|  |
| --- |
|  |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий  Кафедра Математического обеспечения и стандартизации информационных технологий  **Отчет по практической работе №2**  по дисциплине «Технология разработки программных приложений»  по теме «Системы сборки»   |  |  | | --- | --- | | **Выполнил:**  Студент группыИКБО-20-22 | Шумахер М. Е. | | **Проверил:** | преподаватель Мельников Д. А. |   МОСКВА 2024 г. |

**СОДЕРЖАНИЕ**

[Задание (Вариант 15) 3](#_Toc160410799)

[Выполнение заданий 4](#_Toc160410800)

[Ответы на вопросы 9](#_Toc160410801)

[Вывод 10](#_Toc160410802)

Задание (Вариант 15)

**Цель работы**: знакомство с системой сборки Gradle. Возможности gradle. Управление зависимостями.

**Задание для выполнения**

Для выполнения необходимо клонировать git-репозиторий (https://github.com/rtu-mirea/trpp-second-15) 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/book/id). 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.

Выполнение заданий

Найдем отсутствующую зависимость (рис. 1) и укажем ее в соответствующем блоке в build.gradle (рис. 2), чтобы проект снова начал собираться. Проверим соответствующую ошибку (рис. 3).

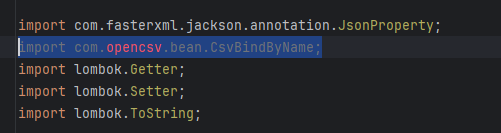


Рисунок 1 – Нахождение отсутствующей зависимости

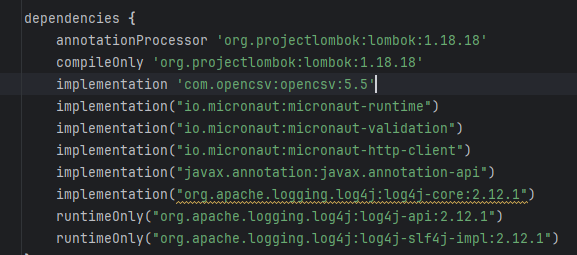


Рисунок 2 – Указание зависимости в build.gradle

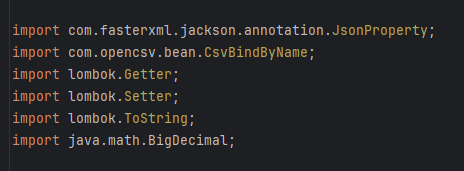


Рисунок 3 – Проверка ошибки в build.gradle

Исправим имя пакета в классе (рис. 4-5). Проверим сборку проекта (рис. 6).

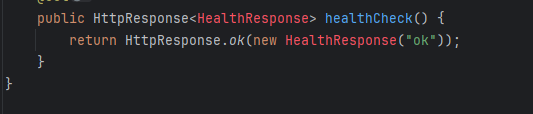


Рисунок 4 – Неправильное имя пакета



Рисунок 5 – Добавление имени пакета

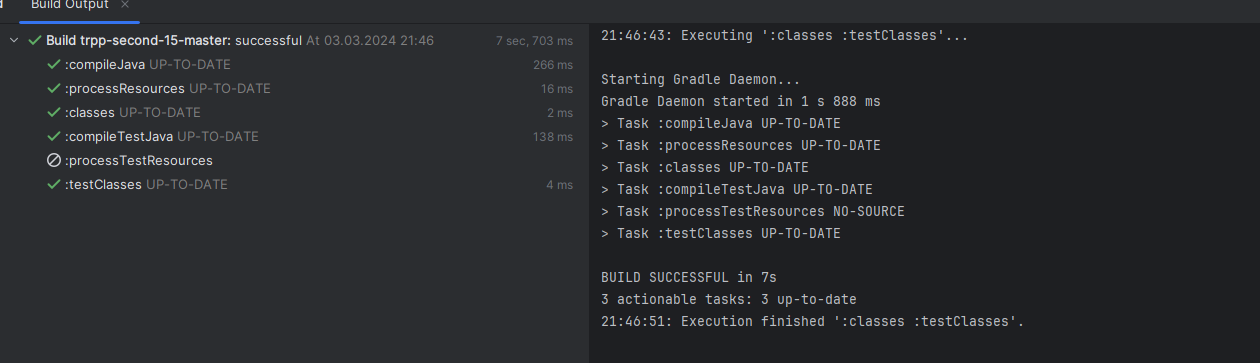


Рисунок 6 – Проверка сборки проекта

Соберем документацию проекта посредством Javadoc, найдем в ней запросы состояния и сущности по идентификатору (рис. 7).

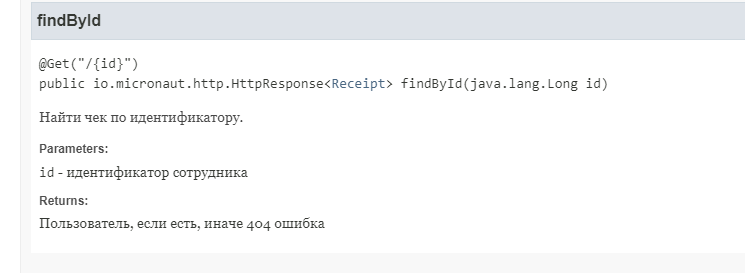


Рисунок 7 – Поиск метода в файле документации

Соберем jar-файл со всеми зависимостями (так называемый UberJar), после чего запустим приложение (рис. 8). По умолчанию, сервер стартует на порту 8080.

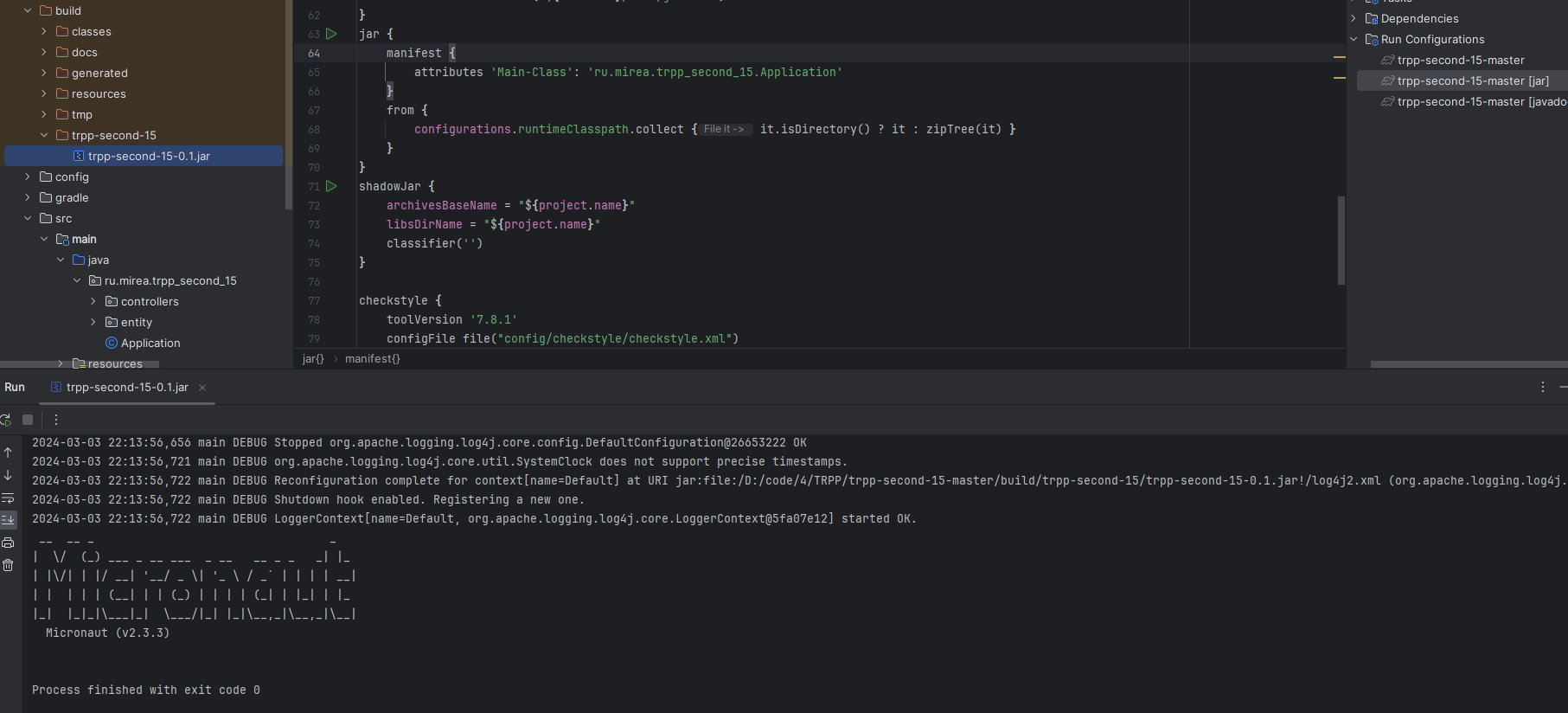


Рисунок 8 – Запуск приложения на 8080 порту

Запросим состояние запущенного сервера (GET запрос по адресу [http://localhost:8080](http://localhost:8080/)) (рис. 9).

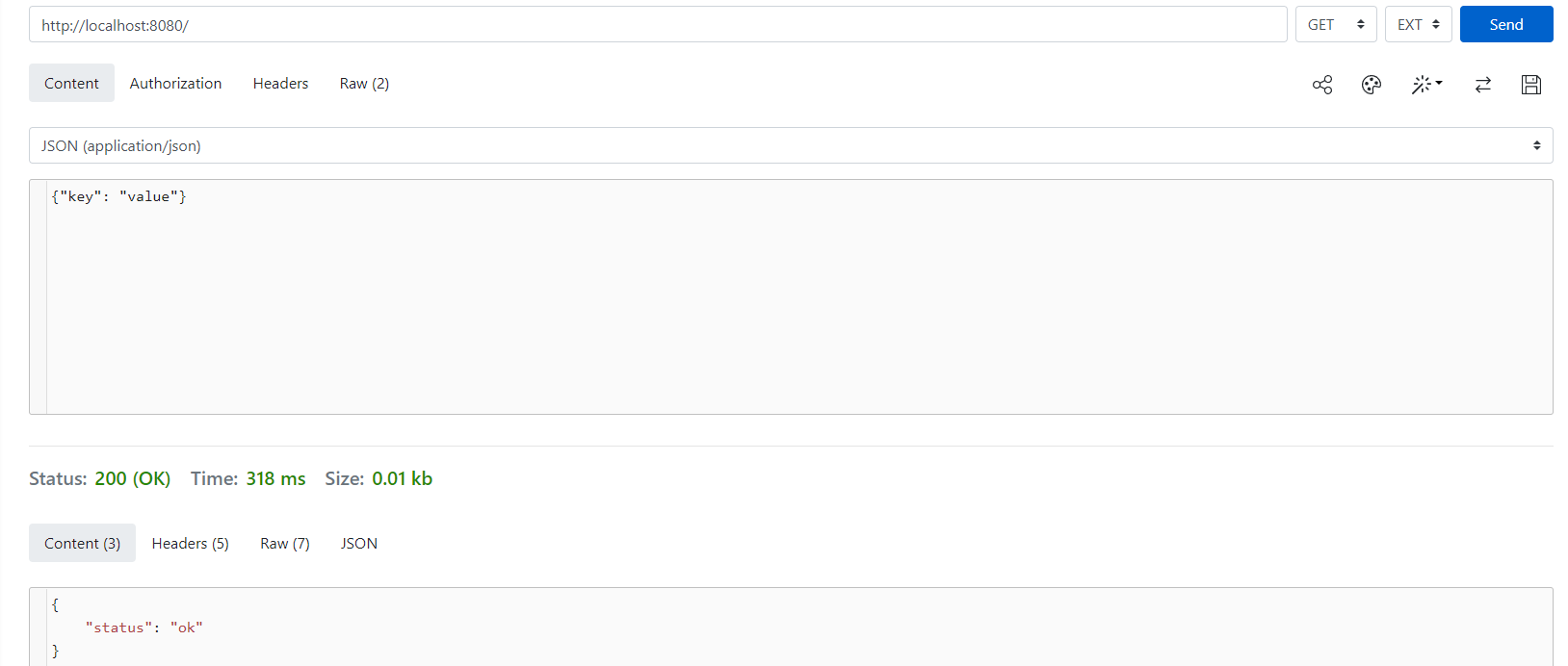


Рисунок 9 – Запрос состояния сервера

Запросим сущность по идентификатору (GET запрос по адресу: [http://localhost:8080/сущность/идентификатор](http://localhost:8080/%D1%81%D1%83%D1%89%D0%BD%D0%BE%D1%81%D1%82%D1%8C/%D0%B8%D0%B4%D0%B5%D0%BD%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%82%D0%BE%D1%80)) (рис. 10). Идентификатором будут 3 последних цифры в серийном номере студенческого билета.

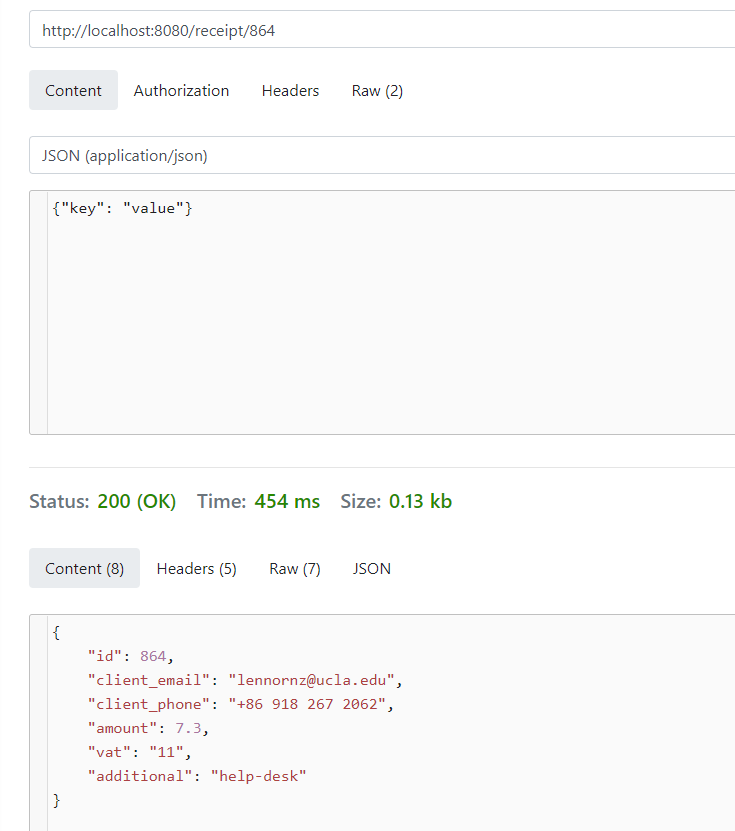


Рисунок 10 – Запрос сущности по идентификатору

В задаче shadowJar добавим к jar-файлу свою фамилию (рис. 11-12).

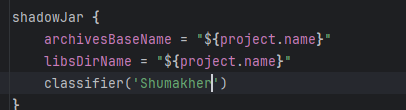


Рисунок 11 – Добавление фамилии к названию jar-файла

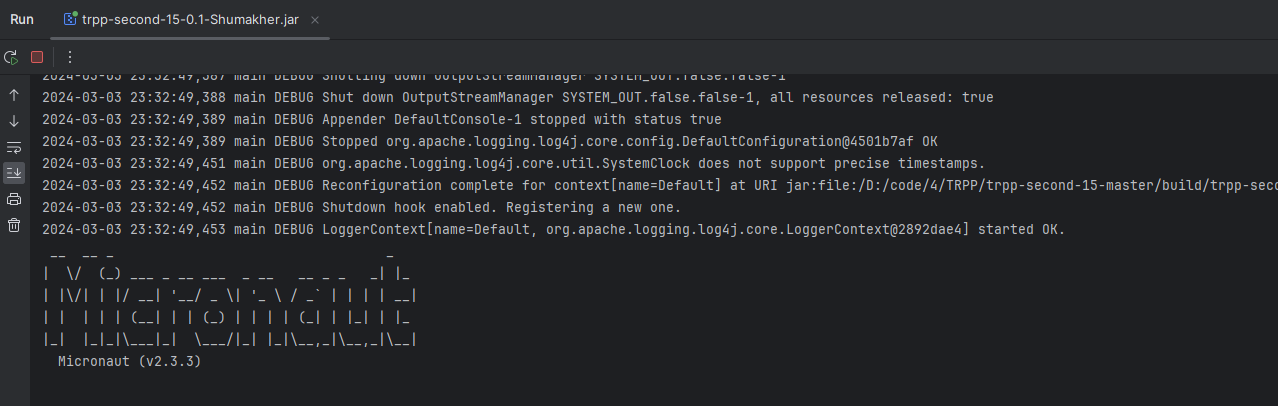


Рисунок 12 – Запуск jar-файла

Выполним задачу checkstyleMain. Посмотрим сгенерированный отчет и устраним ошибки оформления кода (рис. 13-14).



Рисунок 13 – Вывод ошибок по оформлению кода

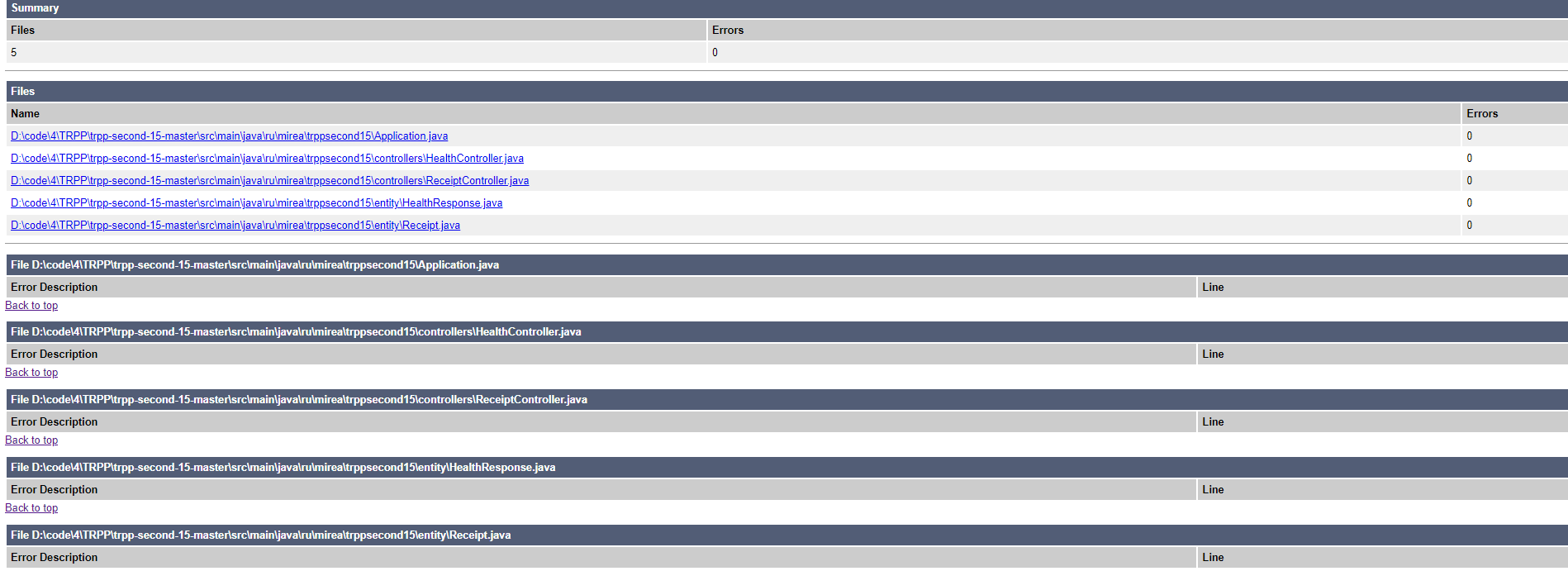


Рисунок 14 – Вывод отсутствия ошибок по оформлению кода

Ответы на вопросы

1. Чем компиляция отличается от сборки?

Компиляция – это процесс преобразования исходного кода программы на языке программирования в машинный код или байт-код, который может быть выполнен на целевой платформе. Компиляция обычно создает промежуточные файлы, такие как файлы классов в Java.

Сборка – это процесс объединения всех необходимых файлов, таких как скомпилированные классы, ресурсы и библиотеки, в исполняемый файл или артефакт, который можно запустить или использовать в приложении.

2. Что такое система сборки?

Система сборки – это инструмент, который автоматизирует процесс сборки приложения из исходного кода. Он позволяет управлять зависимостями, компилировать исходный код, упаковывать приложение и выполнять другие задачи, связанные с разработкой программного обеспечения.

3. Что такое репозиторий?

Репозиторий – это централизованное хранилище для хранения и управления версиями файлов и данных. В контексте разработки программного обеспечения репозиторий часто используется для хранения исходного кода, библиотек, зависимостей и других ресурсов, которые могут быть использованы в проекте.

4. Как указать зависимости проекта?

Зависимости проекта указываются в специальных файлах конфигурации, таких как build.gradle для Gradle или pom.xml для Maven. В этих файлах вы можете добавить зависимости, указав их идентификаторы, версии и другую информацию, необходимую для загрузки и использования библиотек в вашем проекте

5. Что такое Gradle?

Gradle — система сборки проектов с открытым исходным кодом, в которой основной упор идет на гибкость и производительность. В настоящее время поддерживаются 2 языка программирования для написания скриптов сборки: groovy и kotlin.

6. Что такое maven?

Maven – это инструмент для управления проектами и автоматизации сборки проектов на основе конфигурационных файлов pom.xml. Maven предоставляет стандартизированную структуру проекта, управление зависимостями, сборку проекта и другие функции, которые помогают упростить процесс разработки и управления проектами.

Вывод

В результате выполнения работы были получены навыки по работе с Gradle, Javadoc и Micronaut.