# Lab: Spring Introduction MVC

## 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:

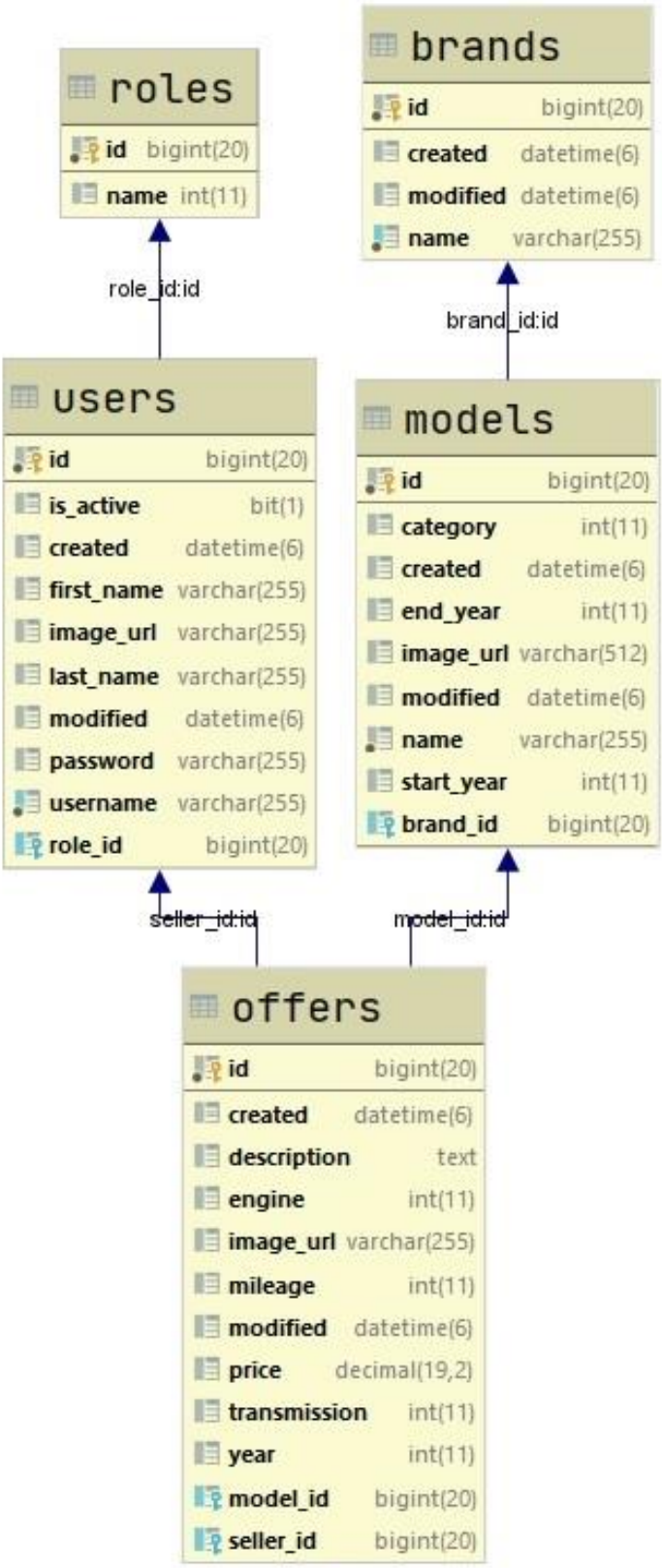
- **`id`** – **uuid or number**.
- **`username`** – username of the **user**.
- **`password`** – password of the **user**.
- **`firstName`** – first name of the **user**.
- **`lastName`** – last name of the **user**.
- **`isActive`** – **true OR false**.
- **`role`** – **user's role (User or Admin)**.
- **`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**



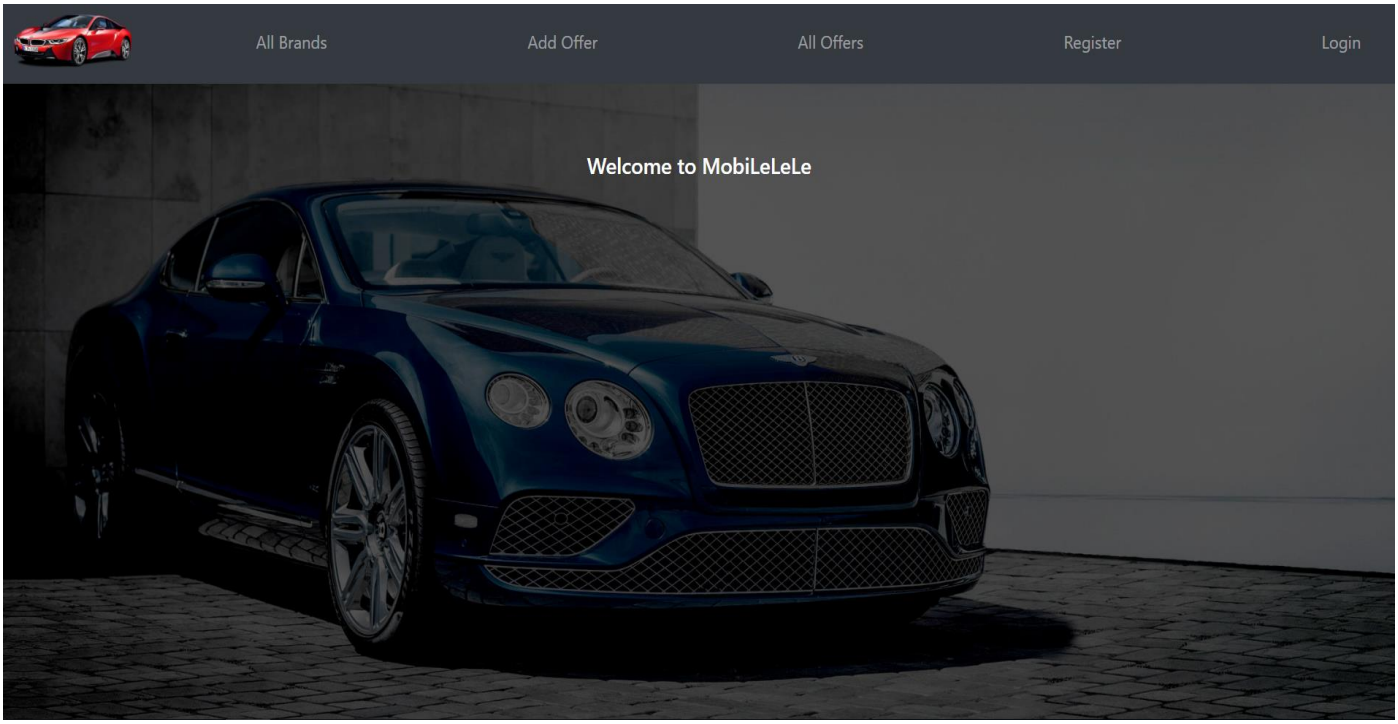
## 2. Populate DB

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.

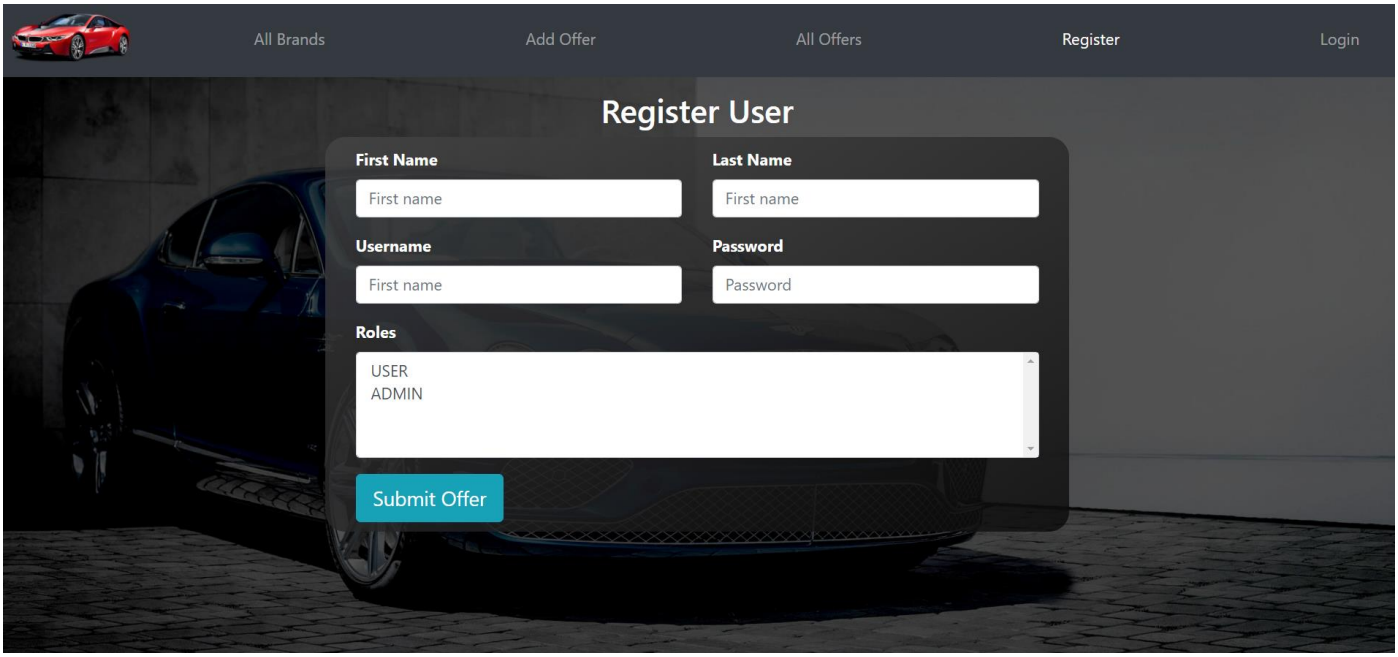



## 4. Register User - route ("/users/register").

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">
```

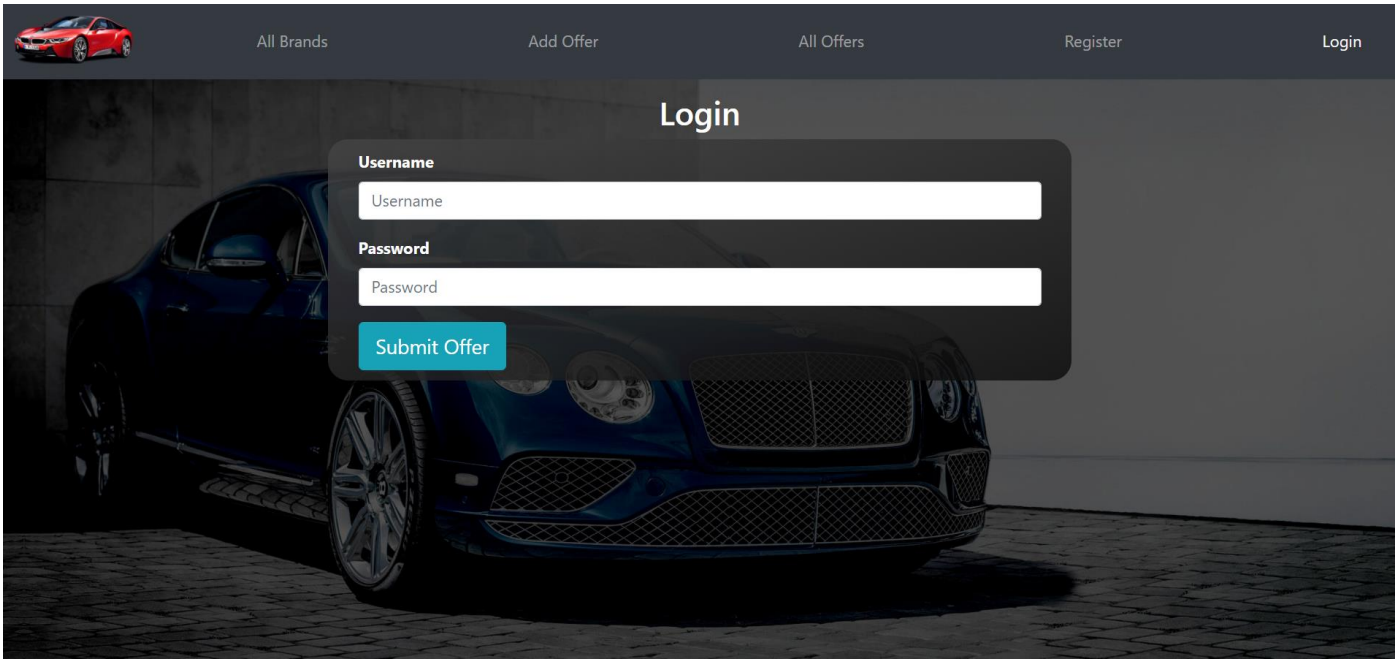
- Also you need to add in are the html form action and method (remember last lecture)**:**

```
<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.

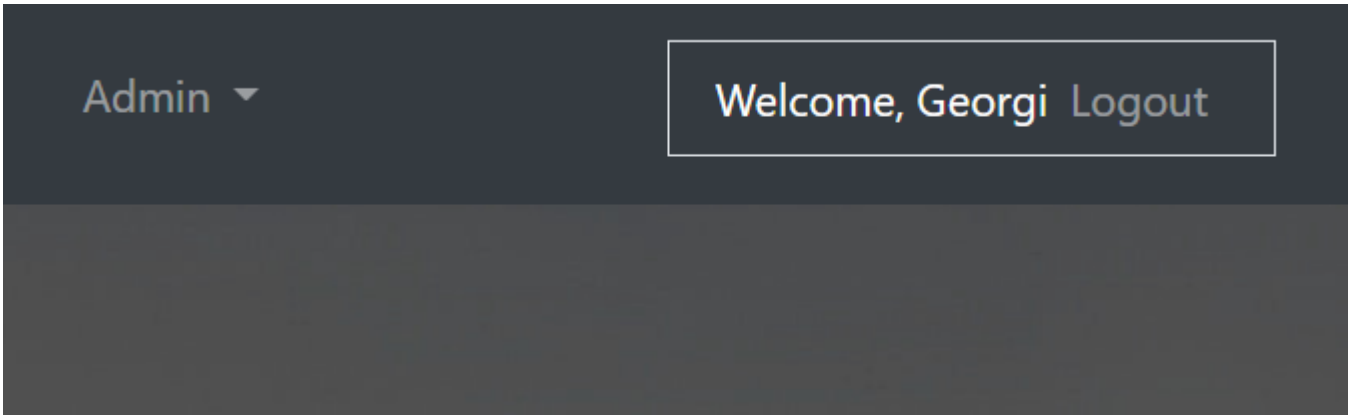
Also, if he has an Admin role, he can see the Admin dropdown.

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>
```

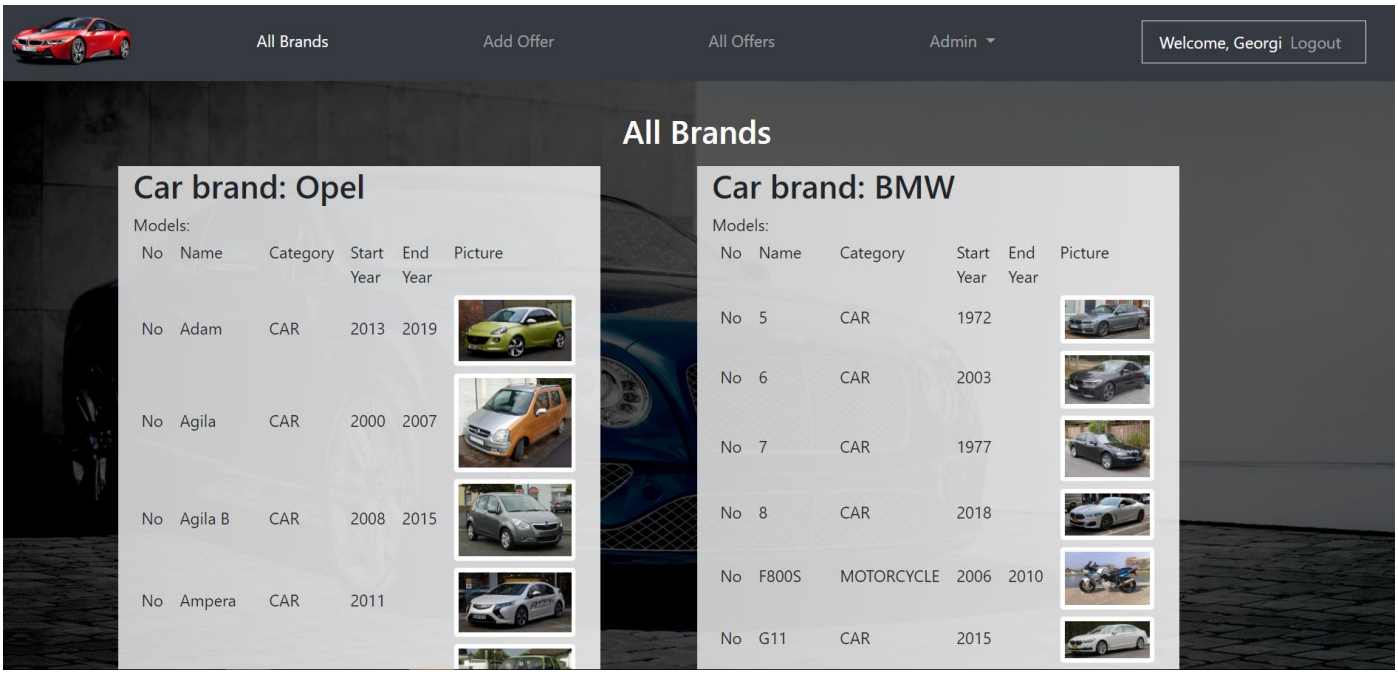
Expected result for login user



# 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.




We continue with more functionality on the next lab. ☺