Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

**Отчет по лабораторной работе №9**

Дисциплина: «Разработка профессиональных приложений»

Docker

Вариант №8

Выполнил:

студент группы ИВТАСбд-21

Зюзин Г.А.

Проверил:

преподаватель кафедры

«Вычислительная техника»

Исхаков И.И.

Ульяновск, 2023

**Задание по варианту.**

Для лабораторной работы №8 подготовить Docker Compose файл, позволяющий запустить 2 контейнера: с автоматическим развертыванием всех необходимых библиотек и запуска gunicorn сервера и СУБД PostgreSQL.

**Описание реализации.**

В данной лабораторной работе работа №8 была перенесена на Docker с созданием контейнеров: Django, gunicorn и PostgreSQL.

Docker Compose файла, который позволит вам запустить два контейнера: один для развертывания сервера gunicorn с необходимыми библиотеками, и второй для СУБД PostgreSQL.

Для создания контейнера использовалось 2 основных файла, описывающие работу контейнера: docker-compose.yml и Dockerfile. Также были созданы конфигурационные файлы для сервера и базы данных, а также для загрузки зависимостей.

В корневой папке проекта были созданы три файла:

1. docker-compose.yml - это файл конфигурации для управления контейнерами с помощью Docker Compose. В нем определяются службы (services), их конфигурация, зависимости и другие параметры, необходимые для запуска и связывания контейнеров.

|  |
| --- |
| **Файл docker-compose.yml:**  version: '3'  services:  db:  image: postgres:latest  volumes:  - ./data:/var/lib/postgresql/data  environment:  - POSTGRES\_USER=postgres  - POSTGRES\_PASSWORD=postgres  django:  build:  context: .  dockerfile: Dockerfile  volumes:  - .:/app  ports:  - "8000:8000"  depends\_on:  - db |

* Версия 3 - версия Docker Compose.
* В разделе services определены две службы: db и django.
* Служба db использует образ postgres:latest, создает том для хранения данных базы данных PostgreSQL и задает переменные окружения для пользователя и пароля базы данных.
* Служба django собирается из текущего контекста с использованием Dockerfile. Также применяются привязки томов и проброс портов. Служба зависит от службы db.

1. Dockerfile - это файл, используемый для создания образа контейнера. Он содержит инструкции для сборки образа, включая базовый образ, установку зависимостей, копирование файлов проекта и определение команды, которая будет выполнена при запуске контейнера.

|  |
| --- |
| **Файл Dockerfile:**  FROM python:3.10.6  WORKDIR /app  COPY requirements.txt .  RUN pip install -r requirements.txt  COPY . .  CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"] |

* Используется образ Python версии 3.10.6.
* Рабочая директория установлена на /app.
* Файлы requirements.txt копируются в контейнер.
* Устанавливаются зависимости, указанные в requirements.txt.
* Все файлы проекта копируются в контейнер.
* Команда CMD задает команду, которая будет выполнена при запуске контейнера. В данном случае, выполняется команда python manage.py runserver 0.0.0.0:8000.

1. requirements.txt - это файл, содержащий список зависимостей Python-проекта. Он используется для установки всех необходимых пакетов и библиотек внутри контейнера.

|  |
| --- |
| Django==4.7.1  gunicorn==20.1.0 |

Здесь перечислены зависимости Python-проекта. В данном случае, указаны Django версии 4.7.1 и Gunicorn версии 20.1.0.

Таким образом, эти файлы используются для настройки и запуска контейнеров с помощью Docker Compose. docker-compose.yml описывает структуру сервисов и их зависимости, Dockerfile определяет, как создать образ для службы django, а requirements.txt перечисляет зависимости проекта Python. Все файлы должны быть помещены в корневую директорию проекта.

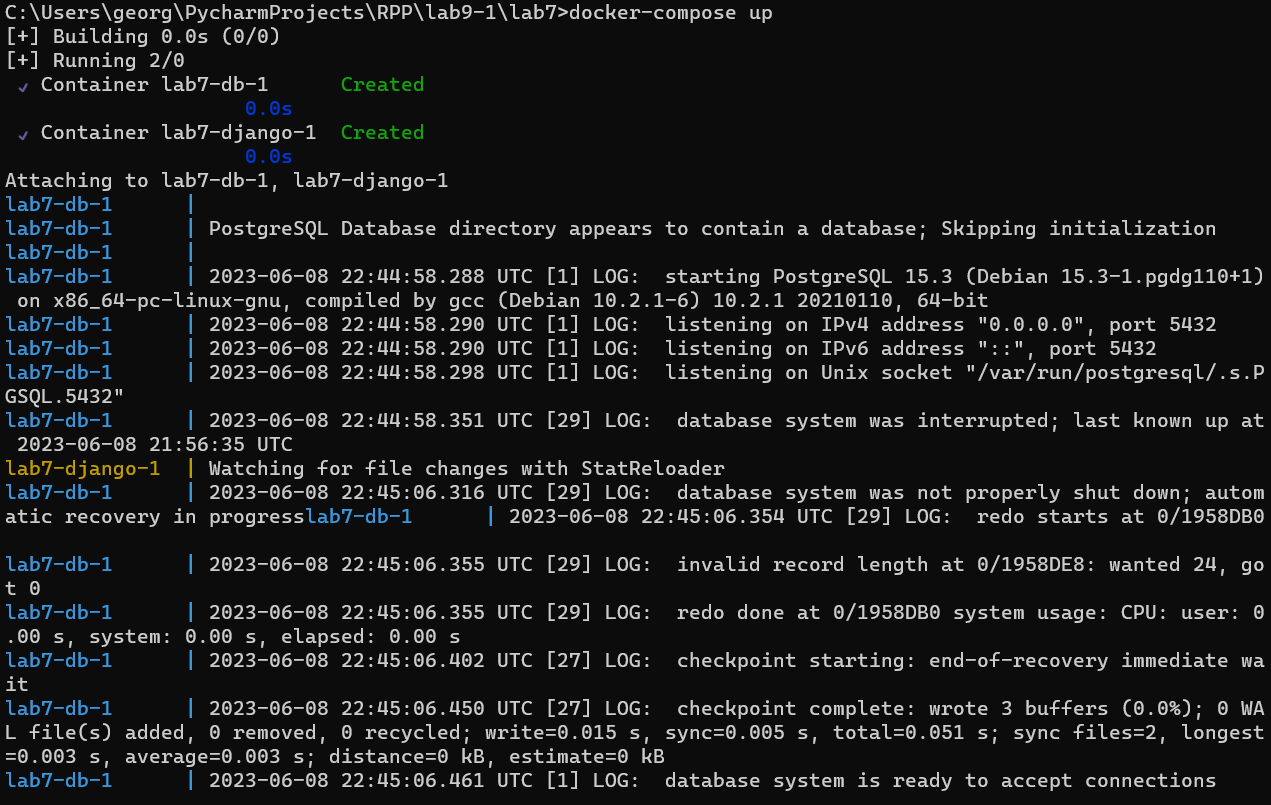
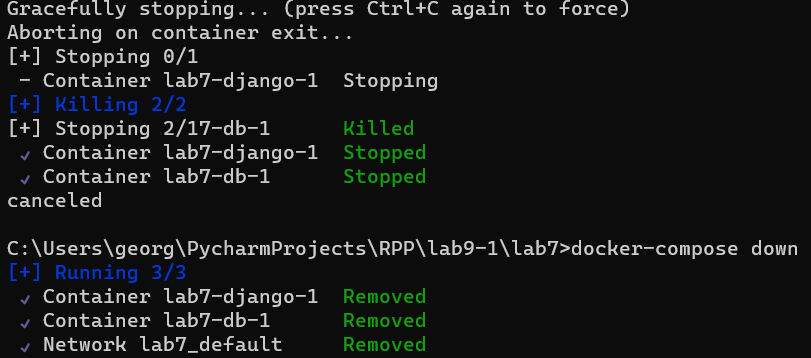
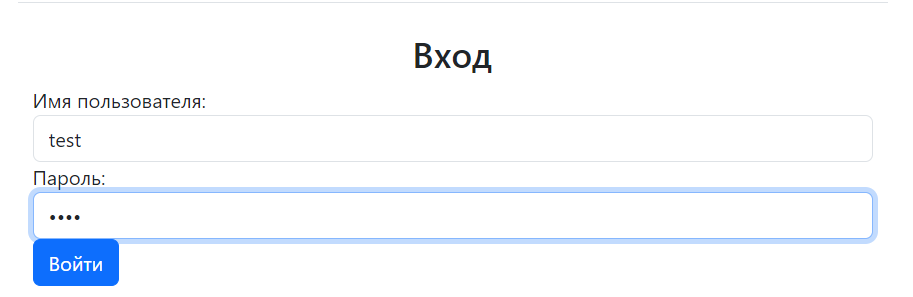
Чтобы запустить контейнеры с помощью Docker Compose, в терминале прописывается команда docker-compose up (docker-compose down – останавливает и удаляет контейнеры).

Рис.1. Запуск контейнеров.

Рис.2. Удаление контейнеров.

Таким образом, первая служба db запускает контейнер с PostgreSQL, который будет использоваться в Django-приложении, а вторая служба django запускает контейнер с Django-приложением, используя образ, созданный на основе Dockerfile, и связывает его с контейнером PostgreSQL для обеспечения работы приложения с базой данных.

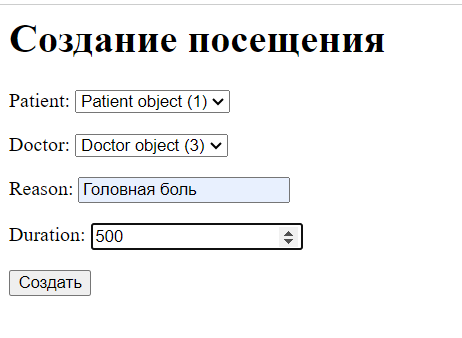
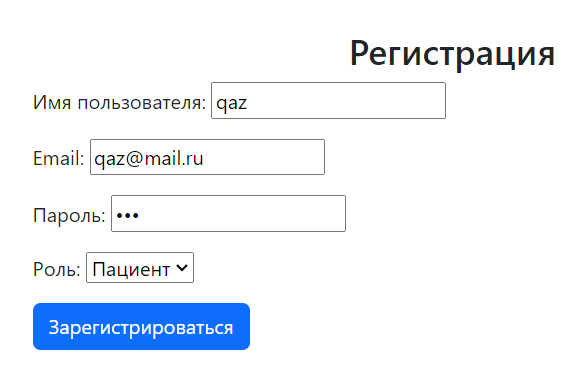
Рис. 3. Авторизация

Рис. 4. Добавление записи.

Рис. 5. Регистрация.

**Описание возникших затруднений.**

Первая трудность, которая возникла, была в том, что на порт 8000 был занят другим привязанным проектом на Django, то есть порт уже был занят другим процессом. По этой причине, чтобы запустить новые контейнеры, необходимо было прервать процессы, происходящие на этом порте.

Выполнив команду netstat -ano | findstr :8000, команда выводит процессы, которыми занят порт. А чтобы прервать процесс в порту используется команда taskkill /F /PID <PID>.

Следующая трудность заключалось в том, что было трудно определить куда расположишь докер-файлы. В итоге они были расположены в корневой папке проекта.

**Описание альтернативных способов решения.**

Альтернативные способы решения этой задачи не рассматривались.

**Вывод.**

В результате выполнения данной лабораторной работы была реализована интеграция Docker в проект на основе Django. Был создан Docker Compose файл, позволяющий запустить два контейнера: один для развертывания gunicorn сервера с необходимыми библиотеками, а другой для СУБД PostgreSQL. Это интеграция Docker позволяет упростить развертывание проекта и обеспечить его портативность и изолированность. Docker Compose файл позволяет определить и управлять несколькими связанными контейнерами одновременно, что облегчает разработку и развертывание комплексных приложений.