Департамент образования города Москвы

Государственное автономное образовательное учреждение

высшего образования города Москвы

«Московский городской педагогический университет»

Институт цифрового образования

Департамент информатики, управления и технологий

ОТЧЕТ

по дисциплине «Интеграция и развертывание программного обеспечения с помощью контейнеров»

Направление подготовки 38.03.05 – бизнес-информатика

Профиль подготовки «Аналитика данных и эффективное управление»

(очная форма обучения)

Лабораторная работа 3.1. Docker Compose для мультиконтейнерных приложений

Выполнил:

st\_89

Руководитель:

Москва  
2025

**Цель работы:** Освоить использование Docker Compose для управления многоконтейнерными приложениями.

**Задачи:**

1. Создать файл docker-compose.yml для указанного многоконтейнерного приложения.
2. Запустить приложение с помощью Docker Compose.
3. Проверить работоспособность приложения и взаимодействие между контейнерами.
4. Выполнить индивидуальное задание.

**Вариант 2**.:

1. Создать файл docker-compose.yml для системы прогнозирования спроса (Python Flask + PostgreSQL).
2. Запустить приложение и проверить корректность записи данных в базу.
3. Реализовать расчет среднего спроса по категориям товаров.

**Ход работы:**

1. Для начала создаем папку python-flask-app и все необходимые файлы.

Структура следующая (Рис. 1 и Рис. 2):

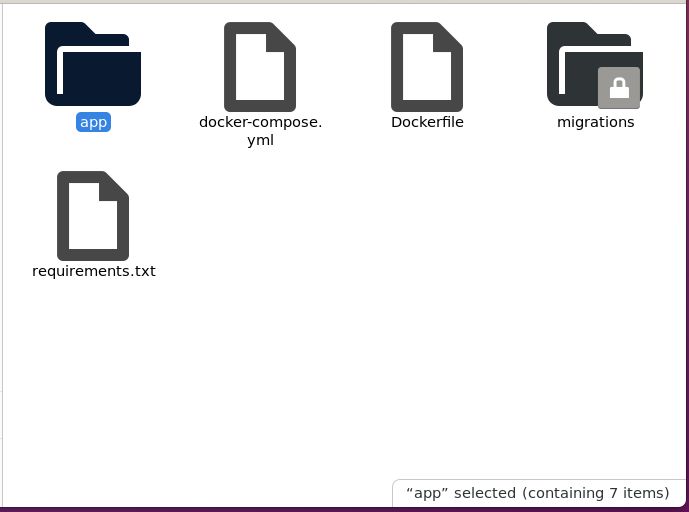


Рис. 1 – Основной каталог python-flask-app

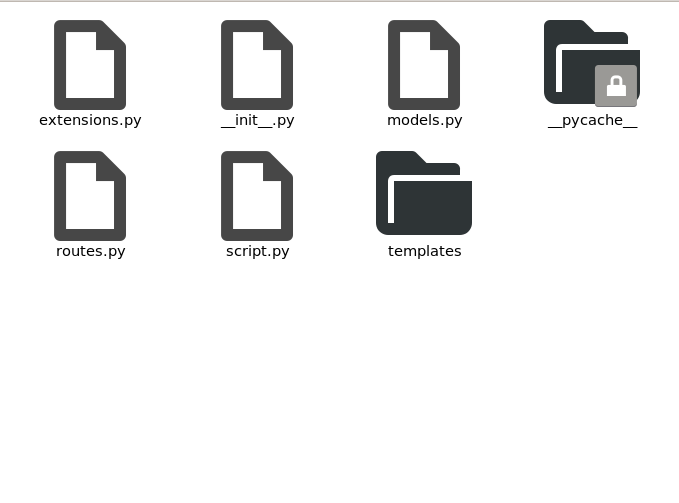


Рис. 2 – подпапка app

1. Листинг docker-compose.yml



Рис. 3 – файл docker-compose.yml

1. Основная часть. Запуск контейнеров, генерация, данных и запись в базу

После запуска контейнера заходим в окружение bash

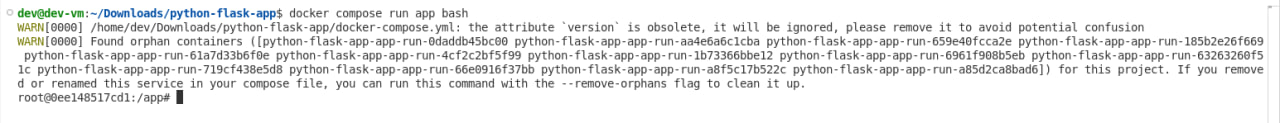


Рис. 4 – Окружение bash

Устанавливаем библиотеку faker для генерации данных

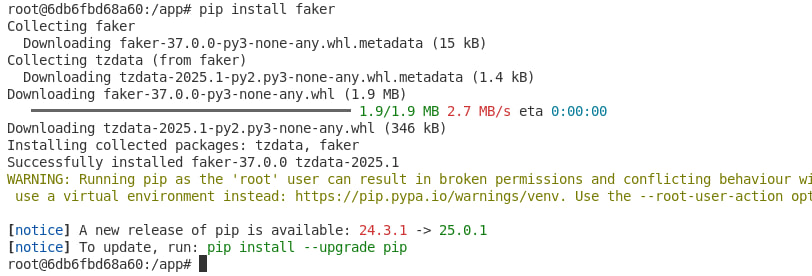


Рис. 5 – Установка faker

Инициализируем базу данных и проводим миграцию для создания базы и последующей загрузки данных в нее.

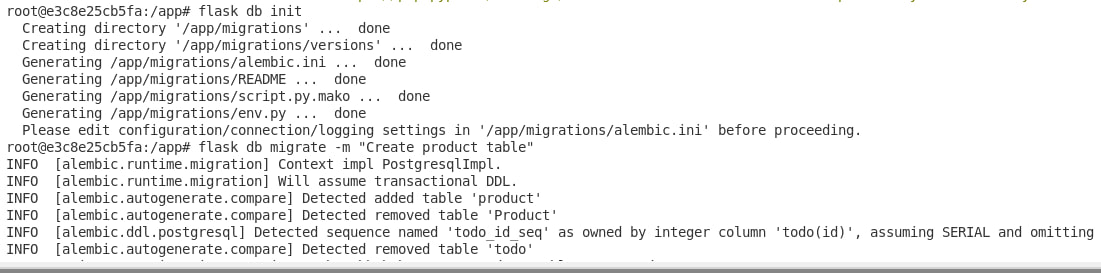


Рис. 6 – Инициализация миграций и базы данных

После чего заходим во flask shell и запускаем скрипт .py для генерации данных

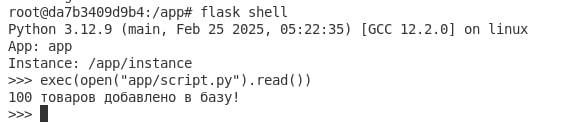


Рис. 7 – Загрузка данных

Заходим в Adminer и видим, что создалась таблица product

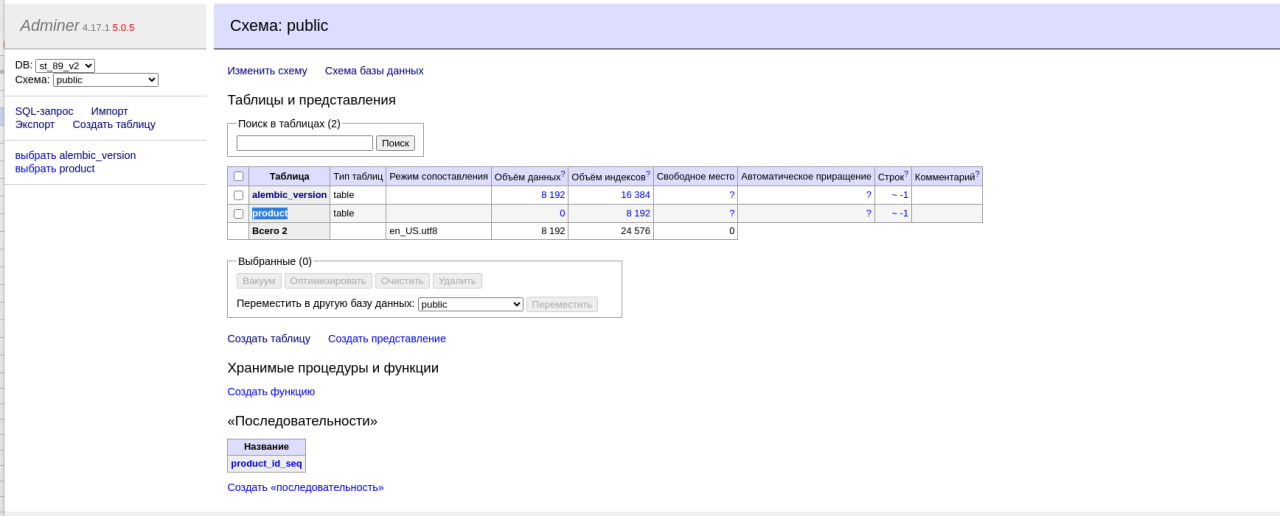


Рис. 8 – Структура базы st\_89\_v2 и таблица product

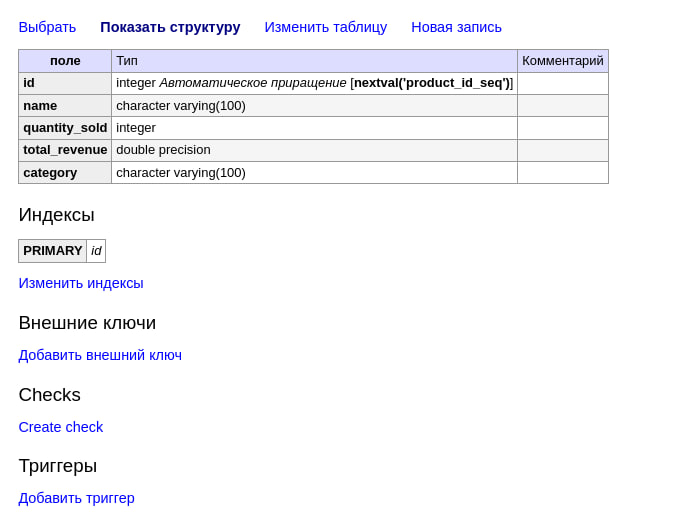


Рис. 9 - Типы данных

С помощью SQL скрипта выведем загруженные данные



Рис. 10 - Загруженные данные

Также, с помощью web-интерфейса можно записывать данные в базу

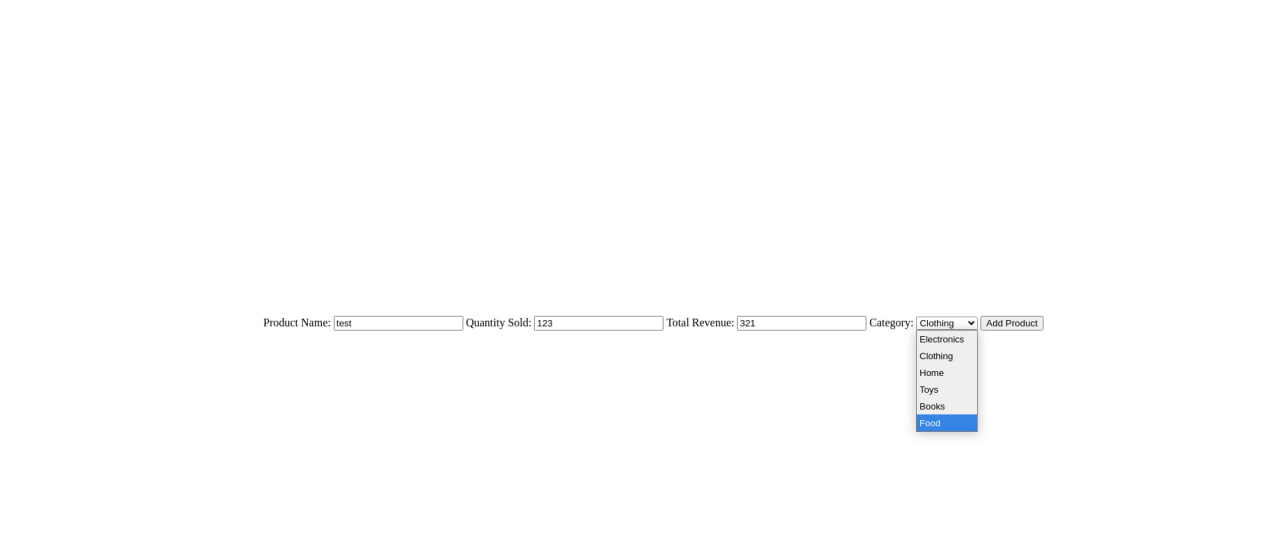


Рис. 11 - Web-интерфейс

Видим, что данные успешно записаны в базу

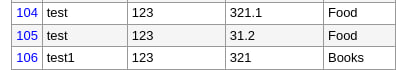


Рис. 12 - Данные из web-интерфейса

Далее был произведен расчет спроса по категории товаров

(все данные фейковые)

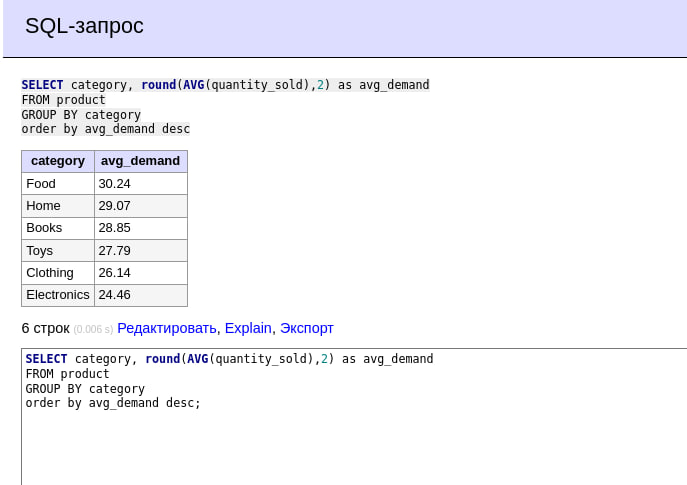


Рис. 13 - Расчет спроса по категории товаров

**Вывод:** Было проведено создание мультиконтейнерного приложение для записи в базу данных с помощью Python+Flask, а также был разработан Web-интерфейс для ручной записи в базу.

**Контрольные вопросы:**

**1. Что такое Docker Compose и для чего он используется?**

Docker Compose — это инструмент для управления многоконтейнерными приложениями в Docker. Он позволяет описывать инфраструктуру приложения в одном файле (docker-compose.yml) и управлять всеми контейнерами с помощью простых команд.

Docker Compose используется для:

1. Упрощённого развертывания многоконтейнерных приложений.
2. Определения сервисов, сетей и хранилищ в одном файле.
3. Автоматизации процессов запуска, масштабирования и удаления контейнеров.

### ****2. Основные преимущества использования Docker Compose****

**Упрощенное управление** – Один YAML-файл описывает всю инфраструктуру.  
**Легкость в развертывании** – Одна команда запускает все контейнеры.  
**Поддержка нескольких сред** – Можно использовать разные конфигурации для разработки, тестирования и продакшна.  
**Легкость в масштабировании** – Простая настройка количества реплик сервисов.  
**Изоляция среды** – Все сервисы работают в одной сети и не конфликтуют с другими контейнерами.

**3. Основные разделы и директивы в docker-compose.yml**

**version** – версия Docker Compose.  
**services** – определение контейнеров (сервисов).   
**image** – Docker-образ, используемый контейнером.  
**ports** – перенаправление портов.  
**volumes** – подключение томов.  
**depends\_on** – задает порядок запуска сервисов.  
**environment** – переменные окружения для контейнера.

version: "3.8" # Версия Docker Compose

services:

web: # Определение сервиса (контейнера)

image: nginx:latest # Используемый образ

ports:

- "80:80" # Проброс порта

volumes:

- ./html:/usr/share/nginx/html # Примонтированные тома

depends\_on:

- db # Запускается после базы данных

db:

image: mysql:5.7

environment:

MYSQL\_ROOT\_PASSWORD: root # Переменные окружения

MYSQL\_DATABASE: app\_db

volumes:

- db\_data:/var/lib/mysql # Том для сохранения данных

volumes:

db\_data: # Определение именованного тома

**4. Как запустить многоконтейнерное приложение с помощью Docker Compose?**

Команда: **docker-compose up –d**

**5. Как остановить и удалить контейнеры, запущенные с помощью Docker Compose?**

docker-compose down – для остановки контейнера

docker-compose stop – остновить без удаления

docker kill $(docker compose ps –q) – остановить и удалить все запущенны контейнеры