Maxime J.

Projet 7 Openclassrooms : Créez GrandPy Bot, le papy-robot 🤔

Livrables :

- Document texte expliquant la démarche choisie, les difficultés rencontrées et les solutions trouvées. Le document doit être en format pdf et ne pas excéder 2 pages A4. Il peut être rédigé en anglais ou en français, au choix, mais prenez bien en considération que les fautes d'orthographe et de grammaire seront évaluées !
- Adresse du site déployé pour consulter votre projet "en vrai" et poser des questions à GrandPy (les mentors aussi ont le droit de s'amuser !) : https://grandpybotte.herokuapp.com/

- Code source hébergé sur Github : https://github.com/Maximedu13/Cr-ez-GrandPy-Bot-le-papy-robot
- Tableau Trello ou Pivotal Tracker contenant les user stories : https://trello.com/b/bpszZ58s/grandpy-bot

I. Generalities

The aim of this project is to create a website providing access to information related to a question raised by the user. He enters a question and GrandPyBot answers him displaying:

- An extract from the Wikipedia page concerning the address of the place retrieved via Wikipedia API.
- A map retrieved via Google Maps API pointing to the place the user is looking for.

Nothing is saved and the user sends his question by pressing enter and the answer is displayed directly on the screen, without reloading the page.

The program is coded in HTML, CSS, JavaScript and Python on Visual Studio Code 1.33.1. The front-end is developed thanks to the Framework Bootstrap and the back-end thanks to the Framework Flask.

This program is hosted on GitHub, here is the link : https://github.com/Maximedu13/Cr-ez-GrandPy-Bot-le-papy-robot and on Heroku server, here is the link : https://grandpybotte.herokuapp.com/

II. Content of the project

Flask gives more freedom (than Django for example) to allow each developer to organize himself as he wishes.

The project contains the following programs and files:

- a repository GrandPyBot : contains the application itself.

- a repository tests : this folder contains the unit tests.
- README.md : a file containing information about other files in the same folder.
- config.py : the configuration file.
- requirements.txt : containing a list of commands for pip that installs the required versions of dependent packages.
- run.py : this file is used to run the Flask application.
- Procfile : useful to specify the commands that are executed by the app on startup.

III. Approach chosen

It was requested to adopt the Doc Driven Development approach, in other words to, respectively, determine the list of features to provide, write the complete documentation, write the code, check if the doc is well respected, and finally iterate.

First, I wrote the steps to follow that I thought I would do for this project. To remind me the tasks to achieve, I created a Trello table, available at this link : https://trello.com/b/bpszZ58s/grandpy-bot. Thus, I divided my program in user stories, and in tasks, but I didn't affect deadlines to my table.

Then, I initialized a Github repository, made my first push and wrote the documentation.

As the same time, I educated myself on the use of Flask by taking the course on OpenClassrooms, on the use of AJAX reading the course on codehandbook. As for AJAX, I found it easier than I expected.

Thirdly, I designed the application front-end via Bootstrap. This design is relatively quick to build: a header with a logo and catchphrases, a footer with my name & surname, and a link to my Github repository and my Twitter. The central area is

made up of an empty area (which is used to display the dialog) and a form field to send a question. Moreover, the website is responsive.

Fourthly, I started to design the back-end by testing the performance of the two APIs (google maps and media wiki) separately. Whether the use of media wiki wasn't a problem, the google maps API is more complicated to use. Afterwards it had been necessary to build an algorithm able to save the useful words, and able to delete those which were unnecessary.

Then, I had to test this parser. So, I chose to use Unittest for my project. It is a tool of the standard python library. Once the tests go green, I deployed my website on Heroku, but I had to take into consideration a problem related with the logs.

I finally succeeded to get a functional app online.

IV. Algorithms choice

In this part, is developed one of the algorithms used in the program.

The parsing algorithm:

It's useful to break long sentences into words, which will be analyzed to keep only relevant key-words (an address for example).

First, we need to pick up a list of stop words from this link : https://github.com/6/stopwords-json/blob/master/dist/fr.json

The method allowing that is parse_mg which has two parameters: self and message. First of all, we create an empty list list_word_input. Then, we call the method lower() on the message to put it in lowercase. We replace all apostrophes with spaces. Through a loop for, we insert each word in the list. If a word from this list is present in the list of stop words, this word is removed from the list. Then, a method join is used to join all items in the list into the message. Finally, the message is capitalized via the method title().

V. Difficulties encountered and solutions found

Many miscoding (errors) were obstacles to the proper functioning of the program:

✗ {"error_message" : "You have exceeded your daily request quota for this API. If you did not set a custom daily request quota, verify your project has an active billing account: http://g.co/dev/maps-no-account",

"results" : [],

"status" : "OVER_QUERY_LIMIT" }

? Almost every Google API has a daily free quota. For Google Places API it's 2500 requests per day and 10 requests per second.
✓ Enable billing.

✗ Launch the unit tests
? How?
✓ python -m tests.test

✗ 'Application Error:
An error occurred in the application and your page could not be served. Please try again in a few moments.
If you are the application owner, check your logs for details.'
? gunicorn is not using the correct port. Heroku assigns a port for the application.
✓ First, create a PROCFILE file and fill it with this instruction:
web: gunicorn GrandPyBot:app

Secondly, add gunicorn==19.9.0 to requirements.txt
Thirdly, check Heroku logs. Fix them.

VI. Possible improvements

To improve the project, it would have been possible to have recourse to a JavaScript framework such as Vue.js, as we used respectively the frameworks Bootstrap for HTML/CSS, and Flask for Python. The

advantage of Vue.js is to be easy to understand and to develop Applications. It facilitates the developers to integrate with the existing applications.

It will still be possible to improve the parsing algorithm. In fact, there are still special cases that do not return any response. For example, the search on the name of the street will not give anything for the streets containing a date (ex: Rue du 11 Novembre 1918) because the algorithm removes each numeral.

These improvement tracks could be the subject of the project 11 on Openclassrooms “Improve an existing project in Python”.