

SAE 2.04

Exploitation d’une base donnée



*Sommaire*

Table des matières

[Introduction 2](#_Toc136868424)

[statique 2](#_Toc136868425)

[DYNAMIQUE 4](#_Toc136868426)

[Carte 6](#_Toc136868427)

[Analyse 7](#_Toc136868428)

[Conclusion 9](#_Toc136868429)

[Analyse-critique 10](#_Toc136868430)

# Introduction

Notre projet consiste à créer une solution permettant de récupérer et de visualiser des données à partir d'un site web. Pour ce faire, nous nous concentrons sur les données liées aux vélibes.

Nous allons tout d'abord collecter deux types de données : les données statiques et les données dynamiques. Les données statiques comprennent des informations telles que l'emplacement des stations de vélibes, les coordonnées géographiques, les adresses, et d'autres caractéristiques qui restent inchangées dans le temps. Les données dynamiques, quant à elles, comprennent des informations qui évoluent régulièrement, telles que le nombre de vélos disponibles dans chaque station, le nombre de places libres, et d'autres informations en temps réel.

Pour stocker ces données, nous allons créer une base de données dans WampServer que nous appellerons "velibes". Cette base de données comportera deux tables distinctes : une table statique et une table dynamique. La table statique contiendra les informations stables, tandis que la table dynamique sera régulièrement mise à jour avec de nouvelles données toutes les 30 minutes pendant une période de deux heures.

Une fois les données stockées dans notre base de données, nous créerons une interface utilisateur conviviale pour visualiser les informations. Nous développerons une vue qui affichera les données des vélibes sous forme de tableaux, permettant aux utilisateurs d'explorer les détails des stations, telles que leur emplacement, les caractéristiques spécifiques et les disponibilités actuelles.

Nous irons encore plus loin en créant des graphiques interactifs qui illustreront les tendances et les variations des données dynamiques au fil du temps. Cela permettra aux utilisateurs de mieux comprendre les fluctuations de la disponibilité des vélos dans différentes stations à différentes heures de la journée.

Enfin, nous intégrerons une carte interactive dans notre application, affichant les emplacements des stations de vélibes dans la région de l'Île-de-France. Cette carte sera mise à jour toutes les 30 minutes pour refléter les informations les plus récentes sur la disponibilité des vélos et les places libres dans chaque station.

# statique

Pour insérer des données statiques à partir d'un fichier JSON dans une base de données MySQL, je dois d'abord importer le module mysql.connector. Ensuite, je crée une variable appelée connection pour établir une connexion avec ma base de données WampServer, en spécifiant la base de données à laquelle je souhaite me connecter. J'ai également besoin d'une variable appelée basedb qui agira comme un curseur.

Ensuite, je crée une fonction nommée creer sans paramètres. À l'intérieur de cette fonction, je crée une table appelée station\_information\_static avec les colonnes suivantes : stationcode (TEXT), name (TEXT), capacity (INT), longitude (DOUBLE), latitude (DOUBLE). Je crée une variable sql qui servira à insérer les données.

Ensuite, j'ouvre mon fichier JSON et je stocke toutes les informations dans la variable data. En utilisant une boucle, je parcours chaque valeur dans data en créant une variable valeur pour récupérer chaque donnée lors de chaque itération. Ensuite, je l'insère dans la base de données. Après cela, j'affiche un message indiquant que la mise à jour a été effectuée avec succès en utilisant la fonction print.

En dehors de la fonction, j'appelle la fonction creer pour l'exécuter et je m'assure que les données ont été correctement insérées.

# DYNAMIQUE

Tout d'abord, j'importe les modules nécessaires tels que pandas, mysql.connector, time, ainsi que certains fichiers personnalisés appelés carte.py et analyse.py.

Ensuite, je crée une variable appelée "connection" pour établir une connexion avec ma base de données WampServer, en spécifiant la base de données à laquelle je souhaite me connecter. J'ai également besoin d'une variable appelée "basedb" qui agira comme un curseur pour exécuter des requêtes sur la base de données.

Ensuite, je crée une table appelée "station\_information\_dynamique" qui va contenir les colonnes suivantes : "stationcode" de type TEXT, "is\_installed" de type TEXT, "numdocksavailable" de type INT(64), "numbikesavailable" de type INT(64), "mechanical" de type INT(64), "ebike" de type INT(64), "nom\_arrondissement\_communes" de type TEXT, et "Date" de type datetime. Cette table va stocker les informations dynamiques sur les stations Velib.

Ensuite, je crée une deuxième table appelée "history\_change" avec les colonnes suivantes : "id" de type INT(64) avec AUTO\_INCREMENT et PRIMARY KEY, "table\_name" de type TEXT, "action" de type TEXT, et "action\_date" de type datetime. Cette table sera utilisée pour enregistrer l'historique des modifications effectuées dans la table "station\_information\_dynamique".

Ensuite, j'ajoute des déclencheurs (triggers). Le premier déclencheur, appelé "check\_is\_installed", vérifie si la valeur de "is\_installed" dans la table "station\_information\_dynamique" est 'OUI' et la modifie en '1' si c'est le cas, sinon en '0'.

Le déclencheur "log\_history\_change" enregistre les actions d'insertion dans la table "station\_information\_dynamique" dans la table "history\_change" avec la date et l'heure actuelles.

Le déclencheur "log\_history\_change\_after\_update" enregistre les actions de mise à jour dans la table "station\_information\_dynamique" dans la table "history\_change" avec la date et l'heure actuelles.

Le déclencheur "log\_history\_change\_after\_delete" enregistre les actions de suppression dans la table "station\_information\_dynamique" dans la table "history\_change" avec la date et l'heure actuelles.

Ensuite, j'utilise la bibliothèque pandas pour lire les données à partir d'une source JSON. Dans ce cas, les données sont récupérées à partir de l'URL "<https://opendata.paris.fr/api/explore/v2.1/catalog/datasets/velib-disponibilite-en-temps-reel/exports/json?lang=fr&timezone=Europe%2FBerlin>". Ces données contiennent les informations en temps réel sur la disponibilité des vélos (Velibs) à Paris.

Ensuite, j'exécute une requête SQL pour créer la table "station\_information\_dynamique" si elle n'existe pas déjà. Cette table a des colonnes telles que "stationcode", "is\_installed", "numdocksavailable", "numbikesavailable", "mechanical", "ebike", "nom\_arrondissement\_communes" et "Date", qui correspondent aux différentes informations sur les stations Velib.

J'exécute également une requête SQL pour créer la table "history\_change" si elle n'existe pas déjà. Cette table est utilisée pour enregistrer l'historique des modifications effectuées dans la table "station\_information\_dynamique".

Ensuite, j'utilise une boucle "for" pour parcourir les valeurs des données récupérées à partir du JSON. J'insère ensuite ces valeurs dans la table "station\_information\_dynamique" en utilisant une requête SQL d'insertion. La date et l'heure actuelles sont ajoutées automatiquement à chaque enregistrement grâce à la fonction NOW() dans la requête SQL.

Après l'insertion des données, j'exécute une requête SQL supplémentaire pour récupérer les données les plus récentes de la table "station\_information\_dynamique", en joignant les informations statiques des stations depuis la table "station\_information\_static". Ces données sont ensuite stockées dans la variable "resultat".

Ensuite, j'appelle différentes fonctions pour effectuer des analyses et des visualisations des données. Par exemple, la fonction "carte(resultat)" utilise les données récupérées pour afficher une carte avec la localisation des stations Velib. D'autres fonctions telles que "ver(resultat)", "ver2(resultat)", "ver3(resultat)", "circuler2(resultat)" sont également appelées pour effectuer des analyses spécifiques.

Enfin, j'introduis un délai de 2 secondes en utilisant la fonction "time.sleep(2)" pour attendre avant de répéter le processus d'insertion des données.

En résumé, j'ai expliqué les différentes étapes de la fonction "insertion()" en détail, en décrivant les importations des modules, l'établissement de la connexion à la base de données, la création des tables, l'insertion des données, les analyses et visualisations, ainsi que le délai entre les itérations du processus.

# Carte

J'importe les modules nécessaires pour la génération de cartes interactives (notamment le module "folium") ainsi que le module "json" pour manipuler les données au format JSON. J'importe également des plugins supplémentaires de Folium.

Je définis une fonction appelée "carte" qui prend un argument "resultat". Cette fonction sera utilisée pour générer la carte des stations de vélos.

Je crée une carte Folium en utilisant la fonction "Map" du module Folium. La carte est centrée sur Paris avec les coordonnées [48.8566, 2.3522] et un niveau de zoom initial de 10.

J'ajoute des fonctionnalités à la carte en incorporant deux plugins de Folium : MiniMap et Geocoder. Cela permet d'inclure une minimap et la possibilité de géocoder des adresses.

J'intègre une fonction interne appelée "arrondisemment" qui sera utilisée pour ajouter les arrondissements à la carte.

Je charge les données des arrondissements à partir du fichier JSON "arrondissements.json" et les stocke dans la variable "data".

J'ajoute les arrondissements à la carte en utilisant les coordonnées géométriques de chaque arrondissement pour créer une couche de données géospatiales. Cette couche est ensuite ajoutée à la couche des marqueurs d'arrondissements.

J'utilise une boucle "for" pour itérer à travers les éléments de la variable "resultat", qui semble être une liste contenant des données sur les stations de vélos. À chaque itération, je crée des marqueurs pour chaque station en fonction de ses caractéristiques (disponibilité, capacité, etc.) et les ajoute aux groupes de marqueurs appropriés en fonction de leur disponibilité.

J'ajoute les groupes de marqueurs à la carte à l'aide de la méthode "add\_child".

J'ajoute un contrôle de couches à la carte, permettant à l'utilisateur de sélectionner les groupes de marqueurs à afficher.

Je crée une légende HTML personnalisée pour les icônes des marqueurs, fournissant une description pour chaque couleur utilisée.

J'ajoute la légende HTML à la carte en utilisant la méthode "add\_child" de l'objet "html" de la racine de la carte.

Enfin, je sauvegarde la carte générée dans un fichier HTML nommé "carte\_velib.html" en utilisant la méthode "save" de la carte.

*Une image contenant texte, capture d’écran, carte

Description générée automatiquement*

# Analyse

j'ai utilisé Python et les bibliothèques matplotlib et mysql.connector pour effectuer des analyses sur des données de vélos en libre-service (Velibs).

J'ai d'abord établi une connexion à la base de données MySQL "velibs" en utilisant les informations de connexion appropriées, telles que l'hôte, le nom d'utilisateur, le mot de passe et le nom de la base de données. Cela m'a permis d'accéder aux données nécessaires pour effectuer les analyses.

Ensuite, j'ai défini une fonction appelée "cree()" qui crée une vue dans la base de données. La vue créée, nommée "stats\_sur\_periode\_donnée", est le résultat d'une requête SQL qui sélectionne les données pertinentes à partir des tables "statique" et "dynamique" et les lie en utilisant une jointure. Les colonnes sélectionnées comprennent des informations telles que le code de la station, le nom de la station, la capacité, la géolocalisation, le nombre de bornes disponibles, et d'autres détails utiles.

Ensuite, j'ai développé une fonction appelée "capacite(resultat, ville, typevelo)" qui permet à l'utilisateur de spécifier une ville et un type de vélo (mécanique ou électrique). En fonction des paramètres fournis, la fonction calcule le nombre de places libres et de vélos disponibles pour le type de vélo spécifié dans la ville spécifiée. J'ai utilisé les résultats obtenus pour créer un graphique circulaire à l'aide de la bibliothèque matplotlib. Ce graphique affiche les pourcentages de places libres et de vélos mécaniques ou électriques disponibles, avec le nom de la ville spécifiée en tant que titre.

En outre, j'ai créé une fonction nommée "historique(resultat2)" qui analyse les résultats pour compter la capacité des stations pour chaque heure. En parcourant les données, j'ai stocké les heures et les capacités correspondantes dans des listes. En utilisant ces informations, j'ai généré un graphique en barres avec matplotlib pour visualiser l'évolution de la capacité des stations au fil du temps. L'axe des abscisses représente les heures et l'axe des ordonnées indique la capacité. Le titre du graphique est personnalisé en fonction de la nature de l'analyse effectuée.

Enfin, j'ai implémenté la fonction "circuler2(resultat, commune\_specifique)" qui permet à l'utilisateur de spécifier une commune spécifique. La fonction extrait les données de cette commune, notamment le nombre de vélos mécaniques et électriques disponibles. J'ai utilisé ces informations pour calculer les pourcentages respectifs de vélos mécaniques et électriques par rapport au total des vélos disponibles dans la commune spécifiée. À l'aide de la bibliothèque matplotlib, j'ai créé un diagramme circulaire qui représente visuellement la répartition des vélos. Ce diagramme affiche les pourcentages de vélos mécaniques et électriques, et le titre est personnalisé en fonction de la commune spécifiée.

Ainsi, en utilisant ces fonctions, il est possible d'effectuer différentes analyses et visualisations des données de vélos en libre-service à partir de la base de données "velibs".

Une image contenant Caractère coloré, diagramme, Graphique, capture d’écran

Description générée automatiquement

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

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

# Conclusion

En conclusion, ce projet de récupération et de visualisation des données liées aux vélibs à partir d'un site web a été une expérience enrichissante. J'ai pu mettre en pratique mes compétences en développement et en gestion de bases de données pour créer une solution efficace. En utilisant une base de données MySQL, j'ai pu stocker et organiser les informations statiques et dynamiques des stations Velib, permettant ainsi un accès rapide et une mise à jour en temps réel des données.

Grâce à l'importation des données statiques à partir d'un fichier JSON et à l'utilisation de requêtes SQL, j'ai pu insérer les informations statiques dans la base de données de manière automatisée. De plus, en récupérant les données dynamiques depuis une source JSON en temps réel, j'ai pu fournir aux utilisateurs des informations actualisées sur la disponibilité des vélos dans chaque station.

Ce projet m'a également permis d'explorer des fonctionnalités avancées telles que les déclencheurs et l'historique des modifications dans la base de données. Ces fonctionnalités ajoutent une couche de sécurité et de traçabilité aux données, garantissant ainsi leur intégrité et permettant d'analyser les changements survenus au fil du temps.

En somme, cette solution de récupération et de visualisation des données liées aux vélibs offre une expérience utilisateur améliorée et facilite l'accès aux informations essentielles pour les utilisateurs de l'application. Ce projet m'a permis d'approfondir mes connaissances en développement web et en gestion de bases de données, renforçant ainsi mes compétences techniques dans ces domaines.

# Analyse-critique

Comme j'étais légèrement en avance, j'aurais dû profiter de ce temps pour travailler sur les aspects complémentaires de mon projet. J'aurais pu organiser mon code de manière plus efficace en le divisant en plusieurs modules distincts, ce qui aurait facilité sa maintenance et sa compréhension. De plus, j'aurais dû ajouter des commentaires pour expliquer clairement chaque partie du code, permettant ainsi à d'autres développeurs (et à moi-même) de le comprendre plus facilement à l'avenir.