

Lab: Spring Essentials

MobiLeLeLe web application

MobiLeLeLe is an application in which you register cars, with several properties.

You will have to create a simple application which has several pages and some object entities.

1. Data

This is the data layer of the application. There are some data object for you to implement.

Brand

Create a **Brand** class, which holds the following properties:

- **id** – a **uuid or number**.
- **name** – a **name of brand**.
- **created** – a **date and time**.
- **modified** – a **date and time**.

Model

Create a **Model** class, which holds the following properties:

- **id** – **uuid or number**.
- **name** – a **model name**.
- **category** – an enumeration (Car, Buss, Truck, Motorcycle)
- **imageUrl** – the **url of image** with size between 8 and 512 characters.
- **startYear** – a **number**.
- **endYear** – a **number**.
- **created** – a **date and time**.
- **modified** – a **date and time**.
- **brand** – a **model brand**.

Offer

Create a **Model** class, which holds the following properties:

- **id** – **uuid or number**.
- **description** – some **text**.
- **engine** – **enumerated** value (GASOLINE, DIESEL, ELECTRIC, HYBRID).
- **imageUrl** – the **url of image**.
- **mileage** – a **number**.
- **price** – the **price of the offer**.
- **transmission** – **enumerated** value (MANUAL, AUTOMATIC).
- **year** – the **year** of offered car.
- **created** – a **date and time**.
- **modified** – a **date and time**.
- **model** – the **model of a car**.
- **seller** – a **user that sells the car**.

User

Create a **User** class, which holds the following properties:

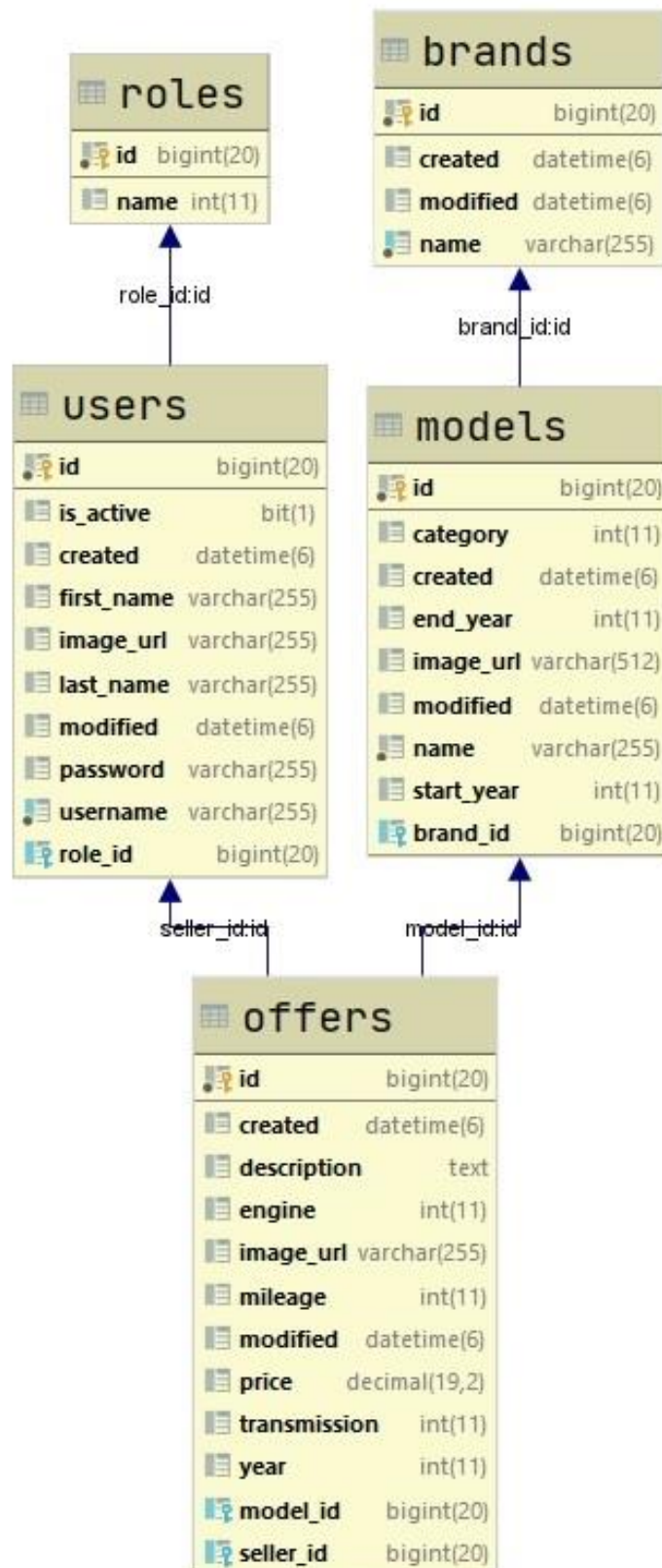
- **id** – **uuid or number**.
- **username** – username of the **user**.
- **password** – password of the **user**.
- **firstName** – first name of the **user**.
- **lastName** – last name of the **user**.
- **isActive** – **true OR false**.
- **role** – **user's role (User or Admin)**.
- **imageUrl** – a **url of user's picture**.
- **created** – a **date and time**.
- **modified** – a **date and time**.

UserRole

Create a **UserRole** class, which holds the following properties:

- **id** – **uuid or number**.
- **role** – **enumerated** value.

This is an example of ER Diagram



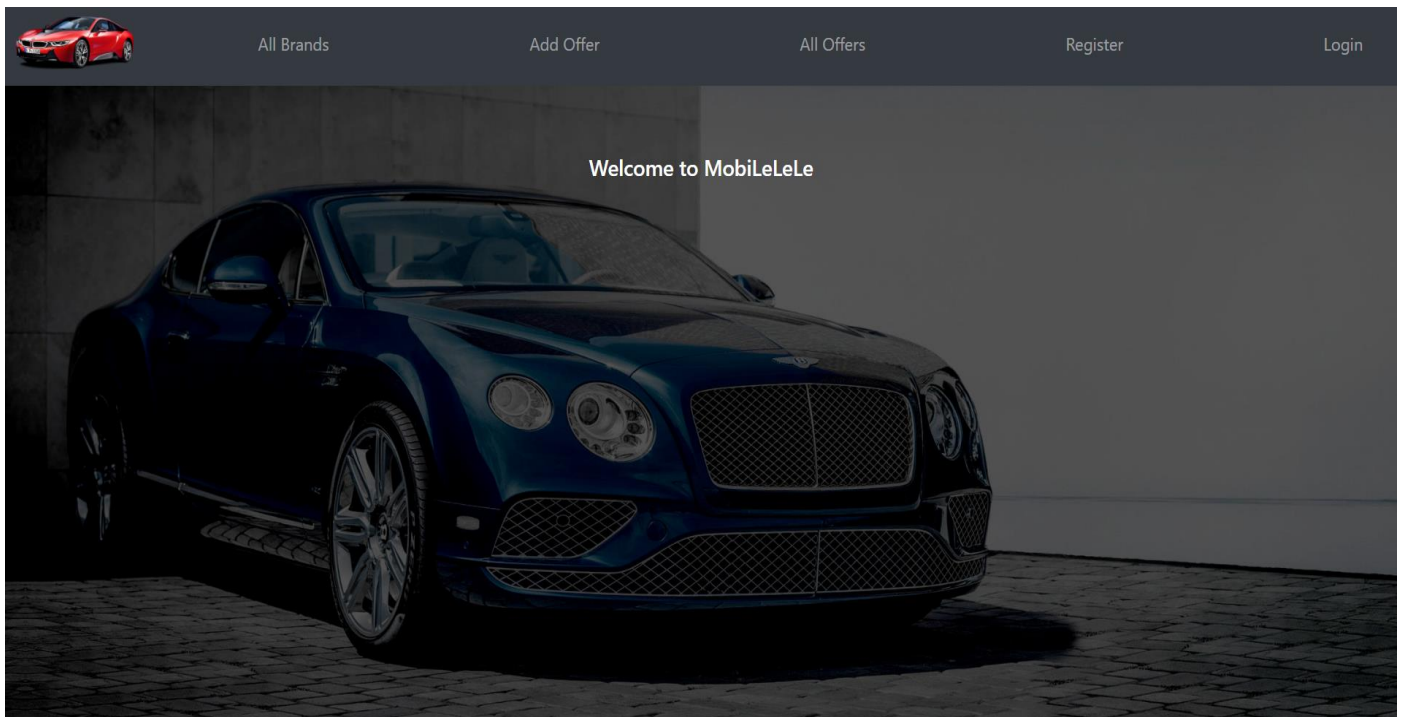
2. Populate DB

Create Data Initializer class, that populate the DB with information about cars when application starts for the first time.

3. Home/index - route ("/")

It should support only a **GET** request.

It should return the following HTML page, upon a **GET** request.

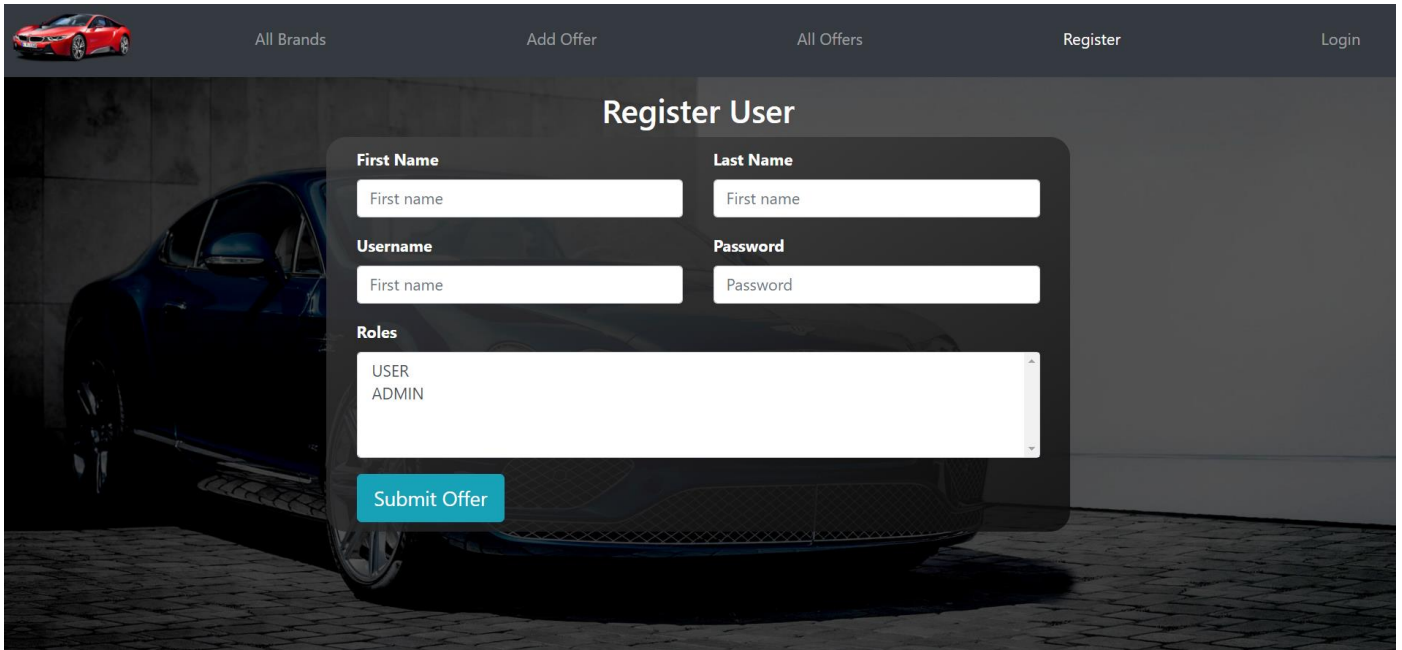


4. Register User - route ("/users/register").

It should support only a **GET & POST** request.

It should return the following HTML page, upon a **GET** request.

First we need to add some users in our DB.



Hint section:

- Because you will learn Thymeleaf in details on the next next lecture, we'll give you hints on how to implement some things
- Do not forget to add Thymeleaf in you pom.xml file
- Do not forget to add Thymeleaf name spaces:

```
<html lang="en" xmlns:th="http://www.thymeleaf.org">
```

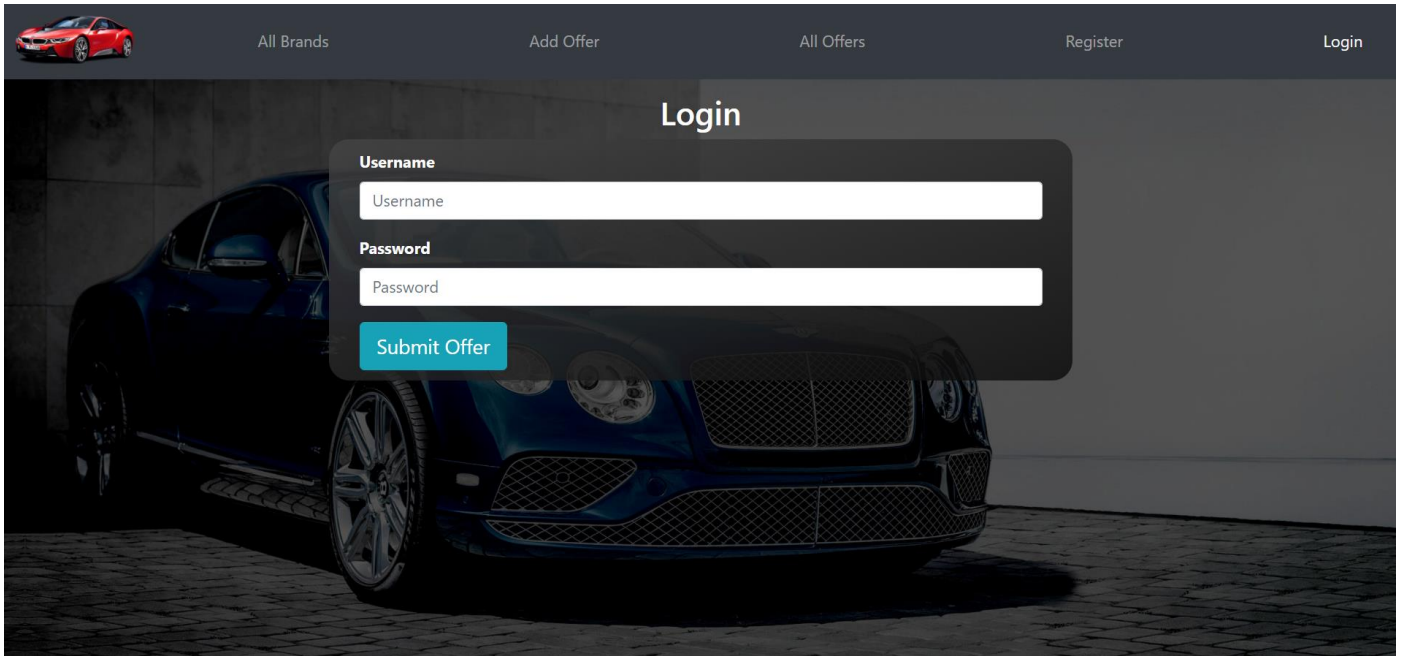
- Also you need to add in are the html form action and method (remember last lecture):

```
<form th:action="@{/users/register}" th:method="POST"></form>
```

5. Login - route ("/users/login")

It should support only a **GET & POST** request.

It should return the following HTML page, upon a **GET** request.



6. Navigation for login user

When a user logs in, in the application, he cannot see the Register and Login buttons, but Logout.

Also, if he has an Admin role, he can see the Admin dropdown.

Because you will learn Thymeleaf in the next lesson, we will give you a little hint how to do this point.

Hint Section:

```
<li th:if="${session.user}" class="nav-item">
  <div class="form-inline my-2 my-lg-0 border px-3">
    <div class="logged-user"
      th:text="|Welcome, ${session.user.firstName}|"></div>
    <a class="nav-link" href="/users/logout">Logout</a>
  </div>
</li>
```

Expected result for login user

7. All brands and models in out DB - route ("/brands/all").

It should support only a **GET** request.

It should return the following HTML page, upon a **GET** request.

All Brands

Car brand: Opel

No	Name	Category	Start Year	End Year	Picture
No	Adam	CAR	2013	2019	
No	Agila	CAR	2000	2007	
No	Agila B	CAR	2008	2015	
No	Ampera	CAR	2011		

Car brand: BMW

No	Name	Category	Start Year	End Year	Picture
No 5		CAR	1972		
No 6		CAR	2003		
No 7		CAR	1977		
No 8		CAR	2018		
No F800S		MOTORCYCLE	2006	2010	
No G11		CAR	2015		

Hint:

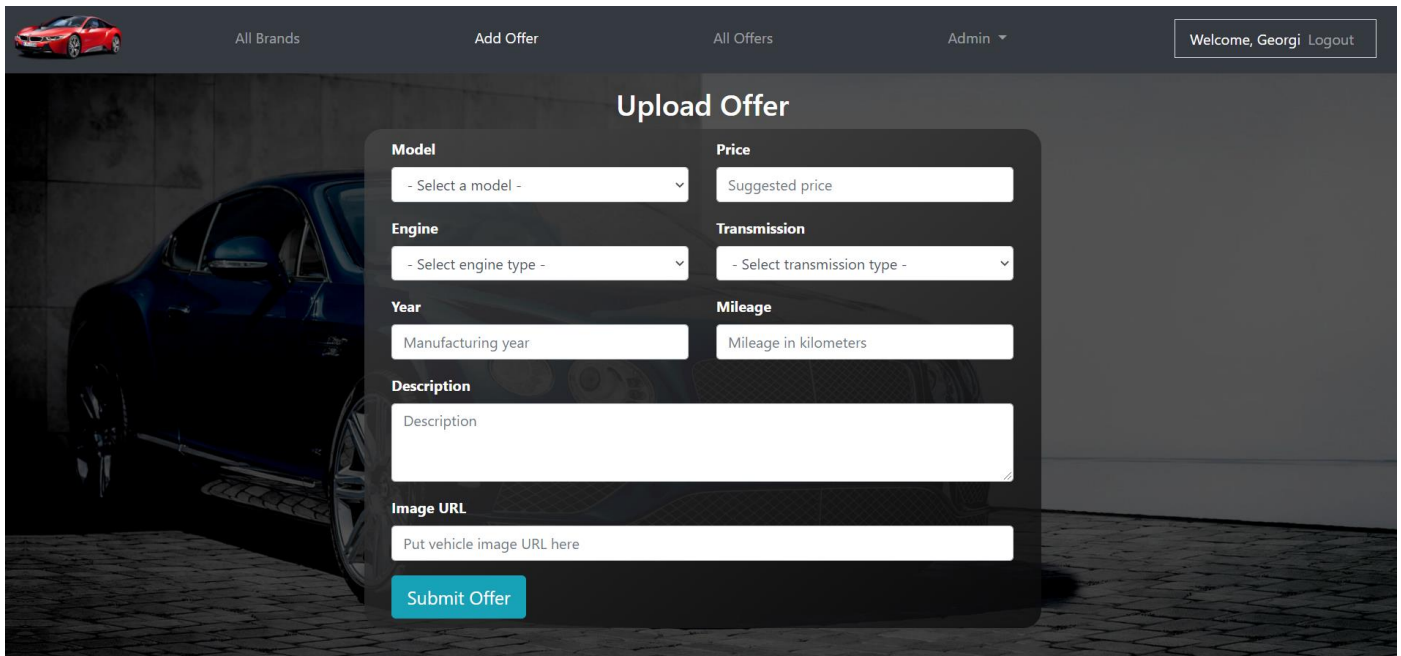
- To make menu (for example "All Brands") to be active may use this:

```
<li class="nav-item"
  th:classappend="${#request.getServletPath() == '/users/register'}? 'active'">
  <a class="nav-link" href="/users/register">Register</a>
</li>
```

8. Upload offers - route ("/offers/add").

It should only support a **GET & POST** request.

It should return the following HTML page, upon a **GET** request.



Upload Offer

Model
- Select a model -

Price
Suggested price

Engine
- Select engine type -

Transmission
- Select transmission type -

Year
Manufacturing year

Mileage
Mileage in kilometers

Description
Description


Image URL
Put vehicle image URL here

Submit Offer




9. All Offers - route ("/offers/all")

It should only support a **GET** request.

It should return the following HTML page, upon a **GET** request.



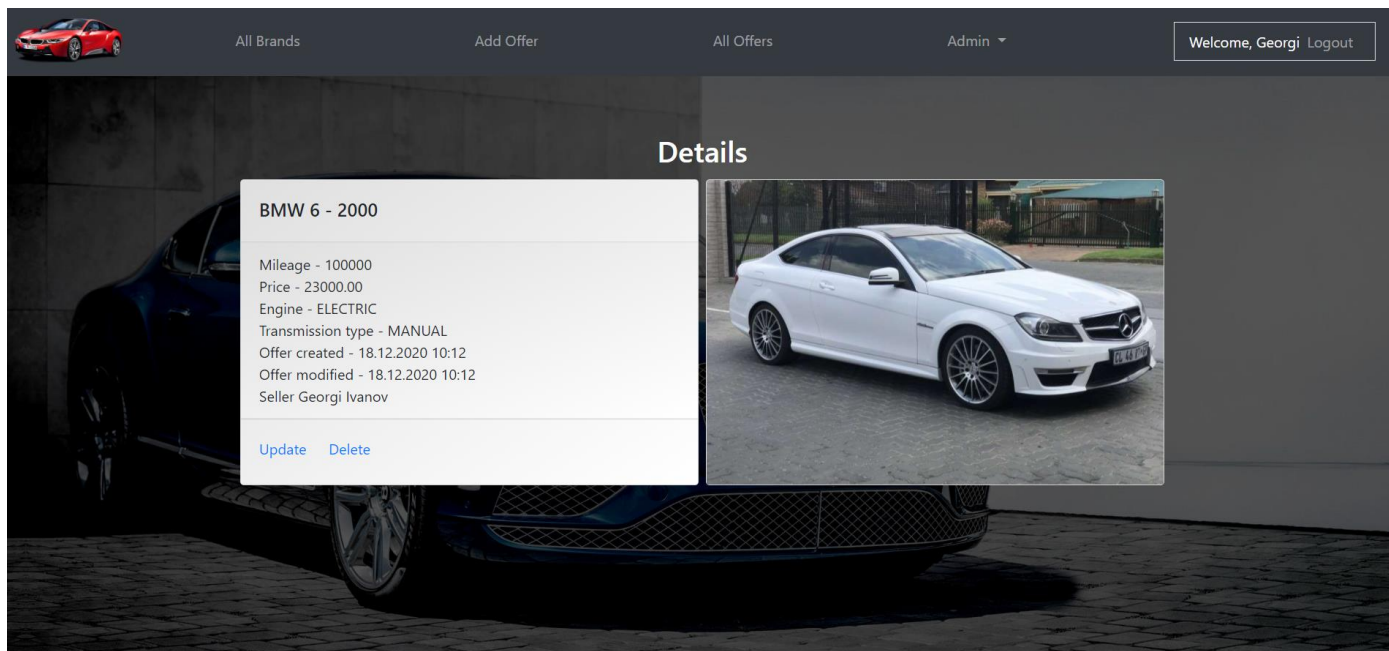
All Offers

 BMW X5 - 2000 Mileage 56000 Price 23000.00 Engine GASOLINE Transmission AUTOMATIC Details	 Opel Agila - 2000 Mileage 7000 Price 23000.00 Engine DIESEL Transmission AUTOMATIC Details	 BMW 6 - 2000 Mileage 100000 Price 23000.00 Engine ELECTRIC Transmission MANUAL Details
--	---	--

10. Offer details - route ("/offers/details")

It should only support a **GET** request.

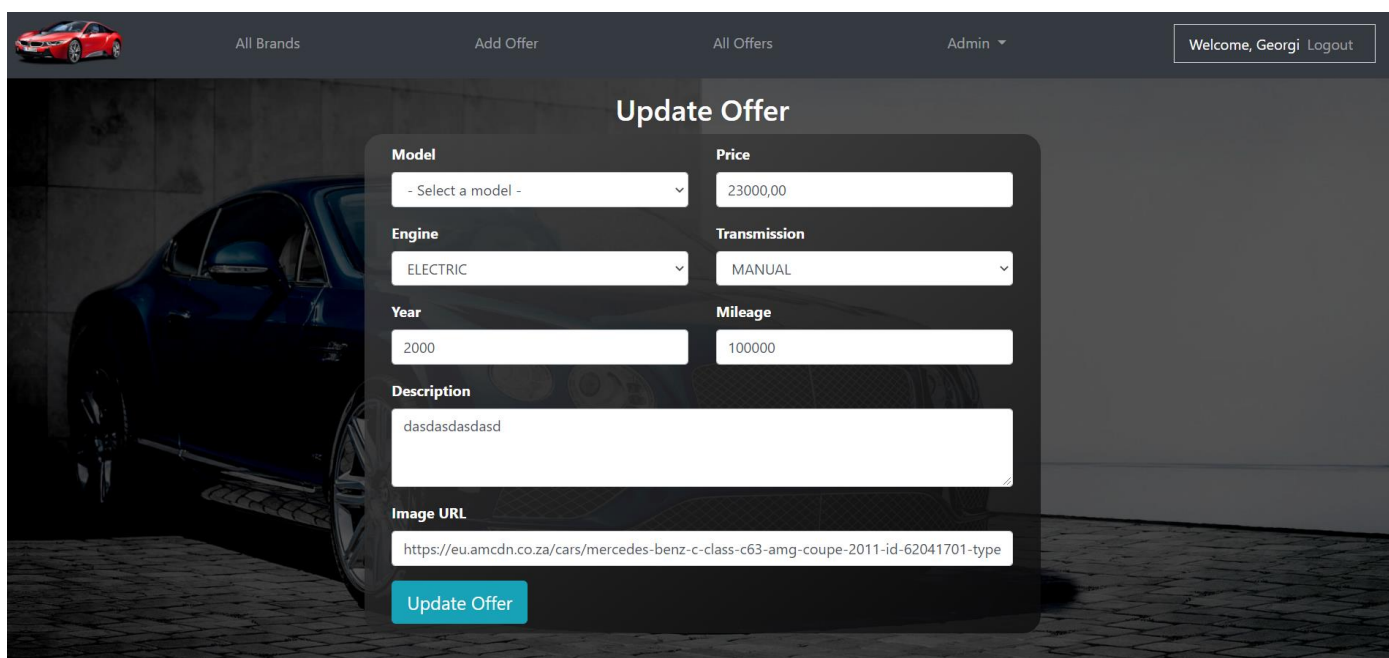
It should return the following HTML page, upon a **GET** request.



11. Update ("/offers/update")

It should support only a **GET & POST** request.

It should return the following HTML page, upon a **GET** request.



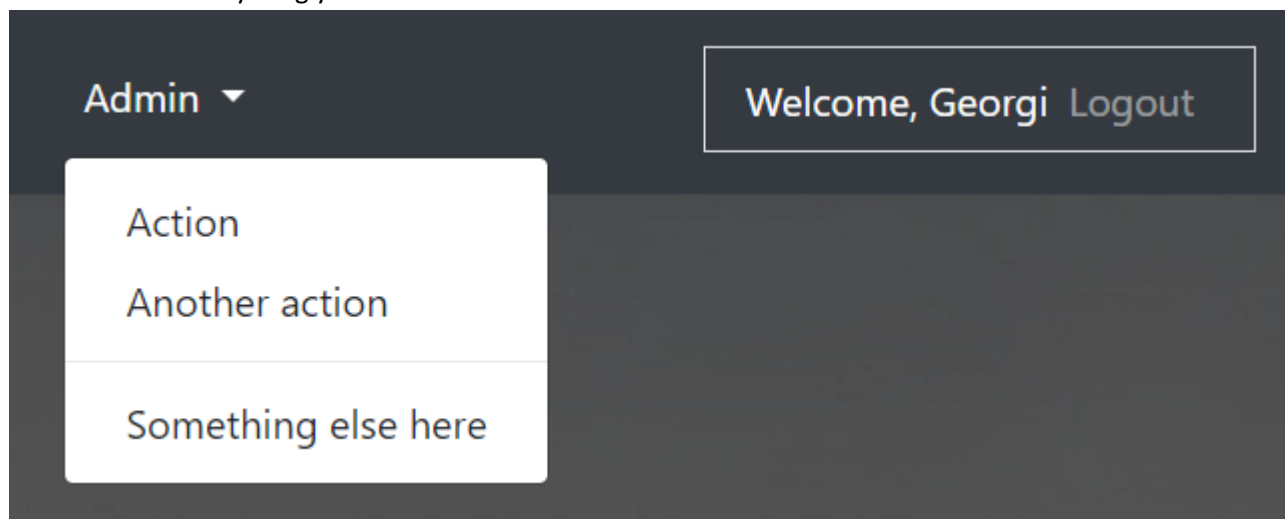
12. Delete ("/offers/delete")

Just delete the offer, after that redirect to route – "/offers/all".

13. Footer & Dropdown

You must implement your custom footer and add new admin functionalities in a drop down menu. You can implement promotions and VIP offers.

Feel free to add anything you want.



14. Security

Now you can know how to work with the Session and you can to secure all routes in the application.

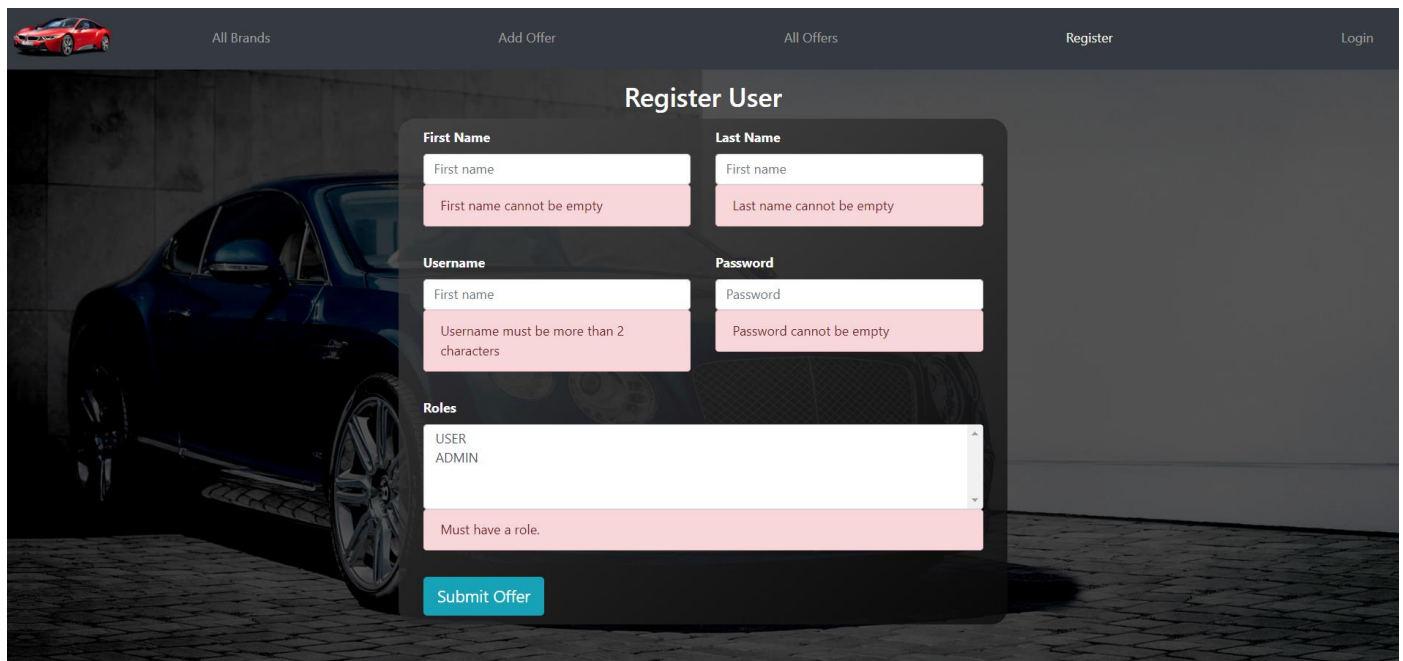
Guest can only visit the index page, login page and register page.

Only user with role admin can see the Admin dropdown with additional functionalities.

15. Better validation *

In the next lesson you will learn more about the different Thymeleaf validation ways. But if you want, you can implement the validation in this application.

- **Validations in Register page**



The screenshot shows the 'Register User' form on a website. The form is overlaid on a background image of a blue Bentley Continental GT. The form has a dark gray header with the title 'Register User'. Below the header, there are four input fields: 'First Name', 'Last Name', 'Username', and 'Password'. Each field has a validation error message displayed below it in a pink box. The 'First Name' error is 'First name cannot be empty'. The 'Last Name' error is 'Last name cannot be empty'. The 'Username' error is 'Username must be more than 2 characters'. The 'Password' error is 'Password cannot be empty'. Below these fields is a 'Roles' section with a dropdown menu showing 'USER' and 'ADMIN'. A validation error 'Must have a role.' is displayed below the dropdown. At the bottom of the form is a blue 'Submit Offer' button.

Register User

First Name
First name
First name cannot be empty

Last Name
First name
Last name cannot be empty

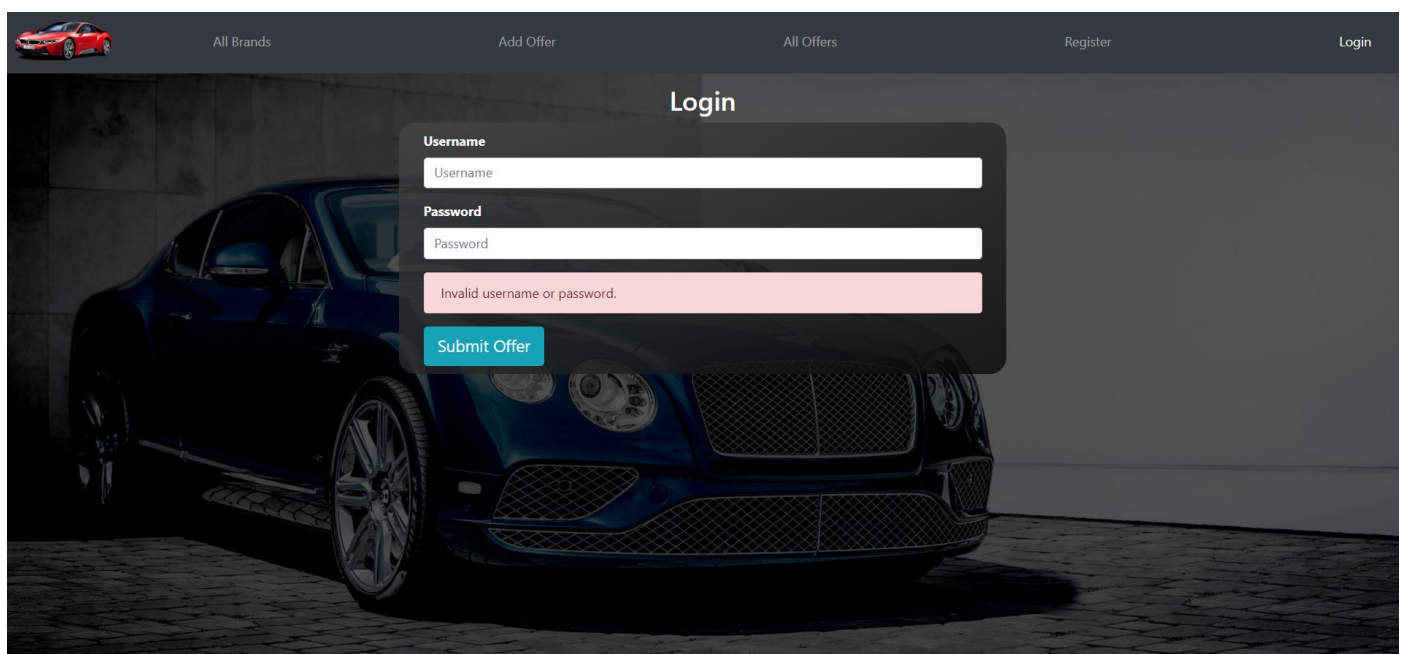
Username
First name
Username must be more than 2 characters

Password
Password
Password cannot be empty

Roles
USER
ADMIN
Must have a role.

Submit Offer

- **Validations in Login page**



The screenshot shows the 'Login' form on a website. The form is overlaid on a background image of a blue Bentley Continental GT. The form has a dark gray header with the title 'Login'. Below the header, there are two input fields: 'Username' and 'Password'. A validation error 'Invalid username or password.' is displayed below the 'Password' field in a pink box. At the bottom of the form is a blue 'Submit Offer' button.

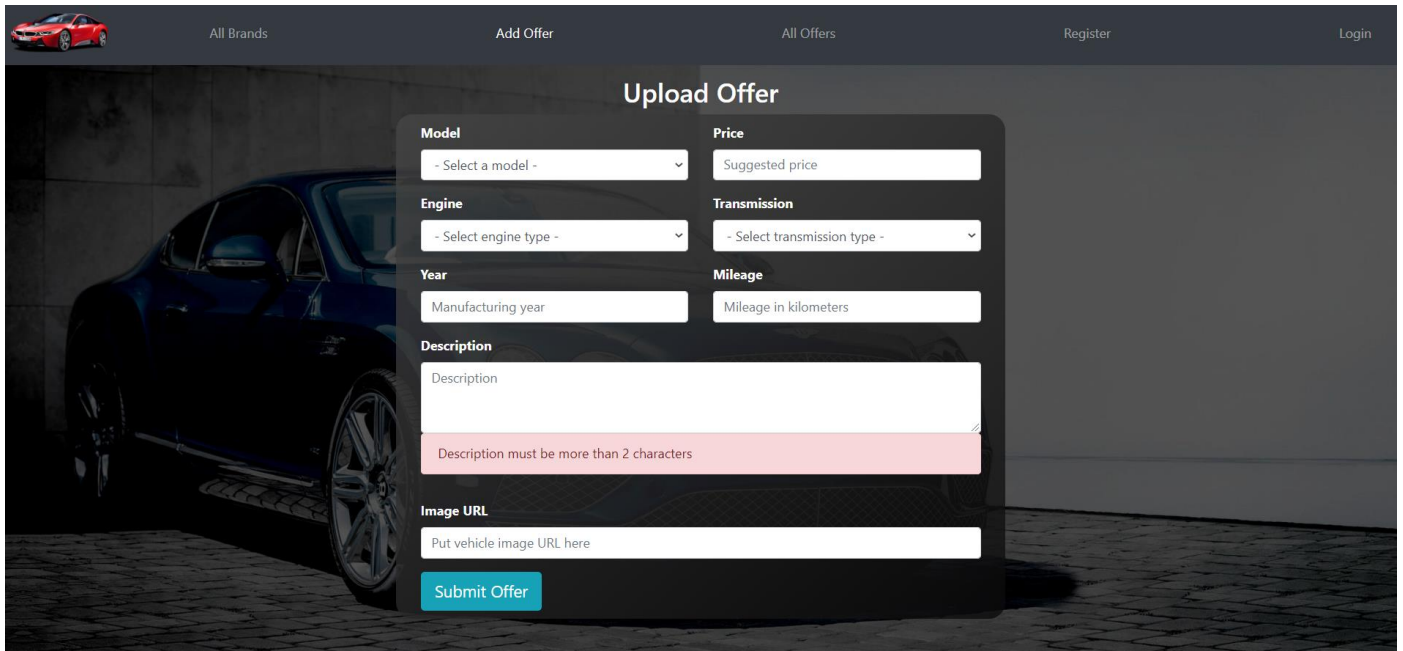
Login

Username
Username

Password
Password
Invalid username or password.

Submit Offer

- **Validations in Add Offer page**



The screenshot shows the 'Add Offer' page of a website. The page has a dark header with navigation links: 'All Brands', 'Add Offer', 'All Offers', 'Register', and 'Login'. A red sports car is visible in the top left corner. The main content area features a dark background with a blue sports car. Overlaid on this is a white 'Upload Offer' form. The form contains the following fields:

- Model:** A dropdown menu with the placeholder text '- Select a model -'.
- Price:** A text input field with the placeholder text 'Suggested price'.
- Engine:** A dropdown menu with the placeholder text '- Select engine type -'.
- Transmission:** A dropdown menu with the placeholder text '- Select transmission type -'.
- Year:** A text input field with the placeholder text 'Manufacturing year'.
- Mileage:** A text input field with the placeholder text 'Mileage in kilometers'.
- Description:** A text area with the placeholder text 'Description'. Below the text area, a red error message states: 'Description must be more than 2 characters'.
- Image URL:** A text input field with the placeholder text 'Put vehicle image URL here'.

At the bottom of the form is a blue 'Submit Offer' button.