Université Gustave Eiffel

Web Services project

FENNI Hassen FREDJ Najeh

Sunday 29 November, 2020

Supervised by :

Mr. Mahdi Zargayouna

Contents

List of Figures

1 Introduction

A **Web service** is an application that can be consumed (accessed and used) over the Internet (or an intranet) using industry-standard protocols. To consume this application, a client (Web service client) invokes operations that are described using Web Services Description Language (WSDL) and sent to the application using Simple Object Access Protocol (SOAP) over HTTP (or HTTPS). The application executes the operations and returns the results to the client using SOAP over HTTP/S. A Web service provider (Web application server) hosts the Web service and manages all communications between it and its clients. Simply put, Web services provide an industry-standard way for all types of client applications to call functions on all types of application servers, over any network configuration that supports SOAP over HTTP, and where the application program interface (API) can be described using WSDL.

In this project, we used web services to make the company Eiffel Corp which has just acquired the company IfsCars, specialized in car rental to enhance its vehicle base, enriched by its employees' notes, and make it accessible to the outside world.

To do that we applied **Remote method invocation** (RMI) first, to make Eiffel Corp's employees benefit from IfsCars's services at a preferential price.

Hence, RMI is a programming interface (API) for the Java language which allows remote methods to be called.

2 Project framework

2.1 Problematic

In the context of applying our knowledge about SOAP Web Services and use of the RMI middleware.

2.2 Objective

The mean objective of this project is to satisfy Eiffel Corp's employees and customers who can be able to run the applications with minimal user intervention in order to rent and return vehicles or to constitute a basket, to validate it and to pay.

2.3 Tools and technologies

2.3.1 Eclipse

Eclipse is an integrated development environment (IDE) used in computer programming.It contains a base workspace and an extensible plugin 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 plugins.

Figure 2.1: Eclipse logo

2.3.2 SoapUI

SoapUI is an open source and it is the world's most widely-used automated testing tool for SOAP and REST APIs. Write, run, integrate, and automate advanced API Tests with ease.

Figure 2.2: SoapUi logo

2.3.3 Java

Java is an object oriented programming language. A special feature of Java is that software written in this language is compiled to an intermediate binary representation that can be executed in a Java Virtual Machine (JVM) without regard to the operating system.

2.3.4 Soap(Simple Object Access Protocol)

Simple Object Access Protocol (SOAP) is a messaging protocol. It allows programs that run on separate operating systems (such as Windows and Linux) to communicate using HyperText Transfer Protocol (HTTP) and its language, Extensible Markup Language (XML).

2.3.5 Tomcat

Apache Tomcat is a free web container for servlets and JSP. From the Jakarta project, it is one of the many projects of the Apache Software Foundation

Figure 2.3: Tomcat logo

2.3.6 Jetty

Eclipse Jetty is an HTTP server and servlet engine based entirely on Java technology.

2.3.7 JSF Primefaces

JavaServer Faces (abbreviated as JSF) is a Java framework for developing web applications.

3 Functional analysis and work architecture

3.1 Use case diagram

To allow users to express their needs and understand the functionality that the system will provide. We will defend who are the users of the application and what do they want to do in the system. This use case (Figure 3.1) represents a discrete unit of interaction between the user and the system, it is a significant unit of work.
In a use case diagram, users are called actors and they interact with use cases: in our case, the actors are the employees and the customers .

3.2 Class diagram

A lass diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. In our case we have three classes: employee, vehicle and account. The emloyee can rent one or more vehicles and for eatch employee we have an account.

3.3 Sequence diagram

The system sequence diagram is the graphic representation of the interactions between the actors and the system for a dynamic analysis. The aim is to describe how actions take place between actors or objects.

In order to describe how the different global actions take place between employee and the vehicle,The system sequence diagram illustrated by the Figure (4.3) describes the general functioning of the system and the different exchanges: In our case the employee have to check if the wanted car is on the free to rent vehicles list then he can rent it.

3.4 Work architecture

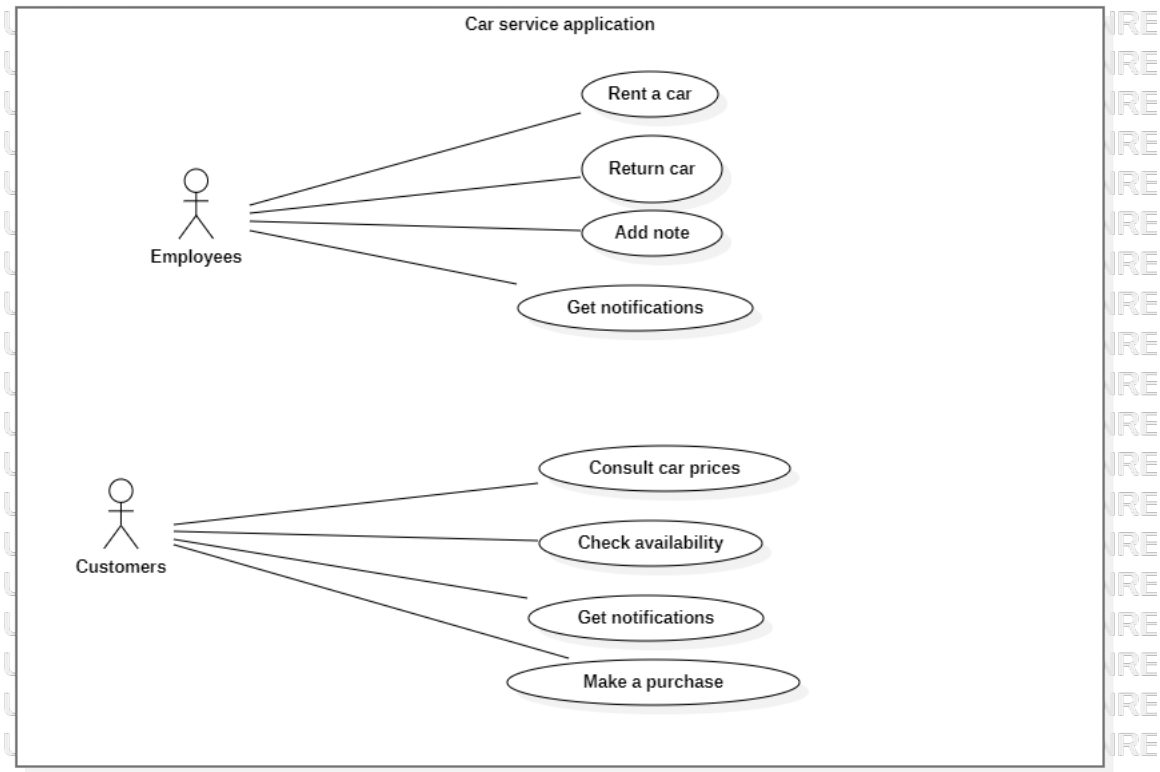

Figure 3.1: Use case diagram

Figure 3.2: Class Diagram

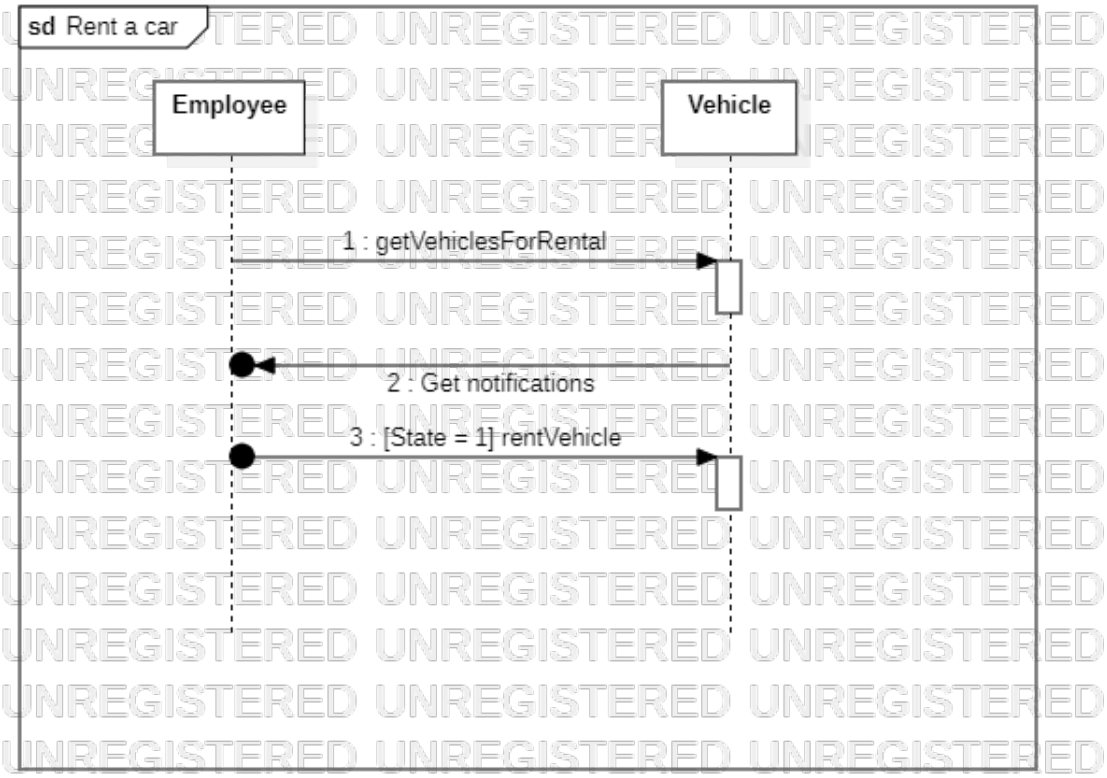

Figure 3.3: Sequence diagram

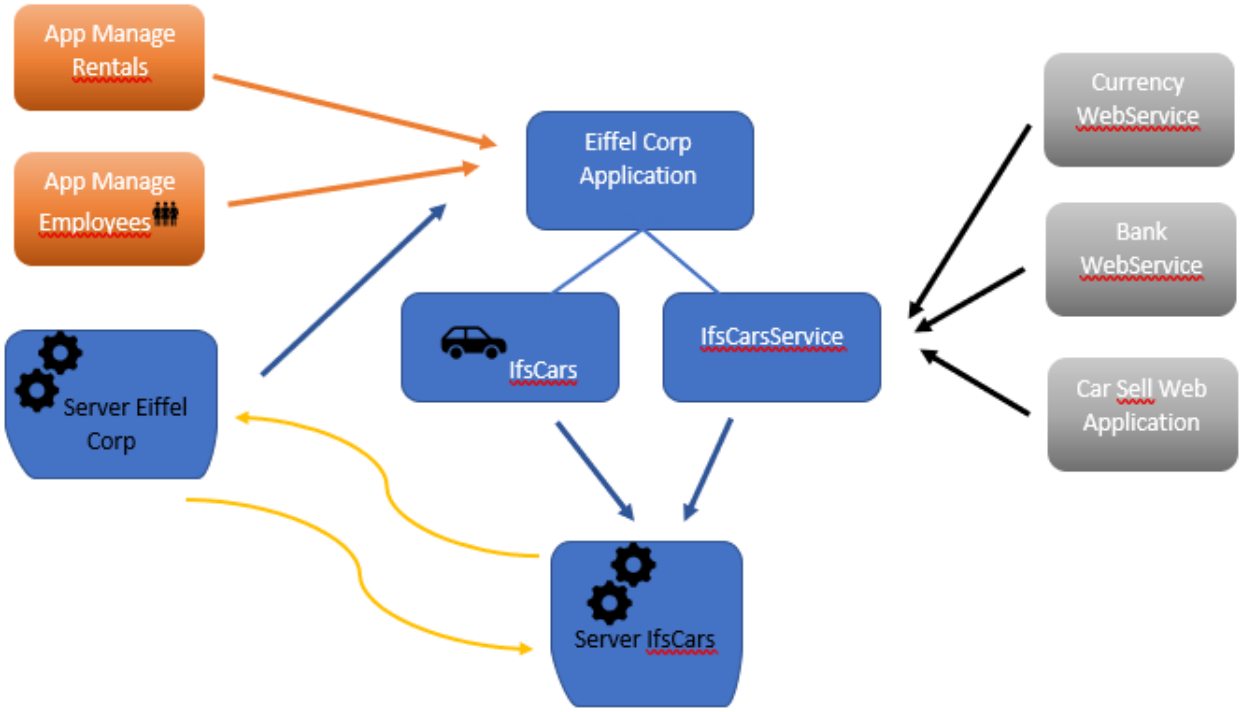

Figure 3.4: SWork architecture

4 Realisation

4.1 the difficulties encountered

We have considered several problems during the resolution of the project especially in the design of the architecture of a robust distributed application and in the configuration of web serices. Difficulty was encountered in classing the RMI's sub-projects and web services. The choice of implementing the right methods represented another difficulty that have been treated. Finally , deploying the graphical interfaces was not an easy task.

4.2 Demostration

4.2.1 design choices

To run the project we need to :

- Run the 2 RMI server Projects:EiffelCorp and IfsCarsRMI. - Add the bank project in Tomcat server 9 then lunch the two servers that will host our websites (The rental and sales websites) each of them runs on a different e jetty server

4.2.2 Use guide

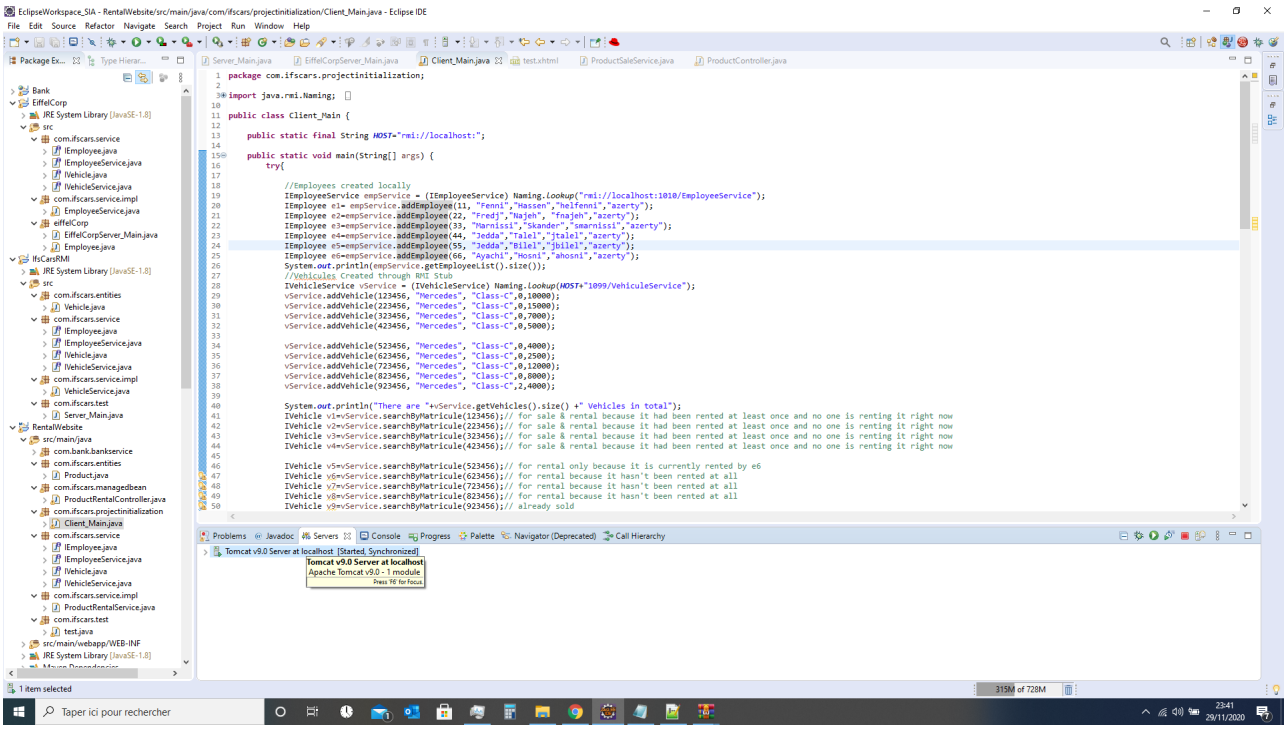

Figure 4.1: Step 1

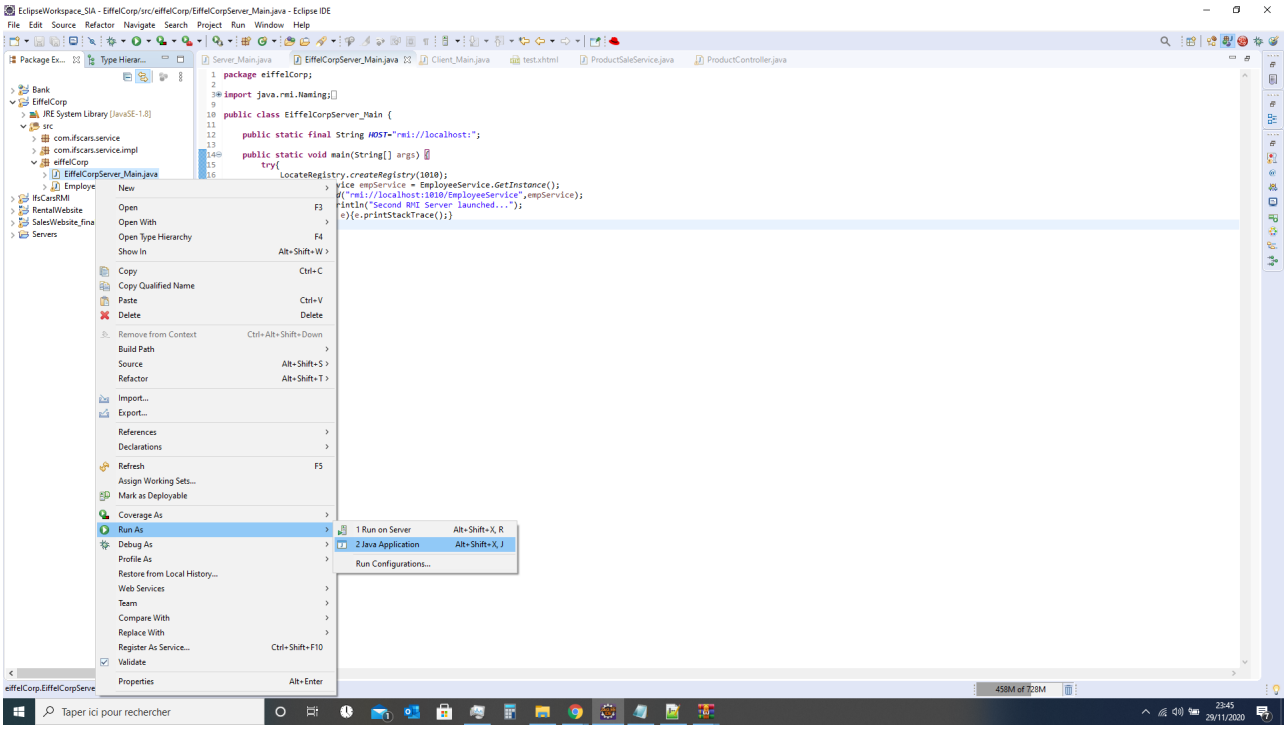

Figure 4.2: Step 2

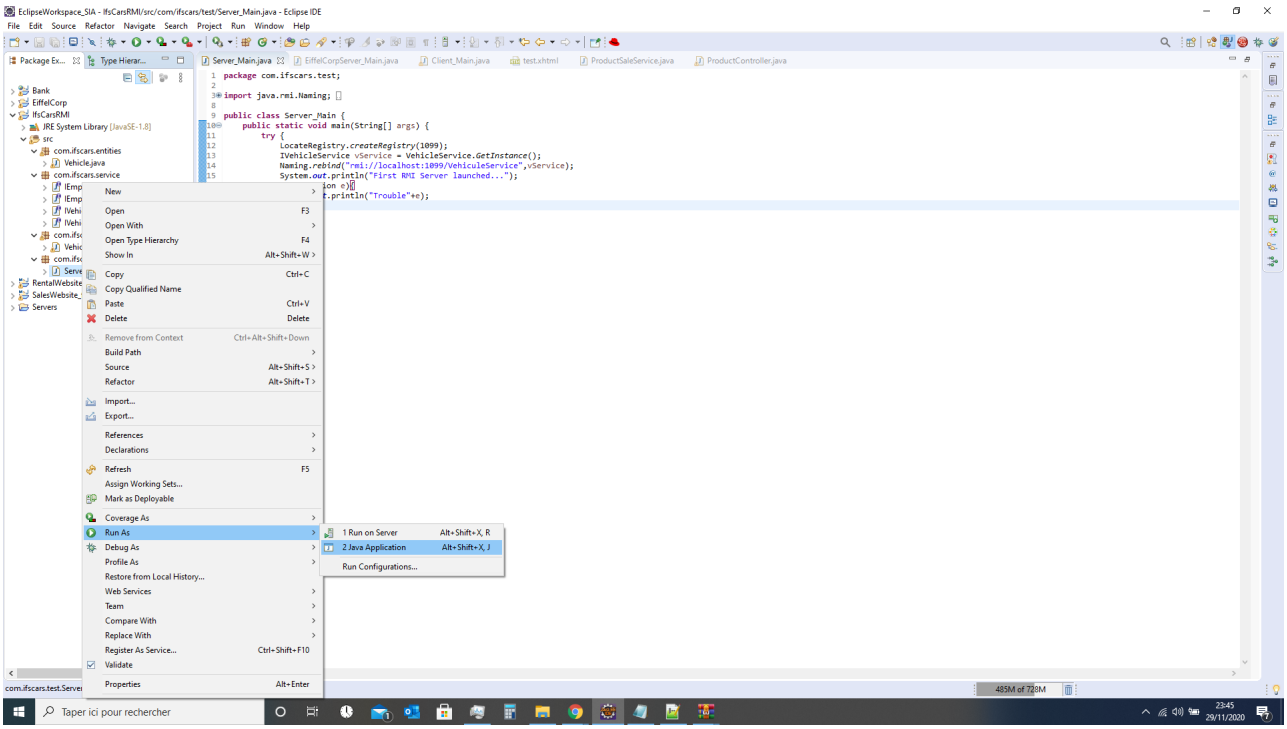

Figure 4.3: Step 2

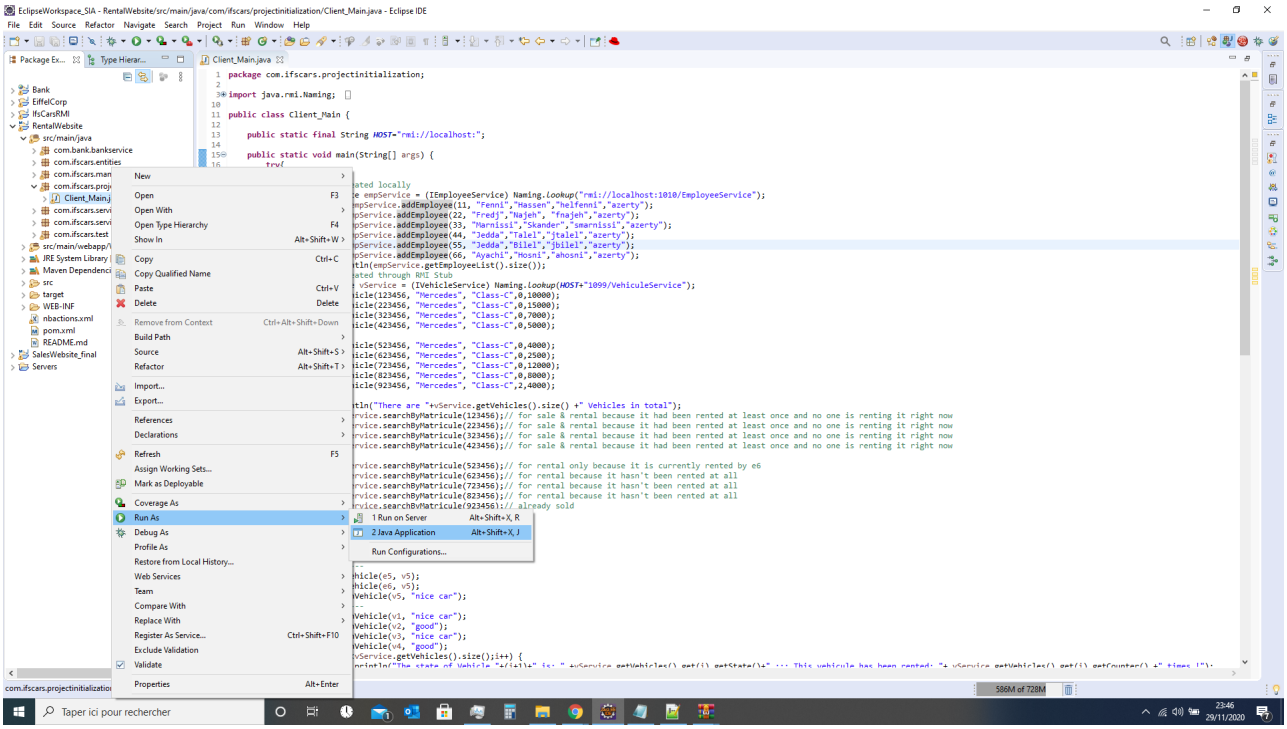

Figure 4.4: Step 3

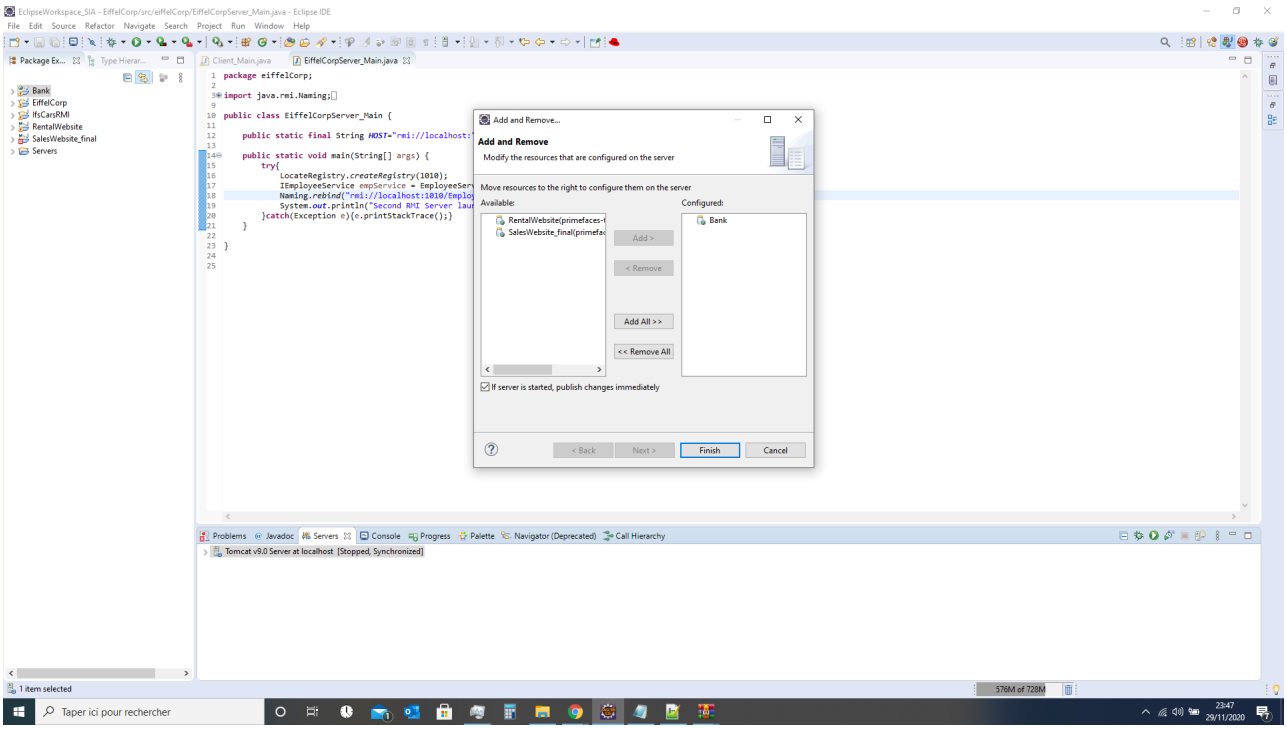

Figure 4.5: Step 4

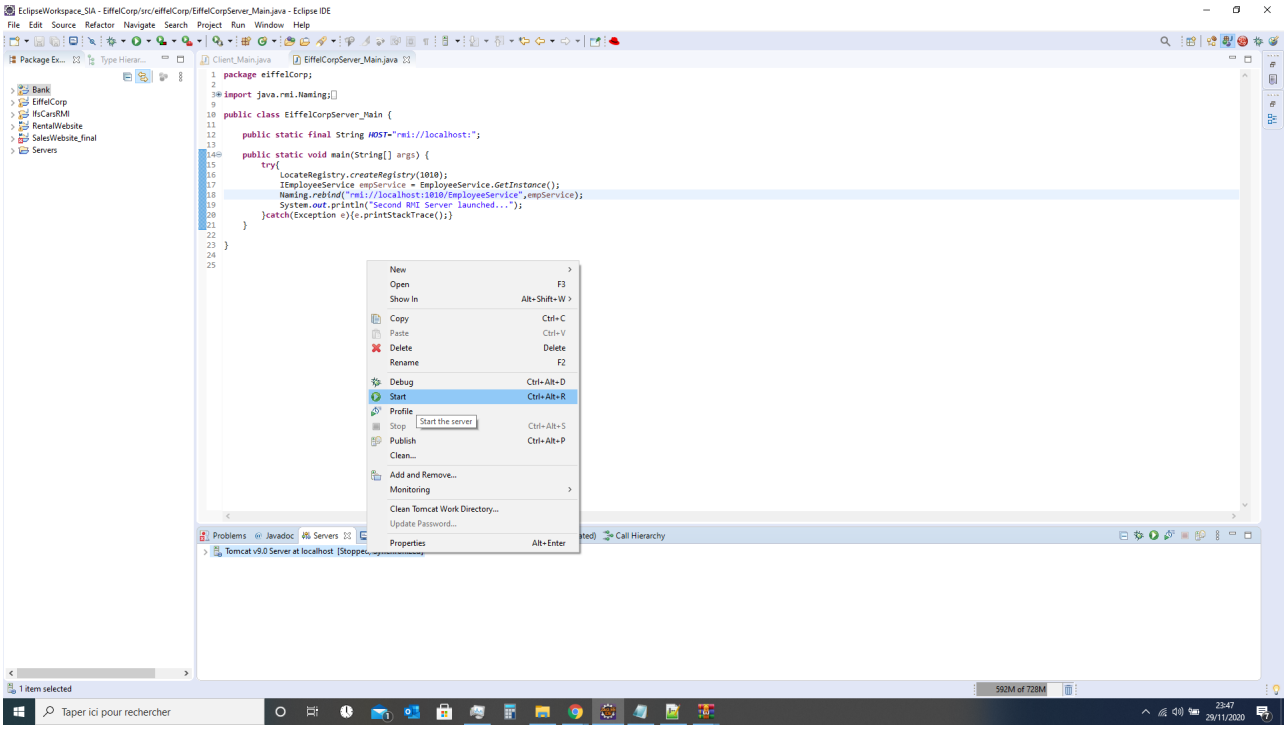

Figure 4.6: Step 4

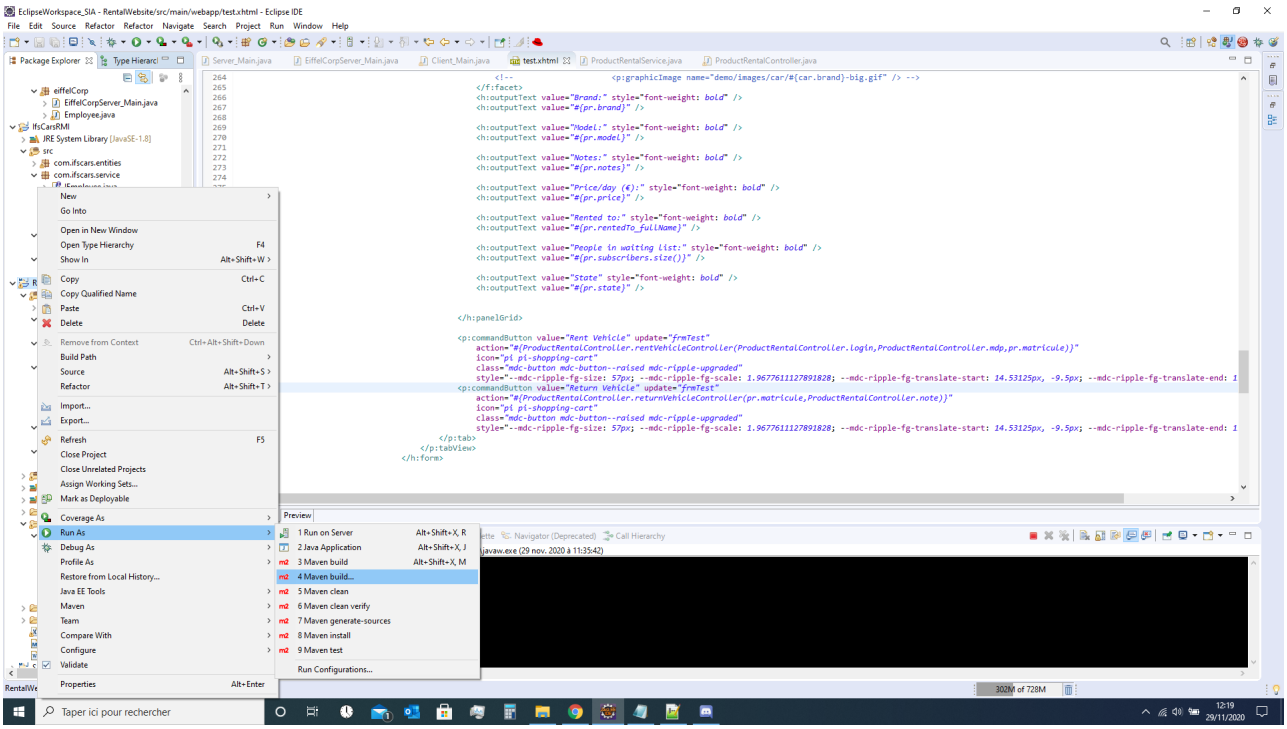

Figure 4.7: Step 5

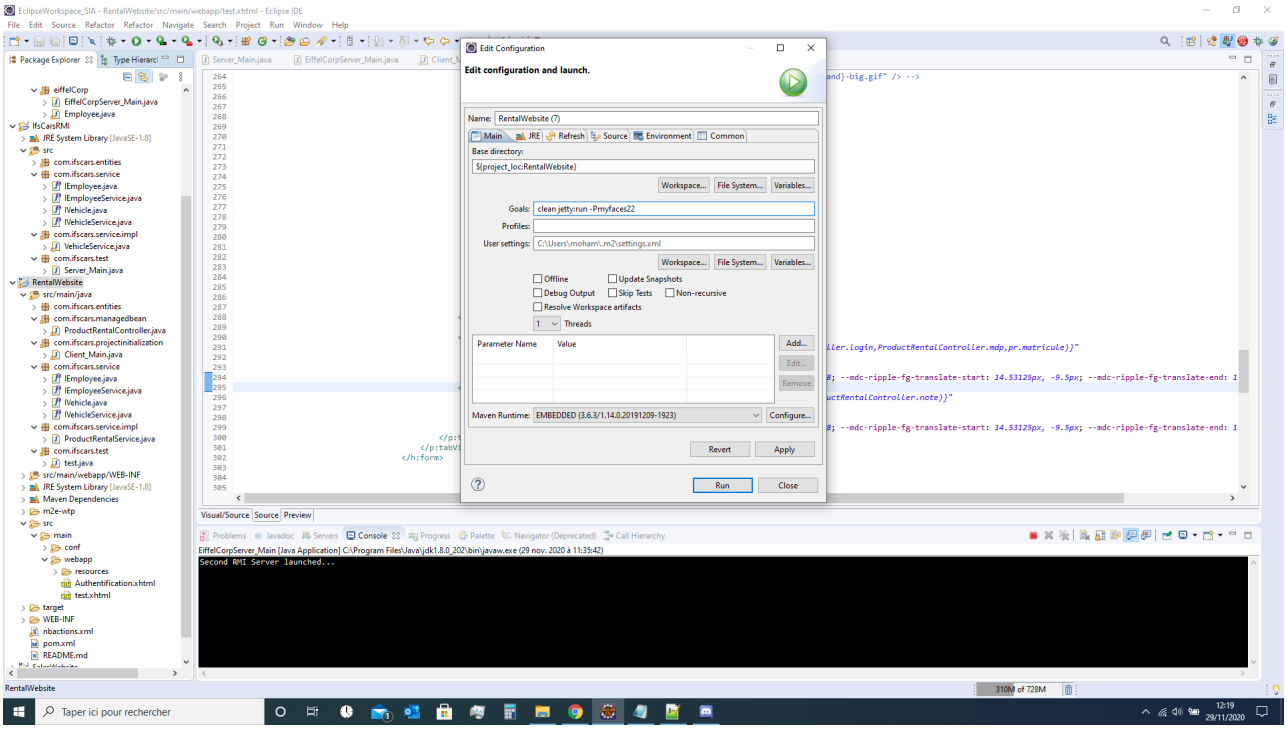

Figure 4.8: Step 5

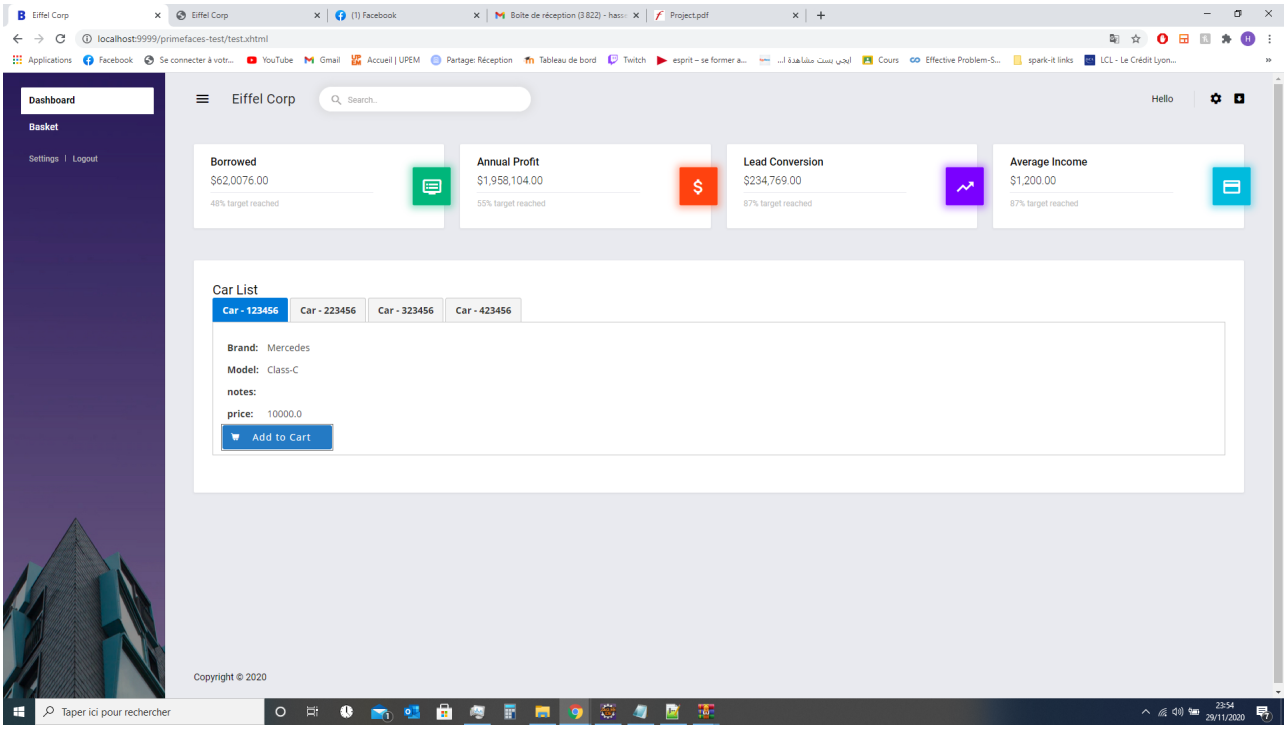

Figure 4.9: Add website

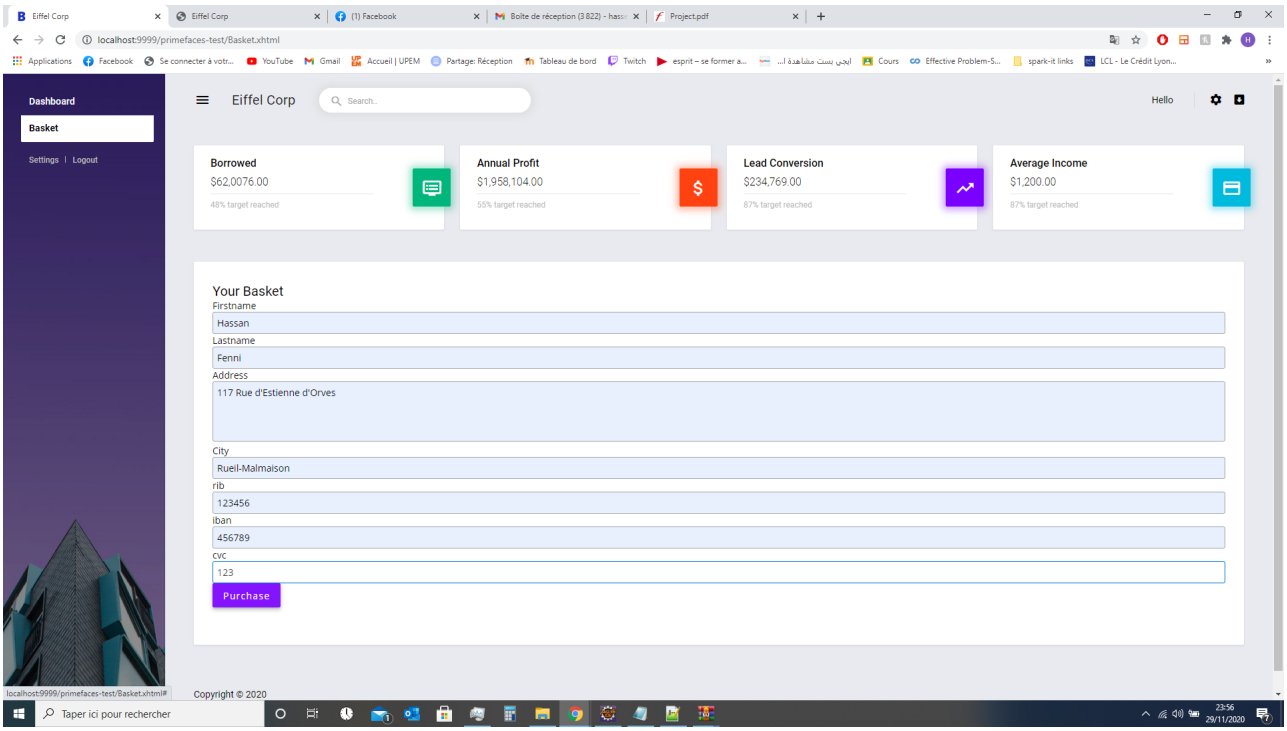

Figure 4.10: Purchase website

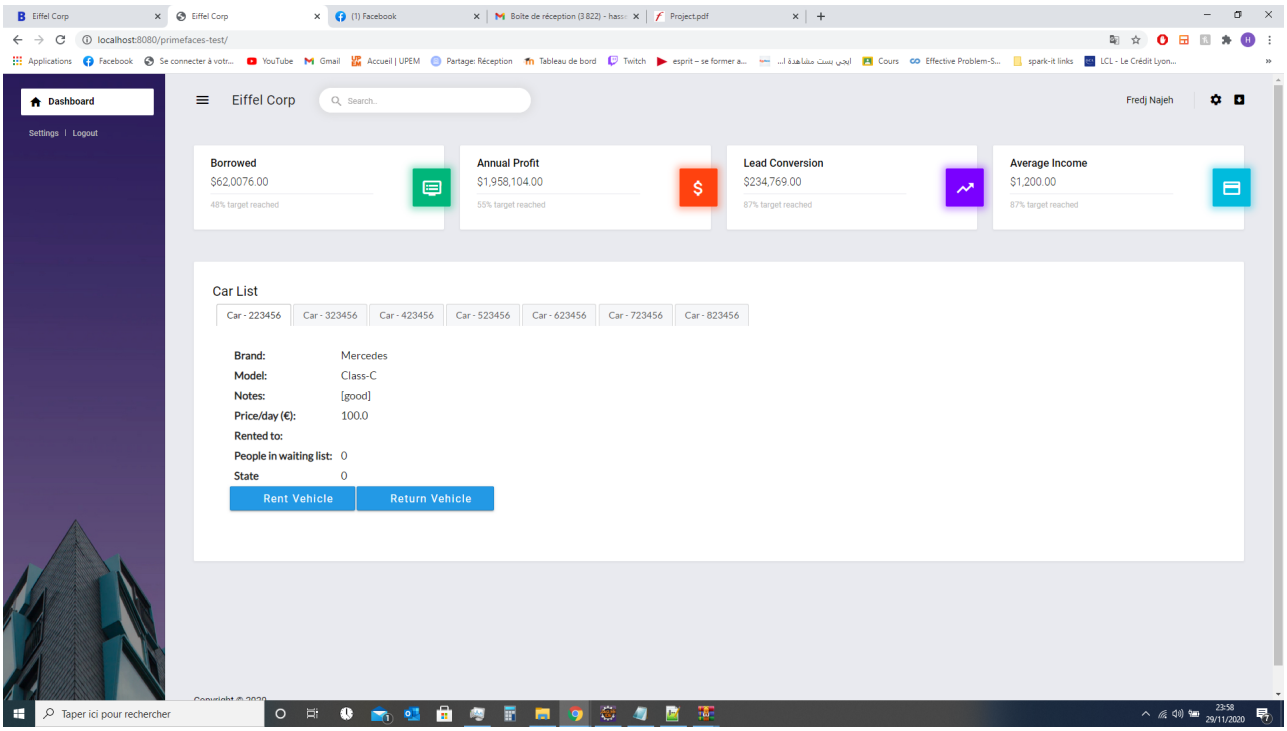

Figure 4.11: rental website

5 Conclusion

In order to realize our web services project,We have achieved our goal by creating our java application to please IEiffel Corp's employees and costumers.