**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное автономное образовательное учреждение

высшего образования «Уральский федеральный университет

имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий - РТФ

Центр ускоренного обучения

Отчет по лабораторной работе № 5

по дисциплине «Технологии Программирования»

Старший преподаватель: Н.А. Архипов

Студент гр. РИВ-220938у М.С. Ермаков

Екатеринбург 2023

**1. Цель работы:**

**Цель:** Доработать простой REST сервис и добавить unit тесты

**Учебные вопросы:**

1) Доработка Сервиса 1 - изменение класса Request и добавления метода расчета заработной платы.

2) Доработка Сервиса 1 для того, чтобы он мог пересылать полученный и модифицированный request.

3) Самостоятельно доработать приложение в соответствии с заданием.

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. Описание задачи:**

Реализация дополнительного функционала

Необходимо самостоятельно доработать приложение:

1. Доработайте метод calculate, чтобы в нем вычислялось количество дней в году, в зависимости от того високосный год или нет.

2. Расширьте enum Positions, добавьте три дополнительных позиции, а также добавьте поле isManager с типом данных boolean, для менеджеров установите значение true, в прочих случаях false.

3. Напишите метод, вычисляющий квартальную премию, метод работает только для менеджеров и иных управленцев (те, у кого isManager == true), добавьте проверку в метод, на признак управленец это или нет.

4. Разработайте unit тест для метода расчета квартальной премии.

5. Произведите рефакторинг приложения – добавьте комментарии к каждому полю в классах Request и Response в соответствии с описанием из лабораторной работы №2, пример комментария к полю показан на рисунке 1 текущей работы.

6. Сделайте так, чтобы в классе MyController стало меньше кода, т.е. приберитесь в нем.

Вопросы

Письменно ответьте на вопросы, если ваш номер в списке группы четный, то нужно ответить на четные вопросы, если нечетный, то нужно ответить на нечетные вопросы.

1. Что такое unit тестирование?

2. Что такое smoke тестирование?

3. Что такое интеграционное тестирование?

4. Какие библиотеки используются в Spring Boot для unit тестирования?

5. Что такое Assert? Какие бывают Assert?

6. Какие аннотациия наиболее распространены при использовании библиотеки JUnit 5?

**3. Ход выполнения:**

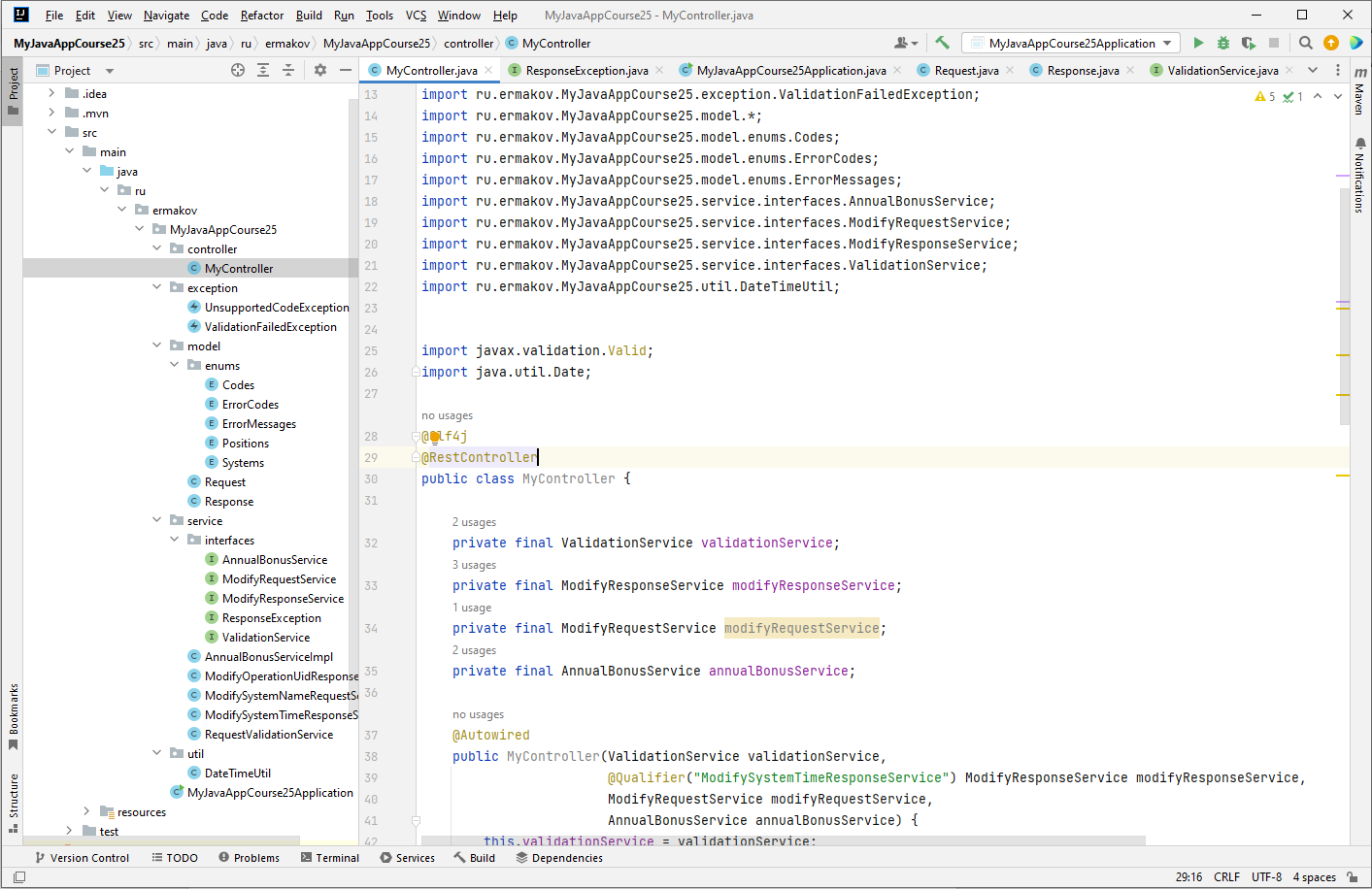


Рисунок 1 – Структура сервиса

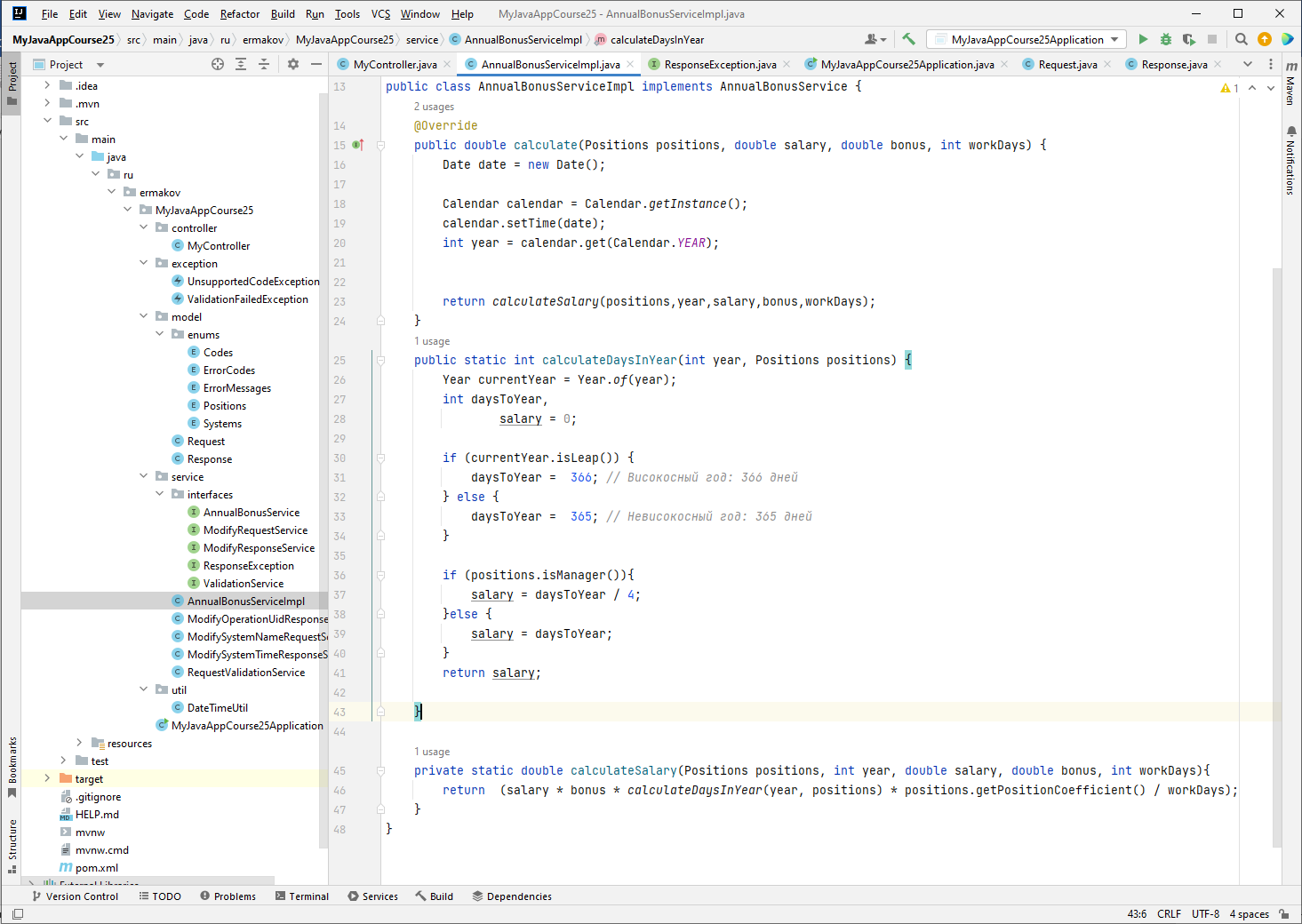


Рисунок 2 – Доработка метода calculate()

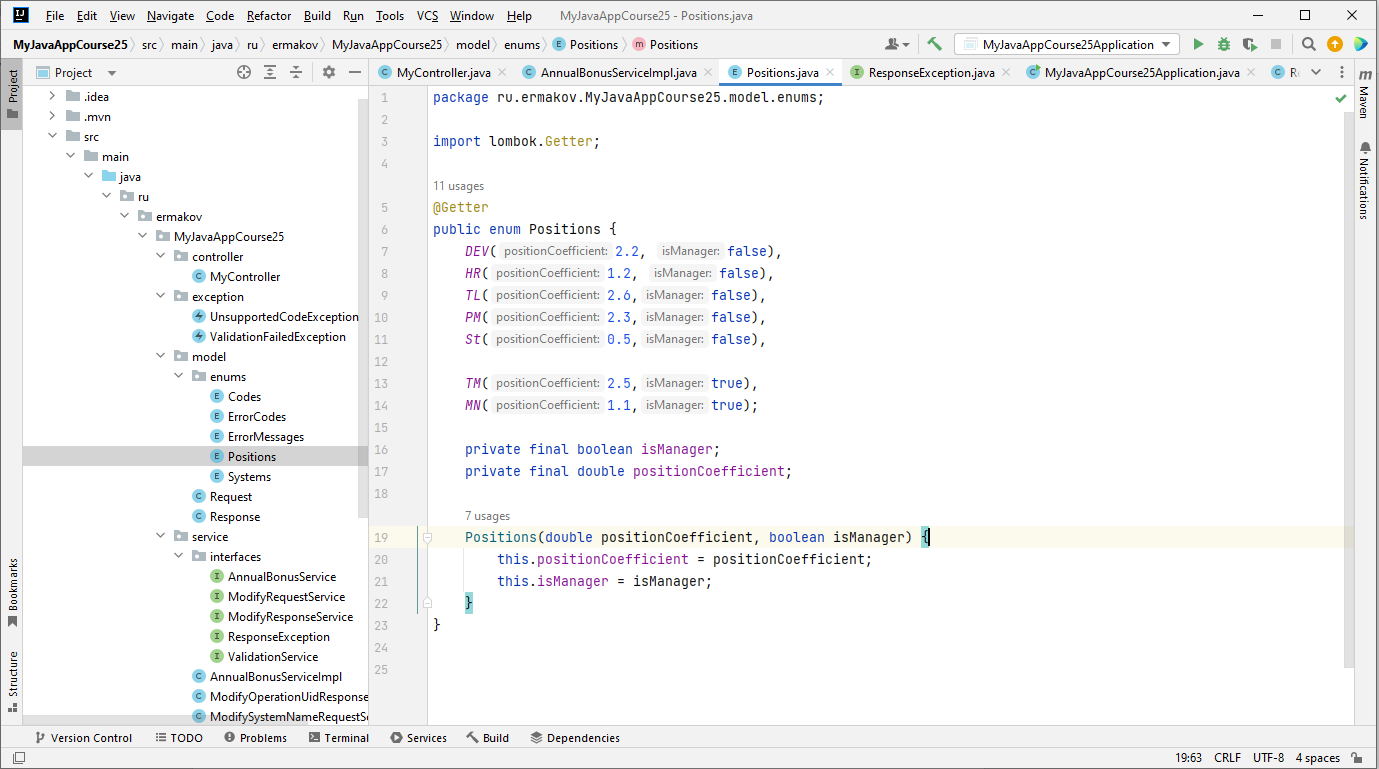


Рисунок 3 – Расширение enumа Positions, добавление поля isManager с типом данных boolean

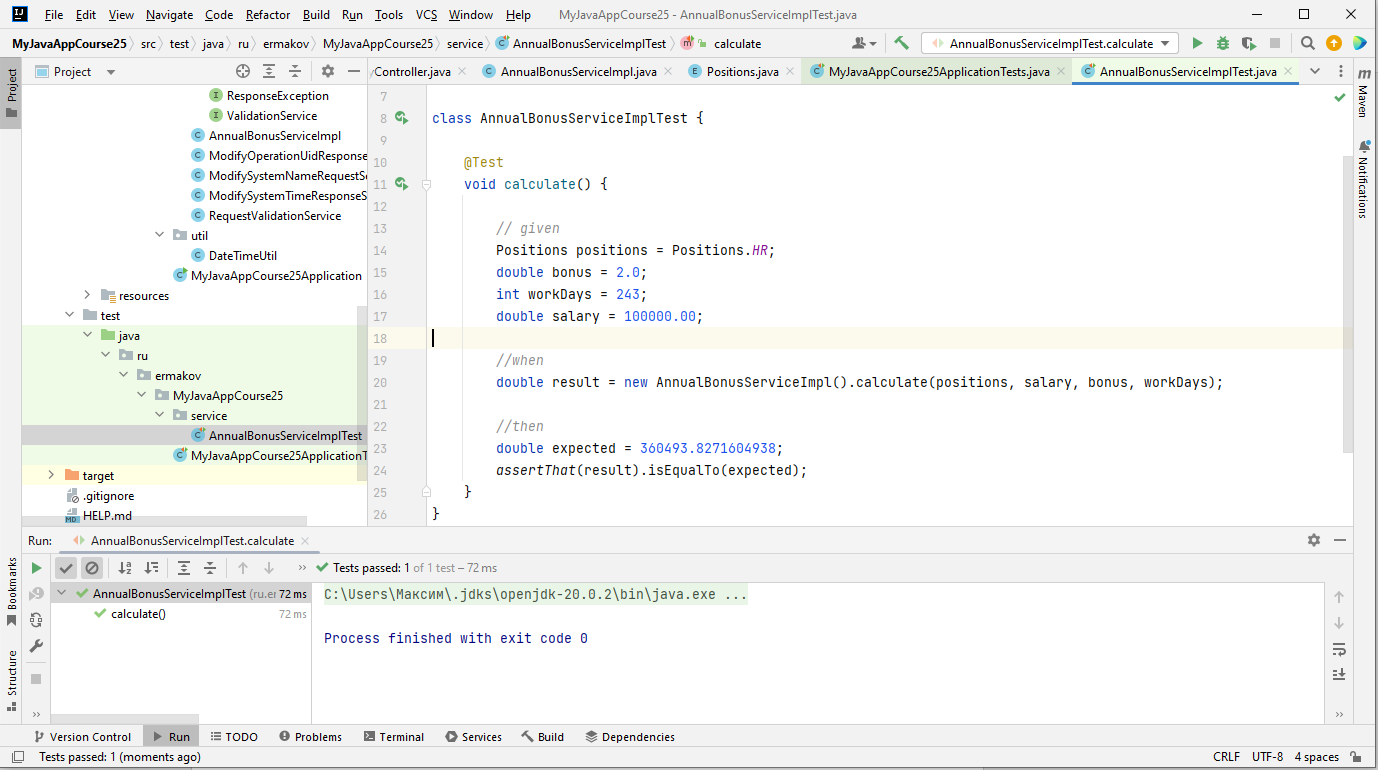


Рисунок 4 – Результат работы Unit теста на методе calculate()

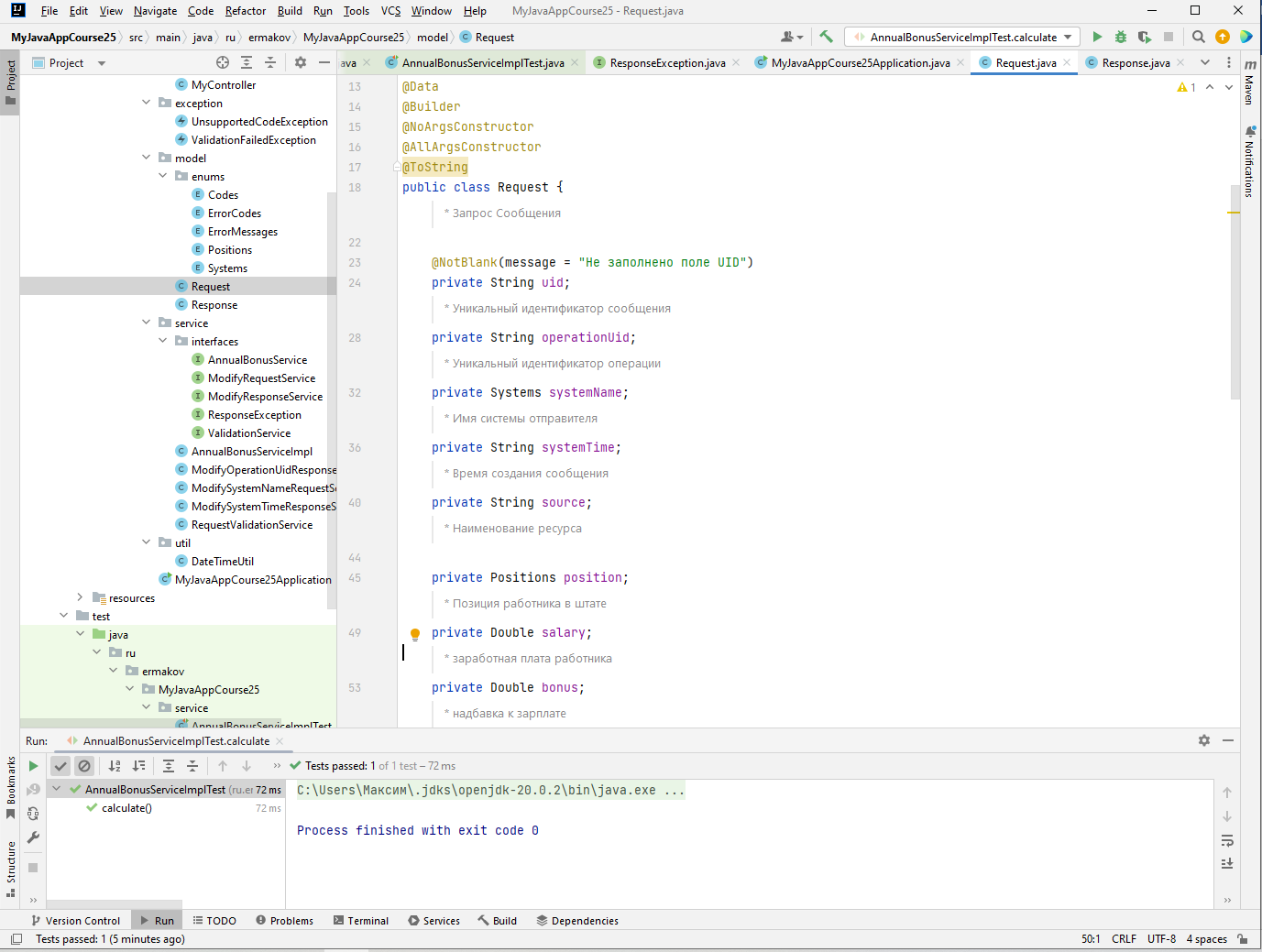


Рисунок 5 – Добавление комментариев в классе Request

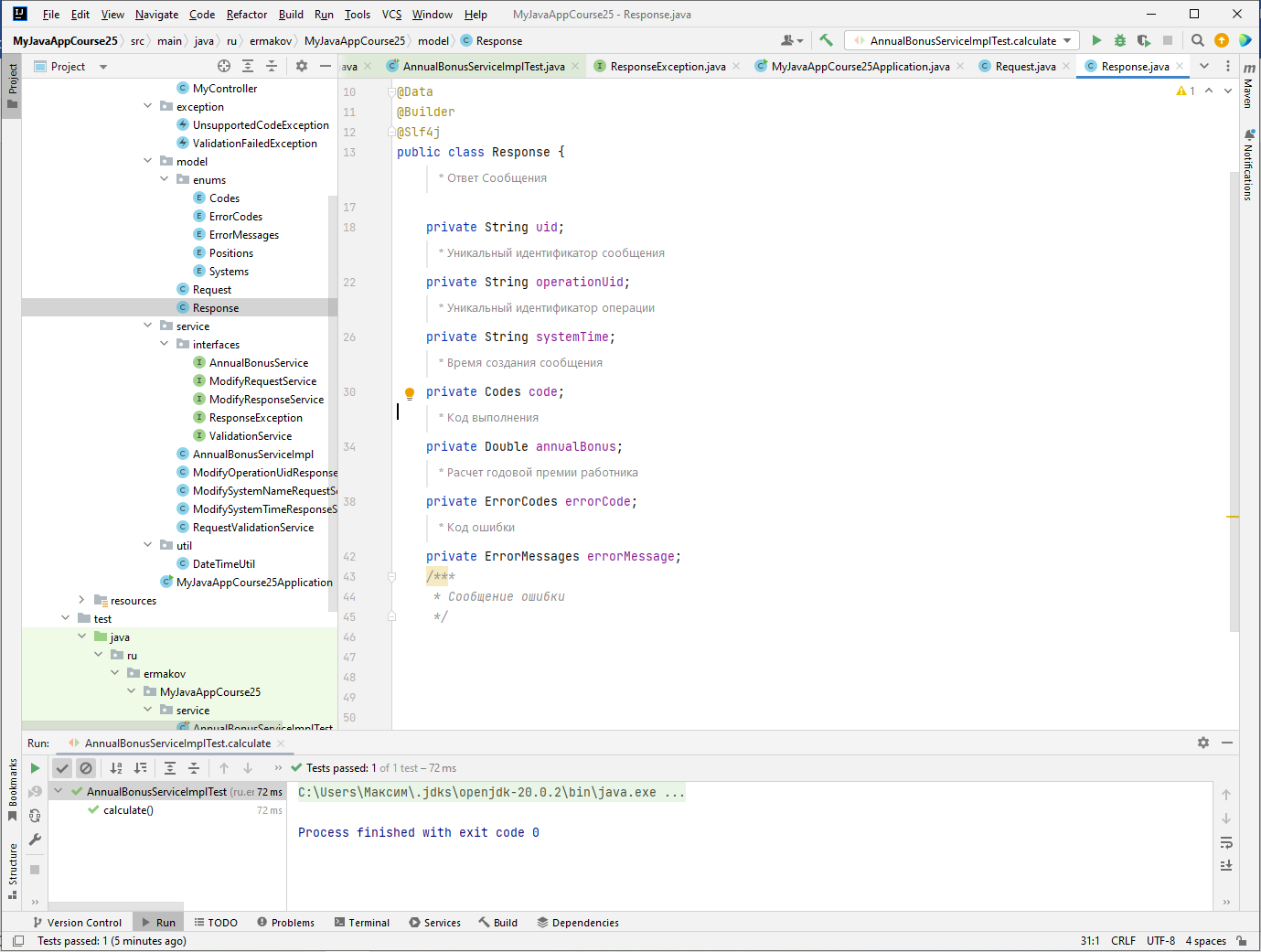


Рисунок 6 – Добавление комментариев в классе Response

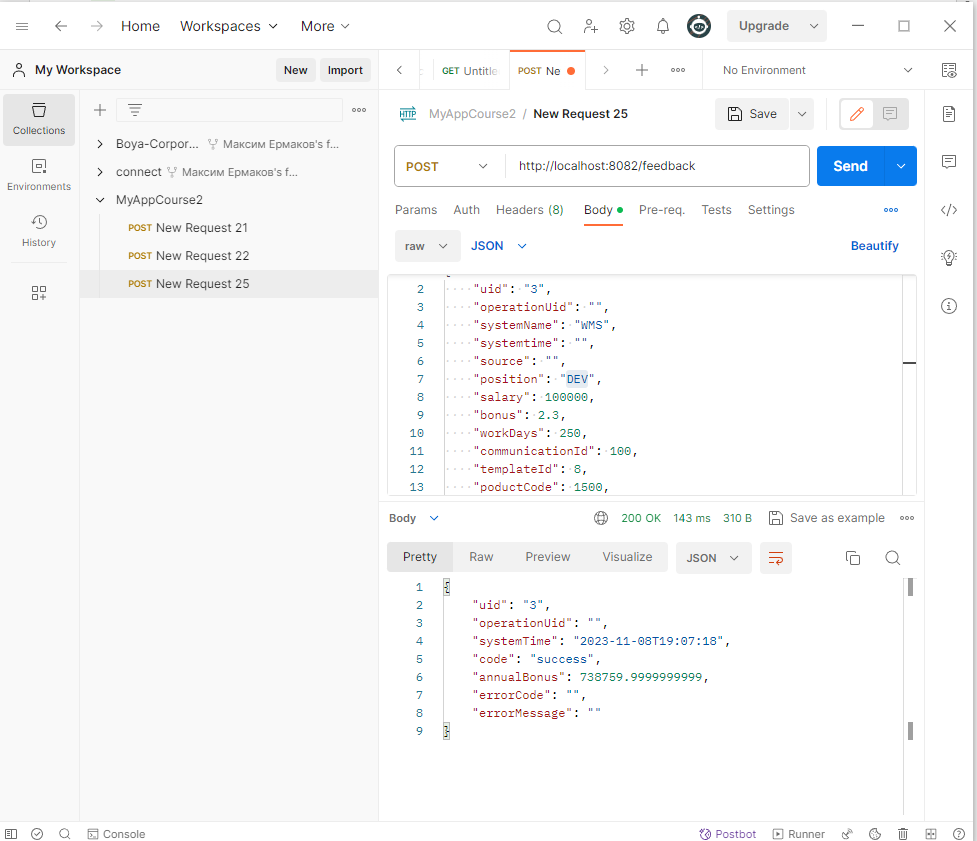


Рисунок 7 – Результат тестирования запроса POST в POSTMAN

**Ответы на вопросы:**

2 Что такое smoke тестирование?

Smoke-тестирование – проверка программного обеспечения на стабильность и наличие явных ошибок. Тест должен подтвердить или опровергнуть правильность выполнения ПО своих основных функций перед его передачей на более глубокое тестирование.

Сам термин появился впервые в среде инженеров, занимавшихся отопительным оборудованием. Когда изготавливалась новая печь или котел, все ее заглушки задвигались, после чего она затапливалась. Если дым шел только из положенных мест (например дымохода), то отопительное устройство было собрано правильно. Позже этот термин стал в ироничном ключе использоваться разработчиками микроэлектроники, которые проверяли устройства путем подачи напряжения на короткое время и таким образом выявляли сильно нагревающиеся компоненты.

Smoke-тест планируется, как правило, в начале проекта командой quality assurance (QA) инженеров в сотрудничестве с менеджером проекта, программистами, заказчиками и пользователями.

4. Какие библиотеки используются в Spring Boot для unit тестирования?

Модульное тестирование с помощью Spring Boot

Модульные тесты составляют основу нашей стратегии тестирования. Каждый проект Spring Boot, который мы запускаем с помощью Spring Initializr, имеет прочную основу для написания модульных тестов. Настраивать практически нечего, так как Spring Boot Starter Test включает в себя все необходимые строительные блоки.

Помимо включения и управления версией Spring Test, этот Spring Boot Starter включает и управляет версиями следующих библиотек:

* JUnit 4/5
* Mockito
* Библиотеки проверки утверждений, такие как AssertJ, Hamcrest, JsonPath и т. Д.

Мы можем найти введение в этот швейцарский нож тестирования и включенных в него библиотеках тестирования в этом сообщении блога.

В большинстве случаев наши модульные тесты не нуждаются в какой-либо конкретной функции Spring Boot или Spring Test, поскольку они будут полагаться исключительно на JUnit и Mockito.

С помощью модульных тестов мы изолированно тестируем, например, свои \*Service классы и имитируем каждого сотрудника тестируемого класса:

6) Какие аннотации наиболее распространены при использовании библиотеки JUnit 5?

Аннотации в JUnit – это специальные метки, которые Java-разработчик размещает перед методами в тестовом классе. Они позволяют настраивать процесс тестирования, указывая фреймворку, как именно их следует обрабатывать. Например, можно явно указать, какие из методов являются тестовыми случаями, какие из них выполнять перед тестами и после и так далее.

Вот несколько базовых аннотаций.

**@Test**. Эту аннотацию ставим перед методами, которые относятся к тестовым случаям. JUnit поймёт, что их следует выполнять в качестве теста, а по завершении проверить результат.

**@Before**. Используется для методов, которые должны быть выполнены перед каждым тестовым случаем. Например, если у нас есть несколько тестов, которые требуют одних и тех же начальных условий, мы можем обозначить метод с аннотацией @Before, задав необходимые условия тестирования один раз.

**@After**. Эту аннотацию используем перед методом, который должен быть выполнен после тестового случая.

**@BeforeClass**, **@AfterClass**. Методы с аннотацией **@BeforeClass** выполняются перед запуском первого теста в классе, а методы с аннотацией **@AfterClass** – после завершения всех тестов в классе.

**@Ignore**. Используется перед методом, чтобы отключить его выполнение в тесте. Это может быть полезно, если мы не уверены в работоспособности отдельных тестов и не хотим их использовать, но должны оставить в коде.

**@BeforeEach** и **@AfterEach**. Аналоги **@Before** и **@After** в JUnit 4.

**Ссылка** на репозиторий **github.com** =

<https://github.com/themakson/MyJavaAppCourse2/tree/main/MyJavaAppCourse25>

**Вывод:**

В ходе лабораторной работы 5 мы доработали созданный ранее простой REST сервис с помощью фреймворка Spring Boot в языке программирования Java, а также познакомились с тестированием своей программы, а именно через unit test, расширили данный сервис с дополнительными enums, произвели рефакторинг и упростили в кол–во кода нашу программу, а также произвели различного рода вычисления.

|  |  |
| --- | --- |
| Сдал: | М.С. Ермаков |
| Проверил: | Н.А. Архипов |