Министерство образования Московской области

ГБПОУ МО «Колледж «Коломна»

09.02.03

КУРСОВОЙ ПРОЕКТ

ПМ. 02 Осуществление интеграции программных модулей

Тема: Разработка программного обеспечения для новостного блога

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

ККОО.КП0214.000ПЗ

|  |  |
| --- | --- |
| Студент: | Оленев О.В. |
| Руководитель: | Карташова Е.В. |
| Нормоконтролер: | Грушникова Т.Н. |
| Дата защиты: | Оценка |

2023

МИНИСТЕРСТВО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ

ГБПОУ МО «КОЛЛЕДЖ «КОЛОМНА»

ЗАДАНИЕ

на курсовой проект по ПМ 02 Осуществление интеграции программных модулей

студента очной формы обучения 4 курса группы 41911

Оленева Олега Вадимовича

Тема: Разработка программного обеспечения для новостного блога

Постановка задачи:

Разработать информационную систему в соответствии с темой и приложение для работы с ней. Разработанная информационная система должна отражать основные бизнес-процессы в соответствии с техническим заданием на разработку. Для защиты информации необходимо разграничить права пользователей системы. Разработанные формы должны обеспечить возможность пользователям осуществлять работу с данными в соответствии с их правами.

При разработке информационной системы необходимо решить следующие задачи:

- сформулировать цель проектирования базы данных;

- описать возможного пользователя базы данных;

- определить круг запросов и задач, которые предполагается решать с использованием созданной базы данных;

- построить концептуальную модель;

- сформулировать требования к базе данных;

- построить реляционную модель и выполнить её нормализацию;

- осуществить выбор СУБД и технических средств;

- создать спроектированную базу данных в среде, выбранной СУБД;

- разработать приложение для реализации запросов и решения задач;

- оценить ИС с точки зрения возможностей её дальнейшего развития.

Объем курсовой работы:

* + 1. Пояснительная записка (25-50 листов печатного текста формата А4):

Введение

* + - * 1. Разработка системного проекта
  1. Назначение разработки
  2. Требования к программному продукту
     + - 1. Обзор используемых технологий
         2. Описание системы
  3. Проектирование базы данных
  4. Разработка проекта
  5. Интерфейс программы
     + - 1. Руководство пользователя

Заключение

Список литературы

* + 1. Презентация для защиты курсового проекта (10-15 слайдов)

Дата выдачи задания \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Срок окончания \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Зав. структурным подразделением \_\_\_\_\_\_\_\_ Емельянова В. А.

Руководитель \_\_\_\_\_\_\_\_\_ Карташова Е.В.

СОДЕРЖАНИЕ

|  |  |
| --- | --- |
| Введение | 5 |
| 1 Разработка системного проекта | 6 |
| 1.1 Назначение разработки |  |
| 1.2 Требования к программному продукту |  |
| 2 Обзор используемых технологий | 8 |
| 3 Описание системы | 10 |
| 3.1 Проектирование базы данных |  |
| 3.2 Разработка проекта | 12 |
| 3.3 Интерфейс программы | 15 |
| 4 Руководство пользователя | 19 |
| Заключение | 25 |
| Список литературы | 26 |
| [Приложение](#_Toc25575737) А – Исходный код компонента «Профиль» | 27 |

Курсовое проектирование является завершающим этапом изучения междисциплинарного курса «Осуществление интеграции программных модулей», в ходе студенты узнают о методах и технологиях интеграции программных модулей, а также о том, как правильно проектировать и тестировать системы, в которых используется несколько программных модулей.

Целью курсового проекта является:

* углубления теоретических знаний в соответствии с заданной темой;
* формирования умений применять теоретические знания при решении поставленных вопросов;
* формирования умений использовать справочную, нормативную и правовую документацию;

Завершением курсового проекта является рабочее, полностью функционирующее приложение, предназначенное для публикации различной информации в интернете в виде статей.

1. РАЗРАБОТКА СИСТЕМНОГО ПРОЕКТА

1.1 Назначение разработки

Разработать информационную систему, предназначенную для публикации различной информации в интернете в виде статей.

Система разрабатывается для всех пользователей интернета. Она позволит публиковать и читать разного рода статьи, а также будет развлекательной площадкой для всех пользователей.

Компоненты системы

Система будет использоваться как веб-сайт, доступный для всех пользователей, но некоторые действия только для авторизованных в системе.

Разработка направлена на создание нового продукта, который будет предоставлять новые функциональные возможности, решать проблемы пользователей или открывать новые рынки.

1.2 Требования к программному продукту

1. Полнота и актуальность информации: Информационный продукт должен содержать полную и актуальную информацию о новостях, обзорах, интервью и т.д. Вся информация должна быть актуализирована как можно быстрее и не оставлять пользователей в неведении.
2. Удобство использования: Информационный продукт должен быть удобен в использовании и иметь простой и понятный интерфейс. Пользователь должен быстро и легко находить нужную ему информацию и иметь возможность персонализировать свой профиль и настройки под свои потребности.
3. Качество контента: Информационный продукт должен содержать качественный и интересный контент, который привлекает пользователей и держит их внимание. Это может быть достигнуто с помощью хорошо написанных статей, качественных фото- и видеоматериалов, а также других интересных и разнообразных форматов контента.
4. Быстродействие: Информационный продукт должен быть быстрым и отзывчивым, чтобы пользователи могли быстро и легко находить нужную им информацию без задержек и проблем.
5. Безопасность и конфиденциальность: Информационный продукт должен обеспечивать безопасность и конфиденциальность пользователей. Должны быть приняты меры для защиты личных данных пользователей и их финансовых средств, а также для предотвращения несанкционированного доступа к сайту.

1. ОБЗОР ИСПОЛЬЗУЕМЫХ ТЕХНОЛОГИЙ

Приложение реализовано с помощью React – это библиотека JavaScript, используемая для создания пользовательского интерфейса веб-приложений. Она позволяет разработчикам создавать масштабируемые и быстро работающие приложения с использованием компонентной архитектуры. React также поддерживает множество плагинов и инструментов, таких как Redux и React Router, что делает его более гибким и удобным в использовании.

Разработка осуществляется на языке TypeScript.

TypeScript – это язык программирования, который является надмножеством JavaScript и добавляет типизацию к коду. TypeScript делает код более читаемым, проще поддерживать и облегчает совместную работу над проектом. Он также обладает более современным синтаксисом, чем JavaScript, и включает в себя множество возможностей, таких как возможность создания интерфейсов и перечислений, что делает код более ясным и понятным.

Серверная часть написана с помощью Nest.Js и Sequelize.

Nest.Js - это фреймворк для создания серверных приложений на Node.js. Он предоставляет множество возможностей для создания масштабируемых и легко поддерживаемых приложений, включая поддержку TypeScript, модульность, инжектирование зависимостей и поддержку множества протоколов, таких как HTTP, WebSocket и TCP.

Sequelize - это ORM (Object-Relational Mapping) для Node.js, который позволяет разработчикам взаимодействовать с реляционными базами данных с помощью объектно-ориентированной модели. Sequelize поддерживает несколько баз данных, включая PostgreSQL, MySQL, SQLite и MSSQL, и обеспечивает множество возможностей, таких как миграции, синхронизация схемы базы данных, создание запросов и многие другие.

PostgreSQL был выбран языком для базы данных.

PostgreSQL - это реляционная база данных с открытым исходным кодом, которая обеспечивает надежное хранение и быстрый доступ к большим объемам данных. PostgreSQL обладает множеством возможностей, таких как поддержка транзакций, индексирование и расширяемость, что делает его очень популярным среди разработчиков.

В качестве средства управления базами данных используется PgAdmin.

Системы контроля версий Git позволяет разработчикам управлять и контролировать изменения в коде. Он позволяет разработчикам работать параллельно, управлять версиями кода, объединять ветки и контролировать доступ к коду.

Zustand - это библиотека управления состоянием для React-приложений. Она предоставляет простой способ хранения и изменения состояния вашего приложения, который может использоваться вместо использования Redux или Context API. Основная идея Zustand заключается в том, что он использует простое API для управления состоянием, что делает его более легким в использовании, чем Redux или Context API. Zustand также предлагает возможность использования механизмов подписки и обновления, что делает его более эффективным, чем обычный React-код.

1. ОПИСАНИЕ СИСТЕМЫ

3.1 Проектирование базы данных

В процессе проектирования были выявлены следующие сущности:

У каждого пользователя есть:

* Имя
* Личное описание
* Пароль
* Email
* Аватар
* Избранные посты

Каждый пост характеризуется следующими параметрами:

* Заголовок
* Содержащийся текст
* Изображение
* Имя автора
* Фото автора

Для того, чтобы хранить токены авторизации пользователей, была создана отдельная таблица, которая содержит:

* Уникальный идентификатор пользователя
* Обновляемый токен

В результате проектирования была получена следующая диаграмма «Сущность-связь» (рисунок 1)

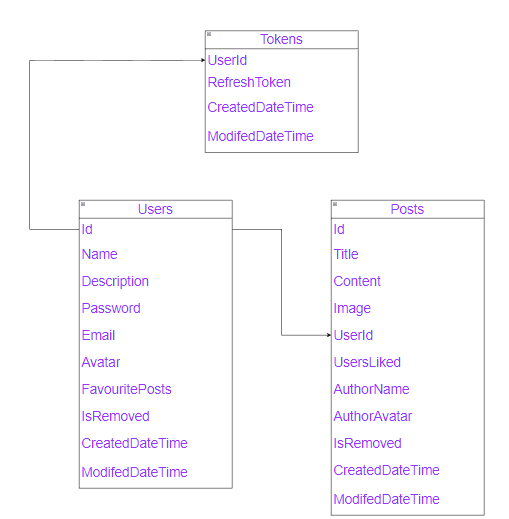


Рисунок 1 - Концептуальная модель

Таблица 1 - Пользователь

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Наименование поля | Тип | Ключи | Примечание |
| 1 | Id | uuid | PK | Not Null |
| 2 | Email | character var 255 |  | Not Null |
| 3 | Password | character var 255 |  | Not Null |
| 4 | Name | character var 255 |  |  |
| 5 | Description | text |  |  |
| 6 | Avatar | character var 255 |  |  |
| 7 | FavouritePosts | uuid[] |  | Not Null |
| 8 | IsRemoved | boolean |  | Not Null |
| 9 | CreatedAt | timestamp |  | Not Null |
| 10 | UpdatedAt | timestamp |  |  |

Таблица 2 - Посты

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Наименование поля | Тип | Ключи | Примечание |
| 1 | Id | uuid | PK | Not Null |
| 2 | Title | character var 255 |  |  |
| 3 | Content | text |  | Not Null |
| 4 | Image | character var 255 |  |  |
| 5 | UserId | uuid | FK | Not Null |
| 6 | UsersLiked | uuid[] |  | Not Null |
| 7 | AuthorName | character var 255 |  | Not Null |
| 8 | AuthorAvatar | character var 255 |  |  |
| 9 | IsRemoved | boolean |  | Not Null |
| 10 | CreatedAt | timestamp |  | Not Null |
| 11 | UpdatedAt | timestamp |  |  |

Таблица 3 - Токены

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Наименование поля | Тип | Ключи | Примечание |
| 1 | UserId | uuid | FK | Not Null |
| 2 | RefreshToken | text |  | Not Null |
| 3 | CreatedAt | timestamp |  | Not Null |
| 4 | UpdatedAt | timestamp |  |  |

3.2 Разработка проекта

В системе существуют 2 типа пользователя: авторизованный и неавторизованный. Необходимо предусмотреть соответствующие уровни доступа для каждого из них.

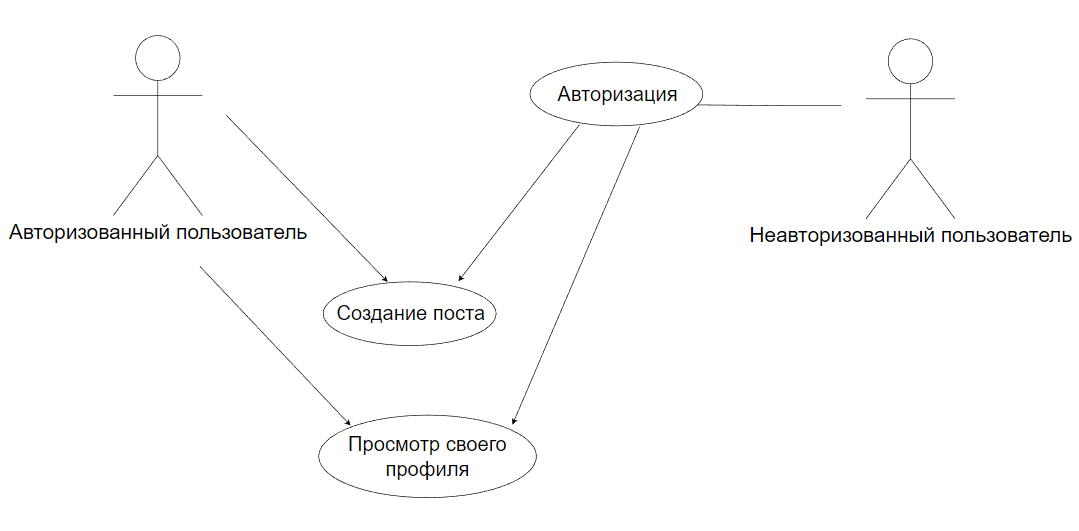


Рисунок 2 - Диаграмма использования

В приложении предусмотрено 9 форм, которые описаны в таблице 4

Таблица 4 - Формы приложения

|  |  |  |
| --- | --- | --- |
| Название | Основное назначение | Функционал |
| Модальное окно авторизации | Авторизация | Если email и пароль введены правильно, откроется основная страница сайта |
| Модальное окно регистрации | Регистрация | Если имя, email и пароль валидны и такого email нет в базе, то откроется основная страница сайта |
| Настройки профиля | Изменение информации о пользователе | Можно изменить информацию о своем профиле, если нужно |
| Мои посты | История постов | Просмотр всех постов, который публиковал пользователь |
| Создание постов | Создание статей и публикация новостей | Авторизованный пользователь может опубликовать любую новость с картинкой |
| Тренды | Отображение самых популярных, на данный момент, постов | Просмотр популярных статей |
| Свежее | Отображение самых последних по времени постов | Просмотр свежих статей |
| Избранное | Отображение избранных постов, которые добавил пользователь | Просмотр избранных статей пользователя |
| Страница поста | Полный просмотр содержимого | Просмотр всего содержимого конкретной статьи |

Для работы с данными использовалась технология Sequelize.

3.3 Интерфейс программы

При разработке приложения были созданы следующие формы, представленные на рисунках с 3 по 9.

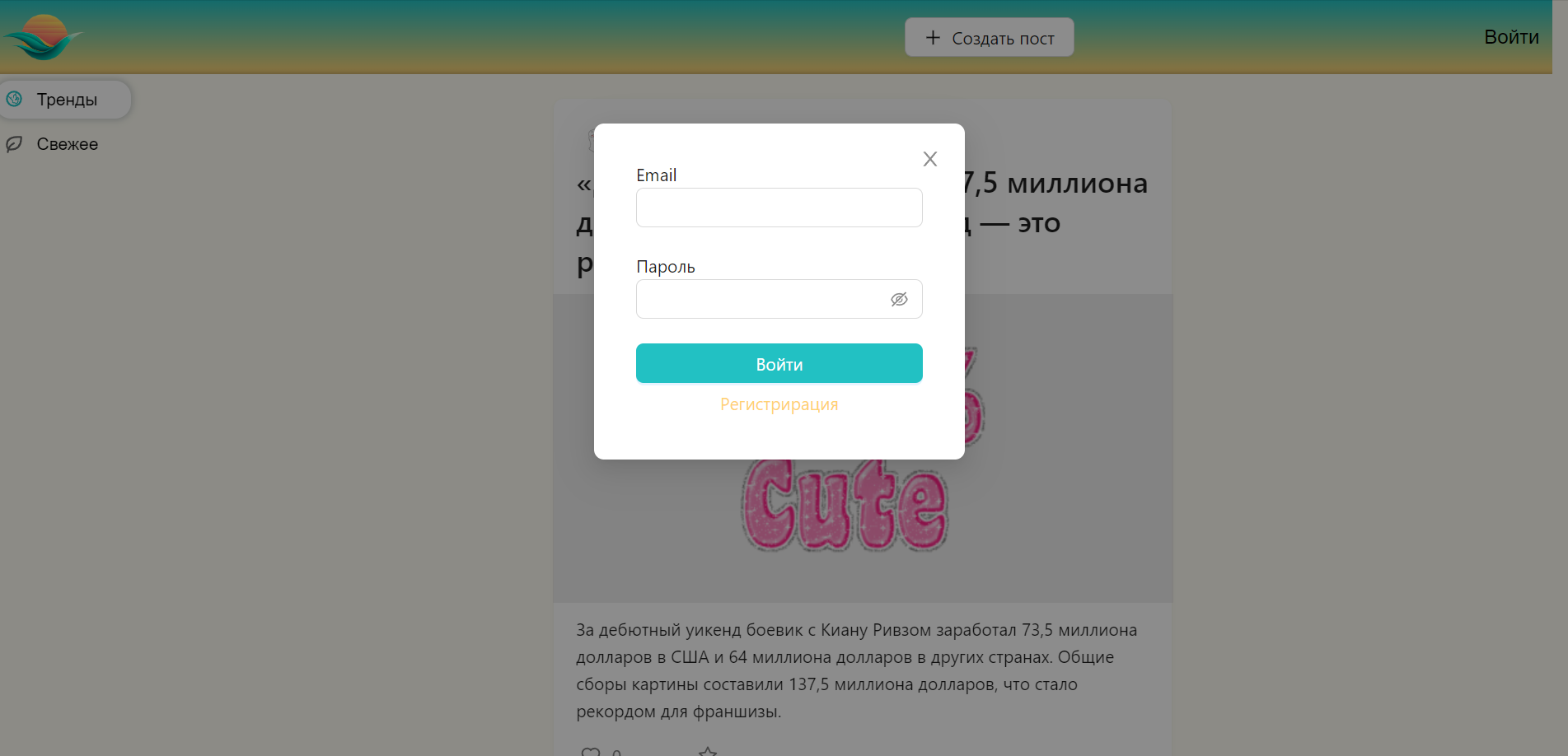


Рисунок 3 – Форма входа

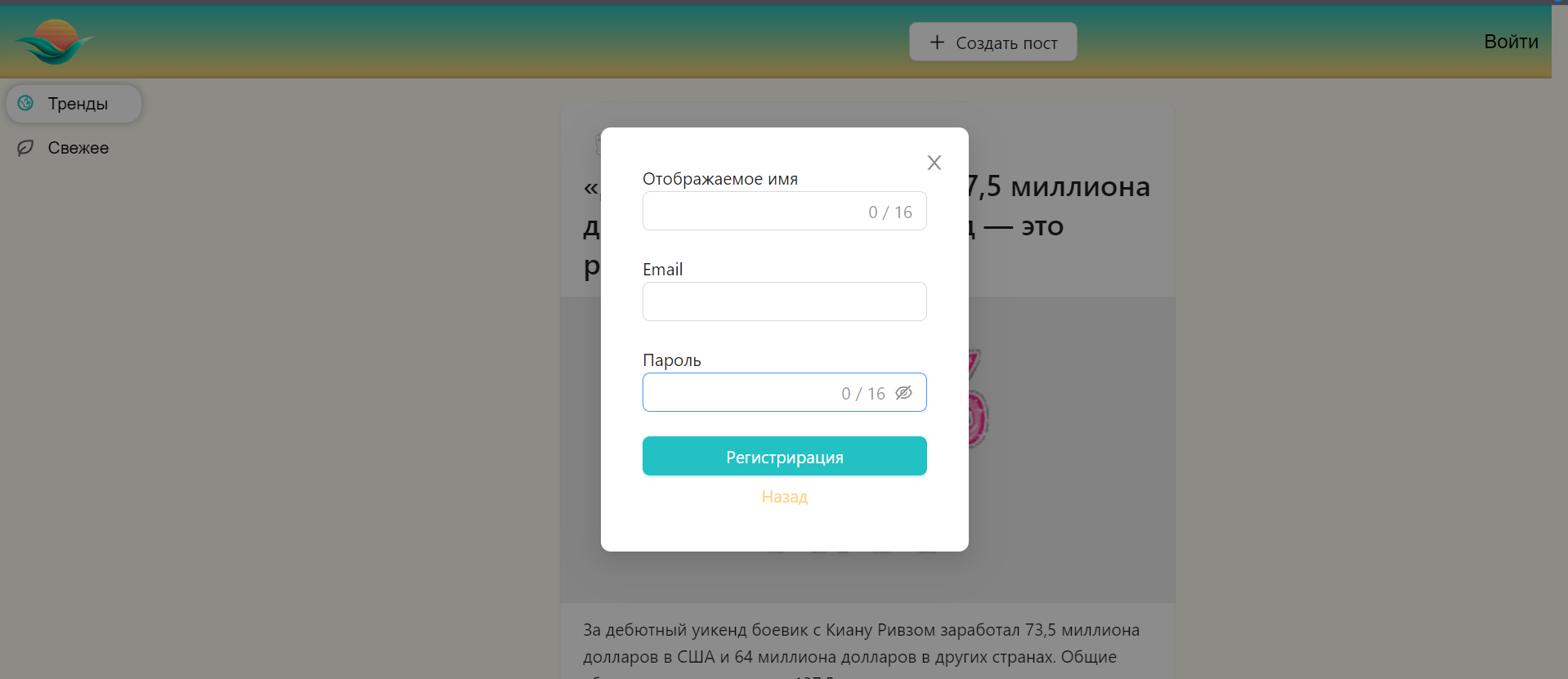


Рисунок 4 – Форма регистрации

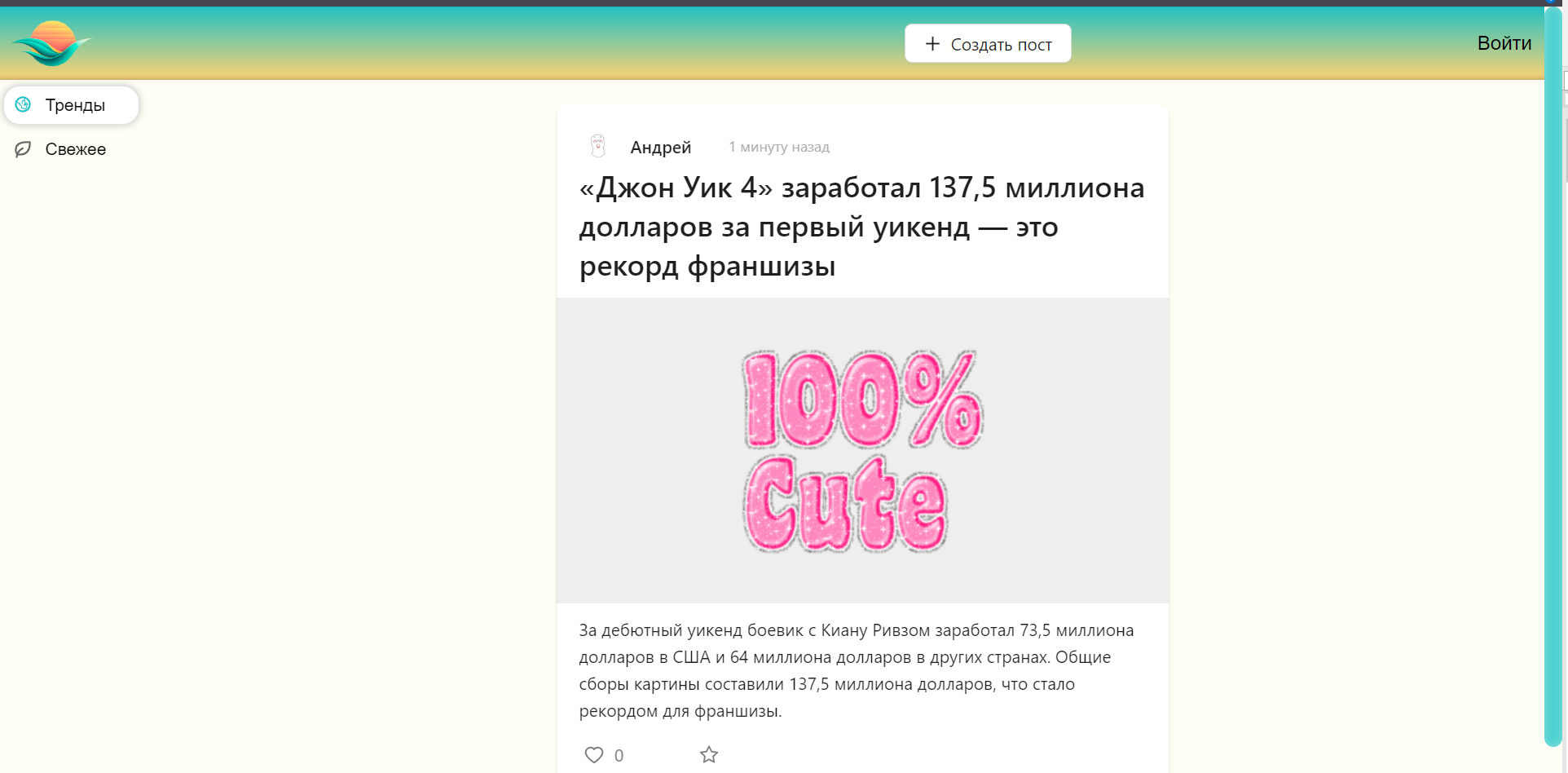


Рисунок 5 – Основная страница с новостями

Страницы «Тренды», «Свежее», «Избранное» ничем не отличаются, кроме сортировки постов по условию.

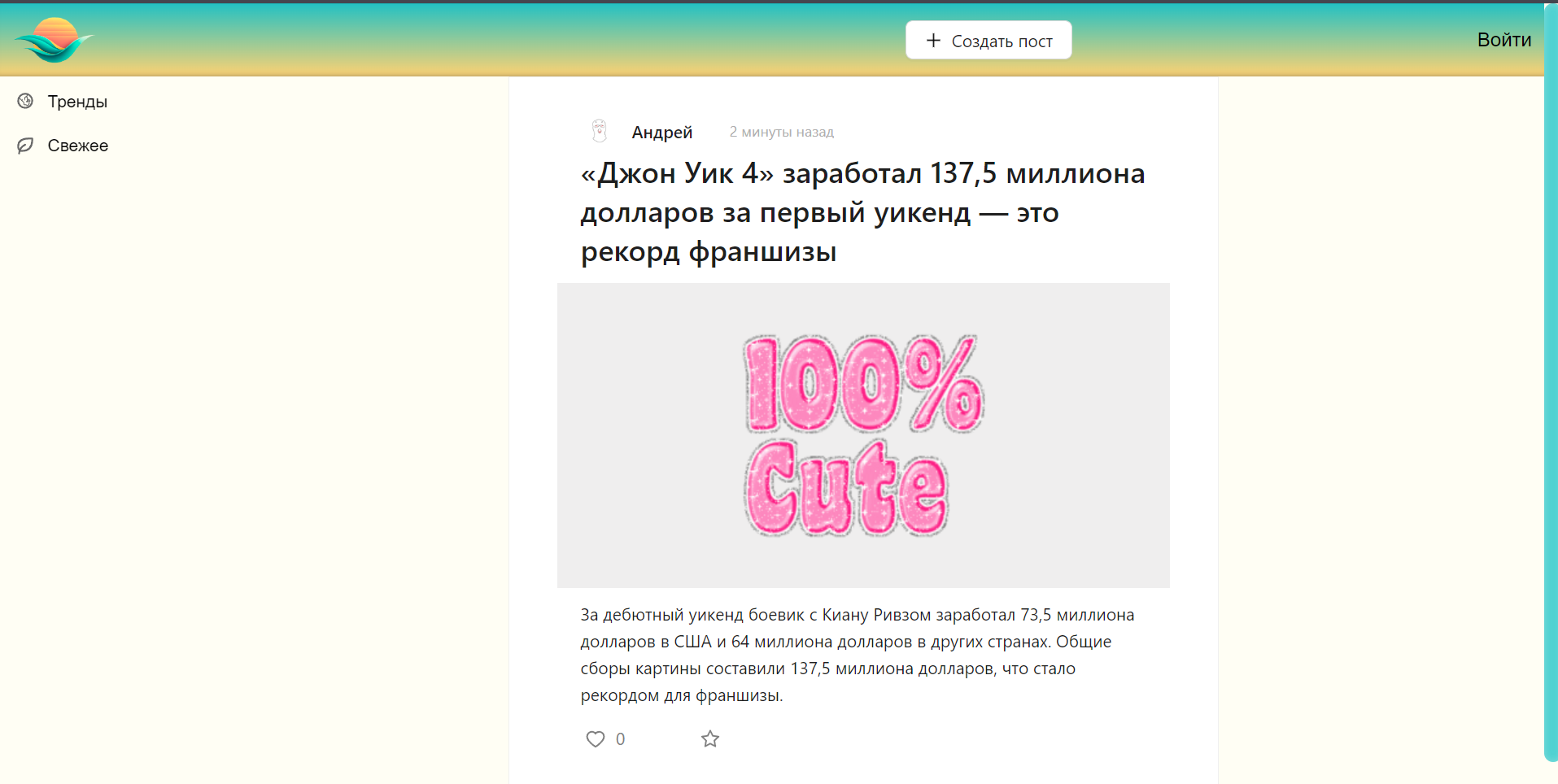


Рисунок 6 – Страница конкретного поста

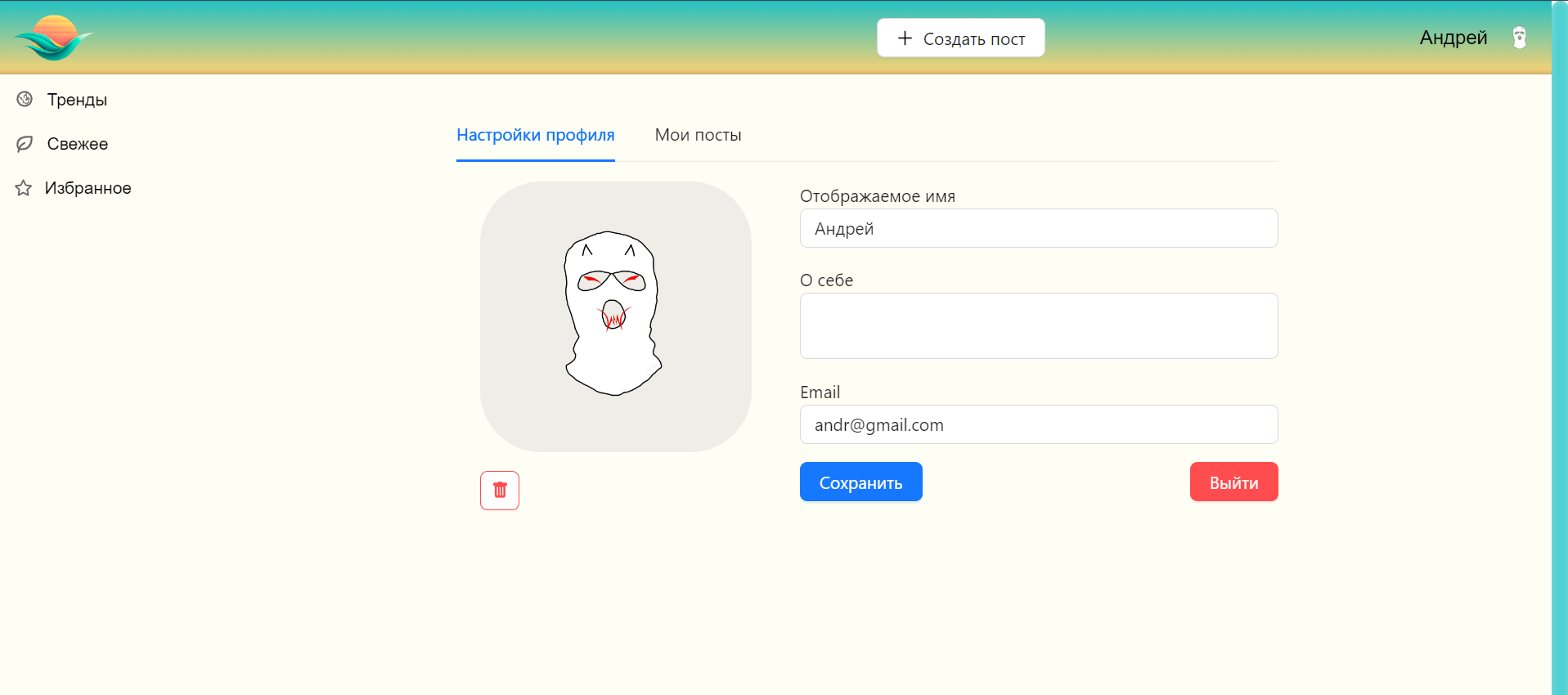


Рисунок 7 – Настройки пользователя

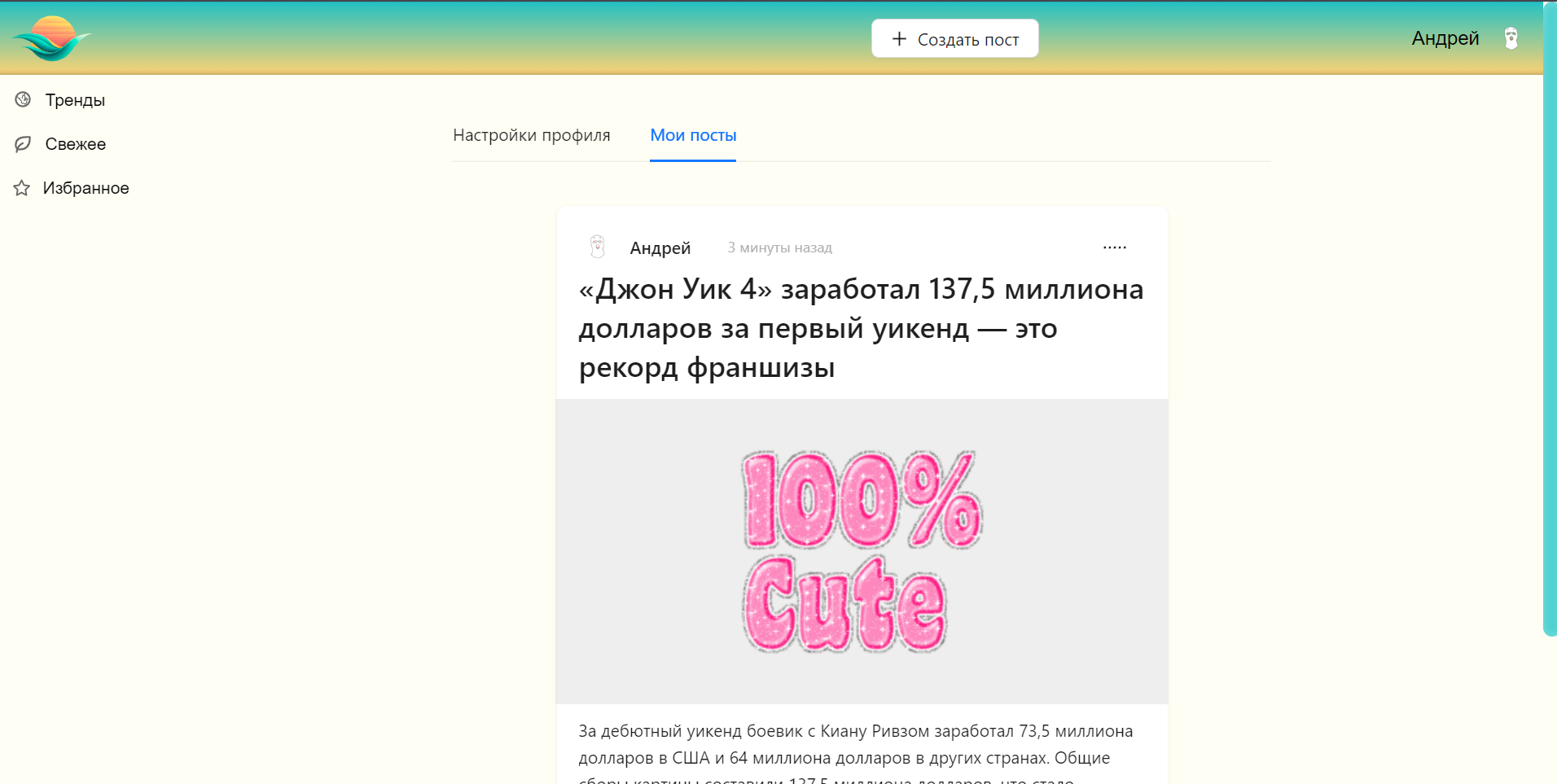


Рисунок 8 – История постов

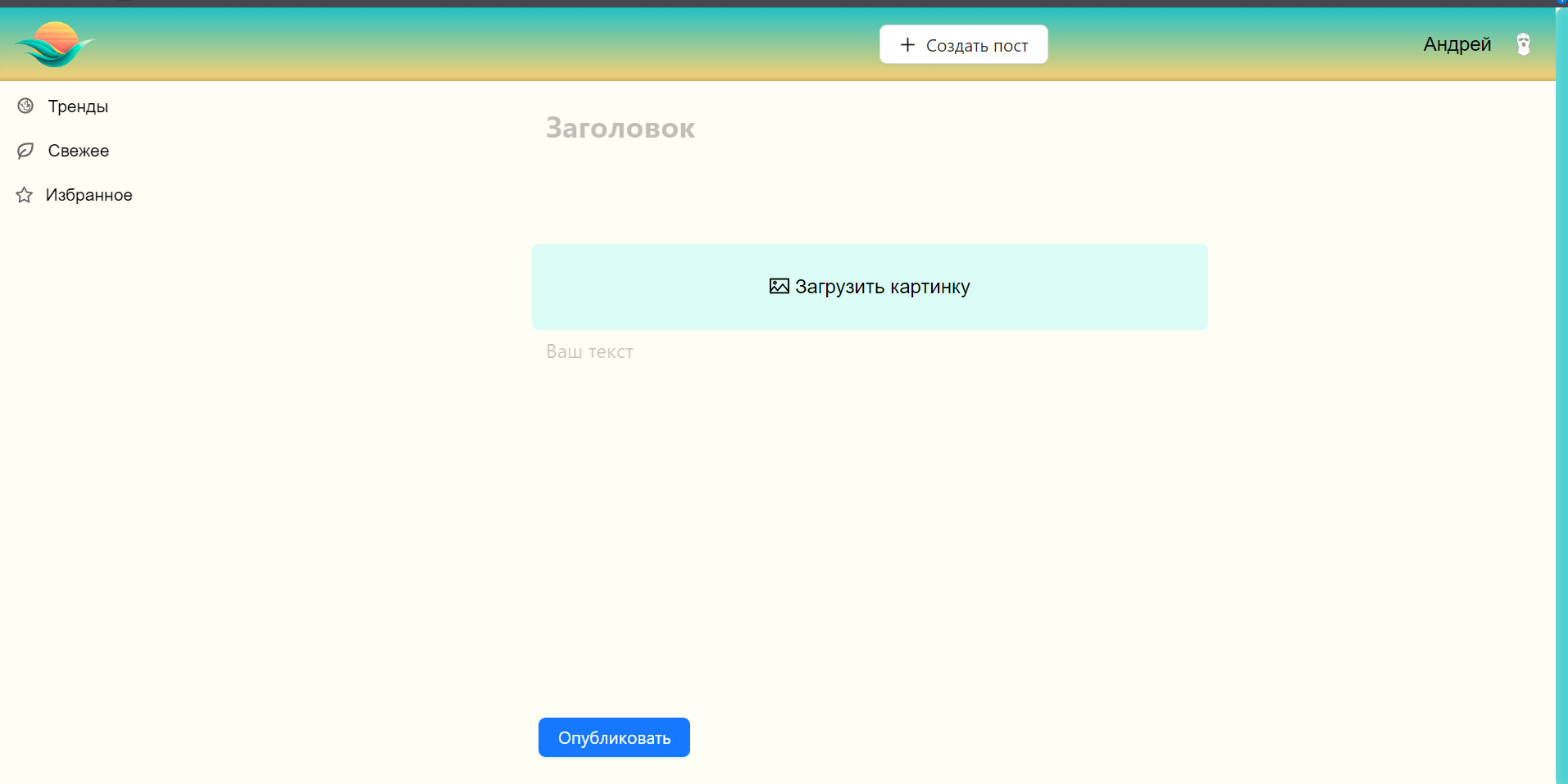


Рисунок 9 – Создание поста

3.3.1 Обработка данных

Всего в части фронтенд есть 2 основные модели: User и Post. Они содержат все поля, которые требуется в этой части проекта. Также для каждой модели существуют контроллеры для обращения к API. Контроллер модели Post:

createPost – создание поста;

getLimitPostsByTime – получение постов, отсортированных по дате;

getLimitPostsByLikes – получение постов, отсортированных по количеству лайков;

getOne – получение одного поста по его id;

getByUserId – получение поста по полю userId;

like – добавление лайка к посту;

getFavoritePosts – получение постов по массиву id;

editPost – редактирование поста;

deletePost – логическое удаление поста из базы;

Контроллер пользователя:

save – сохранение пользователя;

setFavouritePost – добавление поста в избранное;

Пример фронтенда приведен в Приложении А, в котором написана логика страницы «Профиль», где пользователь может просматривать свои посты и редактировать свои данные.

1. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Чтобы начать пользоваться всем, что есть в приложении необходимо зарегистрироваться или авторизоваться, если уже есть аккаунт (рисунок 10).

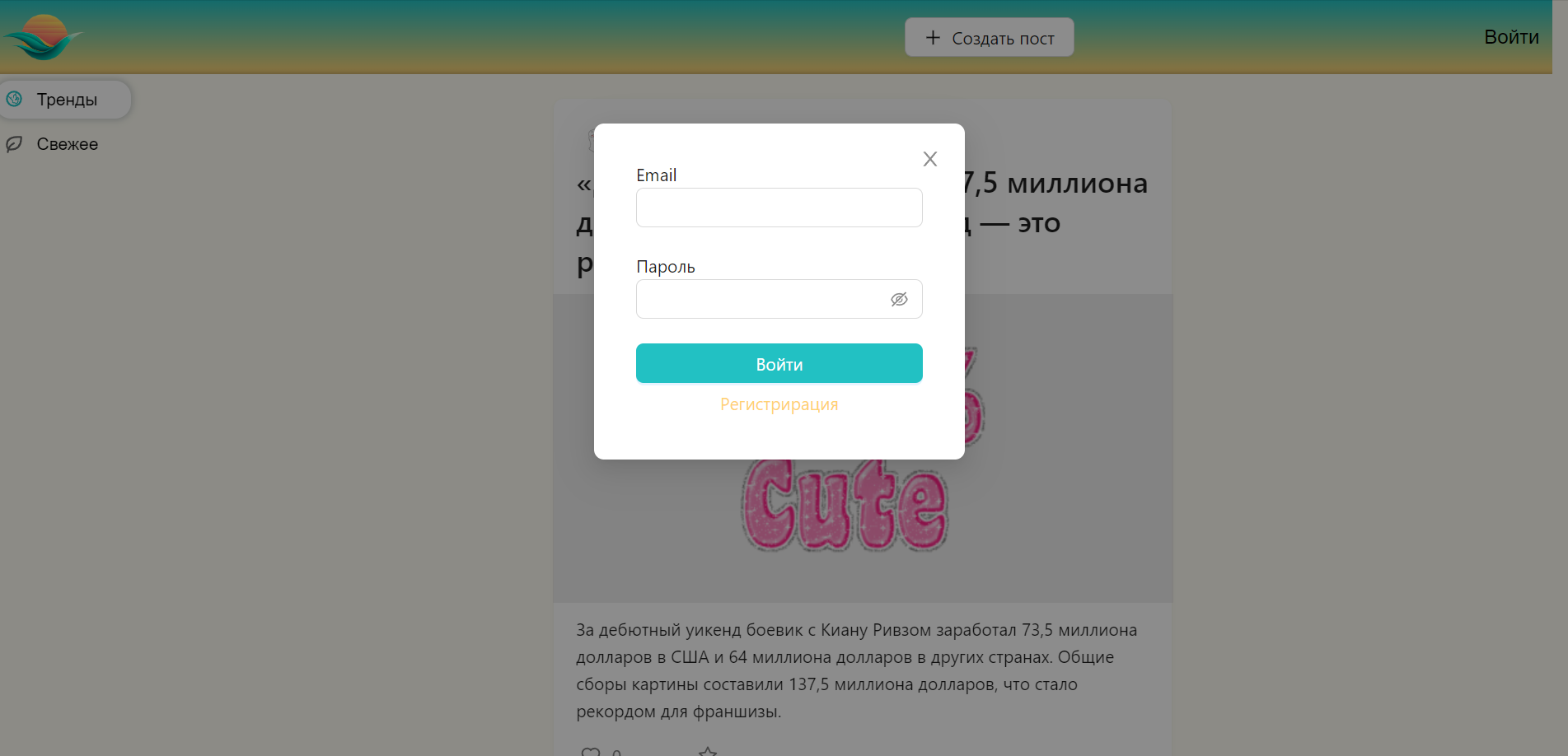


Рисунок 10 – Страница для входа

Чтобы зарегистрироваться, нужно нажать кнопку «Регистрация». После этого появятся другие поля для заполнения. После их успешного заполнения, пользователь будет зарегистрирован в системе (рисунок 11).

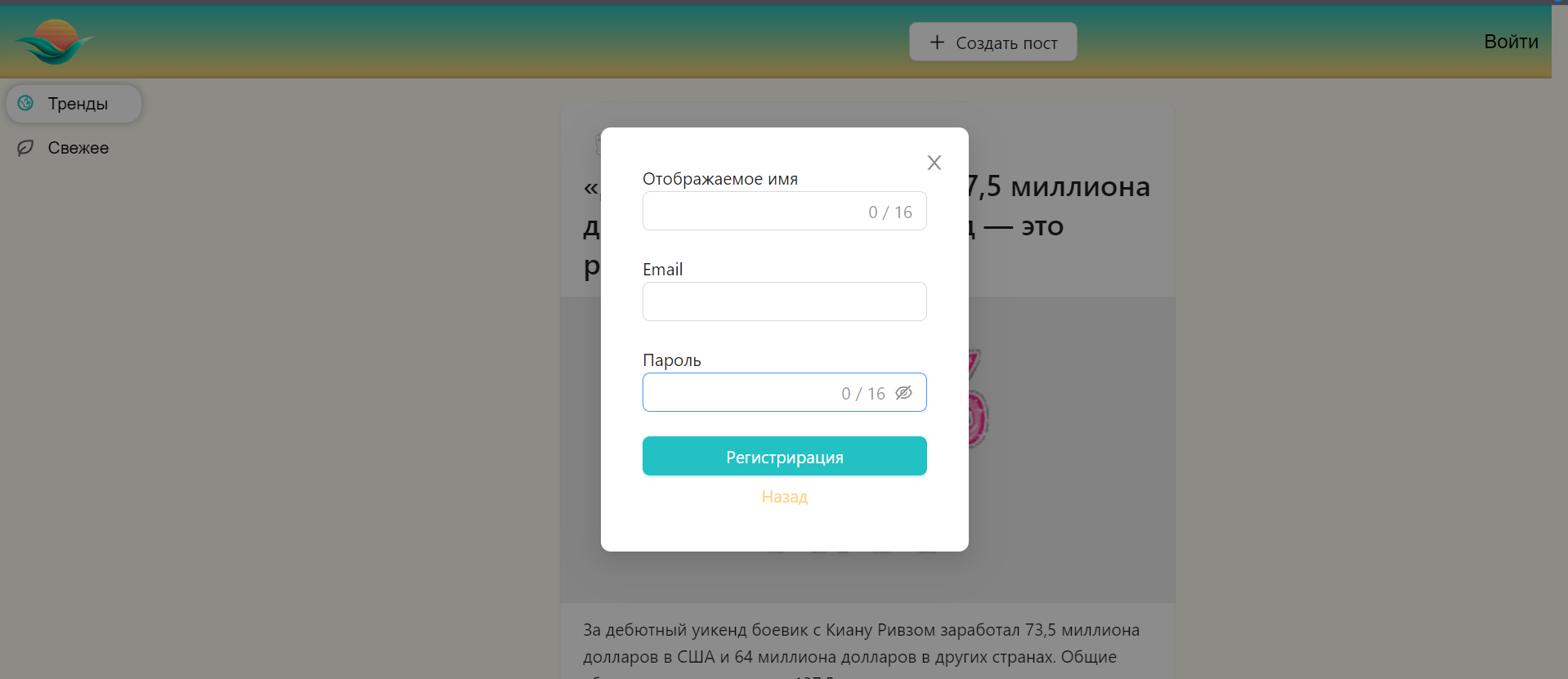


Рисунок 11 – Регистрация

После того, как пользователь авторизовался в системе, у него появляется возможность создавать посты, редактировать свой профиль и просматривать свои ранее опубликованные статьи.

Чтобы создать пост, необходимо нажать кнопку «Создать пост» в верхней части (рисунок 12).

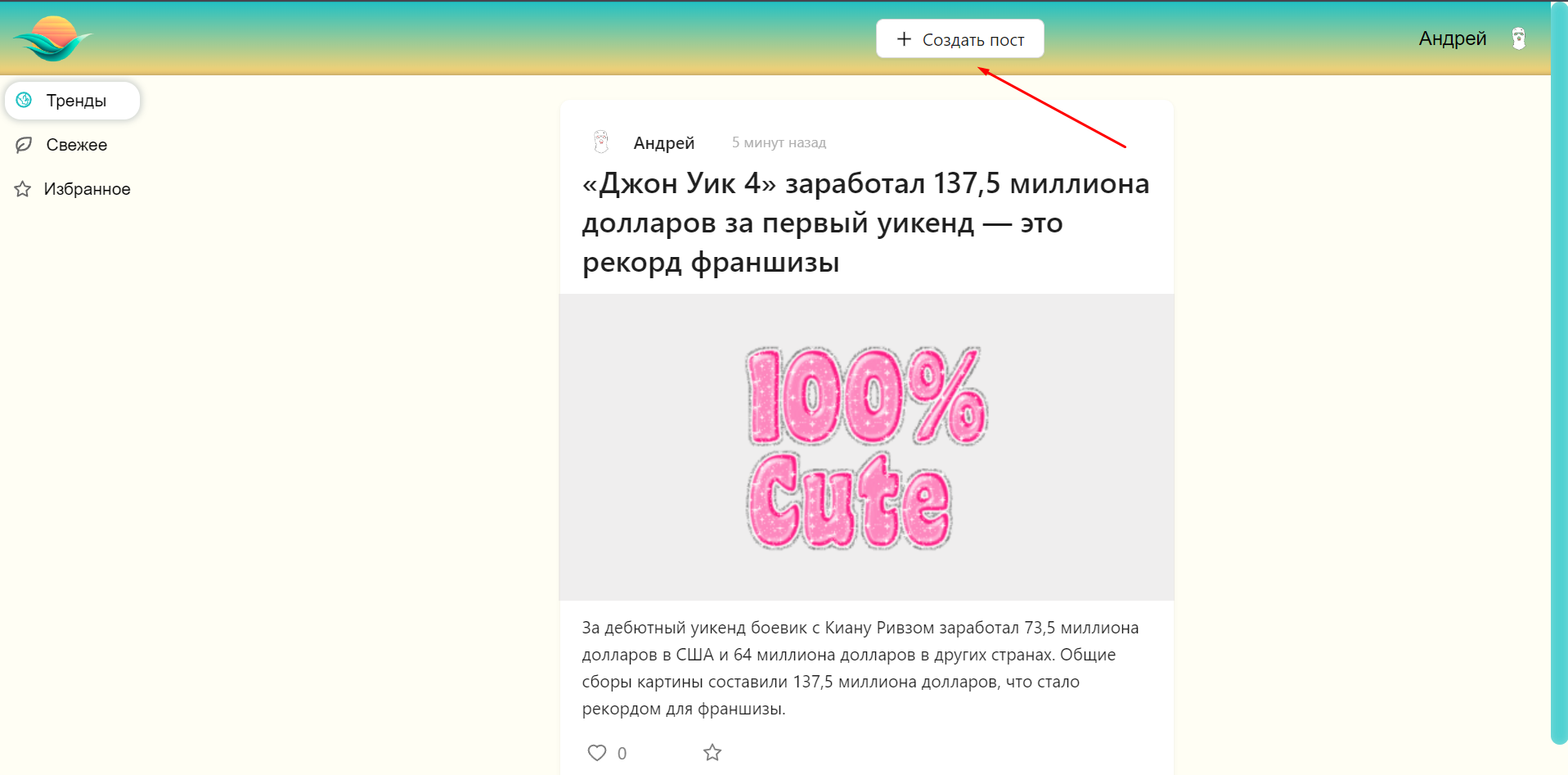


Рисунок 12 – Создание поста

После этого пользователь попадет на страницу для создания поста. Здесь он может добавить заголовок, написать интересную новость и добавить картинку. Для поста важно иметь новость, заголовок и картинка не обязательны для успешной публикации (рисунок 13).

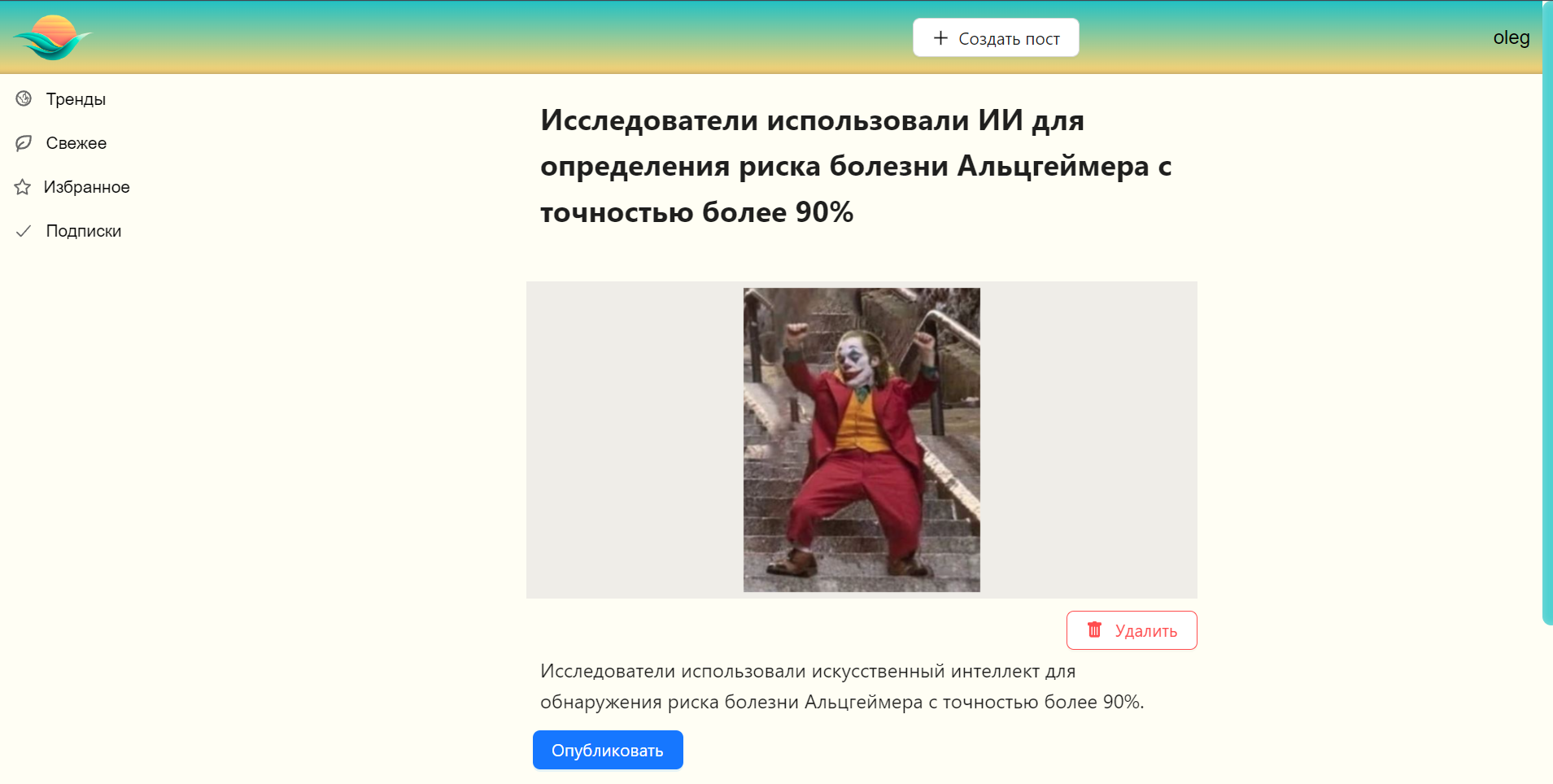


Рисунок 13 – Создание поста

Теперь необходимо нажать кнопку «Опубликовать», чтобы эта новость отобразилась в общей ленте новостей (рисунок 14).

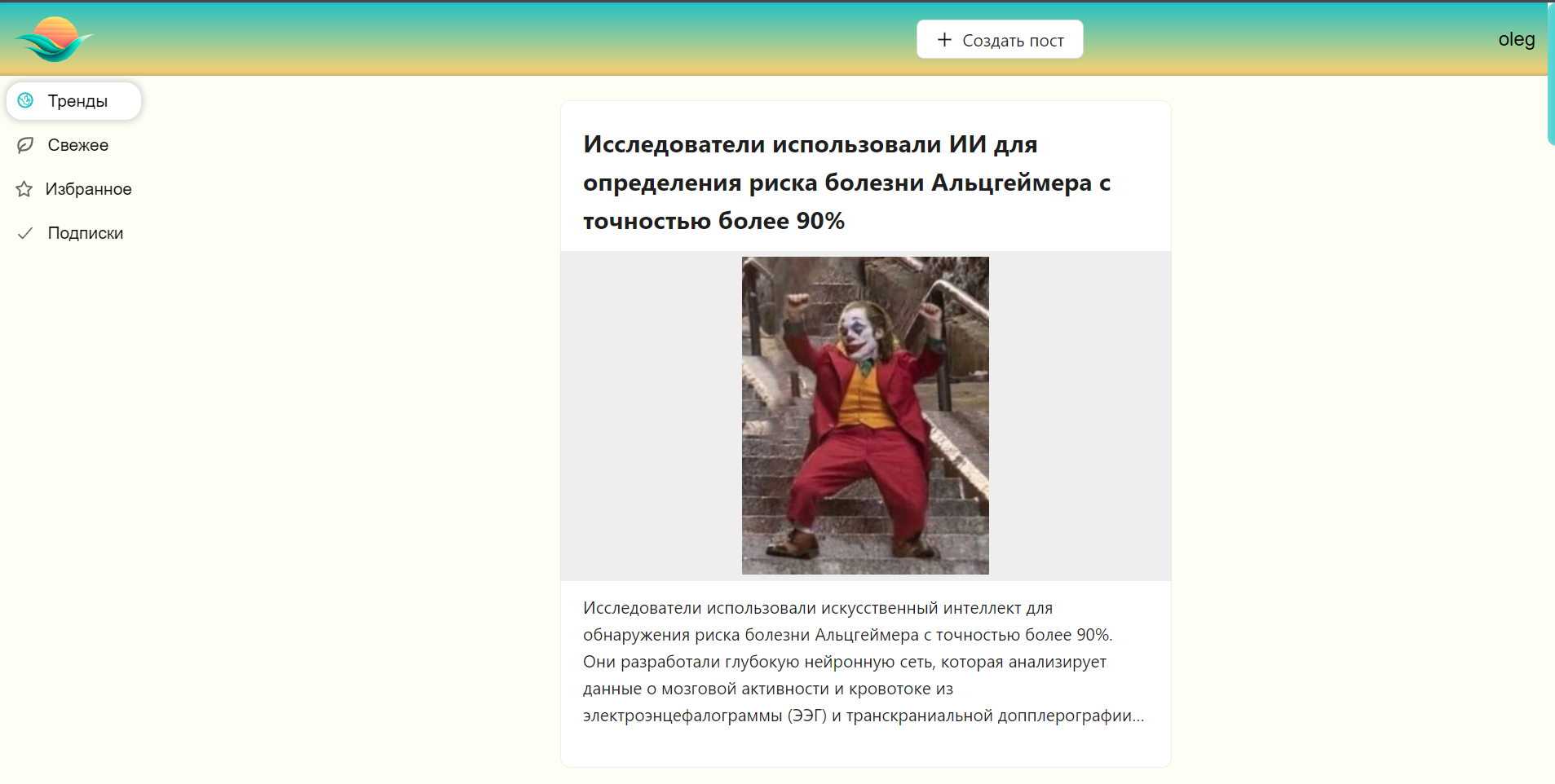


Рисунок 14 – Результат создания статьи

На данном примере видно, что текст выведен не полностью. Для того, чтобы прочитать весь текст, необходимо нажать на пост. После нажатия пользователь перейдет на страницу поста, где сможет прочитать всю информацию.

На странице своего профиля каждый пользователь может изменить свое отображаемое имя, почту и информацию о себе. Для этого достаточно изменить данные и нажать кнопку «Сохранить» (рисунок 15).

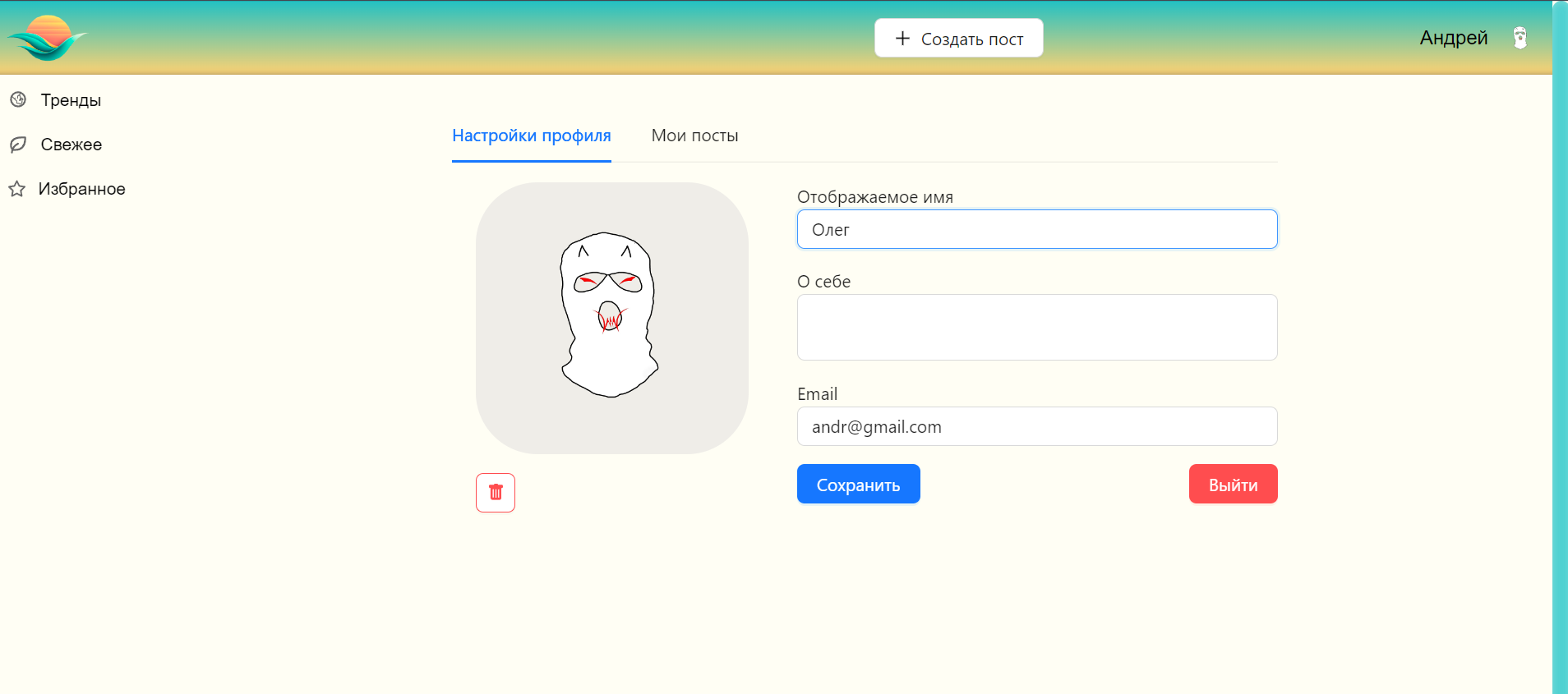


Рисунок 15 – Сохранение изменений профиля

Также на это странице можно посмотреть все свои опубликованные посты. Для этого достаточно переключить вкладку сверху на «Мои посты» (рисунок 16).

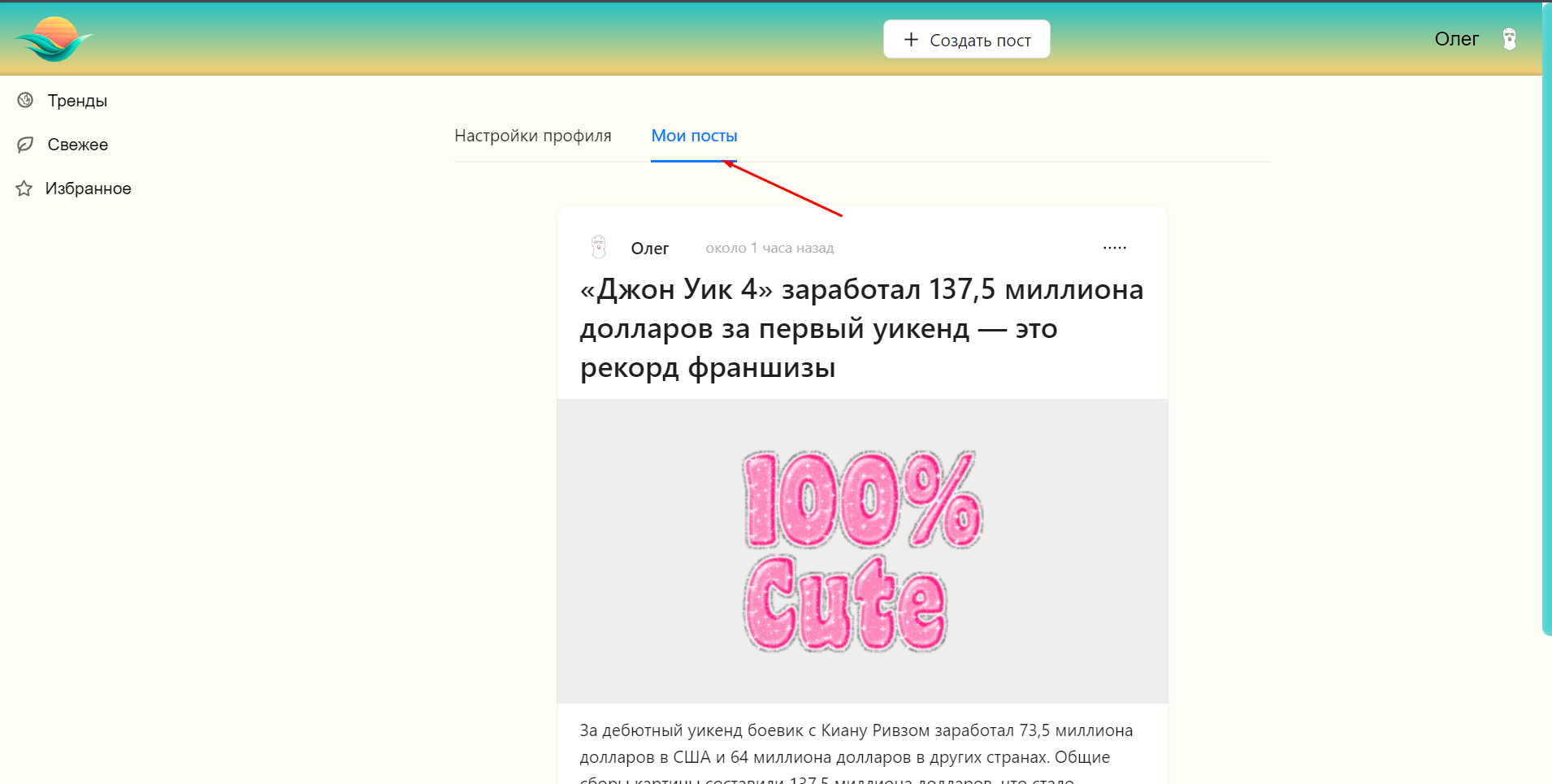


Рисунок 16 – История постов

Для того чтобы удалить или отредактировать созданный пост нужно перейти на страницу «Профиль», нажать «Мои посты» и в карточке каждого созданного поста будет кнопка с выпадающим списком. В ней можно выбрать удаление поста или его редактирование. При выборе «Удалить» откроется модальное окно с подтверждением действия. При выборе «Редактировать» откроется страница «Создание постов», где уже будут заполнены все поля (рисунок 17).

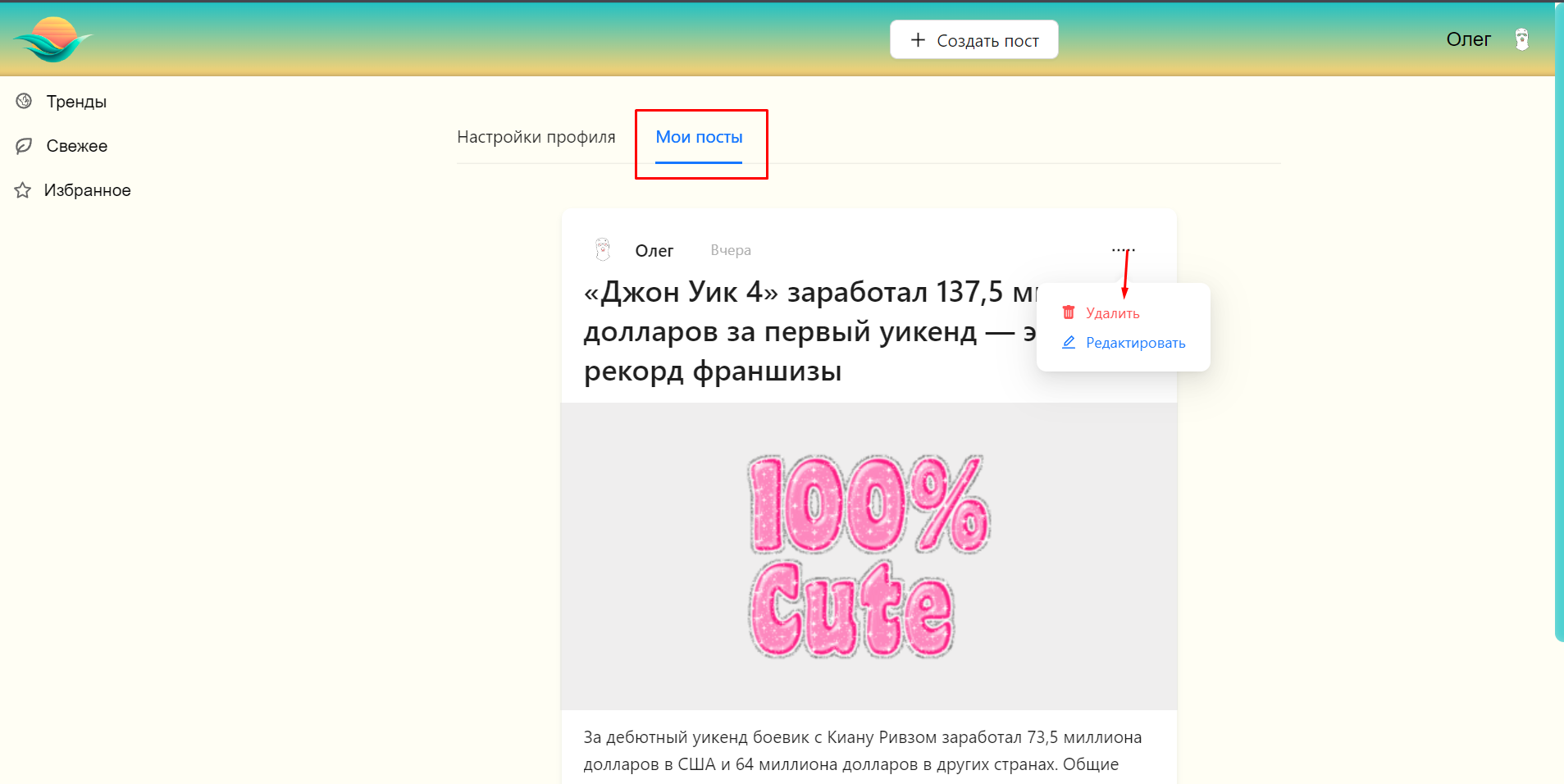


Рисунок 17 – Редактирование/удаление поста

Также можно ставить лайки или добавлять в избранное посты. Для этого достаточно нажать на иконку звезды или сердечка. После добавления поста в «Избранное» их можно найти на странице «Избранное» (рисунок 18).



Рисунок 18 – Лайки и избранное

ЗАКЛЮЧЕНИЕ

Для успешной интеграции программных модулей необходимо проводить тестирование и отладку, а также документировать процесс интеграции и взаимодействия модулей.

В процессе выполнения курсового проекта были изучены основные подходы к интеграции программных модулей, проведено их анализ и сравнение. Были рассмотрены примеры интеграции модулей на различных языках программирования.

В целом, выполнение данного курсового проекта позволило расширить знания и навыки в области интеграции программных модулей и предоставило возможность применить полученные знания на практике.

В результате создано работоспособное обеспечение для новостного блога. Проведенное тестирование не выявило существенных ошибок.

СПИСОК ЛИТЕРАТУРЫ

* Nest.js: A Progressive Node.js Framework, Джей Белл, Грег Маголан, Патрик Хосли (2018).
* Технология разработки программных продуктов. Практикум: учеб. пособие для студ. учреждений сред. проф. образования / А. В.Рудаков, Г. Н. Федорова. — 4-е изд., стер. — М. : Издательский центр «Академия»; 2014. — 192 с.
* «Разработка на JavaScript. Построение кроссплатформенных приложений с помощью GraphQL, React, React Native и Electron», Адам Д. Скотт, 2021.
* «Секреты JavaScript ниндзя», Джон Резиг, Биер Бибо, 2018.
* «Эффективный TypeScript: 62 способа улучшить код», Вандеркам Дэн, 2020

Приложение А – Исходный код компонента «Профиль»

export function Profile() {

    const user = useUserStore(state => state.user)

    const ref = useRef<HTMLInputElement>(null);

    const [messageApi, contextHolder] = message.useMessage();

    const [isAuth, setUser, logout, checkAuth] = useUserStore(state => [state.isAuth, state.setUser, state.logout, state.checkAuth])

    const [name, setName] = useState<string>('')

    const [description, setDescription] = useState<string>('')

    const [email, setEmail] = useState<string>('')

    const [posts, setPosts] = useState<IPost[]>([])

    const [avatar, setAvatar] = useState<File | null>(null)

    const [imageSrc, setImageSrc] = useState<string>('');

    const [isAuthModalOpen, setIsAuthModalOpen] = useState(false);

    const [isLoading, setIsLoading] = useState<boolean>(false)

    useEffect(() => {

        if (user != null) {

            setName(user.name)

            setDescription(user.description)

            setEmail(user.email)

            setImageSrc(user.avatar ? user.avatar : "")

        }

    }, [user])

    async function saveUser() {

        if (user != null)

            try {

                const formData = new FormData()

                formData.append('id', user.id)

                formData.append('name', name)

                formData.append('description', description)

                formData.append('email', email)

                formData.append('avatarSrc', imageSrc)

                if (avatar != null)

                    formData.append('avatar', avatar)

                const response = await UserProvider.save(formData)

                setUser(response.data)

                checkAuth()

                setAvatar(null)

                messageApi.open({

                    type: 'success',

                    content: 'Успешно сохранено',

                    duration: 3

                })

            } catch (e: any) {

                messageApi.open({

                    type: 'error',

                    content: e.response.data.message,

                    duration: 3

                })

            }

    }

    async function changeTabs(key: string) {

        if (key == 'posts' && posts.length == 0)

            getPostPaged()

    }

    async function getPostPaged() {

        if (user != null) {

            setIsLoading(true)

            const postPaged = await PostProvider.getPostsByUserId(user.id)

            setPosts(postPaged.data.rows)

            setIsLoading(false)

        }

    }

    function uploadPicture(e: ChangeEvent<HTMLInputElement>) {

        if (!e.target.files?.length) return;

        if (!['jpg', 'jpeg', 'png'].some(a => e.target.files![0].type.includes(a)))

            return messageApi.open({

                type: 'error',

                content: "Неверный тип файла",

                duration: 3

            })

        const file = e.target.files[0]

        setAvatar(file)

        const src = URL.createObjectURL(file);

        setImageSrc(src)

    }

    function deleteImage() {

        setAvatar(null)

        setImageSrc("");

        if (!ref.current) return;

        ref.current.value = "";

    }

    function changeName(event: React.ChangeEvent<HTMLInputElement>) { setName(event.target.value) }

    function changeDescription(event: React.ChangeEvent<HTMLTextAreaElement>) { setDescription(event.target.value) }

    function changeEmail(event: React.ChangeEvent<HTMLInputElement>) { setEmail(event.target.value) }

    function showAuthModal() { setIsAuthModalOpen(true) }

    function handleOk() { setIsAuthModalOpen(false) }

    const tabItems: TabsProps['items'] = [

        {

            key: 'profile',

            label: 'Настройки профиля',

            children:

                <div className={styles.settings}>

                    <div className={styles.avatar}>

                        {

                            !imageSrc &&

                            <div className={styles.imageInput}>

                                <label htmlFor="file-upload">

                                    <p><PictureOutlined /> Загрузить картинку</p>

                                </label>

                                <input id="file-upload" style={{ display: 'none' }} type="file" accept='.jpg, .jpeg, .png' onChange={e => uploadPicture(e)} ref={ref} />

                            </div>

                        }

                        {

                            imageSrc &&

                            <div className={styles.imageContainer}>

                                <div className={styles.picture}>

                                    <Image src={avatar != null ? imageSrc : process.env.REACT\_APP\_API\_URL + imageSrc} preview={false} />

                                </div>

                                <Button className={styles.deleteButton} icon={<BsFillTrashFill />} danger type='default' onClick={deleteImage} />

                            </div>

                        }

                    </div>

                    <div className={styles.textInputs}>

                        <div>

                            <label>Отображаемое имя</label>

                            <Input type="text" maxLength={25} value={name} onChange={changeName}></Input>

                        </div>

                        <div>

                            <label>О себе</label>

                            <TextArea rows={2} maxLength={150} value={description} onChange={changeDescription} style={{ resize: 'none' }}></TextArea>

                        </div>

                        <div>

                            <label>Email</label>

                            <Input type="text" value={email} onChange={changeEmail}></Input>

                        </div>

                        <div className={styles.buttons}>

                            <Button type="primary" onClick={saveUser}>Сохранить</Button>

                            {

                                isAuth &&

                                <Button onClick={logout} danger type="primary">Выйти</Button>

                            }

                        </div>

                    </div>

                </div>

        },

        {

            key: 'posts',

            label: 'Мои посты',

            children:

                <>

                    {

                        isLoading

                            ? <></>

                            : posts.length != 0

                                ? <div className={styles.feed}>

                                    {

                                        posts.map((p, index) => (

                                            <div className={styles.post}>

                                                <PostCard post={p} needLink previewImage={false} key={index} isEditable refreshPage={getPostPaged} hoverable elipsis />

                                            </div>

                                        ))

                                    }

                                </div>

                                : <EmptyPosts message="У вас еще нет постов. Но вы можете их создать!" />

                    }

                </>

        },

    ]

    return (

        <div className={styles.profileContent}>

            {contextHolder}

            {

                user == null

                    ? <>

                        <Button type="link" onClick={showAuthModal} size="large" className={styles.modalButton}>

                            Авторизируйтесь чтобы попасть на свой аккаунт

                        </Button>

                    </>

                    :

                    <Tabs defaultActiveKey="1" items={tabItems} onChange={changeTabs} />

            }

            <AuthModal isOpenModal={isAuthModalOpen} handleOk={handleOk} />

        </div>

    )

}