

ОГЛАВЛЕНИЕ

1 Постановка задачи (Вариант 6)	1
1.1 Цель работы	1
1.2 Задание	1
2 Проектирование и реализация	2
2.1 Первая сборка	2
2.2 Создание документации	3
2.3 Тесты отправки запросов	4
2.4 Дополнительно	6
3 Ответы на вопросы	8

1 Постановка задачи (Вариант 6)

1.1 Цель работы

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

1.2 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 (604).

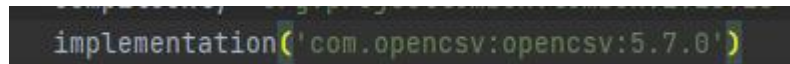
7. В задаче shadowJar добавить к jar-файлу вашу фамилию

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

2 Проектирование и реализация

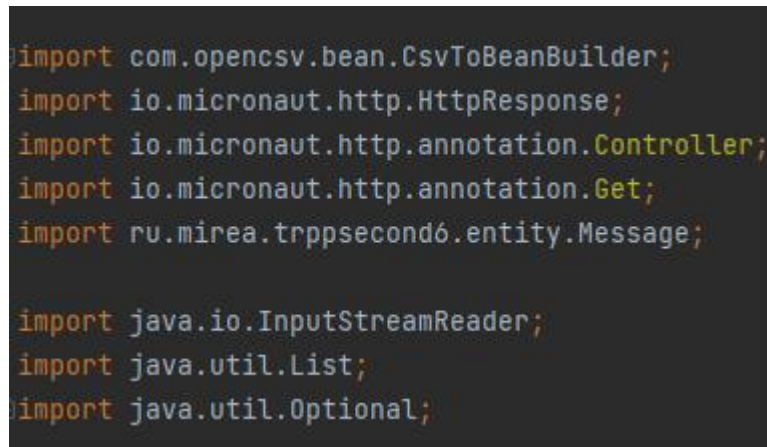
2.1 Первая сборка

Чтобы проект начал собираться, в зависимостях нужно было указать библиотеку для работы с .csv файлами (OpenCSV). Её подключение в gradle.build и импорт в необходимых файлах можно видеть на рисунке 1 и рисунке 2 соответственно.



```
implementation('com.opencsv:opencsv:5.7.0')
```

Рисунок 1 – Подключение OpenCSV в build.gradle



```
import com.opencsv.bean.CsvToBeanBuilder;
import io.micronaut.http.HttpResponse;
import io.micronaut.http.annotation.Controller;
import io.micronaut.http.annotation.Get;
import ru.mirea.trppsecond6.entity.Message;

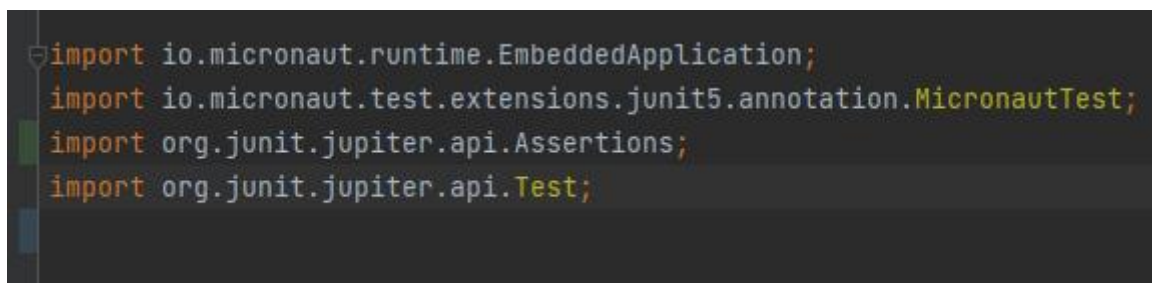
import java.io.InputStreamReader;
import java.util.List;
import java.util.Optional;
```

Рисунок 2 – Добавление нужных импортов в файл EmployeeController

Файл не собирался пока не были исправлены ошибки стиля

Название пакета с ru.mirea.trpp_second_6 было заменено на ru.mirea.trppsecond6.

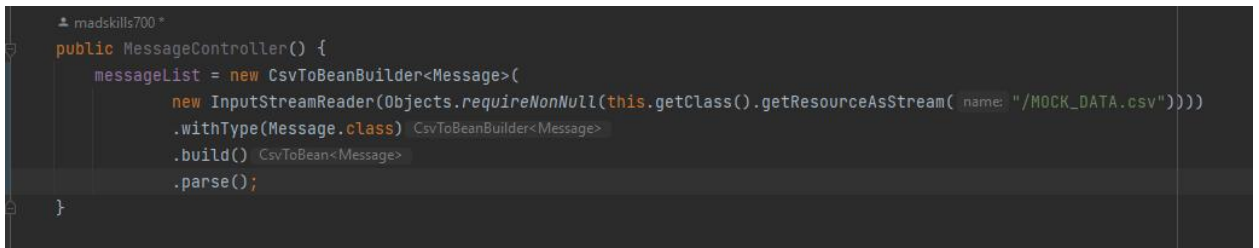
В файле тестов был заменен порядок импортов, результат на рисунке 3.



```
import io.micronaut.runtime.EmbeddedApplication;
import io.micronaut.test.extensions.junit5.annotation.MicronautTest;
import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.Test;
```

Рисунок 3 – Новый порядок импортов в файле тестов

Слишком длинная строка в файле EmployeeController была исправлена, результат на рисунке 4.

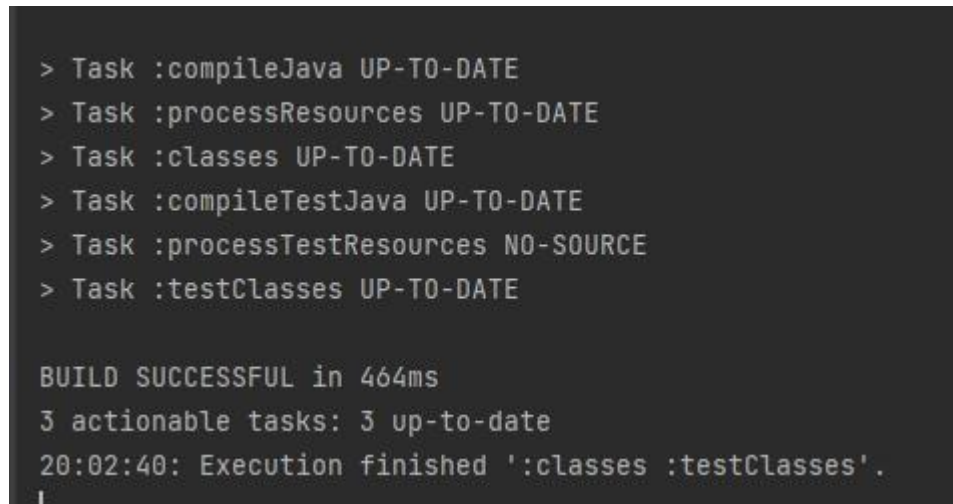


```

madskills700 *
public MessageController() {
    messageList = new CsvToBeanBuilder<Message>(
        new InputStreamReader(Objects.requireNonNull(this.getClass().getResourceAsStream("MOCK_DATA.csv"))))
        .withType(Message.class) CsvToBeanBuilder<Message>
        .build() CsvToBean<Message>
        .parse();
}

```

Рисунок 4 – Исправления в слишком длинной строке
Результат сборки проекта на рисунке 5.



```

> Task :compileJava UP-TO-DATE
> Task :processResources UP-TO-DATE
> Task :classes UP-TO-DATE
> Task :compileTestJava UP-TO-DATE
> Task :processTestResources NO-SOURCE
> Task :testClasses UP-TO-DATE

BUILD SUCCESSFUL in 464ms
3 actionable tasks: 3 up-to-date
20:02:40: Execution finished ':classes :testClasses'.

```

Рисунок 5 – Успешная сборка проекта

2.2 Создание документации

Для сборки документации используем задачу Javadoc, результаты сборки документации и найденные запросы на рисунках 6 – 8 соответственно.



```

PS C:\Users\sidor\Desktop\trpp-second-6> ./gradlew javadoc

BUILD SUCCESSFUL in 1s

```

Рисунок 6 – Сборка документации

Method Detail

getOrganizations

```
@Get public io.micronaut.http.HttpResponse<java.util.List<Organization>> getOrganizations()
```

Получить список организаций.

Returns:

список организаций

findById

```
@Get("/{id}") public io.micronaut.http.HttpResponse<Organization> findById(java.lang.Long id)
```

Найти организацию по идентификатору.

Parameters:

id - идентификатор организации

Returns:

Организация, если есть, иначе 404 ошибка

Рисунок 7 – Методы класса EmployeeController

Method Detail

healthCheck

```
@Get public io.micronaut.http.HttpResponse<HealthResponse> healthCheck()
```

Проверить состояние сервера.

Returns:

ответ 200 - ОК

Рисунок 8 – Метод класса HealthController

2.3 Тесты отправки запросов

Для создания jar файла используем задачу shadowJar, далее запустим собранный файл через java -jar путь к файлу с названием, рисунки 10, 11

```
PS C:\Users\sidor\Desktop\trpp-second-6> ./gradlew shadowJar

BUILD SUCCESSFUL in 704ms
3 actionable tasks: 3 up-to-date
```

Рисунок 10 – Сборка jar файла

```
PS C:\Users\sidor\Desktop\trpp-second-6> java -jar ./build/trpp-second-6/trpp-second-6-0-1.jar
```

Рисунок 11 – запуск сервера через jar файл

Запросы к серверу и к сущности изображены на рисунках 12 и 13. Делались с помощью Postman.

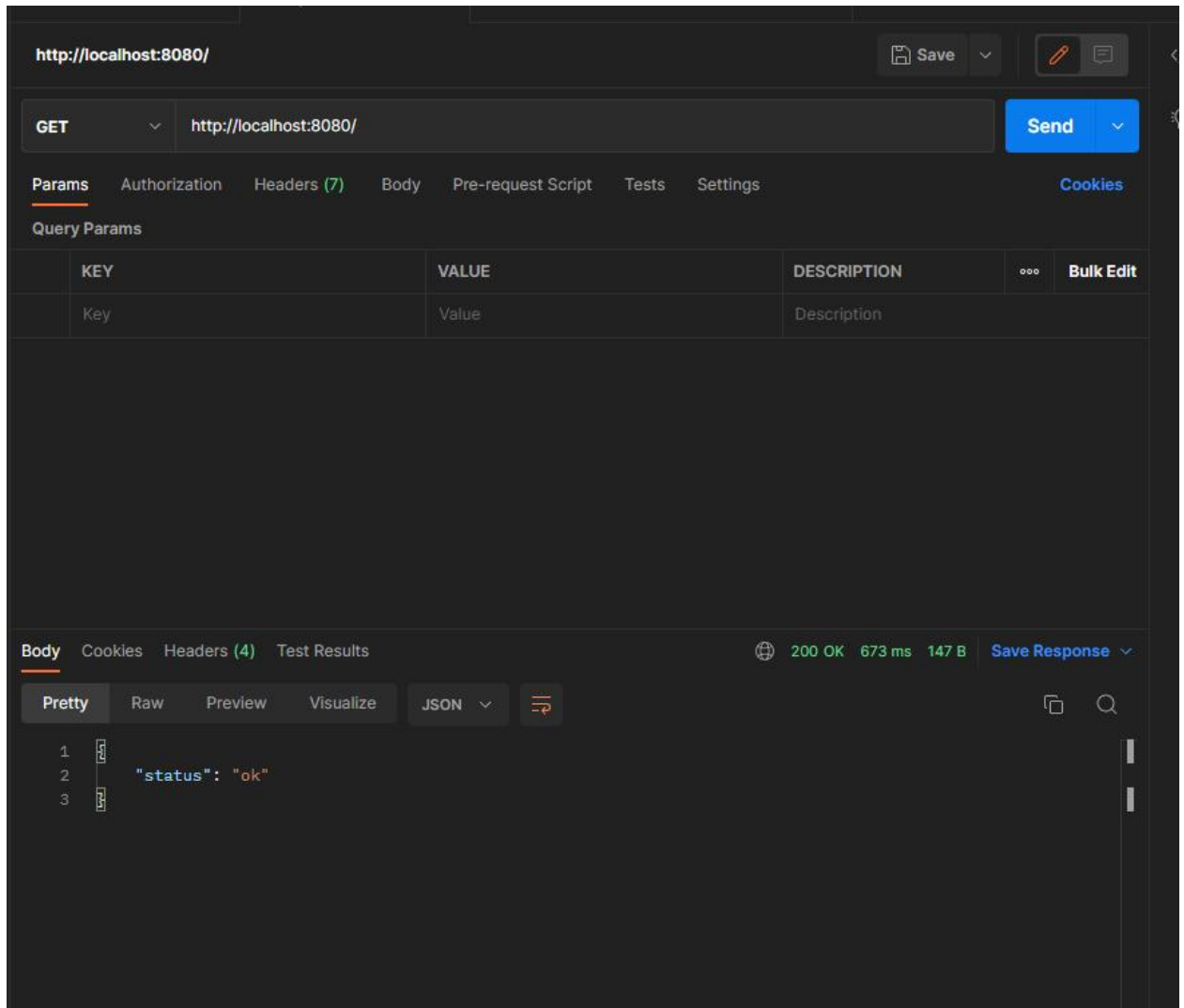


Рисунок 12 – Get запрос к серверу


```
4 actionable tasks: 1 executed, 3 up-to-date
PS C:\Users\sidor\Desktop\trpp-second-6> ./gradlew checkstyleTest

BUILD SUCCESSFUL in 724ms
4 actionable tasks: 4 up-to-date
```

Рисунок 15 – Проверка команды checkstyleTest

3 Ответы на вопросы

1. Чем компиляция отличается от сборки?

Компиляция - это процесс преобразования исходного кода программы на языке высокого уровня в машинный код, который может быть понят и исполнен процессором компьютера.

Сборка - это процесс объединения скомпилированных файлов (или других ресурсов, таких как изображения и шрифты) в единый исполняемый файл или библиотеку. Сборка выполняется после компиляции всех файлов с исходным кодом и может включать этапы оптимизации кода, линковки и упаковки исполняемого файла. Сборка может выполняться несколько раз в течение процесса разработки, чтобы включить изменения в исходный код и обновить исполняемый файл.

5. Что такое gradle?

Gradle - это система автоматической сборки проектов, которая используется в различных областях разработки программного обеспечения, включая Android-приложения, серверные приложения и веб-приложения. Она является современной системой автоматической сборки, которая позволяет создавать сложные проекты с множеством зависимостей и модулей.

6. Что такое Maven?

Maven - это инструмент автоматической сборки проектов, который используется в различных областях разработки программного обеспечения, включая Java-приложения и серверные приложения. Он является популярным инструментом для управления зависимостями, сборки проектов и автоматизации процесса разработки.

7. Что такое mavencentral?

Maven Central - это крупнейший репозиторий для Java-библиотек и плагинов, который используется в системе автоматической сборки Maven. Этот репозиторий хранит множество библиотек и плагинов, которые могут быть использованы в проектах Java.

8. Что делает задача clean?

Задача "clean" в системе автоматической сборки, такой как Maven или Gradle, выполняет очистку сгенерированных ранее файлов, чтобы проект мог быть пересобран с чистого листа.

18. Что такое Postman?

Postman - это инструмент для тестирования API и создания запросов к веб-сервисам.