Vélo Epicurien

GLO-4035

Guillaume HECTOR
Jérémy LEVILAIN
Jonathan BIBAS

Sommaire

Sommaire **2**

Introduction **3**

Révisions **3**

Stratégie d'acquisition des données **4**

Itinéraires 4

Restaurants 4

Technologies utilisées **4**

Langage de programmation 4

Technologies de persistance 5

PostgreSQL 5

Neo4J 5

Processus d'ETL **5**

Obtention des données initiales et incrémentales 5

Transformation des données 6

Annexe **9**

Exemple de données extraites d'OpenStreetMap 9

Exemple de données extraites de Google Place Search 10

Introduction

Ce document vise à décrire le plus précisément possible la direction technique de notre projet afin de prouver sa faisabilité. Il vise aussi à expliquer à la fois notre stratégie d'acquisition des données ainsi que nos choix de technologies au sein du projet.

La ville choisie pour l'étude est celle de **Paris, France**.

Révisions

Cette section permet de garder une trace des modifications du document, permettant une traçabilité des modifications entre les différents rendus.

Date	**Sections modifiées**	**Commentaire**
26/10/2020	Introduction, Stratégie d'acquisition des données, Technologies utilisées, Processus d'ETL, Annexe	Correction des parties en fonction des remarques du premier rendu. Ajout du processus d'ETL.

Stratégie d'acquisition des données

Cette section présente la stratégie d'acquisition des données, soit la source des données et la présentation de celles-ci.

Itinéraires

La récolte des données cartographiques ainsi que les données concernants les routes et accès pour les vélos sera tirée des informations disponibles sur **OpenStreetMap** via leur API ouverte, gratuite et communautaire. Son approche communautaire permet d'avoir des résultats précis, les contributeurs étant généralement plus motivés et précis que des entreprises privées.

Restaurants

La récolte des données des restaurants se fera via l'API de **Google "Place Search"** afin de trouver des restaurants proches, on utilisera en plus pour compléter les données l'API de TheFork.com.

Technologies utilisées

Cette section va présenter et justifier les technologies utilisées par le produit. Dans un premier temps le langage de programmation sélectionné et ensuite les technologies de persistance.

Langage de programmation

Le langage de programmation sélectionné pour ce projet est **Typescript**. Typescript permet de mêler la simplicité du Javascript avec l'assurance qualité d'un langage fortement typé. Ce langage est par ailleurs maîtrisé par l'ensemble des membres de l'équipe, favorisant aussi l'intégration de nouveaux membres. L'écosystème fort et la communauté très présente permet l'accès à des bibliothèques activement maintenues facilitant ainsi le développement du projet. Ce même écosystème apporte énormément de solutions natives permettant de répondre à une charge croissante : ces systèmes garantiront une haute disponibilité pour notre application.

Technologies de persistance

MongoDB

MongoDB est une base de données NoSQL moderne permettant de stocker des données sous forme de document sans se soucier d'un schéma, c'est-à-dire d'un format précis et uniforme. MongoDB est aujourd'hui un des système de gestion de base de données les plus utilisés. MongoDB est un bon candidat, l'équipe possédant déjà de l'expérience dessus. L'approche moderne de MongoDB lui permet d'être facilement mis à l'échelle grâce à ses stratégies de réplications et de sharding simples. De plus, sa capacité à pouvoir manipuler des données géospatiales est une des raisons primordiales ayant orientée notre choix. Cette technologie sera utilisée afin de stocker les informations des restaurants, c'est-à-dire et sans se limiter, son nom, types de gastronomie, la moyenne des avis, et bien évidemment, ses coordonnées géographiques.

Neo4J

Neo4J est un système de gestion base de données structurant les données sous formes de graphes, avec des noeuds et des relations. Elle permet de stocker des données aux nombreuses relations et ce, en très grande quantité. Neo4J permet d'effectuer des opérations géospatiales ce qui nous aidera énormément lors de la création d'itinéraires. Leader dans son domaine, sa documentation complète et les nombreux exemples font de lui un candidat idéal. Une politique de sharding est disponible afin de permettre de répliquer stratégiquement et efficacement les données, favorisant l'ingestion de plus grandes quantités de données. Neo4J sera parfait afin de stocker une grande quantité de données géographiques représentant les pistes cyclables. Les restaurants environnants seront représentés par un identifiant faisant lien avec notre base de données SQL présentée précédemment.

Processus d'ETL

Cette section présente les différentes étapes afin d'ingérer les données de façon à préparer l'utilisation de l'application. Dans un premier temps, l'extraction des données est présentée, puis les différentes étapes de transformation.

Obtention des données initiales et incrémentales

Afin d'obtenir les données à ingérer dans les différentes bases de données de l'application, deux APIs publiques et gratuites sont utilisées comme sources. Aucune action manuelle d'un gestionnaire n'est nécessaire.

Dans un premier temps, une série de requêtes est effectuée sur l'API Google Places Search afin de récupérer tous les restaurants dans le périmètre traité par notre application (se référer à l'introduction du document). Un exemple de réponse est disponible en annexe. Chaque restaurant renvoyé par l'API est accompagné d'un identifiant unique. Le processus d'ETL veille à stocker ce même identifiant afin de s'assurer de ne pas traiter plusieurs fois cette même donnée, et ce même lorsque de plusieurs exécution du script, permettant d'ingérer de façon incrémentale.

Dans un second temps, un batch de requêtes similaire est effectué cette fois-ci sur OpenStreetMap, reprenant aussi le principe d'identifiant unique. Une particularité de cette source de données est qu'elle renvoie énormément d'informations, que le script devra filtrer et transformer pour être conforme à nos besoins.

Une vue globale du processus de récupération des données est présentée ci-dessous en illustration 1.

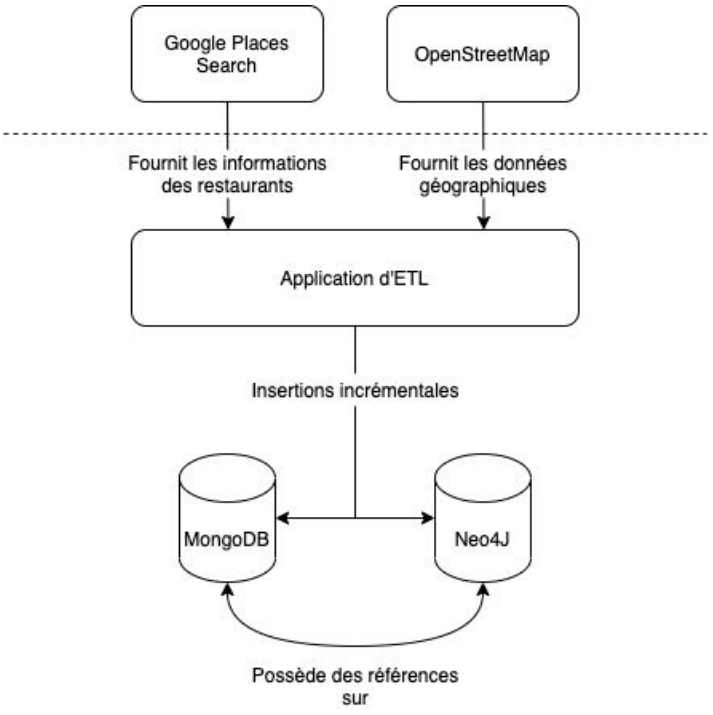

Illustration 1 : Procédure d'ETL

Transformation des données

Les deux bases de données utilisées par l'application ne sont pas adaptées au stockage brut des données récupérées. Elles nécessitent une ou plusieurs transformations. Concernant les données des restaurants, les parties pertinentes sont extraites et directement insérées dans la base MongoDB, dans une collection **`restaurants`**, l'identifiant unique de l'API est ajouté dans un champ spécifique **`placeId`**. De la même manière, les différents points des pistes cyclables sont aisément insérées dans la base Neo4J en tant que noeud, de type **`Way`**. Le script veillera à stocker les liaisons entre les noeuds de son côté en attendant la fin des insertions. Les liaisons sont ensuite ajoutées entre les noeuds connus, les autres sont abandonnées, n'étant pas (ou qu'en partie) dans la zone traitée par l'application.

Les données n'étant ni sensibles, ni personnelles, aucun processus d'anonymisation est nécessaire. Un diagramme du processus de transformation est disponible à l'illustration 2. Un autre décrivant le processus du point de vue applicatif est ajouté à l'illustration 3.

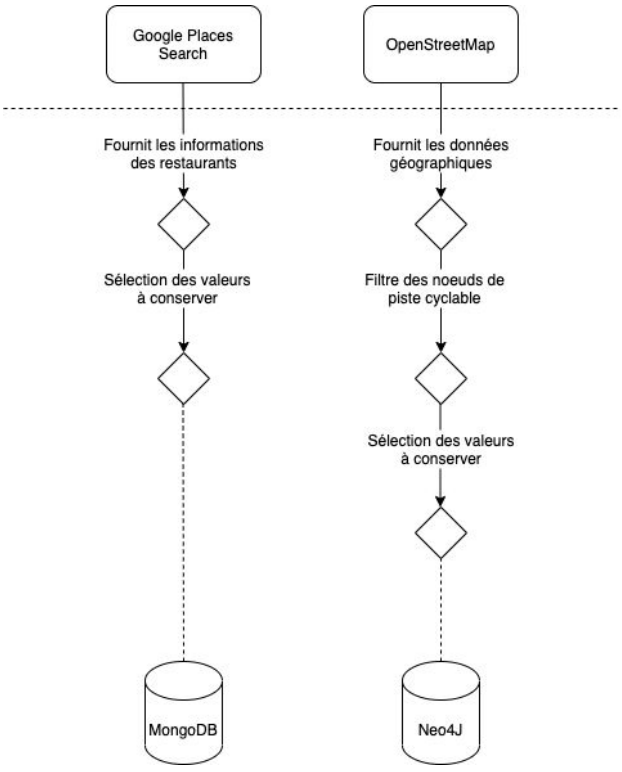

Illustration 2 : Processus de transformation

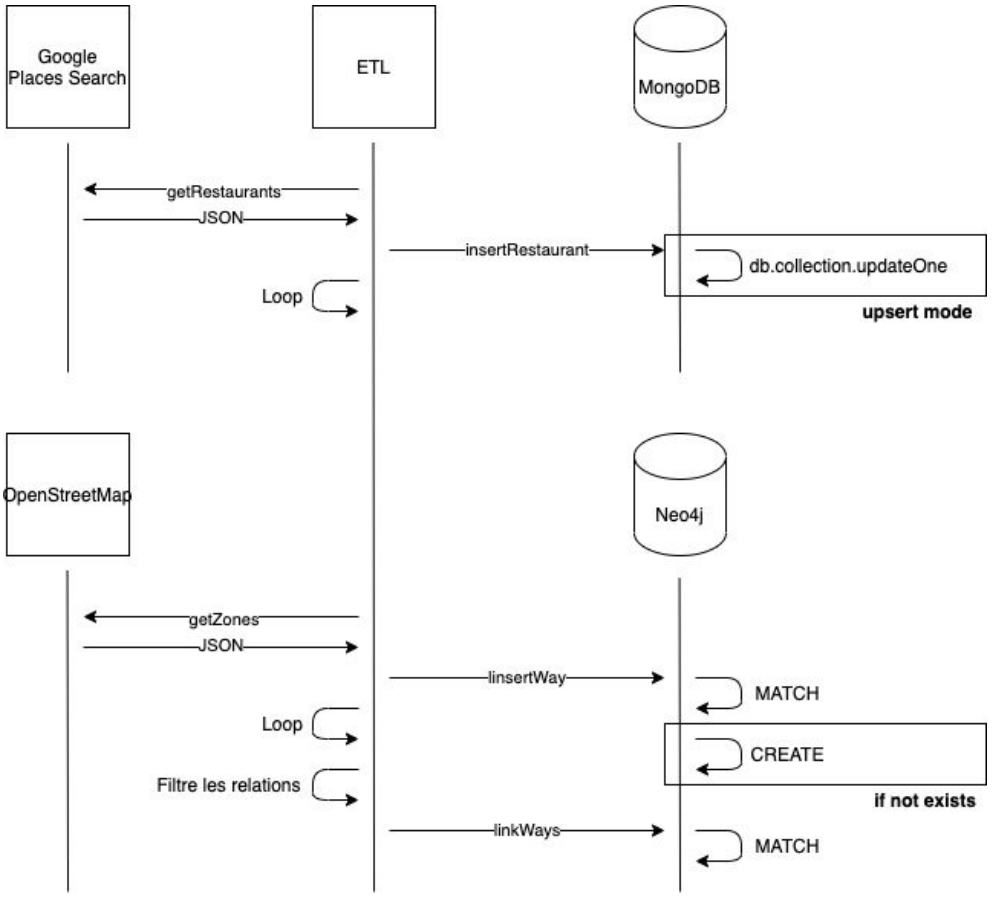

Illustration 3 : Processus d'extraction de l'application

Annexe

Exemple de données extraites d'OpenStreetMap

Comme on peut le voir sur la représentation JSON[1] ci-dessous, l'itinéraire est une succession d'éléments représentant le trajet : signalétique, angles, croisements, routes, etc. Nous utiliserons ce mécanisme d'enchaînement d'éléments afin d'optimiser l'ingestion des données.


```
{
  "version": "0.6",
  "generator": "CGImap 0.8.3 (2868701 spike-08.openstreetmap.org)",
  "copyright": "OpenStreetMap and contributors",
  "attribution": "http://www.openstreetmap.org/copyright",
  "license": "http://opendatacommons.org/licenses/odbl/1-0/",
  "elements": [
    {
      "type": "node",
      "id": 361730,
      "lat": 48.8311993,
      "lon": 2.3543524,
      "timestamp": "2018-10-19T08:14:28Z",
      "version": 8,
      "changeset": 63670386,
      "user": "Chlc",
      "uid": 2322305,
      "tags": {
        "highway": "traffic_signals"
      }
    }
  ]
}
```


[1] https://www.openstreetmap.org/api/0.6/relation/8728882/full.json

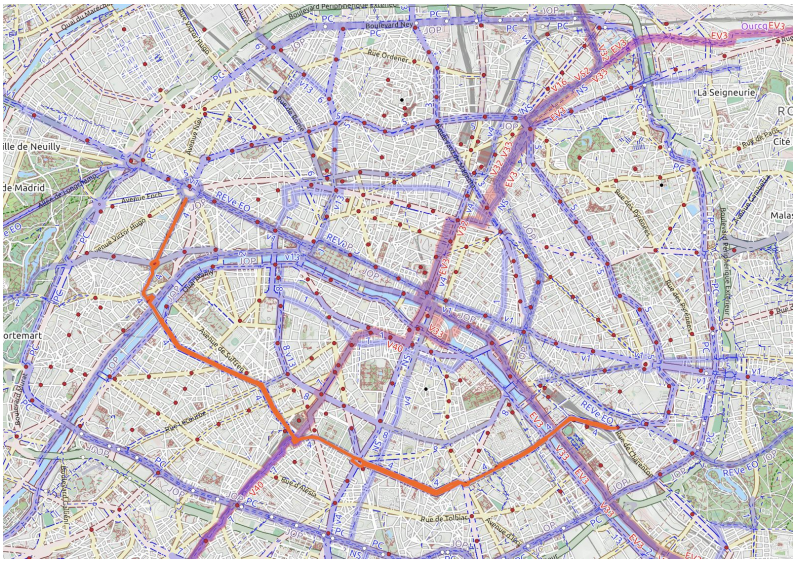

Illustration 4 : Représentation graphique de l'itinéraire

Exemple de données extraites de Google Place Search

Les données renvoyées par Google Places, exemple[2] ci-dessous, contiennent beaucoup plus d'informations que nécessaire mais pourront être utiles si nous souhaitons ajouter plus de fonctionnalités dans un temps futur.

```
{
  business_status: 'OPERATIONAL',
  geometry: {
    location: { lat: 48.8837188, lng: 2.3285891 },
    viewport: { northeast: [Object], southwest: [Object] }
  },
  icon:
'https://maps.gstatic.com/mapfiles/place_api/icons/v1/png_71/restaurant-71
.png',
  name: 'Indiana Café - Clichy',
  opening_hours: { open_now: true },
  photos: [
    {
      height: 275,
      html_attributions: [Array],
      photo_reference:
'CmRaAAAARuKlBDKWPUsMACi_tDnGsKIYSCZbRmwpfA0ESkmbKUAw8fJZUyysE9WAYMWiA4PrF
AYq1jffozDmdIZWYmfNTTfvVt4GrqjKGlUOQD22MUFbi7NKiJPNyHojzThXZTegEhAlU1lNA6I
HbckTM5LBB5lyGhSjIVHYf1o1F-nJhcxAhWDmWVxezg',
      width: 950
    }
```

[2] https://developers.google.com/places/web-service/search

```
  ],
  place_id: 'ChIJsYVcNkxu5kcRjOa9kJwI6dA',
  plus_code: {
    compound_code: 'V8MH+FC Paris, France',
    global_code: '8FW4V8MH+FC'
  },
  price_level: 2,
  rating: 3.7,
  reference: 'ChIJsYVcNkxu5kcRjOa9kJwI6dA',
  scope: 'GOOGLE',
  types: [
    'cafe',
    'bar',
    'restaurant',
    'food',
    'point_of_interest',
    'store',
    'establishment'
  ],
  user_ratings_total: 1106,
  vicinity: '79 Boulevard de Clichy, Paris'
}
```