|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра инструментального и прикладного программного обеспечения (ИППО)**

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**

по дисциплине «Технология разработки программных приложений»

**Практическое задание № 3**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент группы ИНБО-01-17 | ИКБО-16-22 Трусов Е.С. | (подпись) | |
| Ассистент | Петрова А.А. | (подпись) | |
| Отчет представлен | «\_\_\_»\_\_\_\_\_\_\_\_2023г. | |  | |

Москва 2023 г.

Содержание

[ПОСТАНОВКА ЗАДАНИЯ 3](#_Toc163761069)

[Цель работы: 3](#_Toc163761070)

[Задание: 3](#_Toc163761071)

[Индивидуальный вариант №12: 3](#_Toc163761072)

[ХОД РАБОТЫ 4](#_Toc163761073)

[КОНТРОЛЬНЫЕ ВОПРОСЫ 12](#_Toc163761074)

[ВЫВОД 13](#_Toc163761075)

# ПОСТАНОВКА ЗАДАНИЯ

## Цель работы:

Знакомство с системой сборки Gradle. Возможности gradle. Управление зависимостями.

## Задание:

Для выполнения необходимо клонировать (или форкнуть) git-репозиторий согласно варианту, и выполнить следующие задания:

1. Найти отсутствующую зависимость и указать ее в соответствующем блоке в build.gradle, чтобы проект снова начал собираться

2. В некоторых классах поправить имя пакета

3. Собрать документацию проекта, найти в ней запросы состояния и сущности по идентификатору

4. Собрать jar со всеми зависимостями (так называемый UberJar), после чего запустить приложение. По умолчанию, сервер стартует на порту 8080.

5. Запросить состояние запущенного сервера (GET запрос по адресу http://localhost:8080)

6. Запросить сущность по идентификатору (GET запрос по адресу: <http://localhost:8080/сущность/идентификатор>) Идентификатором будут 3 последних цифры в серийном номере вашего студенческого билета.

7. В задаче shadowJar добавить к jar-файлу вашу фамилию

8. Выполнить задачу checkstyleMain. Посмотреть сгенерированный отчет. Устранить ошибки оформления кода.

## Индивидуальный вариант №12:

репозиторий: https://github.com/rtu-mirea/trpp-second-12, сущность ru.mirea.entity.Animal

# ход работы

Перед выполнением практической работы был клонирован репозиторий, указанный в варианте.

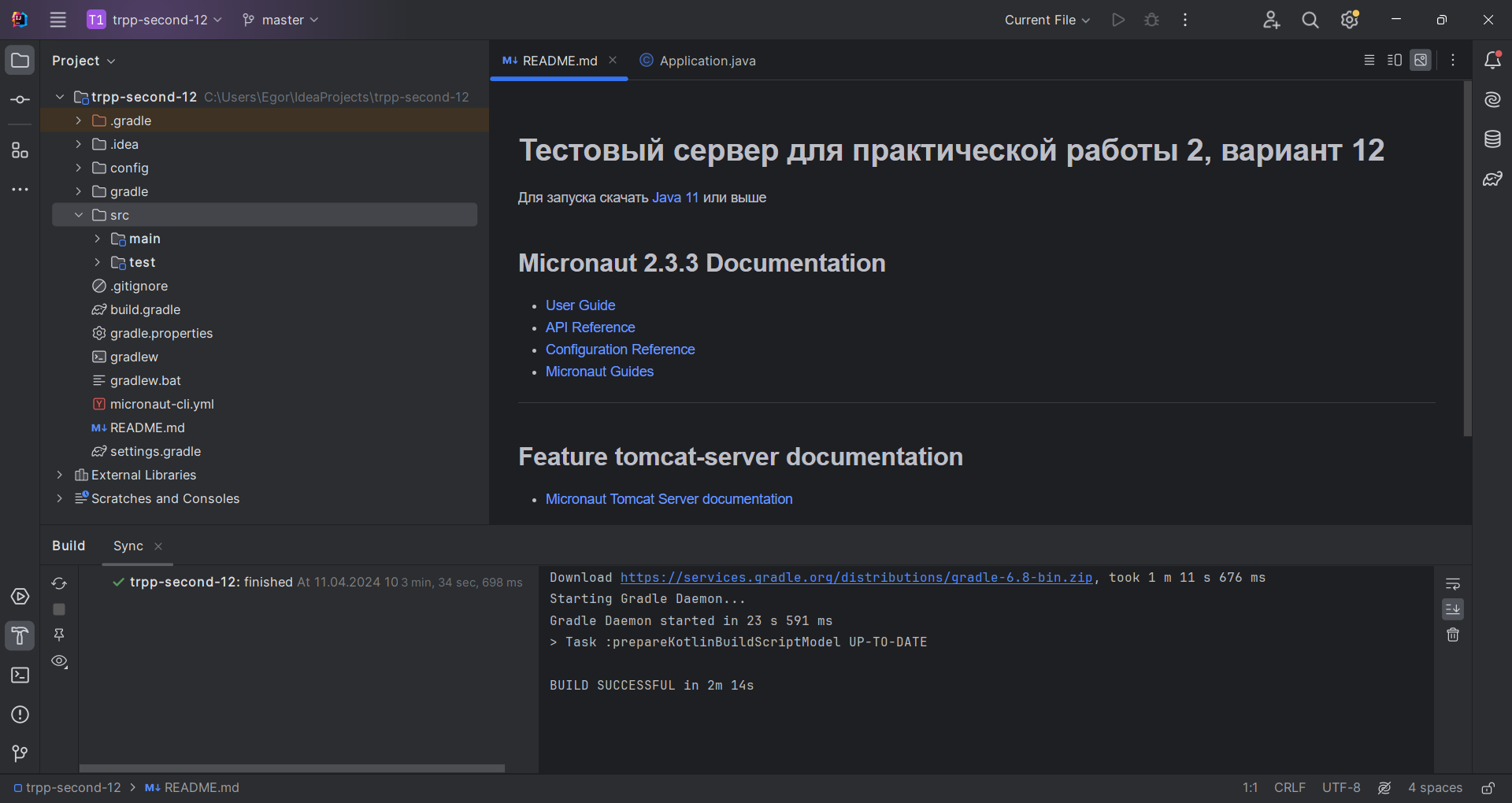


Рисунок 1 – Клонированный репозиторий

1. Найти отсутствующую зависимость и указать ее в соответствующем блоке в build.gradle, чтобы проект снова начал собираться:

Сперва был поправлен код проекта, импортированы необходимые зависимости.

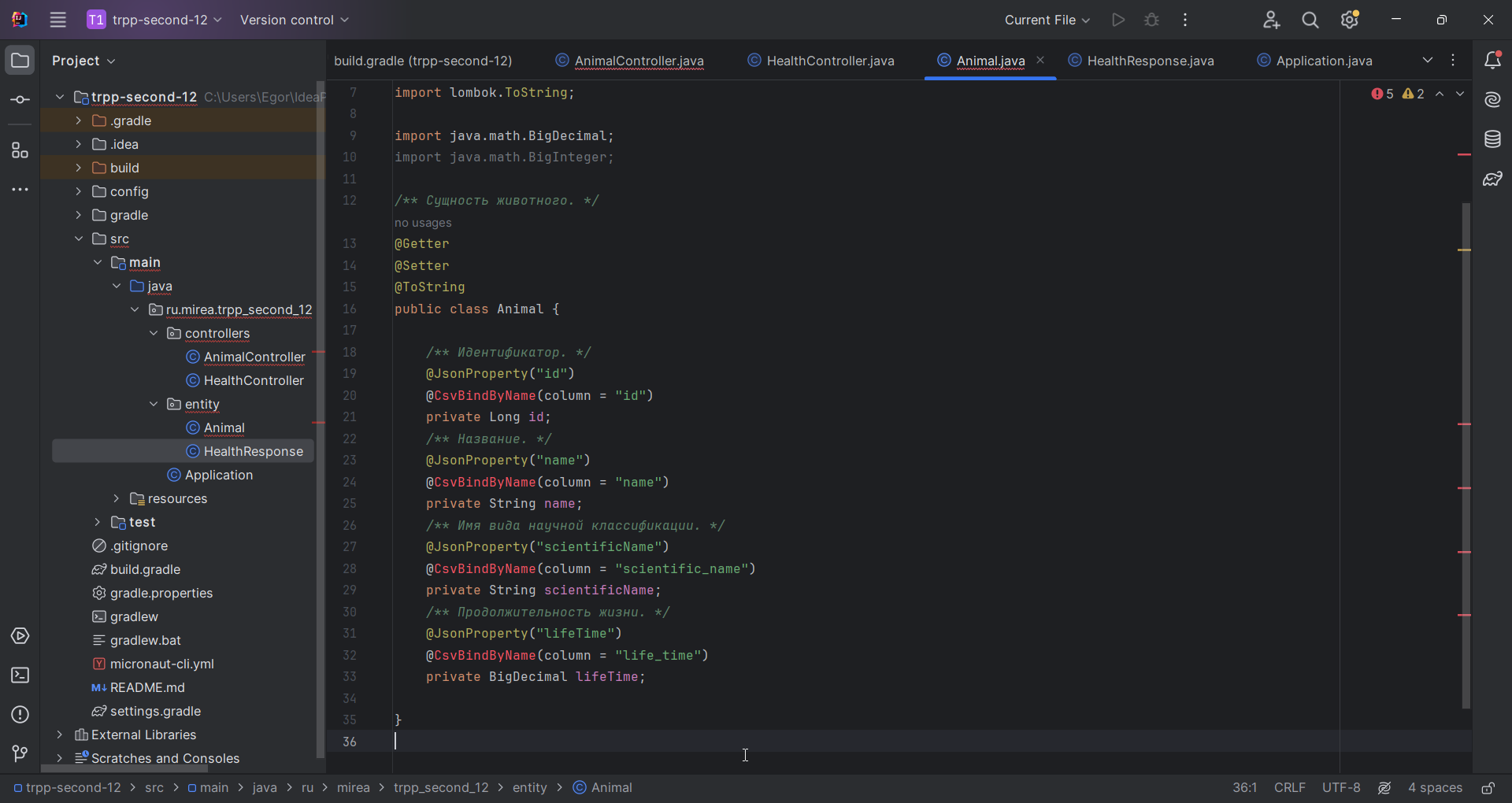


Рисунок 2 – Импорт BigDecimal

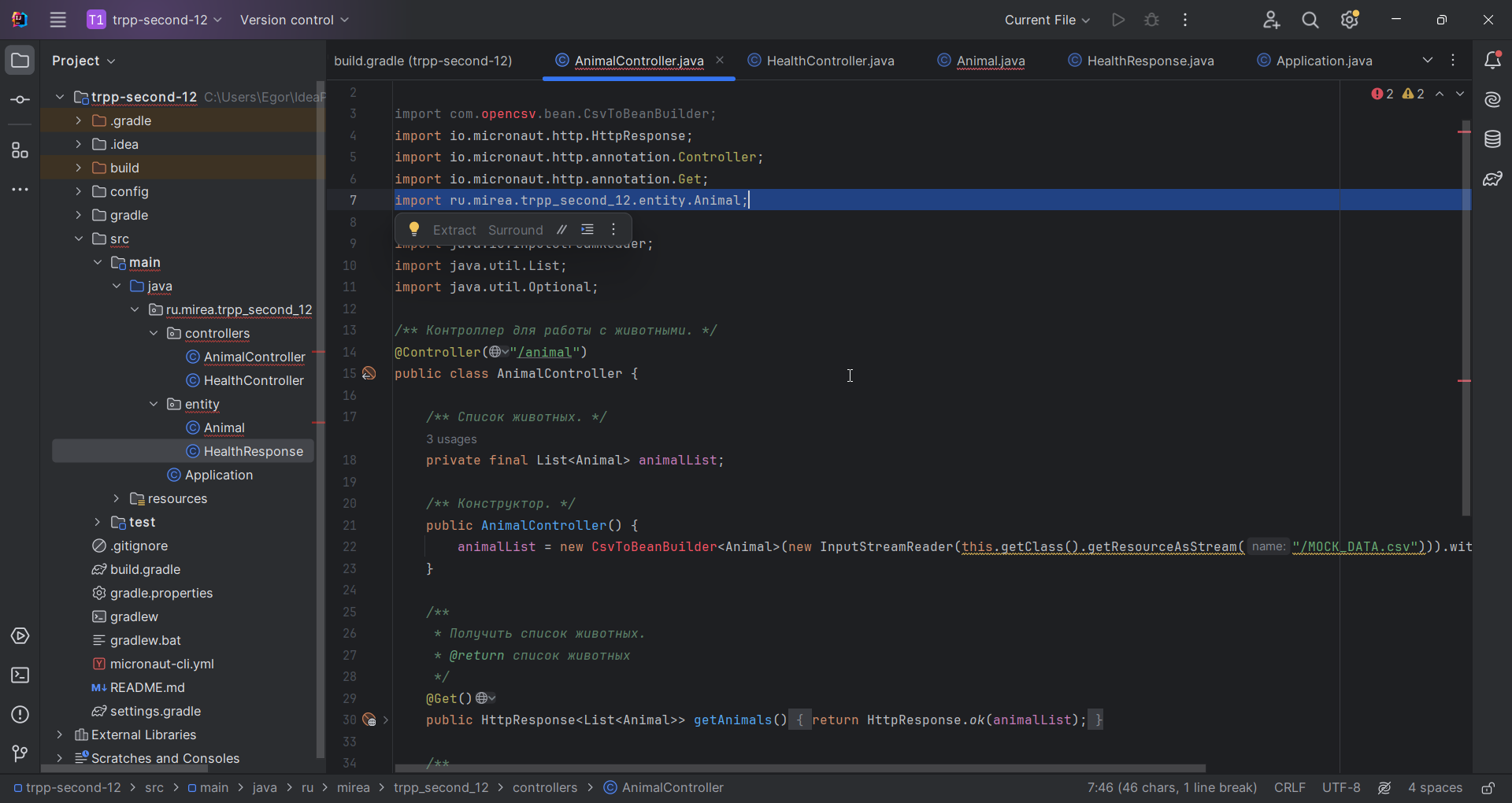


Рисунок 3 – импорт Animal

В ходе исправления ошибок было обнаружено, что одну из зависимостей среда не распознает автоматически при попытке импорта. Необходимо добавить эту зависимость в build.gradle:

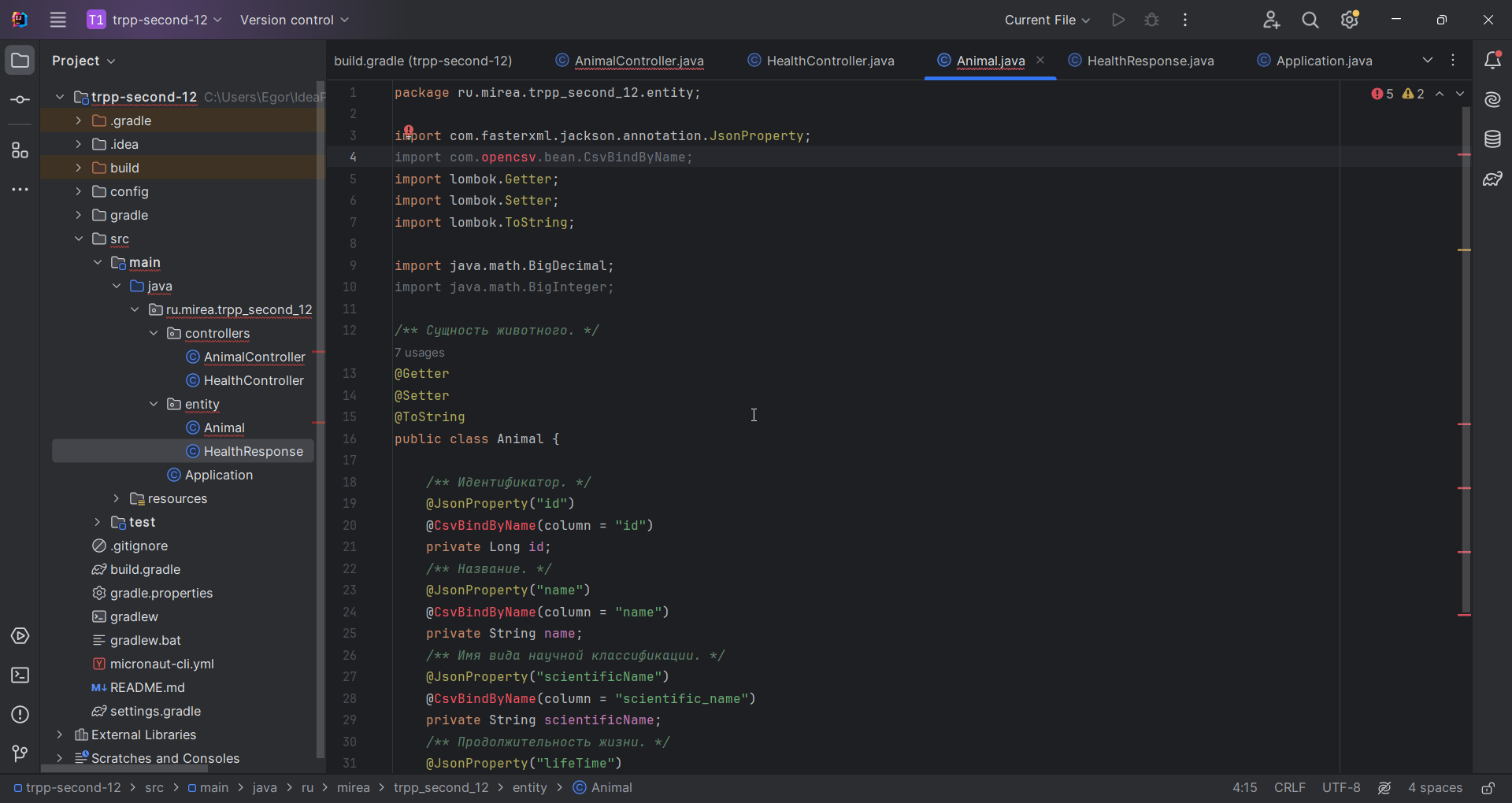


Рисунок 4 – Попытка импорта opencsv

В файле build.gradle была добавлена недостающая зависимость.

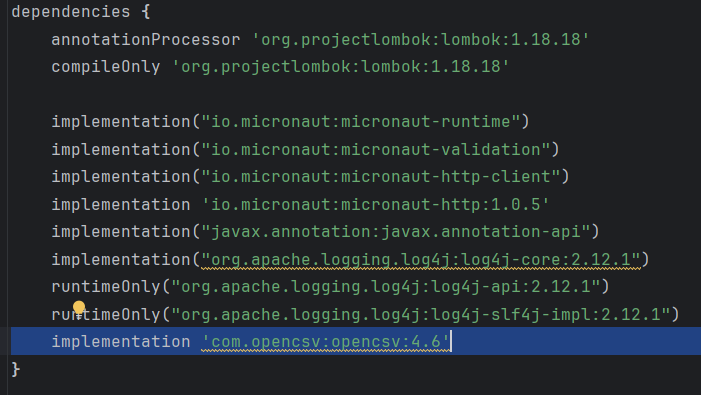


Рисунок 5 – Добавление необходимой зависимости

2. В некоторых классах поправить имя пакета:

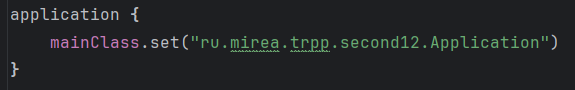


Рисунок 6 – Исправление пути к файлам

3. Собрать документацию проекта, найти в ней запросы состояния и сущности по идентификатору:

Для создания документации в терминале была выполнена команда ./gradlew javadoc. В папке build появилась папка docs с необходимой документацией

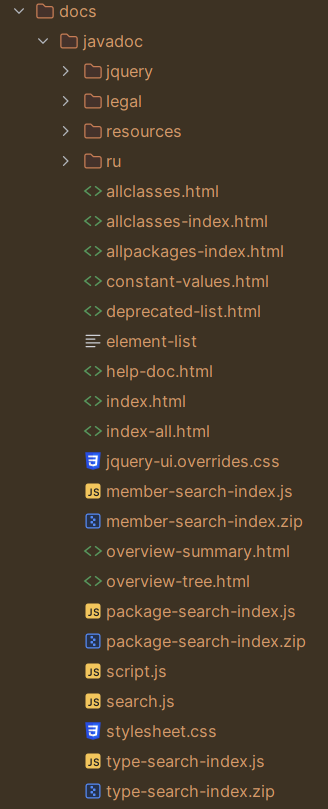


Рисунок 7 – Содержимое папки docs

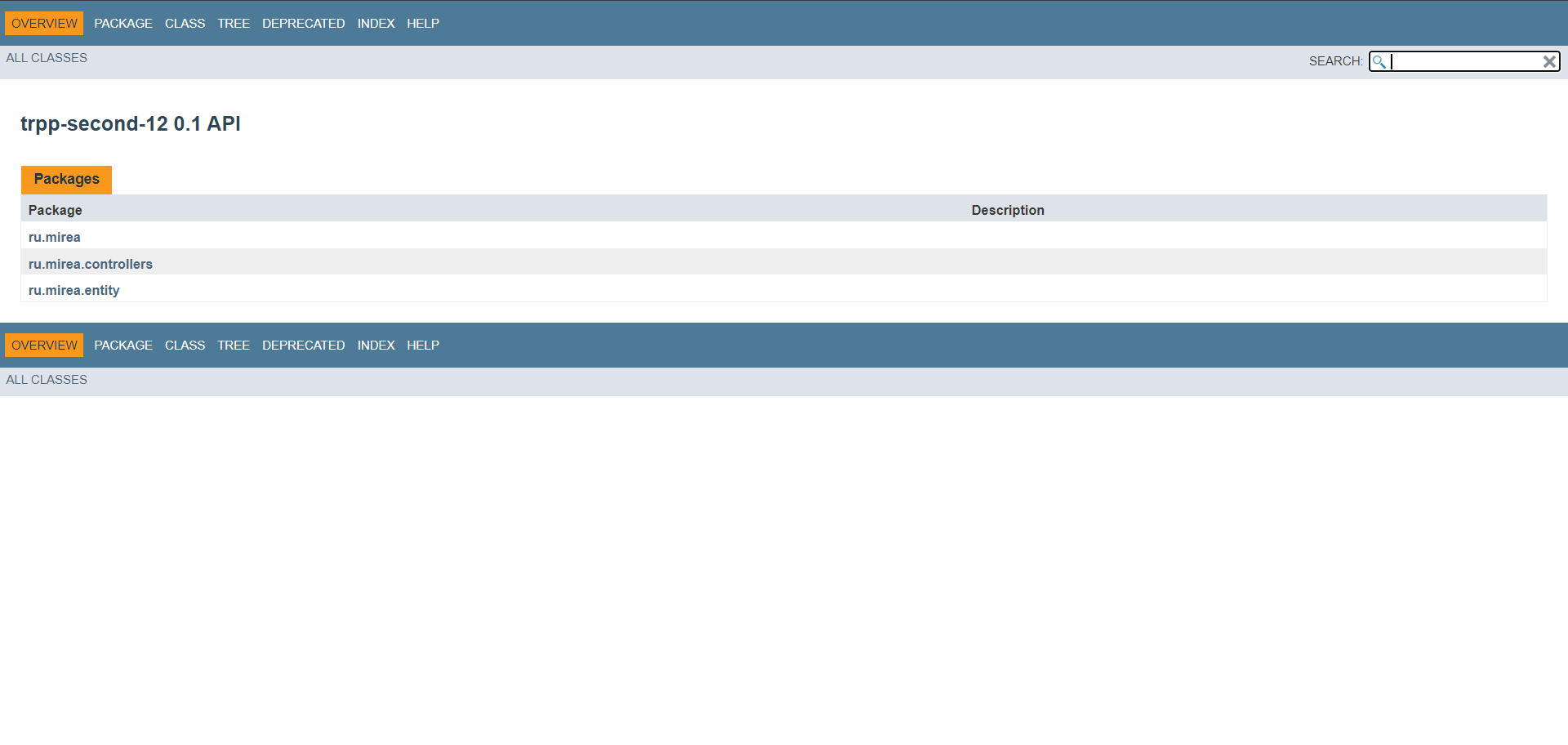


Рисунок 8 – Пример документации

4. Собрать jar со всеми зависимостями (так называемый UberJar), после чего запустить приложение. По умолчанию, сервер стартует на порту 8080:

После выполнения команды ./gradlew shadowJar был собран jar файл.

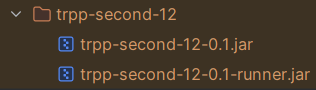


Рисунок 9 – Создание jar-файла

Затем проект был запущен и проверен на localhost

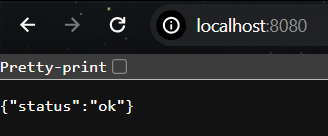


Рисунок 10 – Проверка работоспособности проекта

5. Запросить состояние запущенного сервера (GET запрос по адресу <http://localhost:8080>):

Для отправки GET-запроса был использован postman.

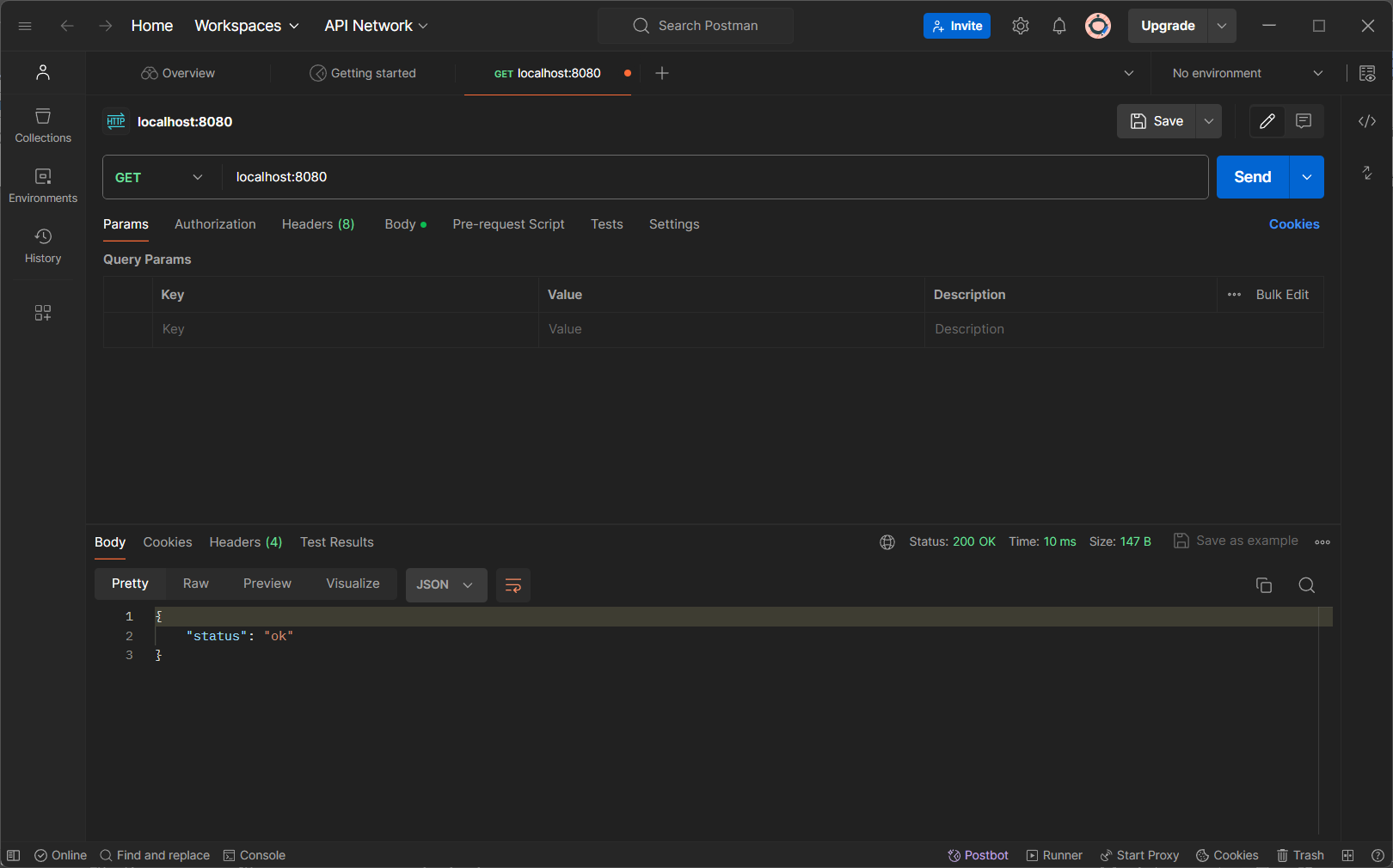


Рисунок 11 – Отправка GET-запроса

6. Запросить сущность по идентификатору (GET запрос по адресу: <http://localhost:8080/сущность/идентификатор>) Идентификатором будут 3 последних цифры в серийном номере вашего студенческого билета:

Для отправки GET-запроса был использован Postman.

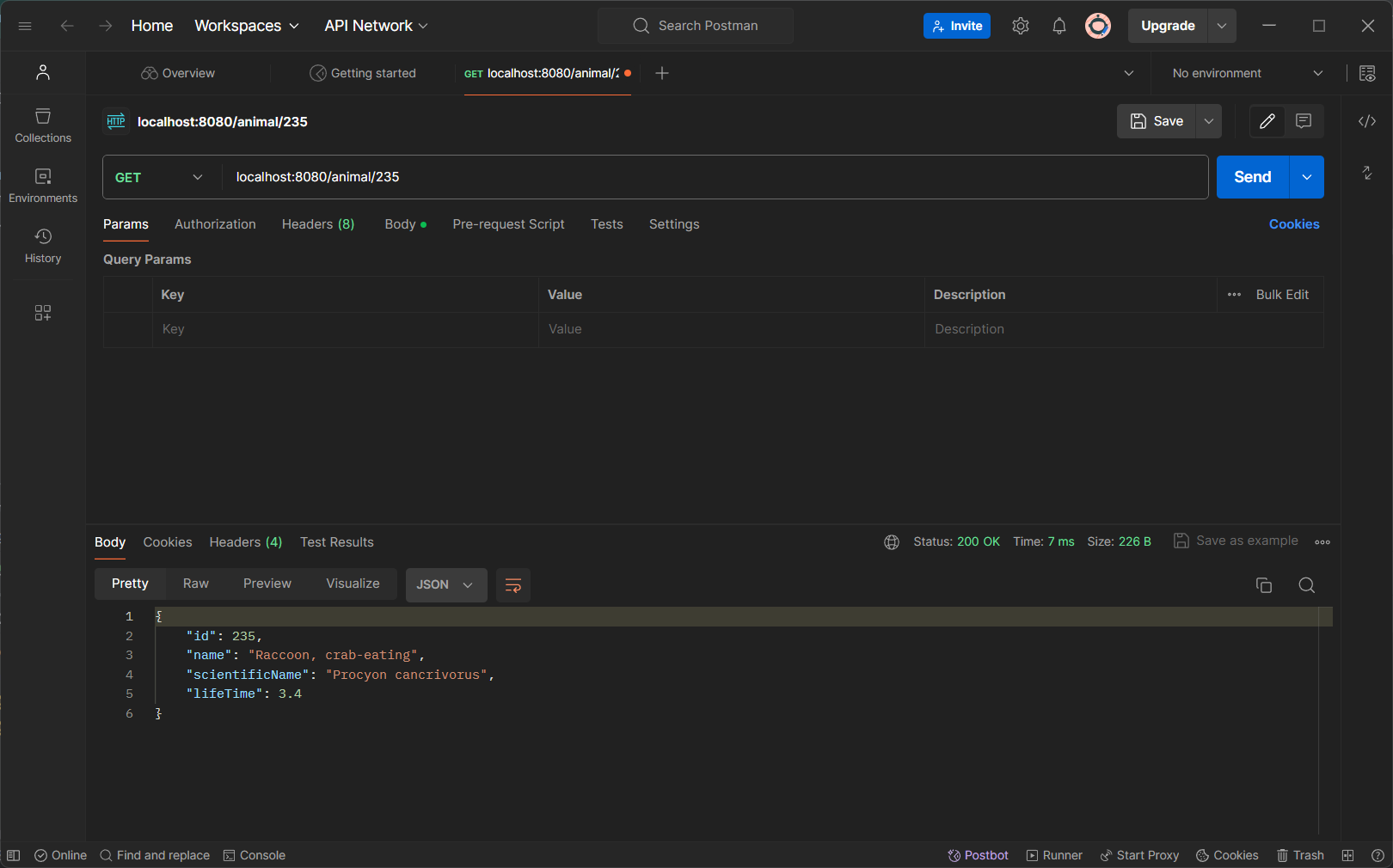


Рисунок 12 – Отправка GET-запроса для сущности

7. В задаче shadowJar добавить к jar-файлу вашу фамилию:

Были данные о себе. При создании jar-файла эти данные учитываются в названии.

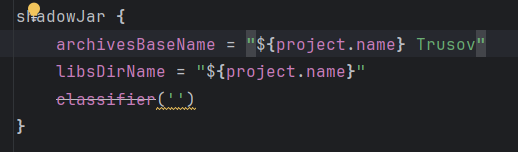


Рисунок 13 – Добавление фамилии в jar-файлу

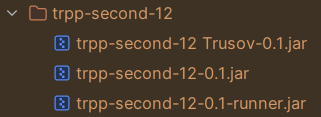


Рисунок 14 – Фамилия в jar-файле

8. Выполнить задачу checkstyleMain. Посмотреть сгенерированный отчет. Устранить ошибки оформления кода.

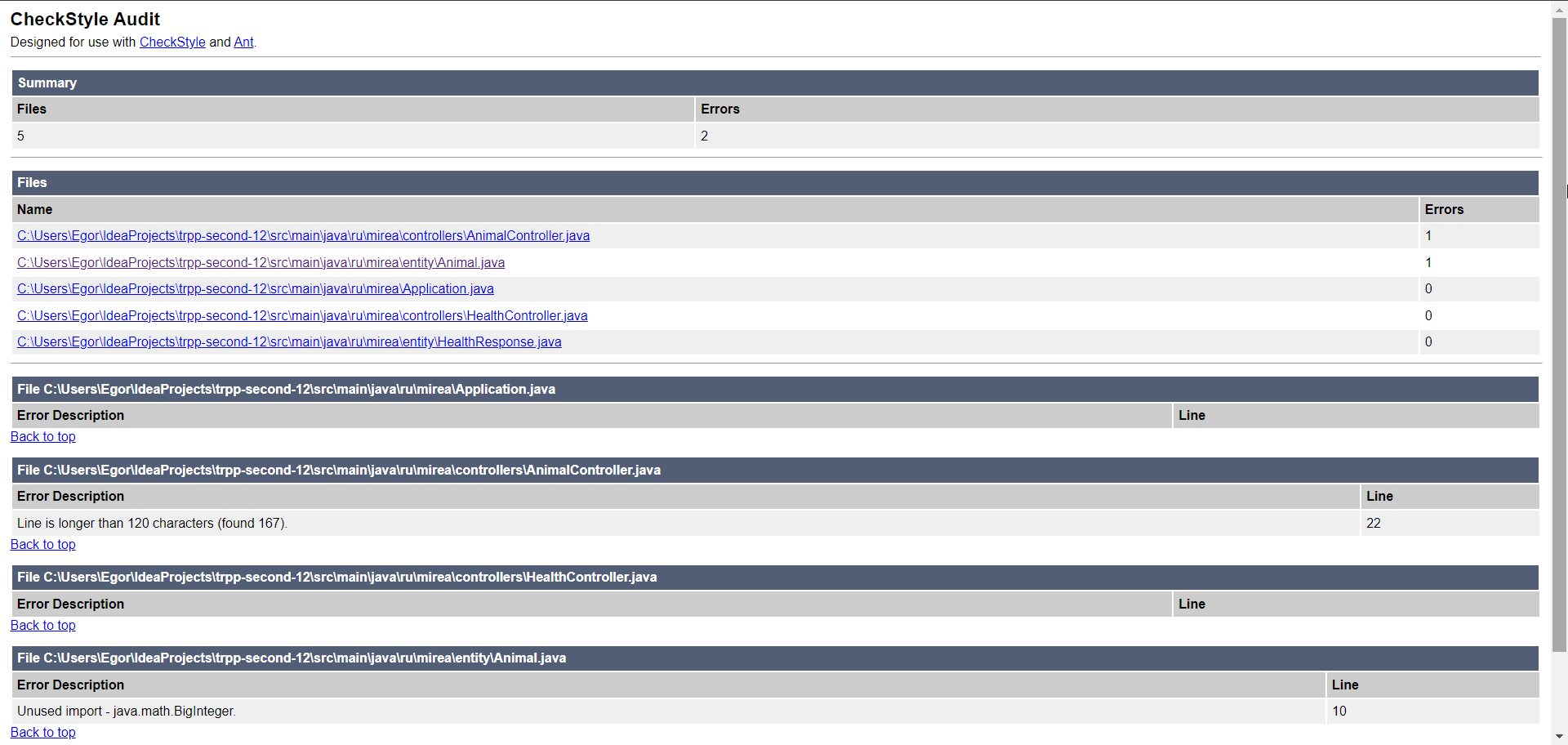


Рисунок 15 – Сгенерированный checkstyle отчёт

Для того, чтобы избежать ошибок с оформлением кода, была исправлена строка длиннее 120 символов.

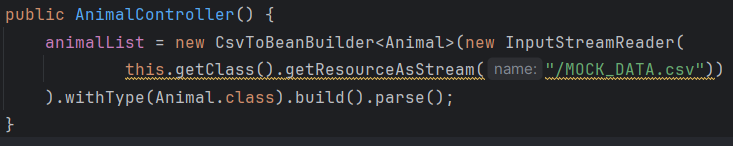


Рисунок 16 – Исправление слишком длинной строки

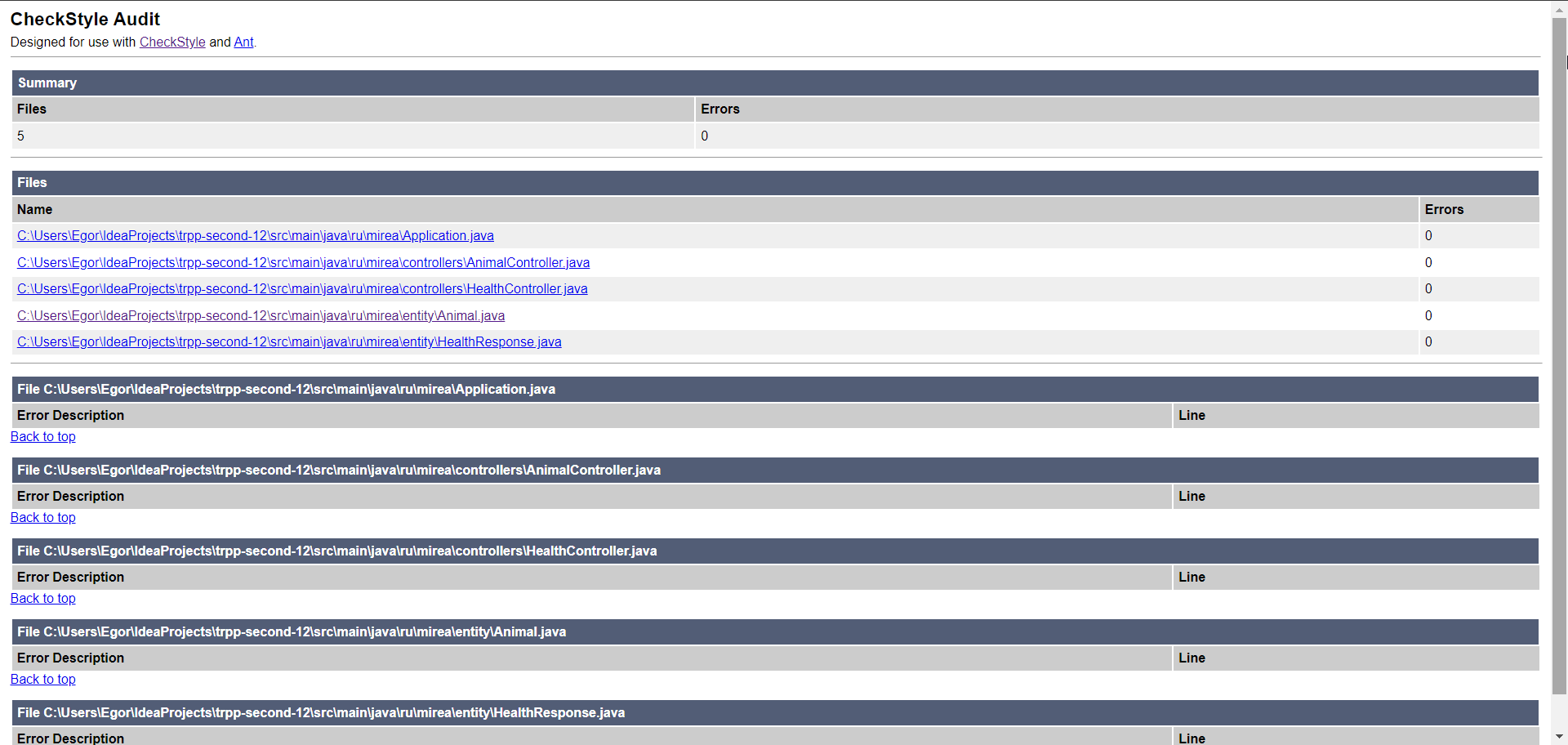


Рисунок 17 – Отчёт без ошибок

# Контрольные вопросы

1. Чем компиляция отличается от сборки?

Компиляция – это преобразование исходного кода в машинный код, а сборка – это объединение машинного кода и других ресурсов в исполняемый файл.

2. Что такое репозиторий?

Репозиторий – это хранилище для хранения и управления файлами проекта, включая исходный код, документацию, конфигурационные файлы и другие ресурсы.

3. Что такое gradle?

Gradle – это система сборки и инструмент автоматизации для разработки программного обеспечения, которая поддерживает множество языков программирования и платформ.

4. Что делает задача clean?

Задача clean – это стандартная задача для систем сборки, которая удаляет все промежуточные и конечные файлы, созданные во время процесса сборки проекта. Это позволяет очистить рабочую область проекта и начать сборку с чистого листа.

5. Что делает задача build?

Задача build – это стандартная задача для систем сборки, которая выполняет полный цикл сборки проекта, включая компиляцию кода, выполнение тестов, создание пакетов и другие операции, необходимые для получения готового к использованию продукта.

6. Что такое javadoc?

Javadoc – это инструмент для генерации документации по исходному коду программ на языке Java. Javadoc позволяет автоматически создавать HTML-документацию для классов, методов, полей и других элементов кода, основанную на специальных комментариях, добавленных в исходный код.

# ВЫВОД

В ходе проделанной работы мы познакомились с системой сборки Gradle, с её возможностями и с управлением зависимостями.