

МИНОБРНАУКИ РОССИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”

Факультет компьютерных наук

Кафедра программирования и информационных технологий

Курсовая работа по дисциплине «Технологии программирования»  
Свободная интернет-энциклопедия «WikiWorld»

Направление 09.03.02 Информационные системы и технологии

Преподаватель \_\_\_\_\_ В.С. Тарасов, ст. преподаватель \_\_\_\_\_.\_\_20\_\_

Обучающийся \_\_\_\_\_ С.П. Иванников, 3 курс, д/о

Обучающийся \_\_\_\_\_ С.Д. Сильнов, 3 курс, д/о

Обучающийся \_\_\_\_\_ И.Р. Саратов, 3 курс, д/о

Воронеж 2023

## Содержание

Содержание.....	2
Введение.....	4
1 Постановка задачи.....	6
1.1 Постановка задачи .....	6
1.2 Обзор аналогов.....	7
1.2.1 Wikipedia.....	7
1.2.2 Wikireality .....	8
1.2.3 Fandom.....	9
2 Анализ предметной области .....	11
2.1 Терминология (гlossарий) предметной области .....	11
2.2 Диаграммы, иллюстрирующие работу системы.....	11
2.2.1 Диаграмма прецедентов .....	11
2.2.2 Диаграмма активностей.....	13
2.2.3 Диаграмма последовательности .....	14
2.2.4 Диаграмма развертывания .....	17
2.2.5 Диаграмма классов .....	18
2.2.6 Диаграмма объектов .....	19
2.2.7 Диаграмма состояний .....	19
2.2.8 ER-диаграмма.....	20
2.2.9 Физическая схема базы данных.....	20
3 Реализация.....	22
3.1 Средства реализации.....	22
3.2 Реализация backend .....	24
3.3 Реализация frontend.....	27

3.4 Навигация по приложению .....	28
3.4.1 Для пользователя .....	28
3.4.2 Для администратора .....	33
3.5 Тестирование .....	35
3.5.1 Модульное тестирование .....	35
3.5.2 Дымовое тестирование .....	36
3.5.3 GUI-тестирование .....	37
Заключение .....	39
Список использованных источников .....	40

## **Введение**

В современном мире, который невозможно представить без вычислительной техники от серверных суперкомпьютеров до смарт-часов, активно встает вопрос доступа к достоверной научной информации. Одним из удобных способов получения энциклопедических знаний являются сетевые энциклопедии, которые позволяют пользователям искать подходящую информацию, вносить свои правки, опираясь на надежные источники, и ликвидировать свою неграмотность в отдельных вопросах и многом другом.

В данной курсовой работе будет рассмотрена разработка клиент-серверного веб-приложения «WikiWorld». Это предложение представляет собой удобную, быструю и свободно доступную всем энциклопедию.

В ходе работы будут рассмотрены основные этапы проектирования и разработки приложения, начиная с постановки задачи и анализа предметной области, и заканчивая созданием базы данных и написанием клиентской и серверной частей приложения. Особое внимание будет уделено вопросам user-friendly интерфейса и безопасности приложения, чтобы обеспечить удобство использования и защитить данные пользователей.

Приложение будет предоставлять пользователям возможность быстрого доступа к разнообразным энциклопедическим статьям, предоставлять возможность редактирования и улучшения материалов, а также поддерживать надежные источники информации. Это приложение будет служить надежным источником знаний для пользователей разных областей и интересов.

Кроме того, в рамках разработки будут учтены и потребности сотрудников, для которых предусмотрены специальные аккаунты с возможностью просмотра и редактирования информации о статьях, обеспечивая эффективное управление контентом и поддержку качества статей в энциклопедии.

Таким образом, разработка приложения не только удовлетворит потребности обычных пользователей в доступе к надежной научной

информации, но также обеспечит эффективные инструменты и интерфейсы для сотрудников, чтобы поддерживать высокое качество и актуальность контента.

## **1 Постановка задачи**

### **1.1 Постановка задачи**

Целью данного проекта, поставленной заказчиком перед разработчиками, является создание высококачественной сетевой энциклопедии, которая будет доступна для всех пользователей и предоставлять полезную информацию. Система имеет социальную значимость, обеспечивая доступ к знаниям, полезным в повседневной жизни и развитии пользователей. Важными аспектами разработки являются качество и достоверность информации, а также удобство использования и доступность энциклопедии для пользователей всех возрастов и уровней подготовки.

Для достижения поставленной цели система будет предоставлять следующие функциональности:

- Просмотр статей;
- Просмотр тематических подборок статей;
- Прослушивание статей;
- Возможность авторизованным пользователям работать над созданием и редактированием статей;
- Возможность администратора редактировать и удалять контент;
- Скачивание аудиофайла со статьей;
- Скачивание текстовых материалов статей;
- Поиск статей.

Система будет состоять из двух основных компонентов: backend (серверная) часть и frontend (клиентская) часть, которые будут взаимодействовать посредством REST API. Backend часть будет обрабатывать запросы, работать с базой данных, управлять доступом пользователей. Frontend часть будет отвечать за отображение информации в браузере и обеспечивать пользовательский интерфейс для взаимодействия с системой.

Таким образом, система «WikiWorld» решает задачи предоставления информации, совместного использования знаний, экспорта файлов и обеспечивает удобный поиск статей.

## **1.2 Обзор аналогов**

### **1.2.1 Wikipedia**

Wikipedia - это свободная сетевая энциклопедия, которая была запущена в 2001 году. Она предоставляет пользователям доступ к обширной базе данных статей по различным темам, созданных и редактируемых добровольцами. Wikipedia является одним из самых популярных и широкоизвестных онлайн-ресурсов, доступных на многих языках. Она позволяет пользователям искать информацию, узнавать о различных темах, и участвовать в коллективном создании и редактировании контента.

У Wikipedia есть несколько недостатков:

- Вандализм: Из-за открытой структуры, где любой пользователь может вносить изменения, Wikipedia подвержена вандализму. Некоторые пользователи могут намеренно вносить ложную информацию, неподобающий контент или удалять существующий контент, что может негативно повлиять на достоверность и качество статей;
- Ошибки и неполнота: Несмотря на усилия сообщества редакторов, ошибки и неполнота информации могут присутствовать в некоторых статьях. Wikipedia полагается на добровольных участников, и не всегда удается обеспечить полную проверку и обновление контента;
- Политика редактирования: Wikipedia имеет свои правила и политику редактирования, которые могут вызывать споры и разногласия. Некоторые пользователи могут считать эти правила слишком строгими или субъективными, что может ограничивать свободу редактирования и создания нового контента.

На рисунке 1 представлен скриншот сайта Wikipedia.

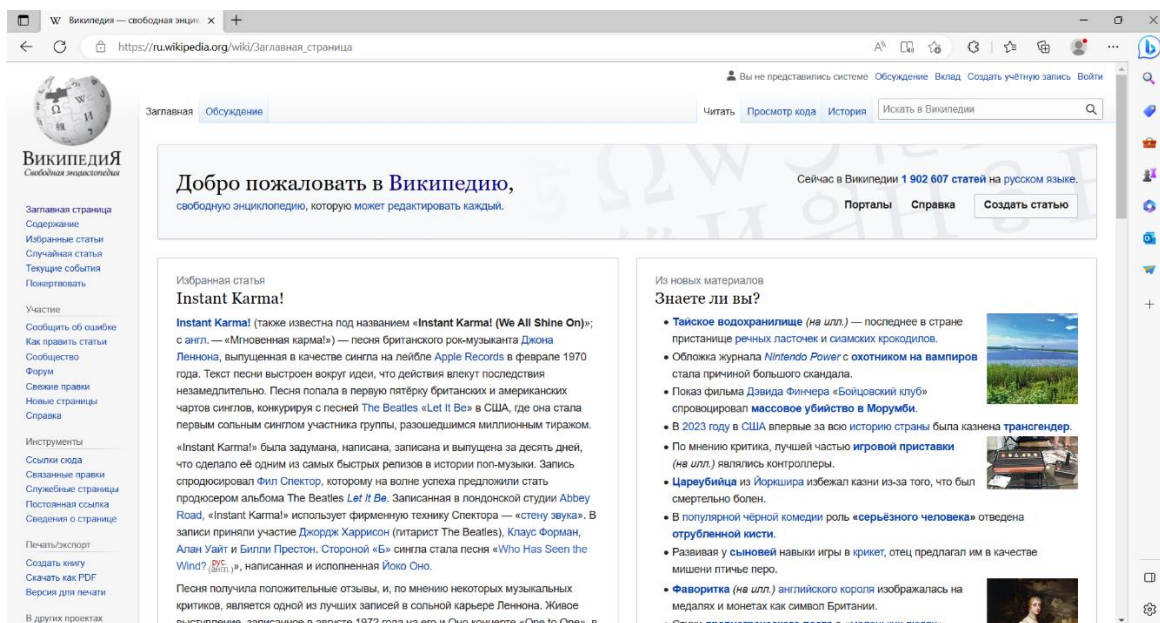


Рисунок 1 - Wikipedia

### 1.2.2 Wikireality

Wikireality — это российский проект, основанный бывшими локальными модераторами русского раздела Википедии, которые были не согласны с тогдашней политикой данного ресурса. Данный проект работает по принципу «больше — лучше», ведется активная работа на количество статей, критерии энциклопедической значимости не так велики, как в русской Википедии, и прежде всего данный ресурс хорошо отражает российские реалии, недостаточно описываемые на других аналогичных сайтах.

У Wikireality есть несколько недостатков:

- Низкое качество модерации;
- Много откровенно рекламных статей.

На рисунке 2 представлен скриншот сайта Wikireality.



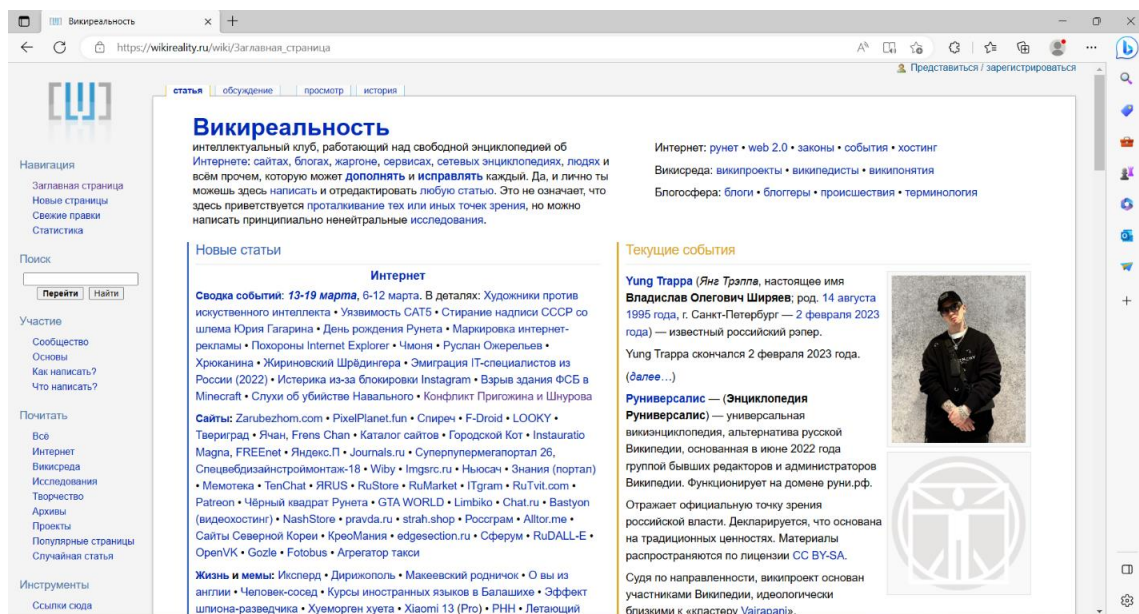


Рисунок 2 - Wikireality

### 1.2.3 Fandom

Fandom — это американский сетевой проект, позволяющий создать собственную тематическую энциклопедию на оптимизированном вики-движке. Особой популярностью пользуется у поклонников видеоигр, аниме, сериалов, желающих создать подробный ресурс о своем любимом творческом продукте.

У Fandom есть несколько недостатков:

- Частичная децентрализация проекта приводит к тому, что на ресурсе имеется много мусорных энциклопедий с низким качеством наполнения;
- Коммерческая модель с высоким количеством рекламы, с которой владельцы энциклопедий не получают никакого дохода;
- Fandom-лицензия, при которой все материалы по праву принадлежат головной компании, а не авторам, что снимает с энциклопедии критерий свободы.

На рисунке 3 представлен скриншот сайта Fandom.

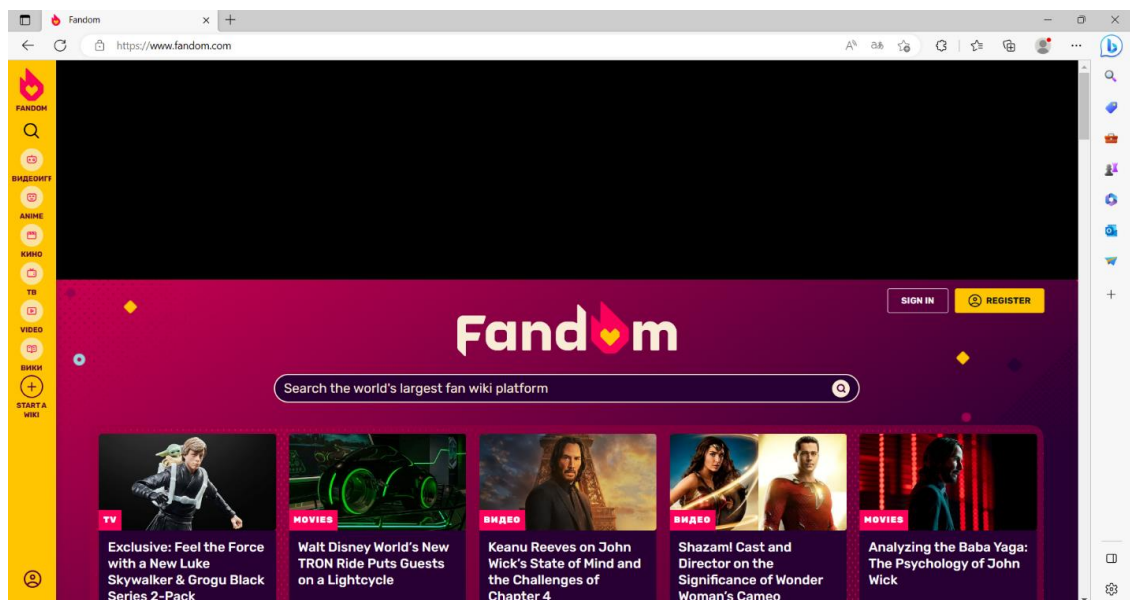


Рисунок 3 - Fandom

## **2 Анализ предметной области**

### **2.1 Терминология (гlossарий) предметной области**

Статья — это публикация, которая содержит информацию об определенной теме.

Тематика статьи — это конкретная тема или предмет, о котором написана статья. Она определяет, какую информацию следует включить в статью и как организовать ее содержание.

Веб-приложение — клиент-серверное приложение, в котором клиент взаимодействует с веб-сервером при помощи браузера.

MVC (Model-View-Controller) — схема разделения данных приложения и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо. Контроллер обрабатывает входящие запросы. Модель отвечает за данные, которые хранятся и обрабатываются на сервере. Представление определяет результат запроса, который получает пользователь. [1]

REST (Representational state transfer) API – это архитектурный стиль, который определяет правила обмена данными между клиентом и сервером.

Backend — логика работы сайта, внутренняя часть продукта, которая находится на сервере и скрыта от пользователя. [2]

Frontend — презентационная часть информационной или программной системы, ее пользовательский интерфейс и связанные с ним компоненты. [3]

Эндпоинт (endpoint) — это конечная точка или URL веб-сервиса или API, к которой можно обратиться для выполнения определенного запроса или получения определенной информации. Эндпоинты определяют, какие операции и ресурсы доступны в системе и каким образом они могут быть использованы.

## **2.2 Диаграммы, иллюстрирующие работу системы**

### **2.2.1 Диаграмма прецедентов**

Диаграмма прецедентов позволяет визуализировать различные типы ролей в системе и то, как эти роли взаимодействуют с системой. На диаграмме, изображенной на рисунке 4, присутствуют 3 актёра: неавторизованный, авторизованный пользователи, администратор.

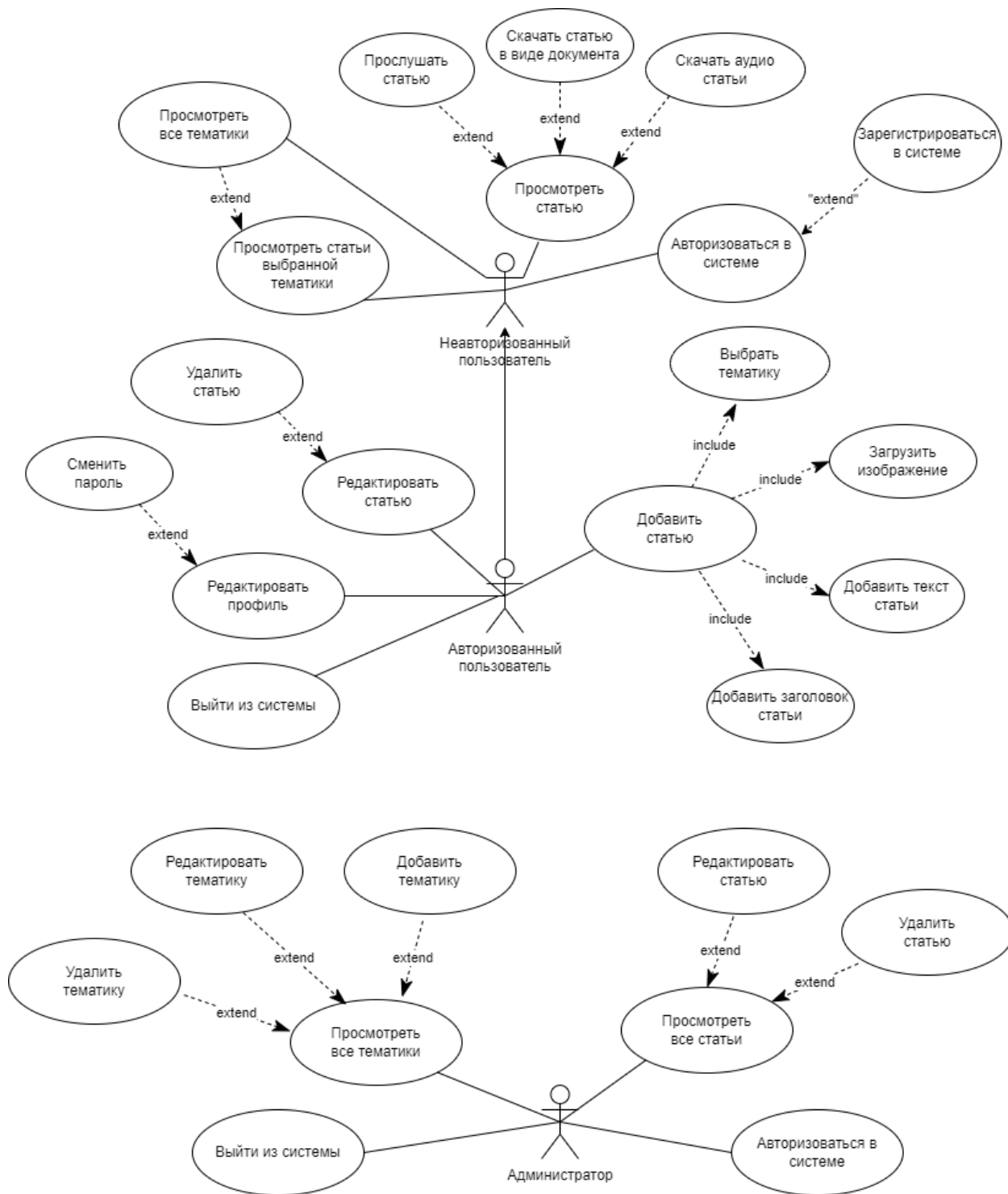


Рисунок 4 - Диаграмма прецедентов

Функции неавторизованного пользователя:

- Просмотр статьи выбранной тематики;
- Просмотреть все тематики;
- Просмотреть статью;
- Прослушать аудио статьи;
- Скачать аудио статьи;
- Скачать статью в виде документа;
- Авторизоваться в системе;
- Зарегистрироваться в системе.

Функции авторизованного пользователя:

- Добавить статью;
- Редактировать статью;
- Удалить статью;
- Редактировать профиль;
- Выйти из системы.

Функции администратора (пользователя системы, обладающего особыми правами):

- Просмотреть все тематики;
- Просмотреть все статьи;
- Добавить тематику;
- Удалить тематику;
- Редактировать тематику;
- Редактировать статью;
- Авторизоваться в системе;
- Удалить статью;
- Выйти из системы.

### **2.2.2 Диаграмма активностей**



На рисунках 6-8 представлены диаграммы последовательности для трех основных актеров системы.

Неавторизованный  
пользователь

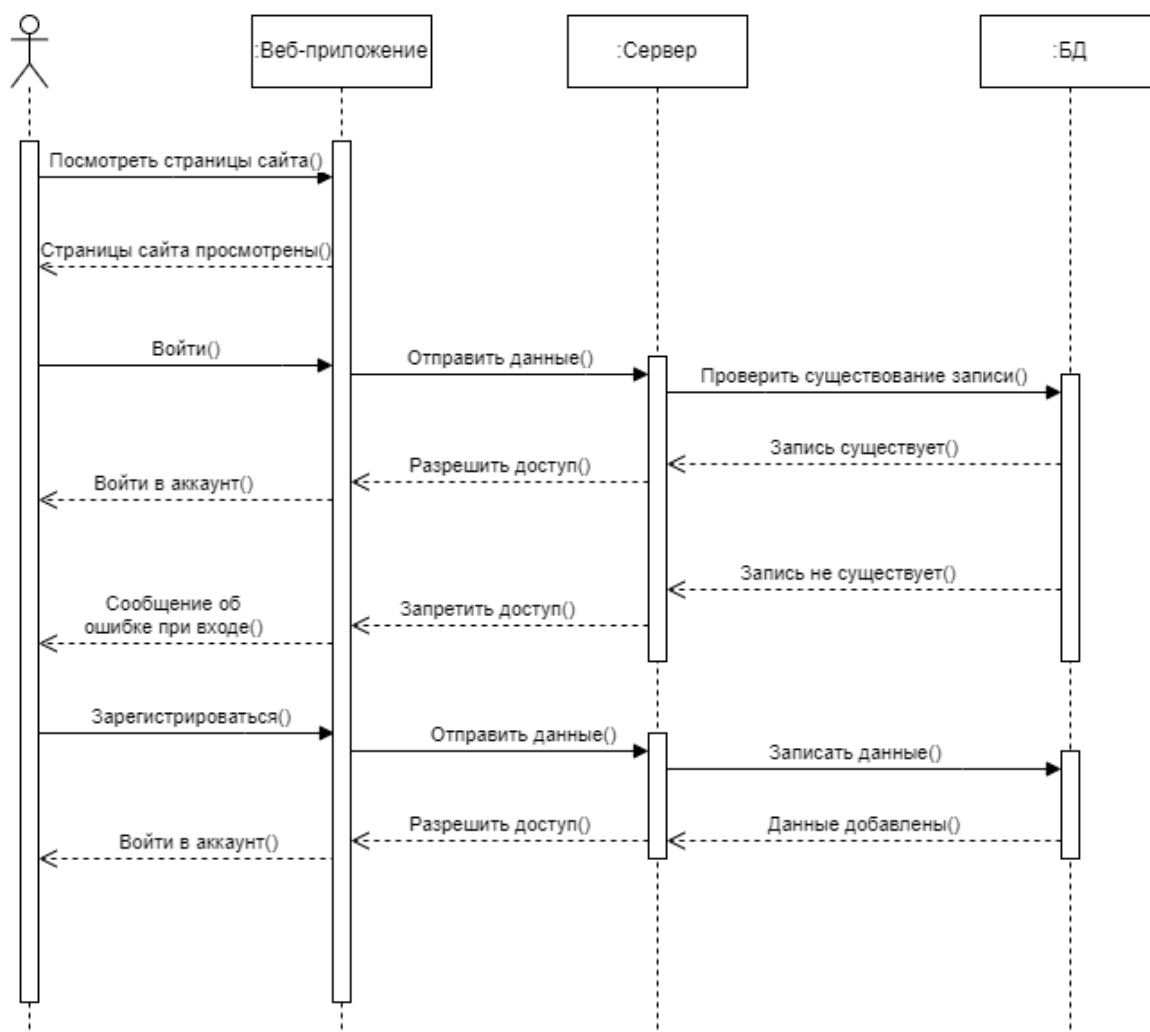


Рисунок 6 - Диаграмма последовательности для неавторизованного пользователя

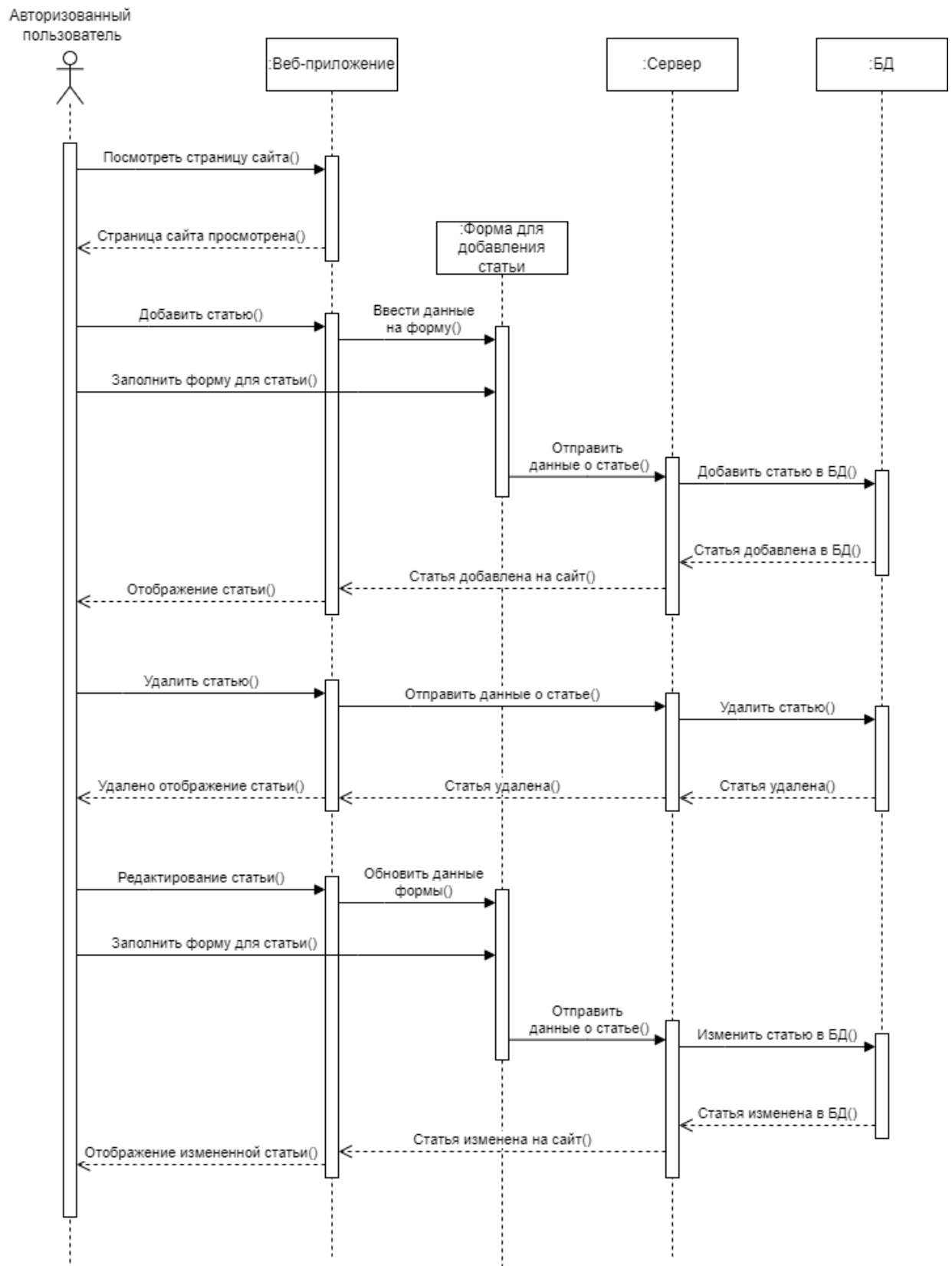


Рисунок 7 - Диаграмма последовательности для авторизованного пользователя



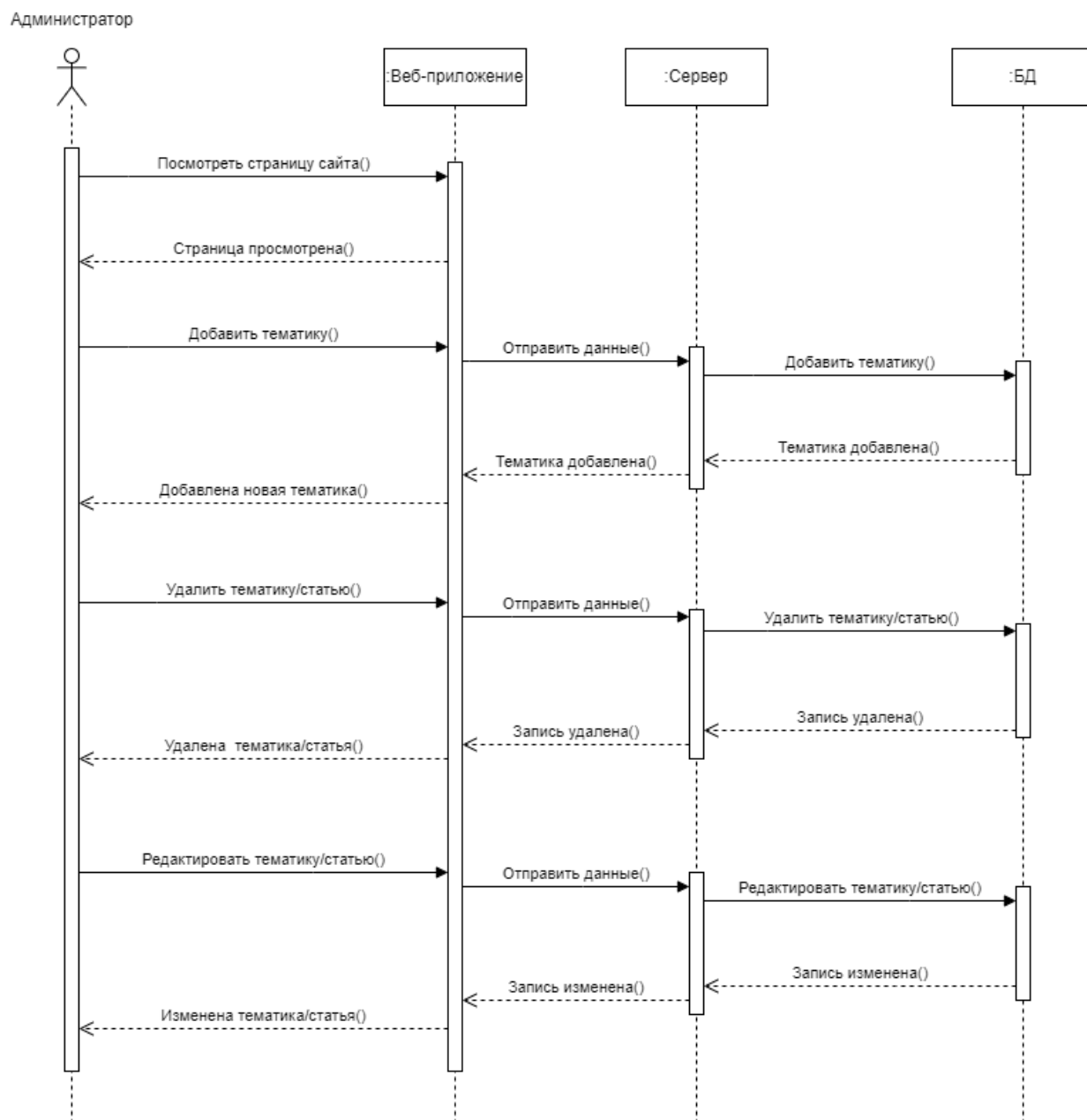


Рисунок 8 - Диаграмма последовательности для администратора

#### 2.2.4 Диаграмма развертывания

Диаграмма развертывания предназначена для представления общей конфигурации или топологии распределенной программной системы.

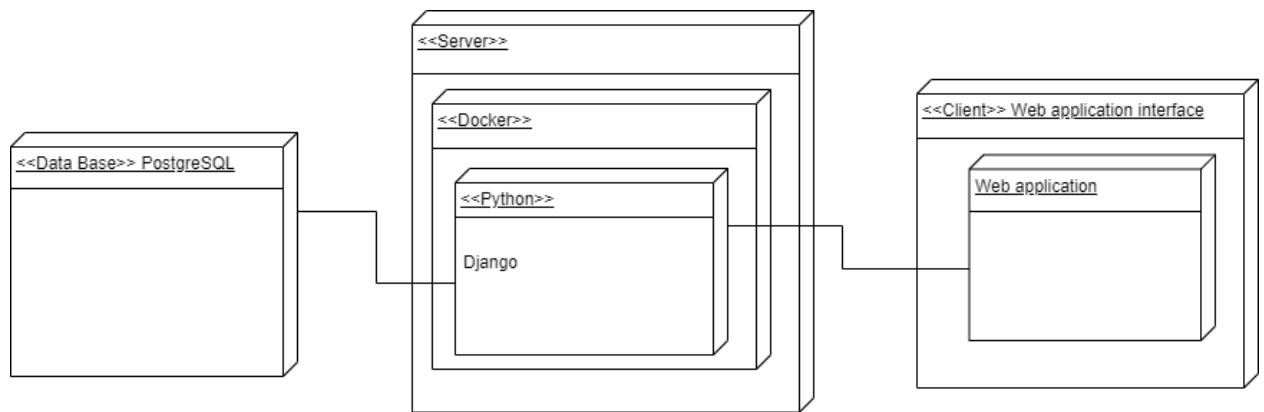


Рисунок 9 - Диаграмма развертывания

На рисунке 9 изображена топология разрабатываемой системы.

### 2.2.5 Диаграмма классов

Диаграмма классов предназначена для представления внутренней структуры программы в виде классов и связей между ними.

На рисунке 10 изображена диаграмма классов для различных категорий объектов.

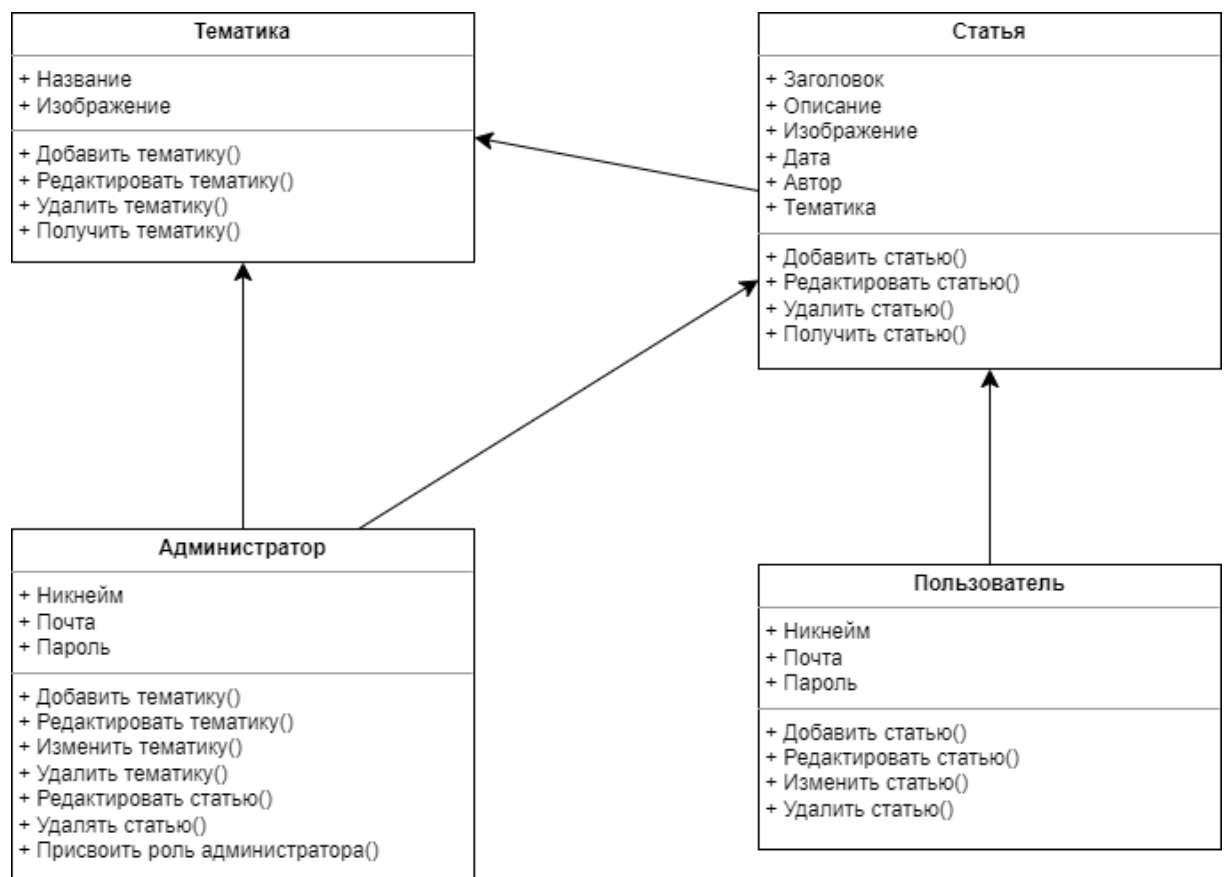


Рисунок 10 - Диаграмма классов

### 2.2.6 Диаграмма объектов

Диаграмма объектов в языке моделирования UML предназначена для демонстрации совокупности моделируемых объектов и связей между ними в фиксированный момент времени.

На рисунке 11 изображена диаграмма объектов для разрабатываемой системы.

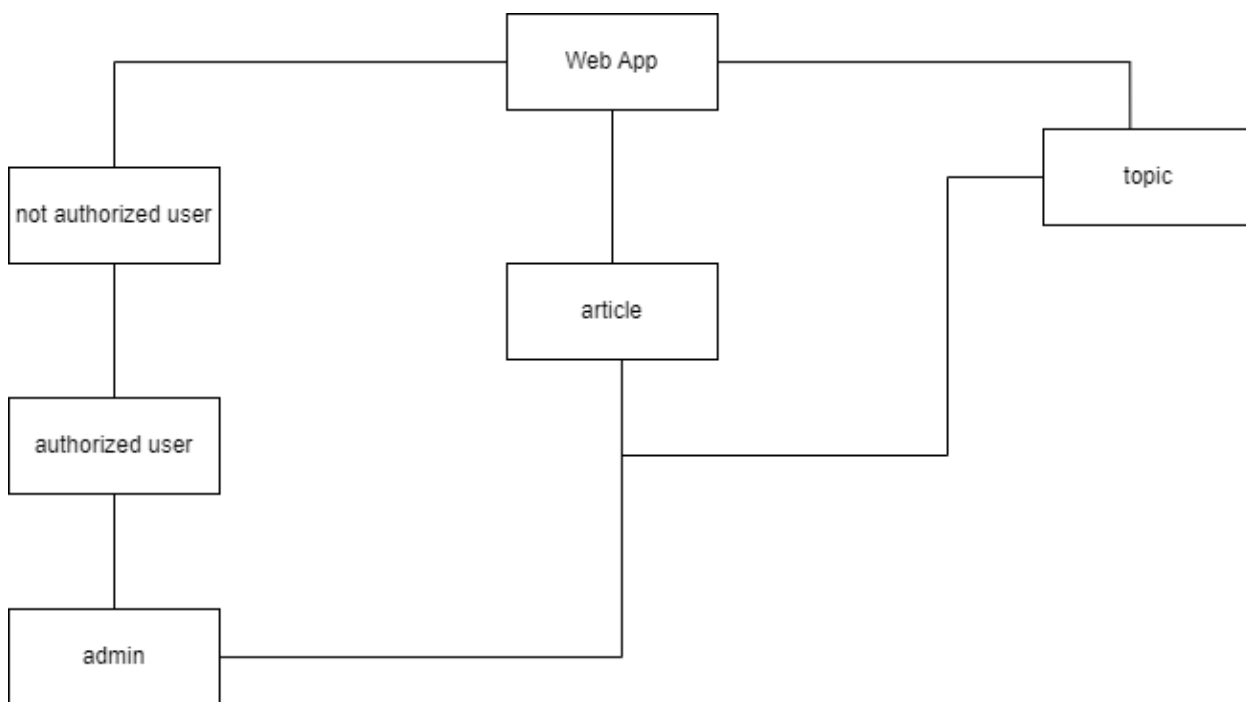


Рисунок 11 - Диаграмма объектов

### 2.2.7 Диаграмма состояний

На диаграмме состояний отображаются все переходы между состояниями изменяющегося объекта системы.

На рисунках 12-14 изображены диаграммы состояний для изменяющихся во времени объектов системы, а именно для статьи, тематики и пользователя.



Рисунок 12 - Диаграмма состояний для статьи



Рисунок 13 - Диаграмма состояний для тематики

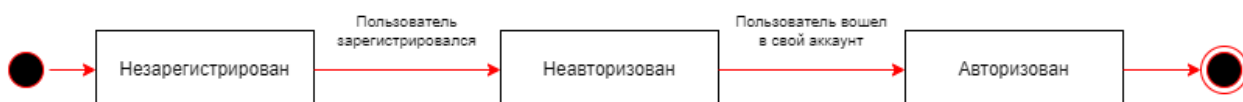


Рисунок 14 - Диаграмма состояний для пользователя

### 2.2.8 ER-диаграмма

Диаграмма «Сущность-связь» (ER-диаграмма) — визуальное представление базы данных, которое показывает, как связаны элементы внутри. В данном случае она иллюстрирует, какие есть сущности и как они связаны внутри системы разрабатываемого веб-приложения.

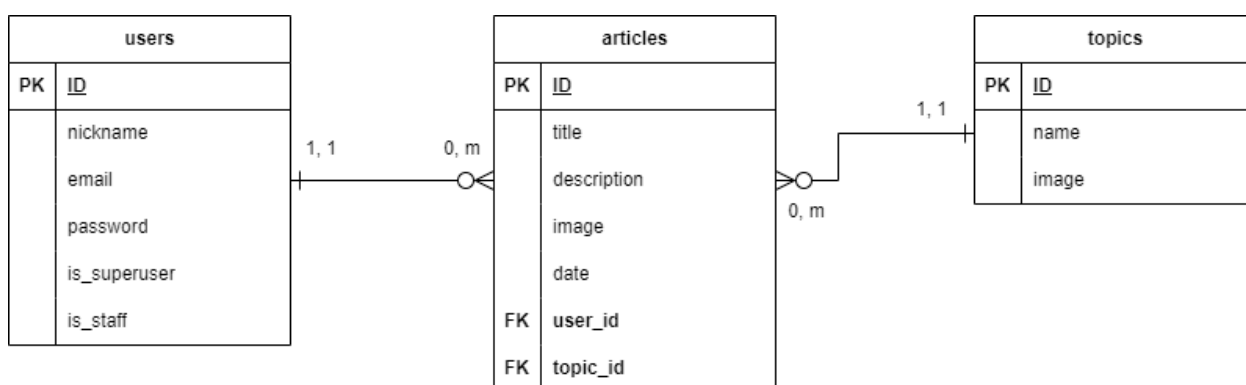


Рисунок 15 - ER-диаграмма

На рисунке 15 можно увидеть, какую информацию хранят в себе сущности, а также взаимодействия между ними.

### 2.2.9 Физическая схема базы данных

Физическая схема базы данных — это модель данных, которая определяет, каким образом представляются данные, и содержит все детали, необходимые СУБД для создания базы данных.

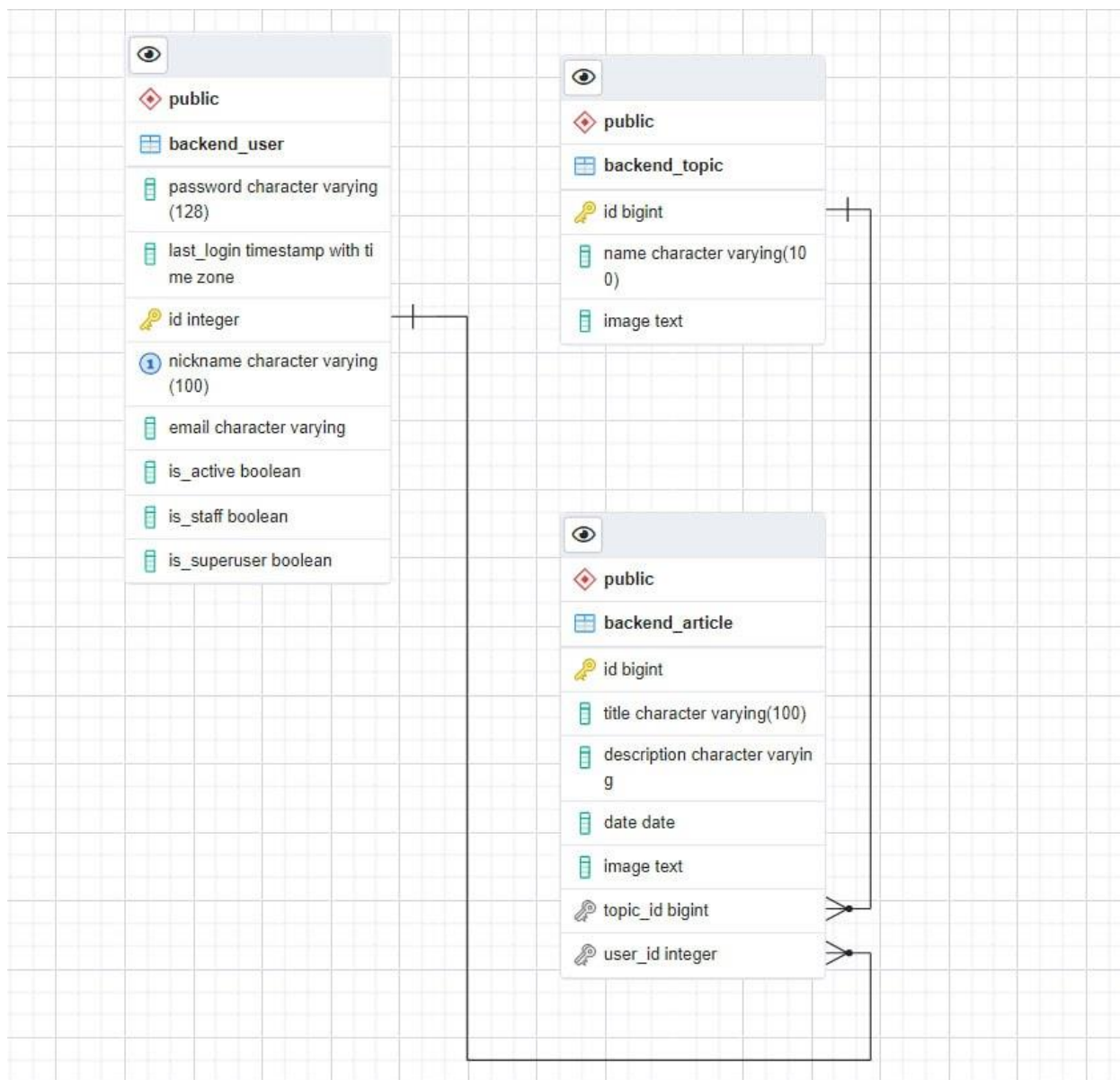


Рисунок 16 - Физическая схема базы данных

На рисунке 16 можно увидеть, как ER-диаграмма преобразовалась в физическую схему базы данных.

### **3 Реализация**

Работа в команде велась в соответствии с гибкой методологией разработки Kanban. В начале разработки была создана доска Kanban, на которой отображались все задачи проекта в виде карточек. Каждая карточка представляла определенную задачу или работу, которую требовалось выполнить.

В рамках работы по методологии Kanban не было фиксированной длительности спринтов. Вместо этого, задачи брались из бэклога и добавлялись на доску Kanban по мере доступности ресурсов и возможностей команды. Команда обсуждала и выбирала задачи, которые они могут выполнить в текущий период времени.

Встречи участников команды проводились по мере необходимости, чтобы обсудить прогресс, планы и трудности. Они могли происходить в формате ежедневных стендап-митингов или других созвонов, когда это было необходимо для синхронизации работы.

#### **3.1 Средства реализации**

Для разработки серверной (backend) части приложения был выбран следующий стек технологий:

- Python — высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ;
- Django REST Framework — это мощный и гибкий инструмент для создания веб-интерфейсов (web APIs);
- Django — это высокоуровневый веб-фреймворк, написанный на языке Python, который позволяет разрабатывать быстрые и масштабируемые веб-приложения. Django предоставляет множество инструментов и функций, упрощающих процесс разработки веб-приложений, таких как

управление базами данных, маршрутизация URL-адресов, обработка форм, аутентификация пользователей и многое другое;

- PostgreSQL — объектно-реляционная система управления базами данных с открытым исходным кодом, основанная на языке SQL. [4] Она обладает следующими преимуществами: поддержка транзакций, возможность использования Hibernate;
- Swagger — инструмент для документирования и тестирования API. [5] Он позволяет создавать интерактивную документацию для веб-сервисов, что упрощает их использование и интеграцию. Swagger автоматически генерирует документацию на основе аннотаций в коде, что позволяет разработчикам сосредоточиться на написании логики приложения, а не на создании и поддержке документации. Благодаря Swagger, разработчики могут изучить доступные эндпоинты, параметры, модели данных и примеры запросов и ответов;
- Docker — это открытая платформа, которая позволяет автоматизировать развертывание, упаковку и запуск приложений в легковесных и изолированных контейнерах. Контейнеры Docker обеспечивают стандартизацию окружения, включая операционную систему, библиотеки и зависимости, и позволяют запускать приложения на различных системах без изменений;
- Yandex Cloud (Яндекс.Облако) — это облачная платформа, предоставляемая компанией "Яндекс", которая предлагает широкий спектр облачных услуг для хранения данных, развертывания приложений, аналитики, машинного обучения и других задач.

Для разработки клиентской (frontend) части приложения был выбран следующий стек технологий:

- JavaScript — высокоуровневый интерпретируемый язык программирования, который применяется в веб-разработке для создания интерактивных элементов на веб-страницах. Он обладает

следующими преимуществами: возможность создания динамических и интерактивных веб-страниц, доступ к различным библиотекам и фреймворкам;

- CSS — язык стилей, используемый для оформления веб-страниц. Он определяет внешний вид и расположение элементов на странице, включая цвета, шрифты, размеры, отступы, границы и другие стилевые свойства. Он обладает следующими преимуществами: возможность изменения внешнего вида всего сайта путем изменения одного файла CSS, возможность переиспользования стилей благодаря созданию классов и переопределения стилей для различных элементов;
- HTML — язык разметки, используемый для создания структуры и содержимого веб-страниц. Он обладает следующими преимуществами: читаемость для разработчиков, возможность создания структурированного контента с использованием семантических элементов, доступность и адаптивности для различных устройств и браузеров. HTML является основным языком для создания веб-страниц и работает в сочетании с CSS для определения внешнего вида и JavaScript для добавления интерактивности;
- Bootstrap — свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML и CSS шаблоны оформления для типографики, веб форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript расширения.

### **3.2 Реализация backend**

Серверная (backend) часть приложения была написана на языке Python с использованием фреймворка Django. Такие проекты обычно состоят из нескольких независимых приложений, каждое из которых выполняет определенные функции. Структура приложения и иерархия пакетов с модулями изображена на рисунке 17.



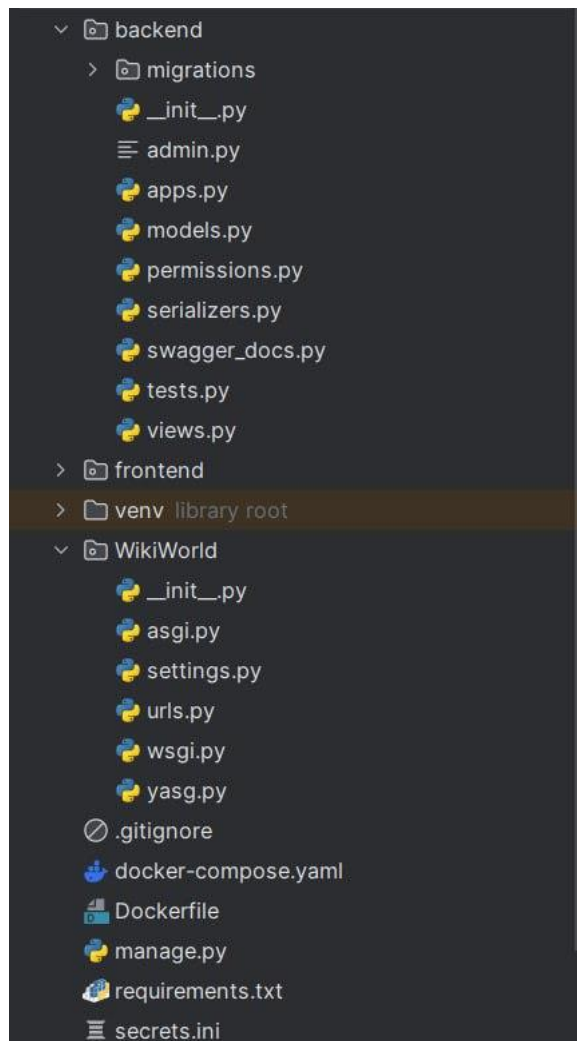


Рисунок 17 - Структура серверной части приложения

Приложение разделено на следующие модули:

- Модуль `apps.py`. Этот модуль является частью структуры приложения и служит для конфигурирования приложения. Он содержит метаданные и настройки, связанные с конкретным приложением;
- Модуль `admin.py`. Этот модуль используется для настройки административного интерфейса (Admin Site) проекта. Он позволяет определить модели, которые можно отображать и редактировать в панели администратора, а также настраивать их отображение и поведение;
- Модуль `models.py`. Этот модуль используется для определения моделей данных, которые представляют структуру и хранение данных

- приложения. Модели определяют таблицы в базе данных и предоставляют способ взаимодействия с данными, включая создание, чтение, обновление и удаление;
- Модуль `permissions.py`. Этот модуль используется для определения прав доступа в приложении. Он позволяет контролировать, какие пользователи или группы пользователей имеют доступ к определенным функциям и ресурсам приложения;
  - Модуль `serializers.py`. Этот модуль используется для определения сериализаторов в приложении. Сериализаторы в Django позволяют преобразовывать сложные типы данных, такие как модели, в простые типы данных, которые могут быть переданы через сеть или сохранены в базе данных;
  - Модуль `swagger_docs.py`. Этот модуль используется для определения декораторов `swagger_auto_schema`, которые являются частью пакета `drf-yasg`. Эти декораторы применяются к представлениям Django REST Framework API для автоматической генерации спецификации Swagger и документации для каждой операции API;
  - Модуль `views.py`. Этот модуль используется для определения представлений API. Он содержит классы, которые определяют различные операции, такие как получение, создание, обновление и удаление данных;
  - Модуль `settings.py`. Этот модуль является основным файлом настроек проекта Django, который используется для настройки и конфигурации DRF и других компонентов приложения;
  - Модуль `urls.py`. Этот модуль используется для определения маршрутов URL API. Он служит для связывания конкретных URL-путей с соответствующими представлениями, которые обрабатывают запросы и формируют ответы;

— Модуль `yasg.py`. Этот модуль используется для настройки и интеграции инструмента Swagger в API.

Сервер был развернут на хостинге Yandex Cloud с использованием Docker.

Для описания спецификации API в проекте использовался Swagger. Для интеграции его в проект, потребовалось добавить соответствующую зависимость (`drf-yasg`) и декораторы к представлениям. На основе этих декораторов и структуры приложения составляется интерактивная документация API.

### **3.3 Реализация frontend**

Клиентская часть приложения (frontend) разработана на языке JavaScript с использованием фреймворка Django. Для взаимодействия с серверной частью приложения на клиентской части используются библиотеки `ajax` и `jquery`, которые позволяют обмениваться данными с сервером без перезагрузки страницы.

Разработанное приложение имеет следующую структуру: модули `urls.py` и `views.py` (для рендеринга страниц), папку `templates` с html страницами для авторизованного, неавторизованного пользователей и администратора, папку `static`, в которой находятся папки `css` (для хранения CSS файлов, описывающих стили), `fonts` (в ней хранятся шрифты, которые используются на страницах), `img` (в ней находятся необходимые приложению изображения) и `scripts` (в этой папке находятся `.js` файлы для обработки запросов).

Описанная структура приложения изображена на рисунке 18.

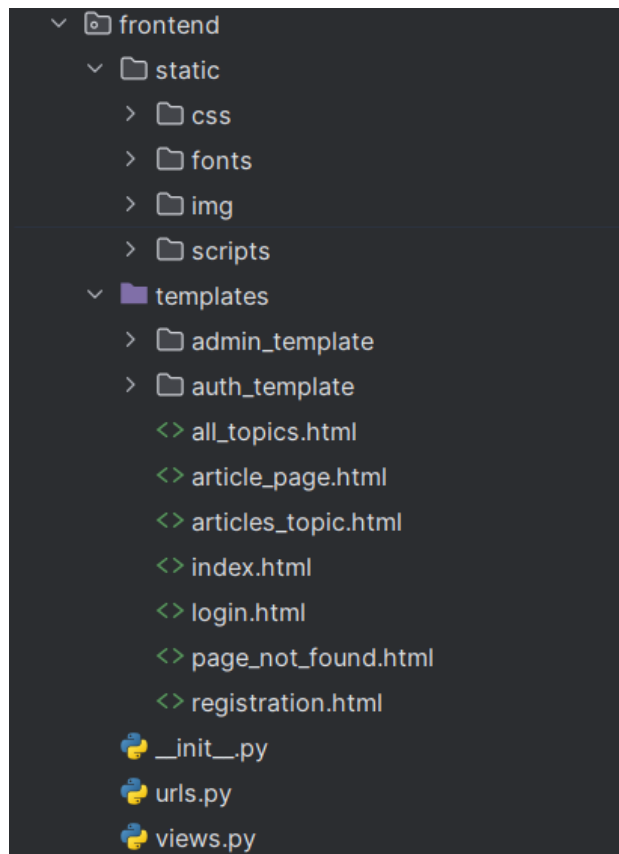


Рисунок 18 - Структура frontend части

### 3.4 Навигация по приложению

#### 3.4.1 Для пользователя

При запуске приложения пользователя встречает главная страница, представленная на рисунке 19.

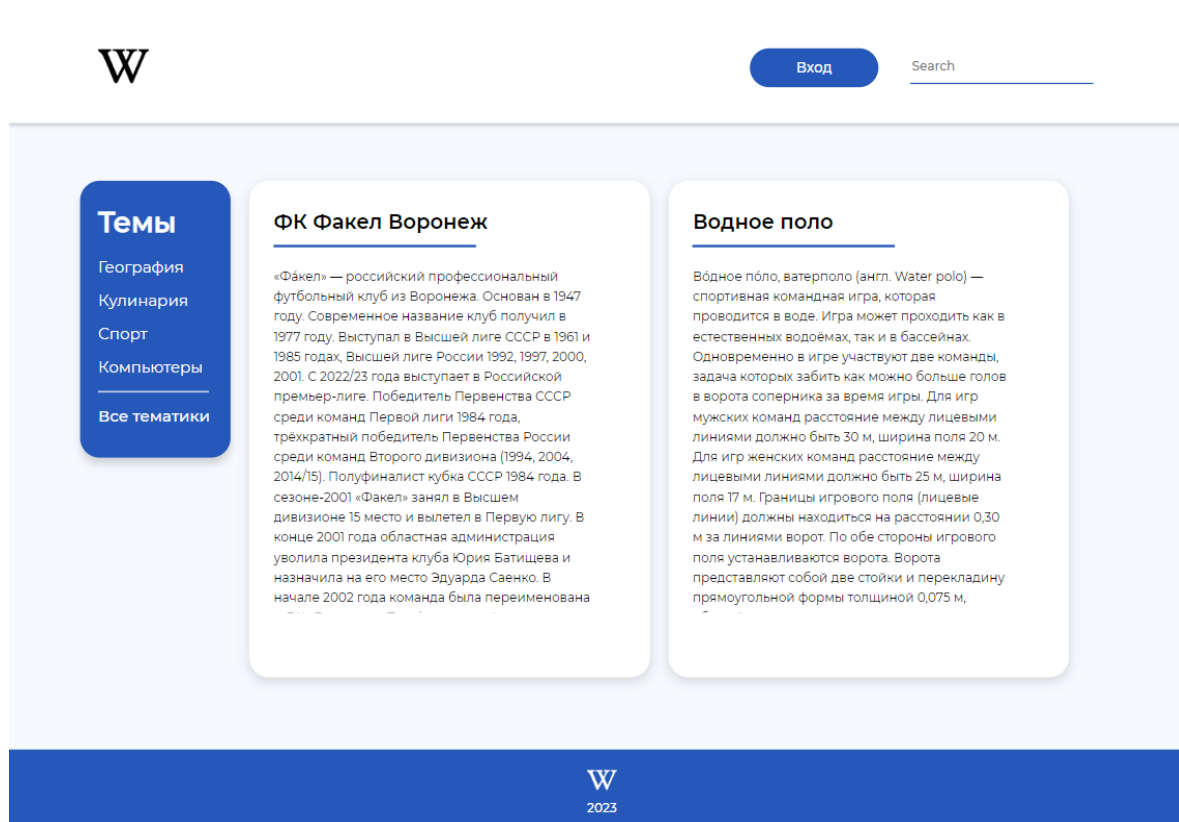


Рисунок 19 - Главная страница

У пользователя есть выбор: войти в существующий аккаунт, пройти регистрацию или продолжить пользоваться приложением без возможности работать с контентом. Страница входа представлена на рисунке 20, страница регистрации — на рисунке 21.

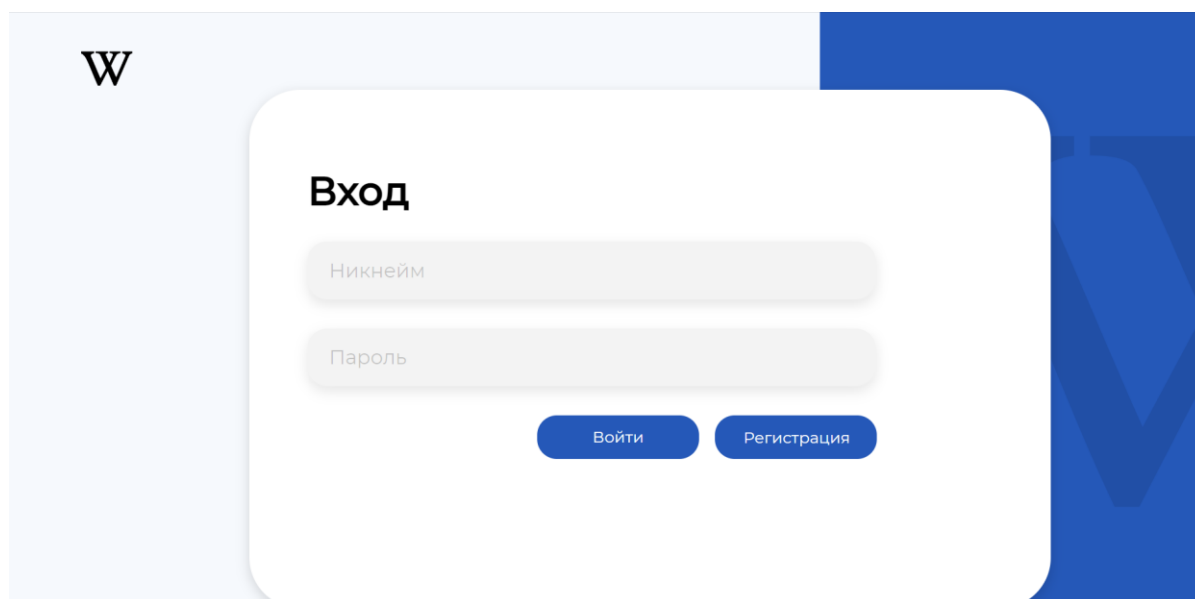


Рисунок 20 - Страница входа в существующий аккаунт

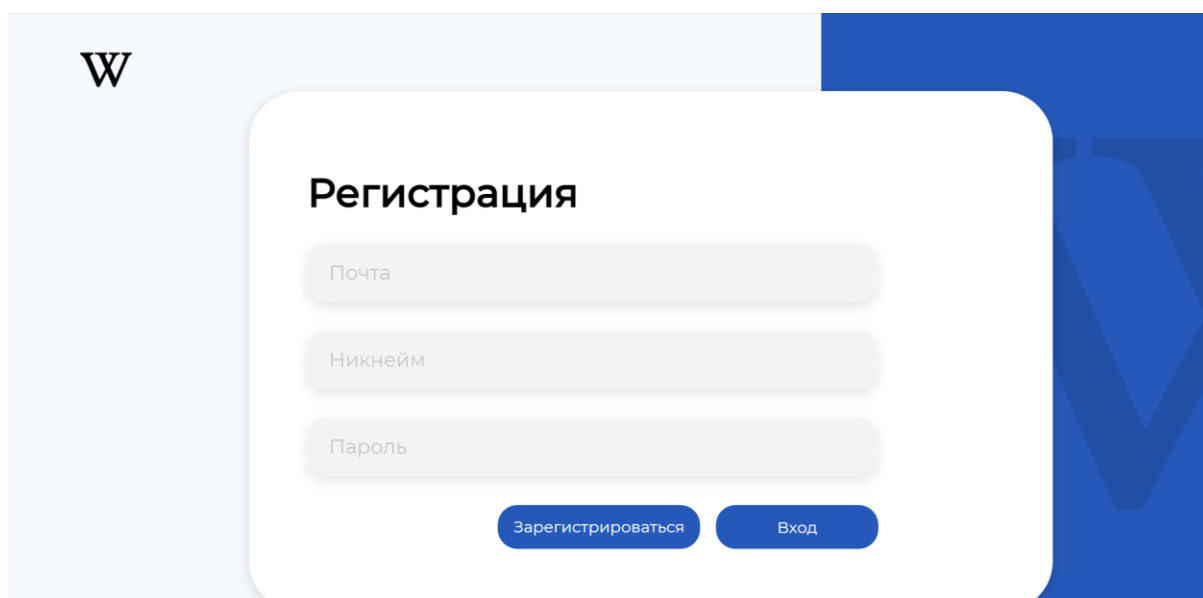


Рисунок 21 - Страница регистрации нового аккаунта

На странице тематик, изображенной на рисунке 22, можно просмотреть список всех тематик статей. И на странице с выбранной тематикой, представленной на рисунке 23, можно просмотреть списков всех статей по этой теме.

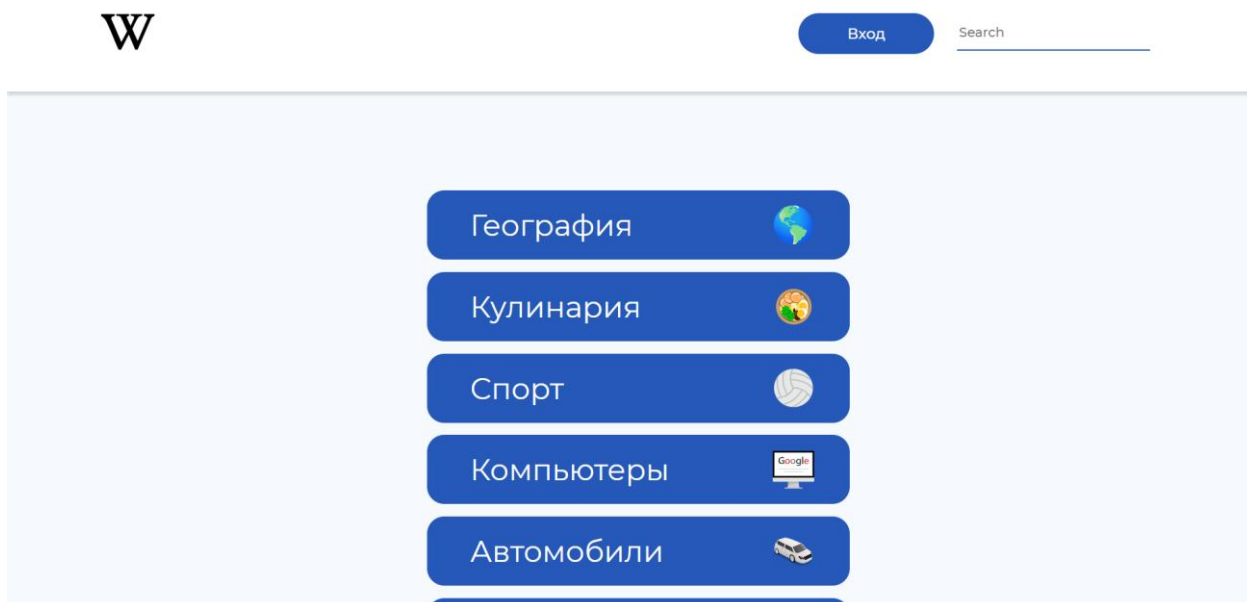


Рисунок 22 - Страница для просмотра всех тематик

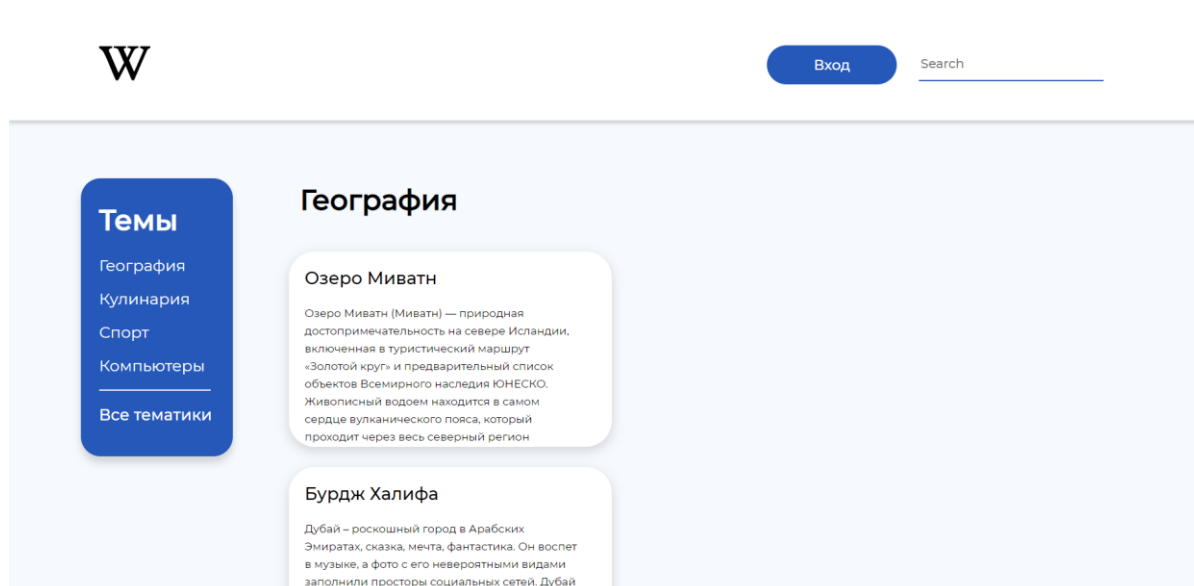


Рисунок 23 - Страница для просмотра статей выбранной тематики

После перехода на страницу со статьей, представленной на рисунке 24, пользователь может изучить материалы статьи, скачать текст, скачать аудио, а также прослушать статью на сайте.

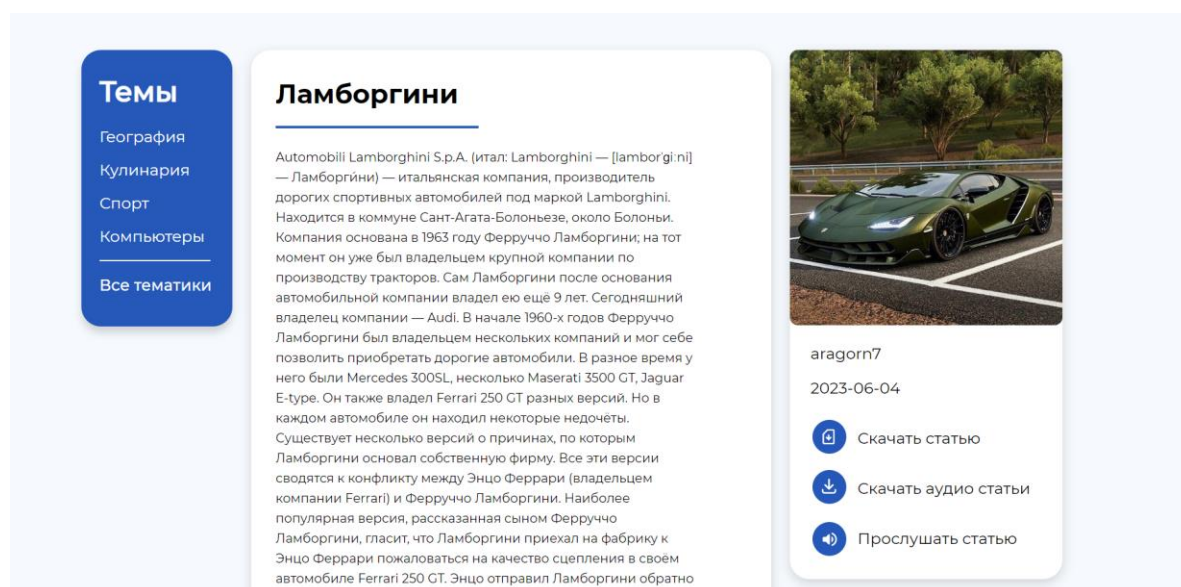


Рисунок 24 - Страница со статьей

После авторизации пользователю становится доступной возможность добавлять, редактировать и удалять свои статьи, а также редактировать профиль. Для этого на странице профиля необходимо нажать на нужную кнопку. Страница профиля представлена на рисунке 25.

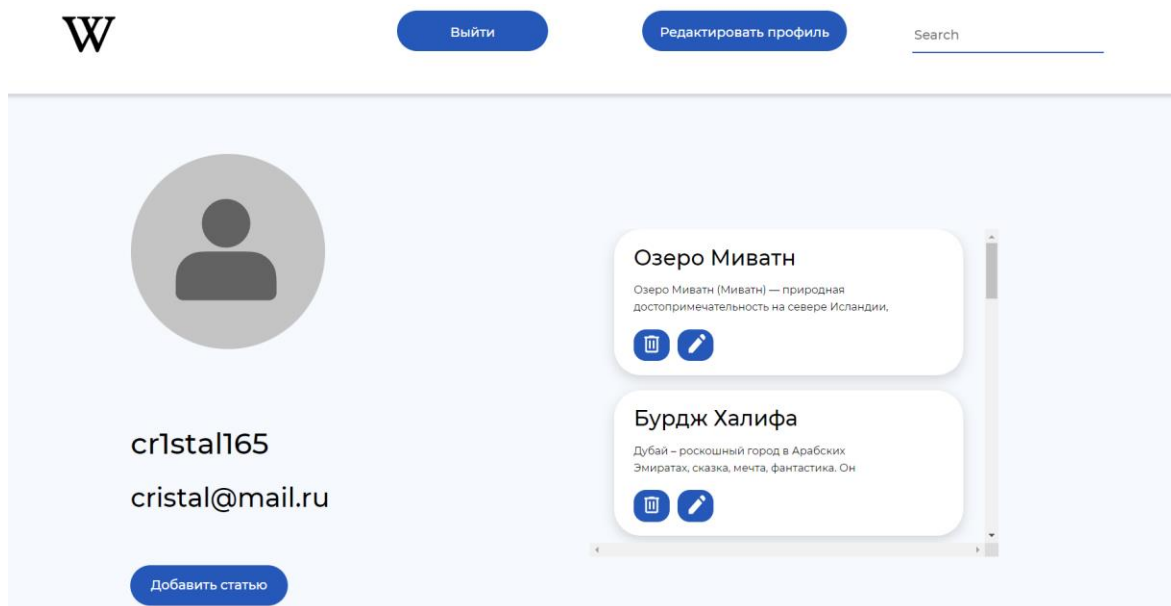


Рисунок 25 - Страница профиля

После нажатия на кнопку «Добавить статью» происходит переход на страницу с формой для добавления, изображенной на рисунке 26, где необходимо ввести данные для статьи.

Рисунок 26 - Страница с добавлением статьи

После нажатия на кнопку редактирования происходит переход на страницу с формой для изменения статьи, изображенной на рисунке 27, где необходимо обновить данные.



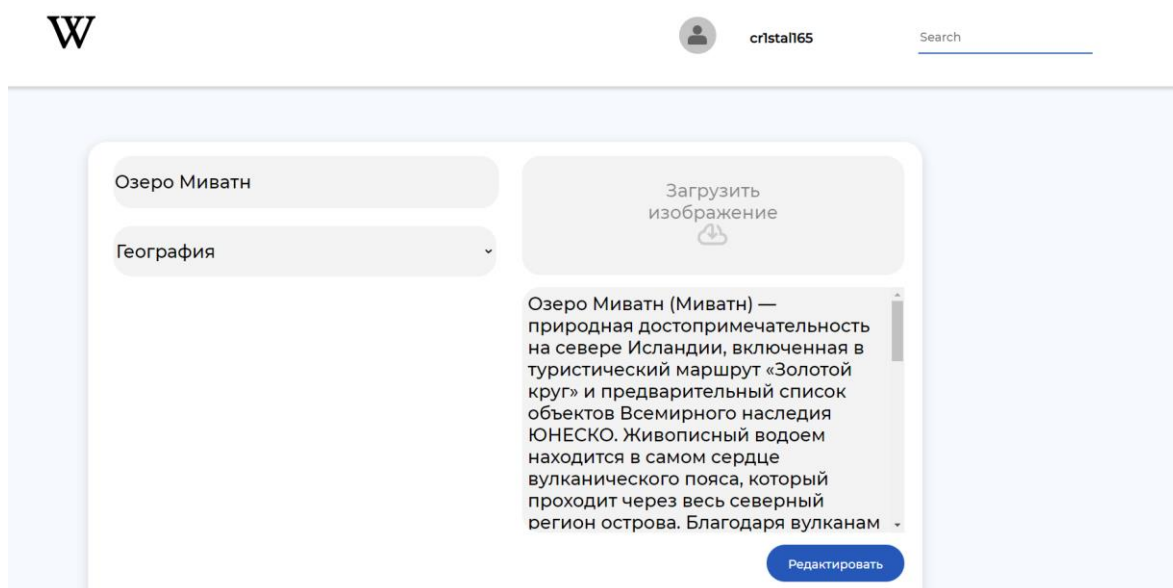


Рисунок 27 - Страница с редактированием статьи

После нажатия на кнопку «Редактировать профиль» происходит переход на страницу с формой для изменения данных пользователя, изображенной на рисунке 28, где необходимо ввести новые данные.

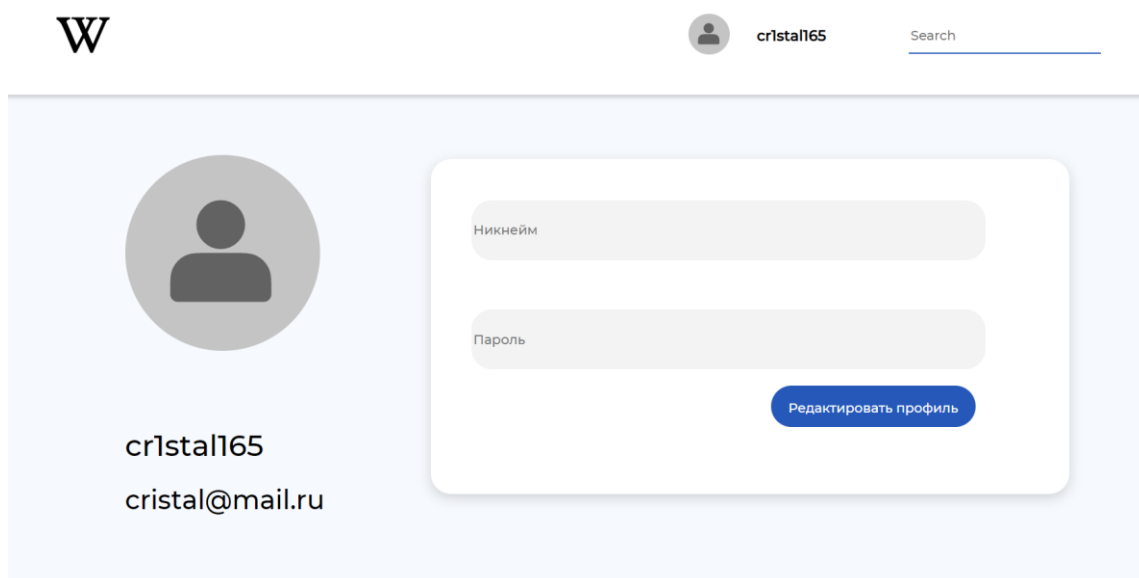


Рисунок 28 - Страница с редактированием профиля

### 3.4.2 Для администратора

Администратору предлагаются возможности для работы с тематиками и статьями. В качестве примера рассмотрим работу с тематиками. На странице

тематик, изображенной на рисунке 29, выводится список доступных тематик, которые можно удалить, изменить или добавить.

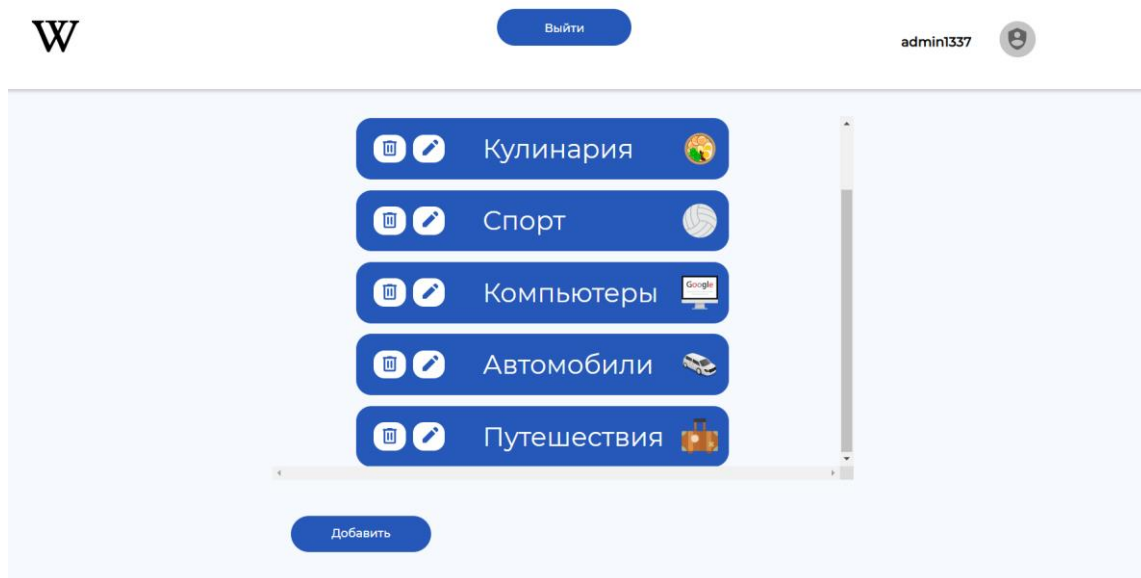


Рисунок 29 - Страница со списком тематик для администратора

При нажатии на кнопку «Добавить» предлагается ввести данные для новой тематики на странице добавления, представленной на рисунке 30.

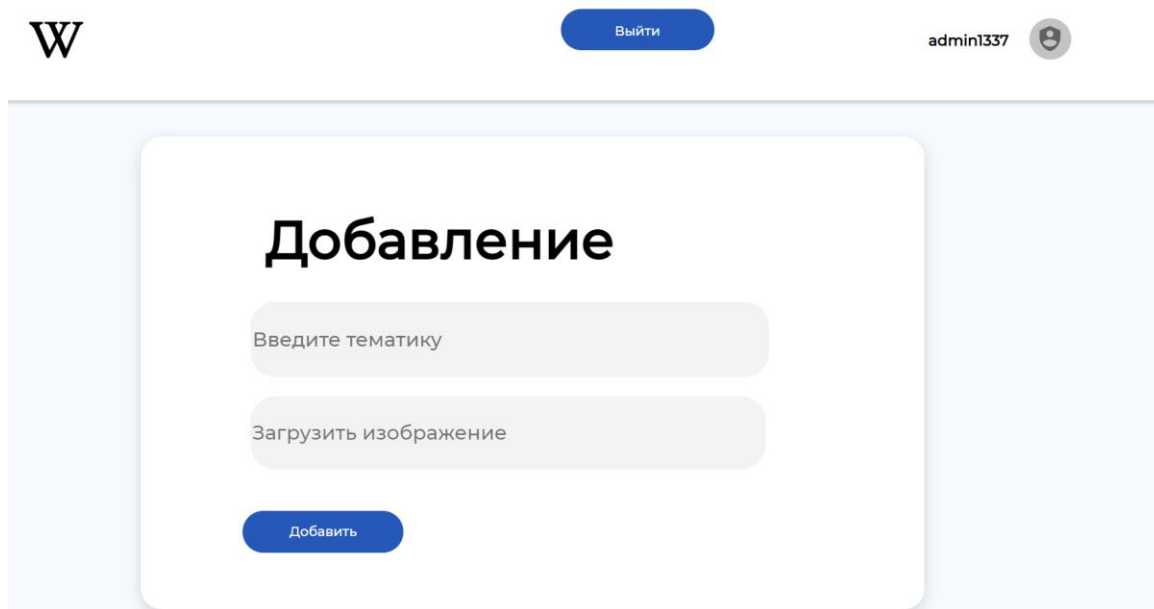


Рисунок 30 - Страница с добавлением новой тематики

При нажатии на кнопку редактирования предлагается изменить данные для тематики на странице, представленной на рисунке 31.

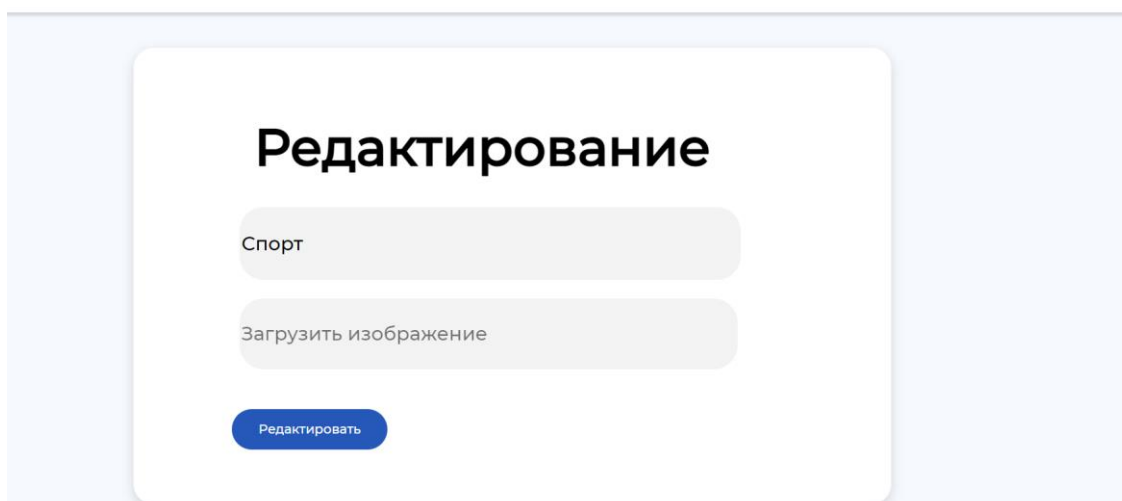


Рисунок 31 - Страница для изменения тематики

### 3.5 Тестирование

Для тестирования разработанной системы были проведены тесты основных типов:

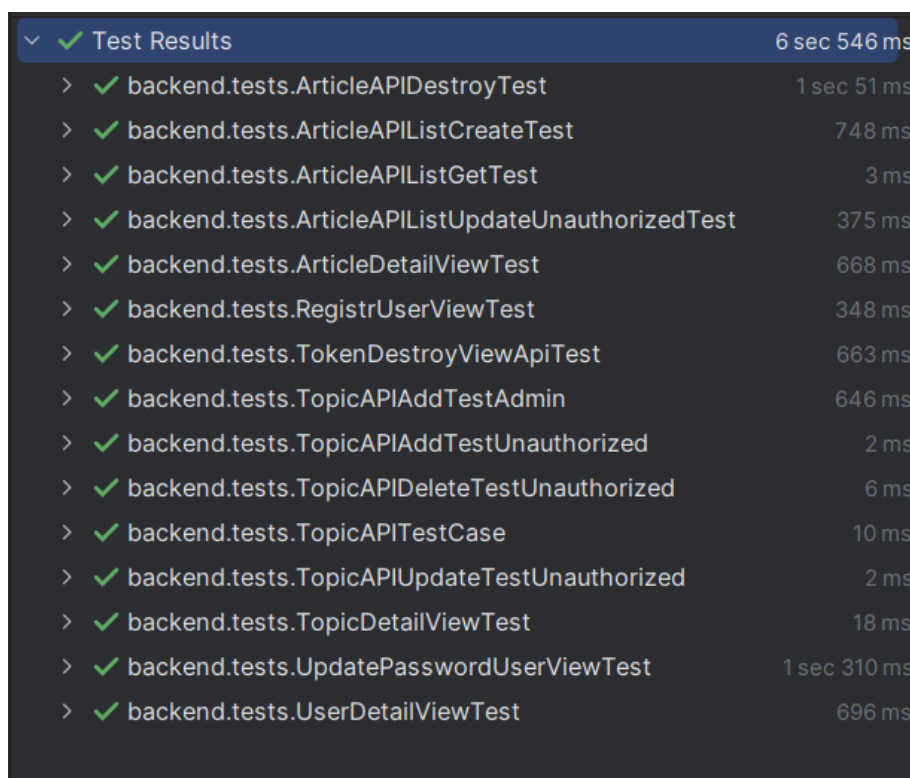
- Модульное тестирование (unit-тесты)
- Тесты, связанные с изменениями (дымовое тестирование);
- Функциональные тесты (GUI-тестирование).

#### 3.5.1 Модульное тестирование

Unit-тесты являются частью процесса разработки программного обеспечения и используются для проверки отдельных модулей или компонентов программы. Они предназначены для тестирования функциональности отдельных единиц кода, таких как отдельные классы, методы или функции.

Целью unit-тестов является проверка, что каждая единица программного кода работает правильно в изоляции от других компонентов системы. Unit-тесты проверяют, что методы возвращают ожидаемые результаты при заданных входных данных, а также что ошибочные ситуации обрабатываются корректно.

На рисунке 32 представлены результаты unit-тестов серверной части приложения.



✓ Test Results	6 sec 546 ms
> ✓ backend.tests.ArticleAPIDestroyTest	1 sec 51 ms
> ✓ backend.tests.ArticleAPIListCreateTest	748 ms
> ✓ backend.tests.ArticleAPIListGetTest	3 ms
> ✓ backend.tests.ArticleAPIListUpdateUnauthorizedTest	375 ms
> ✓ backend.tests.ArticleDetailViewTest	668 ms
> ✓ backend.tests.RegistrUserViewTest	348 ms
> ✓ backend.tests.TokenDestroyViewApiTest	663 ms
> ✓ backend.tests.TopicAPIAddTestAdmin	646 ms
> ✓ backend.tests.TopicAPIAddTestUnauthorized	2 ms
> ✓ backend.tests.TopicAPIDeleteTestUnauthorized	6 ms
> ✓ backend.tests.TopicAPITestCase	10 ms
> ✓ backend.tests.TopicAPIUpdateTestUnauthorized	2 ms
> ✓ backend.tests.TopicDetailViewTest	18 ms
> ✓ backend.tests.UpdatePasswordUserViewTest	1 sec 310 ms
> ✓ backend.tests.UserDetailViewTest	696 ms

Рисунок 32 - Результаты unit-тестов

### 3.5.2 Дымовое тестирование

Дымовое тестирование (smoke testing) — это вид тестирования программного обеспечения, который выполняется для быстрой проверки основной функциональности системы или компонента после внесения изменений или внедрения новой версии. Основная цель дымового тестирования — обнаружить критические ошибки или проблемы, которые могут повлиять на работоспособность системы в целом.

В ходе дымового тестирования выполняются базовые операции или сценарии, которые предполагаются как наиболее важные и часто используемые пользователем.

В таблицах 1, 2 представлены результаты дымового тестирования для основных сценариев пользователя и администратора.

Таблица 1 - Результаты дымового тестирования для пользователя

Тестовый сценарий	Результат теста
Регистрация	Пройден
Авторизация	Пройден
Просмотр всех тематик	Пройден
Просмотр статей с выбранной тематикой	Пройден
Просмотр конкретной статьи	Пройден
Скачивание текста статьи	Пройден
Скачивание аудио статьи	Пройден
Прослушать статью	Пройден
Добавить статью	Пройден
Редактировать статью	Пройден
Удалить статью	Пройден
Редактировать профиль	Пройден

Таблица 2 - Результаты дымового тестирования для администратора

Тестовый сценарий	Результат теста
Авторизация	Пройден
Просмотр списка тематик	Пройден
Просмотр списка статей	Пройден
Добавление тематики	Пройден
Редактирование тематики	Пройден
Удаление тематики	Пройден
Редактирование статьи	Пройден
Удаление статьи	Пройден

### 3.5.3 GUI-тестирование

Тестирование пользовательского интерфейса (GUI-тестирование) — это процесс тестирования элементов управления в приложении, который помогает убедиться, что интерфейс соответствует ожидаемой функциональности.

Задача проведения GUI-тестов — убедиться, что в функциях пользовательского интерфейса отсутствуют дефекты.

В таблице 3 представлена часть результатов тестирования пользовательского интерфейса. В ней отражены тестовые сценарии для неавторизованного пользователя.

Таблица 3 - Результаты GUI-тестирования для пользователя

Тестовый сценарий	Ожидаемый результат	Статус теста
Нажатие на кнопку «Все тематики» на главной странице	Переход на страницу со всеми тематиками	Пройден
Нажатие на заголовок статьи на главной странице	Переход на страницу со статьей	Пройден
Нажатие на логотип	Переход на главную страницу	Пройден
Нажатие на кнопку «Вход» в шапке	Переход на страницу входа	Пройден
Нажатие на кнопку «Зарегистрироваться» на странице входа	Переход на страницу регистрации	Пройден
Нажатие на кнопку «Войти» на странице входа	Переход на страницу профиля	Пройден
Нажатие на кнопку «Добавить статью» на странице профиля	Переход на страницу с добавлением статьи	Пройден
Нажатие на кнопку «Редактировать профиль» на странице профиля	Переход на страницу с редактированием профиля	Пройден
Нажатие на кнопку с редактированием статьи на странице профиля	Переход на страницу с редактированием статьи	Пройден

## **Заключение**

В ходе выполнения данного проекта были успешно достигнуты все поставленные перед разработчиками задачи. Была разработана сетевая энциклопедия под названием «WikiWorld», которая предоставляет полезную информацию и обладает социальной значимостью. Система обеспечивает доступ к знаниям, полезным в повседневной жизни и развитии пользователей.

Важными аспектами разработки были качество и достоверность информации, а также удобство использования и доступность энциклопедии для пользователей всех возрастов и уровней подготовки. Было предусмотрено несколько функциональностей, позволяющих пользователям взаимодействовать с системой.

Пользователи имеют возможность просматривать статьи и тематические подборки, а также слушать статьи в аудиоформате. Авторизованные пользователи могут участвовать в создании и редактировании статей, а администраторы имеют возможность редактировать и удалять контент. Кроме того, система предоставляет возможность скачивания аудиофайлов и текстовых материалов со статей, а также осуществляет поиск по статьям.

Итоги разработки, проверенные в ходе тестирования, позволяют достигнуть поставленных заказчиком целей и решают сформулированные в начале разработки задачи.

## **Список использованных источников**

1. Что такое MVC: рассказываем простыми словами [Электронный ресурс]. — Режим доступа: <https://ru.hexlet.io/blog/posts/chto-takoe-mvc-rasskazyvaem-prostymi-slovami>. — Заглавие с экрана. — (Дата обращения: 15.04.2023).
2. Backend разработка [Электронный ресурс]. — Режим доступа: <https://mobile-erp.ru/backend-razrabotka/>. — Заглавие с экрана. — (Дата обращения: 15.04.2023).
3. Фронтенд [Электронный ресурс]. — Режим доступа: <https://www.gorkilib.ru/events/frontend>. — Заглавие с экрана. — (Дата обращения: 16.04.2023).
4. Система управления объектно-реляционными базами данных PostgreSQL [Электронный ресурс]. — Режим доступа: <https://webcreator.ru/technologies/webdev/postgresql>. — Заглавие с экрана. — (Дата обращения: 20.04.2023).
5. Тестирование API с помощью Swagger: особенности и преимущества [Электронный ресурс]. — Режим доступа: <https://blog.ithillel.ua/ru/articles/api-testing-with-swagger>. — Заглавие с экрана. — (Дата обращения: 30.05.2023).