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 дисциплине: Технология разработки серверных информационных систем |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 4134к |  |  |  | Костяков Н.А. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург

2023

Задание на лабораторную работу.

1 Определите перечень Rest-сервисов, выполняющих те же действия, что и в лабораторной работе 1. Внимательно отнеситесь к вопросу какой HTTP метод использует тот или иной сервис и какие коды HTTP он может возвращать. 2 Опишите Ваш API с помощью OpenAPI v3. 3 Средствами swagger сгенерируйте сервер spring 4 Замените pom файл в сгенерированном сервере на файл из моего примера 5 Поменяйте названия пакетов на уникальные 6 Реализуйте приложение.

# Вариант 4

Расписание поездов, самолетов, кораблей

# Листинг

API

*/\*\*  
 \* NOTE: This class is auto generated by the swagger code generator program (3.0.48).  
 \* https://github.com/swagger-api/swagger-codegen  
 \* Do not edit the class manually.  
 \*/*package io.swagger.api;  
  
import io.swagger.model.Error;  
import io.swagger.model.Time;  
import io.swagger.model.TimeTable;  
import io.swagger.v3.oas.annotations.Operation;  
import io.swagger.v3.oas.annotations.Parameter;  
import io.swagger.v3.oas.annotations.enums.ParameterIn;  
import io.swagger.v3.oas.annotations.responses.ApiResponses;  
import io.swagger.v3.oas.annotations.responses.ApiResponse;  
import io.swagger.v3.oas.annotations.media.ArraySchema;  
import io.swagger.v3.oas.annotations.media.Content;  
import io.swagger.v3.oas.annotations.media.Schema;  
import io.swagger.v3.oas.annotations.security.SecurityRequirement;  
import org.springframework.http.ResponseEntity;  
import org.springframework.validation.annotation.Validated;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RequestHeader;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestMethod;  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.web.bind.annotation.RequestPart;  
import org.springframework.web.multipart.MultipartFile;  
import org.springframework.web.bind.annotation.CookieValue;  
  
import javax.validation.Valid;  
import javax.validation.constraints.\*;  
import java.util.List;  
import java.util.Map;  
  
@javax.annotation.Generated(value = "io.swagger.codegen.v3.generators.java.SpringCodegen", date = "2023-10-19T16:54:06.552045592Z[GMT]")  
@Validated  
public interface TtApi {  
  
 @Operation(summary = "добавить в расписание", description = "", tags={ "tt" })  
 @ApiResponses(value = {   
 @ApiResponse(responseCode = "200", description = "Успех", content = @Content(mediaType = "application/json", schema = @Schema(implementation = TimeTable.class))),  
   
 @ApiResponse(responseCode = "200", description = "Error", content = @Content(mediaType = "application/json", schema = @Schema(implementation = Error.class))) })  
 @RequestMapping(value = "/tt",  
 produces = { "application/json" },   
 consumes = { "application/json" },   
 method = RequestMethod.*POST*)  
 ResponseEntity<TimeTable> createTime(@Parameter(in = ParameterIn.*DEFAULT*, description = "", required=true, schema=@Schema()) @Valid @RequestBody Time body);  
  
  
 @Operation(summary = "удаление", description = "", tags={ "tt" })  
 @ApiResponses(value = {   
 @ApiResponse(responseCode = "200", description = "Успех"),  
   
 @ApiResponse(responseCode = "200", description = "Error", content = @Content(mediaType = "application/json", schema = @Schema(implementation = Error.class))) })  
 @RequestMapping(value = "/tt/{time\_id}",  
 produces = { "application/json" },   
 method = RequestMethod.*DELETE*)  
 ResponseEntity<Void> delTime(@Parameter(in = ParameterIn.*PATH*, description = "Id of Time", required=true, schema=@Schema()) @PathVariable("time\_id") int timeId);  
  
  
 @Operation(summary = "получить расписание", description = "", tags={ "tt" })  
 @ApiResponses(value = {   
 @ApiResponse(responseCode = "200", description = "Успех", content = @Content(mediaType = "application/json", schema = @Schema(implementation = TimeTable.class))),  
   
 @ApiResponse(responseCode = "200", description = "Error", content = @Content(mediaType = "application/json", schema = @Schema(implementation = Error.class))) })  
 @RequestMapping(value = "/tt",  
 produces = { "application/json" },   
 method = RequestMethod.*GET*)  
 ResponseEntity<TimeTable> getTt();  
  
  
 @Operation(summary = "по id", description = "", tags={ "tt" })  
 @ApiResponses(value = {   
 @ApiResponse(responseCode = "200", description = "Успех", content = @Content(mediaType = "application/json", schema = @Schema(implementation = Time.class))),  
   
 @ApiResponse(responseCode = "200", description = "Error", content = @Content(mediaType = "application/json", schema = @Schema(implementation = Error.class))) })  
 @RequestMapping(value = "/tt/{time\_id}",  
 produces = { "application/json" },   
 method = RequestMethod.*GET*)  
 ResponseEntity<Time> getTtById(@Parameter(in = ParameterIn.*PATH*, description = "Id of Time", required=true, schema=@Schema()) @PathVariable("time\_id") int timeId);  
  
}

API controller

package io.swagger.api;  
  
import io.swagger.model.Error;  
import io.swagger.model.Time;  
import io.swagger.model.TimeTable;  
import com.fasterxml.jackson.databind.ObjectMapper;  
import io.swagger.v3.oas.annotations.Operation;  
import io.swagger.v3.oas.annotations.Parameter;  
import io.swagger.v3.oas.annotations.enums.ParameterIn;  
import io.swagger.v3.oas.annotations.responses.ApiResponses;  
import io.swagger.v3.oas.annotations.responses.ApiResponse;  
import io.swagger.v3.oas.annotations.media.ArraySchema;  
import io.swagger.v3.oas.annotations.media.Content;  
import io.swagger.v3.oas.annotations.media.Schema;  
import io.swagger.v3.oas.annotations.security.SecurityRequirement;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.RestController;  
import org.springframework.web.bind.annotation.CookieValue;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RequestHeader;  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.web.bind.annotation.RequestPart;  
import org.springframework.web.bind.annotation.RestController;  
import org.springframework.web.multipart.MultipartFile;  
  
import javax.validation.Valid;  
import javax.validation.constraints.\*;  
import javax.servlet.http.HttpServletRequest;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Map;  
import java.util.concurrent.atomic.AtomicInteger;  
  
@javax.annotation.Generated(value = "io.swagger.codegen.v3.generators.java.SpringCodegen", date = "2023-10-19T16:54:06.552045592Z[GMT]")  
@RestController  
public class TtApiController implements TtApi {  
  
 private static final Logger *log* = LoggerFactory.*getLogger*(TtApiController.class);  
  
 private final ObjectMapper objectMapper;  
  
 private final HttpServletRequest request;  
  
 @org.springframework.beans.factory.annotation.Autowired  
 public TtApiController(ObjectMapper objectMapper, HttpServletRequest request) {  
 this.objectMapper = objectMapper;  
 this.request = request;  
 }  
  
 private AtomicInteger key = new AtomicInteger();  
 public TimeTable TTable = new TimeTable();  
 public ResponseEntity<TimeTable> createTime(@Parameter(in = ParameterIn.*DEFAULT*, description = "", required=true, schema=@Schema()) @Valid @RequestBody Time body) {  
 String accept = request.getHeader("Accept");  
 if (accept != null && accept.contains("application/json")) {  
 synchronized (TTable){  
 body.setTimeId(key.incrementAndGet()) ;  
 TTable.add(body);  
 }  
 return new ResponseEntity<TimeTable>(HttpStatus.*CREATED*);  
 }  
  
 return new ResponseEntity<TimeTable>(HttpStatus.*NOT\_IMPLEMENTED*);  
 }  
  
 public ResponseEntity<Void> delTime(@Parameter(in = ParameterIn.*PATH*, description = "Id of Time", required=true, schema=@Schema()) @PathVariable("time\_id") int timeId) {  
 String accept = request.getHeader("Accept");  
 TTable.remove(timeId);  
 return new ResponseEntity<Void>(HttpStatus.*OK*);  
 }  
  
 public ResponseEntity<TimeTable> getTt() {  
 String accept = request.getHeader("Accept");  
 if (accept != null && accept.contains("application/json")) {  
  
 synchronized (TTable){  
 return new ResponseEntity<TimeTable>(TTable,HttpStatus.*OK*);  
 }  
  
  
 }  
  
 return new ResponseEntity<TimeTable>(HttpStatus.*NOT\_IMPLEMENTED*);  
 }  
  
 public ResponseEntity<Time> getTtById(@Parameter(in = ParameterIn.*PATH*, description = "Id of Time", required=true, schema=@Schema()) @PathVariable("time\_id") int timeId) {  
 String accept = request.getHeader("Accept");  
 if (accept != null && accept.contains("application/json")) {  
 synchronized (TTable){  
 return new ResponseEntity<Time>(TTable.get(timeId),HttpStatus.*OK*);  
 }  
 }  
  
 return new ResponseEntity<Time>(HttpStatus.*NOT\_IMPLEMENTED*);  
 }  
  
}