PROJET 7- CRÉER GRANDPY BOT, LE PAPY-ROBOT

Lien trello : https://trello.com/b/pygroM0W/projet-7
Lien GitHub : https://github.com/yannhamdi/n.projet7
Lien Heroku: https://projet7-yh.herokuapp.com/

Dans le cadre de ce projet, nous avons comme instructions de créer une application qui permettra à l'utilisateur de demander des informations sur un lieu.
L'application devra renvoyer au client une adresse, afficher sur la page une carte du secteur concerné. Il devra fournir également quelques renseignements sur l'emplacement.
Ce document présentera comment j'ai décidé de procéder afin d'offrir une solution adéquate qui satisfera la cahier des charges du projet 7.
Je vous présenterai donc dans un premier temps, les difficultés que j'ai pu rencontrer, les solutions trouvées puis nous survolerons les différentes parties de mon projet.

DIFFICULTES :

Nous avons pour ce projet une nouveauté, nous devons gérer le coté client ; donc il faut penser à l'aspect visuel de la page web qui en plus doit être dynamique afin de réagir au comportement de l'utilisateur ainsi que le comportement de notre serveur.
Puis respecter les fonctionnalités du cahier des charges qui seront en grande partie gérer par notre serveur et notre langage python.
Mais je dois admettre que la partie la plus difficile aura été la communication entre mon serveur et mon client, même si « Flask » aura beaucoup contribué à faciliter l'exécution de cette tâche.
Nous allons donc voir les solutions trouvées pour réaliser de manière à respecter les instructions de l'énoncé.

SOLUTIONS :

La toute première chose que j'ai faite cela a été de suivre les instructions de mon mentor en commençant par faire toute la partie python (coté serveur) soit de faire fonctionner mon application coté console uniquement. On peut donc dire que mon projet a été diviser en plusieurs étapes succinctes.
Mon projet a donc été structuré en plusieurs parties afin de me rendre la tâche plus facile sans quoi, je risquais de m'emmêler les pinceaux…
J'ai donc procédé par étapes que je vais vous citer ci-dessous puis, je rentrerai plus en détails dans la présentation de mes modules.

- **Coder la partie python uniquement et rendre l'application fonctionnel uniquement au niveau back-end**
- **La partie html afin de créer le squelette de la page de mon site**
- **La partie CSS afin de donner un aspect plus esthétique pour l'utilisateur**
- **La partie Framework-Flask afin de permettre de faciliter la communication entre le serveur et client**
- **La partie JavaScript offrant de la dynamique à la page web et par le biais d'Ajax a permis d'envoyer les données à notre serveur**

DESCRIPTION DE LA PARTIE PYTHON :

Une petite nouveauté, développer en TDD, j'ai trouvé que cela facilitait relativement la tâche, du fait qu'on sait avant de coder, ce que l'on attend et sous quelle forme, même au départ j'ai éprouvé beaucoup de difficultés au niveau des mocks mais finalement, j'ai apprécié coder en utilisant le « TEST-DRIVEN DEVELOPMENT »

1. *Module « parsing.py » :* Ce module va traiter la question posée par l'utilisateur afin de supprimer les espaces, les mots inutiles (liste stop-word), et renvoyé une phrase « parsée » afin de faciliter le travail à l'api de Google-Map. Cette fonctionnalité était une instruction primordiale de notre cahier des charges…
2. *Module « googlemapapi » :* ce module va tout simplement récupérer la phrase « parsée » et effectuer une recherche en envoyant une requête ' l'api de Google Map. L'api nous renverra sous forme de dictionnaire JSON, des données, nous ne récupérerons uniquement que les informations dont on a besoin, c'est à dire l'adresse, ainsi que les coordonnées GPS du lieu recherché.
3. *Module « mediawiki.*py » : Ce module va effectuer une recherche à partir des coordonnées gps récupérées par le module « googlemap » et faire une recherche à proximité. Puis de manière aléatoire, il va choisir une page id et faire une recherche de cette page id et nous fournir des quelques lignes d'informations sur

PROJET 7- CRÉER GRANDPY BOT, LE PAPY-ROBOT

le lieu ainsi que l'adresse URL Wikipedia concerné afin de proposer à l'utilisateur d'avoir la page complète s'il le désire.

4. *Module « pybot.*py : ce module va uniquement préparer la réponse à envoyer à JavaScript en renvoyant la réponse sous forme d'un dictionnaire en récupérant tout ce dont il a besoin auprès des autres modules.

DESCRIPTION DE LA PARTIE HTML ET CSS :

J'ai regroupé la partie HTML et CSS, car les 2 langages sont étroitement liés. La partie HTML va construire le « squelette » de notre page et CSS va nous apporter l'esthétique nécessaire.
De ce fait, le code HTML est relativement simple ; l'énoncé nous demande d'avoir un haut de page, la partie centrale et le pied de page. Voici le « squelette » de notre page. J'ai donc utilisé les balises suivantes, <header>, <main> et <footer>, puis la balise <form> pour le formulaire ou notre utilisateur va entrer sa requête, puis j'ai utilisé quelques conteneur afin d'englober les réponses et faciliter ainsi la mise en page avec le CSS.
Le code CSS, lui va intégrer les images pour le footer, définir les tailles, permettre l'affichage ou non, de l'icône de chargement (display :none ou block).
Mon css va également définir la taille de la carte affiché par JavaScript.
Voilà ce que mon CSS fera dans l'ensemble.

DESCRIPTION FRAMEWORK FLASK :

Le travail de flask est de permettre le fonctionnement de l'application. Et surtout la communication entre JavaScript et le serveur par le biais des vues, on rentre l'url dans JavaScript et dans notre vue puis de ce fait, on pourra envoyer les informations à notre serveur, pour ce projet, j'ai utilisé AJAX (POST), qui enverra ce que l'utilisateur à écrit dans notre formulaire, ce formulaire est récupéré et envoyé au serveur qui traitera l'information grâce aux modules que nous avons codés.

DESCRIPTION PARTIE JAVASCRIPT :

Pour la partie JavaScript, j'ai utilisé le « DOM » afin d'afficher les réponses envoyées par le serveur.
Grâce à « queryselector », nous récupérons les informations du formulaire dont l'id est « questionform », puis dès que nous avons cette information il ne reste plus qu'à envoyer cet info à notre serveur en utilisant AJAX et ici par l'intermédiaire de « FETCH » qui envoi l'information et récupère les infos traitées. Une fois ces informations coté JavaScript sous forme de dictionnaire jsonifié, nous les traitons et les ajoutons à notre page web, en utilisant entre autres « appendchild » qui affichera les informations coté client de manière dynamique sans que l'utilisateur rafraîchisse la page.
Nous récupérons également les coordonnées gps afin de les envoyer à l'api Google map, qui nous renverra la carte qui sera affichée sur notre page web.
Puis une autre fonctionnalité, que JavaScript m'a permis d'effectuer est l'icône de chargement pendant que ajax attende la réponse du serveur. Effectivement, nous avions un display none du côté de CSS et avec l'aide de JavaScript, j'ai changé la classe de notre image afin d'obtenir un display block et ainsi faire apparaître l'icône jusqu'à avoir une réponse.
Voilà pour la partie JavaScript.

DEPLOYER SUR HEROKU :

Une fois, notre application fonctionnelle, il nous faut la déployer sur Heroku. Le déploiement est relativement facile, les seules vrais subtilités sont les environnement de variable qui nous font du coup faire quelques modifications sur nos modules python, dans notre cas pour la clé API afin de permettre le fonctionnement de l'application sur heroku et bien configurer le fichier « Procfile ».
Après avoir effectué ces configurations, on envoi vers heroku de la même que l'on enverrait sur github et notre site est en ligne.

Je dois dire que j'ai eprouvé beaucoup de difficultés pour ce projet, du fait de beaucoup de nouveauté à assimiler mais une fois réalisé, je suis fier et content d'avoir compris comment communiquer le coté client et coté serveur et j'ai vraiment hâte d'approfondir ces connaissances car elles semblent infinies tellement il y a des choses à apprendre.