PROJET 8 : Créez une plateforme pour amateurs de Nutella

Lien Heroku : https://purbeurre-ocp8.herokuapp.com/

Lien Github : https://github.com/preudh/P8_Django_Purbeurre

Lien Trello : https://trello.com/b/vGGLnN7o/p8ocpreudh

Démarche suivie :

1) Utilisation de l'outil Trello (Cf. dossier documentation ou lien):

*NFR (non fonctional requirements)

*Démarches

*Products backlog = users stories triées par priorité

*Sprint backlog = mettre les plus importantes en premier

*In progress

*Done

*Difficultés rencontrées : évaluation de la charge de travail par sprint.

2) Lecture de documentations Django :

* https://docs.djangoproject.com/fr/3.2/

* Le blog de Victor Freitas: https://simpleisbetterthancomplex.com/

3) Esquisse :

Ces dessins permettent de clarifier le processus et d'écrire le html, css en utilisant le thème de Bootstrap. J'enrichie les esquisses de la page d'inscription et d'une page où l'utilisateur peut sauvegarder des substituts. Difficultés rencontrées : à ce stade, je n'avais pas envisagé tous les flux. En codant, j'enrichi les flux, les exceptions.

4) Modèle de données :

Objectifs : lister les classes et structures des tables. J'utilise le modèle de données du projet P5

5) Installation d'une base PostgreSQL : Après avoir installé l'outil pdAdmin 4, je configure le fichier setting.py dans Pycharm.

6) Création du projet Django dans l'IDE Pycharm et organisation du projet selon 3 applications :

1/app_data_off, écriture script pour récupérer et alimenter la base de données avec la commande « python manage.py cm_db »,

2/app_user pour gérer les logins, inscriptions et logout,

3/app_management pour gérer principalement la fonction de recherche , de détail, de sauvegarde, des favoris.

7) Initier un repos GitHub :

8) Codage de l'application :

*Ecrire les templates, html, css en utilisant bootstrap (thème du cahier des charges) et en utilisant Crispy Forms pour le login et le register,

*Choix Function-Based Views versus Class-Based Views car FBV simple à mettre en œuvre, facile à lire, flux de code explicite, utilisation simple des décorateurs. Néanmoins, utlisation des CBV sur autres projets à prévoir,

*Ecrire les controllers et les views dans Django avec une attention particulière pour la fonction « search »,

*Combiner les back end avec le front end et répéter jusqu'à ce qu'il n'y ait plus de problème.

9) Tests :

*Fichier tests unitaires dans chaque application et fichier tests fonctionnels Selenium à la racine « functional_tests.py »,

*Coverage à 54% sans omit donc ok sans la pondération des méthodes natives à Django,

*Rédaction d'un TestscenarioTemplate (cf. fichier tableur dans dossier /docs),

*Tests responsives réalisés sur plusieurs téléphones portables.

10) Création d'un pipeline entre Pycharm, Github et Heroku

11) Analyse du log et instalation de Heroku CLI pour affiner l'analyse des bugs et permettre ainsi la mise en production

12) Conclusion et axes d'amélioration :

Apprentissage de Django, Design pattern MVT, Bootstrap, PostgreSQL, Tests Fonctionnels,

Renforcement des compétences html, CSS, débugger dans Pycharm, tests unitaires,

Axes d'amélioration du présent projet :

*CSS à mettre uniquement dans le dossier static/css. Reste du CSS dans le html,

*Tests unitaires à utiliser en amont (TDD),

*Utilisation des slugs dans les models Category et Product pour un meilleur référencement et visbilité des URL.