Parcours développeur d’application - Python

Projet 11 : Améliorez un projet existant en python.

1 Description du projet

Le projet 11 de ce parcours a pour objectif de nous faire revenir sur un projet anciennement effectué et d’y ajouter une nouvelle fonctionnalité. Le but étant d’ajouter cette fonctionnalité en suivant les bonne pratiques liées à la reprise d’un projet existant.

2 Choix de l’amélioration à apporter

Pour réaliser ce projet, j’ai choisi de revenir sur le projet numéro 7 (« Créez GrandPy Bot, le papy-robot ») et d’ajouter une fonction permettant à l’utilisateur de consulter une nouvelle page web lui indiquant des informations sur les 10 dernières demandes faites à GrandPy n’ayant données aucun résultat (date et mots clés de la demande). J’ai également choisi d’afficher, sous forme de graphique, le nombre total de demandes sans résultat par rapport au nombre total de demandes effectuées sur l’application.

3 Démarche suivie pour la réalisation de la fonctionnalité

3.1 Ajout d’une base de données

Le point central de l’ajout de cette fonctionnalité est la mise en place d’une base de données permettant de stocker les demandes effectuées par les utilisateurs. En effet, jusqu’alors le projet numéro 7 était une application web réalisée avec Flask mais qui n’utilisait aucune base de données. J’ai donc commencé par effectuer des test d’utilisation de base de données dans un projet Flask puis j’en ai ajouté une à mon projet.

Pour le développement de la fonctionnalité, j’ai utilisé une base de donnée SQLite qui nécessite moins de paramétrage qu’un SGBD classique tel que MySQL ou PostgreSQL. Cependant, la plateforme Heroku sur laquelle est déployée l’application par la suite ne supporte pas SQLite, il a donc fallu anticiper cela pour le déploiement.
Afin de faciliter la passage d’une BDD à l’autre, j’ai également utilisé l’ORM SQLachemy, qui permet d’effectuer toutes les manipulations de données avec de objets Python et qui se charge par la suite de traduire les opérations en requêtes SQL.

3.2 Utilisation de la librairie 3D JS

Afin d’afficher les données sous forme de graphique, j’ai du utiliser une nouvelle librairie Javascript nommée 3D JS. Il m’a fallu prendre en main cette nouvelle librairie avant d’ajouter la fonctionnalité voulu. En m’aidant d’un tutoriel et de la documentation officielle, j’ai commencé à réaliser différent graphiques pour comprendre la base de la réalisation d’un graphique en svg et son affichage sur une page web.
Après avoir réussi à afficher le graphique voulu avec des données factices, j’ai rajouté le code javascript à mon application en lui passant les données voulues. A savoir, nombre de demandes sans résultat et nombre total de demandes faites sur l’application.

3.3 Nouveau déploiement sur Heroku

J'ai choisi d'effectuer un nouveau déploiement sur Heroku pour garder les deux versions de mon projet (le P7 et le P11), j'ai donc du procéder à un nouveau déploiement avec une base de données.
J'ai profité de ce nouveau déploiement pour réorganiser mon application Flask.
J'ai dû recommencer le déploiement à plusieurs reprises en raison de quelques oublis pour permettre au serveur heroku d'utiliser la base de données PostgreSQL avec l'application (oublie du module psycopg2 dans le « requirements.txt », ajout de la commande «flask db upgrade » dans le fichier « procfile »…).
Malgré les quelques échecs de déploiement, cette étape n'a pas réellement posé de grosses difficultés. Les différents échecs étaient toujours liés à de léger oublis que j'ai réussi assez rapidement à diagnostiquer et corriger.

4 Tâche annexe : Résolution d'un bug

En parallèle de l'ajout de fonctionnalité, la réalisation du projet demandait d'effectuer un « mini » jeu de rôle qui consistait en la découverte d'un bug par le client, et la mise en œuvre de la résolution de ce bug. Le tout devait être retracé dans une conversation par e-mail.

J'ai choisi de traiter cette partie après l'ajout de la nouvelle fonctionnalité.
Pour simuler le bug, j'ai fait volontairement une faute de frappe dans la fonction de la vue « google_api » permettant d'enregistrer la requête utilisateur dans la base de données. Cette erreur bloque l'exécution des opérations suivantes de la vue et l'application n'affiche donc plus rien après que l'utilisateur a fait sa demande.
J'ai ensuite, corrigé cette faute de frappe et refactorisé le code de la fonction comme indiqué dans la consigne, puis j'ai ajouté un test unitaire permettant de voir si l'appel à la vue s'exécutait bien.
Enfin, j'ai rédigé les mails que j'aurais écrits si cette situation c'était présentée dans un cadre professionnel.
Tous ces éléments sont consultables dans les livrables du projet.

5 Livrables associés au projet

L'application web est disponible à l'adresse suivante : https://newgrandpybot.herokuapp.com/
Le code de l'application et les livrables en lien avec le projet sont disponibles sur GitHub à l'adresse suivante : https://github.com/NanroYahel/P11_Ameliorez-un-projet-existant-en-Python