Некоммерческое акционерное общество  
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ  
имени Гумарбека Даукеева»  
Институт автоматизации и информационных технологий  
Кафедра IT-инженерии и искусственного интеллекта

**ОТЧЕТ  
по разработке WEB-приложения**

По дисциплине: WEB-программирование

На тему: Онлайн-платформа для публикации и обмена научными статьями

Образовательная программа 6B06102 – Информационные системы

Выполнил: Плугов М., Пак А., Калентьева А., Сабадах М., Конопатов Р.Д.

Группа ИС-22-5

Проверил: старший преподаватель Сейдазимов С. Б.

\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г.  
(оценка) (подпись)

Алматы 2024

**Цель**: Создание веб-приложения для учёных и исследователей, где они смогут загружать научные статьи, рецензировать работы других пользователей и участвовать в обсуждениях.

**Система контроля версий**: GitHub (все изменения и версии кода хранятся в репозитории для удобства работы в команде).

**Проектная группа:**

* [Сабадах М., Калентьева А., Конопатов Р.] — Front-end разработчик
* [Плугов М., Пак А.] — Back-end разработчик
* [Плугов М.] — Ответственный за тестирование/интеграцию

**Требования к функциональности:**

1. Аутентификация пользователей (регистрация, вход, выход).
2. Роли пользователей: обычный пользователь и администратор (при необходимости).
3. Основной функционал веб-приложения, соответствующий выбранной теме.
4. Административная панель для управления контентом (при необходимости).
5. Отзывчивый интерфейс (адаптивность для мобильных устройств).
6. База данных для хранения информации.
7. Деплой на облачную платформу.

**Этапы работы и критерии оценивания**

**Неделя 1: Планирование проекта (10%)**

**Архитектура приложения Front-end:** HTML, CSS и JavaScript используются для создания интерфейса и отображения содержимого сайта. **Back-end:** Python с использованием фреймворка Django для обработки запросов, аутентификации и управления ролями пользователей. **База данных:** MySQL (phpMyAdmin) для хранения информации о пользователях, статьях, рецензиях и комментариях.

**Роли участников**

* **Front-end разработчик**: отвечает за создание пользовательского интерфейса, разработку формы загрузки статей, адаптацию платформы для мобильных устройств.
* **Back-end разработчик**: разрабатывает серверную логику, функции аутентификации и ролевой модели, маршрутизацию и взаимодействие с базой данных.
* **Ответственный за тестирование/интеграцию**: проводит тестирование функционала, осуществляет интеграцию front-end и back-end частей, отвечает за деплой на облако.

**Неделя 2: Аутентификация и авторизация (15%)**

Во вторую неделю работы была реализована система аутентификации и авторизации для веб-приложения. Эта функция представляет пользователям возможность регистрации, входа в систему и выхода из неё. Было также осуществлено хранение данных в базе данных.

**Функциональные требования**

1. Зарегистрироваться с именем пользователя, почтовым ящиком и паролем.
2. Войти в систему при правильных данных аутентификации.
3. Выход из системы.
4. Защищенный доступ к страницам, требующим авторизации.

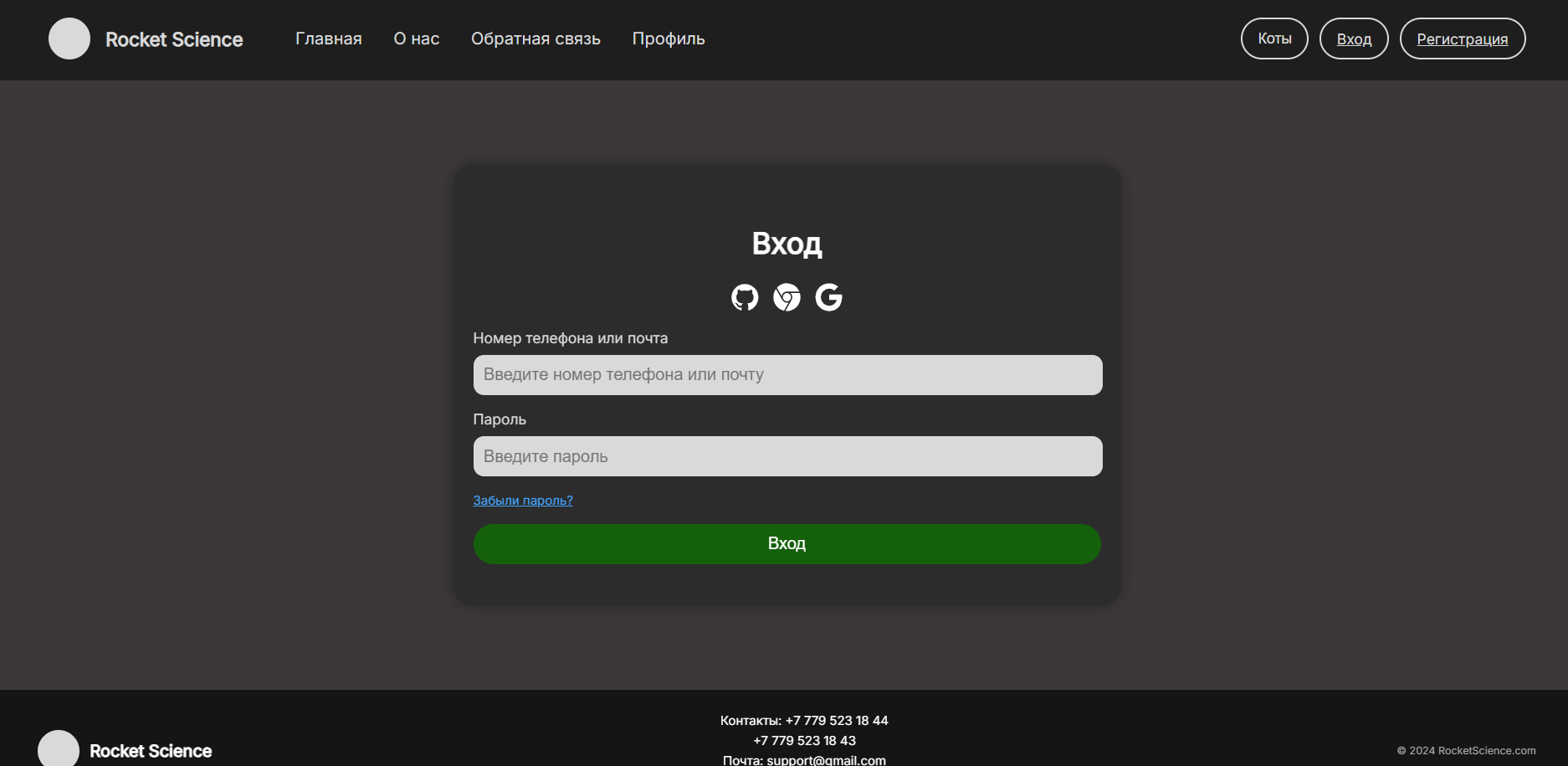


Рисунок 1 - страница входа в личный кабинет

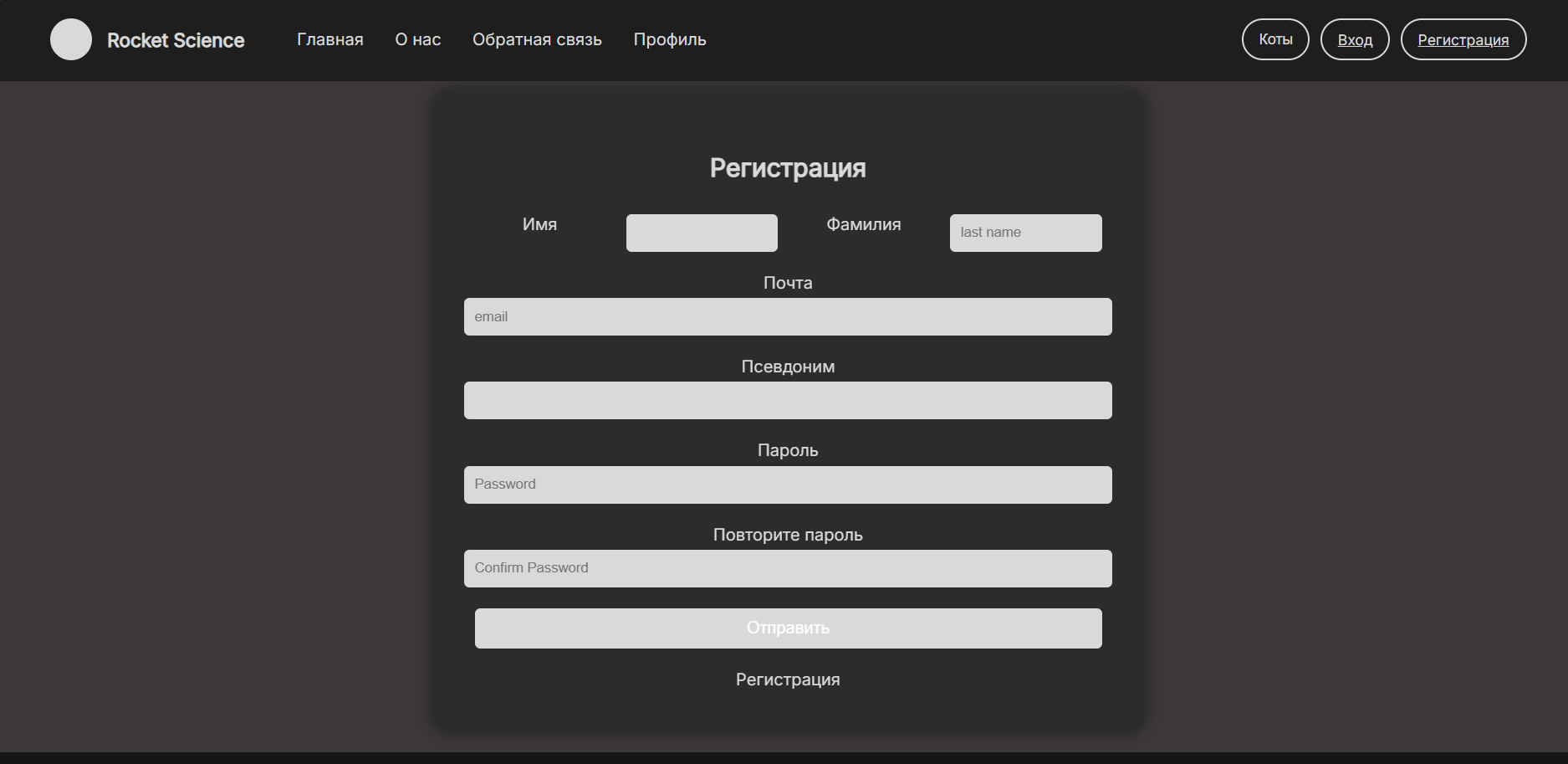


Рисунок 2 – страница регистрации пользователя

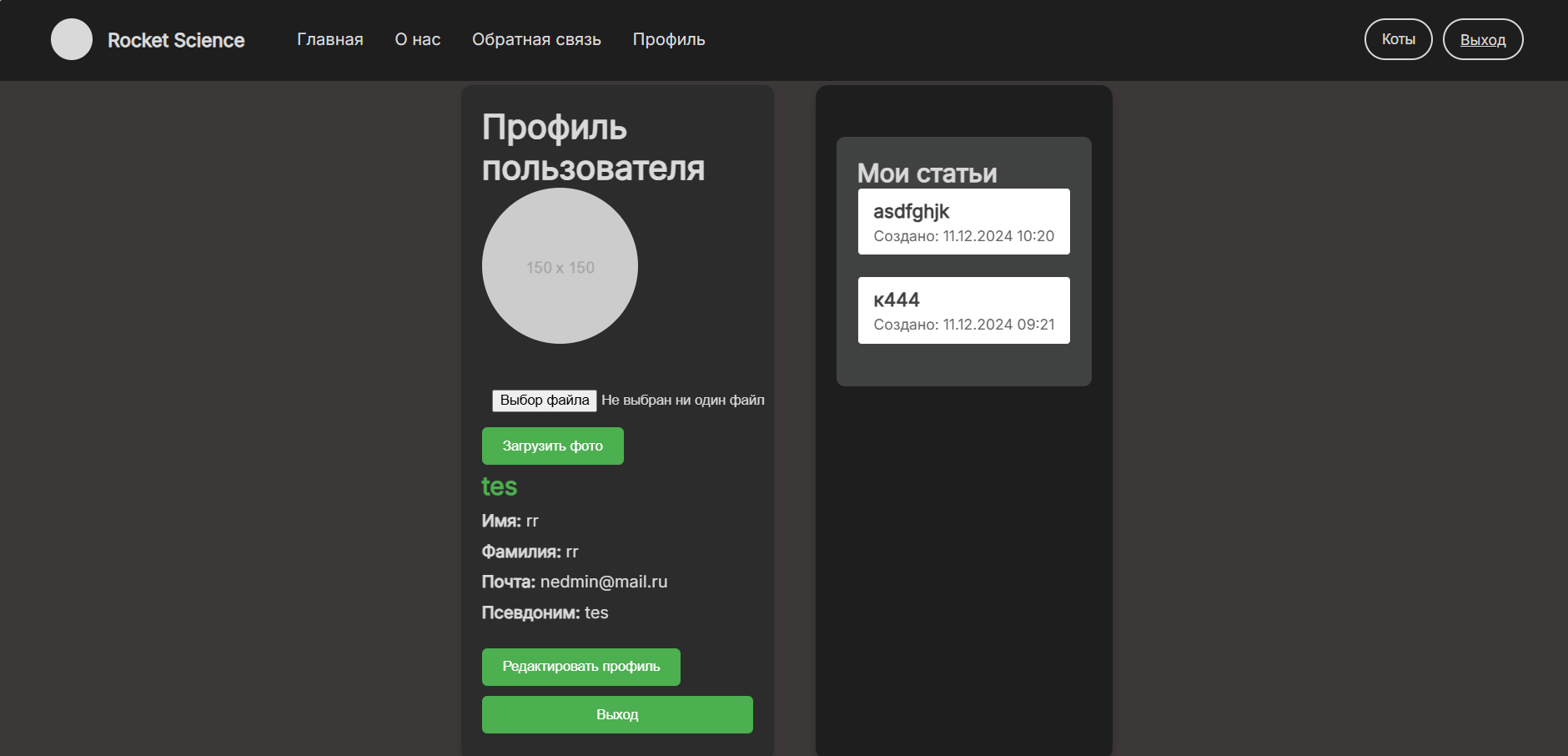


Рисунок 3 – профиль пользователя

**Неделя 2: Аутентификация и авторизация (15%)**

**Технологии и инструменты**

* **Язык программирования:** Python
* **Фреймворк:** Django
* **База данных:** phpMyAdmin
* **Фронтенд:** HTML, CSS, JavaScript
* **Библиотеки:** Django Authentication, Bootstrap (для интерфейса)

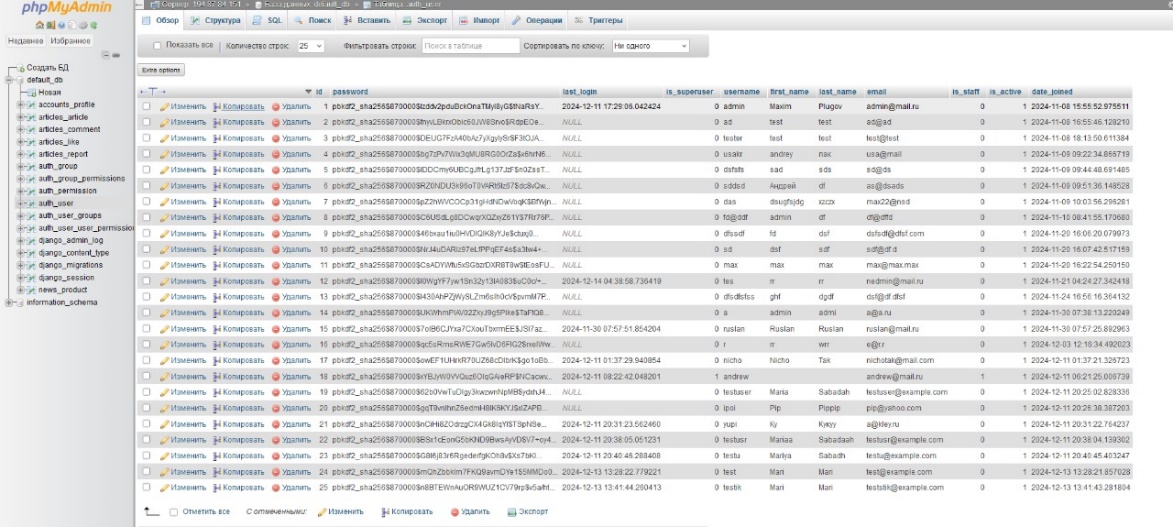


Рисунок 4 – База данных в SQL

В файле views.py определена модель пользователя с основными полями для хранения информации о пользователе.

**Форма редактирования профиля пользователя (EditProfileForm)**

class EditProfileForm(forms.ModelForm):

    class Meta:

        model = User

        fields = ['first\_name', 'last\_name', 'email', 'username']

        widgets = {

            'first\_name': forms.TextInput(attrs={'class': 'form-control'}),

            'last\_name': forms.TextInput(attrs={'class': 'form-control'}),

            'email': forms.EmailInput(attrs={'class': 'form-control'}),

            'username': forms.TextInput(attrs={'class': 'form-control'}),

        }

**Фрагмент: Аутентификация пользователя (login\_user и logout\_user)**

def logout\_user(request):

logout(request)

return redirect('login')

def login\_user(request):

if request.method == "POST":

user = authenticate(username=request.POST['username'], password=request.POST['password'])

if user:

login(request, user)

return redirect('profile')

messages.info(request, 'Invalid credentials')

return render(request, 'accounts/login.html')

**Неделя 3: Основной интерфейс пользователя (15%)**

* Разработайте пользовательский интерфейс для приложения.
* Убедитесь, что интерфейс адаптируется под мобильные устройства.

**Функциональные требования**

1. Просмотр и загрузка научных статей.
2. Рецензирование статей других пользователей.
3. Функция обсуждений и комментариев.
4. Адаптивный дизайн для мобильных устройств и планшетов.

Система аутентификации и авторизации успешно реализована, протестирована и интегрирована с базой данных. Этот функционал обеспечивает защиту пользовательских данных и доступ к основным возможностям веб-приложения.

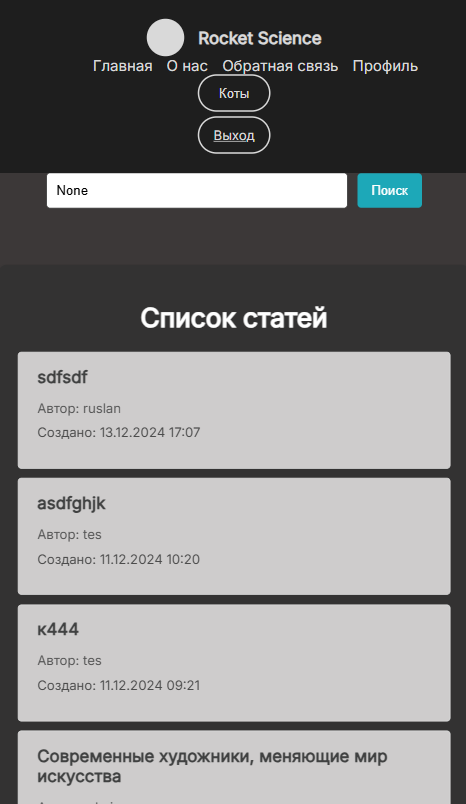


Рисунок 5 – адаптивная версия сайта

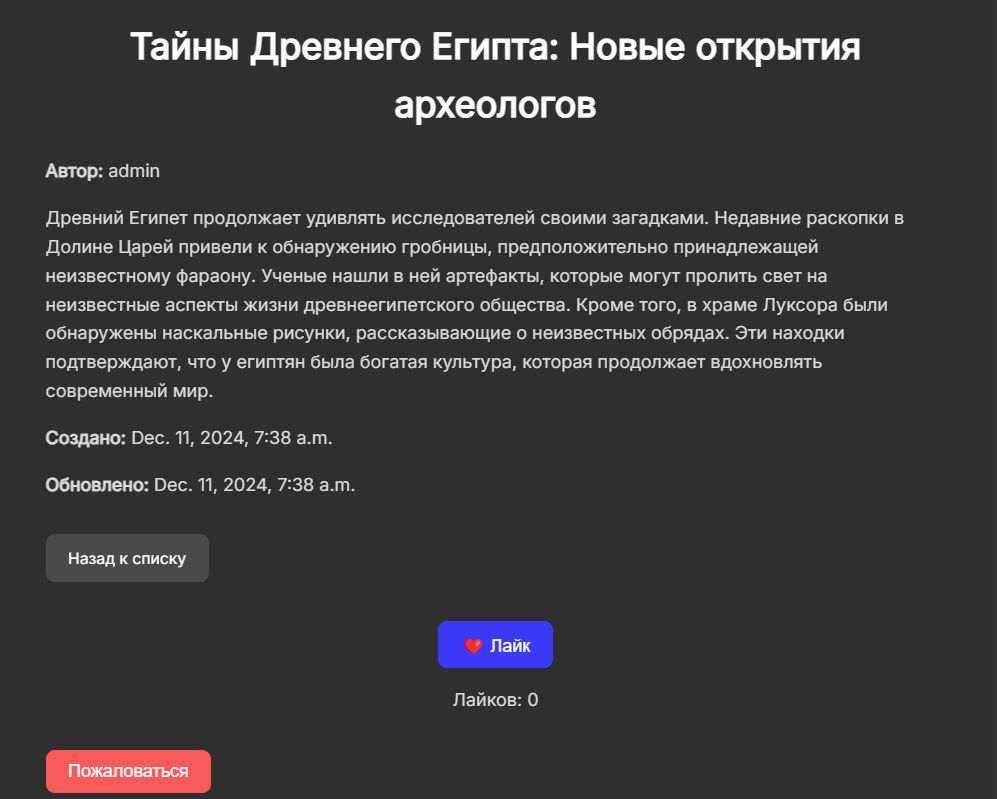


Рисунок 6 – Рецензирование научных статей

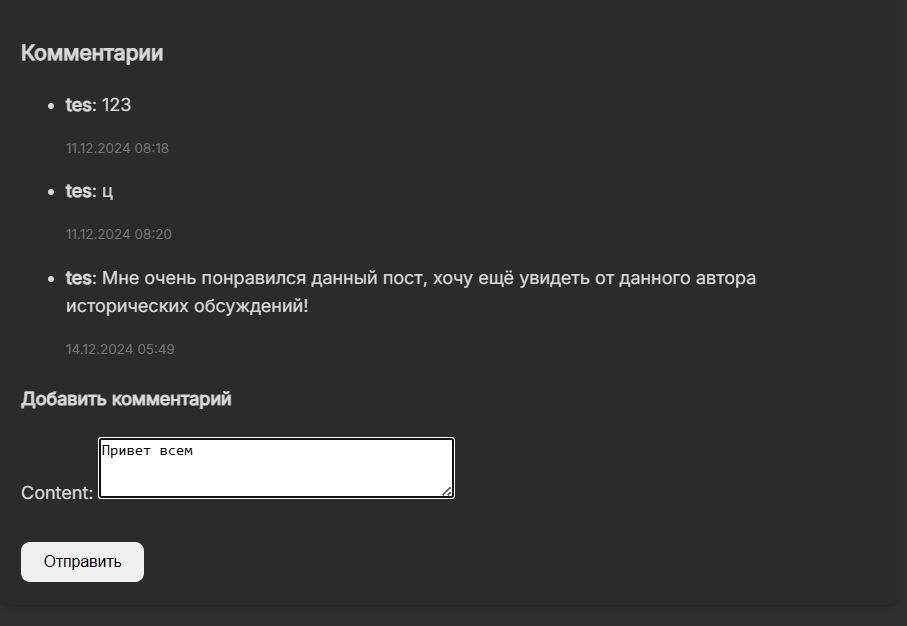


Рисунок 7 - Функция обсуждений и комментариев

Адаптивная настройка поиска для мобильных устройств

@media (max-width: 768px) {

.search-container {

padding: 0 10px;

height: 80px;

margin: 20px 10px 0;

}

.search-input {

font-size: 18px;

}

.icon-magnifying-glass {

top: 15px;

right: 10px;

}

}

**Неделя 4: Администрирование (10%) (если актуально для темы)**

**Технологии и инструменты**

* **Язык программирования:** Python
* **Фреймворк:** Django
* **Административная панель:** Django Admin
* **База данных:** phpMyAdmin

На четвёртой неделе была разработана административная панель для управления контентом приложения. Административная панель позволяет управлять пользователями, загруженными статьями, рецензиями и комментариями.

Административная панель позволяет эффективно управлять пользователями и контентом. Она предоставляет функции добавления, редактирования и удаления пользователей, а также возможность блокировки и разблокировки аккаунтов. Администраторы могут просматривать все загруженные статьи и удалять некорректный или нежелательный контент, обеспечивая контроль над качеством и актуальностью информации в системе.

**Сценарии тестирования:**

1. Успешный вход в административную панель.
2. Добавление и редактирование статей через Django Admin.
3. Удаление комментариев и рецензий.
4. Блокировка и разблокировка пользователей.

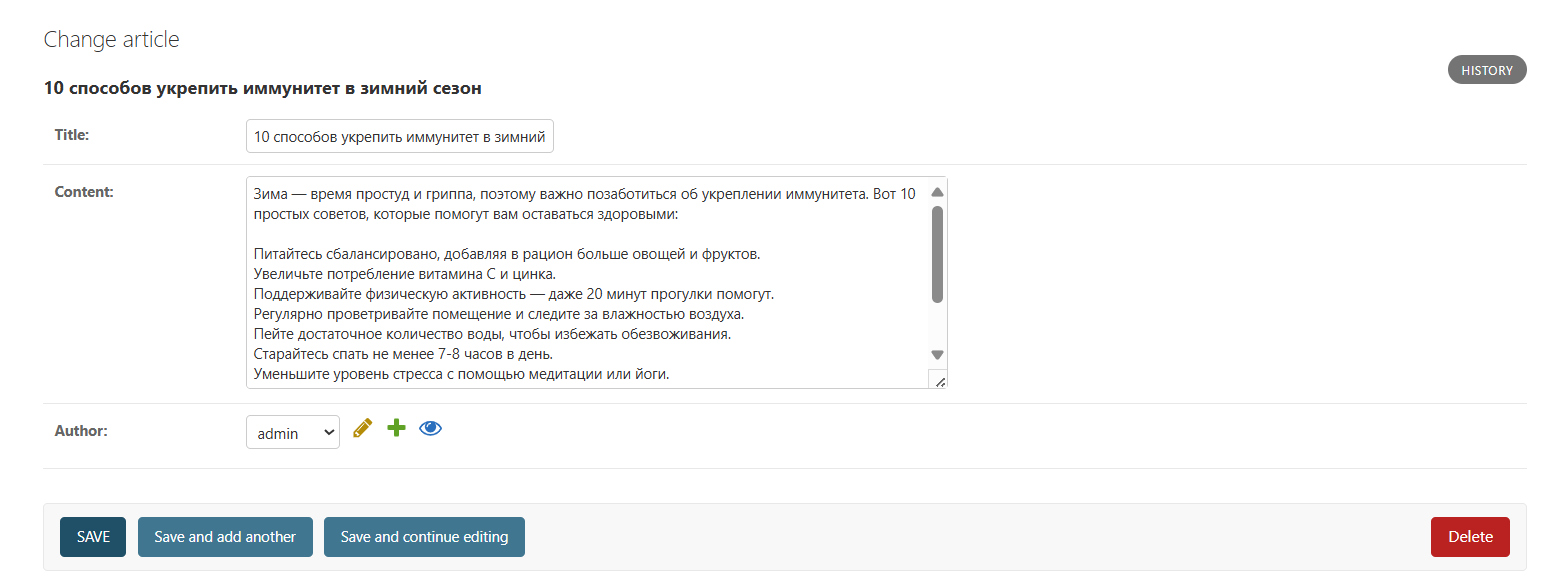


Рисунок 8 – рецензирование постов

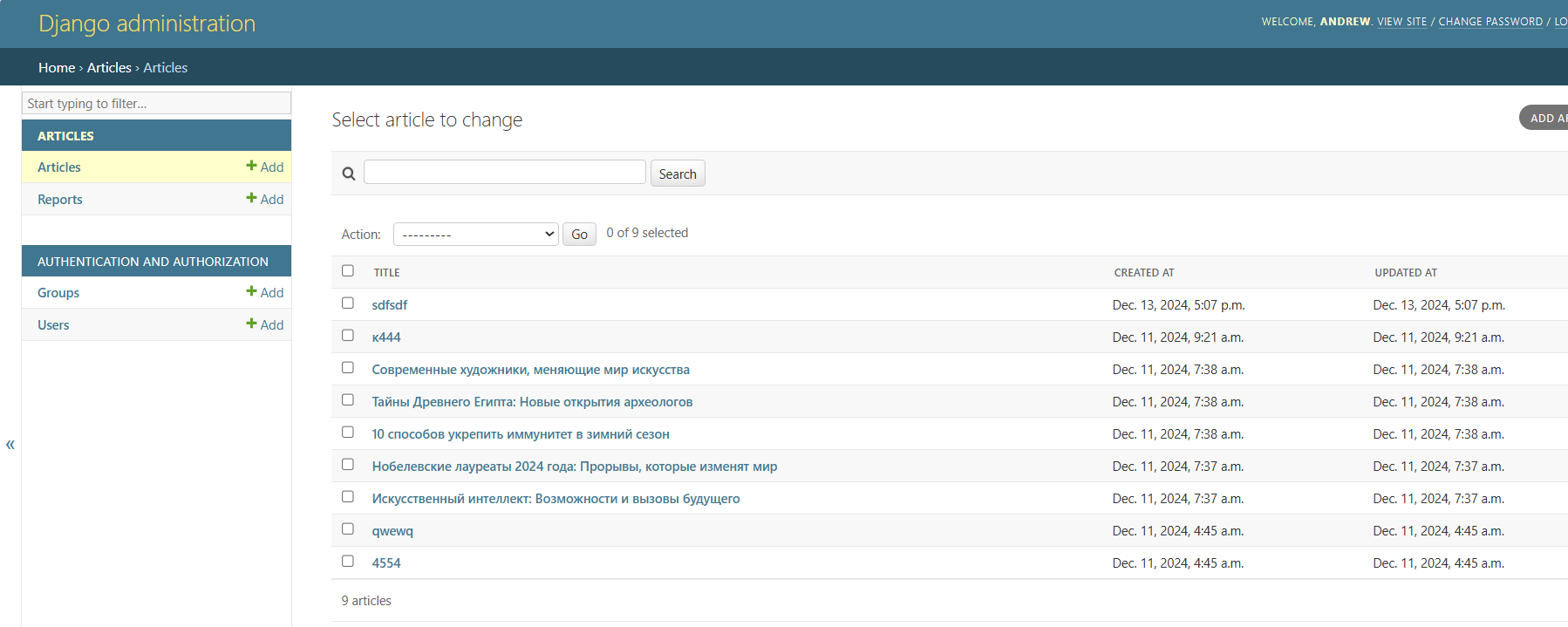


Рисунок 9 – просмотр всех постов

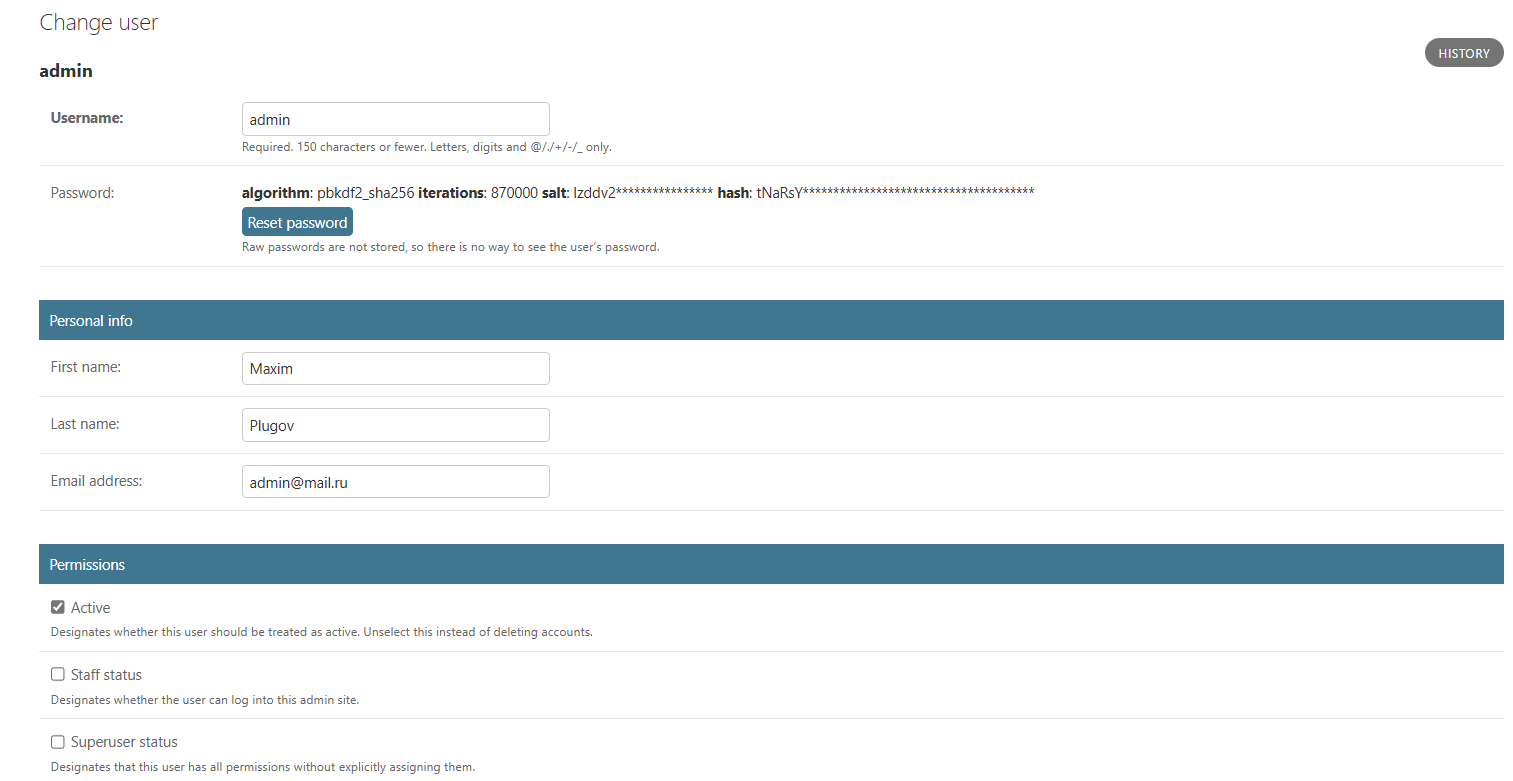


Рисунок 10 – просмотр данных о пользователе

Административная панель успешно реализована с использованием **Django Administration**. Она предоставляет полный функционал для управления контентом, пользователями и комментариями, обеспечивая удобный и интуитивный интерфейс для администратора.

**Неделя 5: База данных и API (15%)**

На пятой неделе была настроена база данных для хранения информации, связанной с проектом, а также реализовано взаимодействие фронтенда и бэкенда с помощью REST API. Для упрощения администрирования данных использовался встроенный инструмент **Django Administration**, который позволяет автоматически создавать интерфейс для управления базой данных.

**База данных** была разработана на основе **phpMyAdmin** и содержит следующие основные сущности: пользователи, статьи, рецензии и комментарии. Django ORM автоматически создал необходимые таблицы в базе данных на основе описанных моделей.

**REST API** обеспечил обмен данными между фронтендом и сервером, предоставляя доступ к данным статей и пользователям. Для создания и тестирования API использовались такие инструменты, как **Django REST Framework**.

Использование **Django Administration** позволило легко управлять данными: добавлять, редактировать и удалять записи, не требуя ручного создания SQL-запросов или интерфейсов для администрирования.

**Заключение**

В результате работы над проектом онлайн-платформы для публикации и обмена научными статьями были успешно реализованы основные компоненты веб-приложения. Разработанная система аутентификации и авторизации обеспечивает безопасность пользовательских данных и доступ к защищённым разделам сайта. Созданный пользовательский интерфейс отличается удобством и адаптивностью для различных устройств. Административная панель на основе **Django Administration** позволяет эффективно управлять пользователями и контентом, а также модерировать рецензии и комментарии. База данных и REST API обеспечивают надёжное хранение данных и взаимодействие между фронтендом и бэкендом. Внедрение системы контроля версий с использованием **GitHub** способствовало эффективной командной работе и прозрачности разработки.

**Список использованной литературы**

MDN Web Docs – ресурсы по веб-разработке:[HTML: HyperText Markup Language | MDN](https://developer.mozilla.org/en-US/docs/Web/HTML)

[CSS: Cascading Style Sheets | MDN](https://developer.mozilla.org/en-US/docs/Web/CSS)

[JavaScript Guide - JavaScript | MDN](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide)

[HTTP | MDN](https://developer.mozilla.org/en-US/docs/Web/HTTP)

W3Schools – справочные материалы и примеры по HTML, CSS, JavaScript:

[W3Schools Online Web Tutorials](https://www.w3schools.com/)

Eloquent JavaScript (Marijn Haverbeke) – книга по основам и продвинутым аспектам JavaScript.

[Eloquent JavaScript](https://eloquentjavascript.net/)