Université Gustave Eiffel

WEB SERVICES PROJECT REPORT

SUPERVISED BY:

DR. ZARGAYOUNA MAHDI

ELABORATED BY:

ARFAOUI FAIROUZ
AYACHI HOSNI
MARNISSI SKANDER

M2 SIA

2020-2021

TABLE OF CONTENT

Introduction 3

I. Problem Definition: 4

II. Implementation and Architecture: 5

III. User Interface: 9

IV. Working Scenarios: 12

Conclusion 14

Introduction

Web services refer to a family of technologies that can universally standardize the communication of applications connecting systems, business partners and customers in a very cost-effective manner utilizing the Web.

Major software vendors such as Microsoft, IBM, Oracle and SUN are all embracing Web services standards and are releasing new products or tools that are Web services enabled, in addition to multiple E-Commerce platforms such as the leader of the field, Amazon.
Using them will ease the constraints of time, cost, and space for every operation needed. Therefore, applications implementing Web Services have a huge lead on those who don’t in terms of organization and optimisation.

Operations such as online search, catalogs and finally payment who are essential in mainly every E-Commerce business nowadays, can be done in a blink of an eye leaving developers without the concern of unnecessary programming and debugging.

Over the years, companies implementing and experimenting Web Services are showing promising results such as greater development productivity gains and easier and faster integration with their partners.

I. Problem Definition:

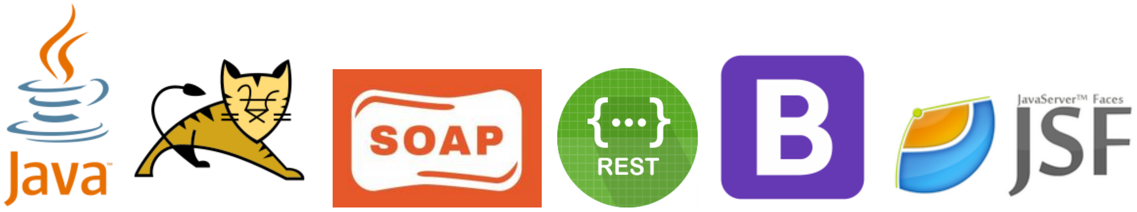
The company *EiffelCorp* has just acquired the company *IfCars*, a company that is specialized in car rental and by doing that, they aim to provide their employees with that service at a preferential price and to reach that objective, a platform will be needed.

That platform will revolve around managing car rental operations and organizing them and therefore, enhance *EiffelCorp*'s vehicle base and allow them to sell them to the outside world. It will consist in an E-commerce website based on ***JAVA*** and the use of ***Web Services*** that would:

- Allow Employees to rent cars for a specific duration of time.
- Allow Employees to rate and leave a note based on their appreciation after returning the car.
- Generate a Waiting List for every car that is requested for rent but already rented so when the car is available, the Employee will be notified and will be able to rent the car based on the "first come, first served" principle .
- Allow outside Customers to be able to access a catalog of previously rented cars.
- Show all car prices in Euros.
- Allow outside Customers to be able to buy previously rented cars.
- Accept all currencies for payment through conversion.

II. Implementation and Architecture:

To go through this project we used the following tools:



- **JAVA 8:**
  Java is an object-oriented programming language that produces software for multiple platforms. When a programmer writes a Java application, the compiled code (known as bytecode) runs on most operating systems. Java derives much of its syntax from the C and C++ programming languages.

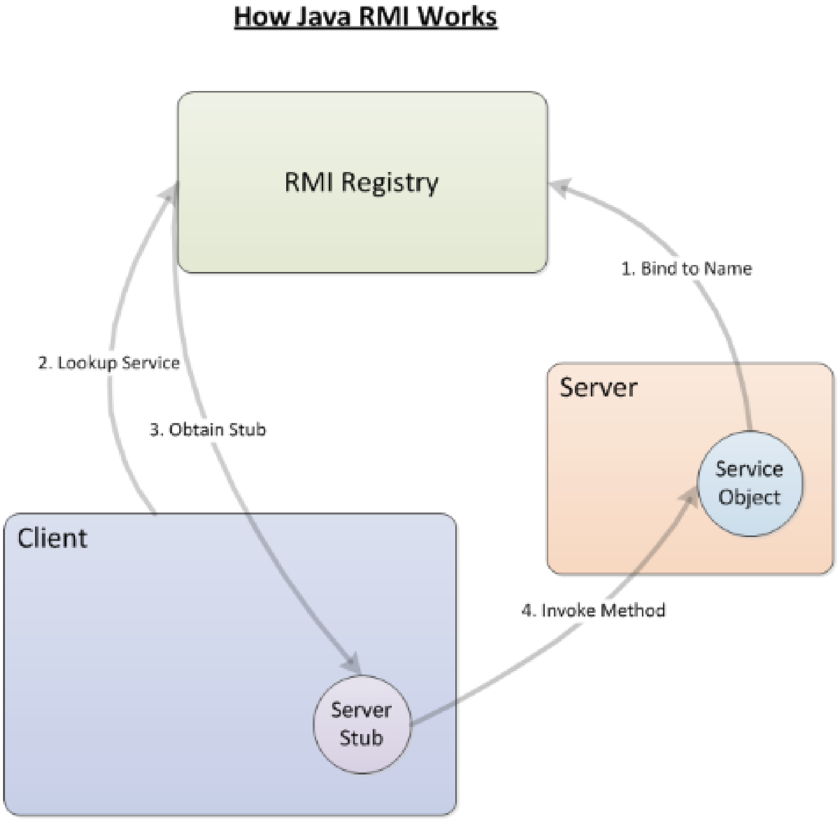
- **JSF:**
  Jakarta Server Faces (formerly JavaServer Faces) is a Java specification for building component-based user interfaces for web applications.

- **Tomcat Server 9.0:**
  Apache Tomcat allows the implementation of Java Servlets and JavaServer Pages (JSP) to promote an effective Java server environment.

- **Maven:**
  Maven is a software project management and comprehension tool primarily used with Java-based projects Maven helps manage builds, documentation, reporting, dependencies, software configuration management (SCM), releases and distribution.

- **RMI:**
  Java RMI is a package for writing and executing distributed Java programs by facilitating object method calls between different Java Virtual Machines (JVM) across a network.

  An RMI application consists of a server interface, a server implementation, a server skeleton and a client stub, and a client implementation. The server implementation creates remote objects that conform to the server interface. These objects are available for method invocation to clients.



- **Web Services:**
  Web services are based on distributed computing technology and provide a standard means of interoperating between different software applications, using XML protocols and formats. Web Services comply with several WWW standards, such as Web Services Definition Language (WSDL), Simple Object Access Protocol (SOAP) and Representational State Transfer (REST).

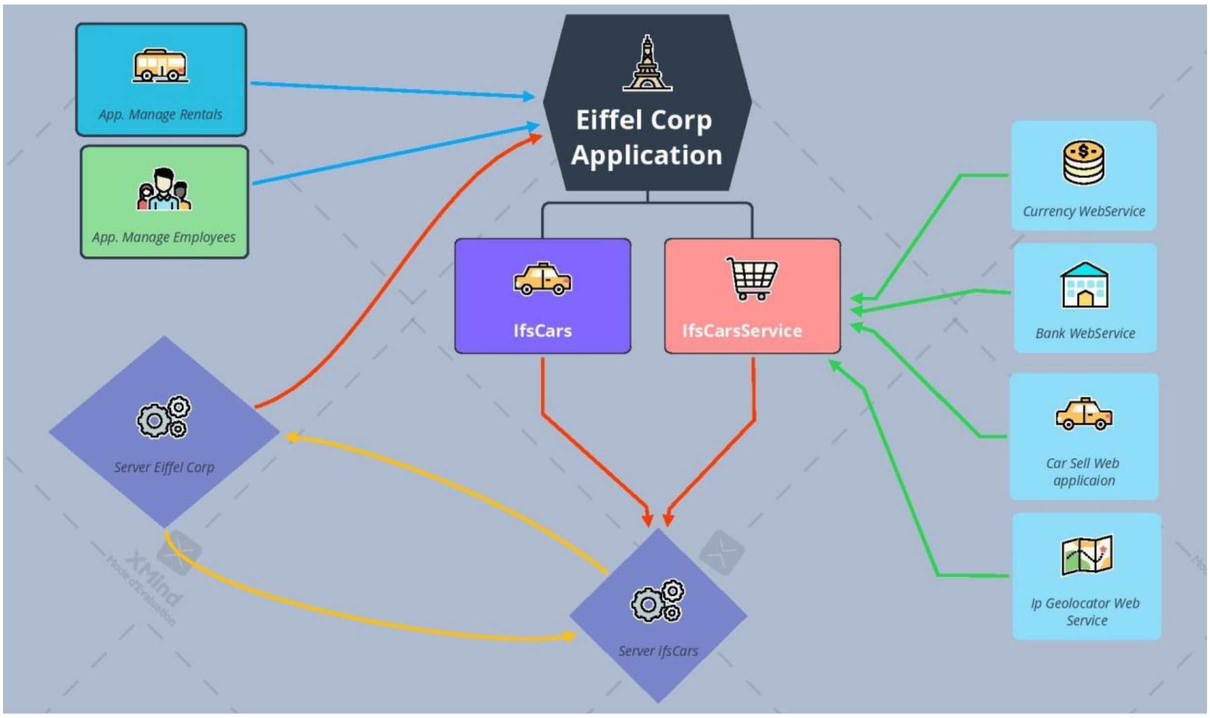
- **WSDL** (Web Service Description Language):
  WSDL is an XML-based language that describes Web services and their uses. A WSDL document is a concrete description of a Web Service and its elements. WSDL describes the abstract functionality of a service and provides a framework for describing the concrete details of a service description. This formal description is required in order for distributed systems and communication between software applications.

- **SOAP** (Simple Object Access Protocol):
  This standard defines the types and formats of XML messages that may be exchanged between peers in a decentralized, distributed environment. One of the main objectives of SOAP is to be a communication protocol that can be used by distinct applications developed using different programming languages, operating systems, and platforms.

- **REST** (Representational State Transfer)
  REST is a distributed system framework that uses Web protocols and technologies. The REST architecture involves client and server interactions built around the transfer of resources.

- **MVC** (Model - View - Controller)
  Model View Controller (MVC) is a design pattern for computer software. It can be considered an approach to distinguish between the data model, processing control and the user interface. It neatly separates the graphical interface displayed to the user from the code that manages the user actions.

- **Bootstrap**
  Bootstrap is a collection of reusable pieces of code written in HTML, CSS, and JavaScript. It is a free and open-source front end web development framework that allows users to create responsive websites and web applications.



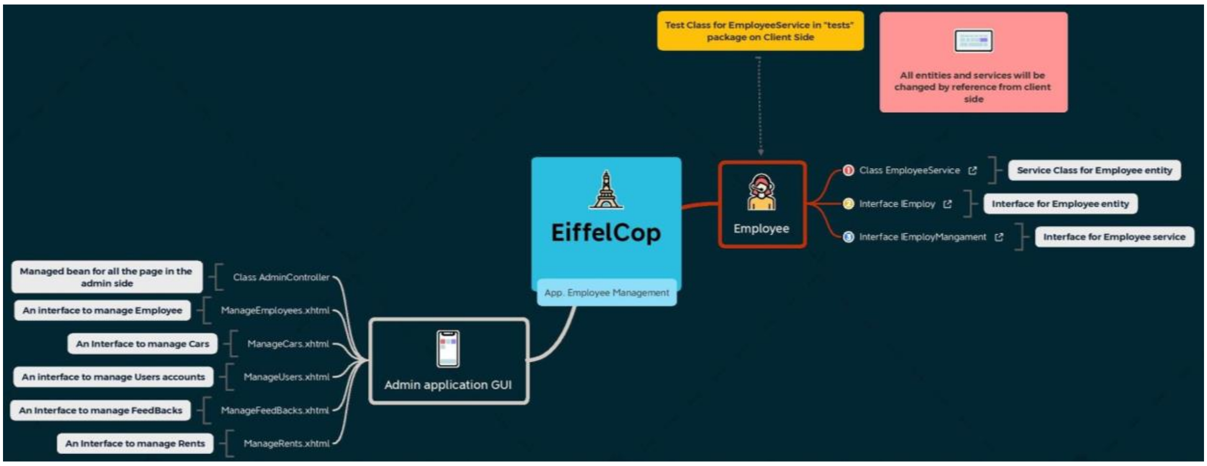
EiffelCorp project Conception

This project is divided into two parts linked together:

- In the first part, we focused on the implementation of RMI to ensure car rental operations for EiffelCorp's Employees.
- In the second part, we used Web Services in order to allow outside Customers to buy previously rented cars.

PART1 : Implementing JAVA RMI

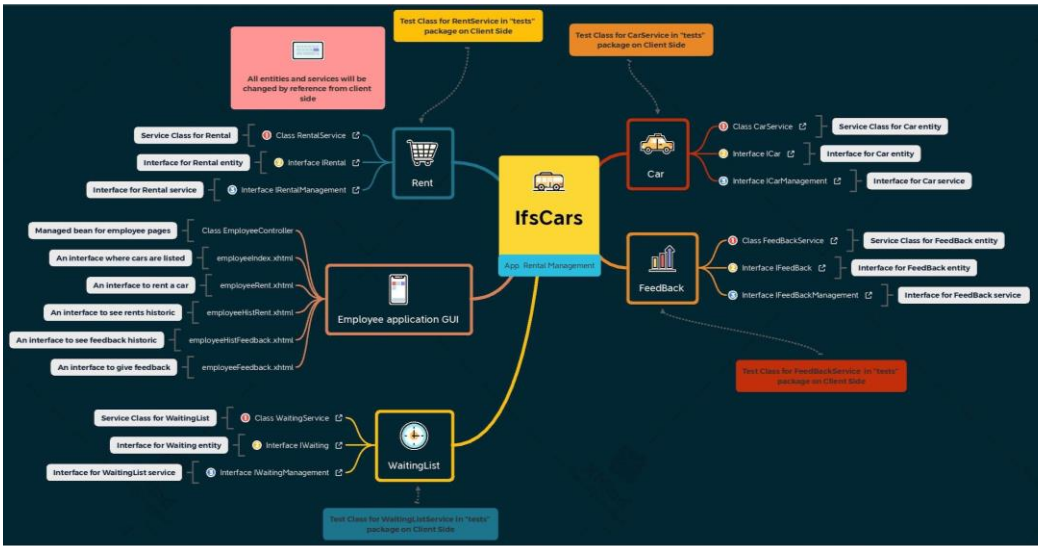
For the first part of this project, we implemented a distributed Java application based on RMI in order to manage Car rental operations such the renting system itself, as well as allowing Employees to leave a note on the Cars they returned. Therefore, this RMI-based application manages both the Cars database and the Employees database.


Test Class for EmployeeService in "tests" package on Client Side
All entities and services will be changed by reference from client side
EiffelCop
App. Employee Management
Employee
Class EmployeeService
Service Class for Employee entity
Interface IEmploy
Interface for Employee entity
Interface IEmployMangament
Interface for Employee service
Admin application GUI
Managed bean for all the page in the admin side
Class AdminController
An interface to manage Employee
ManageEmployees.xhtml
An Interface to manage Cars
ManageCars.xhtml
An interface to manage Users accounts
ManageUsers.xhtml
An Interface to manage FeedBacks
ManageFeedBacks.xhtml
An Interface to manage Rents
ManageRents.xhtml

EiffelCorp Application Conception

PART2 : Implementing Web Services

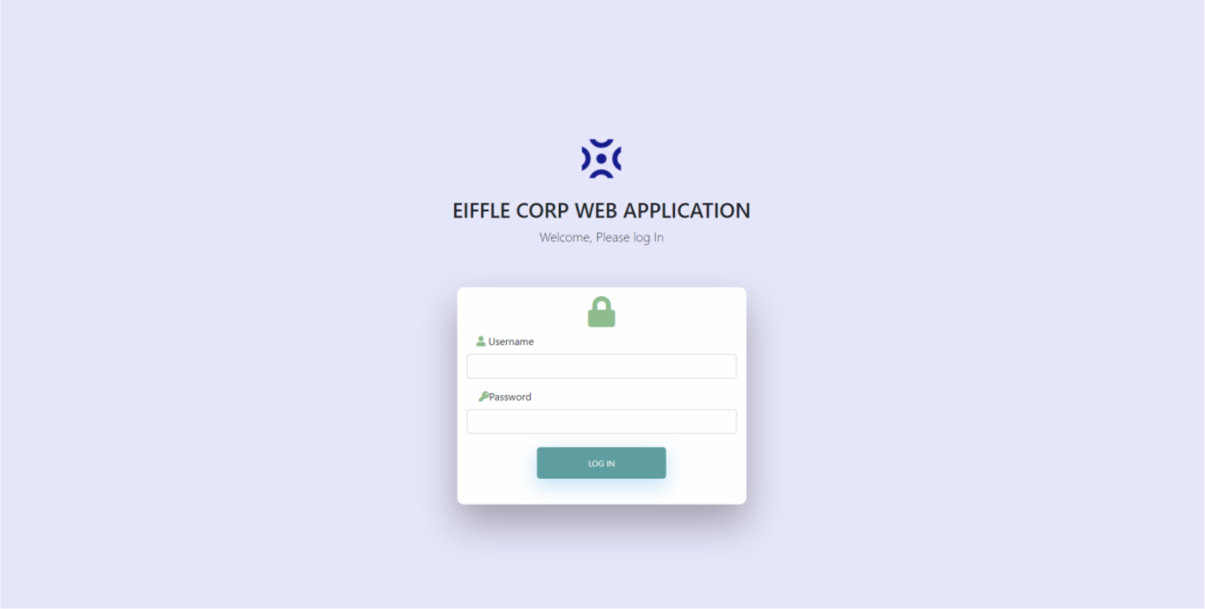
For the second part of this project, the aim was to make the application available to the outside world, and therefore, outside Customers, allowing them to buy Cars that have been rented previously. The use of Web Services revolves around consulting the catalog of available Cars, adding them into a basket and purchasing them, checking the availability of the funds of the buyer and if they’re able to complete the operation no matter what currency their bank account is on, using a real-time currency converter service.


All entities and services will be changed by reference from client side
Test Class for RentService in "tests" package on Client Side
Test Class for CarService in "tests" package on Client Side
IfsCars
App. Rental Management
Rent
Class RentalService
Service Class for Rental
Interface IRental
Interface for Rental entity
Interface IRentalManagement
Interface for Rental service
Car
Class CarService
Service Class for Car entity
Interface ICar
Interface for Car entity
Interface ICarManagement
Interface for Car service
FeedBack
Class FeedBackService
Service Class for FeedBack entity
Interface IFeedBack
Interface for FeedBack entity
Interface IFeedBackManagement
Interface for FeedBack service
Test Class for FeedBackService in "tests" package on Client Side
Employee application GUI
Managed bean for employee pages
Class EmployeeController
An interface where cars are listed
employeeIndex.xhtml
An interface to rent a car
employeeRent.xhtml
An interface to see rents historic
employeeHistRent.xhtml
An interface to see feedback historic
employeeHistFeedback.xhtml
An interface to give feedback
employeeFeedback.xhtml
WaitingList
Class WaitingService
Service Class for WaitingList
Interface IWaiting
Interface for Waiting entity
Interface IWaitingManagement
Interface for WaitingList service
Test Class for WaitingListService in "tests" package on Client Side

IfsCars Application Conception

III. User Interface:

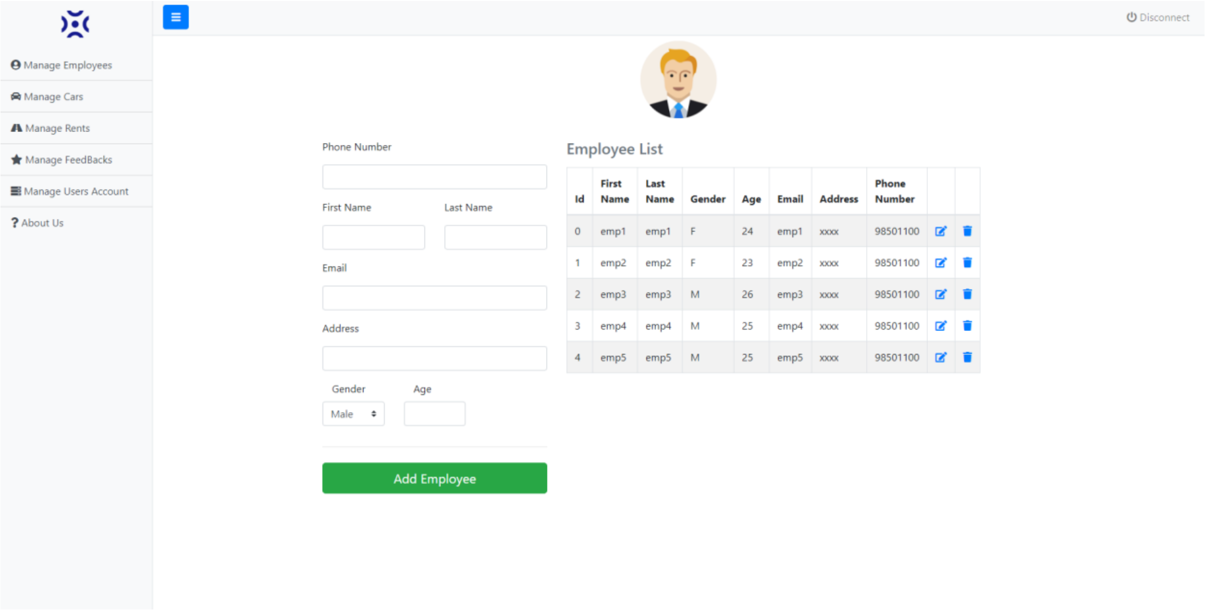
PART1: Implementing JAVA RMI



EIFFLE CORP WEB APPLICATION

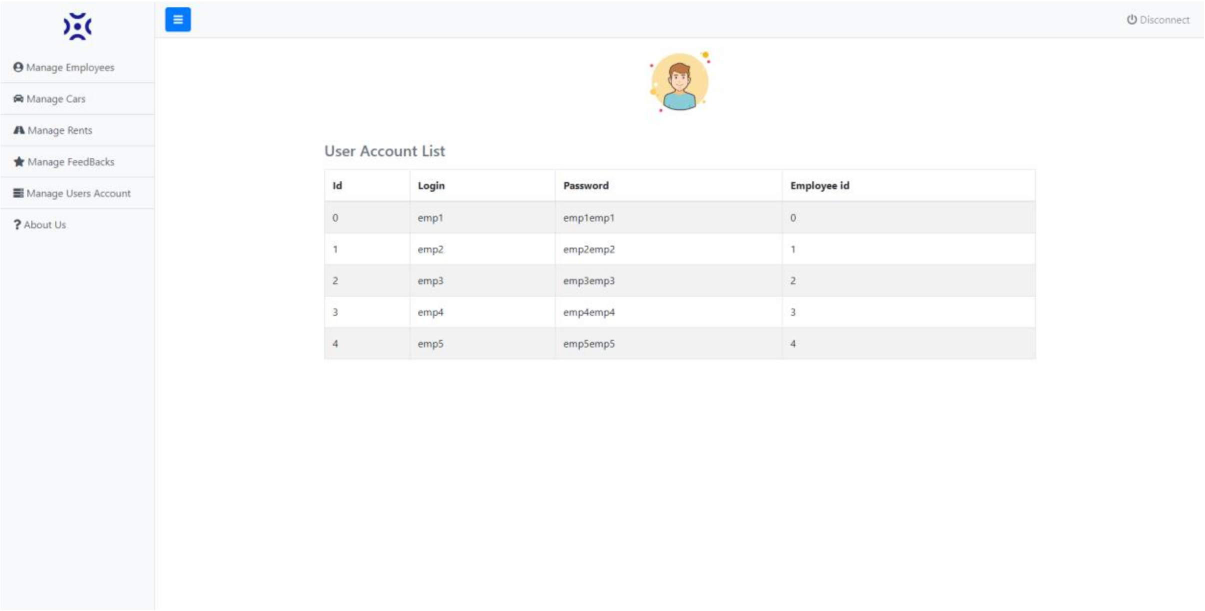
Welcome, Please log In

This is the login screen for the EiffelCorp application, which is a classic one, you have to enter your username and password and then you'll be redirected to a platform depending on whether you are an admin or an employee.



Employee List

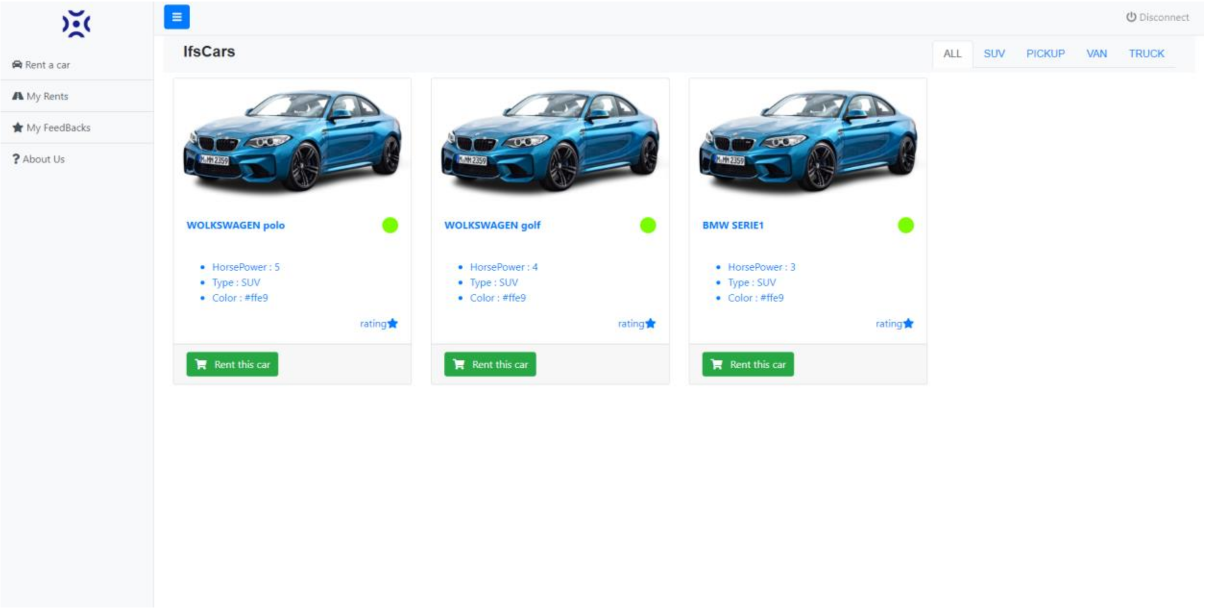
Id	First Name	Last Name	Gender	Age	Email	Address	Phone Number		
0	emp1	emp1	F	24	emp1	xxxx	98501100	✎	🗑
1	emp2	emp2	F	23	emp2	xxxx	98501100	✎	🗑
2	emp3	emp3	M	26	emp3	xxxx	98501100	✎	🗑
3	emp4	emp4	M	25	emp4	xxxx	98501100	✎	🗑
4	emp5	emp5	M	25	emp5	xxxx	98501100	✎	🗑



User Account List

Id	Login	Password	Employee id
0	emp1	emp1emp1	0
1	emp2	emp2emp2	1
2	emp3	emp3emp3	2
3	emp4	emp4emp4	3
4	emp5	emp5emp5	4

This is the Admin's interface where he can manage the employees, cars, rentals, feedbacks on rented cars and the employees accounts as you can see on the side bar. The admin can add an employee or a car, edit their coordinates or delete them from the database. He can edit return dates in the rent menu and he can see the feedback on the cars and employees accounts.



IfsCars

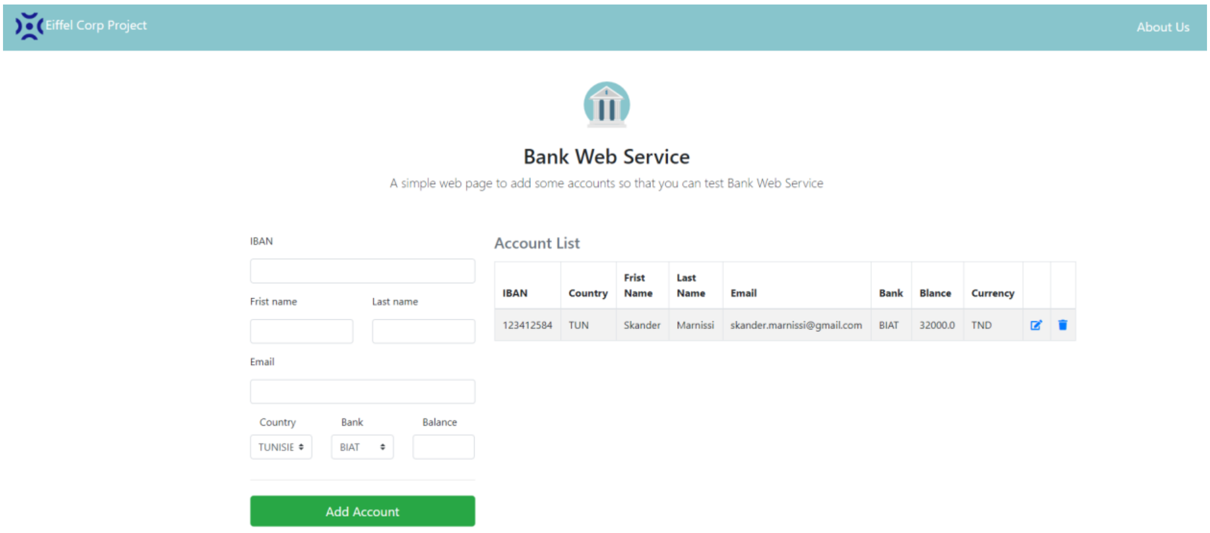
WOLKSWAGEN polo
- HorsePower : 5
- Type : SUV
- Color : #ffe9

WOLKSWAGEN golf
- HorsePower : 4
- Type : SUV
- Color : #ffe9

BMW SERIE1
- HorsePower : 3
- Type : SUV
- Color : #ffe9

This is the Employee's interface where he can choose between a catalog of cars to rent, he has the option to filter based on the type of the car he wants to rent (SUV, Pickup, VAN, Truck), he can also manage his rents and add feedbacks.

PART2: Implementing Web Services



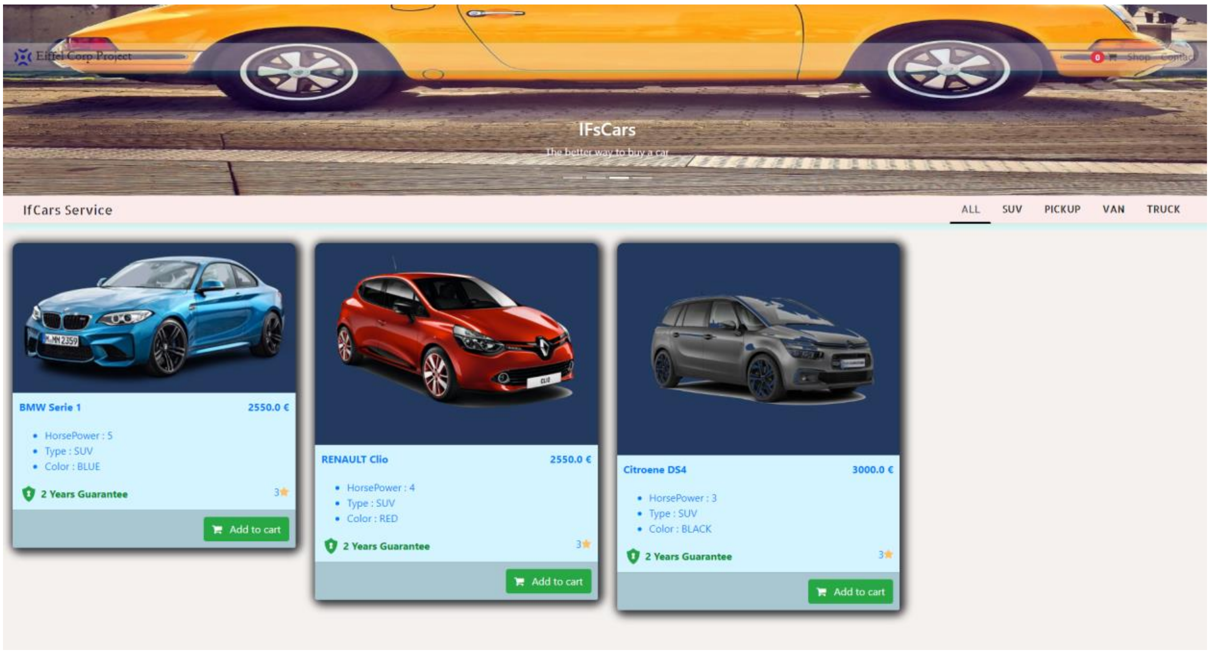
Bank Web Service

A simple web page to add some accounts so that you can test Bank Web Service

Account List

IBAN	Country	Frist Name	Last Name	Email	Bank	Blance	Currency		
123412584	TUN	Skander	Marnissi	skander.marnissi@gmail.com	BIAT	32000.0	TND		

This is the Bank Web Service we will be using for real-time currency conversion, it revolves around a database of bank accounts from different currencies(depending on the country of the owner of the account). Here, you can add an account and set its balance.



BMW Serie 1 2550.0 €

- HorsePower : 5
- Type : SUV
- Color : BLUE

2 Years Guarantee 3★

RENAULT Clio 2550.0 €

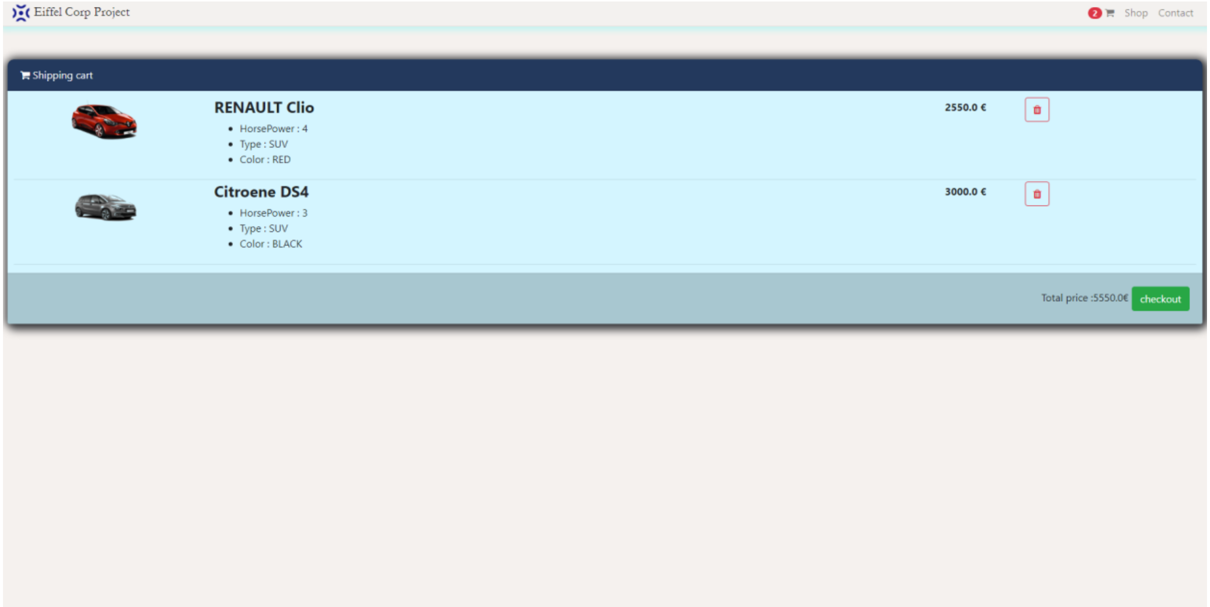
- HorsePower : 4
- Type : SUV
- Color : RED

2 Years Guarantee 3★

Citroene DS4 3000.0 €

- HorsePower : 3
- Type : SUV
- Color : BLACK

2 Years Guarantee 3★



RENAULT Clio
- HorsePower : 4
- Type : SUV
- Color : RED

2550.0 €

Citroene DS4
- HorsePower : 3
- Type : SUV
- Color : BLACK

3000.0 €

Total price :5550.0€

This is the IfCars Website where any outside Customer can go through a catalog of previously rented cars, check their ratings and feedbacks and choose to buy by clicking on the "Add to cart" button. He can then check his cart, edit it by removing a Car or confirming by checking out and fill the check out form.

IV. Working Scenarios:

PART1: Implementing JAVA RMI

- Login into the Admin account.
- Open the "Manage Employees" tab dd a number of employees.
- Open the "Manage Cars" tab and add a number of Cars.

- Login into one of the Employee's accounts.
- Rent a car.

- Login into another Employee's account.
- Rent the same car; Goes into the waiting list.

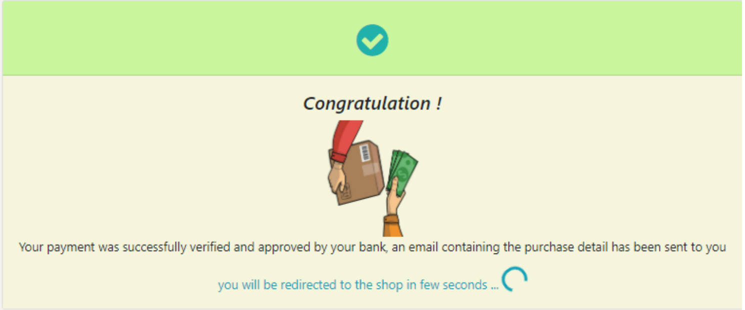
- Login into the Admin account and set the end date of first rent to today.

- Whenever the first Employee logs in, he'll find his rent has ended and he'll have to write a feedback.
- Whenever the second Employee logs in, he'll find his rent has started (State : In progress).

PART2: Implementing Web Services

- Add a Client’s bank account in the Bank Web Service.
- Go to the IfCars Website.
- Choose between a catalog of Cars while consulting feedbacks.
- Add the chosen Car to the basket and confirm the selection.
- Fill the checkout form with information on your bank account.

- When you submit, the currency from the bank account is converted in Euros.
  - If the converted amount > the price of the Car : the operation is a success.

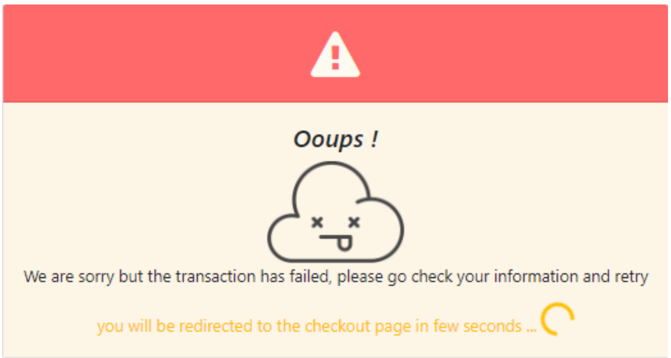


Congratulation !

Your payment was successfully verified and approved by your bank, an email containing the purchase detail has been sent to you

you will be redirected to the shop in few seconds ...

  - If the converted amount < the price of the Car : the operation is failed.



Ooups !

We are sorry but the transaction has failed, please go check your information and retry

you will be redirected to the checkout page in few seconds ...

Conclusion

RMI first and then Web Services were major actors in the accomplishment of this project, facilitating access, use and implementation of JAVA Objects and distributed applications on the Internet like the currency converting system we've encountered. This interoperability is a huge asset that plays an effective part in different operations in different fields as they are great at exposing software functionality to customers. In contrast to previous architectures, adopting Web Services lifts up the burden of a lot of significant resources, in both development and deployment even though the speed of its interoperability is seriously affected by the bandwidth. In fact, Web Services are still in the process of stabilizing the standards.