PROJET 5 : Utilisez les données publiques de l'OpenFoodFacts

Lien Github : https://github.com/preudh/P5_OpenFoodFacts-

Lien Trello : https://trello.com/b/go5lXF7T/p5ocpreudh

Démarche suivie :

1) Utilisation de l'outil Trello (Cf. dossier documentation ou lien):

- NFR (non fonctional requirements)
- Démarches
- Products backlog = users stories triées par priorité
- Sprint backlog = mettre les plus importantes en premier
- In progress
- Done

Difficultés rencontrées : évaluation de la charge de travail par sprint.

2) Esquisse d'un simple diagramme de flux :

Ce dessin me permet de clarifier le processus.

Difficultés rencontrées : à ce stade, je n'avais pas envisagé tous les flux. En codant, j'enrichi les flux, les exceptions. Cf. readme.txt pour le détail du processus.

3) Modèle de données avec l'outil Draw IO :

Objectifs : lister les classes et structures des tables.

Difficultés rencontrées :

- Déterminer les classes,
- Problème de cardinalité (1..*, *..1) avec création d'une table intermédiaire tab_assoc_product_sustitut entre tab_product et tab_substitut car un substitut peut être le substitut de plusieurs produits et un produit peut être substitué par plusieurs produits.

4) Installation de MySQL :

Après avoir installé MySQL directement, je préfère utiliser WampServer (Cf readme.txt) en quelques clics pour installer MySQL et phpMyAdmin. WampServer est un environnement comprenant le serveur MySQL, un interpréteur de script, ainsi que phpMyAdmin pour l'administration Web des bases MySQL

5) Utilitaire JSONView dans le navigateur :

Permet de visualiser les fichiers json et comprendre leur structure. Cet outil m'aide à créer les requêtes vers l'API de OFF

Consultation de la documentation de l'API

6) Création des tables via un script python :

Essais des scripts SQL et visualisation des résultats dans phpMyAdmin.

Puis scripts Python définitifs dans le module database.py

Création du modèle physique de données définitif en utilisant l'outil phpMyAdmin (Cf.dossier Documentation)

7) Organisation du programme selon l'architecture MVC (Modèle-vue-contrôleur) :

Difficultés rencontrées : l'utilisation du modèle MVC rend un peu plus compliquée le développement de l'application mais il permet une meilleure structuration de celle-ci. Création de 4 modules database.py, model.py, controller.py et view.py. Cela devrait faciliter ma compréhension future des frameworks type MVC

8) Conclusion et axes d'amélioration :

- Apprentissage de :
    - Base de données MySQL
    - API
    - Fichier json
    - Architecture MVC (Modèle-vue-contrôleur)
    - Code :
        - MySQL
        - Erreurs et Exceptions (Try…Except)
        - Renforcement des compétences sur les conditions (if, else…)
        - Renforcement des compétences POO

- Axes d'amélioration du présent projet :
    - Format de restitution dans la view
    - Interface graphique avec Tkinter au lieu de la console