

ELLIOT LAGAISE

PROJET vELO

Stations de vélo

GEMA

janvier 2025



SOMMAIRE

01

PROBLÉMATIQUE

02

L'EXISTANT

03

LES API'S

04

RECOLTER LES DONNÉES

05

CLEANING

06

BASE SQL

07

CARTE INTERACTIVE

08

AMELIORATIONS ET
CONCLUSION



1 PROBLEMATIQUE

Créer une application web pour la visualisation des stations de vélos en temps réel.



2 l'existant

Le contexte

Les applications permettant de trouver une station de location de vélos sont peu nombreuses. En effet, à part Google Maps, il est difficile de trouver une application qui recense toutes les villes. La plupart des applications existantes sont spécifiques à une seule ville, comme Ilévia pour Lille ou la RATP pour Paris. C'est pourquoi nous souhaitons créer une application avec une carte interactive qui couvre un plus grand nombre de villes.

3 LES APIS



Pour la collecte des données, je me suis basé sur des APIs fournies par les sites officiels des villes. Voici les liens correspondants pour chaque ville :

- Toulouse : data.toulouse-metropole.fr
- Paris : opendata.paris.fr
- Lille : data.lillemetropole.fr



4 Recolter les données

Pour récolter les différentes données des APIs, deux systèmes de boucles sont utilisés en raison des limitations de collecte imposées par l'API. Pour Lille, aucun problème, il est possible de tout récupérer, alors que pour Toulouse et Paris, il existe une limite.

```
# Check if "results" key is in the response
if "results" in data:
    # Loop through each bike station record in "results"
    for record in data["results"]:
        # Extract relevant fields
        station_info = {
            "stationcode": record.get("stationcode"),
            "name": record.get("name"),
            "is_installed": record.get("is_installed"),
            "capacity": record.get("capacity"),
            "numdockavailable": record.get("numdockavailable"),
            "numbikesavailable": record.get("numbikesavailable"),
            "mechanical_bikes": record.get("mechanical"),
            "electric_bikes": record.get("ebike"),
            "is_renting": record.get("is_renting"),
            "is_returning": record.get("is_returning"),
            "duedate": record.get("duedate"),
            "latitude": record["coordonnees_geo"]["lat"],
            "longitude": record["coordonnees_geo"]["lon"],
            "nom_arrondissement_comunes": record.get("nom_arrondissement_comunes"),
            "code_insee_commune": record.get("code_insee_commune")
        }
        # Add record to the list
        velib_data.append(station_info)

    # If fewer records than the requested limit, we've reached the end
    if len(data["results"]) < rows_per_page:
        break
    else:
        start += rows_per_page # Move to the next page
else:
    print("Key 'results' not found in JSON response.")
    break
except ValueError:
    print("Error decoding JSON.")
    break
else:
    print(f"Request failed with status code {response.status_code}.")
    break

return velib_data
```

```
# Boucle pour extraire les records
for record in records:
    station = {
        'id': record['@id'],
        'nom': record['nom'],
        'adresse': record['adresse'],
        'commune': record['commune'],
        'etat': record['etat'],
        'type': record['type'],
        'nb_places_dispo': record['nb_places_dispo'],
        'nb_velos_dispo': record['nb_velos_dispo'],
        'etat_connexion': record['etat_connexion'],
        'x': record['x'],
        'y': record['y'],
        'date_modification': record['date_modification']
    }
    station_v_lille.append(station)
```



6 Base SQL



Création d'une base sql en local avec phpMyAdmin

Il faut ensuite créer une connexion entre la base de données faite sur Python et une base SQL qui s'actualise automatiquement avec la collecte. Pour cela, il a fallu créer une base en local avec un logiciel nommé MAMP. Après avoir créé des logs et une table sur phpMyAdmin, il est maintenant possible d'établir une connexion entre les deux (cf. code)

Table Sql

phpMyAdmin

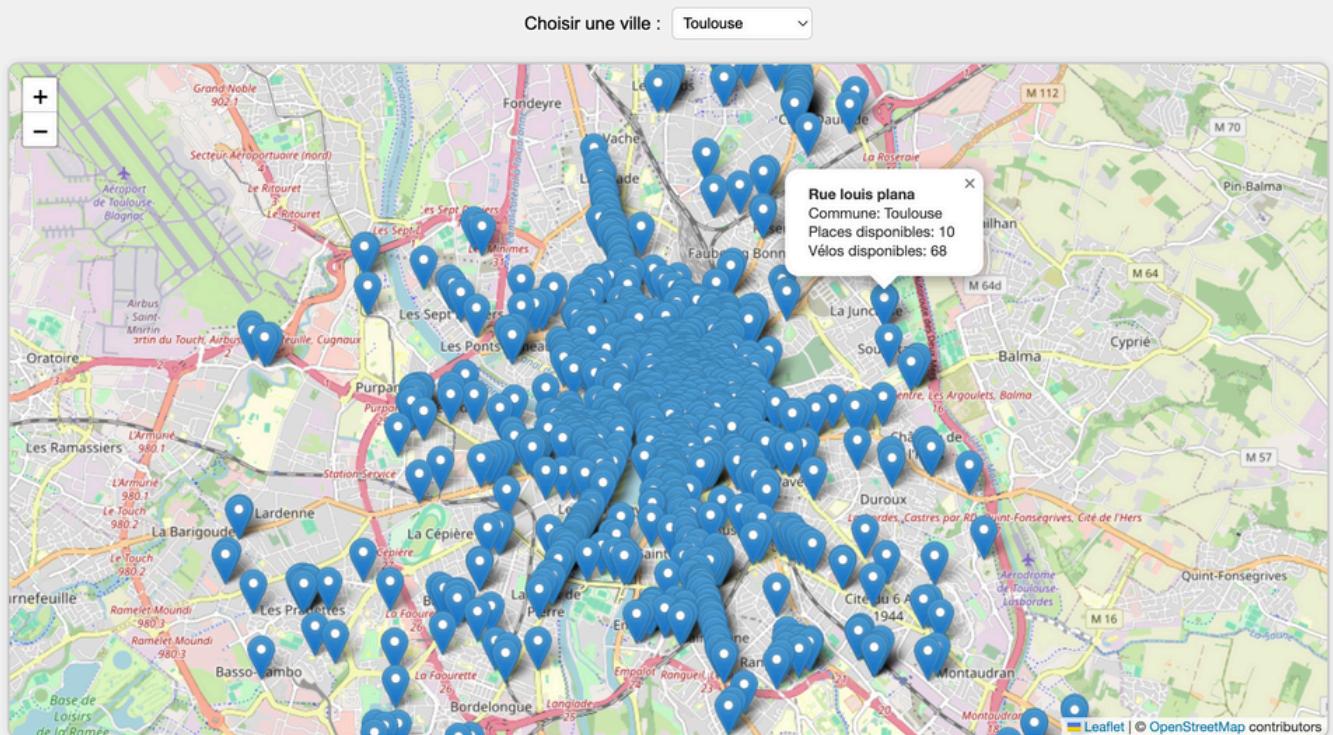
Serveur : localhost:8889 > Base de données : velo > Table : velo

nom_station	commune	nb_places_dispo	nb_velos_dispo	latitude	longitude
Porte des postes	Lille	20	18	3.049251	50.61856
Place colette besson	Lille	21	1	3.063799	50.62276
Cormontaigne	Lille	28	8	3.039911	50.62624
Jb lebas	Lille	19	21	3.068689	50.62482
Rue d'arras	Lille	19	6	3.067004	50.62224
Place deloit	Lille	6	18	3.068387	50.61957
Porte de douai	Lille	6	20	3.07241	50.61843
Gare lille flandres	Lille	15	17	3.069354	50.636399
St sebastien	Lille	24	4	3.059376	50.647236
Faubourg d'arras	Lille	10	6	3.062195	50.614052
P.i. de douai	Lille	11	9	3.074952	50.612194
Rue de la gaite	Lille	0	0	3.091106	50.63665
Boulevard louis xiv	Lille	6	20	3.069797	50.62816
Rue de gand	Lille	15	4	3.067375	50.64297
Boulevard de metz	Lille	0	0	3.041299	50.619312
Bourse du travail	Lille	17	3	3.090898	50.626052
Artois	Lille	1	0	3.061198	50.62629
Gambetta utrecht	Lille	14	4	3.053711	50.629063
Belfort	Lille	1	0	3.077127	50.622223
Place philippe lebon	Lille	8	10	3.06209	50.62789
Parvis rotterdam	Lille	6	14	3.077478	50.637709
Louise de Bettignies	Lille	21	9	3.064974	50.64144
Chaudre rivière	Lille	10	5	3.0822274	50.634417
Angellier	Lille	9	17	3.065015	50.628544
Jeanne d'arc	Lille	14	6	3.064497	50.625597
Place aux bleuets	Lille	4	12	3.067873	50.64142
Metro gambetta	Lille	26	2	3.051729	50.62614
Port de lille	Lille	8	18	3.035681	50.63022
Hopital militaire	Lille	11	5	3.061687	50.63393
Marche de wazemmes	Lille	14	9	3.050173	50.627674
Place casquette	Lille	10	8	3.045321	50.62429
Université catholique	Lille	1	30	3.046134	50.632233
Jean sans neur	Lille	8	5	3.05831	50.632423
Console de requêtes SQL	Lille	0	14	3.023001	50.630596

7 Carte interactive

Carte Interactive des Stations de Vélo

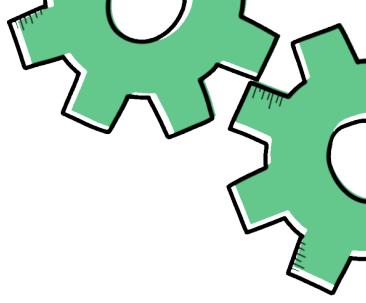
Prochaine mise à jour dans : 105 secondes



Pour la carte interactive, j'ai dû lier la base SQL avec PyMySQL et Flask, qui utilisent à la fois du code Python et du HTML. Elle permet d'afficher, par ville, les stations disponibles. Lorsque la souris passe sur une bulle, des informations sur la station s'affichent, avec en haut à droite une mise à jour automatique de la base qui rafraîchit la page pour actualiser les nouvelles données toutes les 5 minutes.



Pistes d'amélioration



- Ajouter des villes.
- L'API de la ville de Toulouse est mauvaise, en prendre une autre plus complète.
- Amélioration graphique de la carte
- Manque de données dans les API, en croiser avec d'autres.
- Mettre les prix en fonction des villes.
- Faire une carte interactive avec les API de Google Maps.



Conclusion

Pour conclure ce dossier, la création d'une carte interactive est difficile en raison des variations de qualité des API et des informations qu'elles contiennent. Le vrai défi de ce dossier est de créer une cohérence entre les informations.

Avec beaucoup plus de données, il aurait été possible de faire une carte bien plus fournie et, avec du temps, d'ajouter plein de fonctionnalités, comme dit précédemment, pour se rapprocher d'un Google Maps.



ELLIOT LAGAISE

PROJET vELO

