Лабораторная работа №2

По предмету Java

Выполнил: Нечман Дмитрий Андреевич,  
 РИМ-220971

Екатеринбург

2023

**Цель работы:** Разработать простой REST сервис

2. Реализация простого REST сервиса с валидацией и тестирование

Реализация достаточно стандартная, поэтому её скринить не стал, тестирование:

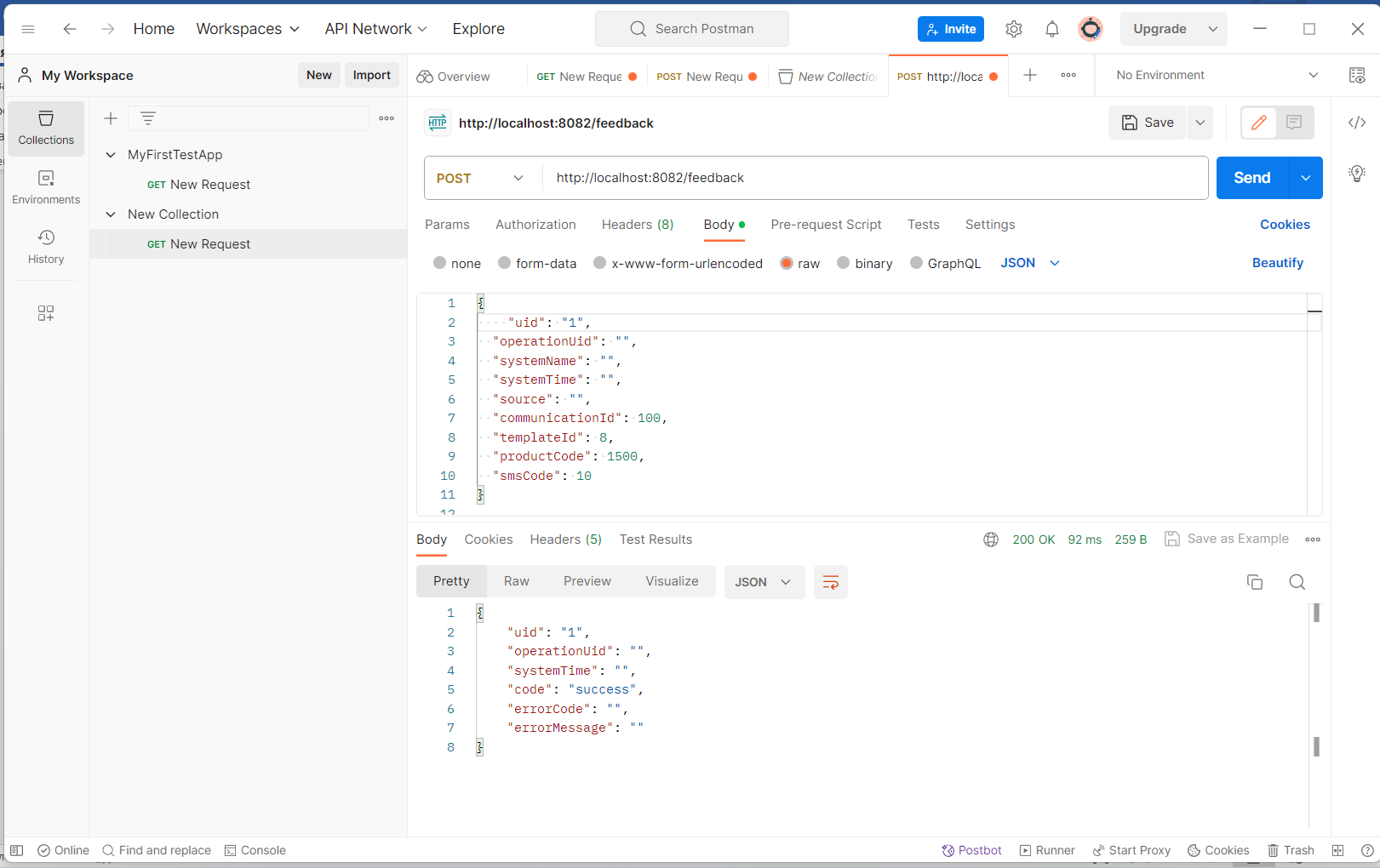


Рисунок – Тестирование базового приложения с помощью Postman

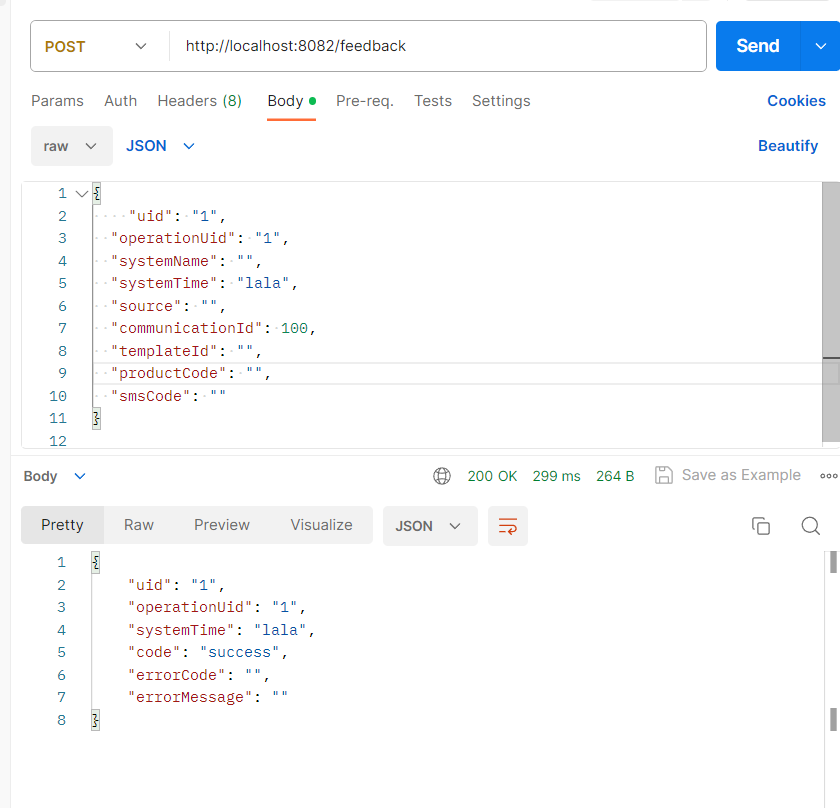
3. Реализация дополнительного функционала

3.1 Добавьте валидацию полей в соответствии с п 1.1 в таблице   
с описанием request, колонка обязательность.

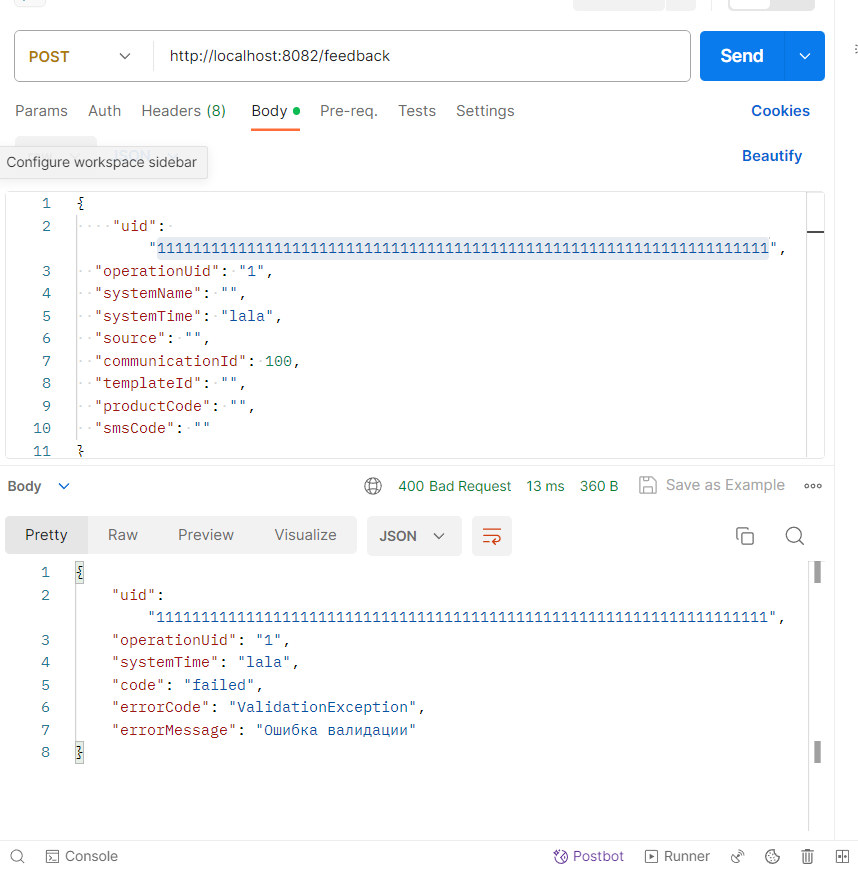
import javax.validation.constraints.Min;  
import javax.validation.constraints.NotBlank;  
import javax.validation.constraints.Size;  
  
  
@Data  
@Builder  
@NoArgsConstructor  
@AllArgsConstructor  
public class Request {  
  
 @NotBlank(message = "Uid is required")  
 @Size(max = 32, message = "Uid cannot exceed 32 characters")  
 private String uid;  
  
 @NotBlank(message = "OperationUid is required")  
 @Size(max = 32, message = "OperationUid cannot exceed 32 characters")  
 private String operationUid;  
  
 private String systemName;  
 @NotBlank(message = "SystemTime is required")  
 private String systemTime;  
  
 private String source;  
 @Min(value = 1, message = "CommunicationId must be at least 1")  
 @Max(value = 100000, message = "CommunicationId cannot exceed 100000")  
 private int communicationId;  
 private int templateId;  
 private int productCode;  
 private int smsCode;  
  
}

2. Тест-кейсы:

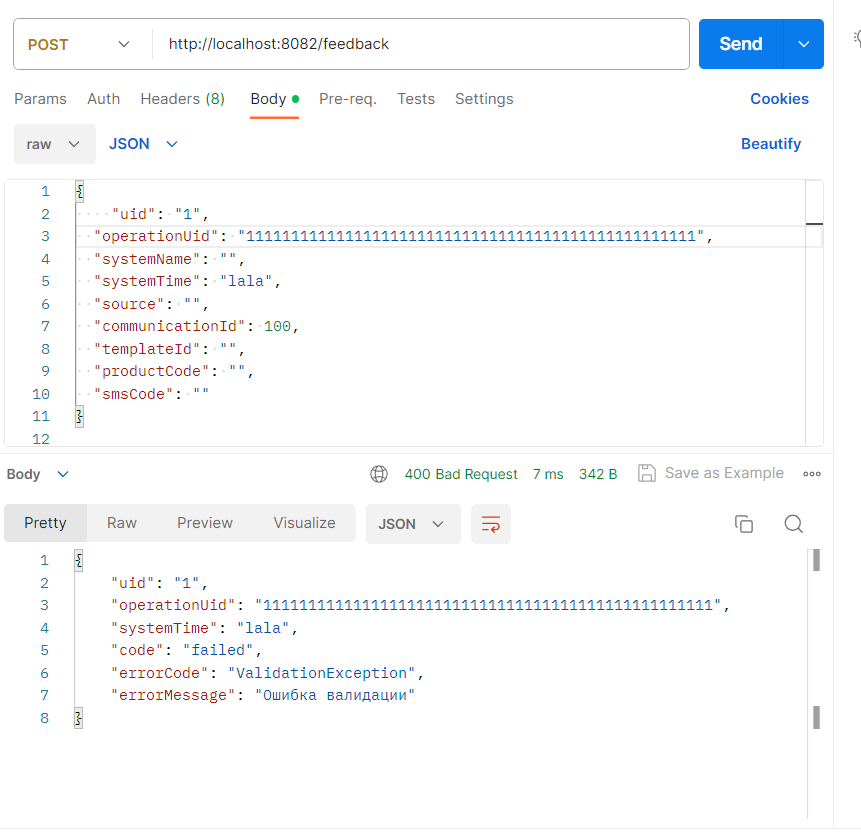
1. Все корректно



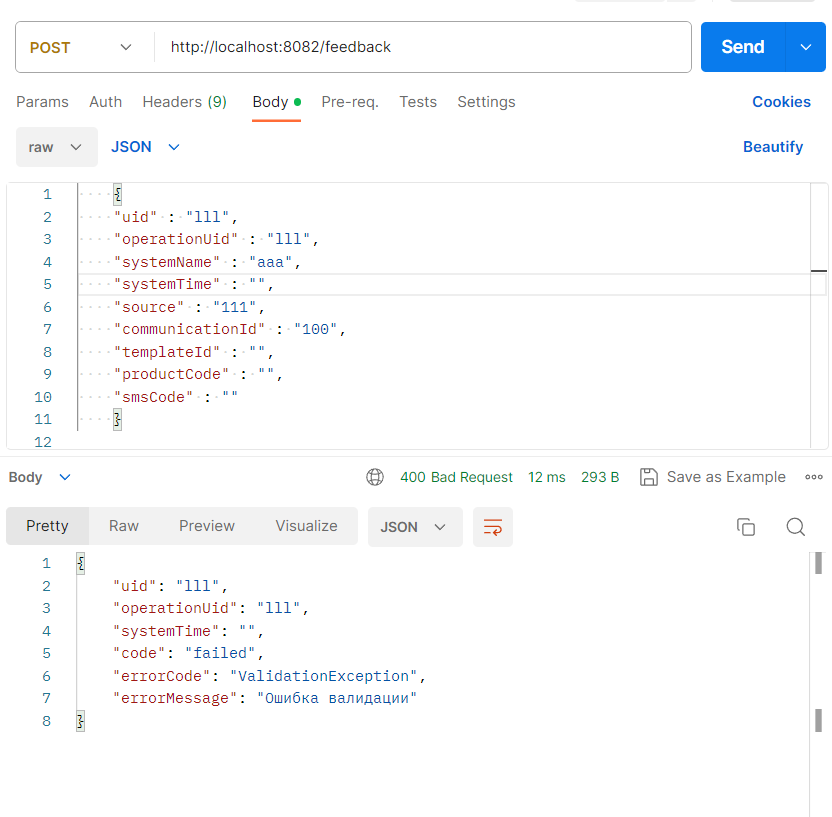
1. Некорретный uid(более 32 символов)



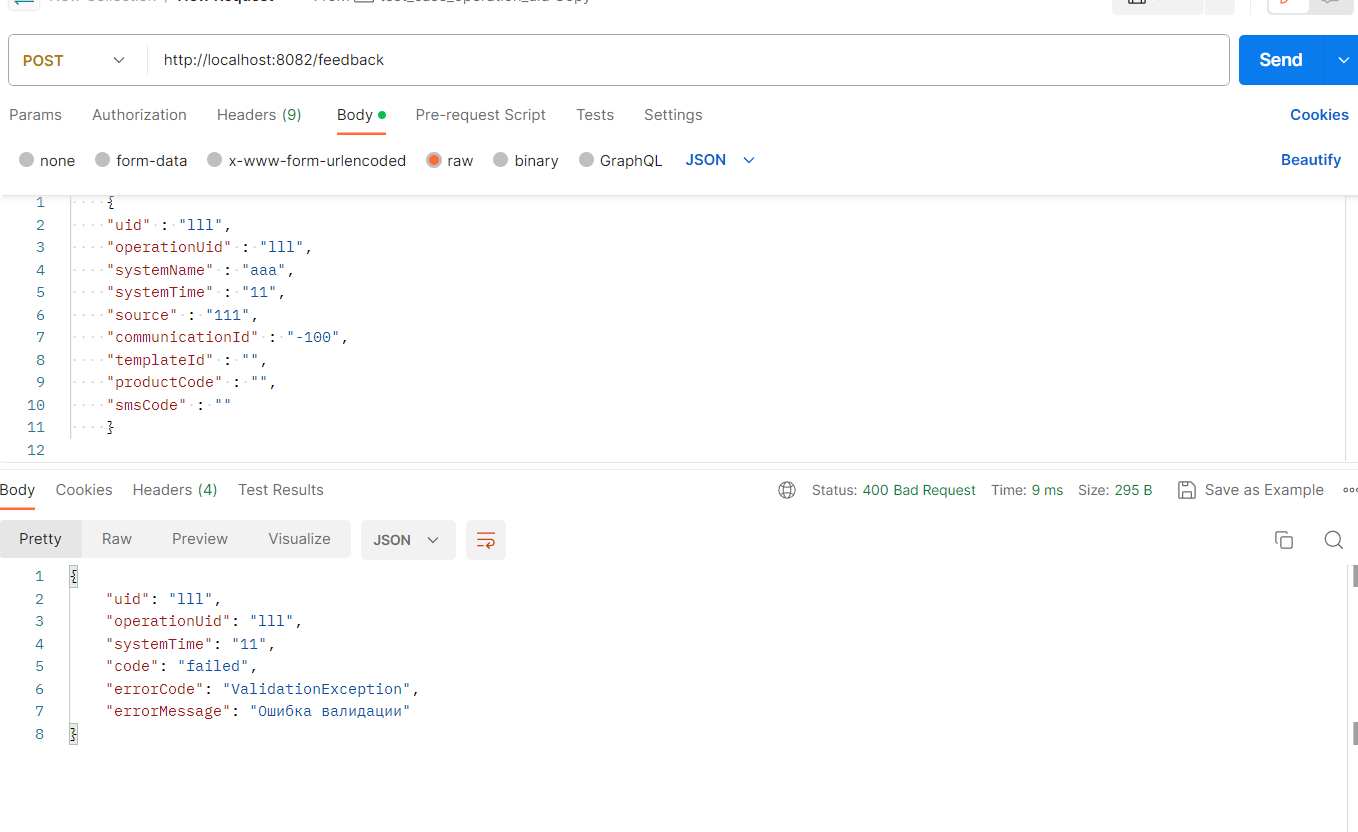
1. Некорректный operationUid (более 32 символов)



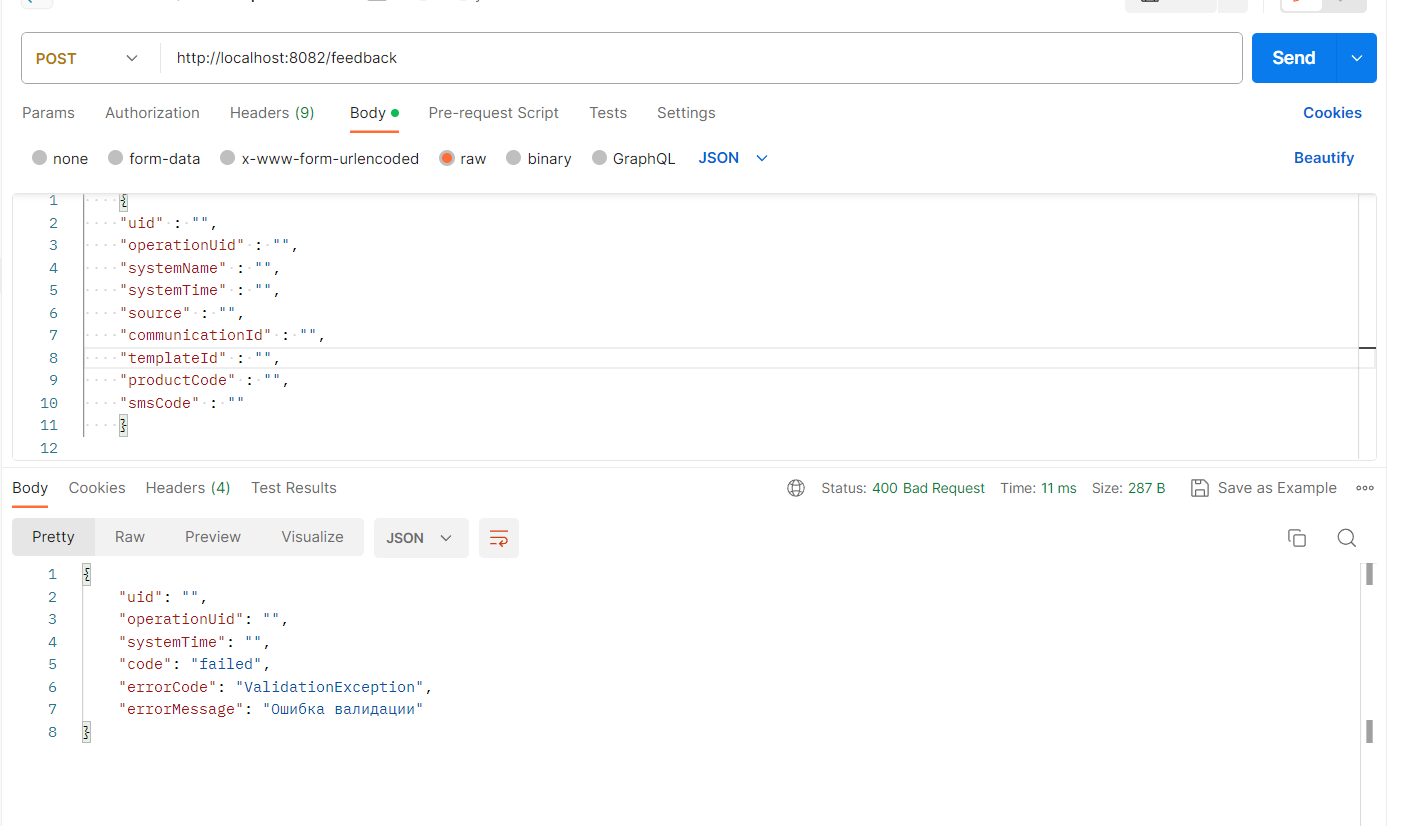
1. Некорректный systemTime(его нет)



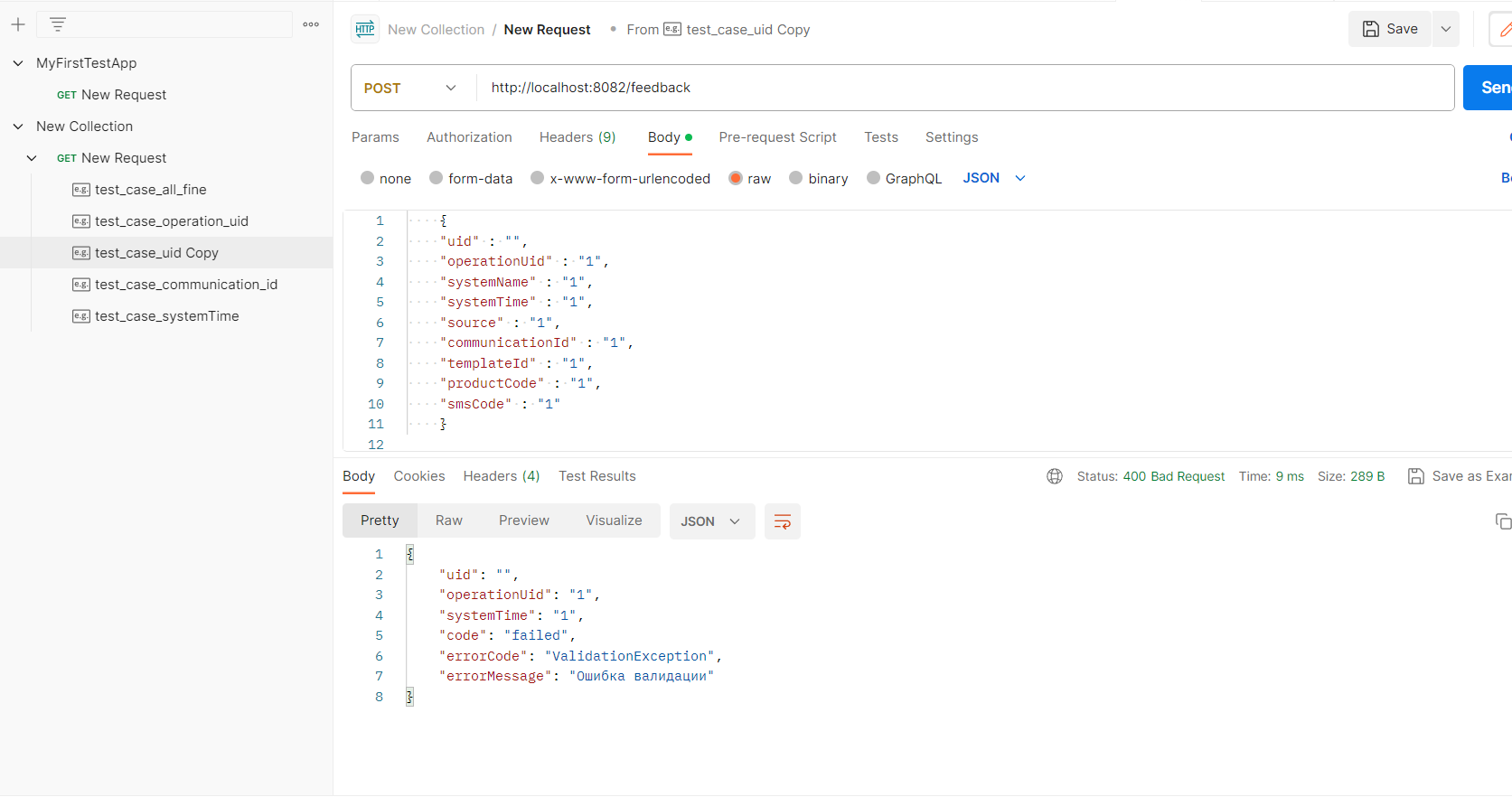
1. Некорректный communicationId( меньше минимума)



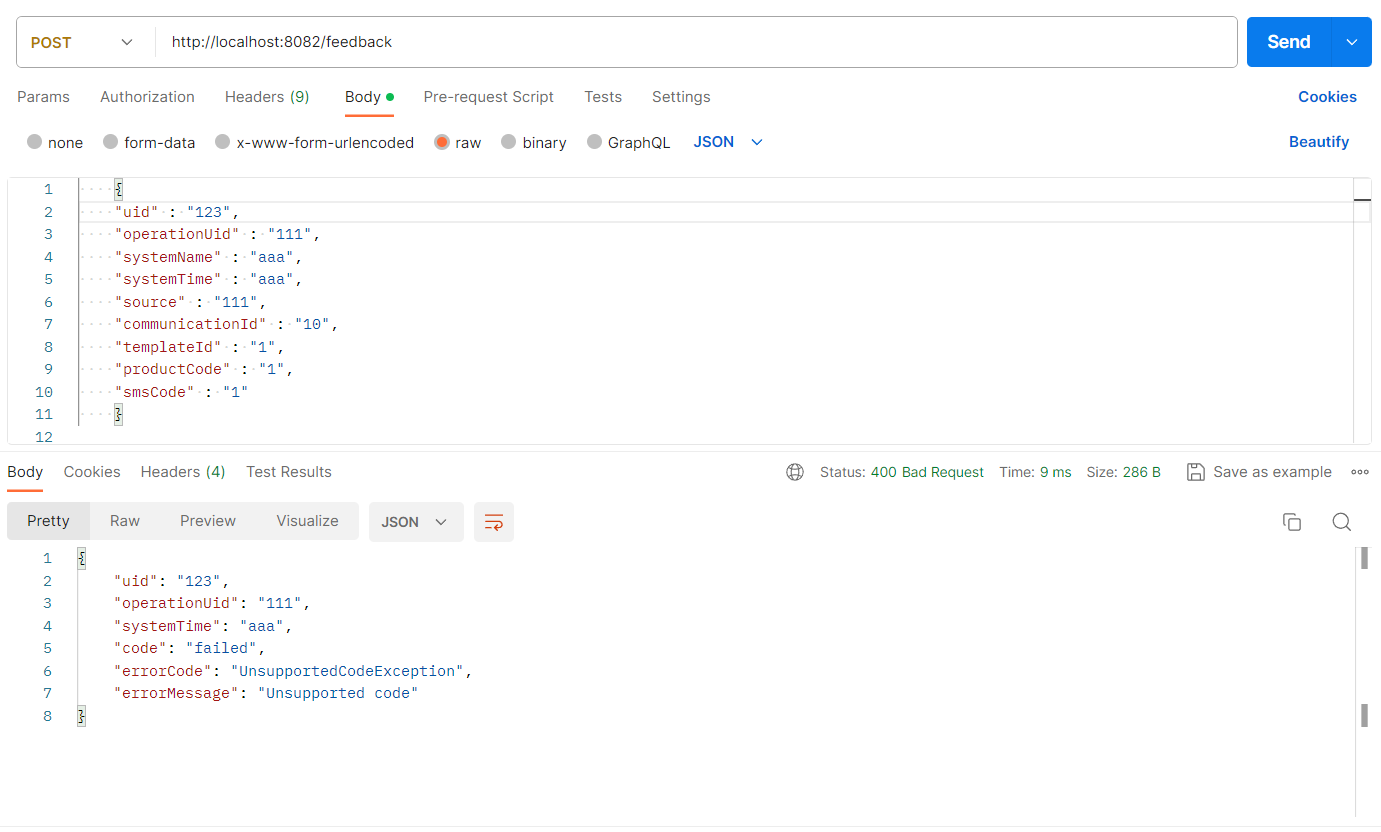
1. Все вышеперечисленное одновременно (кроме п.1)



Тест-кейсы:

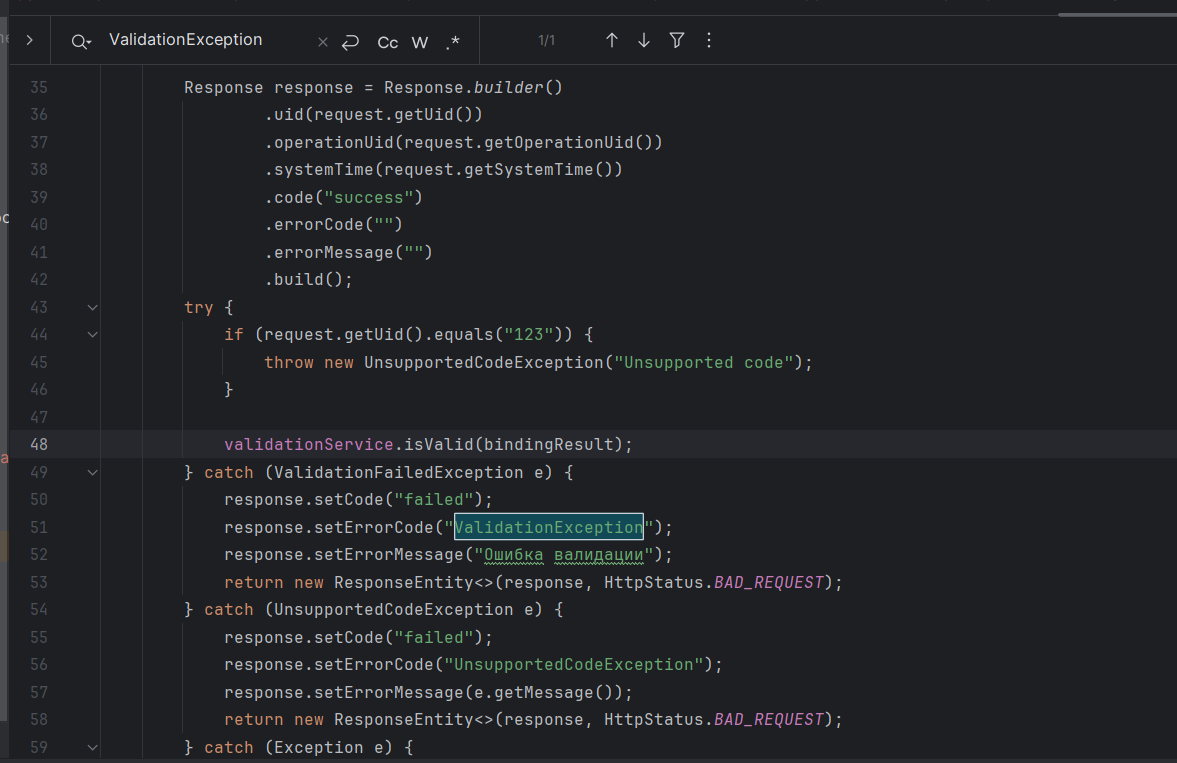


3. Создайте собственное исключение UnsupportedCodeException, выбрасывайте его в случае, когда uid равен 123, добавьте код ошибки в ответное сообщение, приложите скриншот выполненного теста.



Реализация:

package ru.arkhipov.MySecondTestAppSpringBoot.exception;  
  
  
public class UnsupportedCodeException extends Exception {  
  
 public UnsupportedCodeException(String message) {  
 super(message);  
 }  
}



4 Вопросы

2. Какое содержимое может передаваться POST запросом?

POST запросы могут передавать разнообразные данные, включая:

1. Форму HTML: POST запросы могут использоваться для отправки данных, введенных веб-формой на веб-странице. Эти данные обычно отправляются в форме application/x-www-form-urlencoded и могут быть легко обработаны с использованием аннотации @RequestParam или объектов MultiValueMap в Spring.

2. JSON: Вы можете отправлять данные в формате JSON, который будет автоматически преобразован в объект Java с использованием аннотации @RequestBody.

3. XML: Если вы хотите передать данные в формате XML, вы можете использовать аннотацию @RequestBody и настроить Marshalling и Unmarshalling XML данных.

4. Многочисленные параметры (Multipart): POST запросы также могут использоваться для загрузки файлов и данных с использованием enctype="multipart/form-data". Вы можете использовать аннотацию @RequestPart или @RequestParam для обработки таких данных.

5. Бинарные данные: Вы можете отправить бинарные данные, такие как изображения или другие файлы, в теле POST запроса.

6. Параметры запроса: Вы также можете передавать параметры запроса в POST запросе, хотя это не является наилучшей практикой для чувствительных данных.

4. Зачем нужны кастомные исключения?

Кастомные исключения (custom exceptions) в приложениях на Spring Boot предоставляют возможность более гибкого и четкого управления ошибками. Вот несколько причин, по которым они могут быть полезными:

Четкость исключений:

Когда вы создаете свои собственные исключения, вы можете явно указать, что именно пошло не так в вашем приложении. Это помогает быстро идентифицировать проблему и принимать соответствующие меры.

Иерархия исключений:

Вы можете создавать иерархии кастомных исключений, что облегчает обработку различных видов ошибок. Например, у вас может быть базовый класс исключений и несколько подклассов, представляющих конкретные виды ошибок.

Кастомные сообщения об ошибках:

Вы можете предоставить подробные и информативные сообщения об ошибках, которые могут быть использованы для журналирования, мониторинга или отправки пользователю в случае необходимости. Это помогает в диагностике и устранении проблем.

Контрольный уровень:

Кастомные исключения могут использоваться для указания уровня критичности ошибки. Например, у вас может быть критическая ошибка, которую нужно обработать срочно, и менее критичная, которую можно игнорировать или обработать в фоновом режиме.

Обработка ошибок на уровне приложения:

Spring Boot предоставляет механизмы обработки исключений на уровне приложения с использованием аннотаций @ControllerAdvice и @ExceptionHandler. Это позволяет централизованно управлять обработкой исключений и возвращать клиентам приложения информацию об ошибках в удобном формате.

1. Что означает аннотация @RestController?

Аннотация @RestController в Spring Boot обозначает класс как контроллер, который обрабатывает HTTP-запросы и возвращает результаты в виде данных, обычно в формате JSON. Это одна из ключевых аннотаций в Spring Framework, которая облегчает создание RESTful веб-сервисов.

Когда вы помечаете класс аннотацией @RestController, Spring Boot автоматически выполняет следующее:

1. Автоматически добавляет аннотацию @Controller к классу, указывая, что это контроллер.

2. Автоматически добавляет аннотацию @ResponseBody ко всем методам в классе. Это означает, что результаты методов будут автоматически сериализованы в формат JSON и отправлены обратно клиенту.

3. Автоматически обрабатывает маршруты (URL-ы) для методов контроллера.

**Вывод:** в результате работы ознакомился **с** Spring Boot и тем, как создать простейшее веб-приложение с помощью фреймворка.

**Ссылка на репозиторий:**

https://github.com/NechD/Java\_homework/tree/master/2%20курс