

**СОДЕРЖАНИЕ**

[1. ПОСТАНОВКА ЗАДАЧИ 3](#_Toc163598274)

[2. ОСНОВНАЯ ЧАСТЬ 4](#_Toc163598275)

[3. КОНТРОЛЬНЫЕ ВОПРОСЫ 8](#_Toc163598276)

[4. ВЫВОДЫ 9](#_Toc163598277)

# **1. ПОСТАНОВКА ЗАДАЧИ**

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>)

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

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

Номер студента Албахтин И.В. по списку в группе равен N=3. Максимальный номер задания в таблице (число вариантов) M=15. Тогда номер задания V студента Албахтин И.В. будет равен остатку от деления V=Ост((N-1)/M)+1=Ост((3-1)/15)+1= Ост(2/15)+1= 2+1=3

|  |  |
| --- | --- |
| **Вариант** | **Задание** |
| 3 | репозиторий: https://github.com/rtu-mirea/trpp-second-3, сущность ru.mirea.entity.Client |

# **2. ОСНОВНАЯ ЧАСТЬ**

Для начала работы склонируем файлы по ссылке в локальный репозиторий

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

**Рисунок 2.1 — Клонирование репозитория на локальную машину**

Как видно на рисунке 2.2, одна из библиотек не загружена

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Рисунок 2.2 — Найдена отсутствующая зависимость по импортам**

Включим отсутствующую библиотеку в build.grandle



**Рисунок 2.3 — Добавление зависимости в build.gradle**

Также заметим отсутствующий импорт – необходимо добавить импорт класса entity.Organization

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

**Рисунок 2.4 — Найден отсутствующий импорт**

Импортируем нужный класс для устранения ошибки



**Рисунок 2.5 — Добавление отсутствующего импорта**

Убедимся в том, что все ошибки найдены и исправлены, и запустим проект

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**Рисунок 2.6 — Отсутствие нарушение в зависимостях и импортах**

Изображение выглядит как текст, Мультимедийное программное обеспечение, программное обеспечение, Графическое программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**Рисунок 2.7 — Успешный build**

После успешного билда проекта сгенерируем его документацию. Документация проекта представлена на рисунке 2.8

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Контент, сгенерированный ИИ, может содержать ошибки.

**Рисунок 2.8 — Сгенерированная документация**

Применим команду gradlew для запуска проекта. ./gradlew run — это команда, которая запускает приложение, используя Gradle. Она предполагает, что в проекте есть задача run, которая определена build.gradle файле.

Кроме того, ./gradlew позволяет запускать Gradle-команды в проекте, даже если нет установленного Gradle на компьютере. Он автоматически загрузит нужную версию Gradle для проекта.

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Графическое программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**Рисунок 2.9 — Успешный запуск**

После успешного запуска проекта выполним прочие задания практической работы:

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

Изображение выглядит как Мультимедийное программное обеспечение, программное обеспечение, Графическое программное обеспечение, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

**Рисунок 2.10 — Запрос состояния сервера по адресу**

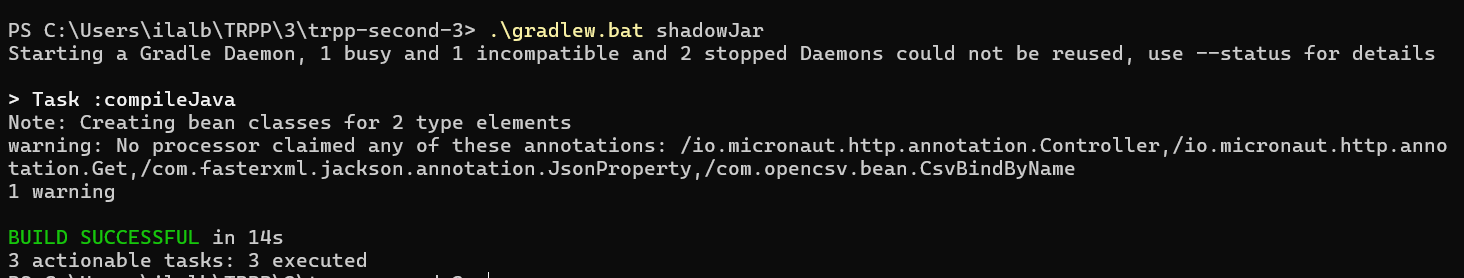
2)Запросить сущность по идентификатору (GET запрос по адресу: <http://localhost:8080/сущность/идентификатор>)

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.

**Рисунок 2.11 — Запрос сущности по идентификатору**

3)В задаче shadowJar добавить к jar-файлу вашу фамилию:



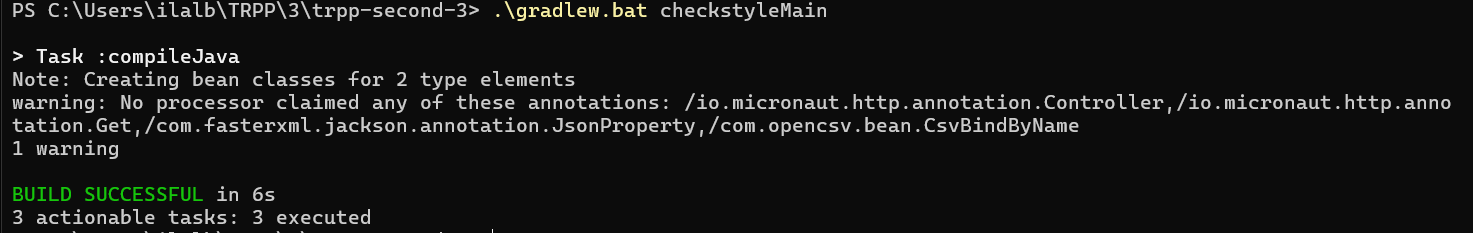
**Рисунок 2.12 — Запуск shadowJar**

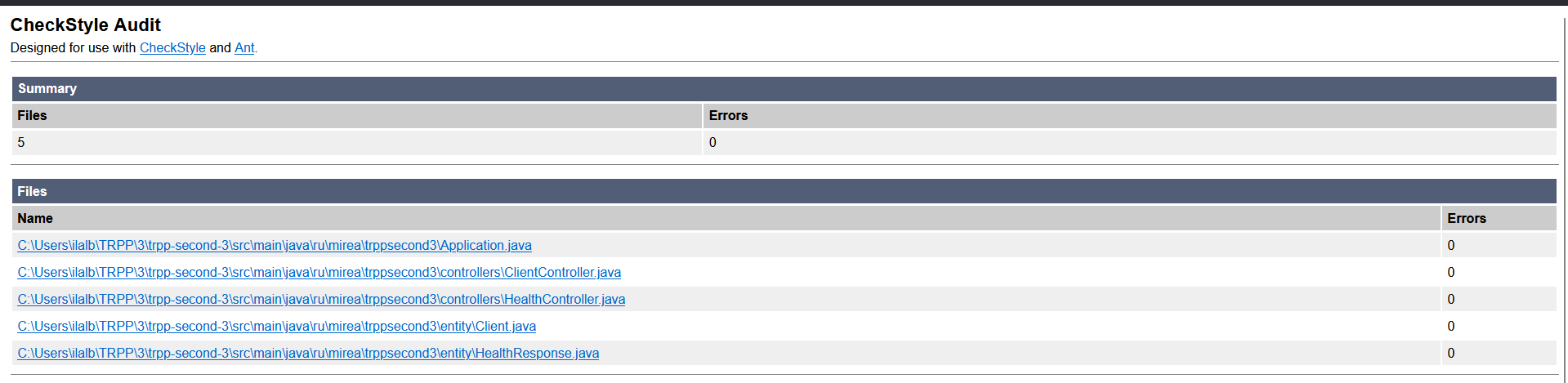


**Рисунок 2.13 — Добавление моей фамилии к jar-файлу**

Выполнить задачу checkstyleMain. Посмотреть сгенерированный

отчет. Устранить ошибки оформления кода.



**Рисунок 2.14 — Генерация отчета**

**Рисунок 2.14 — Просмотр сгенерированного отчета(ошибок нет)**

Выполнение практической работы закончено, перейдем к ответам на контрольные вопросы.

# **3. КОНТРОЛЬНЫЕ ВОПРОСЫ**

**1. Чем компиляция отличается от сборки?**

Компиляция отличается от сборки тем, что компиляция — это процесс преобразования исходного кода программы в машинный код или интерпретируемый код, тогда как сборка включает в себя этапы компиляции, линковки и другие операции для создания исполняемого файла или другого конечного продукта.

**2. Что такое система сборки?**

Система сборки — это инструментарий или программное обеспечение, которое автоматизирует процесс компиляции, тестирования и создания программного продукта из его исходных файлов. Она облегчает управление зависимостями, конфигурацией проекта и другими аспектами разработки.

**3. Что такое репозиторий?**

Репозиторий — это хранилище данных, в котором хранятся файлы исходного кода, библиотеки, зависимости и другие компоненты программного проекта. Репозитории используются для управления версиями и распространения программного обеспечения.

**4. Как указать зависимости проекта?**

Для указания зависимостей проекта в файле проекта или конфигурационном файле (например, build.gradle в Gradle или pom.xml в Maven) обычно используются специальные секции или блоки, где перечисляются необходимые библиотеки или модули, а также их версии.

**5. Что такое gradle?**

Gradle — это инструмент автоматизации сборки проектов, который использует декларативный DSL (язык описания предметной области) на базе Groovy или Kotlin.

**6. Что такое maven?**

Maven — это инструмент автоматизации сборки и управления проектами, который использует XML-файлы для описания структуры проекта, его зависимостей и сценариев сборки.

# **4. ВЫВОДЫ**

Было произведено знакомство с системой сборки Gradle и возможностями gradle, а также управление зависимостями.