

Лабораторная работа 2

Изучение архитектуры MV* на примере Django

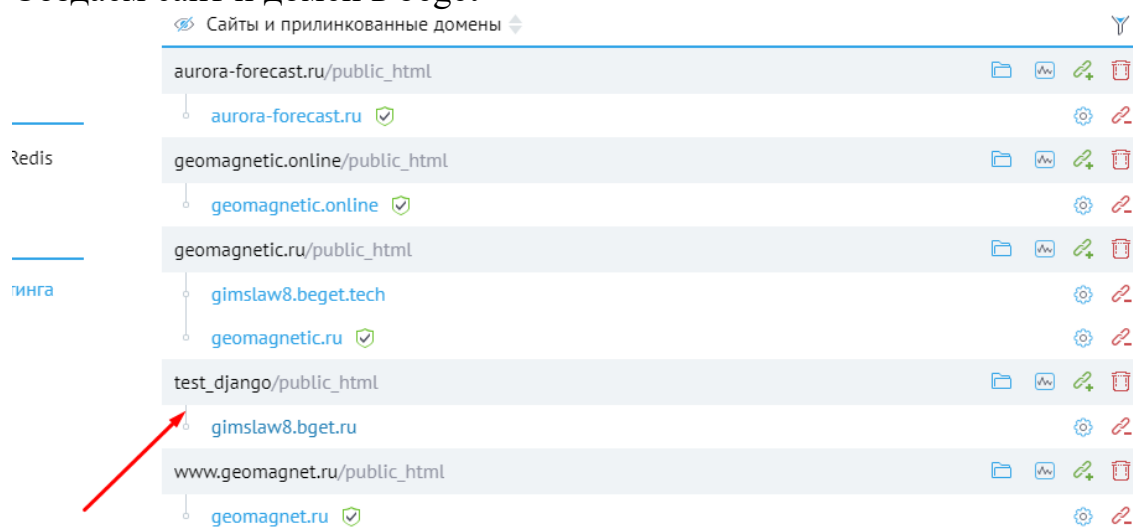
Цель работы: получить навык развертывания фреймворка Django на типовом хостинге.

Ход выполнения

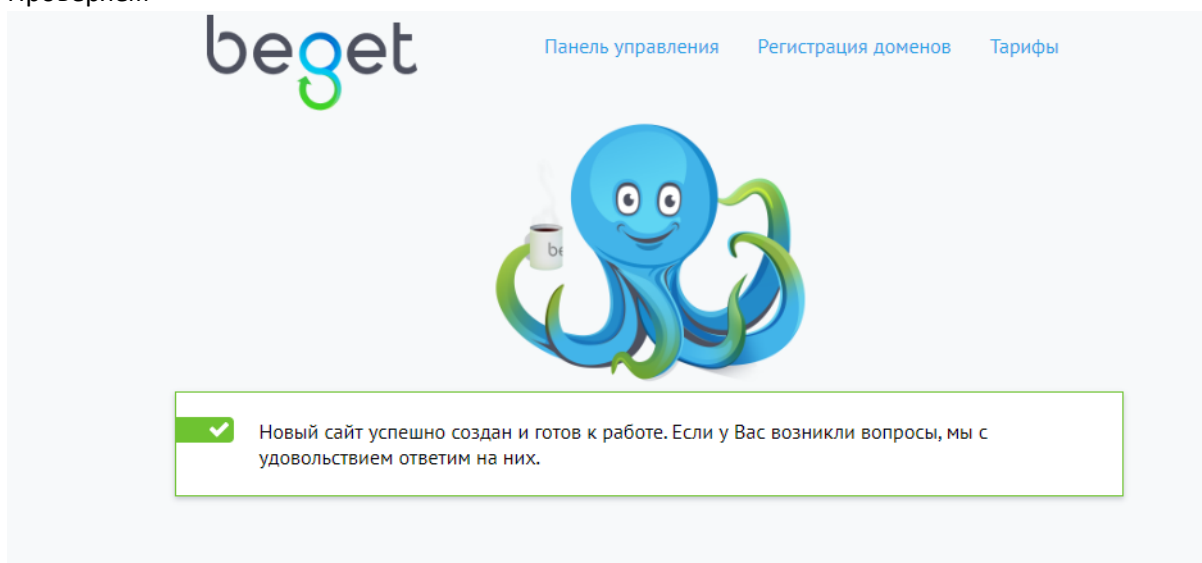
Часть 1. Создание сайта на django на хостинге beget

1. Создание сайта

1.1. Создаем сайт и домен в beget

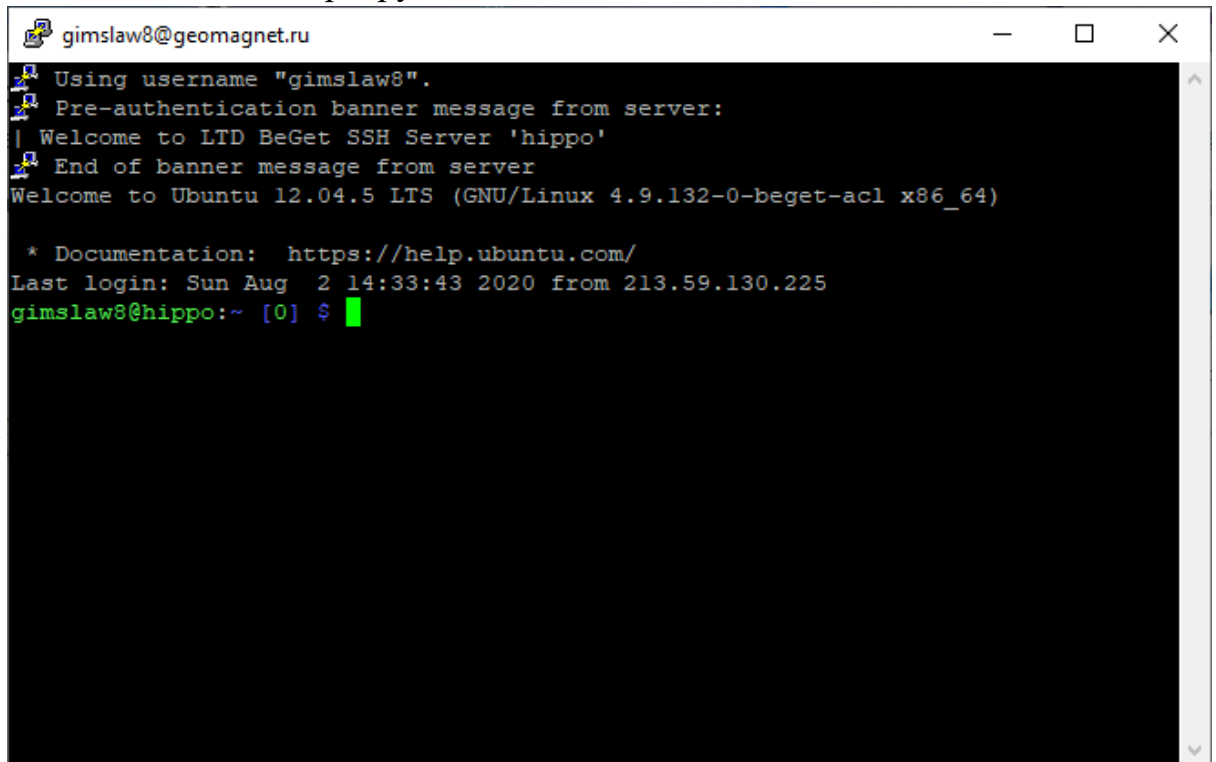


1.2. Проверяем



2. Установка современного Python(на аккаунте gimslaw8 установлен Python3.8)

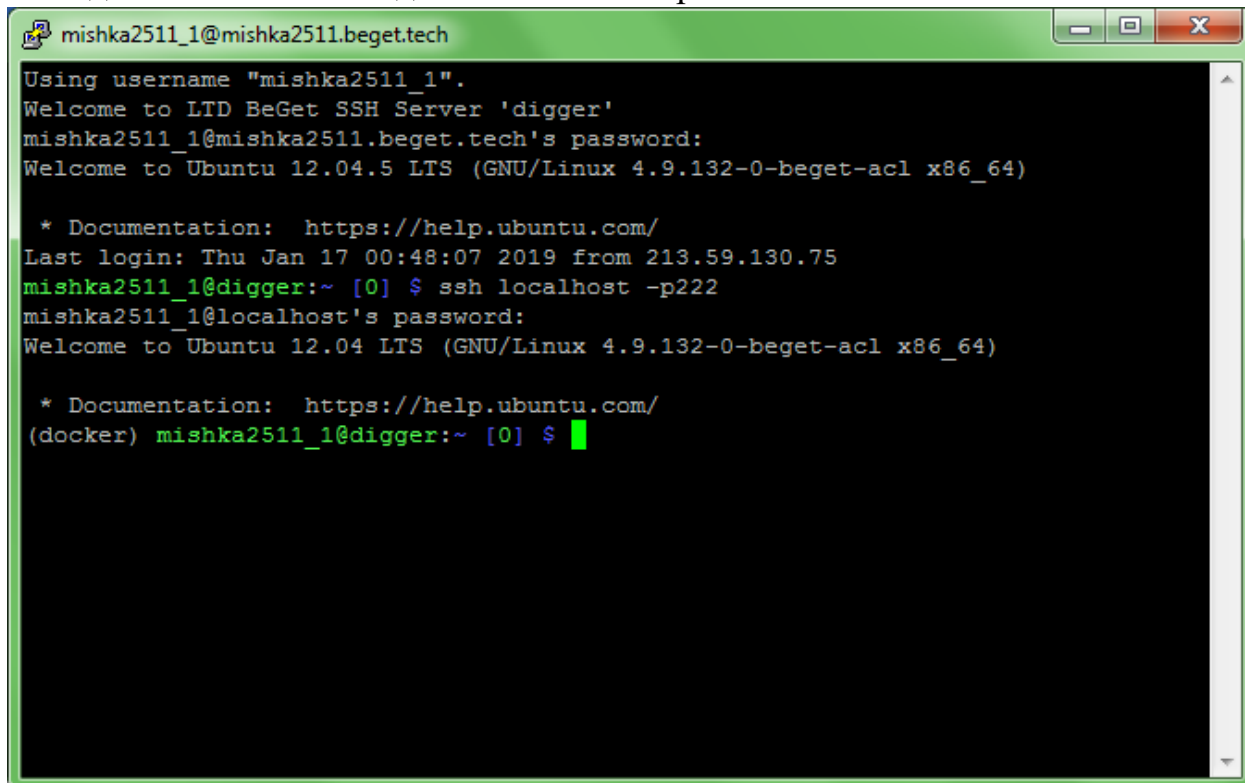
- 2.1. Перед установкой желательно проверить наличие `ssl` на сервере, при необходимости установить через `beget`
- 2.2. Подключаемся к серверу по `ssh`



```
gimslaw8@geomagnet.ru
Using username "gimslaw8".
Pre-authentication banner message from server:
| Welcome to LTD BeGet SSH Server 'hippo'
End of banner message from server
Welcome to Ubuntu 12.04.5 LTS (GNU/Linux 4.9.132-0-beget-ac1 x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Sun Aug  2 14:33:43 2020 from 213.59.130.225
gimslaw8@hippo:~ [0] $
```

- 2.3. Заходим в `docker` командой `ssh localhost -p222`

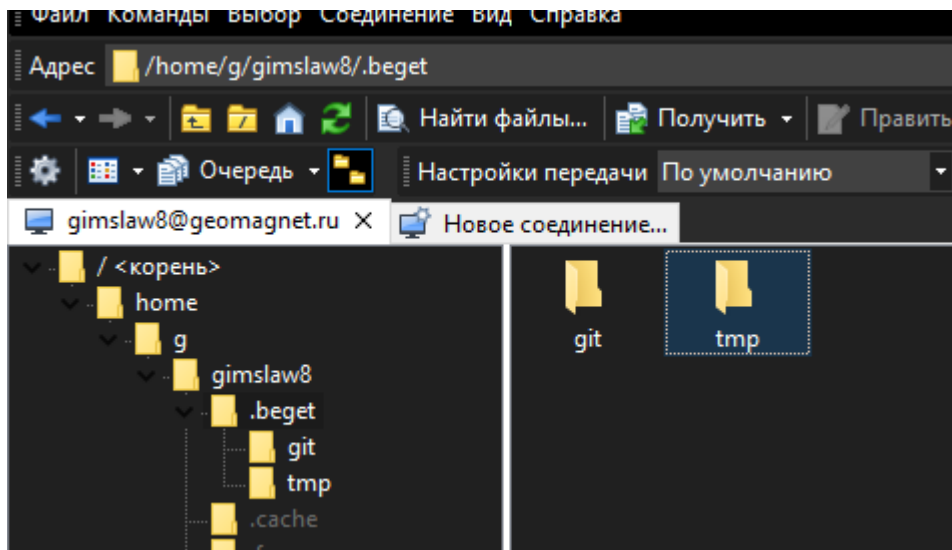


```
mishka2511_1@mishka2511.beget.tech
Using username "mishka2511_1".
Welcome to LTD BeGet SSH Server 'digger'
mishka2511_1@mishka2511.beget.tech's password:
Welcome to Ubuntu 12.04.5 LTS (GNU/Linux 4.9.132-0-beget-ac1 x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Thu Jan 17 00:48:07 2019 from 213.59.130.75
mishka2511_1@digger:~ [0] $ ssh localhost -p222
mishka2511_1@localhost's password:
Welcome to Ubuntu 12.04 LTS (GNU/Linux 4.9.132-0-beget-ac1 x86_64)

 * Documentation:  https://help.ubuntu.com/
(docker) mishka2511_1@digger:~ [0] $
```

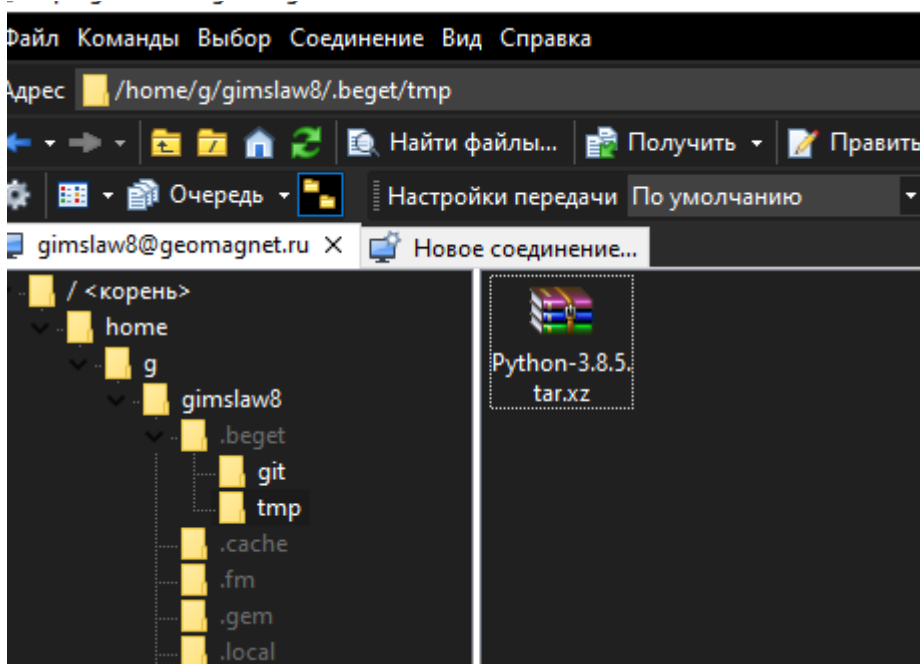
- 2.4. Заходим во временную папку `tmp` командой `cd ~/.beget/tmp/`, если её нет, то предварительно создаем командой `mkdir -p ~/.beget/tmp`
Папка в WinScp



2.5. Находим на сайте Python ссылку на скачивание последней версии(на момент написания Python 3.8.5) -

<https://www.python.org/ftp/python/3.8.5/Python-3.8.5.tar.xz>

2.6. Скачиваем этот Python на сервер командой `wget <ссылка>`
В результате должен появиться архив в папке `tmp`



2.7. Распакуем архив и перейдём в полученную папку командами

```
tar xf <назваине архива>
```

и

```
cd <имя_папки> 1
```

2.8. Подготовим Makefile с помощью утилиты **configure** командой

```
./configure --prefix $HOME/.local
```

После компилируем и устанавливаем командой

```
make -j33 && make install
```

2.9. Проверим установленную версию Python командой

```
python3 -V
```

В результате должна быть установлена 3.8.5

Также проверим версию пакетного менеджера pip

```
pip3 -V
```

Должен быть выведен путь в файлу с указанием версии питона Python3.8.5

2.10. Установка виртуального окружения (рекомендуется – нужно для повышения независимости друг от друга разных Python-проектов на одном хостинге)

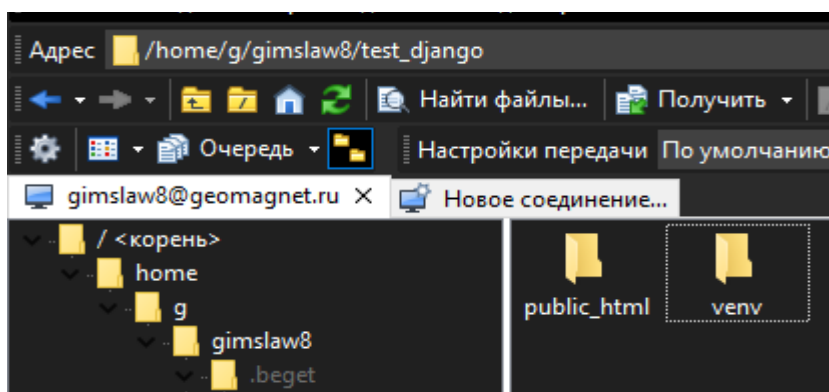
2.10.1. Установим виртуальное окружение командой

```
pip3 install virtualenv --user
```

2.10.2. Переходим в папку сайта и создаем виртуальное окружение командой

```
python3 -m venv venv
```

(Последнее слово в команде – название папки виртуального окружения). В результате должна появиться папка виртуального окружения



2.10.3. Запустим виртуальное окружение командой

```
source venv/bin/activate
```

Важно: Благодаря виртуальному окружению команды `python` и `pip` ссылаются на нужную версию питон, и далее будут использоваться они. Без `venv`-а, надо будет использовать `python` и `pip3`.

3. Установка Django и создание Django-сайта

3.1. Установим Django командой (если используется `virtualenv`, то в нём)

```
pip install django
```

Должна установиться последняя версия.

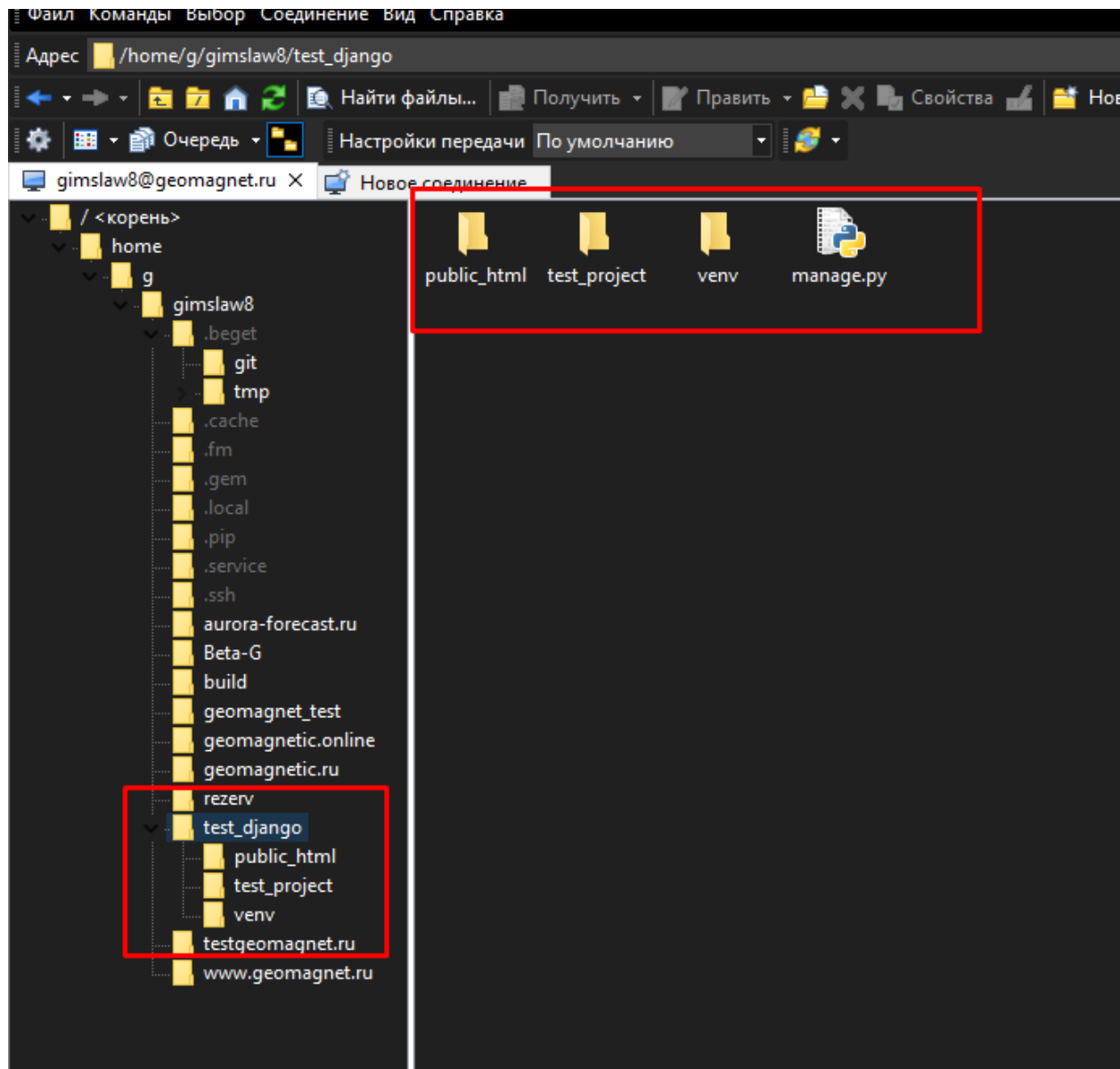
Возможно ошибка при подключении `pip` к репозиторию пакетов PyPi, причина в `ssl`, необходимо установить сертификат и переустановить `python`

3.2. Создаем Django-проект с помощью команды

```
django-admin startproject test_project .
```

Где `test_project` – имя проекта (название папки с конфигурацией проекта), четвертый аргумент «.» нужен для создания в текущей директории, а не уровнем.

Должна получиться такая структура:



3.3. Создадим файл `passenger_wsgi.py` по документации https://beget.com/ru/articles/webapp_python#5

Например

```
No Python interpreter configured for the project
1  # -*- coding: utf-8 -*-
2  import os, sys
3  sys.path.insert(0, '/home/g/gimslaw8/test_django')
4  sys.path.insert(1, '/home/g/gimslaw8/test_django/venv/lib/python3.8/site-packages')
5  os.environ['DJANGO_SETTINGS_MODULE'] = 'test_project.settings'
6  from django.core.wsgi import get_wsgi_application
7  application = get_wsgi_application()
```

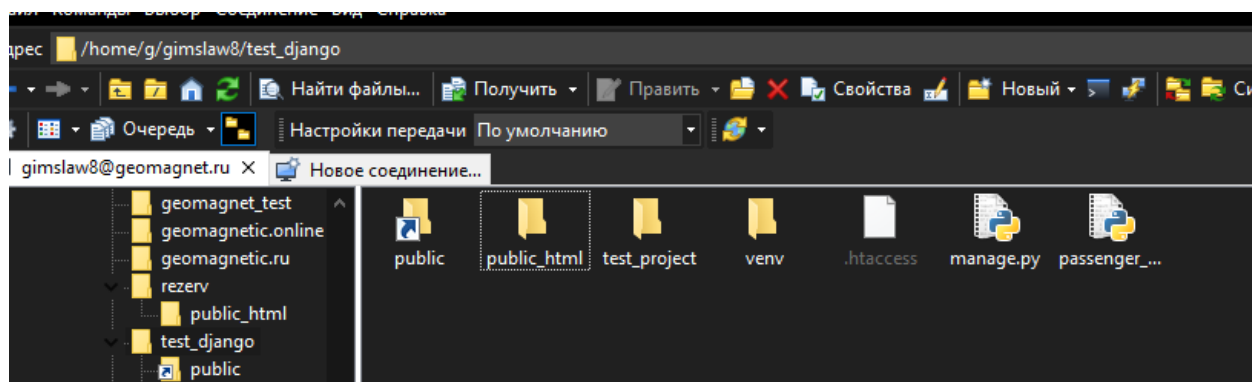
3.4. Создадим файл .htaccess по документации, например

```
tsconfig.json babel.config.js simple.rad 0.310.2_state t_8U0noH.pdb webpack.config.base
1 PassengerEnabled On
2 PassengerPython /home/g/gimslaw8/test_django/venv/bin/python
3
```

3.5. Создадим ссылку public на папку public_html для отдачи статики через nginx командой

```
ln -s public_html public
```

В итоге содержимое папки должно быть таким



3.6. Создадим папку для перезапуска сервера

```
mkdir tmp
```

И перезапустим сервер командой

```
touch tmp/restart.txt
```

3.7. Проверим – откроем сайт в браузере. Если все сделано правильно – откроется ошибка Django, говорящая о том, что не настроены допустимые url-адреса, на которых может работать сайт

DisallowedHost at /

Invalid HTTP_HOST header: 'gimslaw8.bget.ru'. You may need to add 'gimslaw8.bget.ru' to ALLOWED_HOSTS.

```
Request Method: GET
Request URL: http://gimslaw8.bget.ru/
Django Version: 3.0.8
Exception Type: DisallowedHost
Exception Value: Invalid HTTP_HOST header: 'gimslaw8.bget.ru'. You may need to add 'gimslaw8.bget.ru' to ALLOWED_HOSTS.
Exception Location: /home/g/gimslaw8/test_django/venv/lib/python3.8/site-packages/django/http/request.py in get_host, line 122
Python Executable: /home/g/gimslaw8/test_django/venv/bin/python
Python Version: 3.8.2
Python Path: ['/home/g/gimslaw8/test_django',
              '/home/g/gimslaw8/test_django/venv/lib/python3.8/site-packages',
              '/home/g/gimslaw8/test_django',
              '/opt/passenger40/helper-scripts',
              '/home/g/gimslaw8/.local/lib/python3.8.zip',
              '/home/g/gimslaw8/.local/lib/python3.8',
              '/home/g/gimslaw8/.local/lib/python3.8/lib-dynload',
              '/home/g/gimslaw8/test_django/venv/lib/python3.8/site-packages']
Server time: Sun, 2 Aug 2020 20:51:45 +0000
```

4. Настройки Django и создание простейшего приложения

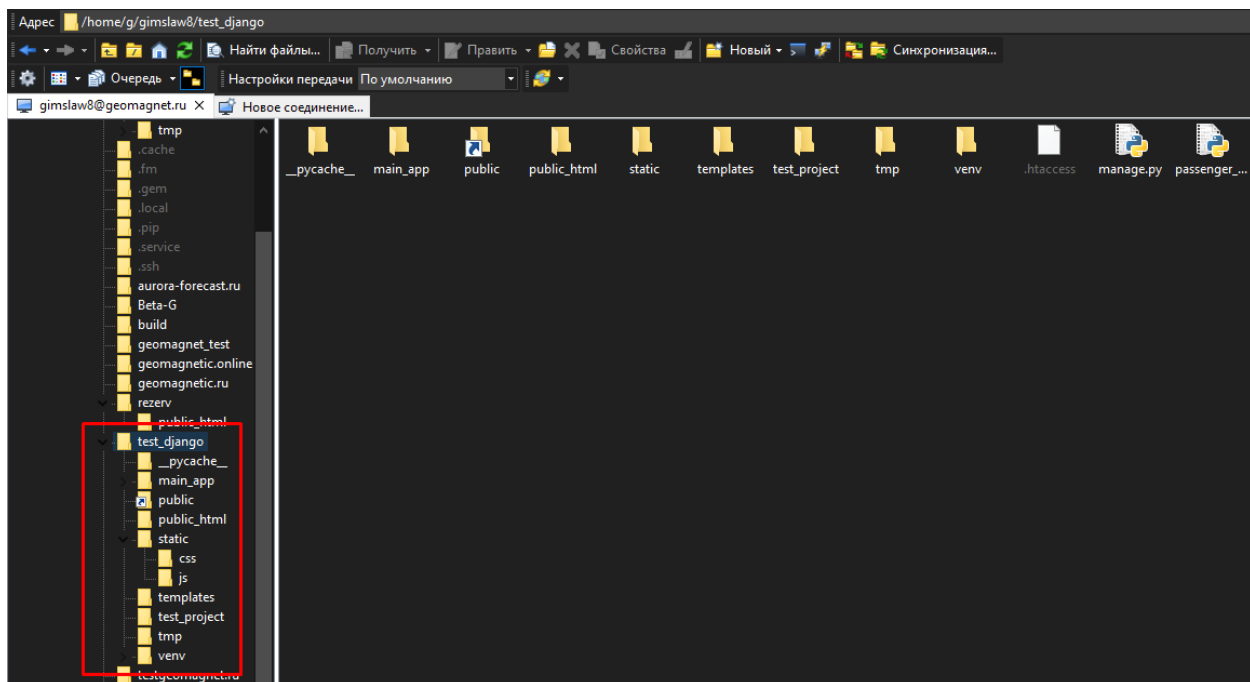
4.1. Создадим Django-приложение командой

```
python manage.py startapp main_app
```

Где `main_app` – название нашего приложения

4.2. Создадим папки для шаблонов и статических файлов (картинки, стили, скрипты) – назовём их `templates` и `static`

Должна получиться примерно такая структура



4.3. Настроим Django – откроем файл настроек (`test_project/settings.py`)

4.3.1. Добавим наш хост (например gimplaw8.bget.ru) в ALLOWED_HOSTS (можно использовать «*» для установки любых хостов)

```
25 # SECURITY WARNING: don't run with debug turned on in production
26 DEBUG = True
27
28 ALLOWED_HOSTS = ['gimplaw8.bget.ru']
29
30
31 # Application definition
32
```

4.3.2. Подключим созданное приложение в INSTALLED_APPS

```
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'main_app',
41 ]
42
43 MIDDLEWARE = [
44     'django.middleware.security.SecurityMiddleware',
45     'django.contrib.sessions.middleware.SessionMiddleware',
46     'django.middleware.common.CommonMiddleware',
47     'django.middleware.csrf.CsrfViewMiddleware',
```

4.3.3. Добавим папку с шаблонами в TEMPLATES.DIRS

```

53 ROOT_URLCONF = 'test_project.urls'
54
55 TEMPLATES = [
56     {
57         'BACKEND': 'django.template.backends.django.DjangoTemplates',
58         'DIRS': [
59             'templates',
60         ],
61         'APP_DIRS': True,
62         'OPTIONS': {
63             'context_processors': [
64                 'django.template.context_processors.debug',
65                 'django.template.context_processors.request',
66                 'django.contrib.auth.context_processors.auth',
67                 'django.contrib.messages.context_processors.messages',
68             ],
69         },
70     ],
71 ]
72

```

4.3.4. Создадим STATICFILES_DIRS и добавим туда папку со статикой

```

121 # https://docs.djangoproject.com/en/3.0/howto/static-fi
122
123 STATIC_URL = '/static/'
124
125 STATICFILES_DIRS = [
126     os.path.join(BASE_DIR, 'static'),
127 ]
128
129

```

Важно! Необходим абсолютный путь, лучше использовать `os.path.join(BASE_DIR, «имя_папки»)`, но можно и просто `'/home/g/gimslaw8/test_django/static '`

Сразу же создадим `STATIC_ROOT` – путь к папке, которую будет использовать `nginx` для статических файлов

4.4. Создадим простую страницу

4.4.1. Создадим простую страницу `index.html` и поместим её в папку `templates`

4.4.2. Откроем файл `views.py`(лежит папке приложения `main_app/views.py`) с контроллерами и создадим простую функцию, возвращающую отрендеренный шаблон `index.html`

```
from django.shortcuts import render

# Create your views here.

def index(request):
    return render(request, 'index.html')
```

4.4.3. Добавим эту функцию в файл настроек `urls.py`(в папке проекта `test_project/urls.py`)

```
from django.contrib import admin
from django.urls import path
from main_app import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.index),
]
```

Важно! Может отличаться в зависимости от версии Django, написано для версии `django3.0`

4.4.4. Проверим – перезапустим сервер командой `touch tmp/restart.txt` и откроем сайт в браузере. Должен быть показан шаблон

4.5.Подключение стилей и скриптов.

4.5.1. Создадим простые файлы `index.css` и `index.js`, разместим их в папке `static`, или в папках `static/css` и `static/js` соответственно

4.5.2. Добавим их подключение в шаблоне `index.html`

```
1 <html lang="ru-Ru">
2 <head>
3   <title>Test</title>
4   {% load static %}
5
6   <script src="{% static 'js/index.js'"></script>
7   <link type="text/css" rel="stylesheet" href="{%static 'css/index.css' %}">
8 </head>
9 <body>
10   <h1>Hello world!</h1>
11 </body>
12 </html>
```

4.5.3. Добавим отдачу статических файлов через Django добавив в путь к ним в urls.py по документации <https://docs.djangoproject.com/en/3.0/howto/static-files/>

```
13 1. Import the include() function: from django.urls import include, path
14 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15 """
16 from django.contrib import admin
17 from django.urls import path
18 from main_app import views
19
20 from django.conf import settings
21 from django.conf.urls.static import static
22
23 urlpatterns = [
24     path('admin/', admin.site.urls),
25     path('', views.index),
26 ] + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
27
```

Перезапустим сервер и проверим, всё должно работать.

Часть 2. Персонализация веб-приложения

Перенести верстку из предыдущей работы в статический шаблон на Django. Проверить работоспособность.