Coulibaly Cheick Ahmed
cheick7cca@gmail.com

Recherche de données

J'ai choisi d'exporter le jeu de données intitulé « **Burkina Faso Populated Places (OpenStreetMap Export)** »
https://data.humdata.org/dataset/hotosm_bfa_populated_places

Description des données : ce jeu de données comprend les lieux habités au Burkina Faso, tels que les villes, villages, hameaux et autres localités. Les données sont extraites d'OpenStreetMap (OSM) et fournissent des informations géospatiales sur ces lieux, notamment leurs noms, types et coordonnées géographiques.

Description du format JSON : les données sont disponibles au format GeoJSON, un format standard pour représenter des objets géographiques. Chaque entrée dans le fichier GeoJSON représente un lieu habité avec les attributs suivants :

- **name** : Nom du lieu
- **name:en** : Nom en anglais (le cas échéant)
- **place** : Catégorie du lieu (par exemple, city, town, village, hamlet, isolated_dwelling)
- **population** : Population estimée (si disponible)
- **is_in** : Indication de la région ou de l'entité administrative dans laquelle se trouve le lieu
- **source** : Source des données
- **name:fr** : Nom en français (le cas échéant)

Chaque lieu est représenté par un objet de type "Feature" avec une géométrie de type "Point" spécifiant les coordonnées géographiques (longitude et latitude).

La période de collecte de la base : la première date de collecte des données de cette base est le 5 décembre 2019, puis elle a été mise à jour 10 février 2025.

La fréquence de mise à jour : les mises à jour se font chaque mois.

Le nom de l'organisme qui la fournie : ces données sont fournies par Humanitarian OpenStreetMap Team (HOT), une organisation qui utilise et soutient OpenStreetMap pour des projets humanitaires et de développement.

Pré-traitement des données

Le fichier téléchargé est-il au format JSON ?

Le fichier **GeoJSON (.geojson)** est bien un fichier JSON structuré spécifiquement pour les données géospatiales. Les autres formats (GPKG, KML, Shapefile) ne sont pas du JSON mais peuvent être convertis en GeoJSON.

La structure du JSON obtenu serait-elle idéale pour une insertion en base MongoDB ?

Pas forcément. MongoDB accepte le GeoJSON, mais certains ajustements peuvent être nécessaires.

Extrait du fichier :


```
{

  "type": "FeatureCollection",
  "name": "hotosm_bfa_populated_places_points_geojson",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "name": "Tombo",
        "name:en": null,
        "place": "hamlet",
        "population": null,
        "is_in": null,
        "source": "NGA",
```

```
        "name:fr": null,
        "osm_id": 7560785332,
        "osm_type": "nodes"
      },
      "geometry": {
        "type": "Point",
        "coordinates": [0.0577324, 14.7802223]
      }
    },
.....
```


Les points corrects :
- La structure **FeatureCollection** est bien un format standard GeoJSON.
- Les objets ont une clé "geometry" qui contient un "type": "Point" et des "coordinates".
- Les informations supplémentaires sont dans "properties", ce qui est logique.

Problèmes pour MongoDB :
- MongoDB ne stocke pas directement "FeatureCollection" : il faut extraire les objets "features" et insérer chaque "Feature" individuellement.
- Nom incorrect pour l'indexation géospatiale : MongoDB attend un champ "location" et non "geometry".
- Certaines valeurs nulles, il faudrait soit les supprimer, soit les remplacer par des valeurs par défaut.

Que faut-il changer ?

Chaque élément "Feature" doit être restructuré pour s'adapter à une base MongoDB avec un index **2dsphere** :

Exemple :


```
{
  "name": "Tombo",
  "place": "hamlet",
  "population": null,
  "source": "NGA",
  "osm_id": 7560785332,
  "location": {
    "type": "Point",
    "coordinates": [0.0577324, 14.7802223]
  }
}
```


Ajout de l'index géospatial avant l'insertion :
db.geoJSONtp2.createIndex({ location: "2dsphere" });

Preprocess.py

1) Utilisation du fichier pour générer un fichier (output.json) dont le format est compatible avec une insertion mongoimport :
python3 preprocess.py -i hotosm_bfa_populated_places_points_geojson.geojson -o output.json

2) Importation des données via mongoimport :
mongoimport --db uvbf --collection geoJSONtp2 --file output.json –jsonArray

3) Compréhension du code : le code app.py est une application web simple utilisant Flask, qui génère des localisations aléatoires autour d'un point donné.

4) Vérification : Les points retournés sont maintenant triés du plus proche au plus éloigné.