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 FLASK В РАЗРАБОТКЕ ВЕБ-ПРИЛОЖЕНИЯ «БЛОГ-ПЛАТФОРМА»

Разработчик:

студент группы 3ИСПР

Пронин М.Д.

(подпись)

Оценка защиты курсового проекта

Руководитель проекта:

преподаватель

Степанов С.О.

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

(подпись)

« »

Дата защиты

« » 2025г.

Ногинск, 2025 г

СОДЕРЖАНИЕ КУРСОВОГО ПРОЕКТА

[ВВЕДЕНИЕ 3](#_Toc199892973)

[1. Описание предметной области 5](#_Toc199892974)

[1.1 Краткая характеристика веб-приложения 5](#_Toc199892975)

[1.2 Описание программных инструментов и средств разработки 6](#_Toc199892976)

[1.2.1 Язык программирования «JavaScript» 6](#_Toc199892977)

[1.2.2 Фреймворк «FLASK» 7](#_Toc199892978)

[1.2.3 Серверная часть 8](#_Toc199892979)

[1.2.4 Среда разработки 9](#_Toc199892980)

[2. Практическая часть 11](#_Toc199892981)

[2.1 Проектирование веб-приложения 11](#_Toc199892982)

[2.2 Подготовка к работе 13](#_Toc199892983)

[2.3 Разработка веб-приложения 14](#_Toc199892984)

[2.3.1 Структура проекта 15](#_Toc199892985)

[2.3.2 Создание и настройка базы данных 15](#_Toc199892986)

[2.3.3 Разработка страниц и маршрутов 17](#_Toc199892987)

[2.3.4 Верстка страниц 18](#_Toc199892988)

[2.4 Размещение веб-приложения на хостинге 23](#_Toc199892989)

[Заключение 26](#_Toc199892990)

[Список использованной литературы 28](#_Toc199892991)

[Приложение 29](#_Toc199892992)

# ВВЕДЕНИЕ

В современном мире веб-приложения становятся неотъемлемой частью нашей повседневной жизни, охватывая различные сферы, от бизнеса до образования. Разработка таких приложений требует применения современных технологий и инструментов, обеспечивающих высокую производительность, безопасность и удобство использования. Одним из наиболее популярных фреймворков для создания веб-приложений на языке Python является Flask.

Он предоставляет разработчикам гибкость и простоту в реализации проектов различной сложности. В рамках данного курсового проекта будет рассмотрен процесс проектирования и реализации системы бронирования услуг.

**Целью данного курсового проекта** является разработка функционального веб-приложения для системы бронирования услуг. Приложение должно обеспечивать удобный интерфейс для пользователей и администраторов, а также включать все необходимые функции для эффективного управления процессом бронирования.

Для достижения поставленной цели необходимо решить ряд задач:

1. Анализ требований пользователей: На первом этапе будет проведен анализ потребностей целевой аудитории, что поможет определить ключевые функции приложения. Необходимо выявить, какие услуги планируется бронировать, как пользователи будут взаимодействовать с системой и какие дополнительные функции могут повысить удобство использования.

2. Разработка архитектуры приложения: На этом этапе будет создана архитектура веб-приложения, включающая структуру базы данных, модели данных и основные модули приложения. Важно учесть масштабируемость и возможность дальнейшего расширения функционала.

3. Реализация основных модулей: После проектирования архитектуры начнется реализация основных компонентов приложения: интерфейса пользователя, функционала для администраторов, системы аутентификации и авторизации пользователей, а также интеграции с базой данных.

4. Тестирование приложения: После завершения разработки необходимо провести тестирование приложения для выявления возможных ошибок и недочетов. Это позволит убедиться в корректной работе всех функций и стабильности системы в целом.

5. Подготовка документации: Завершающим этапом станет подготовка документации, включающей описание функционала приложения, инструкции по установке и использованию, а также технические детали реализации.

**Объект и предмет курсового проекта.** Объектом курсового проекта является процесс разработки веб-приложений с использованием фреймворка Flask. Flask позволяет разработчикам создавать легковесные и эффективные приложения с минимальными усилиями благодаря своей гибкости и простоте использования.

Он поддерживает множество расширений, которые могут быть использованы для добавления новых функций без необходимости переписывать существующий код.

**Предметом курсового проекта** является проектирование и реализация конкретного приложения системы бронирования услуг. Это приложение должно удовлетворять современным требованиям пользователей, обеспечивая интуитивно понятный интерфейс и высокую скорость работы. Актуальность данной темы обусловлена растущим спросом на онлайн-сервисы, позволяющие пользователям удобно бронировать различные услуги — от гостиниц до ресторанов и мероприятий. Эффективная система бронирования не только упрощает процесс для пользователей, но и значительно оптимизирует бизнес-процессы компаний, предлагающих эти услуги.

# Описание предметной области

## 1.1 Краткая характеристика веб-приложения

Веб-приложение – это приложение, к которому пользователи получают доступ через веб-браузер по сети (обычно через Интернет или локальную сеть). В отличие от классических десктопных приложений, которые устанавливаются и запускаются непосредственно на компьютере пользователя, веб-приложения исполняются на веб-сервере и взаимодействуют с пользователем через браузер.

**Ключевые характеристики веб-приложений:**

* **Доступ через браузер:** Пользователю не нужно устанавливать какое-либо программное обеспечение на свой компьютер. Достаточно иметь веб-браузер (например, Chrome, Firefox, Safari, Edge) и подключение к сети.
* **Централизованное хранение данных:** Данные веб-приложения обычно хранятся на сервере, что облегчает их управление, резервное копирование и синхронизацию между пользователями.
* **Независимость от операционной системы:** Веб-приложения работают на любой операционной системе (Windows, macOS, Linux, Android, iOS), если на ней есть веб-браузер.
* **Обновление на сервере:** Обновления веб-приложения выполняются на сервере, и пользователи сразу же получают доступ к новой версии без необходимости загружать и устанавливать обновления.
* **Клиент-серверная архитектура:** Веб-приложения работают по принципу “клиент-сервер”, где браузер является клиентом, а веб-сервер - сервером. Клиент отправляет запросы на сервер, а сервер обрабатывает эти запросы и отправляет ответы обратно клиенту.

Удобство веб приложений и их доступность делают их популярным выбором для разработчиков и пользователей.

**1.2 Описание программных инструментов и средств разработки**

### 1.2.1 Язык программирования «JavaScript»

**JavaScript (JS)** - это динамический, интерпретируемый язык программирования, который в первую очередь используется для создания интерактивных веб-страниц и веб-приложений. Это один из трех основных языков (вместе с HTML и CSS), на которых строится большинство современных веб-сайтов.

**Ключевые особенности JavaScript:**

* **Динамический:** Типы переменных определяются во время выполнения, а не во время компиляции. Это обеспечивает гибкость, но требует более тщательного тестирования.
* **Интерпретируемый:** Код JavaScript выполняется непосредственно браузером без предварительной компиляции.
* **Мультипарадигменный:** Поддерживает различные стили программирования, включая объектно-ориентированное, императивное и функциональное программирование.
* **Слабо типизированный:** Не требует явного объявления типов переменных.
* **Событийно-ориентированный:** Реагирует на действия пользователя (например, клики, наведение мыши) и другие события.

**Основное назначение JavaScript:**

* **Интерактивность:** Добавление динамических элементов и интерактивности на веб-страницы.
* **Управление DOM:** Манипуляция HTML-элементами (DOM - Document Object Model) для изменения внешнего вида и поведения веб-страницы..

### 1.2.2 Фреймворк «FLASK»

Фреймворк представляет собой структурированную платформу для разработки программного обеспечения, своего рода готовый каркас или шаблон, который предоставляет разработчикам необходимые компоненты, библиотеки и правила.

Это значительно упрощает и стандартизирует процесс создания приложений, избавляя от рутинного написания кода с нуля и позволяя сосредоточиться на уникальной логике и функциональности проекта. Фреймворк предоставляет готовые решения для часто используемых задач, задает определенную структуру организации кода, упрощает поддержку и совместную работу, абстрагируется от сложных деталей реализации и, в конечном итоге, существенно ускоряет процесс разработки.

**Flask** — это микрофреймворк для разработки веб-приложений на Python. Он отличается лёгкостью, гибкостью и минимализмом, предоставляя разработчикам свободу выбора компонентов и библиотек.

Преимущества использования Flask:

* **Простота и лёгкость в освоении:** Flask имеет небольшое ядро и понятный синтаксис, что делает его идеальным для начинающих веб-разработчиков и для быстрого создания прототипов.
* **Гибкость и расширяемость:** Flask не накладывает жестких ограничений и позволяет разработчикам выбирать библиотеки и инструменты, которые лучше всего подходят для их задач.
* **Контроль над приложением:** Flask предоставляет больше возможностей для контроля над архитектурой и компонентами веб-приложения по сравнению с более «полноценными» фреймворками.

### 1.2.3 Серверная часть

Node.js — это платформа с открытым исходным кодом для работы с языком JavaScript. Она позволяет писать серверный код для веб-приложений и динамических веб-страниц, а также программ командной строки.

Node.js нужна для решения следующих задач:

- Разработка серверных приложений. Среда позволяет разрабатывать веб-серверы, способные работать с миллионами одновременных соединений.

- Асинхронное программирование. Код выполняется асинхронно, благодаря чему ресурсы сервера расходуются грамотнее, а приложение может обрабатывать множество запросов одновременно.

- Работа с большими объёмами данных. За счёт того, что Node.js хорошо справляется с высокой нагрузкой, её часто применяют для создания приложений, где нужна обработка больших объёмов информации.

- Создание микросервисов. Архитектура среды позволяет разрабатывать небольшие изолированные сервисы, которые легко масштабировать и изменять.

- Разработка сетевых приложений. К примеру, Node.js подходит для разработки чатов, игровых серверов и RESTful API, в которых нужна высокая скорость обработки запросов.

- Интеграция со сторонними инструментами через API. С помощью Node.js настраивают взаимодействие сайта с базами данных, фреймворками и библиотеками.

MySQL — это реляционная система управления базами данных (СУБД), которая распространяется как свободное программное обеспечение. Она позволяет хранить, организовывать большие объёмы данных и манипулировать ими.

MySQL нужна для решения следующих задач:

- Хранение данных. Позволяет сохранять данные в структурированном виде и даёт возможность быстро получать к ним доступ.

- Управление данными. MySQL даёт возможности для управления данными, например, создание, изменение и удаление таблиц, индексов и отношений между ними.

- Обеспечение безопасности данных. У системы продвинутая система контроля прав для разных пользователей, есть шифрование и верификация.

- Возможность контролировать версии базы данных. Можно делать бэкапы, обновлять базу или откатывать назад.

Command Prompt — это текстовый интерфейс для взаимодействия с операционной системой, который позволяет запускать различные команды и выполнять задачи напрямую через текстовый интерфейс, минуя графический пользовательский интерфейс (GUI).

### 1.2.4 Среда разработки

Visual Studio Code — это бесплатный, кроссплатформенный редактор исходного кода, разработанный компанией Microsoft. Он поддерживает работу на операционных системах Windows, Linux и macOS, сочетая в себе легковесность, гибкость и широкие функциональные возможности, которые делают его одним из самых популярных инструментов среди разработчиков по всему миру.

Основные особенности и преимущества VS Code:

- Поддержка множества языков программирования

Включает в себя поддержку таких языков, как JavaScript, TypeScript, Python, Java, C++, PHP, HTML, CSS, Go, C#, и многих других. Благодаря расширениям функциональность можно легко расширить.

- Интеллектуальные подсказки и автодополнение (IntelliSense)

VS Code автоматически предлагает завершение кода, информацию о функциях, переменных и типах, что значительно ускоряет написание кода и уменьшает количество ошибок.

- Встроенный терминал

Позволяет запускать команды прямо из редактора, без переключения на внешний терминал. Это удобно для запуска скриптов, команд Git или установки пакетов.

- Интеграция с системой контроля версий Git и GitHub

- Отладка приложений

Редактор поддерживает встроенную отладку, включая установку точек останова, пошаговое выполнение кода, просмотр переменных и стеков вызовов. Поддерживается множество языков и фреймворков через расширения.

- Live Share — совместная работа в реальном времени

Разработчики могут делиться своим рабочим пространством с коллегами, редактировать и отлаживать код совместно, как будто работают за одним компьютером.

- Гибкая настройка интерфейса

- Работа с удалёнными серверами и контейнерами

С помощью расширений (например, Remote - SSH, Remote - Containers), можно разрабатывать прямо в удалённой среде или в Docker-контейнере, не устанавливая зависимости локально.

- Лёгкость и производительность

Несмотря на богатый функционал, VS Code остаётся быстрым, легко запускается и не перегружает систему.

# Практическая часть

## 2.1 Проектирование веб-приложения

Проектирование веб-приложений представляет собой многогранный процесс, который включает в себя планирование, создание, тестирование и развертывание системы. Веб-приложения предоставляют пользователю доступ к различным функциональным возможностям через интернет-страницы. В случае с нашим проектом — форумом, мы должны были создать приложение, которое бы обеспечивало обмен сообщениями, управление контентом и удобную работу с пользователями. Такой тип веб-приложений предполагает наличие интерфейса для общения, создания контента и взаимодействия через комментарии и профили. Важно, что каждый элемент должен быть спроектирован с учётом удобства использования и безопасности.

Разработка веб-приложения форума началась с тщательного анализа требований. Важно было понять, какие функциональные возможности должны быть в системе. Наши требования включали создание пользовательских профилей, возможность зарегистрироваться и авторизоваться на сайте, создавать посты и комментарии, а также просматривать информацию о других пользователях. Эти базовые функции стали основой для дальнейшего проектирования. На этом этапе также было принято решение использовать фреймворк Flask, так как он предоставляет гибкость и удобство для реализации серверной логики и взаимодействия с базой данных.

Проектирование архитектуры приложения основывалось на принципах MVC (Model-View-Controller), что позволило разделить различные части системы на независимые компоненты. Модели определяли структуру данных для пользователей, постов и комментариев, в то время как представления отвечали за отображение информации на веб-страницах. Контроллеры принимали запросы от пользователей и обрабатывали их с использованием логики приложения, взаимодействуя с базой данных. В качестве базы данных был выбран SQLite, так как это легковесная и простая в использовании СУБД, которая идеально подходит для небольших и средних проектов.

При проектировании пользовательского интерфейса было уделено внимание тому, чтобы пользователи могли легко и удобно взаимодействовать с приложением. Структура интерфейса была спроектирована таким образом, чтобы пользователь мог быстро переходить между страницами создания постов, комментариев и профиля. Для стилизации интерфейса использовался фреймворк Bootstrap, который обеспечил современный и адаптивный дизайн. Также было решено использовать JavaScript для обработки динамических элементов на страницах, таких как формы для ввода текста и комментариев.

В процессе реализации функционала особое внимание было уделено тому, чтобы интерфейс взаимодействовал с сервером корректно. Каждый запрос от пользователя (например, создание нового поста или добавление комментария) обрабатывался на сервере с помощью маршрутов Flask. Для хранения данных использовалась база данных SQLite, которая позволяла хранить информацию о пользователях, постах и комментариях. SQLAlchemy была использована для упрощения работы с базой данных и обеспечения гибкости при выполнении операций с данными.

После завершения разработки основного функционала начался этап тестирования, на котором проверялась стабильность работы приложения и его безопасность. Были проведены юнит-тесты, которые проверяли работу отдельных функций приложения, и интеграционные тесты, чтобы убедиться, что все компоненты системы взаимодействуют корректно. Также было протестировано отображение страниц на различных устройствах, чтобы гарантировать правильную работу интерфейса на мобильных телефонах и десктопах.

Завершающим этапом было развертывание веб-приложения на сервере. Для этого был использован Gunicorn как WSGI сервер и Nginx для проксирования запросов. Это обеспечило высокую доступность приложения и его устойчивость при высоких нагрузках. Настройка базы данных и регулярные бэкапы стали необходимыми для предотвращения потерь данных.

Процесс проектирования и разработки веб-приложения форума был направлен на создание удобного, безопасного и функционального инструмента для общения пользователей. Веб-приложение, реализованное на Flask, обеспечило пользователям возможность создавать контент, взаимодействовать с другими людьми через комментарии и профили, а также делиться своим мнением на обсуждаемых темах.

## 2.2 Подготовка к работе

Подготовка к разработке веб-приложения начинается с установки необходимых инструментов и фреймворков, которые обеспечат основу для работы. Для разработки форума на платформе Flask необходимо установить Python — основной язык программирования для серверной части приложения. Важно настроить виртуальное окружение, которое изолирует зависимости проекта, гарантируя, что установленные пакеты не будут конфликтовать с другими проектами на машине.

Далее, требуется установка ключевых библиотек и фреймворков. Для разработки веб-приложения используется Flask — легковесный фреймворк для создания серверной логики. Вместе с ним устанавливается Flask-SQLAlchemy, который предоставляет удобный инструмент для работы с базой данных через ORM. Для работы с формами, валидации данных и защиты от CSRF атак применяется Flask-WTF. Для реализации системы аутентификации пользователей устанавливается Flask-Login, что позволяет управлять сессиями и обеспечивать безопасность.

Кроме того, потребуется Werkzeug, который отвечает за обработку безопасных операций, таких как хэширование паролей. Для упрощения работы с HTML и шаблонами используется Jinja2, встроенный в Flask. Все эти инструменты обеспечат правильную работу веб-приложения и упрощают реализацию базовых функций, таких как регистрация пользователей, создание постов и комментариев.

На стадии предпроектной подготовки также важно сформировать структуру проекта. Она должна включать файл для серверной логики, папки для хранения статичных файлов и шаблонов, а также место для базы данных. Важно заранее спроектировать структуру базы данных, описав таблицы и их связи, чтобы в дальнейшем не было необходимости в кардинальных изменениях.

После того как все инструменты установлены и проект настроен, наступает этап разработки функционала, где создаются модели данных, маршруты для обработки запросов и интерфейс для взаимодействия пользователей с системой. Подготовка всех компонентов до начала разработки помогает избежать многих проблем и ускоряет процесс создания функционала веб-приложения.

## 2.3 Разработка веб-приложения

Разработка веб-приложения форума является многозадачным процессом, включающим создание серверной логики, проектирование базы данных, разработку интерфейса пользователя и интеграцию всех компонентов в единую систему. В данном разделе мы рассмотрим ключевые этапы этого процесса, начиная с проектирования структуры данных и заканчивая версткой страниц. Также будут приведены примеры кода, что поможет лучше понять реализацию тех или иных функций.

2.3.1 Структура проекта

Прежде чем приступать к кодированию, важно четко представить структуру проекта, которая будет удобной для дальнейшего развития. Структура проекта включает несколько ключевых компонентов: серверную логику, базу данных, статичные файлы и шаблоны.

Для удобства и четкости работы мы создаем отдельные каталоги для каждой из частей. В корневом каталоге будет находиться основной файл проекта — app.py. Это файл, где будет реализована вся серверная логика приложения. В директории instance**/** хранится база данных, которая будет использоваться для хранения данных о пользователях, постах и комментариях. Каталог static**/** предназначен для хранения всех статичных файлов, таких как изображения, стили (CSS) и скрипты (JavaScript). Наконец, папка templates**/** будет содержать все HTML-шаблоны, которые будут рендериться на сервере.

Эта структура имеет несколько преимуществ: она облегчает навигацию по проекту и делает код более организованным. Например, все шаблоны расположены в одном месте, а статичные файлы — в другом. Такая структура позволяет легко масштабировать проект в будущем, если потребуется добавить новые компоненты или страницы.

2.3.2 Создание и настройка базы данных

После того как структура проекта определена, следующим важным шагом является создание базы данных. Для форума нам нужно хранить информацию о пользователях, их постах и комментариях. Для этой цели используется библиотека SQLAlchemy, которая позволяет удобно работать с базой данных с помощью Python-объектов.

Вначале необходимо создать модели, которые будут представлять таблицы базы данных. Каждая модель в SQLAlchemy это классовое представление таблицы. Например, модель User будет описывать таблицу пользователей (Рис. 1), а модель Post — таблицу постов.

Для модели пользователя можно использовать следующий код:

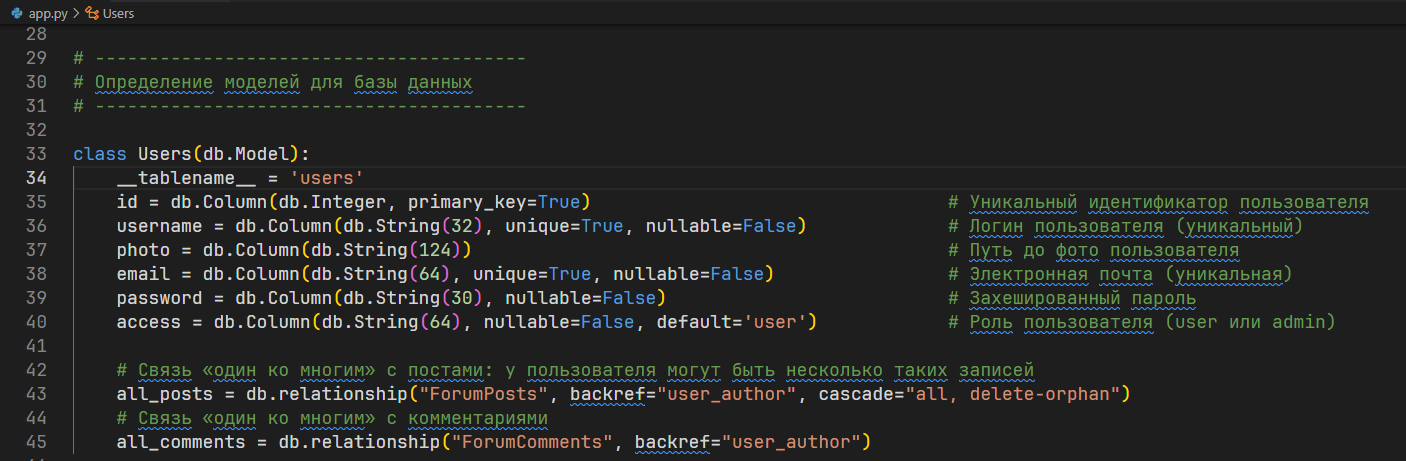


Рисунок 1. Модель User.

Эта модель описывает таблицу пользователей с такими полями, как id, username, email и password. Каждое поле модели будет соответствовать столбцу в таблице базы данных и её конфигурации (Рис. 2).

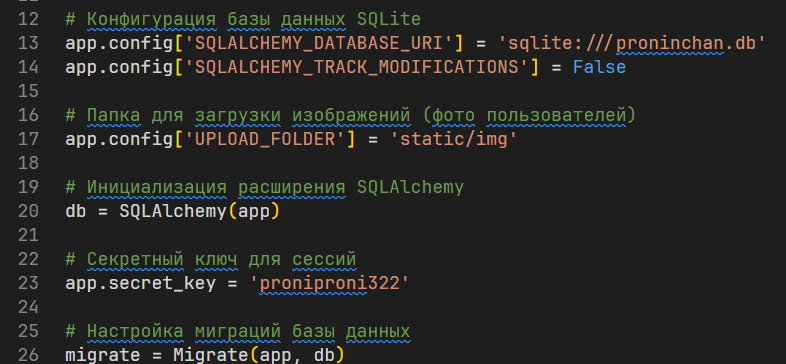


Рисунок 2. Конфигурация БД.

После этого в базе данных будет создана таблица users, и в дальнейшем можно будет выполнять операции, такие как добавление, удаление и обновление пользователей.

Создание других моделей для постов и комментариев происходит аналогично. Важно помнить, что между таблицами можно устанавливать связи. Например, каждый пост будет принадлежать конкретному пользователю. Эта связь устанавливается через использование внешних ключей и отношений в SQLAlchemy. Мы можем определить связь один ко многим, когда один пользователь может иметь несколько постов, а каждый пост связан с одним пользователем (Рис. 3).

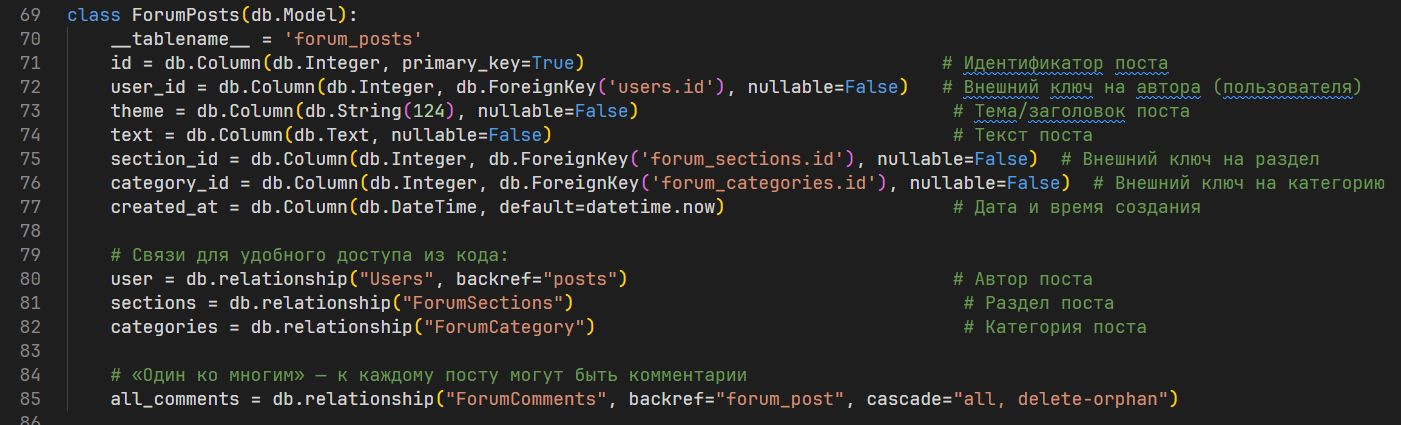


Рисунок 3. Настройка поста в БД.

Здесь мы указываем, что каждый пост связан с пользователем через поле user\_id, которое является внешним ключом, указывающим на id пользователя.

2.3.3 Разработка страниц и маршрутов

После того как база данных настроена, можно приступать к разработке страниц и маршрутов. В Flask маршруты определяются с помощью декораторов. Каждый маршрут отвечает за обработку определенного запроса от пользователя.

Главная страница форума должна показывать список всех доступных постов. Для этого мы создаем маршрут /, который будет извлекать все посты из базы данных и передавать их в шаблон для отображения (Рис. 4).

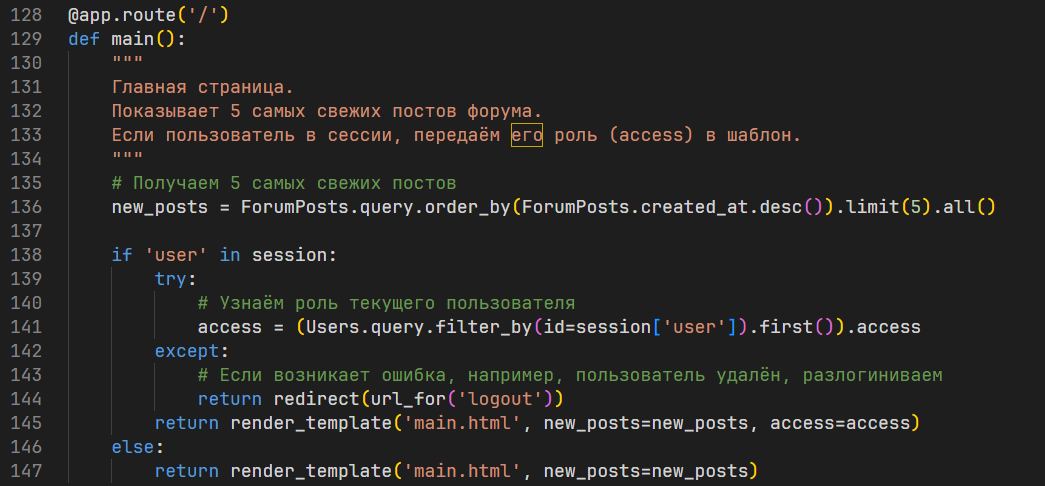


Рисунок 4. Маршрут главной страницы.

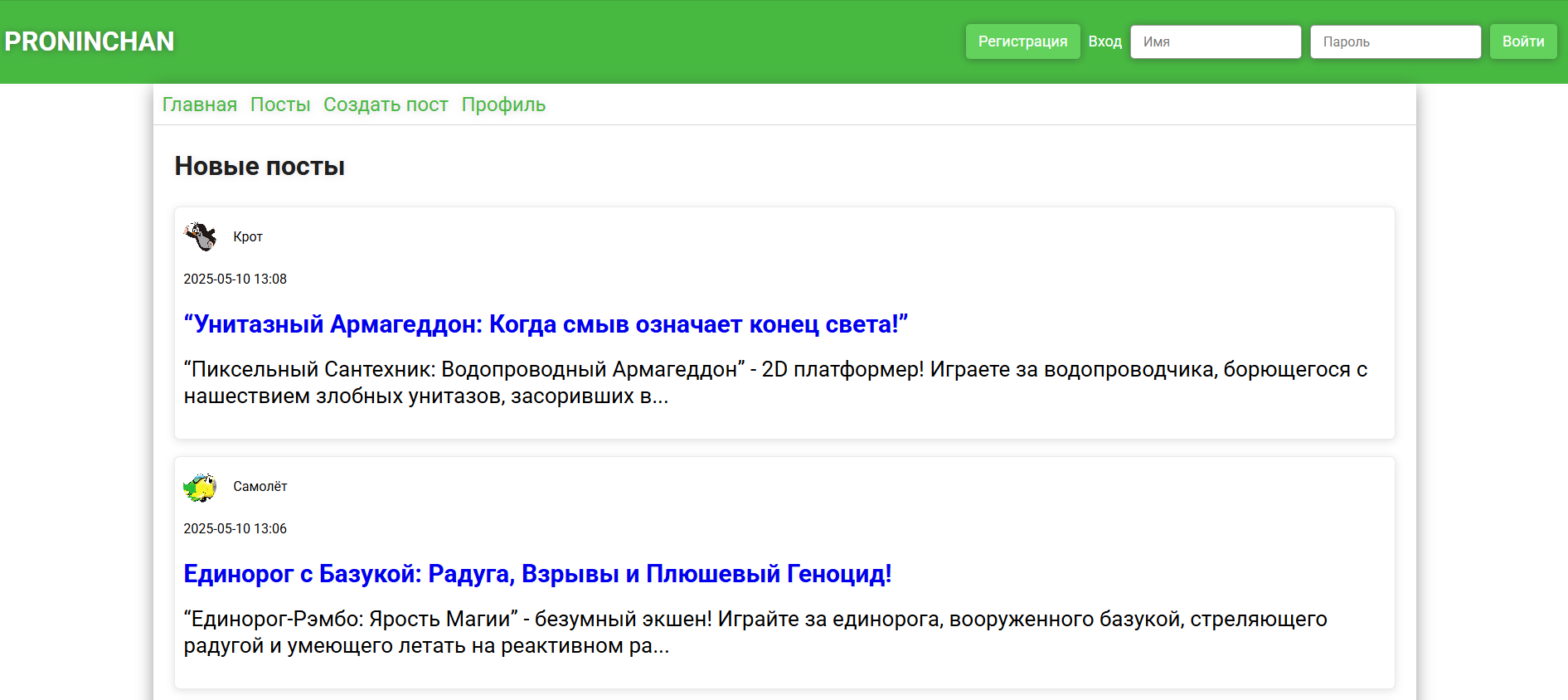
Здесь мы используем метод query.all() для получения всех постов из базы данных. Результат передается в шаблон posts.html, который рендерит их на странице.

Другим важным маршрутом является страница создания поста. Мы создаем маршрут /post/create, который будет обрабатывать как GET-запросы (показ формы для создания поста), так и POST-запросы (сохранение данных в базе данных). Этот маршрут создает новый пост в базе данных. В случае успешного добавления поста происходит перенаправление на главную страницу форума.

2.3.4 Верстка страниц

Верстка страниц веб-приложения форума играет ключевую роль в формировании первого впечатления у пользователя и в обеспечении удобства взаимодействия с функционалом. Каждый элемент интерфейса — от навигационной панели до расположения полей ввода комментариев — должен быть тщательно продуман и оформлен в едином стиле, чтобы пользователь сразу понимал, куда следует кликать и как совершать нужные действия. В нашем проекте основной упор был сделан на чистоту и простоту дизайна, чтобы даже при минимальном опыте работы с форумом посетитель мог быстро освоиться.

Для начала, на главной странице реализован макет с ярко выраженным заголовком и кратким описанием форума (Рис. 5). Под заголовком располагается область с последними публикациями: карточки постов выстроены в один или два столбца в зависимости от ширины экрана. Каждая карточка содержит заголовок, превью текста и дату публикации, а также при наведении подсвечивается рамкой, что обеспечивает дополнительную интерактивность и наглядность. Такой подход позволяет пользователю мгновенно увидеть, какие темы обсуждаются, и быстро перейти к интересующему материалу.

  
Рисунок 5. Главная страница.

Страница просмотра списка постов содержит большую область контента, плавно переходящую в список комментариев при выборе конкретного поста (Рис. 6). Основная часть текста выровнена по ширине, что улучшает читаемость, а поля с боковой информацией (например, количество комментариев или имя автора) аккуратно вынесены в правую сторону карточки поста.

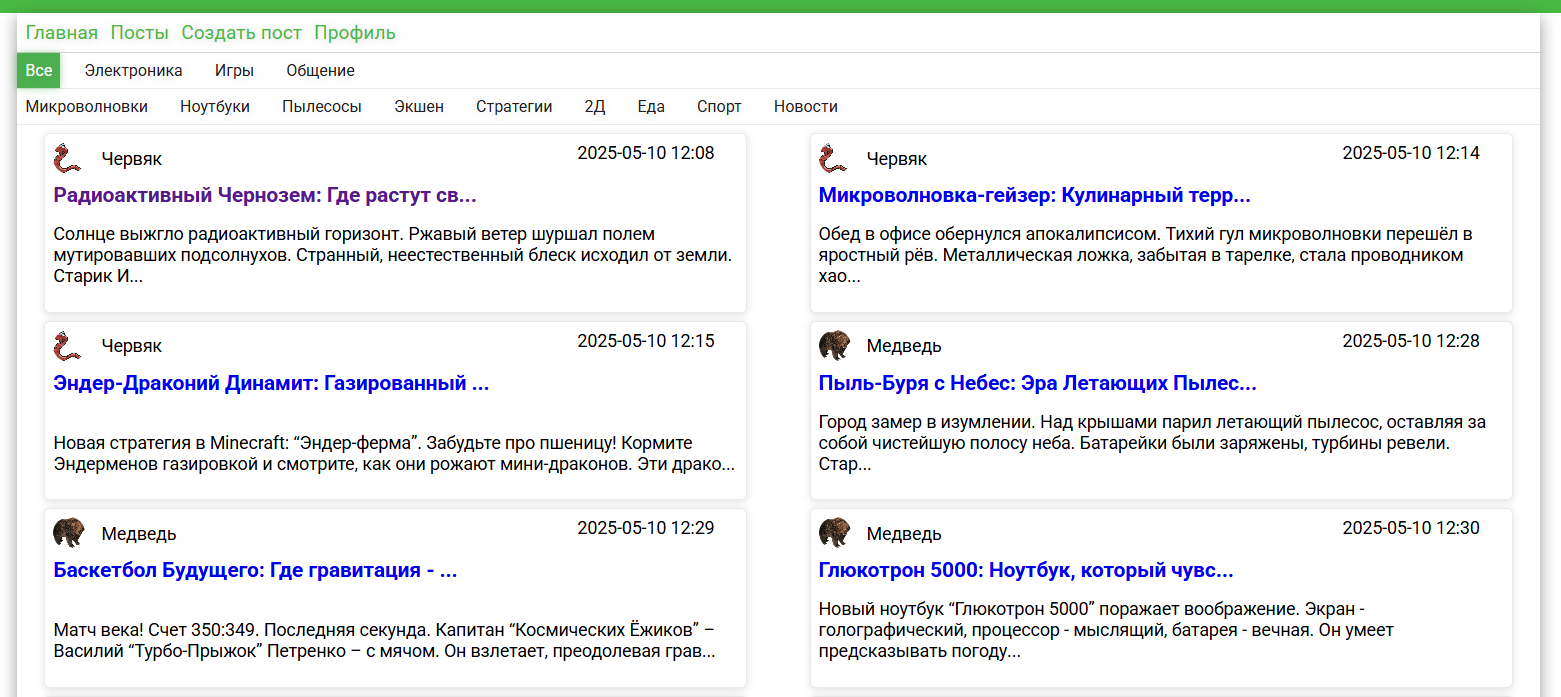
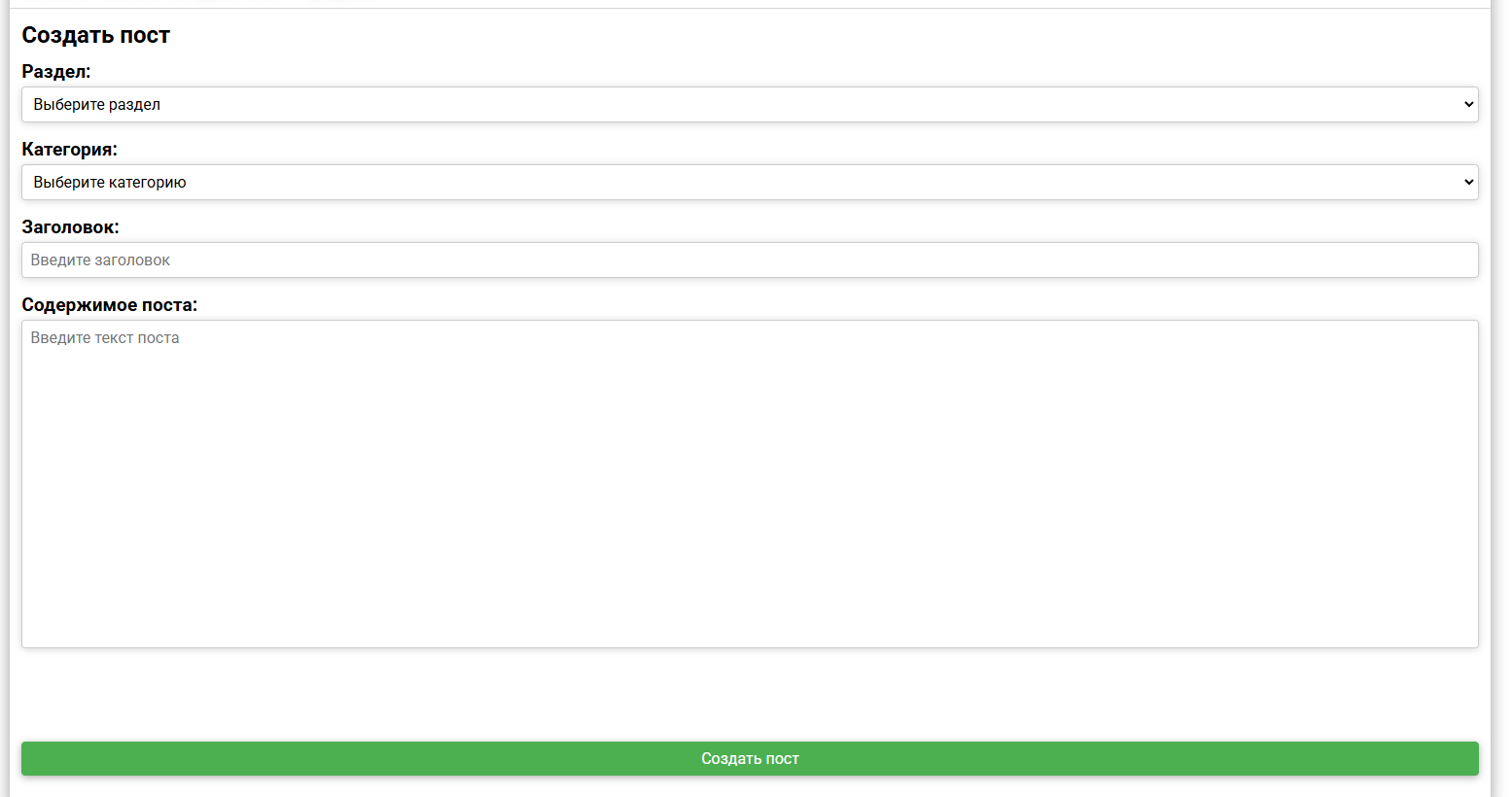
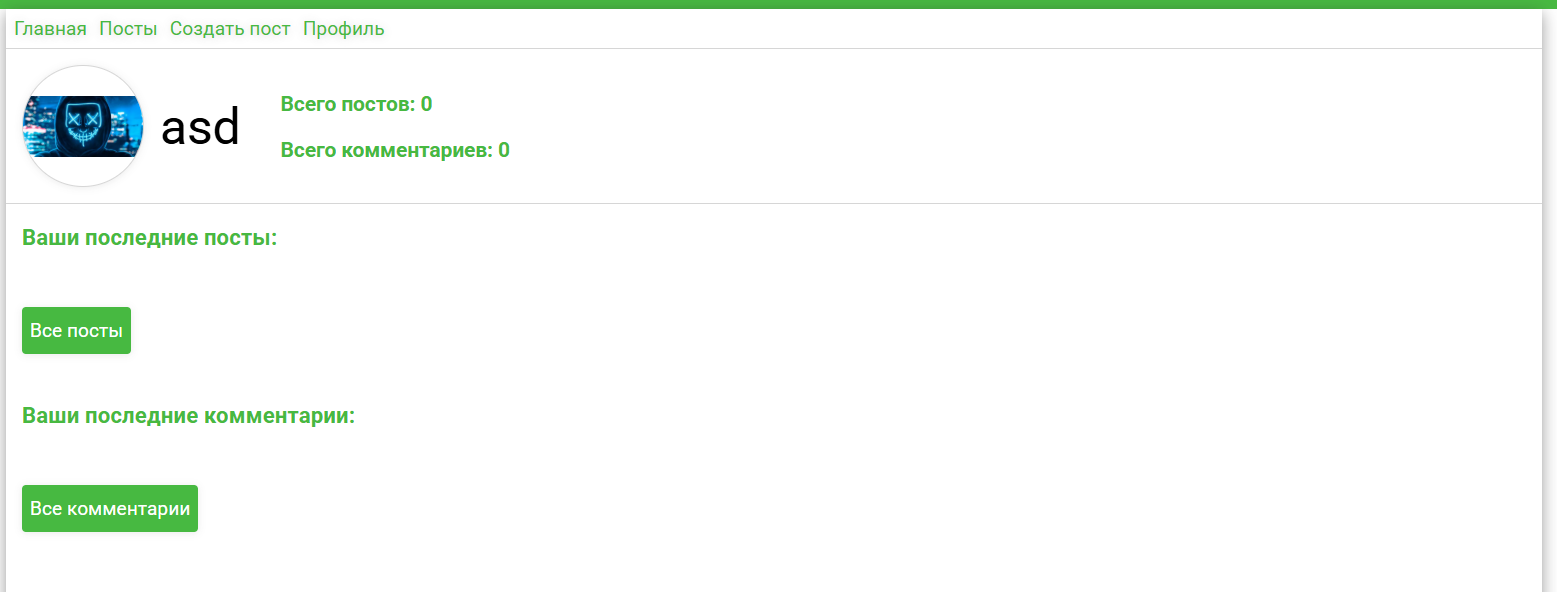


Рисунок 6. Страница списка постов.

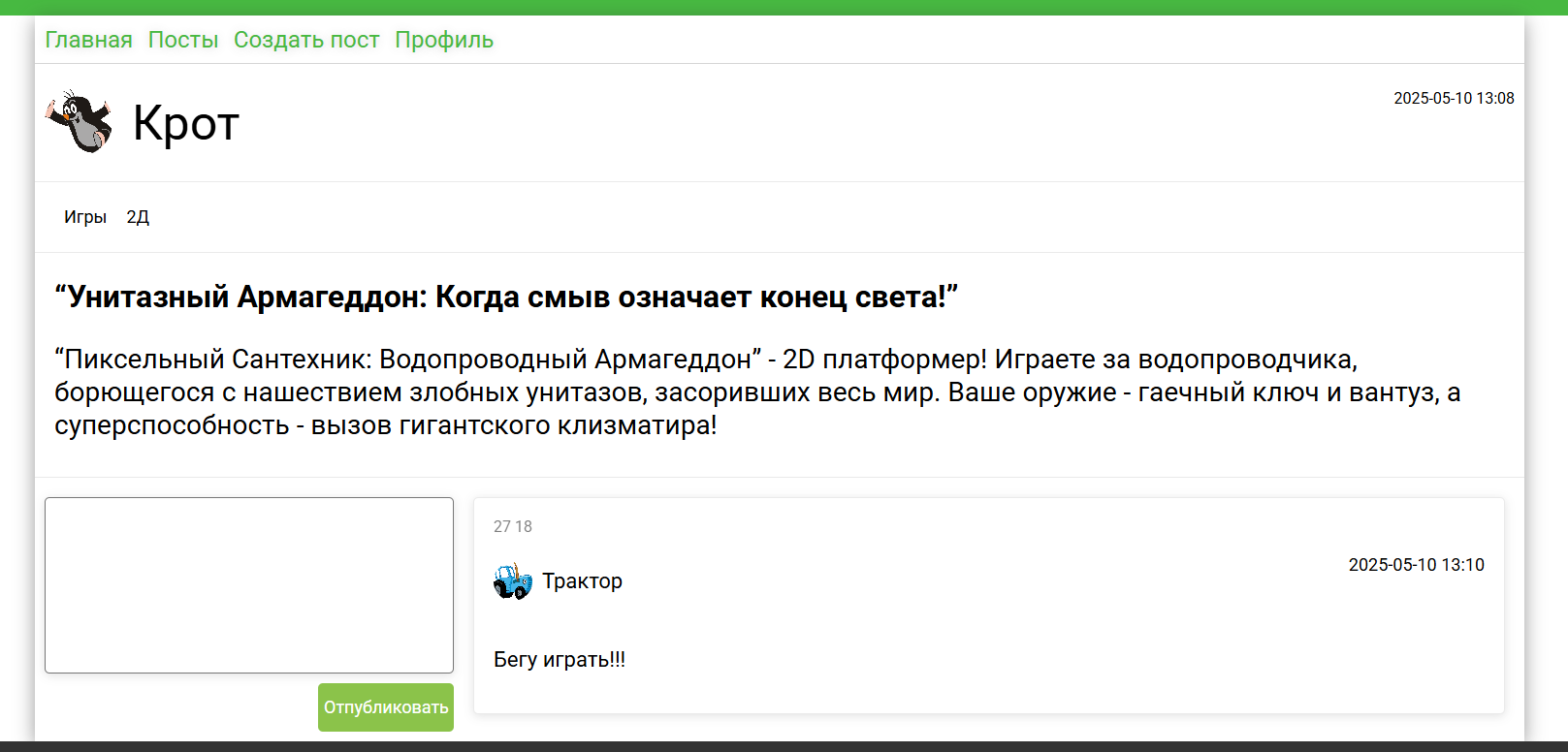
Для страницы создания нового поста была выбрана простая и интуитивная схема: в верхней части размещается поле ввода заголовка, чуть ниже — большое текстовое поле для основного содержимого (Рис. 7). Рядом с формой можно увидеть подсказки по форматированию (например, как вставить картинку или оформить цитату). Важным элементом верстки здесь стала кнопка «Опубликовать», выполненная в цветовой гамме, контрастирующей с фоном, чтобы её было легко найти. Поля формы обведены приятной тонкой границей и имеют внутренние отступы, что делает набор текста более комфортным. При заполнении формы предусмотрена подсветка ошибок — если пользователь оставил поле пустым или ввёл слишком короткий текст, боковая подсказка выдвигается плавно и указывает, что именно нужно исправить.

  
Рисунок 7. Создание поста.

Страница профиля пользователя также оформлена с акцентом на визуальную аккуратность (Рис. 8). В левой части экрана располагается аватарка (аватарку можно загрузить самостоятельно; при отсутствии изображения отображается стандартная иконка-заглушка), справа от неё — основные данные: имя пользователя, количество опубликованных постов и комментариев. Под этими данными расположена вкладка с перечислением всех постов, которые этот пользователь создал. Каждая запись оформлена в виде компактной карточки, в которой видно название темы, дату публикации и несколько строк превью текста. Если пользователь прокручивает страницу вниз, список плавно подгружается, чтобы не перегружать интерфейс сразу всеми записями.

  
Рисунок 8. Страница профиля.

Страница просмотра конкретного поста выполнена с учётом необходимости сконцентрироваться на содержимом (Рис. 9). Вверху отображается заголовок, под ним — имя автора и дата создания поста. Ниже расположен основной текст, обёрнутый в контейнер с фиксированной шириной и достаточными боковыми отступами для комфортного чтения. Справа или снизу (в зависимости от размера экрана) выводится область для комментариев: форма добавления комментария аккуратно встроена в общий макет, а сами комментарии оформлены в виде чётких блоков с указанием автора и времени публикации. Все элементы отделены друг от друга лёгкими тенями и границами, что делает страницу более «воздушной» и читабельной.

  
Рисунок 9. Просмотр поста.

Таким образом, верстка страниц форума стала основой для комфортного взаимодействия пользователей с приложением, обеспечив единый и понятный интерфейс как на десктопах, так и на мобильных устройствах. Каждый элемент — от навигационной панели до формы комментариев — продуман в деталях, а показательные скриншоты помогут продемонстрировать внешний вид и логику работы интерфейса.

## 2.4 Размещение веб-приложения на хостинге

При развертывании Flask-приложения на хостинге первым шагом является подготовка окружения: на сервере устанавливают нужную версию Python и создают изолированную виртуальную среду, куда загружают все библиотеки, перечисленные в проекте. В корне размещают файлы приложения (код, шаблоны, статические ресурсы) и базу данных (например, SQLite-файл или конфигурацию подключения к внешнему СУБД). Помимо этого, проверяют конфигурационные параметры в коде: пути к папке загрузки изображений, секретный ключ для сессий и настройки SQLAlchemy (URI базы данных, отключение трекинга изменений).

Для продакшен-режима запускают WSGI-сервер Gunicorn, который берёт на себя обработку Python-запросов и создание рабочих процессов. При этом указывают количество воркеров, привязку к Unix-сокету и, при необходимости, логирование в отдельный файл. Результатом работы Gunicorn становится файл-сокет (например, app.sock), через который веб-сервер может передавать HTTP-запросы в Flask-код. Этот шаг важен для разделения процессов: приложение не слушает напрямую сетевые порты, а отдаёт обработку на Gunicorn, что повышает устойчивость и масштабируемость.

Затем настраивают Nginx как обратный прокси: он принимает внешние HTTP-запросы и перенаправляет их на Unix-сокет, где «сидит» Gunicorn. В конфигурации выбирают доменное имя или IP-адрес, указывают корневую папку для статических файлов (чтобы CSS, JS и изображения отдавались напрямую, без участия Python), и задают блоки proxy\_pass с заголовками (Host, X-Real-IP, X-Forwarded-For, X-Forwarded-Proto) для корректной передачи информации об источнике запроса. Такая схема разгружает приложение и ускоряет отдачу статики.

Не менее важно уделить внимание правам доступа: директория с кодом и папка для загруженных файлов должны быть доступны только нужным пользователям (обычно www-data или эквивалент), а файл Unix-сокета — обязательно принимать соединения только от Nginx. Для этого меняют владельца и права на папки и сокет, чтобы исключить ошибки «Permission denied». В результате Nginx получает возможность подключаться к Gunicorn без лишних прав, а пользователи вне данной группы доступа не смогут получить прямой доступ к сокету или «сырому» коду.

Кроме того, отображение работы и отладка визуальных ошибок происходит через журналы: обычно выделяют отдельные логи Nginx для ошибок и доступа, а также лог-файл Gunicorn, куда пишет критичные сообщения и tracebacks. При изменении кода в продакшен-окружении можно задействовать перезагрузку Gunicorn (graceful reload) через сигналы, что позволяет вносить обновления без полного отключения сайта.

После всех настроек и проверки доступности по адресу сайта (домену или IP-адресу) приложение оказывается онлайн: Nginx отдаёт статику, передаёт динамические запросы через сокет на Gunicorn, а тот уже вызывает Flask-код, взаимодействует с базой данных и возвращает ответы. Такая архитектура обеспечивает надёжность, разделяет обязанности между компонентами и упрощает дальнейшее сопровождение и масштабирование проекта.

# Заключение

В рамках курсового проекта была поставлена и успешно решена задача разработки веб-приложения «Блог-платформа» с использованием микрофреймворка Flask. Была проведена предварительная аналитическая работа, включающая изучение требований пользователей, выбор технологического стека и построение архитектуры приложения.

В практической части реализована функциональность создания, редактирования и удаления публикаций, система регистрации и аутентификации пользователей, а также раздел комментариев. Для хранения данных использована СУБД SQLite с ORM SQLAlchemy, что обеспечило простоту взаимодействия с базой и гибкость при масштабировании. Интерфейс выполнен на основе Bootstrap и JavaScript, что позволило добиться адаптивности и удобства использования.

Тестирование показало стабильную работу всех модулей: пользовательские сценарии (регистрация, публикация, комментирование) отрабатывают корректно, а при развёртывании на сервере во связке Gunicorn + Nginx приложение демонстрирует ожидаемую производительность и надёжность.

Дальнейшие направления развития проекта могут включать:

* интеграцию более мощной СУБД (например, PostgreSQL) для обеспечения высокой нагрузки;
* расширение возможностей системы модерации и уведомлений;
* внедрение RESTful API для мобильных клиентов или внешних сервисов;
* реализацию более сложных механизмов поиска и фильтрации контента.

Таким образом, поставленная цель достигнута: разработанное на Flask веб-приложение обеспечивает базовый функционал блог-платформы и может служить основой для дальнейшего развития и совершенствования.

# Список использованной литературы

1. Flask documentation [Электронный ресурс]. — URL: https://docs.flaskproject.com/ (Дата обращения: 01.05.2025).
2. MySQL Documentation [Электронный ресурс]. — URL: https://dev.mysql.com/doc/ (Дата обращения: 11.05.2025).
3. Документация Visual Studio Code [Электронный ресурс].–URL: https://code.visualstudio.com/docs/ (Дата обращения: 09.05.2025.)
4. Real Python Tutorials[Электронный ресурс]. — URL: https://realpython.com/ (Дата обращения: 14.05.2025).
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).

# Приложение

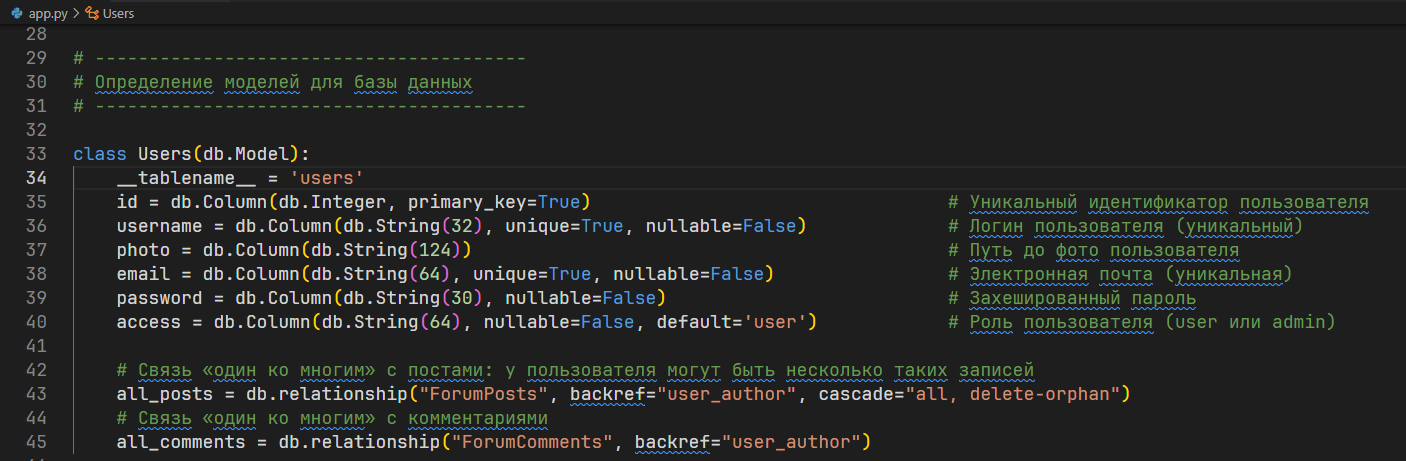


Рисунок 1. Модель User.

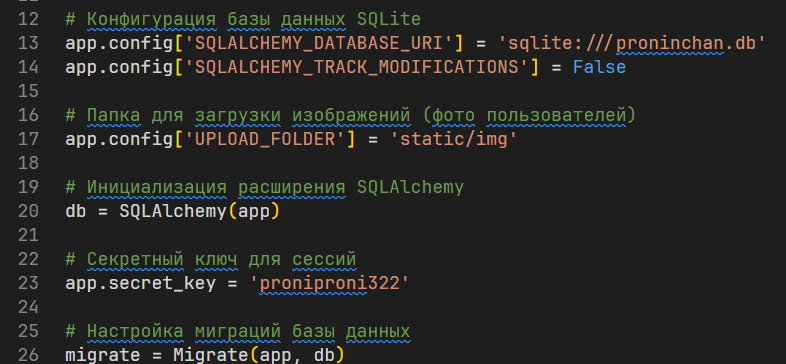


Рисунок 2. Конфигурация БД.

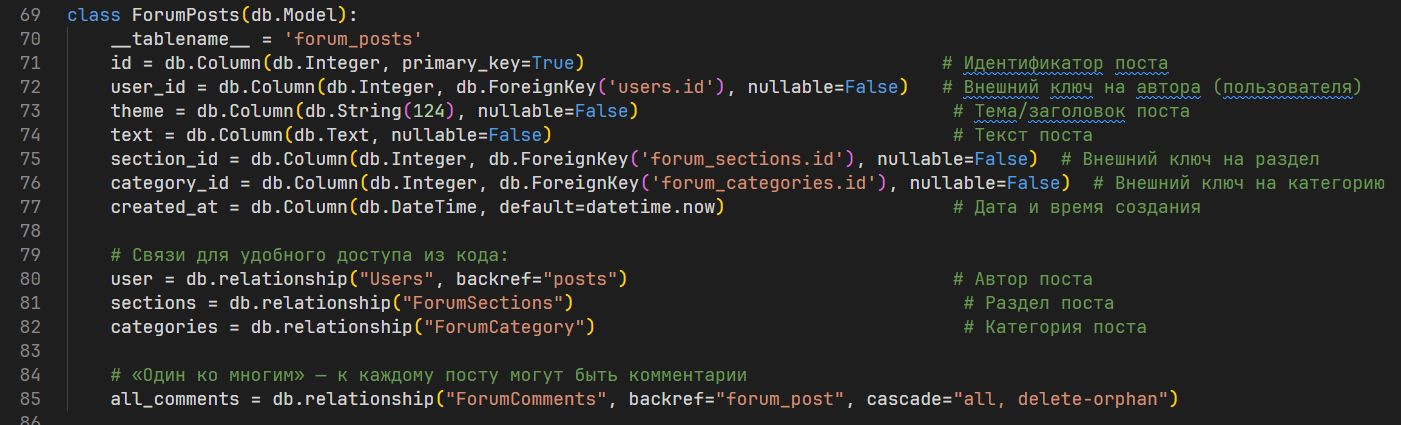


Рисунок 3. Настройка поста в БД.

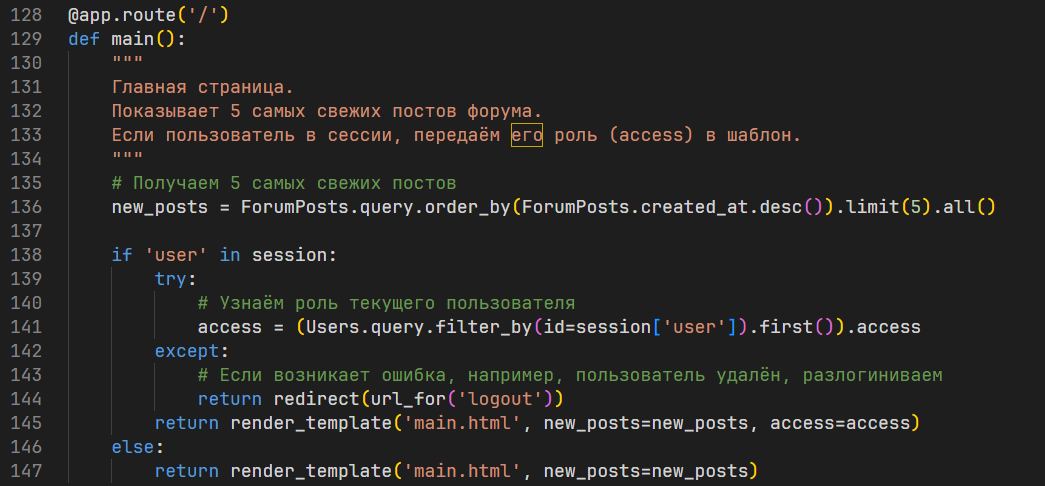
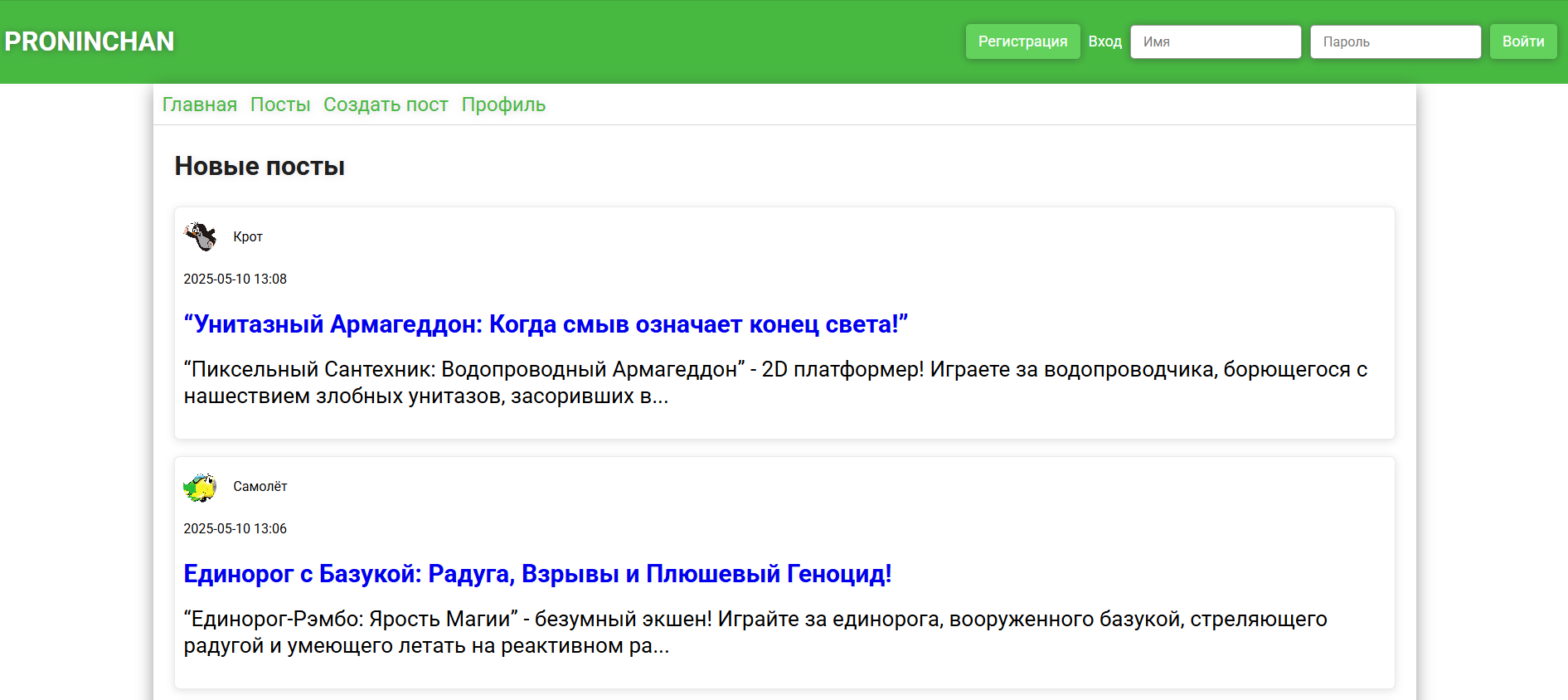


Рисунок 4. Маршрут главной страницы.

  
Рисунок 5. Главная страница.

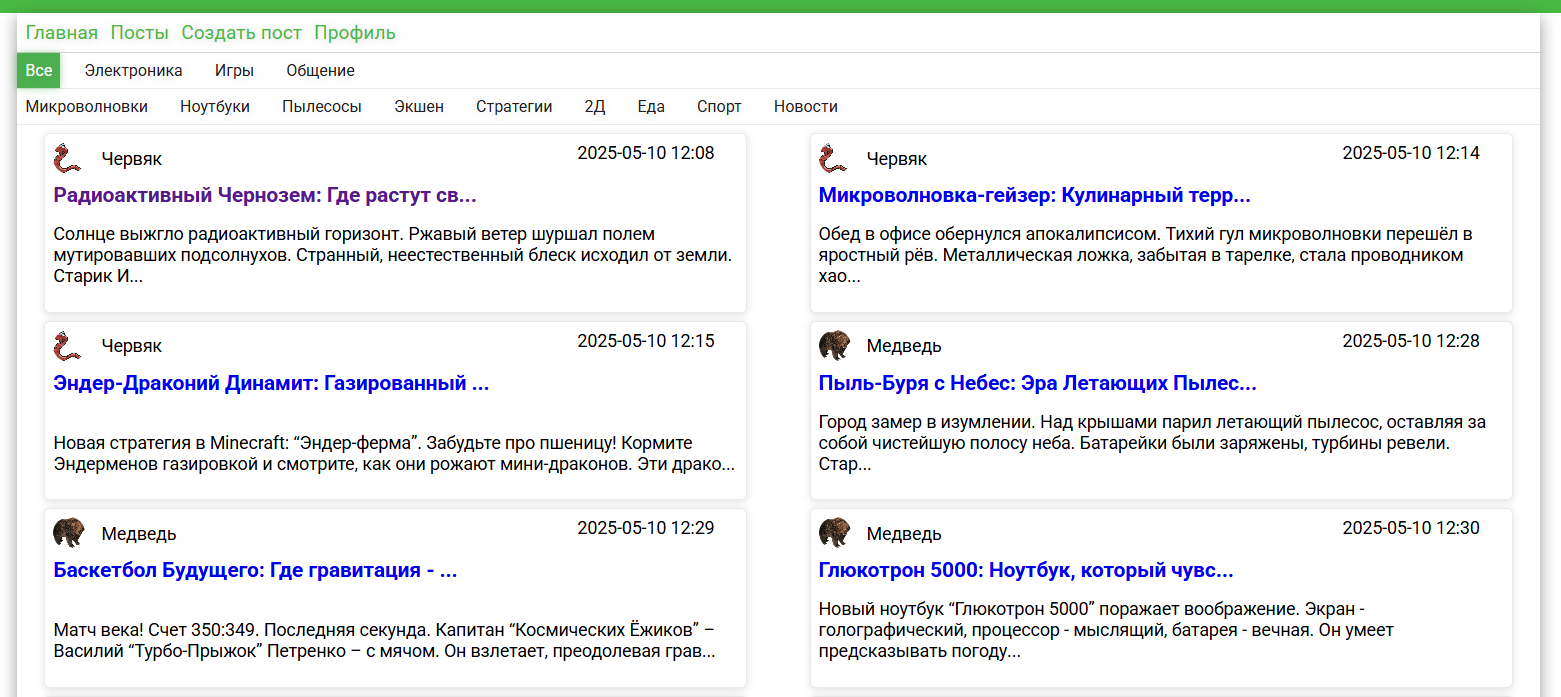
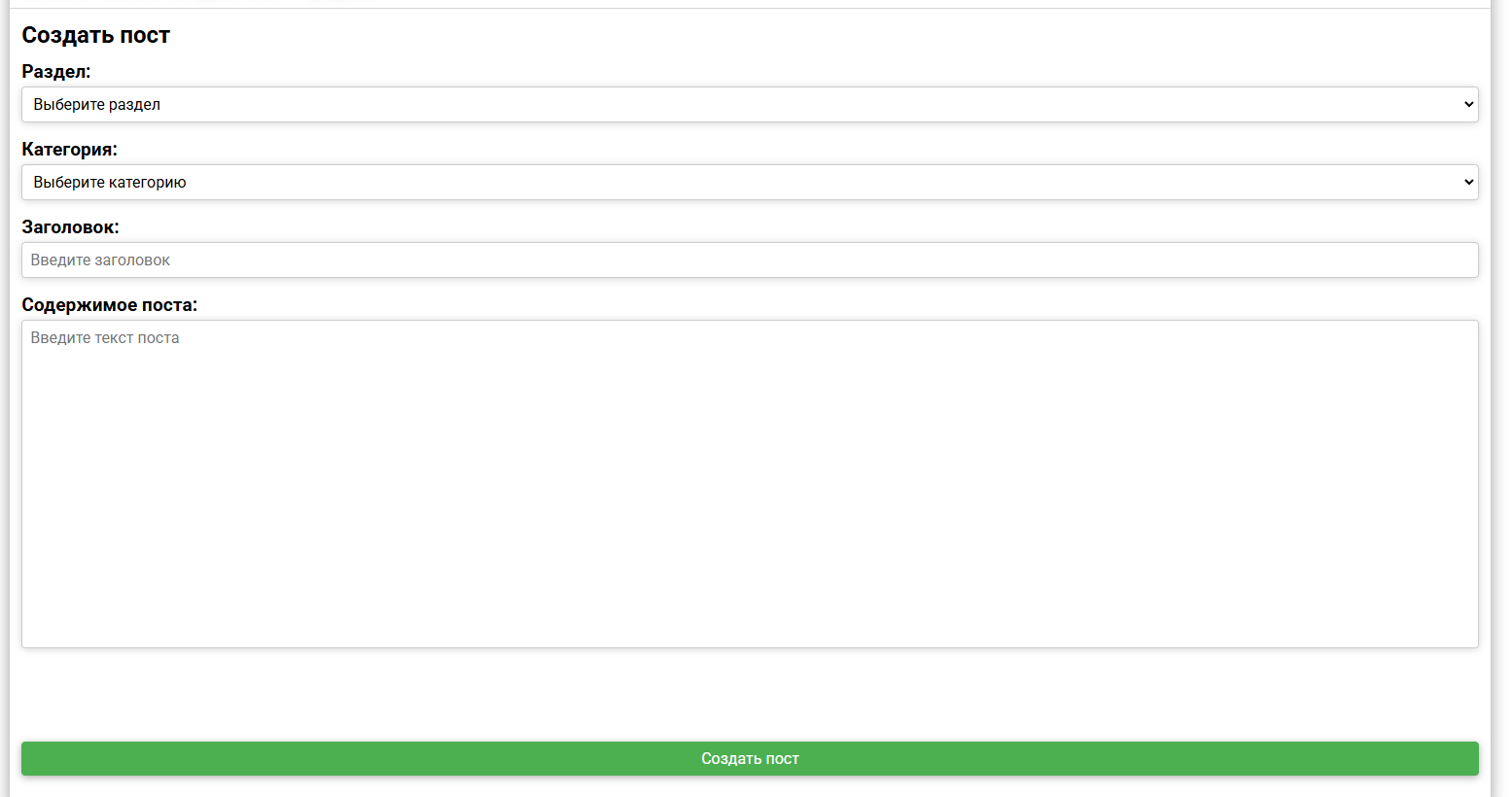
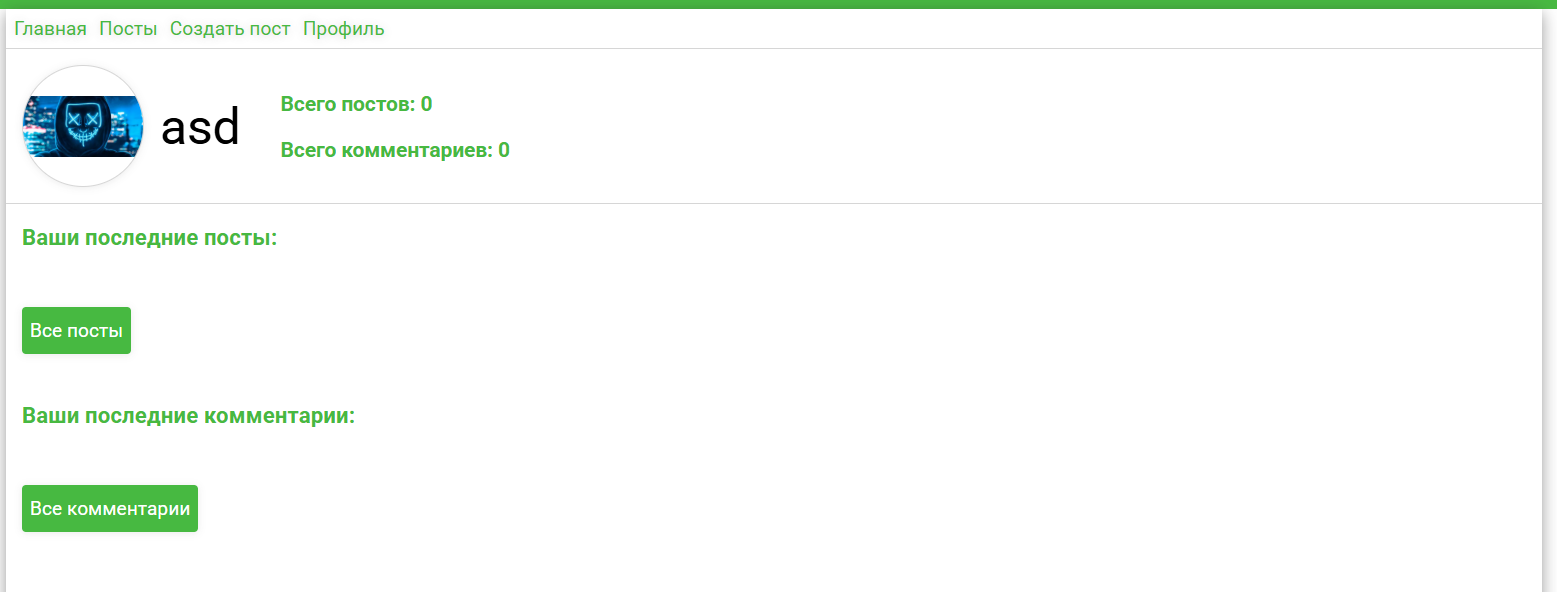
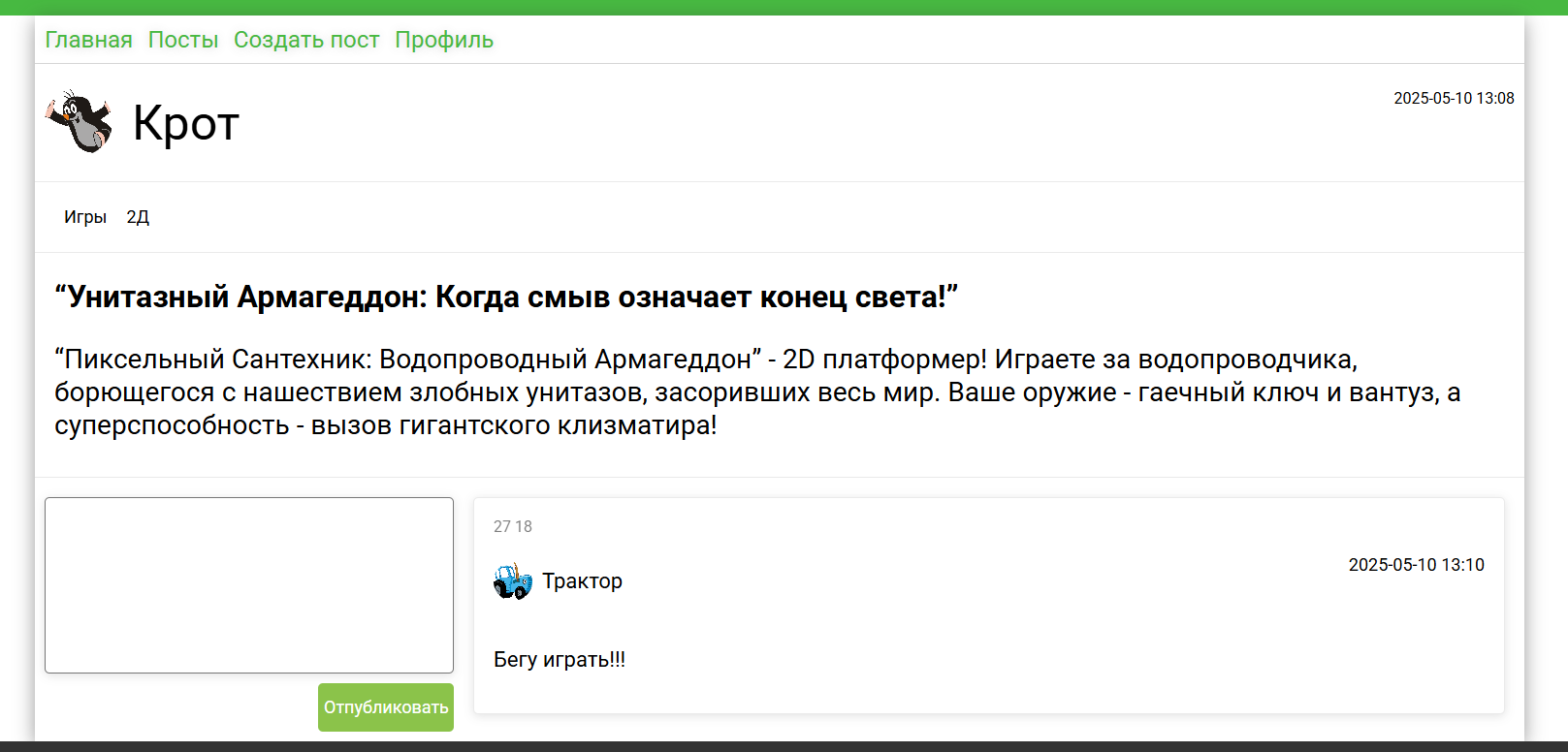


Рисунок 6. Страница списка постов.

  
Рисунок 7. Создание поста.

  
Рисунок 8. Страница профиля.

  
Рисунок 9. Просмотр поста.