|  |
| --- |
| Photo displaying partial image of two pie charts on a canvas-textured page |
| Global Pollution Interface (GPI)  Written report of the second coursework from advance web technologies in Napier University. |
| |  |  |  | | --- | --- | --- | | Vazquez, Adrian | 11/30/16 | Advanced Web Technologies | |

# Introduction

The objective of the project is to create a web application with python flask, in a learning environment Linux-based server set-up, called Levinux [1]. We must demonstrate a good level building python flask applications, covering aspects such as routing, URL hierarchy design, static files, request, templates, redirects, responses, templates, sessions, logging, testing, CSS, JavaScript, multiple users, and data storage. Also, I have implemented other technologies like JSON APIs, and AJAX.

All the source code has been uploaded to a Git repository [2] where is available also the developing process or “commits” that I updated during the project where other possible developers can see helpful messages about the changes that I made in the code. Also, this repository has a README file where explain how set up the web application.

This is a web application which collect information from an API [3] called OpenAQ [4] where collect data from agencies or governments all over the world about pollution in their countries and cities. At this moment, there are 33 different countries and 4.450 cities with different pollution measurements and we can display these data on the web application.

There are different pollution measurements like:

* PM2.5 [5]
* PM10 [6]
* Ozone (O3) [7]
* Sulfur dioxide (SO2) [8]
* Nitrogen dioxide (NO2) [9]
* Carbon monoxide (CO) [10]
* Black carbon (BC) [11]

Inside the website also explain briefly these elements, how they are produce and the impact on health’s population. Not all the cities have these measurements, there are some which just have one are there are some cities that have most of them, and that’s depends on the agency or government who provide the data.

Also, the APIs has a limitation of 1000 request, for that reason, is time limited to the last 1000 measurements which cover around the last two weeks.

A line chart displays these data where *x* represent the time, and line *y* represents the value, depending on the measurement minimum and maximum. For the line chart, I use Morris.js [12] which is an API which generate the graphic wanted with a JavaScript called.

Also, I have implemented a database with for users, which a user can register, log in into the platform, and save their favourites locations.

# Design

Although the information provided is the most important on the web, also how it looks, the interactivity and the user experience is crucial.

For this reason, I decided to user Bootstrap framework [13] which I can easily structure the content and for the colour, fonts and general aspects of the web I have chosen a theme [14], called “Flatly” [15] for my project. I think this theme give the web a serious aspect, a very clean layout, with very white spaces and consistency of the different elements such as buttons, links, tables through the website.

## Home page

|  |  |
| --- | --- |
|  | 1. The navigation bar fixed at the top of the window. The most common layout on internet offers make the user aware of the different pages of the website. 2. A breadcrumb is a very useful element which inform the user in which page of the site is now. It is separate from the navigation bar and the content. 3. This element tries to attract the attention of the user, and will inform the user briefly what is the page about. 4. A block containing information about the different pollution types, with a blue header that I use also for buttons and links. 5. The table of countries available. |

## Country

|  |  |
| --- | --- |
|  | 1. This table contains the information related to the country chosen. To maintain the consistency through the site, all the table have the same patter.   Also, I could display more information, such as last updates, first reading, or number of readings available. But it will be difficult for the user to read and I put what I think are the most important. |

## Location

|  |  |
| --- | --- |
|  | 1. This is a button which will save the location for the user, has a Glyphicon [16], of a heart which mean “Favourites”. If this heart is full, is because was added already but if its empty, it can be added. 2. Headers sorted by importance are the elements displaying in these part, which contains the location, city, country, last update, and source. 3. The parameters available also contains a Glyphicon which represent a line chart. 4. On the left the line chart with a darker blue and grey lines to determine the values range. |

## Registration/Login

|  |  |
| --- | --- |
|  | 1. The register and login form are on the same page to reduce the number of pages. On the left side, the user can see the register form which inside each field display the name field required. Also, has a very bright to button to catch the attention of the visitor. 2. On the right side, it displays the login form. This distribution is because normally users pay more attention what on the top left, and they need to be register before login, therefore the next time the user need to log in will now the login form will be on the right. |

## Profile

|  |  |
| --- | --- |
|  | 1. The update form is consistent with the other forms on the website but has a different colour button, green, which is very common for updates. 2. Location panel, on the right, will display all the favourites locations of the user, and it has again a bright blue to attract the attention of the user. |

# Web structure

The web structure can be divided in three parts:

* Information pages
* API/Data
* Users

### Information pages

**/ about:** The about page contain information related to this project, the purpose, objectives, link to the git repository and information about me.

**/ api:** The api page explain a little bit more about the how I retrieve the information from this API, and explaining about that organization.

**/ 404:** This is a custom error page and it triggers when a page is not found.

**/ 401:** The is a custom error page and it is display when a user does not have permission to see that page.

### API/Data

**/ index:** The landing page, which contain an introduction of the website, the possible measurements that you can see through the website and a table with the countries, number of cities available and the button which will send you to the country chose.

**/ country:** This page contains a list of cities of the country chose. It displays the country, the city, the location (a city can have different location) the agency source of the measurements and the pollution parameters available for that location.

**/ city:** This page has four different parts:

* First at the top we can see a button where you can save the location if we are login or the question **“*Do you want to add it to your favourites? Register here.*”** If you are not login.
* Also at the left is displaying information about the location we have chosen like location name, city, country, last update of the measurements and the source.
* On the right is displaying buttons depending on the parameters available.
* They will display the fourth part of the page where we can see on the left a line chart with the values represented as points and on the right a small brief explaining the pollution particle.

### Users

**/ login:** Login page contains two parts: One for registration, and other for login.

**Registration** form, anyone can register, the user need to fill the form which ask for first name, last name, email, password and repeat the password. There are security measurements to complete the register well, for instance, if the user does not type an email or the password does not match, we will see an error message and what caused the problem.

**Login** from will check if the username is in the database, if this is correct and the password provide is the correct too, it will redirect the user to their profile page.

**/ profile:** The profile page has two parts: On the left, we can see a form to update the user details. These details can be updated but also has the same requirements as registration form, the email is a requirement field and the password must match to update. On the right side, it will display all the location saved for that user. These locations provide direct links to them.

**/ logout:** This page will redirect to the landing page, it will clear the session which will log the user out.

# Enhancements

Although, the most of the objectives of web application are covered, there are few improvements I would like to add.

I think I should create a better structure for the web, for example: ***/GB/Edinburgh/Edinburgh St Leonards*** and now is ***/country***, and ***/city***.

Also, it does not work perfectly well the favourites features because it does not change the text between ***“Add”*** and ***“Remove”***.

I would like also to improve the info or error messages, when for instance, the user login wrong, or the passwords do not match, because now it is attach to the navigation bar and I think it would be better place below the breadcrumbs and before the content.

There are two major improvements I did want to do but I had not time to research the right solution. Give a different colour for different types of measures such as, red for CO or yellow for PM2.5 and a scale of minimum/maximum for each one of the measurements, which now it does display just for PM2.5. I could not make these improvements because I had no time to make a deep research about these elements.

# Critical evaluation

A critical evaluation of your web-­‐app

# Personal evaluation

In my opinion, the objectives of the module have been covered. In my opinion I have learned very well how to build web applications with python flask, how to use routing, URL hierarchy design, static files, request, templates, redirects, responses, templates, sessions, logging, testing, CSS, JavaScript, multiple users, and data storage.

Also, I have learned or improved new technologies such as JSON APIs, and AJAX. Researching a good API service that provide the data that I wanted was very difficult, and was even harder to understand how to gather this information with python [17], and, I had to learn how to retrieve this data on JavaScript using AJAX objects to use it with Morrison.js which displays the line chart.

Learning and developing these elements were the most time consuming during this project.

I believe that I improve my coding abilities, also I improved how I manage the time I had which I think it reflects on the project and in this report.

# Resources and references

A summary of resources used and a list of references

## References

1. <http://levinux.com/#1>
2. <https://github.com/thetwentyseven/GPI>
3. <https://en.wikipedia.org/wiki/Application_programming_interface>
4. <https://docs.openaq.org/>
5. <http://laqm.defra.gov.uk/public-health/pm25.html>
6. <https://www.quora.com/What-is-the-difference-between-PM2-5-and-PM10-with-respect-to-the-atmospheric-pollutants>
7. <https://diamondenv.wordpress.com/2010/12/10/particulate-pollution-pm10-and-pm2-5/>
8. <https://www.epa.gov/so2-pollution>
9. <https://www.environment.gov.au/protection/publications/factsheet-nitrogen-dioxide-no2>
10. <http://www.nhs.uk/Conditions/Carbon-monoxide-poisoning/Pages/Introduction.aspx>
11. <http://www.euro.who.int/__data/assets/pdf_file/0004/162535/e96541.pdf>
12. <http://morrisjs.github.io/morris.js/lines.html>
13. <https://getbootstrap.com/>
14. <https://bootswatch.com/>
15. <https://bootswatch.com/flatly/>
16. <http://glyphicons.com/>
17. <http://stackoverflow.com/questions/40663186/create-in-python-a-dictionary-from-json-url>

## Resources:

Stackoverflow: <http://stackoverflow.com/>

Teaching materials: <http://siwells.github.io/teaching_set09103/>

Flask documentation: <flask.pocoo.org/>

WinSCP: <https://winscp.net/eng/index.php>

DB Browser for SQLite: <http://sqlitebrowser.org/>

SQLite3: <https://sqlite.org/>