**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 №4**

**По дисциплине «Технологии программирования»**

Студент группы РИЗ-240938у Бабенко М.А.

Преподаватель: Архипов Н.А.

**Екатеринбург, 2025 г.**

**Лабораторная работа N4**

Цель: доработать простой REST сервис.

**Задания для самостоятельной работы**

**Задание 1**

Копируем папку с проектом из предыдущего задания, а также переименовываем главный метод; изменяем порты в application.properties. Для проверки делаем запрос через Postman – он успешный.

Результат запроса показан на рисунке 1.

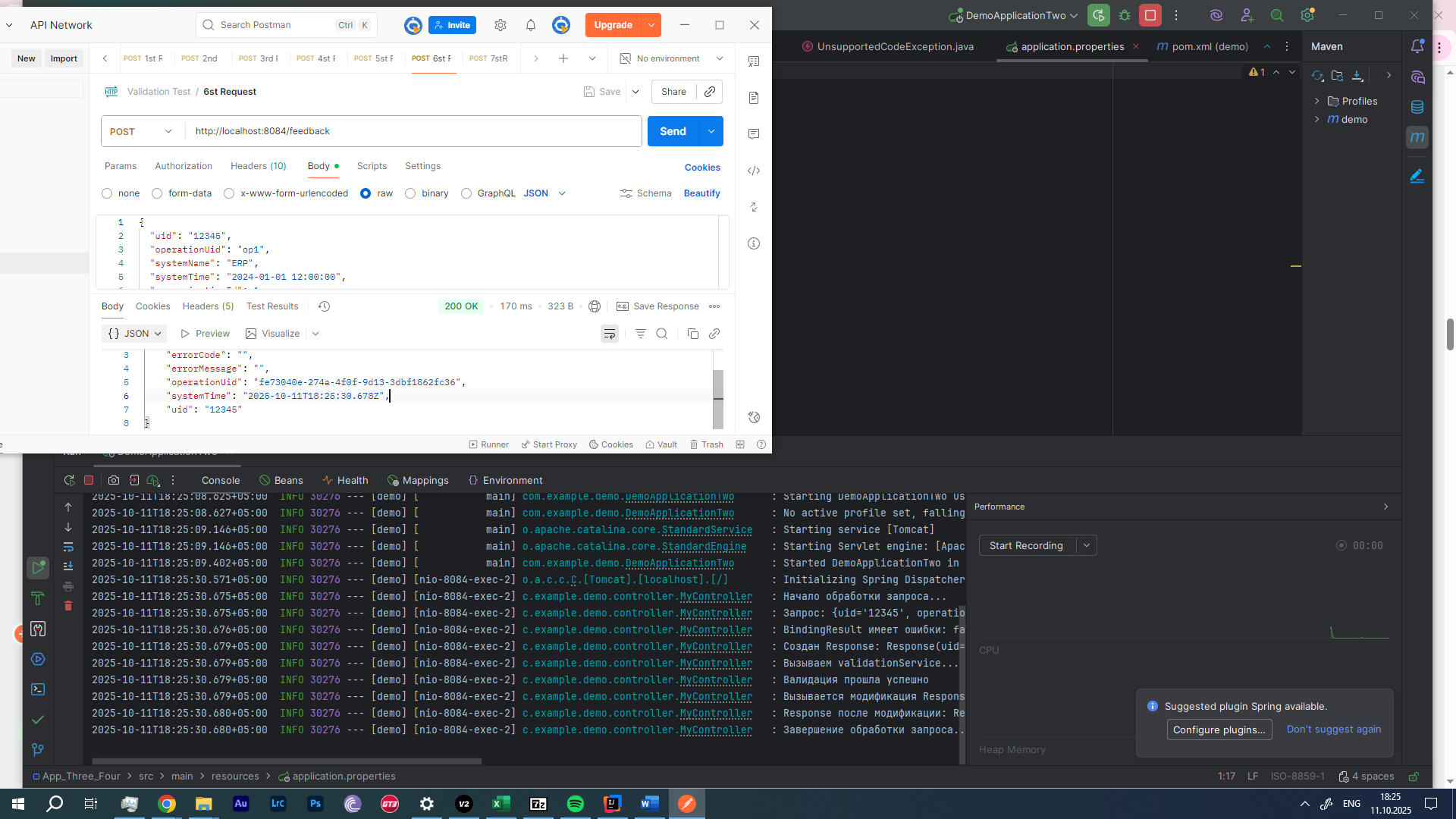


Рисунок 1 – Результат запроса через Postman

Затем, в первом проекте создаем еще два класса – ModifyRequestService и ModifySystemNameRequestService (один из них будет интерфейсом, а другой – сервисом). Попутно правим множество ошибок.

Созданные классы ModifyRequestService и ModifySystemNameRequestService показан на рисунке 2.

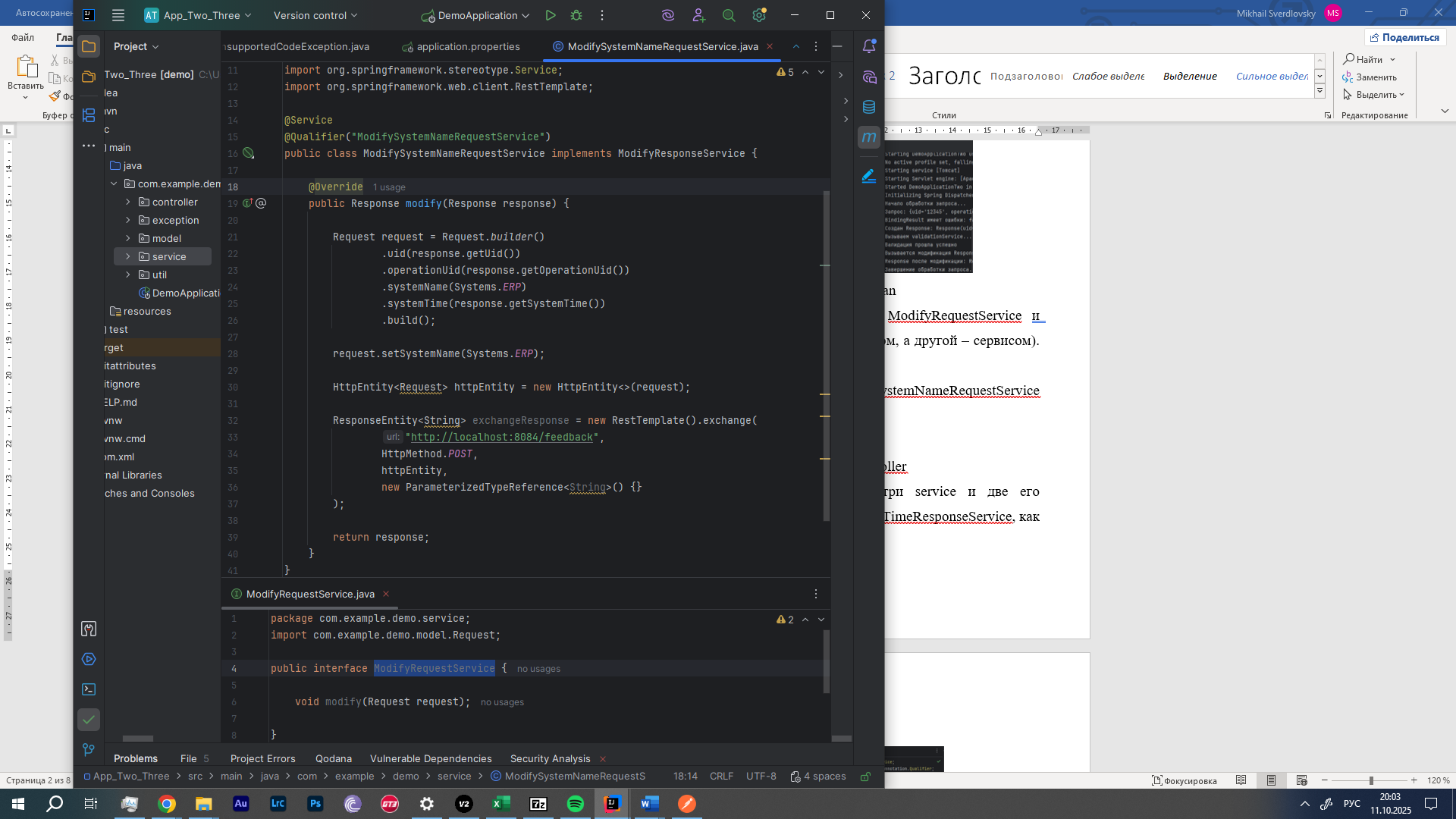


Рисунок 2 – Созданные классы

Затем, создаем интерфейс ModifyResponseService внутри service и две его реализации – ModifyOperationUidResponseService и ModifySystemTimeResponseService, как указано в задании.

Результат создания показан на рисунке 3.

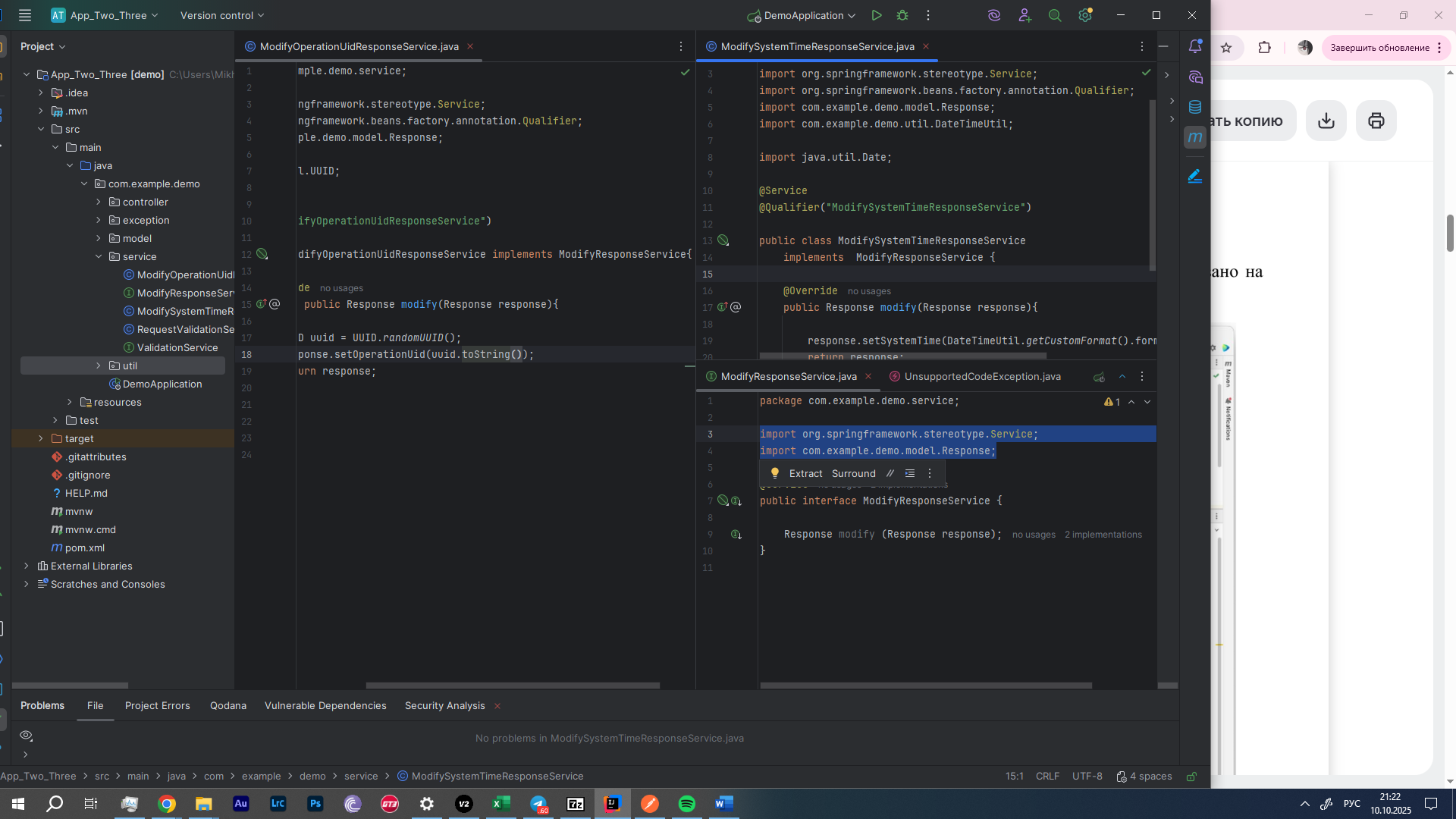


Рисунок 3 – Созданный интерфейс и реализации

Далее, внедряем зависимость на сервис ModifyRequestService в класс MyController.

Результат внедрения показан на рисунке 4.

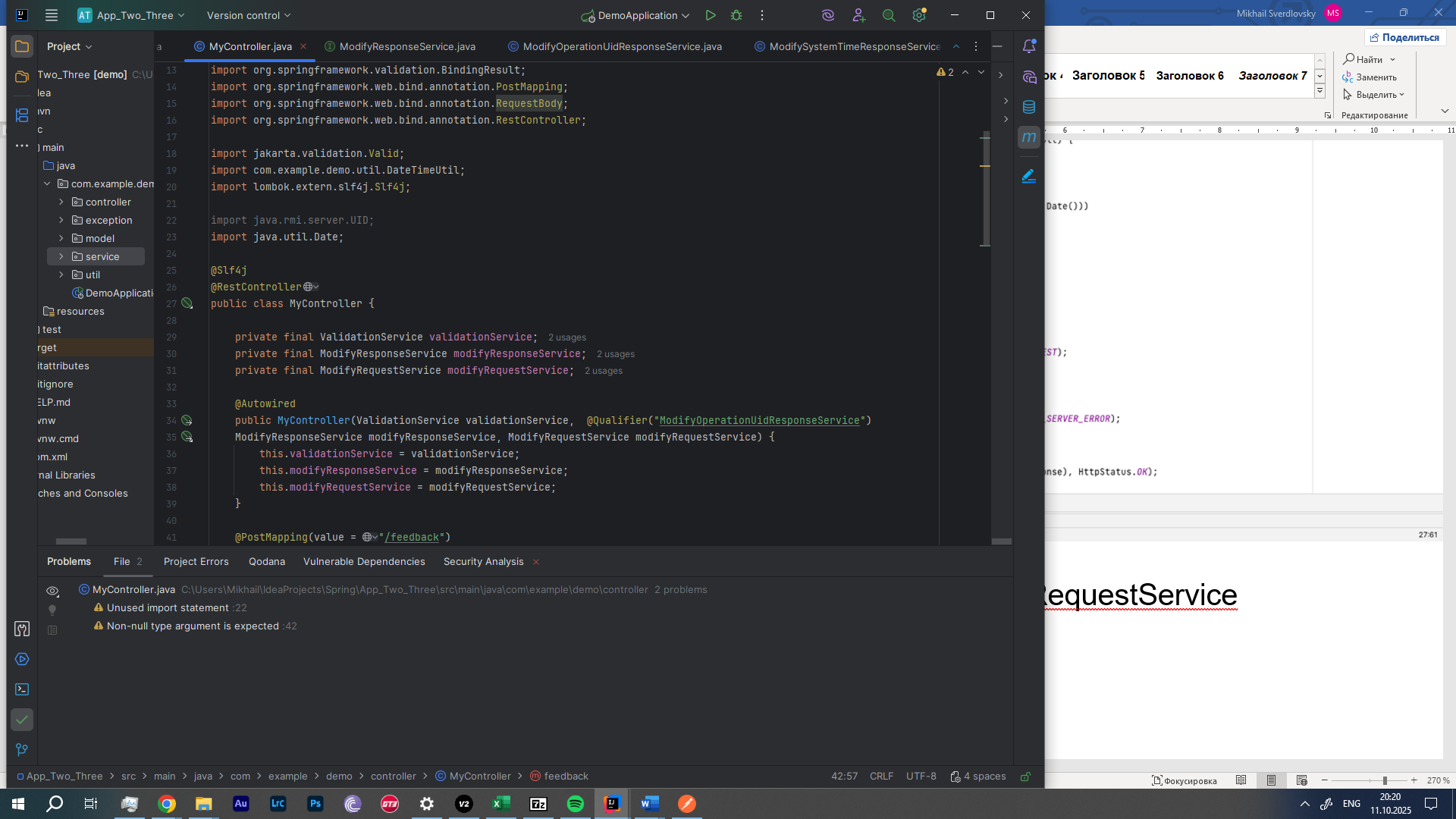


Рисунок 4 – Результат внедрения ModifyRequestService в MyController

Теперь мы должны запустить оба сервиса, и создать реквест таким образом, чтобы первый его принял, отправил второму, и принял уже модифицированным. Создаем реквест в Постмане и успешно отправляем его. Логги при обработке запроса показаны на рисунке 5.

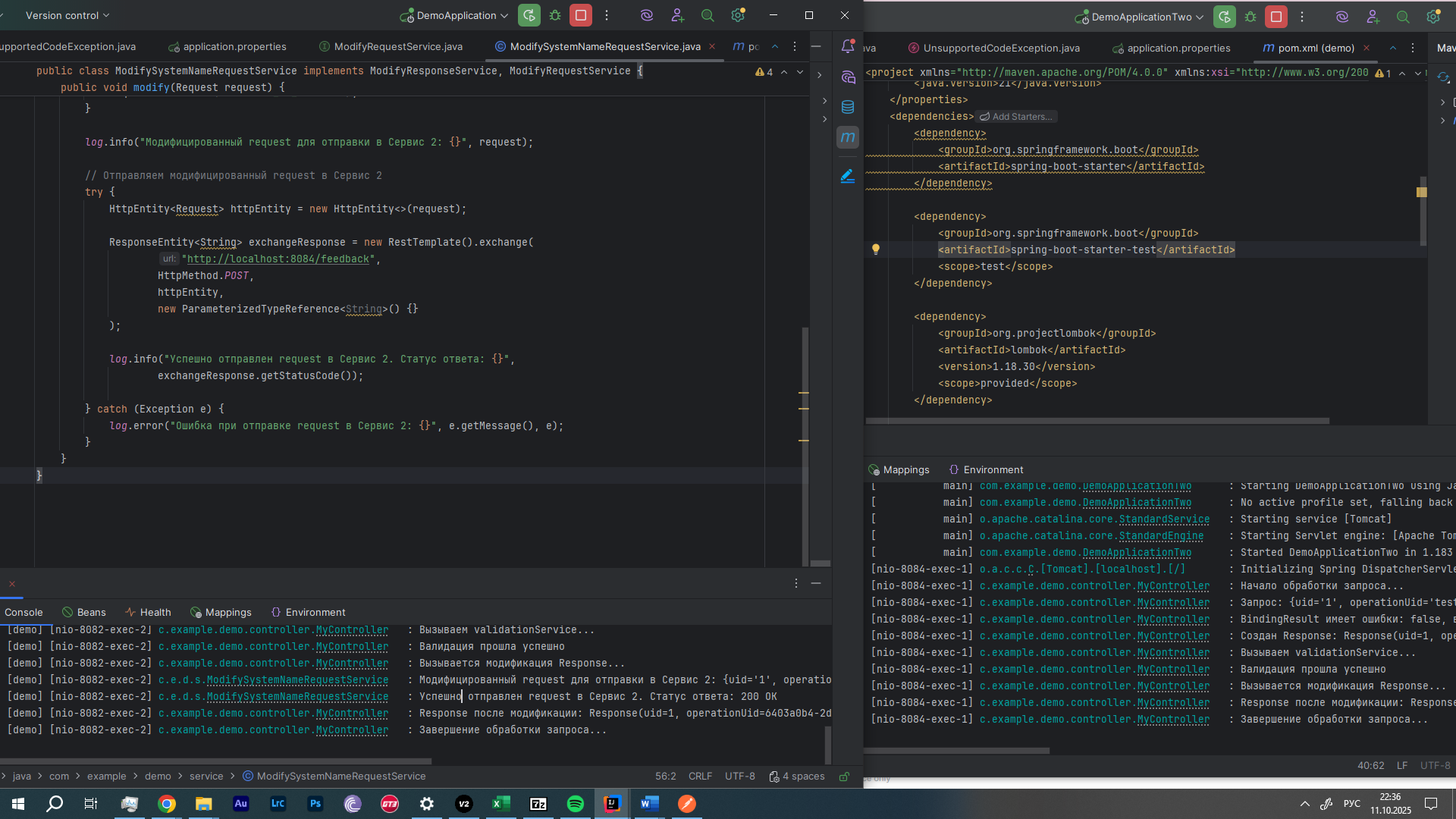


Рисунок 5 – Логги при обработке запроса

Переходим в Postman – смотрим, что запрос прошел успешно. Фиксируем ответ, он показан на рисунке 6.

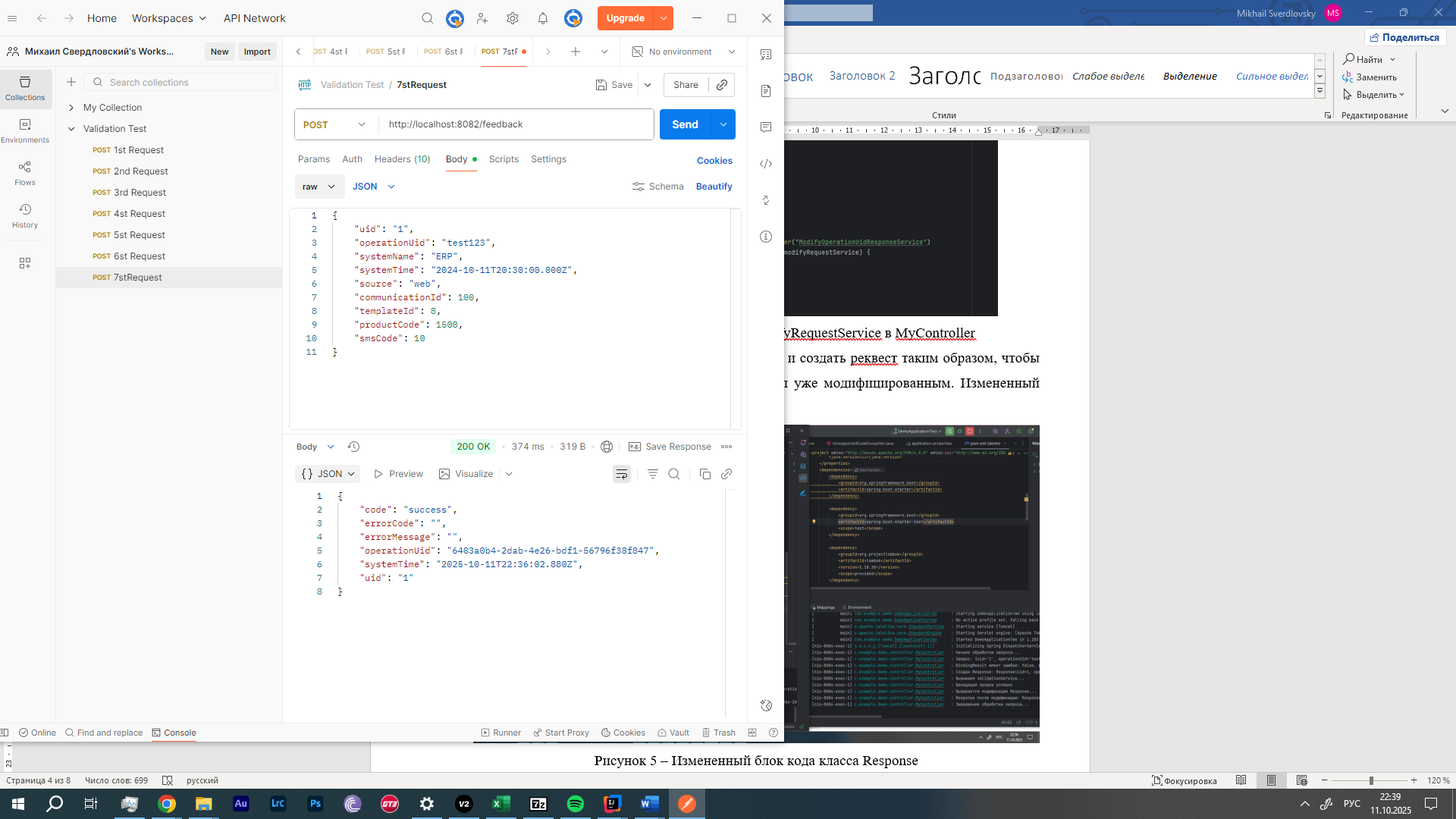


Рисунок 6 – Ответ на запрос через Postman

**Задание 2. Реализация дополнительного функционала.**

1 Добавьте реализацию интерфейса ModifyRequestService, которая изменяет поле source в Сервисе 1;

2 Доработайте Сервис 1 и Сервис 2 таким образом, чтобы в логи Сервиса 2 выводилось сообщение о разнице времени прошедшего с момента получения Сервисом 1 Request и временем получения модифицированного Request полученного Сервисом 2, другими словами измерить время от нажатия кнопки Send в Postman до вывода в лог Сервисом 2 сообщения.

Для начала, добавляем реализацию, изменяющую поле источника во время запроса в Сервисе 1. Для этого модифицируем класс ModifyRequestService, чтобы он изменял поле source. Задаем уникальный квалификатор ModifySourceRequestService и одноименный класс, в нем выводится в логи информация о преобразованиях; если поле источника пустое – в нем будет записано Source\_Modified\_By\_Service\_1; если поле не пустое – к значению (записанному в строку originalSource) записывается название источника и, затем, \_Modified\_By\_Service\_1.

Также модифицируем класс MyController, адаптируем его для работы с двумя сервисами – вносим строки с modifySourceRequestService.

Во время вызова сначала вызывается ResponseService, затем – modifySourceRequestService (изменяется только поле source), и после – modifySystemNameRequestService (замена systemName и отправка в Сервис 2). Формируем для проверки запрос в Postman. Конфигурация запроса показана на рисунке 7.

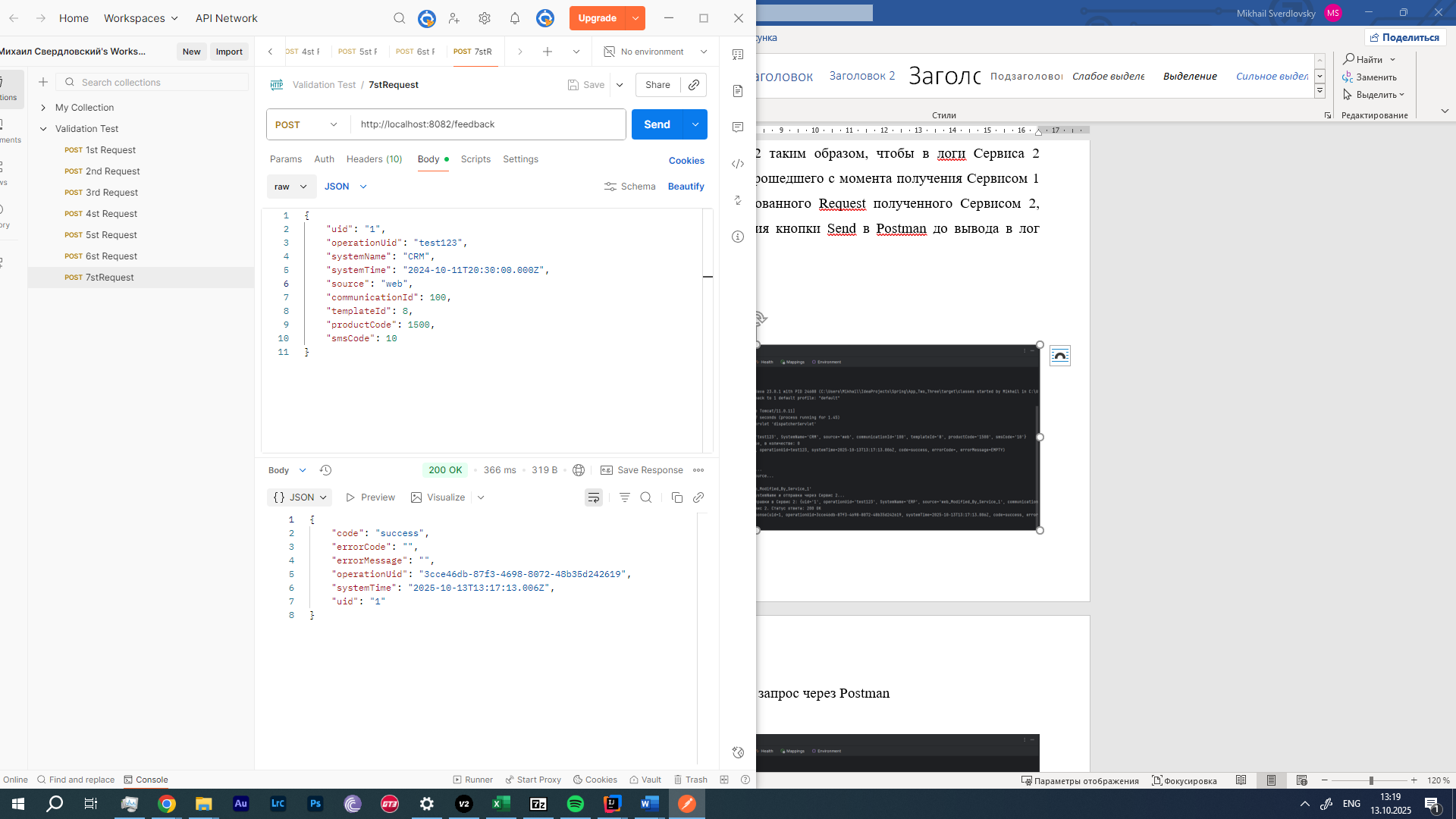


Рисунок 7 – Конфигурация запроса через Postman

В результате запроса, в логах отобразился процесс изменения поля source и systemName, запрос был успешно обработан в обоих сервисах. Логги при запросе в Postman показаны на рисунке 8.

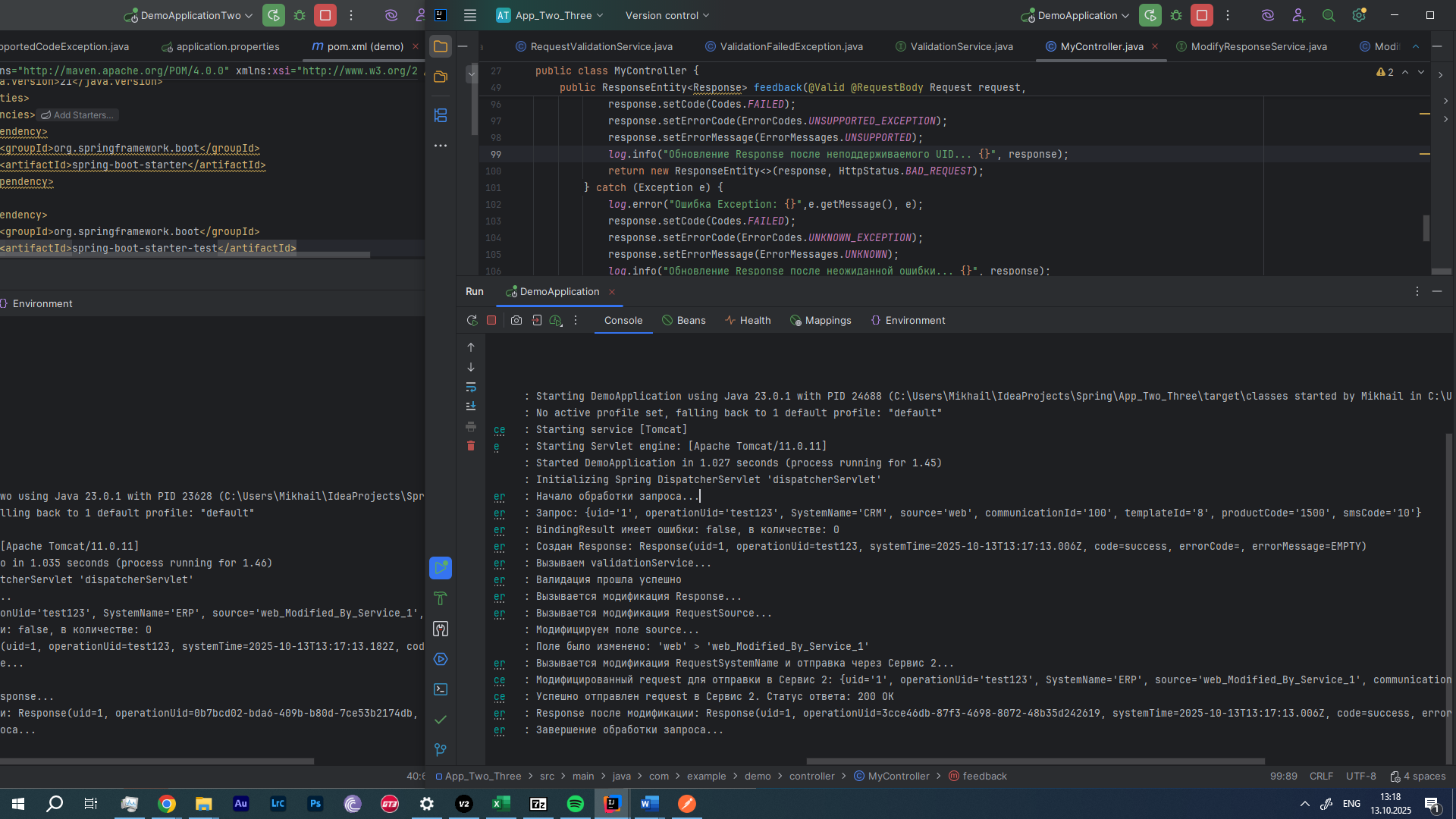


Рисунок 8 – Логги при запросе в Postman

Для реализации второй задачи, вводим новое поле в Сервисе 1 в классе Request.java для записи времени запроса, и добавляем его в шаблон для вывода в логгах при запросе, в MyController записываем время получения запроса. Для Request.java в Сервисе 2 делаем то же самое, что и для Сервиса 1. А в MyController, при условии, что getRequsetsTime не нулевой, рассчитываем время прохождения: для Сервиса 1 – getRequestTime (геттер времени запроса Request), для Сервиса 2 – System.currentTimeMillis() (текущее время). Затем выводим это время через log.info. Сформированный запрос и логги сервисов показаны на рисунке 9.

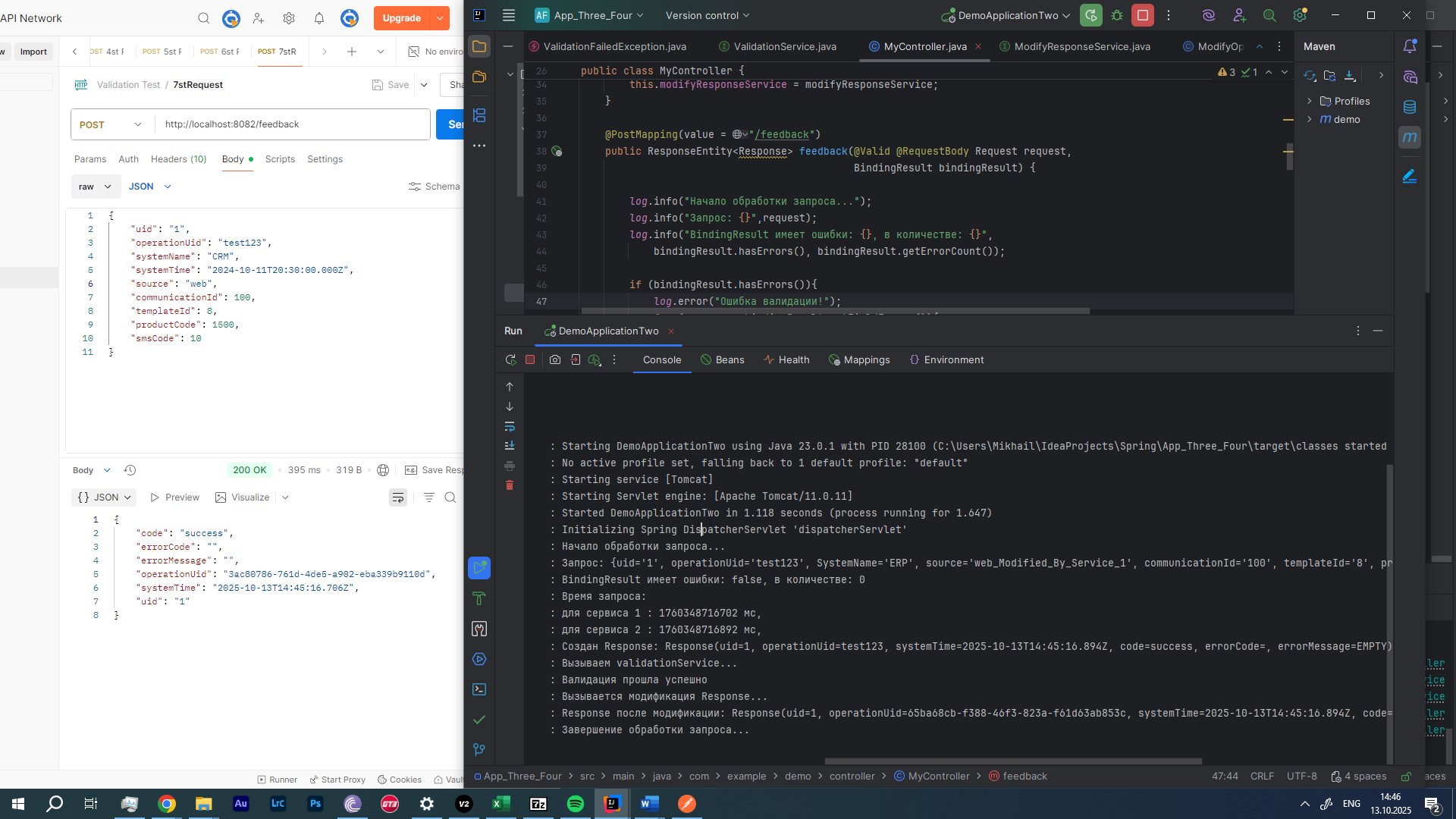


Рисунок 8 – Сформированный запрос и логги сервисов

**Задание 3. Ответы на вопросы.**

1 Зачем нужен параметр server.port?

Для указания порта, на котором будет работать SB сервер (по умолчанию 8080).

2 Зачем нужна аннотация @Builder?

Через Builder удобнее создавать объекты и шаблоны для них.

3 Зачем нужна аннотация @AllArgsConstructor?

Для автоматической генерации конструктора с полями класса сразу.

**Вывод**

В результате лабораторной работы был доработан простой REST сервис.