Démarche choisie

1. Construire la base de données

La première tâche à laquelle je me suis attelé est la structure de la base de données. Afin de trouver rapidement mes produits, j'ai décidé de les laisser tous dans la même table, indépendamment de leurs catégories. Une table des catégories s'est du coup imposée afin de simplifier les requêtes. La table des substitutions a été mise en balance avec une colonne « substitution » dans la table des aliments puis a été conservé pour sa simplicité d'usage et parce qu'elle permettait de mettre en place une jointure.

La question de l'interaction des tables a d'ailleurs été pensé pendant la conception, parce qu'elle permettait de montrer une certaine agilité avec MySQL. L'idée étant de faire le maximum avec MySQL plutôt qu'avec python. Le choix de noter les id dans la table des substituts plutôt que le nom des produits, en plus d'être plus « agile » pour une hypothétique amélioration du programme, permettait de montrer une requête avec jointure interne en html dans le script.

2. Importer des données depuis OpenFoodFact

L'API OpenFoodFact semble plutôt organisée mais souffre d'un défaut de documentation qui est assez problématique pour organiser facilement ses requêtes, il a donc fallu les tester longuement pour importer sereinement les données.

Un deuxième problème est apparu lorsqu'il a fallu faire plusieurs requêtes d'affilée afin de remplir efficacement la table des aliments. L'API bloque par défaut les requêtes en série non sourcée ; j'ai été obligé de réorganiser complètement la façon dont je faisais mes requêtes.

3. Construire l'application

Les fonctionnalités de l'application (observer la liste des produits par catégorie, choisir un substitut, observer la liste des substituts) ont été directement guidées par le cahier des charges du projet. J'ai rajouté assez naturellement une fonctionnalité « initialisation de la base de données » puisqu'elle permettait d'inclure le script de création de la base de données de façon élégante.

Moins évident a été le choix de la programmation orientée objet. Mon premier réflexe a été de réaliser un simple script fonctionnel, au vu de la brièveté de l'exercice, sans me rendre compte que le résultat était difficilement lisible. J'ai donc repris mon architecture dans un deuxième temps, sous l'impulsion de mon mentor et je l'ai refaite orientée objet.

L'architecture finale du programme a été dictée par le souci de séparer la création de la base de données de sa manipulation et in fine du programme final. Les classes principales, InitDataBase, DataBase, et Engine reflètent ce découpage. Elles sont imbriquées les unes dans les autres afin que le connecteur à la base de données puisse être un attribut de la classe Engine et donc qu'il n'ait pas à être reconstruit à chaque requête.

Deux classes subsidiaires servent à amalgamer naturellement les informations extraites de la base de données afin de les manipuler globalement. Un soin particulier a été mené pour rendre la classe Data itérable, principalement dans un but didactique.

La dernière difficulté a été de maîtriser la bibliothèque Python MySQL-Python-Connector afin d'amalgamer script de la base de données et script de l'application python. Comprendre qu'il fallait

sauvegarder les modifications des lignes de la base de données contrairement à un script classique en MySQL a été la seule grosse difficulté de l'exercice.