Министерство науки и высшего образования Российской Федерации

ФГАОУ ВО «Уральский федеральный университет имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий-РТФ

ОТЧЕТ

По лабораторной работе №1

«Основы работы с Docker и PostgreSQL»

По дисциплине «Разработка приложений»

Выполнил: Яшин М.С.

Группа: РИМ-150950

Проверил преподаватель:
Кузьмин Д.

Екатеринбург

2025

Цель работы: Освоить фундаментальные концепции и базовые операции Docker: создание образов, запуск контейнеров, управление ими, работа с сетями и томами. На практике закрепить навыки, запустив изолированную базу данных PostgreSQL и подключившись к ней извне.

Задачи:

1. Установить и проверить работу Docker.
2. Изучить базовые команды Docker.
3. Запустить контейнер с PostgreSQL в изолированном режиме.
4. Запустить контейнер с pgAdmin и подключить его к контейнеру с БД через сеть Docker.
5. Подключиться к БД из pgAdmin, создать схему и выполнить запросы.
6. Обеспечить сохранность данных БД с помощью томов Docker.

Ход работы:

1. Установка и проверка Docker

Был скачен и установлен Docker Desktop.

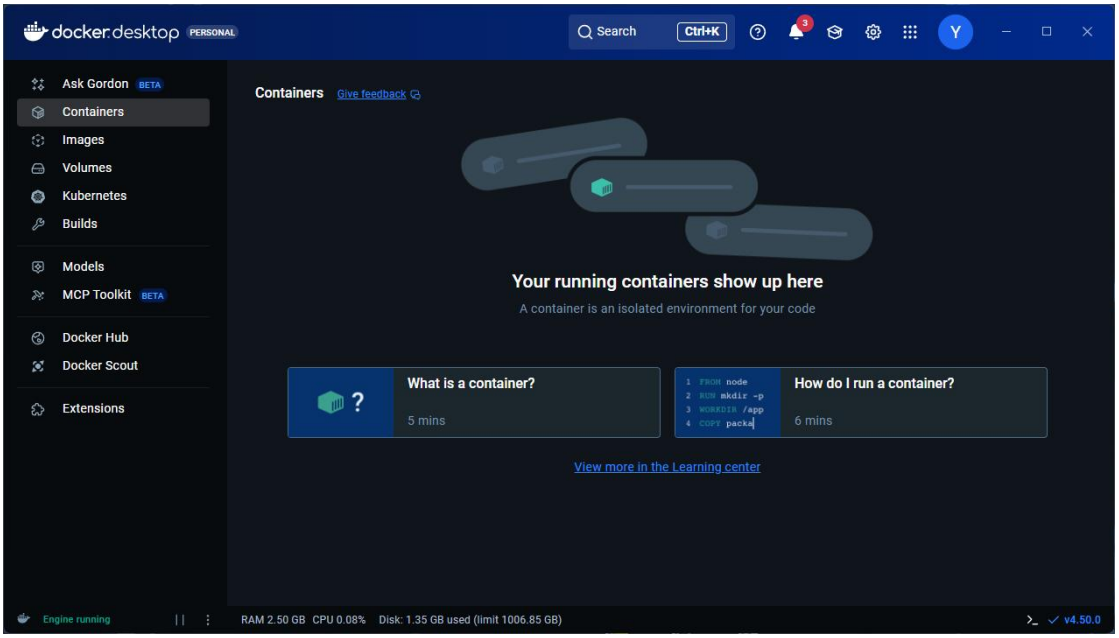

Рисунок 1 – Docker Desktop

Дальше, через WebStorm вписав в терминале команды «docker version», «docker compose version» и «docker hello-world», убедились в правильной работе Docker.

```
PS C:\Users\misha\WebstormProjects\application-development-2025-urfu> cd lr1
PS C:\Users\misha\WebstormProjects\application-development-2025-urfu\lr1> docker version
Client:
 Version:           28.5.1
 API version:       1.51
 Go version:        go1.24.8
 Git commit:        e180ab8
 Built:             Wed Oct  8 12:19:16 2025
 OS/Arch:           windows/amd64
 Context:           desktop-linux

Server: Docker Desktop 4.50.0 (209931)
 Engine:
  Version:          28.5.1
  API version:      1.51 (minimum version 1.24)
  Go version:       go1.24.8
  Git commit:       f8215cc
  Built:            Wed Oct  8 12:17:24 2025
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.7.27
  GitCommit:        05044ec0a9a75232cad458027ca83437aae3f4da
 runc:
  Version:          1.2.5
  GitCommit:        v1.2.5-0-g59923ef
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
PS C:\Users\misha\WebstormProjects\application-development-2025-urfu\lr1> docker compose version
Docker Compose version v2.40.3-desktop.1
```

Рисунок 2 – Команды «docker version» и «docker compose version»

```
PS C:\Users\misha\WebstormProjects\application-development-2025-urfu\lr1> docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
17eec7bbc9d7: Pull complete
Digest: sha256:56433a6be3fda188089fb548eae3d91df3ed0d6589f7c2656121b911198df065
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Рисунок 3 – Команда «docker hello-world»

2) Базовые команды Docker. Работа с образами и контейнерами

Просмотр информации.

```
PS C:\Users\misha\WebstormProjects\application-development-2025-urfu\lr1> docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world   latest    56433a6be3fd   3 months ago   20.3kB
PS C:\Users\misha\WebstormProjects\application-development-2025-urfu\lr1> docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS C:\Users\misha\WebstormProjects\application-development-2025-urfu\lr1> docker ps -a
CONTAINER ID   IMAGE         COMMAND    CREATED              STATUS                          PORTS     NAMES
ee56f1776da3   hello-world   "/hello"   About a minute ago   Exited (0) About a minute ago             funny_satoshi
PS C:\Users\misha\WebstormProjects\application-development-2025-urfu\lr1>
```

Рисунок 4 – «docker images», «docker ps» и «docker ps -a»

Запуск простого контейнера, на примере Nginx.

```
PS C:\Users\misha\WebstormProjects\application-development-2025-urfu\lr1> docker run -d -p 8080:80 --name dz1_webserver nging:alpine
Unable to find image 'nging:alpine' locally
docker: Error response from daemon: pull access denied for nging, repository does not exist or may require 'docker login'

Run 'docker run --help' for more information
PS C:\Users\misha\WebstormProjects\application-development-2025-urfu\lr1> docker run -d -p 8080:80 --name dz1_webserver nginx:alpine
Unable to find image 'nginx:alpine' locally
alpine: Pulling from library/nginx
8f6a6833e95d: Pull complete
ff8a36d5502a: Pull complete
2d35ebdb57d9: Pull complete
194fa24e147d: Pull complete
bdabb0d44271: Pull complete
3eaba6cd10a3: Pull complete
df413d6ebdc8: Pull complete
d9a55dab5954: Pull complete
Digest: sha256:b3c656d55d7ad751196f21b7fd2e8d4da9cb430e32f646adcf92441b72f82b14
Status: Downloaded newer image for nginx:alpine
51abc35572097f6ab2355856deaa849c04f45e30c1e6e6daa1300b3a7923d9d1
PS C:\Users\misha\WebstormProjects\application-development-2025-urfu\lr1>
```

Рисунок 5 – Создание контейнера

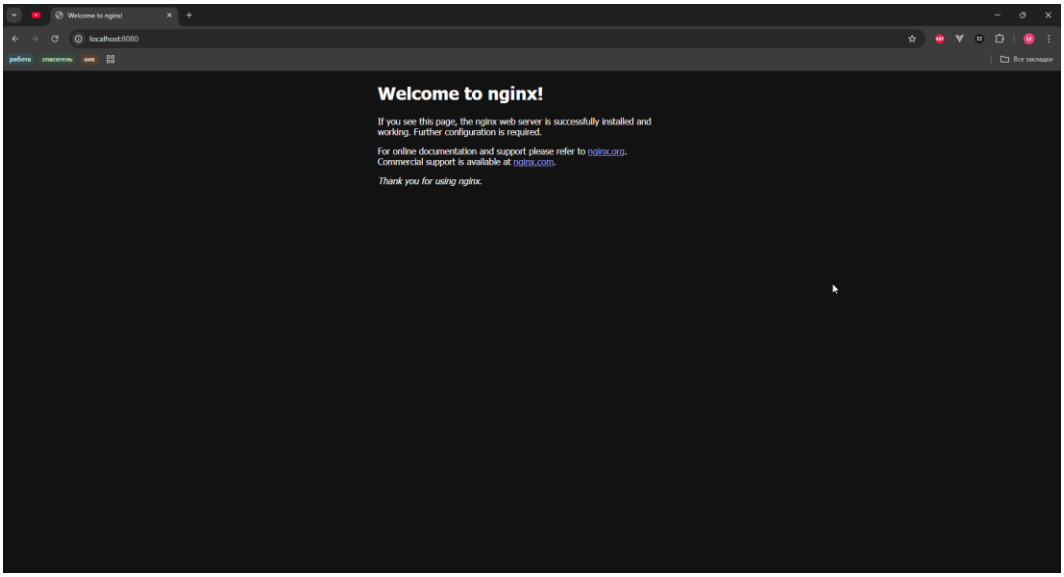

Рисунок 6 – Переход на «localhost:8080»

3) Запуск PostgreSQL в контейнере

Запустим контейнер с PostgreSQL и подключимся к БД из контейнера (Рисунок 7). Создадим таблицу с помощью команды «CREATE TABLE users (id SERIAL PRIMARY KEY, name VARCHAR(50));», вставим данные с помощью «INSERT INTO users (name) VALUES (‘Alice’), (‘Bob’);» и проверим ее с помощью команды «SELECT * FROM users;».

Рисунок 7 – Запуск контейнера с PostgreSQL

Подключение к БД.

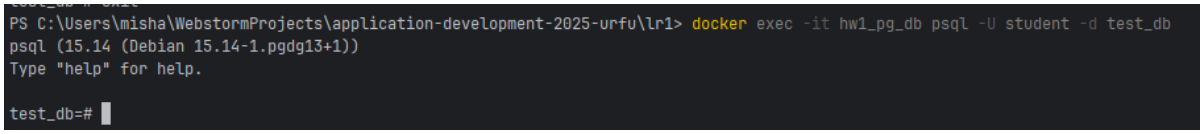

Рисунок 8 – Подключение к test_db

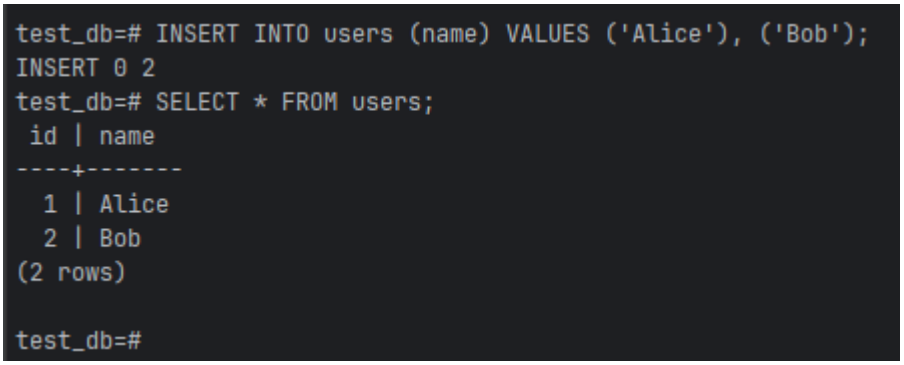

Рисунок 9 – Проверка созданной таблицы

4) Подключение к БД через pgAdmin из второго контейнера

Создадим сеть Docker и проверим, что она появилась в списке сетей.

```
PS C:\Users\misha\WebstormProjects\application-development-2025-urfu\lr1> docker network create hw1_network
c055621900e423b84f6827809d067df25a68eaed7d3440f7c46b1bd4396b8eea
PS C:\Users\misha\WebstormProjects\application-development-2025-urfu\lr1> docker network create ls
c15becef64a0af325268577033805bf9be59f4d9f14b22d3b024bb07753f862d
PS C:\Users\misha\WebstormProjects\application-development-2025-urfu\lr1> docker network ls
NETWORK ID     NAME          DRIVER    SCOPE
29b09c62e9a5   bridge        bridge    local
7580db3bc455   host          host      local
c055621900e4   hw1_network   bridge    local
c15becef64a0   ls            bridge    local
7572b8b77a71   none          null      local
```

Рисунок 10 – Создание сети

Подключим контейнер с PostgreSQL к сети и запустим pgAdmin в той же сети.

```
PS C:\Users\misha\WebstormProjects\application-development-2025-urfu\lr1> docker run -d --name hw1_pg
62a94d6db9d2615195e990835653fd9b2877537532bd684d4c54ccbe134aa5dd
```

Рисунок 11 – Подключение к сети и запуск pgAdmin

Откроем «localhost:8080» и попадем в pgAdmin и подключим сервер с нашей созданной базой данных, далее проверим, что все создано.

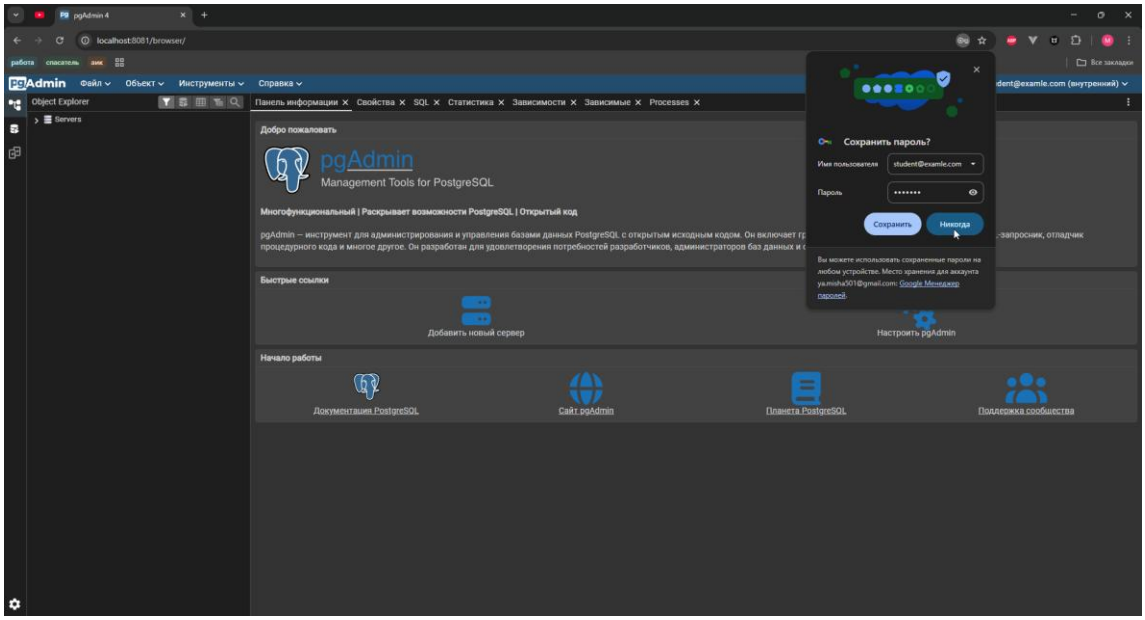

Рисунок 12 –pgAdmin в браузере

5) Сохранение данных с помощью Томов (Volumes)

Остановим текущий контейнер с БД, создадим том и запустим новый контейнер.

```
PS C:\Users\misha\WebstormProjects\application-development-2025-urfu\lr1> docker stop hw1_pg_db
hw1_pg_db
PS C:\Users\misha\WebstormProjects\application-development-2025-urfu\lr1> docker rm hw_postgres_db
Error response from daemon: No such container: hw_postgres_db
PS C:\Users\misha\WebstormProjects\application-development-2025-urfu\lr1> docker rm hw1_postgres_db
Error response from daemon: No such container: hw1_postgres_db
PS C:\Users\misha\WebstormProjects\application-development-2025-urfu\lr1> docker rm hw1_pg_db
hw1_pg_db
PS C:\Users\misha\WebstormProjects\application-development-2025-urfu\lr1> docker volume create postgres_Data
postgres_Data
PS C:\Users\misha\WebstormProjects\application-development-2025-urfu\lr1> docker volume ls
DRIVER    VOLUME NAME
local     8f4f36020a7ffb9c05c66d6d66c5129210c871bc9d2515615bd18be7dd348d85
local     6757d767a7f9de9d0b8ccacf9cd2ffdcf981485629b7b4f0072afb5bd8fbf26d
local     postgres_Data
```

Рисунок 13 – Создание тома

```
PS C:\Users\misha\WebstormProjects\application-development-2025-urfu\lr1> docker run -d --name hw1_pg_db_persistent -e
w1_network postgres:15
4a782d7bc9815866d32c768c58308113705b6439aeed70537fc52b9ca3df0faa
```

Рисунок 14 – Монтирование тоа

Через pgAdmin подключимся к нашей БД, создадим таблицу и введем данные.

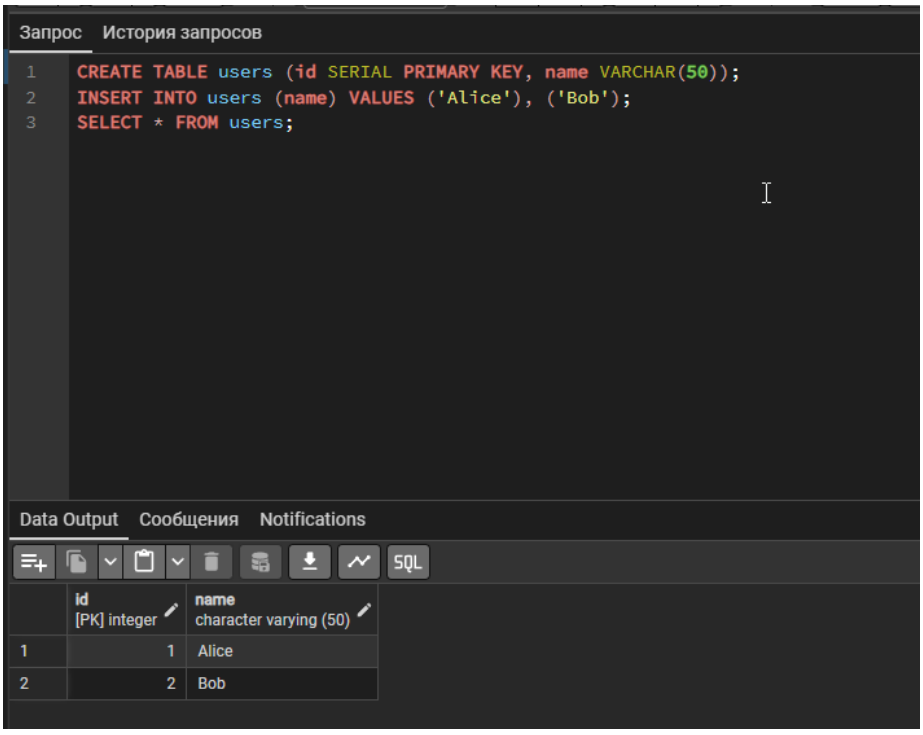

Рисунок 15 – Создание таблицы через pgAdmin

Далее отключим контейнеры, включим заново и проверим сохранность данных.

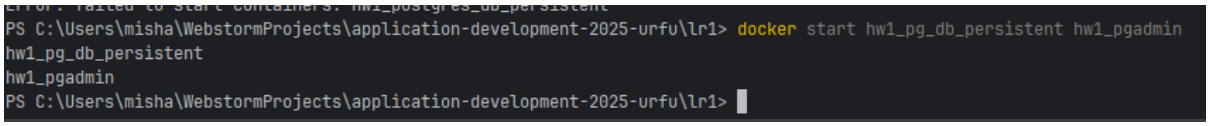

Рисунок 16 – Запуск контейнеров

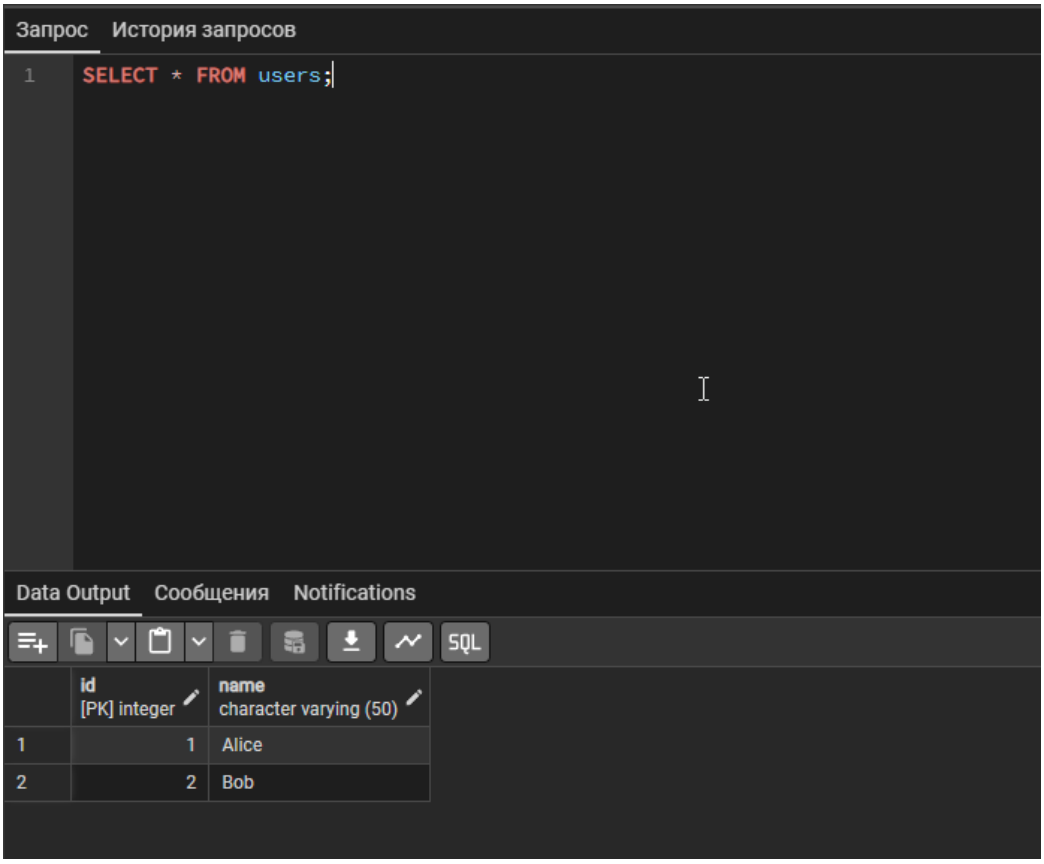

Рисунок 17 – Проверка данных в pgAdmin

Данные сохранились, всё работает.

6) Перенос конфигурации контейнеров в docker-compose.yaml

Создадим файл docker-compose.yaml (файл приложен в репозитории на github). «Поднимем» его командой «docker-compose up -d». Проверим работоспособность, перейдя на localhost:5050 (Рисунок 16).

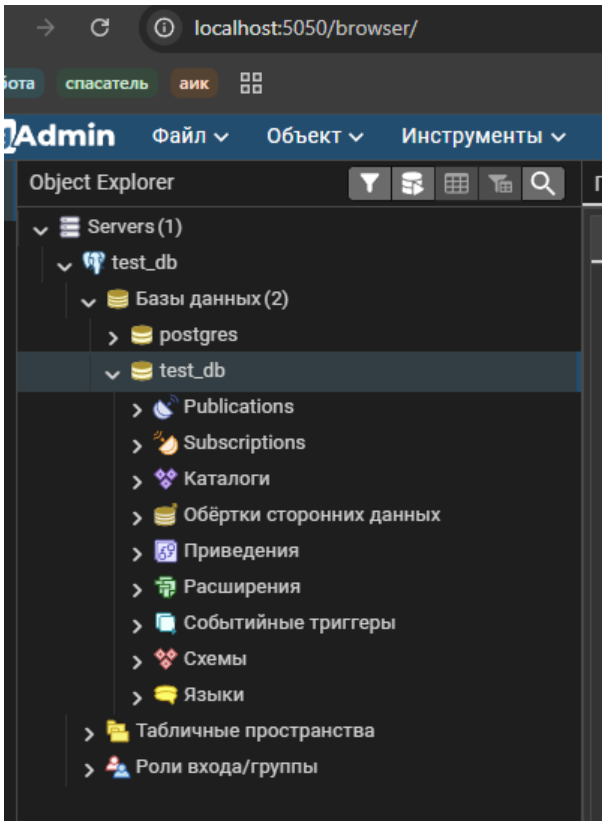

Рисунок 18 – Подключение БД к pgAdmin

Создадим там таблицу с помощью команды «CREATE TABLE users (id SERIAL PRIMARY KEY, name VARCHAR(50));», вставим данные с помощью «INSERT INTO users (name) VALUES (‘Alice’), (‘Bob’);» и проверим ее с помощью команды «SELECT * FROM users;». Затем отключим с помощью команды в терминале «docker-compose down» и «поднимем» заново (Рисунок 17), данные сохранились.

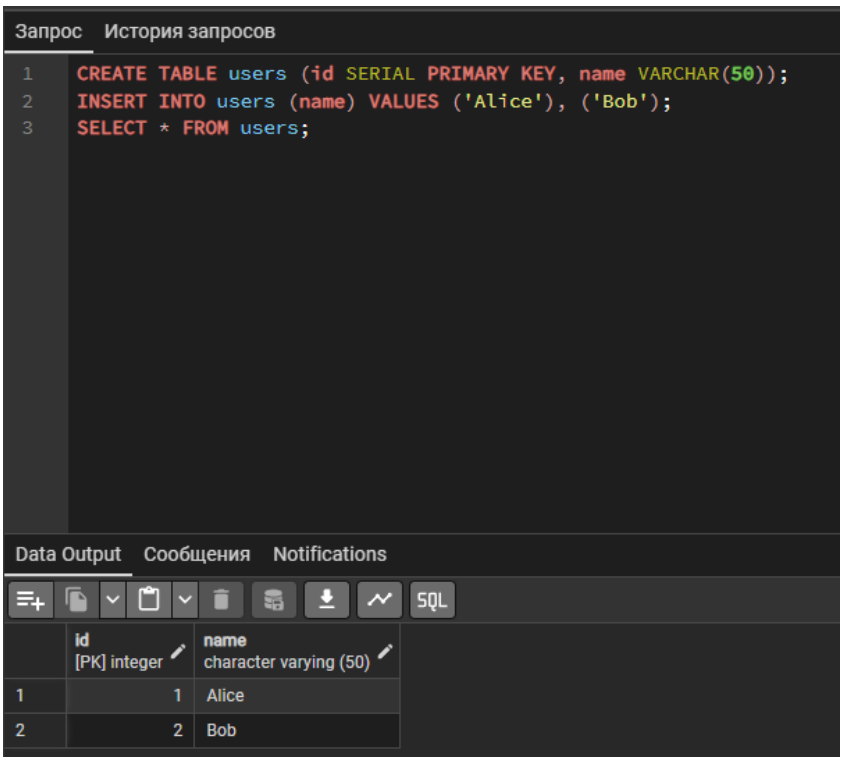

Рисунок 19 – Проверка данных после второго «поднятия»

Вопросы:

1. Что такое docker?

Docker – инструмент, позволяющий упаковывать приложения и всё, что им нужно для работы (код, библиотеки, настройки), в контейнеры. Например, применимо к нашей работе, мы можем запустить веб-сервер NGINX, СУБД PostgreSQL.

2. Для чего нужны тома и сети docker?

Том нужен для сохранения данных. Если подключить том, данные сохранятся даже после перезапуска или удаления контейнера. Сети нужны для обмена данными между контейнерами. По умолчанию контейнеры не видят друг друга, но если подключить их к одной созданной сети, они смогут общаться по имени

3. Как подключится к контейнеру и выполнить в нём команды?

Мы можем подключиться к контейнеру через терминал, используя команду «docker exec -it ***имя_контейнера команда***». В нашей работе мы использовали docker exec -it dz1_postgres psql -U student -d test_db для подключения к базе данных.

4. Для чего нужен pgAdmin?

PgAdmin – веб-интерфейс для управления PostgreSQL, он позволяет подключаться к базе данных через браузер, просматривать таблицы, схемы, индексы, выполнять SQL-запросы и создавать/удалять пользователей, базы, таблицы – без командной строки.

Вывод:

В ходе выполнения лабораторной работы были освоены основные концепции и инструменты Docker, необходимые для развертывания и управления приложениями в контейнерах.

Удалось успешно запустить контейнеры с веб-сервером NGINX и СУБД PostgreSQL, настроить тома (volumes) для сохранения данных базы данных даже после удаления контейнера, создать пользовательскую Docker-сеть, обеспечивающую взаимодействие между контейнерами по имени, развернуть веб-интерфейс pgAdmin и подключить его к PostgreSQL-контейнеру для удобного управления базой данных, автоматизировать запуск всей инфраструктуры с помощью файла docker-compose.yml.

В процессе работы были преодолены ошибки, связанные с отсутствием опыта работы с Docker, например, конфликты портов, некорректный синтаксис yaml, проблемы с аутентификацией и кавычками в SQL. Это позволило глубже понять особенности работы с Docker и важность внимательности при настройке конфигураций.

В результате создана полностью изолированная, воспроизводимая и переносимая среда разработки, состоящая из базы данных и инструмента для её администрирования, что подтверждает практическую применимость Docker в реальных проектах.

Ссылка на репозиторий:

https://github.com/yam501/application-development-2025-urfu