

ENSAI - FILIÈRE SID

- Projet de fin de module -
Technologies NoSql

Charlène Noé

Janvier 2018

1 Ma définition du meilleur endroit pour vivre à New York

Le but de ce projet consiste à trouver le meilleur endroit pour s'installer dans la ville de New York. Étant donné que ce concept est propre à chaque personne, je vais présenter ma vision du « meilleur endroit ». Pour moi, cet endroit doit :

- **Être sûr** : la sécurité est primordiale pour se sentir bien dans un nouveau lieu. Un de mes objectifs est donc de trouver un lieu calme qui respire la confiance.
- **Être dynamique** : sur mon temps libre, j'aime bien aller voir une pièce de théâtre, regarder un film au cinéma, aller dans un bon restaurant, etc. C'est donc primordial que l'arrondissement idéal propose de nombreux lieux culturels.
- **Posséder des espaces verts** : comme mon principal hobby est la course à pied, je souhaiterais avoir des parcs autour de mon logement afin de pratiquer ce sport que j'aime tant. De plus, je suis quelqu'un qui vient de la campagne, j'aimerais donc retrouver un peu de verdure autour de chez moi.

2 Les données

2.1 Présentation des données sélectionnées

Pour trouver ce parfait endroit, la ville de New York met à disposition de nombreuses données réparties en 10 catégories sur leur portail open data. J'ai donc sélectionné 7 bases de données pour tenter de répondre à ma définition du « meilleur endroit » selon les critères que j'ai établi :

1. Sécurité :

- **NYPD Motor Vehicle Collisions** : Liste des collisions impliquant des véhicules motorisés, cyclistes et/ou piétons sur l'année 2013.
- **Water quality complaints** : Base de données des différents types de plaintes sur la qualité de l'eau depuis 2010.
- **NYPD Complaint Data Historic - Felony 2015** : Historique des crimes valides pendant l'année 2015.

2. Dynamisme :

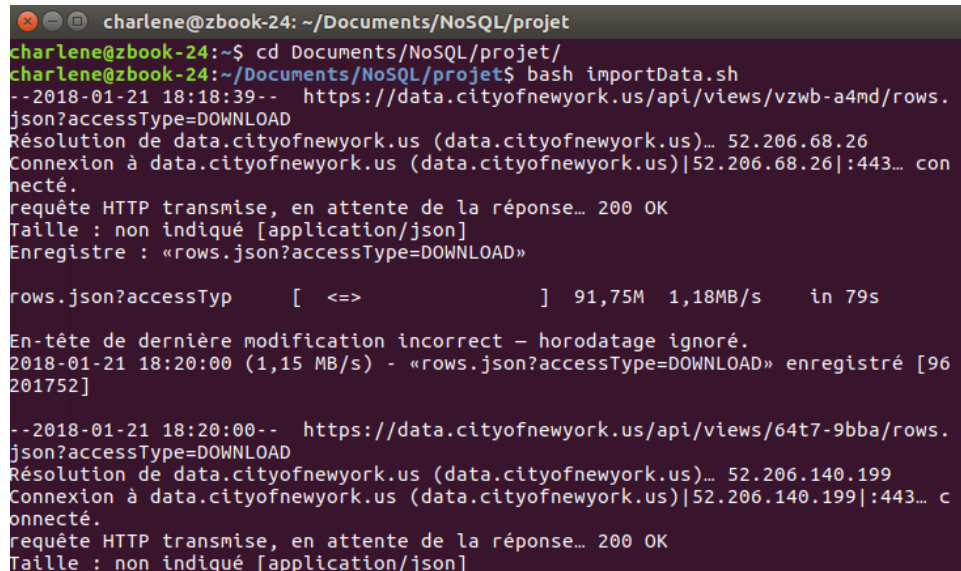
- **DCLA Cultural Organizations** : Base de données des cinémas à NYC.
- **DOHMH New York City Restaurant Inspection Results** : Liste des inspections des restaurants de la ville depuis 2014.

3. Espaces verts :

- **2015 Street Tree Census** : Enregistrements de tous les arbres de New York en 2015.
- **Parks Properties** : Inventaire des différents parcs existants à New York.

2.2 Pour importer les données

Tout d'abord, mettez vous à la racine du répertoire récupéré sur Github. Ensuite ouvrez un terminal et exécutez la ligne de commande `bash importData.sh` (attention, il faut installer python3 avant de le lancer). Dans un premier temps, ce script shell va réorganiser les données en ne gardant que les colonnes qui nous intéressent et va, ensuite, importer les données présentées à la partie précédente dans un répertoire « data ».



```
charlene@zbook-24: ~/Documents/NoSQL/projet
charlene@zbook-24:~$ cd Documents/NoSQL/projet/
charlene@zbook-24:~/Documents/NoSQL/projet$ bash importData.sh
--2018-01-21 18:18:39-- https://data.cityofnewyork.us/api/views/vzwb-a4md/rows.
json?accessType=DOWNLOAD
Résolution de data.cityofnewyork.us (data.cityofnewyork.us)... 52.206.68.26
Connexion à data.cityofnewyork.us (data.cityofnewyork.us)[52.206.68.26]:443... c
onnecté.
requête HTTP transmise, en attente de la réponse... 200 OK
Taille : non indiqué [application/json]
Enregistre : «rows.json?accessType=DOWNLOAD»

rows.json?accessTyp      [ <=>          ] 91,75M 1,18MB/s   in 79s

En-tête de dernière modification incorrect - horodatage ignoré.
2018-01-21 18:20:00 (1,15 MB/s) - «rows.json?accessType=DOWNLOAD» enregistré [96
201752]

--2018-01-21 18:20:00-- https://data.cityofnewyork.us/api/views/64t7-9bba/rows.
json?accessType=DOWNLOAD
Résolution de data.cityofnewyork.us (data.cityofnewyork.us)... 52.206.140.199
Connexion à data.cityofnewyork.us (data.cityofnewyork.us)[52.206.140.199]:443... c
onnecté.
requête HTTP transmise, en attente de la réponse... 200 OK
Taille : non indiqué [application/json]
```

FIGURE 1 – Capture d'écran pour l'importation des données dans le répertoire « data »

3 Installation et imports

3.1 Choix de la base de données

J'ai décidé d'utiliser une base de données NoSQL orienté documents puisque les données de l'Open Data peuvent être téléchargées en format JSON. Les requêtes seront donc effectuées dans une base **MongoDB**. On utilisera la base mongoDB installée dans VirtualBox lors du TP2.

3.2 Installation de MongoDB et Robomongo

Pré-requis : installation de VirtualBox et disposer de la version Ubuntu 17.04.

Avant de mettre les données en base, il faut télécharger le client mongo avec la commande :

```
wget http://91.121.220.23/mongo.vdi.gz
```

Il faut s'assurer que le MD5 du client mongo soit identique à : « 1b0f6ea99737480259e1815f7b490e5e mongo.vdi.gz ». Pour cela, il vous suffit de renseigner la commande suivante (en veillant à se mettre dans le bon dossier) :

```
md5sum mongo.vdi.gz
```

Dans un soucis de simplicité, on exécutera les requêtes à l'aide de Robomongo qui offre une interface plus pratique. Pour l'installer, il faut exécuter la requête suivante :

```
wget https://download.robomongo.org/1.1.1/linux/robo3t-1.1.1-linux-x86_64-c93c6b0.tar.gz
```

Il faut ensuite dézipper le fichier et lancer Robomongo disponible dans le répertoire bin.

3.3 Import des données dans MongoDB

Tout d'abord, il faut lancer MongoDB dans la VirtualBox avec le login « root » et le mot de passe « root » également. Ensuite, dans le shell mongo, il faut saisir la commande `service mongod start` pour activer la base de données. Il faut s'assurer que la base écoute sur le port 27017 grâce à la commande `mongo`.

Pour pouvoir importer les données, il suffit ensuite d'exécuter le bash suivant : `bash importMongo.sh`

Note : Il y a un module Python à installer pour pouvoir exécuter le script associé : `pip3 install pymongo`.

Par la suite, ouvrez Robomongo (dans le répertoire « bin ») et créer une nouvelle connexion sur le port « localhost :27017 ». Vous devez obtenir la vue suivante avec une base de données nommée « projet » qui comporte 7 collections correspondant à nos données :

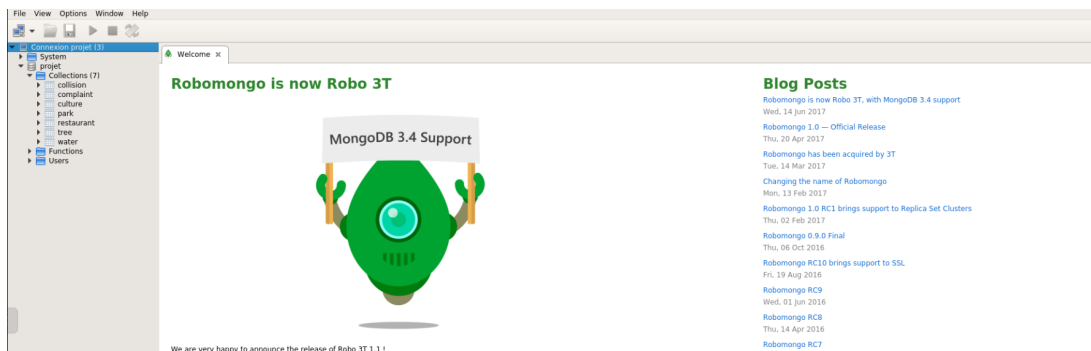


FIGURE 2 – Capture d'écran pour l'importation des données dans MongoDB

4 Recherche du « meilleur endroit »

4.1 Descriptif de la méthode utilisée

Pour chaque base de données importée, on va établir un classement des arrondissements selon les critères exprimés dans le tableau ci-dessous.

Les scripts permettant de réaliser ces différents classements sont présents dans le répertoire « requests » qui sont à exécuter directement dans Robomongo.

Note : le classement consiste à affecter un score de 1 à 5 à chaque arrondissement : 1 désignant le meilleur arrondissement et donc par conséquent, 5 désignant le pire.

Base de données	Nombre de lignes	Critère
Collision	203 720	Hiérarchie des arrondissement selon la mortalité des cyclistes et piétons
Water	9 469	Répartition des plaintes sur la qualité de l'eau selon l'arrondissement
Complaint	37 700	Proportion des plaintes sur les délits selon l'arrondissement
Culture	1 968	Répartition des cinémas et musées de NYC selon l'arrondissement
Restaurant	386 843	Classement du nombre de meilleurs restaurants (grade = A) après inspection dans les différents arrondissements
Tree	683 788	Classification des arrondissements selon leur nombre d'arbres
Park	2 010	Distribution des arrondissements selon leur nombre de parcs

FIGURE 3 – Récapitulatif des critères utilisés

Une fois les arrondissements ordonnés, on agrège cet ordre selon son critère principal : sécurité, dynamisme ou espaces verts. Puis on réalise une moyenne pondérée sur ces 3 axes afin de dégager l'arrondissement « idéal ». J'ai décidé de donner des poids différents à mes critères tout simplement parce que je pense que je suis prête à faire plus de concessions sur certains que sur d'autres. Les poids sont : 0.4 pour les espaces verts et pour la sécurité et 0.2 pour le dynamisme.

4.2 Présentation des résultats

Pour exécuter les requêtes, un script Python est disponible en lançant la commande (Attention : mettez vous à la racine du projet) : `python3 ./DAO/loadData.py`.

Ce script permet d'exécuter les requêtes définies dans le dossier « requests », de récupérer leur résultat et d'afficher les différents classements : le classement par base, par critère et classement final.

Vous devez avoir la fenêtre suivante :

```

charlene@zbook-24: ~/Documents/NoSQL/projet/projetNoSql
charlene@zbook-24:~/Documents/NoSQL/projet/projetNoSql$ python3 ./DAO/request.py

--> Résultat des classements -----
Dans l'ordre : collision, complaint, water, culture, restaurant, tree, park
{'BROOKLYN': [5, 5, 5, 2, 2, 2, 1], 'STATEN ISLAND': [1, 1, 1, 4, 5, 3, 5], 'BRONX': [2, 4, 2, 5, 4, 4, 3], 'MANHATTAN': [4, 2, 3, 1, 1, 5, 4], 'QUEENS': [3, 3, 4, 3, 3, 1, 2]}

--> Résultat des classements par critère global -----
Dans l'ordre : sécurité, dynamisme, espaces verts
{'QUEENS': [3.3333333333333335, 3.0, 1.5], 'BROOKLYN': [5.0, 2.0, 1.5], 'BRONX': [2.6666666666666665, 4.5, 3.5], 'MANHATTAN': [3.0, 1.0, 4.5], 'STATEN ISLAND': [1.0, 4.5, 4.0]}

--> Résultat des classements finaux par ordre croissant -----
[('QUEENS', 2.5333333333333337), ('STATEN ISLAND', 2.9000000000000004), ('BROOKLYN', 3.0), ('MANHATTAN', 3.2), ('BRONX', 3.3666666666666667)]

L'arrondissement idéal est donc le QUEENS !!!
charlene@zbook-24:~/Documents/NoSQL/projet/projetNoSql$

```

FIGURE 4 – Capture d’écran des classements

Explications des sorties

On commence par faire un classement des arrondissements pour chaque base de données. On a donc les résultats suivants :

Critère	Staten Island	Bronx	Manhattan	Queens	Brooklyn
Collision	1	2	4	3	5
Water	1	2	3	4	5
Complaint	1	4	2	3	5
Culture	4	5	1	3	2
Restaurant	5	4	1	3	2
Tree	3	4	5	1	2
Park	5	3	4	2	1

FIGURE 5 – Résultats des classements

Puis, comme expliqué précédemment, je réalise ensuite une moyenne pour chaque critère « global » et obtient le classement moyen en pondérant les résultats obtenus.

Thème	Poids	Staten Island	Bronx	Manhattan	Queens	Brooklyn
Sécurité	0.4	1	2.67	3	3.33	5
Dynamisme	0.2	4.5	4.5	1	3	2
Espaces verts	0.4	4	3.5	4.5	1.5	1.5
MOYENNE	—	2.9	3.4	3.2	2.5	3

FIGURE 6 – Résultat final

C’est donc le Queens qui correspond au mieux à mes critères de recherches !

5 Axes d'amélioration

J'ai vu que certaines données de l'open data de New York étaient disponibles en format GeoJson, un format utilisé dans le cas de position (longitude, latitude). Ces fichiers auraient pu servir à pousser un peu plus loin la recherche du meilleur endroit grâce à leur précision (recherche d'une rue particulière par exemple).

En ce qui concerne le script Python mis en place pour l'interrogation des collections, j'aurais préféré qu'il vienne lire les scripts enregistrés en format javascript et qu'il les exécute. De plus, je n'ai pas réussi à réaliser les requêtes de « proportion » me contentant de réaliser un simple comptage (problème pour stocker une collection temporaire et la requêter ultérieurement avec Python).