Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Ордена Трудового Красного Знамени

федеральное государственное бюджетное

образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра МКиИТ

Проектирование клиент-серверных приложений

Лабораторная работа №2

“Создание web-страницы с текстом”

Выполнил:

студент 3 курса,

группы БФИ2001

Лушин Е. А.

Москва 2023

**Цель работы:** научиться создавать web-страницы с простым текстом и html-шаблоны, настраивать обработку статичных файлов для Django.

**Задание:**

Создание web-страницы с текстом

* Сделать так, чтобы по адресу <http://127.0.0.1:8000/hello/> возвращался тот же самый текст;
* Убрать указание типа возвращаемого ответа (если классу HttpResponse напрямую не указать тип ответа, то будет выставлено значение по умолчанию). Сравнить полученные результаты.

Работа с шаблонами в Django

* Добавить к созданной таблице две строки и один столбец;
* Убрать у созданной таблицы все границы;
* Сделать заголовки списков (нумерованного и маркированного) подзаголовками четвертого уровня;
* Создать абсолютно такой же шаблон, только поменяв название на «**static\_handler.html**». В следующих заданиях при выполнении этой лабораторной работы изменяйте именно новосозданный шаблон.

Настройка обработки статичных файлов для Django

* Установить для заголовка первого уровня шрифт с засечками;
* Добавить картинку высотой 30px;
* Изменить размер шрифта для подзаголовков четвертого уровня с 22px до 14px;
* Сделайте ширину таблицы на 100% экрана;

**Краткая теория**

**Работа с шаблонами в Django**

Как уже упоминалось ранее, шаблон в Django представляет собой строку текста, предназначенную для отделения представления документа от его данных. В шаблоне могут встречаться маркеры и простые логические конструкции (шаблонные теги), управляющие отображением документа. Обычно шаблоны применяются для создания HTML-разметки, но в Django они позволяют генерировать документы в любом текстовом формате.

В каком-то смысле шаблоны в Django — это некий аналог бланков из реального мира. Приходя в любое государственное учреждение, вы можете получить полупустую заготовку какого-либо документа, к примеру заявления, которое вы заполняете лишь отчасти, потому что общие части уже пропечатаны за вас. То есть шаблон — это основа будущей HTML-разметки, которая должна быть заполнена теми данными, что будут переданы в шаблон.

Эти данные, которые необходимо внести в шаблон, в сумме называются контекстом, а процесс, когда данные вносятся в шаблон, называется рендерингом.

Пример шаблона для Django:

*from django.template import Template*

*template = Template(‵‵<h1>Привет, моё имя {{name}}!</h1>′′)*

и если вы пожелаете на место переменной name поставить, например, имя «Василий», то контекстом будет обычный словарь из языка Python, оформленный в специальный класс Context примерно такого содержания:

*from django.template import Context*

*context = Context({‵‵ name′′ : u‵‵ Василий′′})*

Теперь осталось только совместить шаблон и контекст с помощью рендеринга. Рендеринг шаблонов в Django:

*html page = template.render(context)*

*# html page == ‵‵ <h1>Привет, моё имя Василий!</h1>′′*

Помимо прямого вывода переменной в будущий HTML-документ в шаблонах можно пользоваться некими аналогами конструкций из языка Python. Например, в шаблонах можно использовать циклы for и условия if.

Пример использования алгоритмических конструкций в шаблонах:

*from django.template import Context, Template*

*template = Template(‵‵′′′′*

*<h1>Привет, моё имя {{ my name }}!</h1>*

*{% for friend in friend list %}*

*{% if friend.is groupmate %}*

*<p>Мой одногруппник: {{ friend.name }}</p>*

*{% endif %}*

*{% endfor %}*

*‵‵′′′′)*

*context = Context({‵‵my name′′: u‵‵Василий′′,*

*‵‵friend list′′: [*

*{‵‵name′′: u‵‵Георгий′′, ‵‵is groupmate′′: False},*

*{‵‵name′′: u‵‵Леонид′′, ‵‵is groupmate′′: True},*

*{‵‵name′′: u‵‵Константин′′, ‵‵is groupmate′′: True},*

*{‵‵name′′: u‵‵Гавриил′′, ‵‵is groupmate′′: False}, ] })*

*html page = template.render(context)*

*# в итоге получится следующий html-код:*

*# <h1>Привет, моё имя Василий!</h1>*

*# <p>Мой одногруппник: Леонид</p>*

*# <p>Мой одногруппник: Константин</p>*

В данном примере прямо внутри шаблона был произведен обход по всем элементам массива friend list и для каждого из друзей была произведена проверка, является ли друг ещё и одногруппником, и если вся проверка успешно прошла, имя друга выводится в списке ниже.

Естественно, зашивать HTML-код прямо в представления на языке Python не очень хорошая штука (как и любое другое смешивание языков), поэтому хорошей, точнее сказать, обязательной, практикой является вынос шаблона в отдельный файл с форматом .html, и файл этот должен храниться в папке templates любого вашего приложения. Так что файловая структура ваших приложений в Djangoпроектах станет ещё чуточку сложнее. Теперь пример базового приложения стал шире.

Файловая структура приложения с шаблонами:

*your app/*

*templates/*

*# здесь будут храниться ваши шаблоны*

*\_\_init\_\_ .py*

*models.py*

*tests.py*

*views.py*

Также вам обязательно нужно знать, что создавать вручную объекты Template, затем объекты Context, а потом вызывать метод render первого, передавая туда контекст, необязательно. Естественно, как и все уважающие себя программисты, создатели Django создали автоматизированный вариант этого кода, который умещается всего. . . в одну строку кода, куда уж короче! Для этого была создана функция render, которая находится в модуле django.shortcuts. Пользоваться этой функцией очень просто.

Пример использования функции render():

*from django.shortcuts import render*

*def some view(request):*

*return render(request,*

*‵some template.html′,*

*{*

*‵context key1′:‵context value1′,*

*‵context key2′:‵context value2′,*

*}*

*)*

Здесь функции render в качестве первого аргумента передается объект запроса, который должен быть в каждом представлении, вторым аргументом идет имя шаблона, который сохранен в одной из папок templates ваших приложений, а третий аргумент — не что иное, как словарь, содержащий контекст вашего шаблона.

**Работа со статичными файлами в Django-приложениях**

Django-разработчики в основном работают с динамической частью приложения — представлениями и шаблонами, которые чаще всего изменяют свое содержимое при каждом запросе (например, страница профиля /profile/ будет у каждого пользователя разная, хотя каркас для всех будет общим). Но веб-приложения содержат и другую часть: статические файлы (изображения, CSS, Javascript и др.), которые не требуют никакой программной обработки. Для них нет потребности в рендеринге, они не зависят от содержимого базы данных. При каждом запросе к такому файлу веб-серверу достаточно просто вернуть их прямо такими, какими их сохранили в последний раз.

В больших проектах — особенно состоящих из десятков, а то и сотен приложений — работа с большим количеством файлов становится нелегким делом, потому что статические файлы расположены в разных папках. При базовой настройке ваши статичные файлы должны храниться в папке static каждого вашего приложения, и получается, что сколько приложений в вашем проекте, столько папок со статичными файлами нужно обрабатывать.

Для этого было создано приложение django.contrib.staticfiles: оно собирает статические файлы из всех ваших приложений (и остальных мест, которые вы укажите) в одном месте, что позволяет легко настроить выдачу статических файлов на реальном сервере.

Как уже было указано, в число статичных файлов обычно входят картинки всевозможных форматов (только если при каждом запросе вам нет нужды в дополнительной обработке изображения, однако такая потребность из разряда экзотических), сценарии на языке JavaScript и каскадные таблицы стилей (CSS, Cascading Style Sheets). О последних ниже будет краткое описание.

**Выполнение**

Для выполнения данной лабораторной работы создадим директорию Lab2, а в ней, создадим новый проект firstwebpage. Далее, перейдём в директорию firstwebpage и создадим в мой проект новое приложение под именем “app\_lushin\_evgeniy”. Все выше описанные действия продемонстрированы на рисунке 1.

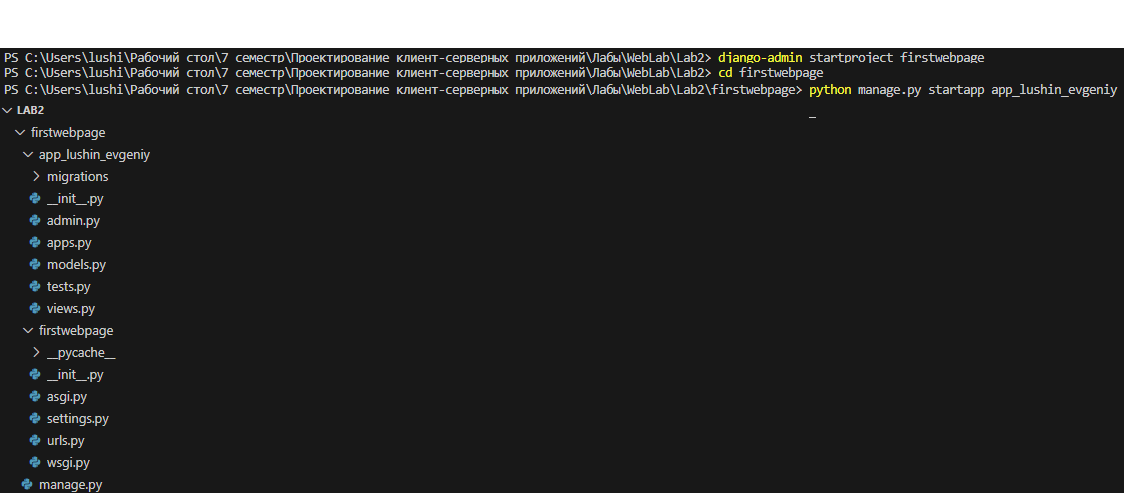


Рисунок 1 – Создание проекта и приложения

В начале работы над проектом необходимо задать базовые настройки: задать имя базы данных и создать ее таблицы, как это было сделано в лабораторной работе №1, а также добавить нужные приложения в проект.

Далее, чтобы добавить приложение app\_lushin\_evgeniy в проект, откроем файл settings.py, найдём кортеж INSTALLED\_APPS и добавим в конец элемента строку 'app\_lushin\_evgeniy', как показано на рисунке 2.

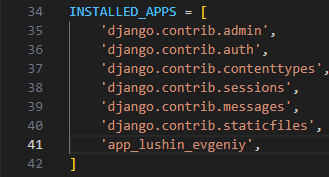
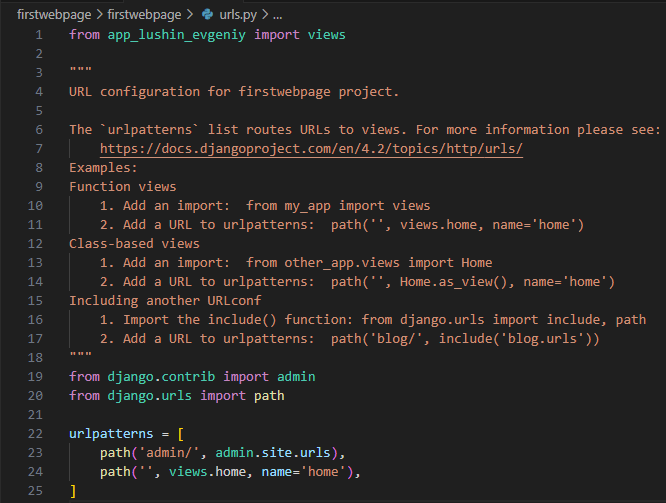


Рисунок 2 – Файл “settings.py”

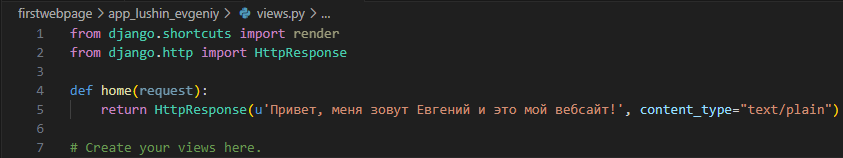
Перед началом работы с файлом urls.py необходимо импортировать views нашего приложения. А также для будущей страницы создадим новый адрес в этом же файле.

Листинг 1. Содержимое файла urls.py



Для того, чтобы в будущем при обращении генерировался ответ, необходимо создать функцию home в файле views.py в директории app\_lushin\_evgeniy.

Листинг 2. Содержимое файла views.py



Для запуска сервера выполним команду, которая запускает локальный сервер на порту 8000. Результат выполнения команды показан на рисунке 3.

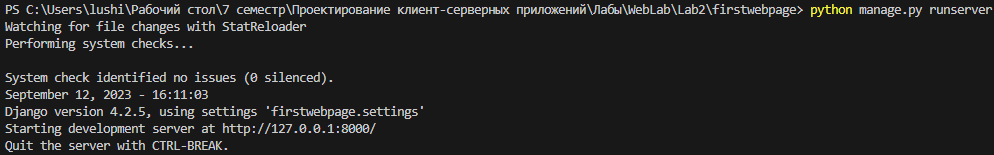


Рисунок 3 – Запуск сервера

После запуска сервера перейдём по адресу (<http://127.0.0.1:8000/>). Начальная страница продемонстрирована на рисунке 4.

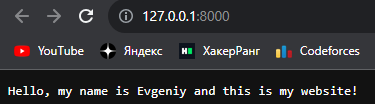
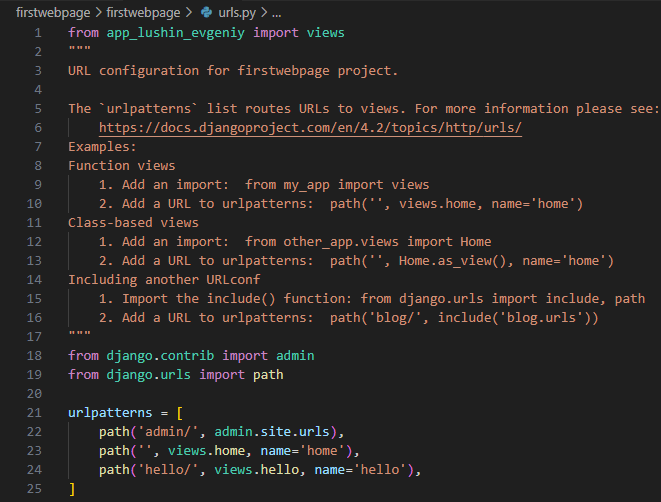


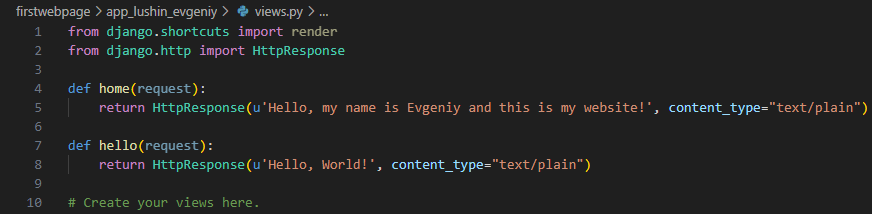
Рисунок 4 – Начальная страница вебсайта

Чтобы по адресу (<http://127.0.0.1:8000/hello/>) возвращался текст “Hello world!” необходимо в файл urls.py добавить следующую строку: **path('hello/', views.home, name='home')**. На рисунке 5 продемонстрирован вывод текста на сайте.

Листинг 3. Содержимое файла urls.py



Листинг 4. Содержимое файла views.py



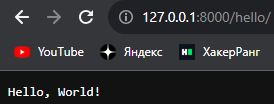
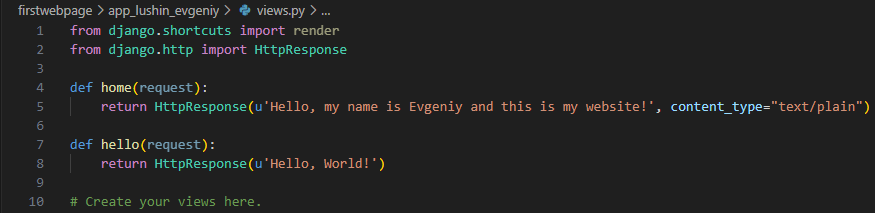


Рисунок 5 – Страница hello

Уберём тип возвращаемого ответа в функции hello.

Листинг 5. Содержимое файла views.py



Откроем страницу сайта (<http://127.0.0.1:8000/hello/>), после изменений можно заметить, что изменился шрифт нашего текста и фон сайта. Это продемонстрировано на рисунке 6.

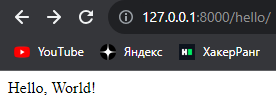


Рисунок 6 – Страница hello после изменения функции

Создадим папку templates в директории app\_lushin\_evgeniy, в данной папке создадим файл index.html. Структура проекта продемонстрирована на рисунке 7 на странице 11.

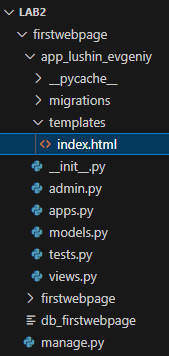
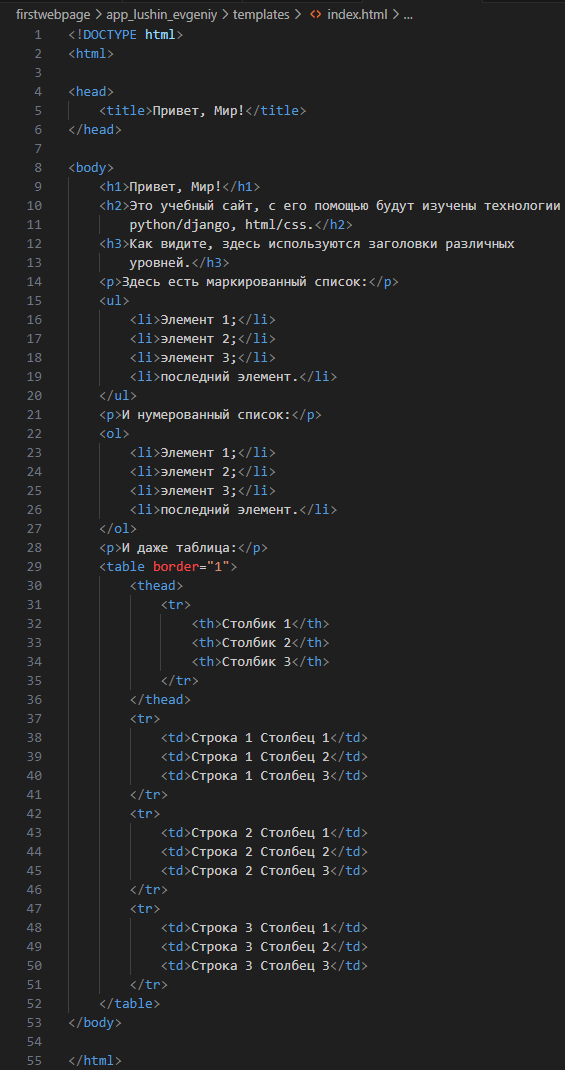
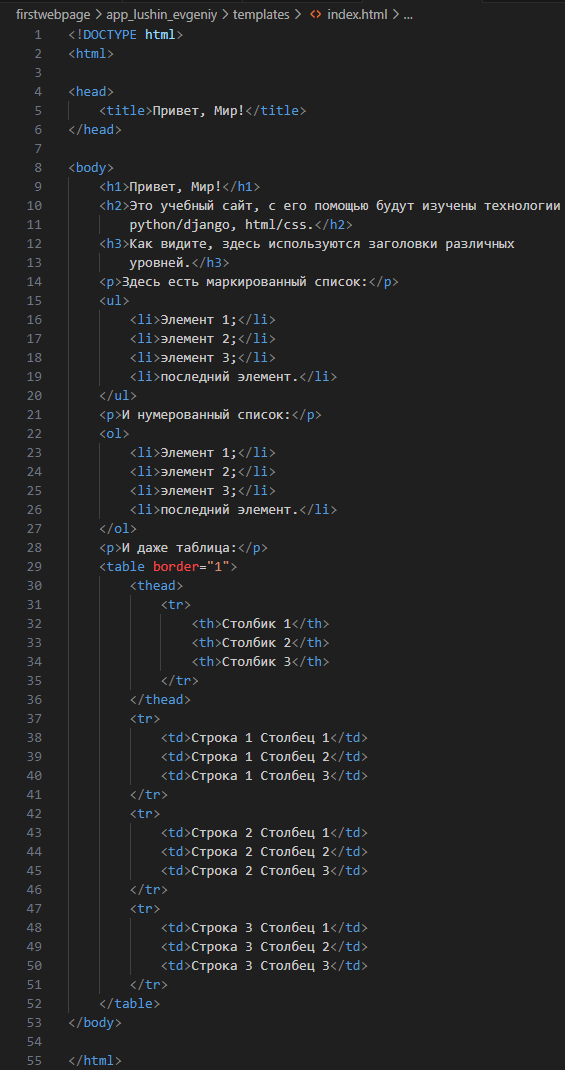


Рисунок 7 – Структура проекта Lab2

Код index.html представлен в листинге ниже.

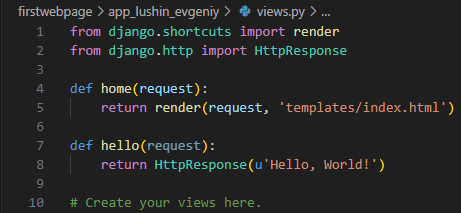
Листинг 6. Содержимое файла index.html





Созданный html-файл необходимо подключить к функции-представлению home. Для этого в файле views.py, который находится в директории app\_lushin\_evgeniy, добавим операции импортирования и функцию-представление home.

Листинг 7. Содержимое файла views.py



Также, для того чтобы файл index.html был найден в директории templates, в файле settings.py изменим поле DIRS в кортеже TEMPLATES. Поле DIRS должно содержать адрес директории, в которой располагается файл index.html. На рисунке 8 продемонстрирован изменённый кусок кода.

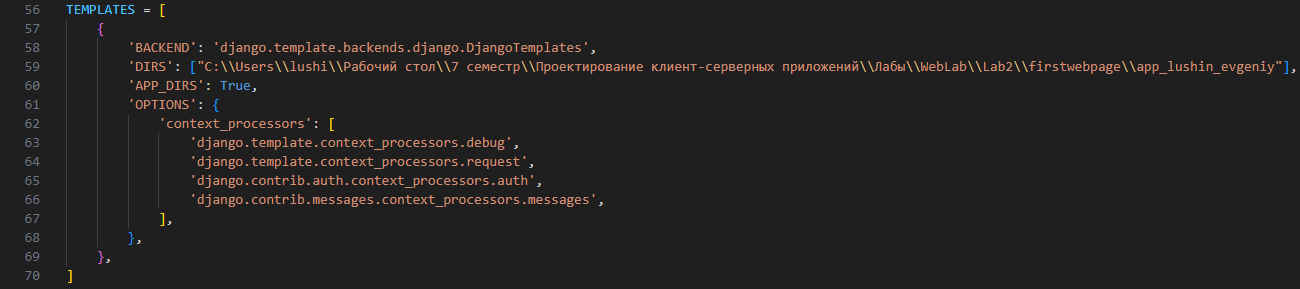


Рисунок 8 – Изменённый кортеж TEMPLATES

Проверим теперь, произошли ли изменения на сайте. На рисунке 9 продемонстрирована новая начальная страница.

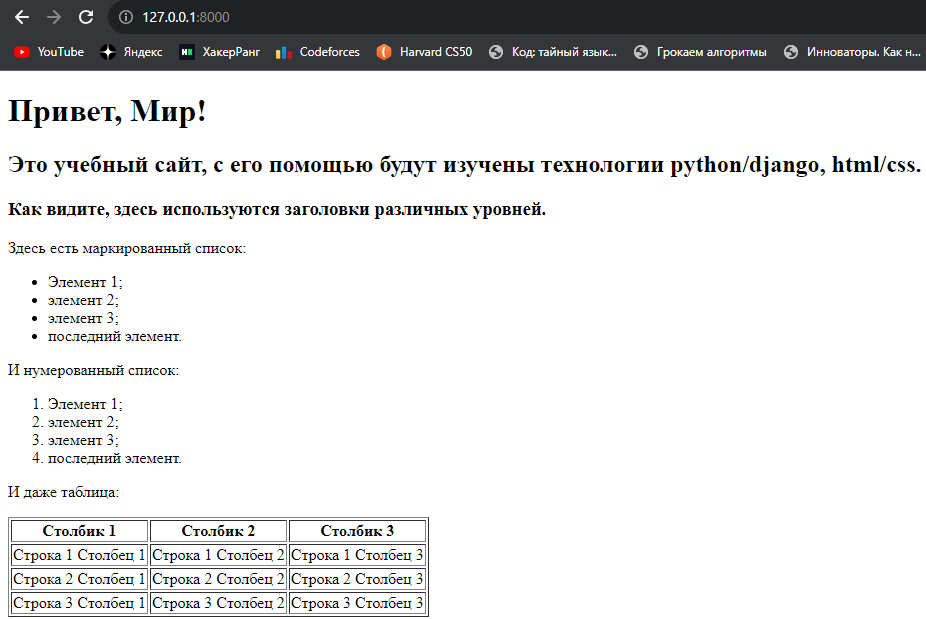


Рисунок 9 – Новая начальная страница

Создадим ещё 1 столбец и 2 строки к заданной таблице. Результат продемонстрирован на рисунке 10 на странице 15.

Листинг 8. Содержимое файла index.html

<!DOCTYPE html>

<html>

<head>

    <title>Привет, Мир!</title>

</head>

<body>

    <h1>Привет, Мир!</h1>

    <h2>Это учебный сайт, с его помощью будут изучены технологии

        python/django, html/css.</h2>

    <h3>Как видите, здесь используются заголовки различных

        уровней.</h3>

    <p>Здесь есть маркированный список:</p>

    <ul>

        <li>Элемент 1;</li>

        <li>элемент 2;</li>

        <li>элемент 3;</li>

        <li>последний элемент.</li>

    </ul>

    <p>И нумерованный список:</p>

    <ol>

        <li>Элемент 1;</li>

        <li>элемент 2;</li>

        <li>элемент 3;</li>

        <li>последний элемент.</li>

    </ol>

    <p>И даже таблица:</p>

    <table border="1">

        <thead>

            <tr>

                <th>Столбик 1</th>

                <th>Столбик 2</th>

                <th>Столбик 3</th>

                <th>Столбик 4</th>

            </tr>

        </thead>

        <tr>

            <td>Строка 1 Столбец 1</td>

            <td>Строка 1 Столбец 2</td>

            <td>Строка 1 Столбец 3</td>

            <td>Строка 1 Столбец 4</td>

        </tr>

        <tr>

            <td>Строка 2 Столбец 1</td>

            <td>Строка 2 Столбец 2</td>

            <td>Строка 2 Столбец 3</td>

            <td>Строка 2 Столбец 4</td>

        </tr>

        <tr>

            <td>Строка 3 Столбец 1</td>

            <td>Строка 3 Столбец 2</td>

            <td>Строка 3 Столбец 3</td>

            <td>Строка 3 Столбец 4</td>

        </tr>

        <tr>

            <td>Строка 4 Столбец 1</td>

            <td>Строка 4 Столбец 2</td>

            <td>Строка 4 Столбец 3</td>

            <td>Строка 4 Столбец 4</td>

        </tr>

        <tr>

            <td>Строка 5 Столбец 1</td>

            <td>Строка 5 Столбец 2</td>

            <td>Строка 5 Столбец 3</td>

            <td>Строка 5 Столбец 4</td>

        </tr>

    </table>

</body>

</html>

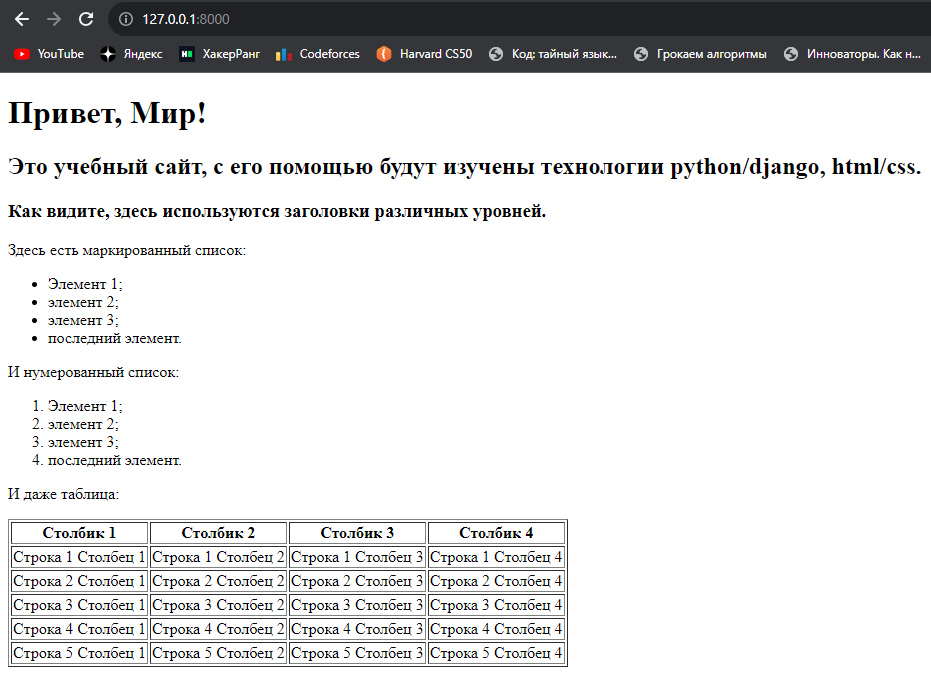


Рисунок 10 – Страница с увеличенной таблицей

Уберём все границы у нашей таблицы. Результат продемонстрирован на рисунке 11 на странице 17.

Листинг 9. Содержимое файла index.html

<!DOCTYPE html>

<html>

<head>

    <title>Привет, Мир!</title>

</head>

<body>

    <h1>Привет, Мир!</h1>

    <h2>Это учебный сайт, с его помощью будут изучены технологии

        python/django, html/css.</h2>

    <h3>Как видите, здесь используются заголовки различных

        уровней.</h3>

    <p>Здесь есть маркированный список:</p>

    <ul>

        <li>Элемент 1;</li>

        <li>элемент 2;</li>

        <li>элемент 3;</li>

        <li>последний элемент.</li>

    </ul>

    <p>И нумерованный список:</p>

    <ol>

        <li>Элемент 1;</li>

        <li>элемент 2;</li>

        <li>элемент 3;</li>

        <li>последний элемент.</li>

    </ol>

    <p>И даже таблица:</p>

    <table>

        <thead>

            <tr>

                <th>Столбик 1</th>

                <th>Столбик 2</th>

                <th>Столбик 3</th>

                <th>Столбик 4</th>

            </tr>

        </thead>

        <tr>

            <td>Строка 1 Столбец 1</td>

            <td>Строка 1 Столбец 2</td>

            <td>Строка 1 Столбец 3</td>

            <td>Строка 1 Столбец 4</td>

        </tr>

        <tr>

            <td>Строка 2 Столбец 1</td>

            <td>Строка 2 Столбец 2</td>

            <td>Строка 2 Столбец 3</td>

            <td>Строка 2 Столбец 4</td>

        </tr>

        <tr>

            <td>Строка 3 Столбец 1</td>

            <td>Строка 3 Столбец 2</td>

            <td>Строка 3 Столбец 3</td>

            <td>Строка 3 Столбец 4</td>

        </tr>

        <tr>

            <td>Строка 4 Столбец 1</td>

            <td>Строка 4 Столбец 2</td>

            <td>Строка 4 Столбец 3</td>

            <td>Строка 4 Столбец 4</td>

        </tr>

        <tr>

            <td>Строка 5 Столбец 1</td>

            <td>Строка 5 Столбец 2</td>

            <td>Строка 5 Столбец 3</td>

            <td>Строка 5 Столбец 4</td>

        </tr>

    </table>

</body>

</html>

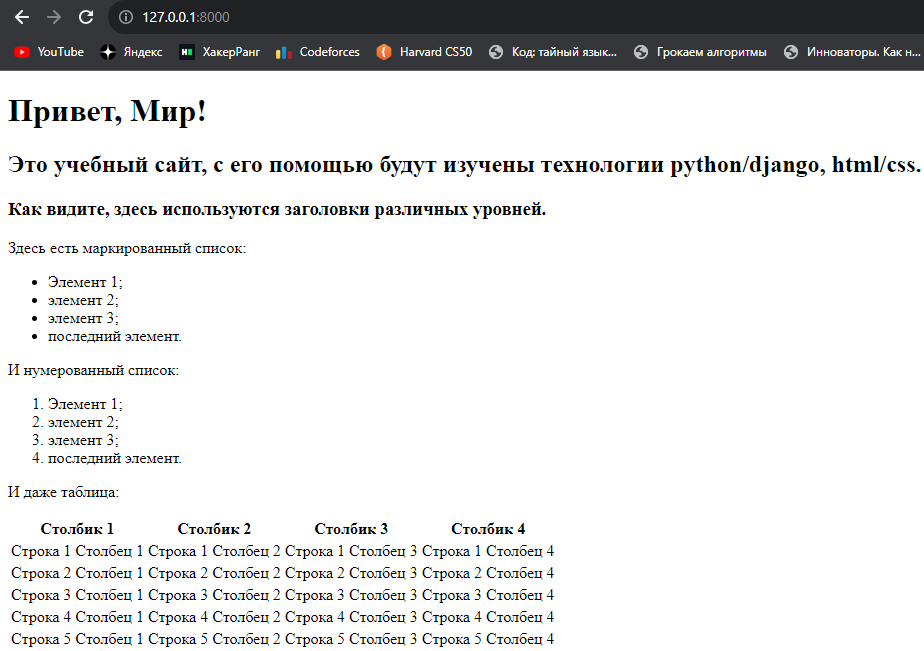


Рисунок 11 – Страница с таблицей без границ

Сделаем заголовки списков (нумерованного и маркированного) подзаголовками четвертого уровня. Результат продемонстрирован на рисунке 12 на странице 19.

Листинг 10. Содержимое файла index.html

<!DOCTYPE html>

<html>

<head>

    <title>Привет, Мир!</title>

</head>

<body>

    <h1>Привет, Мир!</h1>

    <h2>Это учебный сайт, с его помощью будут изучены технологии

        python/django, html/css.</h2>

    <h3>Как видите, здесь используются заголовки различных

        уровней.</h3>

    <h4>Здесь есть маркированный список:</h4>

    <ul>

        <li>Элемент 1;</li>

        <li>элемент 2;</li>

        <li>элемент 3;</li>

        <li>последний элемент.</li>

    </ul>

    <h4>И нумерованный список:</h4>

    <ol>

        <li>Элемент 1;</li>

        <li>элемент 2;</li>

        <li>элемент 3;</li>

        <li>последний элемент.</li>

    </ol>

    <p>И даже таблица:</p>

    <table>

        <thead>

            <tr>

                <th>Столбик 1</th>

                <th>Столбик 2</th>

                <th>Столбик 3</th>

                <th>Столбик 4</th>

            </tr>

        </thead>

        <tr>

            <td>Строка 1 Столбец 1</td>

            <td>Строка 1 Столбец 2</td>

            <td>Строка 1 Столбец 3</td>

            <td>Строка 1 Столбец 4</td>

        </tr>

        <tr>

            <td>Строка 2 Столбец 1</td>

            <td>Строка 2 Столбец 2</td>

            <td>Строка 2 Столбец 3</td>

            <td>Строка 2 Столбец 4</td>

        </tr>

        <tr>

            <td>Строка 3 Столбец 1</td>

            <td>Строка 3 Столбец 2</td>

            <td>Строка 3 Столбец 3</td>

            <td>Строка 3 Столбец 4</td>

        </tr>

        <tr>

            <td>Строка 4 Столбец 1</td>

            <td>Строка 4 Столбец 2</td>

            <td>Строка 4 Столбец 3</td>

            <td>Строка 4 Столбец 4</td>

        </tr>

        <tr>

            <td>Строка 5 Столбец 1</td>

            <td>Строка 5 Столбец 2</td>

            <td>Строка 5 Столбец 3</td>

            <td>Строка 5 Столбец 4</td>

        </tr>

    </table>

</body>

</html>

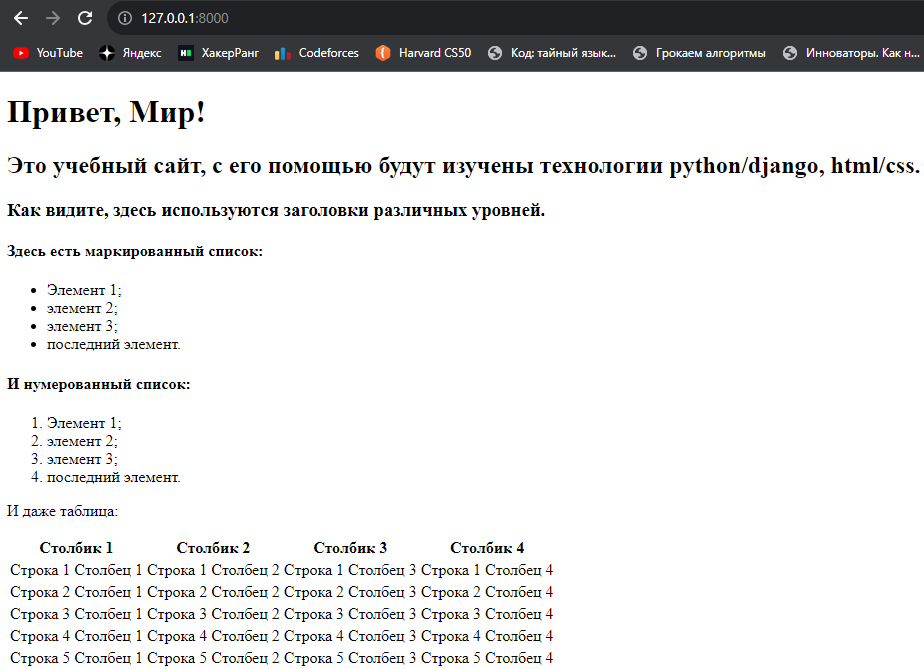


Рисунок 12 – Страница с новыми заголовками списков

Создадим идентичный шаблон с названием: **static\_handler.html**. Для того, чтобы придать документу стиль в соответствии с макетом, будем использовать CSS. Создадим папку static в директории app\_lushin\_evgeniy, а в ней файл index.css. Структура проекта продемонстрирована на рисунке 13.

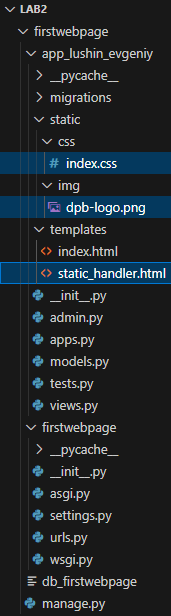


Рисунок 13 – Структура проекта Lab2

В файл страницы static\_handler.html во внутрь тега <head> вставим тег подключения css-скрипта. Установим для заголовка первого уровня шрифт с засечками, добавим картинку и сделаем ее высотой 30px, изменим размер шрифта для подзаголовков четвертого уровня, сделаем ширину таблицы на 100% экрана. Результат продемонстрирован на рисунке 14 на странице 23.

Листинг 11. Содержимое файла static\_handler.html

{% load static %}

<!DOCTYPE html>

<html>

<head>

    <link rel="stylesheet" type="text/css" href="{% static 'css/index.css' %}">

    <title>Привет, Мир!</title>

</head>

<body>

    <h1>Привет, Мир!</h1>

    <h2>Это учебный сайт, с его помощью будут изучены технологии

        python/django, html/css.</h2>

    <h3>Как видите, здесь используются заголовки различных

        уровней.</h3>

        <img src="{% static 'img/dpb-logo.png' %}" alt="Логотип" type="image/png"/>

    <h4>Здесь есть маркированный список:</h4>

    <ul>

        <li>Элемент 1;</li>

        <li>элемент 2;</li>

        <li>элемент 3;</li>

        <li>последний элемент.</li>

    </ul>

    <h4>И нумерованный список:</h4>

    <ol>

        <li>Элемент 1;</li>

        <li>элемент 2;</li>

        <li>элемент 3;</li>

        <li>последний элемент.</li>

    </ol>

    <p>И даже таблица:</p>

    <table>

        <thead>

            <tr>

                <th>Столбик 1</th>

                <th>Столбик 2</th>

                <th>Столбик 3</th>

                <th>Столбик 4</th>

            </tr>

        </thead>

        <tr>

            <td>Строка 1 Столбец 1</td>

            <td>Строка 1 Столбец 2</td>

            <td>Строка 1 Столбец 3</td>

            <td>Строка 1 Столбец 4</td>

        </tr>

        <tr>

            <td>Строка 2 Столбец 1</td>

            <td>Строка 2 Столбец 2</td>

            <td>Строка 2 Столбец 3</td>

            <td>Строка 2 Столбец 4</td>

        </tr>

        <tr>

            <td>Строка 3 Столбец 1</td>

            <td>Строка 3 Столбец 2</td>

            <td>Строка 3 Столбец 3</td>

            <td>Строка 3 Столбец 4</td>

        </tr>

        <tr>

            <td>Строка 4 Столбец 1</td>

            <td>Строка 4 Столбец 2</td>

            <td>Строка 4 Столбец 3</td>

            <td>Строка 4 Столбец 4</td>

        </tr>

        <tr>

            <td>Строка 5 Столбец 1</td>

            <td>Строка 5 Столбец 2</td>

            <td>Строка 5 Столбец 3</td>

            <td>Строка 5 Столбец 4</td>

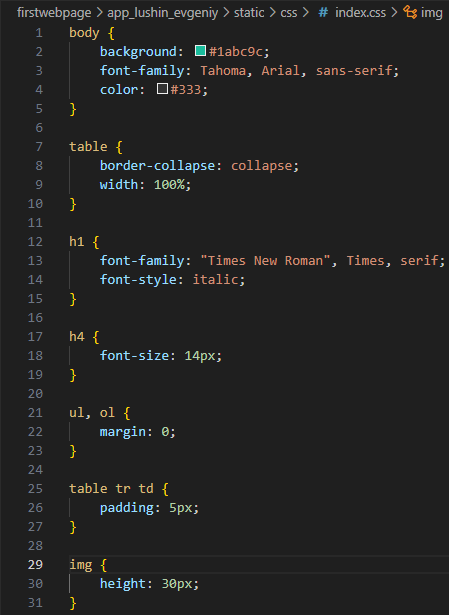
        </tr>

    </table>

</body>

</html>

Листинг 12. Содержимое файла index.css



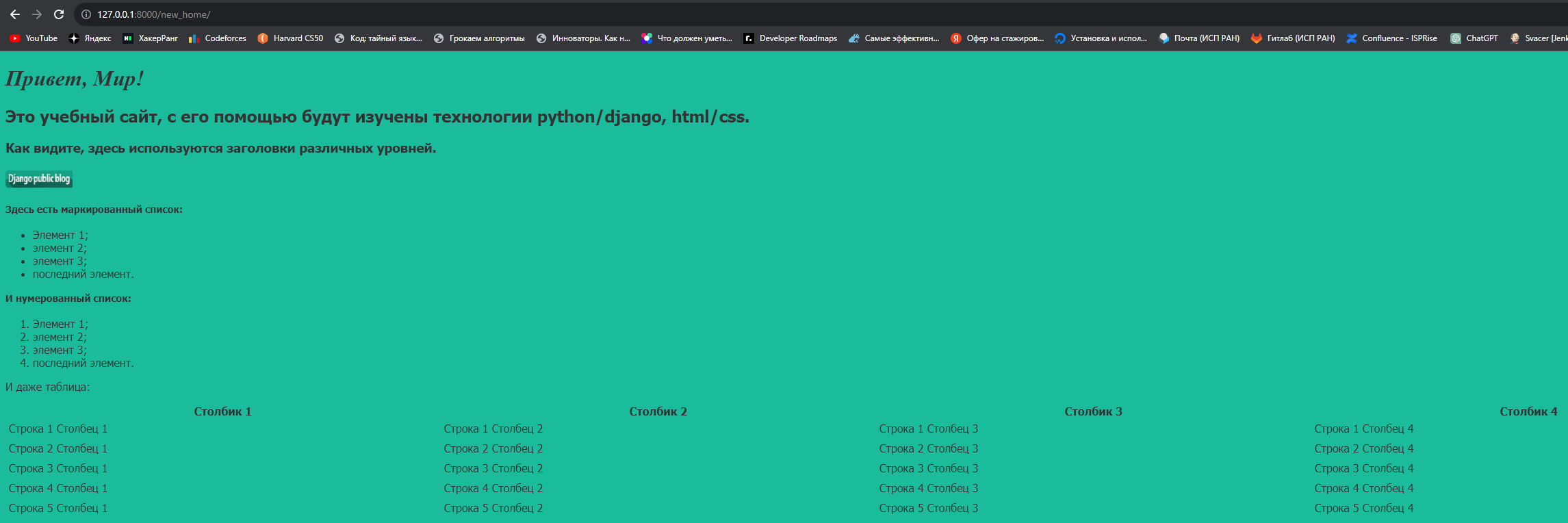


Рисунок 14 – Страница со стилем из шаблона

**Вывод:** В данной лабораторной работе я научился создавать web-страницы с простым текстом и html-шаблоны, а также настраивать обработку статичных файлов для Django.