PARCOURS OPENCLASSROOMS DÉVELOPPEUR D'APPLICATIONS PYTHON

QUENTIN LATHIÈRE

Projet 5: Utilisez les données publiques de l'OpenFoodFacts

https://github.com/Synkied/OC_Projet-5

Le présent document résume la réalisation du 5e projet (Utilisez les données publiques de l'OpenFoodFacts) dans le cadre de mon parcours Développeur d'application - Python.
Ce document recense les grandes lignes de la réalisation du projet. La réflexion pré-projet, la réalisation du projet et les réflexions post-projet.

Définition du projet

Description

Ce projet consiste à créer une application permettant à ses utilisateurs de chercher un produit et d'en trouver un équivalent plus sain.

Fonctionnalités

- Utiliser Python 3 ;
- Créer une base de données MySQL ;
- Utiliser la bae OpenFoodFacts ;
- Faire interagir l'utilisateur dans un terminal (programme CLI).

Contraintes

- Versionner son code en utilisant Git et le publier sur Github ;
- Développer dans un environnement virtuel en utilisant Python 3 ;
- Respecter les bonnes pratiques de la PEP 8 ;
- Code écrit en anglais : nom des variables, commentaires, fonctions...

Pré-projet

Environnement de développement

- Machine sous Windows 10. Environnement virtuel via virtualenv ;
- Sublime Text 3 avec plusieurs plugins (Principaux : Anaconda, GitGutter) ;
- Tests sur un MacBook Pro 2017 sous MacOS X Sierra. Virtualenv

Anticipation des problèmes

- Apprendre à utiliser l'API OpenFoodFacts ou la gestion de fichiers CSV ;
- Structurer les classes et leurs interactions ;
- Structurer la base de données ;
- Utiliser un ORM ;
- Utiliser plusieurs nouvelles librairies ;
- Préparer le projet en DDD (Doc Driven Dev).

Première difficulté rencontrée

- Penser la base de données. Plusieurs itérations ont été nécessaires.

Solution apportée

- Itérer plusieurs fois sur le modèle de base de données, en me renseignant bien sur les meilleures pratiques. Notamment, l'emploi de relations **many-to-many** et de table de jonctions.

Mise en place du projet

Démarche choisie

Comme demandé, j'ai d'abord écris les démarches que je pensais effectuer pour ce projet, et l'ordre dans lesquels je pensais les réaliser. Pour me rappeler des tâches à effectuer, j'ai créé un

Taiga et me suis fait des User Stories.

J'ai ensuite décidé de créer le fichier de création de base de données (.sql) à la main, mais me suis vite rendu compte que *MySQLWorkBench* permettait de créer des modèles de base de données graphiques exportables en fichier .sql.
Je me suis donc rabattu sur cette solution de facilité et ai pu me consacrer de manière plus intense à réfléchir au modèle de base de données.

Ensuite, je pensais faire un programme relativement simple en ligne de commande et faire des **injections SQL** dans du code Python. Mais je me suis renseigné et ai finalement décidé d'utiliser un ORM. Après en avoir testé plusieurs, j'ai décidé d'utiliser **Peewee**.

De fil en aiguille, j'ai itéré sur le projet et ai finalement décidé de créer un script « install.py » qui permet à l'utilisateur de gérer les actions autour de la mise en place de l'application (téléchargement et nettoyage du CSV, création de la base de données...) et un script « menu.py » permettant à l'utilisateur d'utiliser facilement l'application et voir les différents produits de la base et se voir proposer des substituts.

J'ai aussi fait appel à plusieurs librairies Python tierces : *tqdm, colorama, termcolor, terminaltables*, afin de rendre le programme plus agréable à utiliser au sein du terminal.

Je souhaitais que le projet soit le plus agréable possible à utiliser et le moins difficile à mettre en place pour l'utilisateur. Bien sûr, il reste des étapes un peu complexes, comme la création d'un environnement virtuel et l'installation des librairies dépendantes, mais j'ai bien l'intention de rendre mon projet accessible au plus grand nombre après mon parcours OpenClassrooms, car je trouve la démarche très intéressante.

Pourquoi l'utilisation du CSV plutôt que l'API ?

J'ai décidé d'utiliser les données fournies en **CSV** par OpenFoodFacts, car leur API est **expérimentale** et que récupérer les produits était complexe et très long à cause, notamment, de la pagination.
Je pense que l'utilisation du fichier CSV est une solution plus **pérenne** (tant que l'API n'est pas stable), en sachant que j'ai l'intention de **continuer ce projet une fois mon parcours terminé**.

Analyse d'algorithmes

Affichage d'un produit plus sain (substitut)

L'algorithme de la méthode *display_better_product(self, prod_choice)* prend en paramètre le produit choisi par l'utilisateur, qui est simplement un entier (*int*) représentant l'id d'un produit dans la base de données. Via la méthode *Products.get()* de l'ORM Peewee, l'instance (la ligne) correspondant au produit est alors stockée dans la variable *cur_product* sous forme d'objet *Products* (la classe correspondant à la table *Products* dans la base de données).
Finalement, pour trouver un substitut plus sain, l'algorithme vérifie le nutri_grade actuel du produit (fourni par OpenFoodFacts).
Si ce nutri_grade est de « a » (le meilleur possible), alors le programme propose un produit avec un nutri_grade équivalent.
Sinon, il propose toujours un produit avec un nutri_grade meilleur que celui du produit actuel.

Pour se faire, l'algorithme fait tout simplement appel aux méthodes **__eq__()** ou **__lt__()** de la classe *Str*, afin de comparer les lettres du nutri_grade (préalablement passée en *lowercase*, lors de la création du fichier CSV utilisé pour la base de données).

Difficultés
Concrètement, cet algorithme n'a pas été dur à mettre en place, mais toute la partie sous-jacente l'a été.
En effet, je n'avais jamais utilisé un ORM auparavant et ne comprenait pas bien la notion d'objet qui gravitait autour au début. Après avoir lu la documentation de Peewee et m'être beaucoup renseigné sur StackOverflow, j'ai pu comprendre la puissance et l'utilité d'un tel outil et c'est alors naturellement que j'ai pu créer cet algorithme.

Téléchargement du fichier CSV via le terminal
Cet algorithme m'a pris du temps à mettre en place.
En effet, je préférais maîtriser le téléchargement du fichier CSV et le proposer directement à l'utilisateur dans le terminal, afin que le fichier se place directement où je l'ai décidé et que la suite du programme soit aisée à mettre en place pour l'utilisateur.

Difficultés
Au premier abord, le téléchargement d'un fichier CSV via le terminal et un script Python n'est pas compliqué, grâce à des librairies comme Request, Urllib ou bien BeautifulSoup.
Cependant, le fichier CSV d'OpenFoodFacts pèse + d'1Go (et n'a cessé de prendre du poids entre le début de mon projet et au moment où j'écris ces lignes).
L'algorithme semblait donc à priori marcher, mais je ne savais pas combien de temps prendrait le téléchargement ni s'il marchait réellement, car aucune information n'était renvoyée par le terminal.
La vraie difficulté était donc là : afficher les informations de téléchargement au sein du terminal.

Je me suis donc mis à chercher une librairie Python qui proposait cela et suis tombé sur TQDM, une excellente librairie proposant d'afficher une barre de progression au sein d'un terminal pour toute sorte d'opération.

Par la suite, j'ai dû récupérer la taille totale du fichier à télécharger, accessible via *Request* et simplement écrire les données dans un nouveau fichier à l'aide de la clause *with* et la fonction *open*.