МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Кафедра вычислительной математики и программирования

Курсовой проект по курсу «Программная инженерия»

Выполнила: Литвина А.А.
Группа: М8О-109М-21
Преподаватель: Кухтичев А.А.

Москва, 2022

Задание курсового проекта

Задача: Требуется разработать современное веб-приложение. Основные требования:

• Поднять веб-сервер (nginx). Отдавать статические файлы (логотип и т.д.) по location’у /static/. Настроить проксирование запросов на сервер-приложений по отдельному URL;
• В конфиге nginx создать location, которое будет ходить на Django-приложение;
• Поднять сервер-приложений.
• Создать базу данных в PostgreSQL; Написать классы-модели, мигрировать;
• Организовать приём и передачу сообщений с помощью формата JSON, используя REST.
• Реализовать метод API для загрузки файла, использовать для хранения файла облачное S3 хранилище, создать localtion в Nginx для раздачи загруженных файлов, реализовать обработчик в приложении для проверки прав доступа к файлу;
• Реализовать OAuth2-авторизацию для двух любых социальных сетей, навесить
декоратор, проверяющий авторизацию при вызовах API;
• Покрыть тестами все вьюхи и по желанию другие функции; Написать selenium-тест (найти элемент + клик на элемент); Использовать mock-объект при тестировании; Использовать factory boy; Узнать степень покрытия тестами с помощью библиотеки coverage;
• Развернуть и наполнить тестовыми данными Elasticsearch; Реализовать поиск по пользователям, продуктам (сущностям); Реализовать метод API для поиска по указанным сущностям и создать страничку HTML с вёрсткой для поиска и отображения результатов
• Установить и поднять centrifugo; Подключить centrifugo к проекту на стороне клиента и сервера; Организовать отправку/получение сообщений с помощью centrifugo.
• Установить docker и docker-compose; Создание Dockerfile для Django-приложения;
Создание docker-compose для проекта:
– nginx;
– База данных;

– Django-приложение;
– elasticsearch;
Создание Makefile для проекта.

Тема курсовой: Веб-приложение для достопримечательностей.
Вариант веб-сервера: nginx.
Вариант сервера-приложений: Django.
Вариант S3-хранилища: MCS.
Вариант базы данных: PostgreSQL

1. Веб-сервер

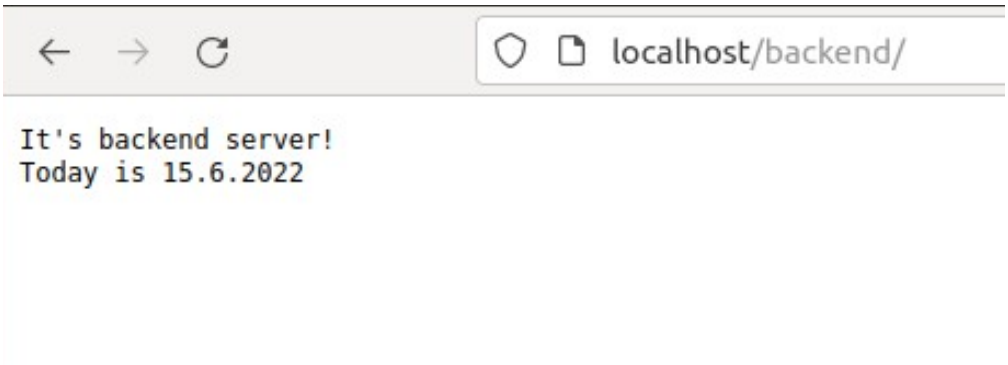
Nginx — веб-сервер и почтовый прокси-сервер, работающий на Unix-подобных операционных системах. Nginx позиционируется как простой, быстрый и надёжный сервер, не перегруженный функциями.
Применение nginx целесообразно прежде всего для статических веб-сайтов и как обратного прокси-сервера перед динамическими сайтами [1].

Адреса:

http://localhost:8000/ или http://localhost/backend/ - backend-сервер

http://localhost/public/index.html — отдача статических файлов

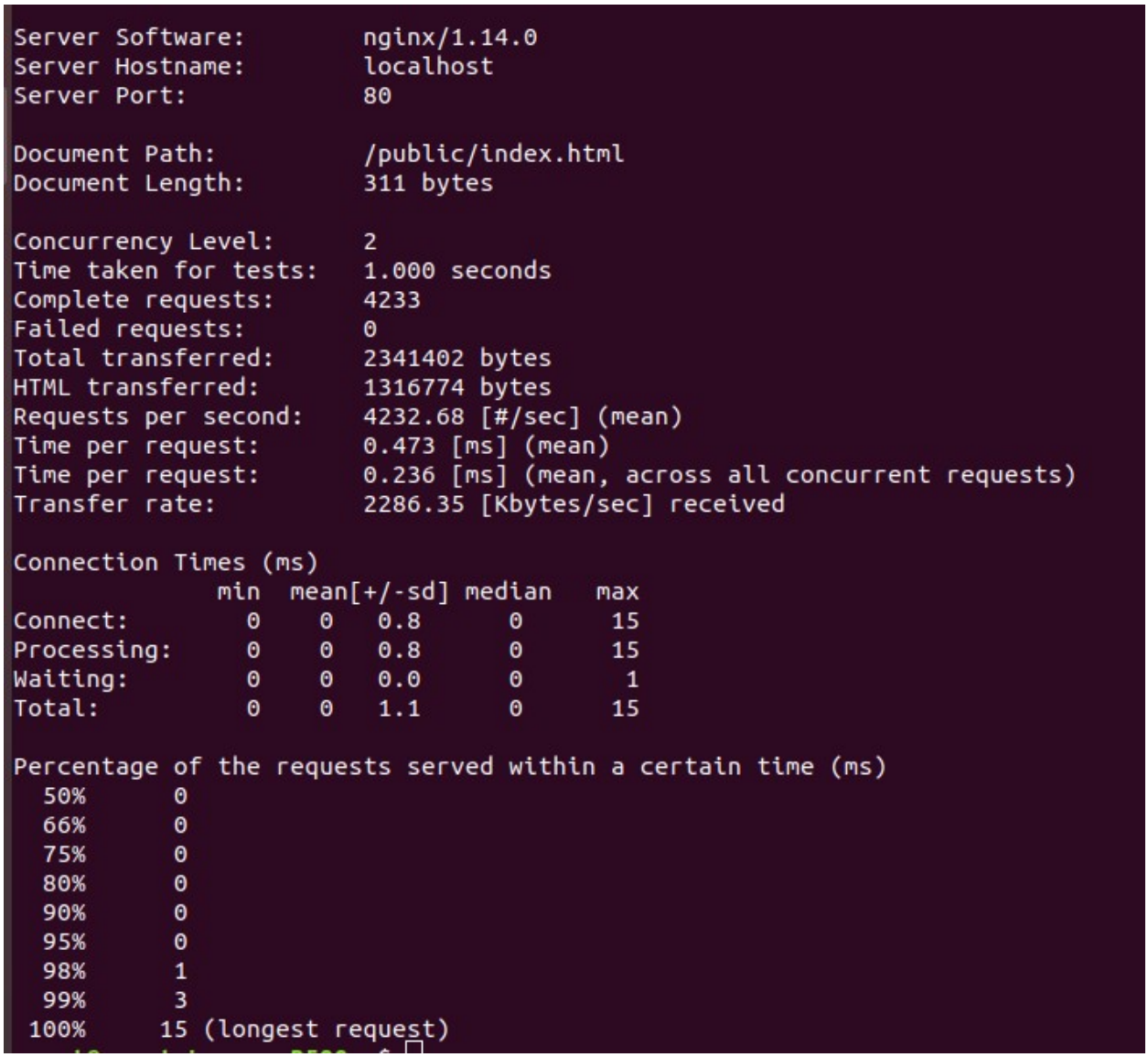
За отдачу статических файлов и проксирование запросов отвечает файл конфигурации nginx — default (находится по адресу /etc/nginx/sites-available).



Отдача статических файлов

Производительность:

1) Статические файлы
ab -n 10 -c 2 -t 1 -v 2 http://localhost/public/index.html



```
Server Software:        nginx/1.14.0
Server Hostname:        localhost
Server Port:            80

Document Path:          /public/index.html
Document Length:        311 bytes

Concurrency Level:      2
Time taken for tests:   1.000 seconds
Complete requests:      4233
Failed requests:        0
Total transferred:      2341402 bytes
HTML transferred:       1316774 bytes
Requests per second:    4232.68 [#/sec] (mean)
Time per request:       0.473 [ms] (mean)
Time per request:       0.236 [ms] (mean, across all concurrent requests)
Transfer rate:          2286.35 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.8      0      15
Processing:     0    0   0.8      0      15
Waiting:        0    0   0.0      0       1
Total:          0    0   1.1      0      15

Percentage of the requests served within a certain time (ms)
  50%      0
  66%      0
  75%      0
  80%      0
  90%      0
  95%      0
  98%      1
  99%      3
 100%     15 (longest request)
```

2) Backend

ab -n 10 -c 2 -t 1 -v 2 http://127.0.0.1:8000/

```
Server Software:        gunicorn/19.7.1
Server Hostname:        127.0.0.1
Server Port:            8000

Document Path:          /
Document Length:        38 bytes

Concurrency Level:      2
Time taken for tests:   1.000 seconds
Complete requests:      2454
Failed requests:        0
Total transferred:      451720 bytes
HTML transferred:       93290 bytes
Requests per second:    2453.67 [#/sec] (mean)
Time per request:       0.815 [ms] (mean)
Time per request:       0.408 [ms] (mean, across all concurrent requests)
Transfer rate:          441.07 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       5
Processing:     0    1   1.7      1      59
Waiting:        0    0   1.2      0      58
Total:          0    1   1.7      1      59

Percentage of the requests served within a certain time (ms)
  50%      1
  66%      1
  75%      1
  80%      1
  90%      1
  95%      1
  98%      3
  99%      3
 100%     59 (longest request)
```

3) Backend с помощью проксирования
ab -n 10 -c 2 -t 1 -v 2 http://localhost/backend/

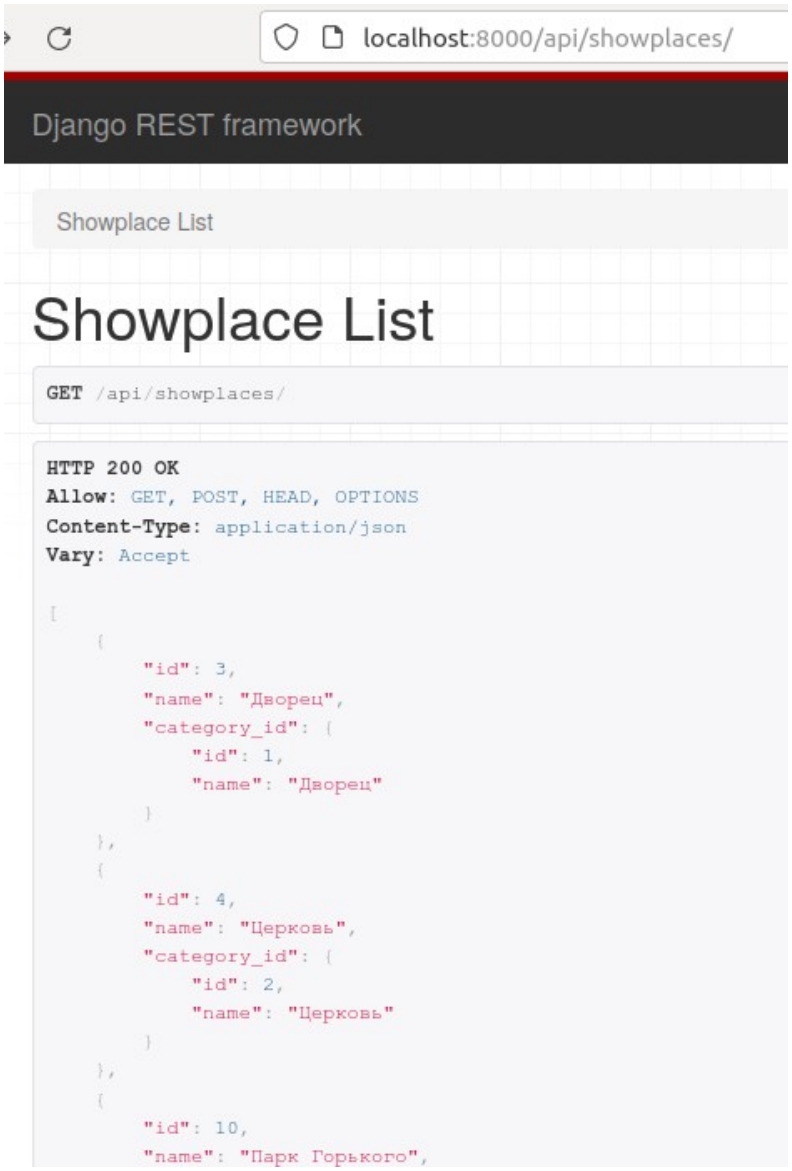
```
Server Software:        nginx/1.14.0
Server Hostname:        localhost
Server Port:            80

Document Path:          /backend/
Document Length:        38 bytes

Concurrency Level:      2
Time taken for tests:   1.000 seconds
Complete requests:      1831
Failed requests:        0
Total transferred:      347890 bytes
HTML transferred:       69578 bytes
Requests per second:    1830.98 [#/sec] (mean)
Time per request:       1.092 [ms] (mean)
Time per request:       0.546 [ms] (mean, across all concurrent requests)
Transfer rate:          339.73 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       3
Processing:     1    1   0.3      1       5
Waiting:        0    1   0.3      1       4
Total:          1    1   0.3      1       5

Percentage of the requests served within a certain time (ms)
  50%      1
  66%      1
  75%      1
  80%      1
  90%      1
  95%      1
  98%      2
  99%      3
 100%      5 (longest request)
```

2. Сервер приложений

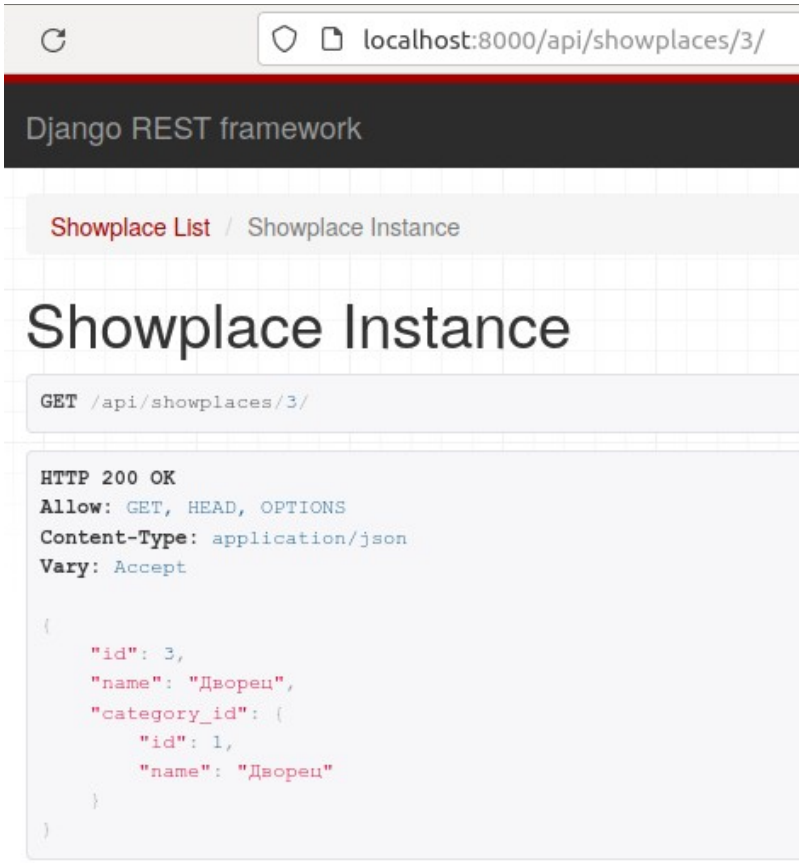
Django — свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC. Сайт на Django строится из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из существенных архитектурных отличий этого фреймворка от некоторых других. Один из основных принципов фреймворка — DRY (Don't repeat yourself) [2].

Приветственная страница приложения

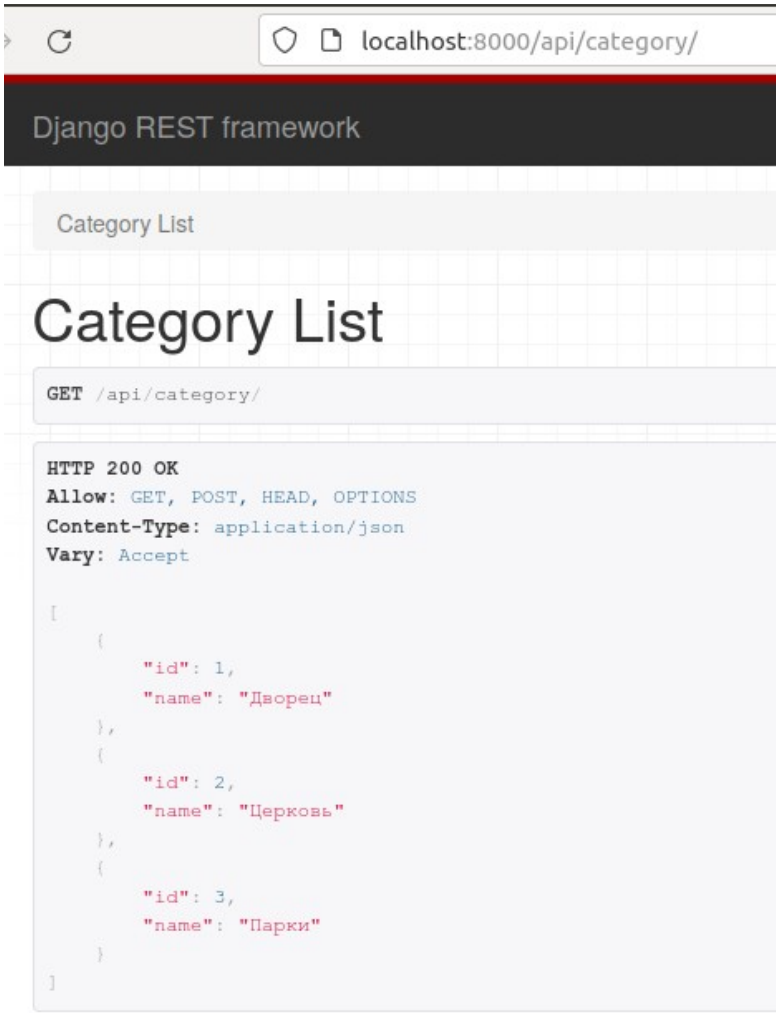
Список достопримечательностей



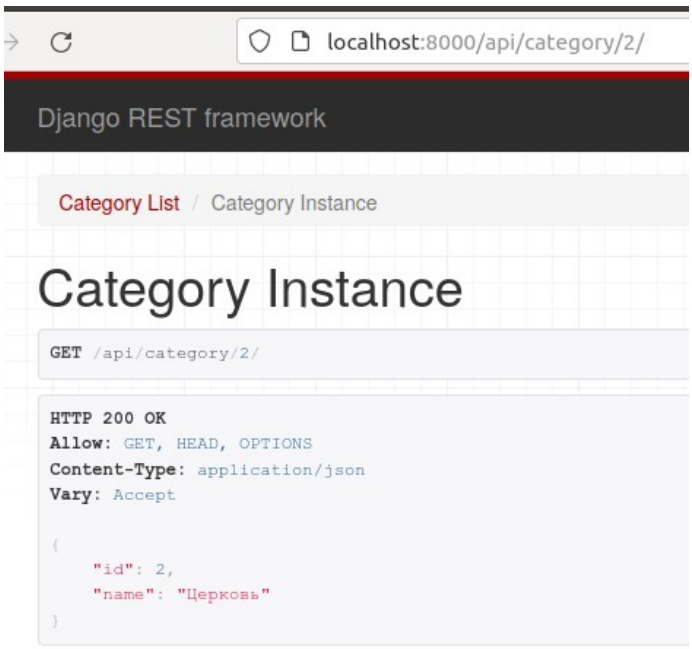
Достопримечательность с id=3



Список категорий



Категория с id=2



3. База данных

PostgreSQL — свободная объектно-реляционная система управления базами данных (СУБД).

Преимущества PostgreSQL:

- высокопроизводительные и надёжные механизмы транзакций и репликации;
- наследование;
- возможность индексирования геометрических (в частности, географических) объектов и наличие базирующегося на ней расширения PostGIS;
- встроенная поддержка слабоструктурированных данных в формате JSON с возможностью их индексации;
- расширяемость (возможность создавать новые типы данных, типы индексов, языки программирования, модули расширения, подключать любые внешние источники данных) [3].

Структура таблиц базы данных описывается с помощью моделей в файлах models.py. Чтобы внести изменения в базу данных, необходимо сделать миграции:
python ./manage.py makemigrations
python ./manage.py migrate

В базе данных используется связь «один ко многим» между достопримечательностью и ее категорией, организованная посредством функции ForeignKey с опцией CASCADE.

4. Контейнеризация

Docker— программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации, контейнеризатор приложений. Позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер, который может быть развёрнут на любой Linux-системе с поддержкой контрольных групп в ядре, а также предоставляет набор команд для управления этими контейнерами [4].

5. Выводы

В данном курсовом проекте я познакомилась с докером и понятием контейнеризации, приобрела навыки работы с веб-сервером nginx, произвела замеры производительности работы веб-сервера с проксированием и отдачей статических файлов. Кроме того, я освежила свои знания по работе с фреймворком Django и базой данных PostgreSQL, вспомнила, что такое миграции и зачем они нужны, написала свой тестовый сервер приложений для достопримечательностей, реализовала методы API.

Список литературы

[1] https://ru.wikipedia.org/wiki/Nginx
[2] https://ru.wikipedia.org/wiki/Django
[3] https://ru.wikipedia.org/wiki/PostgreSQL
[4] https://ru.wikipedia.org/wiki/Docker