Python Web Framework

Carlos Matheus Barros da Silva[1]

Igor Bragaia[1]

Prof. Lourenço Alves Pereira Jr

I. OBJECTIVE

This project aimed to manipulate HTTP requests on the Application Layer using Python. The project have been made for the Second Bimester Trail of CES35. In order to manipulate the HTTP requests, the original Objective was to create a Python Web Framework, It was conducted follwing the Rahmonov's tutorial[2][3][4]. The Python Web Framework was made to support GET, POST, PUT, DELETE requests, and also supported routing.

After the initial development we found out that the core implementation that imteract directly with the HTTP request is the Web Server Gateway Interface (WSGI), therefore the project core migrated to the WSGI developed.

Nlo haverá deploy em produçjo de servidor web, apenas localhost

Cronograma de atividades. 4j semana: planejamento das atividades 5j semana: estudar implementaçjo de exemplo de web framework em Python 6j semana: desenvolvimento da web framework básica, suportando, ao menos, requisiçjes HTTP GET e POST 7j semana: continuaçjo do desenvolvimento e finalizaçjo do desenvolvimento 1j semana de exames: desenvolvimento de material para apresentaçjo final, incluindo simulaçjo cliente/servidor em uma aplicaçjo de exemplo que utiliza o web framework desenvolvido

Referjncia: http://rahmonov.me/posts/write-python-framework-part-one/ http://rahmonov.me/posts/write-python-framework-part-two/ http://rahmonov.me/posts/write-python-framework-part-three/

A* searches techniques. To do that, it was implemented the *Task 2.1* and the *Task 2.2* in Python. The code can be seen in the attached files.

---

[1]Computer Engineering Bachelor Student of ITA

II. PROJECT DEVELOPMENT

What was developed is a Web Framework written in Python running with WSGI also developed by us. The Web Server Gateway Interface (WSGI) is an stadart interface to connect python web frameworks to a server connection socket and interchange data between the socket and the server framework.

The main idea is that the WSGI creates the HTTP server and provides it an HTTP handler. The HTTP server basicly initiate a socket on that port and start to listen to HTTP request. Once it gets a request it passes it to the HTTP handler. The HTTP handler implements the package treatment according to the HTTP protocol, it sets up and read the message headers and handle errors. After that the handler passes all that treated headers and the package to server framework itself, from that point it is the server web framework responsability what will be done regarding that message. After the framework execute what it is to execute to that message, the framework call the WSGI back with the response message and the WSGI delivers that message back to the client.

This interaction between client, WSGI server and web framework application can be seen on the Figure 1, it represents a sequence diagram of that interaction.

The hole intereaction between the classes can be seen on the Figure 2, it shows the project classe, interactions, and hierarch.

III. PROJECT VALIDATION

In order to make every thing work, both the Python Web framework server and the WSGI were successfuly developed and they can be accessed by the *Github Repository*[2].

The usage process can be followed by the repository *Readme* file.

The Web framework also supports templates files. It is

IV. CONCLUSION

A. *Discussion*

omprehension and exemplification. Its difficulty was right-minded, although the amount

---

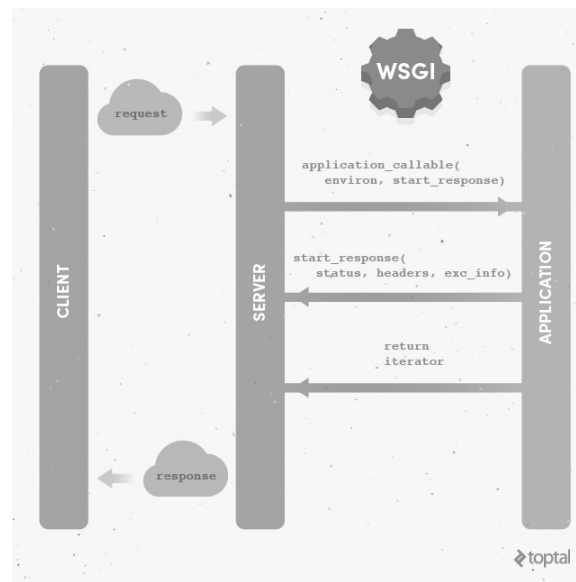[2]https://github.com/CarlosMatheus/Python-Web-Framework

Fig. 1. WSGI Sequence diagram. The Client send a message that when it arrives on the host computer it goes to the WSGI HTTP server socket. The WSGI process the message and passes it to the Web Framework application. The web framework processes the message and send the response message to the WSGI server. The WSGI sends the message back to the client though its HTTP socket.
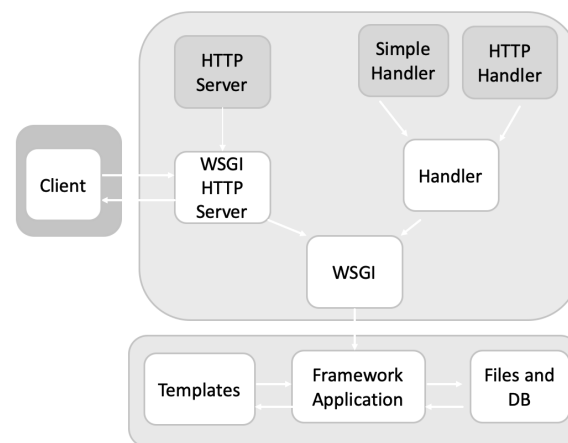

Fig. 2. Project Class architecture. The Classes HTTP Server, Simple Handler, HTTP Handler are from standard Python libraries and they perform basic HTTP protocol operations, therefore they were not implemented for the scope of this project and some classes used them or inherited from them.

B. Conclusion

The work was quite enlightening when it comes to *Greedy* and *A** search methods comprehension and exemplification. Its difficulty was right-minded, although the amount of code required made the work duller than complex.

You can cite an online resource [1].

REFERENCES

[1]   Rebecca Ford. *Earthquake: Twitter Users Learned of Tremors Seconds Before Feeling Them*. Aug. 2011. URL: http://www.hollywoodreporter.com/news/earthquake-twitter-users-learned-tremors-226481.

[2]   Jahongir Rahmonov. *How to write a Python web framework. Part I*. Feb. 2019. URL: https://rahmonov.me/posts/write-python-framework-part-one/.

[3]   Jahongir Rahmonov. *How to write a Python web framework. Part II*. Feb. 2019. URL: https://rahmonov.me/posts/write-python-framework-part-two/.

[4]   Jahongir Rahmonov. *How to write a Python web framework. Part III*. Mar. 2019. URL: https://rahmonov.me/posts/write-python-framework-part-three/.