

**Solent
University
in Partnership with QA (QAHE)**

Module Title: Object Oriented Design and Development

Module Code: QHO543

Module Leader: Tendai Mhlanga

Assessment Title: Underground Train Ticket System

Assessment Type: Software Development

Student name: Felix Constantin Tudorascu

University id: 15077250

ID: 10103209

INTRODUCTION	2
IMPLEMENTATION	3
CLIENTAPPLICATION	3
TFL_JSP	3
VALIDATORGATE.JSP	4
SOURCE PACKAGES	5
JAVA PACKAGES	5
<i>ClientConsumerServiceRestful_TFL_Underground</i>	5
Consumer_TFL_underground	5
<i>ControllersClient_TFL</i>	6
TicketGenerator_TFL_Underground	6
<i>Model_TFL_Underground</i>	6
Station	6
Ticket	7
ValidatorTicket	8
RESTWEBSERVICES_TFL_UNDERGROUND	8
SOURCE PACKAGES	8
<i>Controllers</i>	8
JDBCHandadeller	8
<i>Facade</i>	9
StationsFacade	9
LIBRARIES USED	10
Java EE Web 8 API	10
Connector	10
JDK version	10
HOW CAN YOU CHECK RESTFUL WEB SERVICES?	11
WHAT IS NOT WORKING AND WHY?	11
CLASS DIAGRAMS	12

Figure 1 ClientApplication	3
Figure 2 TFL_JSP	4
Figure 3 ValidatorJSP	4
Figure 4 ClientConsumer_ServiceRestful	5
Figure 5 Ticket Generator	6
Figure 6 Station	7
Figure 7 Ticket	7
Figure 8 ValidatorTicket	8
Figure 9 JDBCHandeller	8

Introduction

In a few words, I'll explain what the project is and how it works, so I created an Underground Ticket System where you can choose the current location at that time, the location where you want to travel and choose and until what date the ticket will be valid. After that you gonna receive an xml code with your chosen location and the time and date. After that start the ValidatorGate.jsp, insert

the code down bellow in the textarea and you will get one of the two messages, you are allowed to Enter or the gate will remain Closed. In the next few chapters it will demonstrate how the code work, what is not working and why. Let's start!

Implementation

Before you start to build the project, you must install the following tools:

Apache NetBeans IDE 12.0

Xampp 8.1.6

MySQL

JAVA

JDK 1.8

Maven

Glassfish

Tomcat or TomEE

After you install all of this, create a project with maven/ Web application, add some Java Dependencies like JDK 11, Javaee-endorsed-api-7.0.jar and in the RestServices add in Libraries Java EE Web 8 API - javaee-web-api-8.0.jar, the connector mysql-connector-java-8.0.28.jar, JDK 8 and add the glassfish server.

In the meantime, let's create a database with 12 stations, so let's use XAMPP, start Apache and MySQL Module, after click on MySQL/Admin and from there you can create a simple database.

Start the Glassfish server in the NetBeans Services, start the Apache Tomcat, connect your database with the NetBeans Services using the connector installed earlier.

Let's dig in and see step by step how the project is working.

ClientApplication

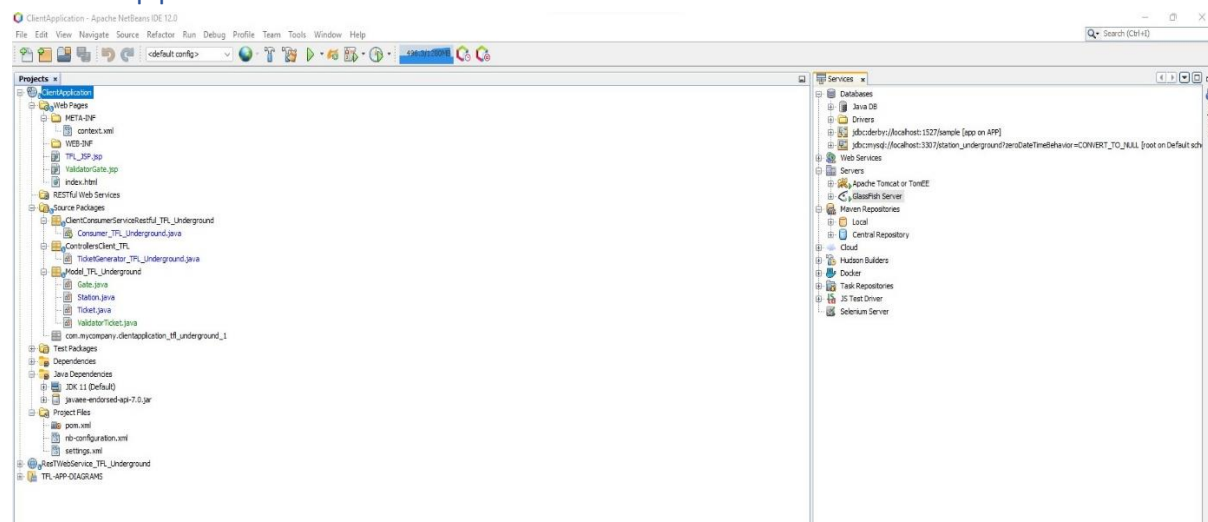


Figure 1 ClientApplication

In figure 1 you can see what ClientApplication contains, what servers you start and database is connected to this project. You gonna see below how everything was created and how is working.

TFL_JSP

In Figure 2, I created the JSP file, the interface that the user sees and uses where you can choose the station from where you leave, the station where you have to arrive and the date until the ticket will

be valid. The form contains embedded java code where I populate first list with my stations from database. Now you can just from the drop down one of your stations. Same thing with the second list but from that drop down you choose the destination station. Pressing the Create XML button will create an XML code with your information.

```

1 Document : TFL_JSP
2 Author : Tudorascu Felix
3
4 <%@page import="java.util.Date"%>
5 <%@page import="java.text.SimpleDateFormat"%>
6 <%@page import="ControllersClient_TFL.TicketGenerator_TFL_Underground"%>
7 <%@page import="Model_TFL_Underground.Ticket"%>
8 <%@page import="java.util.List"%>
9 <%@page import="Model_TFL_Underground.Station"%>
10 <%@page contentType="text/html" pageEncoding="UTF-8"%>
11
12 <!DOCTYPE html>
13 <html>
14 <head>
15 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
16 <title>JSP Page</title>
17 </head>
18 <body>
19 <form action="TFL_JSP.jsp" id="userform">
20 <label for="label_fromstation">From Station:</label>
21 <select name="fromstation" id="dropdownStation">
22
23 <%
24 List<Station> listOfStation_TFL_From = ClientConsumerServiceRestful_TFL_Underground.Consumer_TFL_Underground.getStations();
25 for ( int i = 0; i < listOfStation_TFL_From.size(); i++ ) {
26 out.print("<option value=" + i + ">" + listOfStation_TFL_From.get(i).getStationname() + "</option>");
27 }
28 %>
29 </select>
30 <br>
31 <br>
32
33 <label for="label_toystation">To Station:</label>
34 <select name="toystation" id="dropdownStation">
35
36 <%
37 List<Station> listOfStation_TFL_TO = ClientConsumerServiceRestful_TFL_Underground.Consumer_TFL_Underground.getStations();
38 for ( int i = 0; i < listOfStation_TFL_TO.size(); i++ ) {
39 out.print("<option value=" + i + ">" + listOfStation_TFL_TO.get(i).getStationname() + "</option>");
40 }
41 %>
42 </select>
43 <br>
44 <br>
45
46
47
48

```

Figure 2 TFL_JSP

ValidatorGate.JSP

In this figure you can see how interface look like, how the user see the project, and creating the XML ticket code how you can validate if the gate remains Closed or will be Open.

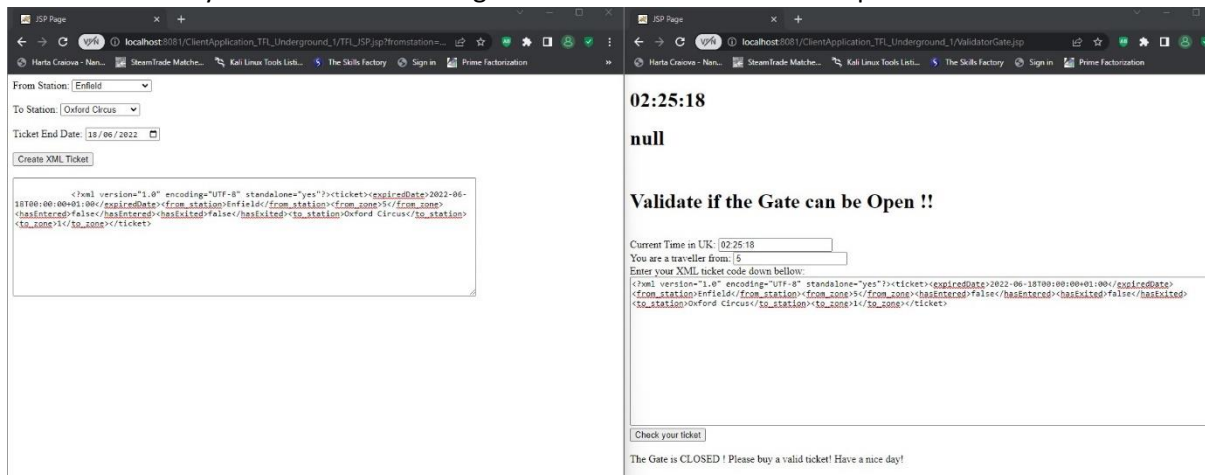


Figure 3 ValidatorJSP

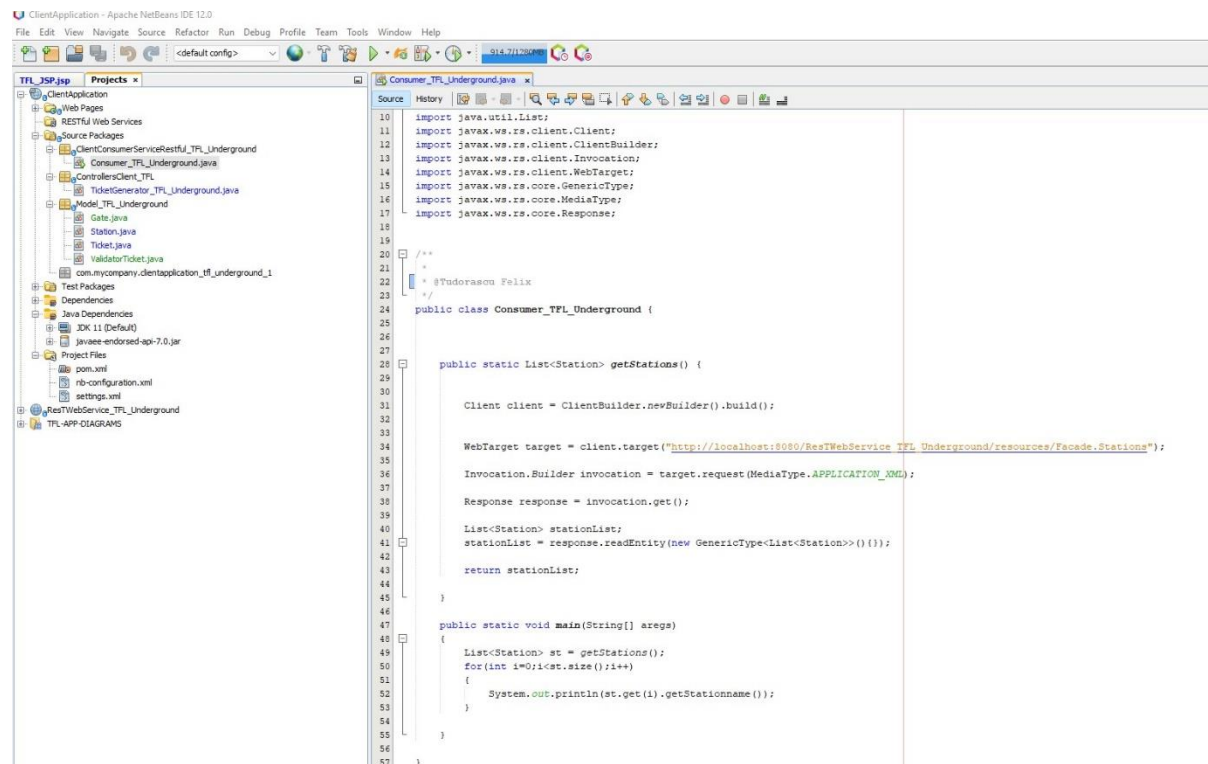
Source Packages

JAVA Packages

ClientConsumerServiceRestful_TFL_Underground

Consumer_TFL_underground

In this figure I created a Java class, where I populate list with our stations from database using the commands down below. You must send requests to database to get a response with your stations. Import Java.util.List and many more. The WebTarget interface is used to represent a specific URI that you wish to call. You may build a WebTarget using one of the target() methods from the Client interface, as shown in Fig. 4.



```
10 import java.util.List;
11 import javax.ws.rs.client.Client;
12 import javax.ws.rs.client.ClientBuilder;
13 import javax.ws.rs.client.Invocation;
14 import javax.ws.rs.client.WebTarget;
15 import javax.ws.rs.core.GenericType;
16 import javax.ws.rs.core.MediaType;
17 import javax.ws.rs.core.Response;
18
19
20 /**
21  *
22  * @Tudorason Felix
23  */
24 public class Consumer_TFL_Underground {
25
26
27
28
29
30     public static List<Station> getStations() {
31
32
33         Client client = ClientBuilder.newBuilder().build();
34
35         WebTarget target = client.target("http://localhost:8080/RestWebService_TFL_Underground/resources/Facade.Stations");
36
37         Invocation.Builder invocation = target.request(MediaType.APPLICATION_XML);
38
39         Response response = invocation.get();
40
41         List<Station> stationList;
42         stationList = response.readEntity(new GenericType<List<Station>>() {});
43
44         return stationList;
45     }
46
47     public static void main(String[] args)
48     {
49         List<Station> st = getStations();
50         for(int i=0;i<st.size();i++)
51         {
52             System.out.println(st.get(i).getStationname());
53         }
54     }
55
56
57 }
```

Figure 4 ClientConsumer_ServiceRestful

ControllersClient_TFL

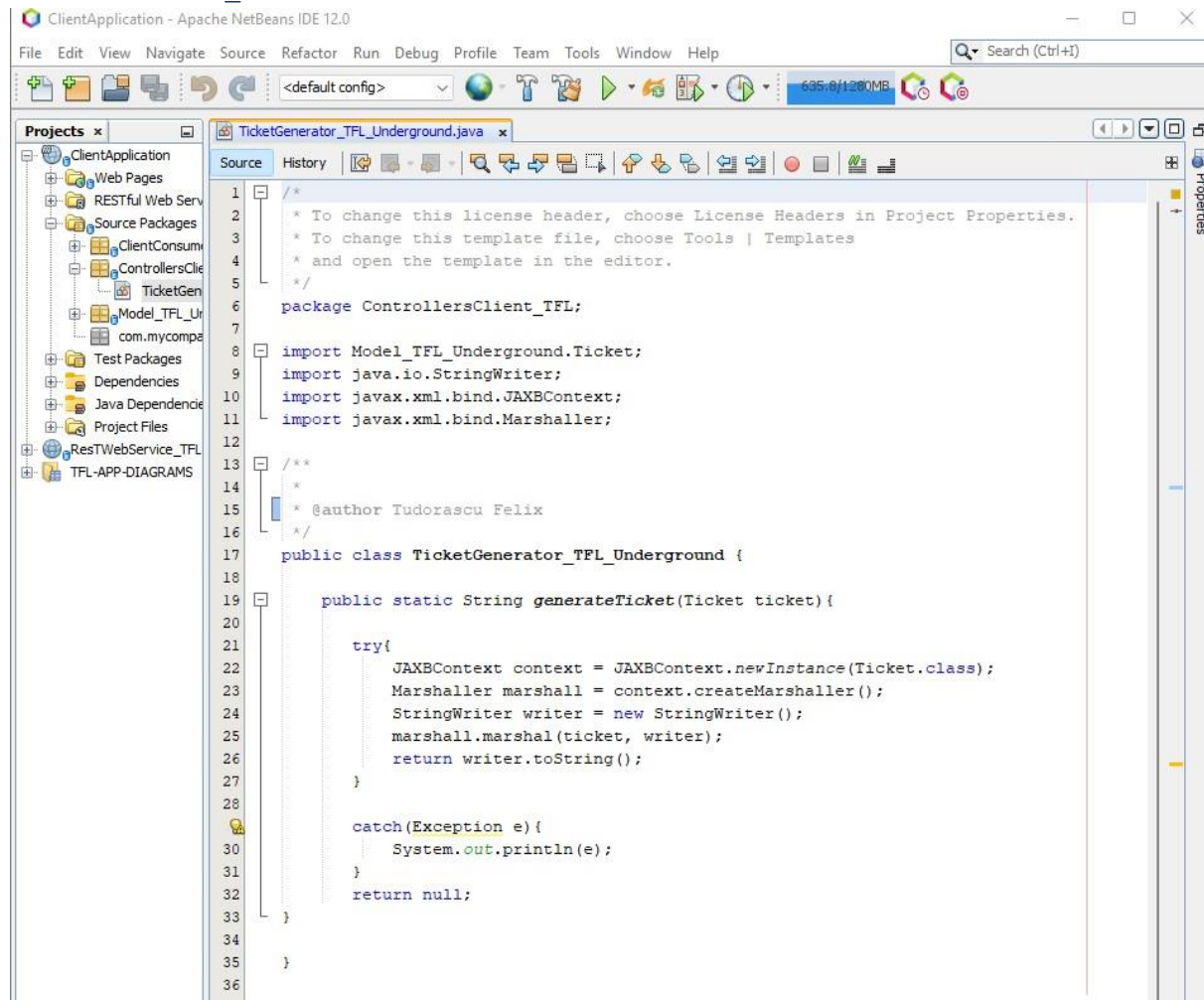


Figure 5 Ticket Generator

The JAXBContext class is used by the client to access the JAXB API. It acts as a wrapper for the XML/Java binding information required to accomplish the JAXB binding framework functions of unmarshal, marshal, and validate.

TicketGenerator_TFL_Underground

Model_TFL_Underground

Station

When the @XmlRootElement annotation is added to a top-level class or enum type, its value is represented as an XML element in an XML document. In this java class, I defined the classes for Stations, like in the Fig down below and it is: String for stationname, int for stationid and zone, I created the constructor and I added the Getter and Setters.

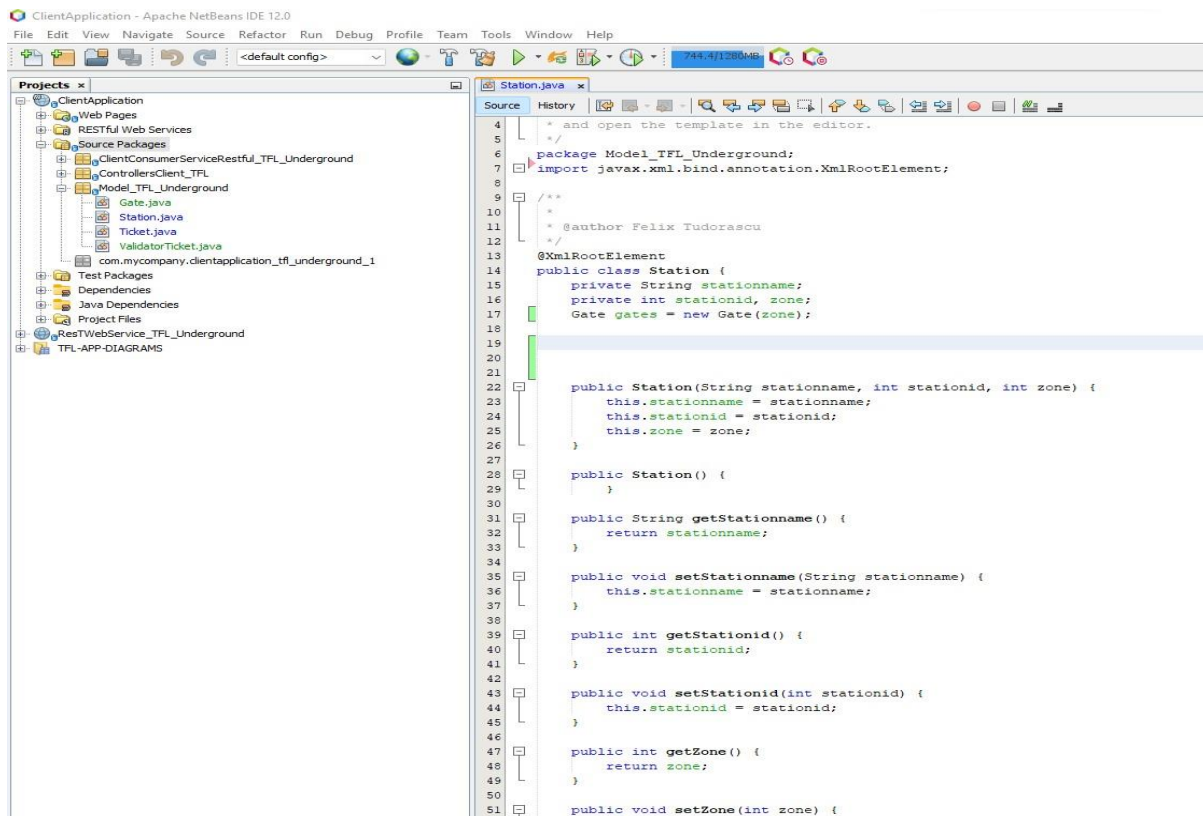


Figure 6 Station

Ticket

The Ticket was basically created the same as Stations but with different values: String from_station, to_station, from_zone, to_zone, Boolean hasEntered, hasExited, Date.

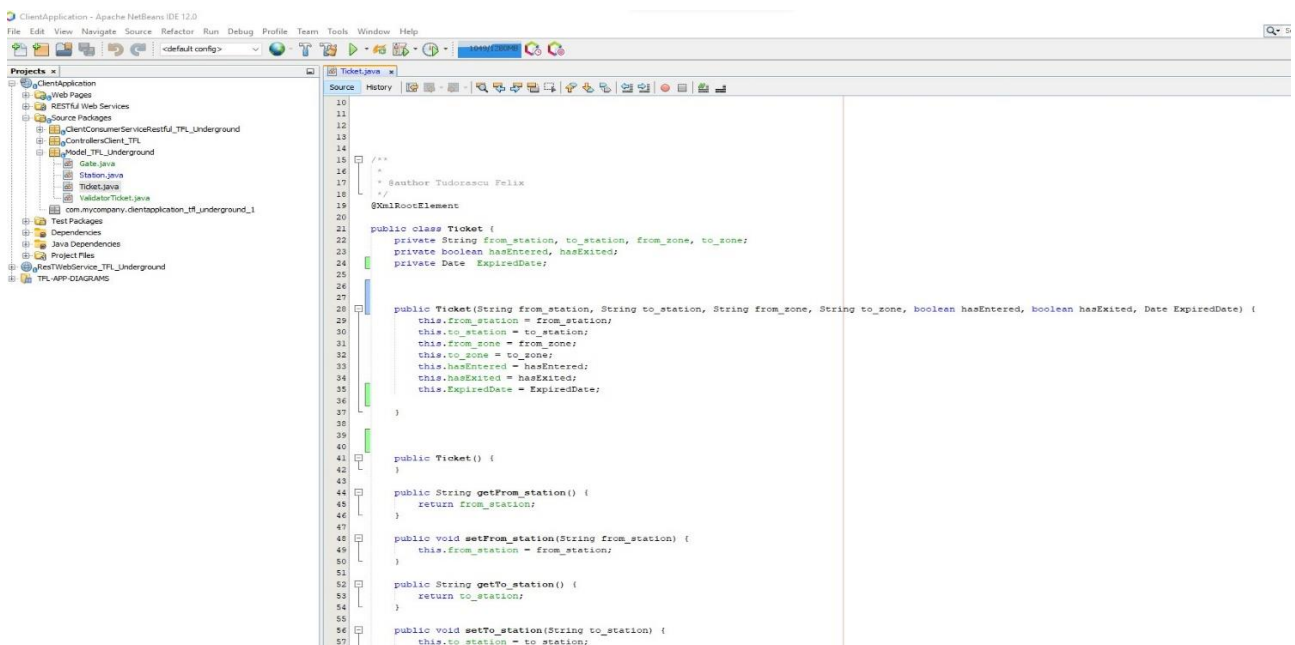


Figure 7 Ticket

ValidatorTicket

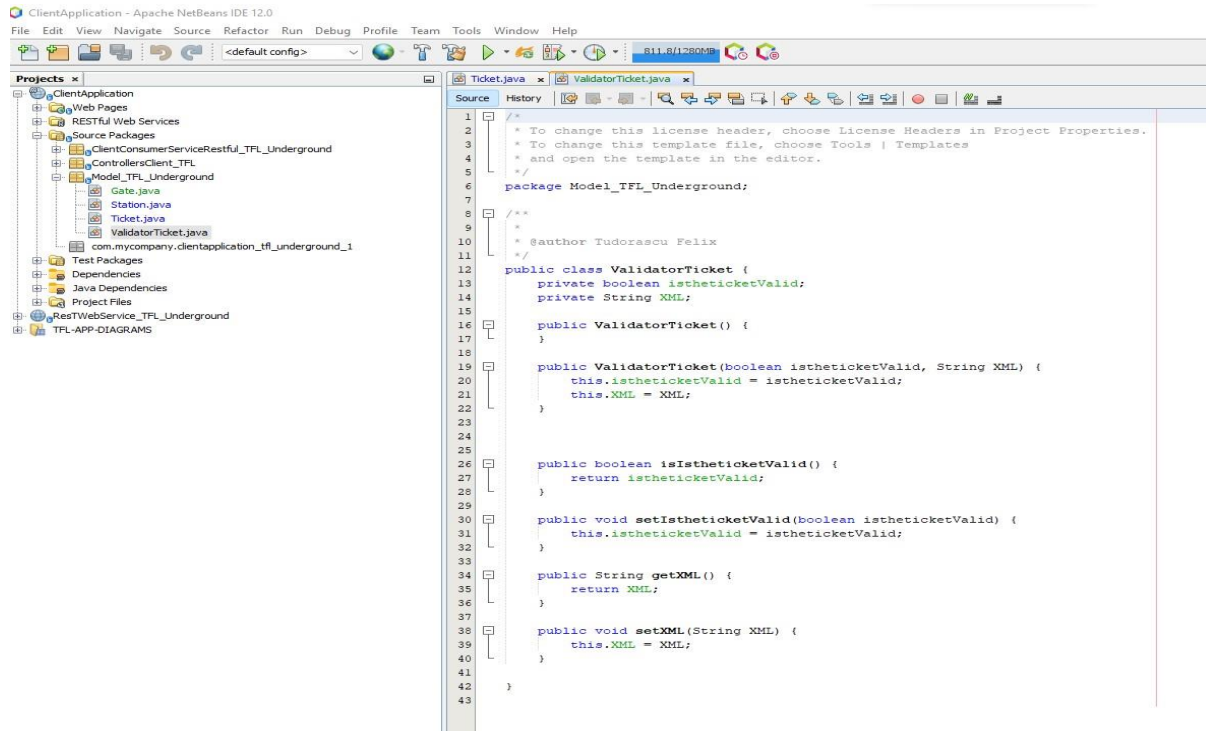


Figure 8 ValidatorTicket

RestWebServices_TFL_underground

Source Packages

Controllers

JDBCHandadeller

The Generic Java Database Connectivity (JDBC) handler allows you to replicate transactional data from a source system or database to a target system or database. How you can see in Figure 9 I create the connection between my Database with my application using my username, password and URL of my local database.

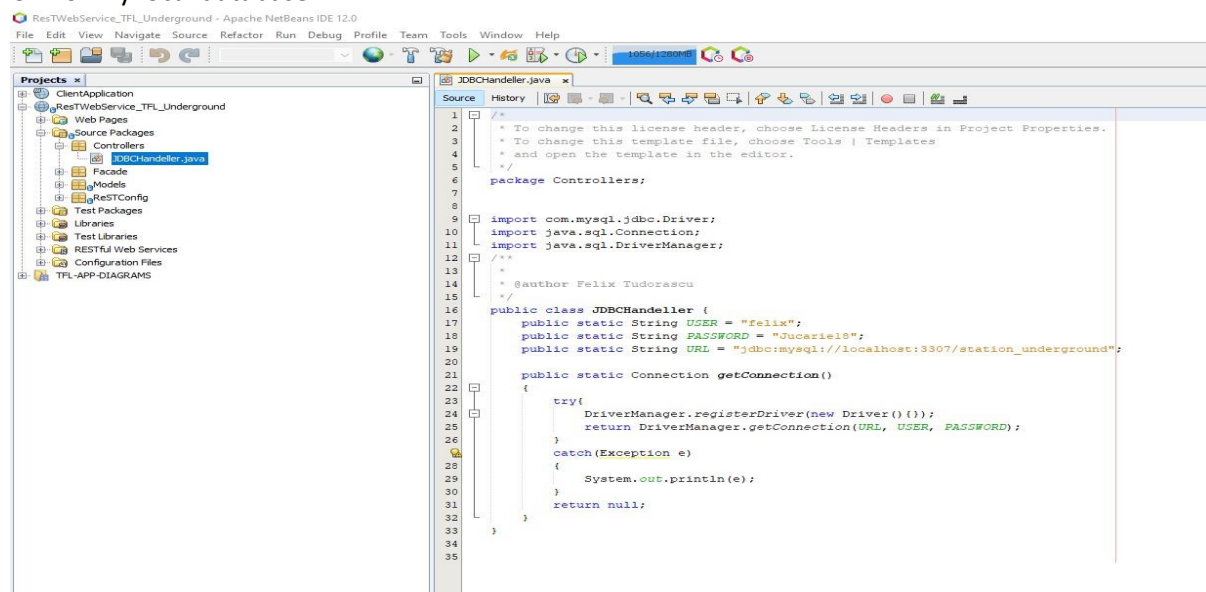
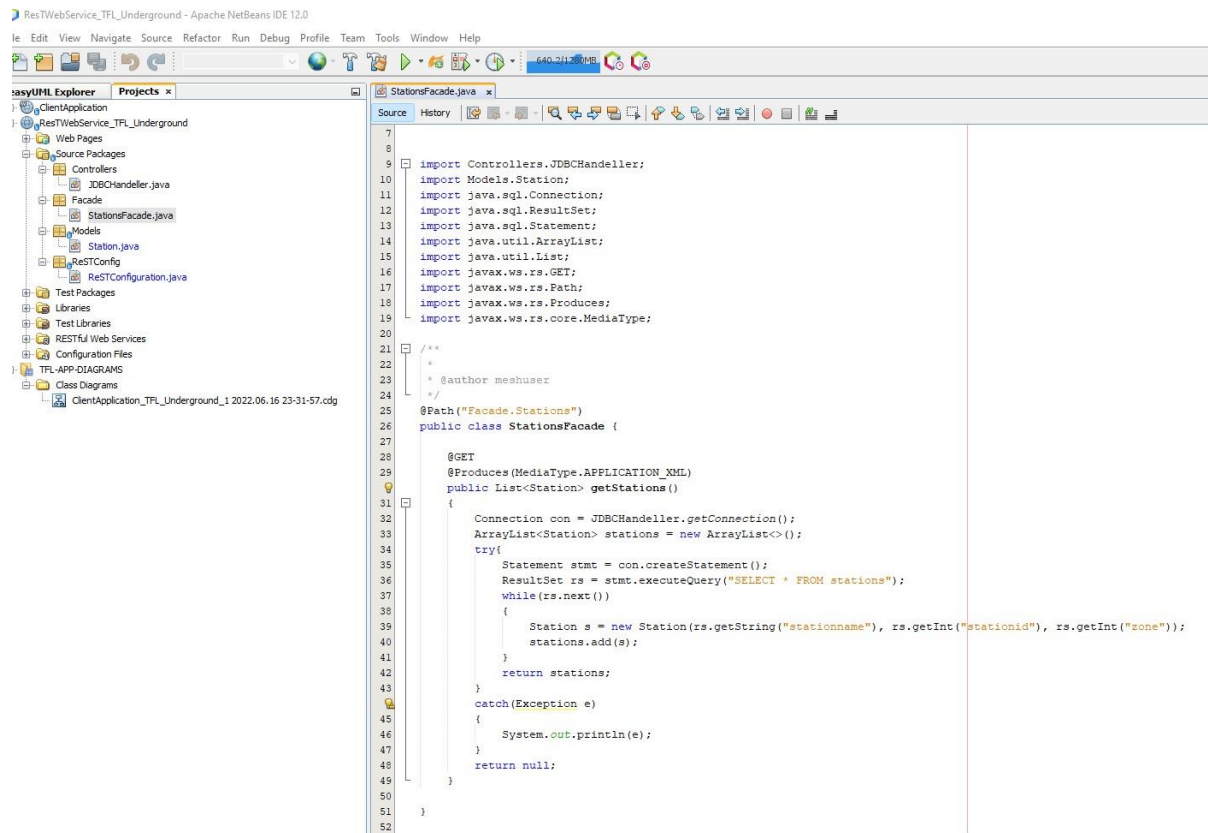


Figure 9 JDBCHandler

Facade

StationsFacade

In this Java class I just used JDBCHandler to create the connection with my database and import and execute the Query to get all my stations with stationname, stationid, zone (id).



```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52

import Controllers.JDBCHandler;
import Models.Station;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

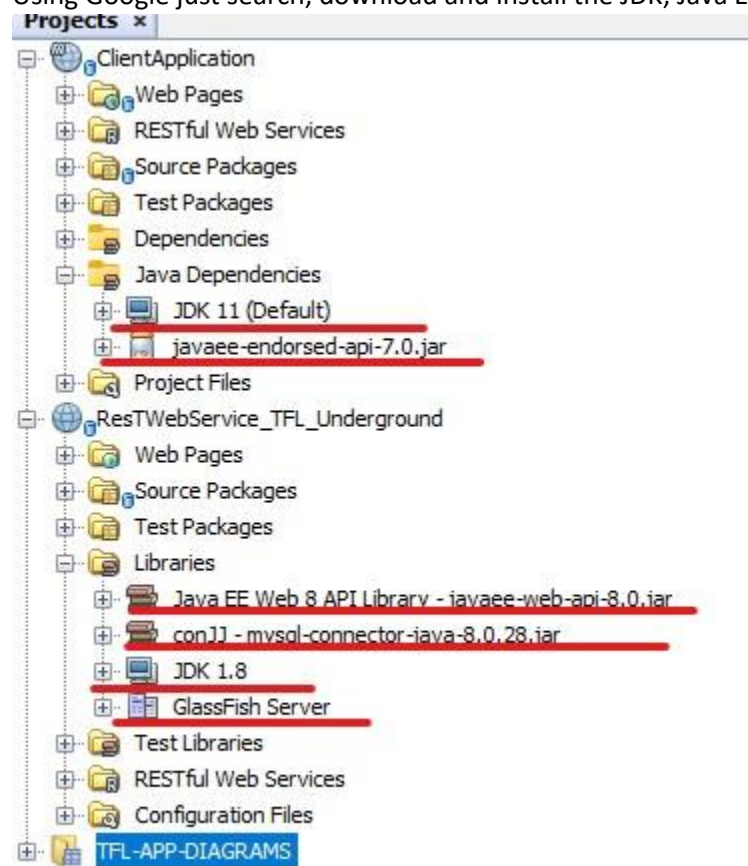
/**
 * @author meshuser
 */
@Path("/Facade.Stations")
public class StationsFacade {

    @GET
    @Produces(MediaType.APPLICATION_XML)
    public List<Station> getStations()
    {
        Connection con = JDBCHandler.getConnection();
        ArrayList<Station> stations = new ArrayList<>();
        try{
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM stations");
            while(rs.next())
            {
                Station s = new Station(rs.getString("stationname"), rs.getInt("stationid"), rs.getInt("zone"));
                stations.add(s);
            }
            return stations;
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
        return null;
    }
}
```

Figure 10 SationFacade

Libraries used

Using Google just search, download and install the JDK, Java EE and install the connector.



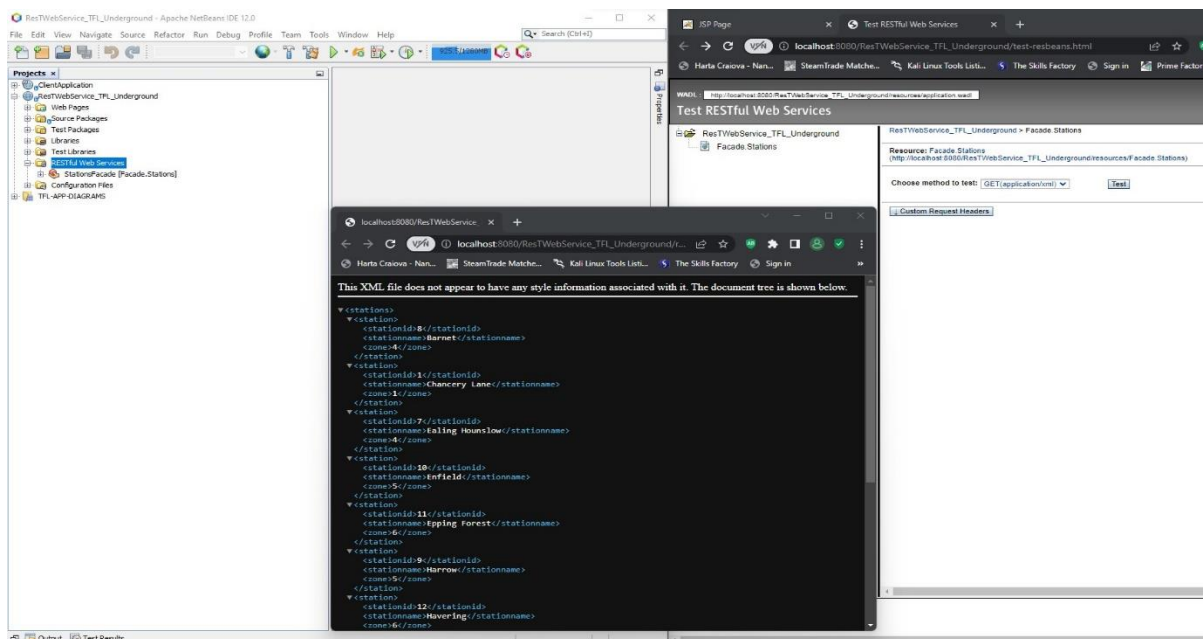
Java EE Web 8 API

Connector

JDK version

How can you check ReSTful Web Services?

If you want to check the Resful_web_services just right click, test Restful web services, press OK and you gonna get this page, so basically everything working perfect.



What is not working and why?

I was trying to finish all the project, but I failed. I was stuck in one point when my XML code is not unmarshalled so I cannot enter in the station. I tried many things, but I did not receive any favourable result. I made a lot of research but in the end nothing helped.

Class Diagrams

A class diagram is a diagram used to illustrate classes and their connections in software design and modelling. Without having to look at the source code, class diagrams allow us to represent software at a high degree of abstraction. A class diagram's classes match to the source code's classes. The diagram depicts the names and properties of the classes, as well as their links and, in certain cases, their methods.

