



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»  
(ДГТУ)**

# Отчет

по дисциплине «Основы проектной деятельности»

Выполнил:  
Студент группы ВИС23  
Дедюхин А.С.

Проверила: Зубарева Е.Г.

Ростов-на-Дону  
2023 г.

# Тема: создание веб-приложений на основе Django

## Краткое описание

В этой работе рассмотрены способы и примеры создания простейших веб-приложений, их элементов или функций. В качестве рабочего редактора был выбран Microsoft Visual Studio Code, а в качестве фреймворка – Django. Основные пользовательские функции и приложения (аппликации) используют язык Python версии 3.7.9 (64-bit), разметки страниц – HTML, язык запросов к базам данных – SQLite.

Для корректной работы с фреймворком необходимо установить все необходимые расширения: Python Extension, SQLite viewer, Django, Pylance. Также для удобства рекомендуется установить расширения Flake8 (облегчает написание кода в соответствии с PEP8) и Russian Language Pack (русификатор интерфейса).

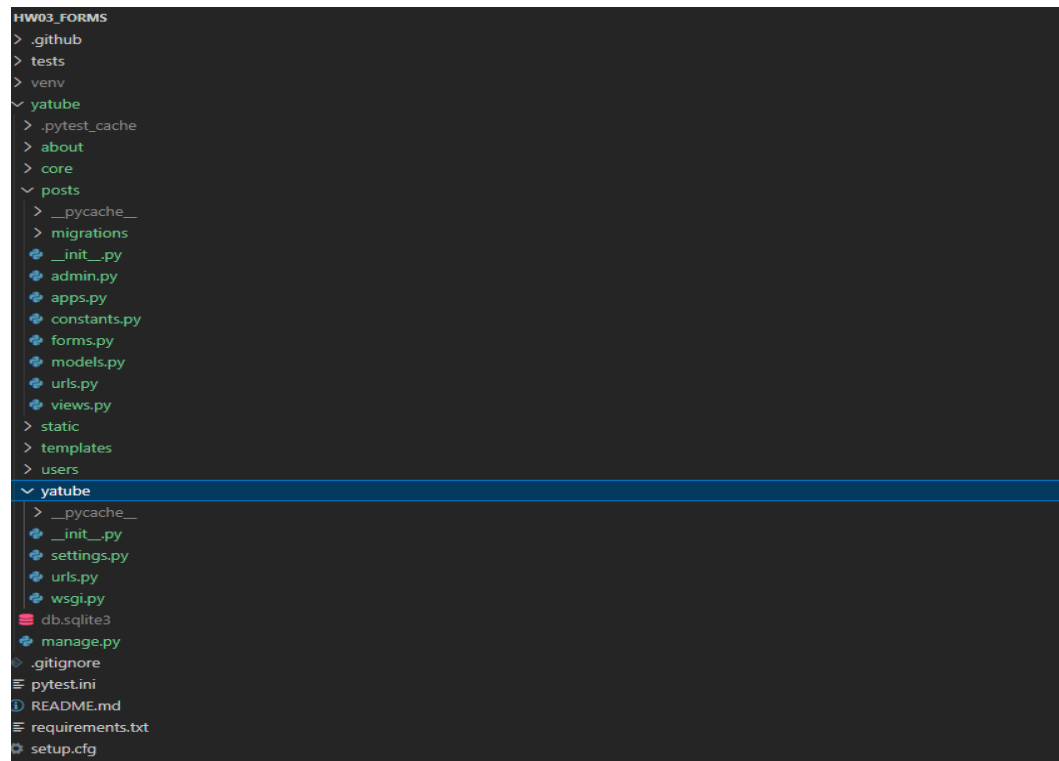
В данной работе мы рассмотрим создание веб-приложения, позволяющего пользователям публиковать на сервере текстовые сообщения. В качестве примера, загруженного в нашу базу данных, хранящуюся на устройстве-хосте используются письма Льва Николаевича Толстого (он же «Пользователь Ieo»). Пользователю будут доступны функции регистрации, log-in и log-out, написание текстового сообщения (далее «пост»), просмотр списка «постов» того или иного пользователя и многое другое.

Все подробности установки необходимых расширений и ПО можно найти в официальной документации к VS Code, Django и Python.

# Начало работы

Для начала работы необходимо установить ПО и все необходимые разрешения. Также необходимо установить в нашей директории виртуальное окружение с помощью менеджера *pip* (подробнее можно прочесть в документации) . В нашем случае в командной строке выполняем команду `pip install virtualenv`. В дальнейшем, находясь в директории на уровне, где установлено виртуальное окружение, для его активации следует ввести в командную строку `source venv/Scripts/activate`. В дальнейшем для сохранения прогресса разработки и возможности вернуться к предыдущим версиям рекомендуется использовать инструментарий **Git**.

Теперь достаточно ввести в командную строку `python manage.py startapp <имя_приложения>`. После выполнения команды создастся наш рабочий репозиторий с необходимой нам архитектурой, который будет выглядеть к концу работы следующим образом:



# Приложение и его функции

В нашем проекте мы будем работать над приложением Yatube. Для того, чтобы оно являлось цельным и имело нужное нам название его необходимо зарегистрировать в файле *settings.py* (см. документацию). По желанию ему можно добавить язык по умолчанию, часовой пояс и прочие свойства.

Для выполнения необходимых нам функций, а именно

- регистрация пользователей
- возможность писать посты и просматривать группы
- возможность входить и выходить из своего аккаунта

нам потребуется:

- создать шаблоны страниц
- создать view-функции отображения нужных страниц
- разработать систему url-адресов и закрепить их за определенными функциями и страницами.

# HTML-шаблоны страниц

Для начала следует создать основу разметки. С её помощью будет удобнее создавать отдельные специализированные страницы в целом. Поэтому создадим документы *base.html*, *header.html* и *footer.html*. Как следует из названий, это будут основное тело страницы, верхняя и нижняя часть страницы. В нашем случае html-код выглядит так:

## Base.html

```
yatube > templates > < base.html
1  {% load static %}
2  <!DOCTYPE html> <!-- Используется html 5 версии -->
3  <html lang="ru"> <!-- Язык сайта - русский -->
4  <head>
5      <meta charset="utf-8"> <!-- Кодировка сайта -->
6      <!-- Сайт готов работать с мобильными устройствами -->
7      <meta name="viewport" content="width=device-width, initial-scale=1">
8      <!-- Загружаем фав-иконки -->
9      <link rel="icon" href="{% static 'img/fav/fav.ico' %}" type="image">
10     <link rel="apple-touch-icon" sizes="180x180" href="{% static 'img/fav/apple-touch-icon.png' %}">
11     <link rel="icon" type="image/png" sizes="32x32" href="{% static 'img/fav/favicon-32x32.png' %}">
12     <link rel="icon" type="image/png" sizes="16x16" href="{% static 'img/fav/favicon-16x16.png' %}">
13     <meta name="msapplication-TileColor" content="#000">
14     <meta name="theme-color" content="#ffffff">
15     <!-- Подключен файл со стандартными стилями бустрап -->
16     <link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}">
17     <title>
18         {% block title %}
19             Последние обновления на сайте
20         {% endblock %}
21     </title>
22 </head>
23 <body>
24     <header>
25         {% include 'includes/header.html' %}
26     </header>
27     <main>
28         {% block content %}
29             <!-- класс ru-5 создает отступы сверху и снизу блока -->
30             <div class="container ru-5">
31                 Не существует одиночества более глубокого, чем одиночество самурая...
32             {% endblock %}
33         </main>
34         <!-- Используются классы бустрапа: -->
35         <!-- border-top: создаёт тонкую линию сверху блока -->
36         <!-- text-center: выравнивает текстовые блоки внутри блока по центру -->
37         <!-- ru-3: контент внутри размещается с отступом сверху и снизу -->
38     <footer>
39         {% include 'includes/footer.html' %}
40     </footer>
41 </body>
42 </html>
```

## Header.html

```
yatube > templates > includes > <> header.html
1  {% load static %}
2  <nav class="navbar navbar-light" style="background-color: lightskyblue">
3      <div class="container">
4          <a class="navbar-brand" href="{% url 'posts:index' %}">
5              
6              <span style="color:red">Ya</span>tube</a>
7          </a>
8          {% with request.resolver_match.view_name as view_name %}
9          <ul class="nav nav-pills">
10             <li class="nav-item">
11                 <a class="nav-link {% if view_name == 'about:author' %}active{% endif %}" href="{% url 'about:author' %}">Об авторе</a>
12             </li>
13             <li class="nav-item">
14                 <a class="nav-link {% if view_name == 'about:tech' %}active{% endif %}" href="{% url 'about:tech' %}">Технологии</a>
15             </li>
16             {% if user.is_authenticated %}
17             <li class="nav-item">
18                 <a class="nav-link" href="{% url 'posts:post_create' %}">Новая запись</a>
19             </li>
20             <li class="nav-item">
21                 <a class="nav-link link-light" href="">Изменить пароль</a>
22             </li>
23             <li class="nav-item">
24                 <a class="nav-link link-light" href="{% url 'users:logout' %}">Выйти</a>
25             </li>
26             <li>
27                 Пользователь: {{ user.username }}
28             </li>
29             {% else %}
30             <li class="nav-item">
31                 <a class="nav-link link-light" href="{% url 'users:login' %}">Войти</a>
32             </li>
33             <li class="nav-item">
34                 <a class="nav-link link-light" href="{% url 'users:signup' %}">Регистрация</a>
35             </li>
36             {% endif %}
37         </ul>
38     {% endwith %}
39     {# Конец добавленного в спринте #}
40 </div>
41 </nav>
42
43
```

## Footer.html

```
yatube > templates > includes > <> footer.html
1  <footer class="border-top text-center py-3">
2  |   <p>© {{ year }} Copyright <span style="color:red">Ya</span>tube</p>
3  </footer>
```

Далее для работы нам понадобятся шаблоны страниц, отвечающие нашим функциям приложения, и использующие при этом созданные нами основы разметки. Рассмотрим на примере главной страницы.

## Index.html

```
yatube > templates > posts > <> index.html
1  <!DOCTYPE html> <!-- Используется html 5 версии -->
2  {% extends 'base.html' %}
3  <title>
4  |   {% block title %}
5  |   Меч мечей
6  |   {% endblock title %}
7  </title>
8  {% block content %}
9  |   <div class="container py-5">
10 |     {% for post in page_obj %}
11 |     <ul>
12 |     |   <li>
13 |     |   |   Автор: {{ post.author.get_full_name }}
14 |     |   |   </li>
15 |     |   |   <li>
16 |     |   |   |   Дата публикации: {{ post.pub_date|date:"d E Y" }}
17 |     |   |   </li>
18 |     |   </ul>
19 |     <p>{{ post.text }}</p>
20 |     <p>
21 |     |   <a href="{% url 'posts:post_detail' post.pk %}">подробная информация </a>
22 |     </p>
23 |     {% if post.group %}
24 |     <a href="{% url 'posts:group_list' post.group.slug %}">все записи группы</a>
25 |     {% endif %}
26 |     {% if not forloop.last %}<hr>{% endif %}
27 |     {% endfor %}
28 |     {% include 'includes/paginator.html' %}
29 |   {% endblock content %}
30
```

Когда все страницы и их основы были созданы, наша директория шаблонов стала выглядеть следующим образом:

```
└─ templates ●
  └─ about ●
    ├── author.html U
    └── tech.html U
  └─ includes ●
    ├── footer.html U
    ├── header.html U
    └── paginator.html U
  └─ posts ●
    ├── create_post.html U
    ├── group_list.html U
    ├── index.html U
    ├── post_detail.html U
    └── profile.html U
  └─ users ●
    ├── logged_out.html U
    ├── login.html U
    ├── signup.html U
    └── base.html U
```



# URL-адреса и view-функции

Теперь, когда у нас уже существуют страницы в виде html-документов, их необходимо привязать к адресам, а адреса – к конкретным функциям для возможности перехода по ним. Для удобной работы с адресами, предназначенными для разных разделов сайта рекомендуется внутри нашей рабочей директории создавать отдельные папки, в которых адреса и функции будут находиться в соответствии с их предназначением, например: адреса и функции, связанные с постами, в одной папке, а адреса и функции, связанные с пользователем непосредственно – в другой.

Обратите внимание, что внутри файлов (в разных вложенных папках) *urls.py* находится только название html-документа, сам путь к нему и связанная с конкретным адресом view-функция.

```
yatube > posts > 📁 urls.py > ...
1  from django.urls import path
2
3  from . import views
4
5
6  app_name = 'posts'
7  urlpatterns = [
8      path('', views.index, name='index'),
9      path('group/<slug:slug>', views.group_posts, name='group_list'),
10     path('profile/<str:username>', views.profile, name='profile'),
11     path('posts/<int:post_id>', views.post_detail, name='post_detail'),
12     path('create/', views.post_create, name='post_create'),
13     path('posts/<post_id>/edit/', views.post_edit, name='post_edit'),
14 ]
15
```

Касаемо view-функций необходимо отметить, что в этой области заложен максимальный творческий потенциал к разработке. На скриншотах ниже можно заметить, что в нашем проекте предусмотрена ошибка №404 («Страница не найдена»), перенаправление пользователя на главную страницу (например, при нажатии на логотип сайта или на кнопку «главная»), а также невозможность оставлять и редактировать записи, если пользователь не вошел в аккаунт.

```
yatube > posts > views.py > post_create
1 from django.core.paginator import Paginator
2 from django.shortcuts import render, get_object_or_404, redirect
3 from django.contrib.auth.decorators import login_required
4
5 from .models import Group, Post, User
6 from .constants import POSTS_PER_PAGE
7 from .forms import PostForm
8
9
10 def index(request):
11     post_list = Post.objects.all().order_by('-pub_date')
12     paginator = Paginator(post_list, POSTS_PER_PAGE)
13     page_number = request.GET.get('page')
14     page_obj = paginator.get_page(page_number)
15     context = {
16         'page_obj': page_obj,
17     }
18
19     return render(request, 'posts/index.html', context)
20
21
22 def group_posts(request, slug):
23     group = get_object_or_404(Group, slug=slug)
24     posts = Post.objects.filter(group=group).order_by('-pub_date')[
25         :POSTS_PER_PAGE]
26     context = {
27         'group': group,
28         'posts': posts,
29     }
30
31     return render(request, 'posts/group_list.html', context)
32
33
34 def profile(request, username):
35     author = get_object_or_404(User, username=username)
36     posts = Post.objects.filter(author=author).order_by('-pub_date')[
37         :POSTS_PER_PAGE]
38     context = {
39         'author': author,
40         'posts': posts,
41     }
42
```

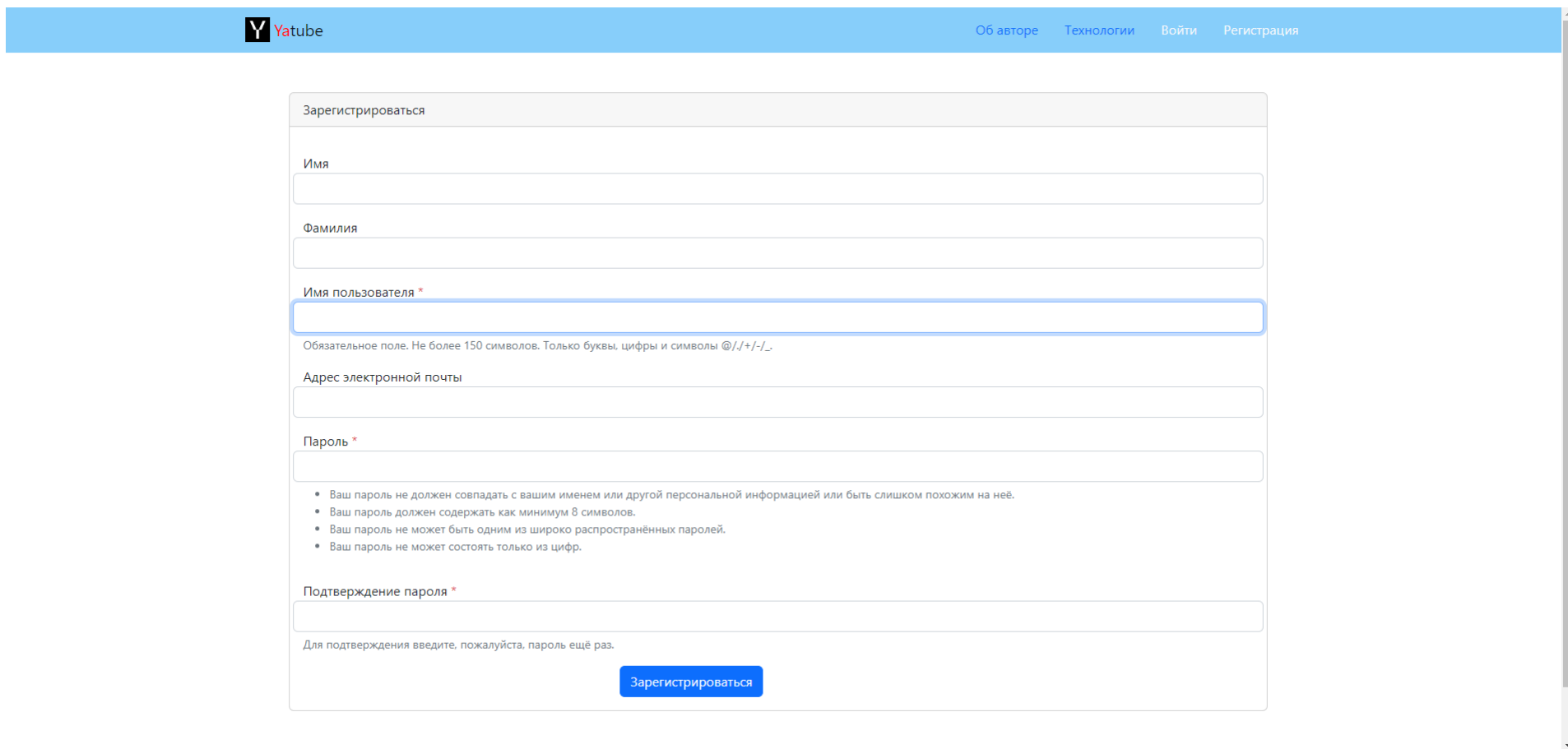
```
46 def post_detail(request, post_id):
47     post = get_object_or_404(Post, pk=post_id)
48     context = {
49         'post': post,
50     }
51
52     return render(request, 'posts/post_detail.html', context)
53
54
55 @login_required
56 def post_create(request):
57     form = PostForm(request.POST or None)
58     if form.is_valid():
59         group_id = form.cleaned_data['group']
60         text = form.cleaned_data['text']
61         if group_id:
62             group = Group.objects.get(id=group_id.id)
63         else:
64             group = None
65         form = Post.objects.create(
66             author=request.user,
67             text=text,
68             group=group
69         )
70         return redirect(
71             'posts:profile', request.user.username
72         )
73     context = {
74         'form': form,
75     }
76     return render(request, 'posts/create_post.html', context)
77
78
79 @login_required
80 def post_edit(request, post_id):
81     post = get_object_or_404(Post, pk=post_id)
82     form = PostForm(request.POST or None, instance=post)
83     if request.method == 'POST':
84         if form.is_valid():
85             form.save()
86             return redirect('posts:post_detail', post_id)
87     return render(request, 'posts/create_post.html', {'form': form})
88     form = PostForm(instance=post)
89     context = {
90         'form': form,
91         'post': post,
92         'is_edit': True,
93     }
```

В официальной документации можно узнать о всех нюансах соотношений view-функций, url-адресов и html-документов.

# Внешний вид

Для создания backend части нашего приложения всего вышеперечисленного будет достаточно. Благодаря Django также существует возможность работать и с оформлением веб-приложения. В нашем примере мы воспользуемся только логотипом, изменением цвета шрифта и фона в отдельных частях посредством языка разметки страницы.

## Страница регистрации пользователя



The screenshot shows a web browser window displaying a registration page for a site named 'Yatube'. The page has a light blue header with the 'Yatube' logo on the left and navigation links 'Об авторе', 'Технологии', 'Войти', and 'Регистрация' on the right. The main content area is a white box with a light gray border. At the top of this box is a tab labeled 'Зарегистрироваться'. Below the tab are several input fields: 'Имя', 'Фамилия', 'Имя пользователя \*' (which is highlighted with a blue border), 'Адрес электронной почты', and 'Пароль \*'. Below the password field is a list of four bullet points providing password requirements. At the bottom of the form is a 'Подтверждение пароля \*' field and a blue button labeled 'Зарегистрироваться'. A small note at the very bottom of the form says 'Для подтверждения введите, пожалуйста, пароль ещё раз.'

Yatube

Об авторе Технологии Войти Регистрация

Зарегистрироваться

Имя

Фамилия

Имя пользователя \*

Обязательное поле. Не более 150 символов. Только буквы, цифры и символы @/./+/-/\_.

Адрес электронной почты

Пароль \*


- Ваш пароль не должен совпадать с вашим именем или другой персональной информацией или быть слишком похожим на неё.
- Ваш пароль должен содержать как минимум 8 символов.
- Ваш пароль не может быть одним из широко распространённых паролей.
- Ваш пароль не может состоять только из цифр.

Подтверждение пароля \*

Для подтверждения введите, пожалуйста, пароль ещё раз.

Зарегистрироваться

## Страница входа в аккаунт



[Об авторе](#) [Технологии](#) [Войти](#) [Регистрация](#)

Войти на сайт

Имя пользователя \*

mega4elk

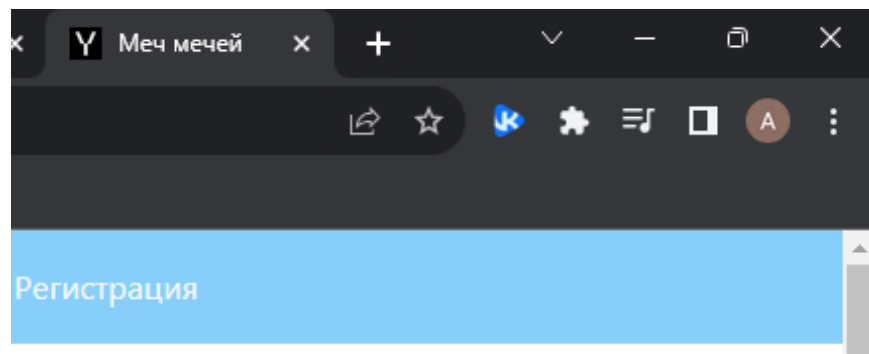
Пароль \*

\*\*\*\*\*

Войти

© 2023 Copyright Yatube

## Шапка страницы и логотип



Главная страница с авторизованным пользователем, добавленными записями и гиперссылками

[Об авторе](#)[Технологии](#)[Новая запись](#)[Изменить пароль](#)[Выйти](#)

Пользователь: mega4el

- Автор:
- Дата публикации: 23 ноября 2022

Утромъ гольдъ Дерсу Узала на повторно заданный вопросъ согласенъ ли онъ поступить проводникомъ изъявилъ свое согласіе и съ этого момента онъ сталъ членом экспедиціи

[подробная информация](#)

---

- Автор:
- Дата публикации: 23 ноября 2022

Oops, I did it again!

[подробная информация](#)

---

## Страница создания поста

Новый пост

Текст поста \*

Hello, World!

Текст нового поста

Группа

-----

-----

Группа: Лев Толстой – зеркало русской революции

Сохранить

Страница информации о посте

Yatube

Об авторе

Технологии

Новая запись

Изменить пароль

Выйти

Пользователь: mega4el

Дата публикации: 28 июля 1854

Группа: Писатели [все записи группы](#)

Автор: Лев Толстой

Всего постов автора:

36

[все посты пользователя](#)

Пишу въ самомъ пріятномъ веселомъ расположеніи духа, въ которомъ я провелъ весь вечеръ. Утро читаль, объѣдался грушами и вмѣсто обѣда Ълъ сыръ. Несмотря на кутежи у Сталыпина и Сержпутовскаго по случаю полученія наградъ, не завидоваль, а провелъ день весело. Вечеромъ хватилъ босонож[ку] и выпилъ бокала два шампанскаго съ Шварц[емъ], Вейлеровскимъ [?] и Гембичомъ, потомъ болталъ съ Шубинымъ и съ Сашей Горчаковымъ. — Исключая праздности, днемъ своимъ очень доволенъ. (1)

# Заключение

Django – очень удобный, простой и понятный фреймворк, а Python считается одним из самых простых языков для изучения. С их помощью можно научиться создавать как простые веб-приложения, подобные рассмотренному нами, так и более сложные с огромным множеством различных функций и уникальных возможностей. Более того, эти инструменты популярны и востребованы на рынке труда, а владение ими выгодно выделяет IT-специалиста. Существует большое количество сопутствующих инструментов и различных особенностей, которые не были полностью раскрыты в этой работе или не были упомянуты вовсе, что делает данную тему только интересней.

**Спасибо за внимание!**



# Список литературы и ресурсы

- 1. Аллен Тейлор - SQL для чайников.
- 2. Марк Лутц – Изучаем Python.
- 3. Семикопенко Алексей Алексеевич – HTML для начинающих.
- 4. <https://docs.github.com/ru/get-started/quickstart>
- 5. <https://pythonworld.ru/osnovy/pep-8-rukovodstvo-po-napisaniyu-koda-na-python.html>
- 6. <https://code.visualstudio.com/docs>
- 7. <https://www.python.org/doc/>
- 8. <https://dev.to/highcenburg/django-admin-startapp-vs-python-managepy-startapp-17ja>
- 9. <https://docs.djangoproject.com/en/4.2/intro/tutorial01/>
- 10. <https://learn.microsoft.com/ru-ru/sql/?view=sql-server-ver16>