**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Государственное бюджетное образовательное учреждение высшего образование**

**Ордена Трудового Красного Знамени**

**«Московский технический университет связи и информатики»**

**Кафедра “Математической кибернетики и информационных технологий”**

**дисциплина «Web-программирование»**

Курсовая работа по дисциплине

«Web-программирование»

Выполнил студент

Группы БФИ1901

Тимонин Григорий Эдуардович

Проверила

Полянцева Ксения Андреевна

Москва 2021

**СОДЕРЖАНИЕ**

Оглавление

[Курсовая работа 3](#_Toc90379446)

[Ход работы 3](#_Toc90379447)

Курсовая работа

Цель работы: разработать REST API-сервис посредством языка Python, протестировать, упаковать его в Docker-контейнер, а также развернуть на сервере.

Ход работы

Что бы изолировать наше приложение и корректно установить зависимости создадим виртуальное окружение командой (python -m venv venv) и активируем его (& venv/Scripts/Activate.ps1) (Рисунок 1)

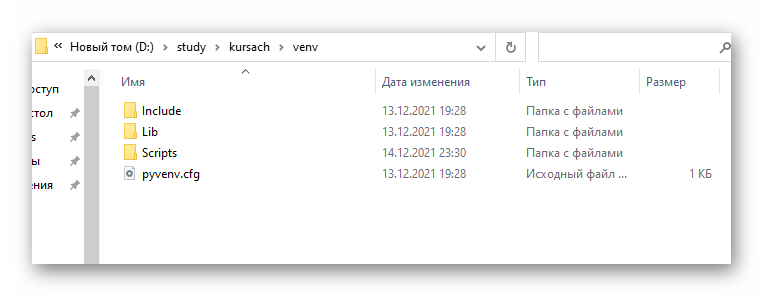


Рисунок 1 – Папка виртуального окружения

Создадим файл requirements.txt и пропишем там зависимости сервиса, чтобы потом установить их одной командой и использовать в последующем в Docker (рисунок 2).

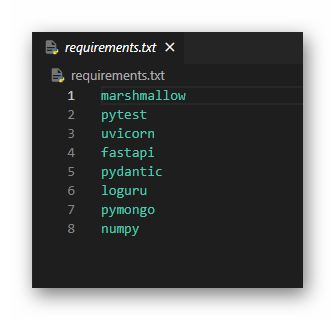


Рисунок 2 – requirements.txt

Для проекта используем следующую структуру (рисунок 3):

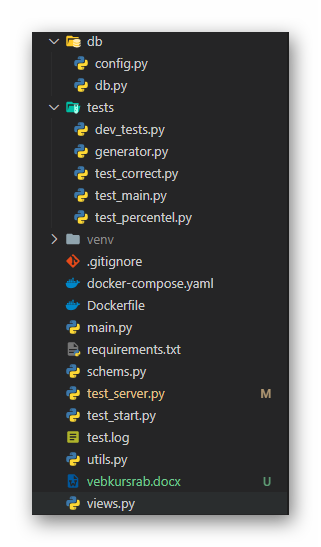


Рисунок 3 – Структура проекта

В качестве базы данных для проекта была выбрана не реляционная база данных MongoDB (Рисунок 4). Эта база была выбрана так как данные поступают на сервер в формате json, а в коллекциях MongoDB его удобно хранить и с ним работать. Так же MongoDB легка в настройке и имеет свой образ для Docker контенера.

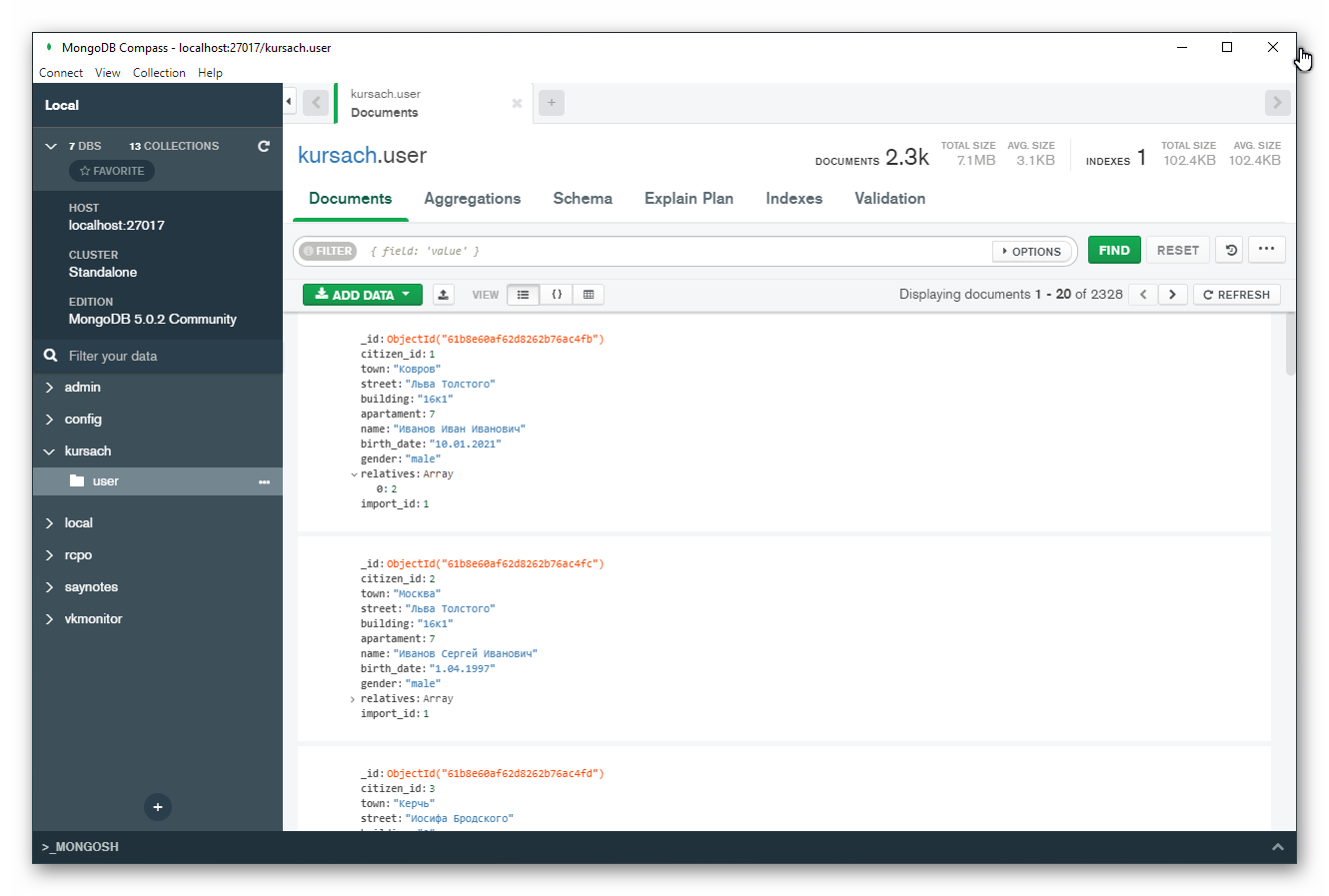


Рисунок 4 – База данных

В качестве веб фреймворка был выбран ассинхронный fastapi, так как асинхронный подход позволяет эффективно обслуживать нескольких клиентов в рамках одного процесса ОС и так как у fastapi хорошая англоязычная документация в интернете. (Рисунок 5)



Рисунок 5 – Содержание файла main.py

Что бы было удобнее работать с базой была созданна папка db с 2 файлами – конфигурацией подключения к базе данных и самим файлом с подключением к коллекции user (Рисунок 6)

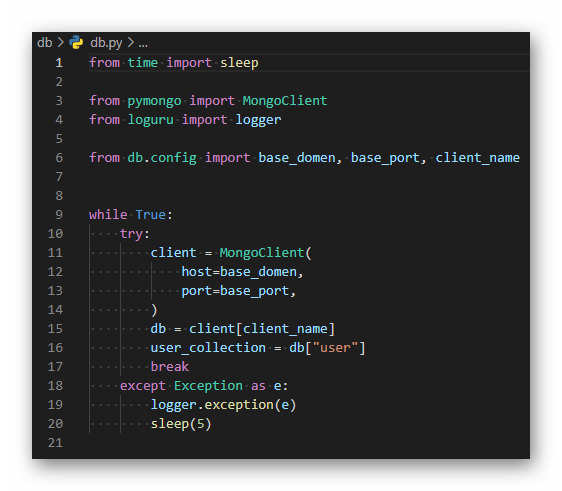


Рисунок 6 – Подключение к базе данных

Так же для того, что бы в случае падения или ошибок сервиса возможно было выявить причину, я настроил логирование. Так как стандартный python logger мне не очень удобен – я скачал библиотеку loguru поставил максимальный размер логов в 100mb и обернул все функции в try – except блоки (Рисунок 7)

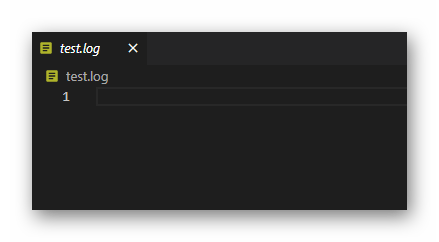


Рисунок 7 – Логирование

Так как поступающие данные нуждаются в проверке – был создан файл schems.py и класс UserSchema, его поля настроенына определенный тип и формат данных, если поступаемые данные будут им несоответствовать, то вызовется исключение, которое будет обработано, а пользователь проинформирован (Рисунок 8)

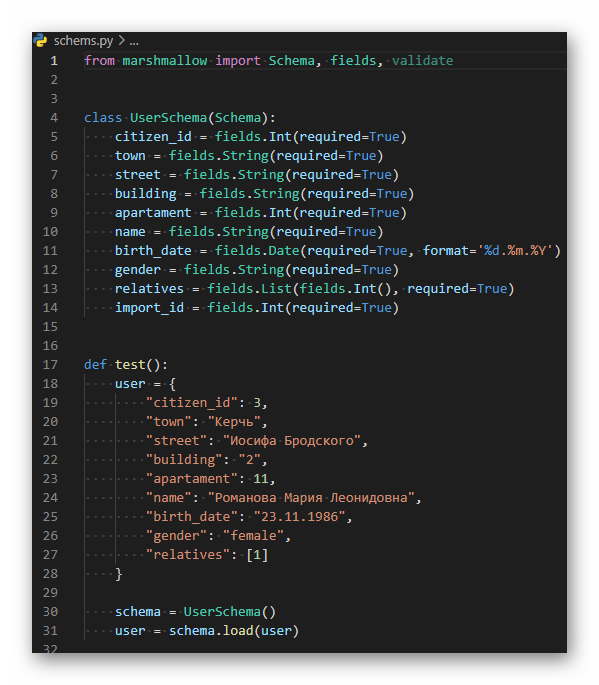


Рисунок 8 – Схема

Что бы быть уверенным в работоспособности сервера были написаны различные тесты с помощью библиотеки pytest, для тестирования поступления больших объемов данных был написан генератор пользователей. Тесты проверяют как отдельные функции так и сам сервер (Рисунок 9)

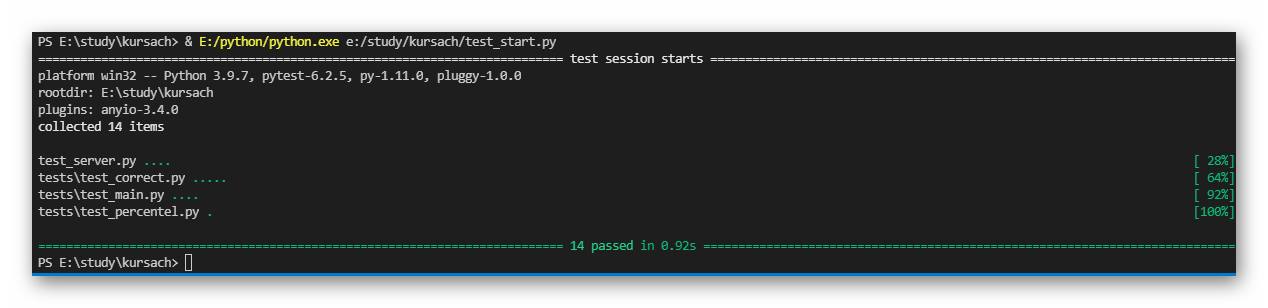


Рисунок 9 – Результат тестирования

Для более легкой настройки на сервере и изоляции api от базы данных, настрою файлы Dockerfile и docker-compose под наш сервис: Полностью их создание я описывать не буду, обращу внимание лишь на запуск fastapi сервиса с помощью скачанной библиотеки uvicorn, настройки хоста и порта которые не будут видимы снаружи – они внутренние – докера. В файле docker-compose установлены порты для контейнера kusach\_api – 8000: порт локального сервера или в последствии сервера на который будет производиться установка соответствует внутреннему 80 порту, который мы прописали в Dockerfile (Рисунки 10, 11)

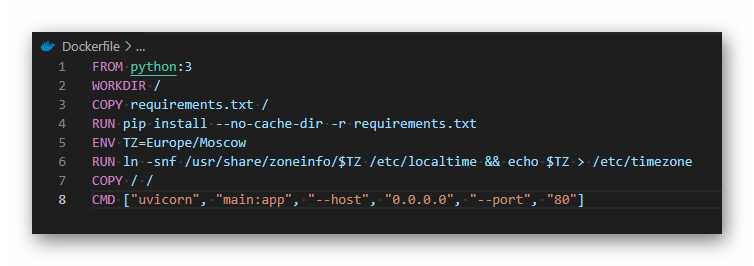


Рисунок 10 – Dockerfile

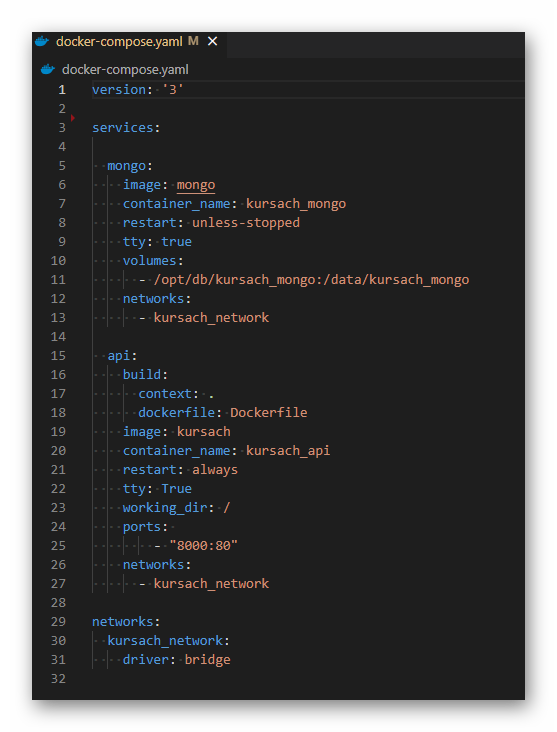


Рисунок 11 – docker-compose

Соберу проект командой docker-compose build и запустим командой docker-compose up. 2 контейнера запустились и работают (Рисунок 12). Далее необходимо отправить проект на гитхаб и подготовить сервер



Рисунок 12 – Работа контейнеров

В качестве сервера для деплоя я выберу vps от хостиноговой reg.ru, так как у них много различных вариантов, хорошая документация и поддержка. У них есть готовый vps с установленным Docker (Рисунок 13). Я выберу самый базовый тариф и закажу сервер

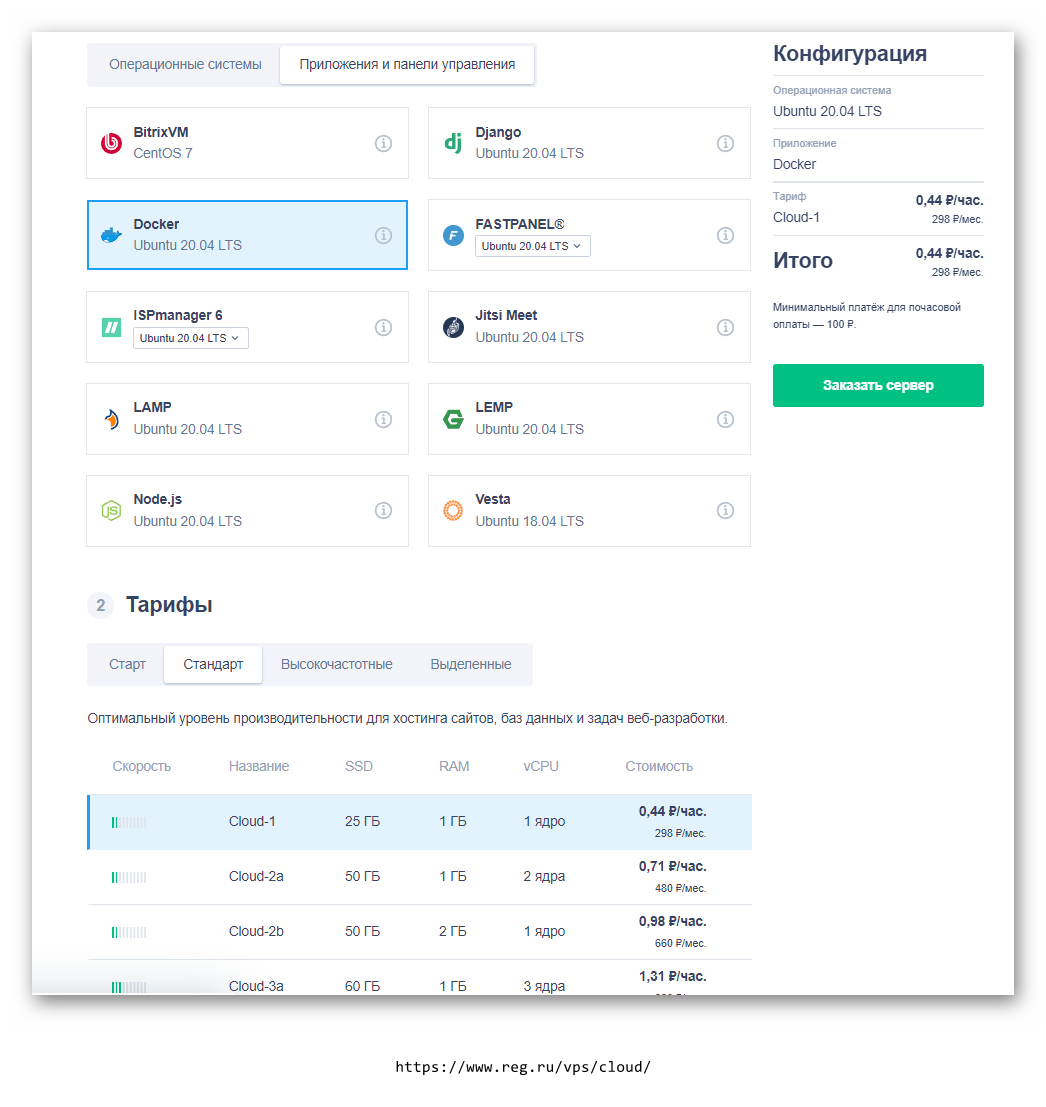


Рисунок 13 – Сайт хостиноговой компании

Для подключения к серверу необходимо скачать ssh клиент я использую PuTTY ввести адрес, логин и пароль. (Рисунок 14). Теперь необходимо скачать сервис с помощью комманды git clone и запустить с помощью команд docker. Теперь сервис доступен по адресу: и можно выполнить некоторые тесты (рисунок 15):

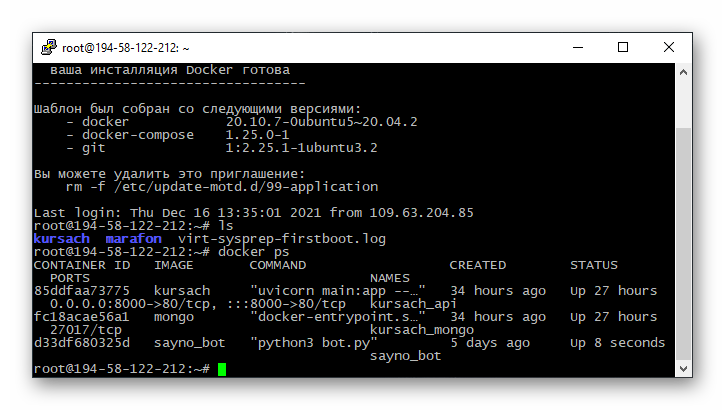


Рисунок 14 – Подключение к серверу

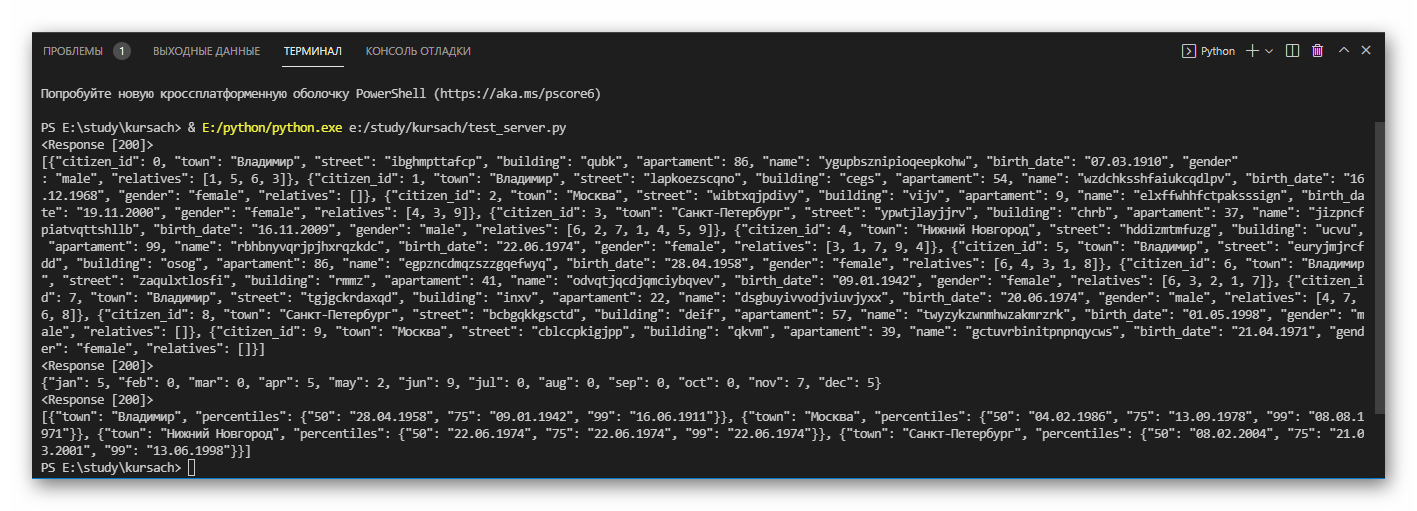


Рисунок 15 – Результаты тестов

Вывод: я разработал REST API-сервис посредством языка Python, протестировал, упаковал его в Docker-контейнер, а также развернул его на сервере.