



МИНОБРНАУКИ РОССИИ

*Федеральное государственное бюджетное образовательное учреждение  
высшего образования*

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

---

Институт Информационных технологий (ИТ)

Кафедра Математического обеспечения и стандартизации информационных  
технологий (МОСИТ)

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 3  
по дисциплине  
«Технология разработки программных приложений»**

**Тема: «Gradle»**

Выполнил студент группы: ИКБО-50-23

Павлов Н.С.

Принял

Степанов П.В.

Практическая работа выполнена

«\_\_» 2025г.

(подпись студента)

«Зачтено»

«\_\_» 2025 г.

(подпись руководителя)

Москва 2025

## **СОДЕРЖАНИЕ**

1 ПОСТАНОВКА ЗАДАЧИ .....	3
2 ВЫПОЛНЕНИЕ РАБОТЫ .....	4
3 ВЫВОДЫ .....	10
4 ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ .....	11

## **1 ПОСТАНОВКА ЗАДАЧИ**

**Цель работы:** Знакомство с системой сборки Gradle. Возможности gradle.

Управление зависимостями.

**Персональный вариант:** репозиторий: <https://github.com/rtu-mirea/trpp-second-9>, сущность ru.mirea.entity.Nomenclature

### **Задания для выполнения:**

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.

## 2 ВЫПОЛНЕНИЕ РАБОТЫ

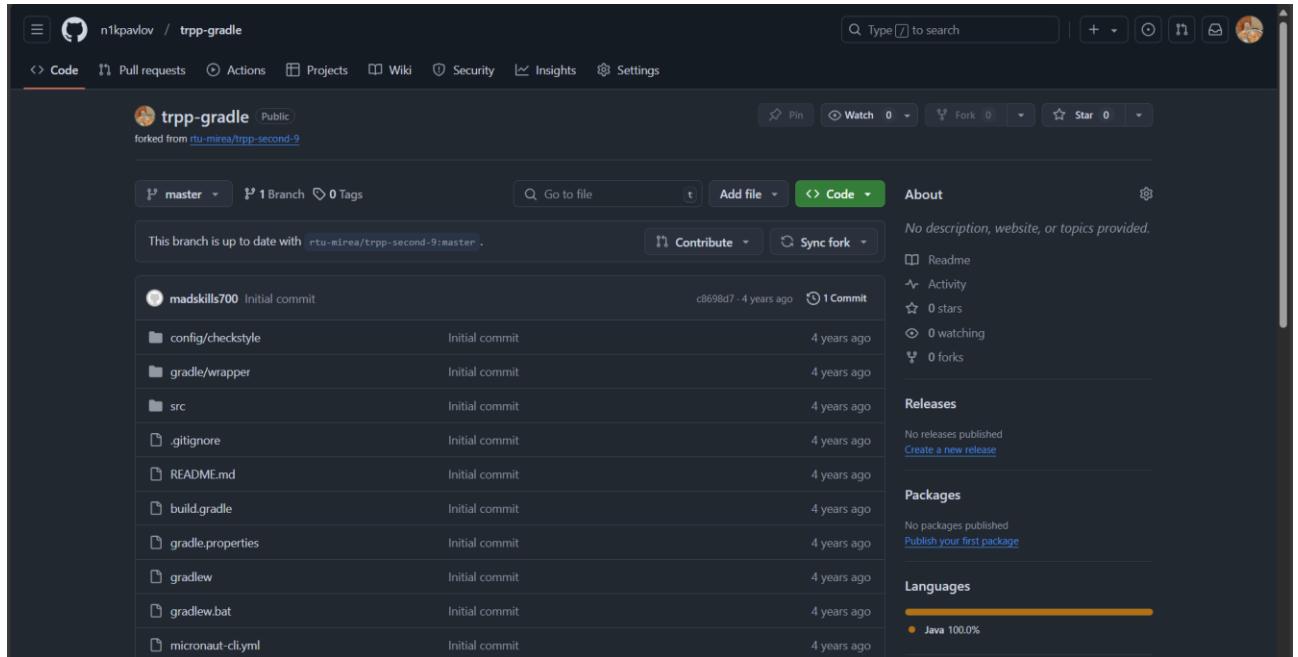


Рисунок 1 – Fork репозитория варианта

```
22
23 > dependencies {
24     annotationProcessor 'org.projectlombok:lombok:1.18.18'
25     compileOnly 'org.projectlombok:lombok:1.18.18'
26
27     implementation("io.micronaut:micronaut-runtime")
28     implementation("io.micronaut:micronaut-validation")
29     implementation("io.micronaut:micronaut-http-client")
30     implementation("javax.annotation:javax.annotation-api")
31     implementation("com.opencsv:opencsv:5.7.1") //new
32     implementation('org.apache.logging.log4j:log4j-core:2.17.1')
33     runtimeOnly("org.apache.logging.log4j:log4j-api:2.12.1")
34     runtimeOnly("org.apache.logging.log4j:log4j-slf4j-impl:2.12.1")
35 }
```

Рисунок 2 – Обновленный список зависимостей в build.gradle

```
© Nomenclature.java ×

1 package ru.mirea.trpp_second_9.entity;
2
3 import com.fasterxml.jackson.annotation.JsonProperty;
4 import com.opencsv.bean.CsvBindByName;
5 import lombok.Getter;
6 import lombok.Setter;
7 import lombok.ToString;
8 import java.math.BigDecimal;//new
```

Рисунок 3 – Импорт BigDecimal в Nomenclature.java

```
© HealthController.java ×

1 package ru.mirea.trpp_second_9.controllers;
2
3 import io.micronaut.http.HttpResponse;
4 import io.micronaut.http.annotation.Controller;
5 import io.micronaut.http.annotation.Get;
6 import ru.mirea.trpp_second_9.entity.HealthResponse;//new
7
```

Рисунок 4 – Импорт класса HealthResponse в HealthController.java

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the project structure under "trpp-second-9". It includes a "main" directory containing "java", "resources", and "test" sub-directories. "java" contains "ru.mirea.trpp\_second\_9" package with "controllers" and "entity" sub-packages, and "Application" class. "resources" contains "application.yml", "log4j2.xml", and "MOCK\_DATA.csv". "test" contains "java" and "ru.mirea" packages with "TrppTest" class.
- Code Editor:** Displays the "Application.java" file with the following code:

```
package ru.mirea.trpp_second_9;
import io.micronaut.runtime.Micronaut;
/** Класс - точка входа в приложение. */
public class Application {
    /**
     * Точка входа.
     * @param args аргументы
     */
    public static void main(String[] args) {
        Micronaut.run(Application.class, args);
    }
}
```
- Gradle Tool Window:** Shows the "trpp-second-9" build configuration with the "Tasks" section expanded. The "javadoc" task is selected.
- Terminal:** Shows the output of the "trpp-gradle [javadoc]" command:

```
> Task :compileJava UP-TO-DATE
> Task :processResources UP-TO-DATE
> Task :classes UP-TO-DATE
> Task :javadoc

BUILD SUCCESSFUL in 2s
3 actionable tasks: 1 executed, 2 up-to-date
13:30:19: Execution finished 'javadoc'.
```
- Status Bar:** Shows system information including battery level (-1°C), network connection (ENG), and system time (13:31 26.02.2025).

Рисунок 5 – Сборка документации проекта

## Method Detail

### getNomenclatures

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

Получить список номенклатур.

**Returns:**

список номенклатур

### findById

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

Найти номенклатуру по идентификатору.

**Parameters:**

id - идентификатор номенклатуры

**Returns:**

Номенклатура, если есть, иначе 404 ошибка

Рисунок 6 – Запросы состояния и сущности по идентификатору

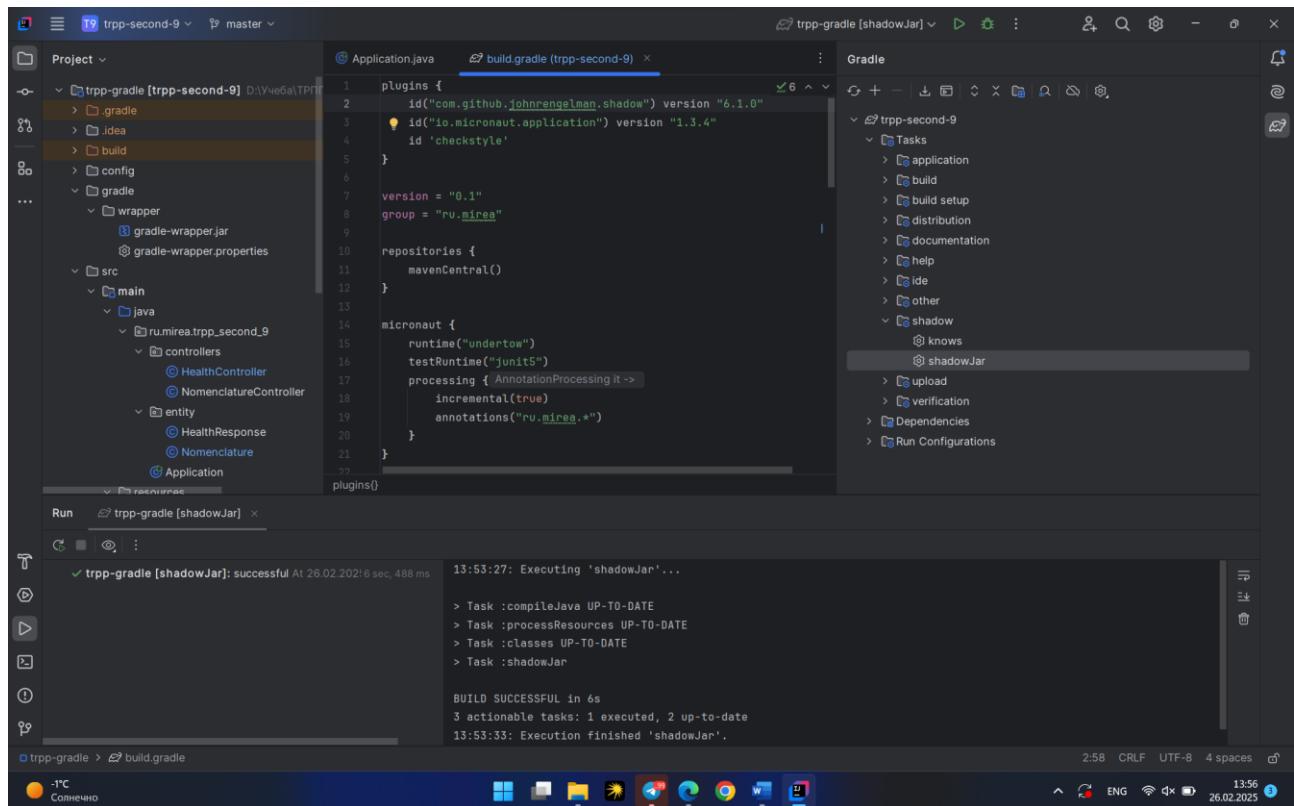


Рисунок 7 – Сборка jar со всеми зависимостями

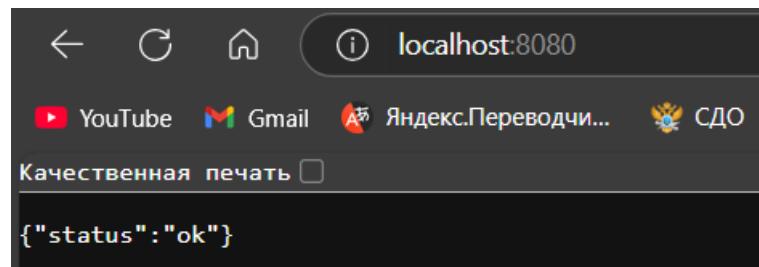


Рисунок 8 – GET запрос по адресу <http://localhost:8080>

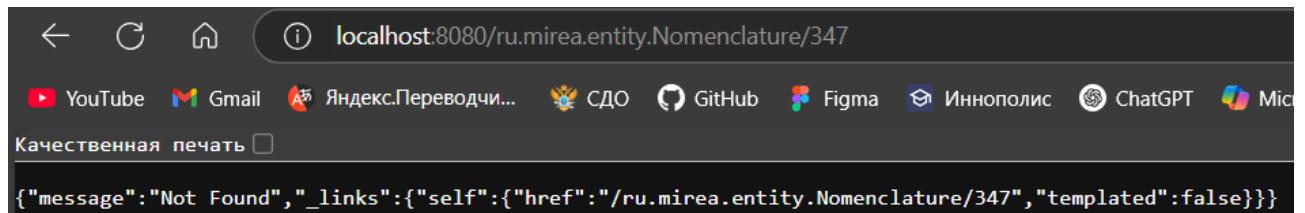


Рисунок 9 – GET запрос по адресу:  
<localhost:8080/ru.mirea.entity.Nomenclature/347>

A screenshot of a code editor showing the 'build.gradle' file. The code defines a 'shadowJar' task with the following configuration:

```
63 }
64
65 > shadowJar {
66     archivesBaseName = "${project.name} - Pavlov"
67     libsDirName = "${project.name}"
68     classifier('')
69 }
```

The line '65 >' has a green arrow icon to its left, indicating it is a comment or part of the code structure.

Рисунок 10 – Добавление в задаче shadowJar фамилии

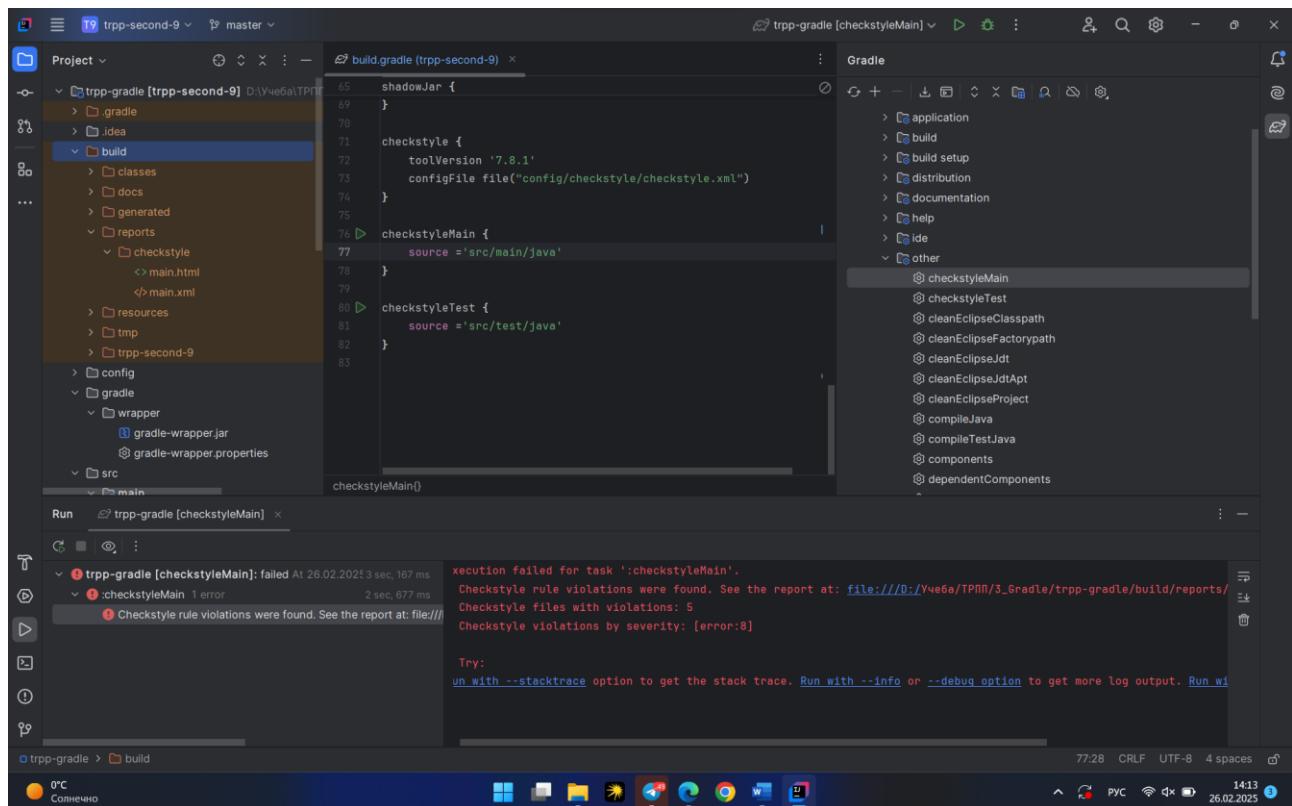


Рисунок 11 – Выполнение задачи checkstyleMain

**File D:\Учеба\ТРПП\3\_Gradle\trpp-gradle\src\main\java\ru\mirea\trpp\_second\_9\Application.java**

Error Description	Line
Name 'ru.mirea.trpp_second_9' must match pattern '^([a-z]+([a-z][a-z0-9]{1,}))*\$'.	1

[Back to top](#)

**File D:\Учеба\ТРПП\3\_Gradle\trpp-gradle\src\main\java\ru\mirea\trpp\_second\_9\controllers\HealthController.java**

Error Description	Line
Name 'ru.mirea.trpp_second_9.controllers' must match pattern '^([a-z]+([a-z][a-z0-9]{1,}))*\$'.	1
'; is not followed by whitespace.	6

[Back to top](#)

**File D:\Учеба\ТРПП\3\_Gradle\trpp-gradle\src\main\java\ru\mirea\trpp\_second\_9\controllers\NomenclatureController.java**

Error Description	Line
Name 'ru.mirea.trpp_second_9.controllers' must match pattern '^([a-z]+([a-z][a-z0-9]{1,}))*\$'.	1
Line is longer than 120 characters (found 185).	22

[Back to top](#)

**File D:\Учеба\ТРПП\3\_Gradle\trpp-gradle\src\main\java\ru\mirea\trpp\_second\_9\entity\HealthResponse.java**

Error Description	Line
Name 'ru.mirea.trpp_second_9.entity' must match pattern '^([a-z]+([a-z][a-z0-9]{1,}))*\$'.	1

[Back to top](#)

**File D:\Учеба\ТРПП\3\_Gradle\trpp-gradle\src\main\java\ru\mirea\trpp\_second\_9\entity\Nomenclature.java**

Error Description	Line
Name 'ru.mirea.trpp_second_9.entity' must match pattern '^([a-z]+([a-z][a-z0-9]{1,}))*\$'.	1
'; is not followed by whitespace.	8

[Back to top](#)

Рисунок 12 – Отчет об ошибках оформления кода

Files	Errors
D:\Учеба\ТРПП\3_Gradle\trpp-gradle\src\main\java\ru\mirea\trppsecond9\Application.java	0
D:\Учеба\ТРПП\3_Gradle\trpp-gradle\src\main\java\ru\mirea\trppsecond9\controllers\HealthController.java	0
D:\Учеба\ТРПП\3_Gradle\trpp-gradle\src\main\java\ru\mirea\trppsecond9\controllers\NomenclatureController.java	0
D:\Учеба\ТРПП\3_Gradle\trpp-gradle\src\main\java\ru\mirea\trppsecond9\entity\HealthResponse.java	0
D:\Учеба\ТРПП\3_Gradle\trpp-gradle\src\main\java\ru\mirea\trppsecond9\entity\Nomenclature.java	0

Рисунок 13 – Отчет после исправления ошибок

### **3 ВЫВОДЫ**

В ходе выполнения практической работы были получены навыки работы с системой сборки Gradle. Изучены функционал и возможности Gradle, исправлены ошибки в исходной сборке, сгенерирована и обработана документация по проекту.

## **4 ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ**

### **1. Как указать зависимости проекта?**

Зависимости проекта указываются в файле build.gradle (или build.gradle.kts для Kotlin DSL) в секции dependencies. В этом файле вы описываете библиотеки и другие модули, от которых зависит ваш проект.

### **2. Что такое Gradle?**

Gradle — это гибкий и высокопроизводительный инструмент автоматизации сборки, используемый для автоматизации процессов сборки программного обеспечения. Он написан на Groovy (или Kotlin) и использует концепцию "build scripts" (скрипты сборки), которые описывают весь процесс сборки проекта.

### **3. Что такое Maven?**

Maven — это инструмент управления проектами и сборки программного обеспечения, основанный на концепции Project Object Model (POM). Он предоставляет стандартизованный способ управления зависимостями, сборки, тестирования и развертывания проектов Java (и не только). Maven использует XML для описания проекта и его зависимостей.

### **4. Что такое Javadoc?**

Javadoc — это инструмент для автоматической генерации документации по коду Java из комментариев в исходном коде. Комментарии, написанные в специальном формате (документационные комментарии), извлекаются и преобразуются в HTML-документацию. Javadoc облегчает создание понятной и хорошо структурированной документации для вашего кода.

### **5. Что такое Checkstyle?**

Checkstyle — это инструмент статического анализа кода, который проверяет соответствие кода Java заданному стилю кодирования. Он помогает поддерживать единообразие стиля в проекте, что улучшает читаемость и поддерживаемость кода.

### **6. Что такое UberJar? При помощи какой задачи его собрать?**

UberJar (или "fat JAR") — это JAR-файл, который содержит не только код вашего приложения, но и все необходимые зависимости. Это удобно для развертывания, так как вам не нужно вручную управлять зависимостями на целевой машине. Для сборки UberJar в Gradle обычно используется плагин shadow.