Lien trello : https://trello.com/b/59qHIEHT/projet10

Lien GitHub: https://github.com/yannhamdi/projet8

Lien Application : 167.99.143.67

Dans le cadre de ce projet, nous avons comme instructions de déployez l’application du projet8 sur un serveur ainsi que d’effectuer quelques différentes tâches telles que suivre notre projet par le biais de « Travis », suivre les logs de l’application en utilisant « sentry » et finalement suivre les performances de notre site grâce à une fonctionnalité disponible sur digital ocean.

**DIFFICULTES :**

Ce que j’ai trouvé difficile sur ce projet du fait de sa nouveauté est le fait de travailler en parallèle sur le serveur et en local. Ainsi que de préparer tous les fichiers de configuration pour le bon fonctionnement de l’application en local et en production. J’ai éprouvé quelques difficultés pour modifier les fichiers settings et séparer les environnements.

**SOLUTIONS :**

Les cours de ce projet étaient relativement bien ficelés et les explications de configuration m’ont donc guidées de manière efficace vers le bon fonctionnement de l’application en production.

J’ai donc pour ce projet suivi minutieusement les différentes instructions du cours chapitre par chapitre et je dois avouer que ce projet s’est plutôt bien déroulé.

Je vais donc vous expliquer quelles ont été les étapes pour ce projet :

* Digital Océan
* Travis
* Suivi performance sur Digital Océan
* Sentry
* Cron

**DIGITAL OCEAN :**

Comme chaque application créée sur un projet Django, on y retrouve quasiment les mêmes modules (models, views, urls…)

J’ai donc commencé par le module « models.py » qui va nous fournir la structure de notre base de données « OpenFoodFacts », j’ai consulté le cahier des charges ainsi que le site d’openfoodacts afin de présélectionner les informations nécessaires pour le bon fonctionnement de notre projet.  
Une fois, la structure de mon modèle en place, il a fallu à l’aide du module « importapi » récupérer les données via l’api d’Openfoodfacts.

Nous avons donc notre modèle et nos données, j’ai donc créé une commande personnalisée « dbopenfood » afin de remplir cette base de données.

Base de données étant fonctionnelle, j’ai conçu les vues pour mon application « products », on y retrouve une vue pour afficher les informations d’un produit sélectionné., une vue qui affichera le produit recherché et le produit de substitution (nous reviendrons sur cette vue lors de la description de l’application favoris), une vue qui va faire appel au module « managers.py » qui procédera à la requête afin d’obtenir un aliment avec un nutrition grade meilleur que le produit recherché.

Puis, nous avons le module « forms.py » qui proposera un formulaire à l’utilisateur afin qu’il effectue sa recherche.

Et enfin un module « urls.py » afin de créer les adresses qui permettront d’accéder aux vues.  
L’application « products » comme son nom l’indique s’occupera de gérer la partie concernant les aliments.

**TRAVIS :**L’application « Users » permet de gérer la partie authentification de notre projet.  
Nous avons la partie modèle qui va nous permettre de définir notre base de données à partir du modèle prédéfini de Django.

Puis nous avons la partie formulaire afin d’inviter l’utilisateur à soit entrer ses identifiants soit s’inscrire ainsi qu’une partie validation afin de renvoyer une erreur dans le cas où l’utilisateur fournit des identifiants erronés.

Et finalement, la partie « Views » qui va faire le lien entre l’interface graphique et le backend.

On aura une vue pour l’inscription, une autre pour se connecter, une vue également pour se déconnecter et enfin une vue pour afficher les informations du compte comme demandé dans le cahier des charges.

**SUIVI PERFORMANCE SUR DIGITAL OCEAN**

L’application « favoris » va gérer les enregistrements des recherches de l’utilisateur et l’affichage des enregistrements.

J’ai donc également créé un modèle avec une colonne qui va pointer vers l’utilisateur afin d’identifier quel utilisateur a enregistré la recherche, une colonne qui va pointer vers le produit recherché et une colonne vers le produit proposé, ensuite les vues de l’application géreront l’affichage.

Il y a également une contrainte pour cette application, les vues sont accessibles uniquement si l’utilisateur est connecté, j’ai utilisé pour ça le décorateur « @loginrequired().

**SENTRY :**

Voilà, la partie Django backend étant terminé, nous passons à la partie Front-end.

Notre client nous fournit une maquette Bootstrap, en effet il souhaite que son site ait l’aspect visuelle relativement identique à celui de « creative bootstrap ».

Nous récupérons donc le git de creative puis l’adaptons à notre projet. Nous récupérons l’html du fichier index.html que nous modifions selon l’esquisse fournit par notre client.

A partir de là, nous travaillons sur les fichiers « scss » afin de modifier le CSS du projet en adaptant les couleurs, les photos et la police fournit par notre client. Puis avec « npm » nous modulons les modifications.

**CRON :**

Une fois, notre application fonctionnelle, il nous faut la déployer sur Heroku. Le déploiement est relativement facile, la seule vrai subtilité exporter nos bases de données vers le serveur Heroku. Une fois les bases de données exportées, nous déployons notre application sur heroku.

L’application est enfin en production.

Je souhaite partager mon ressenti pour ce projet, j’ai trouvé la partie backend relativement facile, Django est relativement facile à prendre en main et le tuto officielle est très bien expliqué et bien détaillé.

Avec ce projet, j’ai découvert que j’avais énormément de lacunes au niveau front-end, il y beaucoup d’élément à gérer positionnement, aspect responsive. Je me rends compte que je dois peaufiner mes compétences au niveau CSS et Bootstrsap. Je me rassure en me disant que ce n’est que le deuxième projet que l’on prépare au niveau front-end alors que le backend, nous avons déjà eu un parcours bien complet.