Πρακτική εξάσκηση

INTERNSHIP REPORT

JAVA WEB SERVICE

Nicolas Pereira

Academic Year 2018-2019

SOFAR

<u>Supervisor</u> : **Dr Dimitris Halvatzaras**

UNIVERSITY INSTITUTE OF TECHNOLOGY
<u>Mentor</u> : **Pr Patrick LEBEGUE**

ΤΕΙ ΙΟΝΙΩΝ ΝΗΣΩΝ

UNIVERSITY INSTITUTE OF TECHNOLOGY
<u>Mentor</u> : **Pr Adamantia KAMPIOTI**

Zakynthos, June 2019

Acknowledgement

First of all, I sincerely would like to thank **Dr. Dimitris Halvatzaras** for his patience, his advice and his help throughout the project. I learnt so many things during the project thanks to him. He gave me the chance to discover this beautiful island and I'm very grateful to him.

Secondly, I greatly thank **SOFAR's team** for their friendliness and their welcome, it was a real pleasure to work with them into the company.

Also, I would like to thank the **TEI* of Zakynthos**, for their welcome and **Dimitris** and **Spiros** for finding us an affordable price and well-placed accommodation.

Last but not least, I am very grateful to **Pr. Patrick Lebègue**, my tutor from the IUT of Lille who followed my internship, as well as the **International Relationship Office** of the IUT who made this internship possible.

Abstract

I chose to do my internship here in Zakynthos first of all to improve my English level by speaking to students, shopkeepers and of course with the members of SOFAR. I had been thinking that the Greece should be a great place to learn English and it was the case! People here are very friendly so I could speak with students and even with elderly persons of the island even if they do not speak English very well.

Secondly, I wanted to come here because I have never been to Greece before. Indeed, I really like discovering new cultures, landscapes and people from different countries. I chose the island of Zakynthos especially for her uncommon landscapes full of mountains and wonderful beaches.

My project topic was about a Java web service which had to get two different list, a bus list and route list. Next, the service had to assign the buses to routes to find the best solution and send back the result.
The best solution was either to pay the smallest cost and use the smallest number of buses or cover every route as fast as possible but you need to make your mind up.
The conditions to find the solution are in a specific file write in Drools. When the problem starts to be solved, the program calculates a score which represents the cost of the solution. For example, if you have to pay 3000€ the score will be -3000, if you use 30 buses the score will be -30. With this score you can follow the evolution of the solving and you can stop it whenever you want. The bus list is a JSON file with different properties like the cost, the capacity or the company of the bus. Same thing for the route list with properties like the demand, the start time or the different bus stops. To solve the problem, I used a library called Optaplanner. I had to adapt my code to be compatible with it.

My internship was a wonderful experience, maybe the best and the most instructive in my life! Professionally this internship brought me new skills like code in drool, server configuration and other. But I also learnt to communicate with an English team and I improved my skills in Java and REST.

Personally, this journey made me grow, I visited a lot of things on this Island and in Greece. I spoke with a lot of person from different countries like Spanish, Albanian, Portuguese, German, Canadian, Dutch and of course Greek! I have discovered new cultures, new ways of thinking and I will never forget it.

Table of contents

Introduction

This internship is the last step of my two years degree at the IUT A in the French University of Lille. After that my degree in computer science will be validate.
I chose to do it abroad because I thought it was a unique opportunity to improve my English level and discover a new part of the world. I could come to the island of Zakynthos in Greece thanks to the Erasmus program and I have been working in a little company named SOFAR.

During my internship, I worked on a Java web service which permitted to answer the following question *"How many buses do I need to cover each bus routes of Zakynthos?"*.

As you can see in the table of contents (page 4), this report is divided in 3 parts. It starts with a description of the place where I did my Erasmus with: Greece, the island of Zakynthos, SOFAR and the TEI. In the second part we will speak about what does my service do precisely. At the end we will have a look on which technologies did I use of this project.

1. Work environment

1.1 Main places

1.1.1 Greece

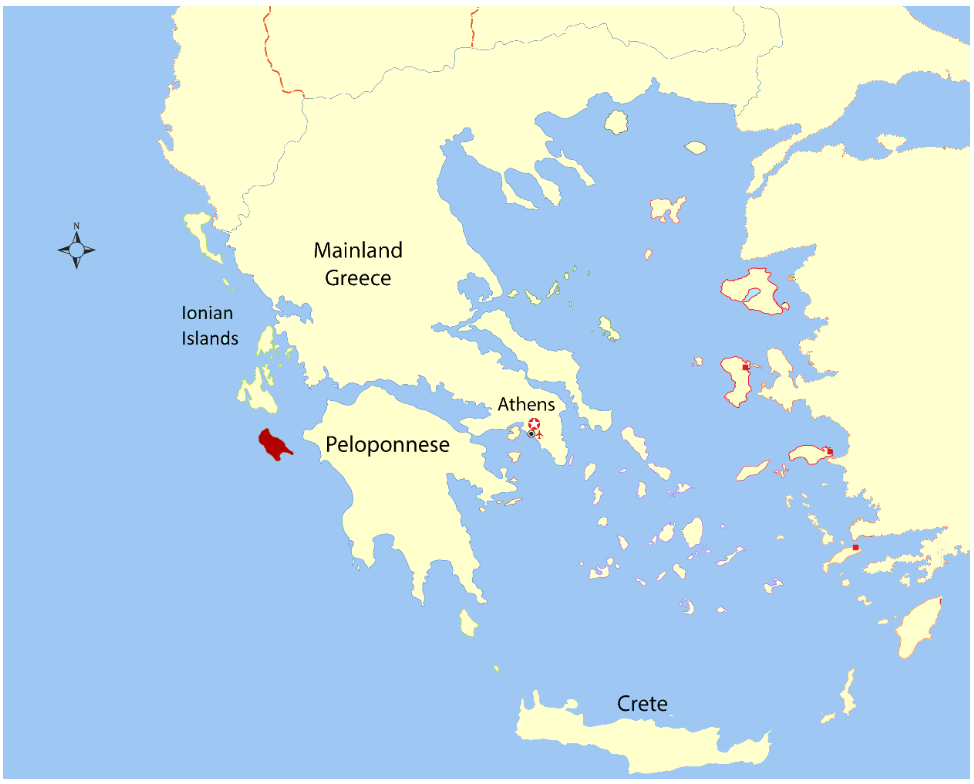
Greece is a country located in the south of Europe and populated by over ten million people for an area of 132 049 km². Greece is member of the European Union since 1981 and the country has for capital Athens (the largest city of the entire country and populated by over three million people). Greece is composed by three parts: Greece continental, the peninsula of Peloponnese, and the islands. Eighty percent of Greece is mountainous and the highest peak is the famous Mount Olympus with 2 917m of altitude.

Greece's location in Europe (red)

Greece has a rich historical past, especially thanks to the ultra-famous books the "Iliade" and the "Odyssey" which describes stories of the antique Greek heroes. One of the best-known places is Olympia, where took place the ancient Olympic games. Also, this country is considered as the cradle of democracy and its great culture still influences many countries around the world. There are more than 6 000 islands around the country and only 200 are inhabited. However, only around 20 islands are very famous and touristic and Zakynthos is one of them.

1.1.2 Zakynthos

Zakynthos is a Greek island, it is one of the 7 Ionian Islands. It is the eleventh biggest island of the Greece with 410 km² populated by around forty-six thousand people. It's a tourist destination with an international airport served by many charter flights and a ferry which join the continent and port of other islands. Zakynthos or Zante is particularly known for one of the most beautiful beaches in the world: "Shipwreck beach". It's the second most visited island after Corfu. During my internship, I was in Zakynthos town, the main city of the island.



Zakynthos's location in Greece (red)

Shipwreck beach

On 12th August 1953, an earthquake of 7.2 intensity on the surface wave magnitude scale hit Zakynthos. It caused widespread damage throughout Kefalonia's island and Zakynthos. In Zakynthos town, all the buildings were destroyed and had to be rebuilt except two.

1.1.2 SOFAR

SOFAR is a private Greek company founded in 2006 by Dr. Dimitris Halvatzaras with a team of talented I.T. experts who love building innovative software applications on the island of Zakynthos. Dr. Halvatzaras studied and worked for several years in different research projects in the field of Human Computer Interaction and Artificial Intelligence. Mainly, the company works on web and mobile applications, in order to increase the tourism on the island like EcoZante, a big project developed by them.

SOFAR's office

SOFAR's team is composed by 4 people. Dimitris is software engineer, he manages projects and clients. Spiros is the web developer, he is the HTML/CSS expert from his long experience on web development and web interface implementation. Constantinos is the web developer as Spiros, he is the HTML/CSS expert specialized in artistic direction. And the last but not least, Georges. He is software developer and even if he is the youngest of the team, he is a very talented programmer. Georges is mainly involved in the development of web based and mobile applications.

1.1.2 TEI

The TEI of Ionian Islands is divided in six departments and in four schools, one in Kefalonia, another in Lefkada, one more in Lixouri and the last is in Zakynthos. The department of Zakynthos is Ecology and EnvironmentTECH but as I did my internship in SOFAR that did not pose any problem. The number of university students in Zakynthos is approximately 400. The department offers two places to the IUT A, so we were two students here in Erasmus.

TEI of Zakynthos

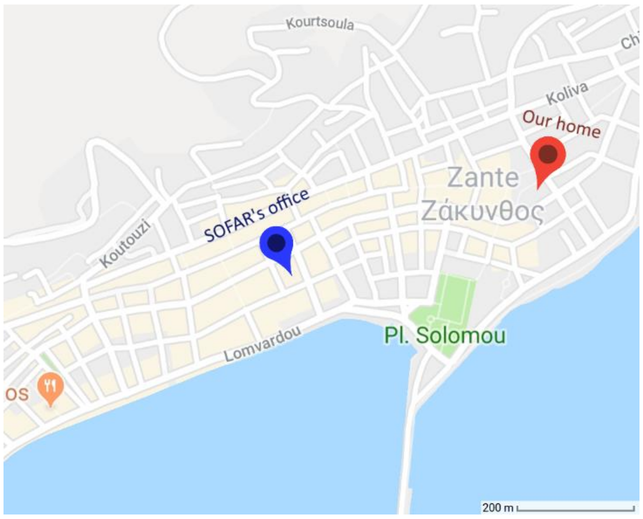
We used to go the TEI to eat, there I met many students who explained me how their life on the island was. They taught me some word in Greek and helped me to improve my English. Even if we have a different culture we seem very similar.

1.2 The beginning

1.2.1 Accommodation

When we arrived in Zakynthos the first problem that we had to overcome was to find an Accommodation. Indeed, the last year house was busy, so we took a week in an AirBnb. I searched a lot on internet but due to the tourist activity of the Island it was impossible for me to find another one. But Dimitris saved us by finding an affordable price and well-placed accommodation. In this way installed us at 7 Arh. Dimitriou street.

We were not too far from SOFAR, maybe 10 minutes on foot. The apartment was really big with two bedrooms, a bathroom and a kitchen. We also had a big balcony and a big roof top. The only problem was the sound insulation, the night I was able to follow conversations of people outside of the house from my bed.



Map of Zakynthos town

1.2.2 First Steps

At the beginning, I was apprehensive about the internship because it was the first time that I worked on a professional project in a company. It was really hard for me to explain what I thought, and which were my difficulties. In other hand, at the beginning I did not understand on what I was working. So, it was hard for me to be motivated by the project, but Dimitris was very comprehensive, and he never put pressure on me. He was always attentive even if he has himself a lot of work. Indeed I will remember forever the ringing of the phone, because it was ringing all the time.

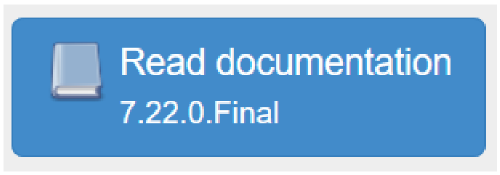
My work condition was very nice, I had a comfortable office and a wonderful working atmosphere despite the warm. Schedule was generous and flexible, I started at around 10am and finished between 5pm and 6pm.

2. My work

2.1 Discovering

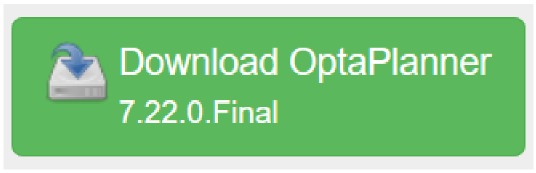
2.1.1 Optaplanner

The first thing that I did when I stared to work on the project was to gather a lot of information on my subject. That is why I began to read the Optaplanner documentation. Also, I looked so many videos on internet in order to understand what was Optaplanner and how does it work. (if you want more information about it you can read the third part called "Technologies Used").



Button to read the Optaplanner documentation

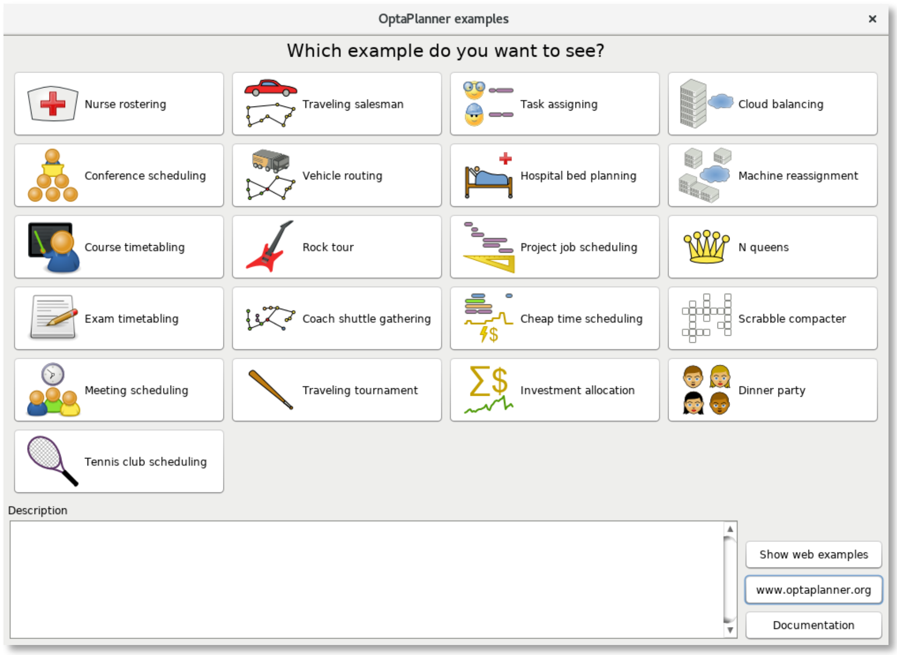
After that I downloaded Optaplanner on my computer and I imported it on my IDE* Eclipse. Optaplanner is divided into two parts themselves divided into subparts. There are the Interfaces, with a java application writen with Swing and a web application that you can put in a web server. These two types of interface use the same back end to model and solve the problem.



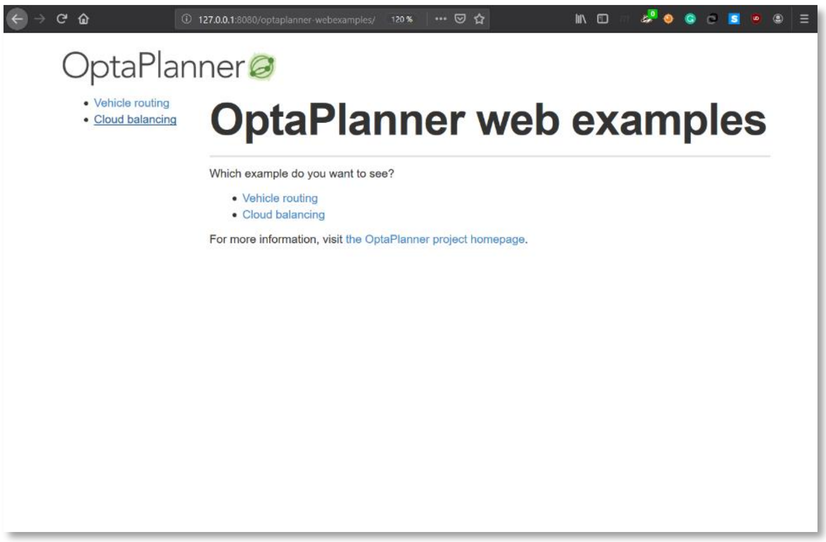
Button to download Optaplanner

2.1.2 Tomcat or Wildfly

As I was able to run the java application, I wanted to test the web application. This one was in WAR format, so I had to put it in a web server. Firstly, I tried with Apache Tomcat 9 web server, but I had some errors with different library. Quickly I changed Tomcat for another web server: WildFly.



Optaplanner java application



OptaPlanner web examples

Which example do you want to see?

- Vehicle routing
- Cloud balancing

For more information, visit the OptaPlanner project homepage.

Optaplanner web application

I spent some time to understand how WildFly works, by reading documentation and looking videos on internet. However with Wildfly I was able to load the WAR file without problem and tried the different functionalities of the web application. During a long time, I used WildFly even if it was slower than Tomcat. But one day I met an accessibility problem and I returned on tomcat.

2.1.3 Maven

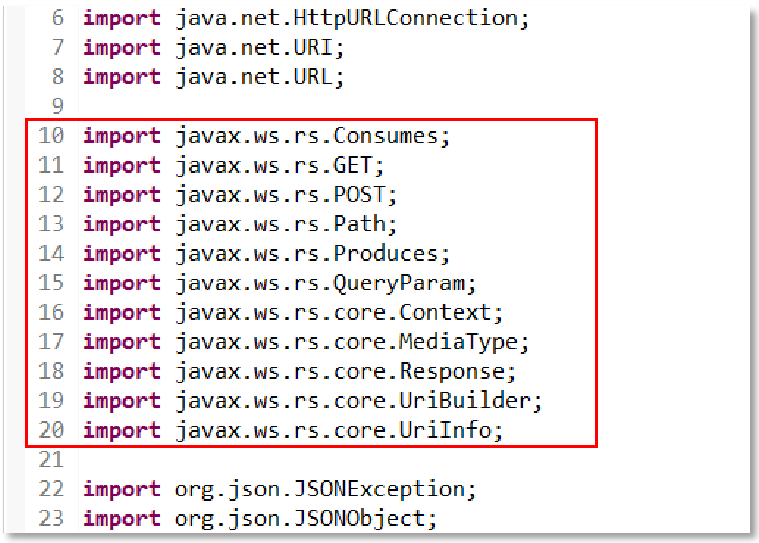
Maven is a build automation tool used primarily for Java projects. Maven addresses two aspects of building software: first, it describes how software is built, and second, it describes its dependencies.

During the project I tried to use Maven because I already did it in my University projects. However, because of compatibility problems with the different version the library used inside Optaplannner and with my own library, I abandoned this one.

2.2 First server REST

2.2.1 Choose the RESTful API

Once the server was chosen it remains me to choose the good library to make it a REST server. At the University we used to use a library called Jersey, that is why I would like to use this one too. The problem was that at the university we normally use Maven. So, I searched on internet how could I do without maven.



```java
import java.net.HttpURLConnection;
import java.net.URI;
import java.net.URL;

import javax.ws.rs.Consumes;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.Context;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.UriBuilder;
import javax.ws.rs.core.UriInfo;

import org.json.JSONException;
import org.json.JSONObject;
```

Screenshot of imports in my java code

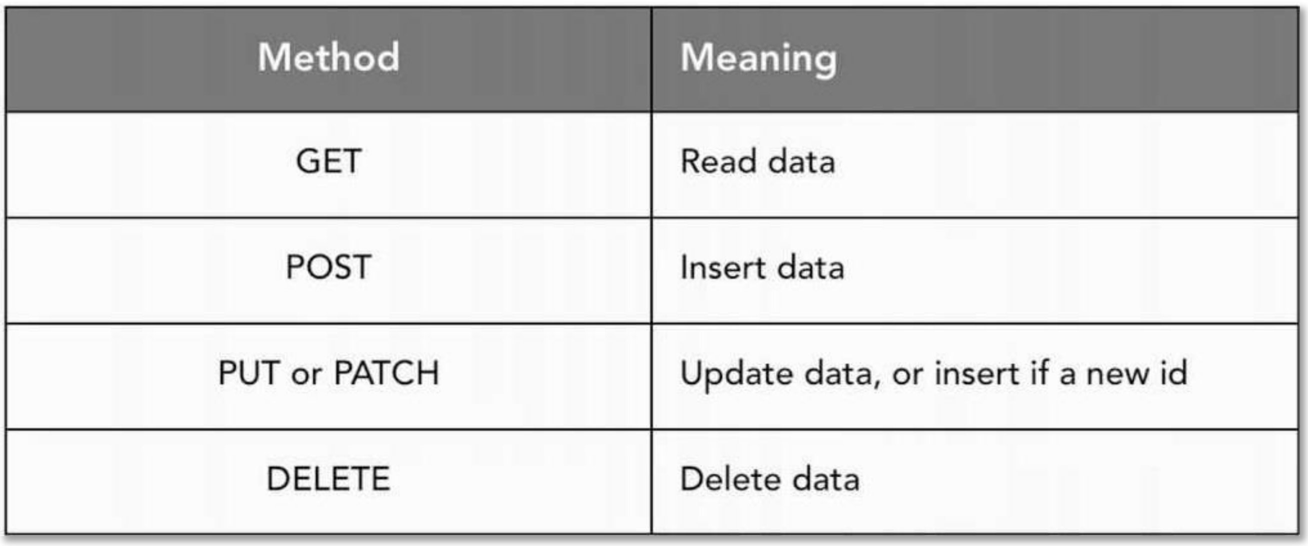
Button to download Jersey

2.2.2 Implementation of Optaplanner

From this point I had solid foundation, a functional web server with a RESTful interface. So, I created a new project, set up the server, my service with Jersey and I began to import Optaplanner, only the parts which interested me to my web service. It was really hard to do it because each class of Optaplanner was dependent. So, I read the code of the classes of Optaplanner and I tried to solve my multiple problems. At the end it remained a quarter of the original code.

2.2.3 GET/POST/PUT/DELETE

My code has changed a lot from the beginning. Some parts are now useless but during the project I created java methods for each HTTP methods. One for the GET, to get the problem during his solving. Another for the POST, it permits to send a problem to solve to my web server. One for the PUT to change something in the problem and to finish one for the DELETE to remove the problem.



Method	Meaning
GET	Read data
POST	Insert data
PUT or PATCH	Update data, or insert if a new id
DELETE	Delete data

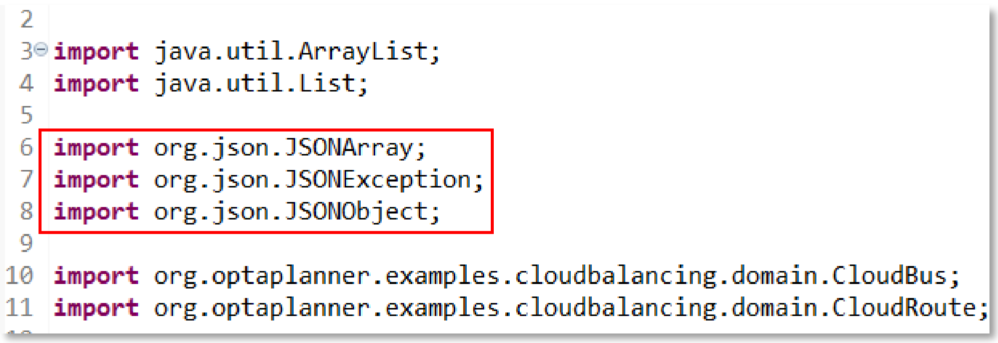
All HTTP methods

Currently, only remains the GET method split in different java methods for example to get the scores, get the best solution or get the final result. Even the POST disappears, because when you start the solving you need to get back a response in order to know if everything is good for the server. Also, the web server gets all the information from a data base, so it does not need to do a POST or a PUT.

2.3 From computers to buses

2.3.1 Use the JSON

After the creation of my service with the parts of Optaplanner that I had chosen and the methods to access to the problem. It was possible to run the solving of the Optaplanner problem example (with process and computers). Nevertheless, one of the aims of the project was to get the problem from a data base, by doing a GET on a specific URL and then get a JSON. Consequently, I needed a library to read JSON object which is "Java-JSON.jar".



```java

import java.util.ArrayList;
import java.util.List;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import org.optaplanner.examples.cloudbalancing.domain.CloudBus;
import org.optaplanner.examples.cloudbalancing.domain.CloudRoute;
```

Screenshot of imports in my java code

So, from this moment we did not use the Optaplanner example anymore, but we did a GET request on Kriton website which contained the bus lists and the route lists. Kriton responded back with JSON, my service read the JSON and transforms that in a Java object usable by Optaplanner.

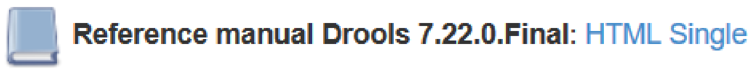
2.3.2 Huge refactoring

From now on it was possible to get a real problem from Kriton, I had to refactor the code of Optaplanner to change the computer class in bus class, the process class in route class and every other modification that I had to do, to fix dependencies bugs. Also, I changed the properties inside the classes, the memory changed in places inside the bus or I had to create new like the start point and the end point of a route. This part was really long and create a lot of problems.

2.4 Hello Drools world

2.4.1 Learn Drools

Optaplanner uses a specific file called “ScoreRules.drl” to adapt his solutioning. This file contains a set of rules written in a specific language: Drools. Drools is a business rule management system with a forward and backward chaining inference-based rules engine, more correctly known as a production rule system, using an enhanced implementation of the Rete algorithm.



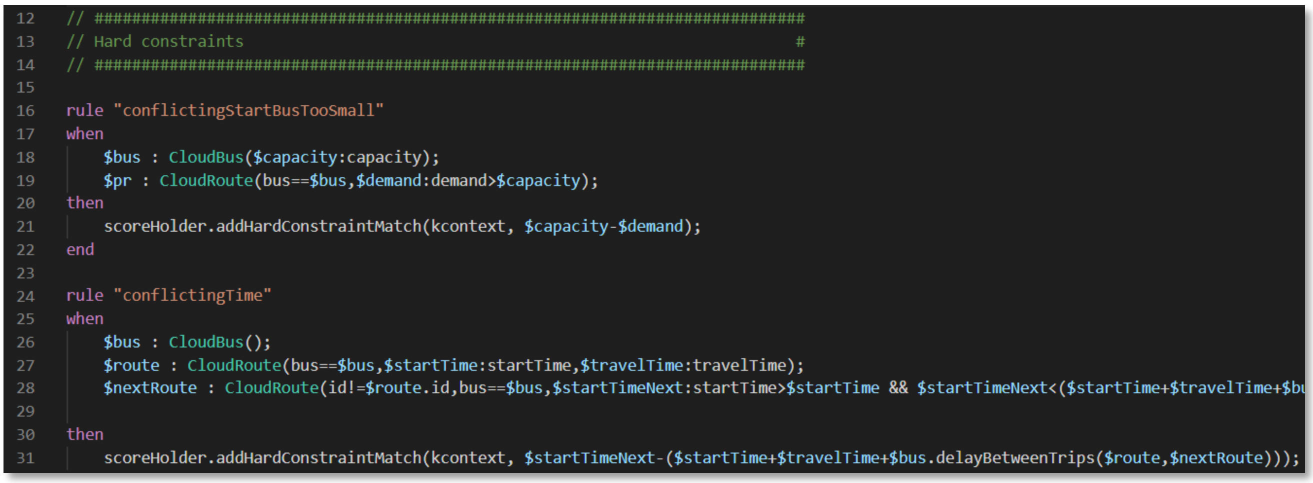
Reference manual Drools 7.22.0.Final: HTML Single

Button to read the Drool documentation

I had never practiced this language before, so I read a lot of tutorials on different forums and I tried to understand what was written in this file. Fortunately, Dimitris gave me an example easier to help me.

2.4.1 Change the DRL file

As you can see on the following screenshot, the DRL file was divided in two parts. The first one is for the Hard constraints, constraints impossible to break, like to putting 100 people in a bus of 25 people or start de new route before the end of the current route.

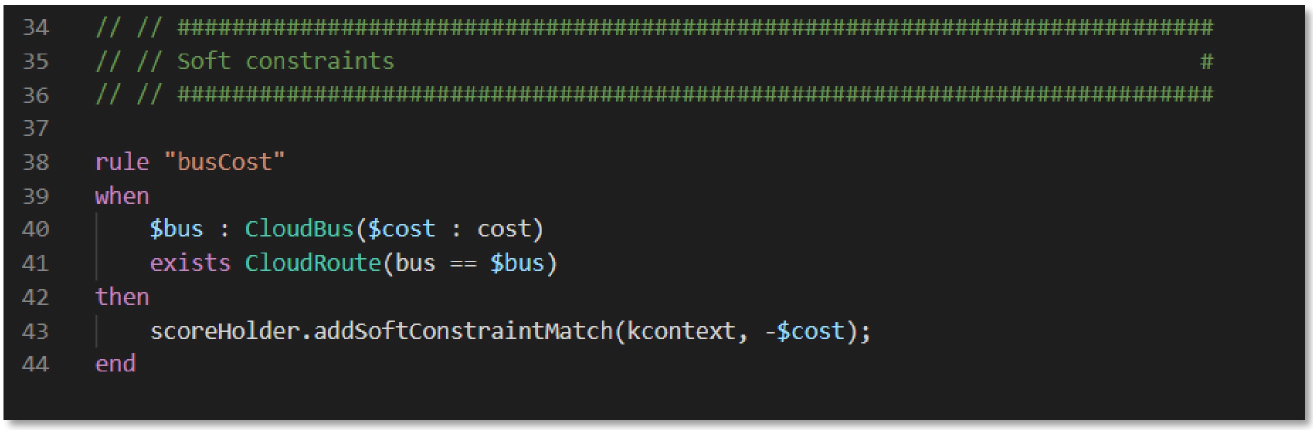


```
// ############################################################################
// Hard constraints                                                           #
// ############################################################################

rule "conflictingStartBusTooSmall"
when
    $bus : CloudBus($capacity:capacity);
    $pr : CloudRoute(bus==$bus,$demand:demand>$capacity);
then
    scoreHolder.addHardConstraintMatch(kcontext, $capacity-$demand);
end

rule "conflictingTime"
when
    $bus : CloudBus();
    $route : CloudRoute(bus==$bus,$startTime:startTime,$travelTime:travelTime);
    $nextRoute : CloudRoute(id!=$route.id,bus==$bus,$startTimeNext:startTime>$startTime && $startTimeNext<($startTime+$travelTime+$bu

then
    scoreHolder.addHardConstraintMatch(kcontext, $startTimeNext-($startTime+$travelTime+$bus.delayBetweenTrips($route,$nextRoute)));
```

Screenshot of the DRL file

When the algorithm tests a solution, it attributes a Hard Score via the constraints of this DRL file. For each Hard constraint broken, the Hard Score increases. A problem with a Hard Score different of zero is an insolvable problem.

The second part of the DRL file is for the Soft constraints, constraints possible to break like pay 500 € for a bus.



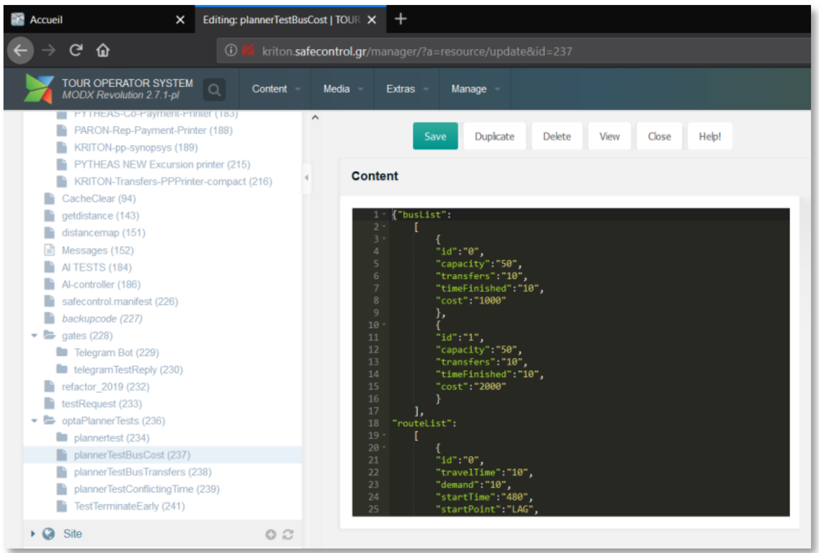
```
// // ############################################################################
// // Soft constraints                                                           #
// // ############################################################################

rule "busCost"
when
    $bus : CloudBus($cost : cost)
    exists CloudRoute(bus == $bus)
then
    scoreHolder.addSoftConstraintMatch(kcontext, -$cost);
end
```

Screenshot of the DRL file

Same thing here, when the algorithm tests a new solution, it attributes a Soft Score via the constraints of this DRL file. It will try to keep the lowest Soft Score possible without breaking a Hard Constraint.

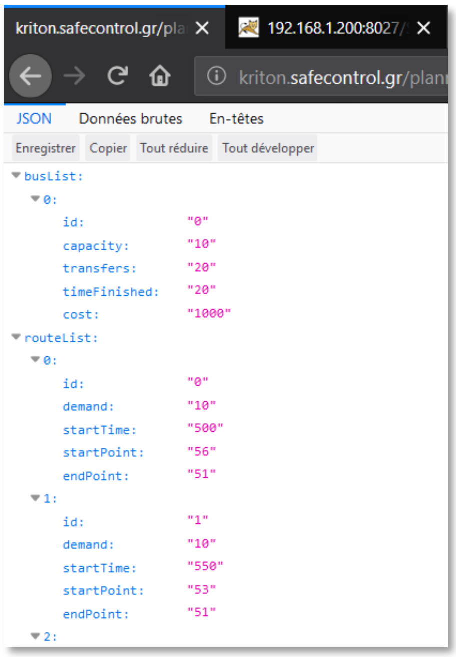
2.4.1 Make some tests

In order to know if the solving was correctly working with my new constraints, I wrote some specifics JSON to test the different cases. One for the cost rule, three for the conflicts timing and others.



```
{"busList":
    [
        {
        "id":"0",
        "capacity":"50",
        "transfers":"10",
        "timeFinished":"10",
        "cost":"1000"
        },
        {
        "id":"1",
        "capacity":"50",
        "transfers":"10",
        "timeFinished":"10",
        "cost":"2000"
        }
    ],
"routeList":
    [
        {
        "id":"0",
        "travelTime":"10",
        "demand":"10",
        "startTime":"480",
        "startPoint":"LAG",
```

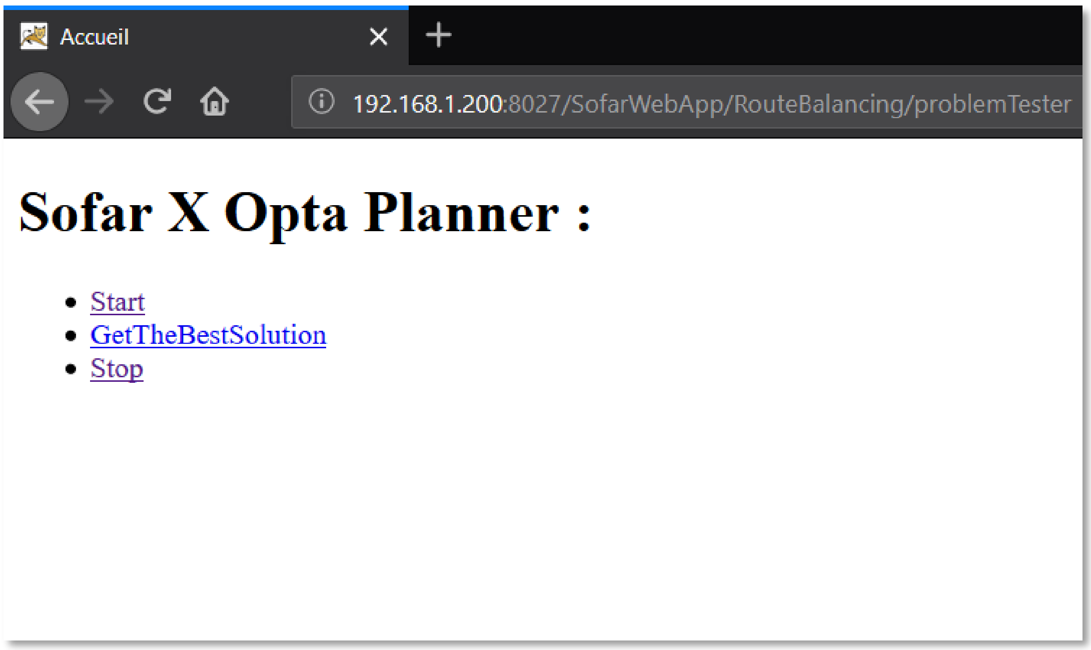
Screenshot of a JSON use to test the cost on the Kriton web site



```
▼ busList:
  ▼ 0:
      id:             "0"
      capacity:       "10"
      transfers:      "20"
      timeFinished:   "20"
      cost:           "1000"
▼ routeList:
  ▼ 0:
      id:             "0"
      demand:         "10"
      startTime:      "500"
      startPoint:     "56"
      endPoint:       "51"
  ▼ 1:
      id:             "1"
      demand:         "10"
      startTime:      "550"
      startPoint:     "53"
      endPoint:       "51"
  ▼ 2:
```

Screenshot of a JSON use to test the conflict timing

Also, I created an interface to run these tests.



Accueil

192.168.1.200:8027/SofarWebApp/RouteBalancing/problemTester

Sofar X Opta Planner :

- Start
- GetTheBestSolution
- Stop

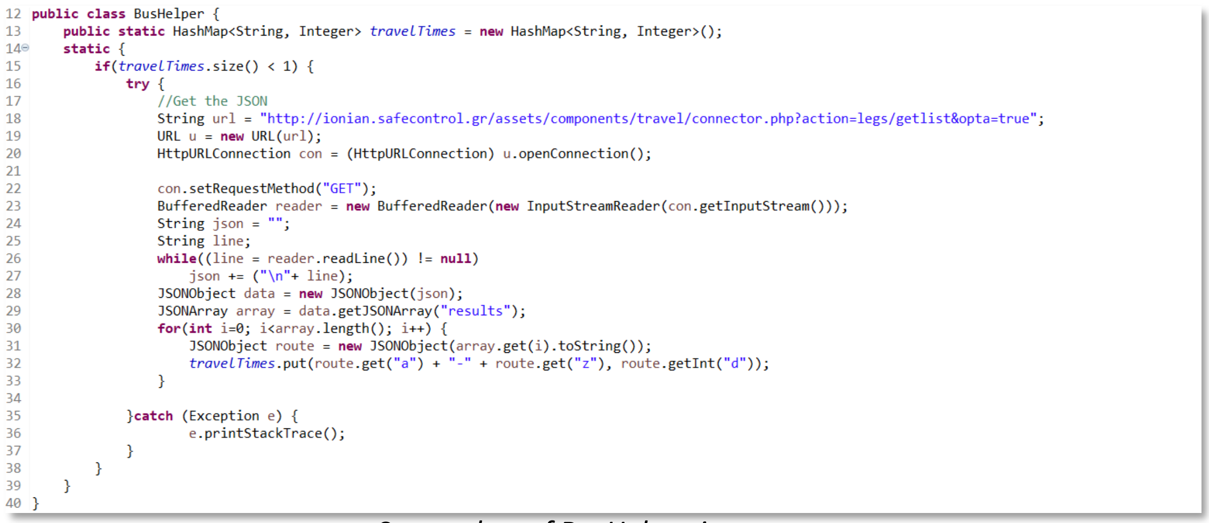
Interface use to run the tests

So, my service did a GET request on the website Kriton on a specific address and it read the JSON which come from this one. It built a java object corresponding to the JSON object and it put it in the Optaplanner solver which solved that with my specific rules.

2.4 More complex problems

2.5.1 Hash map

From now on, solving a real problem with buses and routes was possible, so Dimitris asked me to make some calculus in java. For example, calculate how much time will take a bus to go from point A to point B. This was the goal of my java class "BusHelper".



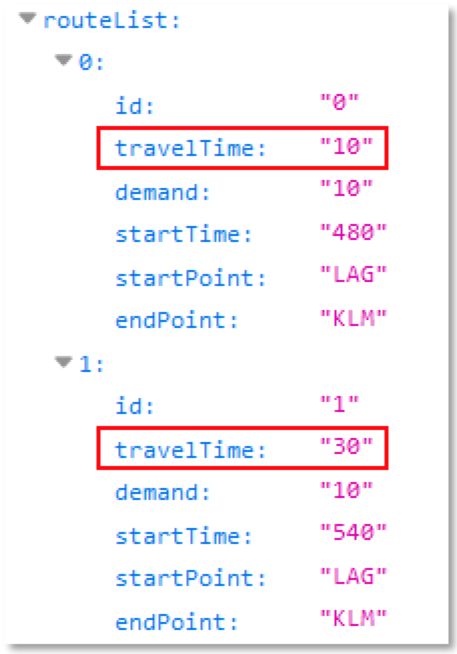
```java
public class BusHelper {
    public static HashMap<String, Integer> travelTimes = new HashMap<String, Integer>();
    static {
        if(travelTimes.size() < 1) {
            try {
                //Get the JSON
                String url = "http://ionian.safecontrol.gr/assets/components/travel/connector.php?action=legs/getlist&opta=true";
                URL u = new URL(url);
                HttpURLConnection con = (HttpURLConnection) u.openConnection();

                con.setRequestMethod("GET");
                BufferedReader reader = new BufferedReader(new InputStreamReader(con.getInputStream()));
                String json = "";
                String line;
                while((line = reader.readLine()) != null)
                    json += ("\n"+ line);
                JSONObject data = new JSONObject(json);
                JSONArray array = data.getJSONArray("results");
                for(int i=0; i<array.length(); i++) {
                    JSONObject route = new JSONObject(array.get(i).toString());
                    travelTimes.put(route.get("a") + "-" + route.get("z"), route.getInt("d"));
                }

            }catch (Exception e) {
                    e.printStackTrace();
            }
        }
    }
}
```

Screenshot of BusHelper.java

The class simply created a static Hash Map and filled it with all the data required. The key was the name of point A and point B and the value was the time to do the travel between these two points in minutes. For example: “key: a-b value: 15”. This Hash Map was used in order to calculate the travel time when we created a new problem with routes. Also, it was used in the DRL file where we needed it to respect the timing constraints.

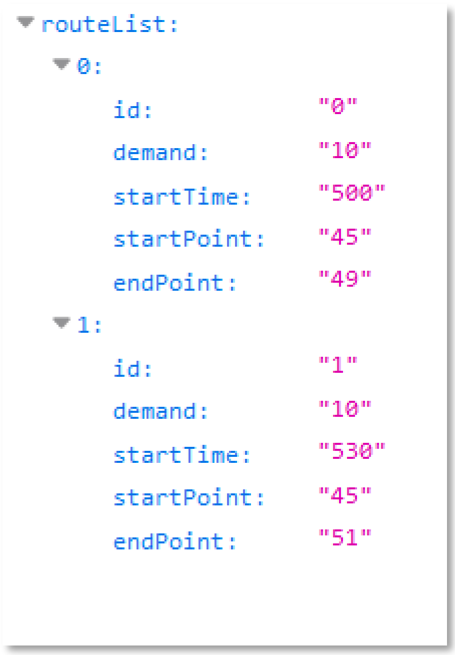
2.5.2 JSON more complex

As the java code changed, the JSON also. After that we did not need the “Travel Time” property anymore in JSON because we were able to calculate it in my java code.



```
▼ routeList:
  ▼ 0:
       id:           "0"
       travelTime:   "10"
       demand:       "10"
       startTime:    "480"
       startPoint:   "LAG"
       endPoint:     "KLM"
  ▼ 1:
       id:           "1"
       travelTime:   "30"
       demand:       "10"
       startTime:    "540"
       startPoint:   "LAG"
       endPoint:     "KLM"
```

Screenshot of an old JSON



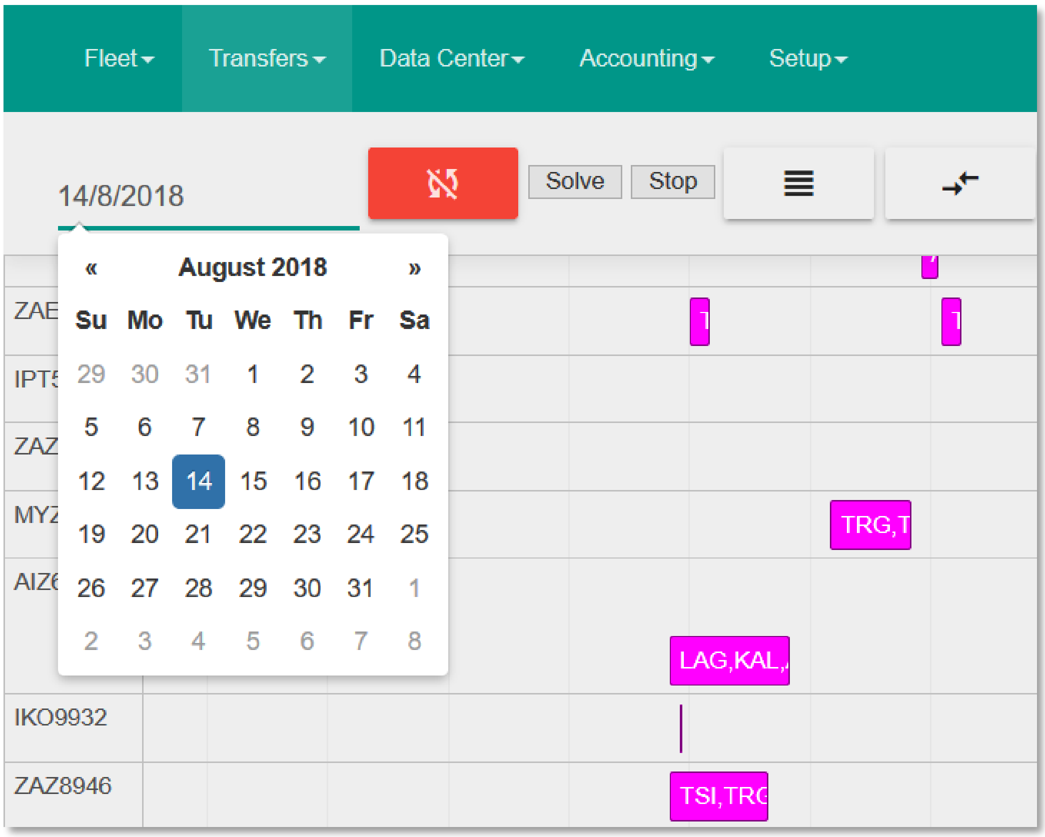
```
▼ routeList:
  ▼ 0:
       id:           "0"
       demand:       "10"
       startTime:    "500"
       startPoint:   "45"
       endPoint:     "49"
  ▼ 1:
       id:           "1"
       demand:       "10"
       startTime:    "530"
       startPoint:   "45"
       endPoint:     "51"
```

Screenshot of a new JSON

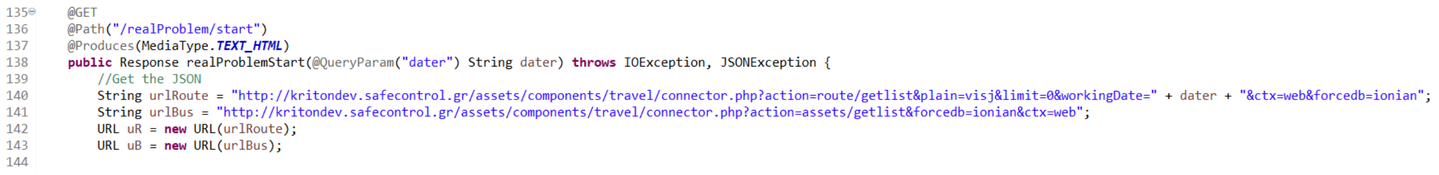
In addition to that, we were able to put more than only 2 bus stops. Of course, we still have the start point and the end point but also, we could have several bus stops between that two point. The java program calculates easily the time travel for all the trip. Finally, there was another modification useful for the rules in the DRL file: we were also able to calculate the time travel between the end of a route and the beginning of another one. It was more realistic.

2.5.3 Choose a date

Once we were certain that everything was working correctly, we stared to solve real problem, with route and buses. The bus list was always the same one, we got it from an URL and we put it in the solver, the route list depends on the date of the day. We used a calendar to choose a date and my service did the GET on the good URL and we got the routes.



Screenshot of the date piker on KritonDev



```
@GET
@Path("/realProblem/start")
@Produces(MediaType.TEXT_HTML)
public Response realProblemStart(@QueryParam("dater") String dater) throws IOException, JSONException {
    //Get the JSON
    String urlRoute = "http://kritondev.safecontrol.gr/assets/components/travel/connector.php?action=route/getlist&plain=visj&limit=0&workingDate=" + dater + "&ctx=web&forcedb=ionian";
    String urlBus = "http://kritondev.safecontrol.gr/assets/components/travel/connector.php?action=assets/getlist&forcedb=ionian&ctx=web";
    URL uR = new URL(urlRoute);
    URL uB = new URL(urlBus);

```

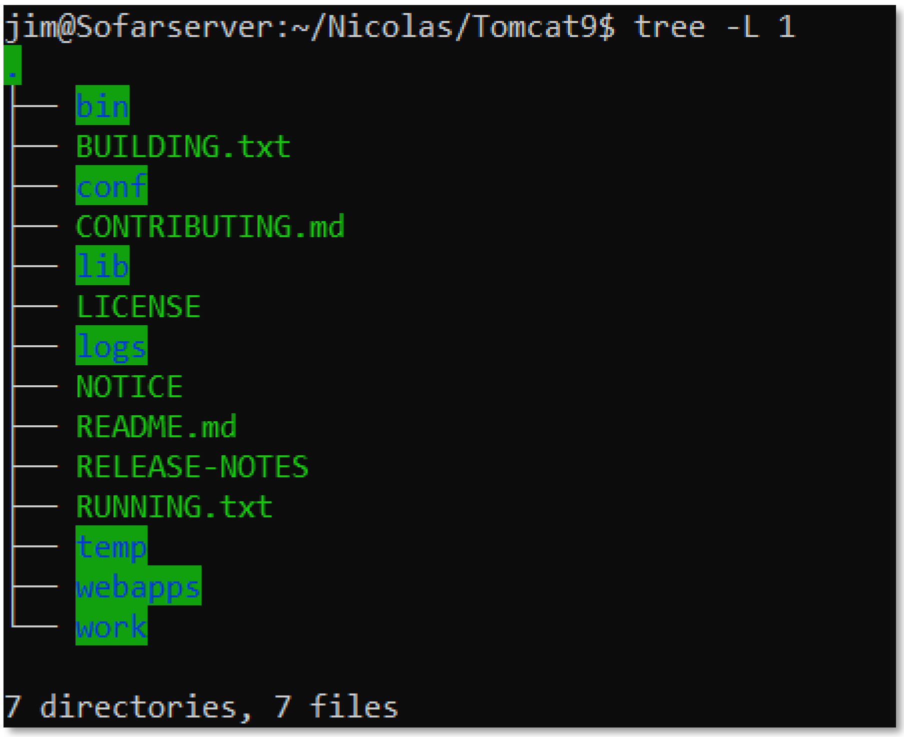
Screenshot of the URL in my java code

Everything was working so Dimitris asked me to create a tomcat server on the SOFAR server and export my web service in a WAR file to load it on this server. To did it, I used Linux commands like SSH or SCP that I had already seen in my University.

2.6 Server problems

2.6.1 Server configuration

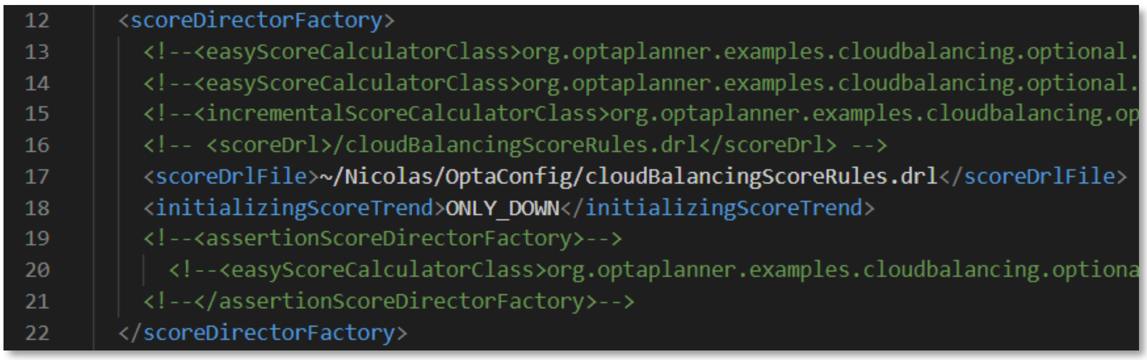
As previously said, my server was an Apache Tomcat 9 for speed reason. Tomcat is lighter than WlidFly and also faster. It permits me to import my project with a WAR file by putting it in the folder: "Tomcat/webapps/". Additionally, Dimitri asked me to extract the DRL file of the WAR file and for a good reason. In fact, when the Tomcat server started, it read the WAR file and loaded it. So, if you want to change something you have to create a new WAR file and this is quite long.



```
jim@Sofarserver:~/Nicolas/Tomcat9$ tree -L 1
.
├── bin
├── BUILDING.txt
├── conf
├── CONTRIBUTING.md
├── lib
├── LICENSE
├── logs
├── NOTICE
├── README.md
├── RELEASE-NOTES
├── RUNNING.txt
├── temp
├── webapps
└── work

7 directories, 7 files
```

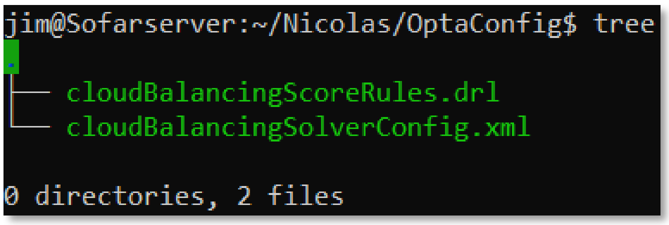
Structure of my Tomcat Server

In order to avoid to recreating WAR file all the time I changed the Optaplanner configuration to load the DRL file from a specific folder called "OptaConfig".



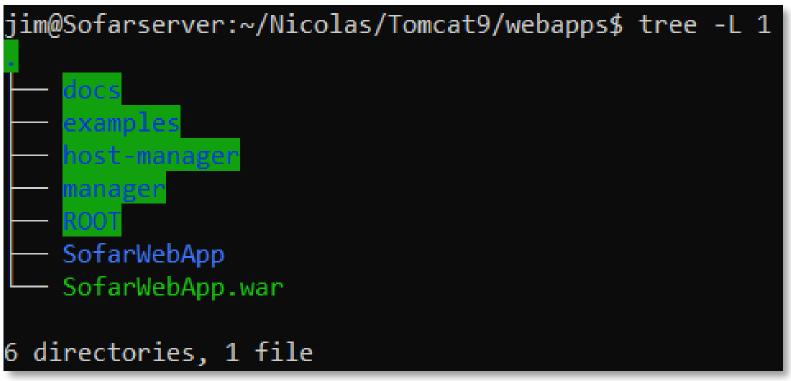
```
<scoreDirectorFactory>
  <!--<easyScoreCalculatorClass>org.optaplanner.examples.cloudbalancing.optional.
  <!--<easyScoreCalculatorClass>org.optaplanner.examples.cloudbalancing.optional.
  <!--<incrementalScoreCalculatorClass>org.optaplanner.examples.cloudbalancing.op
  <!-- <scoreDrl>/cloudBalancingScoreRules.drl</scoreDrl> -->
  <scoreDrlFile>~/Nicolas/OptaConfig/cloudBalancingScoreRules.drl</scoreDrlFile>
  <initializingScoreTrend>ONLY_DOWN</initializingScoreTrend>
  <!--<assertionScoreDirectorFactory>-->
    <!--<easyScoreCalculatorClass>org.optaplanner.examples.cloudbalancing.optiona
  <!--</assertionScoreDirectorFactory>-->
</scoreDirectorFactory>
```

Screenshot of the Optaplanner file configuration



```
jim@Sofarserver:~/Nicolas/OptaConfig$ tree
.
├── cloudBalancingScoreRules.drl
└── cloudBalancingSolverConfig.xml

0 directories, 2 files
```

Content of the OptaConfig folder



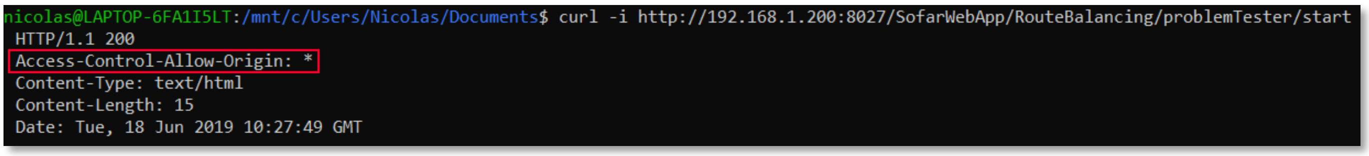
```
jim@Sofarserver:~/Nicolas/Tomcat9/webapps$ tree -L 1
.
├── docs
├── examples
├── host-manager
├── manager
├── ROOT
├── SofarWebApp
└── SofarWebApp.war

6 directories, 1 file
```

Content of the webapps folder

I also created two commands which permited Dimitris to Start and Stop the server, these commands are called “StartOptaplanner” and “StopOptaplanner”.

2.6.2 « Access-Control-Allow-Origin »

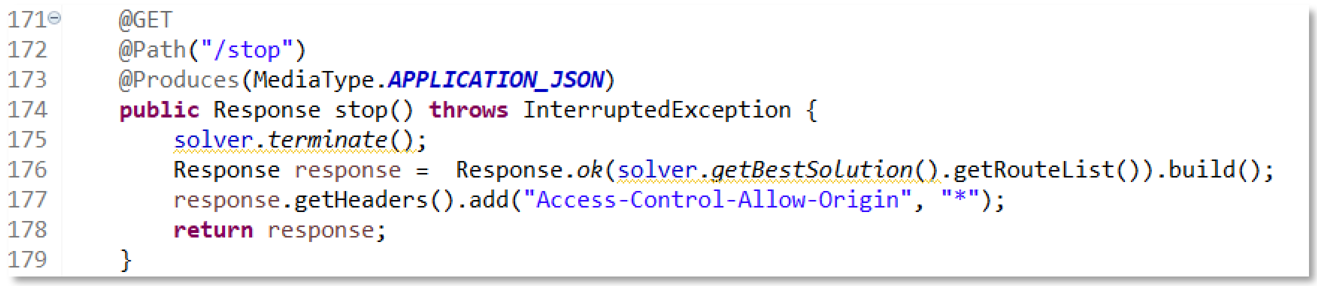
When you work with as many servers, the kriton server where I get my data, the SOFAR’s server where I started my service and of course my tomcat server… It’s impossible to avoid having any problems. One of the problems that I encountered was the famous “Access-Control-Allow-Origin” error. This error mean that a server does not want to communicate with a client.



```
nicolas@LAPTOP-6FA1I5LT:/mnt/c/Users/Nicolas/Documents$ curl -i http://192.168.1.200:8027/SofarWebApp/RouteBalancing/problemTester/start
HTTP/1.1 200
Access-Control-Allow-Origin: *
Content-Type: text/html
Content-Length: 15
Date: Tue, 18 Jun 2019 10:27:49 GMT
```

Header of the response of a request on my server

In fact, you need to have the line that I framed on the picture just above. There are many ways to do that, the one that I used was to write this line directly in my java code when I sent back the result.



```java
    @GET
    @Path("/stop")
    @Produces(MediaType.APPLICATION_JSON)
    public Response stop() throws InterruptedException {
        solver.terminate();
        Response response =  Response.ok(solver.getBestSolution().getRouteList()).build();
        response.getHeaders().add("Access-Control-Allow-Origin", "*");
        return response;
    }
```

Screenshot of the stop function in my java code

2.6.3 « Access Denied » and « Permission Denied »

In my service I had to get the bus list, the route list and the data concerning the time travel. So sometimes one of these links was inaccessible and I obtained a "Access Denied "or a "Permission Denied". But when this happened it was Dimitris who take care of the problem.

3. Technologies used

3.1 Languages

3.1.1 Java

Java is a general-purpose computer-programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible.

3.1.2 JavaScript

JavaScript is a high-level, interpreted, programming language that conforms to the ECMAScript specification. JavaScript has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.

3.1.3 Drools

Drools is a business rule management system with a forward and backward chaining inference based rules engine, more correctly known as a production rule system, using an enhanced implementation of the Rete algorithm. Drools supports the Java Rules Engine API standard for its business rule engine and enterprise framework for the construction, maintenance, and enforcement of business policies in an organization, application, or service.

3.1.4 HTML

Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

3.2 Servers

3.2.1 Tomcat

Apache Tomcat (also referred to as Tomcat Server) implements several Java EE specifications including Java Servlet, JavaServer Pages (JSP), Java EL, and WebSocket, and provides a "pure Java" HTTP web server environment in which Java code can run. Tomcat is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation, released under the Apache License 2.0 license, and is open-source software.

3.1.2 WildFly

WildFly, formerly known as JBoss AS, or simply JBoss, is an application server authored by JBoss, now developed by Red Hat. WildFly is written in Java and implements the Java Platform, Enterprise Edition (Java EE) specification. It runs on multiple platforms.WildFly is free and open-source software, subject to the requirements of the GNU Lesser General Public License (LGPL), version 2.1.

3.3 Tools

3.3.1 Eclipse IDE

Eclipse is an integrated development environment (IDE) used in computer programming, and in 2014 was the most widely used Java IDE in one website's poll. It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages via plug-ins.

3.3.2 Visual Studio Code

Visual Studio Code is a source-code editor developed by Microsoft for Windows, Linux and macOS. It includes support for debugging, embedded Git control and GitHub, syntax highlighting, intelligent code completion, snippets, and code refactoring. It is highly customizable, allowing users to change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.

3.3.3 Firefox

Mozilla Firefox, or simply Firefox, is a free and open-source web browser developed by the Mozilla Foundation and its subsidiary, Mozilla Corporation. Firefox is available for Microsoft Windows, macOS, Linux, BSD, illumos and Solaris operating systems. Its sibling, Firefox for Android, is also available.

3.3.4 Postman

Postman is a great tool when trying to dissect RESTful APIs made by others or test ones you have made yourself. It offers a sleek user interface with which to make HTML requests, without the hassle of writing a bunch of code just to test an API's functionality.

3.3.5 Debian for Windows

Now it's possible to download an application on the Microsoft Store to get a Debian Subsystem. With this one you will be able to use a complete Debian command line environment containing a full current stable release environment.

3.4 Others

3.4.1 Optaplanner

OptaPlanner is an Open Source Constraint Solver written in Java. It solves constraint satisfaction problems with construction heuristics and metaheuristic algorithms. It's sponsored by Red Hat, part of the JBoss community and closely related to the Drools and jBPM projects in the KIE group.

3.4.2 MODx

MODx is a free, open source content management system and web application framework for publishing content on the world wide web and intranets. MODx is licensed under the GPL, is written in the PHP programming language, and supports MySQL and Microsoft SQL Server as the database. It was awarded Packt Publishing's Most Promising Open Source Content Management System in 2007.

3.3.3 JSON

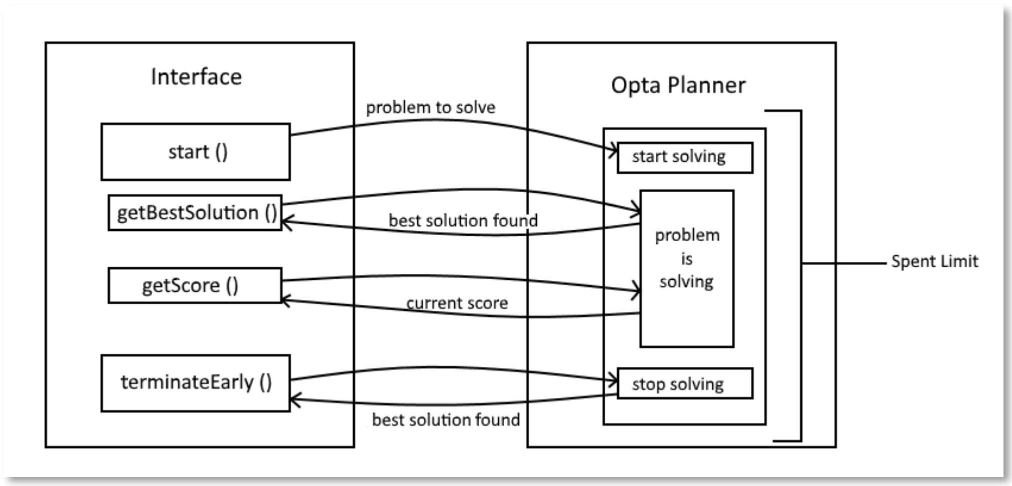
In computing, JavaScript Object Notation (JSON) is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types . It is a very common data format used for asynchronous browser–server communication, including as a replacement for XML in some AJAX-style systems.

3.3.4 XML

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The W3C's XML 1.0 Specification and several other related specifications all of them free open standards define XML. It is a textual data format with strong support via Unicode for different human languages. Although the design of XML focuses on documents, the language is widely used for the representation of arbitrary data structures such as those used in web services.

Summary

To sum it up there were two big part in my web service. The first one did all the GET to get the data, it was also from here that you could use the methods to Start the solving/Get the best solution/Get the score/Stop the solving. This part is called "Interface" on my diagram just beneath. The second part is my modified version of Optaplanner, which used Buses and Routes.



Summary diagram of my project

Conclusion

This internship in Zakynthos was an incredible experience, both with the project and the Erasmus experience. I learnt so much here, new ways for programming in Java, new programming languages, new software and more. It was really awesome to speak in English with Dimitris and also the others members of the SOFAR's team! I have improved my English level exactly as I wanted before coming!

This experience was very enriching about the technologies that I used but also about the culture that I discovered. I met a lot of students from Greece and different countries. It was an incredible life experience to share with all of them, and they made me grew up so fast in just 3 months. I will never be able to thank them enough about what they brought me and this is the same for Dimitris who welcomed us in his company. This erasmus is now a part of my life, a part of my mind, and I will be so glad to spread the Erasmus mind as soon as I will be back to France. Thanks to Dimitris again I was able to discover others part of the Greece like Athens, Mykonos and of course Zakynthos. Greece is a wonderful country, as beautiful as France. It was my first time abroad for a so long time, because I have never been three mounts in another country. In this way I became more independent and adaptable to any situation.

To sum it up, I recommend to everybody to try to live this kind of experience! I would like to say that this island is my second home! And the most important thing I also would like to say thank you to **Pr. Dimitris Halvatzaras!**

Glossary

TEI: Technological Educational Institute (Τεχνολογικό Εκπαιδευτικό Ίδρυμα).

API: A good API makes it easier to develop a computer program by providing all the building blocks, which are then put together by the programmer.

IDE: Integrated Development Environment.

HTML: HyperText Markup Language

CSS: Cascading Style Sheets

XML: Extensible Markup Language

JSON: JavaScript Object Notation

References

Wikipedia

Greeceindex

Bouger-Voyager

SOFAR

Kriton

Java Doc

Stack Overflow

Postman

Firefox

Optaplanner

Drools

Google Maps