

DÉPARTEMENT DE MATHÉMATIQUES, D'INFORMATIQUE ET DE GÉNIE

**INF34515 – Projet en informatique I**

**Automne 2023**

**Professeur: Yacine Yaddaden**

**Présenté par**

**Guillaume Terrière**

[Introduction 3](#_Toc153101247)

[Donnée en temps réelles 4](#_Toc153101248)

[Données archives 5](#_Toc153101249)

[Mise en place d’une base de données 7](#_Toc153101250)

[Stockage dans la base de données 8](#_Toc153101251)

# Introduction

L'objectif de ce projet est d'utiliser des outils et des bibliothèques d'extraction de données pour récupérer et structurer des données à partir de différentes plateformes où les données relatives à la consommation énergétique au Canada sont publiées.  
  
J’ai choisi Énergie NB comme source de donnée.  
  
Ce projet a été réalisé en python avec les bibliothèques suivantes :

* bs4 (BeautifulSoup)
* requests
* selenium
* SQLAlchemy
* pymysql
* pandas
* numpy
* matplotlib

Toutes référencées dans le fichier requirements.txt.

L’IDE Visual Studio Code a été choisi ainsi qu’une base de données MySQL, et MySQL Workbench pour administrer celle-ci.

Enfin, le projet à été réalisé dans un dépôt GitHub (https://github.com/GuyomT/ConsoCA).

# Donnée en temps réelles

En premier lieu, afin de me familiariser avec BeautifulSoup, j’ai découvert comment parser une page HTML assez basique qui donne les données du réseau en temps réel.

Ci-dessous les données du 8 décembre 2023 18:05:18 Heure de l'Atlantique.

Une image contenant texte, capture d’écran, ligne, Police

Description générée automatiquement (<https://tso.nbpower.com/Public/fr/SystemInformation_realtime.asp>)

Afin de récupérer ces données, j’ai commencé par utiliser requests pour vérifier qu’une requête sur la page renvoi bien le code 200. Ensuite, après avoir récupérer l’html avec BeautifulSoup et l’avoir analysé, j’ai compris qu’il fallait que je récupère toutes les balises « <tr> », car les titres se trouvaient dans la 4e ligne et les valeurs dans la 5e.

Ensuite afin de traiter ces données, je les ai mises dans un dictionnaire titres/valeurs.  
N’ayant pas encore de base de données a ce stade du projet, j’en suis resté là.

# Données archives

Ensuite je me suis intéressé au deuxième lien contenant les données archivées du réseau.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

(https://tso.nbpower.com/Public/fr/system\_information\_archive.aspx)  
  
Cependant, ici il ne s’agissait plus seulement de parser l’html, mais d’avoir des interactions avec les boutons javascript de la page pour récupérer les données de chaque mois et de chaque années. J’ai donc du utilisé selenium qui permet d’intéragir de la sorte avec un page web.  
J’ai donc découvert comment créer un webdriver permettant à notre code python de lancer chrome et d’intéragir avec les boutons de la page. J’ai ensuite récupéré le nom des balises « <select> » que je devais utiliser grâce à l’inspecteur de mon navigateur.

Une image contenant texte, Appareils électroniques, capture d’écran, logiciel

Description générée automatiquement

Grâce aux outils de selenium (notamment la fonction find\_element()), j’ai pu créer une fonction qui, donnée une année et un mois récupérait le csv correspondant et le renvoyait sous la forme d’un DataFrame de la bibliothèque pandas. J’ai plus tard eu un bug sur cette fonction qui était lié au fait que lorsque j’envoyais le premier mois (1), je récupérais le 10e, j’ai pu régler cela en utilisant la classe Select de selenium au lieu d’utiliser la fonction send\_keys().

J’ai aussi ajouté une sécurité car certains mois ne sont pas encore présents notamment actuellement les mois de fin 2023.

J’ai alors créé une classe ArchivedNetworkData afin d’implémenter ma boucle qui me permettra de récupérer toutes les données de chaque mois de chaque année. J’ai créer deux fonction get\_min\_year() et get\_max\_year(), et j’ai pu réalisé une boucle, qui enregistrait chaque document dans un csv nommé archive\_{year}\_{month}.csv.  
Après, j’ai fait en sorte de ne pas réenregistrer un document qui existait déjà, puis j’ai agrégé les données par années afin de réduire le nombre de documents.

Maintenant que j’étais capable de récupérer toute ses données de manière organisées, il fallait les stocker.

# Mise en place d’une base de données

Ayant fait de nombreux projet en groupe, j’ai toujours laissé d’autres personnes s’occuper de la gestion des bases de données, et donc c’était la première fois que je ne gérais pas seulement les requêtes, mais aussi la mise en place et la gestion de celle-ci.

J’ai donc installé MySQL Community Server ainsi que MySQL Workbench sur mon ordinateur afin de découvrir comment cela fonctionnait.

J’ai utilisé l’ORM SQLAlchemy, qui est simple d’utilisation et bien documenté.

J’ai donc créé une table reprenant les données présente dans les document d’archive csv dans un fichier InitDatabase.py:

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

J’ai déclaré un Id en temps que primary key mais aussi l’heure comme unique car il n’est jamais censé y avoir deux données pour la même heure.

Afin de ne pas laisser de mot de passe dans le dépôt GitHub, j’ai utilisé un fichier config.ini contenant mon nom d’utilisateur et mon mot de passe de connexion à la base de données (ici en exemple root et password).

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

J’ai pu me connecter alors grâce à pymysql ainsi :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

# Stockage dans la base de données

Maintenant que ma base de donnée MySQL était fonctionnelle, j’ai pu intégrer des fonctions pour insérer mes données dedans dans mes précédentes classe.

Ainsi je récupère tout mes csv et les insère en vérifiant que les données archivées ne sont pas déjà présente, et j’insère les données en temps réelles qui ont les mêmes colonnes (que j’ai seulement du traduire en anglais).  
  
Et grâce à MySQL Workbench, nous pouvons voir que la table est remplie de toutes les données :

Une image contenant texte, logiciel, nombre, Icône d’ordinateur

Description générée automatiquement

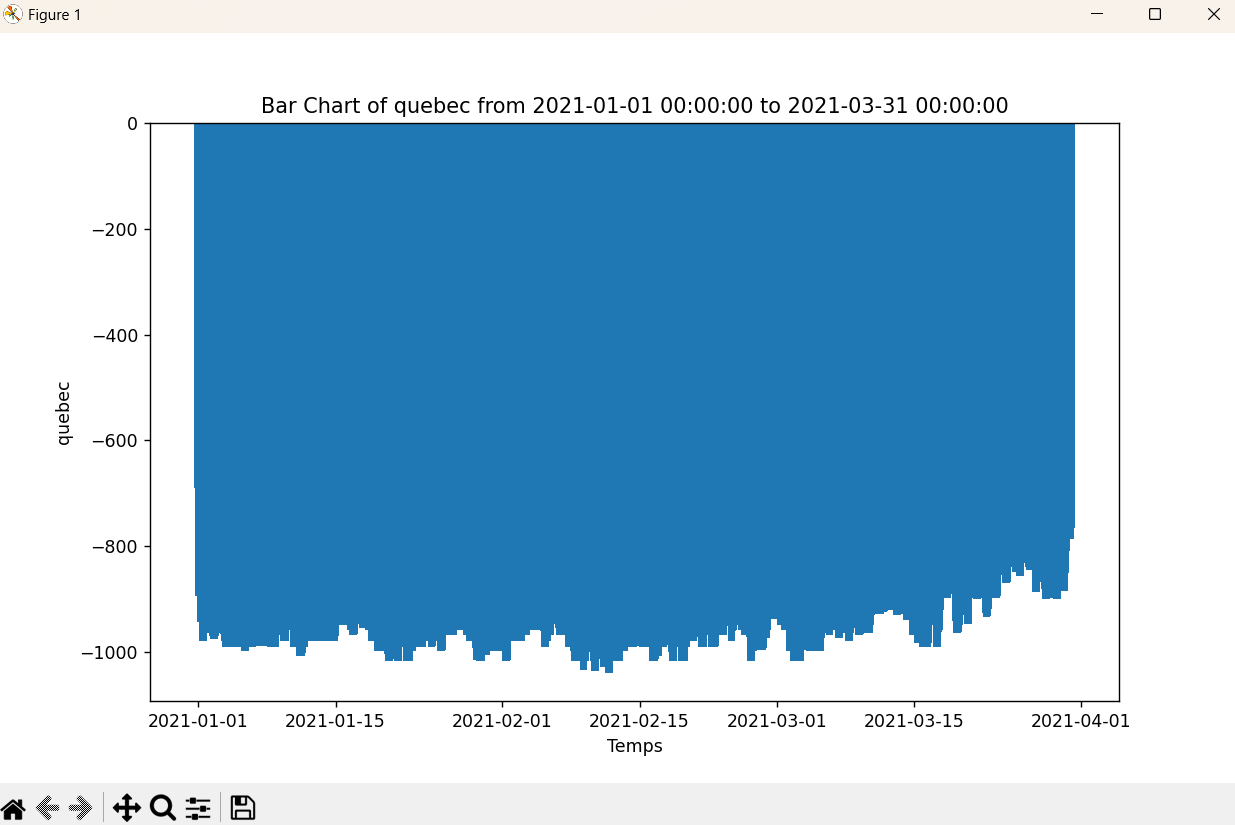
# Visualisation des données

Afin de visualiser les données, j’ai créé une classe PlotData qui, à l’aide de matplotlib, permet de visualiser les données dans un graphique. J’ai implémenté trois graphiques :

* Le premier affichant un champ de la base de données pendant une période

Une image contenant texte, capture d’écran, Tracé, ligne

Description générée automatiquement

* Le second faisant de même mais sous forme de diagramme à bandes
* Le dernier permettant de comparer deux annéesUne image contenant texte, capture d’écran, Tracé, diagramme

  Description générée automatiquement

Bien sur, il est possible de zoomer et de bouger la figure grâce a matplotlib afin de voir en détail ces graphiques. Il est assez simple d’ajouter un nouveau graphique en reprenant les fonctions existantes et modifiant celles-ci au besoin.

# Déploiement avec StreamLit

Afin de déployer le projet en utilisant le streamlit, j’ai du effectuer de nombreuses modifications au projet.  
  
Tout d’abord j’ai du ajouter un requirements.txt contenant les bibliothèques pythons nécessaire au projet. J’ai aussi dû ajouter un packages.txt afin d’installer chromium pour l’utilisation de selenium. Enfin les secrets sont gérés spécifiquement dans streamlit à l’aide d’un fichier .streamlit/secrets.toml (qui n’est évidemment pas pousser sur le dépôt).  
  
J’ai passé beaucoup de temps à découvrir streamlit et l’intégrer au projet, notamment pour la connexion à la base de données. Cependant, une fois toutes les étapes franchises, j’ai pu avoir une meilleure vision sur mes données :

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement  
  
Cependant, après de nombreux essais de déployer dans StreamLit Cloud, j’en suis arrivé à la conclusion que ce n’était pas possible sans une base de données hébergée elle-même ce qui n’est pas le cas pour ce projet étant en local seulement. J’ai donc abandonné cela malgré le temps passé.

# Conclusion du projet ConsoCA

Ce projet ma fait découvrir de nombreuses facettes du développement, tout d’abord le web scrapping avec beautifulsoup et selenium, ensuite le fait de gérer et déployer une base de données SQL, et se gestion avec un ORM (SQL Alchemy). Et enfin j’ai découvert la plateforme de déploiement StreamLit avec ses avantages et ses inconvénients. Je n’ai pas été convaincu par JupyterLab, je ne l’ai donc pas utilisé.

Au niveau de la gestion de projet, j’ai appliqué ce que j’ai appris durant mes nombreux projets à EPITECH, notamment le fait de travailler régulièrement sur le projet, de réaliser des « follow-up » toutes les semaines avec mon professeur référent Yacine Yaddaden afin de montrer les avancements et poser mes questions. J’ai utilisé GitHub simplement étant donné que j’étais seul sur le projet, je n’ai pas eu besoin de faire de pull requests ou de prévoir des issues. J’ai toujours documenté mon code et séparé mes fonctions dans des classes différentes. En effet ce projet pouvant potentiellement être retravaillé par la suite dans le cadre des recherches sur la consommation au Québec, il était important pour moi que le code soit lisible et compréhensible. J’ai aussi expliqué dans le README.md comment lancer le projet, et installer les dépendances.

J’ai choisi d’accorder plus de temps à StreamLit qu’au parsing des deux derniers liens, car ceux-ci me semblait mois important.

Dans l’ensemble, j’ai apprécié faire ce projet, et j’espère qu’il pourra servir à quelque chose de concret à l’avenir.