ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| кандидат технических наук |  |  |  | П. А. Степанов |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2 |
| Разработка ресурса REST/JSON сервиса |
| по курсу: Технология разработки серверных информационных систем |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4134к |  |  |  | Д.В. Самарин |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2023

**Текст и вариант задания:**

16 вариант

Система безопасности предприятия (помещения, люди, права на вход)

**Описание разрабатываемого продукта:**

В данной программе создан сервис по работе с доступом к комнатам (вывод, добавление, удаление и поиск)

**Текст основных фрагментов кода :**

AccessToRoom;

package com.example.lab2.model;

import io.swagger.v3.oas.annotations.media.Schema;

@Schema

public class AccessToRoom {

    @Schema(

            description = "The unique id",

            name = "id",

            type = "long",

            example = "1")

    private long id = 0;

    @Schema(

            description = "The name of the user",

            name = "username",

            type = "String",

            example = "Dmitry")

    private String username;

    @Schema(

            description = "The name of the room",

            name = "roomName",

            type = "String",

            example = "Bathroom")

    private String roomName;

    @Schema(

            description = "The boolean variable to access to room",

            name = "accessToRoom",

            type = "boolean",

            example = "true")

    private boolean accessToRoom;

    public AccessToRoom() {

    }

    public AccessToRoom(String username, String roomName, boolean accessToRoom) {

        this.username = username;

        this.roomName = roomName;

        this.accessToRoom = accessToRoom;

    }

    public void setId(long id) {

        this.id = id;

    }

    public long getId() {

        return id;

    }

    public String getUsername() {

        return username;

    }

    public void setUsername(String username) {

        this.username = username;

    }

    public String getRoomName() {

        return roomName;

    }

    public void setRoomName(String roomName) {

        this.roomName = roomName;

    }

    public boolean isAccessToRoom() {

        return accessToRoom;

    }

    public void setAccessToRoom(boolean isPublished) {

        this.accessToRoom = isPublished;

    }

    @Override

    public String toString() {

        return "AccessToRoom [id=" + id + ", username=" + username + ", desc=" + roomName + ", accessToRoom=" + accessToRoom + "]";

    }

}

AccessToRoomController;

package com.example.lab2.controller;

import java.util.ArrayList;

import java.util.List;

import com.example.lab2.model.AccessToRoom;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

import com.example.lab2.service.AccessToRoomService;

@CrossOrigin(origins = "http://localhost:8081")

@RestController

@RequestMapping("/api")

public class AccessToRoomController {

    @Autowired

    AccessToRoomService accessToRoomService;

    @GetMapping("/access-to-room")

    public ResponseEntity<List<AccessToRoom>> getAllAccessToRooms(@RequestParam(required = false) String username) {

        try {

            List<AccessToRoom> accessToRooms = new ArrayList<AccessToRoom>();

            if (username == null)

                accessToRoomService.findAll().forEach(accessToRooms::add);

            else

                accessToRoomService.findByTitleContaining(username).forEach(accessToRooms::add);

            if (accessToRooms.isEmpty()) {

                return new ResponseEntity<>(HttpStatus.NO\_CONTENT);

            }

            return new ResponseEntity<>(accessToRooms, HttpStatus.OK);

        } catch (Exception e) {

            return new ResponseEntity<>(null, HttpStatus.INTERNAL\_SERVER\_ERROR);

        }

    }

    @GetMapping("/access-to-room/{id}")

    public ResponseEntity<AccessToRoom> getAccessToRoomById(@PathVariable("id") long id) {

        AccessToRoom accessToRoom = accessToRoomService.findById(id);

        if (accessToRoom != null) {

            return new ResponseEntity<>(accessToRoom, HttpStatus.OK);

        } else {

            return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

        }

    }

    @PostMapping("/access-to-room")

    public ResponseEntity<AccessToRoom> createAccessToRoom(@RequestBody AccessToRoom accessToRoom) {

        try {

            AccessToRoom \_accessToRoom = accessToRoomService

                    .save(new AccessToRoom(accessToRoom.getUsername(), accessToRoom.getRoomName(), accessToRoom.isAccessToRoom()));

            return new ResponseEntity<>(\_accessToRoom, HttpStatus.CREATED);

        } catch (Exception e) {

            return new ResponseEntity<>(null, HttpStatus.INTERNAL\_SERVER\_ERROR);

        }

    }

    @PutMapping("/access-to-room/{id}")

    public ResponseEntity<AccessToRoom> updateAccessToRoom(@PathVariable("id") long id, @RequestBody AccessToRoom accessToRoom) {

        AccessToRoom \_accessToRoom = accessToRoomService.findById(id);

        if (\_accessToRoom != null) {

            \_accessToRoom.setUsername(accessToRoom.getUsername());

            \_accessToRoom.setRoomName(accessToRoom.getRoomName());

            \_accessToRoom.setAccessToRoom(accessToRoom.isAccessToRoom());

            return new ResponseEntity<>( accessToRoomService.save(\_accessToRoom), HttpStatus.OK);

        } else {

            return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

        }

    }

    @DeleteMapping("/access-to-room/{id}")

    public ResponseEntity<HttpStatus> deleteAccessToRoom(@PathVariable("id") long id) {

        try {

            accessToRoomService.deleteById(id);

            return new ResponseEntity<>(HttpStatus.NO\_CONTENT);

        } catch (Exception e) {

            return new ResponseEntity<>(HttpStatus.INTERNAL\_SERVER\_ERROR);

        }

    }

}