Project 7

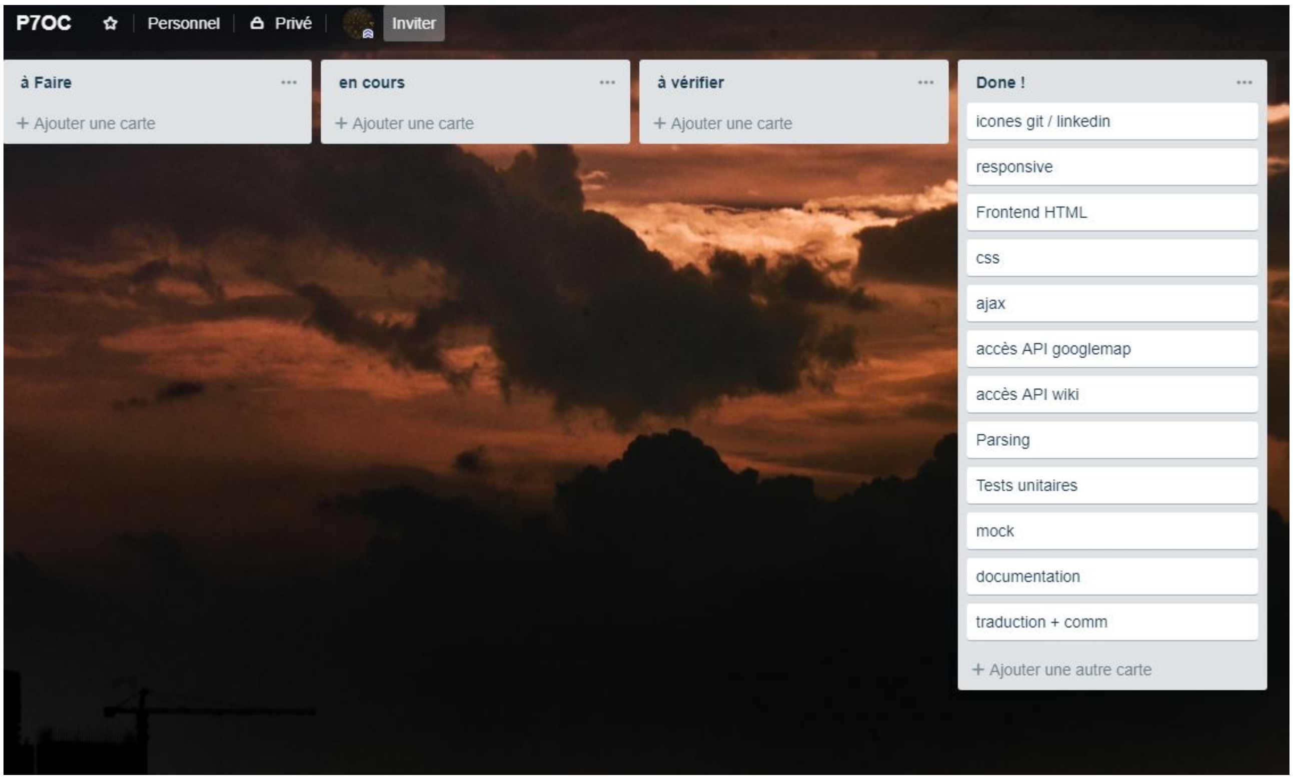
GrandpyBot, the Elder Robot

Camille Maury

https://github.com/Sik4/P7-Papy

https://p7-papybot.herokuapp.com/

Trello :



à Faire
+ Ajouter une carte

en cours
+ Ajouter une carte

à vérifier
+ Ajouter une carte

Done !
- icones git / linkedin
- responsive
- Frontend HTML
- css
- ajax
- accès API googlemap
- accès API wiki
- Parsing
- Tests unitaires
- mock
- documentation
- traduction + comm

+ Ajouter une autre carte

What is Grandpy Bot ? :

Grandpybot is a web page with several functionalities :

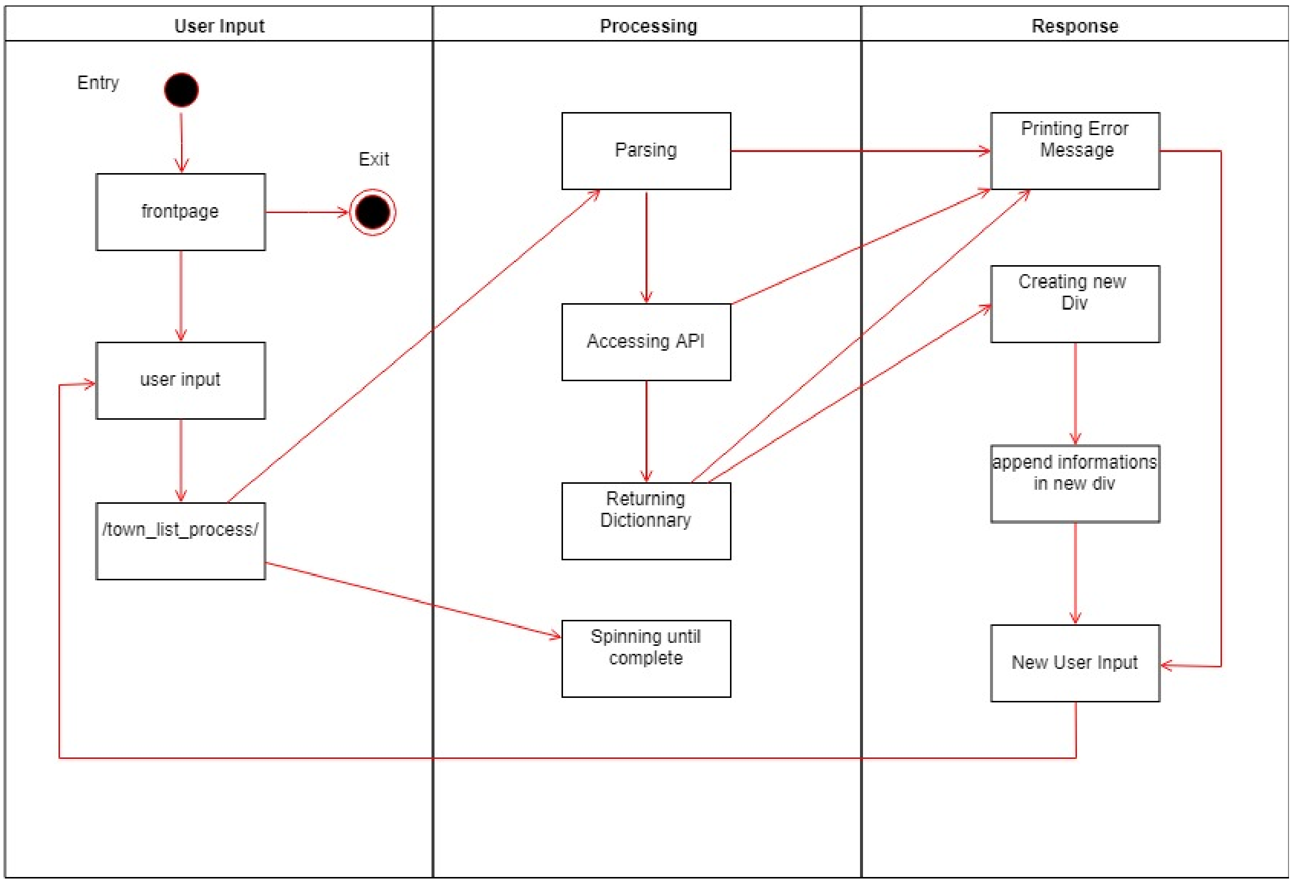
- The User open his web browser, enter the given url, and find a header, some presentation, and a form where he can ask any question about a location to Grandpybot (some rules applies).
- GrandpyBot will parse the question, find the information about the location thanks to regular expressions, and through an Ajax request, interrogate the GoogleMaps and MediaWiki API's to return the exact map (with marker) of the location, and an anecdote provided by Wikipedia.
- While this all process take place, the website will not refresh the page, and will display a beautiful loading animation.
- If the User want to ask several questions, they are going to be displayed one above the others, without any use of Databases. If the User want to refresh the web page, nothing will be saved.
- The website is of course responsive and works on phones and portable devices. The application will be tested, with unitary tests and mocks for the API's.

How does it work ?:

- The main page as a form using a SUBMIT button asking for an user prompt
- The click or Enter is handled by a javascript function, first preventing the default behaviour, then launching the loading animation.
- Papycore.py will handle the /town_list_process route, and will apply the functions of the class "Research" on it.
- The class Research, in the file api_request.py, will execute several operations on the user_input.
- First, from parser.py, it will execute the extract function from the class "PlaceExtractor". The parser will transform the phrase submitted by the user into a word presumably the location asked by the user. The strategy here is in 6 steps (turning user_input into lower case , remove the accents, finding the key words between some fixed start words often used in questions and a punctuation sign, removing all special characters, removing a list of "stopwords" given in a separate standardized json file, and finally removing all words shorter than 3 characters).
- The app will then initialize the google maps and MediaWiki API's, before sending them a request via the GET method, and transforming the data's into a dictionary.

- Thanks to the Ajax method, the javascript function will then create a new div for each time the user submit a form, prepending the results as follow : Grandpy will announce what is the location asked by the user, find a location nearby and submit the summary of the wikipedia page, with a link to the full version, then append an interactive map of the location, with a marker centered on the geocode of the location.

  The user may add any number of request, they will all be displayed as long as he doesn't refresh the page.



Problems :

At first, my approach was completely different. I wanted to cross-check the user_input with a json file with major cities names and location, then make a call to google static map API and wikipedia page, but it was hard to mock, and not a direct call to the API's.

Evolution :

The two main features that could be improved on this project are the parser (more possibilities to understand the questions), and the display of the new divs with a Jinja2 template (already implemented).