Démarche du projet 5 : Utilisez les données publiques de l'Open Food Facts

Ce document a pour but d'expliquer la démarche et l'algorithme que j'ai utilisé pour réaliser le projet 5 de la formation OpenClassRooms développeur d'application Python.

Voici le lien vers mon code source : https://github.com/micktymoon/P5_pelletier_celine.git

Pour commencer, j'ai décidé de créer une ébauche de la base de données, qui sera utilisée par le programme pour stocker les données. Pour cela, j'ai utilisé, MySQL, qui est un système de gestion de base de données relationnelles. Cette ébauche était composée de trois tables.

La table « Product » : qui contient les produits proposés par le programme. Elle est constituée de huit colonnes :

- « id » : qui correspond à un nombre auto-incrémenté à chaque ajout de produit. Il sera notre clé primaire permettant d'identifier chaque ligne de la table et ne pourra pas être nul.
- « name_product » : qui correspond au nom du produit. Il ne pourra pas être nul.
- « brand » : qui correspond à la marque du produit.
- « id_category » : qui correspond à l'ID de la catégorie du produit.
- « nutri_score » : qui correspond au nutri-score du produit.
- « store » : qui correspond aux magasins dans lesquels on peut acheter le produit.
- « ingredient » : qui correspond aux ingrédients utilisés pour fabriquer le produit.
- « url » : qui correspond au lien redirigeant vers la page d'Open Food Facts concernant ce produit.

La table « Category » : qui contient les différentes catégories des produits proposés. Elle est constituée de deux colonnes :

- « id » : qui correspond à un nombre auto-incrémenté à chaque ajout de catégorie. Il sera notre clé primaire et ne pourra pas être nul.
- « name_cat » : qui correspond au nom de la catégorie. Il ne pourra pas être nul.

Et enfin la table « Substitute » : qui contient les différents substituts sauvegardés des produits. Elle est constituée de deux colonnes :

- « id_product » : qui correspond à l'ID du produit à substituer et il ne peut pas être nul.
- « id_product_substitute » : qui correspond à l'ID du produit proposé comme substitut et il ne pourra pas non plus être nul.

Une fois la base de données créée, il fallait que je trouve un moyen de pouvoir la gérer en utilisant le langage Python. Après quelques recherches, j'ai trouvé le pilote de base de données, MySQL Connector / Python.

Une fois installé, j'ai créé une classe qui me permet de gérer la connexion au serveur MySQL et d'exécuter les requêtes MySQL. Ainsi qu'une classe me permettant de gérer la base de donnée entière.

J'ai par la suite créé une classe pour chaque tables de la base de données afin de les gérer. J'ai aussi créé une classe « ApiManager » me permettant de gérer les recherches de produit dans l'API d'Open Food Facts.

Puis j'ai commencé à entrer des données dans la base. Mais j'ai rencontré le problème suivant, les catégories et magasins que je récupérais de l'API étaient sous forme d'un texte et pour chaque produit j'avais plusieurs catégories et plusieurs magasins. J'ai donc décidé dans un premier temps de transformer chaque texte, en liste, afin de pouvoir différencier chaque élément du texte. Pour

cela j’ai trouvé le module « .split » qui transforme un texte en liste en lui mettant en paramètre le séparateur de chaque élément, pour moi ce séparateur était une virgule. Mais je me suis rendu compte que certains mots étaient précédés d’un espace, j’ai donc utilisé le module « .strip » afin de supprimer les espaces avant et après chaque mot.

Voilà, j’avais donc pour chaque produit une liste de catégories et une liste de magasins. Mais comment relier un produit à plusieurs catégories ? La colonne « id_category » ne pouvant contenir qu’un seul nombre, j’ai décidé de supprimer de la table « Product » la colonne « id_category » et de la remplacer par la création d’une table intermédiaire afin de relier chaque produit à plusieurs catégories. Cette tables est donc composée de deux colonnes :

- « id_product » : qui correspond à l’ID du produit auquel on veut associer une catégorie.
- « id_cat » : qui correspond à l’ID de la catégorie à associer.

J’ai procédé de la même manière pour les magasins.

Pour les substituts, j’ai décidé qu’un produit était un substitut potentiel, si il possédait une des catégories du produit à substituer et si son nutri-score était inférieur à celui du produit à substituer. Pour les rechercher, j’ai décidé de faire une première recherche dans la base de donnée et si cette recherche n’aboutit pas, alors un recherche dans l’API est faite. Et enfin, si la recherche dans l’API n’aboutit pas non plus, un message disant qu’aucun substitut n’a pu être trouvé est affiché.

Une fois mes classes créer, j’ai réalisé un script de création et de remplissage de la base de données. Pour cela j’ai créer plusieurs fonctions, une créant la base, une remplissant une première liste de catégories et enfin une remplissant la base avec des produits. Mais lorsque j’executais le script, j’obtenais l’erreur suivante : « mysql.connector.errors.internalerror: 1213 (40001): deadlock found when trying to get lock; try restarting transaction ». Je me suis rendu compte que cette erreur apparaissait lors de l’exécution de la fonction de remplissage de produit, j’ai donc utilisé pour chacune des actions de cette fonction un « try » avec pour « except » le fait de relancer l’action en cas d’une erreur provenant de « MySQL.connector ».

Une fois que tout cela fonctionnait correctement, je me suis lancée dans la création d’une classe gérant l’interface utilisateur. Celle-ci comprends des méthodes pour chacun des menus. Mais ces méthodes étaient bien trop compliquées, j’ai donc décidé de les séparer en plusieurs petites méthodes appelées au fur et à mesure des choix de l’utilisateur.

Une fois cette classe créée, j’ai créé un script qui exécute le programme.

Dans un premier temps il affiche le premier menu, proposant trois choix :

1. Quel aliment souhaitez vous remplacer ?
2. Retrouvez mes aliments substitués.
3. Quitter.

Si l’utilisateur fait le premier choix, alors le programme propose des catégories. L’utilisateur peut donc choisir une catégorie ou retourner au menu précédent. Si il choisit une catégorie, alors un choix de produit s’affiche et il peut soit en choisir un, soit retourner au menu précédent. Si il choisit un produit, alors le programme lui affiche ses substituts possibles et lui propose de les sauvegarder. Il peut donc choisir de les sauvegarder ou non.

Si l’utilisateur fait le deuxième choix, alors le programme lui affiche les produits substitués sauvegardés, il peut donc en choisir un ou retourner au menu précédent. Si l’utilisateur choisit un produit, alors le programme lui affiche les substituts de ce produit, puis retourne au menu principal.

Si l’utilisateur choisit le troisième choix, alors le programme s’arrête.