

CONCESIONARIO



PROYECTO CRUD

Curso: 2ºDAW

Módulo: Desarrollo Web en Entorno Servidor

Nombre de los Integrantes:

- Pilar Guzman Guzman
- Javier Villarreal Hinojo
- Vicente Villarreal Hinojo

- **Planteamiento**

En cuanto a los requerimientos, se trata de un proyecto donde se combinan lenguajes tanto **Java como JSP , Html, Css, comandos de Docker, etc.** Debe tratar de un diseño concreto y consta de una página de Login a través de la cuál le requerirá al usuario un Login y una Contraseña concreta, dicha página de Login estará ligada a un archivo JSP de lógica que accederá a nuestra base de datos levantada previamente y comprobará que dicho usuario se encuentra en nuestro sistema, si el resultado es positivo se redirigirá a la página concreta, en caso de ser negativo se redirigirá al usuario a una página de error previamente creada.

En segundo lugar y situándonos en el caso de que nuestro Login haya sido superado exitosamente se nos mostrará una tabla que accede a la base de datos mostrándonos todas nuestras marcas.

Cada marca que se mostrará tendrá la funcionalidad de poder borrar, actualizar y ver la lista de coches asociados a esa marca. Además se podrá añadir una nueva marca siempre y cuando no esté repetida.

Como tercer punto, si accedemos a la lista de coches de una marca, tendremos varias funcionalidades como añadir coche, borrar o actualizar coche, haciendo así que nuestro proyecto cumpla la funcionalidad de **CRUD**.

Estos procedimientos constan de redirección a formularios Html que recogerán los datos que precisen para poderlos llevar a cabo. Constará en cuanto a la parte lógica de métodos creados en lenguaje java, que ayudados de **Hibernate** accederán a nuestra base de datos y cumplirán dicha funciones.

En cuanto al **Build Path** del proyecto se llevará a cabo en **Java versión 11**. Nuestro servidor será **TomCat 9** y consta de la versión **Dynamic Web Project 4.0**.

También será necesario **mavenizar** el proyecto y añadirle al archivo pom.xml las dependencias de **Hibernate** y conexión a base de datos.

Java Build Path

Source Projects Libraries Order and Export Module Dependencies

JARs and class folders on the build path:

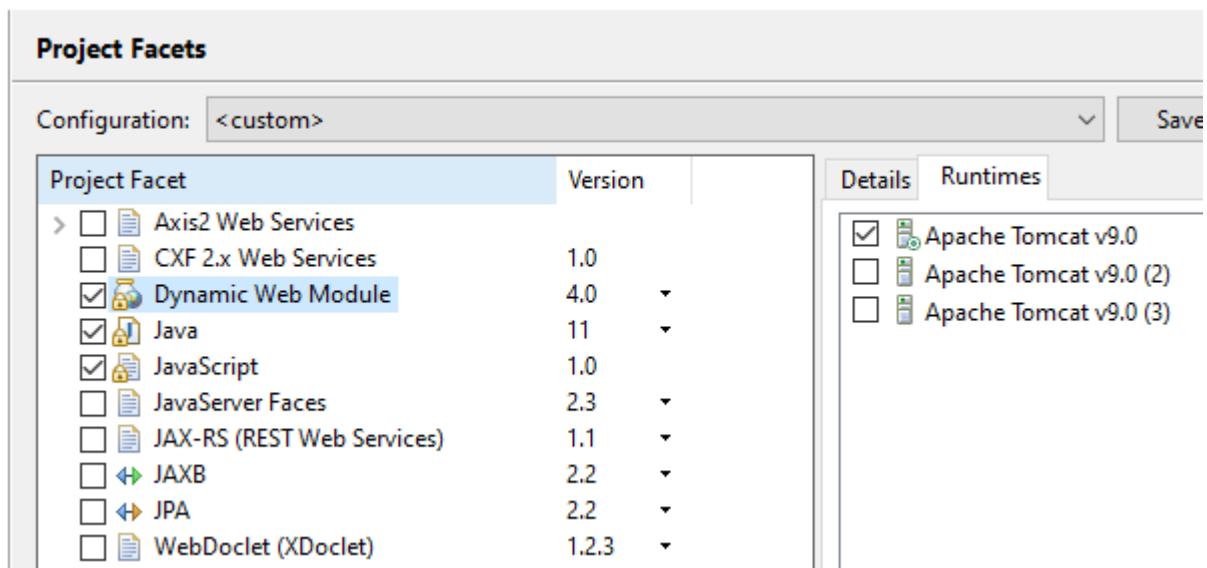
- ✓ Modulepath
 - > JRE System Library [JavaSE-11]
- ✓ Classpath
 - > Maven Dependencies
 - > Server Runtime [apache-tomcat-9.0.62]

Fichero de Hibernate:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
3     "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
4 <hibernate-configuration>
5   <session-factory name="">
6     <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
7     <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/concesionario?useSSL=false</property>
8     <property name="hibernate.connection.username">toor</property>
9     <property name="hibernate.connection.password">toor</property>
10    <property name="hibernate.dialect">org.hibernate.dialect.MySQLInnoDBDialect</property>
11    <property name="hibernate.show_sql">true</property>
12    <property name="hibernate.format_sql">true</property>
13    <mapping class="com.jacaranda.Car"/>
14    <mapping class="com.jacaranda.Brand"/>
15    <mapping class="com.jacaranda.User"/>
16  </session-factory>
17 </hibernate-configuration>
18
```

Fichero pom.xml, con su configuración:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
3   <modelVersion>4.0.0</modelVersion>
4   <groupId>com.jacaranda</groupId>
5   <artifactId>hibernate-core</artifactId>
6   <version>5.4.14-SNAPSHOT</version>
7   <packaging>war</packaging>
8   <url>https://mvnrepository.com/artifact/org.hibernate/hibernate-core</url>
9   <dependencies>
10     <dependency>
11       <groupId>com.mysql</groupId>
12       <artifactId>mysql-connector-java</artifactId>
13       <version>8.0.19</version>
14     </dependency>
15     <dependency>
16       <groupId>org.hibernate</groupId>
17       <artifactId>hibernate-core</artifactId>
18       <version>5.4.14.Final</version>
19     </dependency>
20   </dependencies>
21   <build>
22     <plugins>
23       <plugin>
24         <groupId>org.apache.maven.plugins</groupId>
25         <artifactId>maven-compiler-plugin</artifactId>
26         <version>3.8.1</version>
27         <configuration>
28           <release>11</release>
29         </configuration>
30       </plugin>
31       <plugin>
32         <groupId>org.apache.maven.plugins</groupId>
33         <artifactId>maven-war-plugin</artifactId>
34         <version>3.2.3</version>
35       </plugin>
36     </plugins>
37   </build>
38 </project>
```



```

1 package com.jacaranda;
2
3 import java.sql.Connection;
4
5 /**
6  * Clase con la que realizaremos la conexión a base de datos.
7  */
8 public class Conn {
9
10     private static StandardServiceRegistry sr = new StandardServiceRegistryBuilder().configure().build();
11     private static SessionFactory sf = new MetadataSources(sr).buildMetadata().buildSessionFactory();
12     private static Session session = sf.openSession();
13
14     public static Session getSession() {
15         return session;
16     }
17 }
18
19
20
21
22
23
24
25
26
27
28
29

```

Método de conexión.

- **Diseño**

Como nuestro primer objetivo, nos reunimos y pensamos en conjunto de que iba a tratar nuestro proyecto, llegamos a la conclusión unánime de adentrarnos en la venta de coches.

Decidimos el diseño en unos tonos blancos y azulados para hacer agradable la experiencia del usuario, con un logo personalizado de nuestra marca.

La página de Login será simple e intuitiva para el usuario, así como los formularios que recogen los datos para cumplir las diferentes funciones de nuestra página, como el de añadir, borrar o actualizar.

Nuestros coches tienen diferentes campos, **String**, **Double**, **LocalDate** y **Boolean** que este último será en el script de tipo Varchar debido a la problemática que tuvimos con este campo en Mysql siendo **0= no disponible** y **1=disponible**.

Nuestro proyecto contendrá un **script** de creación con al menos 100 o más registros insertados en cuanto a coches y 5 o más de tipo usuario, consta de dos tables y nos hemos ayudado para realizarlo de la herramienta online (<https://www.mockaroo.com/>)

```
create table Brand (
  name VARCHAR(50),
  country VARCHAR(50),
  address VARCHAR(50),

  CONSTRAINT pk_name PRIMARY KEY (name)
);

create table CAR_DATA (
  model_year VARCHAR(50),
  model_auto VARCHAR(50),
  car_make VARCHAR(13),
  availability VARCHAR(50),
  price DECIMAL(8,2),
  entry_date DATE,
  id VARCHAR(50),

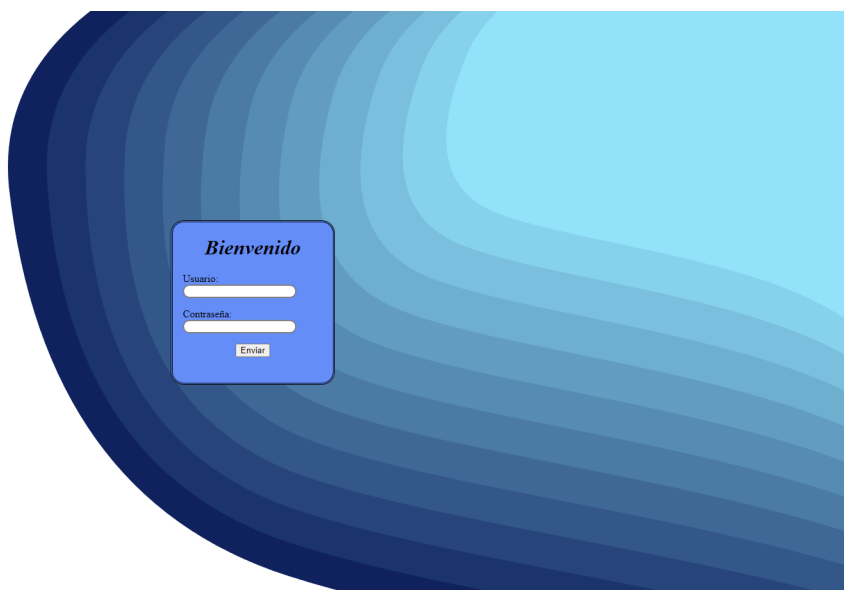
  CONSTRAINT pk_id PRIMARY KEY (id),
  CONSTRAINT fk_car_make FOREIGN KEY (car_make) REFERENCES Brand (name) ON DELETE CASCADE
);

create table USERS (
  name VARCHAR(50),
  pass VARCHAR(50),
  CONSTRAINT PK_USERS PRIMARY KEY (name)
);

insert into USERS (name, pass) values ('pilar', 'pilar');
insert into USERS (name, pass) values ('vicente', 'vicente');
insert into USERS (name, pass) values ('javi', 'javi');
insert into USERS (name, pass) values ('sergio', 'sergio');
insert into USERS (name, pass) values ('inma', 'inma');

insert into Brand (name, country, address) values ('Toyota', 'Bulgaria', '4199 Veith Way');
insert into Brand (name, country, address) values ('Mazda', 'Paraguay', '8 Annamark Junction');
insert into Brand (name, country, address) values ('Lincoln', 'Mozambique', '84775 Alpine Court');
insert into Brand (name, country, address) values ('Ford', 'Russia', '1 Luster Place');
insert into Brand (name, country, address) values ('Land Rover', 'Indonesia', '676 Summer Ridge Trail');
insert into Brand (name, country, address) values ('Chrysler', 'Portugal', '28480 Melrose Point');
insert into Brand (name, country, address) values ('Mercedes-Benz', 'Peru', '4869 Forest Dale Hill');
insert into Brand (name, country, address) values ('Chevrolet', 'Indonesia', '6599 Blue Bill Park Avenue');
insert into Brand (name, country, address) values ('Hyundai', 'Guatemala', '28 Pennsylvania Drive');
insert into Brand (name, country, address) values ('Dodge', 'Serbia', '6 Dorton Alley');

insert into CAR_DATA (model_year, model_auto, car_make, availability, price, entry_date, id) values (1986, '4800s Quattro', 'Chevrolet', true, 135750.25, '1992-10-26', '157-97');
insert into CAR_DATA (model_year, model_auto, car_make, availability, price, entry_date, id) values (1998, 'Dakota', 'Ford', false, 16863.52, '2003-02-14', '374-47-6957');
insert into CAR_DATA (model_year, model_auto, car_make, availability, price, entry_date, id) values (2008, 'Explorer Sport Trac', 'Hyundai', false, 91852.69, '1994-05-23', '83');
insert into CAR_DATA (model_year, model_auto, car_make, availability, price, entry_date, id) values (2004, 'Classic', 'Mercedes-Benz', false, 44333.79, '2008-05-29', '436-44-1');
insert into CAR_DATA (model_year, model_auto, car_make, availability, price, entry_date, id) values (1993, 'Suburban 2500', 'Chrysler', true, 102985.11, '2006-02-18', '160-60-1');
insert into CAR_DATA (model_year, model_auto, car_make, availability, price, entry_date, id) values (2007, 'XJ', 'Hyundai', true, 165569.03, '2005-11-12', '641-83-8281');
insert into CAR_DATA (model_year, model_auto, car_make, availability, price, entry_date, id) values (1995, '9000', 'Chevrolet', true, 110922.93, '2002-11-10', '858-62-5497');
insert into CAR_DATA (model_year, model_auto, car_make, availability, price, entry_date, id) values (2003, 'Tribute', 'Chrysler', true, 29741.39, '2006-12-09', '107-70-5068');
insert into CAR_DATA (model_year, model_auto, car_make, availability, price, entry_date, id) values (2007, 'Avalanche', 'Ford', true, 21457.75, '2022-03-27', '381-34-1670');
insert into CAR_DATA (model_year, model_auto, car_make, availability, price, entry_date, id) values (1993, 'Celica', 'Lincoln', true, 72321.63, '1990-07-08', '519-55-9389');
insert into CAR_DATA (model_year, model_auto, car_make, availability, price, entry_date, id) values (1999, 'Metro', 'Dodge', false, 124521.78, '2018-10-15', '755-18-0479');
insert into CAR_DATA (model_year, model_auto, car_make, availability, price, entry_date, id) values (1993, 'Elan', 'Mercedes-Benz', true, 124424.31, '1982-11-21', '125-95-8667');
```



Bienvenido

Usuario:

Contraseña:

Login.




¡Vaya!

**No hemos podido encontrar
la página que buscas.**

Código de error: 404

Página de error.

← → ↻ localhost:8080/CarHibernate/indexBrand.jsp ☆ 🔒 ≡




Sesion: Javi
Log Out

Añadir Marca

Marca	País	Sucursal	Delete	Update	Ver coches
Chevrolet	Indonesia	6599 Blue Bill Park Avenue			
Chrysler	Portugal	28480 Melrose Point			
Dodge	Serbia	6 Dorton Alley			
Ford	Russia	1 Luster Place			
Hyundai	Guatemala	28 Pennsylvania Drive			
Land Rover	Indonesia	676 Summer Ridge Trail			
Lincoln	Mozambique	04775 Alpine Court			
Mazda	Paraguay	8 Annamark Junction			
Mercedes-Benz	Peru	4869 Forest Dale Hill			
Toyota	Bulgaria	4199 Veith Way			

Inicio con marcas disponibles.

← → ↻ localhost:8080/CarHibernate/AddBrand.jsp ☆ 🔒 ≡



Sesion: Javi
Log Out

Añadir Marca:

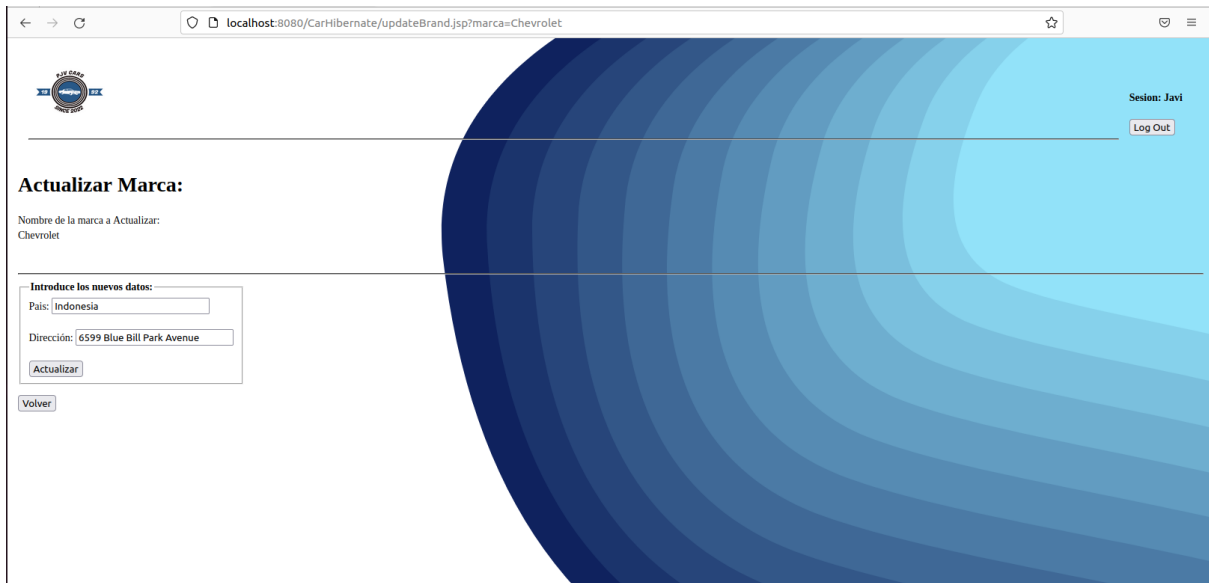
Rellene los siguientes datos:

Nombre de la Marca:

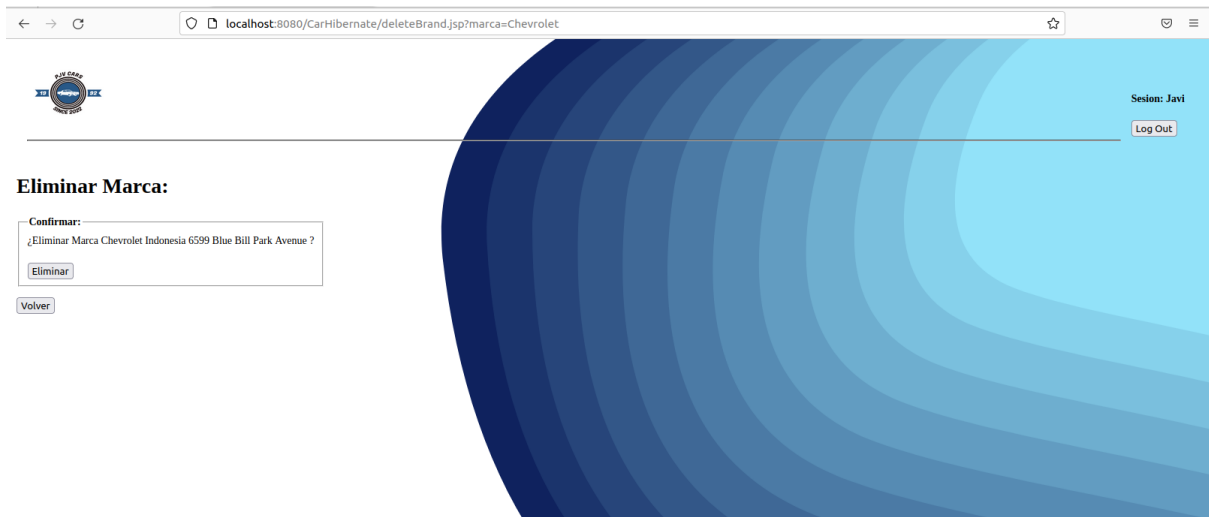
País:

Dirección:


Subpágina para añadir marca



Subpágina para actualizar una marca



Subpágina para eliminar una marca




Sesion: Javi

Log Out

Add car

Model year	Model Car	Car maker	Availability	Price	Entry date	ID	Delete	Update
2004	Bravada	Chevrolet	1	23387.63	1982-01-06	373-81-0506		
2009	Amanti	Chevrolet	0	156888.24	2013-06-12	382-49-9776		
2004	Silverado 2500	Chevrolet	0	150324.66	1994-12-30	492-11-4360		
2005	RX	Chevrolet	1	77976.41	2009-05-26	649-80-1809		
1998	Century	Chevrolet	0	103511.09	2013-10-31	680-70-5122		
2006	Baja	Chevrolet	0	111865.95	2001-09-10	727-93-0318		
1992	98	Chevrolet	0	130379.1	1994-02-27	771-62-0418		
1998	Regal	Chevrolet	0	98215.48	1980-10-20	835-42-7395		
2003	Caravan	Chevrolet	1	101438.63	2006-08-13	854-67-7962		
1995	9000	Chevrolet	1	110922.93	2002-11-10	858-62-5497		

Subpágina con lista de coches filtrados por marca



Sesion: Javi

Log Out

Añadir coche Chevrolet a la venta:

Rellene los siguientes datos:

 Año del vehículo:

 Modelo:

 Disponibilidad: Si ☐ No ☐


 Precio:

 Fecha de entrada:

 Identificador del vehículo:

Subpágina para añadir un coche del tipo de marca por el cual se ha filtrado la tabla.

← → ↻ localhost:8080/CarHibernate/UpdateCar.jsp?value=373-81-0506 ☆ 🛡️ ☰



Sesion: Javi
[Log Out](#)

Actualizar Vehículo:

Identificador del coche a Actualizar:
373-81-0506

Introduce los nuevos datos:

Año del vehículo:

Modelo:

Disponibilidad: ☒ Sí ☐ No

Precio:


Fecha de entrada:

[Actualizar](#)

[Volver](#)

Actualizar un coche específico de la marca seleccionada.

← → ↻ localhost:8080/CarHibernate/deleteCar.jsp?value=373-81-0506 ☆ 🛡️ ☰



Sesion: Javi
[Log Out](#)

Eliminar Vehículo:

Confirmar:

¿Eliminar vehículo Chevrolet Bravada 2004 . Con ID: 373-81-0506 ?

[Eliminar](#)

[Volver](#)

Eliminar un coche específico de la marca seleccionada.

- **Funcionalidades**

A continuación detallaremos cómo realizaremos las diferentes funcionalidades de nuestro proyecto:

1. Clase Car

En la creación de la clase Car, precisamos de todos los campos anteriormente descritos y dicha clase consta de unos atributos definidos en private y un constructor donde crearemos nuestros coches, además de contener dichos getter y setter, ToString, hashCode y Equals.

Deberemos definir el atributo Brand como **“ManyToOne”**, además deberemos indicar como se llama cada columna en nuestra base de datos y como se llama la tabla en la base de datos.

```
1 package com.jacaranda;
2
3 import java.time.LocalDate;
4
5 /**
6  * Clase coche(Car) compuesta por varios campos. ID es su identificador principal
7  * y Brand es una Clave Foránea o FK de la clase Marca, en la base de datos se denomina "car_make".
8  * Cada coche pertenecerá a una marca en concreto.
9  */
10 @Entity
11 @Table(name="CAR_DATA")
12 public class Car {
13
14     @Id
15     private String id;
16     @Column(name="model_year")
17     private int modelYear;
18     @Column(name="model_auto")
19     private String modelAuto;
20     @ManyToOne
21     @JoinColumn(name="car_make")
22     private Brand carMaker;
23     private String availability;
24     private double price;
25     @Column(name="entry_date")
26     private LocalDate dateEntry;
27
28     public Car() {
29     }
30
31     public Car(int modelYear, String modelAuto, Brand carMaker, String availability, double price,
32         LocalDate dateEntry, String id) {
33         super();
34         this.modelYear = modelYear;
35         this.modelAuto = modelAuto;
36         this.carMaker = carMaker;
37         this.availability = availability;
38         this.price = price;
39         this.dateEntry = dateEntry;
40         this.id = id;
41     }
42
43     public int getModelYear() {
44         return modelYear;
45     }
46
47     public void setModelYear(int modelYear) {
48         this.modelYear = modelYear;
49     }
50
51     public String getModelAuto() {
52         return modelAuto;
53     }
54
55     public void setModelAuto(String modelAuto) {
56         this.modelAuto = modelAuto;
57     }
58
59     public Brand getCarMaker() {
60         return carMaker;
61     }
62 }
```

2. Clase Brand

En la creación de la clase Brand, precisamos de nombre, país, dirección y una lista de coches asociados a esa marca, un constructor donde crearemos nuestras marcas, además de contener dichos getter and setter, ToString, hashCode y Equals.

Deberemos definir la lista de coches como “**OneToMany**”, además deberemos indicar como se llama cada columna en nuestra base de datos y como se llama la tabla en la base de datos.

```
1 package com.jacaranda;
2
3 import java.util.ArrayList;
12 /**
13  * Clase Marca (Brand), compuesta por un campo nombre, país y dirección. El campo nombre será su identificador principal.
14  * Cada marca tendrá una lista de coches que pertenezcan a dicha marca.
15  */
16 @Entity(name="Brand")
17 public class Brand {
18     @Id
19     private String name;
20     @Column(name="country")
21     private String country;
22     @Column(name="address")
23     private String address;
24     @OneToMany(mappedBy="carMaker", cascade = CascadeType.ALL, orphanRemoval = true)
25     private List<Car> listCar;
26
27
28     public Brand() {
29
30     }
31
32     public Brand(String name, String country, String address) {
33         super();
34         this.name = name;
35         this.country = country;
36         this.address = address;
37         this.listCar = null;
38     }
39
40     public Brand(String name, String country, String address, List<Car> listCar) {
41         super();
42         this.name = name;
43         this.country = country;
44         this.address = address;
45         this.listCar = null;
46     }
47
48
49     public String getName() {
50         return name;
51     }
52
53
54     public void setName(String name) {
55         this.name = name;
56     }
57
58
59     public String getCountry() {
60         return country;
61     }
62
63
64     public void setCountry(String country) {
65         this.country = country;
66     }
67 }
```

3. Clase User

En la creación de la clase User, precisamos de todos los campos anteriormente descritos y dicha clase consta de unos atributos definidos en private y un constructor donde crearemos nuestros Usuarios.

```
package src.main.java.com.jacaranda;

public class User {

    private String name;
    private String pass;

    public User() {

    }

    public User(String name, String pass) {

        this.name = name;
        this.pass = pass;
    }
}
```

4. Clase UtilsUser

Esta clase la utilizaremos de lógica para la anterior clase, la de User, aquí tendremos varios métodos que nos servirán por ejemplo para ver si un usuario es válido o no entre otras cosas y funcionalidades.

```
17 public class UtilsUsers {
18
19     // Obtiene un usuario en concreto partiendo del parametro name.
20     public static User getUser(String name ) {
21         Session session = Conn.getSession();
22
23         User user = (User) session.getUser(User.class,name);
24         return user;
25     }
26
27
28     // Obtiene una lista con los usuarios
29     public static ArrayList<User> getUsers(){
30         Session session = Conn.getSession();
31
32         Query<User> query = session.createQuery("SELECT p FROM com.jacaranda.User p");
33         ArrayList<User> users = (ArrayList<User>) query.getResultList();
34
35         return users;
36     }
37
38     //Comprueba si el usuario es valido en nuestra base de datos.
39     public static boolean isValid(String name, String pass) {
40         boolean valid = false;
41         try {
42             Session session = Conn.getSession();
43             Query<User> query = session.createQuery("SELECT p FROM com.jacaranda.User p WHERE name='" + name + "'and pass='"+pass+"'",User.class);
44             if(!query.getResultList().isEmpty()) {
45                 valid = true;
46             }
47         } catch (Exception e) {
48             System.out.println(e.getMessage());
49         }
50         return valid;
51     }
52
53
54     /**
55      * Cerrar sesion. Limpia parametros y atributos de la session actual.
56      */
57     public static void closeSession () {
58         Session session = Conn.getSession();
59         session.clear();
60     }
61
62 }
63
```

5. Clase CRUDBrand

En esta clase, recogeremos nuestras funcionalidades para así poder separar la lógica del proyecto de nuestro fin.

En la siguientes imágenes mostraremos algunos de sus métodos.

```
1 package com.jacaranda;
2
3 import java.util.ArrayList;
4
5
6
7
8
9
10
11
12
13 /**
14  * Clase encargada del control de la marca, dispone de varios metodos:
15  * -Buscar la marca por nombre de la marca.
16  * -Añadir una marca a la lista de marcas. Este metodo nos servirá para actualizar la marca
17  * ya que si no existe la crea y si existe la updatea.
18  * -Borrar una marca.
19  * -Obtener todas las marcas.
20  * @author vicente
21  */
22
23 public class CRUDBrand {
24
25     public static Brand getBrand(String name ) {
26         Session session = Conn.getSession();
27         Brand brand = (Brand) session.get(Brand.class,name);
28         return brand;
29     }
30
31     public static boolean saveBrand( Brand brand) {
32         boolean resultado=false;
33         Session session = Conn.getSession();
34
35         try {
36             session.getTransaction().begin();
37             session.saveOrUpdate(brand);
38             session.getTransaction().commit();
39             resultado=true;
40
41         } catch (Exception e) {
42             e.printStackTrace();
43         }
44         return resultado;
45     }
46
47
48     public static boolean deleteBrand(Brand brand) {
49         boolean resultado= false;
50         Session session =Conn.getSession();
51
52         try {
53
54             session.getTransaction().begin();
55             session.delete(brand);
56             session.getTransaction().commit();;
57             resultado=true;
58
59         } catch (Exception e) {
60             e.printStackTrace();
61         }
62
63         return resultado;
64     }
65
66     public static ArrayList<Brand> getBrands(){
67         Session session = Conn.getSession();
68
69         Query<Brand> query = session.createQuery("SELECT p FROM com.jacaranda.Brand p");
70         ArrayList<Brand> brand = (ArrayList<Brand>) query.getResultList();
71
72         return brand;
73     }
74 }
```

6. Clase CRUDCar

En esta clase, recogeremos nuestras funcionalidades para así poder separar la lógica del proyecto de nuestro fin.

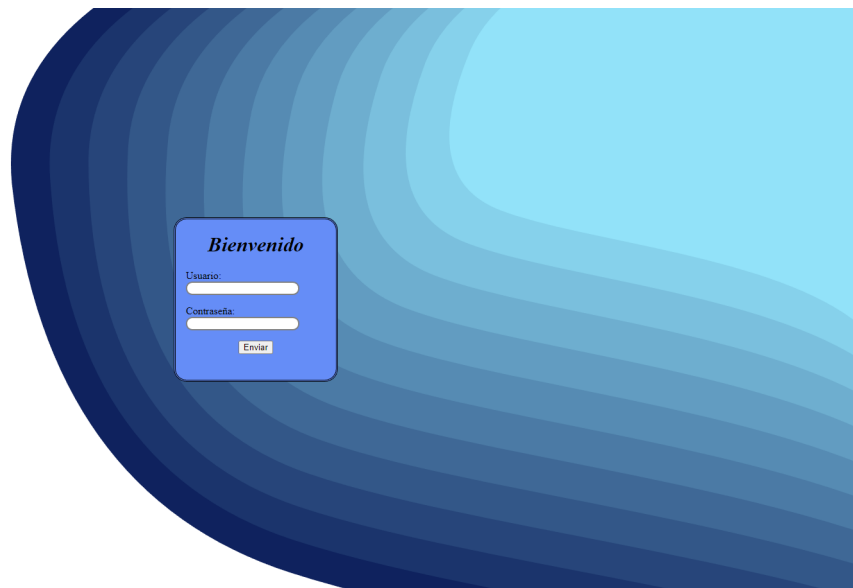
En la siguientes imágenes mostraremos algunos de sus métodos.

```
1 package com.jacaranda;
2
3+ import java.util.ArrayList;
4
5
6 /**
7  * Clase encargada del control de los coches, dispone de varios metodos:
8  * -Buscar el coche por id.
9  * -Añadir un coche a la lista de coches de una marca
10 * -Updatear un coche.
11 * -Borrar un coche.
12 * -Obtener todos los coches de una marca.
13 * @author vicente
14 */
15
16 public class CRUDCar {
17
18     public static Car getCar(String id) {
19         Session session = Conn.getSession();
20         Car car = (Car) session.get(Car.class,id);
21
22         return car;
23     }
24
25     public static boolean saveCar( Car car) {
26         boolean resultado=false;
27         Session session = Conn.getSession();
28
29         try {
30
31             session.getTransaction().begin();
32             session.saveOrUpdate(car);
33             car.getCarMaker().getListCar().add(car);
34             session.getTransaction().commit();
35
36             resultado=true;
37
38         } catch (Exception e) {
39             e.printStackTrace();
40         }
41         return resultado;
42     }
43
44     public static boolean updateCar( Car car) {
45         boolean resultado=false;
46         Session session = Conn.getSession();
47
48         try {
49             session.getTransaction().begin();
50             session.update(car);
51             session.getTransaction().commit();
52             resultado=true;
53
54         } catch (Exception e) {
55             e.printStackTrace();
56         }
57         return resultado;
58     }
59
60     public static boolean carDelete(Car car) {
61         boolean resultado= false;
62         Session session =Conn.getSession();
63
64         try {
```


7. Login

En lo referido al Login, tenemos un formulario de entrada de datos que nos servirá para recoger los parámetros introducidos por el usuario. El usuario una vez introducidos los parámetros procederá a acceder a través de un botón.

Nuestra lógica de programa integrada en la clase **UtilsUser** realizará la ejecución de un método el cuál comprobará en nuestra base de datos si el usuario introducido corresponde a nuestros parámetros, de no ser así se le redirigirá a una pantalla de error.

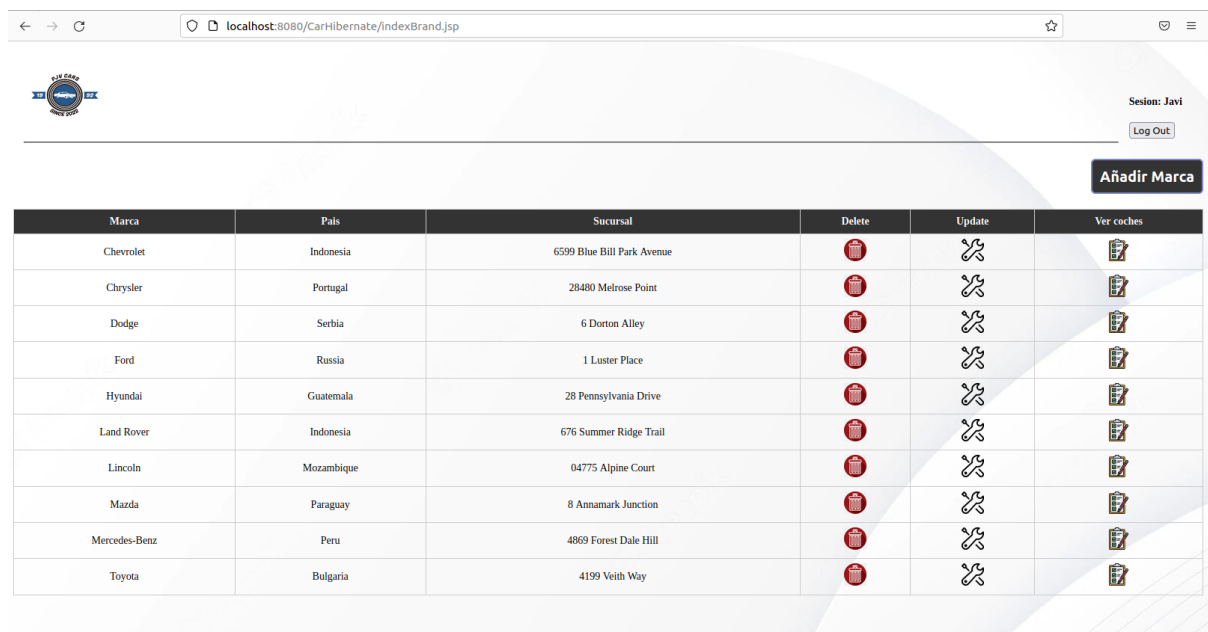


18

8. Inicio

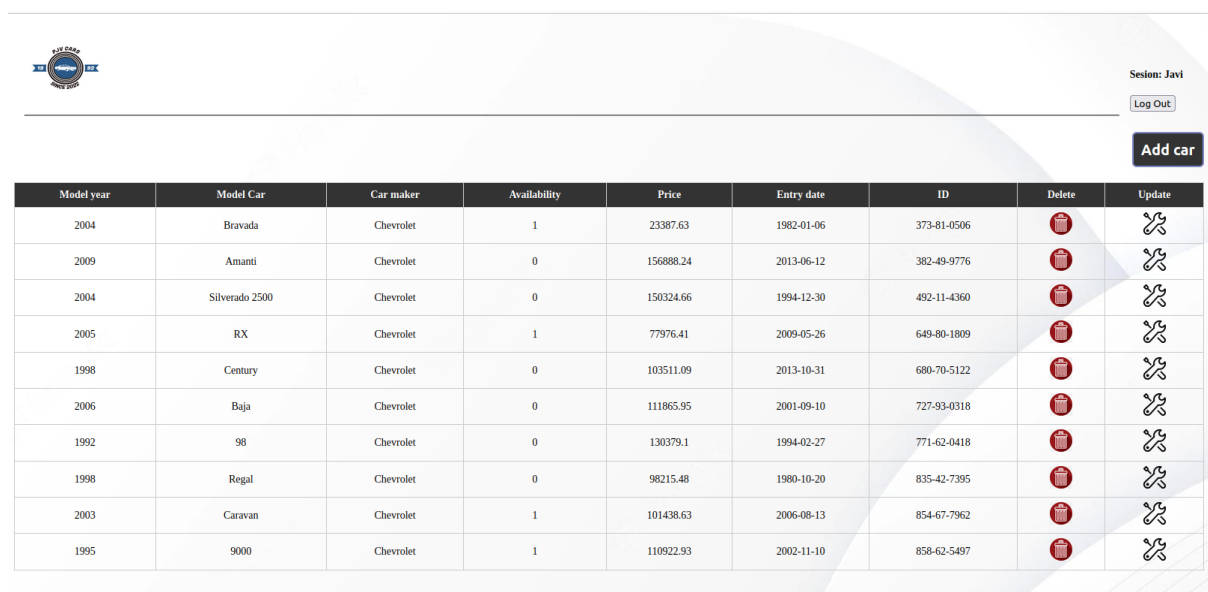
En cuanto al Inicio hemos añadido un archivo .jsp en el cuál recogemos los parámetros que introduce el usuario si previamente está registrado en nuestra base de datos y no muestra la página de error.

En dicho archivo abrimos una sesión que nos servirá para la seguridad de nuestra página, además realizamos una tabla la cuál muestra los registros de las marcas de nuestro concesionario en nuestra base de datos.



Marca	Pais	Sucursal	Delete	Update	Ver coches
Chevrolet	Indonesia	6599 Blue Bill Park Avenue			
Chrysler	Portugal	28480 Melrose Point			
Dodge	Serbia	6 Dorton Alley			
Ford	Russia	1 Luster Place			
Hyundai	Guatemala	28 Pennsylvania Drive			
Land Rover	Indonesia	676 Summer Ridge Trail			
Lincoln	Mozambique	04775 Alpine Court			
Mazda	Paraguay	8 Annamark Junction			
Mercedes-Benz	Peru	4869 Forest Dale Hill			
Toyota	Bulgaria	4199 Veith Way			

Si accedemos a la lista de coches asociados a esa marca tendremos una segunda página donde nos aparecerá nuestra tabla correspondiente de los coches de esa marca.



Model year	Model Car	Car maker	Availability	Price	Entry date	ID	Delete	Update
2004	Bravada	Chevrolet	1	23387.63	1982-01-06	373-81-0506		
2009	Amanti	Chevrolet	0	156888.24	2013-06-12	382-49-9776		
2004	Silverado 2500	Chevrolet	0	150324.66	1994-12-30	492-11-4360		
2005	RX	Chevrolet	1	77976.41	2009-05-26	649-80-1809		
1998	Century	Chevrolet	0	103511.09	2013-10-31	680-70-5122		
2006	Baja	Chevrolet	0	111865.95	2001-09-10	727-93-0318		
1992	98	Chevrolet	0	130379.1	1994-02-27	771-62-0418		
1998	Regal	Chevrolet	0	98215.48	1980-10-20	835-42-7395		
2003	Caravan	Chevrolet	1	101438.63	2006-08-13	854-67-7962		
1995	9000	Chevrolet	1	110922.93	2002-11-10	858-62-5497		

9. Añadir coche

En lo referente a la funcionalidad de añadir, hemos optado por un botón en la tabla que al pulsarlo redirige al usuario hacia un .jsp en el cuál recogemos todos los datos necesarios para que el usuario pueda añadir un coche a nuestra base de datos y por ende aparezca en nuestra tabla de coches de esa marca.

The screenshot shows a web browser window with the address bar displaying 'localhost:8080/CarHibernate/addCar.jsp?marca=Chevrolet'. The page has a blue header with a logo on the left and 'Sesion: Javi' with a 'Log Out' button on the right. The main content area is titled 'Añadir coche Chevrolet a la venta:' and contains a form with the following fields: 'Rellene los siguientes datos:', 'Año del vehículo:' (dropdown), 'Modelo:' (text), 'Disponibilidad: Si ☐ No ☐', 'Precio:' (text), 'Fecha de entrada:' (calendar), and 'Identificador del vehículo:' (text). Below the form is a 'Añadir Coche' button and a 'Volver' button. The background of the page features a large, abstract blue wave pattern.

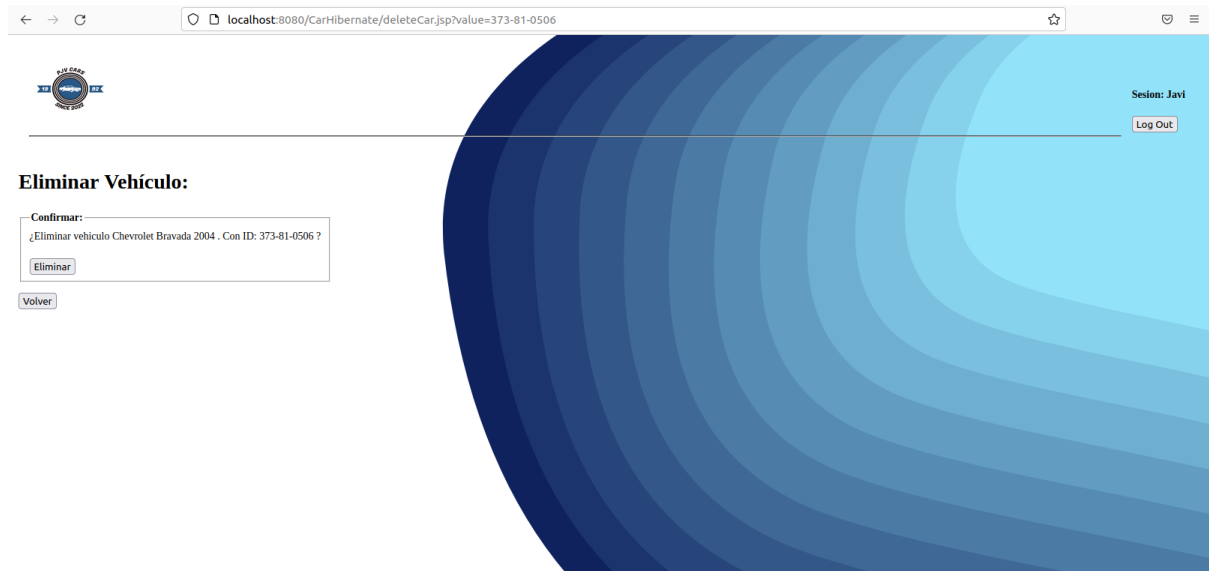
Formulario para añadir coche.

```
public static boolean saveCar( Car car) {  
    boolean resultado=false;  
    Session session = Conn.getSession();  
  
    try {  
  
        session.getTransaction().begin();  
        session.saveOrUpdate(car);  
        car.getCarMaker().getListCar().add(car);  
        session.getTransaction().commit();  
  
        resultado=true;  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
    return resultado;  
}
```

Método de añadir coche

10. Borrar coche

Partiendo de la funcionalidad de borrar un coche, procedimos a colocar una imagen de borrar en la tabla que si el usuario clicca sobre ella lo redirige a un .jsp donde se recoge el id del coche a borrar.



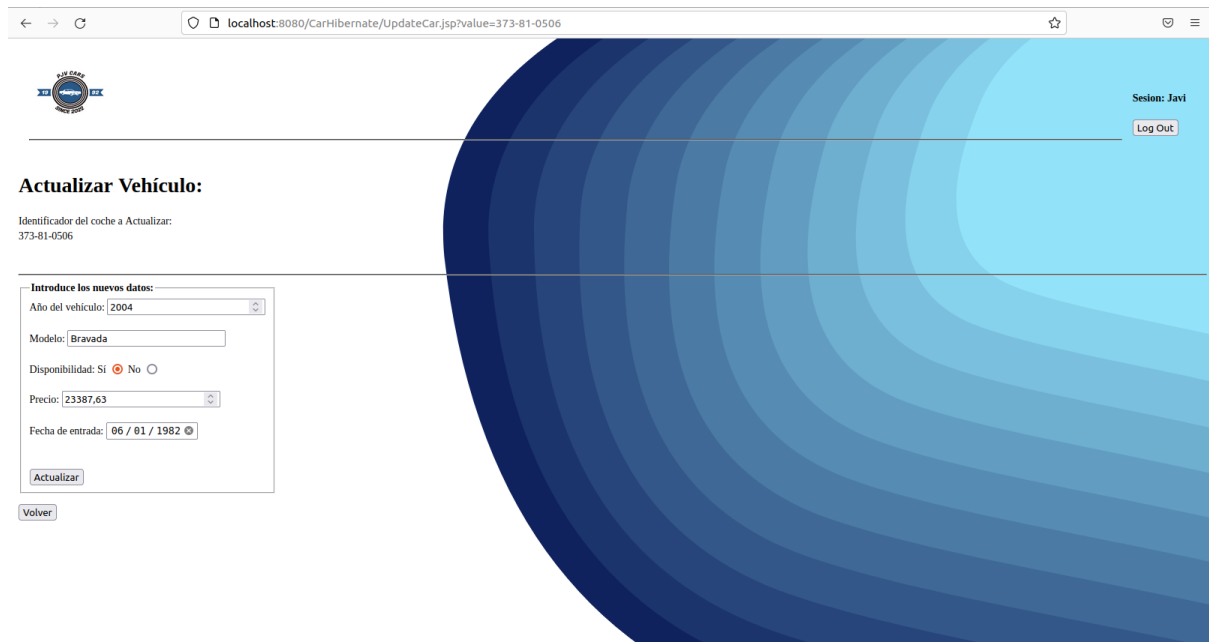
Borrar un coche.

```
public static boolean carDelete(Car car) {  
    boolean resultado= false;  
    Session session =Conn.getSession();  
  
    try {  
  
        session.getTransaction().begin();  
        session.delete(car);  
        car.getCarMaker().getListCar().remove(car);  
        session.getTransaction().commit();  
        resultado=true;  
  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
  
    return resultado;  
}
```

Método de borrar.

11. Actualizar coche

Para la funcionalidad de actualizar, procedimos a colocar una imagen de actualizar en la tabla que si el usuario clicca sobre ella lo redirige a un .jsp donde se recogen los diferentes datos necesarios para poder actualizar un coche.

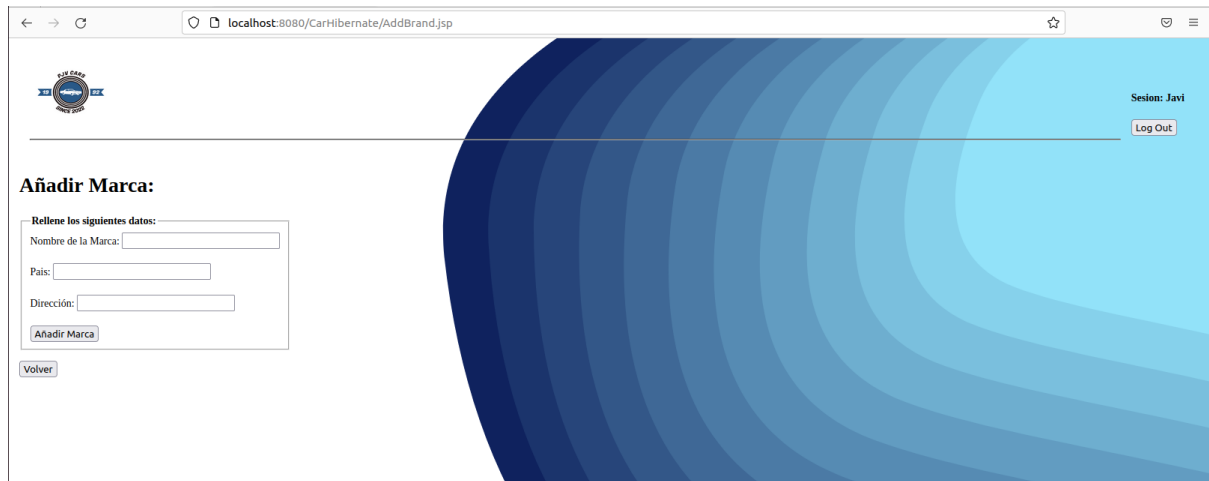


```
public static boolean updateCar( Car car) {  
    boolean resultado=false;  
    Session session = Conn.getSession();  
  
    try {  
        session.getTransaction().begin();  
        session.update(car);  
        session.getTransaction().commit();  
        resultado=true;  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
    return resultado;  
}
```

Método actualizar el coche.

12. Añadir marca:

En lo referente a la funcionalidad de añadir, hemos optado por un botón en la tabla que al pulsarlo redirige al usuario hacia un .jsp en el cuál recogemos todos los datos necesarios para que el usuario pueda añadir una marca a nuestra base de datos y por ende aparezca en nuestra tabla de marcas.

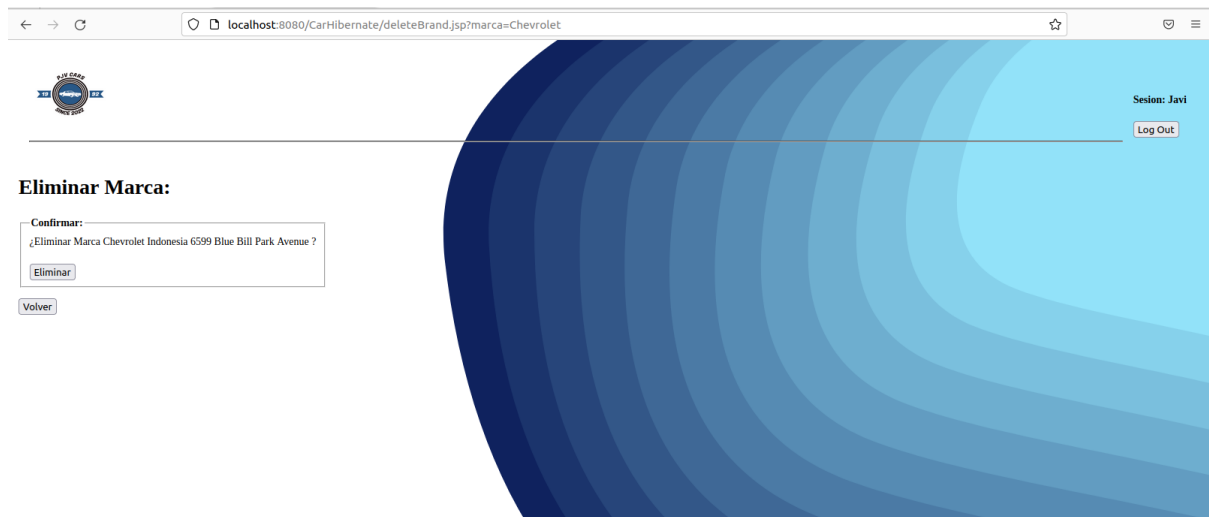


```
public static boolean saveBrand( Brand brand) {  
    boolean resultado=false;  
    Session session = Conn.getSession();  
  
    try {  
        session.getTransaction().begin();  
        session.saveOrUpdate(brand);  
        session.getTransaction().commit();  
        resultado=true;  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
    return resultado;  
}
```

Método para añadir marca:

13. Borrar Marca

Partiendo de la funcionalidad de borrar una marca, procedimos a colocar una imagen de borrar en la tabla que si el usuario clicca sobre ella lo redirige a un .jsp donde se recoge el nombre de la marca a borrar.

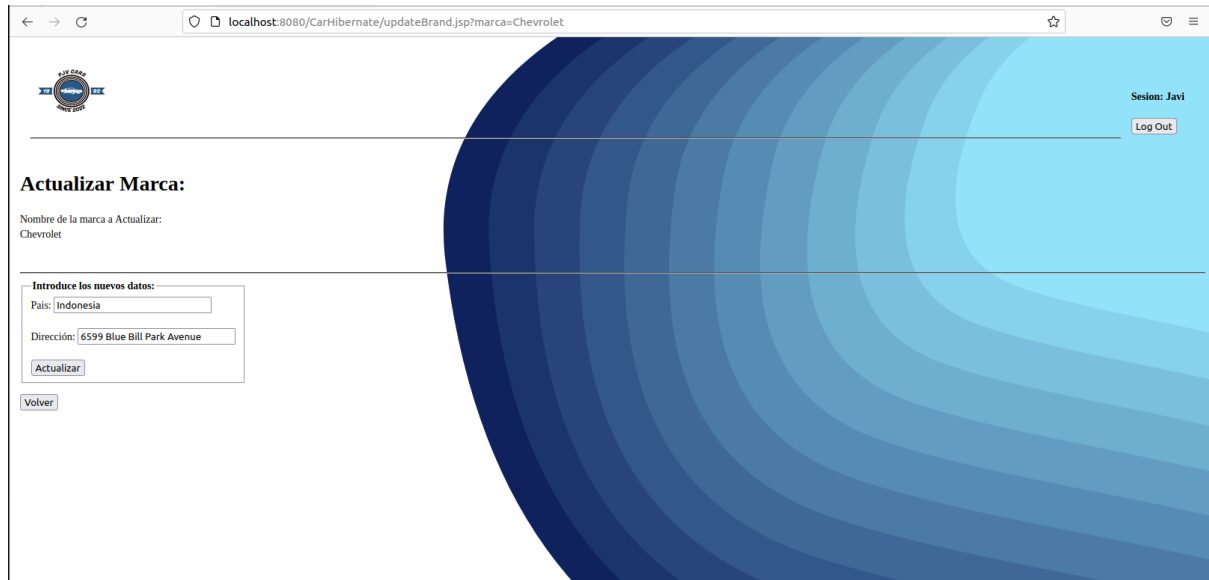


```
public static boolean deleteBrand(Brand brand) {  
    boolean resultado= false;  
    Session session =Conn.getSession();  
  
    try {  
        session.getTransaction().begin();  
        session.delete(brand);  
        session.getTransaction().commit();  
        resultado=true;  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
    return resultado;  
}
```

Método para borrar una marca

14. Actualizar marca:

Para la funcionalidad de actualizar, procedimos a colocar una imagen de actualizar en la tabla que si el usuario clicla sobre ella lo redirige a un .jsp donde se recogen los diferentes datos necesarios para poder actualizar una marca.



```
public static boolean saveBrand( Brand brand) {  
    boolean resultado=false;  
    Session session = Conn.getSession();  
  
    try {  
        session.getTransaction().begin();  
        session.saveOrUpdate(brand);  
        session.getTransaction().commit();  
        resultado=true;  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
    return resultado;  
}
```

Método para añadir marca.