**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)

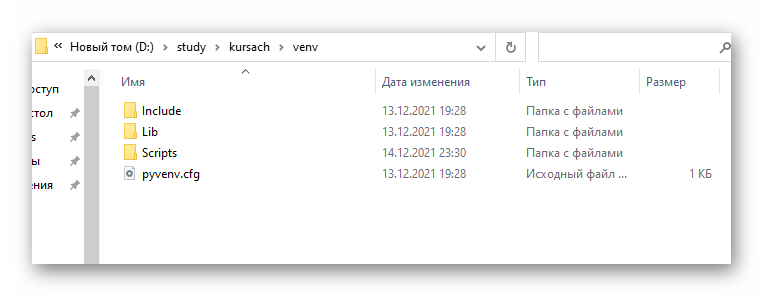


Рисунок 1 – Папка виртуального окружения

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

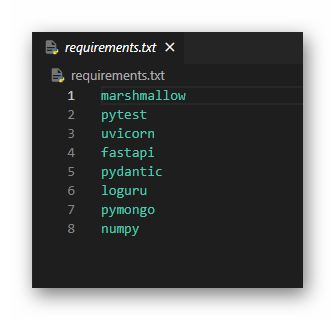


Рисунок 2 – requirements.txt

Для проекта используем следующую структуру (рисунок 3):

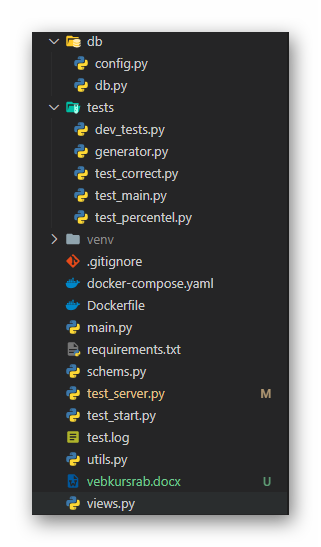


Рисунок 3 – Структура проекта

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.

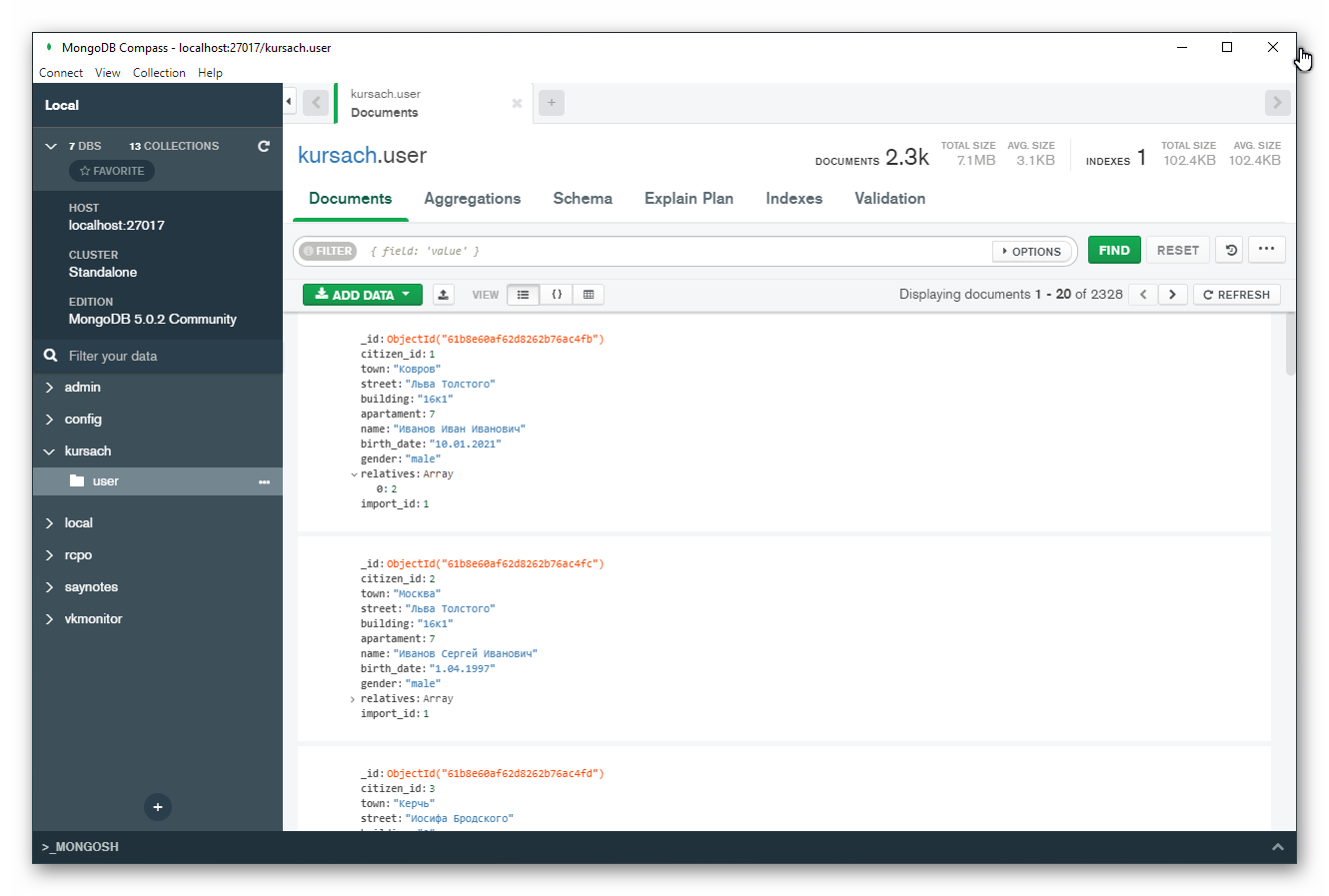
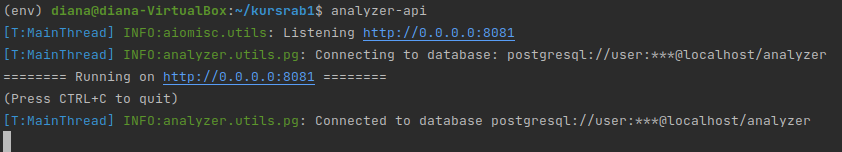


Рисунок 4 – База данных

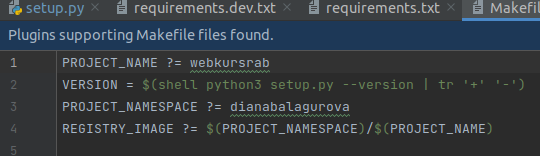
В качестве веб фреймворка был выбран ассинхронный fastapi, так как асинхронный подход позволяет эффективно обслуживать нескольких клиентов в рамках одного процесса ОС и так как у fastapi хорошая англоязычная документация в интернете.

Команда analyzer-api (рисунок 12):



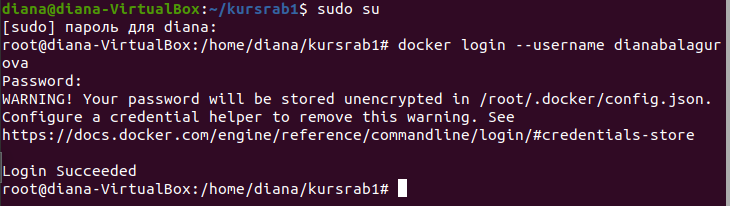
1. Analyzer-api

Изменим файл makefile (рисунок 13):



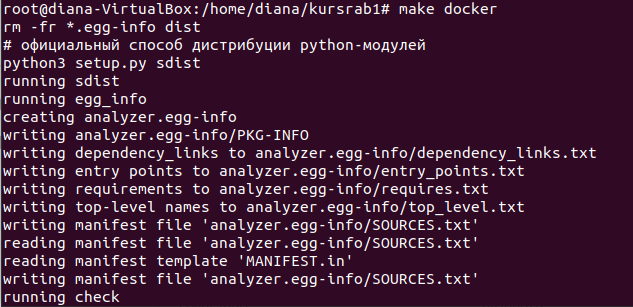
1. Makefile

Войдем в docker (рисунок 14):



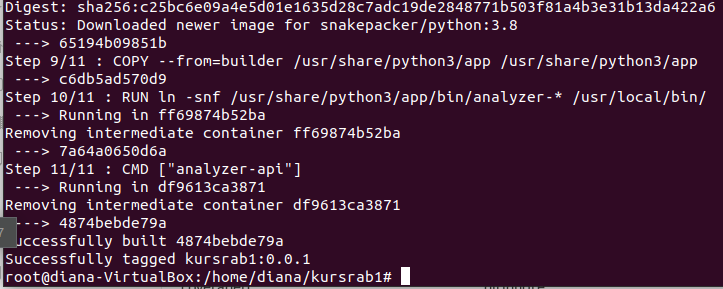
1. Вход

Применим команду make docker(рисунок 15):



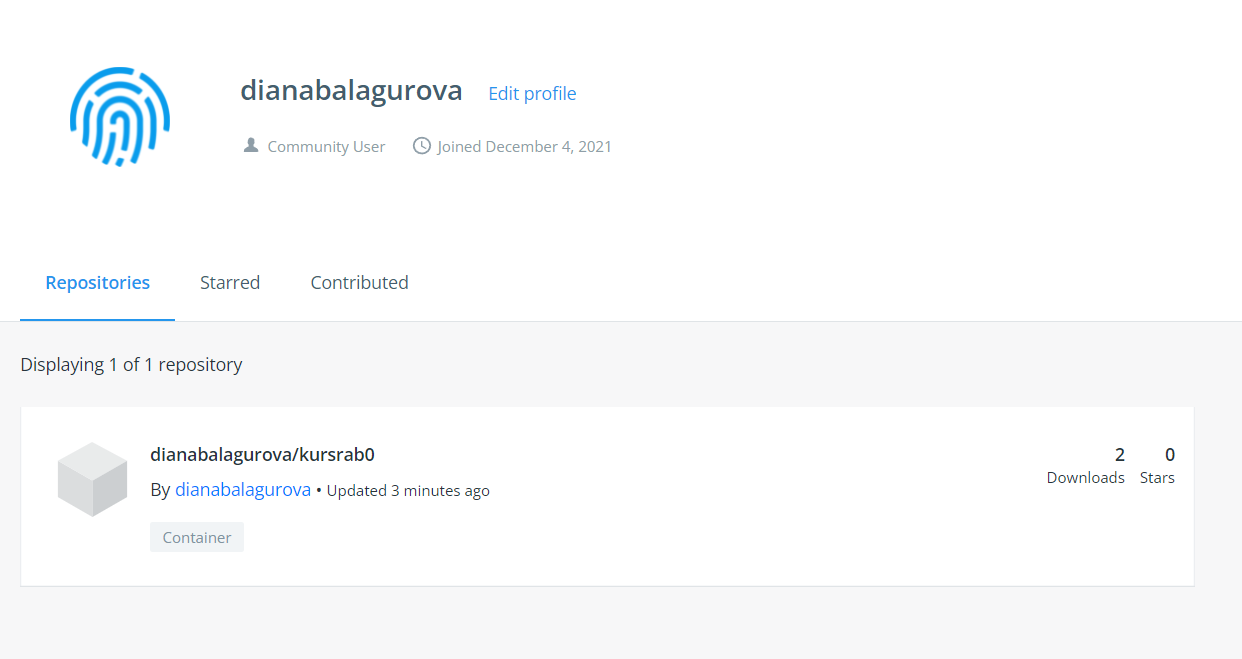
1. Make docker

Применим команду make upload (рисунок 16):



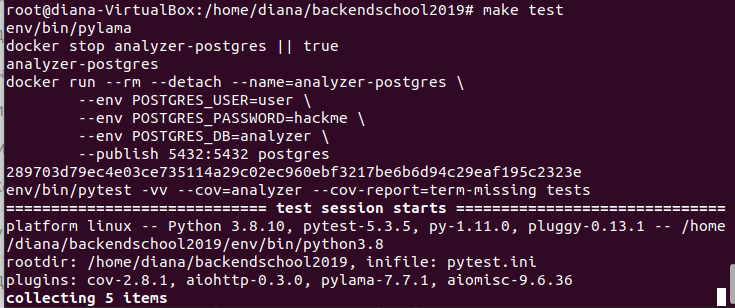
1. Make upload

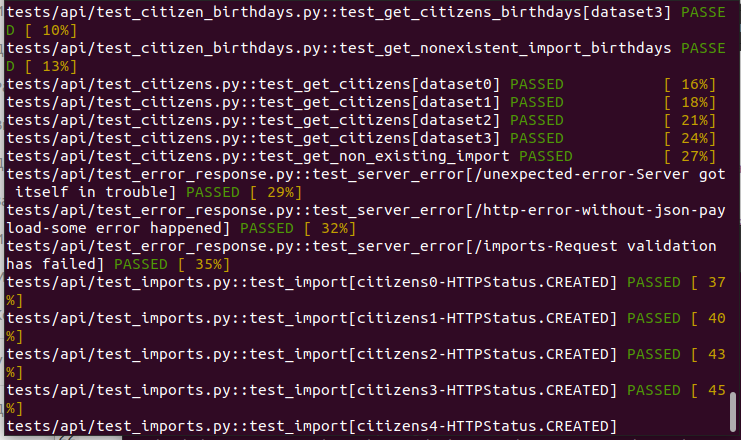
Убедимся в наличии репозитория в docker (рисунок 17):

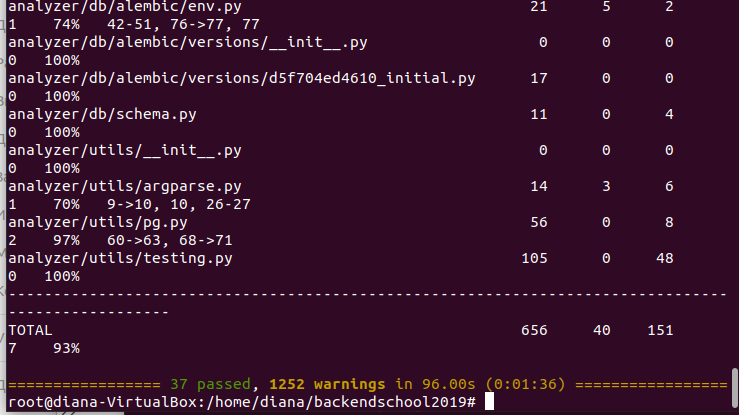


1. Docker

Запустим тестирование командой make test (рисунок 18):

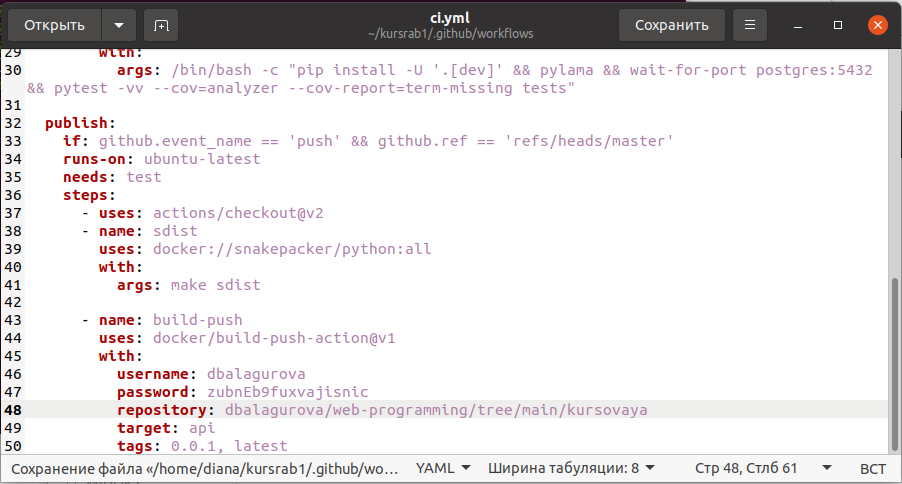






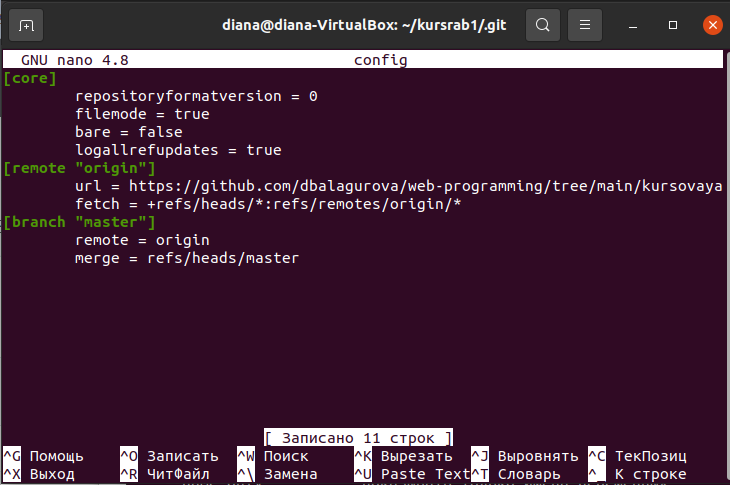
1. Тестирование

Запишем в файл ci.yml свои данные для входа в docker (рисунок 19):



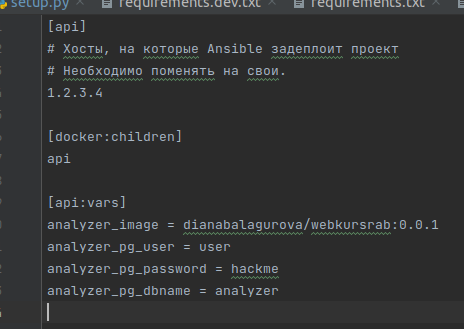
1. Ci.yml

Запишем в файл .git/config (рисунок 20):



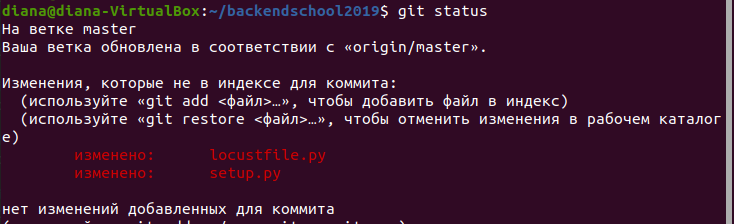
1. Config

В файле deploy/hosts.ini пропишем свой репозиторий (рисунок 21):



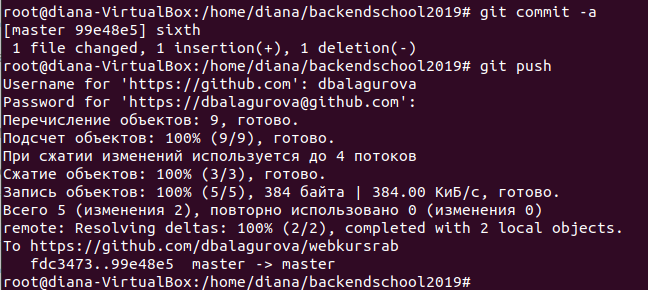
1. Hosts

Проверим статус изменений командой git status (рисунок 22):



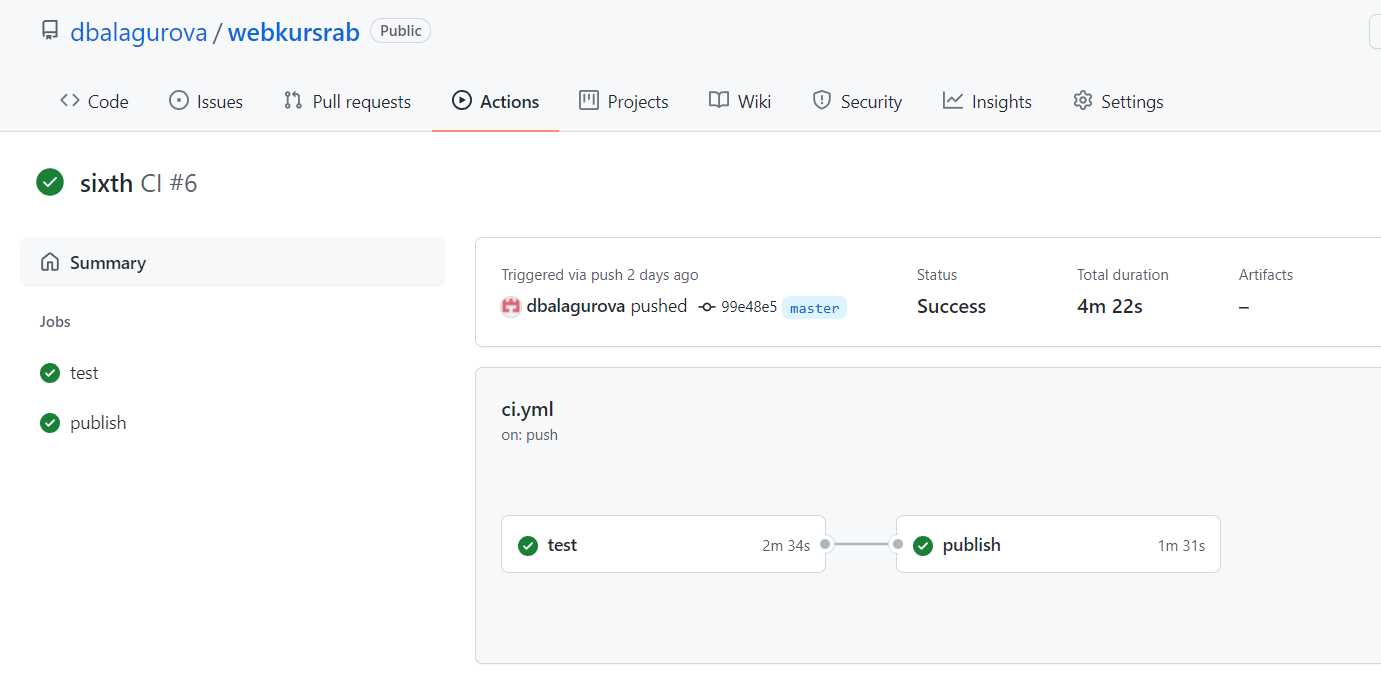
1. Статус

Создадим коммит командой git commit –a и опубликуем изменения командой git push (рисунок 23):



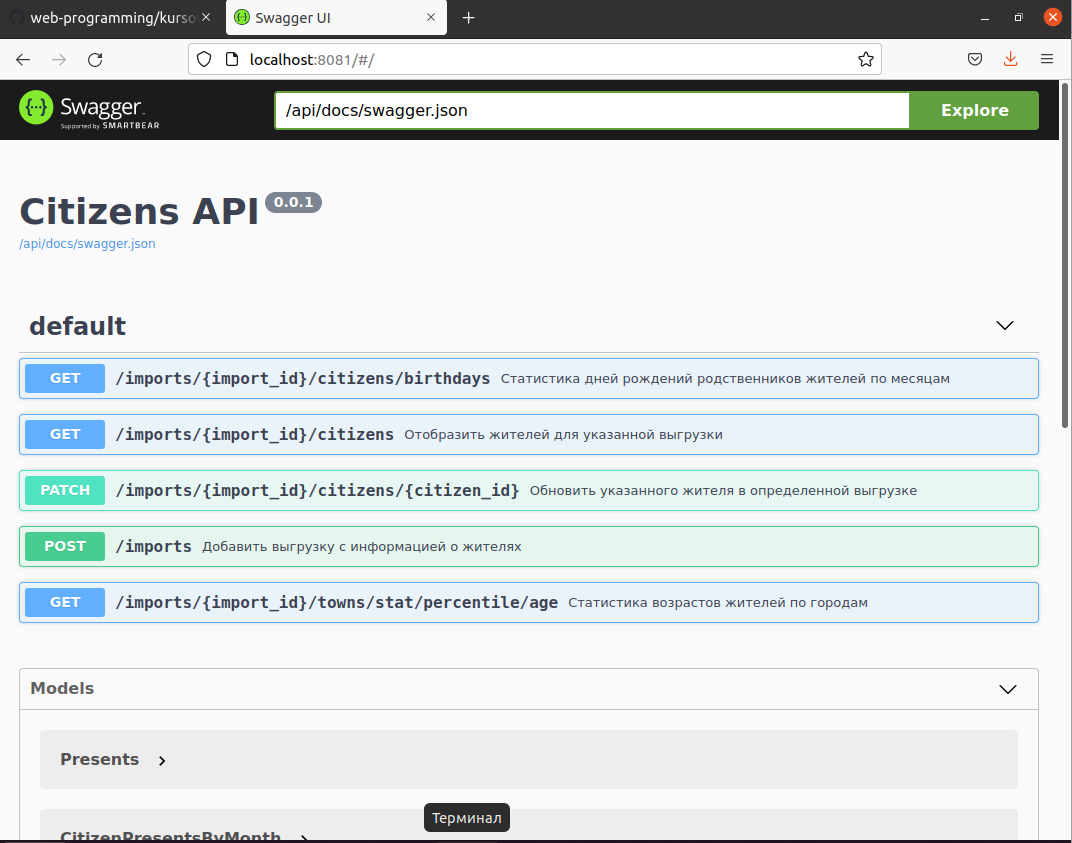
1. Коммит

Убедимся в github что изменения успешно загружены (рисунок 24):



1. Github

Теперь нам доступен интерфейс по адресу localhost:8081 (рисунок 25):



1. Интерфейс

Вывод: я разработала REST API-сервис посредством языка Python, протестировала, упаковала его в Docker-контейнер, а также развернула с помощью Ansible.