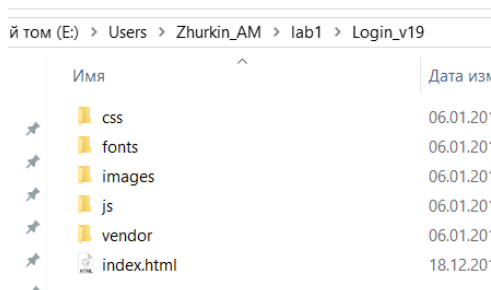
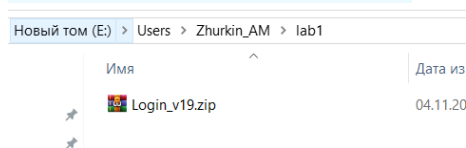
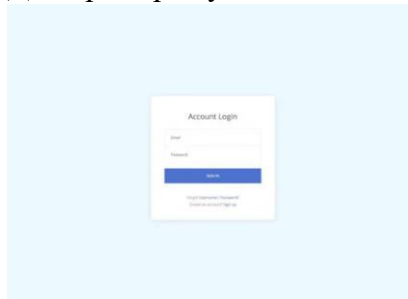


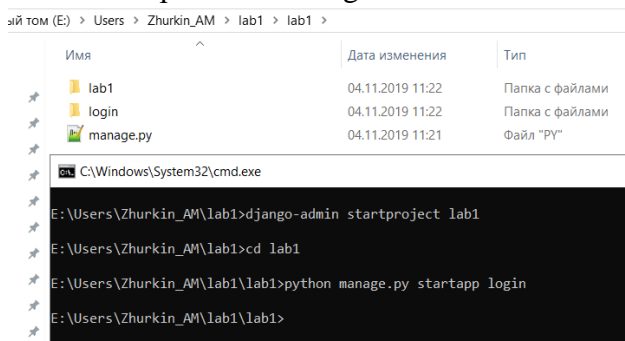
Простой обработчик GET-запросов (Вход на сайт)

1. Зайдите на сайт:
<https://www.internet-technologies.ru/articles/60-besplatnyh-html5-i-css3-form-avtorizacii-dlya-sayta.html>
2. Выберите шаблон входа на сайт (Номер шаблона должен соответствовать номеру компьютера в аудитории). Скачайте его.
Для примера будем использовать шаблон № 19.

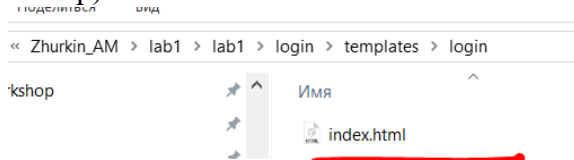


В каталоге шаблона представлены статические (неизменяемые файлы).

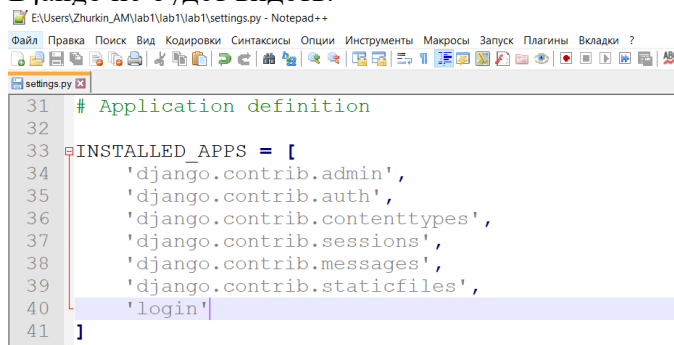
3. Создайте приложение login.



4. Создайте папку templates в каталоге login, а в каталоге templates создайте папку login. Скопируйте index.html из скачанного проекта Login_v19 (У вас другой номер).

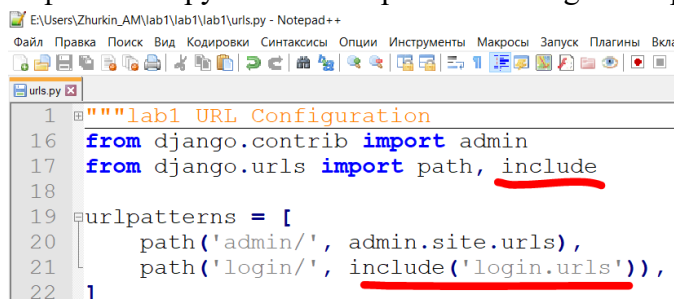


5. Добавьте приложение login в setting.py. Если не добавите, то статические файлы Django не будет видеть.



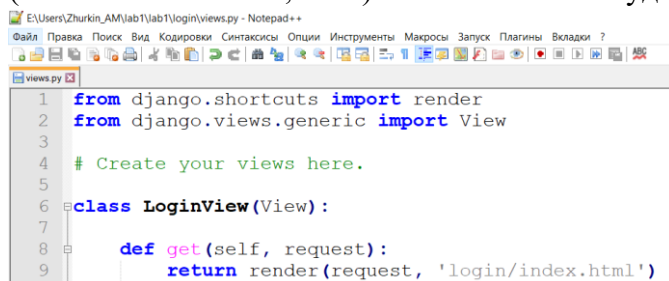
```
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'login'
41 ]
```

6. В файле urls.py включите приложение login в маршрут.



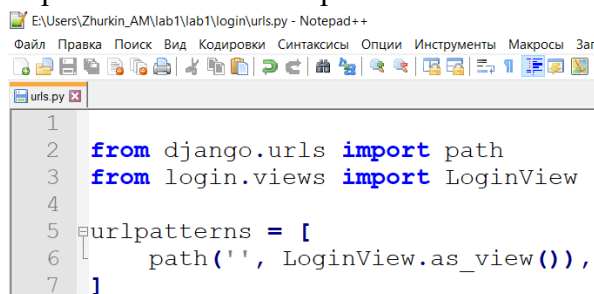
```
1 """lab1 URL Configuration
16 from django.contrib import admin
17 from django.urls import path, include
18
19 urlpatterns = [
20     path('admin/', admin.site.urls),
21     path('login/', include('login.urls')),
22 ]
```

7. Добавьте обработчик (Class based view, CBV). Обработчики на основе функций (Function based view, FBV) использовать не будем (устарело).



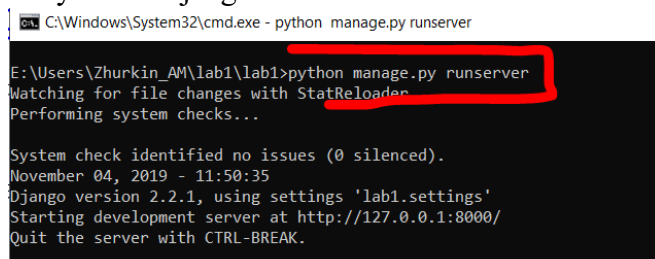
```
1 from django.shortcuts import render
2 from django.views.generic import View
3
4 # Create your views here.
5
6 class LoginView(View):
7
8     def get(self, request):
9         return render(request, 'login/index.html')
```

8. Создайте login/urls.py и добавьте следующий код. Теперь класс LoginView будет обрабатывать GET-запросы.



```
1
2 from django.urls import path
3 from login.views import LoginView
4
5 urlpatterns = [
6     path('', LoginView.as_view()),
7 ]
```

9. Запускаем Django:

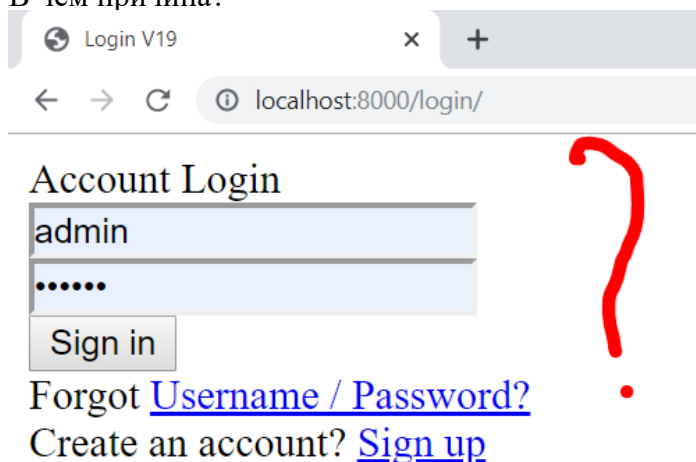


```
C:\Windows\System32\cmd.exe - python manage.py runserver
E:\Users\Zhurkin_AM\lab1\lab1>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

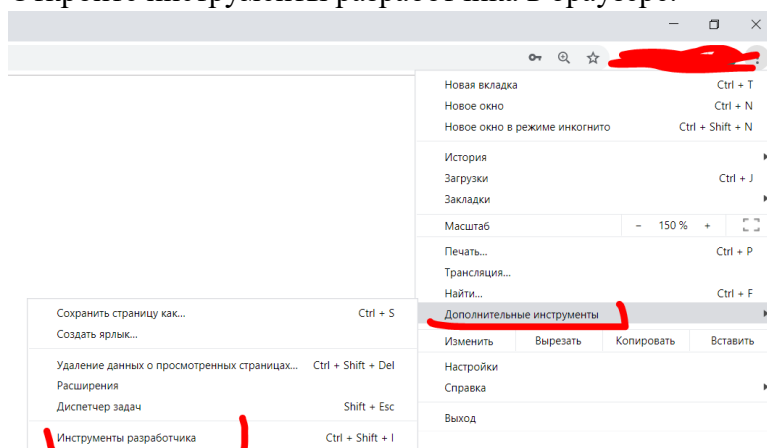
System check identified no issues (0 silenced).
November 04, 2019 - 11:50:35
Django version 2.2.1, using settings 'lab1.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

10. Зайдите на страницу. В результате вы обнаружите, что отображается только html.

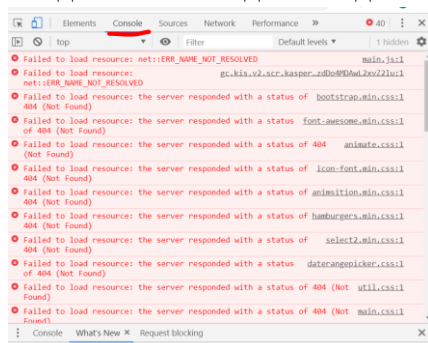
В чём причина?



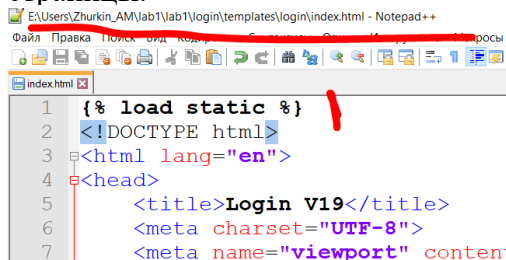
11. Откройте инструменты разработчика в браузере.



12. Откройте Console. В результате обнаруживается, что статические файлы не найдены. Необходимо подготовить html страницу для использования в Django.



13. Откройте `templates/login/index.html` и добавьте тэг `{% load static %}`. Данный тег указывает Django, что необходимо подгрузить статические файлы для данной html страницы.



```

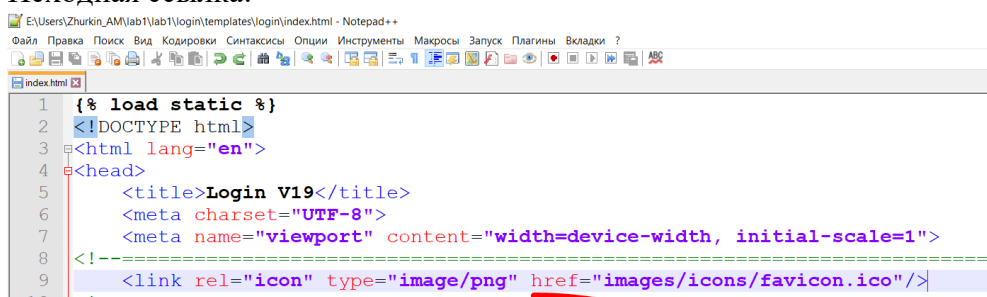
1 {% load static %}
2 <!DOCTYPE html>
3 <html lang='en'>
4 <head>
5 <title>Login V19</title>
6 <meta charset='UTF-8'>
7 <meta name='viewport' content=

```

14. Теперь необходимо откорректировать все ссылки на статические ресурсы href.

15. Пример замены фавикона (Фавикон – значок вкладки WEB-страницы).

Исходная ссылка:

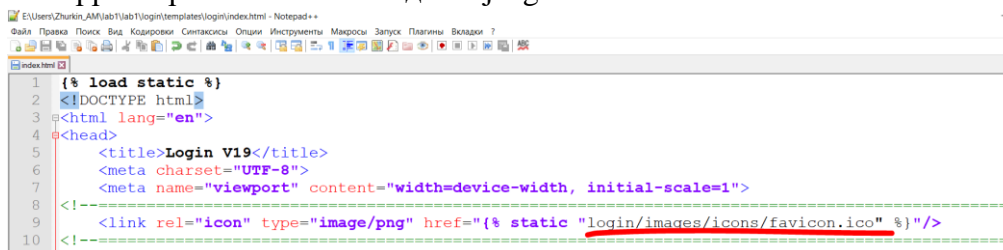


```

1 {% load static %}
2 <!DOCTYPE html>
3 <html lang='en'>
4 <head>
5 <title>Login V19</title>
6 <meta charset='UTF-8'>
7 <meta name='viewport' content='width=device-width, initial-scale=1'>
8 <!--
9 <link rel='icon' type='image/png' href='images/icons/favicon.ico'>
10 <!--

```

Откорректированная ссылка для Django:

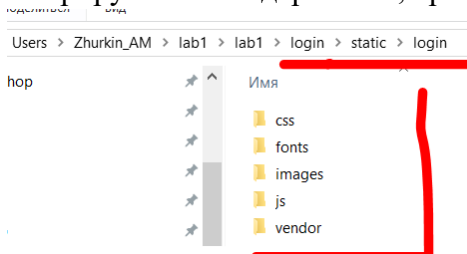


```

1 {% load static %}
2 <!DOCTYPE html>
3 <html lang='en'>
4 <head>
5 <title>Login V19</title>
6 <meta charset='UTF-8'>
7 <meta name='viewport' content='width=device-width, initial-scale=1'>
8 <!--
9 <link rel='icon' type='image/png' href='{% static 'login/images/icons/favicon.ico' %}'>
10 <!--

```

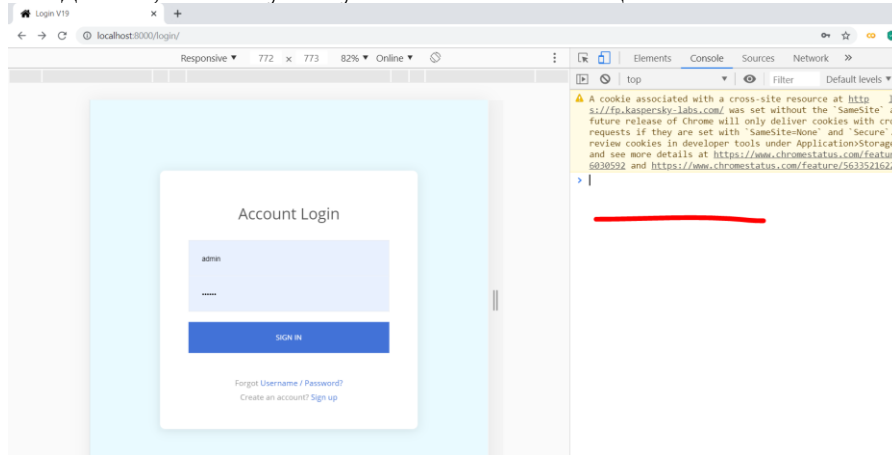
16. Создайте каталог для хранения статических файлов `lab1/login/static/login`. Скоринуйте всё содержимое, кроме `index.html` в этот каталог.



17. Обновите страницу. Фавиком отобразится на вкладке страницы.



18. Отредактируйте все остальные ссылки в вашей html странице. Обновите страницу.
Убедитесь, что отсутствуют в Console сообщения об ошибках.



Создание простого обработчика GET-запросов
на основе классов (CBV) **успешно завершено!**

Простой обработчик POST-запросов (Проверка имя пользователя и пароля)

1. Найдите в login/templates/login/index.html тег с началом (Форма начинается с <form

```

3  <div class="limiter">
4    <div class="container-login100">
5      <div class="wrap-login100 p-l-55 p-r-55 p-t-65 p-b-50">
6        <form class="login100-form validate-form">
7          <span class="login100-form-title p-b-33">
8            Account Login
9          </span>
10
11          <div class="wrap-input100 validate-input" data-validate = "Val:
12            <input class="input100" type="text" name="email" placeholder=
13            <span class="focus-input100-1"></span>
14            <span class="focus-input100-2"></span>
15          </div>
16
17          </div>
18        </form>
19      </div>
20    </div>
21
22    <div class="text-center">
23      <span class="txt1">
24        Create an account?
25      </span>
26
27      <a href="#" class="txt2 hov1">
28        Sign up
29      </a>
30    </div>
31  </div>
32</div>

```

и концом формы (форма завершается </form>

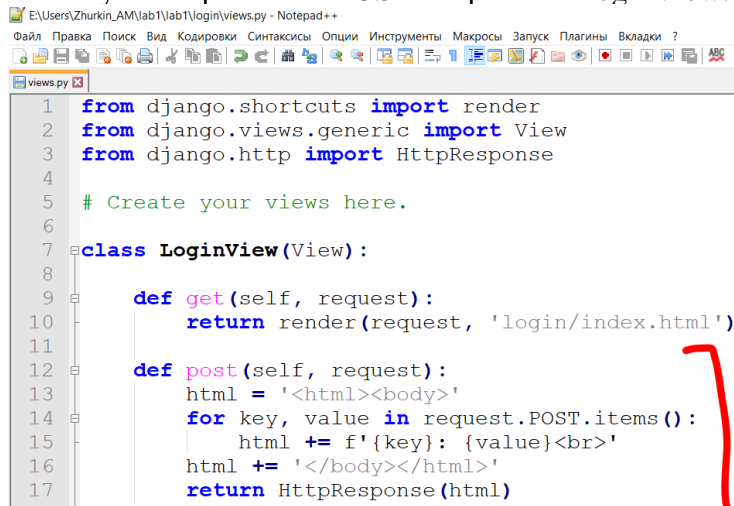
2. Добавьте тег в action URL login, метод запроса POST и csrf token. CSRF токен предназначен для защиты от межсайтовых атак. Т.к. URL POST-запроса совпадает с URL GET-запроса, то необходимо поставить точку.

```

<div class="container-login100">
  <div class="wrap-login100 p-l-55 p-r-55 p-t-65 p-b-50">
    <form action="." class="login100-form validate-form" method="post">
      <div class="wrap-input100 validate-input" data-validate = "Val:
        <input class="input100" type="text" name="email" placeholder=
        <span class="focus-input100-1"></span>
        <span class="focus-input100-2"></span>
      </div>
      <div class="text-center">
        <span class="txt1">
          Create an account?
        </span>
        <a href="#" class="txt2 hov1">
          Sign up
        </a>
      </div>
    </form>
  </div>
</div>

```

3. Реализуйте обработчик POST-запросов в LoginView.



```
1 from django.shortcuts import render
2 from django.views.generic import View
3 from django.http import HttpResponseRedirect
4
5 # Create your views here.
6
7 class LoginView(View):
8
9     def get(self, request):
10         return render(request, 'login/index.html')
11
12     def post(self, request):
13         html = '<html><body>'
14         for key, value in request.POST.items():
15             html += f'{key}: {value}<br>'
16         html += '</body></html>'
17         return HttpResponseRedirect(html)
```

4. Выполните запрос. В результате будет выведен список переменных их значений из тела POST-запроса.

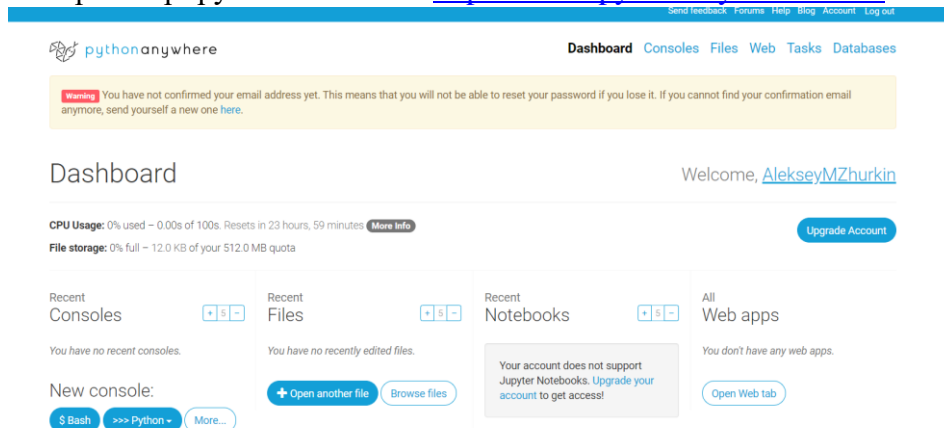
Содание простого обработчика POST-запросов
на основе классов (CBV) **успешно завершено!**

Самостоятельная работа

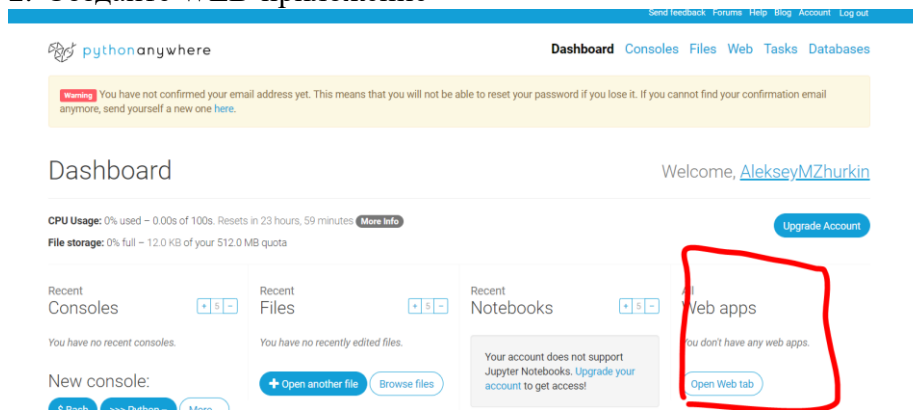
1. Скачайте любой шаблон сайта (должно быть несколько html страниц).
Можно использовать следующий ресурс: <https://freehtmlthemes.ru/>
2. Настройте маршруты.
3. Создайте CBV с обработкой GET и POST запросов. Около 10 GET и 5 POST запросов.

Публикация на сервере

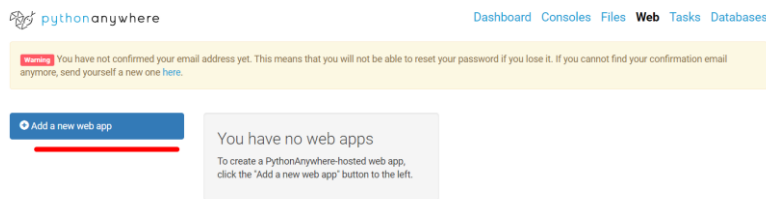
1. Зарегистрируйтесь на сайте <https://www.pythonanywhere.com/>



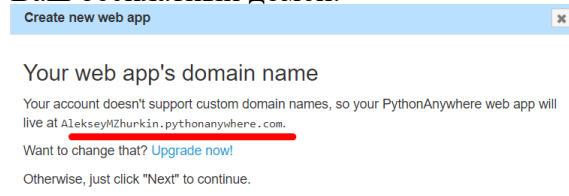
2. Создайте WEB-приложение



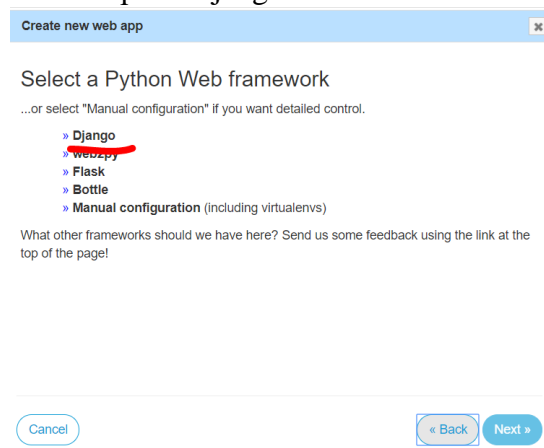
3. Нажмите «Add a new web app»



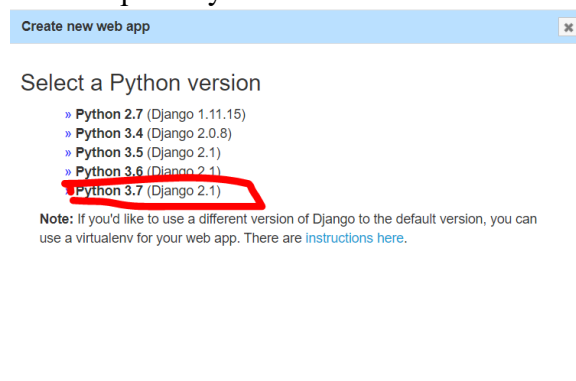
Ваш бесплатный домен.



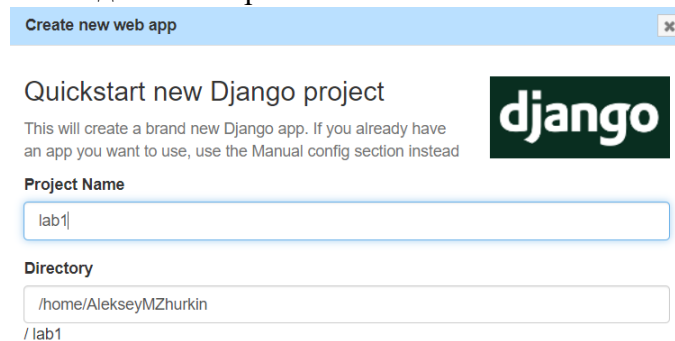
4. Выберите Django



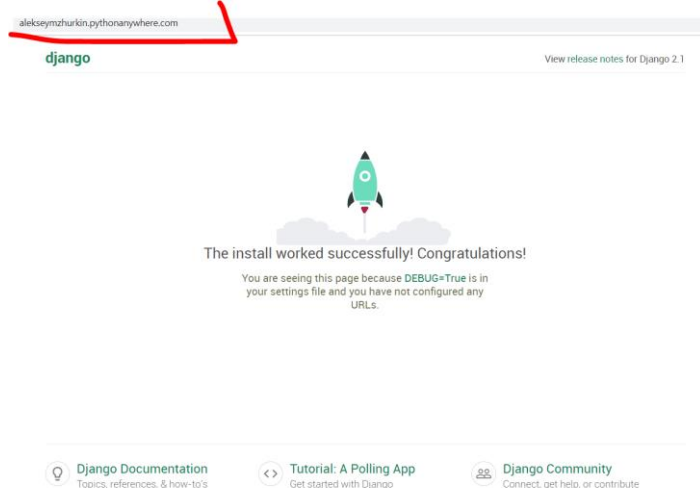
5. Выберите Python 3.7



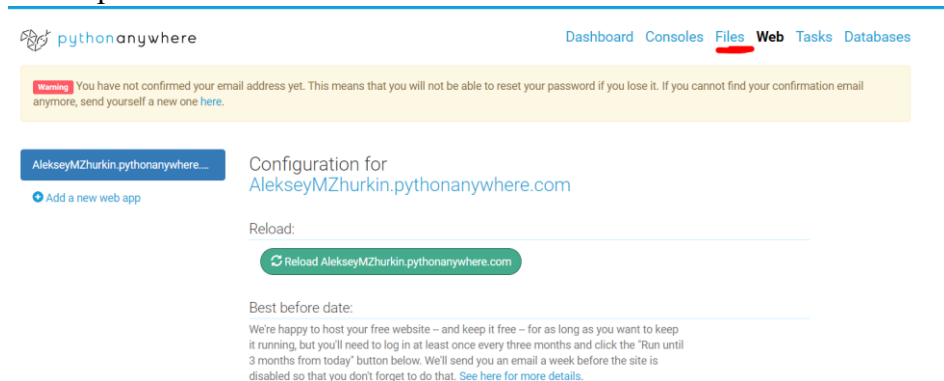
6. Введите имя проекта lab1



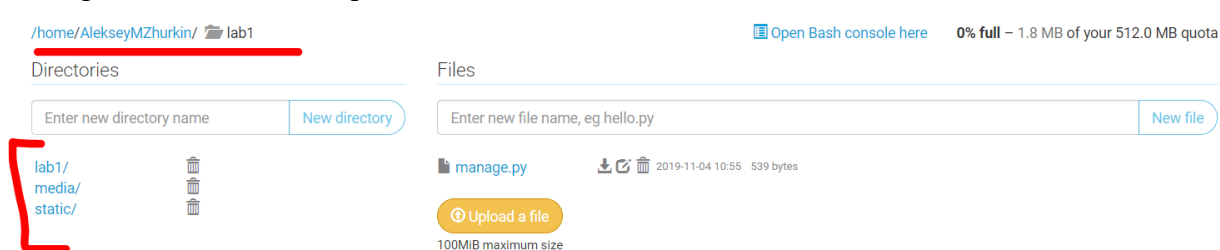
7. Ваш сайт будет работать.



8. Открываем Files



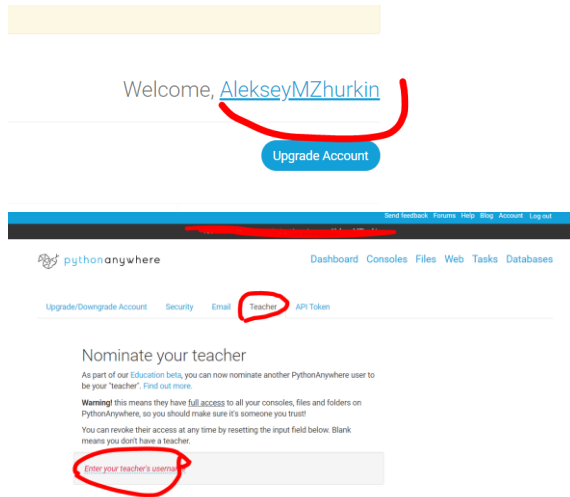
9. Перейдите в каталог проекта lab1



10.

11. Подключите преподавателя к Вашему аккаунту.

Введите USER преподавателя AlekseyMZhurkin.



12. Далее нужно скопировать все файлы. Можно и нужно использовать Git.