МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение   
высшего образования «Самарский национальный исследовательский университет   
имени академика С.П. Королева»

(Самарский университет)

Институт информатики и кибернетики

Кафедра технической кибернетики

**ОТЧЕТ**

по Лабораторной работе 1

Тема: «Разработка базы данных»

Дисциплина: Технологии сетевого программирования

Выполнили студенты

группы 6304-010302D

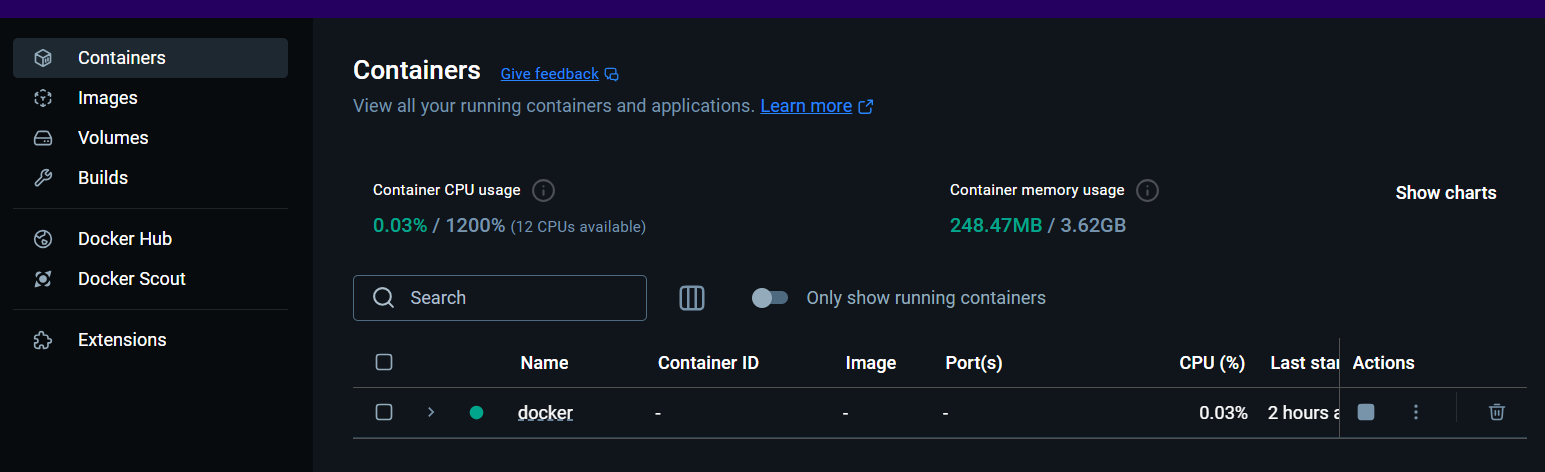
Поликарпова А.М.

Логанова Ю.В.

Самара 2025

В данной лабораторной работы мы развернули PostgreSQL в Docker и подключились к нашей БД через DBeaver.

Контейнер в Docker:



DockerFile:

FROM postgres:latest

ENV POSTGRES\_DB=businessplanner

ENV POSTGRES\_USER=admin

ENV POSTGRES\_PASSWORD=admin

EXPOSE 5432

Разработали модели в соответствии с тз. Миграции настроили при помощи Hibernate.

Файл конфигурации Hibernate:

<hibernate-configuration>

    <session-factory>

        <!-- Database connection settings -->

        <property name="hibernate.connection.driver\_class">org.postgresql.Driver</property>

        <property name="hibernate.connection.url">jdbc:postgresql://localhost:5432/businessplanner</property>

        <property name="hibernate.connection.username">admin</property>

        <property name="hibernate.connection.password">admin</property>

        <!-- SQL dialect -->

        <property name="hibernate.dialect">org.hibernate.dialect.PostgreSQLDialect</property>

        <!-- Automatic schema update -->

        <property name="hibernate.hbm2ddl.auto">update</property>

        <!-- Mapping entities -->

        <mapping class="com.businessplanner.models.User"/>

        <mapping class="com.businessplanner.models.Task"/>

        <mapping class="com.businessplanner.models.Tag"/>

    </session-factory>

</hibernate-configuration>

Структура пользователя: ID, email, name, password.

Функционал для работы с моделями:

1. Заполнение таблиц данными:

// Создаём тестовых пользователей

User user1 = new User();

user1.setName("John Doe");

user1.setEmail("john.doe@example.com");

//user1.setPassword(passwordEncoder.encode("password123"));

user1.setPassword("password123");

userRepository.save(user1);

User user2 = new User();

user2.setName("Jane Smith");

user2.setEmail("jane.smith@example.com");

//user2.setPassword(passwordEncoder.encode("password456"));

user2.setPassword("password456");

userRepository.save(user2);

// Создаём тестовые задачи

Task task1 = new Task();

task1.setTitle("Task 1");

task1.setDescription("Description for Task 1");

//task1.setDueDate(LocalDate.of(2023, 12, 1));

//task1.setStatus("IN\_PROGRESS");

task1.setCreator(user1);

taskRepository.save(task1);

Task task2 = new Task();

task2.setTitle("Task 2");

task2.setDescription("Description for Task 2");

//task2.setDueDate(LocalDate.of(2023, 12, 15));

//task2.setStatus("TODO");

task2.setCreator(user2);

taskRepository.save(task2);

Task task3 = new Task();

task3.setTitle("Task 3");

task3.setDescription("Description for Task 3");

task3.setCreator(user1);

taskRepository.save(task3);

Task task4 = new Task();

task4.setTitle("Task 4");

task4.setDescription("Description for Task 4");

task4.setCreator(user1);

taskRepository.save(task4);

// Создаём тестовые теги

Tag tag1 = new Tag();

tag1.setName("Urgent");

tagRepository.save(tag1);

Tag tag2 = new Tag();

tag2.setName("Important");

tagRepository.save(tag2);

Tag tag3 = new Tag();

tag3.setName("Low Priority");

tagRepository.save(tag3);

1. Создаем связи между задачами и тэгами:

TaskTag taskTag1 = new TaskTag();

taskTag1.setTask(task1);

taskTag1.setTag(tag1);

taskTagRepository.save(taskTag1);

TaskTag taskTag2 = new TaskTag();

taskTag2.setTask(task1);

taskTag2.setTag(tag2);

taskTagRepository.save(taskTag2);

TaskTag taskTag3 = new TaskTag();

taskTag3.setTask(task2);

taskTag3.setTag(tag3);

taskTagRepository.save(taskTag3);

TaskTag taskTag4 = new TaskTag();

taskTag4.setTask(task3);

taskTag4.setTag(tag3);

taskTagRepository.save(taskTag4);

TaskTag taskTag5 = new TaskTag();

taskTag5.setTask(task4);

taskTag5.setTag(tag1);

taskTagRepository.save(taskTag5);

1. Тест вывода данных:

Long userId = 1L;

List<Task> tasksForUser1 = taskService.getTasksByCreatorId(userId);

System.out.println("Задачи для пользователя John Doe:");

tasksForUser1.forEach(task -> System.out.println(

"ID: " + task.getId() +

", Название: " + task.getTitle() +

", Описание: " + task.getDescription()

));

userId = 2L;

List<Task> tasksForUser2 = taskService.getTasksByCreatorId(userId);

System.out.println("Задачи для пользователя Jane Smith:");

tasksForUser2.forEach(task -> System.out.println(

"ID: " + task.getId() +

", Название: " + task.getTitle() +

", Описание: " + task.getDescription()

));

System.out.println("end of 1st test");

// Получаем задачи для пользователя с email "jane.smith@example.com" и тегом с ID = 3

            String email = "jane.smith@example.com";

            Long tagId = 3L;

            System.out.println("vot tut AAAAAAAAAAAA");

            List<Task> tasks = taskService.getTasksByUserEmailAndTagId(email, tagId);

            System.out.println("Задачи для пользователя " + email + " с тегом (ID: " + tagId + "):");

            tasks.forEach(task -> System.out.println(

                "ID: " + task.getId() +

                ", Название: " + task.getTitle() +

                ", Описание: " + task.getDescription()

            ));

            // Получаем задачи для пользователя с email "john.doe@example.com" и тегом с ID = 1

            String email\_2 = "john.doe@example.com";

            Long tagId\_2 = 1L;

            List<Task> tasks\_2 = taskService.getTasksByUserEmailAndTagId(email\_2, tagId\_2);

            System.out.println("Задачи для пользователя " + email\_2 + " с тегом (ID: " + tagId\_2 + "):");

            tasks\_2.forEach(task -> System.out.println(

                "ID: " + task.getId() +

                ", Название: " + task.getTitle() +

                ", Описание: " + task.getDescription()

            ));

            System.out.println("end of 2nd test AAAAAAAAAAAA");

1. Тест на удаление данных из таблиц:

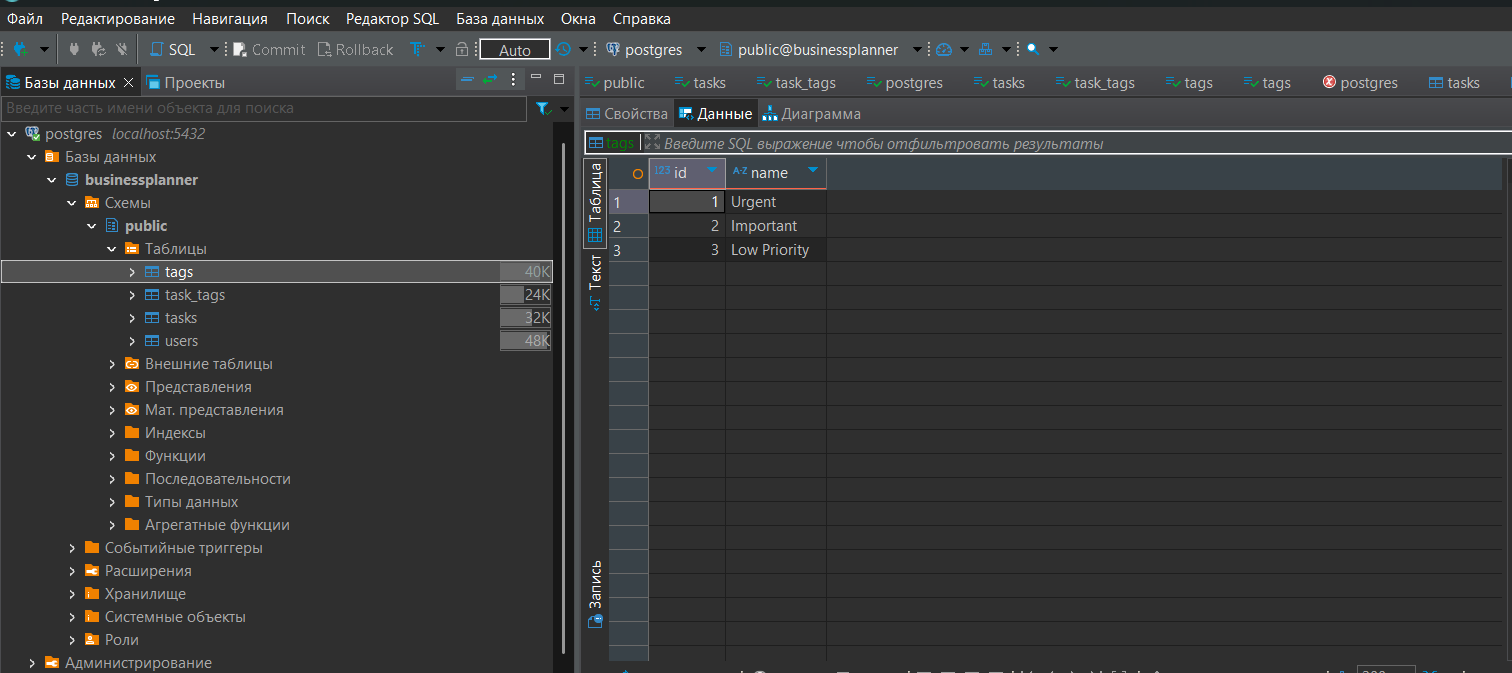
// Удаляем пользователя по email

String emailToDelete = "donkey@example.com";

userService.deleteUserByEmail(emailToDelete);

System.out.println("Пользователь с email " + emailToDelete + " удалён.");.

Подключились к DBeaver:



Дополнительные скриншоты:

Выполнение тестов

