Créez une plateforme pour les amateurs de Nutella

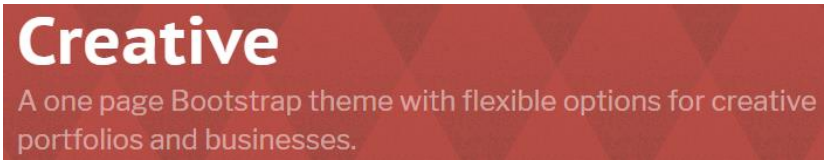
Pour commencer ce projet, j'ai suivi la méthode habituelle à tous les derniers projets :

- Initialiser un repository Git Hub (afin de pouvoir versionner son code tout le long du projet) https://github.com/cenotapHe/Plateforme_Projet8
- Initialiser un tableau Trello (afin de découper le projet en User Story, et de l'effectuer avec une méthodologie agile) https://trello.com/b/djrQoAS5/projet-8-da-python

Le projet 8 tient comme élément central Django, un puissant framework de Python, que j'ai découvert avec ce projet. Dans le P7, nous avions travaillé pour la première fois sur un framework avec Flask. Mais grâce à Django, nous multiplions aujourd'hui les fonctionnalités qui s'offrent à nous. Tout en ayant déjà acquis la prise en main d'un framework.

Django a été pensé pour faciliter la vie des développeurs. En une ligne de commande notre projet Django se créé. Une ligne de commande supplémentaire et nous créons une application reliée à notre projet.

Grace à toutes ces facilités nous pouvons très rapidement nous mettre au travail sur le front. Et pour ça nous allons nous servir de Bootstrap, et d'une librairie en particulier : creative.



Creative nous a été demandé par le client. Elle nous a permis d'avoir un rendu magnifique, presque professionnel. Sur lequel est inséré, entre autres, une barre de navigation fonctionnelle et un site entièrement responsive.

Avant de créer notre base de données, nous avons relié le projet 5 (Utilisez les données publiques de l'OpendFoodFacts) à celui en cours. En premier lieu nous avons récupéré notre script du P5, capable de récupérer les données d'OpenFoodFacts et de les transformer sous la forme de requêtes SQL. L'ancienne application fonctionnait uniquement sous terminal, et ne nécessitait qu'un petit jeu de données pour fonctionner. Donc nous avons modifier les différentes informations à récupérer (supprimer le magasin de vente, et récupérer l'image de l'objet ainsi que l'étiquette des substances nutritives) puis de rajouter plus de catégorie et de produit à récupérer.

Apperçu du fichier sql

```
INSERT INTO catalogue_category
VALUES (1, 'fruits-based-foods');

INSERT INTO catalogue_product
VALUES ('0', 'Mélange de noix', '_Fruits à coq

INSERT INTO catalogue_product
VALUES ('1', 'Confiture de fraises', 'fraises

INSERT INTO catalogue_product
VALUES ('2', 'Les allégées en sucres fraise',

INSERT INTO catalogue_product
VALUES ('3', 'Cocktail Fruits', 'Raisins secs

INSERT INTO catalogue_product
VALUES ('4', 'Confiture orange amere', '', '4'

INSERT INTO catalogue_product
VALUES ('5', 'Spécialité Pomme Fraise-Myrtille

INSERT INTO catalogue_product
VALUES ('6', 'Marmelada', 'Pulpe de coing (coi

INSERT INTO catalogue_product
VALUES ('7', 'Pomme Abricot', 'pommes 57 %, ab
```

Ajout de nouvelles catégories

```
name_category = ['fruits-based-foods', 'legumes-and-their-products', 'juices-and-nectars', 'biscuits', 'ice-creams-and-sorbets',
                 'chocolates', 'breakfast-cereals', 'meats', 'yogurts', 'french-cheeses', 'sauces', 'seafood', 'jams', 'breads', 'chips-and-fries']
```

Une fois le fichier SQL créé et PostgreSQL installé, la base de données fut plutôt simple à créer. Dans notre application, via le modèle MVT de Django, nous avons créé une recherche dans la base de données à parti du site internet. Même si nous parlions d'un site déjà responsive, on continu d'ajuster les détails, comme un affichage de recherche en trois colonnes sur PC et une seule sur smartphone.

Cette recherche nous a resservit sur plusieurs pages. Comme la page de détail de chaque article. Sauf que celle-ci ne dépend pas de l'utilisateur, mais de la comparaison des articles entre eux. Nous avons donc réutilisé l'algorithme de comparaison des articles utilisés au P5. En comparant les articles des différentes catégories, par leur nutriscore, nous pouvons proposer tous les substituts viables.

Django possède une utilisation simplifiée des utilisateurs. Il les gère intégralement, créant même jusqu'à leur table dans la base de données. En possédant en plus des opérations telles que loggin ou loggout.

```
5    from django.contrib.auth.models import User

15    from django.contrib.auth import authenticate, login, logout
```

Grace à la gestion des utilisateurs de Django, il nous suffit de modifier la recherche automatique des pages détails en rajoutant un bouton « substituer » sous chaque article. Ainsi un utilisateur connecté peut facilement remplacer les articles qu'il souhaite. Ceux-ci se gardant en mémoire, et s'affichant dans la page « Mes Aliments » de chaque utilisateur.

Puis il ne restait plus que la mise en ligne avec Heroku. La nouvelle compétence à acquérir venait de la base de données à mettre en ligne. Mais cela ne posa aucun problème car la démarche est très bien expliquée, ligne par ligne, dans le cours OpenClassrooms « Découvrez le framework Django ».

Je me permets d'ailleurs de laisser le lien pour le site fonctionnel en production :

https://plateforme-p8.herokuapp.com

Dans l'ensemble, je n'ai pas eu de gros problèmes ou de blocage dans ce projet. La difficulté vient plutôt de la multiplicité des fichiers et des détails à l'intérieur du projet. L'ajustement de chaque paramètre prend un temps énorme. Mais c'est du temps que j'ai apprécié passer à la découverte de ce framework.