OPENCLASSROOMS

Etudiant: Ly Mickael | Mentor Validateur: Stéphane Nédélec

Soutenance du projet 5: Utilisez les données publiques de l'Openfoodfacts

Lien code source : https://github.com/Goro-Majima/ OpenFoodFacts

Pour le projet 5, l'objectif est de répondre à la problématique de l'entreprise Pur beurre. Elle souhaite que je développe une application qui recommanderait une alternative aux aliments du quotidien d'un client. Les données open source de l'Openfoodfacts seront exploitées pour proposer le substitut, basé sur leur nutrigrade.

Le projet est en programmation orienté objet et la méthode suivie est la méthode agile. Environnement virtuel créé et suivi des recommandations Pep8 (extension black, pylint)

Le challenge de ce projet est d'apprendre à exploiter les données d'une Api et de les parser sur une application connectée entre python et mysql. Le but étant de de se familiariser avec les requêtes sql, un ORM tel que SQLAlchemy ne sera pas nécessaire pour ce projet.

Fonctionnalités et documentations (Trello, Readme)

- Je définis d'abord les fonctionnalités du programme en suivant les besoins du cahier des charges. Je détermine les situations, les tâches et sous-tâches selon l'expérience utilisateur. J'anticipe les erreurs de saisie de l'utilisateur qui peuvent arriver.
- Toutes ces étapes seront enregistrées dans un organisateur de tâches comme Trello qui est un outil collaboratif qui permet de travailler selon la méthode agile.
- **Purebeurre.py** contient le script d'un algorithme plutôt simple avec une boucle évènementielle pour quitter ou revenir au menu et des inputs sous forme de question/réponse. Il faudra effectuer un contrôle des erreurs de valeur d'inputs en utilisant le bloc **Try Except**.

Création de la base de données (MPD, connexion.py, databaseinit.py)

- Je schématise la base de données avec l'application mysql **workbench** pour obtenir 3 tables pour la consultation et l'enregistrement des produits et favoris. J'enregistre le script sql dans **scriptstables.txt** pour insérer la base de données via la console de commandes.
- Après avoir connecté python et mysql avec la bibliothèque **mysql.connector**, je code le script qui autorise l'accès à la base. Je recherche les noms de catégories avec l'application Postman.
- Grâce à l'import de la bibliothèque « **Requests** », je peux récupérer et parser les données de l'API d'Openfoodfact via l'**url** de l'api en format JSON sous forme de listes de tuples (id, nom, nutrigrade, magasin, détails, url).

Consultation de la base de données (display.py)

- Je code ensuite le script qui intéragit sur le terminal avec l'utilisateur et l'application sous forme de questions-reponses en vérifiant le résultat avec le bloc **Try Except** qui répétera la question.

- Une requête SQL est stockée dans une variable qui sera exécutée par les fonctions de la bibliothèque mysql.connector. Son contenu pourra être affiché par une boucle for.
- L'utilisateur saisit l'id d'un produit et le consulte (choix par categorie, puis par liste de 50 produits) ainsi que son substitut au meilleur nutrigrade. Pour afficher un substitut différent s'il y en a plusieurs, j'utilise la fonction « **random** » afin de proposer aléatoirement un substitut d'une liste de « bons produits ».

Insertion du substitut dans la table substitut (display.py, insertfavorite.py)

- Il pourra enregistrer ce produit à ses favoris via sa **clé(id)** s'il le souhaite, ou revenir au menu.
- Le substitut sera affiché avec le produit remplacé (sa clé est aussi enregistrée dans la table substitut) dans l'ordre de la table substitut.
- Il se peut qu'aucun substitut ne soit trouvé car le produit sélectionné est déjà excellent. L'utilisateur retournera au menu ou l'application se fermera.

Difficultés rencontrées

- Organisation et découpage logique des classes et fonctions.
- Connection à python et la base de données
- Récupération des données de l'api qui seraient exploitables pour l'application.
- Affichage des substituts de manière cohérente.

Solutions aux difficultés

- Définition plus précise des tâches et sous tâches qui permettront de développer plus instinctivement les classes et fonctions nécessaires.
- Parcours de tutoriels pour suivre les étapes d'une connexion à une bdd mysql. Création d'un fichier connexion.py exportable pour tout fichier faisant appel à la bdd comme lors de consultations ou d'insertions.
- Ne pas oublier de « commit » lors de l'insertion pour s'assurer de l'enregistrement de la requête.
  Parcours des données de l'api en utilisant l'application « Postman ». Analyse plus organisée pour déterminer quelle catégorie a les informations les plus complètes.
- Amélioration de la fonction « substitutelist » pour afficher les éléments de la table susbstitute. Correction des bugs avec tests.