



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА - Российский технологический университет»

РТУ МИРЭА

**Институт информационных технологий (ИИТ)
Кафедра математического обеспечения и стандартизации ИТ**

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ
по дисциплине «Технологии разработки программных приложений»

Практическая работа № 2

Студент группы *ИНБО-06-21* *Онацкий Е.Е.*

(подпись)

Ассистент *Петрова А. А.*

(подпись)

Отчет представлен « » _____ 2023 г.

Москва 2023 г.

СОДЕРЖАНИЕ

ЧАСТЬ 1.....	3
ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ.....	13
ВЫВОДЫ.....	14

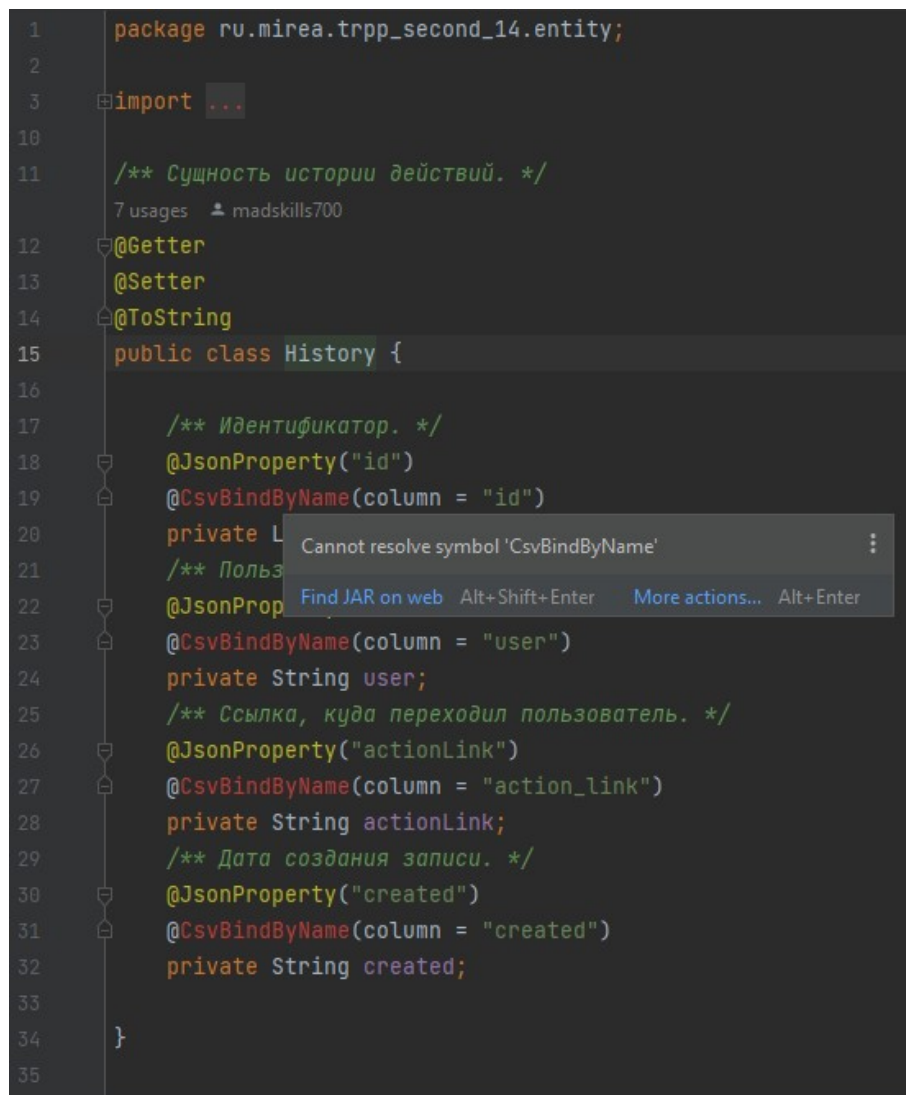
ЧАСТЬ 1.

Задание персонального варианта: репозиторий: <https://github.com/rtu-mirea/trpp-second-14>, сущность ru.mirea.entity.History

Пункт 1

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

На Рисунке 1.1 показана отсутствующая зависимость.



```
1 package ru.mirea.trpp_second_14.entity;
2
3 import ...
4
10
11 /** Сущность истории действий. */
12 @Getter
13 @Setter
14 @ToString
15 public class History {
16
17     /** Идентификатор. */
18     @JsonProperty("id")
19     @CsvBindByName(column = "id")
20     private Long id;
21     /** Пользователь. */
22     @JsonProperty("user")
23     @CsvBindByName(column = "user")
24     private String user;
25     /** Ссылка, куда переходил пользователь. */
26     @JsonProperty("actionLink")
27     @CsvBindByName(column = "action_link")
28     private String actionLink;
29     /** Дата создания записи. */
30     @JsonProperty("created")
31     @CsvBindByName(column = "created")
32     private String created;
33
34 }
35
```

Рисунок 1.1 — Отсутствует библиотека с аннотацией @CsvBindByName

Добавим в dependencies необходимый источник (Рисунок 1.2) и увидим, что библиотека подключилась (Рисунок 1.3)

```
22
23 dependencies {
24     annotationProcessor 'org.projectlombok:lombok:1.18.18'
25     compileOnly 'org.projectlombok:lombok:1.18.18'
26     implementation group: 'com.opencsv', name: 'opencsv', version: '4.6'
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("org.apache.logging.log4j:log4j-core:2.12.1")
32     runtimeOnly("org.apache.logging.log4j:log4j-api:2.12.1")
33     runtimeOnly("org.apache.logging.log4j:log4j-slf4j-impl:2.12.1")
34 }
35
36
```

Рисунок 1.2 — Подключение необходимой библиотеки

```
15     public class History {
16
17         /** Идентификатор. */
18         @JsonProperty("id")
19         @CsvBindByName(column = "id")
20         private Long id;
21         /** Пользователь. */
22         @JsonProperty("user")
23         @CsvBindByName(column = "user")
24         private String user;
25         /** Ссылка, куда переходил пользователь. */
26         @JsonProperty("actionLink")
27         @CsvBindByName(column = "action_link")
28         private String actionLink;
29         /** Дата создания записи. */
30         @JsonProperty("created")
31         @CsvBindByName(column = "created")
32         private String created;
33
34     }
35
```

Рисунок 1.3 — Библиотека подключена правильно, аннотация найдена

Пункт 2

В некоторых классах необходимо поправить имя импортируемого пакета. (см. Рисунки 1.4-1.5)

```
import io.micronaut.http.HttpResponse;
import io.micronaut.http.annotation.Controller;
import io.micronaut.http.annotation.Get;
import работа.entity.HealthResponse;

/** Проверка состояния сервера. */
@madskills700 *
@Controller()
public class HealthController {

    /**
     * Проверить состояние сервера.
     * @return ответ
     * 200 - ОК
     */
    @madskills700
    @Get
    public HttpResponse<HealthResponse> healthCheck() { return HttpResponse.ok(new HealthResponse( status: "OK")); }
}
```

Рисунок 1.4 — Импорт класса HealthResponse в класс HealthController

```
C:\Windows\system32\cmd.exe
> Task :help
Welcome to Gradle 6.8.
To run a build, run gradlew <task> ...
To see a list of available tasks, run gradlew tasks
To see a list of command-line options, run gradlew --help
To see more detail about a task, run gradlew help --task <task>
For troubleshooting, visit https://help.gradle.org
BUILD SUCCESSFUL in 555ms
1 actionable task: 1 executed
```

Рисунок 1.5 — Успешный вызов вращера

Пункт 3

Необходимо собрать документацию проекта, найти в ней запросы состояния и сущности по идентификатору. (см. Рисунки 1.6-1.10)

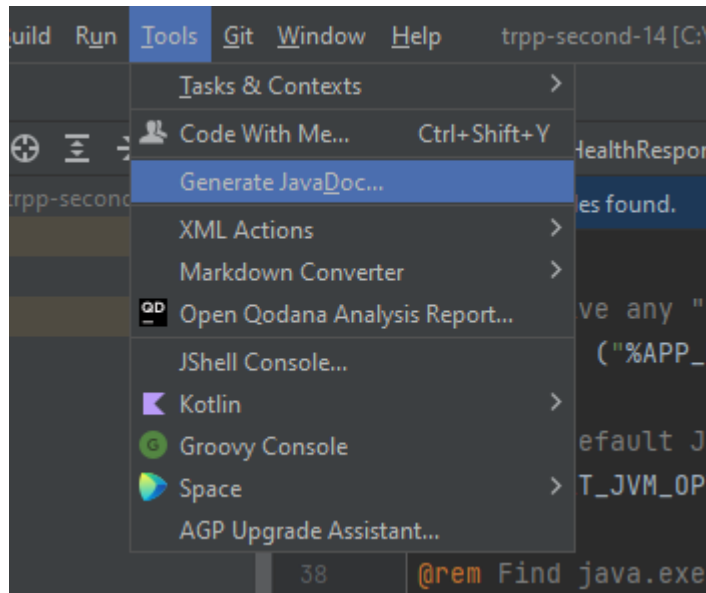


Рисунок 1.6 — Переход во вкладку Tools/Generate JavaDoc

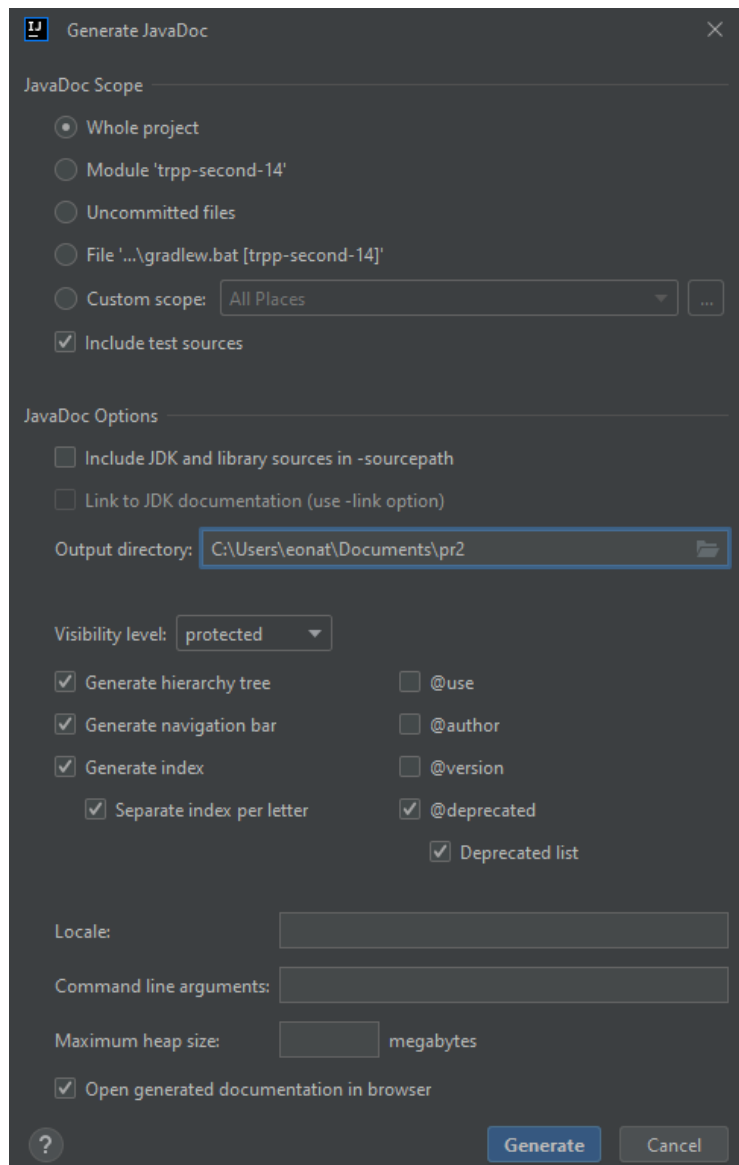


Рисунок 1.7 — Генерация документации JavaDoc

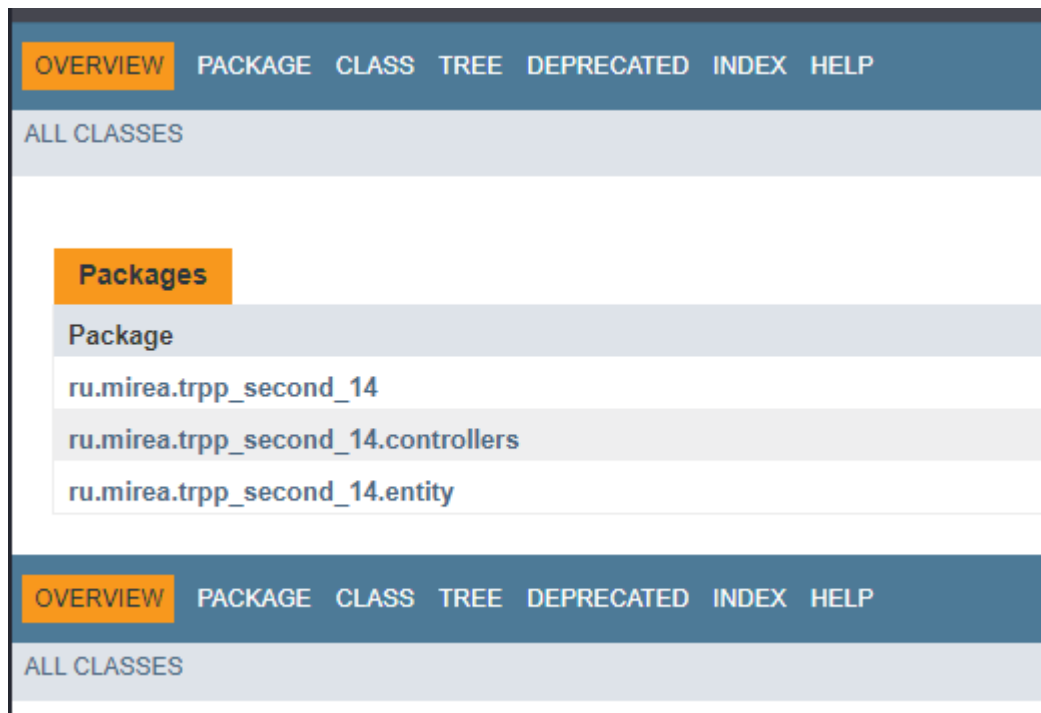


Рисунок 1.8 — Начальная страница документации JavaDoc

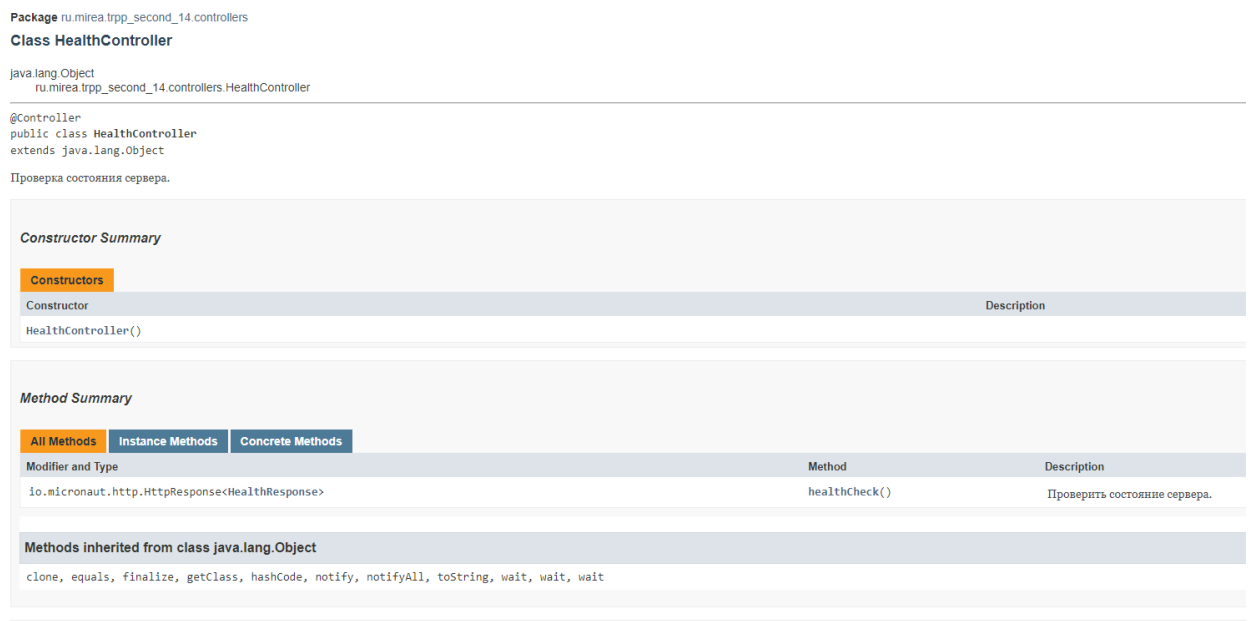


Рисунок 1.9 — Запрос состояния healthCheck в классе HealthController

Package ru.mirea.trpp_second_14.controllers
Class HistoryController
 java.lang.Object
 ru.mirea.trpp_second_14.controllers.HistoryController

```
@Controller("/history")
public class HistoryController
extends java.lang.Object
```

Контроллер для работы с историей.

Constructor Summary

Constructors	Description
HistoryController()	Конструктор.

Method Summary

All Methods	Instance Methods	Concrete Methods
Modifier and Type	Method	Description
io.micronaut.http.HttpResponse<History>	findById(java.lang.Long id)	Найти действие по идентификатору.
io.micronaut.http.HttpResponse<java.util.List<History>>	getEmployee()	Получить список действий.

Рисунок 1.10 — запрос состояния и сущности по идентификатору findById в классе HealthController

Пункт 4

Необходимо собрать jar со всеми зависимостями (так называемый UberJar), после чего запустить приложение. (см. Рисунки 1.11-1.15)

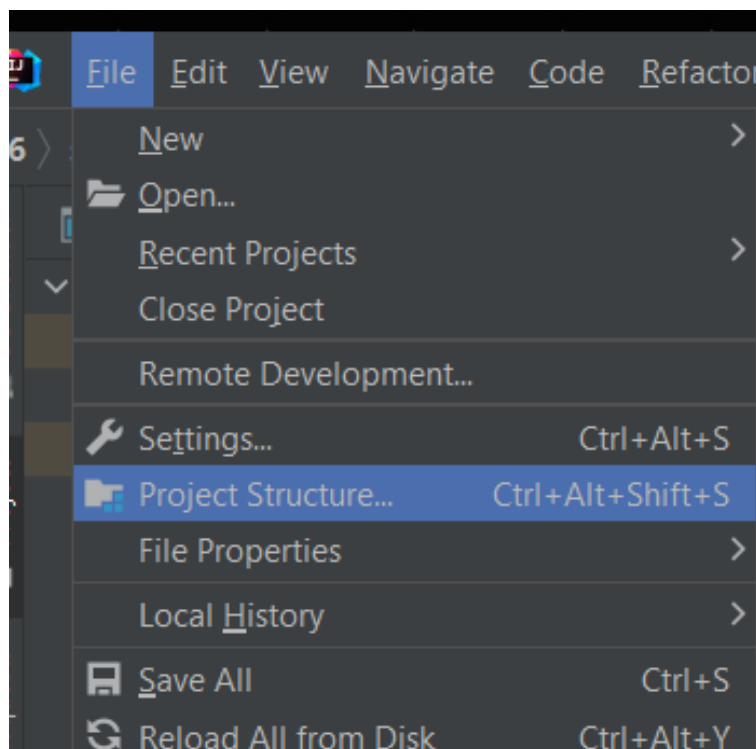


Рисунок 1.11 — Переход во вкладку File/Project Structure

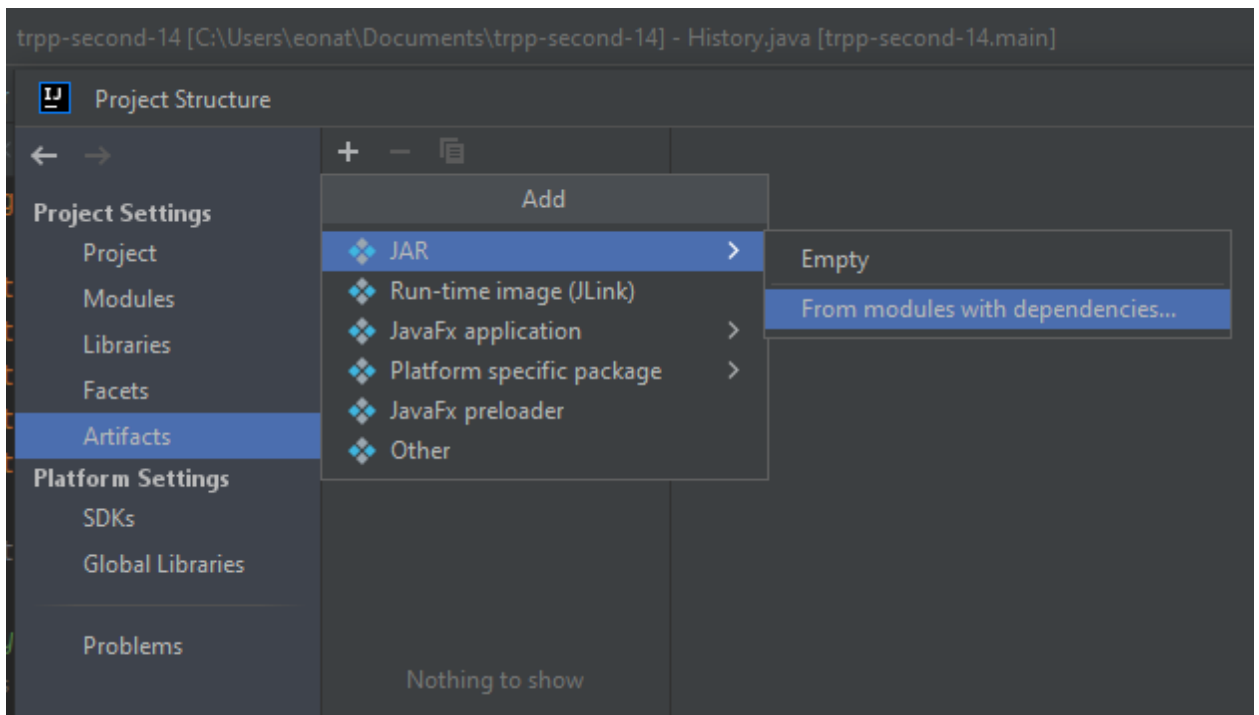


Рисунок 1.12 — Добавление JAR

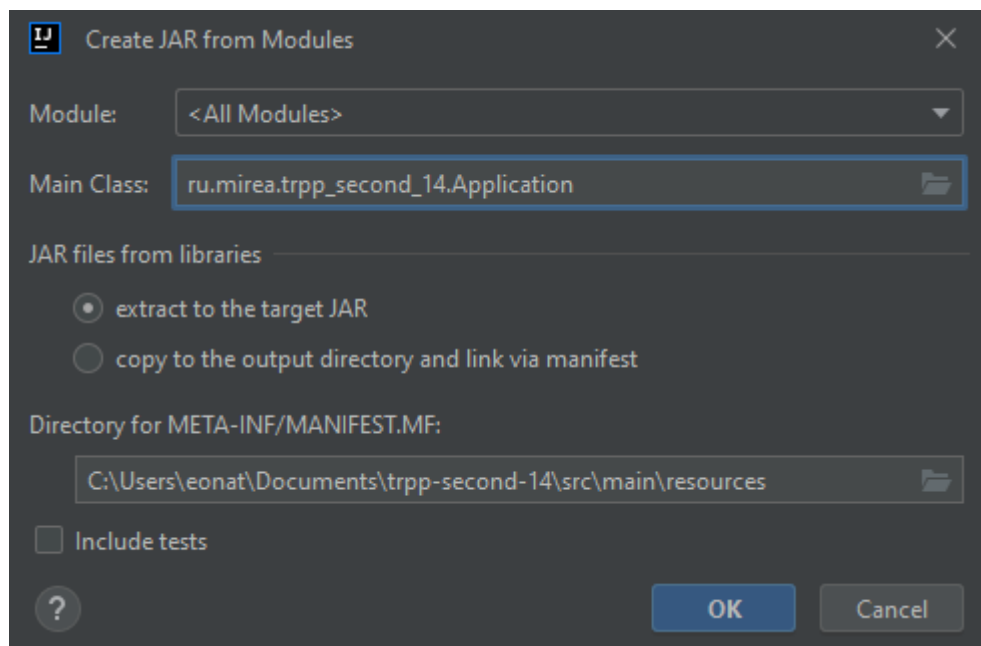


Рисунок 1.13 — Создание JAR на основе главного класса Application.java

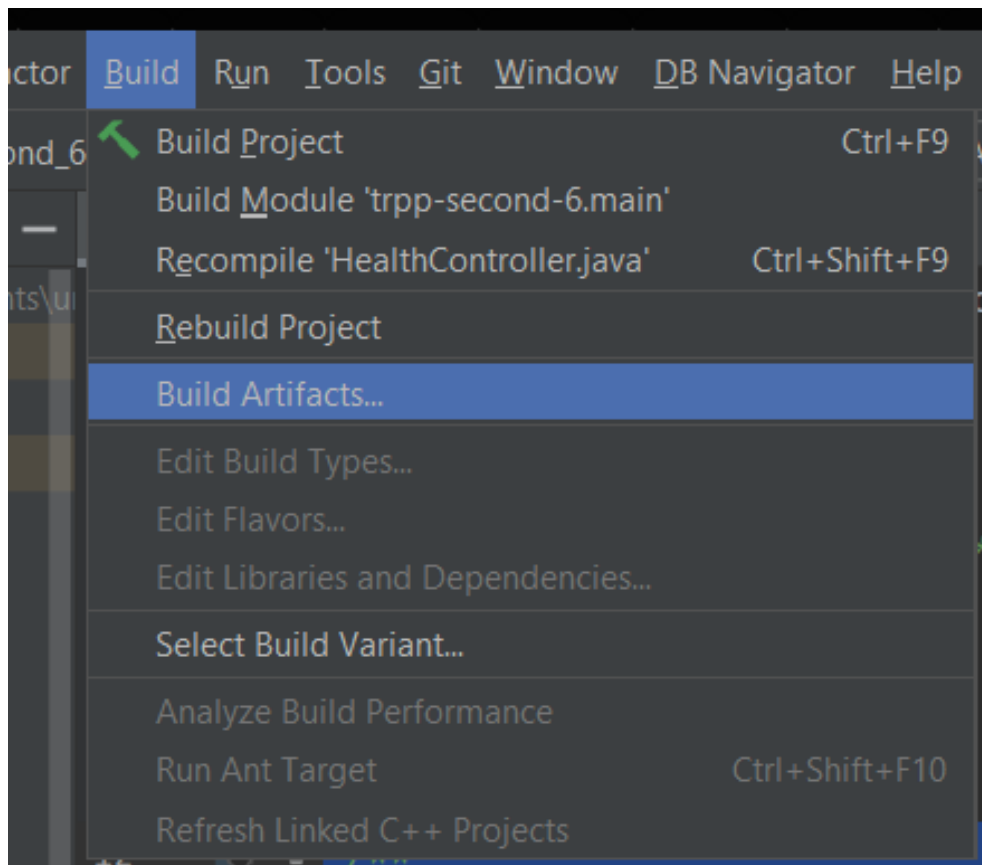


Рисунок 1.14 — Сборка JAR

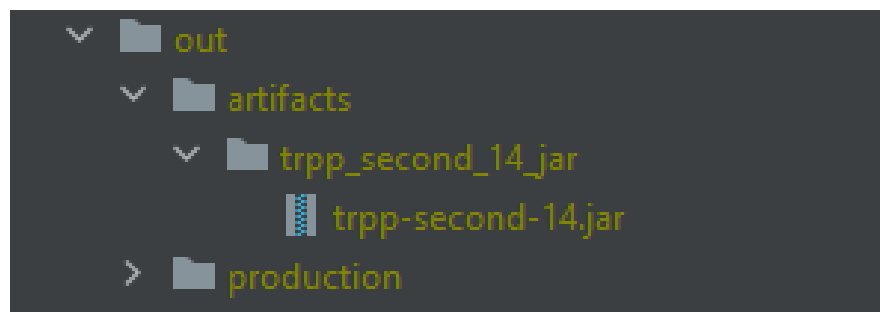


Рисунок 1.15 — Добавленный JAR в файловой системе

Пункт 5

Запросить состояние запущенного сервера (GET запрос по адресу <http://localhost:8080>). (Рисунок 1.16)

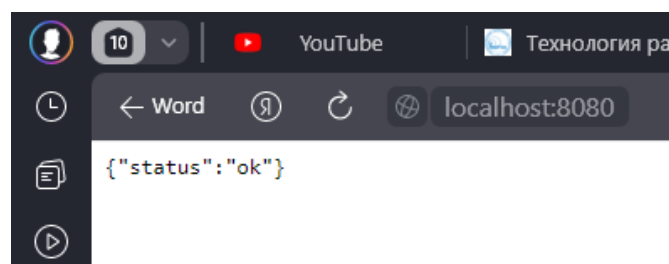
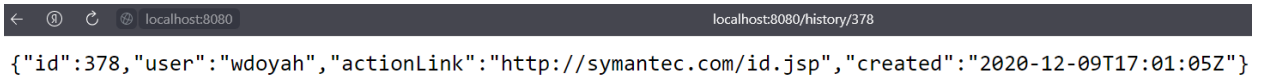


Рисунок 1.16 — Состояние запущенного сервера

Пункт 6

Запросить сущность по идентификатору (GET запрос по адресу: <http://localhost:8080/сущность/идентификатор>). В качестве идентификатора используются три последних цифры номера студенческого билета. (Рисунок 1.17)

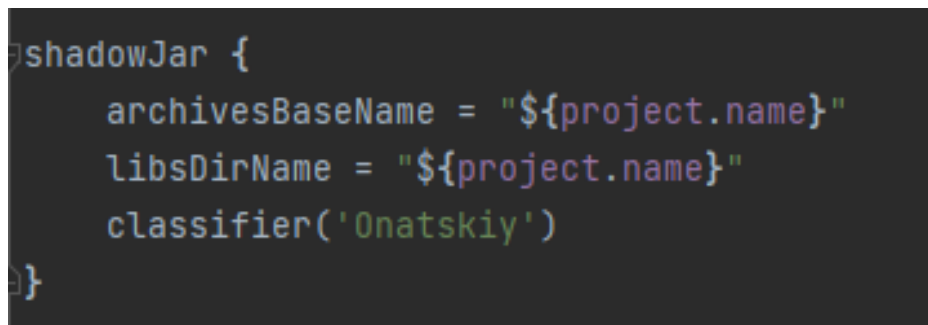


```
{"id":378,"user":"wdoyah","actionLink":"http://symantec.com/id.jsp","created":"2020-12-09T17:01:05Z"}
```

Рисунок 1.17 — Результат запроса сущности по идентификатору 378

Пункт 7

В задаче shadowJar добавим к jar-файлу мою фамилию “Онацкий”. (рисунок 1.18)



```
shadowJar {  
    archivesBaseName = "${project.name}"  
    libsDirName = "${project.name}"  
    classifier('Onatskiy')  
}
```

Рисунок 1.18 — Добавление фамилии в shadowJar

Пункт 8

Выполним задачу checksyleMain. На Рисунке 1.19 видно, что были выявлены некоторые ошибки. Посмотрим информацию о них в отчете по ссылке. (Рисунок 1.20)

```

shadowJar {
    archivesBaseName = "${project.name}"
    libDirName = "${project.name}"
    classifier('0natskiy')
}

checkstyle {
    toolVersion '7.8.1'
    configFile file("config/checkstyle/checkstyle.xml")
}

checkstyleMain {
    source = 'src/main/java'
}

checkstyleTest {
    source = 'src/test/java'
}

```

Execution failed for task ':checkstyleMain'.
> Checkstyle rule violations were found. See the report at: <file:///C:/Users/eonat/Documents/trpp-second-14/build/reports/checkstyle/main.html>
Checkstyle files with violations: 5
Checkstyle violations by severity: [error:7]
* Try:
Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more log output. Run with --scan to get full insights.

Рисунок 1.19 – сообщение об ошибках

CheckStyle Audit
Designed for use with [CheckStyla](#) and [Ant](#)

Summary	
Files	Errors
5	5

Name	Errors
C:\Users\eonat\Documents\trpp-second-14\src\main\java\ru\Mirea\TrppSecond14\Application.java	1
C:\Users\eonat\Documents\trpp-second-14\src\main\java\ru\Mirea\TrppSecond14\controllers\HealthController.java	1
C:\Users\eonat\Documents\trpp-second-14\src\main\java\ru\Mirea\TrppSecond14\controllers\HistoryController.java	1
C:\Users\eonat\Documents\trpp-second-14\src\main\java\ru\Mirea\TrppSecond14\entity\HealthResponse.java	1
C:\Users\eonat\Documents\trpp-second-14\src\main\java\ru\Mirea\TrppSecond14\entity\History.java	1

File C:\Users\eonat\Documents\trpp-second-14\src\main\java\ru\Mirea\TrppSecond14\Application.java	
Error Description	Line
Name 'ruMireaTrppSecond14' must match pattern "[a-z]+(\[a-z\]a-z0-9[1])\$"	1
Back to top	

File C:\Users\eonat\Documents\trpp-second-14\src\main\java\ru\Mirea\TrppSecond14\controllers\HealthController.java	
Error Description	Line
Name 'ruMireaTrppSecond14.controllers' must match pattern "[a-z]+(\[a-z\]a-z0-9[1])\$"	1
Back to top	

File C:\Users\eonat\Documents\trpp-second-14\src\main\java\ru\Mirea\TrppSecond14\controllers\HistoryController.java	
Error Description	Line
Name 'ruMireaTrppSecond14.controllers' must match pattern "[a-z]+(\[a-z\]a-z0-9[1])\$"	1
Back to top	

File C:\Users\eonat\Documents\trpp-second-14\src\main\java\ru\Mirea\TrppSecond14\entity\HealthResponse.java	
Error Description	Line
Name 'ruMireaTrppSecond14.entity' must match pattern "[a-z]+(\[a-z\]a-z0-9[1])\$"	1

Рисунок 1.20 – отчёт об ошибках

Исправим ошибки: уменьшим длину одной из строк, чтобы она не превышала 120 символов, и уберём символы нижнего подчеркивания из названия файла, чтобы оно соответствовало регистру. В результате увидим, что проект успешно забилдился. (Рисунок 1.21)

```
3:48:24: Executing 'checkstyleMain'...

> Task :compileJava UP-TO-DATE
> Task :processResources UP-TO-DATE
> Task :classes UP-TO-DATE
> Task :checkstyleMain UP-TO-DATE

BUILD SUCCESSFUL in 89ms
3 actionable tasks: 3 up-to-date
3:48:24: Execution finished 'checkstyleMain'.
|
```

Рисунок 1.21 – успешно выполненный метод

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое checkstyle?

Checkstyle — это инструмент с открытым исходным кодом, который проверяет код на соответствие настраиваемым наборам правил.

2. Что такое UberJar? При помощи какой задачи его собрать?

По сути, толстый jar (также известный как uber-jar) — это самодостаточный архив, который содержит как классы, так и зависимости, необходимые для запуска приложения. Задача shadowJar — собрать UberJar.

3. Что такое micronaut?

Micronaut — это фреймворк на JVM для построения легковесных модульных приложений.

4. Что такое lombok?

Lombok — это библиотека для сокращения кода в классах и расширения функциональности языка Java.

5. Что такое postman?

Postman — это сервис, который используется для ручного и автоматизированного тестирования HTTP API. С его помощью можно выполнять любые запросы через удобный веб-интерфейс, создавать тесты работы API в автоматическом режиме и многое другое.

6. Что такое аннотация в Java?

Java-аннотация — в языке Java специальная форма синтаксических метаданных, которая может быть добавлена в исходный код. Аннотации используются для анализа кода, компиляции или выполнения. Аннотируемы пакеты, классы, методы, переменные и параметры.

ВЫВОДЫ

В ходе проделанной работы были изучены возможности Gradle посредством включения недостающей зависимости и конфигурации JAR-файла. Были изучены способы отладки ошибок при помощи системы Gradle, генерация отчётов и создание билдов.