Grand-Py web application

Context

The application allows to find the address of a place specified by the user and give also a little description of an interesting site around. The Google Maps and Wiki Media services are used.

Features

- **Dialog view**: The user's query is keyed into a text input field and a dialog box displays the user's query and Grand-Py's reply like chat messages.
- **Identification of the searched place**: The specified place is identified from the user's query sentence.
- **Localization**: Determine the address, the coordinate and the map of the specified place.
- **Place description**: Get description of a point of interest near the specified place.
- **Online hosting**: Access to the web application online.

Architecture

Front-end

HTML / CSS: Content and layout

- **Header**: Displays Logo, title and pickupline
- **Central zone**:
    - The *Dialog box* is a void zone displaying messages from Grand-Py and the user.
        - Loader (Spinner) is an infinite loop animation coordinated by `keyframes`. The animation will be started and interrupted asynchronously according to HTTP request states.
    - The *Query field* is compound of a text input to type a query and a submit button to validate the user's query.
- **Footer**: Lists the author's name and links to contact, source code (Github repository), project planning and social networks.
- **Responsiveness**: Maximal and minimal blocks size are defined and CSS `flex` property is used to stretch block if necessary.

JavaScript: Dynamic update of the application view and interface (data exchange) with the back-end

On submit event (~~enter key pressed or submit image click~~), the user's query is catched from input form and displayed into the dialog box as a text block. The `innerHTML` method is used to update the view content without refresh the web page. Then, a spinner animation is started to simulate Grand-Py's reflection at the same time the query is sent as form data to the flask server via a AJAX post request.

When the response is received asynchronously from the server, a callback function extracts the received JSON-formatted data and displays the Grand-Py's reply and the defined map if necessary into the dialog box.

Back-end

Query analysis: identification of requested place

The goal is to identify the requested place from the user's query received from the front-end side.
The punctuations signs defined in the python `string` class and the most common *French* stop words are discarded from the sentence. In addition, a list of specific stop words such as [*adresse*, *situe*, *où est*, *trouve*, *cherche*, *aller*, *connais*, ...] are defined to delimit text block that could be a requested place. Finally, the only one remaining text block is considered as the searched location. If more than one text block are remaining, a random pre-defined Grand-Py's reply is returned to the front-end, telling the query is not understood.

Localization of a specified place/site

Geocoding

The Google Maps Geocoding API is used to locate the identified place.
The place name is sent to the geocoding service via a HTTP request, as a result the standard address and the coordinates (longitude/latitude) are returned as JSON-formatted data. In order to improve localization result, taking into account that the user is French, the local language as '*french*' and the preferential search region as ccTLD code '*fr*' are also specified to the service.

Map

Knowing the geocode (coodinates), the relative map can be got by sending a HTTP request to the Google Maps Static API. By specifying to the service some parameters like image size, zoom level and coordinates of the pin, as a result the image URL of a static map (non-interactive) is returned.
Then this URL can be transfered to the front-end side without the Google key for security.

Media Wiki

The geocode is also used to get back from MediaWiki API the descritption of points of interest near the place. By sending to the service a HTTP `GET` request as `query` operation, and specifying the *coordinates* and the *maximal perimeter*, a list of Wikipedia page identifiers is returned sorted by distance from specified coordinates.
So, a random page can be selected in order to return Grand-Py's replies more diversified for the same user's query.

With another request speciying the selected *page identifier*, the page text of the point of interest is returned. So, by discarding titles, the first sentences are extracted and used to complete the Grand-Py's reply to the user.

Random reply

For each Grand-Py's reply, predefined text is randomly selected to embellish information returned to the user.

Resources

- Web application
- Planning
- Source Code