


## PROJET 7: DÉMARCHE ADOPTÉE

### Ressources externes

 Code source hébergé sur GitHub

 Tableau agile hébergé sur Trello

#### 1) Étude des cours associés

En s'agissant d'un projet *full-stack*, il y a des compétences requises pour son bon développement soit au niveau du *back-end* soit *front-end*, comme le *framework* Flask, le langage JavaScript, le *Test Driven Development* ou encore jQuery, Bootstrap, HTML et CSS. Au début donc je me suis intéressé à l'étude de ces cours pour acquérir les bases sur lesquelles pouvoir démarrer sa conception et réalisation.

#### 2) Tableau Trello

Avant de commencer à coder, j'ai réalisé un tableau agile Trello ayant le but de diviser le travail en parties logiques et le rendre plus efficace.

J'ai opté pour la division en 4 *sprints*:

- a) partie théorique – monter en compétences comme décrit précédemment;
- b) réalisation d'un produit minimal répondant aux spécifications fonctionnelles;
- c) réalisation du produit final;
- d) déploiement en ligne.

#### 2b) Sprint 2 – Minimum Viable Product

J'ai commencé par interroger l'API de Google Maps et avoir ainsi une idée de comme afficher une carte interactive sur ma page web avec les données en json renvoyées.

Ensuite je me suis intéressé à l'API MediaWiki pour avoir en réponse du texte depuis Wikipedia. Au final j'ai commencé à créer les éléments du DOM à l'aide de JavaScript (images, fenêtre de dialogue entre GrandPy Bot et l'utilisateur etc...).

#### 2c) Sprint 3 – Produit Final

Ici j'ai construit le système de *words parsing* afin de retenir seulement les mots-clé pendant la recherche des résultats par GrandPy Bot. Pour le faire j'ai utilisé la base de *stop-words* mise à disposition et des ressources en ligne (notamment <http://pyparsing.wikispaces.com/Examples> et Stack Overflow m'ont permis de comprendre le fonctionnement d'un *parser* et comment le concevoir). Après les tests fonctionnels comprenant le parsing et l'appel aux API, je me suis consacré au *front-end* de la page web avec l'initialisation de tous les éléments du DOM et l'ajout du style à ma page grâce à CSS et Bootstrap.

En dernier j'ai pensé à l'aspect *responsive* à travers l'ajout des *media queries* dans mon feuille de style.

Pour terminer cette phase, j'ai testé toutes les fonctionnalités de l'application en local sur des navigateurs web différents (Chrome, Mozilla, Safari et Opera) ainsi que son côté *responsive* à l'aide des outils présents dans les paramètres des navigateurs.

#### 2d) Déploiement en ligne

En suivant les instructions sur Heroku, j'ai déployé la page web en ligne et testé ses fonctionnalités sur des dispositifs différents (tablette, ordinateur et mobile) en faisant attention que le parcours utilisateur était le même que celui décrit sur le cahier des charges.

#### 3) Test Driven Development

Tout au long du projet, j'ai essayé de développer en TDD, donc à la conception d'une nouvelle fonctionnalité correspondait un test avec Pytest qui devait être vert avant de continuer.

#### 4) Organisation du travail

Pour ce projet j'ai créé un environnement virtuel où j'ai installé tous les composants nécessaires à son bon déroulement.

Les dossiers relatifs à la production de l'application se divisent en:

- *utils* – contenant les scripts d'interaction avec les API de Google Maps et Wikipedia, ainsi que le système de *words parsing*;
- *static* – contenant les scripts pour l'appel AJAX, l'initialisation des éléments du DOM, les images utilisées dans ma page web, le feuille de style, les importations de Bootstrap et des fonts;
- *templates* – contenant le fichier HTML relatif à la page web;
- *tests* – contenant les tests d'appel aux API et de *parsing*.

#### 5) Difficultés rencontrées

- JavaScript: je croyais que sa compréhension était plus facile, alors que j'ai passé beaucoup de temps sur le DOM, l'initialisation des objets et son intégration à l'application.
- TDD: j'avoue que développer en ce manière ne m'excite pas car selon moi j'ai perdu beaucoup de temps à cause de la conception des tests à réaliser. Je pense c'est vraiment abstrait de concevoir des test avant encore d'avoir réalisé une fonctionnalité.
- Le système de *parsing*: bien qu'en mineur partie respect aux deux points en dessus, au début il a été difficile de comprendre son fonctionnement (d'où mes recherches sur le web).