



Rapport

Projet Services Web

Étudiants :

Volkan ZULAL, Bilal BOUGUERRA, Zacharie SOUIOUED

Enseignant :

Mahdi ZARGAYOUNA

Cours :

Web Service

Table des matières

<i>Introduction</i>	4
Contexte et Objectifs.....	4
<i>Conception</i>	5
Les choix	5
Schéma de la conception	6
.....	6
<i>Difficultés rencontrées</i>	7
Lors de la conception	7
Lors du développement	7
<i>Utilisation</i>	8
Après avoir dé zippé et préparation de l'environnement	8
Utilisation.....	9
<i>Conclusion</i>	14

Introduction

Contexte et Objectifs

Durant ce projet nous amené à mettre en place une application distribué en JAVA fondée sur RMI pour une société nommé Eiffel Corp. vient d'acquérir la société IfsCars, spécialisée dans la location de véhicules. Elle souhaite mettre en place cette application distribué afin de louer des véhicules à ces employés et de les vendre à des clients extérieurs.

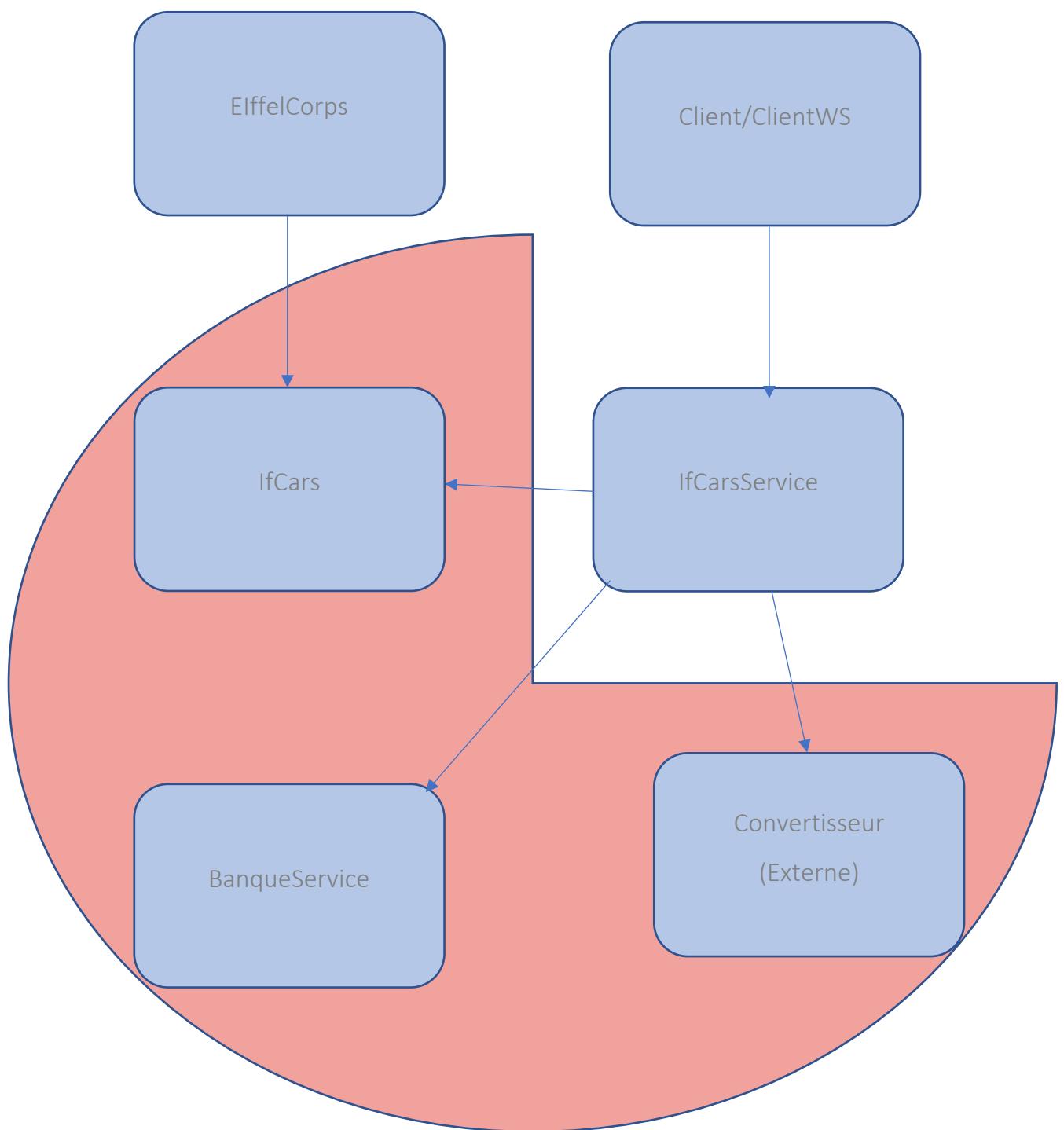
Conception

Les choix

Nous avons choisi de mettre en place la conception suivante :

- Tous d'abord nous avons implémenté une base qui permettait de stocker les voitures (IfCars) et on l'a rendu disponible via un serveur RMI qui se base sur une JVM.
- Nous avons implémenté la base des employés en RMI. L'employé se connecte au serveur des voitures (SellCarServeur) et permet à cette employée de louer des véhicules via le client de son RMI.
- Nous avons implémenté une base des Client en RMI qui est aussi disponible en Service Web via un serveur tomcat. Nous avons deux implémentations un Client en RMI et l'autre ClientWS en Web Service. Tous les deux ont les fonctionnalités ont pris la décision de faire ainsi pour faire une distinction entre les employés qui sont interne à l'entreprise et les clients qui sont externe à l'entreprise.
- Nous avons implémenté un Service Web BanqueService qui permet de gérer le compte en banque d'un client.
- Nous avons implémenté un Service Web IfCarsService pour qui rend la base de véhicules accessible aux clients et de plus fait appel au service web BanqueService et un service web externe de conversion de monnaie en temps réel. Ce service permet de vérifier si les clients ont les fonds suffisants pour s'offrir la voiture en question. De plus il met à jour le compte en banque du client et fait la conversion de monnaie vers l'euro (€) par rapport au pays du client.

Schéma de la conception



→ Désigne que le sert d'un autre (ex. IfCarsService utilise IfCars, BanqueService, et le Convertisseur)

■ Ce sont les services que Client/ClientWS utilisent indirectement grâce à IfCarsService

Difficultés rencontrées

Lors de la conception

Tous d'abord les difficultés de conception son arrivé très tôt lors de la conception. Sur tout au niveau du web service IfCarsService on ne pensez pas qu'un service pouvais implémenter d'autres services web et des services RMI en même temps.

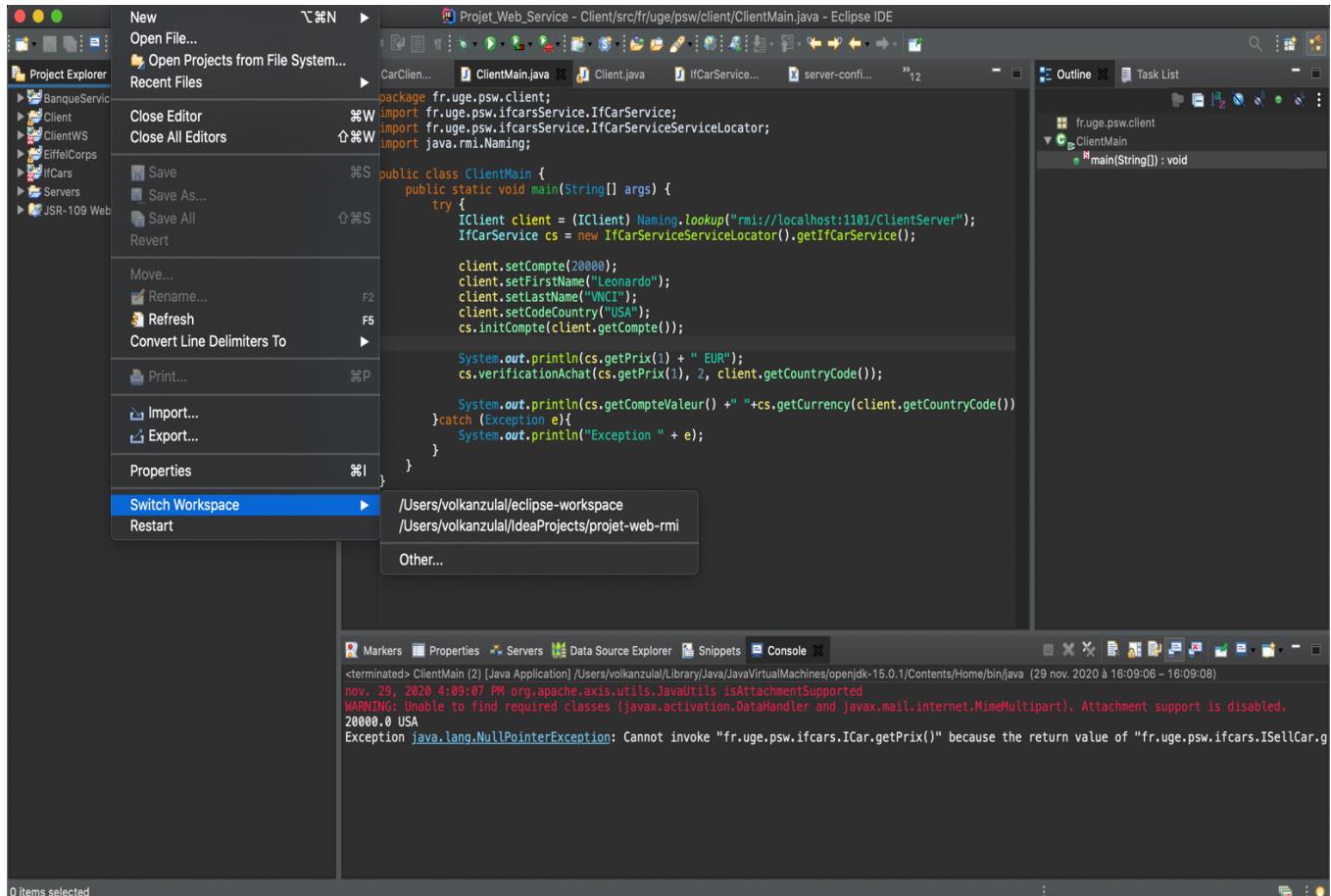
Lors du développement

- Lors du développement du web service IfCarsService, on avait du mal à générer un WSDL correcte du fait qu'elle utilise d'autre services.
- Coexistence de deux RMI nous eux des problèmes à utiliser un serveur RMI qui ne se trouvait pas dans le même projet où l'on avait besoin par exemple les employées nous avons un serveur RMI et pour les véhicules aussi du coup pour que ça marche nous avions mis tous dans un project Eclipse et on les distinguait grâce aux noms de packages au départ. Puis nous avons réussi à les séparer.
- Utilisation d'un web service extérieur via WSDL une fois l'avoir importé nous avions eu des problèmes a utilisé les fonctionnalités de ce client nous nous sommes déboggé grâce au documentation en ligne.
[\(http://fx.currencysystem.com/webservices/CurrencyServer5.asmx?wsdl\)](http://fx.currencysystem.com/webservices/CurrencyServer5.asmx?wsdl)

Utilisation

Après avoir dé zippé et préparation de l'environnement

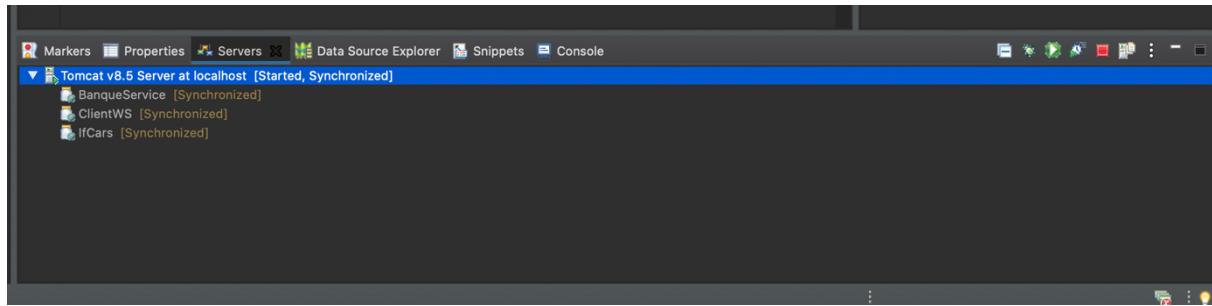
- Il y a génération d'un dossier « Projet_Web_Service » puis sur eclipse vous utiliser le chemin de ce dossier comme workspace.



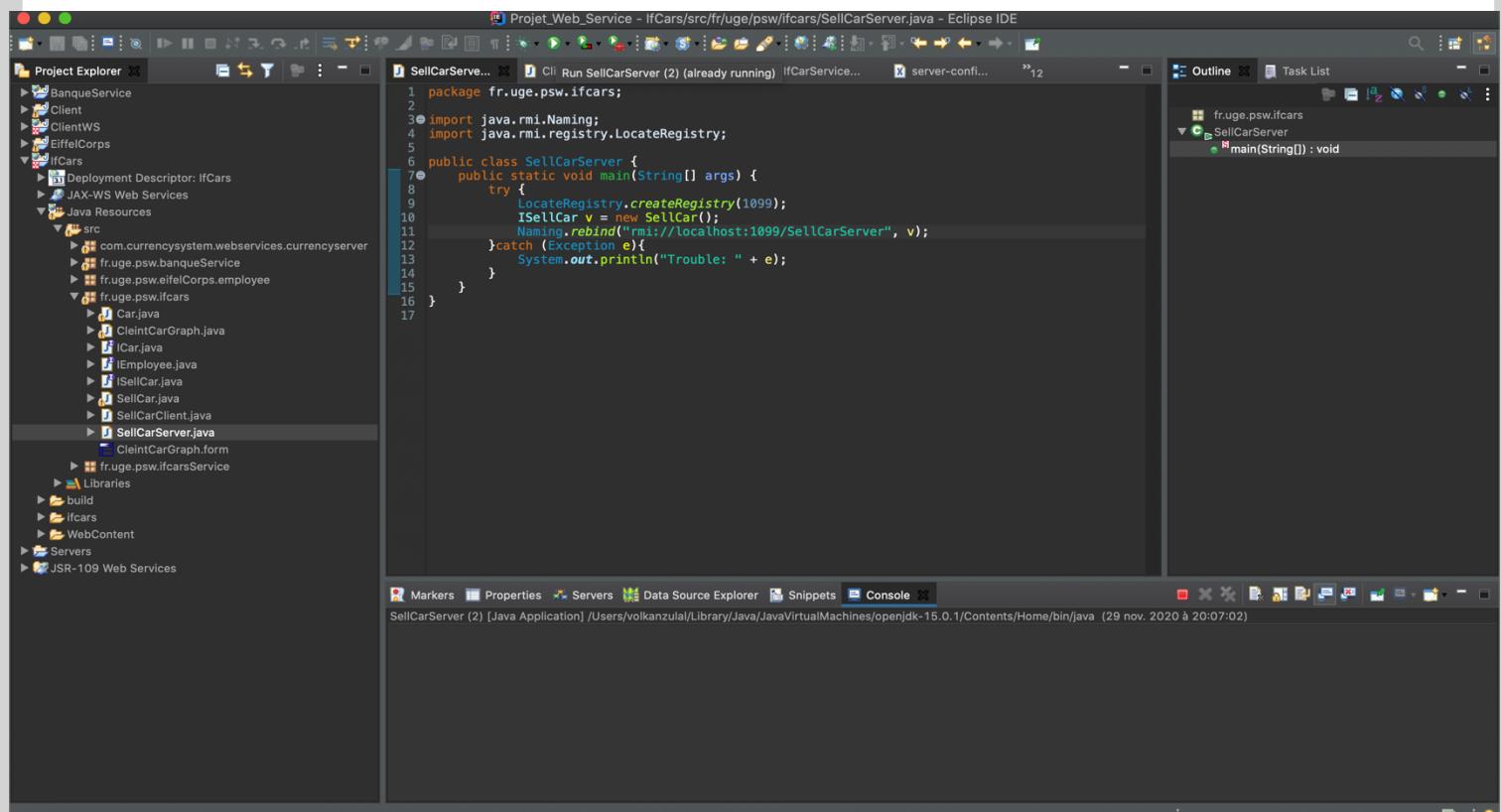
- Puis on met place un serveur Tomcat(7 ou 8) comme expliquer dans les slides du lien suivant : <http://mahdi.zargayouna.free.fr/SSIOLOG/Cours2-SSIOLOG.pdf> . Si vous ne possédé pas tomcat télécharger sur le lien suivant : <https://tomcat.apache.org/download-80.cgi>

Utilisation

Une fois avoir fait un run sur les dynamic web project IfCars, ClientWS, BanqueService et avoir lancée votre serveur tomcat vous aurez les noms des services web en question qui utiliserons votre serveur tomcat.



On lance le serveur de la base des véhicules IfCars on voit en bas a gauche que le serveur tourne bien.



Puis on lance le client de ce serveur pour remplir la base de véhicules et on voit la base se remplir.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with packages like BanqueService, Client, ClientWS, EiffelCorps, IfCars, and Java Resources.
- Code Editor:** Displays the `SellCarClient.java` file containing Java RMI code to interact with a car server.
- Console:** Shows the output of the application execution, displaying information about cars added to the database.

```

Projet_Web_Service - IfCars/src/fr/uge/psw/ifcars/SellCarClient.java - Eclipse IDE

package fr.uge.psw.ifcars;
import java.rmi.Naming;
import java.rmi.RemoteException;

public class SellCarClient {
    public static void main(String[] args) {
        try {
            ISellCar v = (ISellCar) Naming.lookup("SellCarServer");
            v.add("Peugeot", "206", 1, 1400, 0);
            v.add("Renault", "Clio", 2, 2000, 1);
            v.add("Mercedes", "AMG", 3, 2000, 1);
            v.add("Volvo", "S60", 4, 8000, 1);
            v.add("Mercedes", "Class C", 5, 20000, 1);
            v.add("Volvo", "C90", 6, 88000, 1);
            //v.setNote(9,77);
            //List<ICar> l = v.listModel("Clio");
            v.listCars().forEach(e-> {
                try {
                    System.out.println(e.getInformation());
                } catch (RemoteException remoteException) {
                    remoteException.printStackTrace();
                }
            });
        } catch (Exception e){
            System.out.println("Exception " + e);
        }
    }
}

```

Console Output:

```

<terminated> SellCarClient (2) [Java Application] /Users/volkanzula/Library/Java/JavaVirtualMachines/openjdk-15.0.1/Contents/Home/bin/java (29 nov. 2020 à 20:12:30 ~ 20:12:32)
Nb Location 1
Marque:Mercedes
Model:Class C
Prix:20000.0
Note : -1
Etat: CORRECT
Loué: false

ID:6
Nb Location 1
Marque:Volvo
Model:C90
Prix:88000.0
Note : -1
Etat: CORRECT
Loué: false

```

Puis on lance de la même manière le serveur des employés.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with packages like BanqueService, Client, ClientWS, EiffelCorps, IfCars, and Java Resources.
- Code Editor:** Displays the `EmployeeServeur.java` file containing Java RMI code to interact with an employee server.
- Console:** Shows the output of the application execution, indicating the creation of a registry and binding the service.

```

Projet_Web_Service - EiffelCorps/src/fr/uge/psw/eiffelCorps/employee/EmployeeServeur.java - Eclipse IDE

package fr.uge.psw.eiffelCorps.employee;
import java.rmi.Naming;

public class EmployeeServeur {
    public static void main(String[] args) {
        try {
            LocateRegistry.createRegistry(1100);
            IEmployee e = new Employee();
            Naming.rebind("rmi://localhost:1100/EmployeeServeur", e);
        } catch (Exception e){
            System.out.println("Trouble: " + e);
        }
    }
}

```

Console Output:

```

EmployeeServeur [Java Application] /Users/volkanzula/Library/Java/JavaVirtualMachines/openjdk-15.0.1/Contents/Home/bin/java (29 nov. 2020 à 20:19:07)

```

On lance le main() du EmployeeClient afin de louer une voiture présent dans la base. On voit dans la console que l'on loue bien la voiture d'id 4 qui est une « Volvo S60 ».

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Displays the project structure for "Projet_Web_Service". It includes packages like BanqueService, Client, ClientWS, EiffelCorps, IfCars, and Java Resources. Under Java Resources, there are sub-folders for src, build, Ifcars, WebContent, and Servers.
- EmployeeClient.java Content:**

```

10 public class EmployeeClient {
11     public static void main(String[] args) {
12         try {
13             IEmployee employee = (IEmployee) Naming.lookup("rmi://localhost:1100/EmployeeServer");
14             ISellCar cars = (ISellCar) Naming.lookup("rmi://localhost:1099/SellCarServer");
15             employee.setFirstName("John");
16             employee.setLastName("Doe");
17             employee.setId(0);
18
19             System.out.println(employee.getFirstName() + " " + employee.getLastName());
20             System.out.println("ID: " + employee.getId());
21
22             cars.getCar(4).rentCar(employee);
23
24             System.out.println(cars == null);
25
26             // cars.getCar(1).renduCar("RAS", 8);
27
28             cars.listCars().forEach(c -> {
29                 try {
30                     System.out.println(c.getInformation());
31                 } catch (RemoteException remoteException) {
32                     remoteException.printStackTrace();
33                 }
34             });
35         } catch (Exception e) {
36             System.out.println("Exception " + e);
37         }
38     }
39 }

```
- Console Output:**

```

<terminated> EmployeeClient [Java Application] /Users/volkanzula/Library/Java/JavaVirtualMachines/openjdk-15.0.1/Contents/Home/bin/java (29 nov. 2020 à 20:22:45 – 20:22:46)
Note : -1
Etat: CORRECT
Loué: false

ID:4
Nb Location 2
Marque:Volvo
Model:S60
Prix:8000.0
Note : -1
Etat: CORRECT
Loué: true

ID:5
Nb Location 1
Marque:Mercedes

```

On lance lance le serveur du Client ainsi ClientServer.java.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Displays the project structure for "Projet_Web_Service". It includes packages like BanqueService, Client, ClientWS, EiffelCorps, IfCars, and Java Resources. Under Java Resources, there are sub-folders for src, build, Ifcars, WebContent, and Servers.
- ClientServer.java Content:**

```

1 package fr.uge.psw.client;
2
3 import java.rmi.Naming;
4
5 public class ClientServer {
6     public static void main(String[] args) {
7         try {
8             LocateRegistry.createRegistry(1101);
9             IClient c = new Client();
10            Naming.rebind("rmi://localhost:1101/ClientServer", c);
11        } catch (Exception e) {
12            System.out.println("Trouble: " + e);
13        }
14    }
15 }

```
- Console Output:**

```

ClientServer (1) [Java Application] /Users/volkanzula/Library/Java/JavaVirtualMachines/openjdk-15.0.1/Contents/Home/bin/java (29 nov. 2020 à 20:26:10)

```

Puis on achète une voiture ainsi en lançant le client du ClientMain.java .

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with packages like BanqueService, Client, EiffelCorps, and IfCars.
- Code Editor:** Displays the `ClientMain.java` file containing Java code for creating a client and performing a purchase.
- Console Output:** Shows the terminal output of the application running. It includes the following text:


```
<terminated> ClientMain (1) [Java Application] /Users/volkanzulal/Library/Java/JavaVirtualMachines/openjdk-15.0.1/Contents/Home/bin/java (29 nov. 2020 à 20:31:46 – 20:31:52)
nov. 29, 2020 8:31:47 PM org.apache.axis.utils.JavaUtils isAttachmentSupported
WARNING: Unable to find required classes (javax.activation.DataHandler and javax.mail.internet.MimeMultipart). Attachment support is disabled.
2000.0 EUR
17615.6 USD
```

On voit 2000 EUR qui est le prix de la voiture et le client posséde 20000 USD et le 17615.6 USD désigne l'argent qu'il reste dans le compte en banque du client après l'achat.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with packages like JRE System Library, ClientWS, EiffelCorps, and IfCars.
- Code Editor:** Displays the `SellCarClient.java` file containing Java code for interacting with the SellCarServer.
- Console Output:** Shows the terminal output of the application running. It includes the following text:


```
<terminated> SellCarClient (2) [Java Application] /Users/volkanzulal/Library/Java/JavaVirtualMachines/openjdk-15.0.1/Contents/Home/bin/java (29 nov. 2020 à 20:38:24 – 20:38:25)
ID:1
Nb Location 0
Marque:Peugeot
Model:206
Prix:1400.0
Note : -1
Etat: CORRECT
Loué: false

ID:3
Nb Location 1
Marque:Mercedes
Model:AMG
Prix:2000.0
Note : -1
Etat: CORRECT
```

On voit que la voiture d'id 2 qui été une Renault Clio de 2000 € a été supprimé de la base après l'achat effectué par le client.

Ont utilisé maintenant le ClientWS pour effectuer un achat de la manière que le Client RMI mais maintenant on pas besoin de lancer un Serveur vu que celle-ci tourne sur le tomcat.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with packages like `fr.uge.psw.client`, `fr.uge.psw.eiffelCorps`, and `IfCars`.
- Code Editor:** Displays the `ClientMain.java` file containing Java code for creating a client service locator and interacting with a car service.
- Outline View:** Shows the class `ClientMain` and its method `main(String[] args)`.
- Console Output:** Shows the terminal output of the application's execution, including the printed values of the car's price and value.

```
Projet Web Service - Client/src/fr/uge/psw/clientws/ClientMain.java - Eclipse IDE

1 package fr.uge.psw.clientws;
2
3 import fr.uge.psw.ifcarsService.IfCarService;
4
5
6 public class ClientMain {
7     public static void main(String[] args) {
8         try {
9             Client client = (Client) new ClientServiceLocator().getClient();
10            ((ClientSoapBindingStub) client).setMaintainSession(true);
11            IfCarService cs = new IfCarServiceServiceLocator().getIfCarService();
12            client.setCompte(20000);
13            client.setFirstName("Leonardo");
14            client.setLastName("NCI");
15            client.setCodeCountry("USA");
16            System.out.println(client.getCompte() + " " + client.getCountryCode());
17            cs.initCompte(client.getCompte());
18
19            System.out.println(cs.getPrix(3) + " EUR");
20            cs.verificationAchat(cs.getPrix(3), 3, client.getCountryCode());
21
22            System.out.println(cs.getCompteValeur() + " " + cs.getCurrency(client.getCountryCode()));
23        } catch (Exception e) {
24            System.out.println("Exception " + e);
25        }
26    }
27
28
29 }
```

```
<terminated> ClientMain (2) [Java Application] /Users/volkanzulai/Library/Java/JavaVirtualMachines/openjdk-15.0.1/Contents/Home/bin/java (29 nov. 2020 à 20:41:34 – 20:41:37)
nov. 29, 2020 8:41:35 PM org.apache.axis.utils.Javaultils isAttachmentSupported
WARNING: Unable to find required classes (javax.activation.DataHandler and javax.mail.internet.MimeMultipart). Attachment support is disabled.
20000.0 USA
2000.0 EUR
17615.6 USD
```

Conclusion

Pour conclure ce projet nous étions très bénéfique nous avons travaillé sur une application distribuée qui nous a permis de perfectionner notre maîtrise du langage JAVA. De plus ça nous a permis de faire interagir deux différentes JVM et d'utilisé et implémenté des serveurs sous JAVA via Eclipse.