

Projet 11 - Améliorez un projet existant en Python

Étudiant : Guillaume OJARDIAS Mentor : Erwan KERIBIN
Mentor évaluateur : Ranga GONNAGE

Sommaire

I. Présentation	2
I.1. Liens du projets	2
II. Tâches du projet	2
II.1. Modification du workflow	2
Tox	2
GitHub Actions	2
II.2. Correction d'un bug	2
II.3. Envoi d'une fiche produit par e-mail	3
III. Bilan du projet	3
III.1. Modification du workflow	3
III.2. Envoi d'e-mails	3
III.3. Conclusion	4

I. Présentation

Ce projet reprend l'application réalisée lors des projets 8 et 10.

Le but est d'ajouter une fonctionnalité à un projet existant et de corriger un bug introduit par un tiers.

I.1. Liens du projets

Le code source du projet est disponible sur la plateforme GitHub à cette adresse : <https://github.com/GuillaumeOj/P11-AddAFeature>.

L'application en production est visible à cette adresse : <https://projet-11.ojardias.io/>.

Le tableau Notion du projet est accessible ici : <https://www.notion.so/guillaumeoj/>.

II. Tâches du projet

Les modifications apportées à ce projet sont :

- ajout et configuration de *tox*;
- utilisation des *GitHub Actions* à la place de *Travis CI*;
- correction d'un bug;
- envoi par e-mail d'une fiche produit.

II.1. Modification du workflow

Tox

J'utilise *Tox* quotidiennement pendant mon stage. Ce projet est l'occasion de mieux comprendre son fonctionnement et d'apprendre à le configurer par moi même.

Le fichier `tox.ini` configure plusieurs environnements :

- `py39` : lance les tests;
- `pep8` : vérifie la conformité du code avec `black`, `isort` et `flake8`;
- `coverage` : calcul le taux de couverture des tests;
- `start` : démarre le serveur de développement;
- `prod` : exécute les tâches nécessaires à la mise en production de l'application;
- `init-db` : initialise la base de données.

Les environnements `py39`, `pep8` et `coverage` sont utilisés par le CI. Les trois autres sont dédiés à une utilisation manuelle et ponctuelle.

La mise en place de `tox` se fait à partir de ce commit : 936f512. La liste complète des commits qui concernent `tox` est disponible *ici*.

GitHub Actions

Désormais, ce projet comporte deux workflows un CI et un CD. Le premier pour s'assurer de la bonne exécution des tests, le second pour déployer l'application sur le serveur.

Cette transition de `Travis CI` à `GitHub Actions` est visible à partir de ce commit : 04dd5af.

La mise en place du workflow de déploiement est visible sur ce commit : 22f0168.

II.2. Correction d'un bug

Le bug est lié à une mauvaise pratique de ma part. En développement, l'application utilisait SQLite comme base de données. Alors qu'en production, elle utilise PostgreSQL. La conséquence était que les tests échouaient en utilisant PostgreSQL mais pas SQLite.

Pour remédier au problème, il a fallu :

- ajouter des paramètres pour utiliser PostgreSQL avec le serveur de développement;
- corriger les modèles provoquant l'erreur;
- modifier les tests associés.

La correction de ce bug est visible sur le commit suivant : bcdfa1c

II.3. Envoi d'une fiche produit par e-mail

L'utilisateur doit pouvoir recevoir la fiche d'un produit par e-mail simplement en cliquant sur un bouton.

Pour la mise en place de la nouvelle fonctionnalité, j'ai découpé mon travail en trois parties :

- mise en place de la vue nécessaire sur la partie back-end de l'application;
- création du template du mail en utilisant le framework *MJML*;
- réalisation du front-end en ajoutant le bouton d'envoi de la fiche du produit.

Le fonctionnement général de cette fonctionnalité est le suivant :

- l'utilisateur clique sur le bouton d'envoi de la fiche;
- on redirige les utilisateurs non connectés vers le formulaire de connexion;
- une fois connecté l'application génère un e-mail avec le template l'e-mail;
- puis l'e-mail est envoyé via le serveur SMTP de *Sendgrid*.

L'ajout de cette fonctionnalité est visible sur les commits suivants : 59f6f72, 84b2dde et d2208e6.

III. Bilan du projet

III.1. Modification du workflow

La partie la plus difficile dans la modification du workflow est la mise en place du déploiement de l'application. Je me suis retrouvé confronté à plusieurs problèmes :

- peu de documentation exhaustive dans les docs GitHub;
- pas d'exemple convaincant sur le sujet;
- une mauvaise compréhension des actions à effectuer.

Finalement, après plusieurs tentatives, j'ai fini par saisir les étapes à suivre :

- créer une paire de clef SSH dédiée à GitHub et une paire propre au serveur;
- enregistrer la clef privée de GitHub dans les **secrets** du dépôt;
- ajouter la clef publique du serveur dans les clefs SSH autorisées sur le compte GitHub;
- ajouter la clef publique de GitHub dans les clefs autorisées du serveur;

Ce qui donne le workflow suivant :

- connexion au serveur depuis GitHub en SSH;
- récupération des nouveaux commits avec `git pull --rebase` en SSH;
- application des migrations, collecte des fichiers statiques, etc;
- redémarrage de l'application;

III.2. Envoi d'e-mails

La création de cette fonctionnalité m'a fait prendre conscience de la difficulté que peut représenter la création d'un e-mail transactionnel :

- impossible d'utiliser un framework CSS tel que Bootstrap;
- impossible de faire appel au système de feuilles de styles CSS;
- disparité importante de prise en charge du HTML et du CSS selon le client de messagerie;
- difficile d'attacher les images à l'e-mail directement.

Pour répondre à la problématique de la mise en forme de l'e-mail, le template utilise le framework *MJML*. Les objectifs de ce framework sont les suivants :

- utiliser un langage spécifique pour simplifier la mise en page d'e-mails;
- garantir un affichage correct quelque soit le client de messagerie et quelque soit le support de lecture (ordinateur, smartphone, etc.)

L'intégration du template au sein de l'application se fait simplement grâce au plugin dédié `django-mjml`. Ce plugin permet de renseigner le template au format `mjml` directement dans la vue, sans avoir besoin de lancer le build de celui-ci.

Pour l'intégration des images, l'idéal est de faire appel à un CDN. Cependant pour les besoins de ce projet, le logo est celui déjà présent sur le serveur de l'application et l'image du produit est délivrée par Open Food Facts.

III.3. Conclusion

Le plus dur a été de ne pas s'égarer. La différence d'expérience entre les projets 8 et 11 fait que la tentation est grande de refactorer son code. Mais en se plaçant dans le contexte de ce projet, lorsque l'on a des délais et des coûts à tenir, pas questions de perdre du temps.

Pour finir, je suis très heureux de constater mon évolution depuis le début de mon apprentissage. Découvrir de nouveaux outils ou de nouvelles technologies devient plus facile. La lecture des documentations associées est aussi plus rapide. Bref, je suis en bonne voie vers la fin de mon parcours !