GrandPy Bot / Projet 7

Objet de ce document :

Expliquer la démarche choisie, les difficultés rencontrées et les solutions trouvées pour réaliser ce projet.

Demarche :

1 ere étape : acquisitions techniques

La réalisation de ce projet exige de nouvelles compétences techniques nécessitant la l'assimilation des cours proposés, la complétion des exercices et quelques recherches personnelles pour parfaire les acquis.
Ces «découvertes» comprennent notamment la gestion du **front** et de son triptyque **JS/CSS/HTML** mais également les principales fonctions du serveur **Flask**, les test dans l'intégration continue, le déploiement de l'application sur des **services** tels que **Heroku,** Amazon ou Gcp, l'assimilation du fonctionnement des **API tierces** (**Google Place**, **Wikimedia**).

2 eme étape : démarrage du projet

Organisation du projet en local.
- Initialisation du dépôt du projet sur Gihub initialisation d'un suivi de projet type Kaban sur Jira
- Validation des tests automatisée avec Circle-Ci.
- Réalisation d'une application fonctionnelle de type «hello world» avec installation d'un environnement de test minimal.
- Mise en place des» **mocks**» et des tests unitaires.
- Conception de la présentation, du plan de la page d'index, définition du «**user story**» et de la stratégie de constitution de la réponse.

3 eme étape : développement des fonctionnalités, 1ers déploiements

- Instancier les objets «Question» «Reponse» + interactions API.
- Activer les échanges avec les différentes APIs (gestion des **API_KEY** pour Google).
- Gérer l'aspect «responsive» du site. (tester les configurations bootstrap).
- Gérer les problématique de mise en production et les spécificités de l' environnement Heroku (interface Gunicorn, initialisation des «dynos»).
- 1er Tests fonctionnels. Gestion des logs.

4 eme étape : mise en production

- publier et tester l'activité de l'application sur le Web.
- inviter testeurs «externes», mettre le mentor à contribution.
- constituer jeu de test nominal (questions pieges)
- Corriger les problèmes remontés.
- Vérifier respect pep8, pylint, documentation.

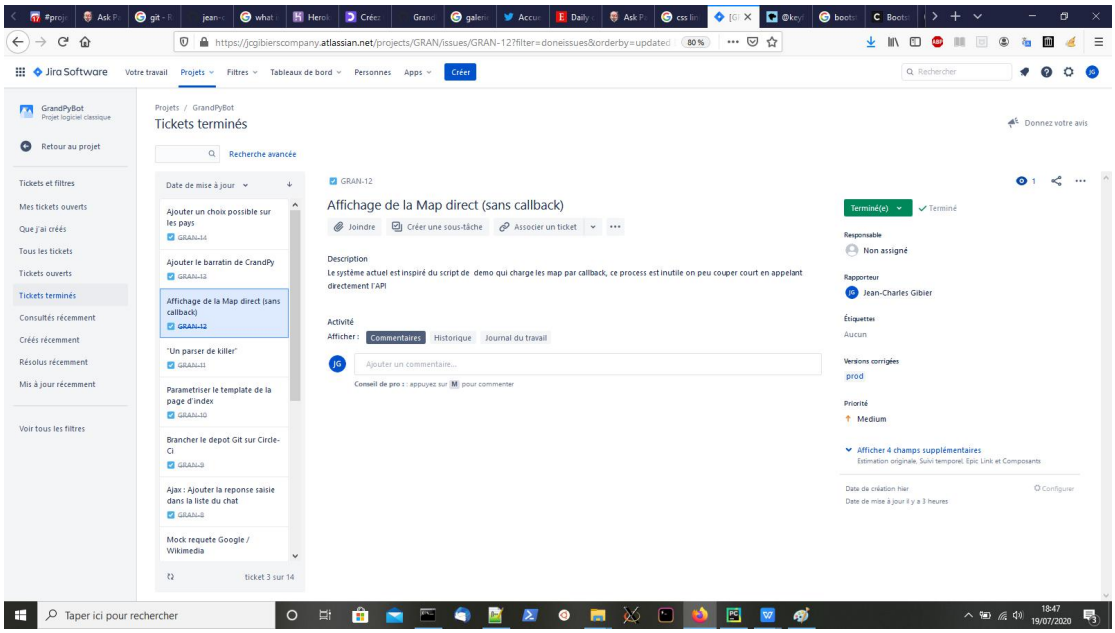
5 eme étape : Livrer

... sans commentaire.
Déploiement accessible sur :
https://yagp.herokuapp.com/

Yagp : comme «Yet Another Gran Py».
Repository Git : https://github.com/jean-charles-gibier/GrandPyBot/blob/master/README.md

Acces Jira :
https://jcgibierscompany.atlassian.net/projects/GRAN/issues/GRAN-12?filter=doneissues&orderby=updated%20DESC

(alias «Trello» (Accès privatif Me demander un accès en envoyant un adresse mail)



Les difficultés rencontrées

Les difficultés ayant mobilisé plus de temps à trouver une solution :
- La gestion des API-key google :
le paramétrage des différentes applications (Google `API place` ,«`findplacefromtext`» , pour différents environnements (local, Heroku) avec différentes protections/restrictions (selon le service, selon l'IP, selon le «refer») à été assez fastidieux à calibrer. Il n'y a pas de configuration uniforme, donc pas de solution universelle.
En occurrence j'ai créé 3 clés différentes :
1 pour le chargement des cartes sur le navigateur (restriction à l'API).
1 pour la fonction «find place» en mode développeur (restriction ip + API)

1 pour la fonction «find place» en mode production (referer + API)

- La mise au point d'une interface conviviale et «responsive» en mode «chat-box» avec des messages s'empilant en sens inverse de leur apparition avec une justification gauche / droite en fonction de l'intervenant Utilisateur / robot. Affichage comprenant l'affichage de la map google. Pour ce faire j'ai procédé empiriquement en testant plusieurs solutions type Boostrap glanées sur le net, et avec l'aide de mon mentor pour des problèmes d'affichage des Maps (certaines s'affichant mal ou partiellement à cause d'un rechargement multiple (donc inutile) de l'API JS Google).