

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**

**«Технологии разработки программных приложений»**

**Тема: «Системы сборки»**

Выполнил студент группы ИКБО-15-22 Оганнисян Г.А.

Принял Исабекова О.А.

Практическая работа выполнена «\_\_»\_\_\_\_\_\_\_202\_\_г.  *(подпись студента)*

«Зачтено» «\_\_»\_\_\_\_\_\_\_202\_\_ г. *( подпись руководителя)*

# СОДЕРЖАНИЕ

**ВЫПОЛНЕНИЕ ЗАДАНИЯ ....................................................................... 6**

**КОНТРОЛЬНЫЕ ВОПРОСЫ .................................................................. 16**

**ВЫВОД......................................................................................................... 17**

# ВЫПОЛНЕНИЕ ЗАДАНИЯ

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

Был клонирован заранее форкнутый репозиторий (см.рис.1).

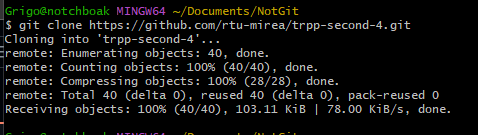
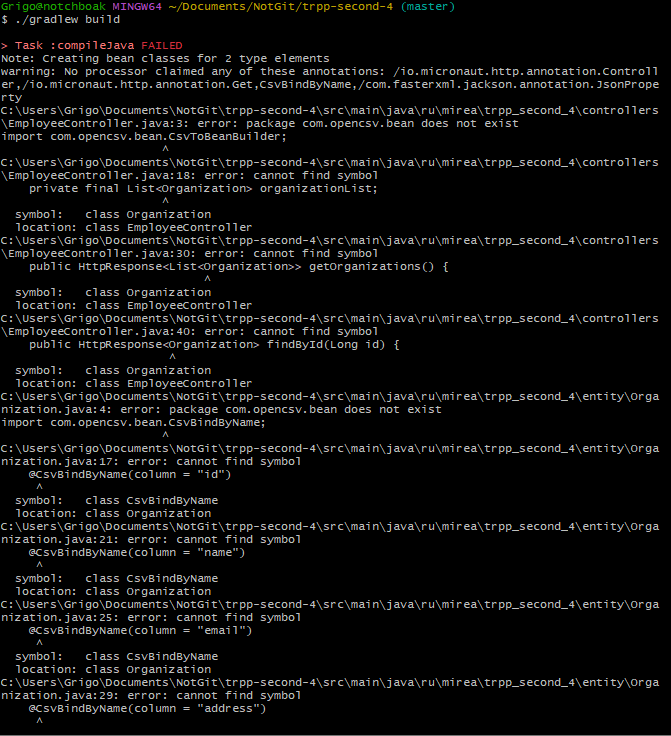


Рисунок 1 – клонирование репозитория

Затем была произведена попытка сборки проекта (см.рис.2).

Исходя из текста ошибки, была добавлена отсутствующая зависимость

(см.рис.3).



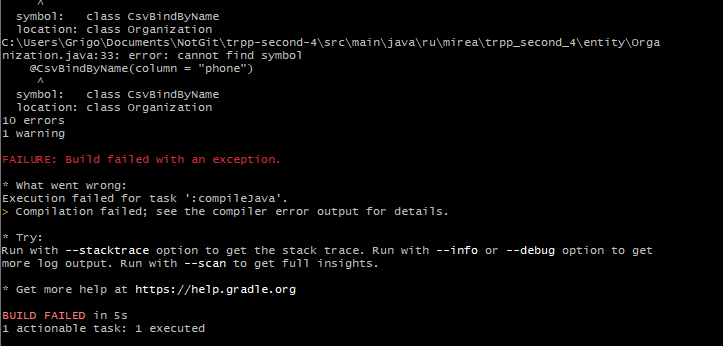


Рисунок 2 – первая попытка сборки

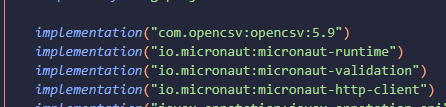


Рисунок 3 – добавленная зависимость

При второй попытке сборки была получена новая ошибка (см.рис.4). Ошибка была решена добавлением недостающего import`а в класс HealthControll (см.рис.5).

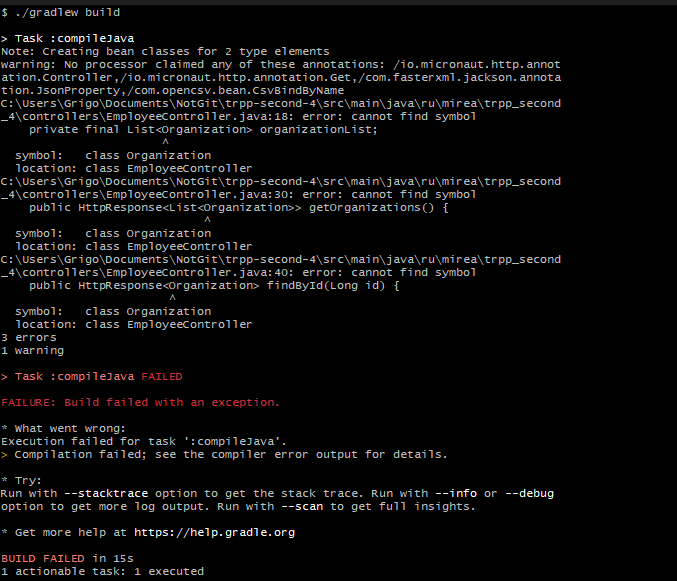


Рисунок 4 – вторая попытка сборки

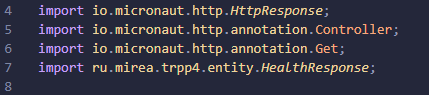
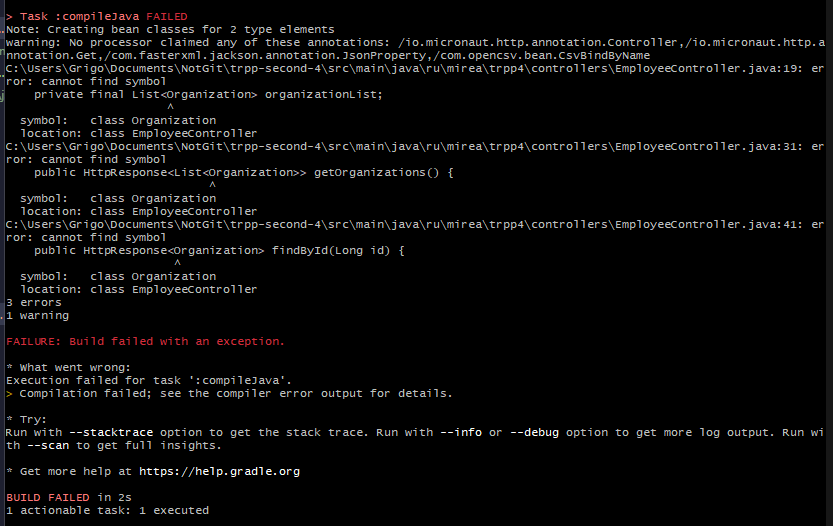


Рисунок 5 – добавление недостающего import`а

При следующей попытке сборки получена ошибка, указывающая на слишком длинную строку в классе EmployeeController (см.рис.6), решение проблемы (см.рис.7).

Рисунок 6 – попытка сборки, выдающая ошибку в классе EmployeeController

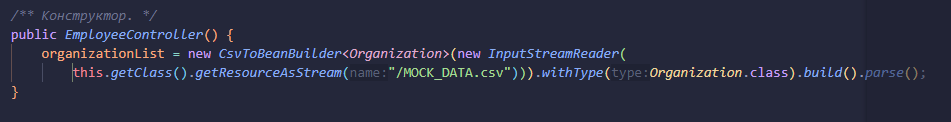


Рисунок 7 – исправление ошибки в классе EmployeeController

При следующей попытке сборки получена ошибка, указывающая на неправильный порядок import`ов в классе TrppTest (см.рис.8), решение проблемы (см.рис.9).

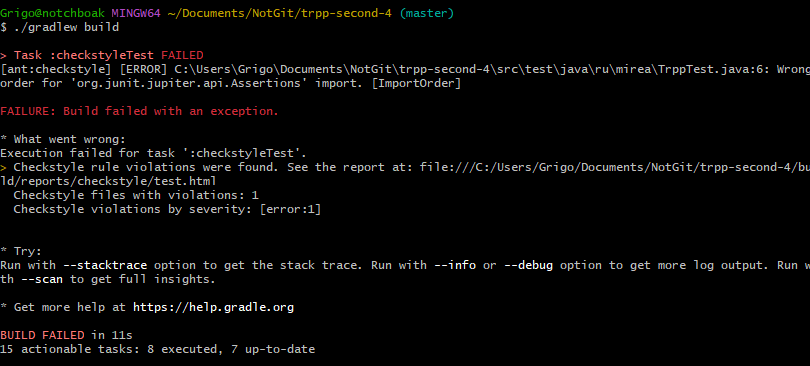


Рисунок 8 – попытка сборки, выдающая ошибку в классе TrppTest

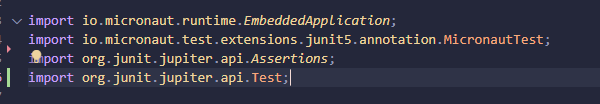


Рисунок 9 – исправление ошибки в классе TrppTest

2. В некоторых классах поправить имя пакета

При попытке сборки проекта были найдены классы, в которых требуется поправить имя пакета. Были внесены необходимые правки:

Замена \*.trpp-second-4\* на \*.trpp4\*



Рисунок 11 – изменение имени пакета в классе

В итоге проект стал успешно собираться (см.рис.12).

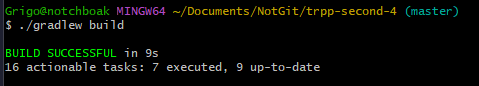


Рисунок 16 – успешная сборка проекта

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

Была собрана документация проекта при помощи команды javadoc

(см.рис.17). В полученной документации были найдены запросы

состояния (см.рис.18) и сущности (см.рис.19).

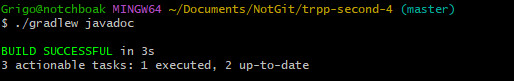


Рисунок 17 – сборка документации

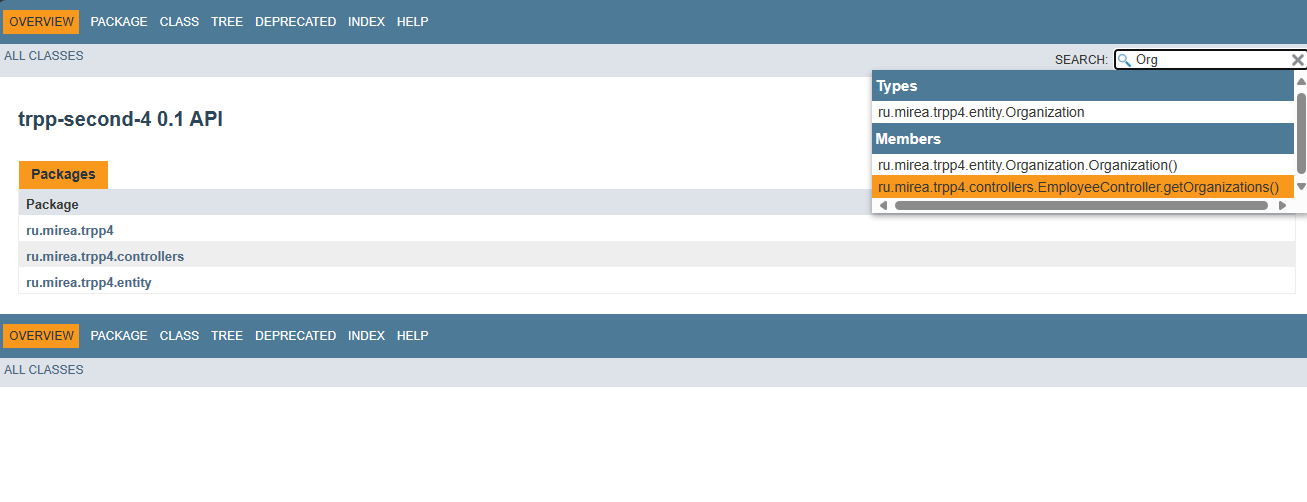


Рисунок 18 – документация о методе getOrganizations()

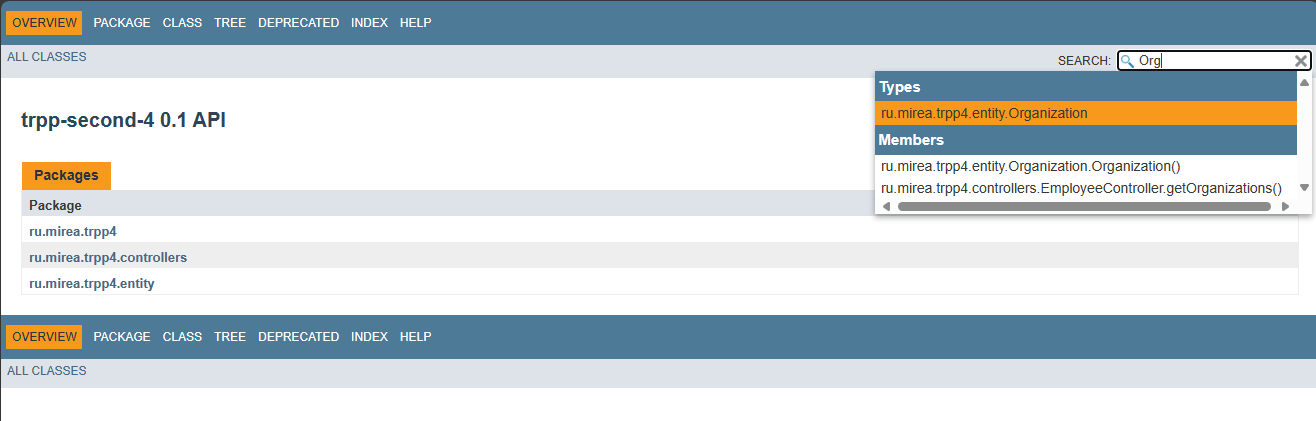


Рисунок 19 – документация о сущности Organizations

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

Был собран UberJar (см.рис.20) при помощи команды shadowJar. Затем было запущено приложение (см.рис.21).

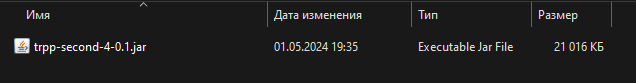
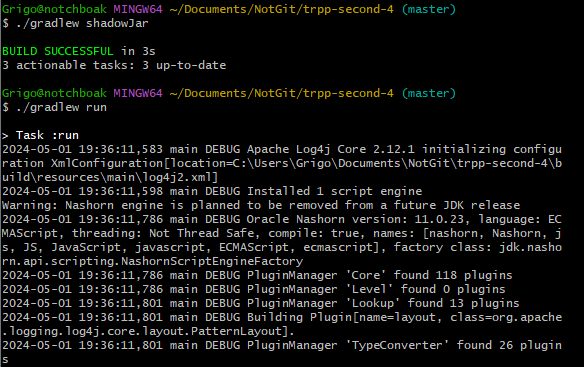


Рисунок 20 – собранный UberJar



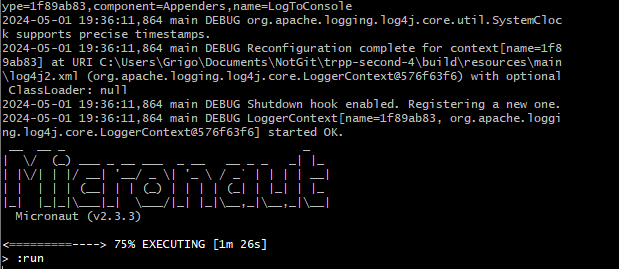


Рисунок 21 – запуск приложения

1. Запросить состояние запущенного сервера (GET запрос по адресу

http://localhost:8080)

Было запрошено состояние сервера (см.рис.22).

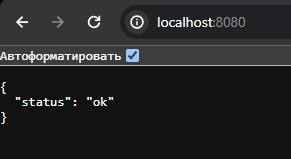


Рисунок 22 – состояние сервера

1. Запросить сущность по идентификатору (GET запрос по адресу: [http://localhost:8080/сущность/идентификатор)](http://localhost:8080/сущность/идентификатор)

Была запрошена сущность Organization по идентификатору 361 – последним трем цифрам в серийном номере студенческого билета (см.рис.23).

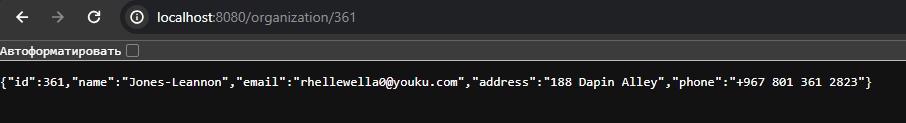


Рисунок 23 – запрошенная сущность

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

В задаче shadowJar была добавлена фамилия к jar-файлу (см.рис.24 – 25).

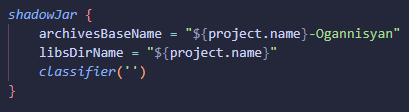


Рисунок 24 – добавление фамилии

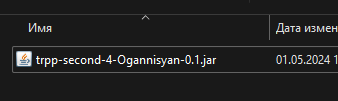


Рисунок 25 – jar файл с добавленной фамилией

1. Выполнить задачу checkstyleMain. Посмотреть сгенерированный отчет.

Устранить ошибки оформления кода.

Была выполнена задача checkstyleMain (см.рис.26), был проверен сгенерированный отчет на предмет ошибок оформления кода (см.рис.27)

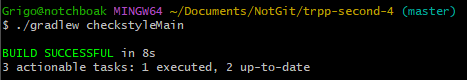


Рисунок 26 – выполнение задачи checkstyleMain

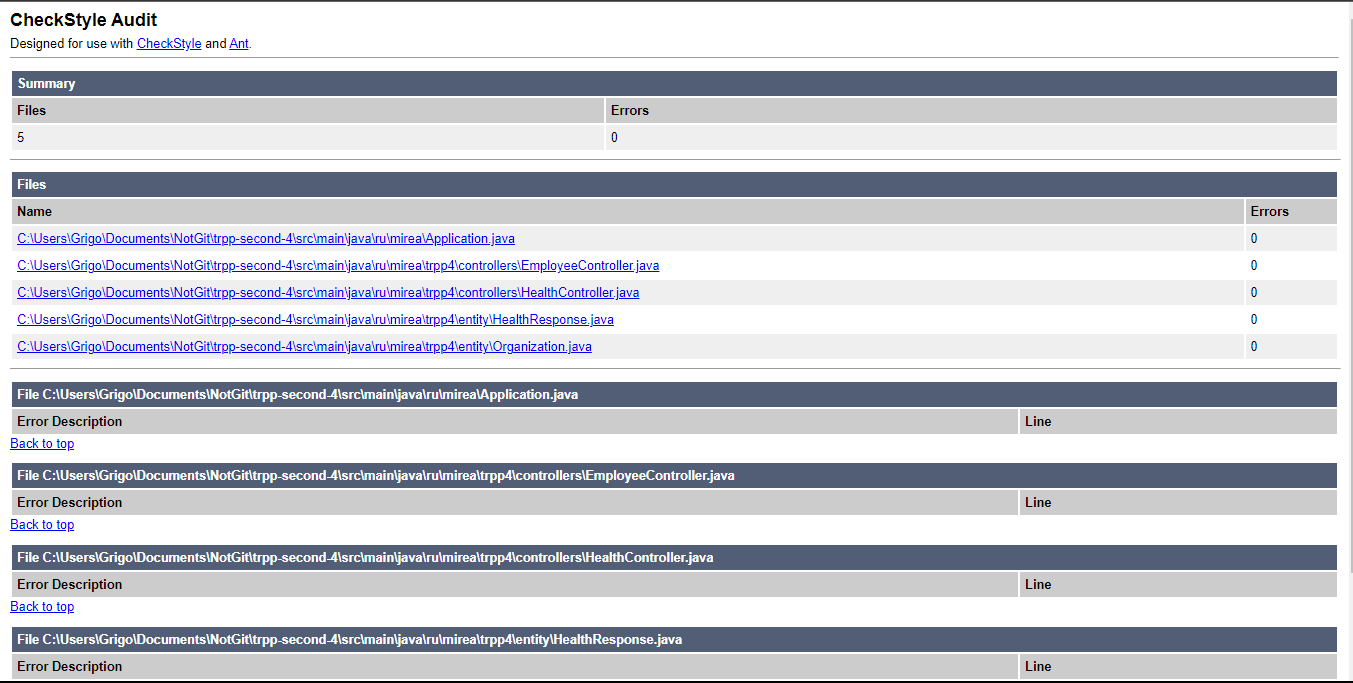


Рисунок 27 – сгенерированный отчет

# КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Чем компиляция отличается от сборки?

Компиляция — это процесс преобразования исходного кода программы в машинный код. Сборка - процесс преобразования файлов с исходным кодом и их компиляция в артефакты, составляющие приложение, такие как бинарные и исполняемые файлы. То есть сборка включает в себя не только компиляцию, но и другие задачи.

4. Как указать зависимости проекта?

Зависимости указываются в build.gradle в блоке dependencies

1. Что делает задача clean?

Задача clean - очистка всех сборочных директорий.

1. Что делает задача build?

Задача build — собирает приложение целиком.

1. Что делает задача run?

Задача run – запускает приложение.

1. Что делает задача test?

Задача test – запускает прогон тестов.

# ВЫВОД

Были получены навыки по работе с системой сборки gradle.