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: Гуламов А.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

С официального сайта был установлен Docker Desktop (Рисунок 1).

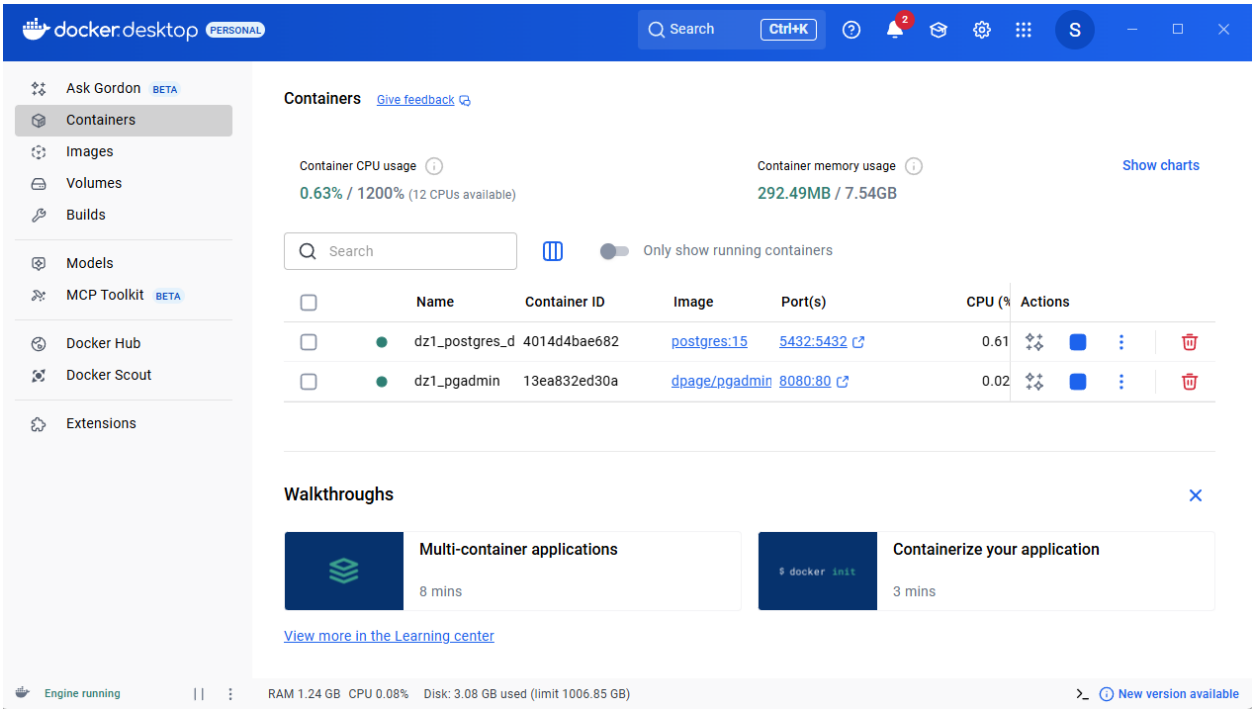

Рисунок 1 – Docker Desktop

Далее, запустив Visual Studio Code и вписав в терминале команды «docker version», «docker compose version» и «docker hello-world», убедились в правильной работе Docker (Рисунки 2-3).

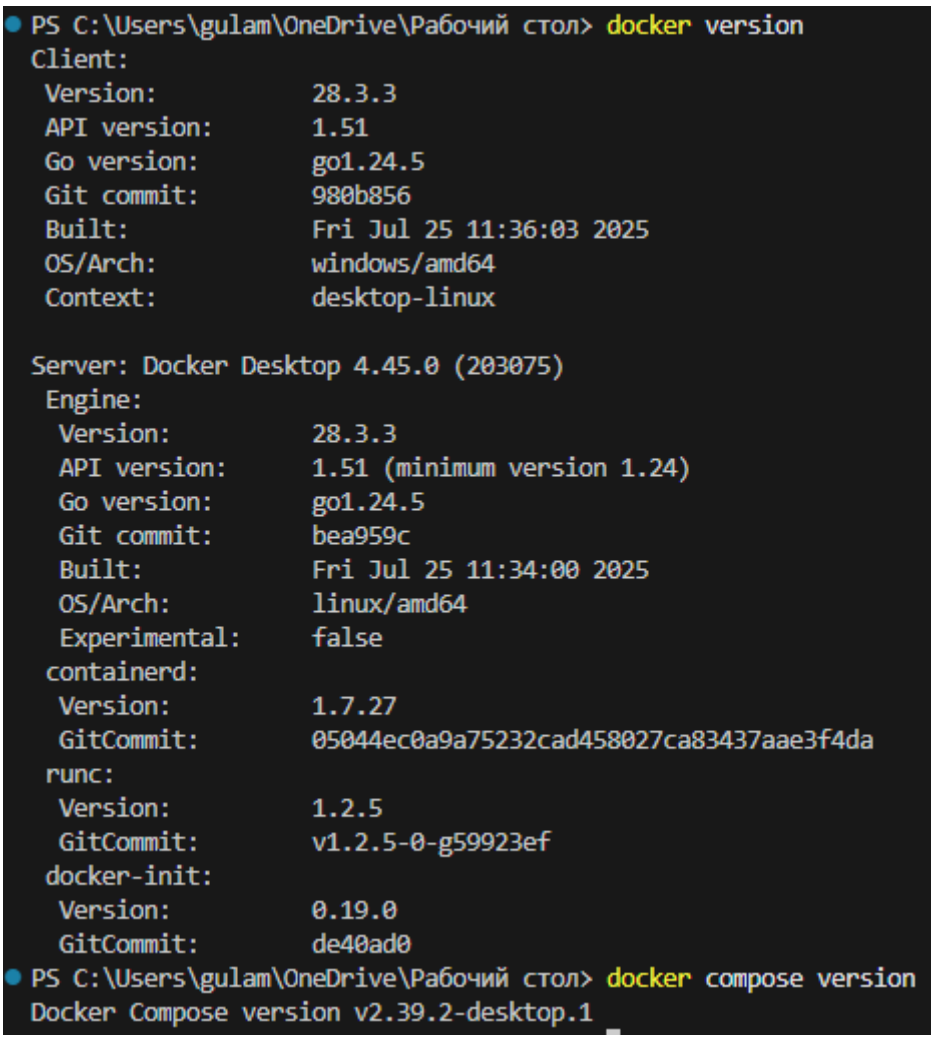

Рисунок 2 – Команды «docker version» и «docker compose version»

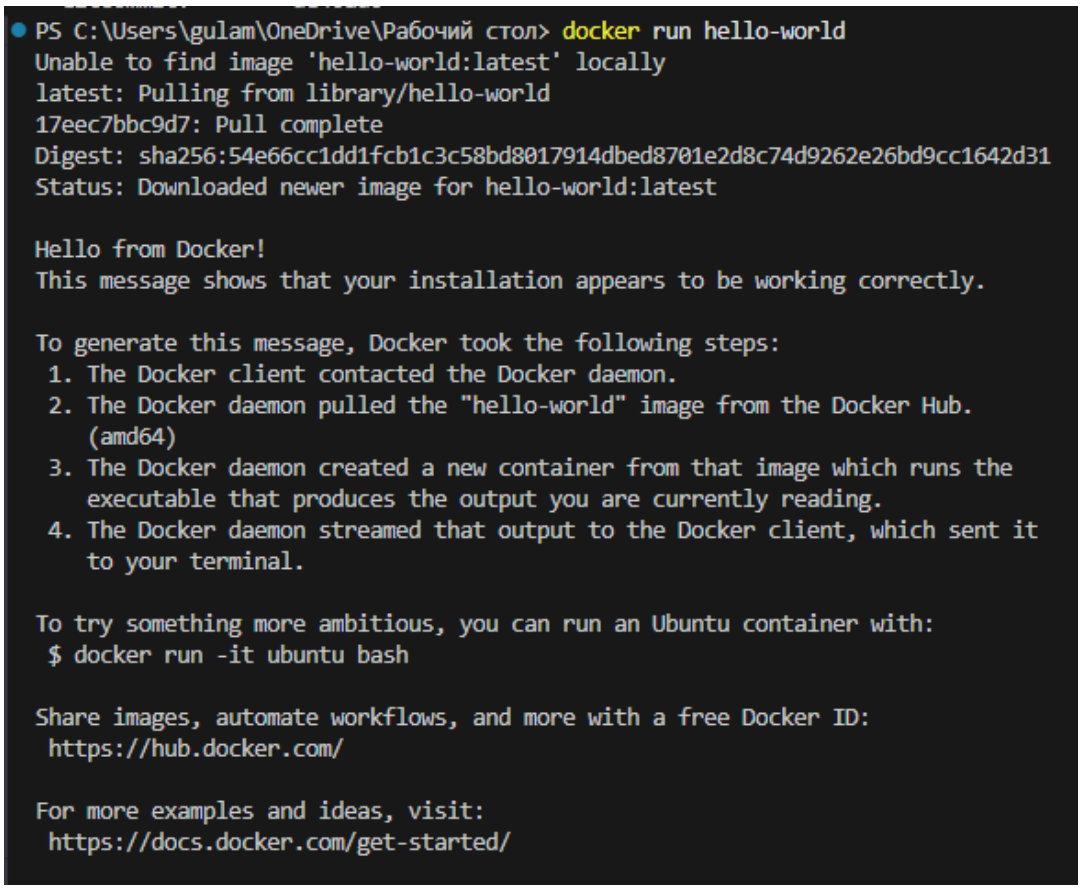

Рисунок 3 – Команда «docker hello-world»

2) Базовые команды Docker. Работа с образами и контейнерами

Воспользуемся базовыми командами для просмотра информации (Рисунок 4).

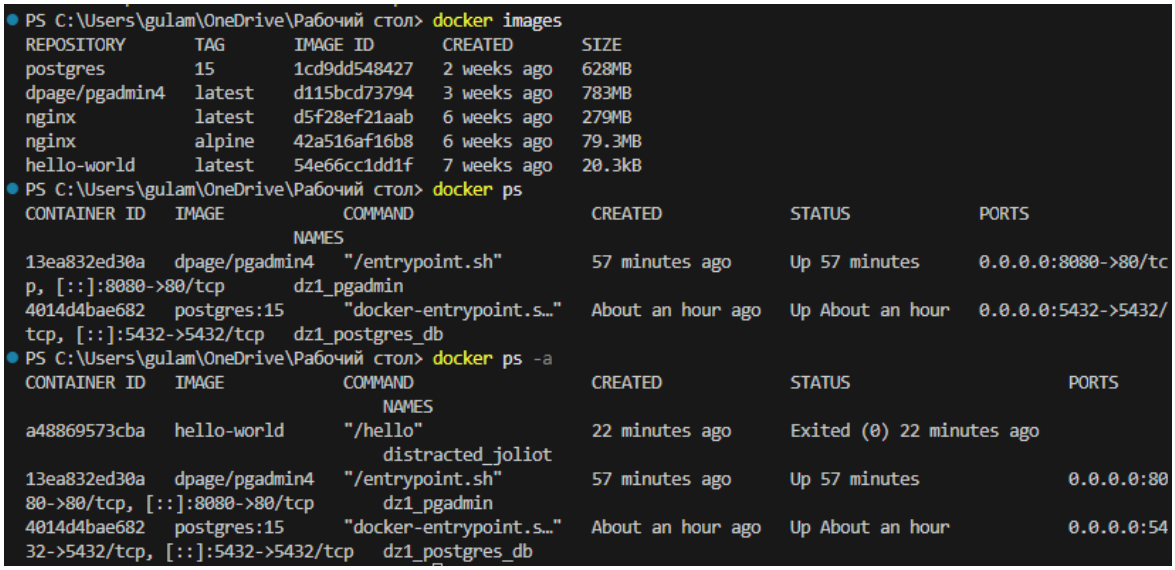

Рисунок 4 – Ввод команд «docker images», «docker ps» и «docker ps -a»

Создадим простейший контейнер и проверим его работу, набрав в браузере «localhost:8080» (Рисунки 5-6).

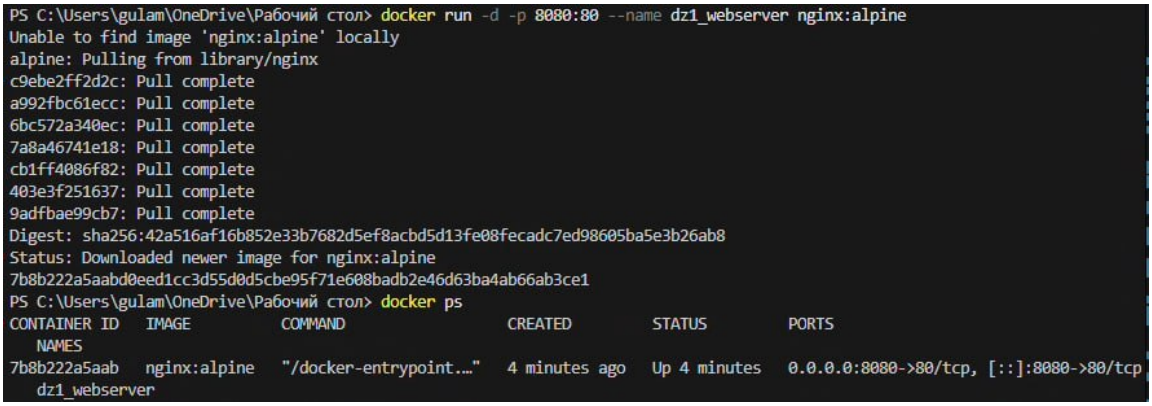

Рисунок 5 – Создание контейнера

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 8).

```
PS C:\Users\gulam\OneDrive\Рабочий стол> docker run -d --name dz1_postgres_db -e POSTGRES_USE
R=student -e POSTGRES_PASSWORD=homework -e POSTGRES_DB=test_db -p 5432:5432 postgres:15
4014d4bae682eb8df66c525162e716d52275ed32ee6dcd023cd242145c933c45
PS C:\Users\gulam\OneDrive\Рабочий стол> docker exec -it dz1_postgres_db psql -U student -d t
est_db
psql (15.14 (Debian 15.14-1.pgdg13+1))
Type "help" for help.
```

Рисунок 7 – Запуск контейнера с PostgreSQL

```
test_db=# SELECT * FROM users;
 id | name
----+-------
  1 | Alice
  2 | Bob
(2 rows)

test_db=# \q
```

Рисунок 8 – Проверка созданной таблицы и выход из консоли

4) Подключение к БД через pgAdmin из второго контейнера

Создадим сеть Docker и проверим, что она появилась в списке сетей (Рисунок 9, случайно была создана сеть «ls»)

```
PS C:\Users\gulam\OneDrive\Рабочий стол> docker network create dz1_network
2d8c87c10cfa2a78239645cc77abc87b54226d6a718300bf0418bafc30436e75
PS C:\Users\gulam\OneDrive\Рабочий стол> docker network create ls
4807c4c451dbc93bcbeff04a024b10fd03633ac21c5d3e114faac78d41e5c6c1
PS C:\Users\gulam\OneDrive\Рабочий стол> docker network ls
NETWORK ID     NAME                 DRIVER    SCOPE
8f133fe0c016   bridge               bridge    local
360ff32dd156   dockerlab_my_network bridge    local
2d8c87c10cfa   dz1_network          bridge    local
a3d5d46a333a   host                 host      local
25d6b2f16f9c   lab1_default         bridge    local
4807c4c451db   ls                   bridge    local
329cb3126fee   none                 null      local
```

Рисунок 9 – Создание сети

Подключим контейнер с PostgreSQL к сети и запустим pgAdmin в той же сети (Рисунок 10).

```
PS C:\Users\gulam\OneDrive\Рабочий стол> docker network connect dz1_network dz1_postgres_db
PS C:\Users\gulam\OneDrive\Рабочий стол> docker run -d --name dz1_pgadmin -e PGADMIN_DEFAULT_
EMAIL=student@example.com -e PGADMIN_DEFAULT_PASSWORD=student -p 8080:80 --network dz1_networ
k dpage/pgadmin4
c925f7231090158b4cbcf35eca1b1c1c8c638c8b6a21fd9f5201dc2c8c766249
```

Рисунок 10 – 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 11).

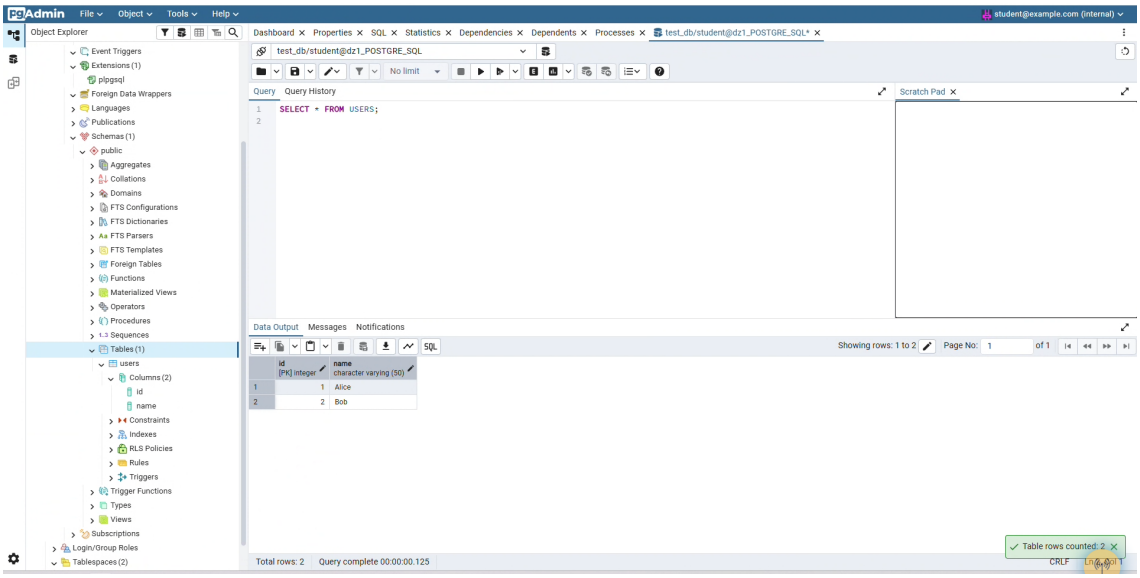

Рисунок 11 – Окно с pgAdmin

5) Сохранение данных с помощью Томов (Volumes)

Остановим текущий контейнер с БД, создадим том и запустим новый контейнер (Рисунок 12).

```
PS C:\Users\gulam\OneDrive\Рабочий стол\practicum> docker stop dz1_postgres_db
dz1_postgres_db
PS C:\Users\gulam\OneDrive\Рабочий стол\practicum> docker rm  dz1_postgres_db
dz1_postgres_db
PS C:\Users\gulam\OneDrive\Рабочий стол\practicum> docker volume create postgres_data
postgres_data
PS C:\Users\gulam\OneDrive\Рабочий стол\practicum> docker volume ls
DRIVER    VOLUME NAME
local     8dbb85c431a37c1f8eb78a2cbd48581b0049f973324aefe712789e495702c625
local     088c1c0691c2b578920894948a6ea97329010db331638cddad25a54138eba9b8
local     207de1ff6b07c87e31dd463b385c6d563d8bb1907f3b4f004638e3c41029e085
local     40595d1dfd79e97fab96edd4a359fccd409d58d45709ad1aac1f49fc7e181956
local     b9f1ccecb0df34700a6f3cfeef37ab2f68046000c15c0e60f7f3e22cab365727
local     bfc83171f379f2f850f27ebc95feae53cf8db6ce00e847567e64ef3ae52e0c78
local     c83e9a9621198d7683bd3eb43cbb58b6b8375e6625d2a6bd7d1d05dc0c63769c
local     d47e6e7e2e249081b41682a506d5c07ee62d186d4340200961305a07a5eda09c
local     dockerlab_postgres_data
local     postgres_data
PS C:\Users\gulam\OneDrive\Рабочий стол\practicum> docker run -d --name dz1_postgres_d
b_persistent -e POSTGRES_USER=student -e POSTGRES_PASSWORD=homework -e POSTGRES_DB=tes
t_db -p 5432:5432 -v postgres_data:/var/lib/postgresql/data --network dz1_network post
gres:15
dac25b3ee035d36f7c7bf612d6836e86cc98f5f54d69f46db5d412c3a45e7051
```

Рисунок 12 – Создание тома

Через pgAdmin подключимся к нашей БД, создадим таблицу и введем данные (Рисунок 13).

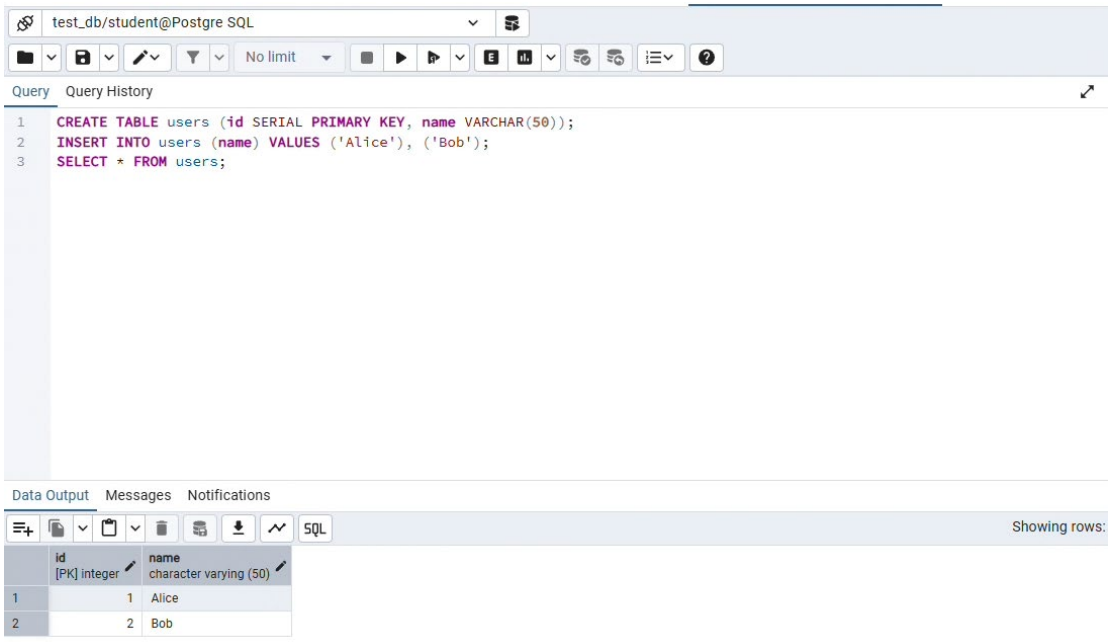

Рисунок 13 – Создание таблицы через pgAdmin

Далее отключим контейнеры, включим заново и проверим сохранность данных (Рисунки 14-15).

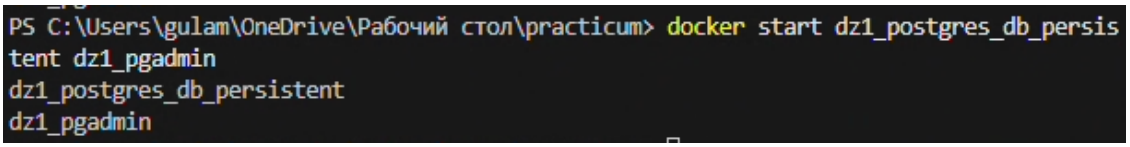

Рисунок 14 – Запуск контейнеров

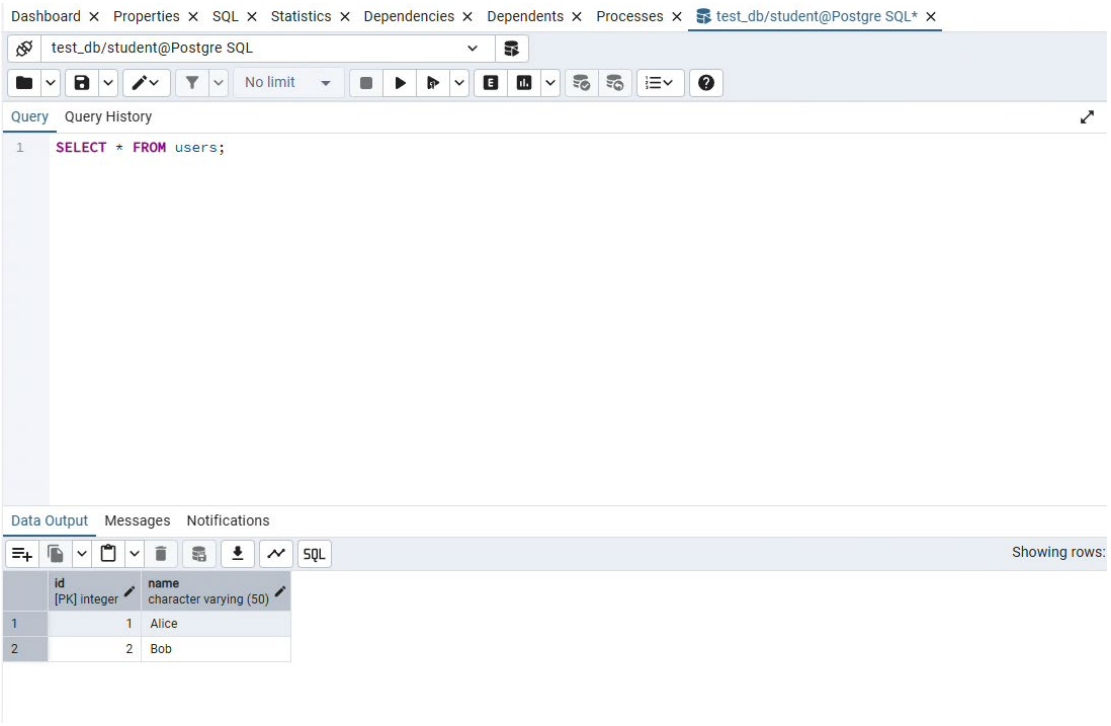

Рисунок 15 – Проверка данных в pgAdmin

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).

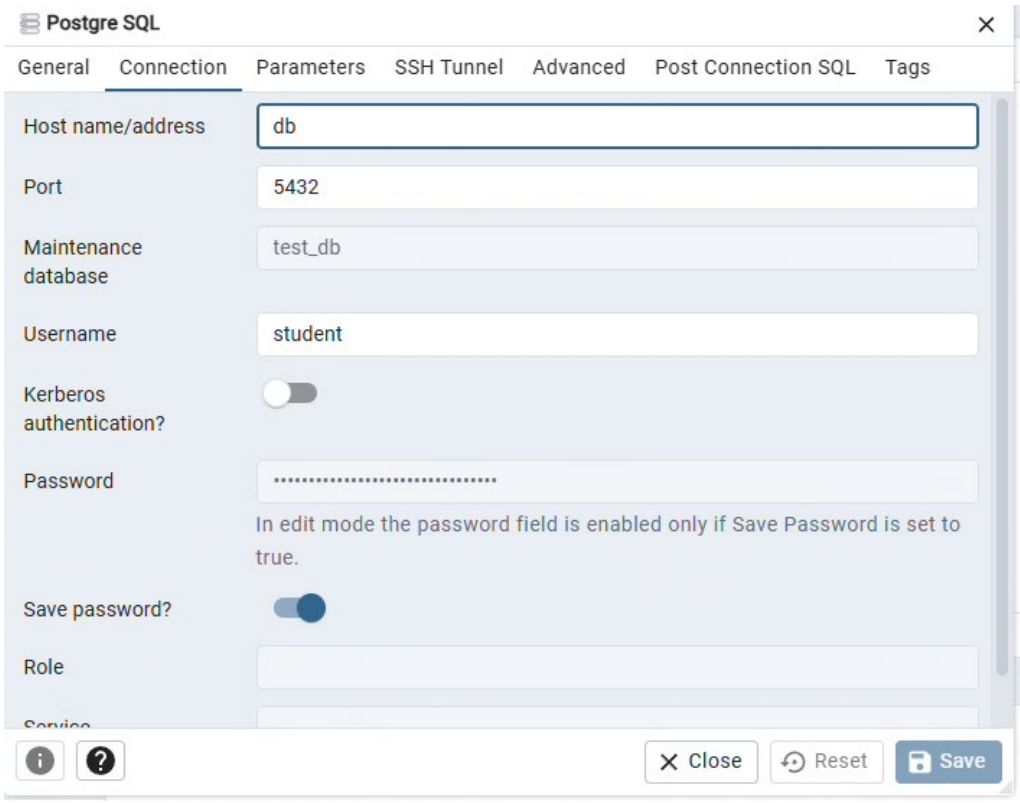

Рисунок 16 – Подключение БД к pgAdmin

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.

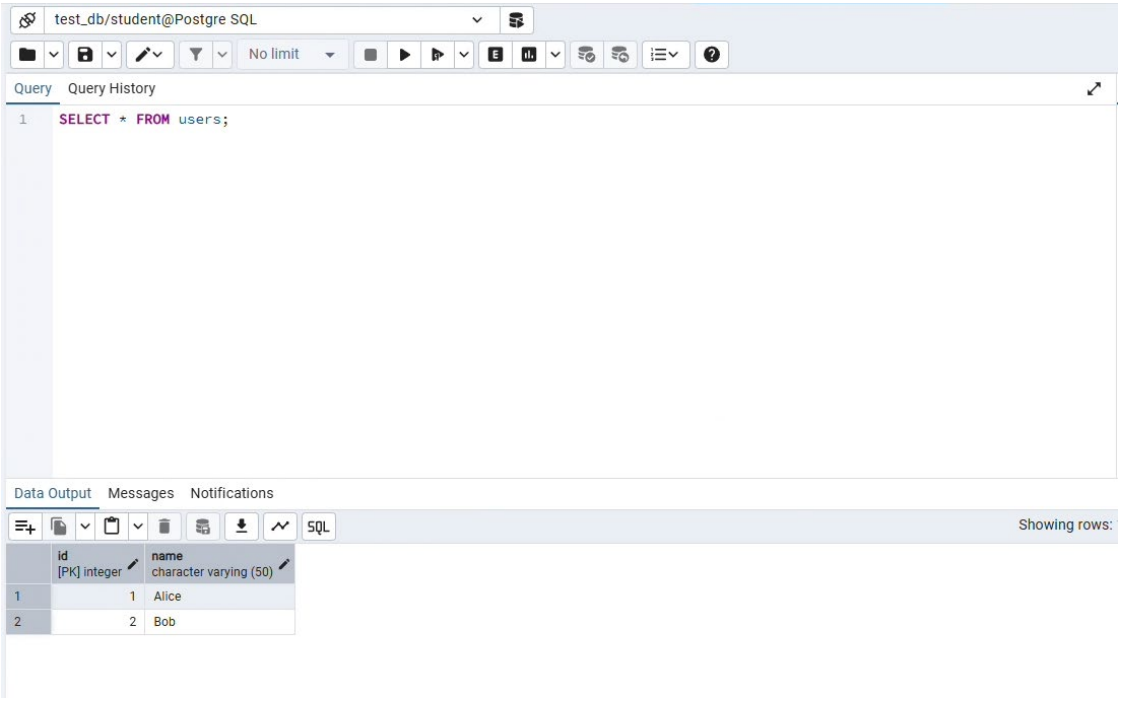

Рисунок 17 – Проверка данных после второго «поднятия»

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/bepis-art/application-development-2025-urfu.git