# Basic CRUD – Himalayas Rescue Register



*You have received information from a Sherpa, who was on his way back from Annapurna, that your friend was seen with a group of climbers a few days ago and you need to register all the information you can gather for them and give it to the mountain rescue team. You can’t stay and help them, you have to go as soon as possible, so you’ve decided to create a program that registers climbers and keeps the needed information about them that could help in some way.*

You have been tasked to create a simple application for registering missing people in the mountains. The application should hold **mountaineers**, which are the main app **entities**. The app is called RescueRegister.

The functionality of the application should support **creating**, **listing, editing**, **deleting** mountaineers.

The application should **persist** the data into a **database**.

## Overview

Your application should be built on **one** of the **following technologies**:

### JavaScript

* **NodeJS** + **ExpressJS** frameworks
* **Handlebars.js** view engine
* **Mongoose** ORM
* **MongoDB**

### PHP

* **Symfony** framework
* **Twig** view engine
* **Doctrine** ORM
* **MySQL** database

### Java

* **Spring** framework (**Spring MVC** + **Spring Boot** + **Spring Data**)
* **Thymeleaf** view engine
* **JPA** / **Hibernate ORM** + **Spring Data** data access
* **MySQL** database

### C#

* **ASP.NET Core** framework (**ASP.NET MVC** + **Entity Framework Core**)
* **Razor** view engine
* **Entity Framework Core** ORM
* **SQL Server** database\

## Data Model

The **Mountaineer** entity holds **5 properties**:

* id – technology-dependent identifier (ObjectID for JavaScript, int for all other technologies)
* name – non-empty text
* age – non-null integer number
* gender – non-empty text
* lastSeenDate – non-empty text

## Project Skeletons

You will be given the applications’ skeletons, which hold about **90%** of the logic. You’ll be given some **files** (**controllers**, **models** etc.). The files will have **partially implemented logic**, so you’ll need to write some code for the application to **function properly**.

The application’s views will be given to you fully implemented. You only need to include them in your business logic.

**Each technology** will have its **own skeleton**, and the **different** **skeletons** may **differ** in **terms** of **what is given to you** and **what is to be implemented**.

Everything that has been given to you inside the skeleton is **correctly implemented** and if you write your code **correctly**, the application should work just fine. You are free to change anything in the Skeleton on your account.

## User Interface

This is the user interface or how the application’s pages should look in their final form (fully implemented). You have several pages, described below:

### Index Page

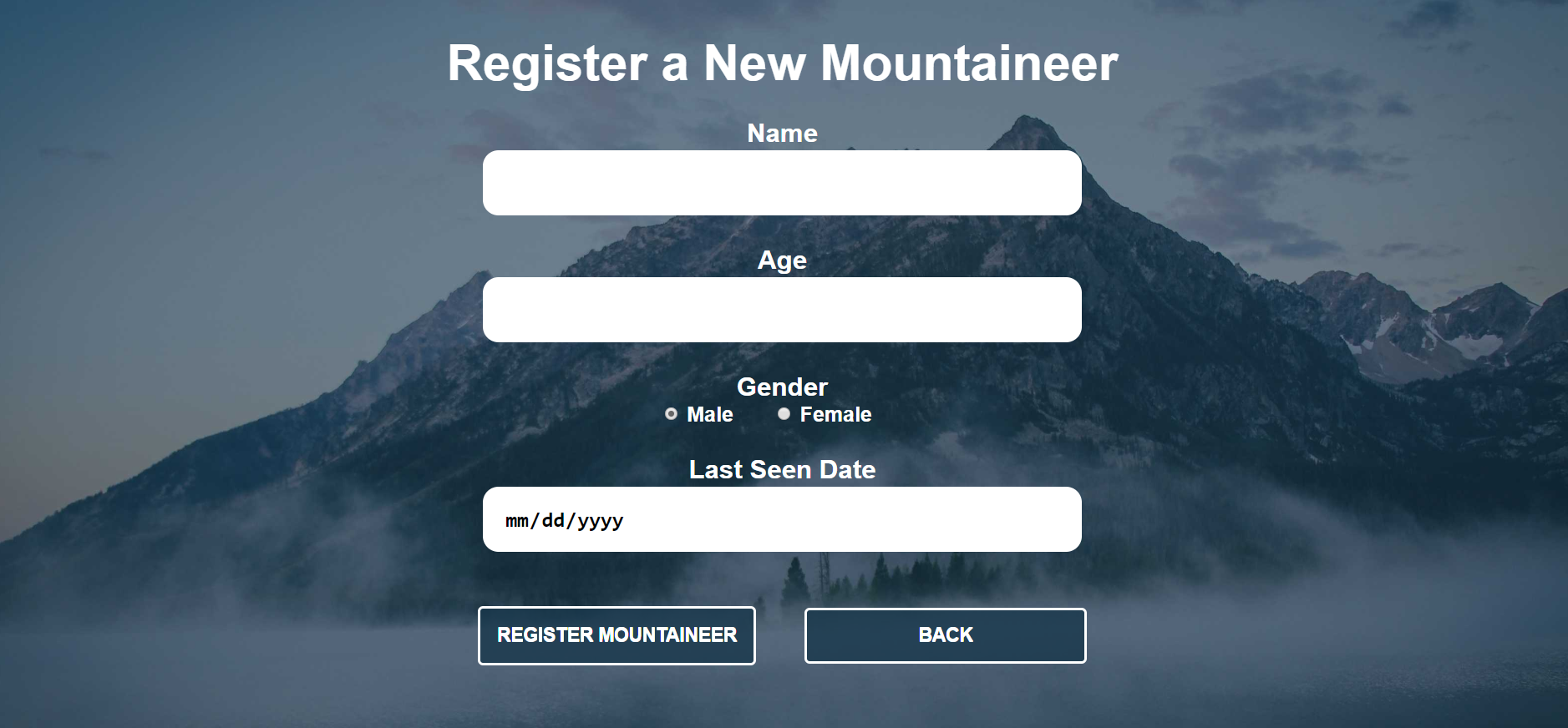
#### Route: **"/"**

Displays **all** the **mountaineers** from the database with **options** to **edit** or **delete** them.



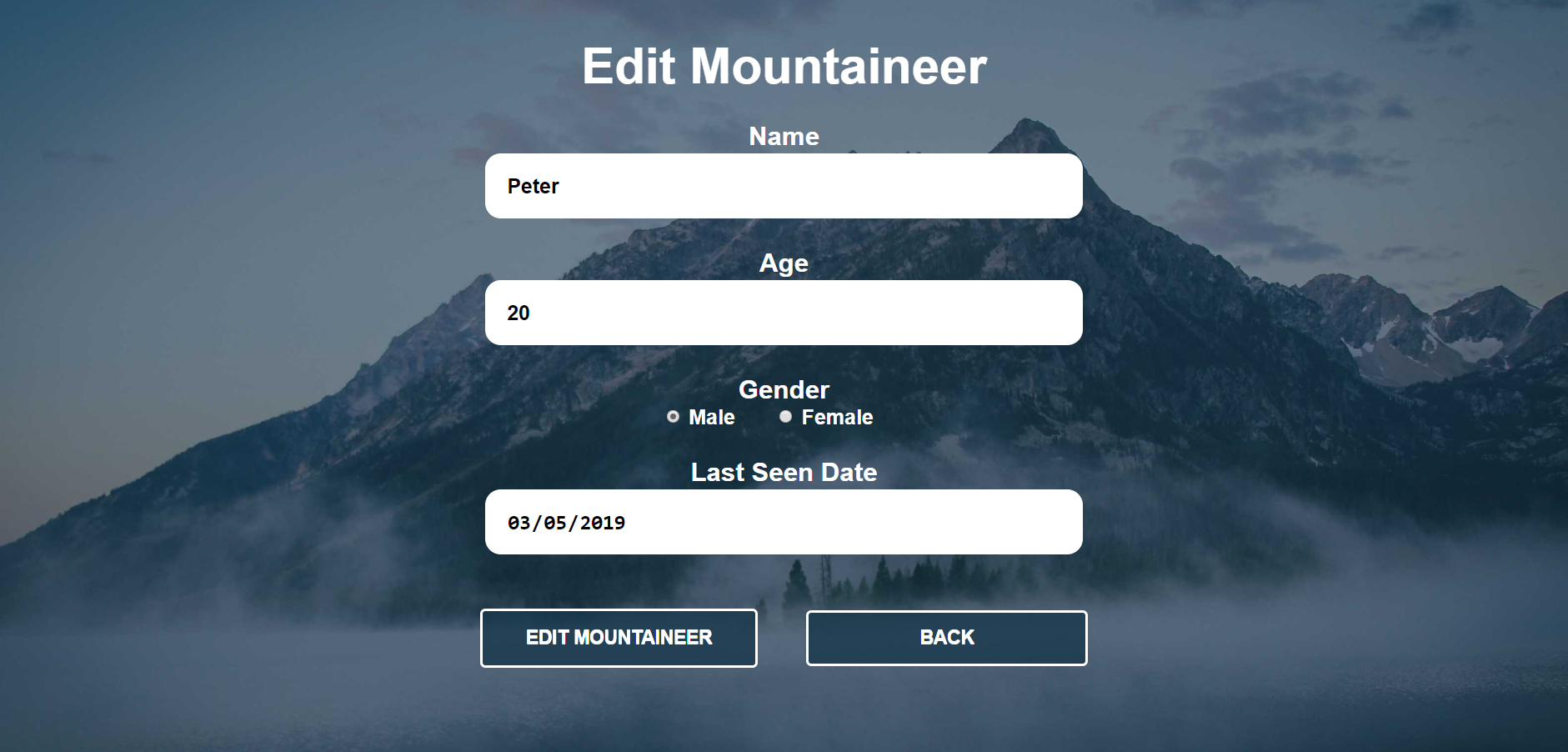
### Create Page

#### Route: **"/create"**



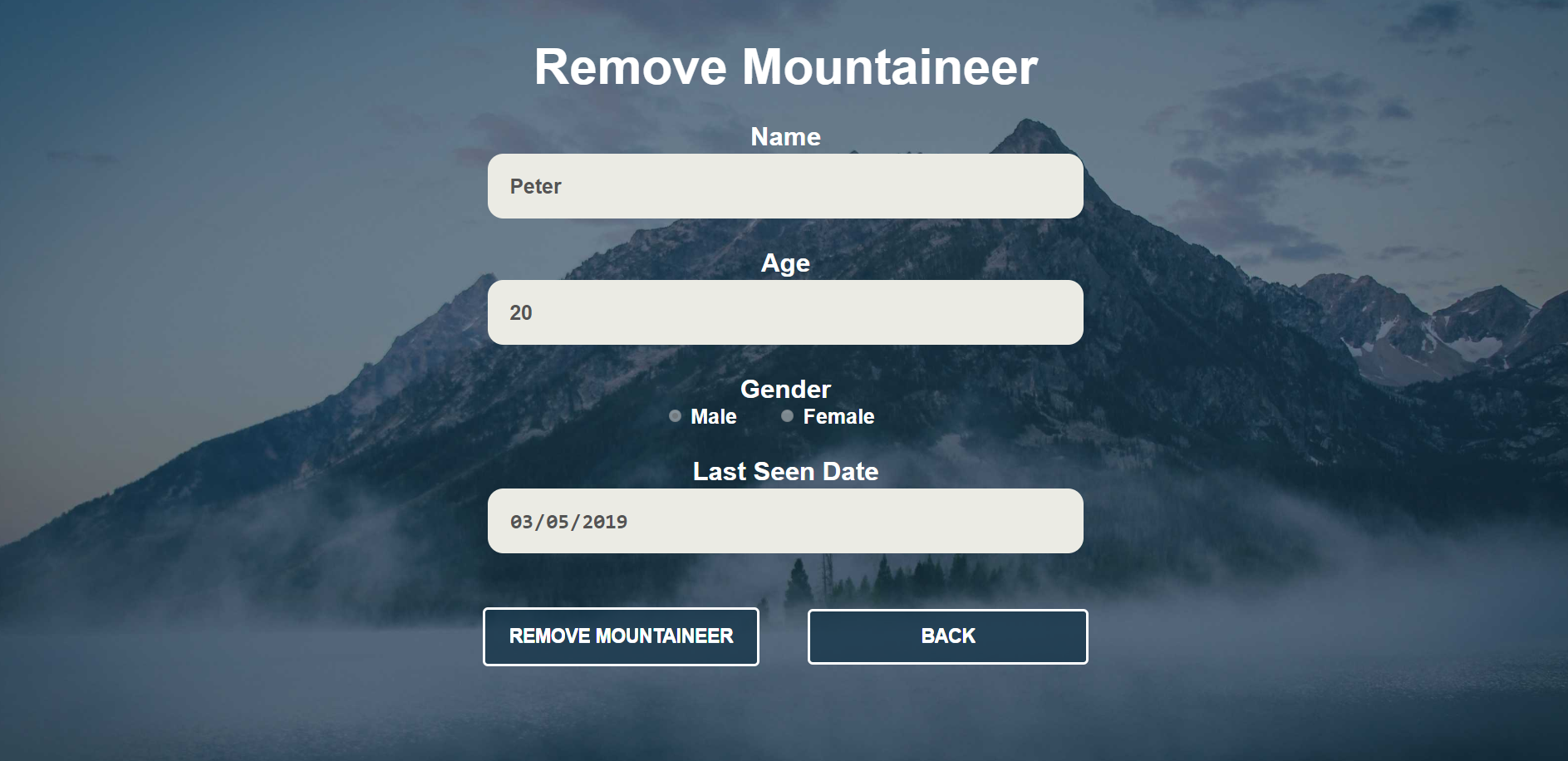
### Edit Page

#### Route: **"/edit/{id}"**



### Delete Page

#### Route: **"/delete/{id}"**



## Problem

As you can see the different pages are on different routes. Most of the routing logic will be given to you in the **Skeleton**, but you should make sure that the application **works properly**.

Implement the "**RescueRegister" app** using only **your technology.**

## Setup

Before you start working, make sure you **download all the dependencies** (packages) required for your technology and **set up** the **databases**! Below are instructions on how to do this:

### PHP and Symfony

1. Make sure you've started your **MySQL server** (either from **XAMPP** or standalone)
2. Open a **Terminal in PHPStorm** or **shell** / **command prompt** / **PowerShell** window in the **root directory**: [Shift] + [Right click] 🡺 [Open command window here]
3. Enter the "composer install" command to restore its **Composer dependencies**   
   (described in composer.json)
4. Enter the "php bin/console doctrine:database:create --if-not-exists" command
5. Done!

### JavaScript and Node.js

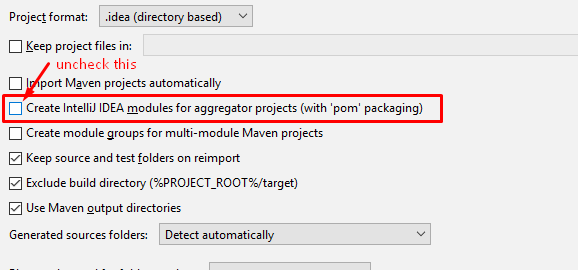
1. Go into the **root directory** of the project (where the index.js file is)
2. Open a **shell** / **command prompt** / **PowerShell** window in the **root directory**: [Shift] + [Right click] 🡺 [Open command window here]
3. Enter the “npm install” command to restore its **Node.js dependencies** (described in package.json)
4. Type **node index.js** to start the server
5. Done!

### C# and ASP.NET

The C# project will automatically resolve its **NuGet dependencies** (described in packages.config) using the NuGet package restore when the project is built.

### Java and Spring MVC

When you import your project, you should **uncheck "**Create IntelliJ IDEA modules for aggregator projects (with 'pom' packaging)**"**:



This project is **set up to use Java jdk 1.8.** If your version is different, you can change it in **Maven dependencies** like this:



The Java project will automatically resolve its **Maven dependencies** (described in pom.xml) when the project is built.