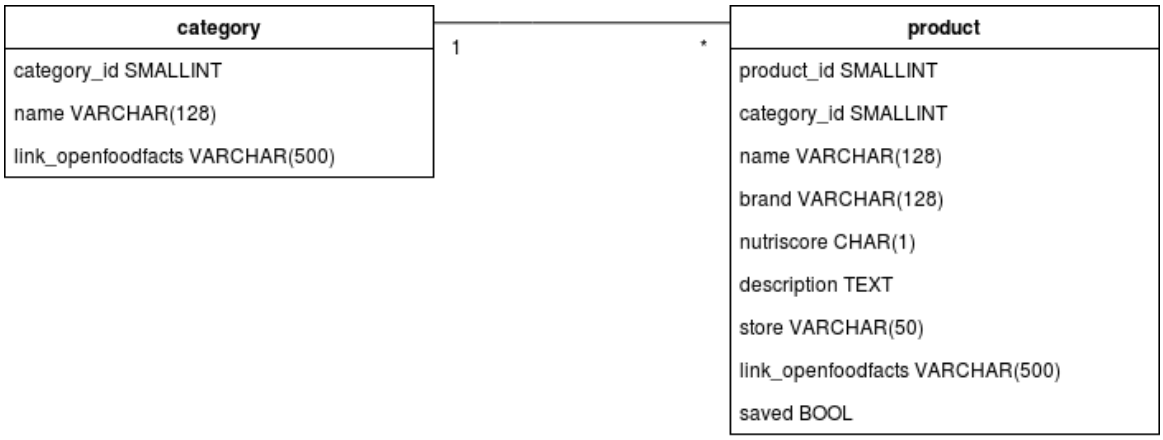
P5 : Utilisez les données publiques de l'OpenFoodFacts

Pour commencer ce projet, j'ai tout d'abord étudié la façon dont le site Open Food Facts permet d'accéder aux données dont j'allais avoir besoin, à savoir la liste des produits d'une catégorie donnée avec les informations de chaque produit, qui est accessible via l'API dans un format de données JSON.

J'ai ensuite testé le module Requests pour Python, qui permet de requêter des données sur internet à partir d'une URL. J'ai utilisé ce module pour récupérer les informations du JSON dans une liste.

J'ai ensuite établi le schéma de la base de données que j'allais utiliser :



La base de données contient deux tables: une pour les catégories, et une pour les produits.
Après avoir modélisé la base de données, j'ai installé MySQL, créé la base, et inséré manuellement les catégories sur lesquelles j'avais choisi de travailler. J'ai testé le module MySQL Connector pour Python afin de vérifier le fonctionnement des requêtes de la base depuis Python.

Pour alimenter la table produit, j'ai développé une fonction d'ajout des produits dans la base de données avec les informations du site Open Food Facts. Cette fonction prend en entrée un id de catégorie, requête la base SQL pour l'URL de la catégorie correspondante, puis le site Open Food Facts pour cette URL. Le résultat retourné par le site est au format JSON et contient une page de produits pour la catégorie. On utilise une boucle pour itérer sur chaque page sur tous les produits de la page. Les produits avec leurs attributs sont stockés dans une liste qu'on utilise pour insérer les données dans la table *product*.

Pour développer l'application, j'ai suivi le scénario d'utilisation décrit dans le cahier des charges du projet, et développé les fonctionnalités une par une dans leur ordre d'utilisation. La première étape de l'interface était le menu principal, qui présente deux choix et attend un input de l'utilisateur. J'ai fait le choix d'utiliser la fonctionnalité "clear" de la ligne de commande (qui efface les entrée précédentes dans le terminal) entre chaque menu, pour une interface plus claire avec uniquement l'affichage du menu courant.

J'ai développé le premier choix possible de l'utilisateur depuis le menu principal : la sélection de catégorie. Le menu pour choisir une catégorie affiche le résultat d'une requête de la table category.
J'ai donc utilisé le module MySQL Connector pour obtenir la liste de toutes les catégories, et les options de formatage de chaîne de caractères offertes par Python avec str.format(), pour afficher le résultat.

Lorsque l'utilisateur sélectionne une catégorie, il faut qu'il choisisse un produit. Il est possible que les produits de la catégorie sélectionnée ne soient pas encore renseignés dans la base. Dans ce cas on utilise la fonction évoquée précédemment qui permet d'ajouter les produits d'une catégorie donnée dans la base de données.

Les catégories contiennent un important nombre de produits. Pour l'affichage des produits dans le menu de sélection, il est donc nécessaire d'utiliser un système de pagination dans notre logiciel.
Pour la pagination, on commence par déterminer le nombre total de produits dans la base MySQL pour la catégorie choisie, puis le nombre de pages (avec le nombre de produits affichés par page qui est une constante dans le code). On établit ensuite la requête SQL qui renvoie les produits de la page courante avec "OFFSET" qui décale la plage de données renvoyées.

On affiche les produits renvoyés en les formatant avec str.format() : En première colonne un entier allant de 1 au nombre de lignes par page, qui reste identique sur chaque page et qui sera utilisé pour la sélection du produit par l'utilisateur, suivi du nom du produit et sa marque, ainsi que le lien Open Food Facts pour la page du produit.

Lorsque l'utilisateur sélectionne un produit, on requête d'abord la base MySQL en interne pour avoir accès à toutes les informations sur ce produit, notamment le nutriscore.

On recherche ensuite le meilleur nutriscore parmi les produits de la même catégorie que celle du produit sélectionné.
On gère les différents cas possibles:

- Le cas où le produit a un nutriscore moyen, on propose à l'utilisateur le produit avec le meilleur nutriscore dans la catégorie.
- Le cas où le produit choisi possède le meilleur nutriscore de sa catégorie, dans ce cas on propose un autre produit qui pourrait être une alternative avec un nutriscore identique. S'il n'y a pas d'autre produit, on en informe l'utilisateur.
- Le cas où le nutriscore n'est pas renseigné pour le produit sélectionné. Dans ce cas, on propose un produit avec le meilleur nutriscore pour cette catégorie.

On propose à l'utilisateur d'enregistrer le produit qui a été proposé. Si l'utilisateur choisit d'enregistrer le produit, on met à jour dans la base la colonne "saved" qui permettra de retrouver les produits qui ont été sauvegardés par l'utilisateur au cours de son utilisation du logiciel.

L'utilisateur peut par la suite consulter ces produits depuis la deuxième option du menu principal.

Le projet répond ainsi aux éléments demandés par le cahier des charges. Des améliorations pour le confort de l'utilisateur sont cependant envisageables :
Parcourir les pages pour sélectionner un produit peut prendre du temps, et une recherche textuelle permettrait d'accéder plus vite au produit souhaité.
Le logiciel propose une seule alternative aux produits, qui est le produit ayant le meilleur score de la catégorie. On pourrait choisir de présenter différents choix de substituts.

Lien GitHub :
https://github.com/Cocomows/P5-Openfoodfacts
Lien Trello :
https://trello.com/invite/b/gcqUBPPH/32b003071e4237d59b23c8f5637a680e/p5-openfoodfacts