Содержание

[Введение 5](#_Toc170731613)

[**АСУП** 7](#_Toc170731614)

[Выполнение задачи 7](#_Toc170731615)

[**Описание интерфейса сайта** 10](#_Toc170731616)

[**Заключение** 13](#_Toc170731617)

[**Список использованной литературы** 13](#_Toc170731618)

[**Приложение** 14](#_Toc170731619)

[**(Обязательное)** 14](#_Toc170731620)

[**Листинг программы** 14](#_Toc170731621)

Введение

Целью данной учебной практики является создание сайта для личного блога с использованием фреймворка Django языка программирования Python.

Задачи данной практики:

* ознакомиться с основами работы фреймворка Django;
* ознакомиться с работой с базой данных в фреймворке Django;
* ознакомиться с языком гипертекстовой разметки HTML;
* ознакомиться с формальным языком декорирования и описания внешнего вида документа CSS;
* разработать функционирующий сайт личного блога.

Django - это специальный инструмент, который помогает разработчикам создавать веб-сайты и приложения. Он называется фреймворком, потому что предоставляет готовые инструменты и правила для работы с веб-сайтами.

Django нужен для того, чтобы упростить процесс разработки веб-сайта. Вместо того, чтобы создавать каждую часть сайта с нуля, разработчики могут использовать Django, чтобы получить уже готовые компоненты и функциональность.

Django имеет множество встроенных инструментов для создания веб-сайтов по готовым шаблонам, такие как классы представления, шаблонизатор и др. В наши дни веб-разработчику на Python жизненно необходимо иметь опыт работы с этим фреймворком.

Django также имеет инструментарий для работы с базами данных с помощью понятных для программистов функций и объектов языка Python. Подавляющая часть современных сайтов содержит базу данных для хранения информации, и умение работы с базами данных так же является крайне важным навыком для веб-разработчика на Python.

Язык гипертекстовой разметки HTML и язык таблицы каскадных стилей CSS являются базой каждой веб-страницы. С помощью HTML пишется текст сайта и оформляется в отдельные блоки, к тексту применяются различные теги, меняющие его внешний вид и организацию текста на сайте. С помощью CSS веб-страница стилизуется, меняется расположение блоков текста, цвета, шрифт, размер текста и многое другое. Без знания этих языков невозможно создать даже веб-страницу, не говоря уже о сайте.

# **АСУП**

<https://technogroupp.com/articles/avtomatizirovannaya-sistema-upravleniya-predpriyatiem/>

<https://avielsy.ru/informatsiya/blog/93-avtomatizirovannaya-sistema-upravleniya-proizvodstvom>

<https://soware.ru/categories/enterprise-management-systems>

<https://saby.ru/accounting?utm_source=external&utm_medium=foc&utm_campaign=ex_buhgalteriya&utm_term=saby-bu&utm_content=buhgalteriya_ex_soware>

<https://a0.group/blog/avtomatizirovannaya-sistema-upravleniya-predpriyatiem/>

Выполнение задачи

Первое, что необходимо сделать, это создать новое виртуальное окружение, в котором будет вестись работа. Это нужно для того, чтобы при создании проекта сохранялись версии импортируемых модулей, с помощью которых создается проект. В противном случае, после обновления какого-либо модуля работоспособность проекта может быть нарушена. Создание окружения осуществляется одной командой в командной строке и не представляет сложностей.

Затем необходимо создать проект Django, что тоже делается одной командой. После создания проекта необходимо так же командой создать приложение внутри проекта Django. Фреймворк сам создает все необходимые директории и файлы. Созданные файлы включают в себя все необходимые инструменты для реализации веб-сайта. Наиболее интересными являются файлы urls.py и views.py. Файл urls.py содержит список возможных адресов, по которым сможет перейти пользователь, а файл views.py содержит программную реализацию всех маршрутов сайта. В самом простом случае, когда пользователь просто переходит на определенный адрес и получает определенную HTML страницу, в Django было бы необходимо прописать сам путь в urls.py и его реализацию в views.py.

Реализацию пути в файле views.py можно сделать в качестве функции представления или класса представления. Функция представления позволяет определить блок программного кода, который будет исполняться при переходе по определенному маршруту.

Класс представления работает несколько иначе. Класс представления наследуется от так называемого «generic view», это определенный внутри Django класс, организующий тот или иной вывод страницы. Классы представления зачастую взаимодействуют с БД, что будет кстати при выполнении задачи.

Для работы с базами данных в Django предусмотрена свой подход, позволяющий использовать интуитивно понятный программный код для обращения к базе данных. Так же Django позволяет создавать свои «модели», по сути классы, содержащие информацию о полях таблицы базы данных. Благодаря такой модели можно создавать, изменять и удалять таблицы и данные в них, а также использовать их в коде как объекты языка Python. Такая работа с базой данных будет реализована при создании сайта.

Также одним из основных понятий в Django являются шаблоны и статические файлы. Шаблоны – это HTML страницы, содержащие блоки, которые могут быть расширены другими HTML документами. На практике это позволяет создать, например, главную страницу с ее верхней и нижней оформленными частями, а блок посередине дополнять необходимым HTML кодом. Этот механизм будет повсеместно использован в реализуемом сайте. Например, для создания страницы для отображения конкретного поста в блоге будет использован один и тот же шаблон, который будет расширяться блоком текста из базы данных. Статические файлы – это любые файлы, которые не изменяясь возвращается пользователю: картинки, видео, файлы css, js и т.д. Они хранятся в определенных папках проекта, как правило называемых static, и заполняют шаблон в определенном месте. Реализуемый сайт будет использовать статические файлы, такие как картинки и css-файлы.

Для начала необходимо создать главную страницу сайта. Для этого в вышеупомянутых файлах проекта создается путь, в случае главной страницы он остается пустым, и его реализация. Реализация на данном этапе будет представлять из себя простой возврат определенной HTML страницы. Шаблон главной страницы создается с помощью языков HTML и CSS. Страница будет иметь верхнюю оформленную часть (header) и нижнюю часть (footer). Посередине будет располагаться расширяемый блок, который в будущем будет расширен другими файлами. Можно запустить локальный сервер и посмотреть на результат.

Затем необходимо создать модель для таблицы базы данных, которая будет содержать статьи блога. Это делается в отдельном файле под названием models.py. Таблица статей будет содержать следующие поля: ID, название, содержание, дата создания и дата последнего изменения.

Затем необходимо выводить некоторое количество статей из базы данных на главной странице. Это можно реализовать классом представления, наследуемым от ListView, что и было сделано. Для этого класса создаем новый шаблон, который будет расширять шаблон главной страницы и содержать в себе вывод статей. Шаблонизатор позволяет вставлять в шаблоны код языка Python используя особый синтаксис. Статьи будут выводится не полностью, а только первые несколько слов, после которых будет следовать кнопка, приглашающая прочитать пост целиком. После того как некоторое количество статей выводится, надо добавить возможность листать страницы со статьями, то есть организовать на сайте постраничную навигацию. Метод ее создания подробно описан в документации Django.

После того, как реализован вывод статей и постраничная навигация, реализуем вывод полного поста. Для этого кнопке возле каждой неполной статьи предпишем передавать на сервер определенное значение, а именно ID поста. По этому ID из базы данных будет выбран нужный пост, содержание которого из соответствующего поля таблицы будет помещён в шаблон главной страницы.

Далее реализуем регистрацию и авторизацию пользователей. В Django есть инструменты, позволяющие быстро реализовать эти функции, в том числе встроенная таблица для пользователей. Для регистрации и авторизации пользователей используем встроенные в Django формы, стилизуя их под реализуемый сайт.

Далее реализуем добавление комментариев авторизованными пользователями. Это потребует написания своей формы, которая будет принимать написанный текст от пользователя. Так же потребуется создание еще одной модели, которая охарактеризует таблицу с комментариями в базе данных. Она будет содержать такие поля как ID пользователя, ID статьи, время создания и содержание. Шаблонизатор позволяет отображать определенную часть страницы только при выполнении какого-либо условия. Неавторизованным пользователям будем выводить предложение авторизоваться, а авторизованным поле для ввода текста комментария. При получении данных из поля ввода, будем создавать новую запись в таблице комментариев. Вывод комментариев добавим в шаблон отображение полного поста. Все записи из таблицы комментариев будут появляться на странице поста, благодаря чему новый комментарий сразу же попадет на свое место.

Для дальнейшего заполнения сайта статьями необходима админ панель сайта. Django предоставляет свою функциональную админ панель, для использования которой необходимо лишь зарегистрировать нужные модели в файле admin.py. После этого можно будет через интерфейс админ панели управлять данными в базе данных сайта.

# **Описание интерфейса сайта**

При переходе на сайт пользователь попадает на главную страницу (Рисунок 1).

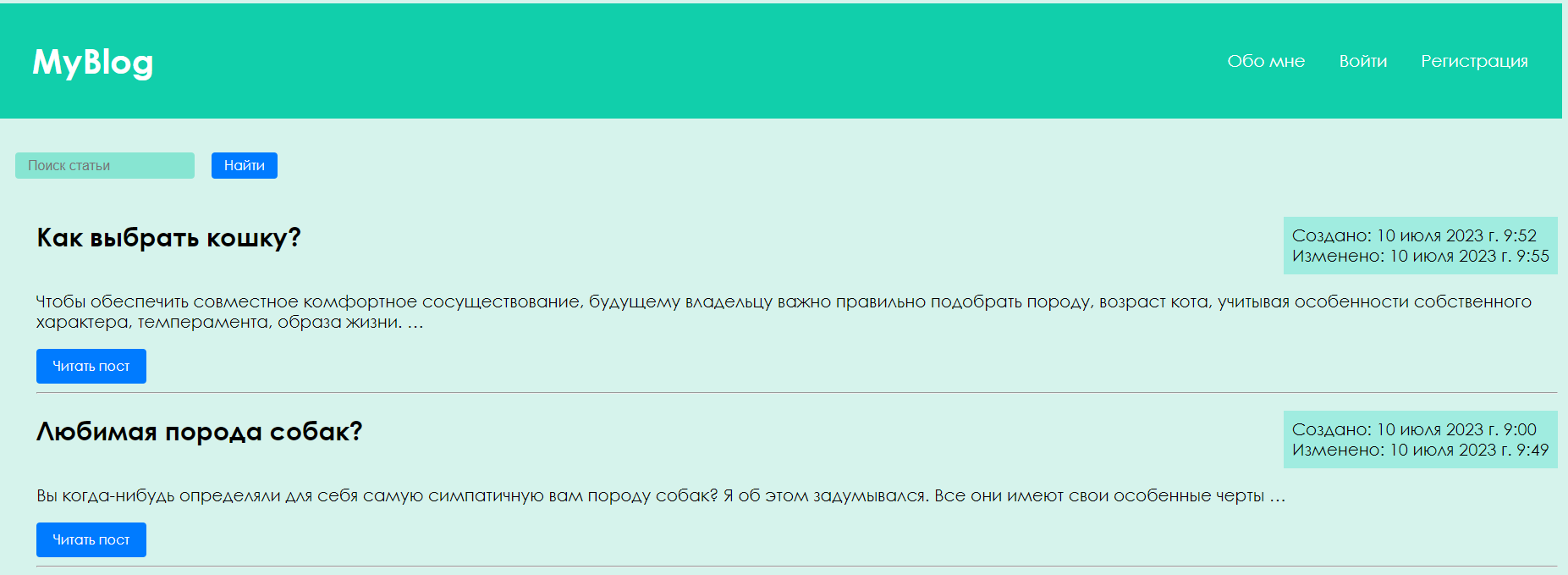


Рисунок 1 – Главная страница сайта

На ней пользователь может выполнить поиск поста по названию, перейти к полной версии поста, зарегистрироваться, войти в учетную запись, или выйти из учетной записи в случае если пользователь авторизован.

При регистрации или входе открываются окна с соответствующими формами (Рисунки 2, 3).

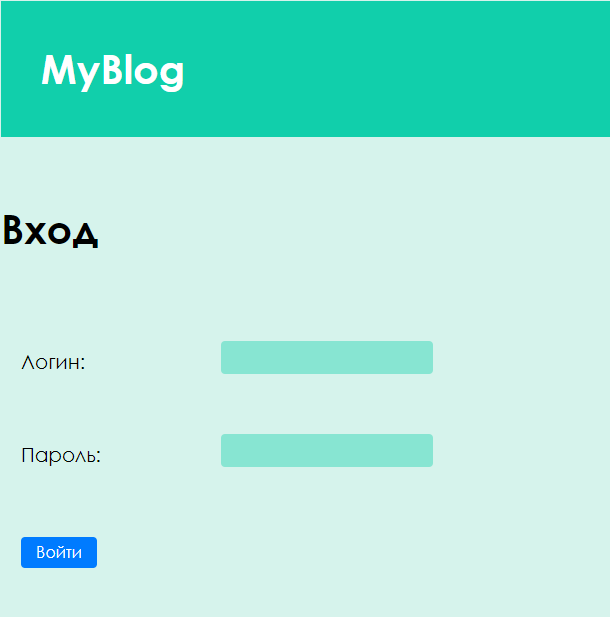


Рисунок 2 – Форма авторизации

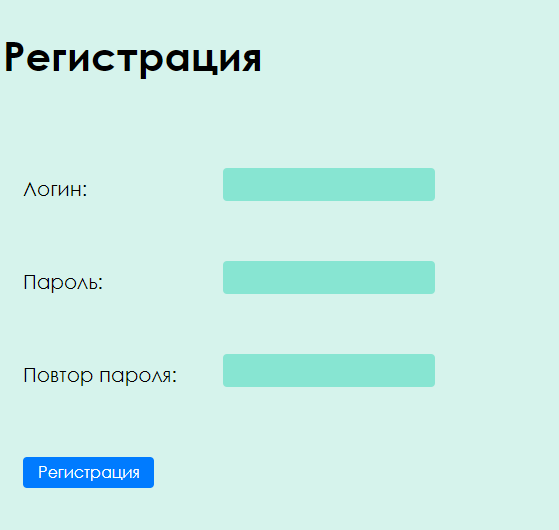


Рисунок 3 – Форма регистрации

Перейдя к полному описанию поста, пользователь увидит весь текст поста, а ниже приглашение к вводу комментария. В случае, если пользователь не авторизован, он увидит строчку, приглашающую авторизоваться на сайте (Рисунки 4,5).

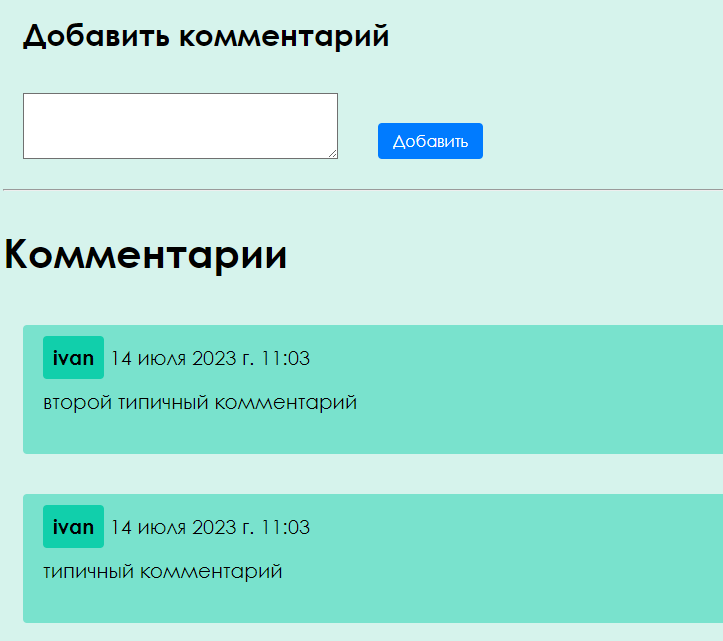


Рисунок 4 – Вид комментариев для авторизованного пользователя



Рисунок 5 – Вид комментариев для неавторизованного пользователя

Перейдя в админ панель сайта можно обращаться к базе данных через ее интерфейс (Рисунок 6).

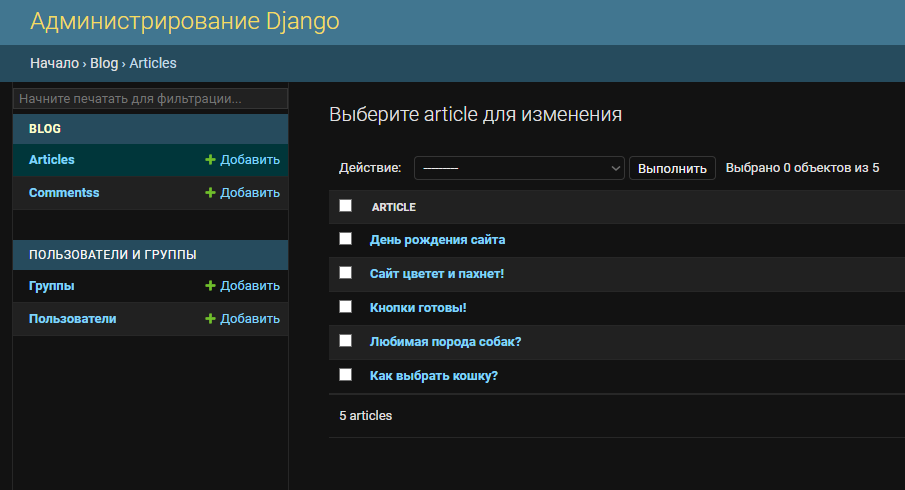


Рисунок 6 – Админ панель сайта

# **Заключение**

В ходе выполнения задания были выполнены все поставленные задачи. Я ознакомился с основами работы Django, языком гипертекстовой разметки HTML, языком каскадных стилей CSS и создал функциональный сайт личного блога.

# **Список использованной литературы**

1. Владимир Дронов, «Django: практика создания Web-сайтов на Python»
2. Анатолий Постолит «Python, Django и PyCharm для начинающих»

**Приложение**

**(Обязательное)**

**Листинг программы**