Департамент образования и науки города Москвы
Государственное автономное образовательное учреждение высшего образования города Москвы
«Московский городской педагогический университет»
Институт цифрового образования
Департамент информатики, управления и технологий

ДИСЦИПЛИНА:

Интеграция и развертывание программного обеспечения с помощью контейнеров

Лабораторная работа 2.1

Создание Dockerfile и сборка образа

Выполнила: st_105

Москва
2025

Цель работы: научиться создавать Dockerfile и собирать образы Docker для приложений.

Задачи:

— Создать Dockerfile для указанного приложения.

— Собрать образ Docker с использованием созданного Dockerfile.

— Запустить контейнер из собранного образа и проверить его работоспособность.

— Выполнить индивидуальное задание.

Вариант 15. Создайте Dockerfile для приложения на Python, которое использует библиотеку Flask для создания простого REST API с одним конечной точкой.

(REST API (Representational State Transfer Application Programming Interface) — это архитектурный стиль для создания веб-сервисов, который позволяет клиентам (например, браузерам или мобильным приложениям) взаимодействовать с сервером через HTTP-запросы.

Конечная точка (endpoint) — это URL-адрес, по которому клиент (например, браузер или другое приложение) может обратиться к серверу для выполнения определённой операции.

Одна конечная точка означает, что ваше API будет иметь только один URL-адрес, который будет обрабатывать запросы.)

ХОД РАБОТЫ:

1. Создана директория python-app в которой проведена вся дальнейшая работа (рисунок 1)

Рисунок 1 – создание директории

2. Создан файл с расширением .py (рисунок 3)

Главная страница (/):

Возвращает приветственное сообщение в формате JSON.

Маршрут /api/hello:

Возвращает сообщение "Hello, World!" и выводит в терминал цветное сообщение о том, что кто-то обратился к этому маршруту.

Маршрут /api/colors:

Возвращает список цветов в формате JSON и выводит в терминал цветное сообщение о доступе к этому маршруту.

Маршрут /api/quote:

Возвращает случайную цитату в формате JSON и выводит в терминал цветное сообщение с этой цитатой.

Используется библиотека colorama для цветного вывода сообщений в терминале.

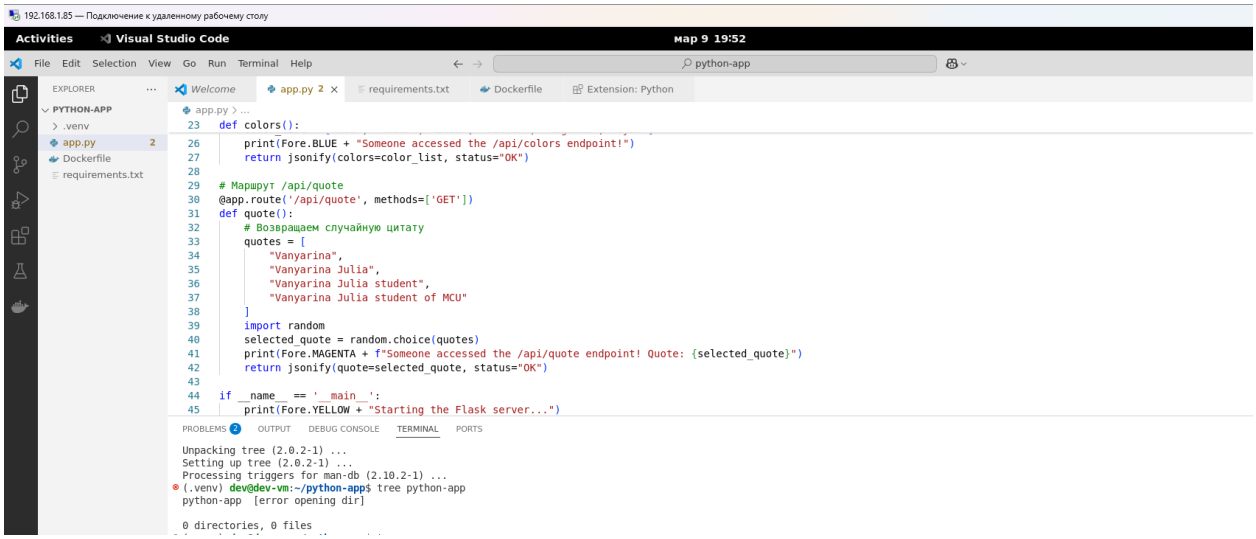

Рисунок 3 – Файл python

3. Созданы файлы requirements.txt и Dockerfile (рисунок 4-5)

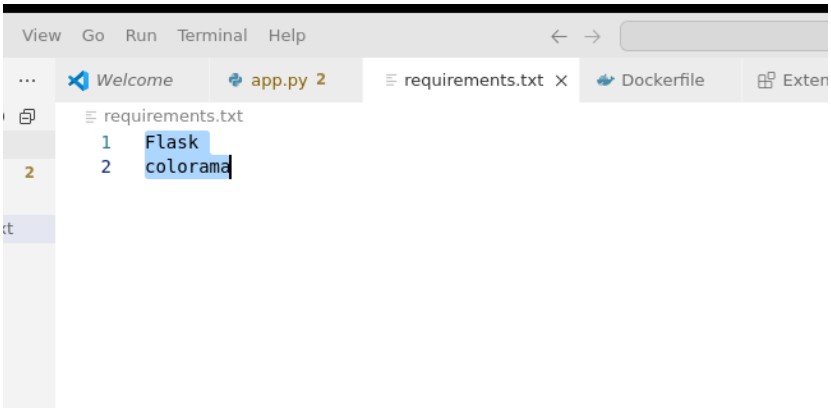

Рисунок 4 – файл для установки

Рисунок 5 – Dockerfile

4. С помощью docker build собираем приложение (рисунок 6)

```
(.venv) dev@dev-vm:~/python-app$ docker build -t colorful-flask-app-py .
2025/03/09 20:20:23 in: []string{}
2025/03/09 20:20:23 Parsed entitlements: []
[+] Building 1.4s (10/10) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 228B
 => [internal] load metadata for docker.io/library/python:3.9-slim
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/5] FROM docker.io/library/python:3.9-slim@sha256:d1fd807555208707ec95b284afd10048d0737e84b5f2d6fdcbed2922b9284b56
 => [internal] load build context
 => => transferring context: 1.61kB
 => CACHED [2/5] WORKDIR /app
 => CACHED [3/5] COPY requirements.txt .
 => CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt
 => [5/5] COPY app.py .
 => exporting to image
 => => exporting layers
 => => writing image sha256:ad558bd0560914f65b57dfd2ab774cb5c13290d16a997e71c0ae75415f8c1824
```

Рисунок 6 – сборка приложения

5. Запуск контейнера и вывод логов (рисунок 7)

```
(.venv) dev@dev-vm:~/python-app$ docker run -t --name my-colorful-app -p 5000:5000 colorful-flask-app-py
Starting the Flask server...
 * Serving Flask app 'app'
 * Debug mode: off
Someone accessed the /api/hello endpoint!
Someone accessed the /api/colors endpoint!
Someone accessed the /api/quote endpoint! Quote: Vanyarina Julia
Someone accessed the /api/quote endpoint! Quote: Vanyarina Julia student of MCU
Someone accessed the /api/quote endpoint! Quote: Vanyarina Julia
Someone accessed the /api/hello endpoint!
Someone accessed the /api/quote endpoint! Quote: Vanyarina
```

Рисунок 7 – цветной вывод

6. Проверка доступа на localhost (рисунок 8-10) и с помощью curl (рисунок 11)

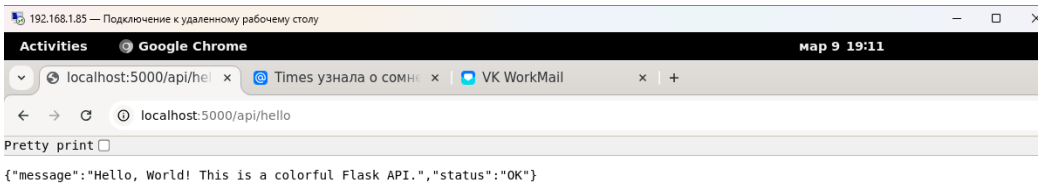

Рисунок 8 –проверка api/hello

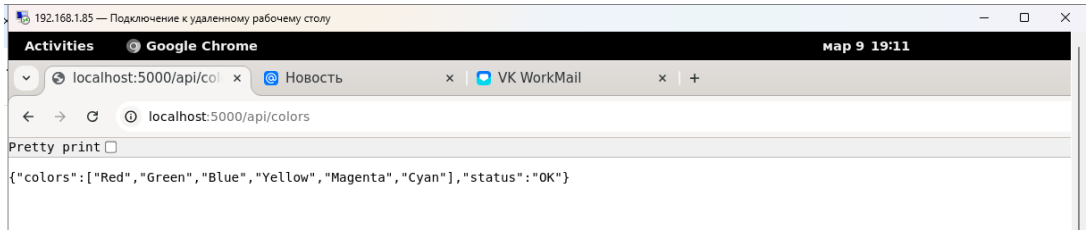

Рисунок 9 – проверка api/colors

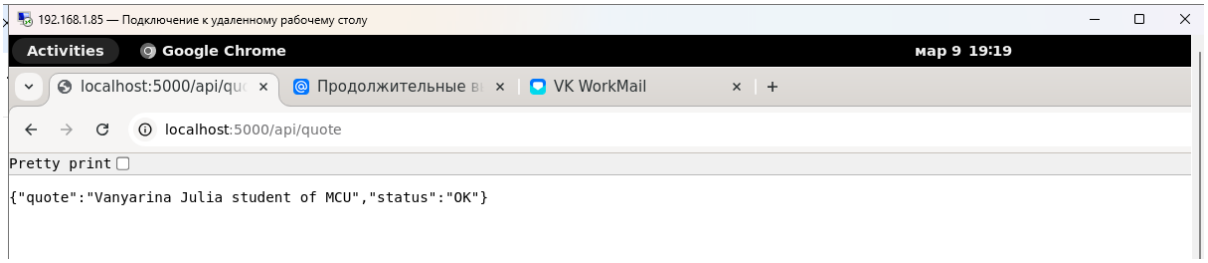

Рисунок 10 – проверка api/quote

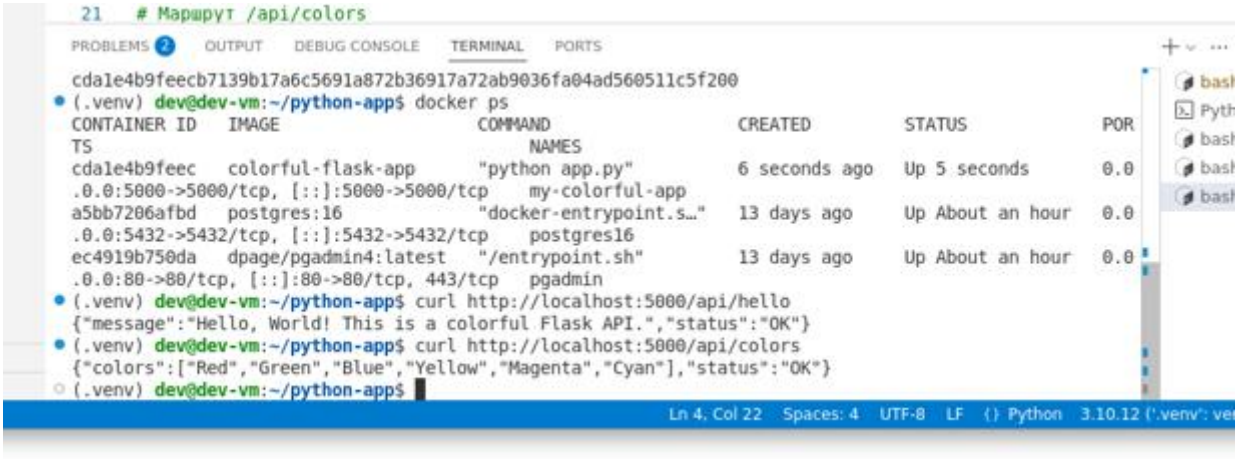

Рисунок 11 – проверка с помощью curl

Итоговая структура файлов (рисунок 12):

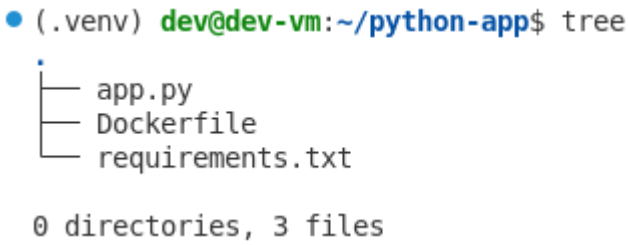

Рисунок 12 – дерево директории

Выводы по работе: В ходе лабораторной работы я научилась создавать Dockerfile для упаковки Python-приложения с использованием Flask, а также собирать и запускать Docker-контейнеры. Было реализовано простое REST API с одной конечной точкой, а также добавлен цветной вывод в терминал с помощью библиотеки colorama.