Projet 13 - Extreme Programing

Parcours OpenClassrooms - Développeur d'application Python

Étudiant : Guillaume OJARDIAS Mentor : Erwan KERIBIN
Mentor évaluateur : Jimmy Kuassi KUMAKO

I. Les règles de l'Extreme Programming

Il existe un certain nombre de règles pour cette pratique. En me basant sur cette source : *http://www.extremeprogramming.org/rules.html*, j'en ai sélectionné quelques une.

1. Planning - User stories are written

Dans le cadre de ce projet les users stories ont étaient simplifiées sous forme de cartes de fonctionnalités. Ces cartes sont mise en place sur un tableau Kanban.

2. Planning - Make frequent small releases

Le workflow du projet est fait de telle manière que les releases sont faites en continue. Globalement, chaque nouvelle fonctionnalité correspond à une pull-request sur le repository GitHub. Dès que celle-ci passe les tests, Mergify la merge dans la branche `master`. A chaque commit sur cette branche, un déploiement sur le serveur de production est exécuté.

3. Designing - Simplicity

Le découpage de l'application a été fait de manière à avoir de petites fonctionnalités. De plus l'application fonctionne suivant un modèle simple :

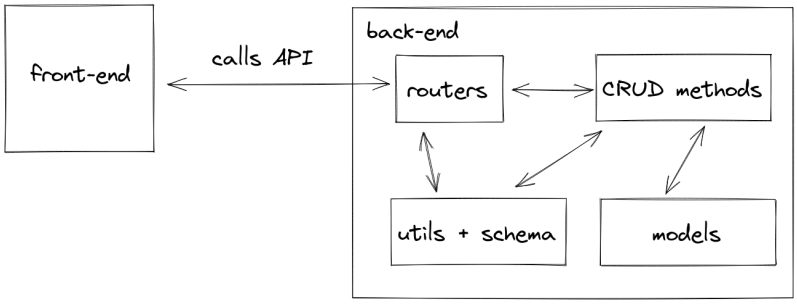

Figure 1: Schéma application

4. Designing - No functionnality is added early

La liste des fonctionnalités, nécessaires, en début de projet m'a permis d'ordonner celles-ci par ordre de priorité. Devant achever le parcours au plus vite, j'ai pu ainsi mettre de côté certaines fonctionnalités, lorsque cela c'est révélé nécessaire, pour les créer dans le futur.

5. Designing - Refactor whenever and wherever possible

Ce projet est partie de rien. La conception que j'ai établie au début pouvait présenter des défauts, des manquements, de redondances. En avançant dans la réalisation de l'application, je me suis rendu compte que certains bout de codes pouvait être optimisés. J'ai donc pris le temps, sur certaines partie, de refactorer le code et de le simplifier.

6. Coding - Code must be written to agreed standards

Pour les standards du code j'utilise systématiquement plusieurs outils d'aide tels que :

- `black` pour le formatage;
- `flake8` pour le respect de la PEP8;
- `mypy` pour la vérification statique du typage.

Ces outils font partis des tests exécuté sur la CI.

7. Coding - Code the unit test first

Dès le projet 7 du parcours, j'ai été initié à l'écriture des tests et l'importance qu'ils peuvent avoir pour la stabilité et la fiabilité de l'application.

La plupart de ces tests ont été écrit dans l'esprit du TDD donc avant l'écriture des fonctions/class/methods.

8. Coding - Integrate often

Ce point rejoint un peu le point 2. Planning - Make frequent small releases.

Le workflow GitHub met en production le code à chaque nouveau commit sur la branche `master`. Hors une pull-request est créée pour chaque nouvelle fonctionnalité ou fix.

9. Coding - Integration computer

Dans le cadre de ce projet, l'intégration computer est tout simplement le serveur qui est utilisé pour la mise en production de l'application.

10. Testing - All code must have Unit test

La partie back-end de l'application est testée sur les points les plus stratégiques (`routers` et méthodes `CRUD`) et sur certains points spécifiques tels que les fonctions `utils`.

11. Testing - All code must pass unit test

Le code passe par une étape de CI lors de la création de la pull-request. Grâce à l'utilisation de Mergify, celles-ci est mergé automatiquement une fois que les tests sont valides.

12. Testing - Create test for each bug

Lors de la création de l'application front-end, j'ai rencontré quelques bug sur le back-end. Pour les corriger, j'ai écrits de nouveaux tests pour répondre au cas de figure rencontré, puis j'ai refactoré/corrigé le code existant.