|  |
| --- |
| Пермский филиал федерального государственного автономного образовательного учреждения высшего образования  «Национальный исследовательский университет  «Высшая школа экономики»  *Факультет социально экономических и компьютерных наук* |
| Антипин Егор Вячеславович  **Социальная сеть**  **«Minds»**  *Курсовая работа*  студентов образовательной программы «Программная инженерия» по направлению подготовки *09.03.04 Программная инженерия*  Руководитель  Доцент кафедры информационных технологий в бизнесе  В. О. Кушев |

Пермь, 2023 год

Аннотация

В данном документе представлен отчет о разработке социальной сети. Автор работы: Антипин Егор Вячеславович, студент факультета социально экономических и компьютерных наук, первого курса образовательной программы «Разработка информационных систем для бизнеса».

Руководитель: Кушев Вадим Олегович.

Объектом исследования является публикационная деятельность людей в сети Интернет.

Предметом исследования являются социальные сети.

Целью данного проекта является разработка социальной сети с использованием технологий React, TypeScript, Node.js, PostgreSQL .

В рамках проекта был проведен анализ существующих решений на рынке. На основе полученных данных было составлено техническое задание (ТЗ) на разработку приложения.

Работа включает 33 страниц, 4 главы, 1 страницу приложения.

Оглавление

[Введение 4](#_Toc164201561)

[1 Анализ публикационной деятельности в сети Интернет 6](#_Toc164201562)

[1.1 Обоснование выбора темы 6](#_Toc164201563)

[1.2 Анализ существующих решений 6](#_Toc164201564)

[1.3 Заключение анализа 10](#_Toc164201565)

[2 Проектирование приложения 12](#_Toc164201566)

[2.1 Принципы взаимодействия с приложением и основные возможности 12](#_Toc164201568)

[2.2 Проектирование пользовательского интерфейса 12](#_Toc164201569)

[2.3 Проектирование базы данных 17](#_Toc164201570)

[2.4 Связь клиентской и серверной частей 18](#_Toc164201571)

[3 Разработка приложения 20](#_Toc164201572)

[3.1 Серверная часть 20](#_Toc164201574)

[3.2 Клиентская часть 25](#_Toc164201575)

[3.3 Тестирование приложения 33](#_Toc164201576)

[ПРИЛОЖЕНИЕ А 35](#_Toc164201577)

[Техническое задание 35](#_Toc164201578)

[Код программы 35](#_Toc164201579)

# Введение

Социальные сети являются одним из самых популярных и востребованных видов интернет-сервисов, которые позволяют людям общаться, обмениваться информацией, выражать свои мнения и интересы. Социальные сети также играют важную роль в формировании общественного мнения, распространении новостей, продвижении брендов и товаров, создании сообществ и групп по интересам. Существует множество различных социальных сетей, которые отличаются по своей целевой аудитории, функциональности, дизайну и концепции. Однако, не все социальные сети одинаково популярны и успешны на разных рынках. Например, на российском рынке доминируют такие социальные сети, как ВКонтакте, Одноклассники, Мой Мир, а также мессенджеры, такие как Telegram, WhatsApp, Viber. Данные соцсети начинают обрастать широким функционалом, из-за чего становятся слишком загружены новыми функциями, которыми не пользуется большая часть основной аудитории. Из-за этого, среди всех этих социальных сетей нет такой, которая была бы аналогом Twitter - платформы для публикации коротких сообщений (твитов), которые могут содержать текст, изображения, видео, ссылки, хэштеги, упоминания других пользователей.

Цель работы: создать удобное веб-приложение с интуитивно понятным интерфейсом, где могут просто делиться своими мыслями друг с другом, а также получить опыт разработки веб-приложений с использованием React и TypeScript, node.js, и использования баз данных.

Для достижения цели потребуется выполнить ряд задач:

1. проанализировать, изучить существующие решения.
2. спроектировать интерфейс приложения и будущие механики приложения.
3. реализовать серверную и клиентскую части.
4. протестировать приложение.
5. развернуть его на сервере в сети Интернет.

Объектом исследования является публикационная деятельность людей в интернете. Среди тематик приложений была выбрана социальная сеть. Именно разработка социальной сети является предметом исследования.

Наиболее популярным инструментом для использования социальных сетей является смартфон и компьютер.

Для работы над проектом был выбран метод анализа. Он включается в себя исследование аналогов, а также создания приложения, путем разработки дизайна, клиентской и серверной части.

# Анализ публикационной деятельности в сети Интернет

При выполнении проектно-исследовательской работы на основании выполненного анализа формируются требования к программной системе и концепция программного продукта.

## Обоснование выбора темы

В последнее время все больше людей начинают предпочитать несколько социальных сетей с точечным функционалом большим разросшимся социальным сетям. Это подтверждает то, что люди чаще используют мессенджеры для общения, чем другие виды социальных сетей. Тоже самое наблюдается и в других сферах социальных сетей. Однако, в нашей стране нет массового решения, которое позволило бы людям обмениваться короткими мыслями, хештегами, фото и видео. Поэтому мной была выбрана данная тема.

Процесс разработки веб-приложения позволяет приобрести новые и закрепить уже приобретенные навыки использования баз данных, которые изучались нами на протяжении второго курса программы «Бизнес информатика» , а также получить опыт в разработке веб-приложений с использованием библиотеки React и языка TypeSctipt.

## Анализ существующих решений

Для выбора стиля интерфейса, режимов печати и методики обучения, нужно выделить плюсы и минусы уже существующих решений. Таким образом, было выбрано несколько самых популярных и посещаемых веб-приложений, выделил критерии для сопоставления и сравнил их.

Были выделены следующие критерии:

* **есть русский язык** – приложение должно быть локализовано для удобного пользования;
* **современный интерфейс –** приложение должно быть визуально не нагруженным, простым и минималистичным, а также удобным в использовании;
* **доступность в РФ –** социальная сеть должна быть доступна из РФ напрямую без использования сторонних методов обхода блокировок;
* **нет рекламы –** приложение не должно иметь рекламу, чтобы пользователь не испытывал дискомфорт при его использовании;
* **Концентрированность на определенно цели –** приложение должно выполнять свою определенную функцию, а не пытаться реализовать максимально возможный функционал.

### Социальная сеть X

**Социальная сеть X**, ранее известная как **Twitter**, представляет собой американский сервис микроблогов и социальную платформу, где пользователи публикуют короткие сообщения, называемые “твитами”, и взаимодействуют с ними. X позволяет пользователям общаться через браузер, мобильное приложение или API. Основанная в 2006 году, она была популярна благодаря ограничению в 140 символов на твит, которое позднее было увеличено до 280 символов для большинства языков. Интерфейс программы представлен на рисунке 1.

Теперь давайте рассмотрим недостатки:

1. **отсутствие русского языка**: X не поддерживает русский язык, что делает его менее удобным для российских пользователей.
2. **реклама**: Наличие рекламы может раздражать пользователей и ухудшать общее впечатление от платформы.
3. **недоступность в России**: X ограничен в России, что делает его менее привлекательным для местных пользователей.

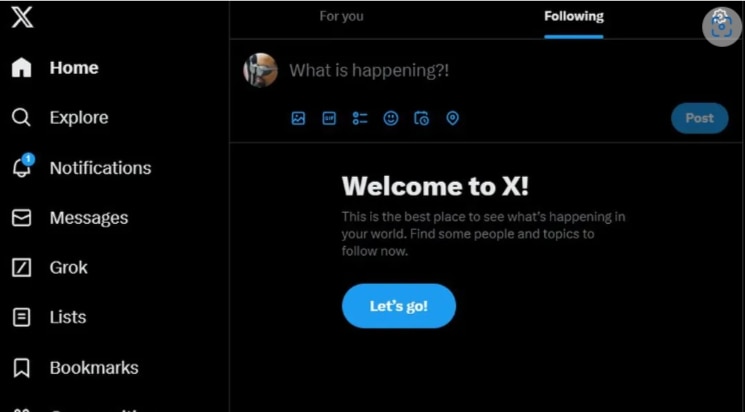


Рисунок 1. – X

### VK

Самая популярная социальная сеть в РФ. Имеет широкий функционал, в который входит создание постов, мессенджер, маркетплейс, доставка еды, сервис такси и множество других. Но в данном случае это не является большим плюсом, так как малое количество пользователей действительно пользуется данными функциями, а большую часть времени они нагружают интерфейс, что больше отпугивает их. В действительность большинство пользователей используют отдельные приложения для каждой потребности. Графическая составляющая продукта представлена на рисунке 2.

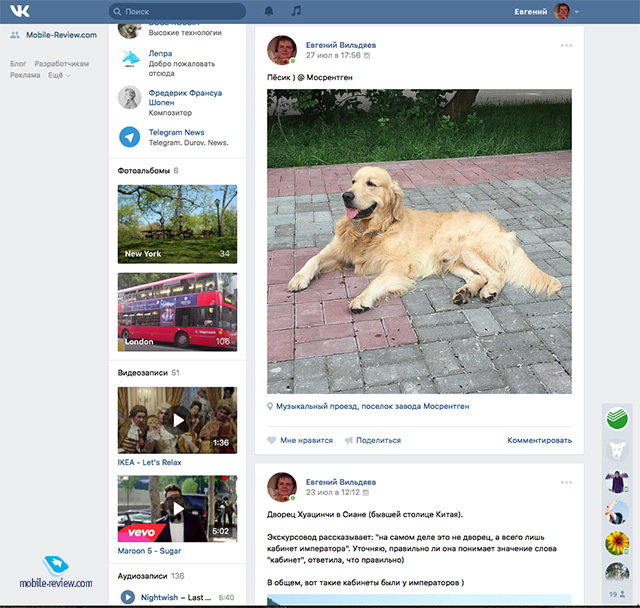


Рисунок 1. – VK

Плюсы данного решения:

* наличие русского языка и доступность в РФ;
* современный интерфейс, хорошая поддержка продукта и частые обновления.
* минусы данного решения:
* частая реклама в ленте, музыке и видео, что уменьшает удовольствие от использования социальной сети;
* чрезмерная перегруженность различным функционалом, что может быть неудобно для пользователей, которые этим всем не пользуются.

### Reddit

Reddit совмещает в себе черты социальной сети и форума. В нем пользователи могут писать и отдельные посты, но главным функционалом реддита являются тематические форумы (сабреддиты), где пользователи могут общаться по интересам, поэтому это веб-приложение не является прямым аналогом приложению нашей темматики. Также хорошим плюсом является то, что пользователи могут анонимно участвовать в обсуждениях. Графическая составляющая приложения представлена на рисунке 3.

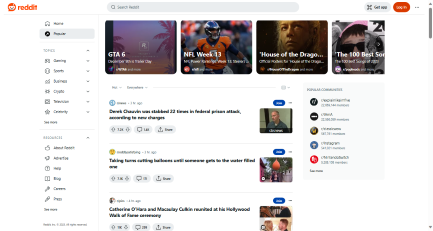


Рисунок 1. – Reddit

Подводя, итог можно выделить следующие минусы данного решения:

* наличие интерфейса только на английском языке, что может быть не удобным для пользователя, который его не знает;
* другой вектор направленности: Reddit используется больше для тематический обсуждений, нежели для публикации своих мыслей.

### Tumblr

Данное веб-приложение имеет уже устаревший дизайн и по функционалу схоже с Reddit. Также тут присутствует много рекламы, что не нравится пользователям. Интерфейс приложения представлен на рисунке 4.

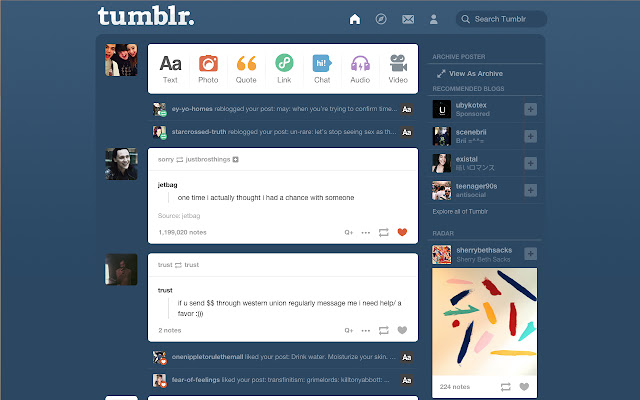


Рисунок 1. – Typing Study

Таким образом, можно сказать, что:

* приложение имеет устаревший дизайн;
* много рекламы;
* похожее на Reddit направление, не подходящее нам.

## Заключение анализа

Сравнение веб-приложений представлена в Таблице №1.

Таблица 1 – оценка аналогов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерий | X | VK | Reddit | Tumblr |
| Есть Русский язык | - | + | - | - |
| Современный интерфейс | + | + | + | - |
| Доступность в РФ | - | + | + | - |
| Концентрированность на определенной цели | + | - | + | + |
| Нет рекламы | - | - | + | - |
| Возможность писать пользовательские публикации | + | + | + | + |
| Возможность создать собственный профиль пользователя | + | + | + | + |
| Возможность дать реакцию на публикации пользователя с помощью лайков и комментариев | + | + | + | + |

Исходя из данных, представленных в таблице №1, можно сделать вывод, что нет существующего решения, которое бы удовлетворяло всем критериям, поэтому, было принято решение разработать собственный продукт, который позволяет:

* создать профиль пользователя;
* писать посты;
* реагировать на чужие публикации.

Функциональные и нефункциональные требования полностью сформулированы в техническом задании (см. Приложение А).

# Проектирование приложения



## Принципы взаимодействия с приложением и основные возможности

Первым этапом проектирования приложения стало определение функционала приложения. Основным функционалом приложения будет являться написание пользователями постов с возможностью добавления к ним фото и видео. Также, чтобы оставить реакцию на посты, должен быть реализован функционал лайков и комментариев. Таким образом, чтобы данные конкретного пользователя сохранялись, нужно также реализовать возможность создания персонального аккаунта и управления им.

С учетом описанного выше, можно выделить следующий функционал приложения:

* **Управление аккаунтом:** регистрация, авторизация, изменения пользовательских данных;
* Написание постов, вывод ленты пользовательских постов, возможность добавить лайк или оставить комментарий;
* **Взаимодействие с другими пользователями:** возможность подписки на пользователя для приоритетного отображения в ленте;
* **Уведомления:** отправка уведомления пользователю при взаимодействии других пользователей с его постами.

Стоит отметить, что пользователь сможет оставлять лайки и комментарии и писать собственные посты только при входе в собственный аккаунт, в противном случае данные функции будут не доступны для сохранения достоверных данных и исключения возможности накрутки.

## Проектирование пользовательского интерфейса

Я решил создать минималистичный пользовательский интерфейс, чтобы пользователь не отвлекался на лишние детали. Также интерфейс будет выполнен в темных тонах, т.к. по моему мнению, темные темы в современных приложениях намного приятнее в использовании, особенно в темное время суток.

Первым делом было спроектировано боковое меню навигации и логотип, внешний вид которых представлен на рисунке 2.1:

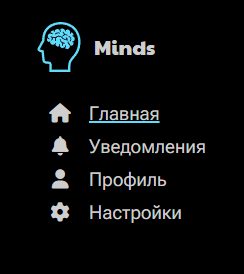


Рисунок 2.1 – Навигационное меню

Стоит обратить внимание, что вкладки «Уведомления» и «Настройки» появляются только после авторизации, а вкладка «Профиль» заменяется на «Вход» у анонимных пользователей.

### Страница настроек

На странице настроек будет расположен функционал смены пароля, логина, имени и почты, а также изображения профиля пользователя. Внешний вид страницы представлен на рисунке 2.3:

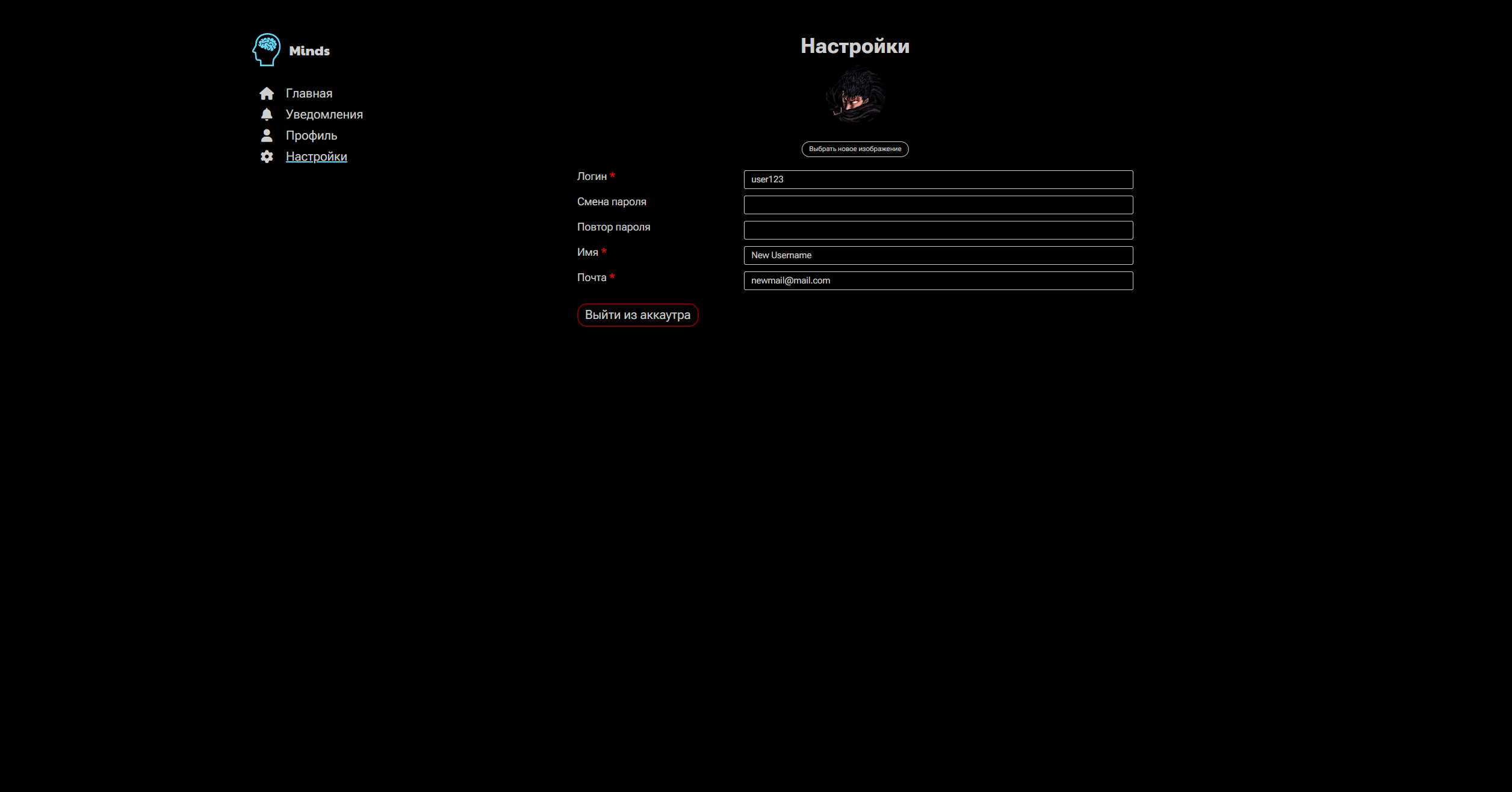


Рисунок 2.3 – Внешний вид страницы настроек

### Страница авторизации

Страница авторизации выполняет роль входа и регистрации пользователей. Внешний вид формы входа представлен на рисунке 2.4

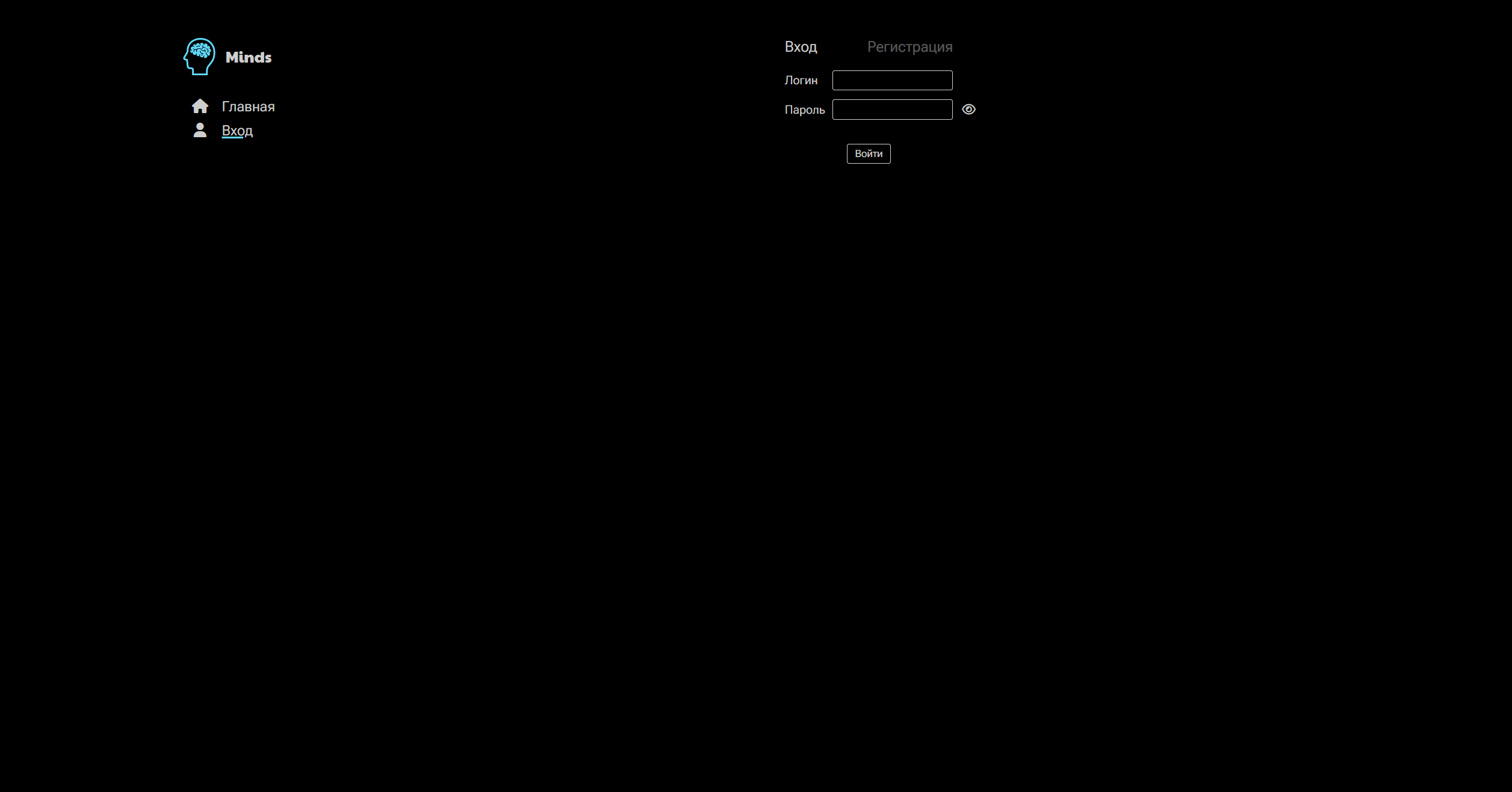


Рисунок 2.4 – форма входа

Как видно из рисунка, вход происходит по логину и паролю пользователя, также справа от поля пароля располагается кнопка в форме глаза, при нажатии на которую тест поля пароля вместо точек отображается как обычный текст.

Форма регистрации представлена на рисунке 2.5:

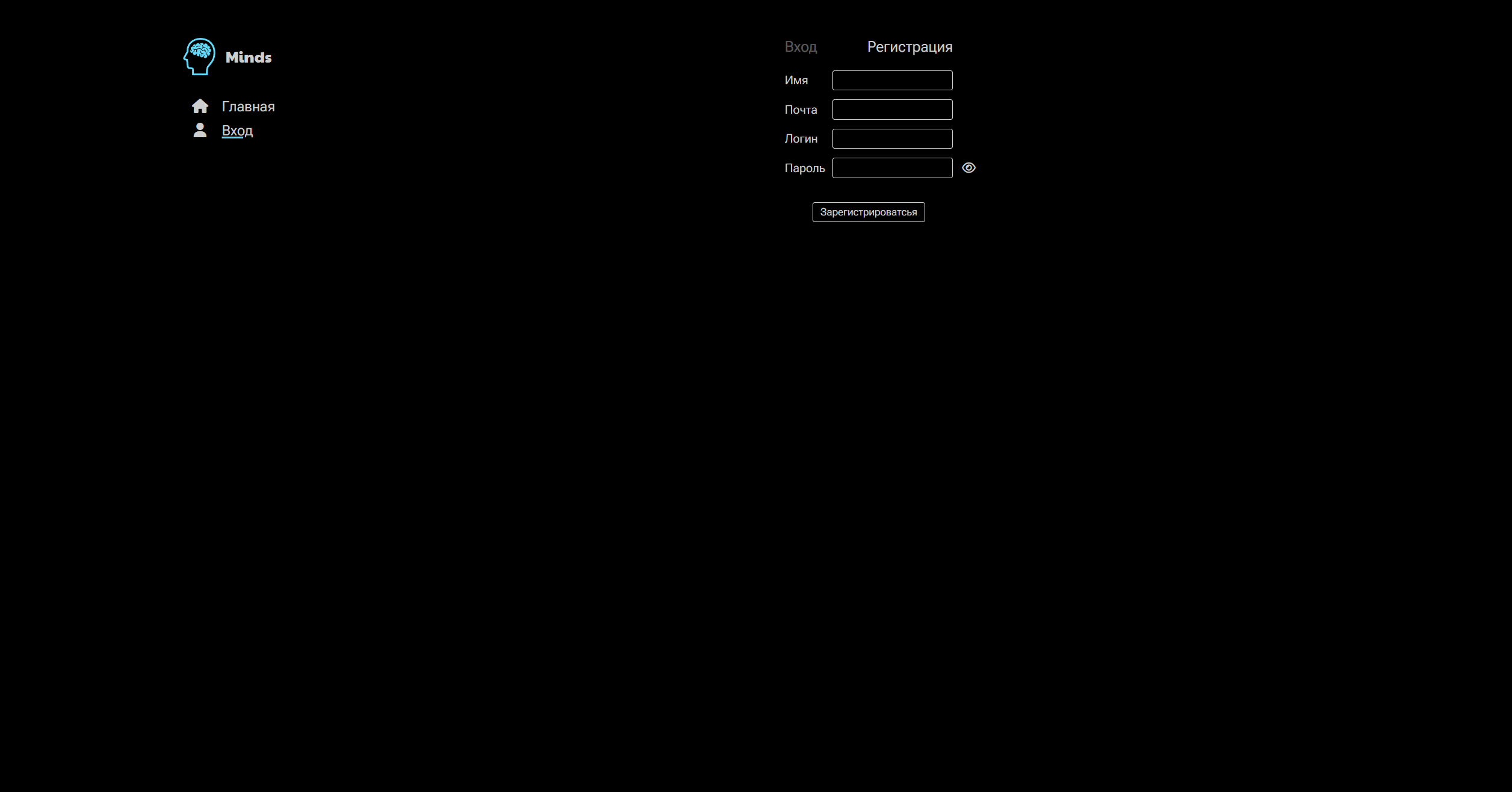


Рисунок 2.5 – Форма регистрации

В форме регистрации добавлены дополнительные поля имени и почты, а также подтверждения пароля, чтобы пользователь точно не ошибся при одиночном вводе.

### Страница пользователя

Внешний вид страницы пользователя представлен на рисунке 2.6:

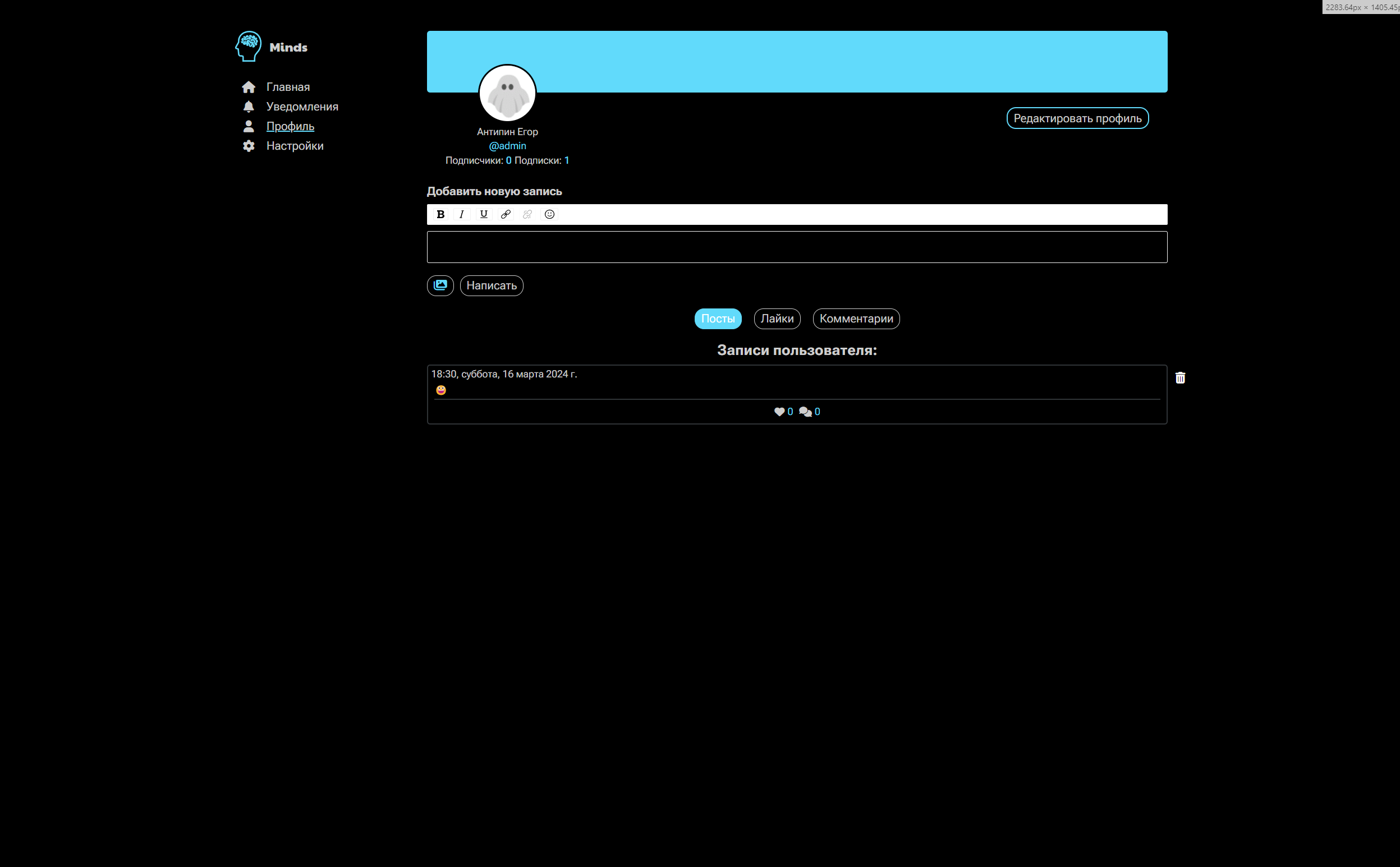


Рисунок 2.6 – Страница пользователя

Она является как страницей любого пользователя, так и личным профилем конкретного пользователя. Разница будет состоять в том, что, если пользователь заходит на свой профиль, то у него будут отображаться кнопки «Редактировать профиль», а также кнопки удаления постов справа от них.

### Страница уведомлений

Страница уведомлений предназначена для вывода уведомлений пользователя о взаимодействиях с его постами, а также подписок на пользователя, внешний вид страницы представлен на рисунке 2.7:

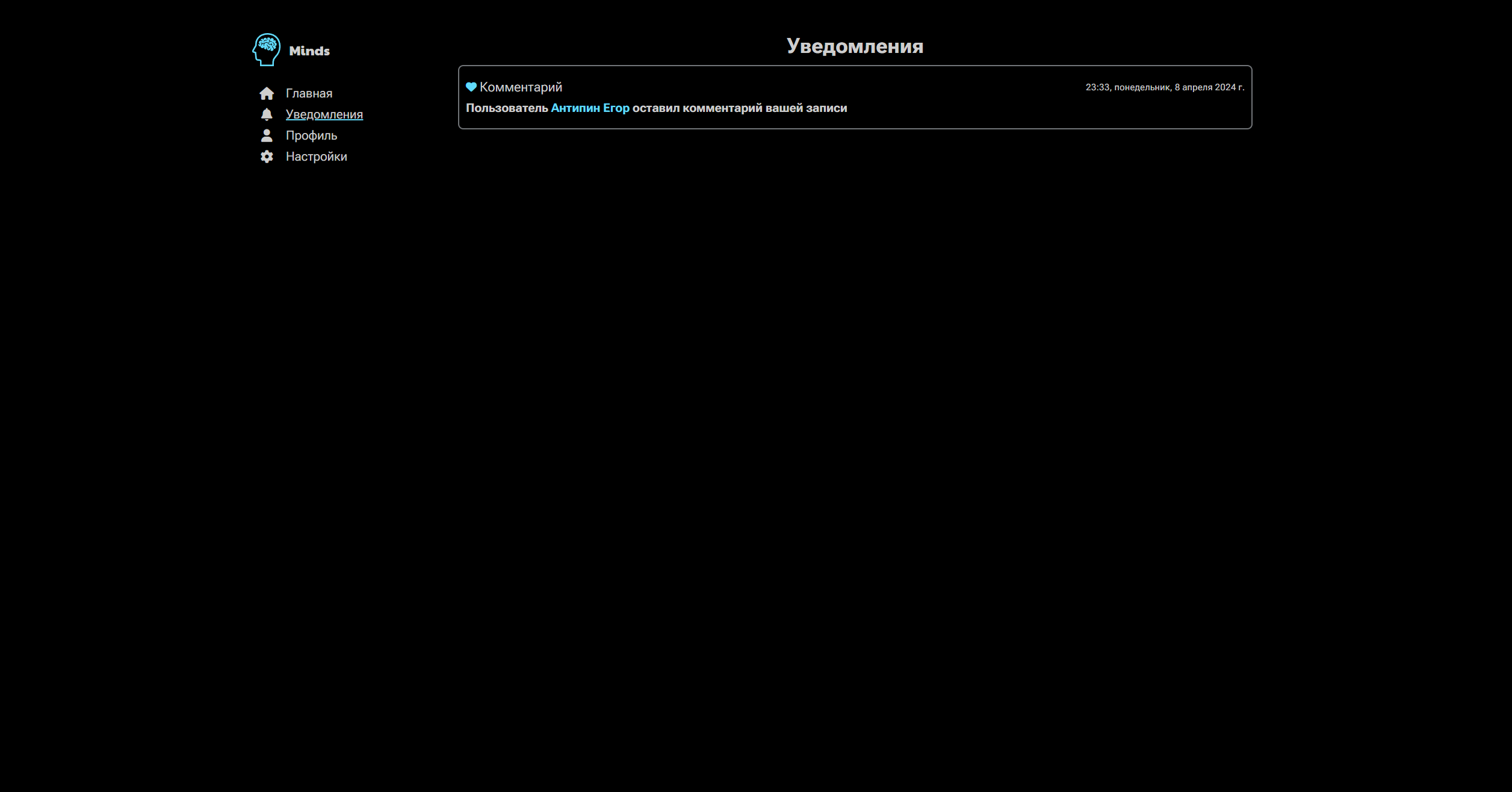


Рисунок 2.7 – Страница уведомлений

Уведомления будут располагаться друг под другом, сортируемые в порядке убывания по дате.

### Главная страница

Основная страница с лентой пользователя, которая может быть доступна даже без регистрации. Внешний вид представлен на рисунке 2.8:

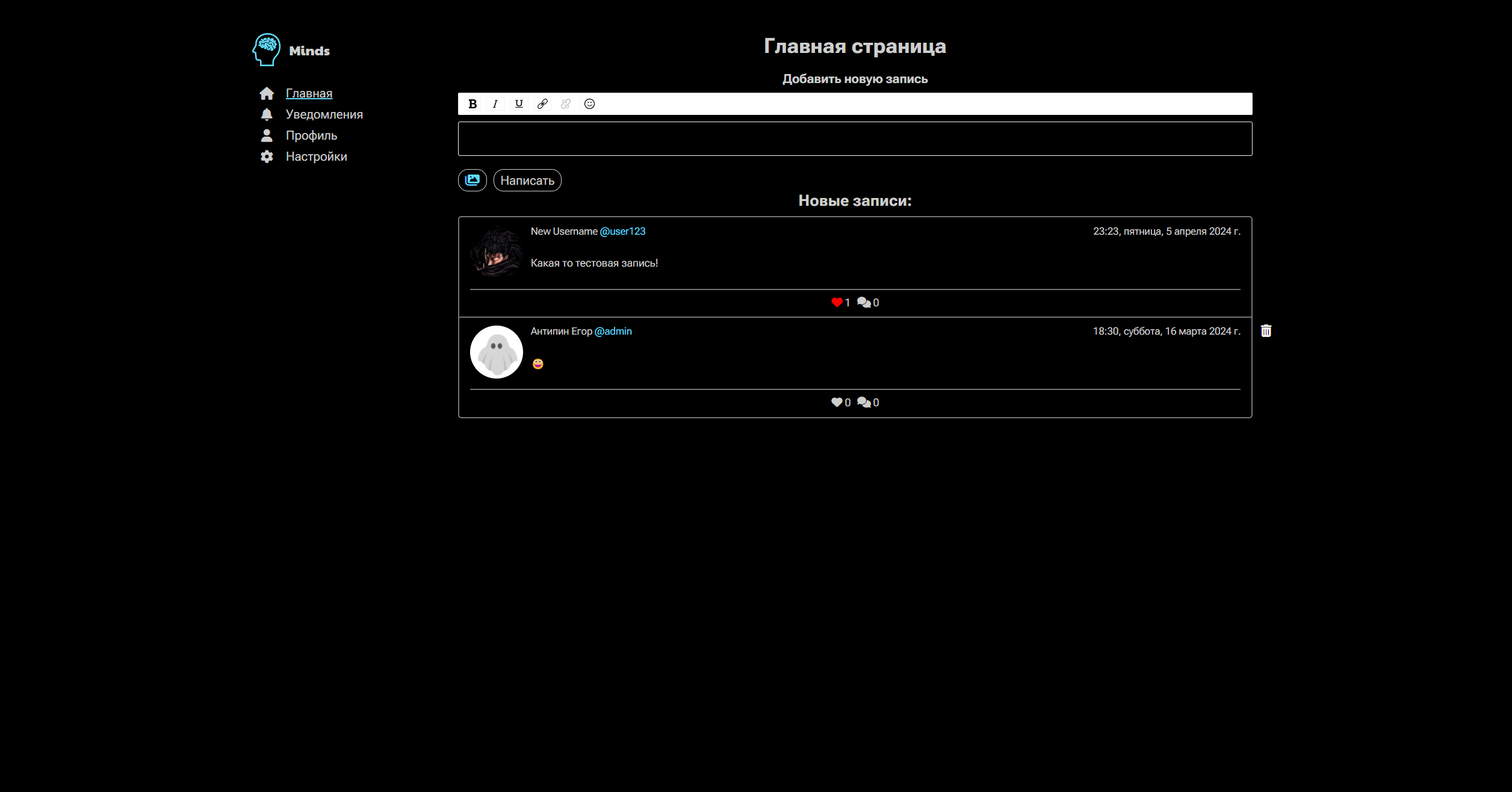


Рисунок 2.8 – Главная страница

Здесь будут отображаться посты пользователей, отсортированные с приоритетом по подписке и дате. Также, если пост принадлежит текущему пользователю, то справа от поста будет расположена кнопка удаления поста.

## Проектирование базы данных

Основные пользовательские данные приложения должны храниться в базе данных для удобного добавления, изменения и получения информации, поэтому в самом начале стоит выбрать базу данных и спроектировать ее. Мной было принято решение использовать базу данных PostgreSQL, так мы уже использовали ее во время курса, связанного с базами данных, а также эта БД является хорошим современным производительным решением.

Вторым этапом является проектирования таблиц БД. Это важный этап, т. к. от него зависит удобство получения данных, а также производительность. Диаграмма таблиц и их связей представлена на рисунке 2.9:

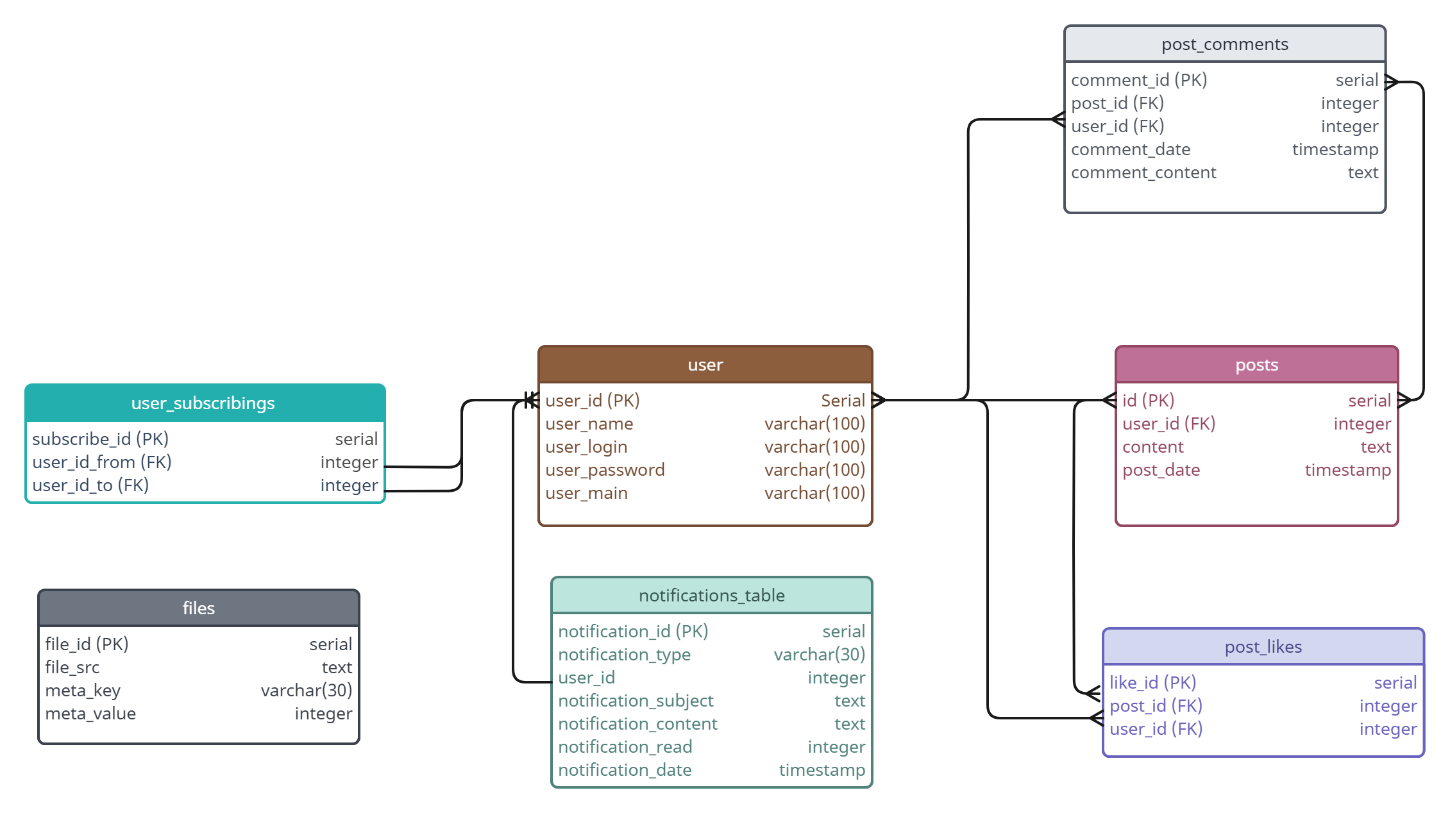
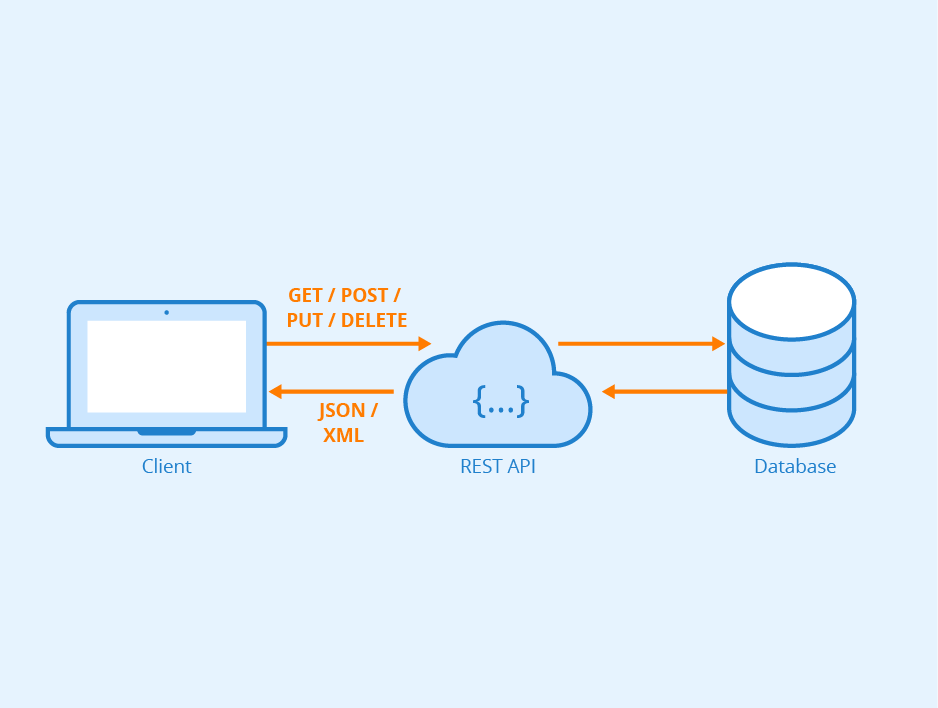


Рисунок 2.9- Диаграмма таблиц базы данных

Стоит отметить некоторые решения, которые я принял во время проектирования базы данных, а также описать функционал таблиц. Основными сущностями будут являться пользователи и посты (таблицы user и posts). Остальные таблицы по сути являются подсущностями этих таблиц. Таблица user\_subscribings предназначена для хранения данных о подписках пользователей друг на друга, таблица post\_likes нужна для хранения данных о лайках пользователей, в таблице post\_comments хранятся комментарии пользователей к постам, а в таблице notifications\_table хранятся уведомления пользователей. Таблица files по сути никак не связана внешними ключами с другими таблицами. Так как у нашего серверного приложения будет реализован маршрут для получения статичных файлов, специально для этого в таблице files будут храниться локальные пути файлов. Столбец meta\_key нужен для определения типа файла, а столбец meta\_value нужен для отношения к определенной сущности. Например, изображение профиля пользователя с ID = 1,которое можно получить по пути url\_сервера+/statis/user/1/profile\_image.png будет храниться в таблице files со значениями meta\_key = «user\_image», meta\_value= 1 (ID пользователя), file\_src = «/user/1/profile\_image.png». Точно также можно будет хранить изображения постов с meta\_key = «post\_image».

## Связь клиентской и серверной частей

Так как клиентская и серверная части по сути будут представлять самостоятельные отдельные приложения, чтобы их связать, будет использована архитектура REST API. Она позволяет удобно передавать данные по протоколу HTTP, что на сегодняшний день является самым безопасным решением. Схема работы REST API представлена на рисунке 2.10.

Рисунок 2.10 — Схема работы REST API

Таким образом, клиентская и серверная части будут обмениваться запросами, отправляя данные и получая ответ.

Благодаря проектированию приложения можно избежать некоторые ошибки в разработке, связанные с непродуманными техническими решениями. После завершения этапа проектирования можно приступить к этапу реализации приложения.

# Разработка приложения



## Серверная часть

В самом начале стоит определиться, на каком языке и фреймворке будет написана серверная часть. Было принято решение использовать язык JavaScript и его фреймворк Express, так как они имеют хорошую интеграцию с PostgreSQL. Также, Express был создан специально для простого и быстрого написания REST API, поэтому он подходит для данного проекта.

Код основного файла сервера представлен на рисунке 3.1:

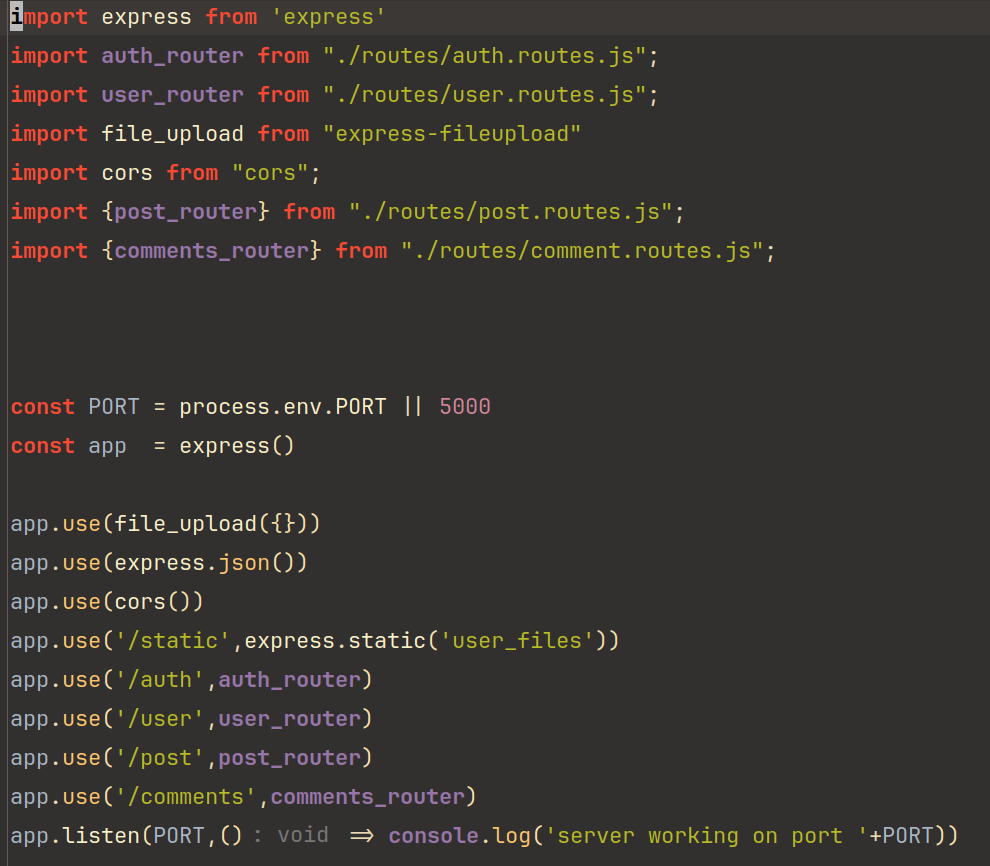


Рисунок 3.1 — Код файла index.js

В нем создается экземпляр класса express, который по сути и является нашим серверным приложением. Далее в него передаются маршруты и роутеры, которые за них отвечают. Роутеры — функции, которые принимают подмаршруты и также сопоставляют с функциями, которые за них отвечают. Пример Роуерта пользователя представлен на рисунке 3.2:

На рисунке видно, что в коде создается экземпляр класса Router, а затем поочередно передаются связки. Также маршруты принимают определенные методы запросов REST API (GET, POST, UPDATE, DELETE).

Рисунок 3.2 — код Роутера пользователя

### Миддлвейры

Иногда в такие «связки» между маршрутом и функцией передается третьи функции, например AuthMiddleware. Они называются миддлвейры и предназначены для каких-либо промежуточных вычислений до получения запроса основной функцией. Разберем один из таких миддлвейров на том же самом примере AuthMiddlware, код которого представлен на рисунке 3.3:



Рисунок 3.3 - код функции AuthMiddleware

Данная функция предназначена для проверки токена пользователя (речь о котором пойдет в главе 3.1.2), его декодировании, а затем дополнении исходного запроса данными пользователя. Если же токе пользователя не переда, то в ответе вернется ошибка с кодом 401 (пользователь не авторизован). Таким образом, в других функциях нам не нужно выполнять постоянную процедуру проверки авторизации пользователя и декодирования его данных из токена, а нужно добавить один миддлвейр.

### Класс обработки авторизации

Для связи маршрутов и функций авторизации и регистрации реализован отдельный статический класс Auth\_controller. В нем реализованы два метода login и register, предназначенные для авторизации и регистрации пользователей. Код функций представлен ниже:



Рисунок 3.4 - код класса AuthController

Нужно пояснить как работает авторизация и регистрация пользователя. Данные функции схожи по принципу работы, за исключением того, что функция register дополнительно сохраняет данные пользователя в базу данных, предварительно проверив на наличие повторяющихся логина или почты. Обе функции создают из данных пользователя токен с помощью функции generateAccessToken, который представляет собой закодированные данные пользователя. Данный токен будет храниться на клиентской части и отправляться на сервер при необходимости получить конфиденциальные данные пользователя.

### Класс User\_controller

Класс User\_controller предназначен для обработки запросов, связанных с данными пользователя. Все запросы, за которые он отвечает, представлены на рисунке 3.2. Полный код класса, как и самого проекта, выложен в репозитории на гитхаб (см. приложение), так как он слишком большой, чтобы представлять полный код в отчете. Стоит отметить основные функции данного класса. Он позволяет получить основные данные пользователя (логин, почту, посты, изображение пользователя), уведомления пользователя, а также его персональную ленту. При связи с роутером пользователя также используется миддлвейр AllowAnonymusMiddleWare, код которого также есть в репозитории. Он схож с AuthMiddleware, за тем исключением, что, если пользователь не авторизирован, то он не прекращает работу с кодом 401, а продолжает без данных пользователя. Таким образом лента может быть как персонализированной, так и для незарегистрированных пользователей.

### Post\_router и Post\_controller

Объект post\_router и класс Post\_controller отвечают за работу с постами пользователя. Код post\_router представлен на рисунке 3.5.

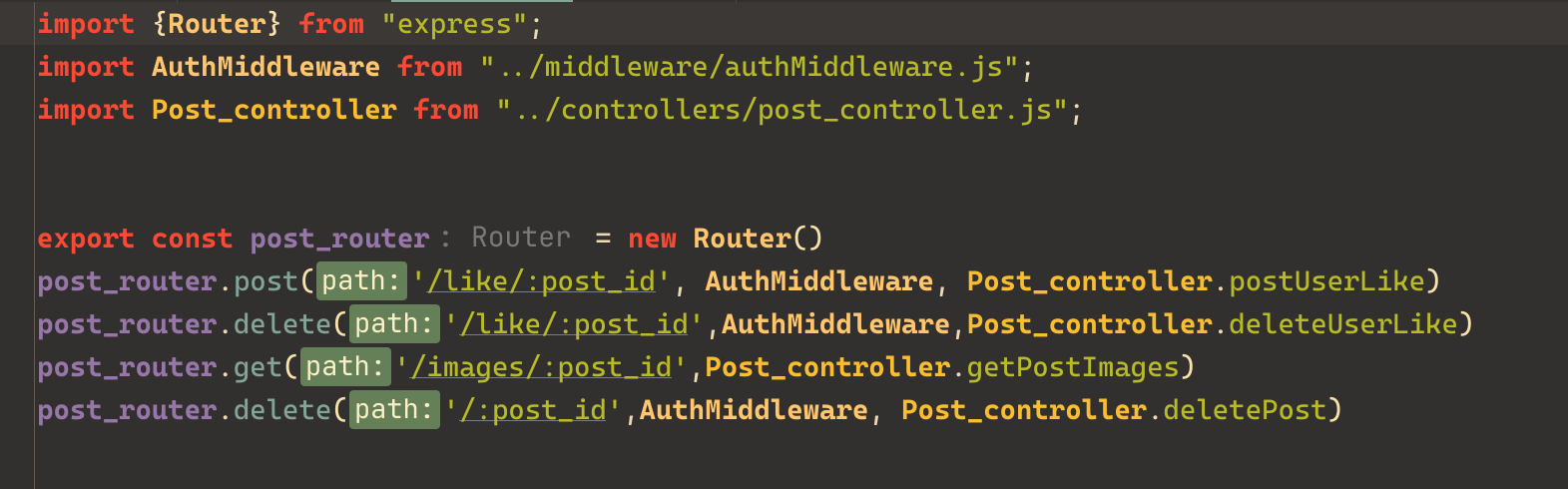


Рисунок 3.5 - код объекта post\_router

С помощью данного роутера обрабатывается добавление и удаление лайка на пост, получение изображений поста, а также добавление и удаление поста. Весь данный функционал реализован в классе Post\_controller, код которого представлен в репозитории (см. приложение). В данном классе также реализован функционал добавления и удаления комментариев, так как он достаточно маленький, чтобы выносить его в отдельный класс.

### Роутер comments\_router

Заключительный роутер, код которого представлен ниже:

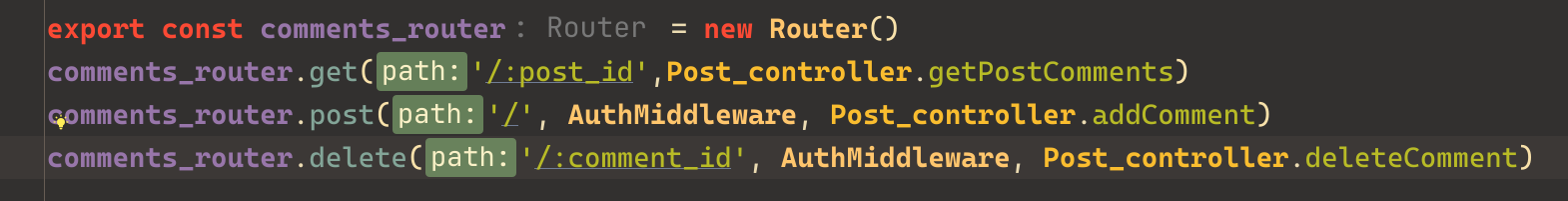


Рисунок 3.6 - код comments\_router

Он предназначен для добавления, получения и удаления комментариев. Как было сказано выше, за функционал здесь отвечает Post\_controller.

## Клиентская часть

В данной главе будет описаны принципы работы клиентской части, а точнее основных ее модулей. Вся клиентская часть написана на языке TypeScript, который является строготипизированной версией обычного JavaScript. Благодаря данной типизации получается отловить некоторые ошибки с данными на этапе разработки. Также основной библиотекой для написания является React. Это современный, быстрый и мощный инструмент, благодаря которому можно написать производительное приложение любой сложности. Также стоит отметить, что вместо языка CSS для написания стилей был использован SCSS. Это тоже измененная версия CSS, которая добавляет вложенность, а также другие дополнительные возможности в простые таблицы стилей, такие как функции, переменные, циклы и т.д.

Хочется отдельно рассказать про библиотеку React. Она была создана компанией Facebook [2] для создания фронтенд-приложений. Основной ее принцип заключается в разбиении всей клиентской части на компоненты, которые являются функциями со своими внутренними переменными, состоянием, событиями и т.д. Они возвращают разметку, схожую с обычным HTML. При изменении состояния компоненты, она автоматически перерисовывается, отображая новые данные. Таким образом, можно переиспользовать данные компоненты в различных местах своего приложения, собирая его как дом из кирпичей.

Чтобы управлять основным состоянием приложения используется менеджер состояния Redux. Он позволяет централизовать управления и получения состояния в одном месте. Таким образом, кроме локальных состояний компонент, появляется возможность хранить общие состояние, например, данных пользователя и получать их из различных компонент.

### Роутер приложения

У данного веб-приложения есть различные пути. Доступ к ним осуществляется при переходе по различным url-путям. Чтобы связать путь и страницу, которая должна отображаться, используется дополнительный пакет React-router-dom. Приложение оборачивается в компоненту данного пакета под названием RoterProvider, в которую передается специальный объект – router. В нем прописывается связь между url-путем и страницей. Также с помощью компоненты Outlet указывается, в каком месте приложения должно отображаться содержания страницы. Код основной компоненты App, которая является входной точкой приложения представлен ниже:

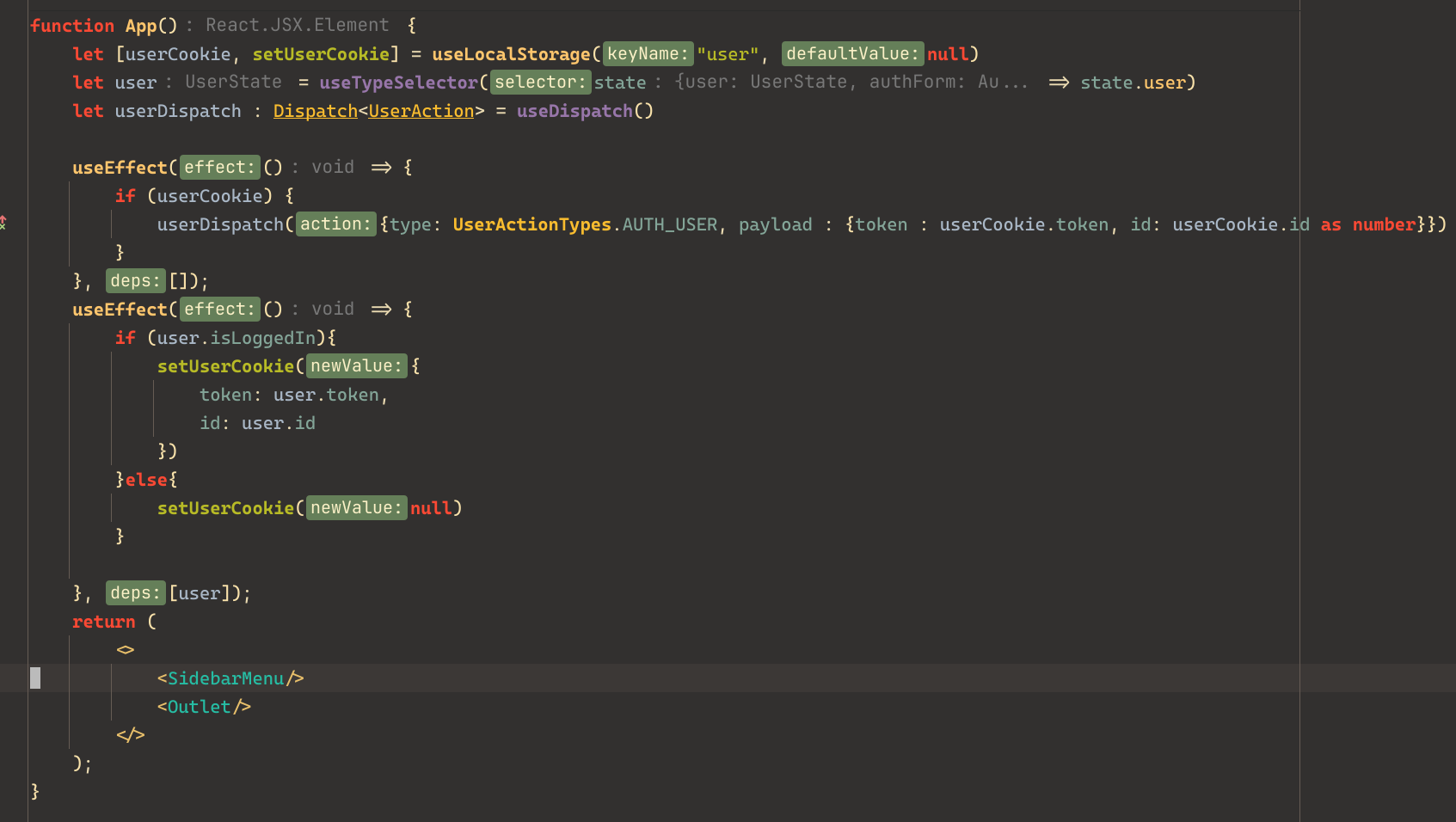


Рисунок 3.6 - компонента App

Таким образом, какая-то часть приложения может оставаться статичной (в нашем случае это боковое меню), а какая-то динамически изменяться и отображаться вместо компоненты Outlet.

### Страница авторизации

Первая страница, которая встречает пользователя при запуске проекта. Её внешний вид уже был представлен выше на рисунке 2.4. Она представляет собой форму, у которой есть основное состояние типа формы (вход или регистрация), от которой зависит, какие поля будут в форме, а также состояния полей формы (по сути данные, которые ввел пользователь). Часть кода страницы представлена ниже:

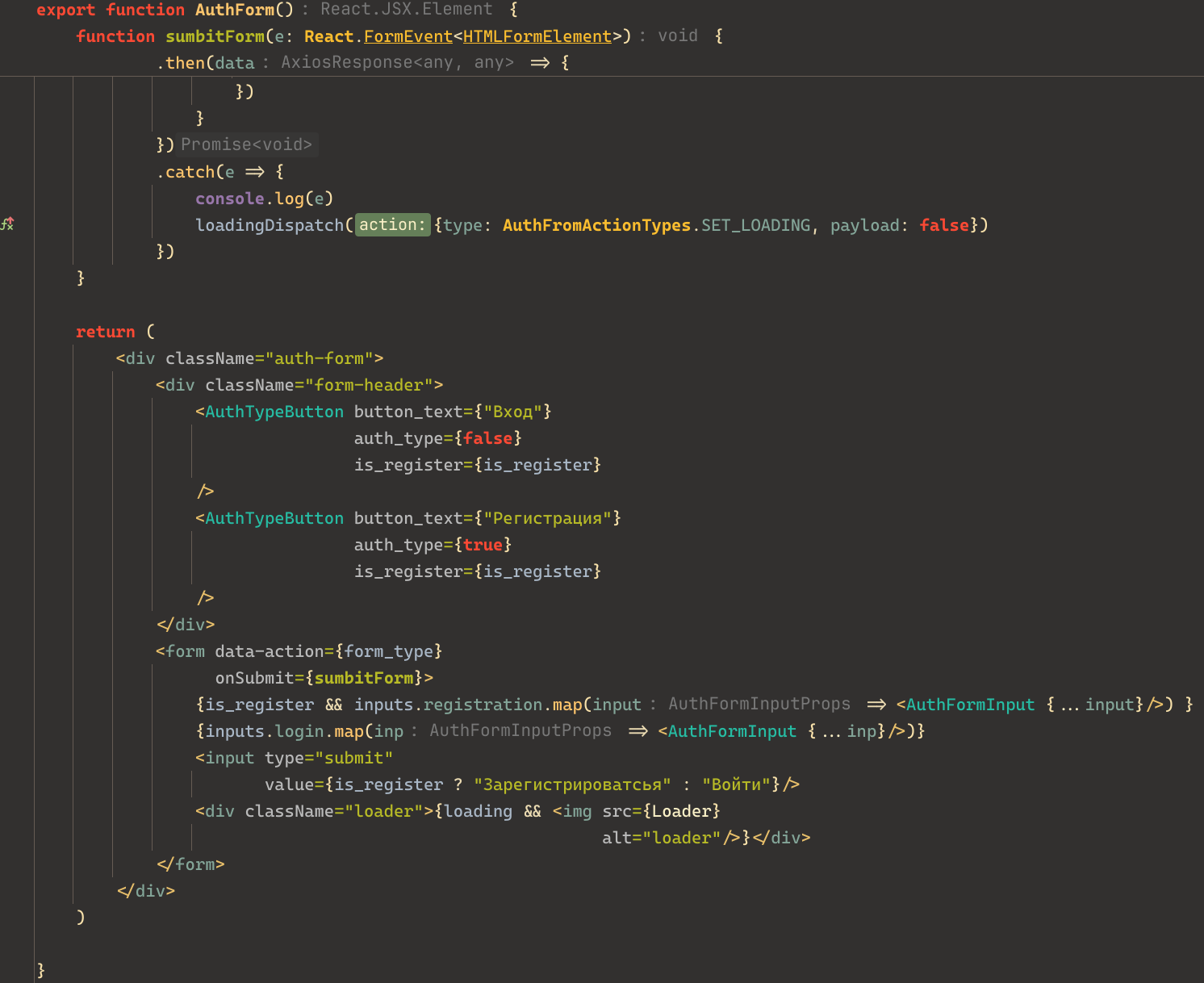


Рисунок 3.7 - компонента AuthForm

Поля формы отрисовываются из отдельной подкомпоненты, которая имеет поля названия поля и типа поля. По нажатию на кнопку отправки формы, происходит валидация всех полей. Если есть какие-либо ошибки, то, над формой всплывает соответствующее сообщение. Если же все хорошо, то форма отправляется на сервер. При успешном ответе сервера, клиентская часть запоминает токен пользователя и перенаправляет его на профиль.

### Страница настроек

Данная страница реализована в отдельной компоненте Settings. Она, также, как и страница авторизации, представляет собой форму с полями, которые может поменять пользователь, а также кнопкой выхода из аккаунта, которая очищает данные о пользователе, тем самым происходит выход из аккаунта. Внешний вид страницы представлен на рисунке 2.5.

Из интересных решений, здесь реализована функция смены изображения профиля с помощью скрытого поля типа файла (input type= «file»). При нажатии на кнопку смены изображения открывается диалоговое окно выбора файла. При выборе файла, он отправляется на сервер и сохраняется в папке user\_files.

Также, кнопка «сменить данные пользователя» появляется только при наличии данных, отличных от первоначальных данных пользователя. При нажатии на эту кнопку данные также валидируются перед отправкой, и при наличии ошибок выходит отдельное сообщение.

### Страница профиля пользователя

Данная страница отличается от других тем, что получает данные о том, какого пользователя этот профиль, через url-путь. Таким образом данная страница является профилем не только текущего пользователя, но профилем любого другого пользователя.

На этой странице отображается общая информация о пользователе, а также 3 вкладки: посты пользователя, лайкнутые посты и посты, на которые он оставил комментарий, вкладки переключаются с помощью кнопок над ними.

Если данная страница является профилем текущего пользователя, то начинают отображаться кнопка настроек профиля, форма добавления нового поста и кнопки удаления постов.

Внешний вид страницы представлен на рисунке 2.6.

### Посты и комментарии

Посты и комментарии как в ленте постов, так и в профиле пользователя сделаны через отдельную компоненту Post. Из внешних данных она получает внешние данные о посте и его авторе, такие как дата, контент, изображение пользователя, логин, имя и лайк поста. Внутри себя она отправляет запрос на медиафайлы поста, чтобы при наличии отобразить их в посте. Часть кода компоненты представлен ниже:

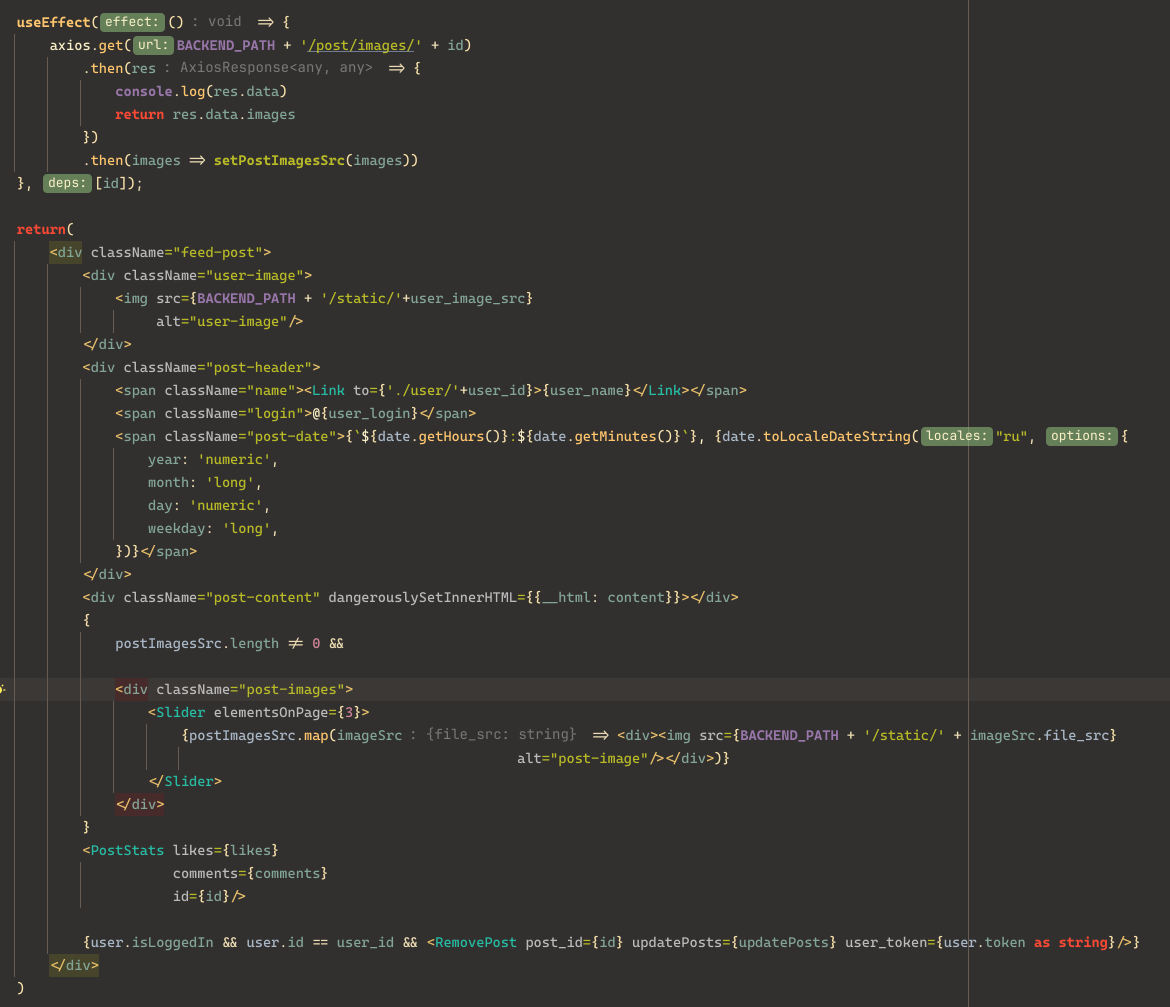


Рисунок 3.8 - код компоненты Post

Комментарии пользователя открываются поверх основного контента. Внешний вид представлен на рисунке 3.9:

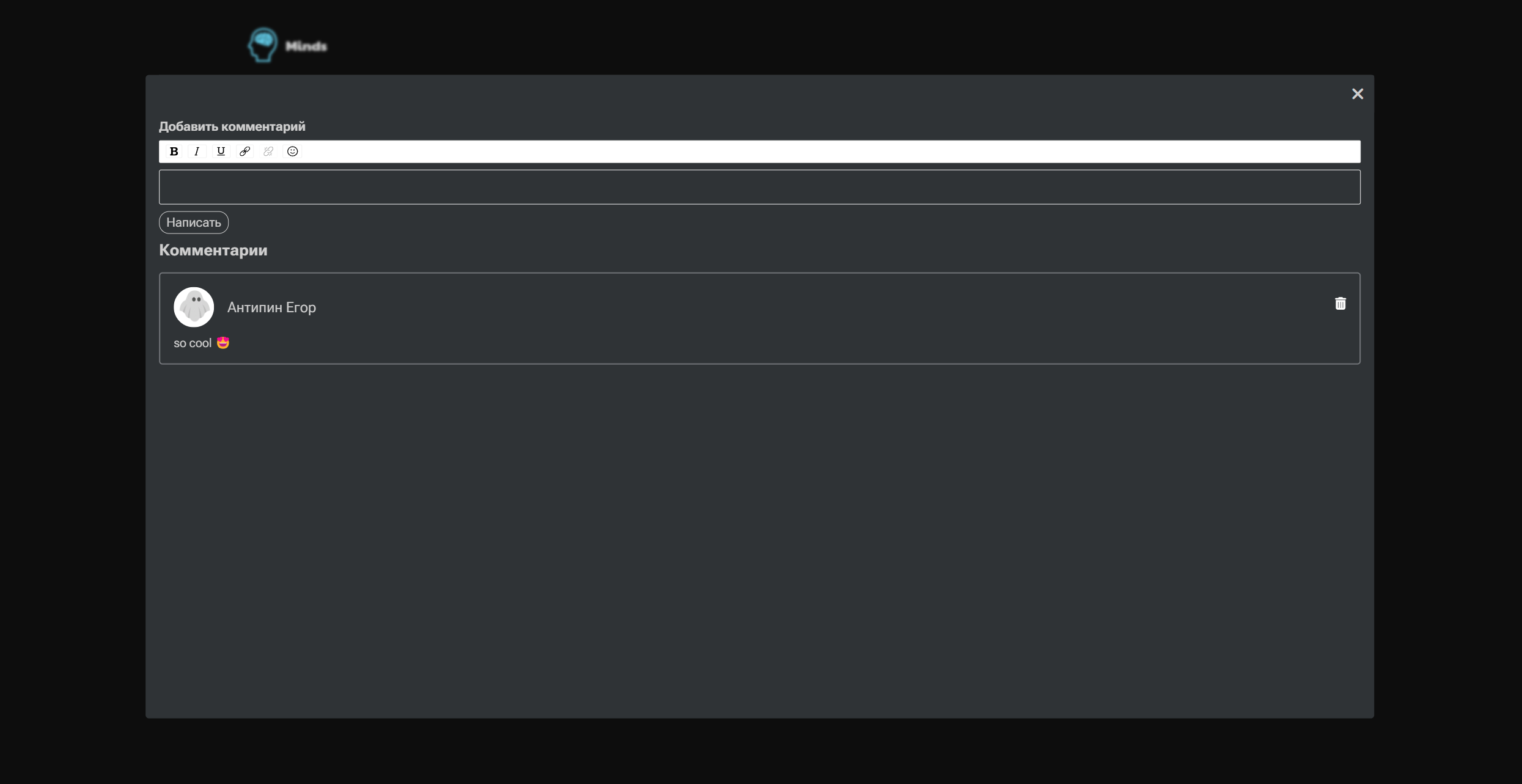


Рисунок 3.9 - внешний вид страницы комментариев

Если пользователь авторизирован, то на данной странице отображается форма добавления комментария. Ниже отображаются комментарии с именем и изображением пользователя, а также контентом комментария. Ели комментарий принадлежит конкретному пользователю, то справа в комментарии добавляется кнопка удаления.

### Страница уведомлений

Внешний вид страницы уведомлений представлен на рисунке 2.7. Она появляется только если пользователь авторизован. Загружая данные с сервера, она отображает все уведомления пользователя. Также, если есть непрочитанные уведомления, то в боковом меню сверху иконки в красном кружке отображается количество непрочитанных уведомлений. Как это выгладит показано на рисунке ниже:

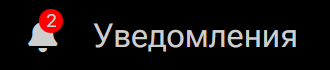


Рисунок 3.10 - внешний вид непрочитанных уведомлений

### Главная страница

Внешний вид данной страницы представлен на рисунке 2.8, код компоненты представлен ниже:



Рисунок 3.11 - код компоненты Feed

Данная страница подгружает данные о ленте пользователя и с сервера и отрисовывает их с помощью компоненты Post, которая используется на странице профиля пользователя. Также, если пользователь авторизован, отображается форма добавления нового поста.

### Форма добавления поста и комментария

Отдельно хочется выделить то, как была реализована форма добавления постов и комментариев. На изображениях выше видно, что они представляют собой одну и ту же форму, потому что являются одинаковой компонентой. Данный подход позволяет реализовать основной плюс библиотеки React - переиспользование компонент. Они реализованы с помощью отдельной компоненты – TextEditor, код которой представлен на рисунке 3.12:

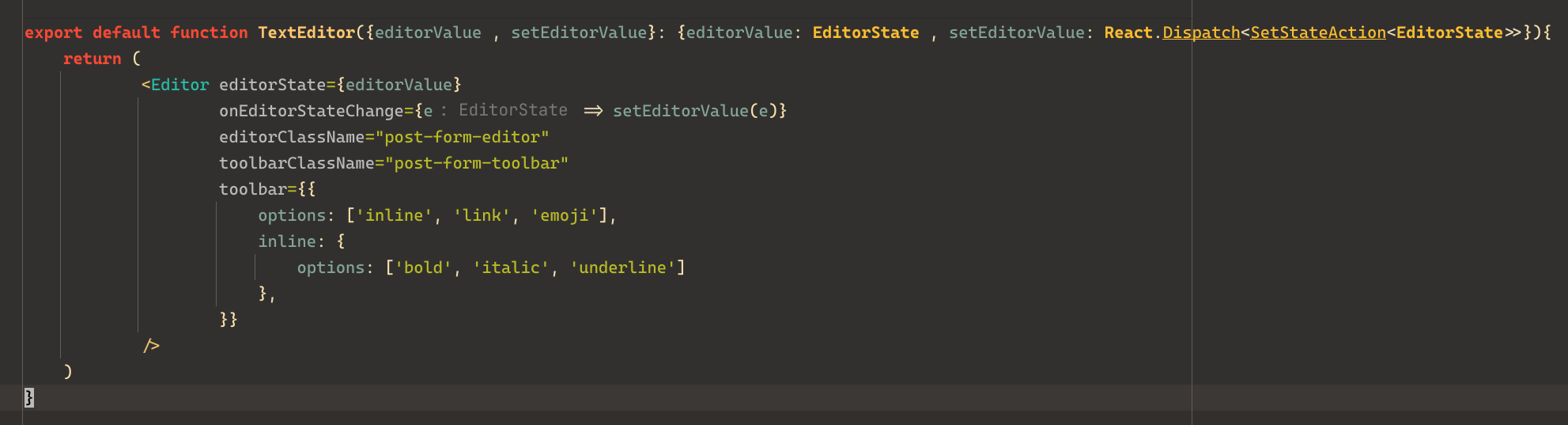


Рисунок 3.12 - код компоненты TextEditor

Данная компонента принимает извне состояние и функцию смены состояния, что позволяет получить и изменить данные редактора снаружи. Сама форма возвращает модернизированную компоненту Editor, которая предоставляется дополнительным пакетом react-draft-wysiwyg.

## Тестирование приложения

Так как данное приложение является веб-страницей, направленной на взаимодействие с пользователями, мною было принято решение использовать ручное тестирование. Для этого я попросил своих знакомых попользоваться данным приложением, предварительно его развернув на хостинге. Благодаря данному тестированию были обнаружены и исправлены следующие ошибки:

* форма добавления поста не отчищается после публикации поста;
* уведомления отправлялись с ошибкой: были поменяны местами ссылки на пользователя, который поставил лайк, и на запись;
* при попытке загрузить профиль несуществующего пользователя не выходило соответствующее сообщение;
* при попытке изменить логин или почту пользователя не было проверки на наличие существующих аккаунтов с такими данными.

При разработке приложение был получен опыт разработки на языке JavaScript с использованием библиотек React и Express, а также реализации базы данных PostgreSQL и подключения ее к клиент-серверному приложению.

Библиографический список

1. Учебный план образовательное программы «Программная инженерия» Национального исследовательского университета Высшая школа экономики, Пермский кампус [электронный ресурс] // URL: <https://perm.hse.ru/ba/se/base_curriculum>
2. Документация библиотеки React [электронный ресурс] // URL: https://react.dev/

ПРИЛОЖЕНИЕ А

## Техническое задание

Техническое задание представлено в файле «ТехническоеЗадание.docx»

## Код программы

Ссылка на код приложения на github: <https://github.com/yosheeeee/social_network>