Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

Разработка профессиональных приложений

(название дисциплины)

**Отчёт по лабораторной работе №9**

«Docker»

(название (тема) работы)

Выполнил:

студент группы ИВТАПбд-21

Аронов В. В.

Проверил(а):

преподаватель кафедры «ВТ»

Исхаков И. И.

Ульяновск

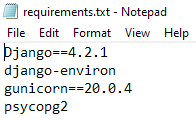
[2023](https://github.com/ghub-ayrtom/DPA-lab5)

[**Постановка задачи**](https://github.com/ghub-ayrtom/DPA-lab5)

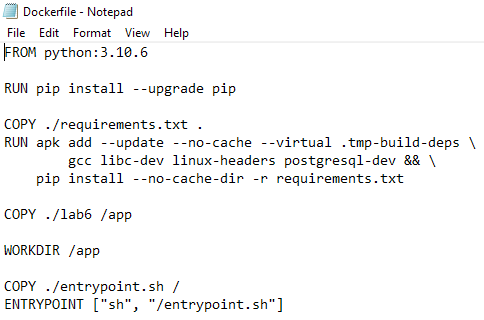
Для лабораторной работы №8 необходимо подготовить Docker Compose файл, позволяющий запустить 2 контейнера: 1. с автоматическим развертыванием всех необходимых библиотек и запуска gunicorn сервера; 2. для СУБД postgresql.

**Описание реализации**

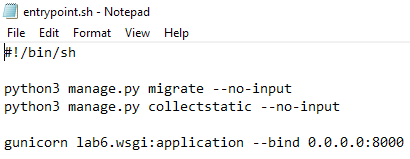
Так как данную лабораторную работу я делал по туториалу из YouTube, то её выполнение я начал может быть не с самой очевидной вещи, а именно с создания в папке с python проектом текстового файла requirements.txt, в котором прописал все необходимые библиотеки, которые понадобятся для развёртывания Docker-контейнеров:



После этого, в той же папке с python проектом был создан файл Dockerfile. Это файл для предварительной работы, набор инструкций, который нужен для записи образа. В нем описывается, что должно находиться в образе, какие команды, зависимости и процессы он будет содержать. При запуске команды docker run программа сначала проверяет, есть ли нужный образ в локальном хранилище:



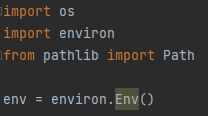
В его конце можно увидеть запуск скрипта entrypoint.sh, который выглядит следующим образом:

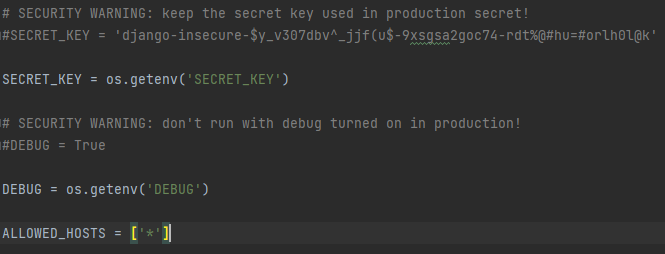


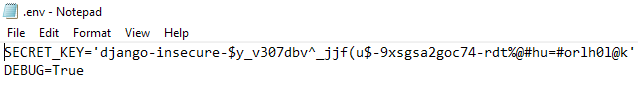
После этого был создан самый основной файл при работе с технологией Docker – это docker-compose.yml — конфигурационный файл в YAML-формате, описывающий логику запуска и взаимодействия контейнеров между собой и внешним миром. В сущности инструкции заложенные в docker-compose.yml по логике работы идентичны ключам команды docker run:

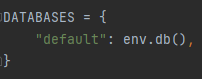


Для работы с Docker Compose также следующим образом был изменён файл settings.py нашего Django-приложения:





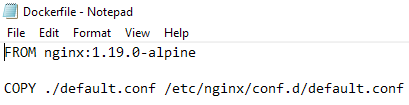






Дополнительно был создан 3 контейнер для развёртывания и запуска nginx сервера, но, как оказалось после, можно было обойтись и без него.

Для начала, в папке с python проектом DPA-lab6 был создан подкаталог nginx и в нём 2 файла: default.conf и ещё один Dockerfile, который имеет следующую структуру:

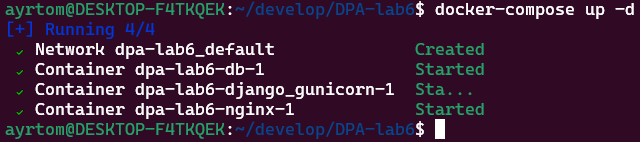


В нём мы командой FROM импортируем nginx версии 1.19.0. и заменяем конфиг nginx по умолчанию тем конфигом, который мы создали сами и который выглядит следующим образом:

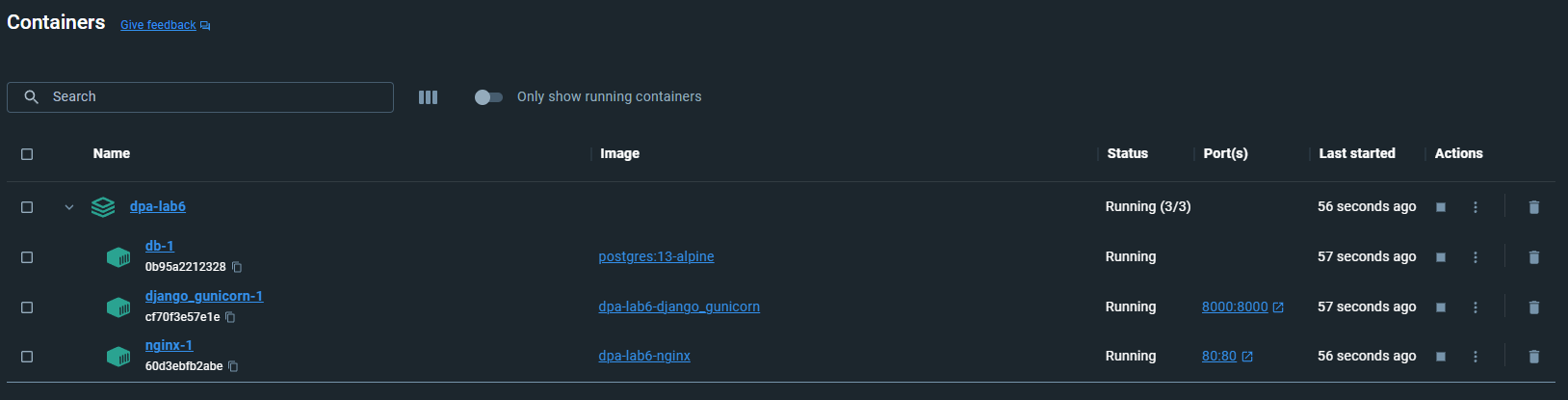


Как видно, веб-сервер nginx занимает 80 порт, в свою очередь WSGI-сервер gunicorn занимает стандартный порт 8000, он расчитан на более серьёзную нагрузку, нежели локальный django-сервер, а также является более безопасным.

После того как все шаги выше были успешно выполнены, можно попробовать запустить docker-compose файл следующей командой в терминале WSL:



Развернулось 3 Docker-контейнера, которые выглядят следующим образом:



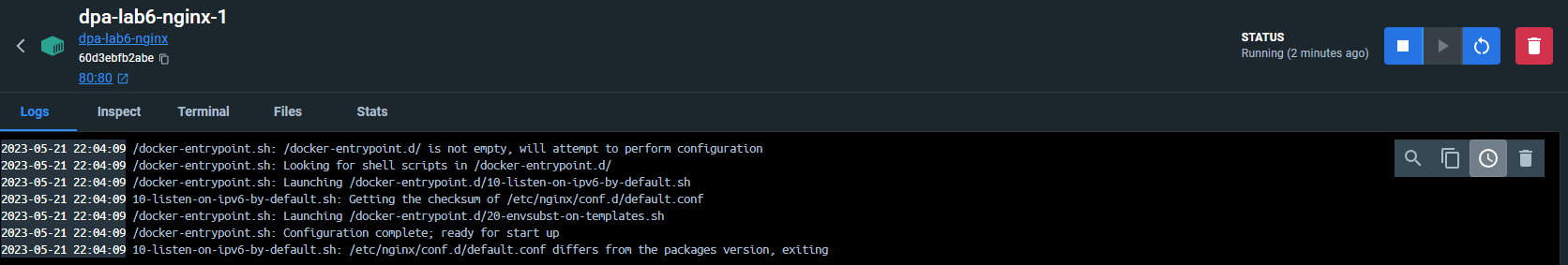
Контейнер dpa-lab6-db-1 отвечает за развёртывание СУБД postgresql:



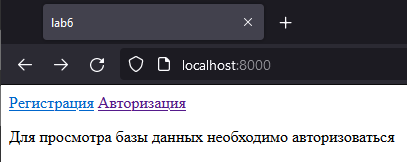
Контейнер dpa-lab6-django-gunicorn-1 отвечает за автоматическое развёртывание всех необходимых библиотек и запуска gunicorn сервера:



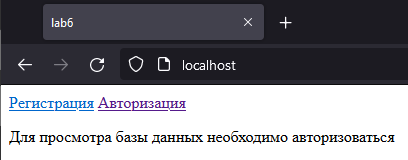
Контейнер dpa-lab6-nginx-1 отвечает за запуск nginx-сервера:



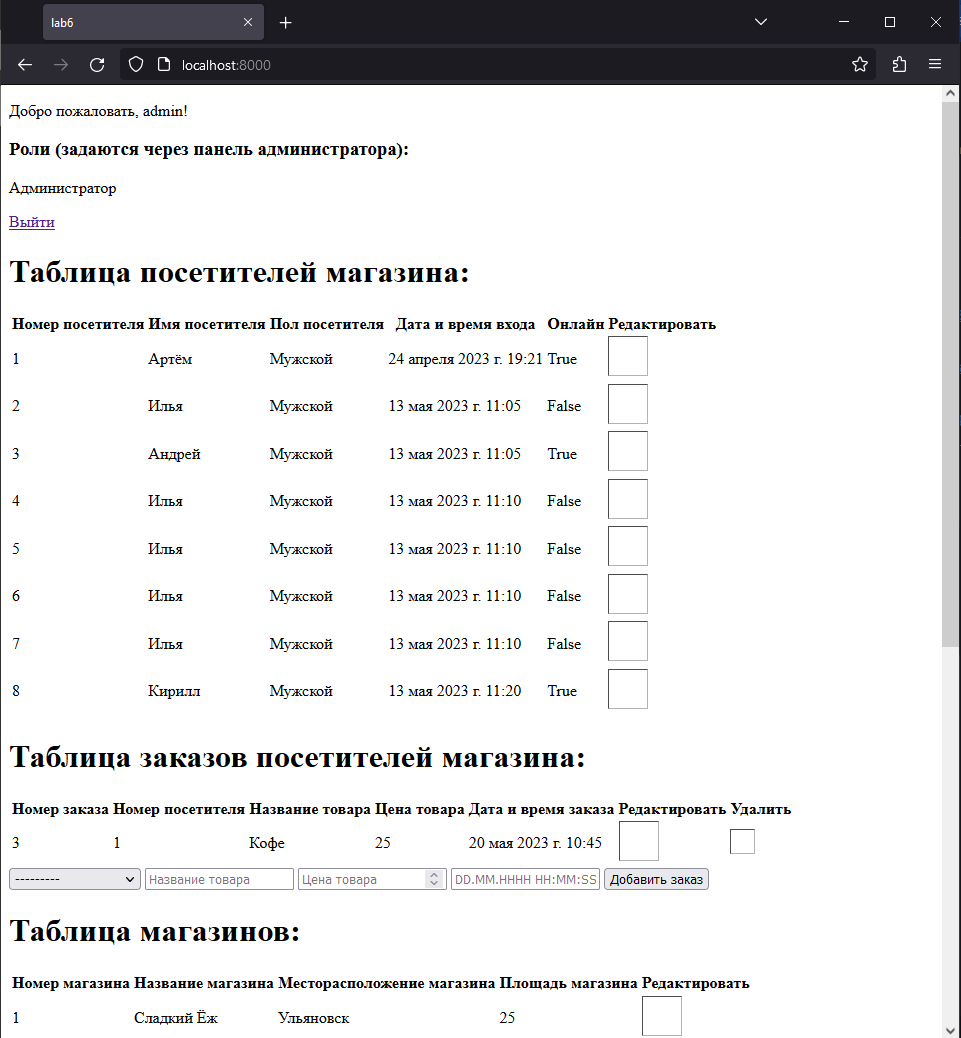
Теперь можно попробовать зайти на сам локальный сервер gunicorn (порт: 8000) и посмотреть, всё ли работает:



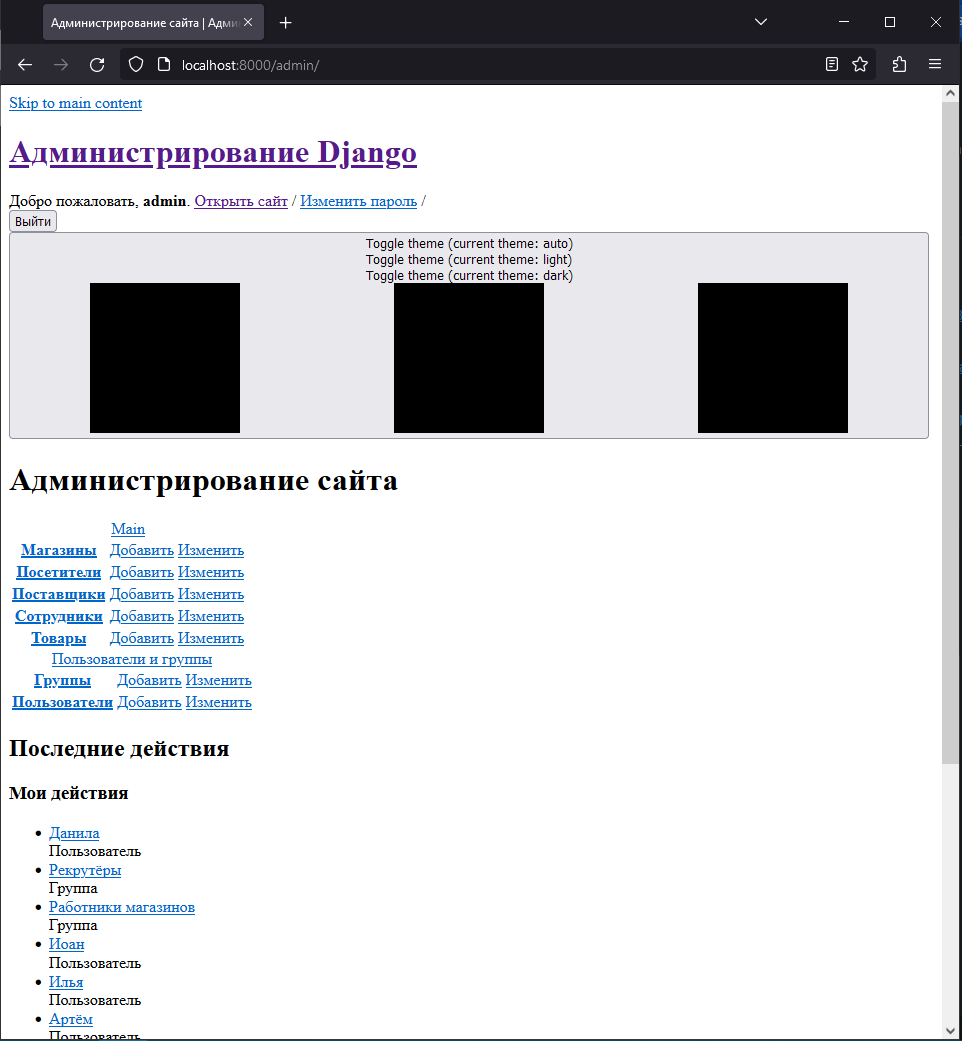
Или же зайти на nginx-сервер (порт: 80) и увидеть всё то же самое, что и на gunicorn-сервере:



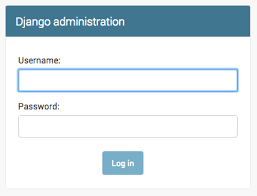
Казалось бы, что всё работает корректно, однако, если попробовать перейти на страницу (авторизоваться), которая имеет хотя бы минимальное количество css стилей, то можно увидеть следующую неприятную картину:



Стандратная панель администратора Django без стилей:

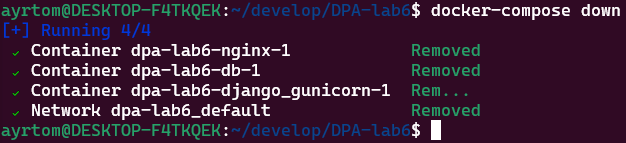


И как она должна выглядеть со стилями:



Нетрудно заметить, что все css стили и другие статические файлы не были найдены и корректно загружены при развёртывании Docker-контейнеров. К сожалению, мне так и не удалось исправить эту проблему, так как в туториалах, по которым я выполнял данную лабораторную работу использовались простейшие новые Django-проекты с их стандартной стартовой страницей (с ракетой) на которой отсутствуют css стили.

По завершению работы с развёрнутым из Docker Compose файла приложением, контейнеры необходимо свернуть обратно следующей командой в терминале WSL:



**Описание возникших затруднений**

Основные затруднения при выполнении данной лабораторной работой были связаны с первичным знакомством с технологией контейнеризации Docker, так как до этого мне не приходилось с ней работать. Было достаточно непросто понять с чего именно нужно начать, с создания какого файла, а после этого было трудно разобраться как во внутренней структуре самих файлов, так и в том, в каком порядке они должны создаваться и подключаться друг к другу.

Также, немало проблем возникло с подключением и установкой необходимых библиотек и их совместимостью. Некоторые проблемы, например с css стилями, решить, к сожалению, так и не удалось.

**Вывод**

По итогу выполнения данной лабораторной работы был создан Docker Compose файл и множество вспомогательных для него файлов, которые позволяют запустить 3 контейнера: 1. с автоматическим развертыванием всех необходимых библиотек и запуска gunicorn сервера; 2. для СУБД postgresql и 3. с запуском nginx сервера