P5_02_Démarche

I. Explication de la démarche:

L'essentiel de la démarche est simple.

D'abord, j'ai analysé le travail demandé en terme d'éléments, fonctionnalités, étapes et contraintes. Ensuite, en adoptant la programmation orientée objet, j'ai défini une modélisation possible du projet pour assurer une réponse fonctionnelle au *brief* presenté.

En effet, l'expérience utilisateur est bien détaillée dans le brief, ce qui m'a permis de voir clair l'utilité de l'application.

Une première étape était de bien analyser l'API *Open food facts* à savoir comment récupérer les data de l'API, sous quel format et suivant quelles requêtes.

A la lumière de cette analyse, et dans une démarche qui converge avec la raison d'être de l'application, j'ai choisi de présenter à l'utilisateur des produits malsains puis lui fournir un substituant sain ayant le plus de catégorie en commun avec sa sélection dans la liste des produits malsains.

Dans cette optique, le choix des catégories était fait d'une façon qu'elles contiennent suffisamment de produit sains et produits malsains. Le critère est le grade nutritionnel, qui suivant cette démarche est soit 'A', soit 'E'.

Une fois la liste des catégories est figée, le nombre de 1000 produits par catégories entre produits sains et malsains est assuré, j'ai commencé à mettre en place l'architecture modulaire capable de m'importer les données nécessaires, tout en étant conforme aux recommandation de l'API *Open food facts.*

Les données sont donc :

- Récupérées à l'aide du module *offparser.py* qui lui trouve les paramètres nécessaires dans le module *configuration.py*
- Nettoyées et mises sous le bon format dans le module *offcleaner.py, ce* qui me permettra de bien exploiter ces données
- Modélisées dans le module *models.py* pour assimiler les entités manipuler (produits, catégories, store) à des objets issus de classes afin de mieux les manipuler ainsi que contrôler leurs affichages.

Maintenant qu'on possède ces données, on passe à l'étape d'exploitation qui se définit comme suit :

- Préparer le terrain pour stocker les données en créant notre base de données vers laquelle l'utilisateur fera ses requêtes suivant *le brief* de départ
- Créer le modèle physique de données qui contrôle la définition des tables qui recevront les données ainsi que les relations entre ces tables et donc la nécessité ou pas de créer des tables d'association (chaque fois qu'on une relation plusieurs à plusieurs).

- Créer les tables, choisir les types de colonnes, les clés primaire et étrangère avec des commandes SQL, le tout dans un module python grâce à la librairie *mysql.connector*
- Créer les fonctions nécessaires à l'exploitation de notre base de données suivant notre objectif final décrit dans *le brief* de départ.
- Ce travail est la responsabilité du module *dbmanager.py* qui regroupe toute la gestion de la base de donnée puis du *dbbuilder.py* où on a regroupé exceptionnellement les appels des fonctions de création et remplissage des tables.

La dernière étape est de gérer les vues ou les menus qui seront présenter à l'utilisateur. C'est la responsabilité des modules *dbexplorer.py* et le module *application.py*

En effet, le module *dbexplorer.py* est simplement un conteneur qui tire juste les fonctions nécessaires du *dbmanager.py,* afin de répondre directement aux use cases demandés par *le brief.*

L'idée est d'avoir un module *application.py* où on a un menu par fonction qui elles se reposant sur des fonctions d'exploitation de la base de données regroupées toutes dans le module *dbexplorer.py*

Une dernière remarque importante est qu'on suppose que l'utilisateur ait déjà sa base de donnée crée et qu'il rentre ses identifiants en modifiant le module *dbidentifier.py* afin de garantir la connexion du script à la base de donnée de l'utilisateur.

En fin le module healthy_substitute.*py* est le point d'entrée qui lance l'application.

II. Les difficultés rencontrées:

a. Comment récupérer des données d'un API et comment écrire des requêtes *sql* via python

-->La question m'a mené à découvrir deux librairie python qui sont *requests* et *mysq.connector*.

Lire la documentation, les utiliser et résoudre les erreurs qui ont surgit a permis une bonne prise en main de ces librairies

b. Comment gérer et exploiter une base de donnée relationnelle --> Pour cela, je me suis documenté sur la création et la gestion des bases de données, les clients, les serveurs et spécialement mysql ainsi que les différentes fonctionnalités qui lui sont associées

c. Comment organiser mon travail en module -->Dans ce projet, c'était moins évident que le projet 3, donc, des cours et webinaire avait un plus-value important pour m'aider à voir clair comment gérer en module et surtout dans une démarche orienté objet l'ensemble du projet.

III. Le lien vers votre code source (sur GitHub)

https://github.com/BENBELGACEM-Bassem/P5