Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

**«Уральский федеральный университет**

**имени первого Президента России Б.Н. Ельцина»**

**Институт радиоэлектроники и информационных технологий**

**Отчёт**

**по лабораторной работе №2**

Выполнила: студентка гр. РИЗ-230916y

Кузнецова Екатерина Алексеевна

Проверил: преподаватель

Архипов Николай Александрович

**ВВЕДЕНИЕ**

Цель: Разработать простой REST сервис

Необходимое программное обеспечение:

IntelliJ IDEA CE, Spring Boot 3, Java 17, Browser, Postman

**1 ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ**

1. Описание входящего (request) и исходящего (response) сообщений, схема работы сервиса
2. Реализовать простой REST сервис
3. Реализовать дополнительный функционал в разработанном по примеру REST сервисе
4. Ответить письменно на вопросы в соответствии с номером в списке
5. Оформить отчет и прикрепить его на сайте https://edu.itlearn.ru/ в соответствующем курсе, в соответствующем разделе

**2 ХОД ВЫПОЛНЕНИЯ**

**2.1 Реализация простого REST сервиса**

package vs.korzhina.SpringBootApp.controller;

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

import jakarta.validation.Valid;

import vs.korzhina.SpringBootApp.exception.UnsupportedCodeException;

import vs.korzhina.SpringBootApp.exception.ValidationFailedException;

import vs.korzhina.SpringBootApp.model.Request;

import vs.korzhina.SpringBootApp.model.Response;

import vs.korzhina.SpringBootApp.service.IValidationService;

import java.text.SimpleDateFormat;

import java.util.Date;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.validation.BindingResult;

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

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

@RestController

public class MyController {

private final IValidationService validationService;

@Autowired

public MyController(IValidationService validationService) {

this.validationService = validationService;

}

@PostMapping(value="/feedback")

public ResponseEntity<Response> feedback(@Valid @RequestBody Request request,

BindingResult bindingResult) {

SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS'Z'");

Response response = Response.builder()

.uid(request.getUid())

.operationUid(request.getOperationUid())

.systemTime(simpleDateFormat.format(new Date()))

.code("success")

.errorCode("")

.errorMessage("")

.build();

try {

validationService.isValid(bindingResult);

} catch (ValidationFailedException e) {

response.setCode("failed");

response.setErrorCode("ValidationException");

response.setErrorMessage("Ошибка валидации");

return new ResponseEntity<>(response, HttpStatus.BAD\_REQUEST);

} catch(Exception e){

response.setCode("failed");

response.setErrorCode("UnknownException");

response.setErrorMessage("Произошла непредвиденная ошбика");

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

}

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

}

}

package vs.korzhina.SpringBootApp.exception;

public class ValidationFailedException extends Exception {

public ValidationFailedException(String message){

super(message);

}

}

package vs.korzhina.SpringBootApp.model;

import jakarta.validation.constraints.\*;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@Builder

@NoArgsConstructor

@AllArgsConstructor

public class Request {

@NotBlank

private String uid;

private String operationUid;

private String systemName;

private String systemTime;

private String source;

@Max(100000)

@Min(1)

private int communicationId;

private int templateId;

private int productCode;

private int smsCode;

}

package vs.korzhina.SpringBootApp.model;

import lombok.Builder;

import lombok.Data;

@Data

@Builder

public class Response {

private String uid;

private String operationUid;

private String systemTime;

private String code;

private String errorCode;

private String errorMessage;

}

package vs.korzhina.SpringBootApp.service;

import org.springframework.stereotype.Service;

import org.springframework.validation.BindingResult;

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

import vs.korzhina.SpringBootApp.exception.\*;

import vs.korzhina.SpringBootApp.model.Request;

@Service

public interface IValidationService {

void isValid(BindingResult bindingResult) throws ValidationFailedException;

}

package vs.korzhina.SpringBootApp.service;

import org.springframework.stereotype.Service;

import org.springframework.validation.BindingResult;

import vs.korzhina.SpringBootApp.exception.UnsupportedCodeException;

import vs.korzhina.SpringBootApp.exception.ValidationFailedException;

import vs.korzhina.SpringBootApp.model.Request;

import vs.korzhina.SpringBootApp.model.Response;

@Service

public class RequestValidationService implements IValidationService {

@Override

public void isValid(BindingResult bindingResult) throws ValidationFailedException {

if (bindingResult.hasErrors())

throw new

ValidationFailedException(bindingResult.getFieldError().toString());

}

}

* 1. **Реализация дополнительного функционала**

1. Валидация

package vs.korzhina.SpringBootApp.model;

import jakarta.validation.constraints.\*;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@Builder

@NoArgsConstructor

@AllArgsConstructor

public class Request {

@NotBlank

@Size(max=32)

private String uid;

@NotBlank

@Size(max=32)

private String operationUid;

private String systemName;

@NotBlank

private String systemTime;

private String source;

@Max(100000)

@Min(1)

private int communicationId;

private int templateId;

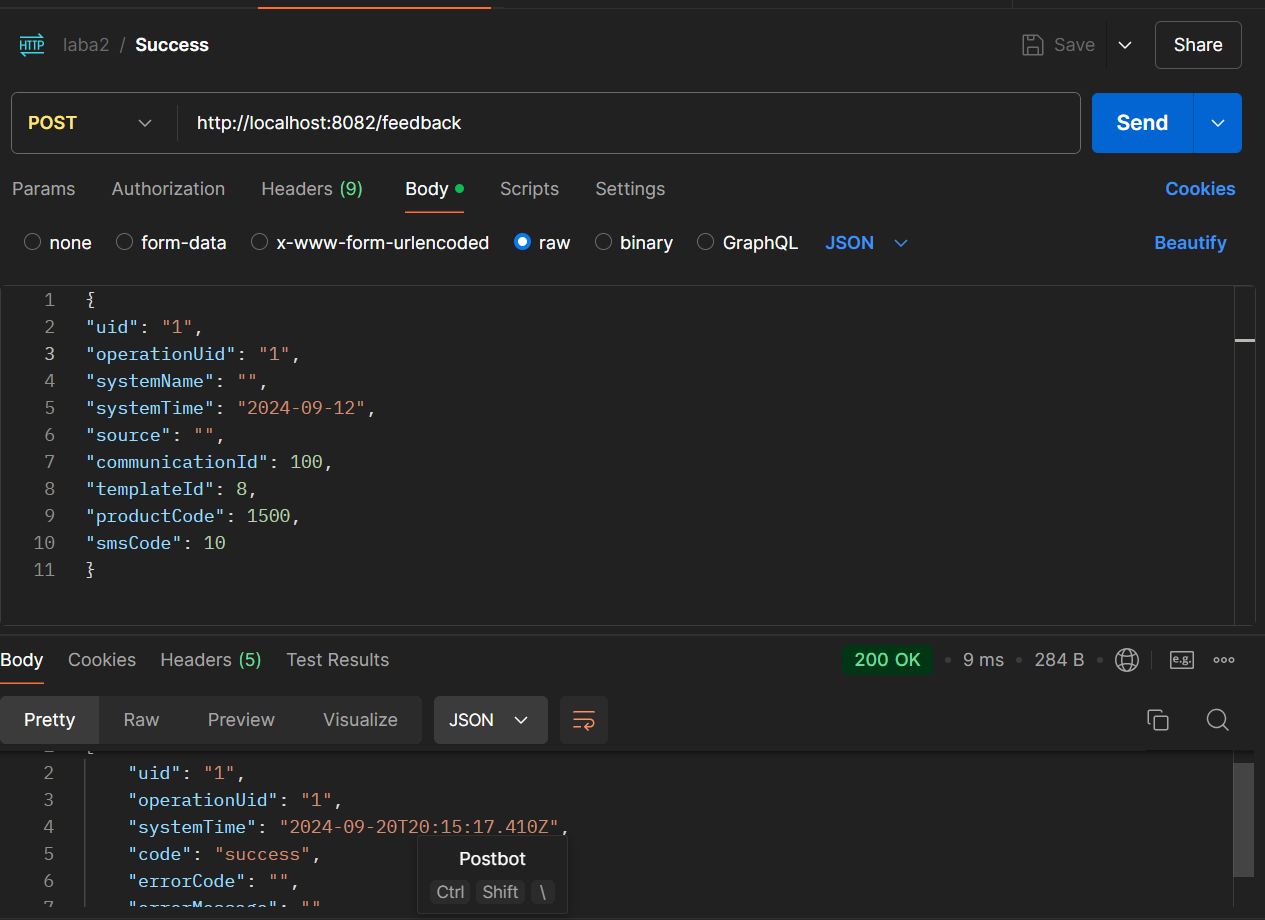
private int productCode;

private int smsCode;

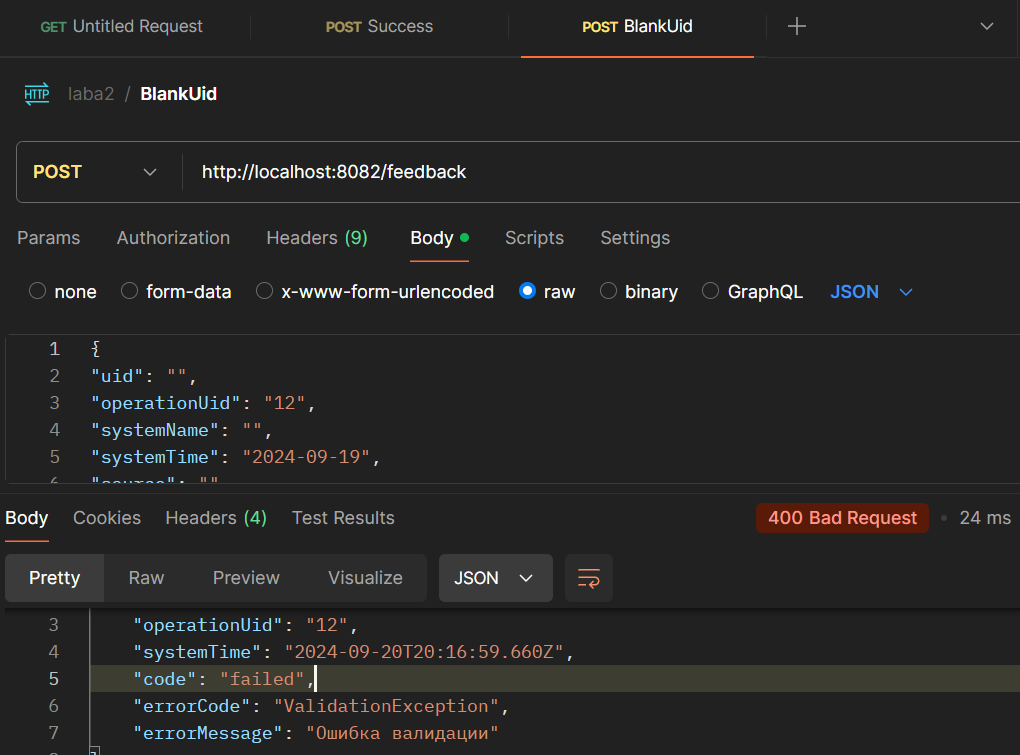
}

1. Тестирование в Postman

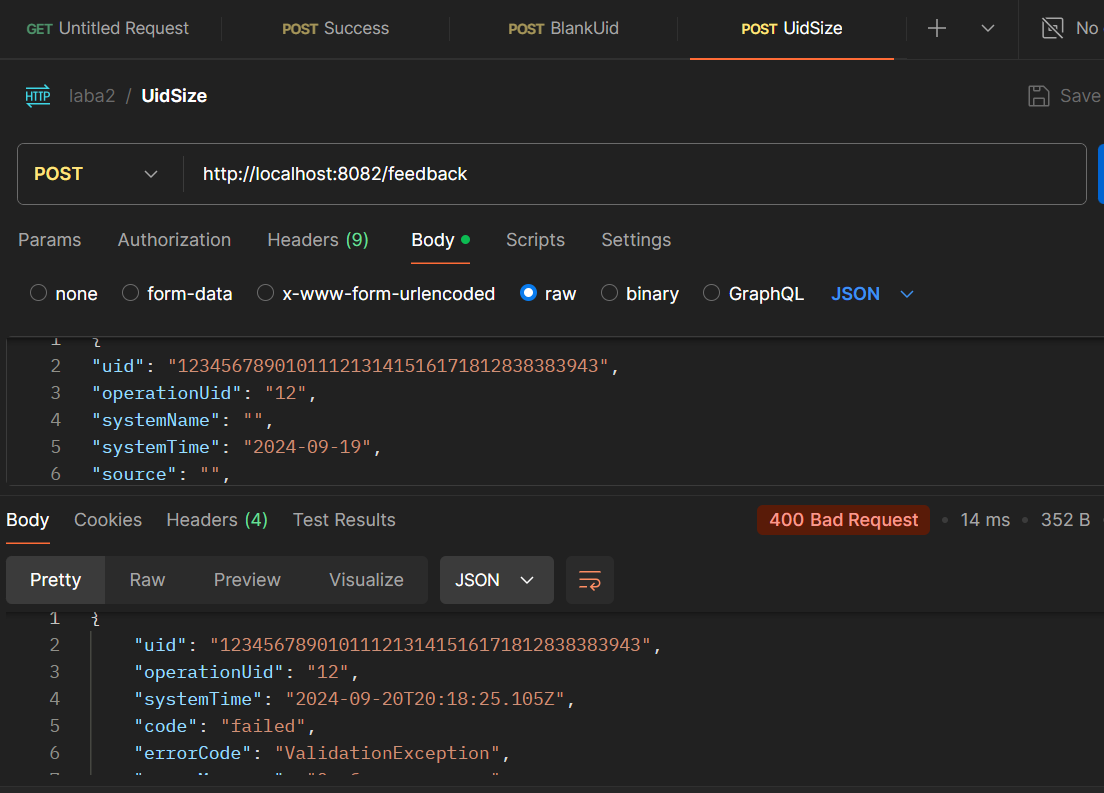
Успешный сценарий:



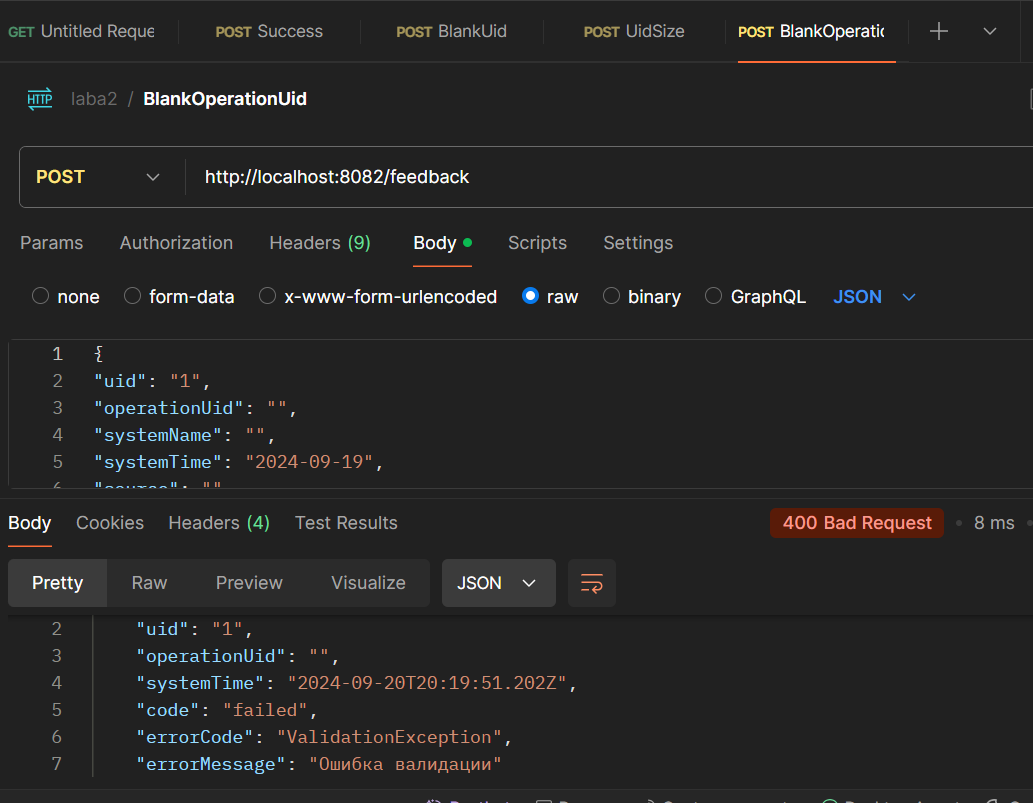
Пустой Uid:



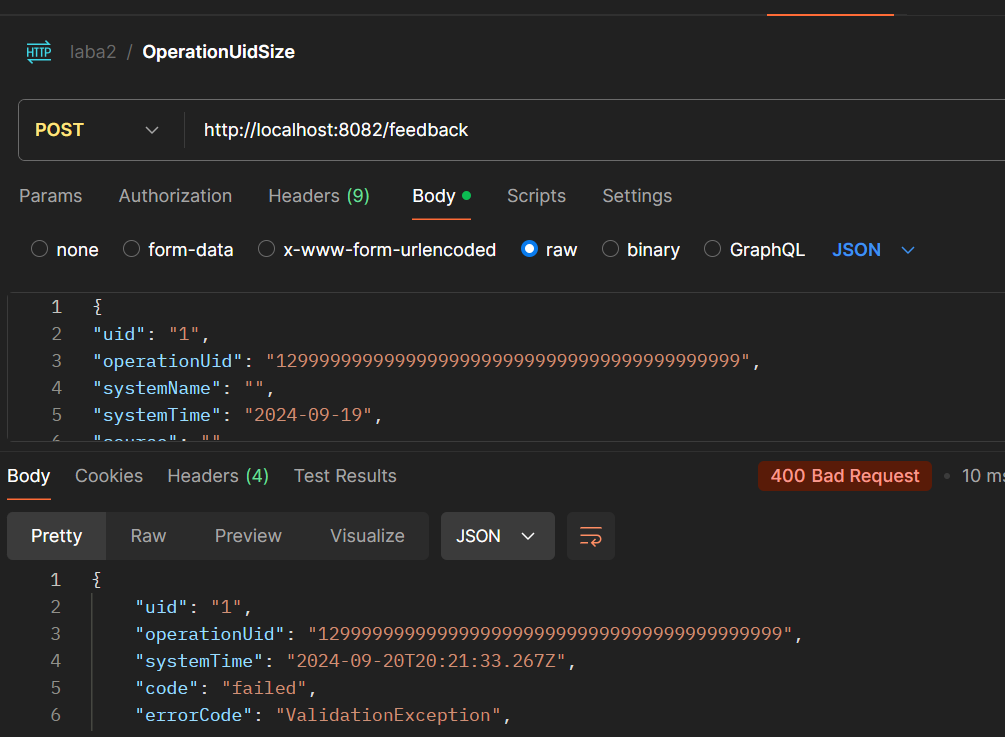
Uid больше 32-х символов:



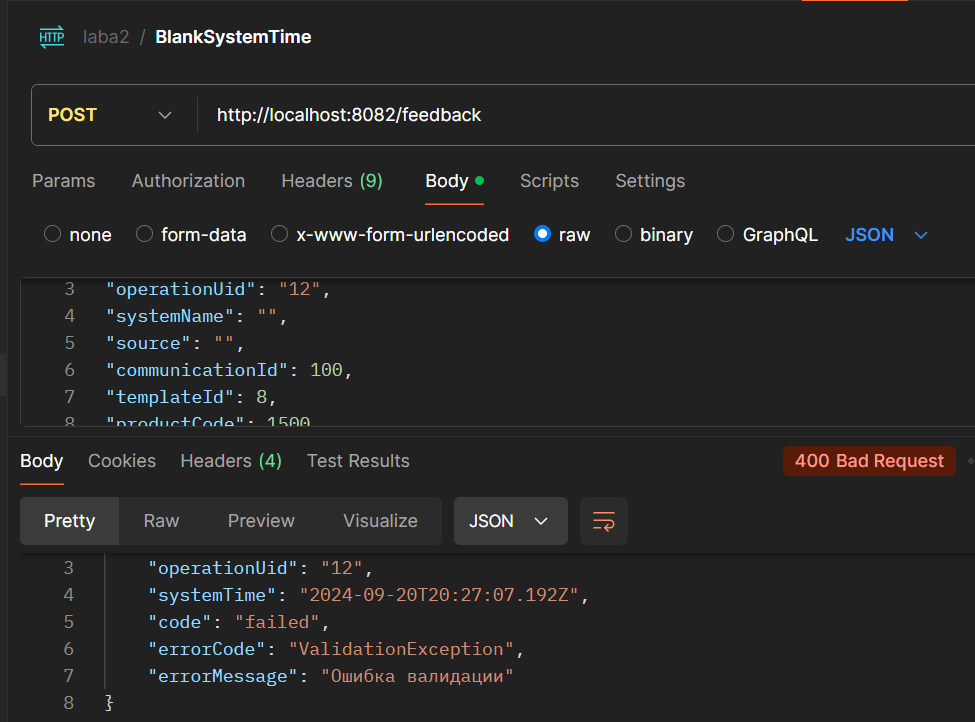
Пустой OperationUid:



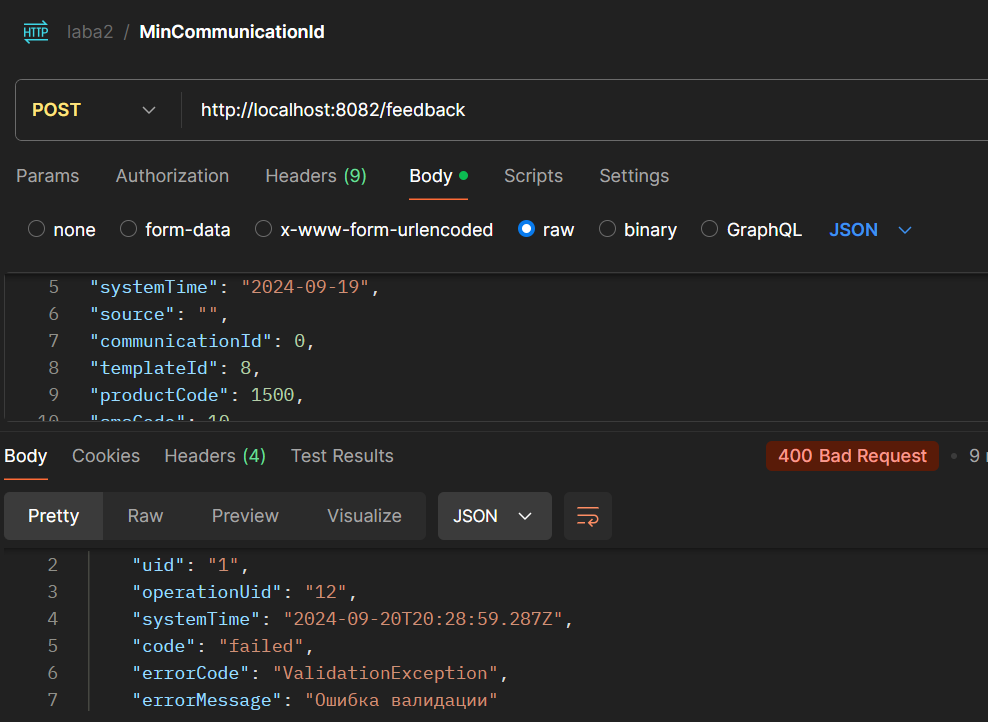
OperationUid больше 32-х символов:



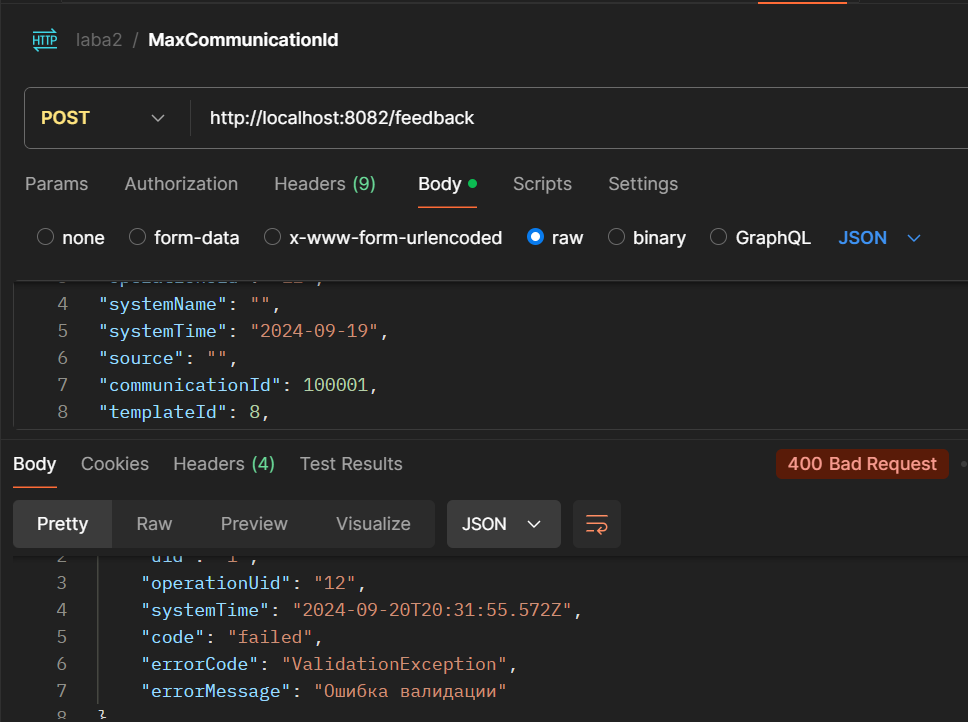
Пустой SystemTime:



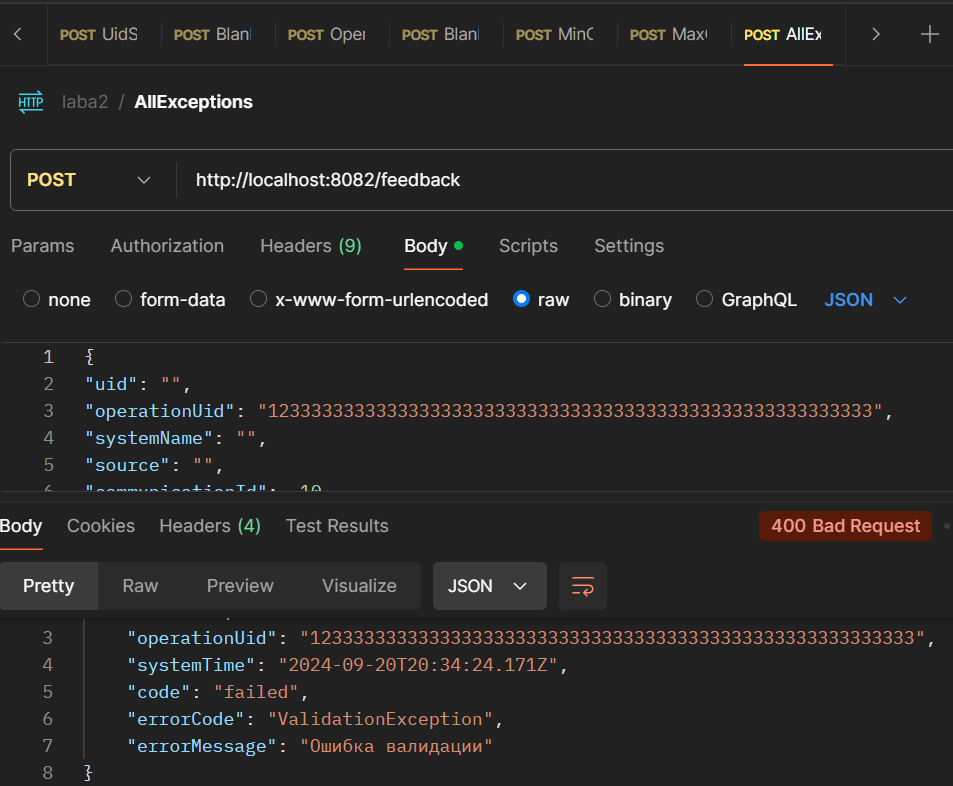
CommunicationId меньше 1:



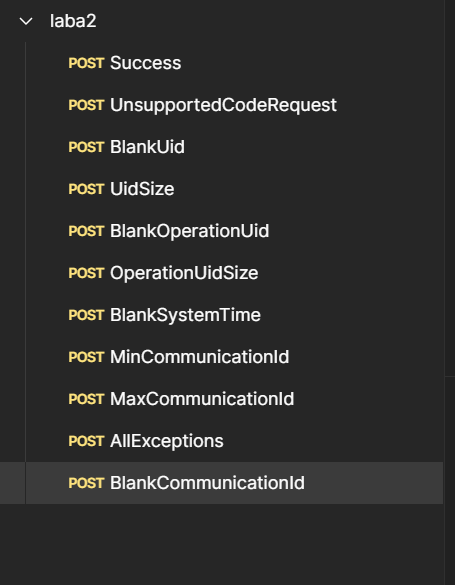
CommunicationId больше 100000:



Все кейсы:



Каталог:



1. Собственное исключение

package vs.korzhina.SpringBootApp.exception;

public class UnsupportedCodeException extends Exception {

public UnsupportedCodeException(String message){

super(message);

}

}

package vs.korzhina.SpringBootApp.service;

import org.springframework.stereotype.Service;

import org.springframework.validation.BindingResult;

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

import vs.korzhina.SpringBootApp.exception.\*;

import vs.korzhina.SpringBootApp.model.Request;

@Service

public interface IValidationService {

void isValid(BindingResult bindingResult) throws ValidationFailedException;

void isUnsupportedCodeException(@RequestBody Request request) throws UnsupportedCodeException;

}

package vs.korzhina.SpringBootApp.service;

import org.springframework.stereotype.Service;

import org.springframework.validation.BindingResult;

import vs.korzhina.SpringBootApp.exception.UnsupportedCodeException;

import vs.korzhina.SpringBootApp.exception.ValidationFailedException;

import vs.korzhina.SpringBootApp.model.Request;

import vs.korzhina.SpringBootApp.model.Response;

@Service

public class RequestValidationService implements IValidationService {

@Override

public void isValid(BindingResult bindingResult) throws ValidationFailedException {

if (bindingResult.hasErrors())

throw new

ValidationFailedException(bindingResult.getFieldError().toString());

}

@Override

public void isUnsupportedCodeException(Request request) throws UnsupportedCodeException {

if (request.getUid().equals("123"))

throw new UnsupportedCodeException("Не поддерживаемая ошибка");

}

}

package vs.korzhina.SpringBootApp.controller;

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

import jakarta.validation.Valid;

import vs.korzhina.SpringBootApp.exception.UnsupportedCodeException;

import vs.korzhina.SpringBootApp.exception.ValidationFailedException;

import vs.korzhina.SpringBootApp.model.Request;

import vs.korzhina.SpringBootApp.model.Response;

import vs.korzhina.SpringBootApp.service.IValidationService;

import java.text.SimpleDateFormat;

import java.util.Date;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.validation.BindingResult;

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

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

@RestController

public class MyController {

private final IValidationService validationService;

@Autowired

public MyController(IValidationService validationService) {

this.validationService = validationService;

}

@PostMapping(value="/feedback")

public ResponseEntity<Response> feedback(@Valid @RequestBody Request request,

BindingResult bindingResult) {

SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS'Z'");

Response response = Response.builder()

.uid(request.getUid())

.operationUid(request.getOperationUid())

.systemTime(simpleDateFormat.format(new Date()))

.code("success")

.errorCode("")

.errorMessage("")

.build();

try {

validationService.isValid(bindingResult);

validationService.isUnsupportedCodeException(request);

} catch (UnsupportedCodeException e) {

response.setCode("failed");

response.setErrorCode("UnsupportedCodeException");

response.setErrorMessage(e.getMessage());

return new ResponseEntity<>(response, HttpStatus.BAD\_REQUEST);

} catch (ValidationFailedException e) {

response.setCode("failed");

response.setErrorCode("ValidationException");

response.setErrorMessage("Ошибка валидации");

return new ResponseEntity<>(response, HttpStatus.BAD\_REQUEST);

} catch(Exception e){

response.setCode("failed");

response.setErrorCode("UnknownException");

response.setErrorMessage("Произошла непредвиденная ошбика");

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

}

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

}

}

GitHub:https://github.com/Kuznecova2033

**2.3 Ответы на вопросы**

1. Какое содержимое может передаваться POST запросом?

POST запросы могут передавать различные типы данных, включая текст, JSON, XML, бинарные данные, файлы и другие форматы. Это позволяет отправлять более сложные структуры данных, такие как объекты, массивы или файлы, в теле запроса. В контексте Spring Boot часто используется передача JSON через тело POST-запроса для взаимодействия с REST API.

2. Зачем нужны кастомные исключения?

Кастомные исключения используются для того, чтобы более точно описывать ошибки, которые могут возникать в приложении. Они помогают улучшить читаемость и поддержку кода, предоставляя разработчикам возможность создавать исключения, которые лучше отражают специфические ошибки приложения, такие как валидация данных или нарушения бизнес-логики.

3.Что означает аннотация @RestController?

Аннотация @RestController в Spring Boot обозначает класс как контроллер для обработки HTTP-запросов. Она объединяет функциональность @Controller и @ResponseBody, что означает, что каждый метод этого класса возвращает данные непосредственно в теле HTTP-ответа (чаще всего в формате JSON), а не требует возвращать представление (как в случае с MVC-контроллерами). @RestController обычно используется для построения RESTful API.

**ВЫВОД**

В ходе работы был реализовано простой REST сервис, написан дополнительный функционал к нему, протестирован в Postman, и были получены ответы на вопросы.

Задания выполнены, цель достигнута.