|  |  |
| --- | --- |
| ГУАП  КАФЕДРА № 43  ОТЧЕТ  ЗАЩИЩЕН С ОЦЕНКОЙ  ПРЕПОДАВАТЕЛЬ |  |
|  |  |
| ассистент | Кочин Д.А. |

подпись, дата

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1

Разработка простого серверного приложения J2EE с использованием сервлетов

по дисциплине: ТЕХНОЛОГИИ РАЗРАБОТКИ СЕРВЕРНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4936 Петровнина Д.В.

подпись, дата фамилия, инициалы  
  
  
  
  
 Санкт-Петербург 2021

**Лабораторная работа №1**

Разработка простого серверного приложения J2EE с использованием сервлетов

**Цель работы:**   
Разработать серверное веб-приложения на основе Maven с использованием сервлетов и возможным применением Spring и баз данных, разработать экранную форму и установить взаимодействие между формой и серверным приложением на основе GET и POST запросов.  
  
**Задание на лабораторную работу:**

1. В соответствии со своим вариантом разработайте набор экранных форм приложения (порядка 5)
2. Соберите проект веб-приложения (war) на Maven (можно без использования Spring)
3. Реализуйте формы средствами сервлетов. Проект должен как минимум содержать формы просмотра, добавления и удаления данных
4. Аргументируйте почему были выбраны HTTP методы GET, POST или оба
5. Использовать базу данных можно, но не обязательно

Вариант №16  
  
Система безопасности предприятия (помещения, люди, права на вход)

**Ход работы:**

1. Создадим проект на основе Spring с помощью среды разработки. В разработке использовался язык программирования Java и сборщик Gradle.
2. Разработаем необходимые экранные формы с помощью языка разметки html с использованием thymeleaf и Bootstrap, обеспечивающие все необходимые функции (добавление, удаление и просмотр информации соответствующей варианту задания).
3. С помощью функций фреймворка Spring создадим необходимые сервлеты:

Полный код программы содержится в Приложение 1.

@Controller  
public class EstateController {

@GetMapping("estates")  
String getEstates(Model model) {  
 model.addAttribute("estates", *estateList*);  
 return "estates";  
}  
  
@GetMapping("estates/add")  
String getEstatesAdd() {  
 return "add";  
}  
  
@PostMapping("estates/add")

String addEstate(@RequestParam String room,  
 @RequestParam String people,  
 @RequestParam String right) {  
  
 if (room.isBlank() ||  
 people.isBlank() ||  
 right.isBlank()) {  
  
 return "redirect:/error";  
 }  
 *estateList*.add(new Estate(*ID*++, room, people, right));  
 return "redirect:/estates";  
}

Аннотация @Controller помечает выбранный класс в качестве сервлета, обрабатывающего запросы, приложение автоматически будет перенаправлять необходимые запросы к соответствующему методу обработчика с пометкой @Get/PostMapping. Аннотация @GetMapping помечает метод в качестве обработчика для GET запросов по выбранному url, а PostMapping соответственно POST запросы. @RequestParam используется для получения определенных полей из поступающих POST запросов на сервер.

1. GET запросы предназначены для просмотра данных и соответственно вывода пользователю, направлены именно на получение информации с возможностью при необходимости конкретизации с использованием параметров (в нашем случае ответом на такой запрос является одна из html форм).   
   POST запросы используются для передачи каких-либо данных из форм на сторону сервера в неявном виде для пользователя.
2. Разработаем модель данных, соответствующую варианту с полями идентификатора, названия помещения, имени человека и права на вход.

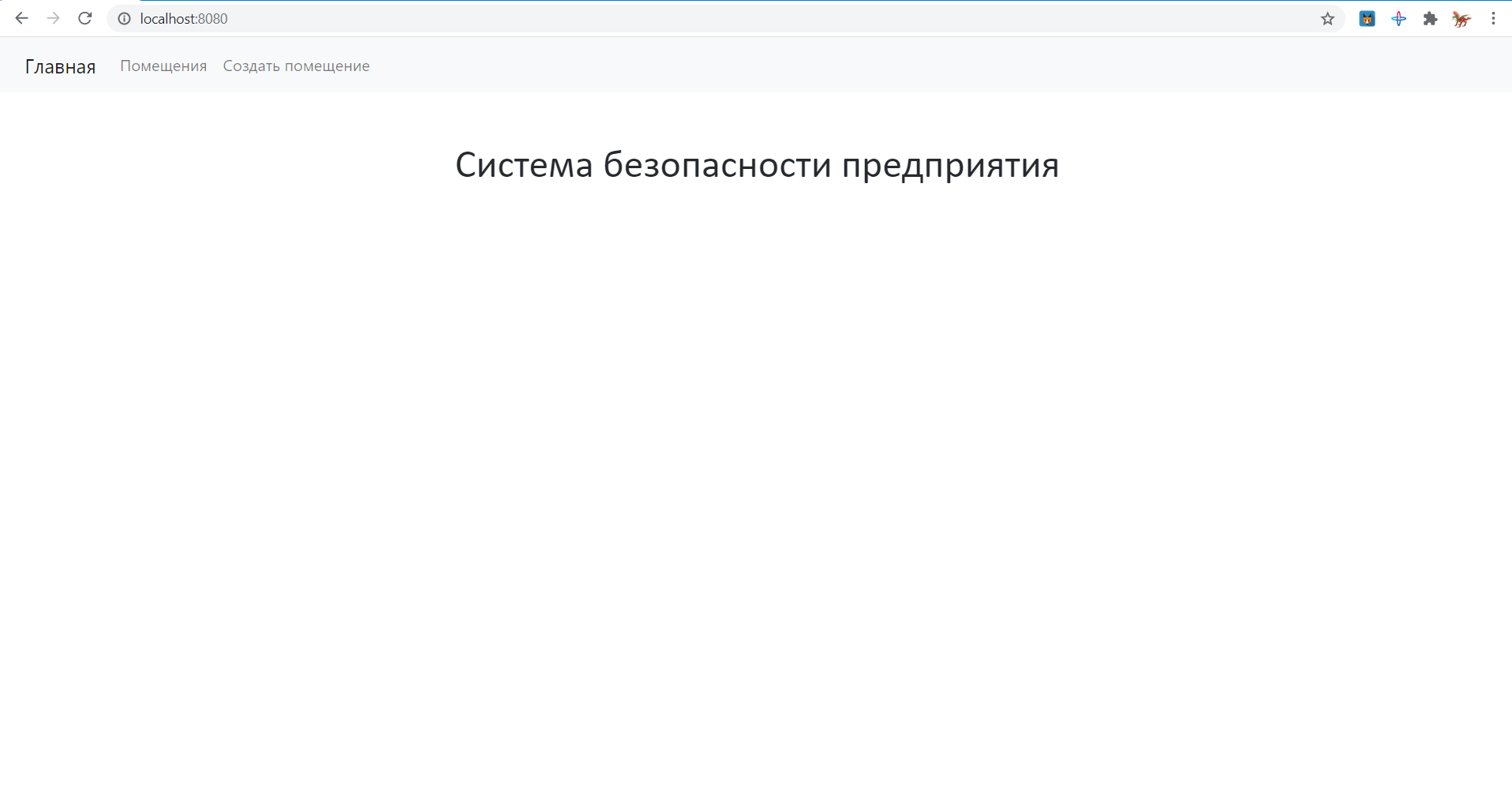
public class Estate {  
  
 private final Integer id;  
 private final String room;  
 private final String people;  
 private final String right;

1. С помощью фреймворка Exposed создадим базу данных (H2) с таблицей на основе раннее разработанной модели:

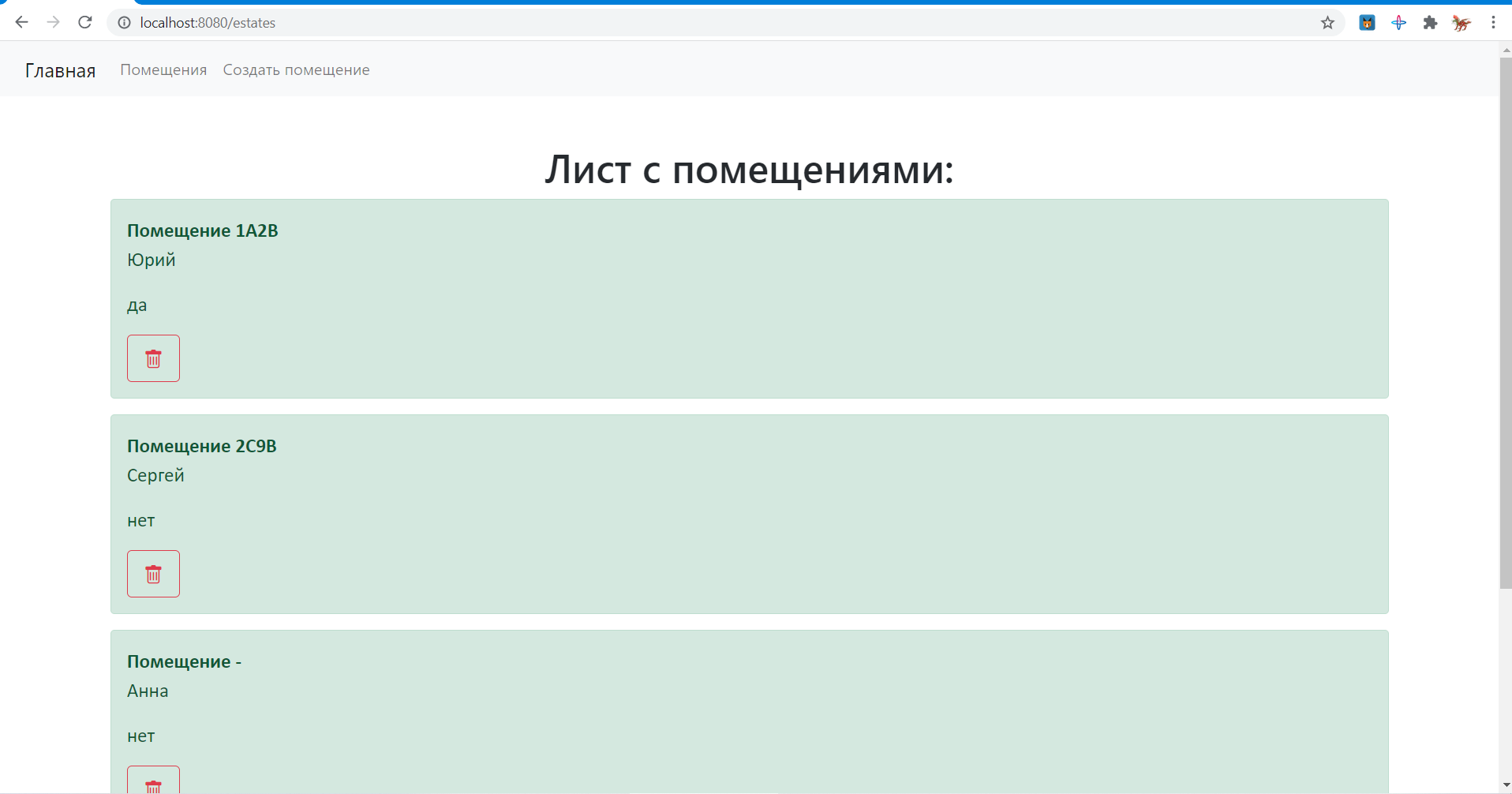
private final static ArrayList<Estate> *estateList* = new ArrayList(List.*of*(  
 (new Estate(1, "Помещение 1А2В", "Юрий", "да")),  
 (new Estate(2, "Помещение 2С9В", "Сергей", "нет")),  
 (new Estate(3, "Помещение -", "Анна", "нет")),  
 (new Estate(4, "Помещение 3С3F", "Софья", "да"))  
));  
private static Integer *ID* = *estateList*.size() + 1;

Далее взаимодействие с базой данных в коде будет происходить посредством Data Access Object (CollectionDao), в котором будут выполнять все основные необходимые транзакции (добавление, удаление, получение данных).

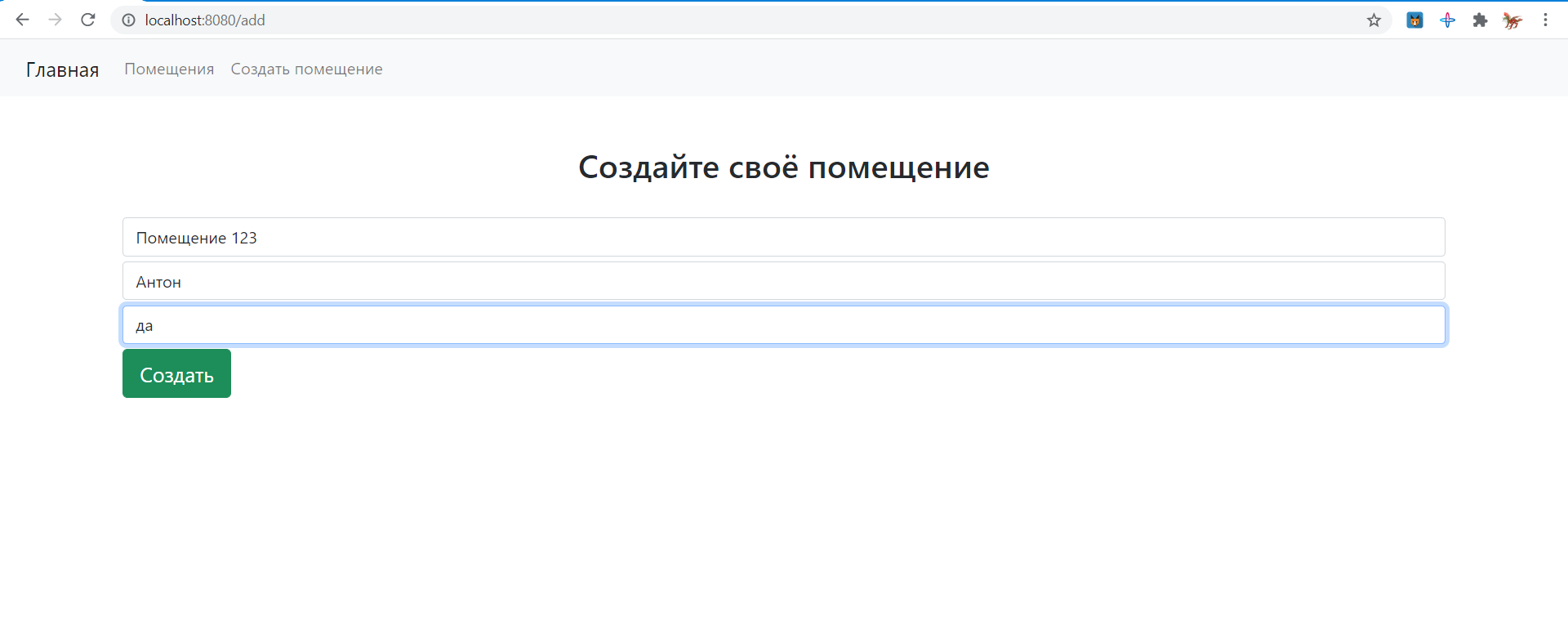
1. Проверим работы программы:

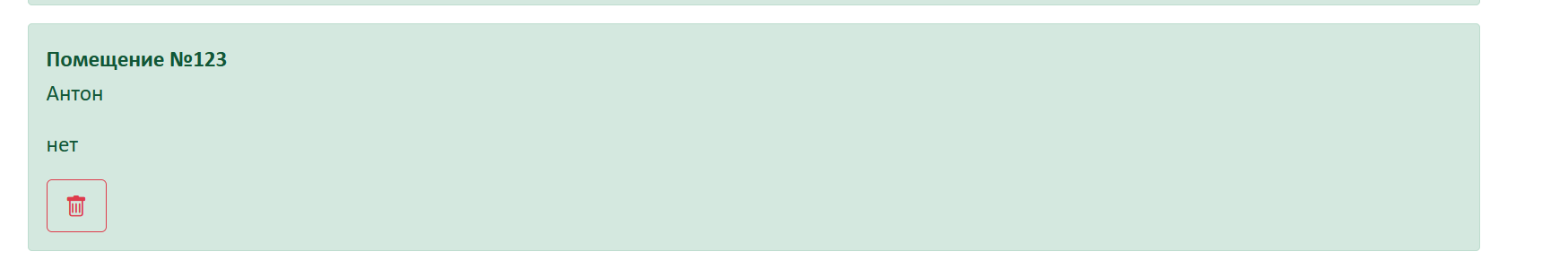


Просмотр данных (GET):

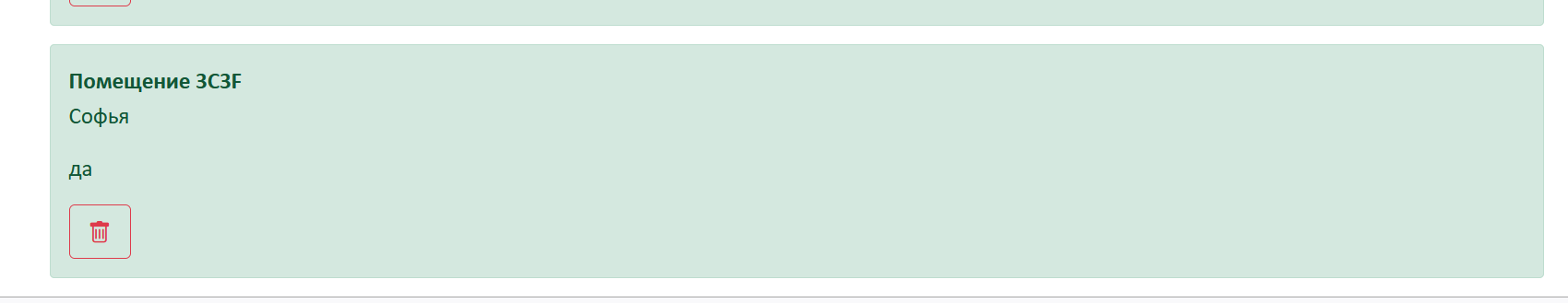


Добавление данных (POST):





Удаление данных (POST):

  
Добавленного ранее помещения нету в списке

**Вывод:** в ходе лабораторной работы были получены основные навыки разработки простых серверных приложений с помощью фреймворка Spring, изучены основы взаимодействия клиент-сервер посредством http запросов, освоен язык разметки html и проведена работа с созданием и применением базы данных.

**Приложение 1.**

**ServLab1Application.java:**

package com.example.serv1;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class Serv1Application {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(Serv1Application.class, args);  
 }  
  
}

**MainController.java:**

package com.example.serv1.controllers;  
  
import org.springframework.stereotype.Controller;  
import org.springframework.web.bind.annotation.GetMapping;  
  
@Controller  
public class MainController {  
  
 @GetMapping("/")  
 String getHome() { return "home"; }  
  
 @GetMapping("/error")  
 String getError() { return "error"; }  
}

**RoomsController.java:**

package com.example.serv1.controllers;import org.springframework.ui.Model;import com.example.serv1.models.Rooms;import org.springframework.stereotype.Controller;import org.springframework.web.bind.annotation.\*;import java.util.ArrayList;import java.util.List;@Controllerpublic class RoomsController { private final static ArrayList<Rooms> *roomsList* = new ArrayList(List.*of*( (new Rooms(1, "Помещение 1А2В", "Юрий", "да")), (new Rooms(2, "Помещение 2С9В", "Сергей", "нет")), (new Rooms(3, "Помещение -", "Анна", "нет")), (new Rooms(4, "Помещение 3С3F", "Софья", "да")) )); private static Integer *ID* = *roomsList*.size() + 1; @GetMapping("rooms") String getRooms(Model model) { model.addAttribute("rooms", *roomsList*); return "rooms"; } @GetMapping("add") String getRoomsAdd() { return "add"; } @PostMapping("add") String addRoom(@RequestParam String room, @RequestParam String people, @RequestParam String right) { if (room.isBlank() || people.isBlank() || right.isBlank()) { return "redirect:/error"; } *roomsList*.add(new Rooms(*ID*++, room, people, right)); return "redirect:/rooms"; } @PostMapping("rooms/remove/{id}") String removeRooms(@PathVariable Integer id) { *roomsList*.removeIf(rooms -> id.equals(rooms.getId())); return "redirect:/rooms"; }}

**Rooms.java**

package com.example.serv1.models;  
  
public class Rooms {  
  
 private final Integer id;  
 private final String room;  
 private final String people;  
 private final String right;  
  
 public Rooms(  
 Integer id,  
 String room,  
 String people,  
 String right) {  
  
 this.id = id;  
 this.room = room;  
 this.people = people;  
 this.right = right;  
 }  
  
 public Integer getId() { return id; }  
 public String getRoom() { return room; }  
 public String getPeople() { return people; }  
 public String getRight() { return right; }  
}

**home.html**

<!DOCTYPE html>  
<html lang="ru">  
<head>  
  
 <meta charset="UTF-8">  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-F3w7mX95PdgyTmZZMECAngseQB83DfGTowi0iMjiWaeVhAn4FJkqJByhZMI3AhiU" crossorigin="anonymous">  
  
 <title>Главная</title>  
  
 <style>  
 h1, h2 {  
 font-family: Calibri, serif;  
 }  
 </style>  
  
</head>  
<body>  
<header th:insert="header :: header"></header>  
  
 <div class="container mt-5">  
 <h1 align="middle">Система безопасности предприятия</h1>  
 </div>  
  
</body>  
</html>

**header.html**

<div th:fragment="header"> <div> <nav class="navbar navbar-expand-lg navbar-light bg-light"> <a class="navbar-brand" href="/" style="margin-left: 25px">Главная</a> <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation"> <span class="navbar-toggler-icon"></span> </button> <div class="collapse navbar-collapse" id="navbarNav"> <ul class="navbar-nav"> <li class="nav-item"> <a class="nav-link" href="/estates">Помещения</a> </li> <li class="nav-item"> <a class="nav-link" href="/add">Создать помещение</a> </li> </ul> </div> </nav> </div></div>

**rooms.html**

<!DOCTYPE html><html xmlns:th="http://www.thymeleaf.org"><head> <meta charset="UTF-8"> <title>Лист с помешениями</title> <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-icons.css"> <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-F3w7mX95PdgyTmZZMECAngseQB83DfGTowi0iMjiWaeVhAn4FJkqJByhZMI3AhiU" crossorigin="anonymous"> <style> p, strong { font-family: Calibri, serif; font-size: 20px; } </style></head><body><header th:insert="header :: header"></header><div class="container mt-5"> <div th:if="${!rooms.isEmpty()}"> <h1 align="middle">Лист с помещениями:</h1> <div th:each="el : ${rooms}" class="alert alert-success mt-2"> <strong th:utext="${el.room}"></strong> <p th:utext="${el.people}"></p> <p th:utext="${el.right}"></p> <form th:action="'rooms/remove/' + ${el.id}" method="post"> <button th:href="'rooms/remove/' + ${el.id}" type="submit" class="btn btn-outline-danger btn-circle btn-lg"><i class="bi bi-trash"></i></button> </form> </div> </div> <div th:unless="${!rooms.isEmpty()}"> <h1>Лист с помещениями пустой</h1> </div></div></body></html>

**error.html**

<!DOCTYPE html><html lang="ru"><head> <meta charset="UTF-8"> <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-F3w7mX95PdgyTmZZMECAngseQB83DfGTowi0iMjiWaeVhAn4FJkqJByhZMI3AhiU" crossorigin="anonymous"> <title>Ошибка</title> <style> h1, h2 { font-family: Calibri, serif; } </style></head><body><header th:insert="header :: header"></header><div class="container mt-5"> <h1 align="middle">Это страница с ошибкой.</h1> <h2 align="middle">Ты сделал что-то не так.</h2></div></body></html>

**add.html**

<!DOCTYPE html><html lang="ru"><head> <meta charset="UTF-8"> <title>Создание помещения</title> <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-F3w7mX95PdgyTmZZMECAngseQB83DfGTowi0iMjiWaeVhAn4FJkqJByhZMI3AhiU" crossorigin="anonymous"> <style> input { margin-bottom: 5px; } </style></head><body><header th:insert="header :: header"></header><div class="container mt-5"> <h2 align="middle">Создайте своё помещение</h2> <form method="post"> <br> <input type="text" name="room" placeholder="room" class="form-control" aria-label="room"> <input type="text" name="people" placeholder="people" class="form-control" aria-label="people"> <input type="text" name="right" placeholder="right" class="form-control" aria-label="right"> <button type="submit" class="btn btn-success btn-lg btn-block">Создать</button> </form></div></body></html>