# Exercise: Spring-Boot and Spring-MVC

# Pathfinder

Because Lucho and Chocho love nature and walks very much, they decided to create a place where people with similar interests can share roads, photos, videos, and comments. So, when a person manages to break away even for a day from the hectic daily life, he will be able to easily find a place to spend a few really energizing hours.

### Entities:

### Role

Create a Role class, which holds the following properties:

* **id** - Accepts **UUID String or Long** values
* **name** - Accepts **String** values
  + **USER, MODERATOR and ADMIN**

### User

The **User** **Entity** should hold the following properties

* **id** - Accepts **UUID String or Long** values
* **username** - Accepts **String** values
  + Accepts values, which should be at least 2 characters
* **password** - Accepts **String** values
  + Accepts values, which should be at least 2 characters
* **full name** - Accepts **String** values
  + Accepts values, which should be at least 2 characters
* **email** - Accepts **String** values
  + Accepts values, which contain the '@' symbol
* **role** - Accepts **Role Entity** values
  + Each registered user should have a "**User**" role
* **level** - Accepts **a level of the user** (BEGINNER, INTERMEDIATE, ADVANCED)

### Comments

The **Comments** **Entity** should hold the following properties

* **id** - Accepts **UUID String or Long** values
* **approved** - Accepts **boolean** values
* **created** - Accepts **Date and Time** values
  + The values should not be future dates
* **text content** - Accepts **very long text** values
* **author** -Accepts **User Entities** as values
* **route** -Accepts **Route Entities** as values

### Pictures

The **Pictures** **Entity** should hold the following properties

* **id** - Accepts **UUID String or Long** values
* **title** - Accepts **String** values
* **url** - Accepts **very long String** values
* **author** -Accepts **User Entities** as values
* **route** -Accepts **Route Entities** as values

### Route

The **Route** **Entity** should hold the following properties

* **id** - Accepts **UUID String or Long** values
* **gpx coordinates** - Accepts **very long text** values
* **level** - Accepts the **levels of the routes** (**BEGINNER**, **INTERMEDIATE**, **ADVANCED**) as values
* **name** - Accepts **String** values
* **author** -Accepts **User Entities** as values
* **video url** – Accepts the **ids of youtube videos** as values

### Categories

The **Categories** **Entity** should hold the following properties

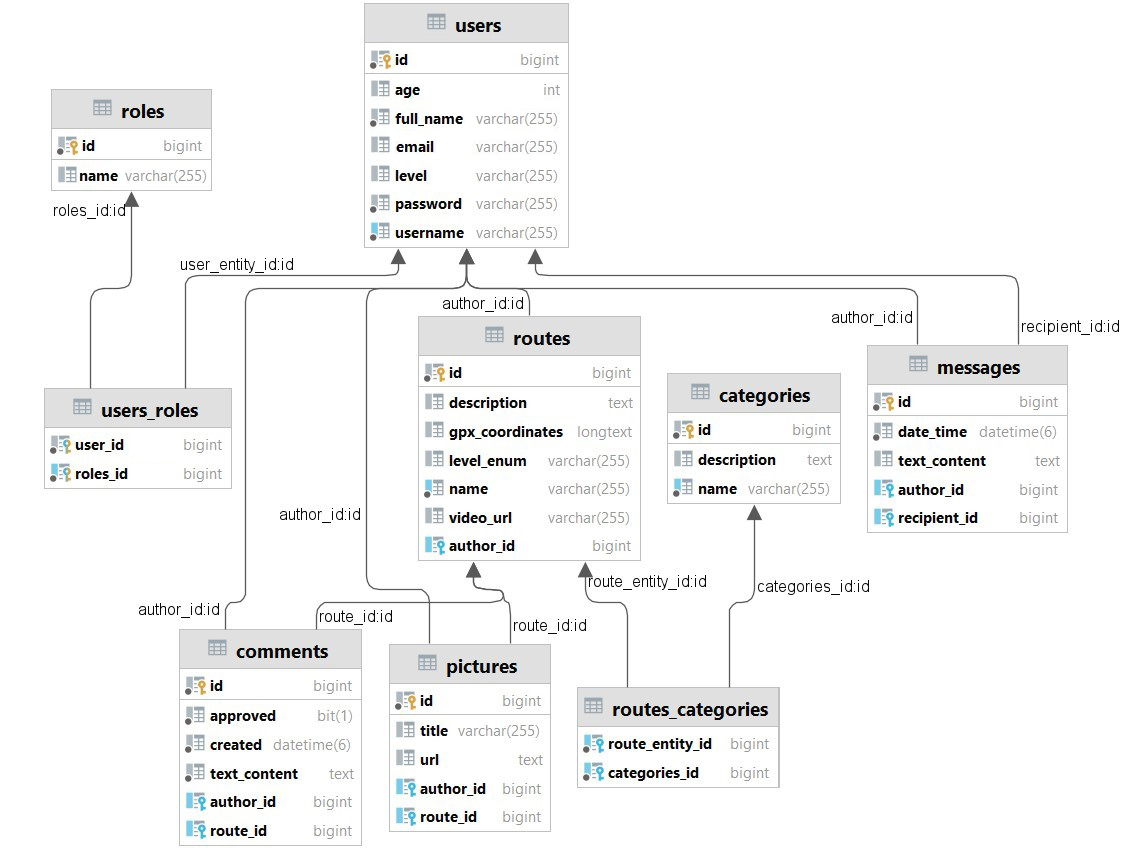
* **id** - Accepts **UUID String or Long** values
* **name** - Accepts **String** values (PEDESTRIAN, BICYCLE, MOTORCYCLE, CAR)
* **description** - Accepts **very long String** values

### Messages

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

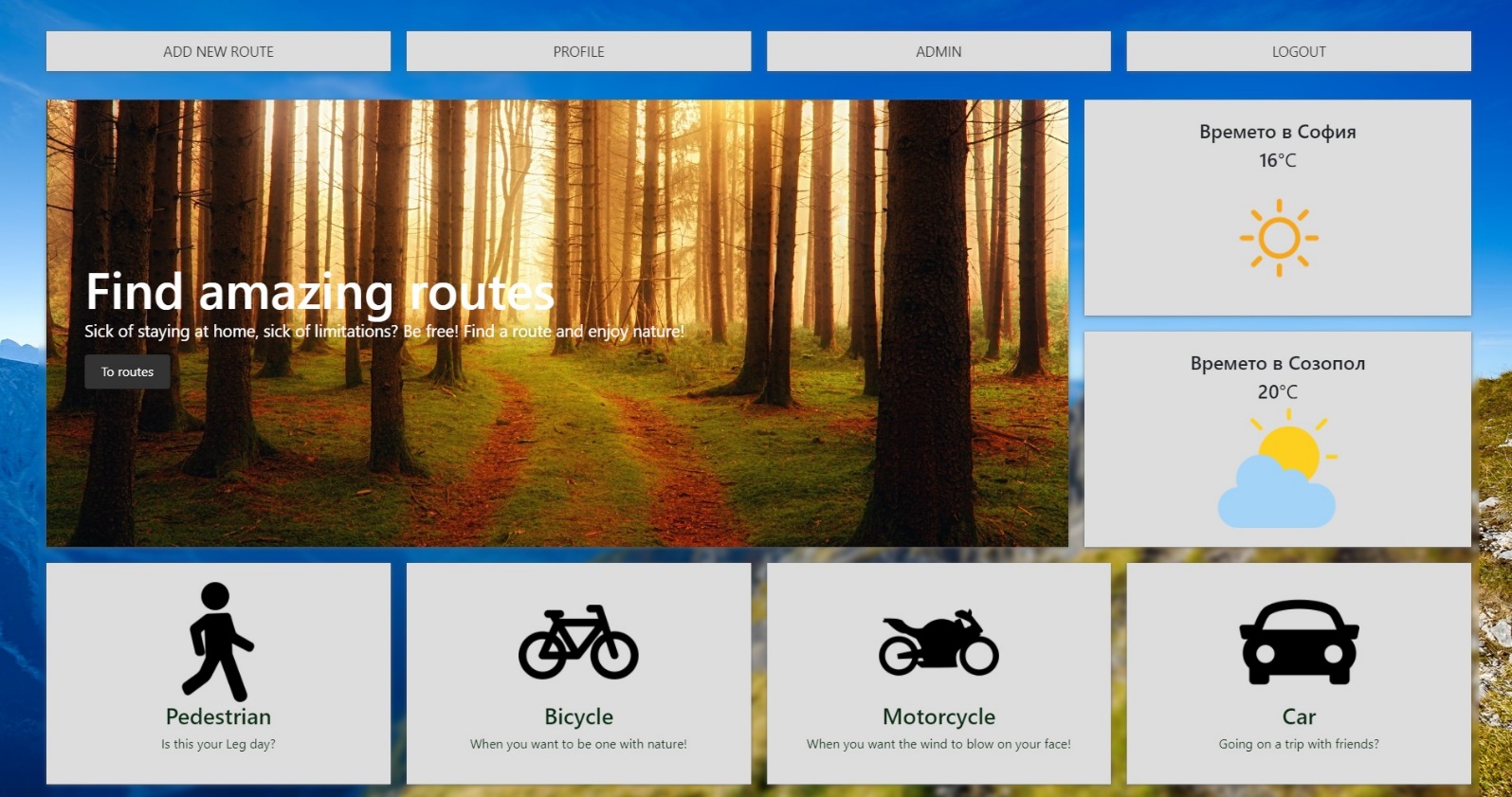
* **id** - Accepts **UUID String or Long** values
* **date time** - Accepts **Date and Time** values
* **text content** - Accepts **very long String** values
* **author** -Accepts **User Entities** as values
* **recipient** -Accepts **User Entities** as values

**Example for ER Diagram**

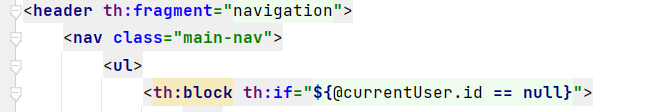


## The Index Page - ("/")

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

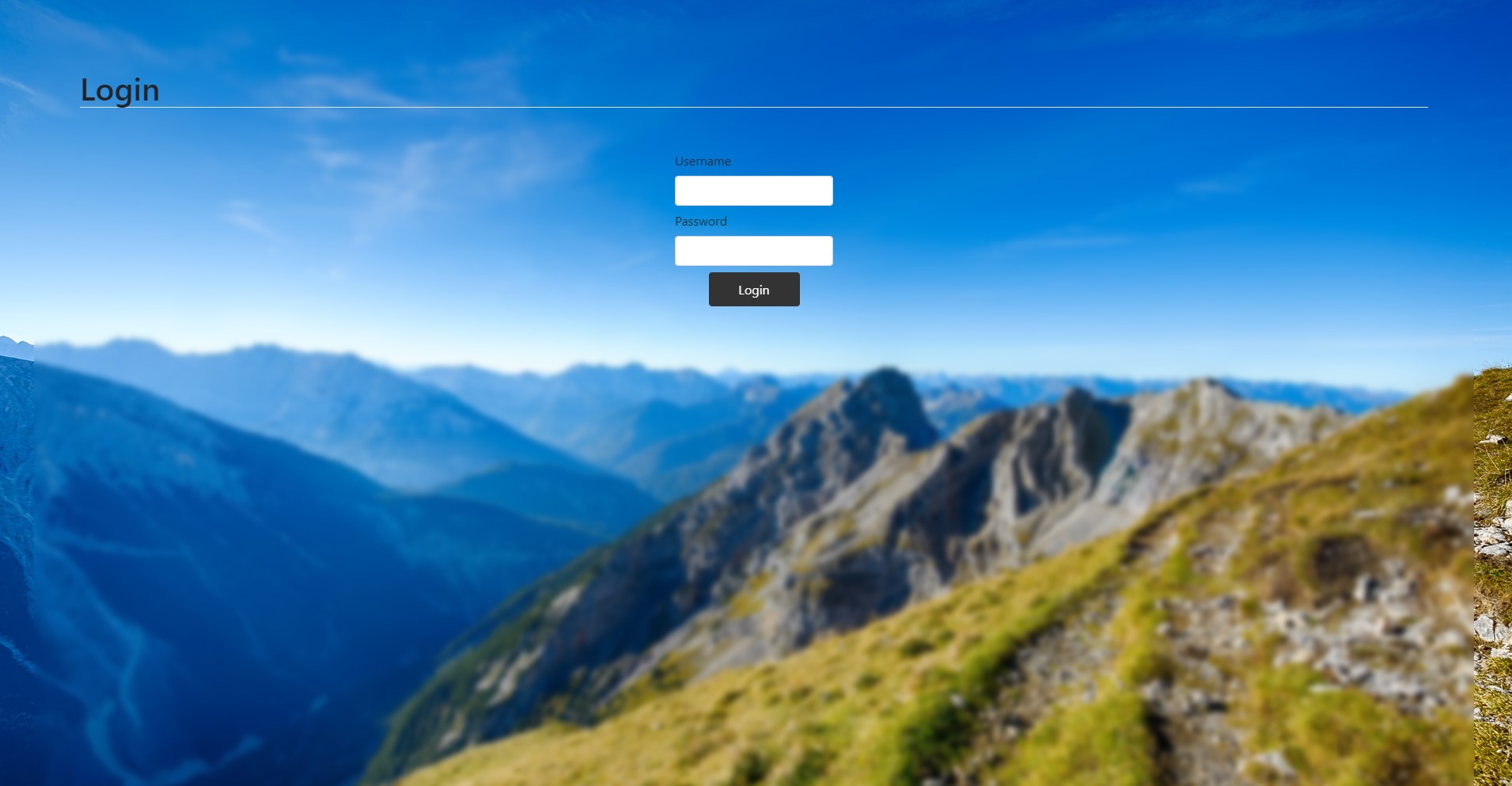


* Let’s create our fragments
  + This is the example of navigation fragment



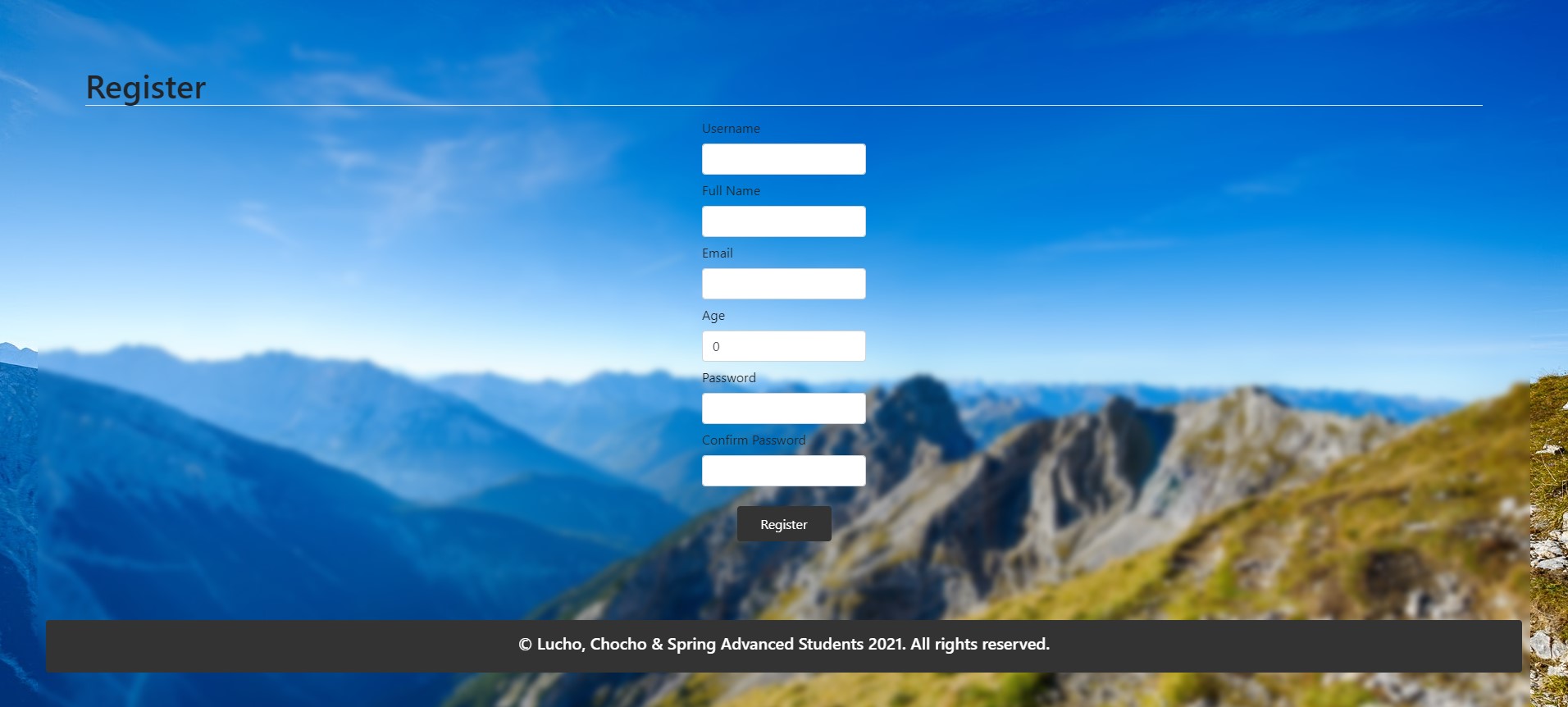
## The Login Page - ("/users/login").

* It should support **GET** & **POST** requests.
* It should return the following HTML page, upon a **GET** request.
* When login successfully, redirect to the "**/home**" page.

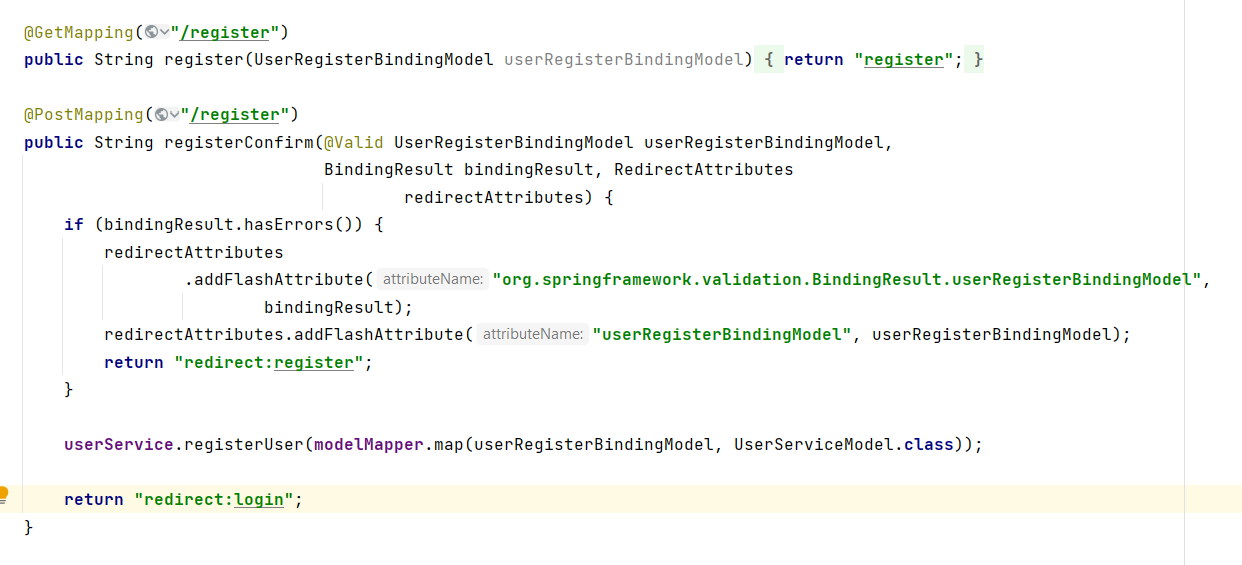


## The Register Page - ("/users/register").

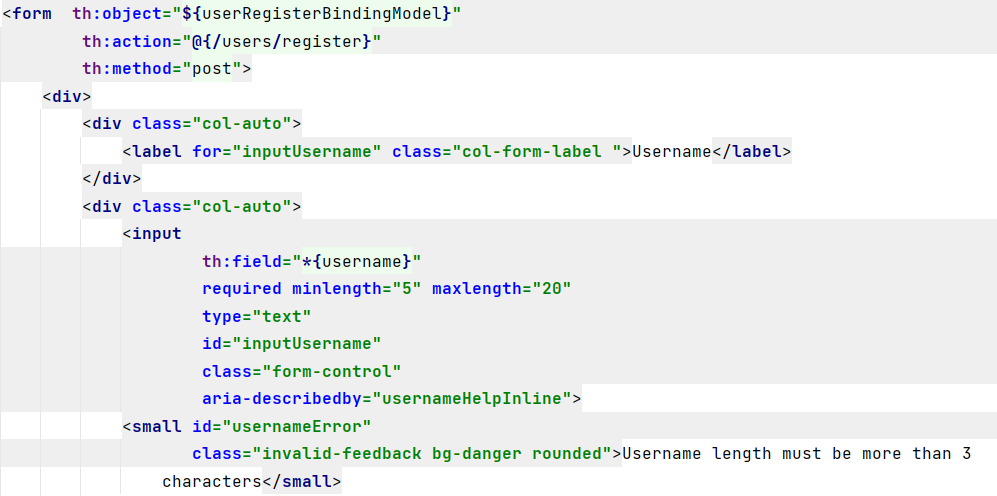
* It should support **GET** & **POST** requests.
* It should return the following HTML page, upon a **GET** request.
* When register successfully redirect to "**/users/login**".



* Example of the UsersController @PostMapping()
  + When the binding model have errors, then we must redirect again to the register page, also we must to keep the date after that.



* Example of the register.html template
  + Attach object to the form
  + Add action and method
  + Select the fields



## Navigation

* If the user is not authenticated



* If the authenticated user is not an admin

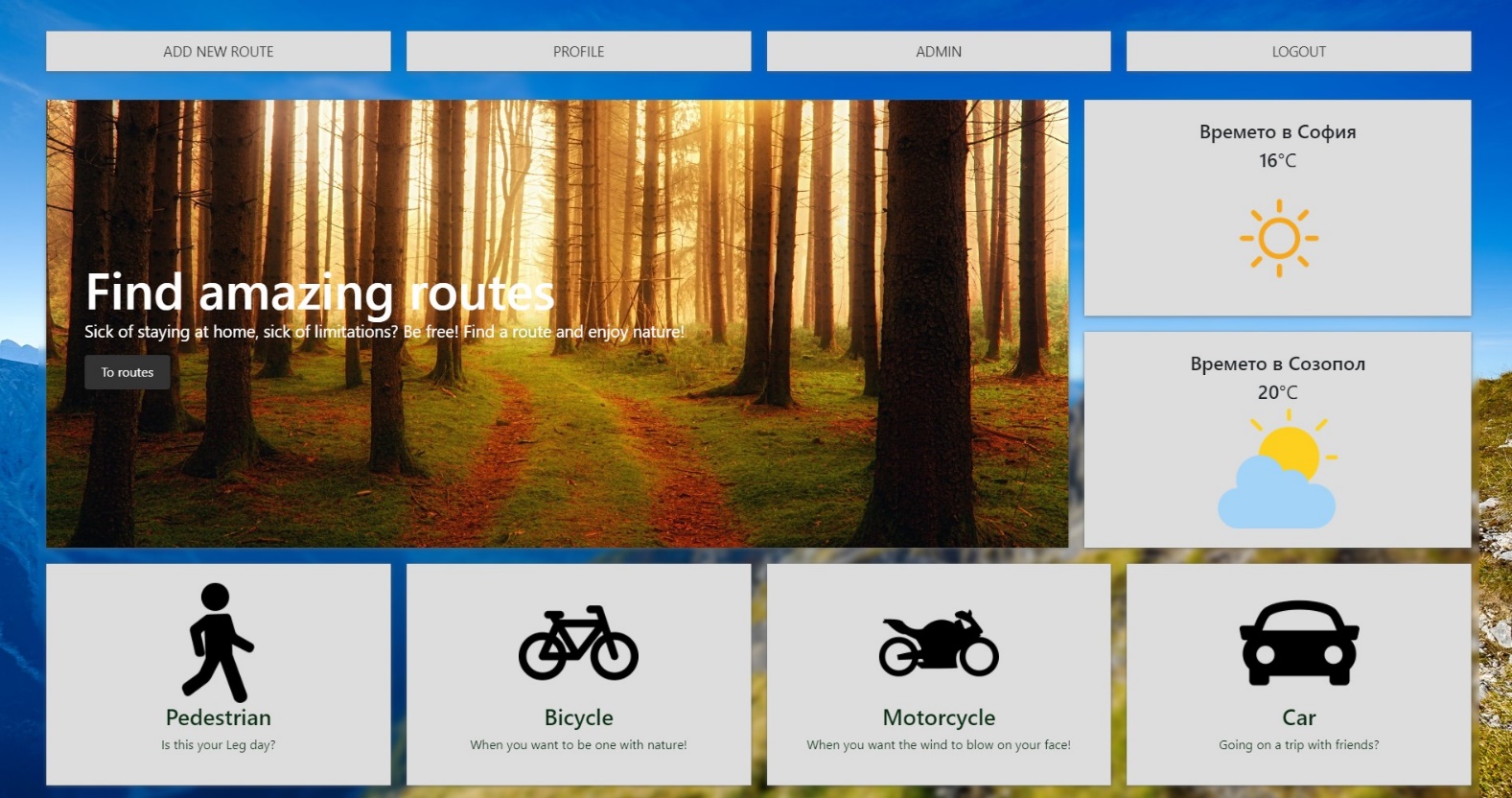


* If the authenticated user is an admin

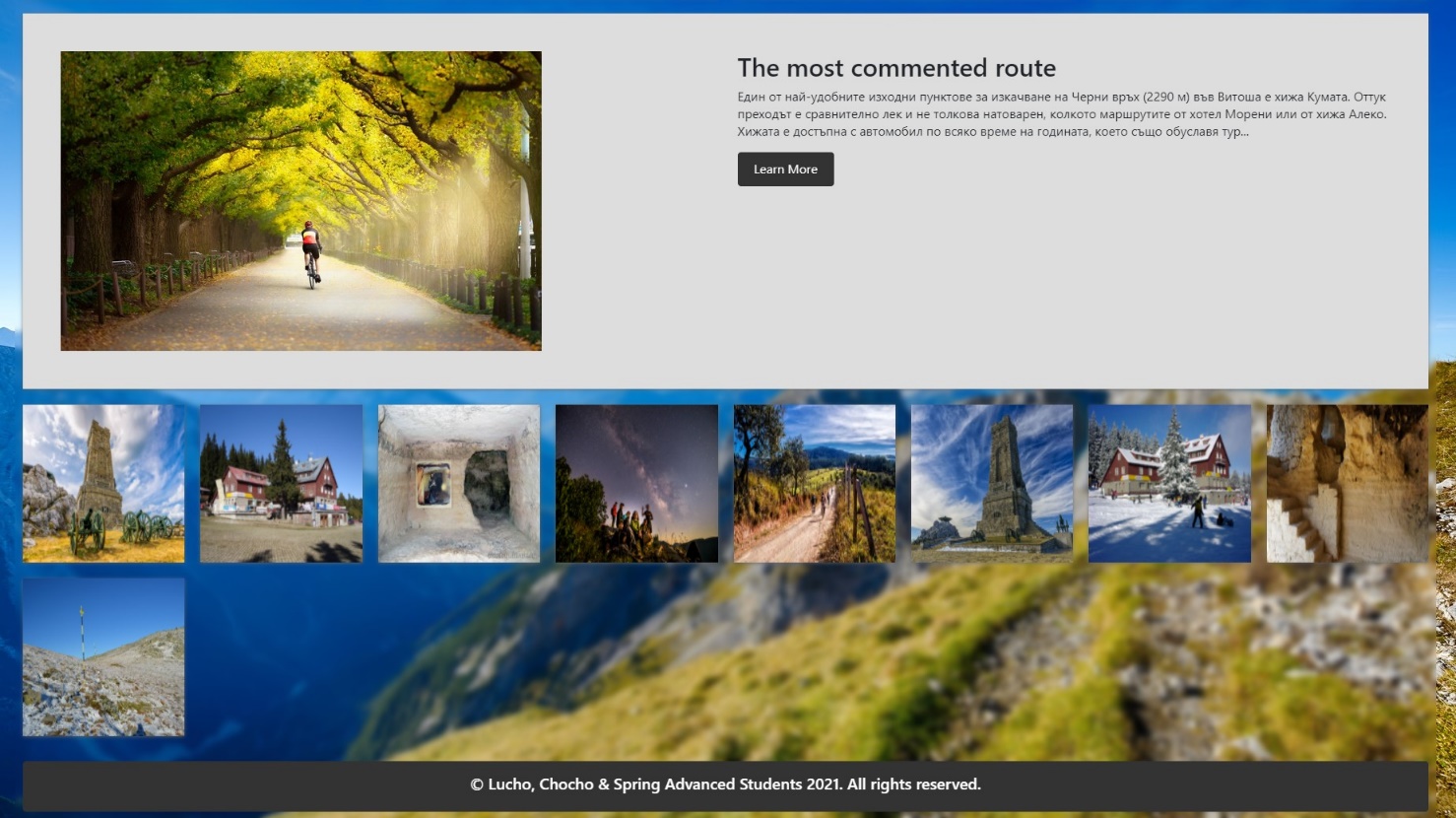


## The Home Page - ("/home").

* It should support a **GET** request.
* It should return the following HTML page, upon a GET request.
* For now, just create home page for the logged in user, who are not admins.
* Later we will explain all for this page in details, for now just show it to the user on this route.

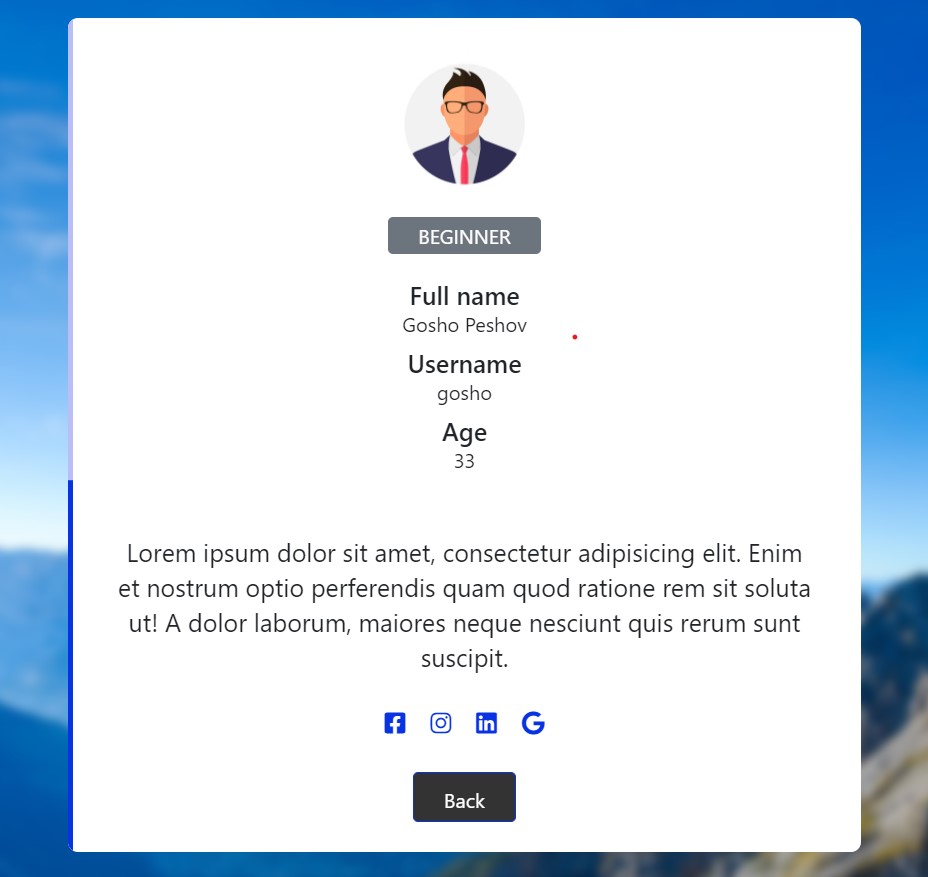


* And the second part of the index page.



## The Profile Page - ("/users/profile").

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



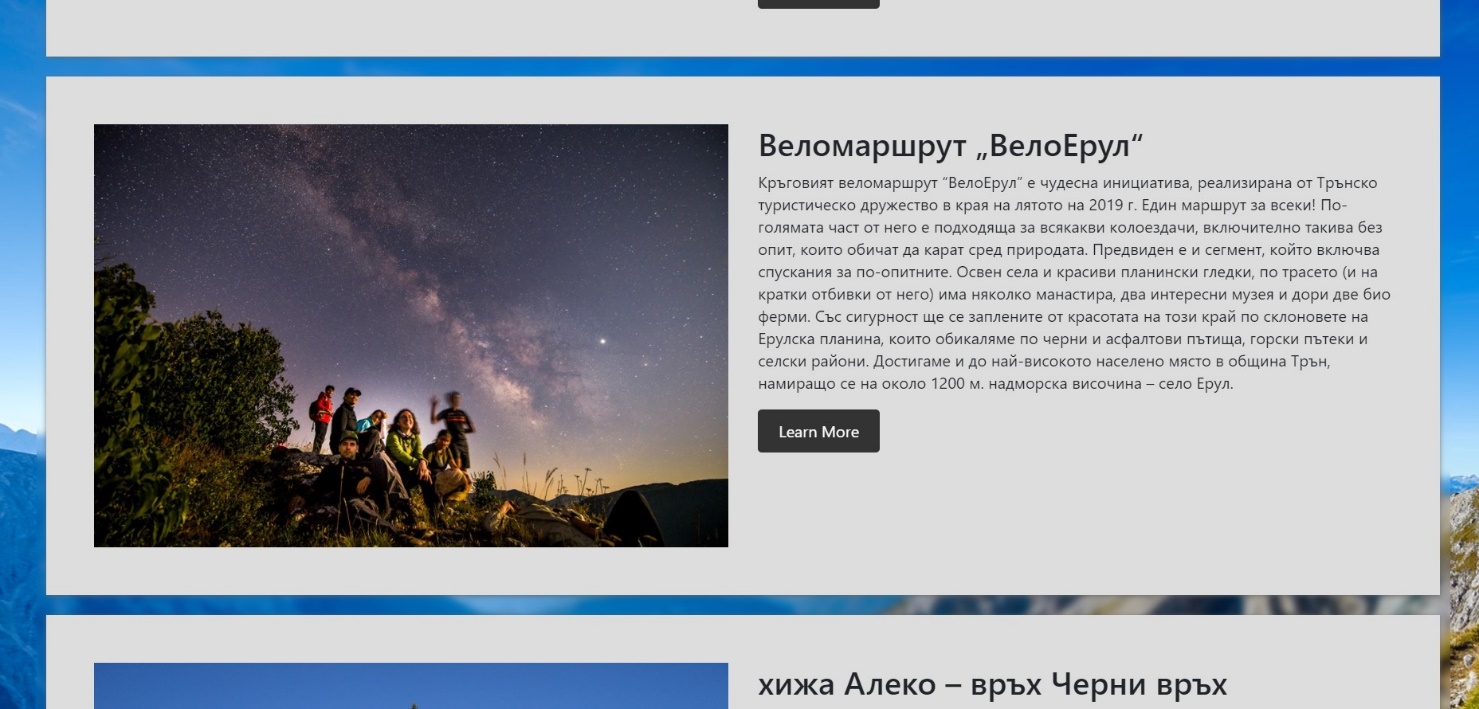
* Using path variable to send the id of the logged in user

Graphical user interface, text, application

Description automatically generated

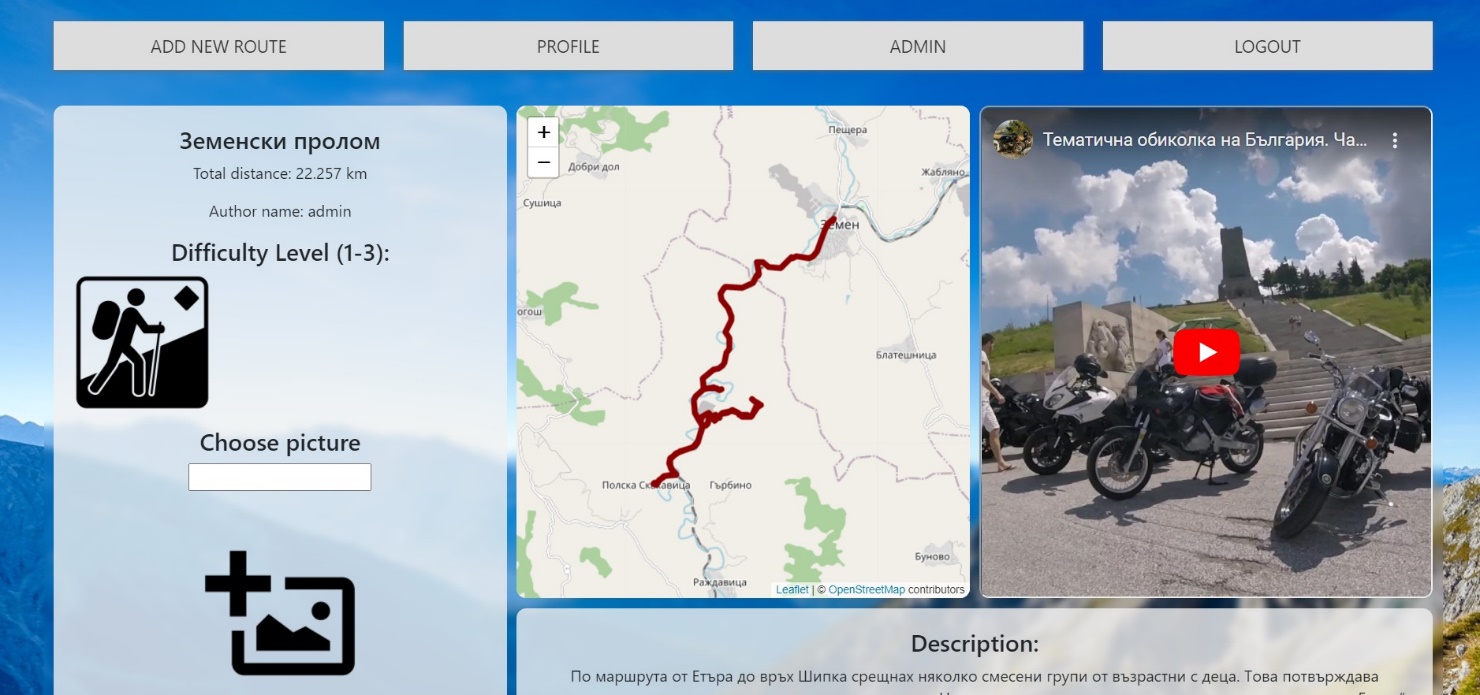
## All routes page - route ("/routes ").

* It should support a **GET** request.
* It should return the following HTML page, upon a GET request
* Lists all routes in the DB



## The Route Details Page - ("/routes/details/{id}").

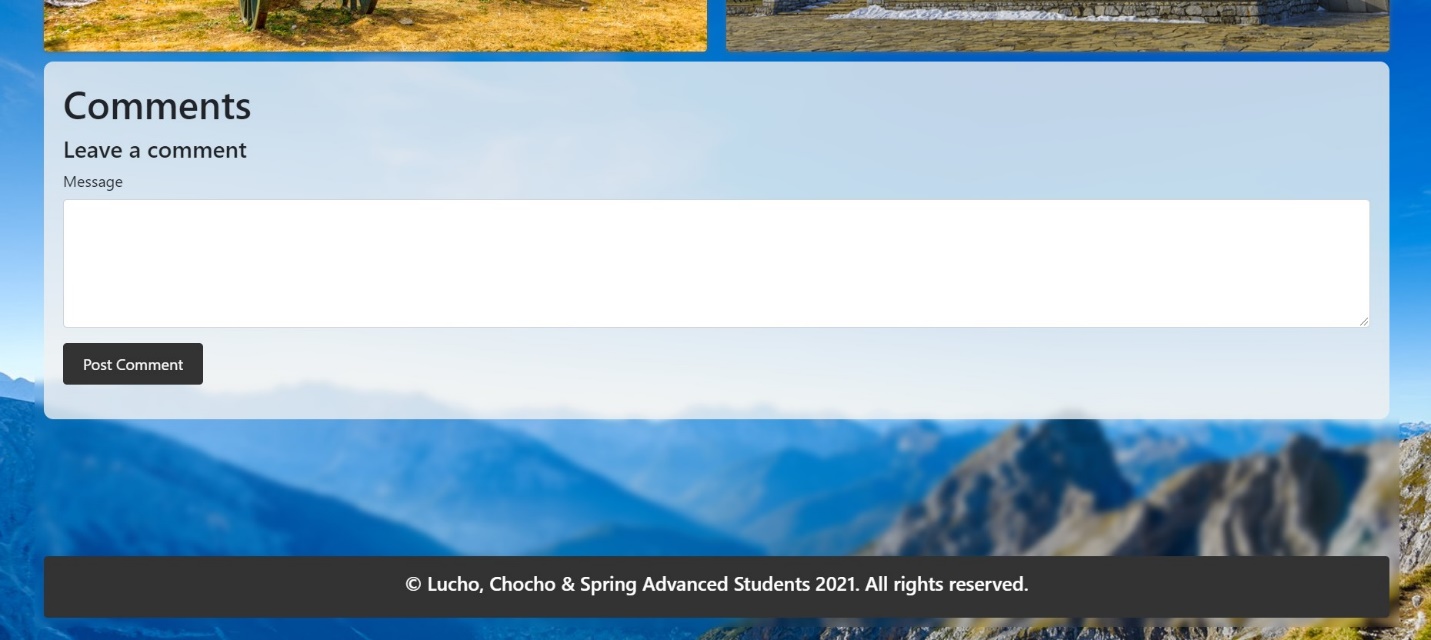
* It should support a **GET** request.
* It should return the following HTML page, upon a GET request



* The second path from the details page



* The third path from the details page



## The work of workshop will continue in the next course 😊