Document : recherche sur les API de cartes pour l'itération 3 + guide Itération 4

Après nous être rendu compte que l'API Mapsforge était très peu documentée et une fois votre accord obtenu nous nous sommes intéressés à utiliser une autre API.
J'ai commencé (@jacquesmoati) à travailler en me documentant sur l'API de google Maps qui en propose une multitude et ayant des utilités différentes (geocoding, reverse geocoding, route optimization etc). J'ai commencé à installer l'API dans un projet test sur eclipse, créé mon compte google cloud, obtenue ma clé de sécurité pour lancer l'api mais je me suis rendu compte que le nombre de requête sur l'API Google Maps est très limitée depuis peu. Même si Google offre 200$ de bons pour utiliser cette api mais je préfère utiliser une solution totalement gratuite (5$ pour 10k requête).

J'ai donc continué mes recherches mais la plupart des API en java pour afficher des cartes sont disponibles uniquement pour des applications android.
Ainsi, il reste une petite dizaine d'API abordable ou gratuite à condition de ne pas excéder.

1) Utilisation de l'API **LocationIQ (https://locationiq.com/)**

J'ai donc testé en créant un petit projet en local où j'ai cloné ce projet https://github.com/location-iq/locationiq-java-client.git qui est l'api complète. J'ai pu lire la doc et les méthodes que l'on pourrait éventuellement exploiter.
Cependant, j'ai eu des erreurs lorsque j'ai voulu changer le pom dans notre projet, il faut lors de l'itération 4 s'en occuper d'en un premier temps.
Ajout de la dépendance :

```
<dependency>
 <groupId>com.locationiq</groupId>
 <artifactId>locationiq-java-client</artifactId>
 <version>1.0.0</version>
 <scope>compile</scope>
</dependency>
```

J'ai créé mon compte locationIQ pour générer la clé secrète (demandée pour appeler l'api) afin de pouvoir requêter et travailler sur la classe BalanceAPI et SearchAPI pour comprendre leur fonctionnement. Cette seconde classe permet à l'aide d'une simple recherche en brute, par exemple on recherche : « universite paris Descartes » mais nous n'avons pas l'adresse, ni rien d'autres, l'output est un json que l'on pourra ensuite exploiter. Ce json est riche en information car en plus de nous donner le lieu exacte de l'endroit entré, il nous donne le quartier, le type de lieu (university), et surtout les positions GPS qui pourront être exploitées dans un second temps pour l'itinéraire ou l'affichage du point GPS.

Exemple : en entrant « université paris Descartes paris » en input (string q)

```
ApiClient defaultClient1 = Configuration.getDefaultApiClient();

// Configure API key authorization: key
ApiKeyAuth key2 = (ApiKeyAuth) defaultClient1.getAuthentication("key");
key2.setApiKey("5edd30041af660");
// Uncomment the following line to set a prefix for the API key, e.g. "Token" (defaults to n
//key.setApiKeyPrefix("Token");

SearchApi apiInstance1 = new SearchApi();
String q = "universite paris descartes paris"; // String | Address to geocode
String format = "json"; // String | Format to geocode. Only JSON supported for SDKs
Integer normalizecity = 1; // Integer | For responses with no city value in the address sect
Integer addressdetails = 1; // Integer | Include a breakdown of the address into elements. D
String viewbox = ""; // String | The preferred area to find search results.  To restrict res
Integer bounded = 1; // Integer | Restrict the results to only items contained with the viewl
Integer limit = 1; // Integer | Limit the number of returned results. Default is 10.
String acceptLanguage = "fr"; // String | Preferred language order for showing search result
String countrycodes = "fr"; // String | Limit search to a list of countries.
Integer namedetails = 1; // Integer | Include a list of alternative names in the results. Th
Integer dedupe = 1; // Integer | Sometimes you have several objects in OSM identifying the s
Integer extratags = 0; // Integer | Include additional information in the result if availabl
Integer statecode = 0; // Integer | Adds state or province code when available to the statec
try {
    List<Location> result = apiInstance1.search(q, format, normalizecity, addressdetails, vi
    System.out.println(result);
} catch (ApiException e) {
    System.err.println("Exception when calling SearchApi#search");
    e.printStackTrace();
}
// API token goes here
```

On obtient ce type de json exploitable :

[class Location {
  placeId: 95214486
  licence: https://locationiq.com/attribution
  osmType: way
  osmId: 56645234
  boundingbox: [48.8509564, 48.8519751, 2.3400064, 2.3418225]
  lat: 48.85139235
  lon: 2.34111160499217
  displayName: Université Paris Descartes - Paris V, Rue de l'École de Médecine, Quartier de la Monnaie, Paris 6e Arrondissement, Paris, Île-de-France, France métropolitaine, 75006, France
  propertyClass: amenity
  type: university
  importance: 0.69996057842905
  address: class Address {
    houseNumber: null
    road: Rue de l'École de Médecine
    residential: null
    borough: null
    neighbourhood: null
    quarter: null
    hamlet: null
    suburb: Quartier de la Monnaie
    island: null
    village: null

```
    town: null
    city: Paris
    cityDistrict: Paris 6e Arrondissement
    county: Paris
    state: Île-de-France
    stateDistrict: null
    postcode: 75006
    country: France
    countryCode: fr
    stateCode: null
  }
  namedetails: class Namedetails {
    name: Université Paris Descartes - Paris V
  }
}]
```

→ Itération 4

Pour l'**itération 4**, il faudrait alors modifier l'interface graphique en ajoutant, lieu de l'ordre de mission et lieu de départ (exemple université paris dauphine → université paris Descartes ou n'importe), cette première API va ensuite générer deux json pour le départ et l'arrivée. On pourra détailler sur l'ordre de mission le détail des adresses, quartier etc. Puis utiliser une seconde api comme OSM pour afficher le trajet entre les 2 points.
On doit donc remplacer la classe ReverseGeoCode du projet par une nouvelle classe faisant appelle à la classe SearchAPI de l'API LocationIQ.

2) Utilisation de MapBox, Google Maps ?
autres choix : https://openlayers.org/ …
Check : https://github.com/eppleton/leaflet4j