# Workshop-Authentication-and-GitHub

# 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

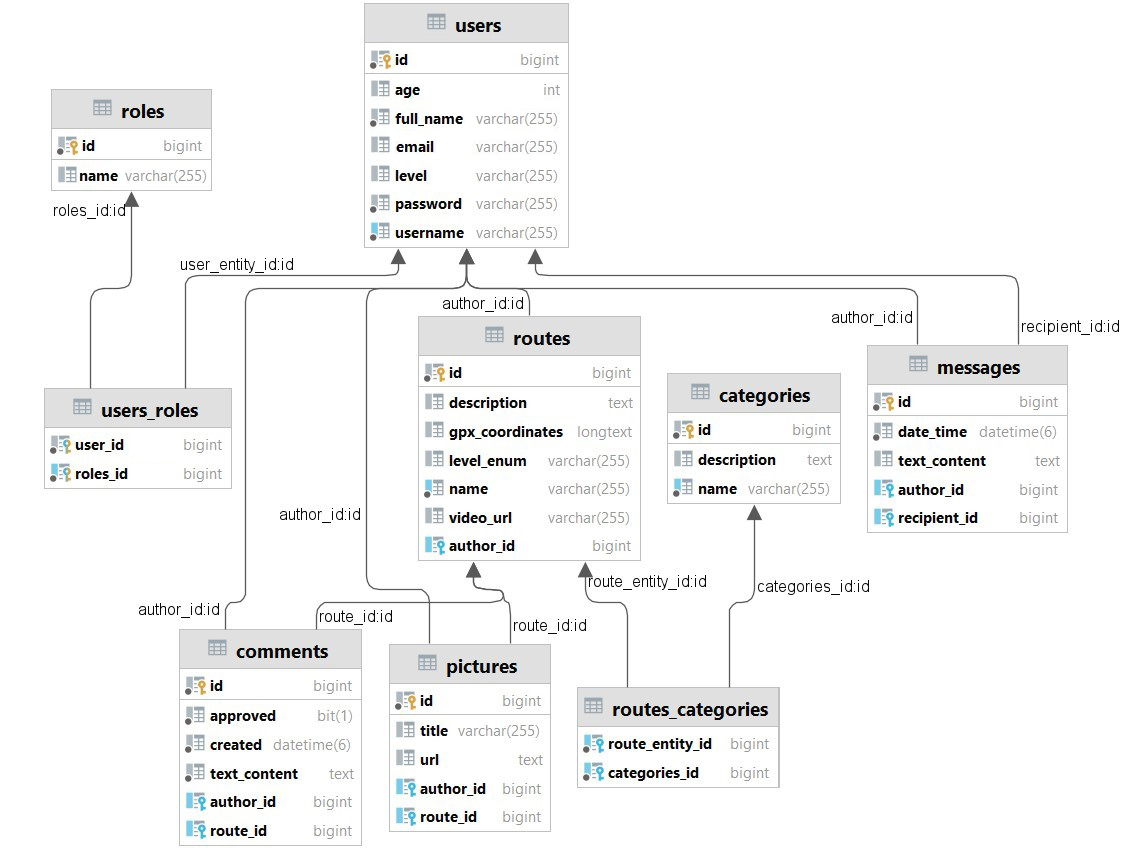
* **id** - Accepts **UUID String or Long** values
* **name** - Accepts **String** values (PEDESTRIAN, BICYCLE, MOTORCYCLE, CAR)
* **description** - Accepts **very long String** values

### Messages

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

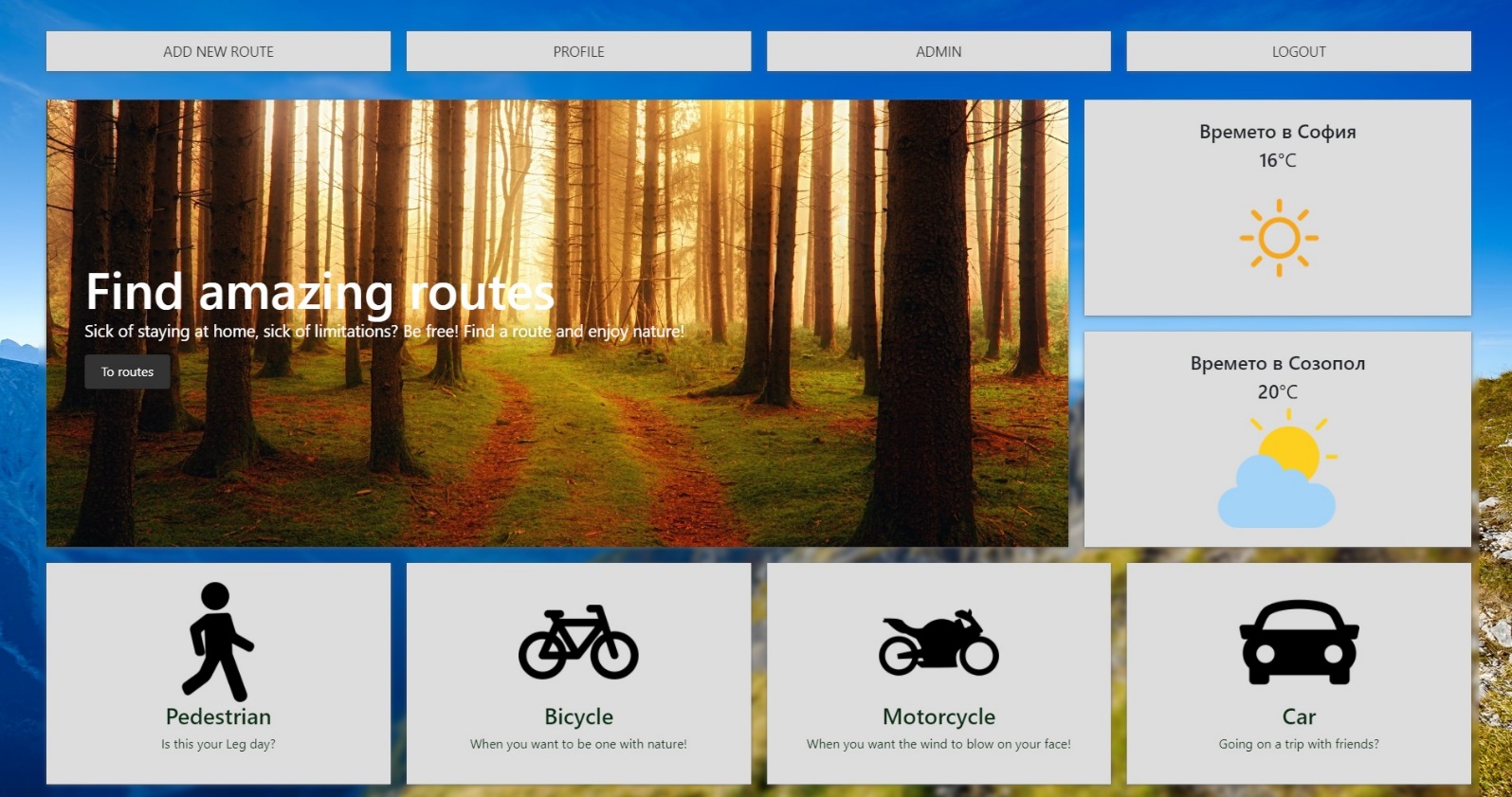
* **id** - Accepts **UUID String or Long** values
* **date time** - Accepts **Date and Time** values
* **text content** - Accepts **very long String** values
* **author** -Accepts **User Entities** as values
* **recipient** -Accepts **User Entities** as values

**Example for ER Diagram**

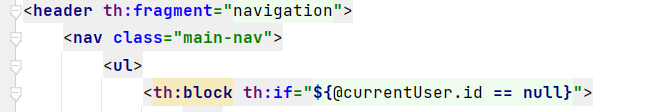


## The Index Page - ("/")

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

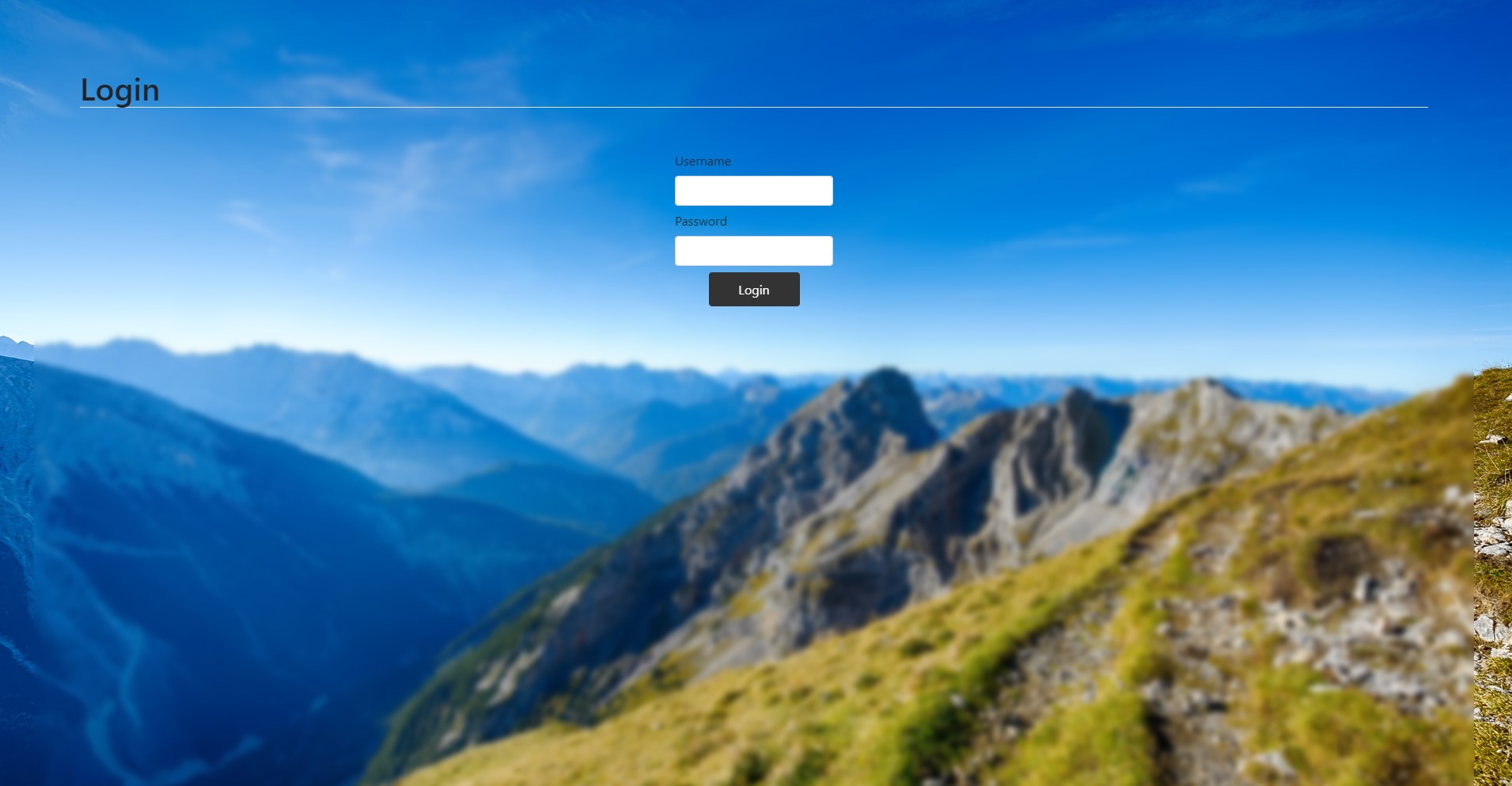


* Let’s create our fragments
  + This is the example of navigation fragment



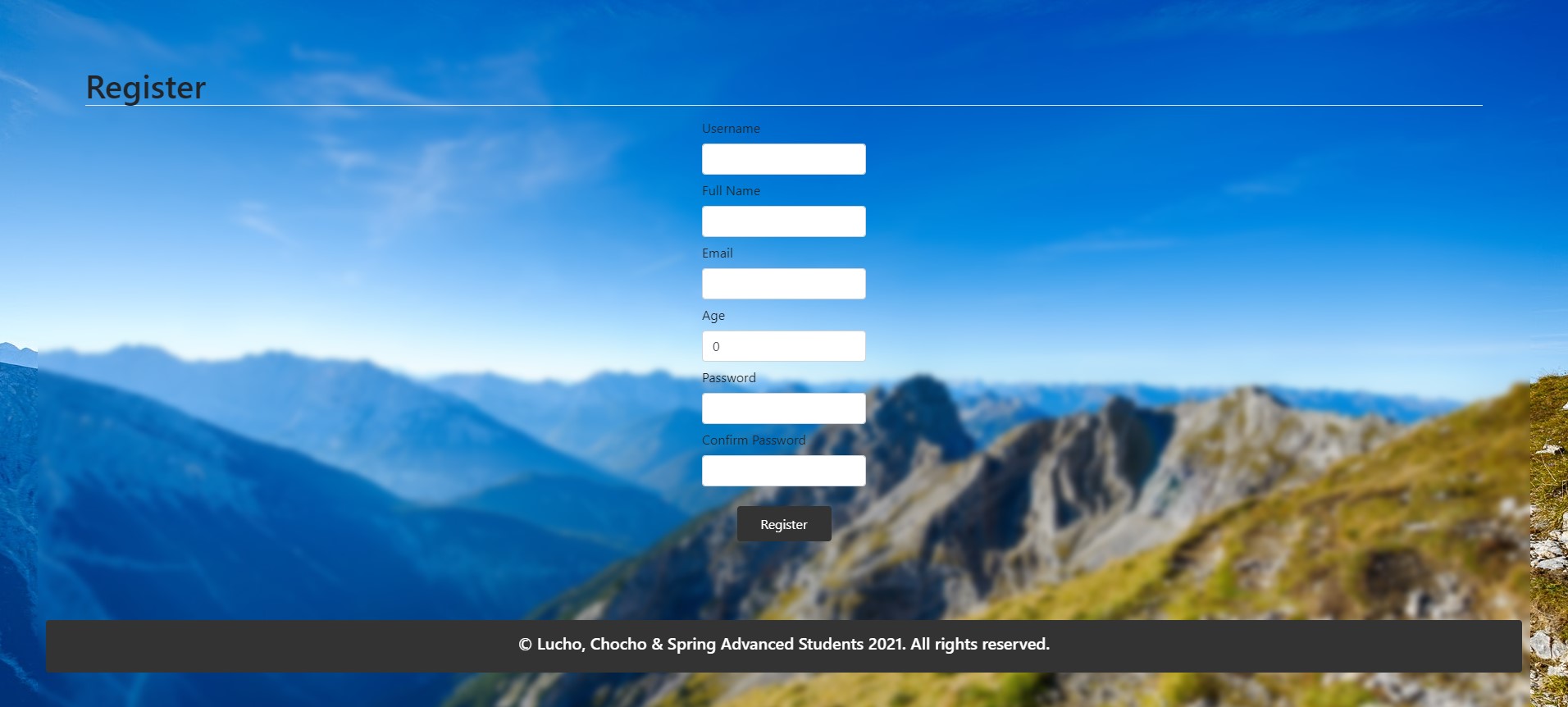
## The Login Page - ("/users/login").

* 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.

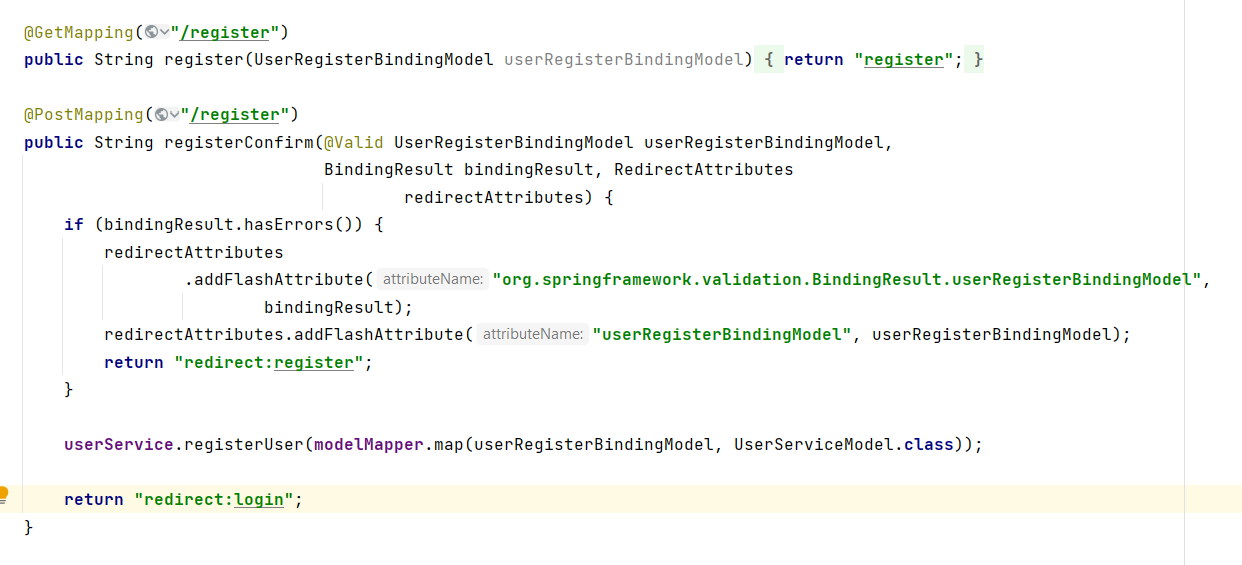


## The Register Page - ("/users/register").

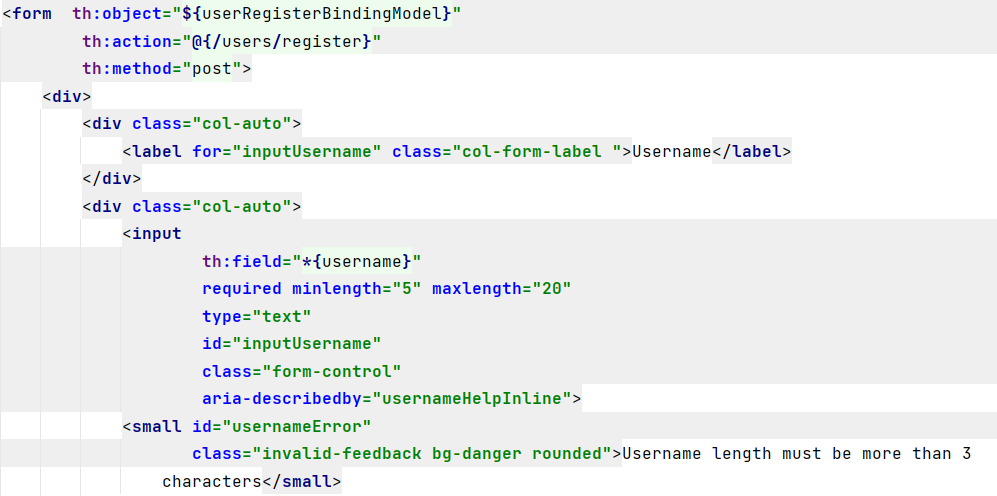
* It should support **GET** & **POST** requests.
* It should return the following HTML page, upon a **GET** request.
* When register successfully redirect to "**/users/login**".



* 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



## What is GitHub

GitHub is a web-based platform and service that hosts software development projects using the Git version control system. It offers a range of features that facilitate collaboration, code management, and project organization for both open-source and private projects.

## Creating a profile on GitHub

* **Visit the GitHub Website**: Open your web browser and navigate to the GitHub website. The URL is <https://github.com>.
* **Sign Up**: On the GitHub homepage, you will find a "**Sign up**" button. Click on it to start the registration process.
* **Provide Account Details**: You will be directed to the registration page where you need to provide the following information:
* **Username**: Choose a unique username for your GitHub account. This will be used to identify you on GitHub.
* **Email Address**: Enter your email address. This will be used for account verification and communication from GitHub.
* **Password**: Create a strong password to secure your GitHub account

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

## Initialize Git Repository locally:

* Open your Spring project in IntelliJ IDEA.
* Go to the **VCS** menu.
* Select Enable **Version Control Integration**.
* Choose **Git** as the version control system.
* Click **OK** to initialize the Git repository.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

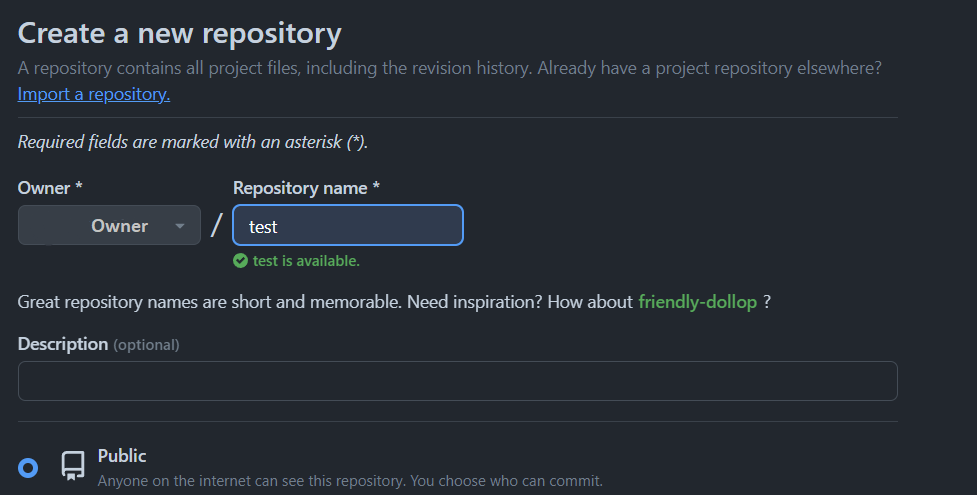
## Create a new repository in GitHub:

* **Login** to your GitHub **Account**.
* On the main page select "**New**" to create a new empty repository.

A screenshot of a computer

Description automatically generated

* Give a name to your new repository.

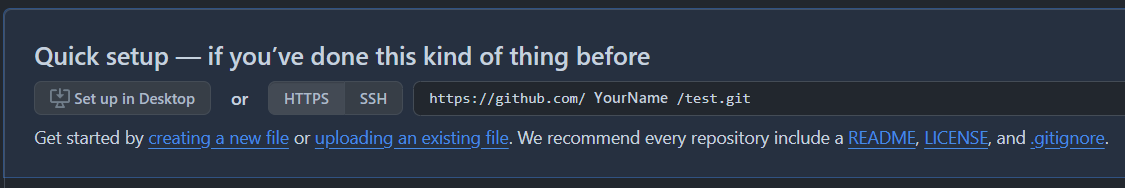


* And finally click on the "**Create repository**" button.

A screenshot of a computer

Description automatically generated

* Now copy the provided link – this is the link to your repository, keep it for later.



## Commit Changes:

* IntelliJ IDEA will show the changes in your project.
* Go to the Commit button (usually at the top left).

A screenshot of a computer

Description automatically generated

* You'll see the files that have changed. Select the files you want to commit.
* Right-click on the selected files and choose Git -> Add.
* Enter a commit message describing the changes.

A screenshot of a computer

Description automatically generated

* Click on the Commit button to commit your changes.

## Pushing commits to repository:

* Select "**master**" from the top left than "**Push**".

A screenshot of a computer

Description automatically generated

* Select "**Define remote**" from the top left then "**Push**".

A screenshot of a computer

Description automatically generated

* In the URL field paste the **link** you copied earlier from your new GitHub repository:

A screenshot of a computer

Description automatically generated

* If you succeed with pushing, you must see this window and then press "**Push**":

A screenshot of a computer

Description automatically generated

* You receive confirmation in bottom right corner of the screen:

A black screen with white text

Description automatically generated

**Now your repository is deployed**.

