Démarche du projet 5 (Open Food Facts)

J'ai commencé par travailler sur la partie métier du projet c'est-à-dire sur la récupération des données utiles grâce à l'API d'Open Food Facts et sur la création de la base de données MySQL afin de pouvoir ensuite utiliser ces informations pour construire la partie "gestion" de l'application.

Tout d'abord, il m'a fallu identifier les données qui me seraient nécessaires pour le projet : ***les produits*** (nom, marque, nutriscore, ingrédients...), ***les catégories*** - à choisir manuellement sur le site OFF -, et ***les substituts***.

- **au niveau de l'API :** j'ai écrit puis testé - avec le logiciel Postman - les endpoints permettant de récupérer les informations sur les produits pour chacune des 5 catégories que j'avais choisies précédemment (par exemple, pour les plats préparés, https://fr.openfoodfacts.org/categorie/plats-prepares.json). J'ai alors regardé le type d'informations et la structure des données renvoyées par l'API d'Open Food Facts et je me suis aperçu qu'il fallait être plus précis sur les requêtes et pour cela il m'a fallu étudier la documentation de l'API d'Open Food Facts ce qui a été particulièrement compliqué car la rédaction des requêtes à concevoir n'étaient pas "intuitives" contrairement à celle utilisées dans le cours.

- **au niveau de la base de données :** j'ai réalisé un schéma de BDD grâce au logiciel Workbench puis j'ai exporté le script de création de cette base de données afin de l'utiliser pour créer mes tables 'Product', 'Category' et Substitutes'.

Une fois tous ces éléments techniques réalisés, j'ai recherché comment exécuter une requête SQL depuis Python et il m'a donc fallu étudier les méthodes de la *librairie "mysql.connector"*.
De même, pour pouvoir effectuer des requêtes avec du code http à partir d'un script Python afin de récupérer les données sur l'API d'Open Food Facts, j'ai utilisé la *librairie "requests"*.

J'ai alors créé le répertoire Github OPENFOODFACTS (https://github.com/MINCARELLI13/OpenFoodFacts.git) et réalisé la classe permettant de créer (et supprimer) les tables de la base de données MySQL puis la classe permettant de remplir ces mêmes tables avec les informations collectées (cette classe intégrait les requêtes à l'API d'Open Food Facts).

Après avoir créer les classes "métier" permettant de récupérer et de stocker les données, je me suis attelé à la partie gestion de l'application et notamment à celui des menus, ce qui est vite devenus très compliqués étant donné le nombre de menus à concevoir ainsi que toutes les informations nécessaires à intégrer et à gérer dans les menus (produit avec ou sans détails, liste de produits, liste de substituts, liste de catégories...).
Un des plus gros - et des plus inattendus - problèmes rencontrés a été la grande quantité de méthodes à utiliser d'où une difficulté certaine à créer des noms de méthodes pertinents.

Une fois l'application finie et opérationnelle, mon mentor m'a conseillé d'utiliser la méthode MVC (Model-View-Controller) pour clarifier mon code.
J'ai donc créé un nouveau répertoire Github "Projet_OpenFoodFacts" (https://github.com/MINCARELLI13/Projet_OpenFoodFacts.git) et j'ai passé plusieurs jours à réorganiser voire à réécrire toutes mes classes avec cette méthode, ce qui a nettement clarifié la partie "menus" du code puisque la librairie "View" m'a permis d'isoler toute la partie affichage du code.
Cette méthode a aussi simplifié la partie "métier" du programme puisqu'elle a notamment permis de scinder les requêtes relatives à l'API de celles concernant la BDD.

Enfin, j'ai dissocié le module "Menu" en deux autres modules :

- un module "Controller_requests" qui gère les appels à la partie métier (les modules de la librairie "Model") et le traitement des données en retour,
- un module "Controller_main" qui gère les menus ainsi que le lien entre le module "Controller_requests" et la partie affichage (la librairie "View"),

ce qui permet de bien scinder les différents types d'actions à gérer.

Après cela, mon mentor m'a invité à étudier la méthode ORM (Mapping Objet-Relationnel) concernant les classes "métiers" et m'en a expliqué rapidement son principe de fonctionnement.
J'ai donc créé un dernier répertoire Github "Projet_OFF" (https://github.com/MINCARELLI13/Projet_OFF.git) et me suis appliqué à réécrire le code de la librairie "Model".
Jusque-là, pour les requêtes SQL, chaque classe correspondait à un type de méthode (par exemple, créer une table) et chaque méthode de cette classe ne permettait d'agir que sur une seule table (par exemple, créer la table 'Product', créer la table 'Category', créer la table 'Substitutes').
Avec le principe ORM, une classe contient toute les méthodes (créer une table, lire une table, remplir une table, supprimer une table...) et chaque méthode est utilisable pour tous les objets de même type (la table 'Product', la table 'Category', la table 'Substitutes').
J'ai donc réécrit les classes métiers correspondantes ce qui m'a effectivement permis de grandement simplifier le code de l'application et de clarifier toute la partie "métier".