# Projet 7: GrandPy Bot

**lien Github: https://github.com/Fabe59/Projet7**
**lien vers l'app : https://grandpubotoc.herokuapp.com/**
**lien Trello : https://trello.com/b/A2QNwUcj/ocp7-grandpy-bot**

## Introduction du projet

Dans ce projet 7 du parcours développeur Python sur OpenClassRooms, il nous a été demandé de créer une application web imitant un grand-père plein de connaissances et de sagesse.
Ainsi, lorsque nous lui posons une question relative à un lieu, notre papy robot répond (si il le connait), en nous donnant l'adresse du lieu, un description ou une anecdote sur le lieu, et affiche une carte Google permettant de bien le situer.
Pour cela, il a été demandé d'utiliser :

- Python pour le backend ;
- Ajax pour la réalisation des requêtes ;
- Utiliser les API Google Maps et Wikipédia.

## Organisation du projet

Ce projet a débuté par l'écriture de *user stories*. Celles-ci m'ont permis de penser à chaque objectif à atteindre pour avancer dans ce projet et ainsi créer des tâches et sous-tâches.

J'ai ensuite suivi la démarche TDD et écris les tests (test/test_models.py) avant d'écrire les fonctions correspondantes, en vue de les faire passer de FAILED à PASSED.

La conception a alors débuté par :

1/ le module « parser » :

L'objectif de ce module est travailler sur la phrase soumise par l'utilisateur à l'application afin d'en faire ressortir l'objet principal. Ainsi, on transforme tout d'abord la phrase en minuscule. On ôte ensuite toute ponctuation puis on split chaque mot de la phrase. Enfin, on supprime les mots non nécessaires à la recherche grâce à une liste de stopwords.

2/ le module « googlemaps »

L'élaboration de ce module a débuté par une recherche et une lecture sur son fonctionnement (inscription, clé, paramètres de requête). Cette étape passée, nous avons pu utiliser le sujet recherche indiqué précédemment par l'utilisateur pour réaliser une requête et ainsi récupérer l'adresse du lieu ainsi que ses coordonnées (latitude et longitude).

3/ le module « mediawiki »

De la même manière, la conception de ce module a débuter par divers lectures afin de comprendre le fonctionnement de l'API Wikipédia. Cette étape a demandé davantage de temps que pour l'API de google car la documentation m'a semblé moins claires. Néanmoins,

grâce à mes recherches et échanges avec d'autres étudiants, il est ressorti que l'obtention d'informations de la part de l'API Wikipédia devait se faire en 2 étapes :

1: récupérer la valeur de la clé « pageid » grâce aux coordonnées précédemment obtenues

2: se servir du « pageid » pour récupérer des infos sur le lieu de recherche de l'utilisateur.

### 4/ Flask

La suite du projet a consister en l'initiation d'un projet Flask avec une base HTML. Cette étape nous a permis de comprendre l'organisation d'un projet avec ce framework.

### 5/ Views.py et Ajax

Ajax (Asynchronous Javascipt And XML) est une techno javascript permettant de réaliser des mises à jour du contenu d'une page web sans qu'elle nécessite le moindre rechargement. Pour ce projet il nous a été demandé de l'utiliser afin de réaliser l'envoi et la réception du contenu du formulaire soumis par l'utilisateur.

Ainsi le script *views.py* contient une fonction *ajax()* qui va permettre l'instanciation des classes utilisées dans le backend de l'application (parser, googlemaps, mediawiki).

### 6/ Frontend

La partie frontend contient le HTML pour définir la structure de base de notre application. La partie CSS permet quant à elle une mis en forme plus esthétique. Enfin le javascript présent dans le fichier *javascript.js* permet un affichage dynamique des différents éléments de réponse obtenus par la requête Ajax.

## Difficultés

Plusieurs difficultés rencontrées durant ce projet :

- Créer les tests avant d'avoir créé le code.Afin de solutionner ce problème, nous avons suivi de nombreux cours afin de bien intégrer cette notion de TDD.
- L'API Wikipédia a été la seconde difficulté. En effet, la documentation sur cette API étant relativement peu claire et complexe. Il a fallu plusieurs recherches, lectures et tests pour aboutir à une réponse correspondant au test initialement codé.
- La troisième difficulté a concerné le fonctionnement d'Ajax et des promesses.
- Enfin, la dernière difficulté a concerné le mise en place du responsive design. La solution a été trouvé dans l'utilisation des media queries afin que notre application s'adapte parfaitement à la taille de chaque écran (smartphone, tablette, moniteur).

## Conclusion

En conclusion ce projet a été pour moi l'un des plus interessants et des plus complets depuis le début du parcours. Il m'a permis de renforcer mes compétences en Python, dans l'utilisation des API, et d'avancer encore dans l'amélioration de la structuration d'un projet.

Il m'a, de plus, permis de m'initier à la conception d'un projet avec le framework Flask, d'utiliser le javascript et ajax, et enfin de déployer mon projet grâce à Heroku.