|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА– Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Инструментального и прикладного программного обеспечения

**ПРАКТИЧЕСКИЕ РАБОТЫ №11-12**

по дисциплине «Разработка серверных частей интернет-ресурсов»

**Тема практической работы: Python**

**Студент группы** ИКБО-03-21 Хречко Сергей Викторович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

**Руководитель практической работы** ассистент Благирев М.М.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Работа представлена «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Допущен к работе «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Москва 2023

**СОДЕРЖАНИЕ**

[1. Практическая работа №11 3](#_Toc9218)

[1.1. Теоретическая часть 3](#_Toc9338)

[1.2. Задание 7](#_Toc17965)

[1.3. Ход работы 8](#_Toc2481)

[1.4. Вывод 12](#_Toc20351)

[2. Практическая работа №12 13](#_Toc28279)

[2.1. Задание 13](#_Toc7190)

[2.2. Ход работы 14](#_Toc8340)

[2.3. Вывод 20](#_Toc9270)

[2.4. Ответы на контрольные вопросы 21](#_Toc31345)

[2.5. Использованная литература 23](#_Toc22286)

# Практическая работа №11

## Теоретическая часть

### Python

Python код может выполняться разными способами. Можно написать файл c расширением .py, который бы содержал некий скриптовый код, далее его можно исполнить командой python, например, мы находимся в некотором каталоге где лежит main.py в котором есть код:

Листинг 1 – код примера

**print("Helloworld")**

Далее необходимо запустить код командой:

Листинг 2 – запуск кода на Python

D:\RSCHIR>pythonmain.py

Hello world

PS D:\RSCHIR>

Но такой подход обычно используется простыми программами или скриптами и использует глобальную виртуальную среду интерпретатора Python. В этой среде обычно определены пакеты, получаемые с помощью пакетного менеджера python – pip. pip берет пакеты из собственного удаленного репозитория пакетов [PyPi](https://pypi.org/). Эти пакеты представляют собой zip архивы которые содержат в себе код Python, некоторую метаинформацию, определяющую зависимости библиотеки и тому подобные вещи.

Соответственно, пакеты, которые будут стоять глобально, могут быть использованы всеми программами, которые выполняются глобально как было описано выше, но проблема в том, что пакеты постоянно обновляются, выходят новые патчи, исправления и, если устанавливать эти обновления глобально, какие-то из проектов могут перестать корректно запускаться, не только из-за несовместимости версий используемых пакетов в конкретных проектах, но и из-за несовместимой версии самого интерпретатора Python.

Поэтому был придуман механизм изоляции приложений, написанных на языке Python – это создание отдельных виртуальных сред интерпретатора Pythonvenv. Такой подход позволяет использовать только те зависимости, которые нужны проекту. Можно сказать, что это своя папка для каждого проекта на Python. Такой подход является наиболее рекомендуемым.

### Django

Таким пакетом и является Django. Чтобы его установить необходимо создать виртуальную среду интерпретатора Python. Для начала воспользуемся командной строкой и создадим рабочую папку и создадим в ней виртуальную среду

Листинг 3 – Создание виртуальной среды и ее активация

PS D:\RSCHIR\helloworld> python -m venv .venv

PS D:\RSCHIR\helloworld> ./.venv/Scripts/activate.bat

(.venv) PS D:\RSCHIR\helloworld>

Далее нужно скачать пакет Django. Делается это с помощью pip

Листинг 4 – Установка пакета

(.venv) PS D:\RSCHIR\helloworld> pip install django

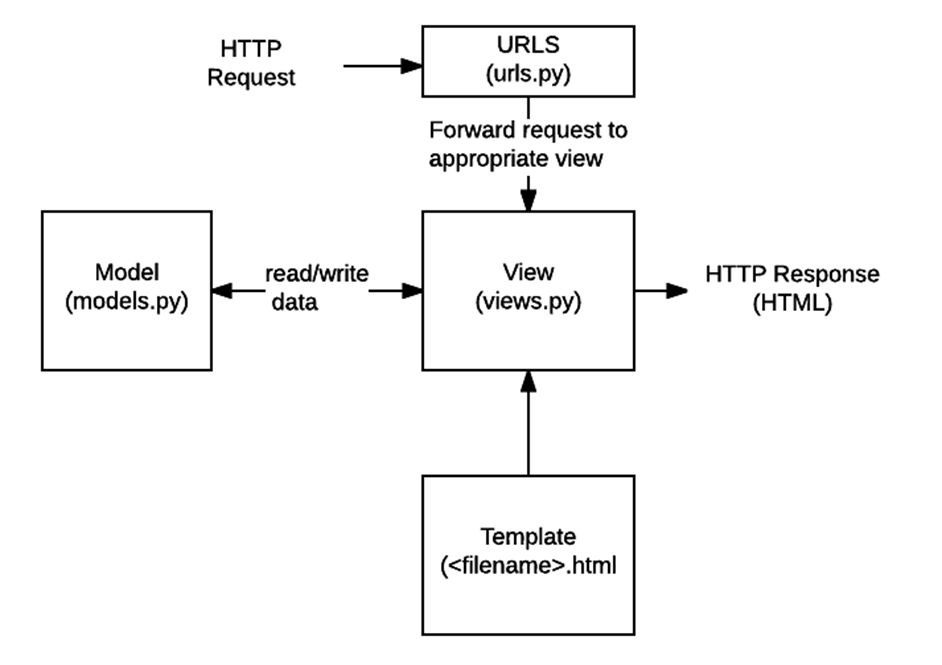
Django установлен последней возможной версии. Далее необходимо создать проект Django, делается это с помощью утилиты django-admin, которая входит в поставку пакета django.

Листинг 5 – Созданиепроекта

(.venv) PS D:\RSCHIR\helloworld> django-admin startproject example

Описание файловой архитектуры созданного проекта

Файловая архитектура организации джанго (стандартная) приведена на рисунке 1.



1. Файловая архитектура организации джанго (стандартная)

### Библиотеки

Python имеет множество библиотек и фреймворков, созданных специально для анализа данных. NumPy, Pandas, Matplotlib и SciPy предоставляют мощные инструменты для работы с числовыми данными, выполнения операций с массивами, обработки и визуализации данных.

NumPy

NumPy это open-source модуль для python, который предоставляет общие математические и числовые операции в виде пре-скомпилированных, быстрых функций. Они объединяются в высокоуровневые пакеты. Они обеспечивают функционал, который можно сравнить с функционалом MatLab. NumPy (Numeric Python) предоставляет базовые методы для манипуляции с большими массивами и матрицами.

SciPy

SciPy (Scientific Python) расширяет функционал numpy огромной коллекцией полезных алгоритмов, таких как минимизация, преобразование Фурье, регрессия, и другие прикладные математические техники.

Matplotlib

Библиотека на языке программирования Python для визуализации данных двумерной и трёхмерной графикой. Получаемые изображения могут быть использованы в качестве иллюстраций в публикациях. Matplotlib написан и поддерживался в основном Джоном Хантером и распространяется на условиях BSD-подобной лицензии.

### Описание файлов приложения

**URLs**: Хотя можно обрабатывать запросы с каждого URL-адреса с помощью одной функции, гораздо удобнее писать отдельную функцию для обработки каждого ресурса. URL-mapper используется для перенаправления HTTP-запросов в соответствующее представление на основе URL-адреса запроса. URL-mapper также может извлекать данные из URL-адреса в соответствии с заданным шаблоном и передавать их в соответствующую функцию в виде аргументов.

**View:** Представление (view) - это функция обработчика запросов, которая получает HTTP-запросы и возвращает ответы. View имеет доступ к данным через модели (необходимым для удовлетворения запросов и делегирования ответа в шаблоны).

**Models:** Модели представляют собой объекты Python, которые определяют структуру данных приложения и предоставляют механизмы для управления (добавления, изменения, удаления) и выполнения запросов в базу данных.

**Templates:** Template (шаблон) - это текстовый файл определяющий структуру или разметку страницы (например HTML страницы), с полями для подстановки используемыми для представления актуального содержимого. View может динамически создавать HTML страницы, используя HTML шаблоны и заполняя их данными из модели (model). Шаблон может быть использован для определения структуры файлов любых типов, не обязательно HTML.

## Задание

Вам предлагается написать простое веб приложение Hello world, которое должно получать имя пользователя и выводить на экран, “Hello, {полученное имя}”. Ваша фамилия обязательно должна фигурировать в пути к приложению (localhost:8080/ivanov). Обязательно обработать случай, когда пользователь не передал имя.

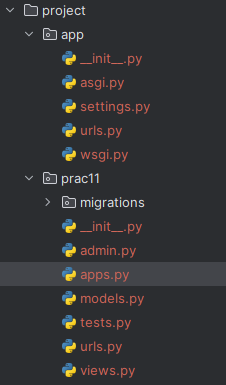
Код обязательно должен быть написан на языке Python c использованием фреймворка Django.

Код обязательно должен иметь отдельную venv Python, и должен быть запущен в контейнере Docker.

## Ход работы

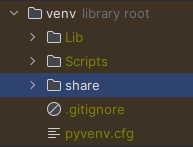
### Создание проекта

Создадим django-проект командой «django-admin startproject app», после чего создадим приложение командой «python manage.py startapp Pr11». Утилита автоматически создаст большую часть необходимых файлов, рисунок 2.



1. Структура файлов проекта

Создадим venv для проекта и установим туда необходимые библиотеки, рисунок 3.



1. Директория созданной venv

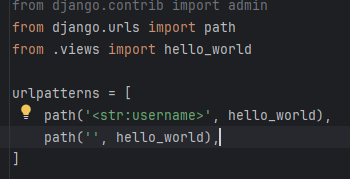
### Создание эндпоинта

Опишем в файле views.py функцию, возвращающую требуемую строку, рисунок 4.



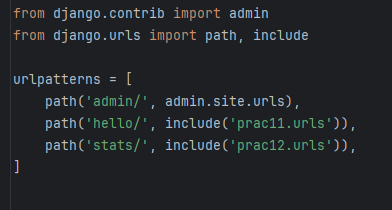
1. Файл views.py

В файле urls.py зададим соответствие пути – обработчику, рисунок 6.



1. Файл urls.py

В файле urls.py проекта подключим обработчики из нашего приложения, рисунок 6.

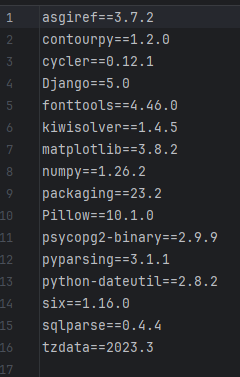


1. Файл urls.py проекта

### Развёртывание

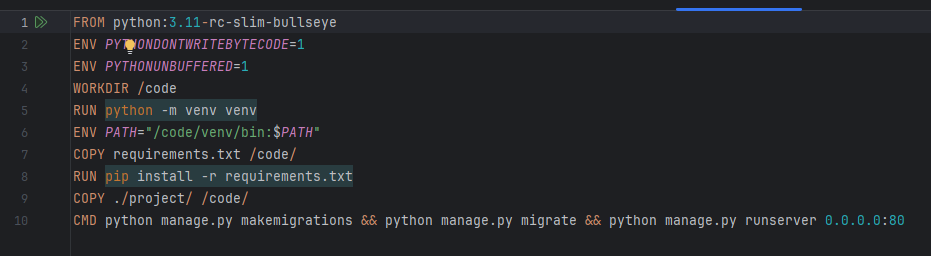
Т.к. реализовывать функционал следующей работы будем в том же проекте, заранее настроим развёртывание так, чтобы его было достаточно и для её функционала.

Создадим файл requirement.txt и заполним его с помощью команды “pip freeze > requirements.txt”. Содержимое файла показано на рисунке 7.



1. Содержимое файла requirements.txt

Опишем dockerfile и docker-compose файл для нашего приложения, рисунки 8 и 9.



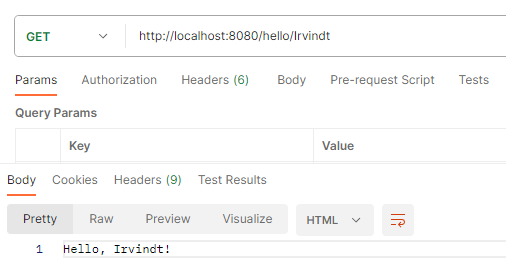
1. Dockerfile



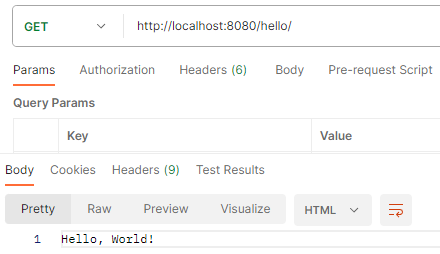
1. Docker-compose файл

### Тестирование

Протестируем приложение, отправив два запроса: с указанием имени и без указания, рисунки 10 и 11.



1. Запрос с указанием имени



1. Запрос без указания имени

## Вывод

В ходе данной работы был реализован простейший web-сервер на python с использованием django, способный обслуживать один эндпоит и возвращать в ответ на запрос динамически генерируемый контент.

# Практическая работа №12

## Задание

Вам предлагается реализовать простой сервис аналитики, которому бы на вход поступали данные из предыдущих практик (Java и Go), на выходе сервис выдает картинку-диаграмму. Картинки обязательно сохранять в БД(Рекомендуемая – Postgres), и при получении данных которые уже были, использовать график, который уже был сгенерирован.

Необходимо реализовать сервис на основе предыдущей работы с использованием REST.

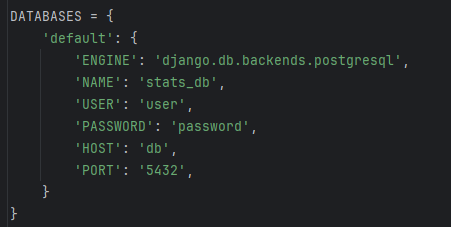
Сервис обязательно должен работать на Docker контейнере, и написан на Python Django с объединением с предыдущими работами. Приветствуется использование схожих библиотек, описанных в введении для создания картинок и анализа данных.

## Ход работы

### Настройка приложения

Создадим новое приложения с использованием команды “python manage.py startapp prac12”.

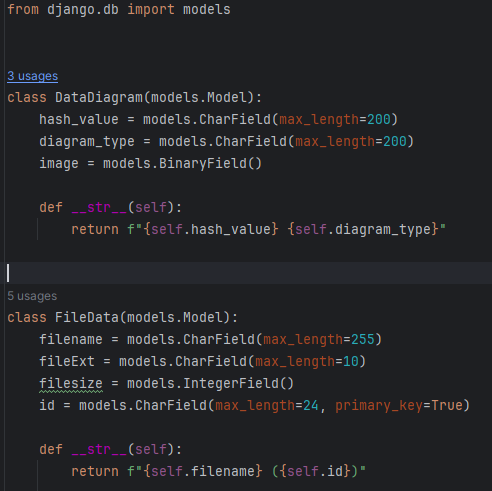
Настроим подключение к базе данных в файле settings.py, рисунок 12.



1. Настройки для подключения базы данных

### Настройка БД

Опишем модели для базы данных: модель записи о файле и модель для хранения диаграммы, рисунок 13.



1. Модели базы данных

Создадим и применим миграцию командами “python manage.py makemigrations” и “python manage.py migrate” соответственно.

### Создание эндпоинтов

Для выполнения задания требуется два эндпоинта: для отправки данных и для получения графика.

В обработчике получения данных парсим тело запроса, проверяем, что все необходимые поля на месте и сохраняем информацию в базу данных. В случае какой-либо ошибки вернём сообщение с ошибкой. Код обработчика показан на рисунке 14.



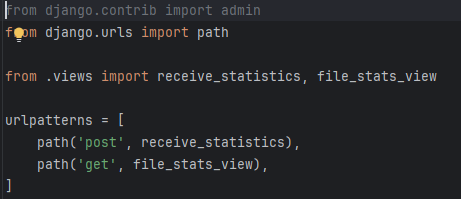
1. Обработчик получения данных

В обработчике, который должен возвращать график, получаем из базы данных информацию о записях. После этого вычисляем хэш от полученного списка и проверяем, существует ли уже сгенерированное изображение. Если изображение существует – вернём его, иначе сгенерируем новый график, сохраним его в базу данных а также отправим его в ответе, рисунок 15.



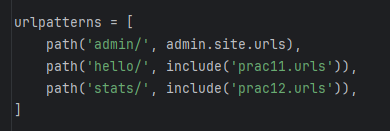
1. Обработчик отправки графика

Укажем в файле urls.py пути, на которые должны откликаться обработчики, рисунок 16.



1. Файлurls.py

Аналогично добавим в файл urls.py основного приложения импорт путей из приложения, рисунок 17.

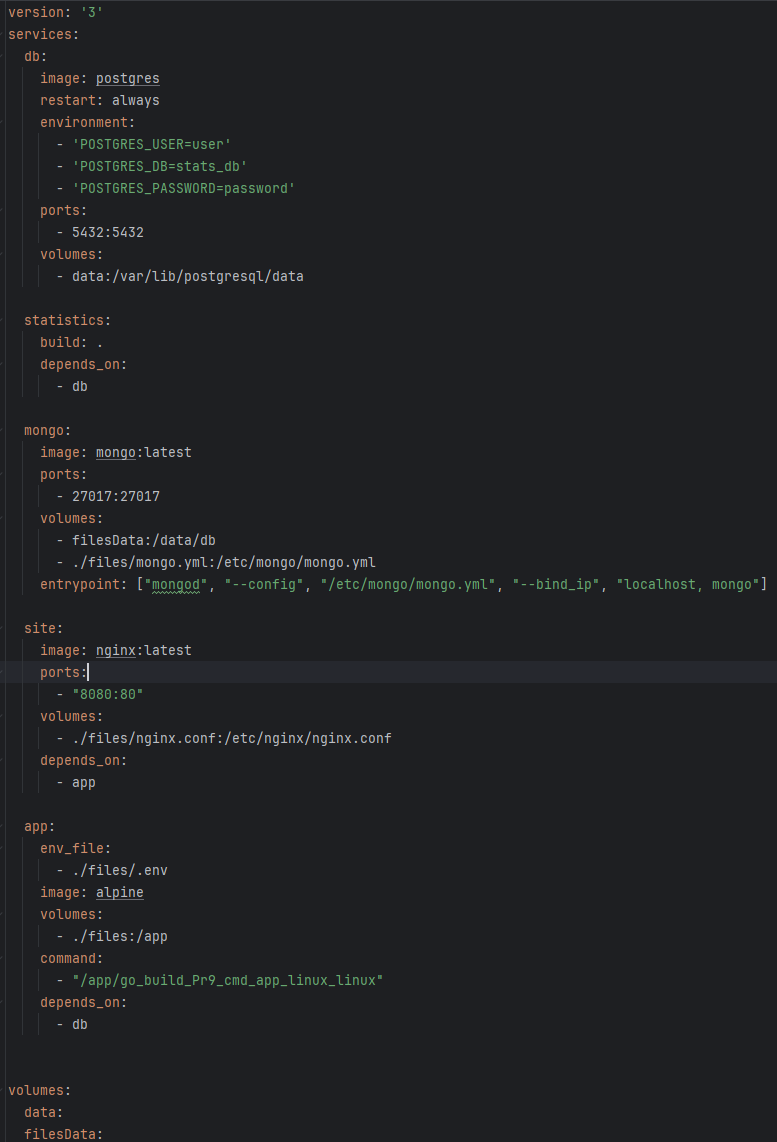


1. Обновлённый файл urls.py приложения

Дополнение микросервиса, который отправляет статистические данные оставим за скобками, так как это не относится напрямую к текущей работе.

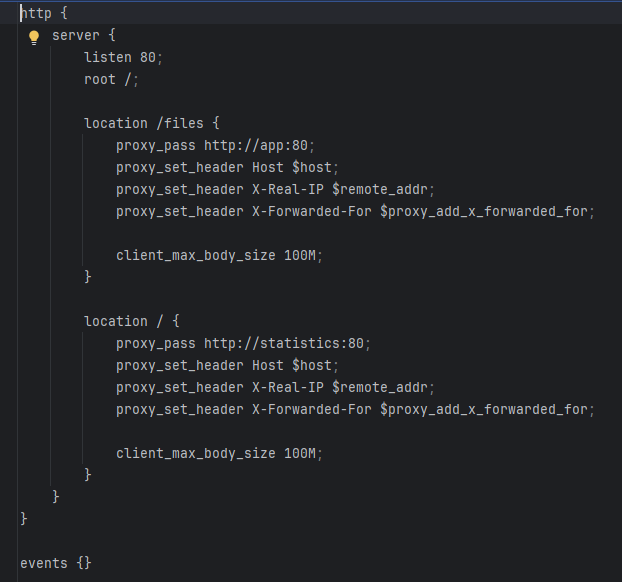
### Развёртывание

Дополним docker-compose файл, добавив туда сервис с mongoDB, которая хранит файлы, сервис с приложением, обрабатывающим загрузку и получение файлов, а также сервис с Nginx для проксирования запросов и распределения их между микросервисами, рисунок 18.



1. Дополнение в docker-compose файл

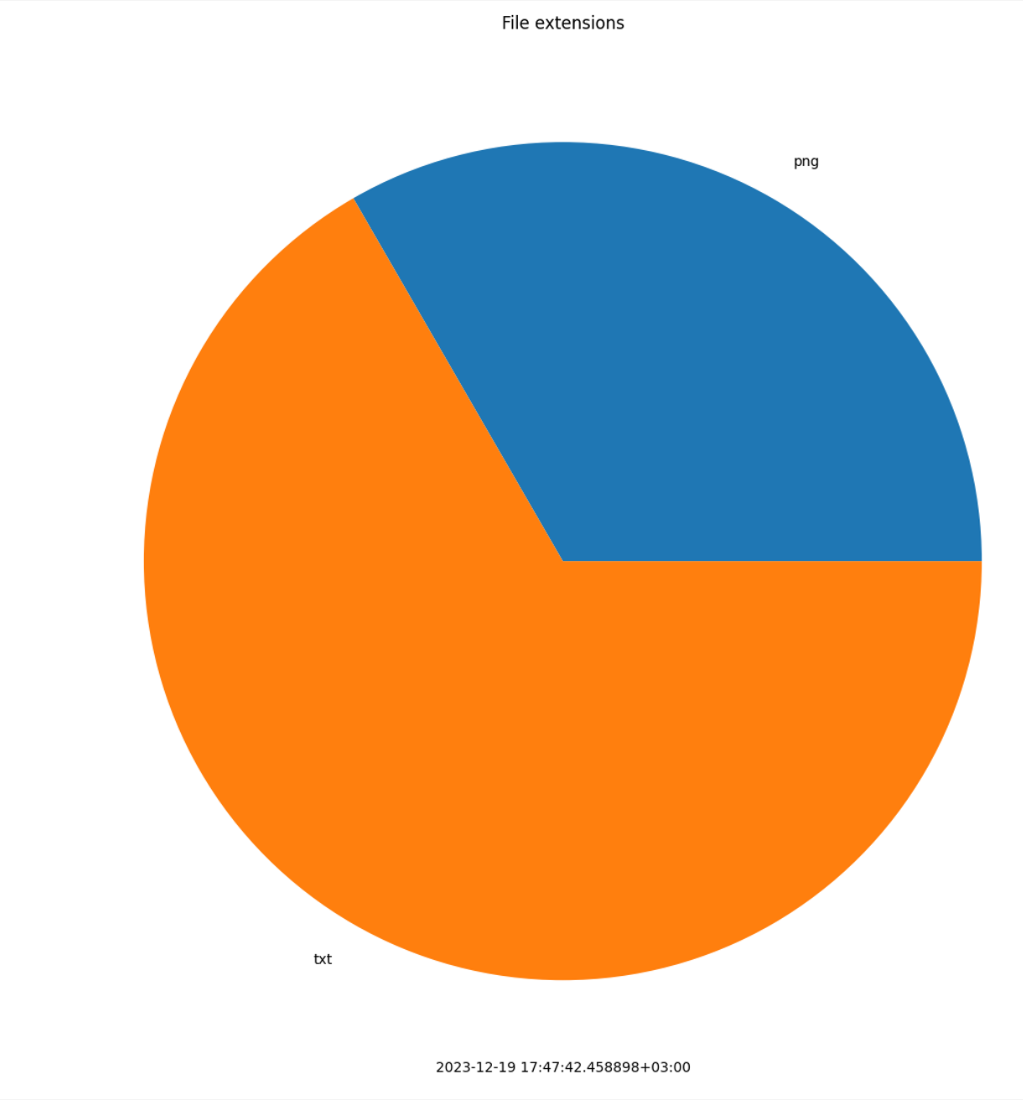
Опишем файл конфигурации nginx, рисунок 19.



1. Файл конфигурации nginx

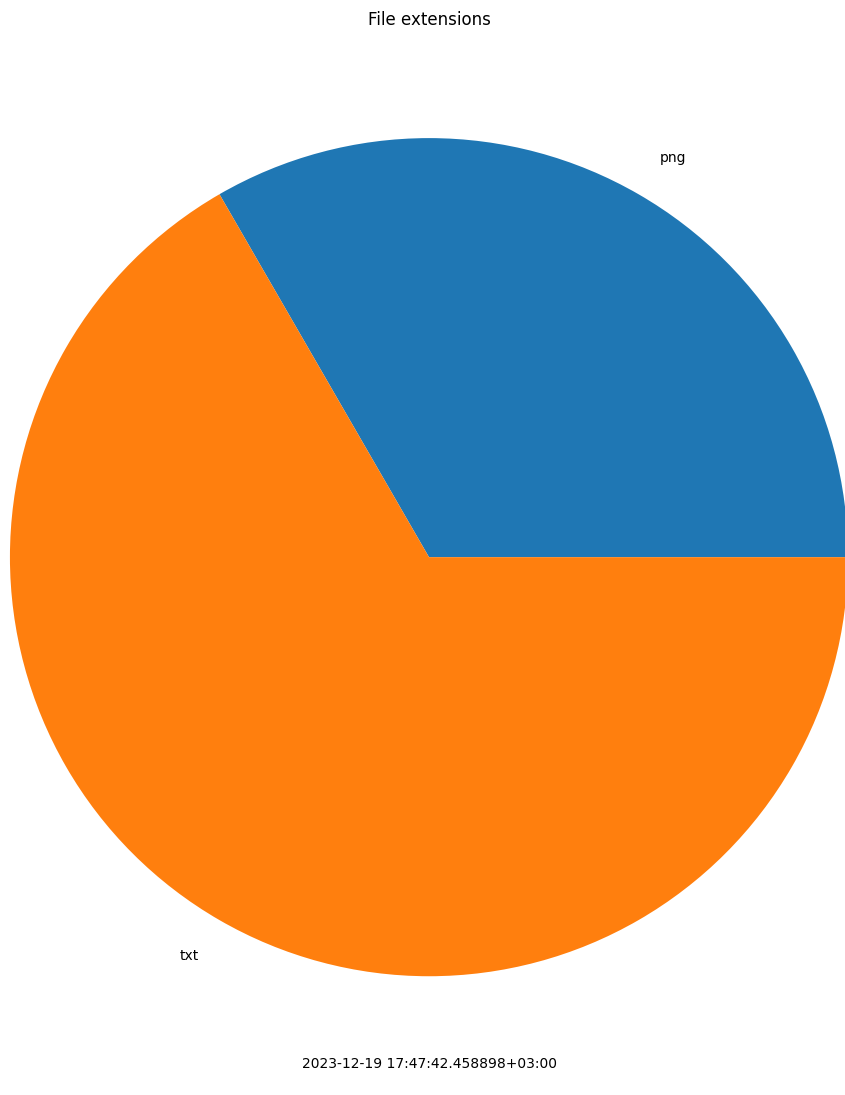
### Тестирование

Отправим GET запрос на адрес “/stats/get” и получим график, рисунок 20.



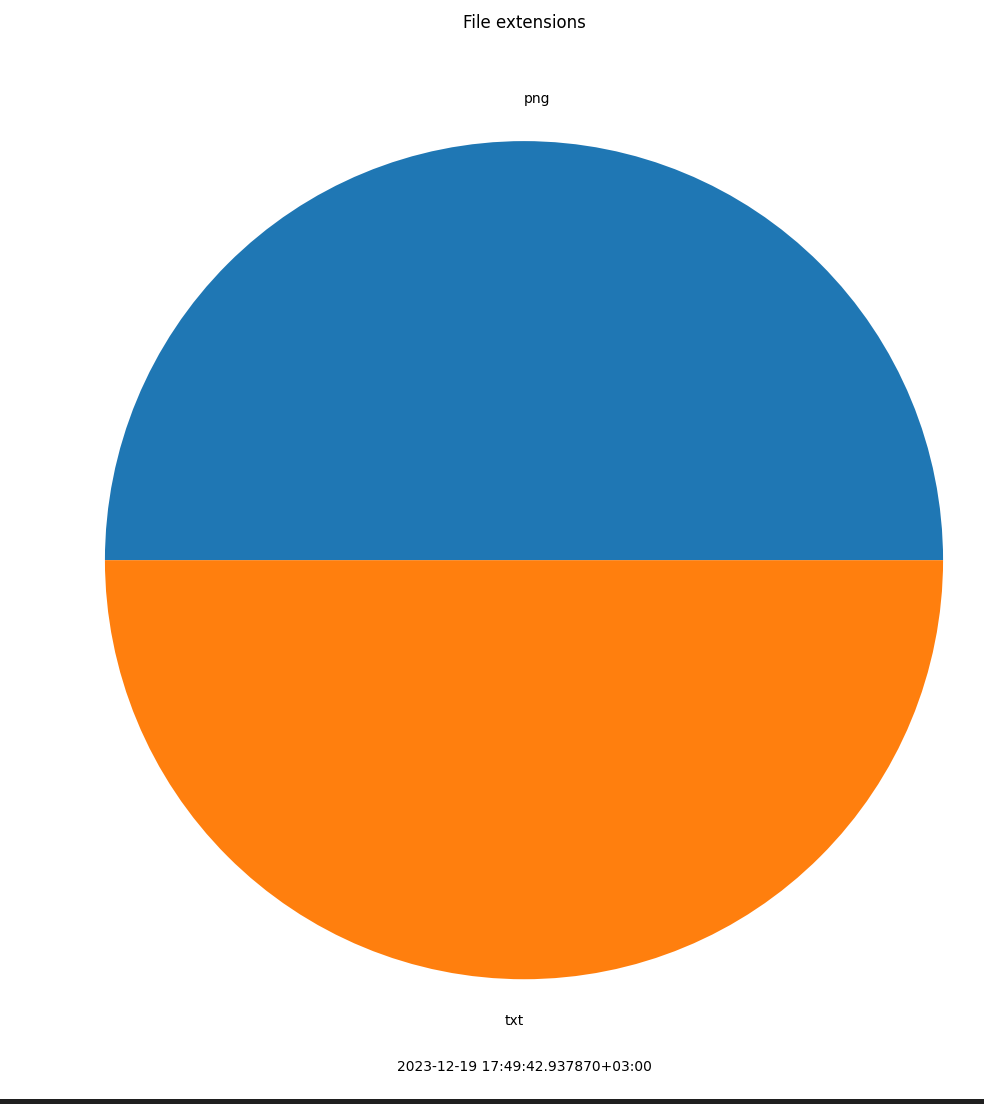
1. Полученный график

Повторно получив график убедимся, что нам вернулась ранее сгенерированная картинка – это видно по запечатанной в изображение временной метке, рисунок 21.



1. Повторно полученный график

Загрузим новый файл с расширением “.png”, после чего заново получим график. Видим, что график изменился в соответствии с новой статистикой, рисунок 22.



1. Обновлённый график

## Вывод

В ходе данной работы был разработан простейший микросервис статистики, способный получать и аккумулировать информацию, а также представлять собранную статистику в виде графиков, генерируемых на стороне сервера.

## Ответы на контрольные вопросы

### Из каких частей состоит обычный проект на Django?

Модели (Models): Определяют структуру базы данных.

Отображения (Views): Обрабатывают запросы и возвращают ответы.

Шаблоны (Templates): Определяют структуру и внешний вид HTML-страниц.

URL-шаблоны (URLs): Определяют маршруты для обработки запросов.

### Что такое Отображения (views) в рамках Django?

Отображения обрабатывают HTTP-запросы. Они принимают запрос, выполняют необходимые действия и возвращают HTTP-ответ.

### Что такое модель в рамках Django?

Модель определяет структуру данных в базе данных.

Она описывает поля, типы данных и отношения между данными.

### Что нужно сделать чтобы развернуть Django приложение внутри Docker?

Создать Dockerfile, определить зависимости и установить их.

Определить порты, которые приложение использует.

Запустить приложение в контейнере с помощью команды docker run.

### Какие преимущества предоставляет Django для разработки аналитических сервисов?

Удобная работа с базой данных через ORM, лёгкость создания динамических веб-страниц, готовая админ-панель для управления данными.

### Как можно визуализировать аналитические данные в Django-приложении?

Использование библиотек для графиков, таких как matplotlib или plotly.

Интеграция JavaScript-библиотек для визуализации на стороне клиента.

### Зачем нужна библиотека matplotlib?

Библиотека matplotlib предназначнена для создания различных типов графиков и диаграмм в Python. Применяется для визуализации данных и для построения графиков для анализа.

### Какие действия нужно произвести чтобы Django мог работать с Postgres?

Установить библиотеку psycopg2 для взаимодействия с PostgreSQL.

В файле настроек (settings.py) указать параметры подключения к базе данных PostgreSQL, такие как имя базы данных, пользователя и пароль.

## Использованная литература

1. Оф. документация Django - <https://docs.djangoproject.com/en/4.2/>
2. RESTful на Django - <https://habr.com/ru/articles/722054/>
3. Views в Django - <https://ru.hexlet.io/courses/python-django-basics/lessons/views/theory_unit>
4. Установка Python - <https://devpractice.ru/python-lesson-1-install/>
5. Docker - <https://habr.com/ru/articles/253877/>
6. Docker compose - <https://habr.com/ru/companies/ruvds/articles/450312/>
7. <https://habr.com/ru/articles/352678/> - NumPy
8. <https://pythonworld.ru/novosti-mira-python/scientific-graphics-in-python.html> - Matplotlib
9. <https://django.fun/ru/articles/tutorials/sozdajte-rest-api-za-30-minut-s-pomoshyu-django-rest-framework/> - REST на Django
10. <https://habr.com/ru/articles/729610/> - Posgres, Django, Docker
11. <https://blog.skillfactory.ru/glossary/pandas/> - Pandas