1. **Démarche de conception**

Pour bien débuter la conception du projet, nous avons commencé à lister les différentes fonctionnalités demandées. Nous avons écarté neuf propriétés demandées :

|  |  |
| --- | --- |
| **1** | Création du schéma relationnel de la base de données |
| **2** | Etude des réponses de l’API Open Food Facts |
| **3** | Création du découpage du programme |
| **4** | Création d’un jeu de données à importer dans la base de données local |
| **5** | Création de la base de données |
| **6** | Créer les différentes requêtes sql |
| **7** | Remplissage de la base avec le jeu de données créé |
| **8** | Créer l’affichage pour que l’utilisateur puisse interagir avec le programme |

Par la suite, nous avons ouvert le projet dans un répository git pour y mettre les différentes ressources du projet et effectuer les différentes sauvegardes dont nous aurons besoin. Ce repository se trouve via [ce lien](https://github.com/ZasshuNeko/OC-Projet_5).

1. **Etude de l’algorithme**

Nous avons scindé le projet en 6 parties.

Deux parties reposant sur des fichiers extérieurs. L’un servant à la configuration du programme et permettant de garder des informations importantes au sein d’un même fichier sans encombrer le code, que l’on a nommé  « config.ini ». L’autre fichier étant le script de création des bases de données pour permettre à chaque utilisateur de partir d’une base commune lors de l’utilisation du programme que l’on nomme « script\_sql.sql ».

Le fichier à lancer, qui contient le squelette du programme, est appelé « OFF\_main.py ». Il est constitué d’une classe « main » qui est interpellée lors du lancement pour générer l’objet programme. C’est à partir de ce fichier que les autres classes seront interpellées pour former la vie du programme et son interaction avec l’utilisateur. Lors de ce lancement, plusieurs options sont mises à disposition de l’utilisateur et il important d’étudier l’algorythme selon ces différentes options :

La première option proposée est la création de la base de données et l’intégration d’un jeu de données dans la base de données locale. Cela fait appel à trois classes : Classe affichage, Classe requete, Classe bdd\_mysql

La classe affichage est initialisée et crée une instance d’affichage, ainsi, nous appellerons cet objet à chaque fois qu’une interaction avec l’utilisateur est demandée ou à chaque fois qu’une information doit apparaitre à l’écran. Chaque choix est ramené à une fonction vérifiant l’intégrité des données pour éviter les erreurs (tel que mettre une lettre ou un choix n’étant pas dans la liste).

La classe requête présente dans le fichier « class\_requete.py » contient l’ensemble des demandes vers l’API d’Open Food Facts par ces différents modules : req\_catégorie, req\_produit,req\_store. Il permet de faire créer une réponse en jsons utilisable par la suite pour former un jeu de données que nous allons fournir à notre base de données locales. Cette classe permet aussi de gérer le lot de données obtenues en lui retirant les données inutiles venant de l’API et le rendant plus malléable pour le projet. Pour cela, chaque catégorie choisie est envoyée vers la classe requête pour créer un dictionnaire formé de chaque produit provenant de l’API. Lors de la création de ce dictionnaire dans le module « crea\_dictionnary », les produits sont liés à leurs catégories et vendeurs. Une fois chaque produit des catégories extraites dans le dictionnaire, les doublons sont retirés avec le module « dbl\_listing ».

La classe « bdd\_mysql » est initialisée pour instancier la connexion entre le programme et la base de données. Cette dernière est vérifiée par une fonction permettant de faire remonter les soucis de connexion. Une fois la connexion instanciée, la classe va permettre la création de l’architecture de la base de données puis prendre le jeu de données généré par l’API de Open Food Facts retravaillé par la class requête afin de l’intégrer dans la base de données locale.

La seconde interaction avec le programme permet d’interroger la base de données locale pour permettre de rechercher un substitut alimentaire. Cette interaction fait appel à la classe « bdd\_mysql » et à la classe affichage.

Lors de ce choix, la classe « bdd\_mysql » suit alors une simple logique : elle cherche les catégories dans la base de données locales dans la table « categories », puis les donne à afficher à la classe affichage. Une fois la réponse effectuée par l’utilisateur, la réponse est donnée à l’objet « bdd\_mysql » qui va alors interroger la base de jointures entre les catégories et les produits. La réponse de cette requête est fournie à l’objet affichage pour le choix utilisateur. Quand le produit à substituer est choisi, on compare le « nutri\_score » : si celui-là n’est pas renseigné, on compare alors les rapports calorifiques de chaque produit pour extraire un substitut au produit sélectionné par l’utilisateur.

Une fois le substitut sélectionné, les données sont envoyées dans la classe affichage où l’utilisateur a la capacité d’enregistrer son substitut dans la base de données.

Le troisième choix est celui de charger les recherches faites par l’utilisateur. Lors de cette sélection, la classe « bdd\_mysql » est interpellée pour interroger la table « substitut\_save », puis afficher par la classe affichage.

Le dernier choix est la possibilité de configurer la base de données directement dans le programme. Cela fait appel à la classe affichage qui à travers la fonction « aff\_configbdd » va demander les informations pour se connecter à la base de donnée et l’enregistrer dans le fichier « config.ini ».

Nous listons alors les étapes que nous allons devoir exécuter pour réaliser ce projet, nous étudierons ces étapes selon les interactions entre l’utilisateur et le programme :

1. **Etablir un plan de travail**
   1. Méthode agile
   2. Déterminer les demandes et les contraintes du projet
   3. Etude de l’API Open Food Facts et des résultats des jsons obtenus
   4. Etablir un schéma relationnel de base de données
2. **Création du programme**
   1. Etablir les différentes classes dont aura besoin le programme
   2. Etablir le fichier de configuration
   3. Installer, créer la base de données
3. **Création des requêtes API et sql**
4. **Finaliser la logique utilisateur**
   1. Lier les différentes classes
   2. Permettre une interaction utilisateurs dynamique
   3. Enregistrement du substitut
5. **Difficultés & solutions**

Les premières difficultés furent de s’approprier l’API de Open Food Facts, bien comprendre chaque requête, faire la part des données inutiles pour le projet et trouver la meilleure solution de présentation pour le programme. Puis, travailler au mieux le format json pour parvenir à créer un contenant de données utiles pour réinjecter dans la base de données locale. Une fois ce format de données établi, nous avons installé et configuré Mysql. Une des réflexions était de bien déterminer le nombre de table, le type de données dont nous allons avoir besoin. Une fois cela fait, réfléchir à la liaison entre les tables, déterminer les jointures pour avoir une idée plus précise du type de requêtes dont on aura besoin. Nous avons alors établi quatre tables gérant chacune quatre objets différents : les produits, les catégories, les vendeurs et les sauvegardes. Comme nos jointures produits catégories et produit vendeur sont des liaisons « un vers plusieurs », nous avons créé deux tables de jointures supplémentaires. Chaque table est constituée d’un champ « id » utilisé comme clé primaire. Nous avons par la suite généré des clés étrangères pour garantir une cohérence de données. Ainsi, il semblait primordial que ces clés soient présentes sur les tables de jointures ainsi que sur la table de sauvegarde. Nous remplîmes les tables catégories, vendeurs en utilisant les listes créées puis chaque produit, effectuant dans un même temps les liens dans les tables de jointure. Une fois réalisée, nous structurâmes les différentes requêtes sql pour répondre aux demandes du projet.