Démarche du projet 8 : Créez une plateforme pour amateurs de Nutella

Ce document à pour but d'expliquer la démarche et l'algorithme que j'ai utilisé pour réaliser le projet 8 de la formation OpenClassrooms développeur d'application Python.

Voici les liens vers :

- mon code source : https://github.com/micktymoon/P8_pelletier_celine
- mon site web : https://celine-bienmanger.herokuapp.com/
- mon Trello : https://trello.com/b/Q8yMl2D7/pur-beurre

J'ai commencé par créer les vues de façon simple en utilisant la classe View de Django. Mais après avoir créé la vue qui affiche les produits en liste, mon mentor m'a expliqué qu'il existait d'autre classes très pratiques pour ce genre de vues et bien d'autres. J'ai donc fais quelques recherches dans la documentation de Django et j'ai découvert les classes ListView, TemplateView et DetailView. Je les ai donc utilisées pour créer un certain nombre de mes vues, mais pas toutes.

Certaines vues ne pouvant pas utiliser les classes de Django, j'ai du créer des vues plus compliquées. Notamment pour les vues d'enregistrement de compte, de recherche de produits dans la base de données et de sauvegarde des produits que l'utilisateur souhaite garder en mémoire.

Pour la vue d'enregistrement, j'ai d'abord utilisé le formulaire UserCreationForm, puis je me suis rendu compte qu'il manquait le prénom, le nom et l'adresse email. J'ai donc créé mon propre formulaire héritant de la classe UserCreationForm auquel j'ai ajouter les données manquantes.

Pour la vue de recherche, j'ai utilisé le formulaire SearchForm que j'ai créé, il hérite de la classe Form et ne possède qu'un seul attribut « search » qui contiendra la recherche. Cette vue recherche d'abord dans la base de donnée les produits contenant dans leur nom, la recherche. Ensuite, il récupère le premier produit trouvé dans la base de données et recherche des produits ayant un nutri-score inférieur ou égal au produit et qui possède une ou plusieurs des catégories de ce produit.

La vue de sauvegarde récupère l'ID du produit que l'utilisateur souhaite sauvegarder et le retrouve dans la base de données. Ensuite, elle vérifie qu'un utilisateur est bien connecté. Si oui, elle relie le produit à l'utilisateur. Si non, elle redirige vers la page de connexion.

Bien sûr pour relier le produit à l'utilisateur j'ai du créer un nouveau modèle d'utilisateur, car le modèle User de Django est très simple. Ce nouveau

modèle s'appelle aussi User, il hérite de la classe AbstractUser et possède un attribut « product » qui est une relation de plusieurs à plusieurs entre les modèles User et Product. Pour utiliser ce nouveau modèle, j'ai du ajouter la ligne « AUTH_USER_MODEL = 'website.User' » au fichier settings.py et la ligne « admin.site.register(User) » au fichier admin.py.

Par la suite, j'ai créer une fonction, « search_product », me permettant de récupérer les produits d'Open Food Facts. Cette fonction récupère une liste de produit correspondant au nom du produit qu'on lui met en argument. Pour chaque produits trouvé, il récupère son nom, sa marque, ses catégories, les magasins dans lesquels on le trouve, son nutri-score, l'URL redirigeant vers sa page Open Food Facts, l'URL de son image et ses repères nutritionnels pour 100g.

Je comptait utiliser cette fonction pour remplir ma base de donnée, mais je ne savais pas encore comment. Après quelques recherches et l'aide de mon mentor, j'ai découvert les commandes personnalisées de Django. J'en ai donc créé une, « fill_db ». Cette commande contient une liste de nom de produit. Pour chaque nom de cette liste elle va récupérer, grâce à la fonction « search_product », une liste de produit qu'elle va par la suite enregistrer dans la base de données, en vérifiant au préalable que le produit n'existe pas déjà dans celle-ci.

Pour ce qui est des tests, je n'ai malheureusement pas pu faire du Test-Driven Development car je ne suis pas encore à l'aise avec la création de fonctionnalités. Je n'arrive pas encore à savoir à l'avance comment vont fonctionner mes fonctions et donc comment je devrais les tester. J'ai donc créé mes tests une fois que mes fonctionnalités ont été terminées.

Pour les tests de vues, de formulaires et de modèles j'ai utilisé la classe « TestCase » de Django, celle-ci utilise le module « unittest » de la bibliothèque de Python.

Et pour les tests des fonctions, que j'ai créé pour faire la recherche dans l'API d'Open Food Facts, j'ai utilisé le module « unittest » et notamment la classe « TestCase » ainsi que des Mock pour remplacer la recherche dans l'API d'Open Food Facts.