DA Python Projet 5

Données publiques de l'OpenFoodFacts

Lien vers Github : https://github.com/ridialass/PureBeurre_p5
Lien vers Trello : https://trello.com/b/ZdiUeqiC

Introduction :

Ce projet est développé dans le cadre de la formation Openclassrooms "Développeur d'application Python". Il a pour but de se familiariser avec l'utilisation d'API REST et MySQL. L'API utilisée dans ce projet est celle du site Open Food Facts qui compile des aliments et leurs propriétés nutritionnelles.

Algorithme :

L'algorithme pour ce programme se divise en 2 parties, la première est de s'occuper de la base de données, imaginez sa structure en analysant les données contenant les informations sur les aliments. Une fois la structure réalisée, il reste à penser à l'algorithme pour trier les données et sélectionner uniquement celles demandées et les afficher à l'utilisateur.

La première étape de ce projet a été d'imaginer la création de la base de données en identifiant les tables nécessaires. Le script de création (db_creation.py) dispose d'une fonction de suppression si une base de données ayant le même nom existe déjà, et d'une autre fonction de nettoyage de tables pour les cas où le module est lancé plusieurs fois successivement dans le processus de rédaction de code. L'utilisateur final n'a besoin de lancer ce module qu'une seule fois. Il en va de même pour le module de remplissage de la base de données (db_populate_tables.py), il contient une fonction de nettoyage des tables s'il est lancé plusieurs fois.

La deuxième étape concerne le client qui interagira avec le serveur. Avec la base de données remplie avec du code python et en utilisant la bibliothèque « Records » et le connecteur « mysql-connector », il fallait songer à l'algorithme de l'interaction utilisateur avec la base de données. D'où le module (request.py) qui se connecte au serveur et exécute les requêtes et le module (client.py) qui fait office d'interface utilisateur.

Les difficultés rencontrées :

_ **Conception de la base de données :** les liens entre les tables de la base de données fut un peu flou sur les cardinalités.

_ Une partie des **requêtes SQL** concernant plusieurs tables avec les « JOIN »s. A force de réfléchir, la solution est venue toute seule. Trouver comment construire la requête de manière efficace pour le remplissage de la base de données grâce à l'API, opération sur

laquelle je suis resté bloqué un certain moment. Pour m'aider dans cette tâche, je me suis servi de Postman afin de vérifier comment étaient agencées les données de l'API.

_ Quelques complications mineures sur la **mise en mémoire de certaines valeurs** sélectionnées par l'utilisateur. Un exemple : le choix du produit à remplacer pour l'enregistrement dans la table « favoris ».
La difficulté majeure était un souci avec Wamp-server sous Windows 10. Avec les dépendances avec Visual-Studio, il fallait installer les VC 2008, 2010, 2012, 2013 et 2017.

Une fois tous ces points complétés et résolus, l'application est testée une dernière fois pour vérifier que toutes les fonctionnalités marchent correctement.
En plus des modules rédigés en python, le projet contient ce document, le lien vers le code stocké sur Github, et un modèle physique de données représentant l'architecture de la base de données.

En conclusion, ce projet m'a été plus instructif que les précédents. Tout d'abord, parce que c'était la première fois que je devais interagir avec une API et aussi une base de données sous python. J'ai eu au début du mal à trouver la documentation nécessaire mais avec l'aide de mon mentor et de mon collègue « Google », j'ai fini par trouver des solutions.