Développement d'Applications Python – Projet 08
« Créez une plateforme pour amateurs de Nutella »

1. Lien vers le code source du projet sur GitHub

https://github.com/etienne86/oc_p08_pur_beurre

2. Reformulation succincte du sujet

L'objectif de ce projet est de concevoir un site web, composé de plusieurs pages, dont les fonctionnalités sont les suivantes :
- l'utilisateur saisit, dans un champ de recherche, un produit qu'il souhaite remplacer
- le programme lui propose des produits de substitution
- si l'utilisateur est connecté, il a la possibilité d'enregistrer un ou plusieurs produits, qu'il pourra retrouver ultérieurement dans ses favoris
- l'utilisateur a la possibilité de consulter la fiche détaillée d'un produit

Dans le cadre de la progression dans le parcours DA Python, ce projet vise notamment à mettre en œuvre de nouvelles compétences liées à :
- l'apprentissage du framework web Django côté serveur
- l'utilisation d'un thème avec le framework CSS Bootstrap côté client

Il permet également de renforcer la pratique du développement web avec HTML, CSS et JavaScript, qui a été abordée au projet 07.

L'application est accessible en suivant ce lien : https://purbeurre-etienne86.herokuapp.com/

3. Gestion du projet

Depuis le projet 04, j'utilise la méthodologie agile pour planifier au mieux mon travail. Voici le tableau de suivi que j'ai utilisé pour piloter le projet 08, avec des user stories et des cartes techniques : https://trello.com/b/jnUY454w/e-barbier-oc-da-py-p08-pur-beurre.

Comme pour le projet 07, j'ai principalement utilisé la méthode du *Test Driven Development*.

4. Déroulement du projet

4.1. Utilisation de l'API d'OpenFoodFacts

L'API ayant déjà été utilisée lors du projet 05 pour des besoins très proches de ceux du projet 08, la prise en main de cette interface a été très rapide. En effet, seules quelques données supplémentaires ont été utiles ici en plus des précédentes, ce qui m'a permis d'ajuster le diagramme de classes par rapport à celui du projet 05.

4.2. Utilisation du framework Django

J'ai trouvé l'utilisation de Django plus simple et plus intuitive que celle de Flask dans le projet 07. J'ai cependant rencontré des difficultés concernant les problèmes d'imports dans les différents modules. En effet, de fréquentes exceptions ont été levées par Django à ce sujet, pendant le développement, mais aussi pendant le déploiement sur Heroku (cf. paragraphe 4.5).

J'ai choisi d'organiser mon projet Django autour de deux applications :
- « auth » pour gérer l'authentification des utilisateurs
- « off_sub » qui est l'application principale, qui s'appuie sur les données de l'OpenFoodFacts pour proposer des substituts à un aliment donné

4.3. Plan de tests

J'ai élaboré un plan de tests contenant trois types de tests :
- les tests unitaires pour les modèles, les vues et la commande de chargement de la base de données
- les tests fonctionnels pour la mise en place des différentes fonctionnalités du projet
- les tests d'intégration pour les interactions entre ces fonctionnalités

Concernant les tests unitaires, j'ai utilisé le *Test Driven Development* (TDD) pour les modèles de l'application principale. Pour cela, j'ai pu m'appuyer sur mon expérience du projet 07 de manière profitable. Les tests unitaires concernant les vues permettent de vérifier que la page demandée renvoie le code de statut attendu, ils ont été codés après les vues elles-mêmes. De même, les tests unitaires éprouvant la commande de chargement de la base de données ont été écrits après la commande, notamment parce que je me suis fortement inspiré du programme existant réalisé au projet 05. Avec l'expérience du projet 08, j'essaierai autant que possible de travailler avec le TDD pour mes futurs projets Django.

J'ai utilisé la librairie *selenium* pour mettre en place les tests fonctionnels et les tests d'intégration. J'ai trouvé cette partie agréable, et je mesure l'utilité de ce type de tests, en particulier lors de l'ajout d'une nouvelle fonctionnalité dans un projet. Au départ, j'ai éprouvé des difficultés puisque je ne parvenais pas à utiliser mon code JavaScript dans la simulation (de même pour les fichiers de style CSS, mais je pensais alors que c'était normal), puis j'ai constaté en parcourant la documentation de Django que mes classes de tests n'héritaient pas de la bonne classe, à savoir *StaticLiveServerTestCase* (et pas *LiveServerTestCase*).

4.4. Utilisation du framework Bootstrap

Afin de prendre en main Bootstrap, j'ai choisi de démarrer le développement de l'interface utilisateur (front-end) sans intégrer le thème fourni. Cela m'a permis de bien comprendre le mode de fonctionnement du framework, puisque j'ai dû résoudre des problèmes en partant d'une page blanche. Cela m'a ensuite été très profitable lors de l'intégration du thème *Creative*.

4.5. Déploiement sur Heroku

Le déploiement de l'application sur Heroku s'est globalement bien passé, avec l'utilisation de la librairie *django-heroku*. Cependant, j'ai rencontré une difficulté importante avec des exceptions levées concernant les imports (comme mentionné dans le paragraphe 4.2). Le problème était que mon arborescence de projet Django n'était pas celle requise, en particulier pour garantir la similitude entre le répertoire de travail, la racine du projet et la racine des imports. J'ai contourné la difficulté pendant le développement en me positionnant manuellement sur le bon répertoire de travail. Pour le déploiement, j'ai donc adapté mon arborescence de projet, ce qui m'a permis de résoudre le problème.

5. Retour d'expérience

J'ai trouvé ce projet très riche, car composé de plusieurs nouveautés pour moi :
- l'intégration d'une interface front-end bien plus élaborée qu'au projet 07, et surtout plus professionnelle et plus moderne avec l'apport de Bootstrap et du thème
- la mise en place d'un plan de tests permettant de rendre le projet plus robuste
- l'utilisation du framework Django, à la fois puissant et agréable, dont j'ai trouvé la documentation très bien rédigée

Afin de mettre à profit les apprentissages liés à la réalisation de ce projet, en particulier en vue du projet 13, je retiens principalement que :
- de façon générale, la documentation est la source d'informations à privilégier et à ne pas négliger, passer du temps à la lire est souvent une bonne idée et un bon investissement
- respecter le principe [répertoire de travail = racine du projet = racine des imports] permet d'éviter un certain nombre de problèmes dans un projet Python, notamment avec Django