Parcours développeur d'application - Python

Projet 7 : Créez GrandPy Bot, le papy-robot

1. Description du projet

Le but du projet est de réaliser un site web statique permettant de simuler un conversion avec un robot. L'utilisateur entre une question et le robot lui répond en lui affichant :
- Une carte miniature récupérée via l'api google maps, pointant sur l'endroit cherché.
- L'adresse complète de l'endroit cherché.
- Un extrait d'article wikipedia concernant l'adresse du lieu.

Le tout doit s'articuler sous la forme d'une conversion de type « chat ».
Le site doit ensuite être déployé grâce au service heroku.
Enfin, le projet est réalisé en suivant une approche Test Driven Developpement.

2. Conception du Front-end

L'interface de l'application est relativement simple. Outre le header et le footer, le corps du site n'a besoin pour fonctionner que d'une zone de formulaire et d'une zone affichant le dialogue entre l'utilisateur et le bot et permanentant d'afficher la carte de google maps. J'ai utilisé le framework bootsrap ainsi que quelques media querries afin d'avoir un design responsive.
J'ai également choisi de séparer la carte de la zone de chat (où je l'avais initialement positionnée). En effet, la fonction de l'API google maps permettant d'afficher la carte nécessite l'utilisation de l'« id » « map » dans la balise HTML accueillant la carte. Un « id » étant unique sur une page HTML, en affichant systématiquement une nouvelle carte à chaque nouvelle question de l'utilisateur, les cartes apparues précédemment dans le chat se modifiaient et affichaient le résultat de la nouvelle recherche. C'est pourquoi j'ai fait le choix d'une unique zone unique pour afficher la carte.

3. Conception du Programme

Avant de concevoir l'application Flask, j'ai commencé par tester le fonctionnement des deux api (google maps et media wiki) de façon isolée, afin d'en comprendre le fonctionnement et d'utiliser par la suite uniquement les fonctions nécessaires.
Dans un premier temps j'ai laissé également de côté l'aspect AJAX et affichait donc le résultat des requêtes sur une nouvelle page web chargée après l'envoi du formulaire.
A côté de ça je me suis entraîné à créer le mécanisme du chat en javascript avec un affichage de la question de l'utilisateur puis une réponse standard de la part du robot, le tout créant une conversation dans la zone de chat.
J'ai ensuite fusionné ces deux parties puis je me suis penché sur l'appel des fonctions en AJAX via Jquery.

4. Algorithme de recherche

La recherche dans google maps ne pose pas de réel problème, l'outil est très efficace et un lieu donné n'a souvent qu'une adresse.

Par contre, afin que GrandPy puisse renvoyer une phrase issue de wikipedia en rapport avec la recherche de l'utilisateur, cela m'a demandé plus d'effort. J'ai choisi des règles de priorité dans les choses à rechercher via l'api media wiki. Ma recherche s'effectue donc selon les critères suivants :
1. L'adresse (celle renvoyée par google) sans le numéro ni le code postal. → Donne des réponses très précises pour les noms de rues dans les grandes ville.
2. Uniquement le nom de la rue → Permet d'avoir un résultat notamment pour les rue ayant un nom de personnage célèbre.
3. Les mots clés de la recherche de l'utilisateur → S'il n'y a aucun résultat relatif à la situation géographique, l'utilisateur peut au moins avoir une info sur le lieu qu'il cherche.
4. Les mots clés sans le nom de la ville → Permet d'éviter d'avoir uniquement une phrase relative à la ville elle même, notamment dans le cas où l'utilisateur rentre une question du type : « nom de rue + ville ».
Afin de pouvoir effectuer ces différentes recherches, j'ai dû créer plusieurs fonctions permettant de parser différemment, soit la phrase de l'utilisateur, soit l'adresse retournée par google maps.

5. Tests

J'ai choisi d'utiliser Unittest pour test mon projet. C'est un outil de la librairie standard de python. Le test des fonctions de parser étaient relativement simples. Mais cela s'est corsé pour tester l'utilisation des API via des mocks. Finalement, après une relecture attentive de la documentation, j'ai pu réaliser les tests pour ces fonctions.

6. Déploiement

Lors de la phase de déploiement j'ai du modifier la structure de mon projet. En effet, je me suis rendu compte de l'importance de la structure d'un projet python et de l'impact de l'endroit d'où est lancé le programme dans les différents imports du modules. Après quelques modifications dans plusieurs fichiers concernant les imports, j'ai finalement réussi à obtenir un application fonctionnelle une fois en ligne.

7. Pistes d'amélioration.

Le programme est encore largement perfectible et l'amélioration de celui-ci pourrait passer dans un premier temps par :
 - L'amélioration des résultats reçus de wikipedia. Il existe encore des cas particuliers qui ne renvoient pas de réponse au niveau de l'appel à l'API media wiki, ou qui retournent des résultats peu cohérents avec la recherche de base. Notamment, la recherche sur le nom de la rue ne donnera rien pour les rues contenant une date (ex : rue du 8 mai 1945) car pour le moment le parser supprime tous les chiffres. De plus, les cas d'homonymie sont encore mal gérés.
- L'amélioration du design du site qui est extrêmement « light » aujourd'hui.

Le site est herbegé sur Heroku et disponible à l'adresse suivante : https://grandpybot-opc.herokuapp.com/
L'intégralité des livrables liés à ce projets sont disponibles sur GitHub à l'adresse suivante : https://github.com/NanroYahel/P7_Creez-GrandPy-Bot-le-papy-robot
Enfin, le projet à été effectué selon les critères d'une méthode agile. Les différentes étapes de la création sont détaillées sur le tableau suivant :
https://trello.com/b/VWgpFgEY/p7grand-py-bot