P7 GrandPy Bot

Le projet est disponible à l'adresse suivante : https://grandpy-bot-cc.herokuapp.com

1 : Planification du projet

Pour planifier le projet, j'ai suivi les étapes décrites dans la description du projet. Les différentes étapes et sous étapes sont listées dans un tableau Trello. J'ai ensuite initialisé un repo GitHub.

2 : Initialisation de Flask

Après avoir pris connaissance du fonctionnement de Flask, je l'ai installé et créé une toute première version de site basique avec uniquement une page d'accueil en HTML brut. J'ai utilisé le Test Driven Development pour vérifier le bon fonctionnement du serveur Flask.

3 : Interface Utilisateur

Une fois mon site fonctionnel, j'ai créé le formulaire qui allait servir à entrer le texte utilisateur. Je me suis alors penché sur l'utilisation de Bootstrap afin d'avoir un design moderne et responsive.

J'ai ensuite étudié les possibilités offertes par JavaScript et JQuery pour traiter l'envoi des données du formulaire avec AJAX dans Flask. La première version fonctionnelle de l'appel AJAX prenait en entrée le message envoyé par l'utilisateur dans le formulaire, et en retour, l'affichait directement dans la zone de chat.

4 : Gestion du parser

Le parser a pour but d'analyser la phrase entrée dans le formulaire par l'utilisateur et d'en extraire les termes qui seront pertinents pour une recherche sur Google Maps et Wikipedia.

On prend en entrée le texte entré par l'utilisateur et on le modifie, tout d'abord en retirant tous les signes de ponctuation, et en passant tous les caractères en minuscules.

Ensuite on utilise une liste de stopwords pour savoir quels termes ne seront pas pertinents dans une recherche. On boucle sur l'ensemble des mots composant la phrase entrée par l'utilisateur, et si le mot courant est présent dans la liste de stopwords, on le retire de la phrase.

J’ai également utilisé le Test Driven Development afin de vérifier que les résultats correspondaient à mes attentes pour différentes phrases entrées, avec différents cas d’utilisation (Signes de ponctuation et apostrophe, répétition de mots, phrase constituée uniquement de stopwords…)

5 : Utilisation de l’API Media Wiki

J’ai choisi de commencer par l’implémentation des appels à l’API Media Wiki pour des raisons de simplicité par rapport à l’API Google Maps : l’API Media Wiki est ouverte et ne requiert pas de clé pour son utilisation, et le résultat que l’on attend en retour dans le projet est uniquement sous forme de texte. J’ai donc pris connaissance du fonctionnement des requêtes et du format de JSON retourné par l’API. J’utilise le texte entré par l’utilisateur après qu’il ait été parsé pour faire une requête sur Wikipedia en utilisant la recherche textuelle, en limitant le résultat à un seul article, et en retournant les 5 premières phrases de l’introduction de l’article Wikipédia.

Pour les tests de requête à l’API Wikipedia, j’utilise un mock qui contient le contenu JSON d’une requête simple.

6 : Utilisation de l’API Google Maps

J’ai ensuite étudié l’API Google Maps, tout d’abord la partie JavaScript, pour afficher une carte dans une page web. J’ai rencontré une difficulté car le code JavaScript de l’API Google Maps n’était pas chargé au moment où le code JavaScript de l’application se lançait, ce qui produisait des erreurs. Pour résoudre ce problème, j’ai utilisé la fonction JQuery getScript(), qui permet d’exécuter du code uniquement après chargement d’un script donné.

Une fois que j’étais en mesure d’afficher un message dans la zone de chat avec une carte incrustée, j’ai utilisé l’API Google Places pour obtenir les coordonnées géographiques d’un lieu en fonction d’une chaîne de caractères donnée. J’ai ensuite étudié les possibilités d’ajout de marqueur et de modification du zoom de la carte afin d’afficher les informations les plus pertinentes pour la demande de l’utilisateur.

J’ai également utilisé un mock dans mes tests pour le retour JSON de l’API Google Maps**.**

7 : Mise en ligne sur Heroku

Une fois toutes ces étapes terminées, lorsque le projet était complet en environnement de développement, je l’ai mis en ligne sur Heroku en suivant la documentation pour l’installation, la création du fichier Procfile, et le push du repo en production.

Améliorations possibles

Il serait possible d’améliorer le projet, par exemple en permettant à l’utilisateur d’avoir la liste des lieux qu’il a recherchés ainsi que le trajet entre ces lieux.