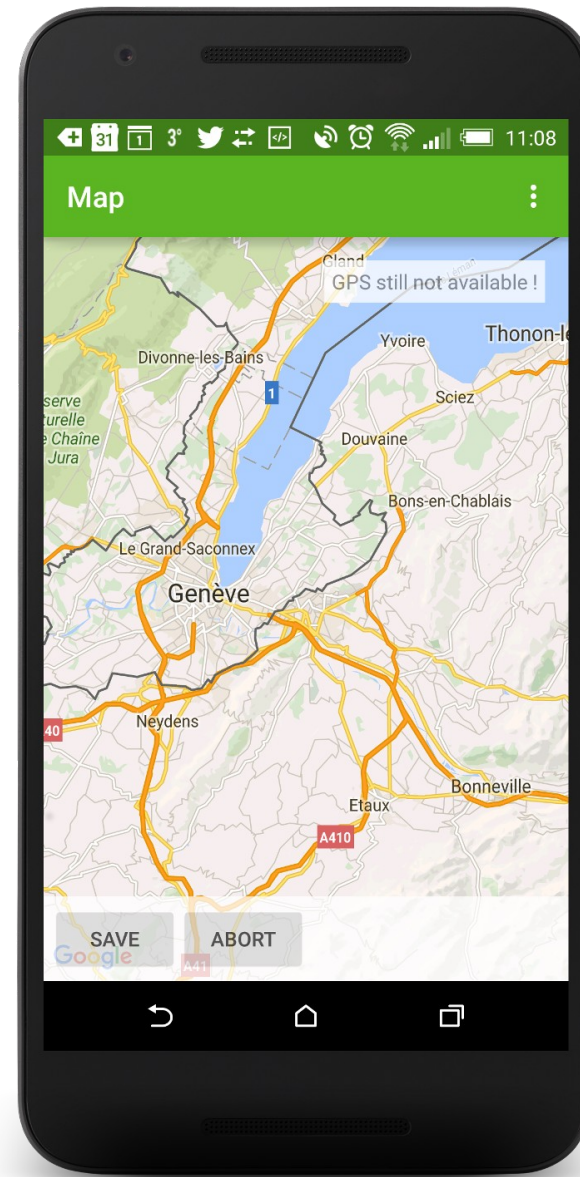
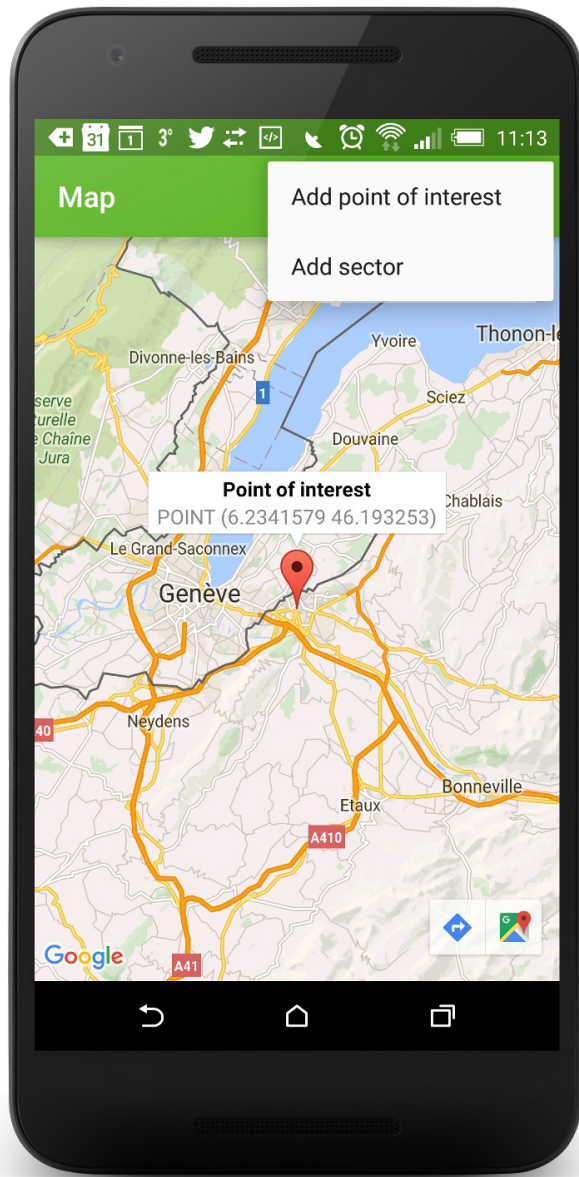
- Dessiner le secteur au fut et à mesure
 - ➔ Il faut convertir le Polygon en PolygonOptions
 - ➔ Il faut gérer l'animation (remove and draw)
- Passer en vue satellite ? Ou hybride ?
- Gérer le GPS avec onPause et onResume
 - ➔ `GPSTils.removeUpdates`
 - ➔ `GPSTils.requestLocationUpdates`



Créer une fenêtre d'info personnalisée (optionnel)

- Créer un layout dans le répertoire `res/layout`
- Au chargement de `googleMap`, assigner l'info window avec la méthode :
`googleMap.setInfoWindowAdapter`
- Instancier un `InfoWindowAdapter` et y surcharger la méthode `getInfoContents`





IN01 – TP

Spatialite

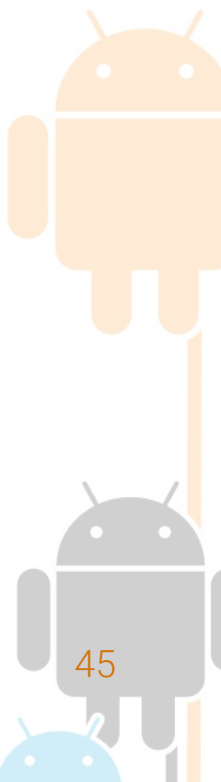


Remarques

- ORMLite n'est pas compatible avec spatialite (utilisation de asText())
- On va écrire un accès aux données simple (requête / affichage)
- Inconvénient : mauvaise pratique, code SQL un peu partout
- Si possible, refactoring en fin de projet

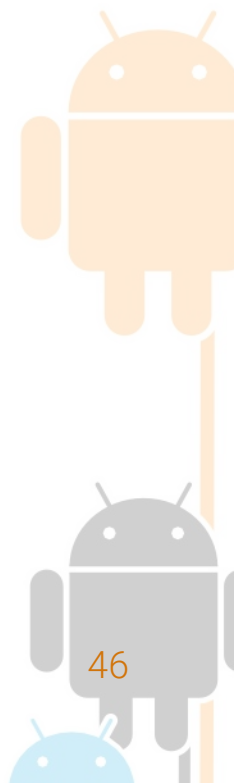
Initialiser la base de donnée

- Créer une classe qui hérite de `SpatialOpenHelper`
 - ➔ `ForesterSpatialiteOpenHelper`
 - ➔ Bonne pratique quand on spécialise une classe [nom][nom superclass]
- Même comportement que `SqliteOpenHelper`
- Gérer le nom et la version dans le constructeur
 - ➔ `DB_FILE_NAME = "Forester.sqlite"`
 - ➔ `VERSION = 1`



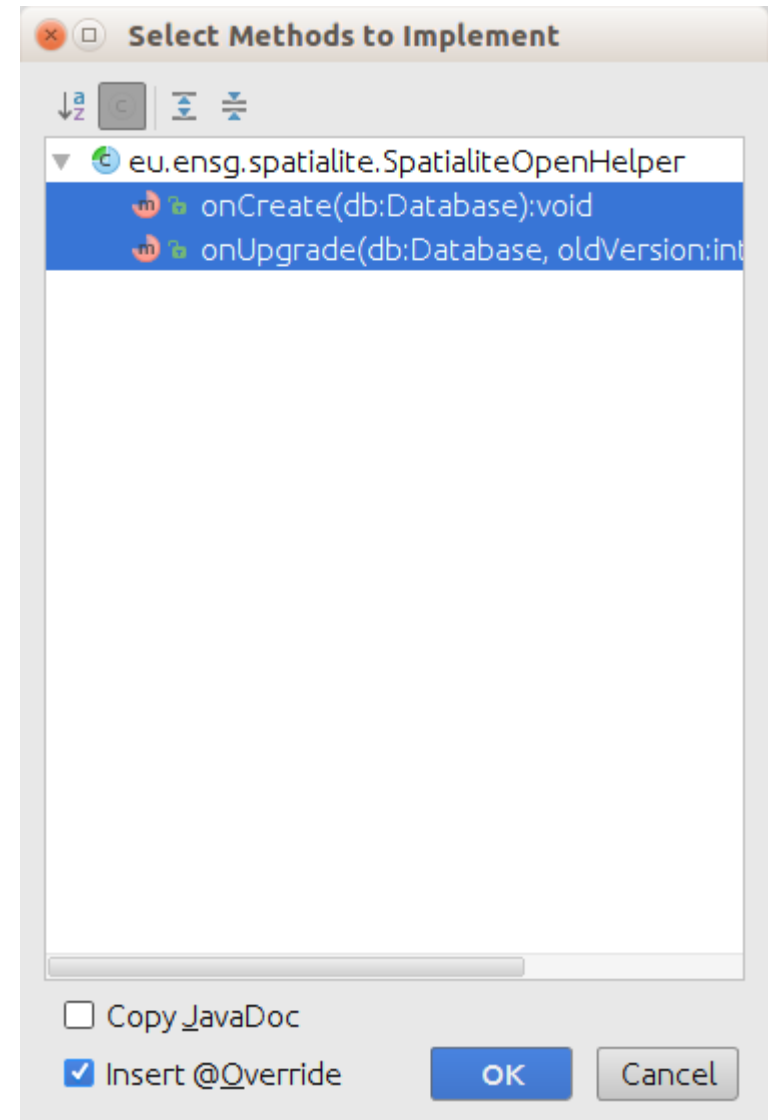
OpenHelper

- Surcharger la méthode `onCreate`
 - ➔ Pour créer le schema de notre base de données
 - ➔ Utiliser la méthode `super.exec`
- La méthode `onUpgrade` reste vide pour le moment

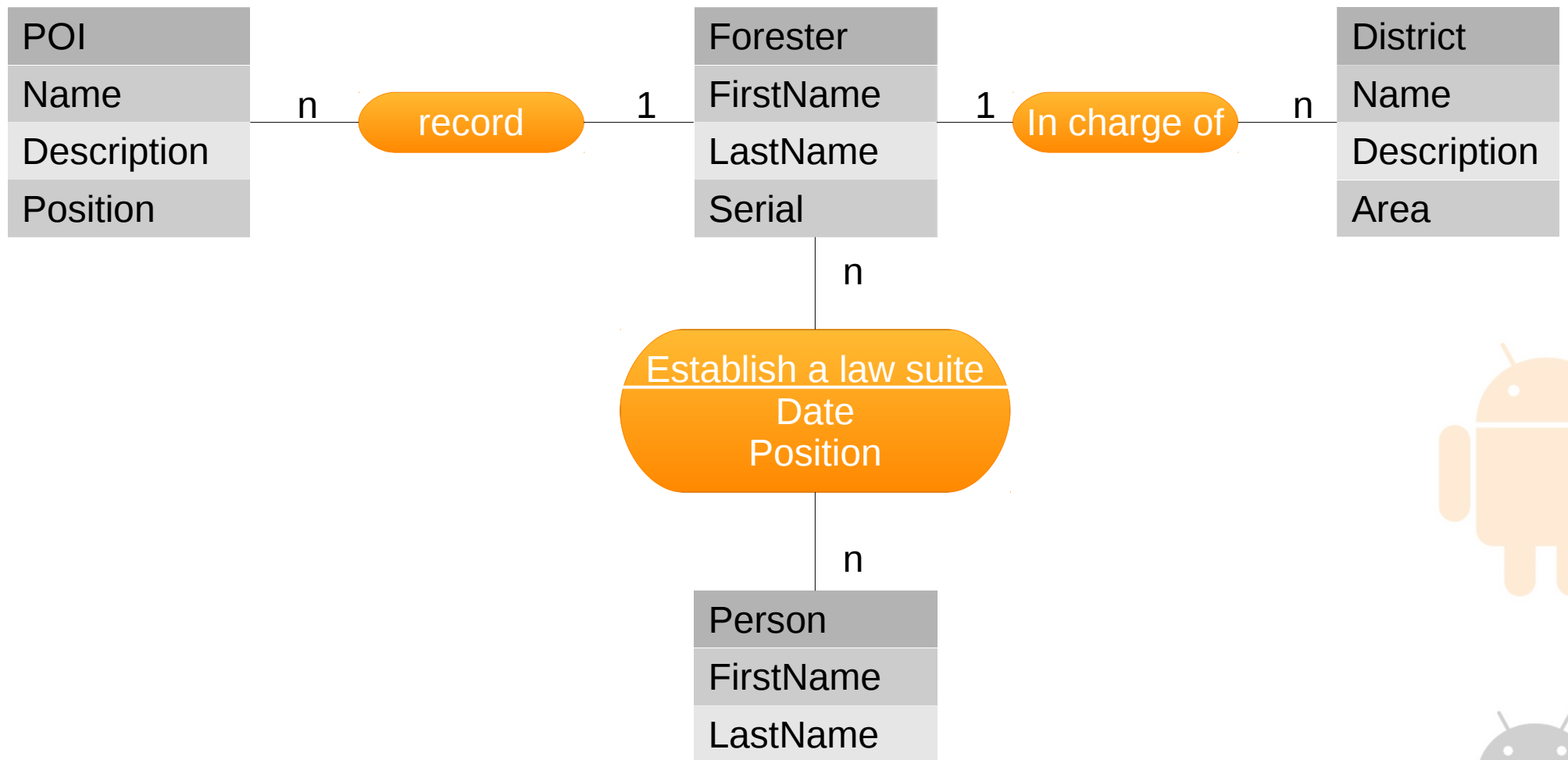


Quelques raccourcis clavier

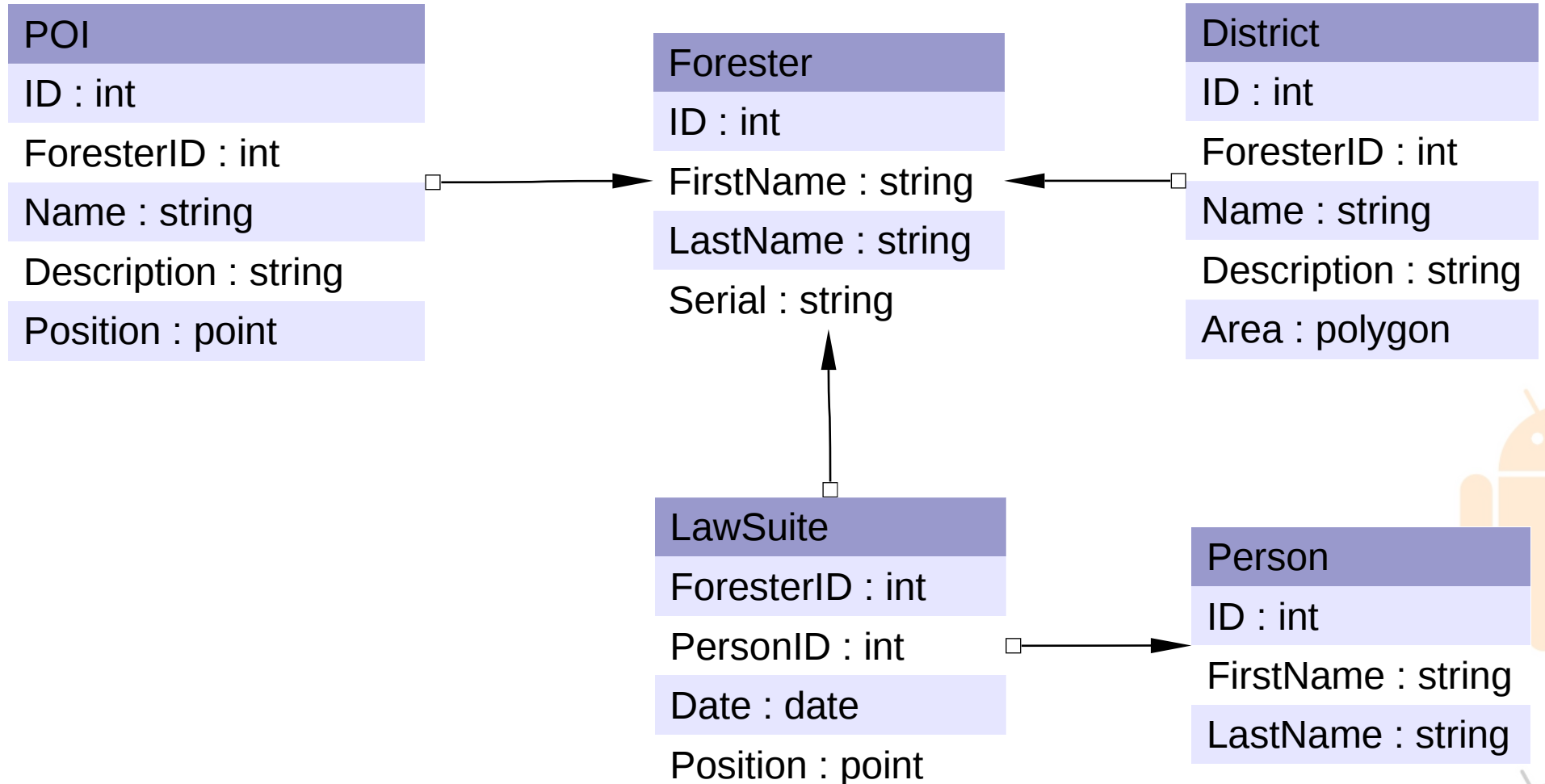
- Quand il y a l'ampoule rouge <alt> + <enter>
- Dans la classe <alt> + <ins>
- Implement methods



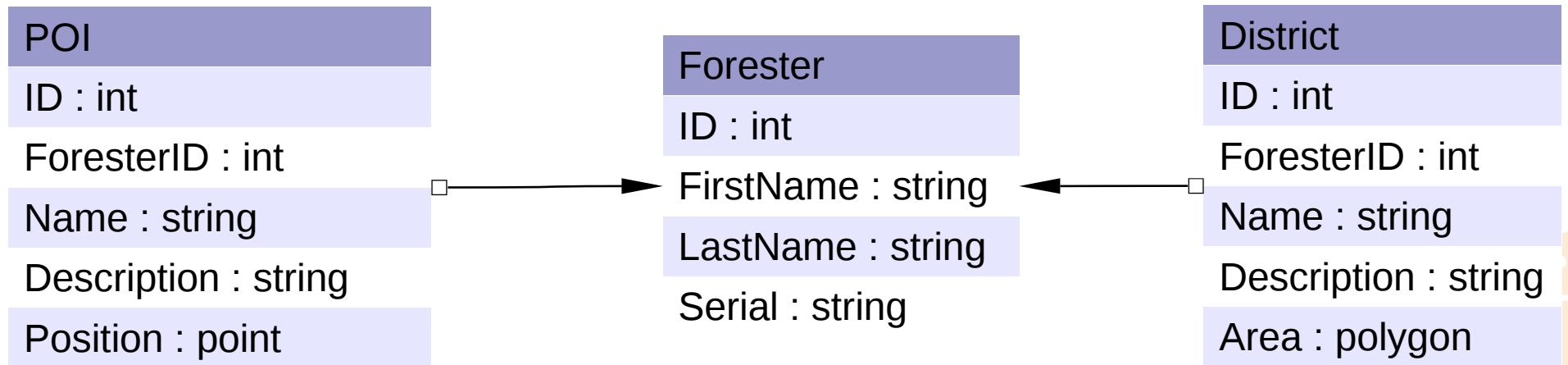
MCD



MLD



MLD Simplifié



Création de table

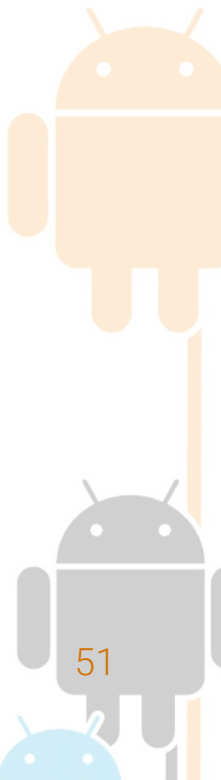
■ `CREATE TABLE Forester (`
 `ID integer PRIMARY KEY AUTOINCREMENT,`
 `FirstName string NOT NULL,`
 `LastName string NOT NULL,`
 `Serial string NULL`
`);`

Nom de la table



Clé primaire

Nullable

Non nullable



Création de table

```
■ CREATE TABLE PointOfInterest (  
    ID integer PRIMARY KEY AUTOINCREMENT,  
    ForesterID integer NOT NULL,  Clé étrangère  
    Name string NOT NULL,  
    Description string,  
    CONSTRAINT FK_poi_forester  
        FOREIGN KEY (foresterID)  
        REFERENCES forester (id)  Contrainte clé étrangère  
);
```

Création de la colonne geometrique

- **SELECT**
`AddGeometryColumn('PointOfInterest'`
`, 'position', 4326, 'POINT', 'XY',`
`0);`

Géométrie Nullable

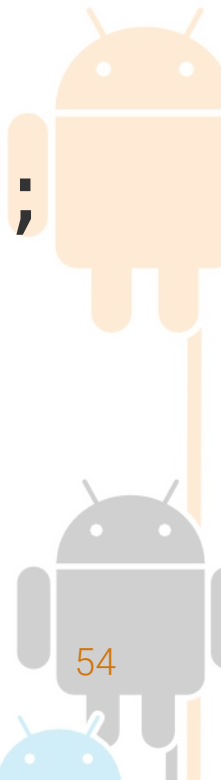
SRID

Type de géométrie

Nb dimensions

Table district

- La même structure que la table PointOfInterest
- Mais avec le champ géométrique au format POLYGON
- ```
SELECT
AddGeometryColumn('District',
'Area', 4326, 'POLYGON', 'XY', 0);
```



# Utilisation de la base

- Dans l'activity LoginActivity
- Au moment de la création de l'activity
- Il faut aussi gérer les exceptions

```
private void initDatabase() {
 try {
 SpatialiteOpenHelper helper = new ForesterSpatialiteOpenHelper(this);
 database = helper.getDatabase();
 } catch (SQLException | IOException e) {
 e.printStackTrace();
 Toast.makeText(this,
 "Cannot initialize database !", Toast.LENGTH_LONG).show();
 System.exit(0);
 }
}
```

# Requêtes

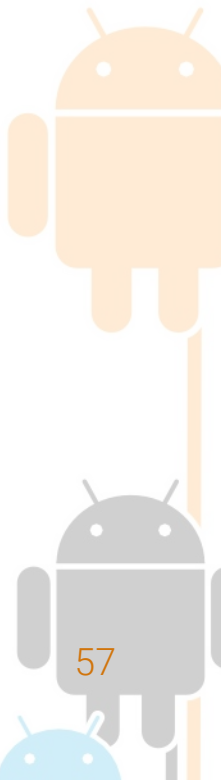
- Lorsque l'utilisateur s'authentifie, il faut vérifier que l'utilisateur existe dans la base de donnée
  - ➔ `login_onClick()`
- Une requête de recherche sur le `SerialNumber`
- Dans une vraie gestion de comptes utilisateur :
  - ➔ Gère un couple login / password
  - ➔ Les password est d'abord encrypté (algorithme de hachage SHA-1 ou MD5) avant d'être stocké dans la base de données





# Requêtes - Rappels

- Utilisation des objects :
  - ➔ `SpatialDatabase`, `Stmt`
- Utilisation des méthodes :
  - ➔ `Stmt.step`
  - ➔ `Stmt.column_Int`
  - ➔ `Stmt.column_String`



# Requêtes - suite

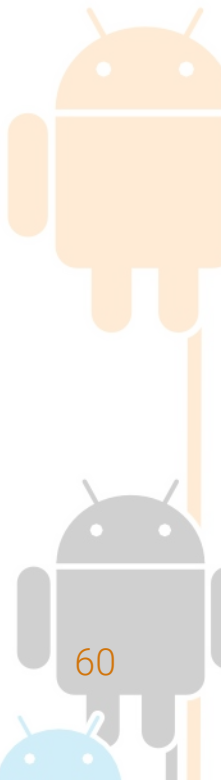
- Il faut pouvoir créer un nouvel utilisateur
- Écrire l'initialisation du helper et de la base de données ainsi que les requêtes nécessaires dans l'activity  
`CreateUserActivity`
- Une fois logué, l'ID de l'utilisateur devra être passé en paramètres à chaque Activity
  - `getIntent().getIntExtra()`
- Attention à l'injection SQL :
  - `DatabaseUtils.sqlEscapeString()`

# Requêtes - spatiales

- Au chargement de la `mapActivity`, il faut initialiser la connexion à la base de données
- Lorsque l'utilisateur ajoute un point d'intérêt, il faut le stocker dans la base de données spatiale
  - ➔ `INSERT INTO PointOfInterest ...`
  - ➔ Colonne géométrique `Position`
  - ➔ `ST_GeomFromText`
  - ➔ `Point.toSpatialiteQuery` attention au SRID

# Requêtes - spatiales

- Lorsque l'utilisateur ajoute un district, il faut la stocker dans la base de données spatiale
  - ➔ Même chose que pour les points d'intérêts
- Au chargement de la map activity, il faut lire les données existantes de la base de données et les tracer
  - ➔ `database.prepare`
  - ➔ `ST_asText`
  - ➔ `Geom.unmarshall`



# IN01 – TP

## Web services

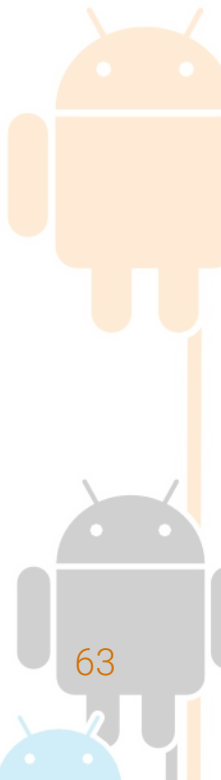


# Reverse Geocoding

- Il faut maintenant ajouter l'adresse des points d'intérêts
- Principe, appeler le web service de Geocoding de Google
  - ➔ `https://maps.googleapis.com/maps/api/geocode/json?latlng=40.714224,-73.961452&key=YOUR_API_KEY`
- Et interpréter le message en retour

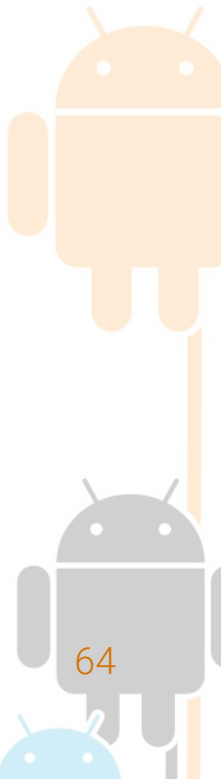
# Chargement du message en Java

- Utilisation de l'objet `URL` et de sa méthode `openConnection()`
- Qui renvoie un objet `HttpURLConnection` lequel possède la méthode `getInputStream()`
- J'ai préparé un utilitaire pour vous : `eu.ensg.commons.io.WebServices`




# Pour que cela fonctionne

- `NetOnMainThreadException`
  - ➔ Il faut que l'appel au webservice se fasse dans un thread "à part"
  - ➔ L'objet `AsyncTask` est prévu pour cela





# AsyncTask

- Créer une tâche asynchrone et une fenêtre d'attente pour l'utilisateur
- `new AsyncTask<Location, Void, String>( )`
  - ... 
- `executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR, currentLocation);`

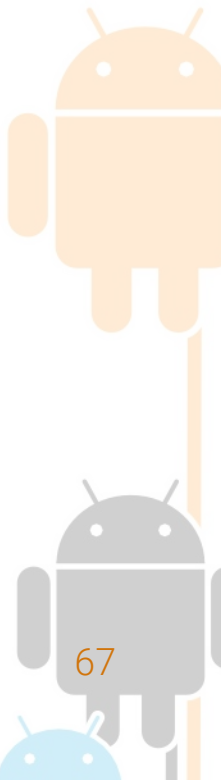
# AsyncTask

- Surchargeons les méthodes `onPreExecute`, `doInBackground`, `onPostExecute`
- Dans `preExecute` nous créons la fenêtre de dialogue avec l'object `ProgressDialog`
- Et nous instrumentons (refactoring) le code de l'appel au web service dans l'`asyncTask`
- La partie `doInBackground` charge les données
- La partie `onPostExecute` s'occupe du parsing



# Pour que cela fonctionne - bis

- Il faut également activer l'API
  - ➔ Depuis le site :  
`https://console.developers.google.com`

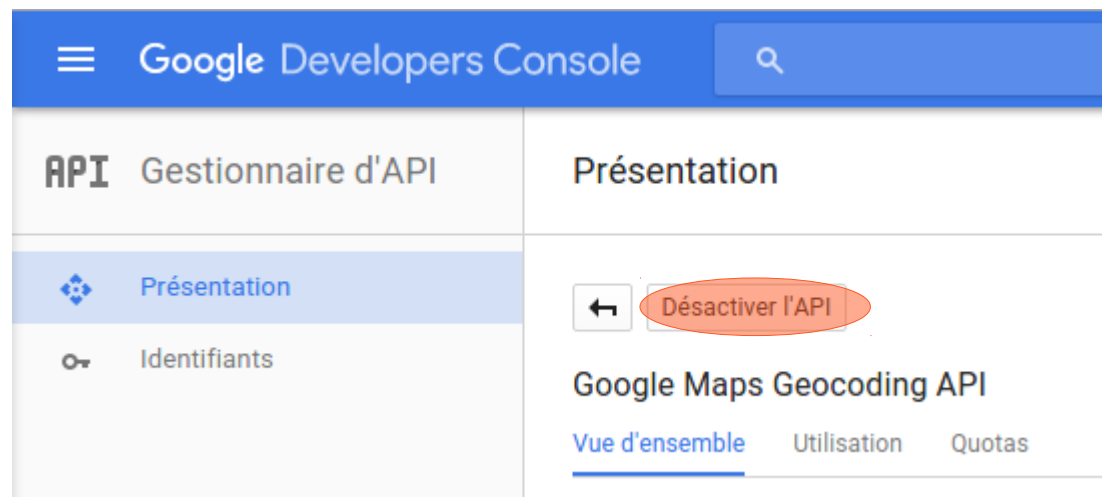


# Activation de l'API

The screenshot shows the Google Developers Console API Manager interface. The top navigation bar is blue with the text "Google Developers Console" and a search icon. Below the navigation bar, the left sidebar contains the "API Gestionnaire d'API" section with sub-items: "Présentation" (highlighted with a blue bar and a diamond icon) and "Identifiants" (with a key icon). The main content area is titled "Présentation" and shows "API Google" and "API activées (9)". A search bar is present with the placeholder text "Rechercher dans plus de 100 API". Below the search bar, the "API populaires" section is displayed. It is divided into two columns. The left column, titled "API Google Cloud" with a hexagonal icon, lists: "Compute Engine API", "BigQuery API", "Cloud Storage Service", "Cloud Datastore API", "Cloud Deployment Manager API", "Cloud DNS API", and a "Plus" link. The right column, titled "API Google Maps" with a map icon, lists: "Google Maps Android API", "Google Maps SDK for iOS", "Google Maps JavaScript API", "Google Places API for Android", "Google Places API for iOS", "Google Maps Roads API", "Google Static Maps API", "Google Street View Image API", "Google Maps Embed API", "Google Places API Web Service", "Google Maps Geocoding API" (highlighted with a red oval), "Google Maps Directions API", "Google Maps Distance Matrix API", "Google Maps Geolocation API", "Google Maps Elevation API", "Google Maps Time Zone API", and a "Moins" link.

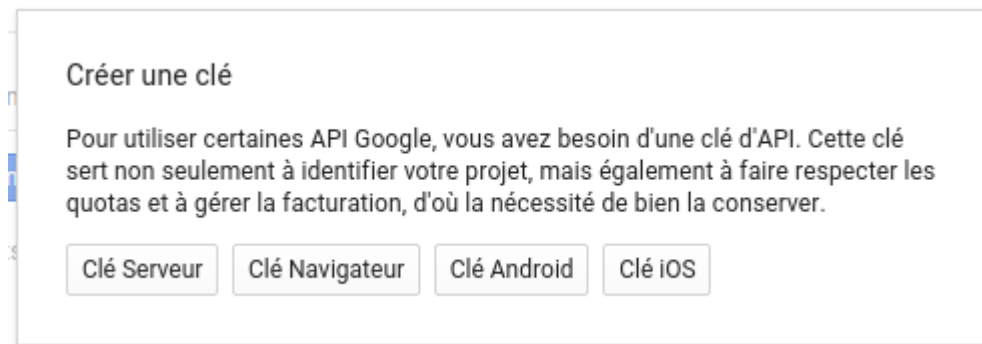
# Activation de l'API

- Cliquer sur Activer l'API



# Création d'une clé serveur

- Puis dans identifiant, il faut créer une clé Serveur



- Et copier celle-ci dans le projet

| <input type="checkbox"/> Nom             | Date de création ▼ | Type    | Clé                                     |
|------------------------------------------|--------------------|---------|-----------------------------------------|
| <input type="checkbox"/> Clé Serveur API | 9 févr. 2016       | Serveur | AlzaSyBfsTbWE7s5g6nn_LfegbvnJF1Z_Pcf61s |

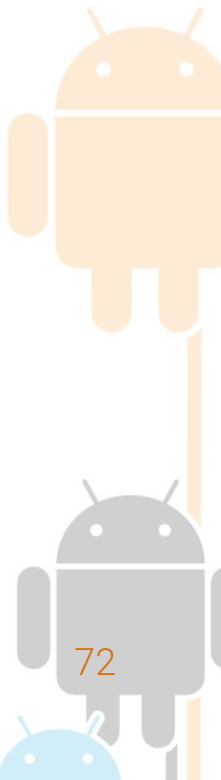
# JSon

- Interpreter le message en retour

```
{
 "results": [
 {
 "address_components": [...],
 "formatted_address": "1 Place de l'Hôtel de ville,
74100 Annemasse, France",
 "geometry": {
 "location": {
 "lat": 46.1931525,
 "lng": 6.234087199999999
 }, ...
 }
 }
]
}
```

# Parsing

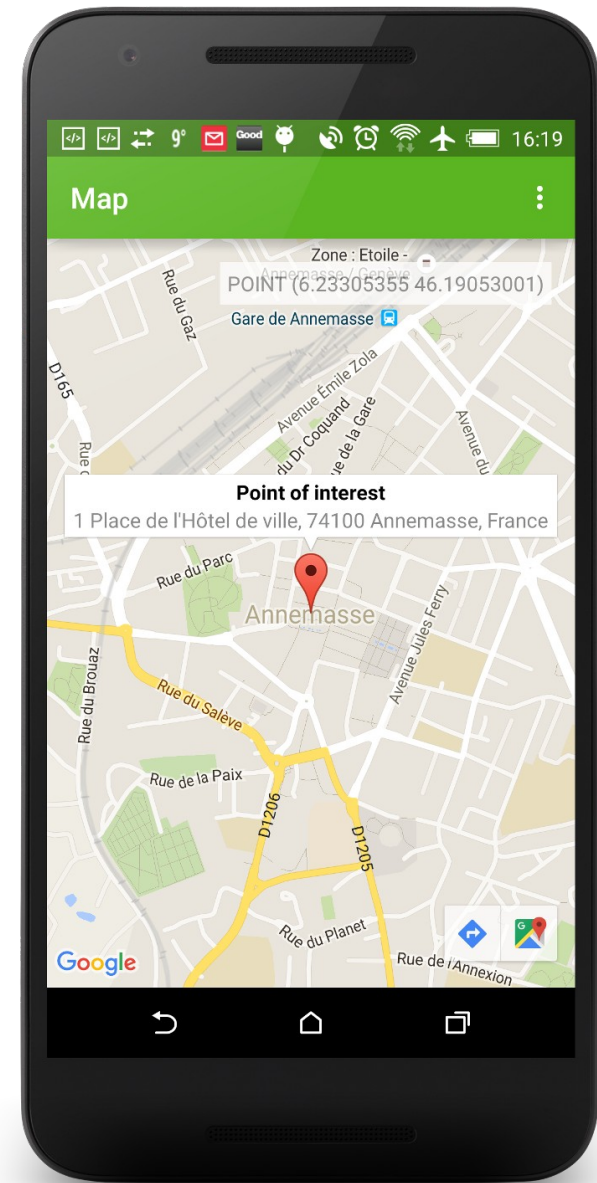
- Utilisons l'object `JsonObject`
- `new JsonObject(String json) : parcourt et crée un objet json`
- Et les méthodes
  - ➔ `getJsonObject(String name) : parcourt un nouveau sous objet`
  - ➔ `getJSONArray(String name)`
  - ➔ `getString(String name)`





# Résultat

- Et voilà le travail !



# Web services météo (bonus)

- Afficher la météo actuelle
- Service météo au format XML ou Json
- <http://www.geonames.org/>
- Construisons l'URL :  
`http://api.geonames.org/findNearByWeatherJSON?  
lat=%.4f&lng=%.4f&username=cyann`

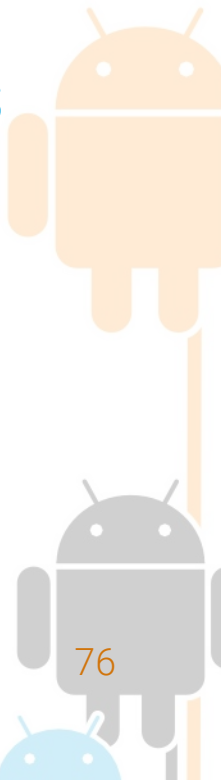
# JSON

- Dans un navigateur internet cette fois

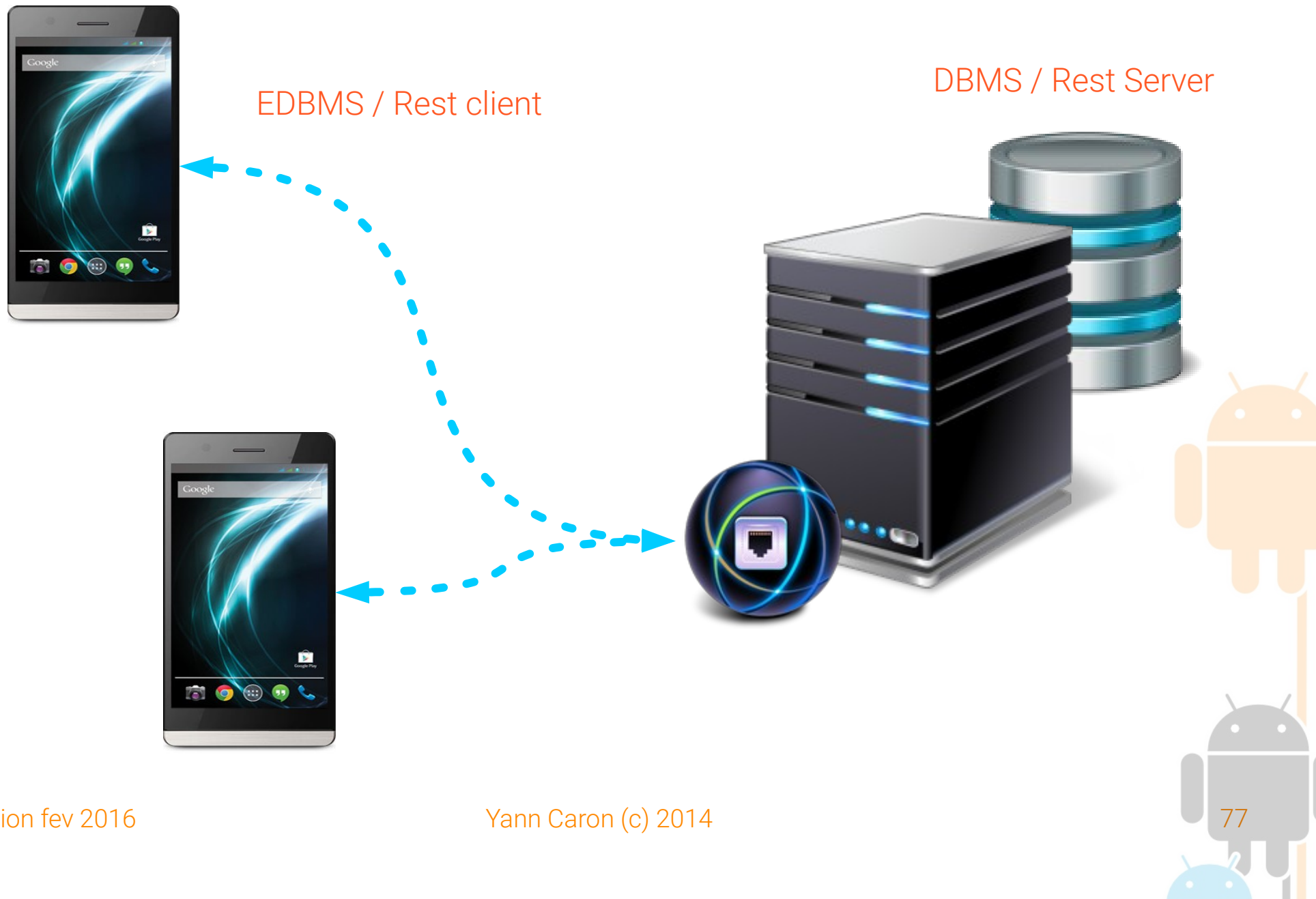
```
{
 "weatherObservation": {
 "elevation": 8, "lng": -1.8, "observation": "LESO 150930Z 03007KT 350V050
9999 SCT020 BKN040 09/07 Q1026",
 "ICAO": "LESO", "clouds": "scattered clouds",
 "dewPoint": "7", "cloudsCode": "SCT",
 "datetime": "2016-01-15 09:30:00",
 "countryCode": "ES", "temperature": "9", "humidity": 87,
 "stationName": "San Sebastian / Fuenterrabia",
 "weatherCondition": "n/a", "windDirection": 30,
 "hectoPascAltimeter": 1026, "windSpeed": "07", "lat": 43.35
 }
}
```

# Ouvertures

- Pourquoi les webservices sont ils si importants ?
- Possibilité de partager les données avec un serveur de base de données centralisé
- Réflexion, comment garantir que les données seront synchronisées qu'une seule fois ?



# Architecture



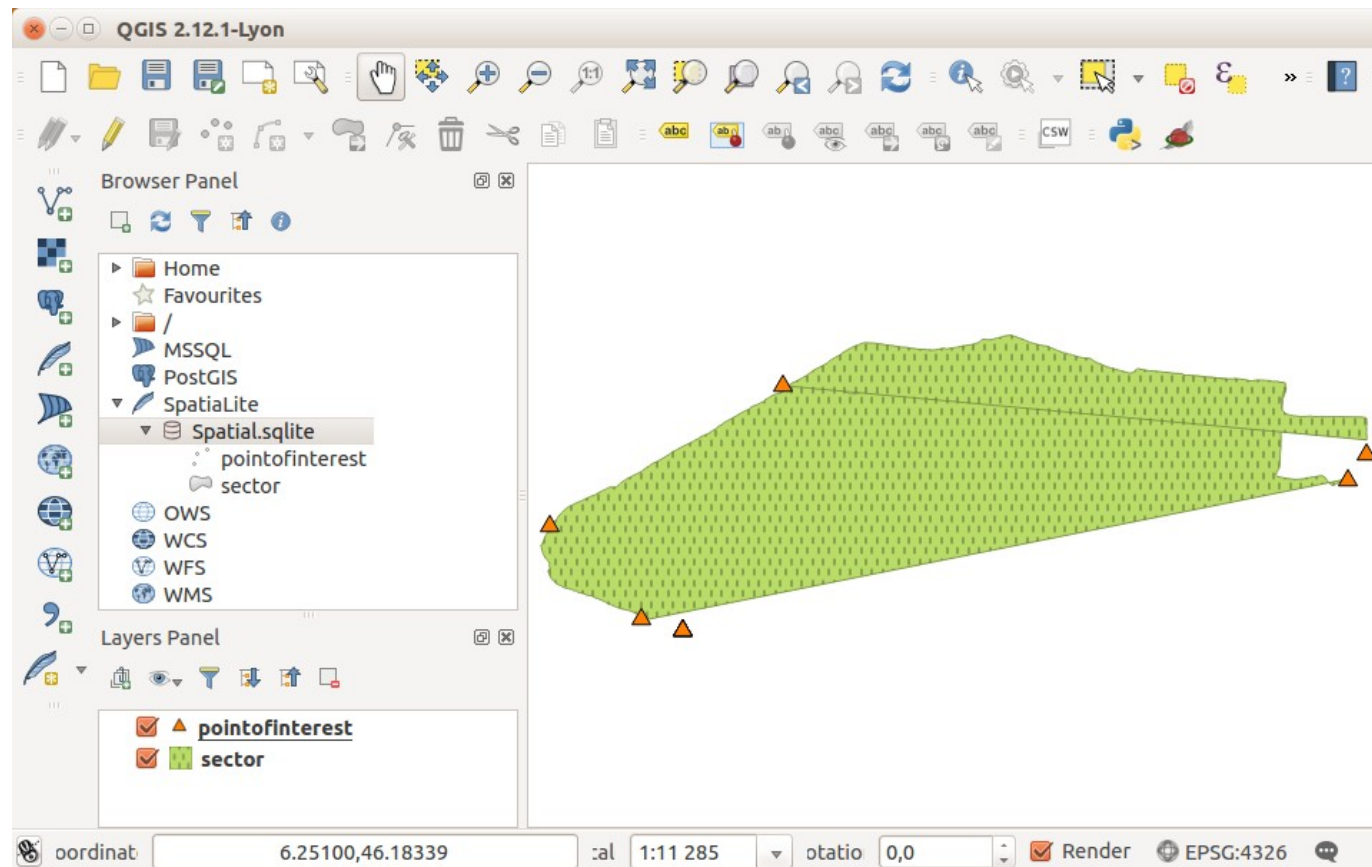
# IN01 – Séance 07

Boni



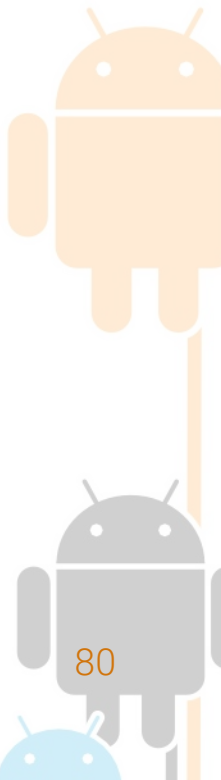
# Copier la base de donnée

- Nous voulons exploiter les données dans un GIS



# Copier la base de donnée

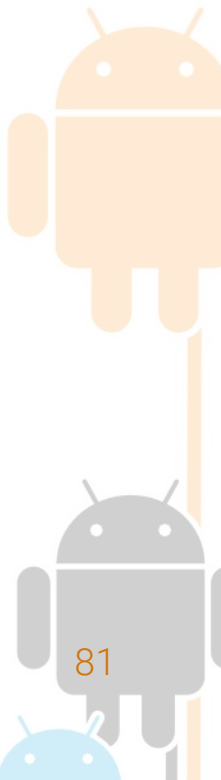
- Principe : extraire le fichier sqlite de l'appareil
- Problème : le gestionnaire des droits Android nous interdit d'avoir accès au fichier interne à l'application
- Solution : depuis l'application, copier le fichier sqlite dans le répertoire `/sdcard`
  - ➔ `context.getDatabasePath([DB_NAME])`
  - ➔ `new File(Environment.getExternalStorageDirectory(), [DB_NAME])`





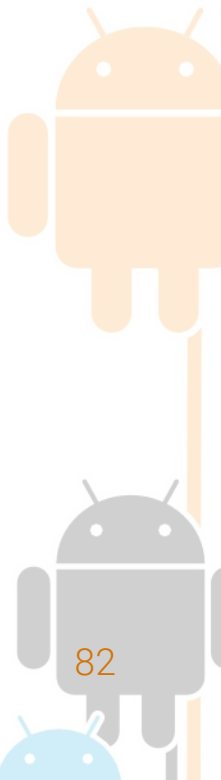
# ADB

- L'appareil branché, utiliser la commande `adb pull`
- `adb pull /sdcard/Spatial.sqlite`
- Que l'on peut maintenant exploiter avec QGis



# Autres opérations

- Restaurer la base de données
- Vider celle-ci
- Ne pas oublier les fenêtres de confirmation

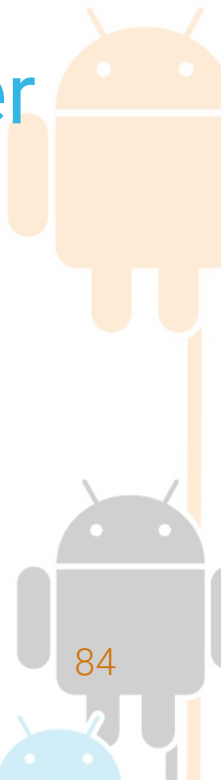


# GridView

- Depuis l'appareil, nous voulons visualiser les tables de la base de données sous la forme d'une grille
- Android dispose d'une vue : GridView
- <http://developer.android.com/guide/topics/ui/layout/gridview.html>
- Nous voulons gérer les deux tables avec une seule activity et une seule grille
  - ➔ Comment faire ?

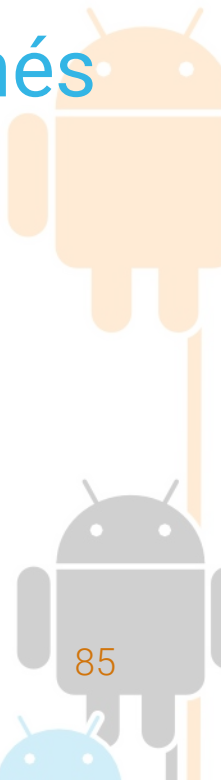
# GridView

- Il faut créer une nouvelle activity (commune aux deux tables à visualiser)
- Il faut créer les menus
- Et lors de l'appel à l'activity, il faut passer en paramètre quel requête SQL on veut visualiser (EXTRA\_SQL)



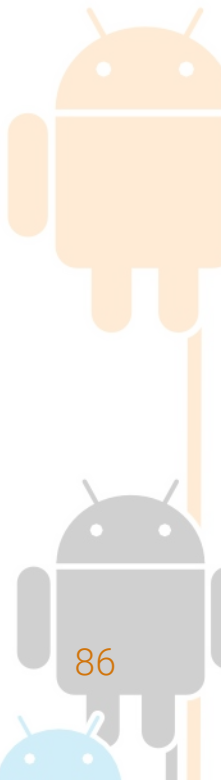
# Adapter

- Design pattern « Template Method »
- Il faut créer une classe qui implémente la super classe BaseAdapter
- Les méthodes getCount, getItem(pos), getItemId(pos) et getView doivent être renseignés
- Le plus simple est de passer une List<String>
- Le plus élégant est de passer une List<PointOfInterest> et de gérer la généricité



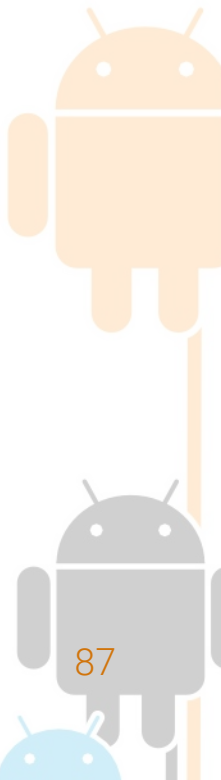
# Charger la liste

- Principe ; boucler sur toutes les valeurs du statement
- Mettre les données dans la liste
- Créer un adapter avec la liste des données précédemment renseignés
- Renseigner le nombre de colonne : `gridView.setNumColumns`.



# Publication

- Android studio prend en charge cette partie
- Menu : Build / Generate Signed APK
- Cliquons sur le bouton create new (keystore)



# Signer l'APK

**New Key Store**

Key store path:  ...

Password:  Confirm:

Key

Alias:

Password:  Confirm:

Validity (years):

Certificate

First and Last Name:

Organizational Unit:

Organization:

City or Locality:

State or Province:

Country Code (XX):

OK Cancel

**Generate Signed APK**

Key store path:

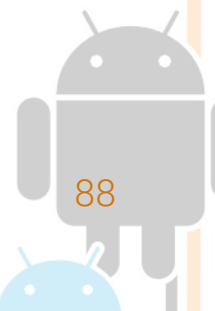
Key store password:

Key alias:  ...

Key password:

☒ Remember passwords

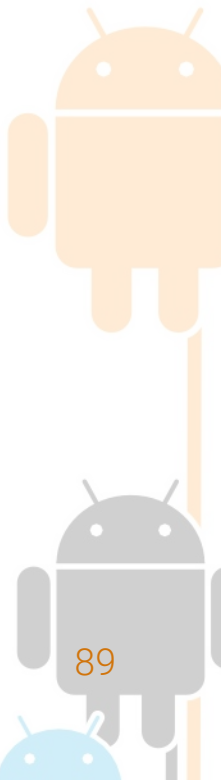
Help Previous Next Cancel





# Publier sur le play store

- Il faudra un compte sur <https://play.google.com/apps/publish>
- L'adhésion est payante (20\$ une seule fois)
- Il faudra enregistrer l'application et la renseigner



# Publier sur le play store

The screenshot shows the Google Play Developer Console interface for the application 'Algoid - Programming language'. The app is published and its status is 'Publication standard'. The version '38' is in production. The console provides options to manage the app, including uploading new APKs, testing with beta or alpha users, and viewing analytics. The left sidebar contains navigation links for various developer tools and app management tasks.

**Google Play Developer Console**

**Algoid - Programming language**  
fr.cyann.algoid [Afficher sur le Play Store](#)

**PUBLIÉE** 11 décembre 2015 Annuler la publication de l'application

Publication standard ▼  
Traitement de la mise à jour en cours...  
[Envoyer la mise à jour](#)

[Passer en mode avancé](#)

**FICHIERS APK**

**PRODUCTION**  
Version  
**38**

**TESTS BÊTA**  
Configurez des tests bêta pour votre application.

**TESTS ALPHA**  
Configurez des tests alpha pour votre application.

**CONFIGURATION DE LA VERSION EN PRODUCTION** [Importer un nouveau fichier APK en version production](#)

**FICHIER APK ACTUEL** date de publication : **11 déc. 2015 07:42:57**

**Appareils compatibles**  
**11376**  
[Afficher la liste](#)

**Appareils exclus**  
**0**  
[Gérer les appareils exclus](#)

| ▼ VERSION         | IMPORTÉ LE          | ÉTAT                 | ACTIONS |
|-------------------|---------------------|----------------------|---------|
| <b>38 (1.3.0)</b> | <b>11 déc. 2015</b> | <b>en production</b> |         |

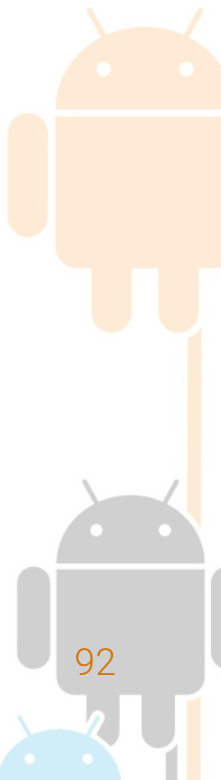
# IN01 – Séance 07

Pour aller plus loin !



# Ouvertures ?

- Synchroniser les données sur un serveur spatial centraliser / via web-service
- Une extension ORMLite spatiale
- Une bibliothèque REST / Spatiale



# Fin

- Merci de votre attention
- Des questions ?

