First week started with a company's presentation and a meeting with my project coordinator.

The project was introduced as well and it will be a chatbot creation that gives us the weather (only for Lisbon, in a beginning state, but it will give the weather for a current time, for today, yesterday and for tomorrow, using machine learning techniques to create weather forecasts). Lately were presented some tools that could be used to make the

project. On the second and third day, besides the received knowledge about the Agile Method, CI/CD and sprints, the search about IBM Watson's documentation started. The project's first draw was created

too.

On the first day of second week the draw was finished and showed in the meeting. On the next two days, by using the work breakdown structure, the arquitecture was improved. For an even better arquitecture, there where searched new tools. The choosed tools were:

\* IBM Watson Assitant that will be used for user - bot's answers interaction

\* IBM Watson Machine Learning that will predict the weather

\* MongoDB that will save the weather informations in a BSON (binary JSON) file

\* GSON that will be used to translate java objects to JSON and vice versa

\* JUNIT that will be used to test all the developed operations

Class Applet will be used to allow the Java's application on Web. Their has no tool to get the weather information. For a Web GUI creation, it will be necessary Web languages. Because of that, it will be

used HTML, CSS and Javascript. The prove of concept was started, in other words, it will be created funcional examples to learn how to use each tool choosed previously. According with the planning, the PoC started earlier.

On the third week, the PoC was continued. For Watson, the example created tests for dialog nodes, intents, examples and values manipulation. For MongoDB was tested objects creation, the translation of that objects to JSON files (using GSON) and their read/write/delete in the database. JUNIT was used for test all created examples. By creating examples to test the weather tools, the chosen one was APIXU, because the simplicity use for requests creation and has all the information needed in a compact way. IBM Watson Machine Learning's example creation was not possible because of country restrictions. So, to solve this unexpected problem, was found a Machine Learning framework (encog) that is very complex and has everything, more than the needs.

On this week's last day was started the brain's structure and the services implementation which are the Watson's services. This step started earier too.

On the forth week second day, tests to the Machine Learning tool were done. The test was the prediction of the sunspots and for that it was used a file from Nasa. On the last day was created Database manipulation's structure and implementation that uses MongoDB services. Database implementation was started earlier because, after the project's final structure, it is necessary to save all the informations and not only the machine learning's "predictions".

On the first day of the fifth week, the application development was continued and it was made some improvements to IBM Watson's workspace.

The project until that day (sprint), the arquitecture, and some possible improvements were presented in the review meeting. Everything made until that day and the Watson's Machine Learning problem (country restrictons) was discussed and the need to get another tool. The data to train the model was missing and because of that it was impossible to create a program that process this data, so the planning's next step was the GUI creation.

The sixth and seventh week were exclusively dedicated to the application's web page.

In the beggining of the sixth week was found a problem on make the java application available to the web, because in the preparation was thought to use the class extension Applet but this class is deprecated since Java 7. To fix this problem, a search was made to find another way to put the application online. The first tool found was Apache Tomcat. Some tests and some tries were made to create a server using this tool. A simple example was used that worked but, when it was adapted to the project, which has many class imports and needs a XML file configuration, it turns the tool into

something with complex configuration and for the needs, it's to complex.

The only thing needed is a HTTP server so the jersey API was imported and then was crated a simple REST server.

This server has only two operations, one GET that returns the initial message from the chatbot and a POST that the user "post" the input and waits for the response.

The HTTP requests are made by a Javascript file that create the requests to the REST server and then sends the received responses to an HTML page, where the conversation between the user and the chatbot will occur. At this moment, a CSS file is used to customize the Web page.

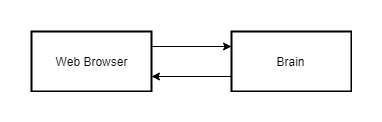
During this weeks, it was presented a new Machine Learning tool, Microsoft Azure Machine Learning. Until now it is very intuitive tool, but were saw demos only, which means that the tool isn't mastered.

Chat bot

What is it?

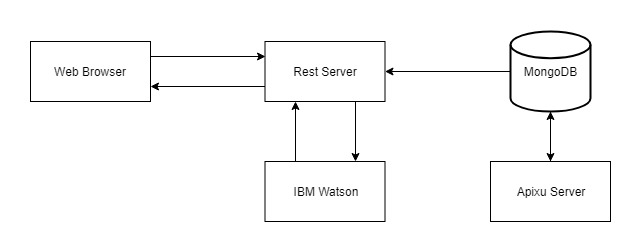
A simple weather chat bot that give the Lisbon weather to a user.

Architecture



The basic architecture is composed by two components:

* Web browser: This component uses a HTML (web page) and a JavaScript file. The HTML is a simple GUI and the JavaScript is used to manage the HTML’s elements functionalities.
* Brain: Components is represented by four sub components (explained above).



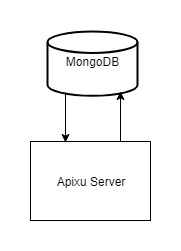
The brain components are:

* Rest Server: This component is a Java Rest Server that can be called as the Brain of the operation. Besides being a Rest server and handle the http requests, this component’s role is insert the updated database information, into the chat-bot “knowledge”, to create the most accurate response.
* IBM Watson: The server used API. The requests are made by HTTP and everything can be changed online (service workspace) or by consuming the API. This API was chosen because of his powerful computing, because it is very easy to use it and because it is almost full customizable by using the API. The chat bot is represented by dialog nodes that recognize some specific expressions.
* MongoDB: This component represents the weather information database. All the weather data from the requests to the weather API are saved into this database. It is saved the current, yesterday and today until the current weather information.
* Apixu Server: The server that has the weather information. This server’s API was chosen because it is free and has a simple response.

Workflow

There are three workflows:

The first one is the weather information update.



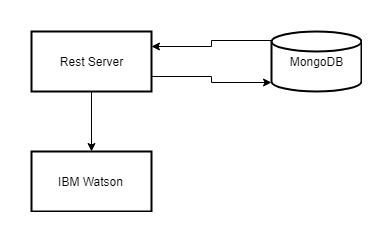
For the information update, the steps are:

1. There is initialized threads to create http requests to the weather server. It will be updated the current time, yesterday and today until the current time weather
2. The response comes in JSON format. When we got it, we transform into an Object using GSON and then it will be saved into the MongoDB database

Only the database update is executed by threads. The chatbot’s dialog update is manual.

The second and thirst are made by the same components, which are:

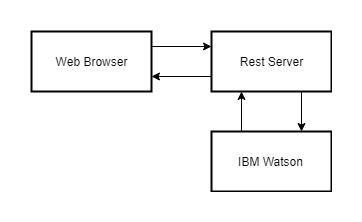
Update the chatbot dialog nodes: When we prefer, we manually update the dialog nodes with the current database weather information.



The steps of this workflow are:

1. Access the MongoDB database and get the information that we are updating (the update is separated by the current node, yesterday and today nodes)
2. It is used the IBM Watson API to get the dialog node that we want to change
3. We insert the new information and send it to the server with this information update

Handle the user HTTP requests:



When the user wants to know the weather, what happen is:

1. The user inserts a weather message into the GUI
2. The message is inserted into the web page as a chat bubble (DOM element)
3. This message is sent to the Rest Server by using JavaScript XMLHttpRequest function. This function will POST the user message into the HTTP request body
4. The Rest Server receives the request and extract the message from it
5. The Server creates an HTTP request to the chatbot server (IBM Watson), inserting the message into the request body
6. The response comes in JSON format and it is parsed to a Java Object (using GSON). The JSON file has all the possible responses order by answer confidence
7. To send as a response to the web page, the most confident answer is sent as the Rest Server response to the JavaScript request
8. The JavaScript receives this message and creates a chat bubble (DOM element) with the response received

Documentation

All the developed code is commented, so for a good explanation about all the implemented functions, read the code’s comments.

What is missing?

There are some improvements that can be made to this project:

1. Change the IBM Watson’s API usage from Java to JavaScript. By doing this improvement, the Java Rest server is not more necessary. (Now, the Rest Server is working as a proxy between the web browser and the Watson’s service).
2. Transform the dialog nodes manual update into automatic.
3. Add forecast by using Machine Learning prediction algorithms
4. Add more cities and countries to the chatbot’s knowledge

References:

Java Dependencies Documentation

https://www.ibm.com/watson/developercloud/conversation/api/v1/java.html?java

http://mongodb.github.io/mongo-java-driver/3.7/driver/getting-started/quick-start/

https://github.com/google/gson/blob/master/UserGuide.md

https://www.apixu.com/api-explorer.aspx

<https://jersey.github.io/documentation/latest/index.html>