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 проекту «Web-сервер для взаимодействия с базой данных с помощью REST API»

по дисциплине «Технологии разработки программных приложений»

Выполнил:
Студент группы ИКБО-20-23 Комисарик М.А.

Проверил: Доцент кафедры МОСИТ,
кандидат технических наук, доцент
Чернов Е.А.

Москва 2025

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ

1.1 Система контроля версий

Создайте репозиторий для своего проекта и обеспечьте к нему удаленный доступ для всех членов команды. В процессе разработки программного продукта необходимо пользоваться выбранной системой контроля версий.

1.2 Разработка проекта и системы сборки

1. Напишите файл README.md с общим описанием проекта, опишите зависимости проекта и команду для его запуска.
2. Настройте систему сборки для вашего проекта.

– В случае проекта на Java: надо использовать Gradle или Maven
– В случае проекта на Python: надо сделать файл requirements.txt со списком зависимостей и подключить setuptools для сборки пакета с проектом

1.3 Развертывание приложения

Создайте Dockerfile, в который запакуйте ваше приложение. Если приложению нужны внешние ресурсы (например, базы данных), то необходимые параметры для подключения приложение должно получать через переменные окружения. Также можно сделать docker-compose.yml.

1.4 Вариант

В качестве задания проекта была выбрана реализация web-сервера для взаимодействия с базой данных PostgreSQL. Языком программирования был выбран Java.

2 ВЫПОЛНЕНИЕ ЗАДАНИЯ

2.1 Система контроля версий.

Перед началом разработки проекта был создан удаленный GitHub репозиторий, который был клонирован на хостовую машину (Рисунок 1).

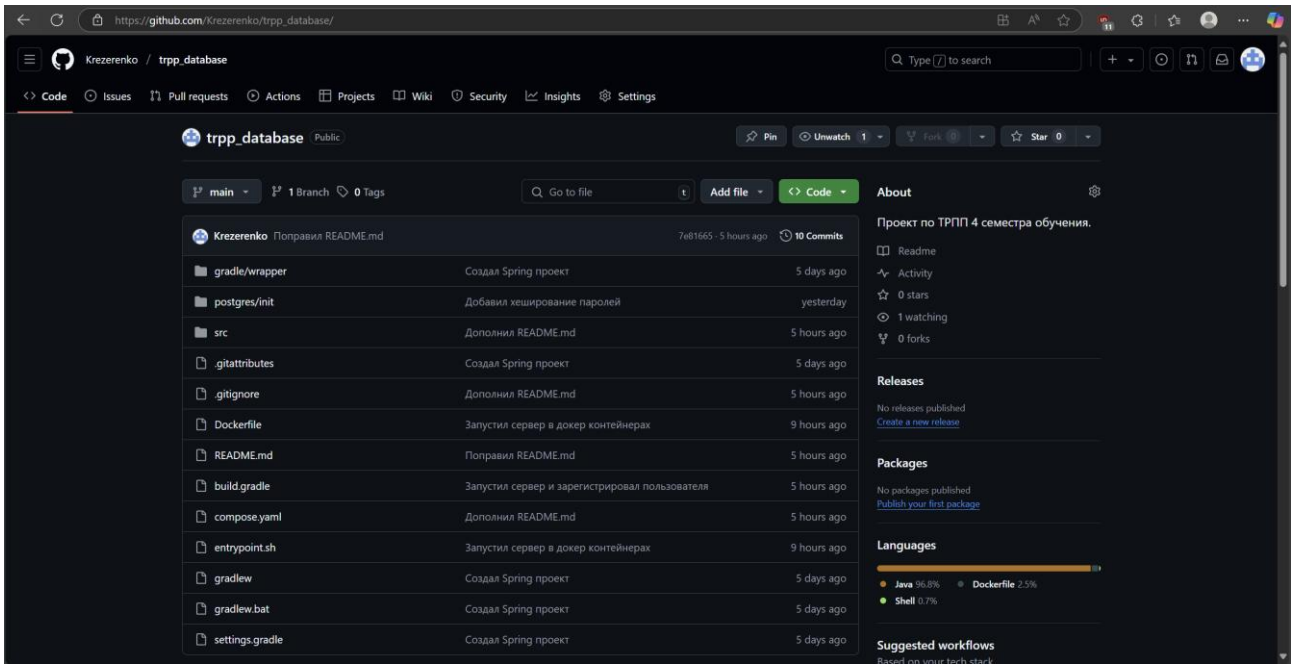

Рисунок 1 – GitHub репозиторий проекта

Доступ к репозиторию можно получить по адресу https://github.com/Krezerenko/trpp_database/.

На рисунке 2 представлена история коммитов репозитория в свернутом формате.

```
PS C:\Users\krezo\Education\SoftDev\Practics\trpp_database> git lg
* fc878fb (HEAD → main, origin/main, origin/HEAD) Дополнил README.md
* 1a006c8 Запустил сервер и зарегистрировал пользователя
* 2f44e0b Запустил сервер в докер контейнерах
* ef0edb6 Добавил хеширование паролей
* 9c94ae1 Доработал докер файлы
* 44c59c2 Добавил REST api классы для работы с таблицей пользователей
* 75832e3 Добавил файл docker compose
* 44f2995 Создал Spring проект
* c984983 Initial commit
```

Рисунок 2 – История коммитов проекта

На рисунках 3-5 представлена подробная история коммитов.

```
commit fc878fb702be4e6403a1ecb61070b33cc5334242 (HEAD → main, origin/main, origin/HEAD)
Author: Krezon <krezony@gmail.com>
Date:   Fri May 16 13:13:18 2025 +0300

    Дополнил README.md

    Добавил команды для сборки и запуска

commit 1a006c852b106e08a5103e550a73d96ff200ad67
Author: Krezon <krezony@gmail.com>
Date:   Fri May 16 12:33:03 2025 +0300

    Запустил сервер и зарегистрировал пользователя

    Ура ура ура ура побееда впереееед
    Добавил DTO для запросов, два исключения дополнил код существующих классов для регистрации

commit 2f44e0b025f648c0b93be8cb24ade8599c73d53f
Author: Krezon <krezony@gmail.com>
Date:   Fri May 16 09:19:18 2025 +0300

    Запустил сервер в докер контейнерах

    Для этого подправил весь код для докера
```

Рисунок 3 – История коммитов, часть 1

```
commit ef0edb6970a054e4d0dca03c6e72b66ae50a1e79
Author: Krezon <krezony@gmail.com>
Date:   Thu May 15 14:49:19 2025 +0300

    Добавил хеширование паролей

    Также добавил поле email

commit 9c94ae1dd598e171dd40f600874d745911043206
Author: Krezon <krezony@gmail.com>
Date:   Thu May 15 14:44:01 2025 +0300

    Доработал докер файлы

    Также создал файлы инициализации

commit 44c59c2edae47ee949baa80b1b732e868f8d0411
Author: Krezon <krezony@gmail.com>
Date:   Thu May 15 11:54:12 2025 +0300

    Добавил REST api классы для работы с таблицей пользователей

    Также была добавленая необходимая конфигурация
```

Рисунок 4 – История коммитов, часть 2

```
commit 75832e30f0c8ee17e5743419fb15317e30bffc43
Author: Krezon <krezony@gmail.com>
Date:   Thu May 15 04:58:27 2025 +0300

    Добавил файл docker compose

    Также начал писать код, изменил имеющийся Dockerfile. Для тестирования работы сервера был создан класс DemoControlle
r

commit 44f29952ca4de2961618087c1c8d58cf72553a08
Author: Krezon <krezony@gmail.com>
Date:   Sun May 11 20:16:29 2025 +0300

    Создал Spring проект

    Для создания использовался Spring Initializr
    Проект предполагается быть сервером

commit c984983cf1569206c11db23d66209c3bec001838
Author: Krezon <121864112+Krezerenko@users.noreply.github.com>
Date:   Sun May 11 19:27:49 2025 +0300

    Initial commit
```

Рисунок 5 – История коммитов, часть 3

2.2 Разработка проекта и система сборки

Для реализации проекта был выбран фреймворк Spring Boot, так как предоставляет обширные возможности для создания web-сервисов и баз данных благодаря большому количеству плагинов.

В качестве IDE для разработки был выбран IntelliJ IDEA, так как предоставляет интеграцию как с PostgreSQL, так и Spring Boot, а также всеми известными системами сборки.

В качестве системы сборки проекта был выбран Gradle, так как это один из наиболее часто используемых систем сборки для языка Java. В процессе реализации было создано несколько задач, которые необходимо делать для сборки проекта, а также задача для запуска REST API сервера и сервиса базы данных.

Используемые плагины представлены на рисунке 6.

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.4.5'
    id 'io.spring.dependency-management' version '1.1.7'
}
```

Рисунок 6 – Плагины, используемые в проекте

Среди них Java, Spring Framework и Spring Dependency Management – необходимые плагины для разработки на Spring.

Зависимости проекта представлены на рисунке 7.

```
dependencies { Add Starters...
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-security'
    implementation 'org.springframework.boot:spring-boot-starter-validation'
    annotationProcessor 'org.projectlombok:lombok'
    compileOnly 'org.projectlombok:lombok'
    runtimeOnly 'org.postgresql:postgresql'
}
```

Рисунок 7 – Зависимости проекта

Среди зависимостей проекта:

- Spring Boot Starter Web – модуль предоставляющий все основные возможности для web разработки;
- Spring Boot Starter Data JPA – модуль для взаимодействия с базами данных, включает в себя также Hibernate;
- Spring Boot Starter Security – модуль для произведения защищенных операций, включает в себя классы для шифрования сообщений, а также предоставляет поддержку аутентификации для доступа к серверу и др.;
- Spring Boot Starter Validation – модуль для проверки введенных данных на корректность, например email;
- Lombok – модуль для генерации boilerplate кода, такого как getter и setter методы для всех полей, конструктор для инициализации всех полей и др.;
- PostgreSQL – драйвер для работы с PostgreSQL.

Были созданы следующие задачи:

- wipeDatabase – для полной очистки папки с базой данных на локальном хостовом устройстве;
- dockerBuild – для сборки образа контейнера с сервером, обновляя находящийся на нем jar файл проекта;
- dockerStart – для выгрузки сервера и сервиса базы данных в контейнеры соответствующие контейнеры и запуска сервера;
- dockerStop – для завершения работы контейнеров и остановки работы сервера;

Все задачи проекта представлены на рисунке 8.

```
29  tasks.register("wipeDatabase", Delete) { Delete it ->
30      delete "postgres/data/pgdata"
31  }
32
33  tasks.register("dockerBuild", Exec) { Exec it ->
34      dependsOn tasks.named("dockerStop")
35      commandLine "docker", "build", "-t", "restapi", "."
36  }
37
38  tasks.register("dockerStart", Exec) { Exec it ->
39      dependsOn tasks.named("build")
40      commandLine "docker", "compose", "up", "-d"
41  }
42
43  tasks.register("dockerStop", Exec) { Exec it ->
44      commandLine "docker", "compose", "down"
45  }
46
47  tasks.named("build") { Task it ->
48      finalizedBy tasks.named("dockerBuild")
49  }
```

Рисунок 8 – Описание всех задач проекта

Структура проекта представлена на рисунке 9.

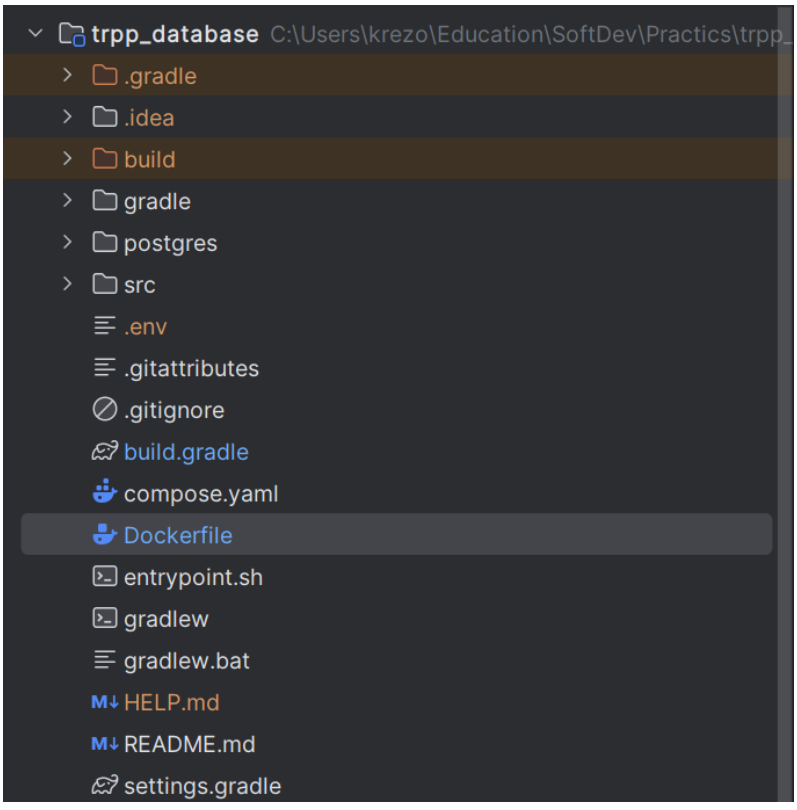

Рисунок 9 – Структура проекта

2.3 Развертывание приложения

Развертывание приложения осуществляется с помощью Docker контейнеров.

Для запуска проекта был написан docker compose файл compose.yaml. В этом файле было создано два сервиса: postgres и restapi, для запуска сервиса базы данных PostgreSQL и сервера REST API соответственно (Рисунки 10-11).

```yaml
services:
  postgres:
    image: postgres:17.4
    environment:
      POSTGRES_DB: ${POSTGRES_DB}
      POSTGRES_USER: ${POSTGRES_USER}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
      PGDATA: "/var/lib/postgresql/data/pgdata"
    volumes:
      - ./postgres/init:/docker-entrypoint-initdb.d
      - ./postgres/data:/var/lib/postgresql/data
    healthcheck:
      test: [ "CMD-SHELL", "pg_isready -U ${POSTGRES_USER} -d ${POSTGRES_DB}" ]
      interval: 10s
      retries: 5
      start_period: 30s
      timeout: 10s
    restart: unless-stopped
    ports:
      - "5433:5432"
```

Рисунок 10 – Сервис postgres в файле compose.yaml

```yaml
restapi:
  image: restapi
  build: .
  environment:
    SPRING_DATASOURCE_URL: "jdbc:postgresql://postgres:5432/${POSTGRES_DB}"
    SPRING_DATASOURCE_USERNAME: ${POSTGRES_USER}
    SPRING_DATASOURCE_PASSWORD: ${POSTGRES_PASSWORD}
    SPRING_SECURITY_USER_NAME: ${SPRING_USER}
    SPRING_SECURITY_USER_PASSWORD: ${SPRING_PASSWORD}
  depends_on:
    postgres:
      condition: service_healthy
  ports:
    - "8080:8080"
```

Рисунок 11 – Сервис restapi в файле compose.yaml

Для доступа к контейнерам были выбраны порты 5433 и 8080 соответственно.

В файле используются данные из файла .env, в котором задаются переменные среды в виде КЛЮЧ="значение".

Для запуска контейнера с сервисом PostgreSQL был выбран официальный образ postgres:17.4.

Для запуска контейнера с сервером создается образ restapi, основанный на eclipse-temurin:21, Dockerfile которого представлен на рисунке 12.

```
FROM eclipse-temurin:21
LABEL authors="krezo"

RUN mkdir /opt/app
COPY ./build/libs/trpp_database.jar /opt/app/app.jar
COPY ./entrypoint.sh /opt/app/entrypoint.sh

RUN chmod +x /opt/app/entrypoint.sh
CMD ["/opt/app/entrypoint.sh"]

```

Рисунок 12 – Содержание Dockerfile

Файл entrypoint.sh запускается при создании сервера (Рисунок 13).

```
#!/usr/bin/env sh

/opt/java/openjdk/bin/java -jar /opt/app/app.jar

```

Рисунок 13 – Содержание файла entrypoint.sh

ЗАКЛЮЧЕНИЕ

В ходе проделанной работы были реализован web-сервер для обработки REST API запросов по работе с базой данных PostgreSQL. В процессе реализации были использованы система контроля версий Git, система сборки Gradle и для развертывания был использован Docker.