МИНИСТЕРСТВО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ

Государственное бюджетное профессиональное образовательное учреждение Московской области «Ногинский колледж»

Курсовой проект

по МДК.09.01 Проектирование и разработка веб-приложений

ПМ.09 Проектирование, разработка и оптимизация веб-приложений

Тема:

ИСПОЛЬЗОВАНИЕ ФРЕЙМВОРКА DJANGO В РАЗРАБОТКЕ ВЕБ-ПРИЛОЖЕНИЯ «БЛОГ-ПЛАТФОРМА»

Разработчик:

студент группы 3ИСПР

Хомутенников В.Р.

(подпись)

Оценка защиты курсового проекта

Руководитель проекта:

преподаватель

Степанов С.О.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

« »

Дата защиты

« » 2025г.

Ногинск, 2025 г

Содержание

[ВВЕДЕНИЕ 3](#_Toc199627714)

[1. Описание предметной области 5](#_Toc199627715)

[1.1 Характеристика веб-приложения 5](#_Toc199627716)

[1.2 Описание программных инструментов и средств разработки 7](#_Toc199627717)

[1.2.1 Язык программирования «Python» 7](#_Toc199627718)

[1.2.2 Фреймворк «Django» 8](#_Toc199627719)

[1.2.3 Серверная часть 9](#_Toc199627720)

[1.2.4 Среда разработки 10](#_Toc199627721)

[2. Практическая часть 12](#_Toc199627722)

[2.1 Проектирование веб-приложения 12](#_Toc199627723)

[2.2 Подготовка к работе 14](#_Toc199627724)

[2.3 Разработка веб-приложения 16](#_Toc199627725)

[2.3.1 Создание моделей и структуры базы данных 16](#_Toc199627726)

[2.3.2 Настройка маршрутов (URLconf) 18](#_Toc199627727)

[2.3.3 Разработка представлений (Views) 19](#_Toc199627728)

[2.3.4 Создание шаблонов (Templates) 20](#_Toc199627729)

[2.3.5 Работа с формами 21](#_Toc199627730)

[2.3.6 Верстка и стилизация 22](#_Toc199627731)

[2.3.7 Обработка медиафайлов 23](#_Toc199627732)

[2.4 Размещение веб-приложения на хостинге 24](#_Toc199627733)

[Заключение 26](#_Toc199627734)

[Список использованной литературы 28](#_Toc199627735)

[ПРИЛОЖЕНИЕ 29](#_Toc199627736)

# ВВЕДЕНИЕ

Развитие информационных технологий и широкое распространение интернета способствуют росту популярности различных веб-приложений, среди которых особое место занимают блог-платформы. Такие сервисы позволяют пользователям публиковать собственные материалы, делиться мнениями и обмениваться информацией. Для создания современных, надёжных и масштабируемых веб-приложений активно используются мощные инструменты разработки, такие как язык программирования Python и фреймворк Django.

**Целью курсового проекта** является проектирование и разработка веб-приложения — блог-платформы с использованием языка Python и фреймворка Django.

Для достижения поставленной цели необходимо решить следующие задачи:

* изучить особенности работы с фреймворком Django;
* спроектировать архитектуру веб-приложения;
* разработать пользовательский интерфейс и серверную часть системы;
* реализовать основные функции блог-платформы: регистрацию пользователей, создание и редактирование публикаций, комментирование записей;
* протестировать и оценить работоспособность созданного приложения.

**Объектом курсового проекта** является процесс разработки веб-приложений на фреймворке Django.

**Предметом курсового проекта** является проектирование и реализация блог-платформы с использованием возможностей фреймворка Django.

**Актуальность темы** обусловлена высоким спросом на функциональные и удобные веб-приложения для обмена информацией в интернете. Django как один из самых популярных фреймворков для Python обеспечивает быструю разработку, безопасность и масштабируемость проектов, что делает его идеальным инструментом для создания современных блог-платформ.

# 1. Описание предметной области

## 1.1 Характеристика веб-приложения

Веб-приложение — это программное обеспечение, которое работает в веб-браузере и взаимодействует с пользователем через интернет. В отличие от традиционных десктопных программ, веб-приложения не требуют установки на компьютер — для их использования достаточно иметь устройство с доступом в интернет и браузер. Они размещаются на удалённых серверах, и все вычисления и обработка данных происходят на стороне сервера, а пользователю передаётся только результат в виде веб-страницы.

Примеры веб-приложений включают интернет-магазины, онлайн-банкинг, социальные сети, форумы, почтовые сервисы, блоги и многие другие сервисы, с которыми мы сталкиваемся ежедневно. Такие приложения позволяют выполнять широкий спектр задач: от чтения новостей до оформления заказов, общения, публикации контента и управления бизнесом.

Главная особенность веб-приложения заключается в том, что пользователь взаимодействует с ним через интерфейс в браузере. Данные, которые вводит пользователь (например, логин и пароль, форма заказа или комментарий), отправляются на сервер, обрабатываются, и обратно возвращается результат. Это обеспечивает высокую гибкость, возможность централизованного обновления (не нужно обновлять программу на каждом устройстве) и удобный доступ с любых устройств.

Современные веб-приложения разрабатываются с использованием языков и технологий, таких как HTML, CSS, JavaScript (на клиентской стороне), а также Python, PHP, Ruby, Java, Node.js и других (на серверной стороне). Для упрощения и ускорения разработки активно используются фреймворки — готовые инструменты и библиотеки, позволяющие быстро создать устойчивое, безопасное и масштабируемое приложение.

Блог-платформа — это специализированное веб-приложение или онлайн-сервис, предназначенный для создания, публикации и управления блогами. Блог представляет собой сайт или раздел сайта, где пользователь (автор) регулярно публикует записи (посты), которые могут содержать текст, изображения, видео и другие мультимедийные элементы. Основное назначение блог-платформы — упростить процесс создания контента и предоставить инструменты для взаимодействия с аудиторией.

Блог-платформы бывают двух типов: самостоятельные (устанавливаемые на собственный сервер) и облачные (работающие на серверах компании-разработчика). Примеры популярных платформ включают WordPress, Blogger, Medium, Tumblr. WordPress, например, может быть как облачным сервисом (wordpress.com), так и устанавливаемой системой управления контентом (wordpress.org).

Пользователь регистрируется на платформе, получает доступ к интерфейсу редактирования, где может создавать статьи, оформлять их, вставлять изображения и видео, добавлять теги и категории. Современные платформы позволяют также настраивать внешний вид блога с помощью тем (шаблонов), расширять функциональность с помощью плагинов, вести комментарии, подключать аналитику, а также зарабатывать с помощью рекламы.

Главное преимущество блог-платформ заключается в том, что они не требуют от пользователя глубоких знаний в программировании или веб-разработке. Даже новичок может создать свой блог всего за несколько шагов. В то же время, более опытные пользователи могут получить доступ к расширенным функциям, таким как редактирование HTML-кода, подключение собственной базы данных или домена.

Блог-платформы играют важную роль в современном цифровом мире — они позволяют людям делиться мнениями, знаниями, творчеством, вести персональные дневники или профессиональные журналы. Для бизнеса блог может стать инструментом продвижения, улучшения SEO и привлечения клиентов.

## 1.2 Описание программных инструментов и средств разработки

1.2.1 Язык программирования «Python»

**Python** — это высокоуровневый, интерпретируемый язык программирования общего назначения, созданный Гвидо ван Россумом и впервые выпущенный в 1991 году. Он отличается простым, понятным синтаксисом, который делает код легко читаемым и удобным для изучения даже для начинающих программистов.

Одной из главных целей Python является повышение читаемости кода. Благодаря этому программы на Python обычно короче и понятнее, чем аналогичные на других языках. Например, блоки кода в Python отделяются отступами, а не фигурными скобками, что способствует лучшей структуре и чистоте кода.

Python используется в самых разных областях:

* **веб-разработка** (например, с помощью фреймворков Django, Flask),
* **анализ данных и машинное обучение** (библиотеки Pandas, NumPy, TensorFlow, Scikit-learn),
* **автоматизация и написание скриптов**,
* **разработка игр, приложений, десктопных программ**,
* **встраиваемые системы и научные расчёты**.

Python — кроссплатформенный язык, он работает на Windows, macOS и Linux. Он также активно используется в образовании, благодаря своей простоте и логике, близкой к естественному языку.

Сильной стороной Python является большое и активное сообщество разработчиков. Существует огромное количество открытых библиотек и инструментов, которые значительно упрощают и ускоряют процесс разработки. Кроме того, Python имеет официальную документацию на многих языках, включая русский.

Таким образом, **Python** — это мощный, универсальный и доступный язык программирования, который подходит как для простых задач, так и для разработки сложных систем, и широко применяется в самых разных сферах IT.

### 1.2.2 Фреймворк «Django»

**Django** — это бесплатный и открытый фреймворк (каркас) для разработки веб-приложений на языке программирования **Python**. Он был создан для того, чтобы ускорить и упростить процесс создания сложных и безопасных сайтов и веб-сервисов. Первая версия Django была выпущена в 2005 году, и с тех пор он стал одним из самых популярных веб-фреймворков в мире.

Основная цель Django — позволить разработчику быстро создать полноценный сайт или веб-приложение с чистой архитектурой и минимальными затратами времени. Слоган Django — **“The web framework for perfectionists with deadlines”**, что в переводе означает: «Веб-фреймворк для перфекционистов с жёсткими сроками».

Фреймворк Django следует архитектуре **Model–View–Template (MVT).** Это означает, что:

* **Model (Модель)** — отвечает за работу с базой данных,
* **View (Представление)** — обрабатывает запросы и возвращает ответы,
* **Template (Шаблон)** — отвечает за отображение данных пользователю (HTML-страницы).

Django содержит множество встроенных инструментов, которые позволяют:

* Создавать формы и обрабатывать пользовательский ввод,
* Управлять пользователями и авторизацией,
* Работать с базами данных без написания «чистого» SQL,
* Реализовывать панели администратора для управления сайтом,
* Защищать сайт от типичных угроз (SQL-инъекции, CSRF, XSS и др.).

Одним из главных преимуществ Django является **встроенная админ-панель**, которая автоматически создаётся на основе моделей и позволяет удобно управлять контентом сайта без необходимости писать дополнительный код.

Django также хорошо масштабируется — его можно использовать как для небольших проектов, так и для крупных веб-сервисов. Многие известные сайты и компании используют Django, например: **Instagram, Pinterest, Mozilla, NASA** и другие.

Таким образом, **Django** — это мощный инструмент для веб-разработки на Python, который позволяет быстро создавать безопасные, масштабируемые и удобные в сопровождении веб-приложения.

### 1.2.3 Серверная часть

Серверная часть интернет-магазина реализована с использованием Django — одного из самых популярных фреймворков для разработки веб-приложений на языке Python. Django предоставляет удобные инструменты для быстрого создания веб-сайтов, работы с базой данных, а также для обеспечения безопасности и производительности.

Используемые технологии

Django — это высокоуровневый веб-фреймворк на языке Python, который упрощает создание веб-приложений за счет множества встроенных функций, таких как управление базой данных, обработка форм, аутентификация пользователей и защита от распространенных угроз. В дополнение к этому, Django также предоставляет систему шаблонов, которая позволяет динамически генерировать HTML-страницы на сервере и отправлять их пользователю.

Django ORM (Object-Relational Mapping) используется для взаимодействия с реляционной базой данных (в нашем случае MySQL). ORM позволяет работать с базой данных, не пишущими напрямую SQL-запросы, а используя Python-объекты и их методы.

Основные компоненты серверной части

1. Маршруты и обработка запросов

Серверная часть приложения обрабатывает различные типы запросов от клиента, такие как получение списка товаров, создание заказов, а также добавление и редактирование постов. В Django обработка запросов осуществляется через views (представления), которые связаны с конкретными маршрутами, определенными в urls.py.

1. Работа с базой данных

Для работы с базой данных используется Django ORM, который позволяет определить модели данных, такие как товары, заказы, посты, и затем легко взаимодействовать с ними. Модели — это классы, которые наследуют django.db.models.Model, и каждая модель представляет собой таблицу в базе данных.

1. Аутентификация и авторизация

Django предоставляет встроенные инструменты для аутентификации и авторизации пользователей. Для защиты доступа к определенным страницам, таким как создание и редактирование постов, используются встроенные декораторы, например, @login\_required, который требует, чтобы пользователь был авторизован.

1. Шаблоны и рендеринг страниц

Django Templates используется для генерации HTML-страниц на сервере. Когда сервер получает запрос от пользователя, соответствующее представление обрабатывает запрос, извлекает необходимые данные из базы данных и передает их в шаблон для рендеринга страницы.

### 1.2.4 Среда разработки

Среда разработки включает в себя набор инструментов, которые позволяют эффективно разрабатывать, тестировать и отлаживать проект. Для работы с Django-проектом и его зависимостями используются различные инструменты.

Visual Studio Code (VS Code) — это популярный текстовый редактор с открытым исходным кодом, который поддерживает множество языков программирования, включая Python и Django. Он имеет встроенную поддержку Git, отладку кода и управление виртуальными окружениями. VS Code поддерживает различные расширения, такие как Python, Django и Pylance, что позволяет повысить эффективность работы. Этот инструмент используется для написания и редактирования Python-кода, шаблонов Django, стилей CSS и других файлов проекта.

Django Debug Toolbar — инструмент для отладки, который добавляется в проект Django и предоставляет подробную информацию о выполнении запросов и времени рендеринга страницы. Он показывает показатели запросов к базе данных, время выполнения, количество запросов и другие метрики, которые помогают в диагностике и отладке. Это удобный инструмент для диагностики, который отображает полезные данные на каждой веб-странице при разработке.

MySQL Workbench — графический инструмент для работы с базой данных MySQL. Он позволяет управлять базами данных, выполнять SQL-запросы, анализировать структуру таблиц и выполнять миграции. MySQL Workbench предлагает удобный интерфейс для работы с таблицами, создание схем, выполнение запросов, а также управление пользователями и их правами. Он используется для работы с MySQL базой данных проекта, а также для выполнения миграций и тестирования запросов.

# 2. Практическая часть

## 2.1 Проектирование веб-приложения

Проектирование веб-приложения является одним из ключевых этапов разработки любого современного информационного ресурса, обеспечивающего взаимодействие между пользователями, обработку данных и визуализацию контента. На данном этапе закладываются архитектурные принципы, логика взаимодействия компонентов, структура хранения данных и особенности интерфейса пользователя. В рамках данной работы разрабатывалось веб-приложение «Блог-платформа» на основе фреймворка Django, представляющее собой систему публикации и управления пользовательскими статьями с возможностью регистрации, аутентификации и комментирования.

Архитектура приложения

Выбор архитектурного подхода основан на принципах Model-View-Template (MVT), который является адаптацией классического паттерна MVC. Этот подход реализован в Django и обеспечивает чёткое разделение:

* Модель (Model) — отвечает за взаимодействие с базой данных и бизнес-логику;
* Представление (View) — определяет, какие данные отображаются пользователю;
* Шаблон (Template) — определяет внешний вид страницы, представленной пользователю.

Такой подход способствует масштабируемости и сопровождаемости проекта, поскольку изменение одного уровня минимально влияет на остальные.

Функциональные требования

Веб-приложение «Блог-платформа» реализует следующие ключевые функции:

* Регистрация и аутентификация пользователей;
* Создание, редактирование и удаление публикаций (постов);
* Система комментариев к публикациям;
* Использование медиафайлов (например, изображений в постах);
* Навигация и просмотр содержимого блога с использованием шаблонов.

Дополнительно обеспечена интеграция с системой управления пользователями через встроенную административную панель Django, что облегчает поддержку и развитие проекта.

Логическая структура проекта

В структуре проекта можно выделить следующие основные компоненты:

* Приложение accounts отвечает за регистрацию, вход и выход пользователей, а также управление профилем;
* Приложение blog реализует основную функциональность: модели публикаций и комментариев, соответствующие представления и шаблоны;
* Папка templates**/** содержит HTML-шаблоны, используемые для формирования страниц;
* Папка static**/** — для хранения CSS, JavaScript и других статических ресурсов;
* Папка media**/** — для хранения пользовательских изображений и других загружаемых файлов.

Между приложениями реализованы связи через маршруты (URLconf), а также общие механизмы авторизации и передачи данных.

База данных и модели

Проект использует встроенный ORM Django, что позволяет определять модели на языке Python и автоматически выполнять миграции в базу данных. Основными сущностями являются:

* User — встроенная модель пользователя;
* Post — модель публикации с полями заголовка, содержания, даты создания и автора;
* Comment — модель комментария, связанная с публикацией и пользователем.

Структура данных оптимизирована для эффективного поиска, фильтрации и отображения информации. Использование отношений один-ко-многим (например, один пользователь — много постов) обеспечивает гибкость и расширяемость проекта.

Процесс проектирования

Процесс проектирования включал следующие этапы:

1. Анализ требований — определение ключевых пользовательских сценариев;
2. Разработка схемы данных — описание моделей и связей между ними;
3. Создание интерфейсных прототипов — определение визуальной структуры страниц;
4. Настройка маршрутов и представлений — реализация логики отображения и обработки данных;
5. Интеграция пользовательского интерфейса и шаблонов — реализация HTML-шаблонов и их связка с представлениями.

В ходе проектирования особое внимание уделялось вопросам безопасности, удобства навигации и масштабируемости.

## 2.2 Подготовка к работе

Перед тем как приступить к написанию кода нашей блог-платформы на Django, необходимо подготовить и настроить среду разработки, которая станет основой для стабильной и удобной работы над проектом.

Первым шагом является установка языка программирования Python, на котором базируется Django. Важно использовать современную версию Python 3.10 или выше, чтобы воспользоваться всеми актуальными возможностями и получать поддержку со стороны фреймворка. После установки Python автоматически устанавливается менеджер пакетов pip, с помощью которого будет производиться установка всех необходимых библиотек.

Далее создаётся виртуальное окружение — изолированное пространство, в котором будут устанавливаться зависимости проекта. Это позволяет избежать конфликтов с другими Python-проектами на компьютере и облегчает управление версиями библиотек. В терминале создаётся виртуальная среда и активируется, после чего можно быть уверенным, что все пакеты будут устанавливаться локально именно для этого проекта.

Следующим шагом является установка самого фреймворка Django. Это делается при помощи команды pip install django, которая скачивает и устанавливает последнюю стабильную версию. Помимо Django, в проект могут входить и другие вспомогательные библиотеки, например для работы с загрузкой файлов или расширения функционала, которые также прописаны в специальном файле requirements.txt. Для удобства все зависимости можно установить командой pip install -r requirements.txt, что гарантирует соответствие версий.

После установки основных инструментов производится создание и инициализация самого проекта Django. Эта команда формирует структуру папок и базовые файлы, без которых невозможно дальнейшее развитие. Затем создаются отдельные приложения, которые отвечают за ключевые части системы: авторизацию пользователей и работу с блог-постами. Такое разделение помогает поддерживать порядок и облегчает масштабирование.

Также необходимо позаботиться о базе данных. В процессе разработки используется MySQL. Однако, если планируется работа с более серьёзными данными, рекомендуется подготовить и настроить PostgreSQL или другую СУБД. На этом этапе создаются таблицы и связи между ними при помощи миграций — автоматизированного механизма обновления структуры базы данных, который интегрирован в Django.

Чтобы обеспечить удобство работы с интерфейсом, создаются каталоги для шаблонов HTML и статических файлов — стилей, скриптов, изображений. В настройках проекта прописываются пути к этим каталогам, чтобы Django мог корректно обрабатывать и отдавать необходимые ресурсы при запросах от пользователей.

Наконец, после всех этапов подготовки запускается локальный сервер разработки командой python manage.py runserver. Это позволяет увидеть приложение в действии в и начать тестирование базовых функций, а также вносить корректировки в режиме реального времени.

Таким образом, подготовительный этап — это комплекс действий, направленных на создание надёжной и удобной основы для дальнейшей разработки. От тщательности и правильности подготовки зависит скорость реализации, качество кода и удобство поддержки веб-приложения в будущем.

## 2.3 Разработка веб-приложения

Разработка современного веб-приложения — это комплексный, многоступенчатый процесс, включающий в себя как проектирование архитектуры, так и реализацию конкретных функциональных модулей. В основе нашего проекта — блог-платформа, построенная на популярном и проверенном временем фреймворке Django. Благодаря его возможностям, процесс создания приложения становится более структурированным, а конечный продукт — надёжным и масштабируемым. В этом разделе мы подробно рассмотрим все ключевые этапы разработки: от проектирования базы данных и настройки маршрутов до создания шаблонов и стилизации интерфейса, а также работы с медиафайлами и административной панелью.

2.3.1 Создание моделей и структуры базы данных

Проектирование моделей — это фундаментальный этап, на котором закладывается логика хранения и взаимодействия данных. В Django модели представляют собой Python-классы, которые отражают таблицы базы данных и содержат поля с типами данных, необходимыми для бизнес-логики. К тому же Django автоматически генерирует SQL-команды для создания таблиц и управления ими, что облегчает работу разработчика.

Для блог-платформы мы определили две основные модели — Post и Comment. Модель Post представляет публикацию в блоге и содержит следующие важные поля: заголовок, основной текст, связь с автором, дату создания и обновления, флаг публикации, а также поле для изображения. Такая структура обеспечивает хранение всей необходимой информации для отображения и управления постами (Рис. 1).

Важно, что поле изображения — необязательное, оно позволяет расширять функционал, добавляя визуальный контент, который делает блог более привлекательным для пользователей и разнообразит подачу информации.

Модель Comment описывает комментарии пользователей, привязанные к конкретным публикациям. Она содержит ссылку на пост, автора комментария, текст сообщения и дату создания. Связи между моделями реализуются с помощью полей ForeignKey, что обеспечивает целостность данных и упрощает навигацию внутри системы.

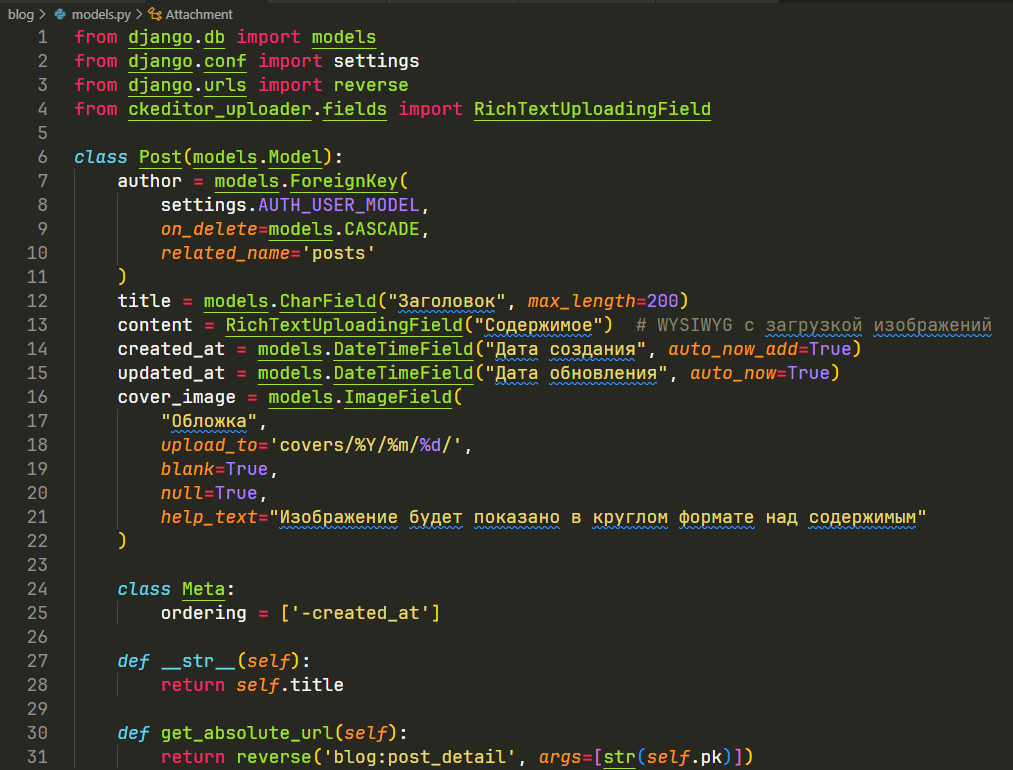


Рисунок 1. Пример ключевых полей модели Post.

Это описание отражает основные характеристики публикации, позволяя гибко управлять контентом и отображать его в нужном виде.

После завершения проектирования моделей в Django используется механизм миграций. Он позволяет автоматически создавать или обновлять таблицы в базе данных согласно описанию моделей, избавляя разработчиков от необходимости писать SQL вручную. Для этого применяются команды:

* python manage.py makemigrations
* python manage.py migrate

Этот процесс является стандартной практикой в Django-разработке и обеспечивает надёжность и удобство поддержки базы данных при эволюции проекта.

2.3.2 Настройка маршрутов (URLconf)

Для того чтобы пользователи могли взаимодействовать с различными страницами приложения, необходимо настроить систему маршрутизации. В Django она реализована через конфигурационные файлы URLconf, где каждому URL адресу сопоставляется определённое представление, которое обрабатывает запрос и возвращает ответ.

В нашем проекте основными маршрутами являются (Рис. 2):

* Главная страница со списком опубликованных публикаций;
* Страница детального просмотра отдельного поста, доступная по уникальному идентификатору;
* Страница создания нового поста для авторизованных пользователей.



Рисунок 2. Настройка маршрутов в файле blog/urls.py.

Такое распределение URL адресов обеспечивает интуитивно понятную навигацию по сайту и упрощает расширение функциональности в будущем.

2.3.3 Разработка представлений (Views)

Представления — это программные компоненты, которые обрабатывают HTTP-запросы, извлекают необходимые данные и подготавливают их для отображения пользователю. В Django широко применяются классовые представления (Class-Based Views), которые обеспечивают простоту и повторное использование кода.

В нашем проекте реализованы (Рис. 3):

* PostListView — отображает список всех опубликованных постов, сортируя их по дате публикации и обеспечивая удобную пагинацию (постраничный вывод);
* PostDetailView — показывает полное содержимое конкретного поста, включая текст, автора и дату;
* PostCreateView — предоставляет форму для создания нового поста, автоматически связывая его с текущим авторизованным пользователем.

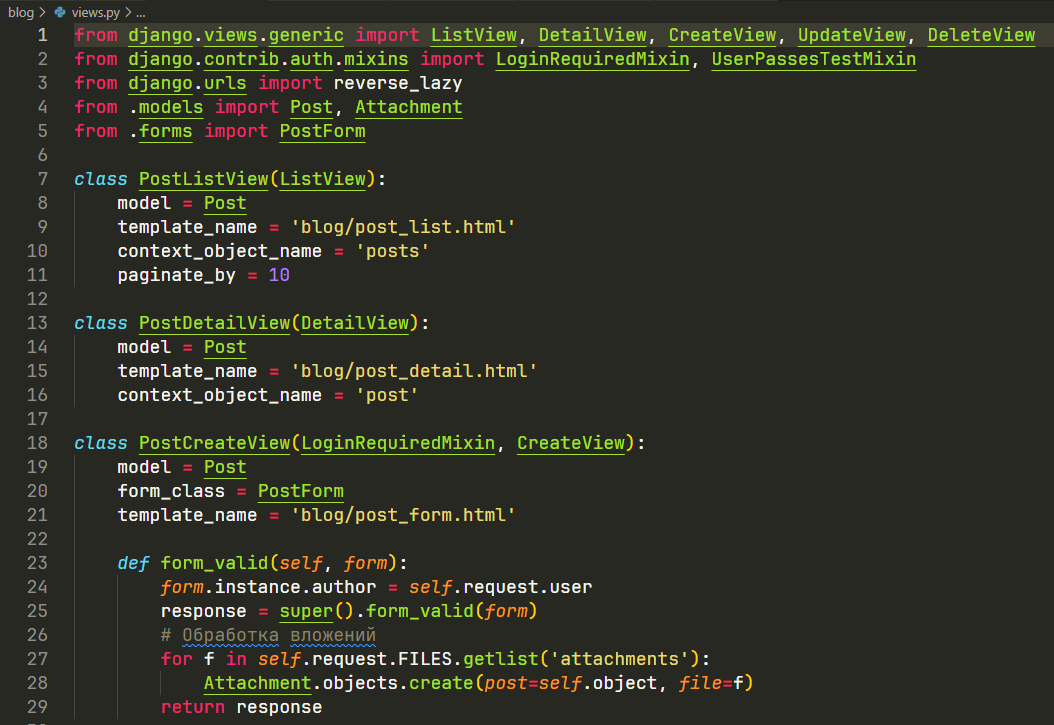


Рисунок 3. Основные представления.

Здесь фильтрация позволяет отображать только опубликованные записи, а пагинация делает интерфейс удобным (Рис.4).

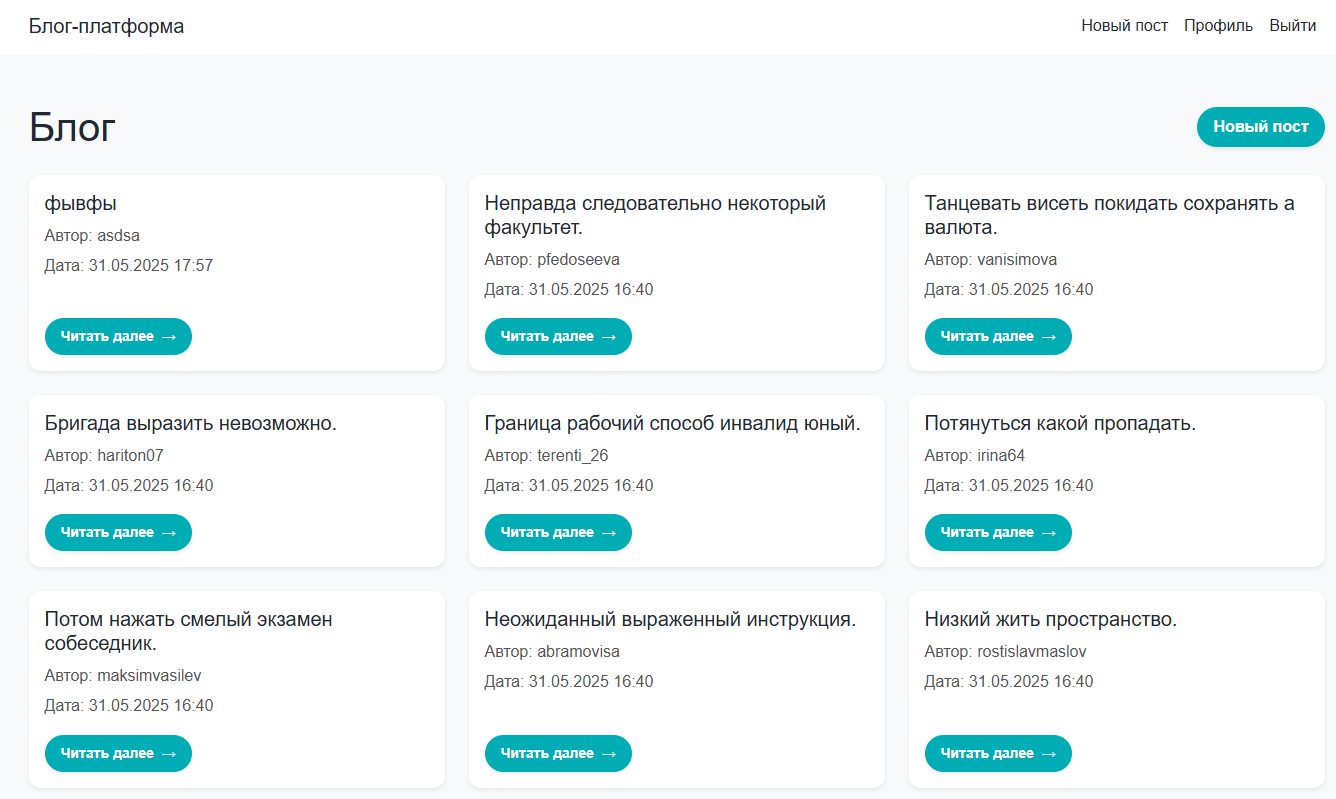
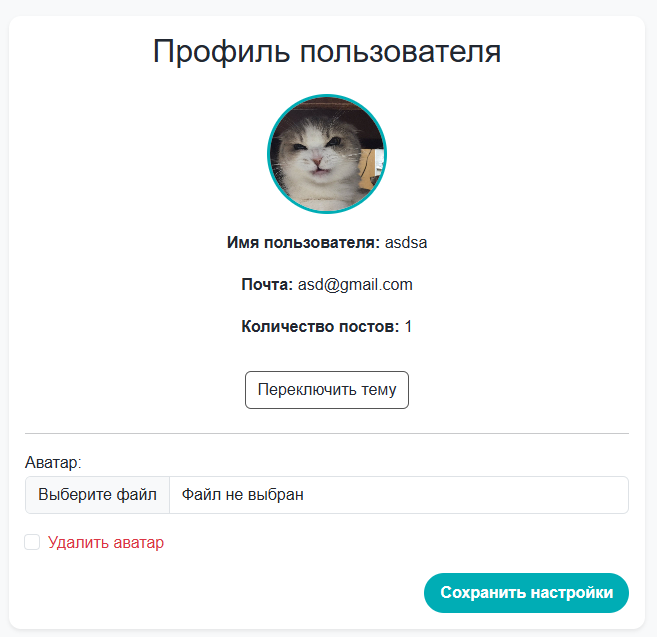


Рисунок 4. Главная страница с постами.

2.3.4 Создание шаблонов (Templates)

Шаблоны формируют визуальную часть приложения, определяя, как будет выглядеть каждая страница. В Django реализован удобный механизм наследования шаблонов, который позволяет определить базовую структуру сайта — шапку, меню, подвал — и расширять её на отдельных страницах, добавляя специфическое содержимое.

В шаблоне профиля выводится список с личной информацией, кнопка с настройкой темы и возможностью изменять фото (Рис.5).

  
Рисунок 5. Внешний вид страницы профиля публикаций с динамическими данными и элементами управления.

2.3.5 Работа с формами

Формы служат для получения данных от пользователей и обеспечения их корректности. Django облегчает эту задачу с помощью форм, автоматически генерируемых на базе моделей.

В нашем проекте форма создания поста содержит поля для заголовка, текста, изображения и статуса публикации (Рис.6). При отправке формы данные проходят валидацию, и если всё корректно, запись сохраняется в базе. При этом автор автоматически назначается текущим пользователем, что позволяет контролировать права и принадлежность контента.

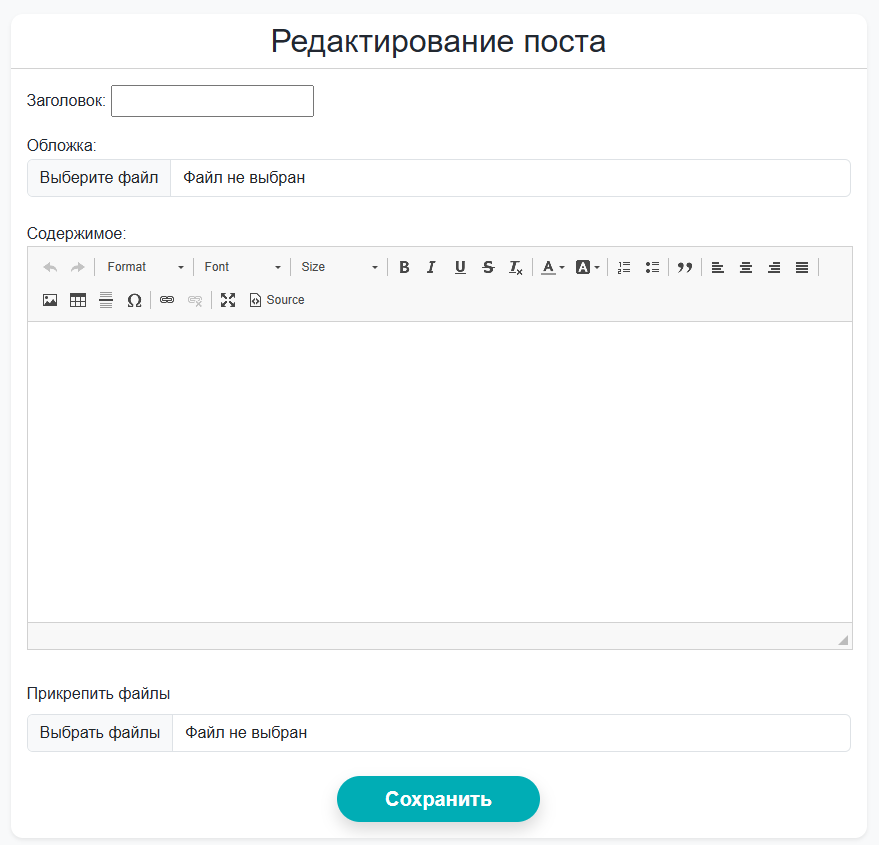
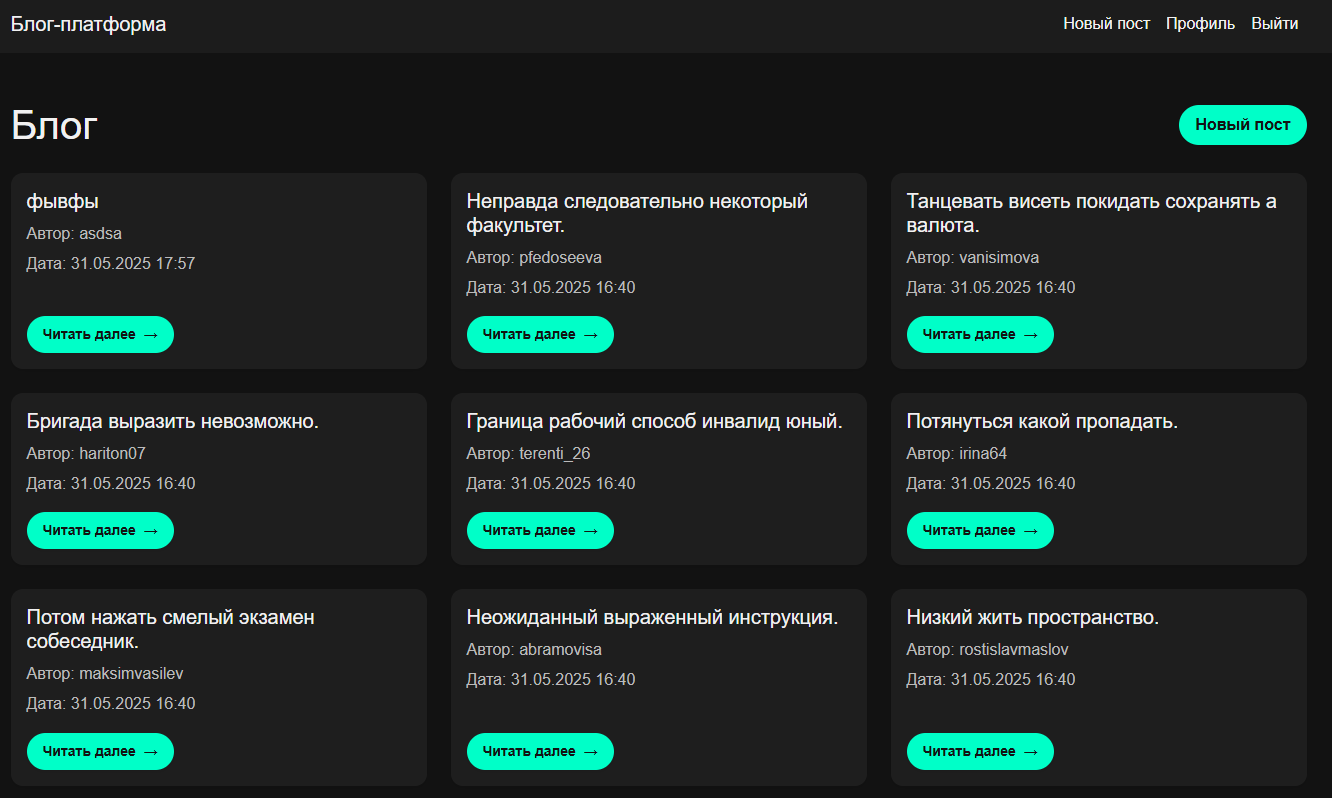


Рисунок 6. Веб-форма для создания новой публикации.

2.3.6 Верстка и стилизация

Для визуального оформления сайта применяются CSS-стили, которые подключаются через статические файлы (Рис. 7). В базовом шаблоне прописано подключение основного стиля, задающего внешний вид текста, ссылок, кнопок и прочих элементов.

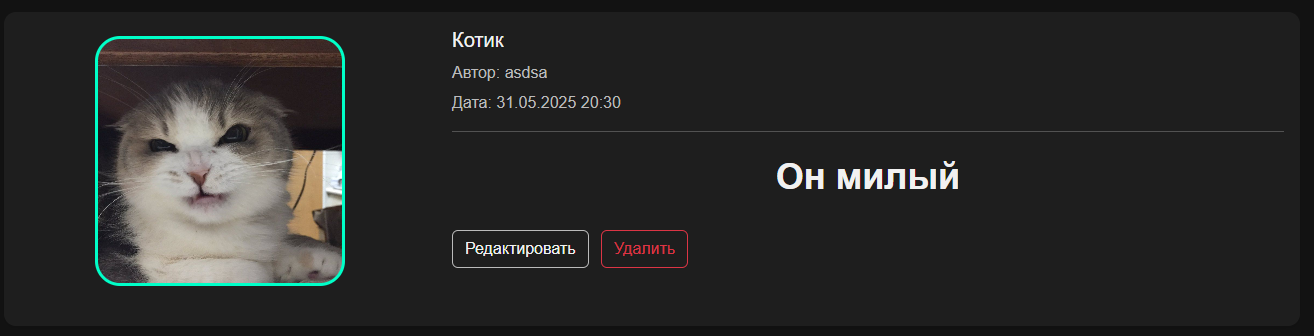
Стили написаны с учётом удобства восприятия информации и простоты навигации, что положительно влияет на общее впечатление пользователя и удержание аудитории.

  
Рисунок 7. Главная страница сайта с применёнными стилями.

2.3.7 Обработка медиафайлов

Загрузка и отображение изображений значительно расширяют возможности блога, делая контент более привлекательным и информативным. В настройках проекта определены пути для хранения и отдачи медиафайлов, что позволяет удобно работать с загруженными пользователями картинками.

В шаблоне страницы публикации реализована проверка наличия изображения и его вывод при наличии, что позволяет не отображать пустые места (Рис. 8).

  
Рисунок 8. Публикация с изображением и текстом.

Итог

Таким образом, процесс разработки блог-платформы на Django — это чётко организованный цикл действий, включающий проектирование моделей данных, настройку маршрутов, разработку представлений и шаблонов, оформление интерфейса и работу с медиафайлами. Использование мощных и гибких инструментов фреймворка позволяет создавать современное, удобное и функциональное веб-приложение, готовое к масштабированию и дальнейшему развитию.

## 2.4 Размещение веб-приложения на хостинге

Для размещения веб-приложения на общедоступном хостинге был выбран сервер с использованием Beget — одного из популярных провайдеров VPS. Приложение разработано с использованием фреймворка Django, базы данных MySQL, и веб-сервера Nginx. В этом разделе описываются шаги, которые были предприняты для развертывания проекта на сервере.

Первым этапом установки было подключение к серверу через SSH с использованием PuTTY, а также настройка Ubuntu VPS, на котором будет работать приложение. После подключения к серверу, было выполнено несколько подготовительных шагов: установка необходимых пакетов и зависимостей. В частности, была установлена последняя версия Python, MySQL, Nginx и Gunicorn — последнего для запуска самого приложения.

Для изоляции зависимостей проекта было создано виртуальное окружение с помощью virtualenv. В нем были установлены все зависимости, указанные в файле requirements.txt. Эти шаги гарантировали, что все пакеты, необходимые для работы приложения, были правильно установлены, не влияя на другие процессы, которые могли работать на сервере.

После установки всех компонентов, следующим этапом был перенос файлов проекта на сервер. Для этого был использован протокол SCP, через который все файлы были загружены в папку проекта на сервере. Путь к директории проекта на сервере был установлен как /home/blog-platform. Важно было проверить правильность структуры проекта, чтобы избежать ошибок при дальнейшем запуске.

Для подключения к базе данных на сервере был создан новый пользователь в MySQL с необходимыми правами для работы с базой данных myblog. База данных была импортирована с помощью дампа, после чего настройки подключения к базе данных были изменены в файле settings.py, указывая правильного пользователя и пароль, а также хост базы данных.

На следующем этапе была выполнена настройка веб-сервера Nginx для проксирования запросов на сервер приложений Gunicorn. Был создан конфигурационный файл для Nginx, который включает в себя настройку обратного прокси для передачи запросов на локальный адрес 127.0.0.1:8000, где работает Gunicorn. Также были настроены пути для обслуживания статических файлов и медиафайлов, что обеспечивало быструю загрузку контента, как изображений, так и стилей, с помощью Nginx.

После завершения настроек сервера было проведено тестирование приложения. Запустив сервер с помощью Gunicorn и настроив Nginx, проект стал доступен по адресу http://82.202.131.86/. При этом все запросы корректно обрабатывались, а статические и медиафайлы отображались без ошибок.

Следующим шагом был запуск проекта через Supervisor, что позволило автоматически управлять процессами Gunicorn, гарантируя их перезапуск в случае сбоя. С помощью Supervisor приложение было настроено на автоматический старт при загрузке сервера.

Таким образом, размещение веб-приложения на хостинге с использованием Beget включало в себя не только установку и настройку всех необходимых компонентов, но и настройку безопасности, правильное управление процессами, а также регулярный мониторинг работы сервера и приложения.

# Заключение

Разработка блог-платформы на базе фреймворка Django стала комплексным и многогранным процессом, включающим в себя не только реализацию базовой функциональности, но и продуманное архитектурное проектирование, что является залогом успешной работы и дальнейшего развития приложения. Благодаря использованию возможностей Django удалось эффективно организовать структуру данных, реализовать гибкую маршрутизацию и создать удобный для пользователя интерфейс, сочетающий динамическое содержимое с продуманной версткой.

Особое значение имело разделение проекта на отдельные модули и приложения, что позволило достичь высокой степени масштабируемости и удобства сопровождения. Такой подход облегчает интеграцию новых функций, будь то расширение возможностей комментариев, внедрение системы рейтингов, интеграция с социальными сетями или реализация мультиязычной поддержки.

Использование встроенных механизмов Django, таких как ORM, системы форм и административной панели, значительно ускорило процесс разработки и снизило вероятность ошибок, повышая качество конечного продукта. Также была реализована поддержка загрузки и отображения медиафайлов, что улучшило визуальную составляющую платформы и повысило вовлечённость пользователей.

С точки зрения перспектив, блог-платформа обладает потенциалом к дальнейшему развитию — начиная от улучшения пользовательского интерфейса, внедрения адаптивного дизайна для мобильных устройств, и заканчивая оптимизацией производительности и безопасности. Помимо этого, проект может быть интегрирован с внешними сервисами аналитики и маркетинга, что расширит возможности по привлечению и удержанию аудитории.

Таким образом, созданное веб-приложение является не только решением для публикации и обмена контентом, но и готовой платформой для расширенного взаимодействия с пользователями и развития сообщества. Опыт, полученный в ходе разработки, сформировал прочную базу знаний, необходимых для реализации более сложных и масштабных проектов в области веб-разработки.

# Список использованной литературы

1. Django documentation [Электронный ресурс]. — URL: https://docs.djangoproject.com/ (Дата обращения: 15.05.2025).
2. MySQL Documentation [Электронный ресурс]. — URL: https://dev.mysql.com/doc/ (Дата обращения: 16.05.2025).
3. Документация Visual Studio Code [Электронный ресурс].–URL: https://code.visualstudio.com/docs/ (Дата обращения: 10.05.2025.)
4. Real Python Tutorials[Электронный ресурс]. — URL: https://realpython.com/ (Дата обращения: 11.05.2025).
5. MDN Web Docs: HTML [Электронный ресурс]. – URL: https://developer.mozilla.org/ru/docs/Web/HTML (Дата обращения: 12.05.2025).
6. MDN Web Docs: CSS [Электронный ресурс]. – URL: https://developer.mozilla.org/ru/docs/Web/CSS (Дата обращения: 13.05.2025).
7. Bootstrap Documentation [Электронный ресурс].– URL: https://getbootstrap.com/docs/5.3/getting-started/introduction (Дата обращения: 17.05.2025).

# ПРИЛОЖЕНИЕ

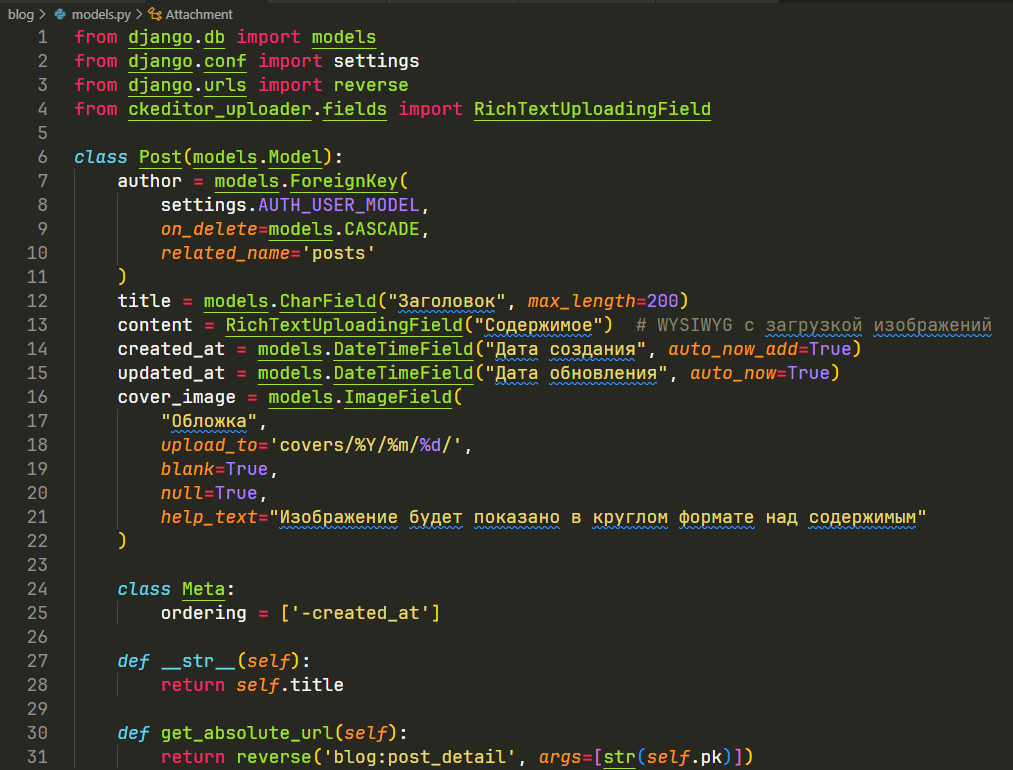


Рисунок 1. Пример ключевых полей модели Post.



Рисунок 2. Настройка маршрутов в файле blog/urls.py.

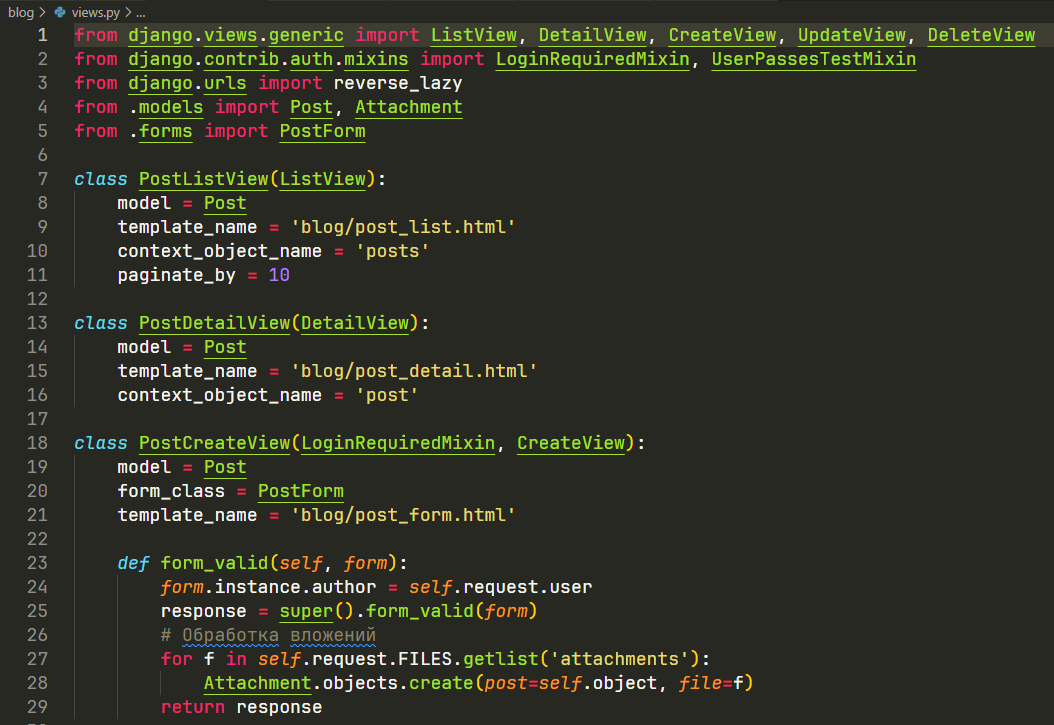


Рисунок 3. Основные представления.

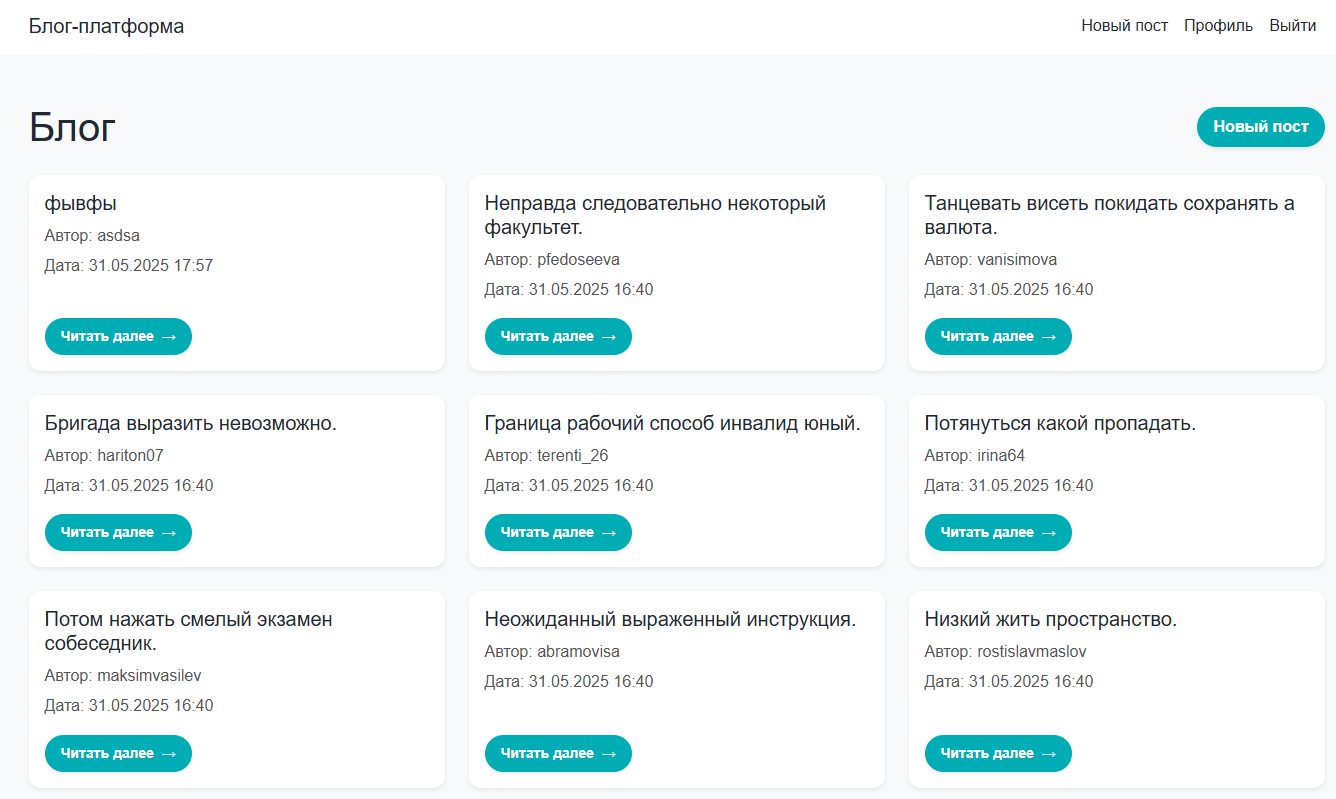
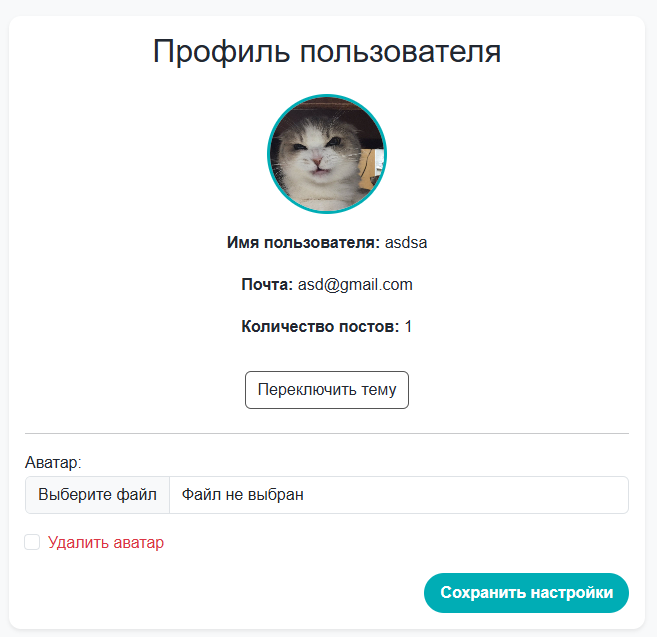


Рисунок 4. Главная страница с постами.

  
Рисунок 5. Внешний вид страницы профиля публикаций с динамическими данными и элементами управления.

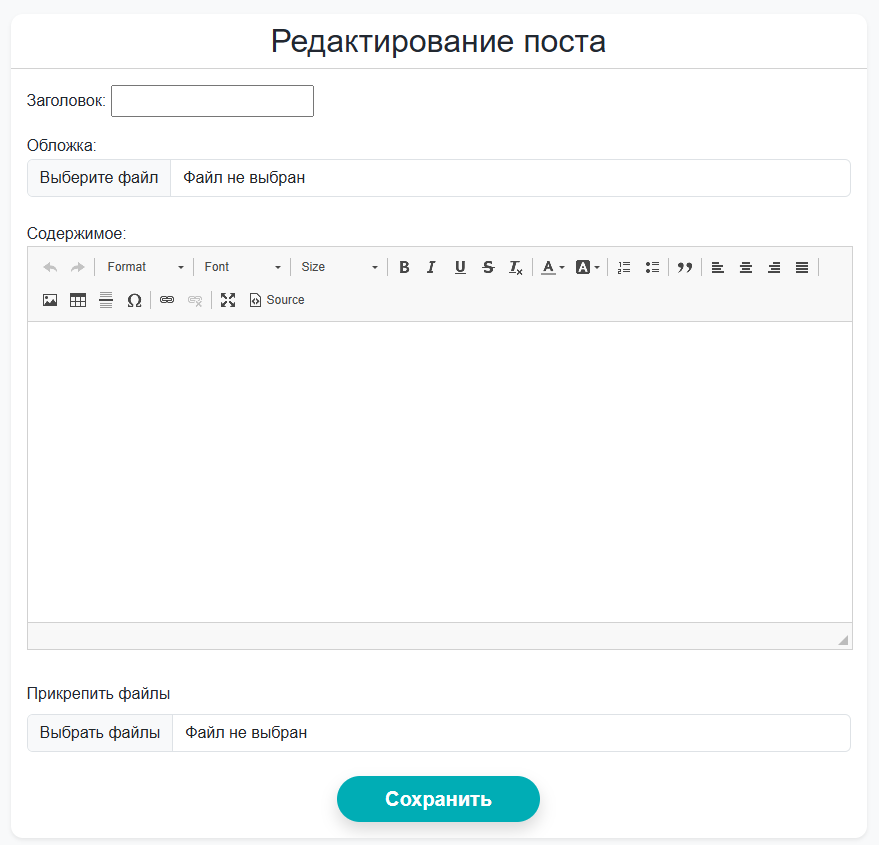
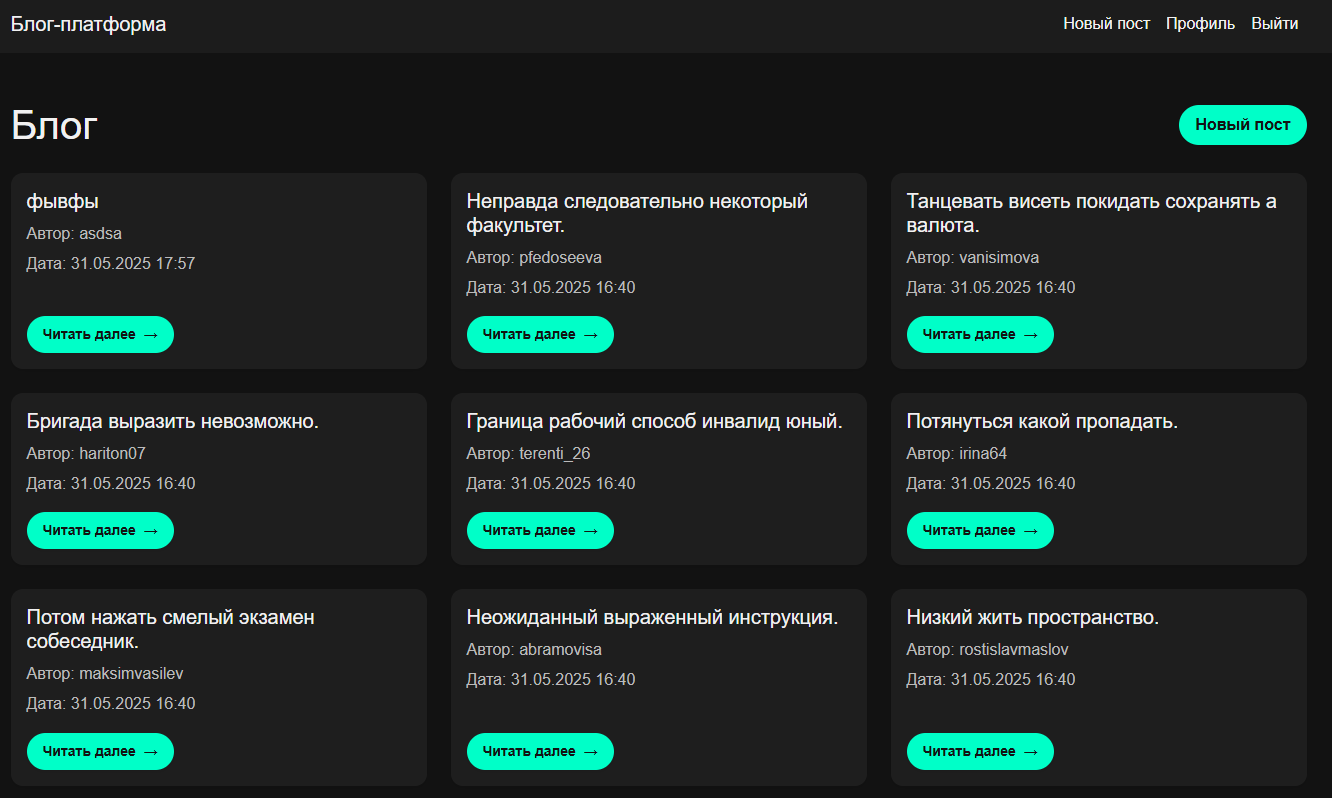
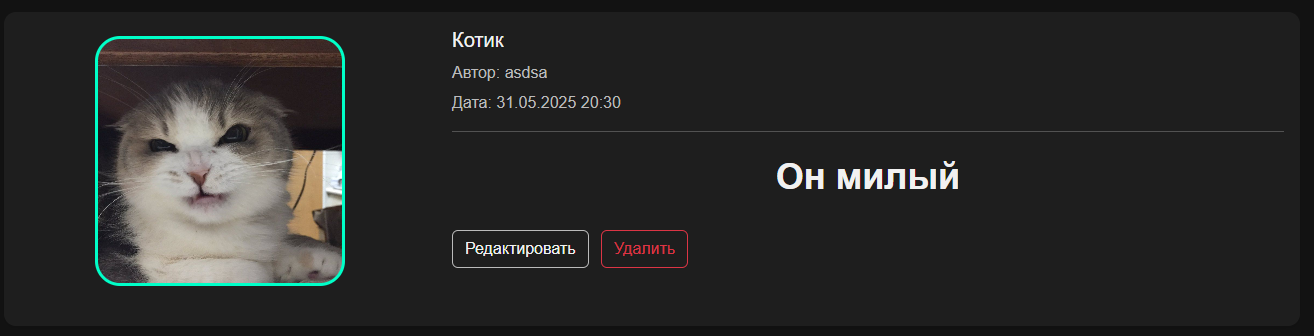


Рисунок 6. Веб-форма для создания новой публикации.

  
Рисунок 7. Главная страница сайта с применёнными стилями.

  
Рисунок 8. Публикация с изображением и текстом.