

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий  
Кафедра Информационные системы и технологии  
Специальность 1–40 05 01 Информационные системы и технологии  
Специализация Программирование в Internet

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
КУРСОВОГО ПРОЕКТА**

по дисциплине «Программирование в Internet»  
Тема Web-приложение «Форум ItHub»

Исполнитель  
студентка 4 курса группы 2

\_\_\_\_\_  
подпись, дата

Ю.С. Гринцевич  
инициалы и фамилия

Руководитель  
асистент  
должность, учена степень, ученое звание

\_\_\_\_\_  
подпись, дата

А.В. Харланович  
инициалы и фамилия

Курсовой проект защищен с оценкой \_\_\_\_\_

Руководитель \_\_\_\_\_  
подпись

А.В. Харланович  
инициалы и фамилия

Минск 2023

## Оглавление

<b>Введение .....</b>	<b>3</b>
<b>1 Постановка задачи .....</b>	<b>4</b>
1.1 Обзор аналогов.....	4
<b>2 Проектирование web-приложения .....</b>	<b>10</b>
2.1 Диаграмма вариантов использования .....	10
2.2 Структурная схема приложения.....	11
2.3 Проектирование базы данных.....	11
<b>3 Разработка web-приложения .....</b>	<b>17</b>
3.1 Серверная часть .....	17
3.1.1 Конфигурация .....	18
3.1.2 Реализация REST API. Контроллеры .....	18
3.1.3 Взаимодействие с БД.....	20
3.1.4 JWT. Аутентификация и авторизация .....	20
3.1.5 Обмен сообщениями с клиентом в реальном времени .....	21
3.2 Клиентская часть .....	21
3.2.1 Хранение состояния.....	22
3.2.2 Маршрутизация .....	23
3.2.3 Обмен сообщениями с сервером в реальном времени .....	24
3.3 Развертывание web-приложения на Docker.....	25
<b>4 Тестирование web-приложения.....</b>	<b>29</b>
<b>5 Руководство пользователя .....</b>	<b>33</b>
5.1 Возможности неавторизованного пользователя .....	33
5.2 Регистрация пользователя .....	34
5.3 Аутентификация и авторизация пользователя.....	36
5.3 Возможности авторизованного пользователя.....	37
5.4 Возможности Администратора .....	39
<b>Заключение .....</b>	<b>41</b>
<b>Список используемых источников.....</b>	<b>42</b>
<b>ПРИЛОЖЕНИЕ А.....</b>	<b>43</b>
<b>ПРИЛОЖЕНИЕ Б .....</b>	<b>46</b>
<b>ПРИЛОЖЕНИЕ В.....</b>	<b>47</b>

## Введение

В настоящее время Интернет стал неотъемлемой частью повседневной жизни, предоставляя людям широкие возможности для общения, обмена информацией и развлечений. С развитием социальных сетей, блогов и онлайн-платформ, возрос интерес пользователей к созданию своих собственных сообществ и форумов. Форумы позволяют людям обмениваться идеями, задавать вопросы, находить ответы и налаживать контакты с единомышленниками по различным интересам и профессиональным сферам.

В связи с растущим интересом к форумам и их значимостью в онлайн-сообществах, разработка собственного форума становится актуальной задачей для многих организаций, сообществ и предпринимателей. Форумы позволяют создать взаимодействие и обмен информацией между пользователями, способствуя росту сообщества, улучшению взаимодействия и обмену знаниями.

Для разработки форума будет использована современная комбинация веб-технологий, включающая Node.js и React:

- Node.js – это среда выполнения JavaScript, построенная на движке V8 Chrome. Она позволяет разрабатывать серверные приложения с использованием JavaScript, что обеспечивает единый язык программирования и на стороне клиента, и сервера;

- React – это JavaScript-библиотека для создания пользовательских интерфейсов. Она позволяет разрабатывать компоненты, которые могут быть масштабируемы в ходе последующей разработки.

- Целью данного курсового проекта выбрано веб-приложение «Форум».

На основе требований пользователей будет разработана архитектура форума. Это включает определение структуры, базы данных, модулей и функциональных компонентов, необходимых для обеспечения эффективной работы форума.

Следующим шагом будет разработка форума на основе разработанной архитектуры. Будут использованы современные веб-технологии и методы программирования для создания функционального и привлекательного интерфейса, способного обеспечить удобство использования и удовлетворение потребностей пользователей, включая авторизированных и неавторизированных.

После завершения разработки форума будет проведено тестирование его функциональности, производительности и безопасности. Будут выявлены возможные ошибки, недочеты и улучшения, которые будут внесены в процессе оптимизации форума, например валидация входящих данных.

## 1 Постановка задачи

В процессе анализа задач, поставленных в данном курсовом проекте, были рассмотрены различные web-приложения, которые могут быть использованы для их решения.

### 1.1 Обзор аналогов

Для эффективной разработки приложения важно провести анализ уже существующих аналогов. Этот анализ поможет выявить основные требования к разрабатываемому приложению:

- Удобный и интуитивно понятный интерфейс: Форум должен иметь простой и понятный интерфейс, который обеспечивает легкую навигацию и быстрый доступ к основным функциям. Пользователи должны легко находить нужные разделы, темы и сообщения, а также иметь возможность удобно взаимодействовать с другими пользователями;

- Форум должен предоставлять различные функции для общения и обмена информацией между пользователями. Возможности отправки сообщений, создания постов, комментирования, обсуждения и добавления файлов должны быть удобными и интуитивно понятными;

Первым аналогом было выбрано web-приложение «Reddit» [1].

На главной странице виден основной функционал форума:

- Вход и регистрация: на главной странице «Reddit» присутствуют элементы, позволяющие пользователям войти в свои аккаунты или зарегистрироваться новым пользователям. Это обеспечивает возможность управления контентом, подписки на интересные разделы и взаимодействия с другими участниками форума.

- Случайные посты: «Reddit» отображает популярные в данный момент времени посты на главной странице. Это позволяет пользователям быть в курсе актуальных обсуждений и искать интересные темы для участия.

- Популярные сообщества: «Reddit» также предоставляет список популярных сообществ на главной странице. Это позволяет пользователям быстро найти сообщества, соответствующие их интересам, и присоединиться к ним для общения и обмена своими мыслями по заданной тематике.

Анализ «Reddit» помогает выделить некоторые основные функциональные элементы форума, которые могут быть применены и в разрабатываемом приложении. Важно учитывать удобство использования, легкость навигации и доступность информации при разработке форума, чтобы обеспечить положительный опыт пользователей. Пользовательский интерфейс форума должен быть интуитивно понятным, с простым и понятным навигационным меню, позволяющим пользователям легко перемещаться по разделам и темам обсуждений.

Кроме того, важно предусмотреть возможность пользовательского взаимодействия и обмена информацией. Это может включать функции комментирования, лайки или общий чат.

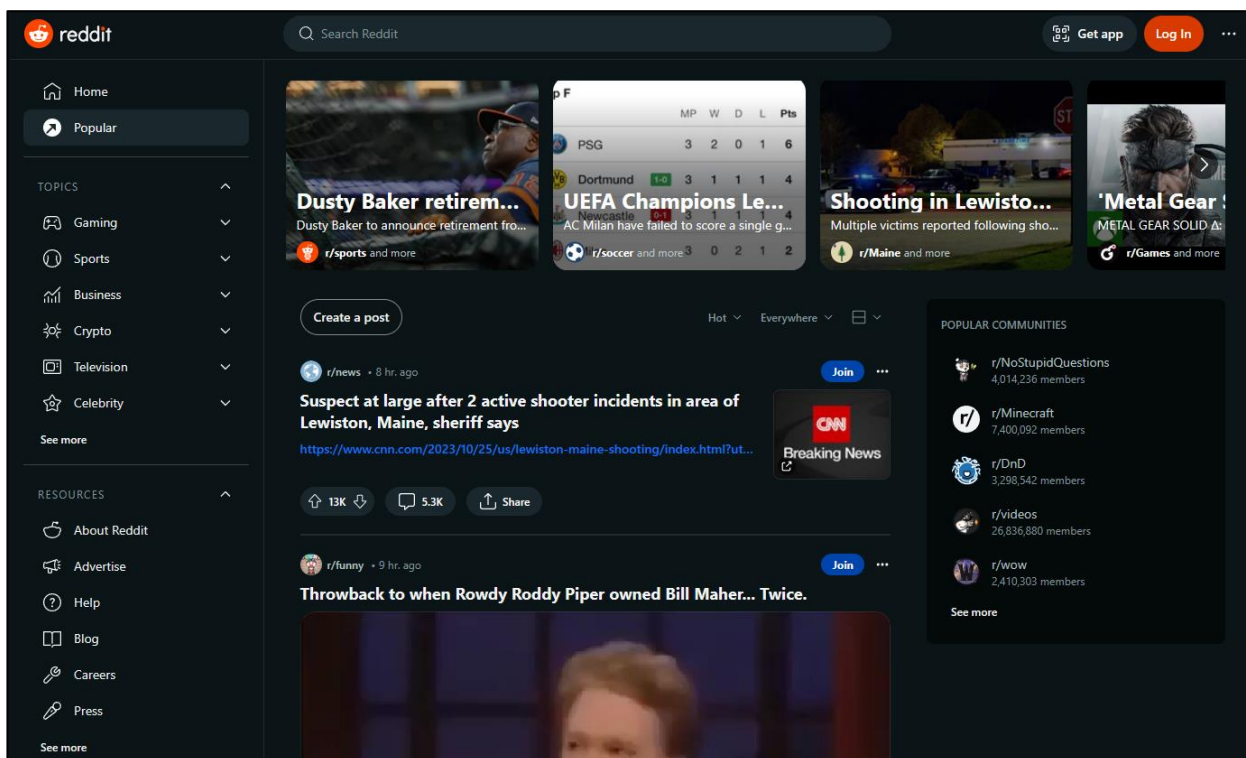


Рисунок 1.1 – Главная страница «Reddit»

При нажатии на название обсуждения открывается страница, на которой видны комментарии под заданной темой. Это позволяет пользователям участвовать в обсуждении, оставлять свои комментарии и отвечать на комментарии других участников.

Страница с комментариями должна быть удобной для чтения и навигации. Комментарии могут быть организованы в виде древовидной структуры, где каждый комментарий может иметь один или несколько ответов под ним. Это позволяет легко отслеживать иерархию комментариев и участвовать в конкретных обсуждениях.

Страница с комментариями также может содержать элементы управления, позволяющие пользователям фильтровать комментарии по различным параметрам, таким как популярность, последние комментарии или активность авторов, количество просмотров или количество лайков. Это помогает пользователям быстро находить интересные и актуальные комментарии в рамках обсуждения. Данная страница, демонстрирующая продвижение обсуждения на основе статистика составленной на основании действий пользователей представлена на рисунке 1.2.

Преимуществами данного web-приложения являются:

- разнообразие тем и сообществ: «Reddit» предлагает широкий спектр различных тематических разделов и сообществ, в которых пользователи могут находить интересующие их темы для обсуждения;

- активное сообщество: «Reddit» привлекает миллионы активных пользователей со всего мира. Благодаря этому формируется живое и динамичное сообщество, где можно обмениваться мнениями, делиться опытом и получать ответы на свои вопросы.

- голосование и рейтинг комментариев: «Reddit» предлагает систему голосования, где пользователи могут оценивать комментарии и посты других

участников. Это позволяет выделить наиболее полезные и интересные комментарии, которые получают большую видимость и рейтинг;

– анонимность и конфиденциальность: «Reddit» предоставляет возможность пользователям оставаться анонимными, если они этого желают. Это может быть важным фактором для тех, кто хочет свободно выражать свое мнение или делиться информацией, не раскрывая свою личность;

– многообразие контента: на «Reddit» можно найти различные форматы контента, включая текстовые посты, изображения, видео и ссылки на другие источники.

Недостатками данного приложения являются:

– интерфейс и навигация: Некоторые пользователи могут считать интерфейс "Reddit" сложным и запутанным. Навигация по разделам и поиск нужной информации может потребовать времени и привыкания;

– зависимость от сообщества: Опыт использования "Reddit" в значительной степени зависит от активности и качества сообществ, которые формируются на конкретной платформе.

Также для сравнения было выбран форум «4PDA» [2]. Интерфейс этого сервиса представлен на рисунке 1.2.

На главной странице данного сервиса можно увидеть случайны обсуждения сайта, меню, строку поиска и кнопки для перехода на страницу авторизации.

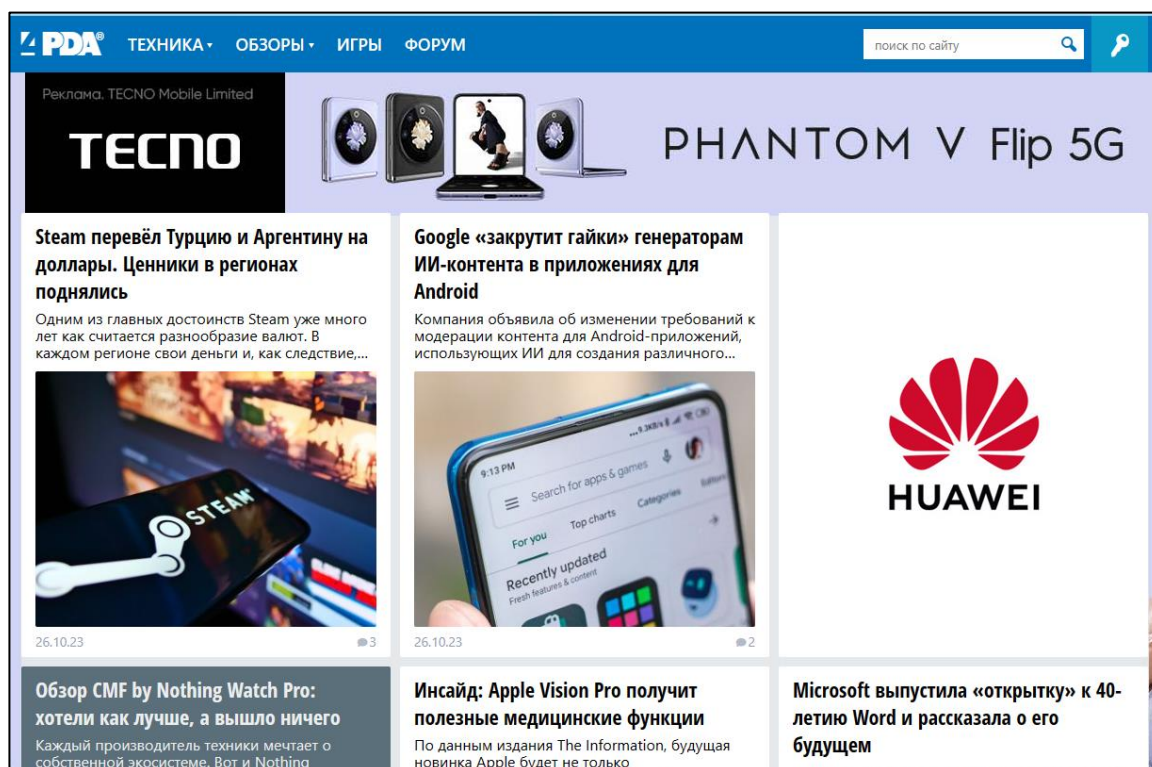


Рисунок 1.2 – Главная страница «4PDA»

При просмотре какого-либо поста мы можем увидеть текст самого поста, комментарии и лайки к посту. Так же если в посте идет речь о каком-то сравнении, то предоставляются также диаграммы для просмотра разницы с аналогами.

Преимуществами данного web-приложения являются:

- большое сообщество: 4PDA имеет огромную активную пользовательскую базу, что создает возможность получить обширную информацию и советы от опытных пользователей по заданным тематикам форума в комментариях;
- разнообразие разделов;
- качественный контент;
- актуальность и обновления.

Недостатками данного приложения являются:

- модерация и контент: в некоторых случаях качество контента на форуме может быть неравномерным;
- интерфейс и оформление: Дизайн и пользовательский интерфейс форума могут казаться немного устаревшими и не всегда удобными в использовании;
- отсутствие общего чата.

Третьим аналогом было выбрано web-приложение «Pikabu» [3].

Интерфейс этого сервиса представлен на рисунке 1.3.

На главной странице виден основной функционал форума:

- Вход и регистрация: на главной странице «Pikabu» также присутствуют элементы, позволяющие пользователям войти в свои аккаунты или зарегистрироваться новым пользователям. Это обеспечивает возможность управления контентом, подписки на интересующие разделы и взаимодействия с другими участниками форума.

- Популярные посты: «Pikabu» отображает популярные посты на главной странице. Это позволяет пользователям быть в курсе актуальных обсуждений и искать интересные темы для участия.

- Категории и теги: «Pikabu» предлагает разделение постов по категориям, таким как юмор, истории, политика, наука и техника и другие. Кроме того, пользователи могут добавлять теги к своим постам, чтобы указать более конкретную тематику. Это позволяет пользователям быстро найти посты, соответствующие их интересам, и фильтровать контент по разным параметрам.

Анализ «Pikabu» помогает выделить некоторые дополнительные функциональные элементы форума, которые могут быть применены и в разрабатываемом приложении. Важно учитывать разнообразие контента, которое может привлекать разную аудиторию и удовлетворять разные потребности пользователей. Категории и теги могут помочь организовать контент по тематикам и упростить поиск нужной информации. Пользовательский интерфейс форума должен быть адаптивным и поддерживать различные форматы контента, включая текст, изображения, видео и ссылки.



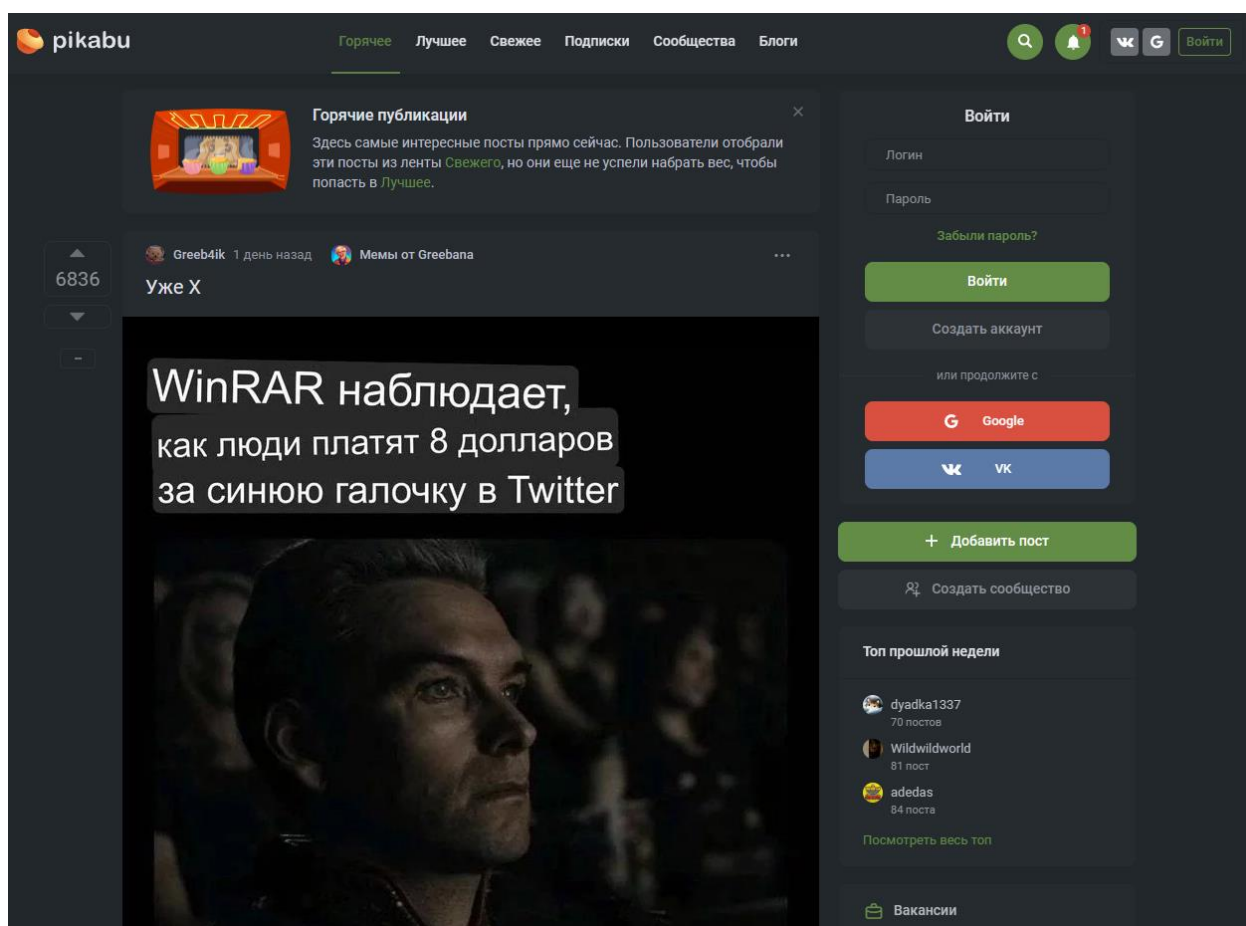


Рисунок 1.3 – Главная страница «Pikabu»

При нажатии на название обсуждения открывается страница, на которой видны комментарии под заданной темой. Это позволяет пользователям участвовать в обсуждении, оставлять свои комментарии и отвечать на комментарии других участников. Страница с комментариями должна быть удобной для чтения и навигации. Комментарии могут быть организованы в виде древовидной структуры, где каждый комментарий может иметь один или несколько ответов под ним. Это позволяет легко отслеживать иерархию комментариев и участвовать в конкретных обсуждениях. Страница с комментариями также может содержать элементы управления, позволяющие пользователям фильтровать комментарии по различным параметрам, таким как популярность, последние комментарии или активность авторов, количество просмотров или количество лайков. Это помогает пользователям быстро находить интересные и актуальные комментарии в рамках обсуждения.

Преимуществами данного web-приложения являются:

- Юмор и развлечения: «Pikabu» известен своим юмористическим и развлекательным контентом, который может поднять настроение и расслабиться. На форуме можно найти множество смешных историй, анекдотов, мемов, видео и других форматов, которые могут вызвать улыбку или смех.

- Обратная связь и поддержка: «Pikabu» также предоставляет возможность пользователям получать обратную связь и поддержку от других участников форума.

Недостатками данного приложения являются:



– Модерация и цензура: «Pikabu» подвергается критике за свою модерацию и цензуру контента. Некоторые пользователи жалуются на то, что их посты или комментарии удаляются без объяснения причин или нарушаются правила форума. Это может приводить к конфликтам в сообществе.

– Токсичность и троллинг: «Pikabu» также страдает от токсичности и троллинга некоторых пользователей, которые могут оскорблять, унижать или провоцировать других участников форума. Это может портить атмосферу общения и отпугивать потенциальных пользователей.

В разработке собственного web-приложения были учтены и исправлены все недостатки вышеописанных аналогов. Так же были учтены и преимущества. Были выделены основные требования к приложению форум. Было уделено внимание его масштабируемости, чтобы оно могло обслуживать большое количество пользователей и масштабироваться по мере необходимости. Особое внимание было уделено безопасности, включая защиту от взлома, атак на сервер и утечки пользовательской информации.

Функционально web-приложение должно:

- обеспечивать возможность регистрации и авторизации;
- поддерживать роли администратора и пользователя;
- позволять пользователям создавать посты;
- позволять пользователям создавать темы для поста при помощи тегов;
- позволяет оставить комментарий под постом;
- позволяет редактировать и удалять посты их владельцам;
- позволяет поддерживать связь между авторизованными пользователями форума при помощи общего чата;
- предоставлять возможность оценить пост лайком;
- предоставлять возможность подписаться на автора интересующих постов;
- предоставлять возможность поиска поста по названию.

На основе функциональных требований были выделены технические требования к проектируемому web-приложению.

Требования к дизайну интерфейса:

- дизайн должен быть простым и интуитивно понятным для пользователей;
- использованные цвета должны сочетаться между собой
- используемые иконки должны соответствовать контексту использования.
- Требования к безопасности:
  - для защиты от несанкционированного доступа к базе данных использовать шифрование паролей пользователей;
  - для защиты от хакерских атак проводить валидацию входных данных.

## 2 Проектирование web-приложения

### 2.1 Диаграмма вариантов использования

Диаграмма использования для приложения форум представляет собой графическое изображение функциональности, которая доступна пользователям при использовании данного приложения. Она помогает визуализировать взаимодействие между пользователями и системой, а также показывает основные действия, которые могут быть выполнены в приложении форума.

Диаграмма использования для приложения представлена на рисунке 2.1.

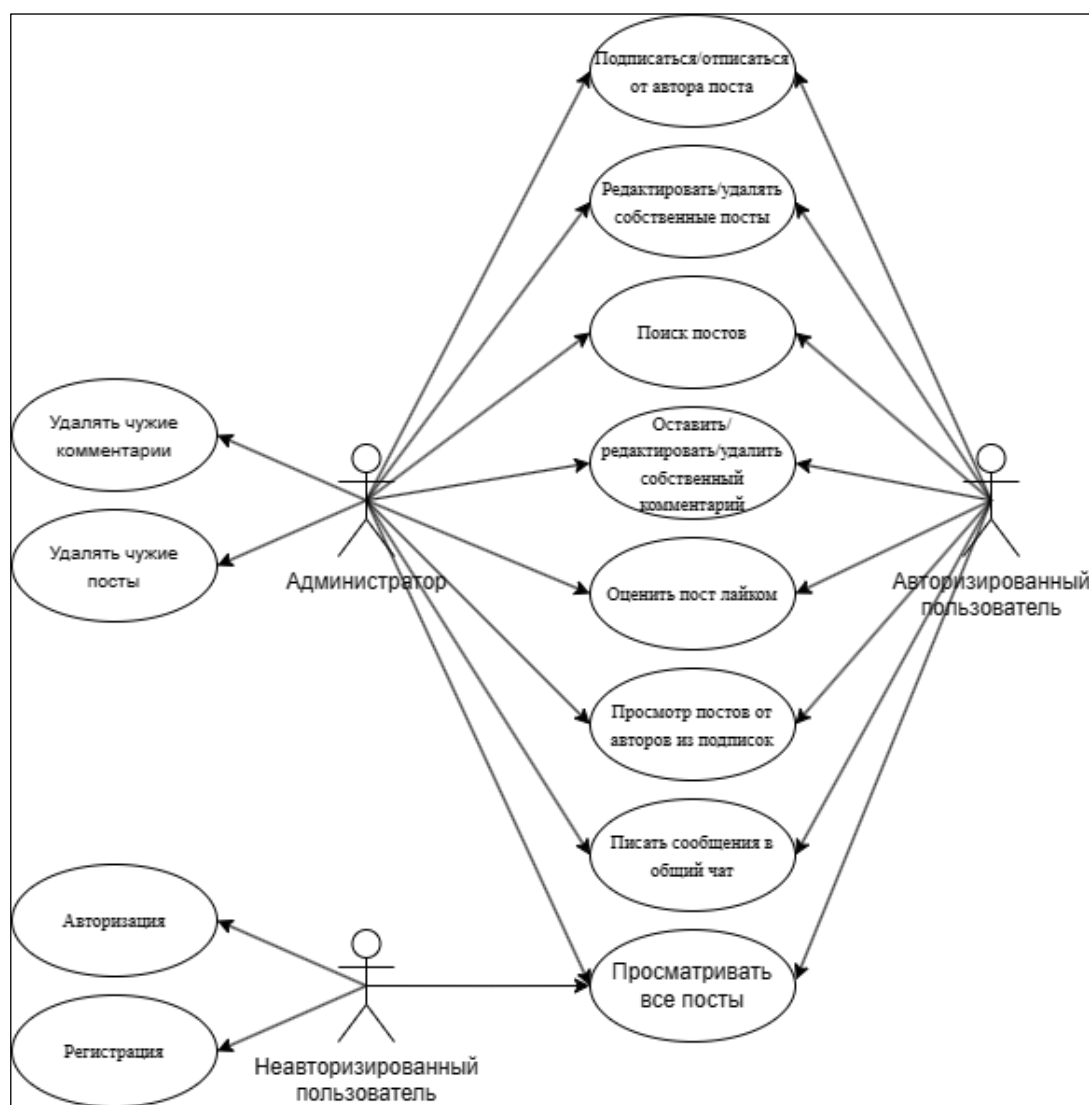


Рисунок 2.1 – Диаграмма вариантов использования

Как видно из диаграммы использования, приложение поддерживает две основные роли: пользователь и администратор. Авторизованный администратор, в отличие от пользователя, имеет доступ к функциям редактирования, удаления чужих обсуждений, а так же, удалять чужие комментарии.

## 2.2 Структурная схема приложения

Клиентская часть:

- Разработана с использованием React и Redux;
- React компоненты отображают пользовательский интерфейс и обрабатывают пользовательские действия;
- Redux хранит состояние приложения и управляет его изменениями;
- Когда пользователь взаимодействует с интерфейсом, React компоненты отправляют запросы к серверной части через API.

Серверная часть:

- Разработана на платформе Node.js с использованием фреймворка Express;
- Express обрабатывает входящие HTTP-запросы от клиента и маршрутизирует их к соответствующим обработчикам;
- При получении запросов, серверная часть взаимодействует с удаленной MongoDB для чтения или записи данных;
- Для взаимодействия с MongoDB используется библиотека Mongoose.
- Удаленная MongoDB:
- MongoDB является базой данных, развернутой на удаленном сервере или хостинге;
- Серверная часть (Node.js и Express) устанавливает соединение с удаленной MongoDB с помощью URI-строки, указывающей на адрес и доступные учетные данные;
- Запросы серверной части отправляются к удаленной MongoDB для выполнения операций чтения и записи данных.

Схема сайта веб приложения для авторизованного и неавторизованного пользователей представлена на рисунке 2.2.

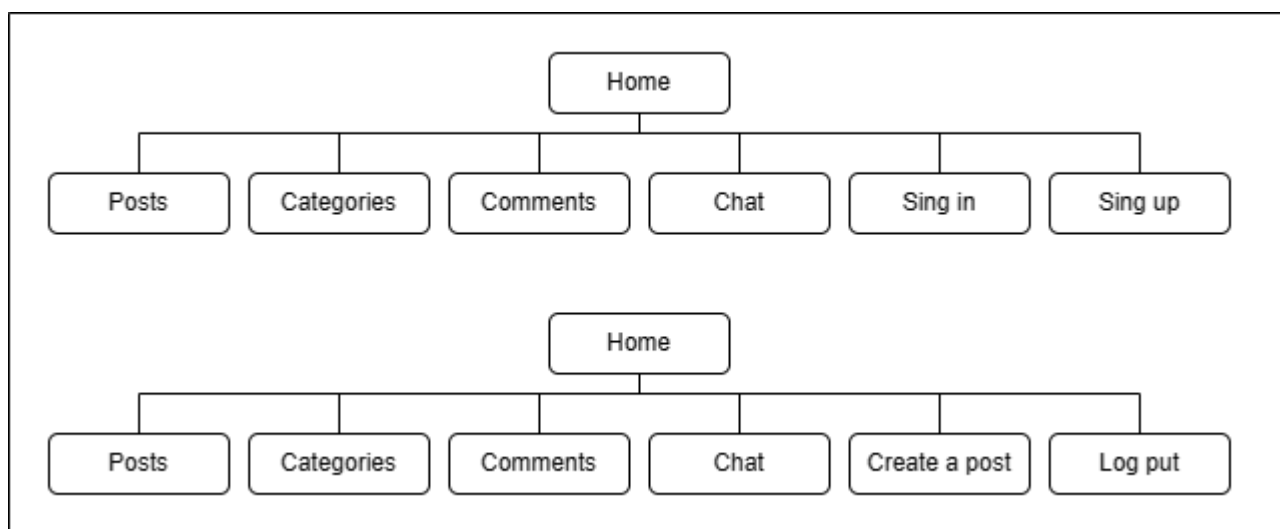


Рисунок 2.1 – Диаграмма вариантов использования

## 2.3 Проектирование базы данных

Важный этап разработки приложения, так как оно строится на основе структурированного хранения и доступа к данным. Правильное проектирование базы

данных позволяет достичь оптимальной производительности, обеспечить безопасность данных и удобство использования приложения. Основной задачей проектирования базы данных является определение структуры, которая соответствует требованиям приложения и обеспечивает эффективное хранение и извлечение данных. Это включает определение сущностей и их атрибутов, установление связей между сущностями и создание таблиц для их представления.

При проектировании базы данных также необходимо учесть правила целостности данных, которые помогают поддерживать точность и надежность данных. Эти правила включают ограничения, проверки и уникальные индексы, которые предотвращают ошибки ввода данных и обеспечивают целостность данных в базе. Важно также учесть масштабируемость базы данных при проектировании, чтобы она могла эффективно обрабатывать растущие объемы данных. Необходимо предусмотреть возможность расширения базы данных в будущем, чтобы она могла легко адаптироваться к увеличивающимся потребностям приложения.

Таким образом, проектирование базы данных является критическим этапом, гарантирующим эффективность и надежность работы приложения, а также способствующим его дальнейшему развитию и масштабируемости. Для хранения данных в приложении используется удаленная база данных.

На рисунке 2.3 изображена схема базы данных программного средства.

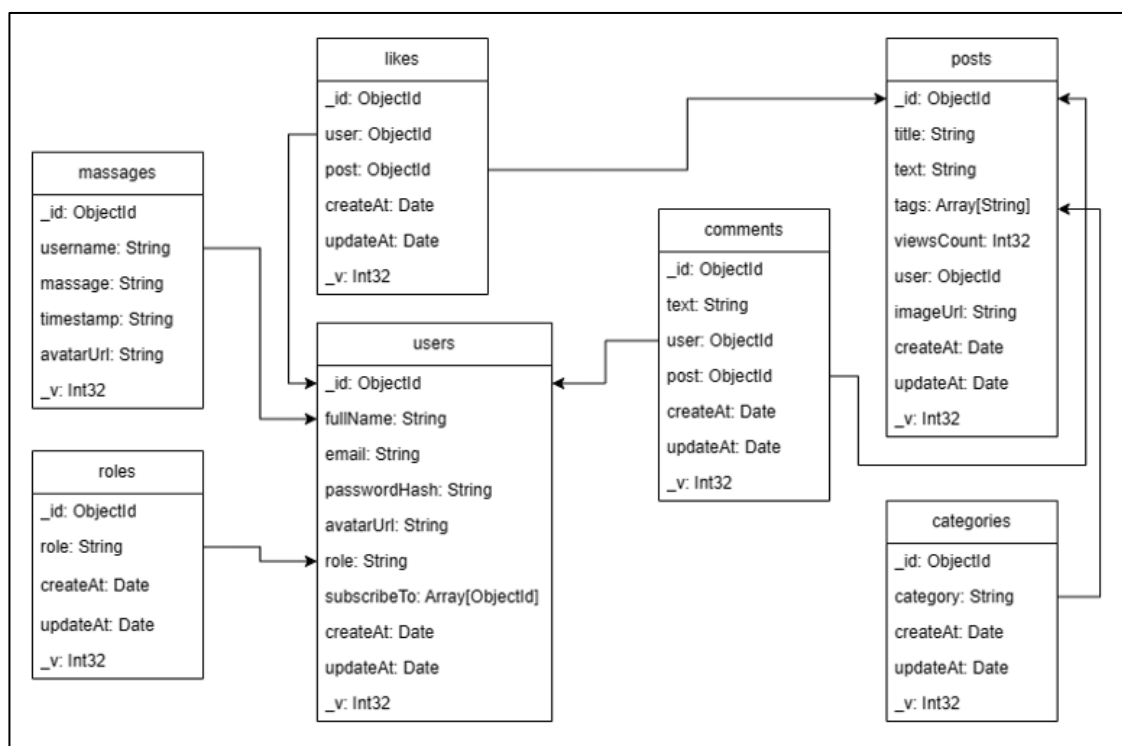


Рисунок 2.3 – Модель базы данных

Далее будет описана каждая таблица базы данных.

В таблице «categories» хранятся все основные разделы форума. Описание структуры таблицы «categories» представлено в таблице 2.1.

Таблица 2.1 – Структура таблицы «categories»

Название столбца	Тип	Описание столбца
_id	ObjectId	Идентификатор категории
category	String	Названии категории
createdAt	Date	Дата создания документа
updatedAt	Date	Дата обновления документа
_v	Int32	Версия документа коллекции

В таблице «comments» хранится информация о комментариях. Описание структуры таблицы «comments» представлено в таблице 2.2.

Таблица 2.2 – Структура таблицы «comments»

Название столбца	Тип	Описание столбца
_id	ObjectId	Идентификатор комментария
text	String	Текст комментария
user	ObjectId	Идентификатор пользователя
post	ObjectId	Идентификатор поста
createdAt	Date	Дата создания документа
updatedAt	Date	Дата обновления документа
_v	Int32	Версия документа коллекции

Таблица «likes» содержит информацию о лайках, которые ставили пользователи определённым постам. Описание структуры наглядно представлено в таблице 2.3.

Таблица 2.3 – Структура таблицы «likes»

Название столбца	Тип	Описание столбца
_id	ObjectId	Идентификатор лайка
user	ObjectId	Идентификатор пользователя
post	ObjectId	Идентификатор поста
createdAt	Date	Дата создания документа
updatedAt	Date	Дата обновления документа
_v	Int32	Версия документа коллекции

В таблице «messages» содержится информация о последних десяти сообщениях в групповой чат. Описание структуры таблицы «messages» представлено в таблице 2.4.

Таблица 2.4 – Структура таблицы «messages»

Название столбца	Тип	Описание столбца
_id	ObjectId	Идентификатор сообщения
username	String	Имя пользователя
message	String	Сообщение от пользователя
timestamp	String	Время отправки сообщения
avatarUrl	String	Ссылка на аватар пользователя
_v	Int32	Версия документа коллекции

В таблице «posts» содержатся посты, созданные пользователем. Описание структуры таблицы «posts» представлено в таблице 2.5.

Таблица 2.5 – Структура таблицы «posts»

Название столбца	Тип	Описание столбца
_id	ObjectId	Идентификатор лайка
title	String	Заголовок поста
text	String	Текст поста
tags	Array[String]	Тэги поста
viewsCount	Int32	Количество просмотров поста
user	ObjectId	Идентификатор пользователя
imageUrl	String	Путь до картинки поста
createdAt	Date	Дата создания документа
updatedAt	Date	Дата обновления документа
_v	Int32	Версия документа коллекции

Таблица «roles» содержит роли, поддерживаемые на форуме. Описание структуры таблицы «roles» представлено в таблице 2.6.

Таблица 2.6 – Структура таблицы «roles»

Название столбца	Тип, ограничение целостности	Описание столбца
_id	ObjectId	Идентификатор роли
createdAt	Date	Дата создания документа
updatedAt	Date	Дата обновления документа
_v	Int32	Версия документа коллекции

Таблица «users» содержит информацию о пользователях форума. Здесь можно увидеть полное имя, почту, хеш пароля, ссылку на картинку профиля и роль на форуме. Описание структуры таблицы «users» представлено в таблице 2.7.

Таблица 2.7 – Структура таблицы «users»

Название столбца	Тип	Описание столбца
_id	ObjectId	Идентификатор пользователя
fullName	String	Полное имя пользователя
Email	String	Почта пользователя
passwordHash	String	Хеш пароля
avatarUrl	String	Ссылка на картинку профиля
role	String	Роль на форуме
subscribedTo	Array[ObjectId]	На кого пользователь подписан
createdAt	Date	Дата создания документа
updatedAt	Date	Дата обновления документа
_v	Int32	Версия документа коллекции

Общий вывод по представленным таблицам базы данных для форума:

- Каждая таблица представляет определенный аспект функциональности форума, такой как категории, комментарии, лайки, сообщения, обсуждения и пользователи.

- Использование ObjectId в качестве идентификаторов позволяет уникально идентифицировать каждую запись и обеспечивает эффективность при выполнении операций поиска и связывания данных.

- Все таблицы содержат общие столбцы, такие как идентификатор (\_id), даты создания (createdAt) и обновления (updatedAt) записи, а также версию документа коллекции (\_v). Эти столбцы позволяют отслеживать и управлять изменениями данных в базе данных.

- Каждая таблица имеет свои уникальные столбцы, соответствующие своей функциональности. Например, таблица «posts» содержит столбцы, такие как заголовок, текст, тэги, количество просмотров и путь к изображению.

- Связь между таблицами может быть установлена с использованием идентификаторов. Например, таблица «comments» имеет столбец «post», который ссылается на идентификатор поста в таблице «posts».

- Организация данных в структурированной форме позволяет эффективно хранить и извлекать информацию, связанную с форумом.

В целом, структура таблиц базы данных обеспечивает удобный и эффективный способ хранения и управления данными для форума. Структура таблиц базы данных в приложении форум представляет собой совокупность моделей данных, которые определяют схему и поведение каждой коллекции в базе данных. Каждая модель соответствует определенному типу данных и содержит поля, описывающие структуру объектов.

Одной из ключевых преимуществ структуры таблиц базы данных является возможность определения связей между коллекциями. Например, модель "Пользователь" может иметь поле, содержащее ссылку на модель "Обсуждение". Это позволяет связывать комментарии или ответы с конкретными обсуждениями и обеспечивает структурированность данных.



Также структура таблиц позволяет определить различные типы полей для хранения различных данных. Например, можно использовать поле типа "Строка" для хранения текстовых комментариев или поле типа "Дата" для сохранения даты и времени публикации комментария. Это способствует эффективному и точному хранению информации.

### 3 Разработка web-приложения

Архитектура приложения форум основана на клиент-серверной модели, где серверная часть отвечает за обработку запросов и хранение данных, а клиентская часть предоставляет пользовательский интерфейс для взаимодействия с сервером.

Взаимодействие между клиентом и сервером осуществляется посредством REST API (Representational State Transfer Application Programming Interface), что обеспечивает простоту и гибкость взаимодействия между различными компонентами приложения.

Для работы с базой данных MongoDB в приложении используется ODM (Object-Document Mapping) Mongoose. Он предоставляет удобный интерфейс для работы с данными MongoDB, позволяя определить модели данных и проводить операции чтения, записи, обновления и удаления объектов в базе данных.

Использование Mongoose позволяет сократить количество кода, необходимого для работы с базой данных, и упростить процесс разработки и поддержки приложения. Он также предоставляет возможности для валидации данных, определения связей между коллекциями и другие функциональности, упрощающие работу с данными.

Такая архитектура и выбранные инструменты позволяют эффективно разрабатывать и масштабировать приложение форум, обеспечивая удобство использования и надежность взаимодействия с данными.

#### 3.1 Серверная часть

Для разработки серверной части приложения использовался фреймворк Express.js. Основная концепция Express.js заключается в определении маршрутов и обработчиков запросов. Маршруты представляют собой URL-адреса, на которые сервер должен реагировать, а обработчики запросов определяют действия, которые должны быть выполнены при обращении к определенному маршруту.

Определение маршрутов в Express.js очень просто и понятно. Можно указать путь к маршруту, тип запроса (GET, POST, PUT, DELETE и т. д.) и соответствующий обработчик запроса. Обработчик представляет собой функцию, которая выполняет необходимые действия при обращении к данному маршруту, например, извлекает данные из базы данных, обрабатывает их и отправляет ответ клиенту.

Express.js также предоставляет множество дополнительных функций и middleware-компонентов, которые облегчают разработку серверной части приложения. Middleware позволяет выполнять определенные операции перед или после обработки запросов, такие как аутентификация, валидация данных, обработка ошибок и другие. Это способствует повышению безопасности и эффективности приложения. Кроме того, Express.js предлагает гибкую настройку сервера. Можно легко определить порт, на котором сервер будет прослушивать запросы, настроить маршруты статических файлов, задать параметры обработки запросов, включая обработку тела запроса, заголовков и других параметров.

### 3.1.1 Конфигурация

Для работы приложения строка подключения к MongoDB, которая располагается на сервере Mongo, предварительно в настройках базы данных нужно включить подключение из любого места на планете. Пример строки подключения и использованием ODM mongoose представлен на листинге 3.1.

```
mongoose.connect('mongodb+srv://admin:admin@cluster0.klivmt.mongodb.net/blog?retryWrites=true&w=majority')
  .then(() => console.log('DB ok'))
  .catch((err) => console.log('DB error', err));
```

Листинг 3.1 – Пример строки подключения

Для запуска и настройки сервера определено создание экземпляра приложения Express, настройка хранилища для загружаемых файлов с использованием Multer, создание HTTPS-сервера с использованием предоставленных сертификатов. Ниже представлен листинг 3.2 с кодом файла index.js.

```
const app = express();import { AppModule } from './app.module';
const upload = multer({ storage });
app.use(express.json());
app.use(cors());
app.use('/uploads', express.static('uploads'));
const server = https.createServer(
  {
    key: fs.readFileSync('./cert/localhost.key', 'utf-8'),
    cert: fs.readFileSync('./cert/localhost.crt', 'utf-8'),
  },
  app
);
```

Листинг 3.2 – Часть файла index.js

Общий смысл этого кода заключается в настройке сервера Express для обработки запросов, разбора JSON-данных, обработки загрузки файлов с использованием Multer, обеспечения поддержки CORS и предоставления статических файлов, а также создания HTTPS-соединения с использованием предоставленных сертификатов.

### 3.1.2 Реализация REST API. Контроллеры

Контроллеры в Express представляют собой обработчики, которые определяют функциональность и логику обработки каждого маршрута (роута) в REST API. Они связываются с определенными URL-адресами и методами запроса, и выполняют необходимые действия, такие как обработка данных, взаимодействие с базой данных и отправка ответов клиенту. На листинге 3.3 приведен пример контроллера для регистрации и авторизации.

```
app.get('/auth/me', checkAuth, UserController.getMe);
```

```
app.post('/auth/login', loginValidation, handleValidationErrors,
UserController.login);
app.post('/auth/register', registerValidation,
handleValidationErrors, UserController.register);
```

### Листинг 3.3 – Маршруты и методы для авторизации и регистрации

Первый маршрут предназначен для получения информации о текущем аутентифицированном пользователе. Перед вызовом обработчика `UserController.getMe` будет выполнена функция промежуточной обработки `checkAuth`, которая проверяет, аутентифицирован ли пользователь.

Второй маршрут предназначен для аутентификации пользователя путем входа в систему. Перед вызовом обработчика `UserController.login` будут выполнены функции промежуточной обработки `loginValidation` (для проверки входных данных) и `handleValidationErrors` (для обработки возможных ошибок валидации). Если данные прошли проверку, вызывается метод `login` из `UserController`, который обрабатывает запрос, проверяет учетные данные пользователя и возвращает результат аутентификации.

Третий маршрут предназначен для регистрации нового пользователя. Перед вызовом обработчика `UserController.register` будут выполнены функции промежуточной обработки `registerValidation` (для проверки входных данных) и `handleValidationErrors` (для обработки возможных ошибок валидации). Если данные прошли проверку, вызывается метод `register` из `UserController`, который обрабатывает запрос, создает нового пользователя и сохраняет его в базе данных. Ниже представлен листинг 3.4 с кодом проверки валидации.

```
export const loginValidation = [
  body('email', 'Неверный формат почты').isEmail(),
  body('password', 'Пароль должен быть минимум 5
символов').isLength({ min: 5 }),
];export const registerValidation = [body('email', 'Неверный
формат почты').isEmail(), body('password', 'Пароль должен быть минимум
5 символов').isLength({ min: 5 }),
  body('fullName', 'Укажите имя').isLength({ min: 3 }),
  body('avatarUrl', 'Неверная ссылка на
аватарку').optional().isURL(),];
```

### Листинг 3.4 – Класс validations

Здесь используется библиотека `express-validator`, которая предоставляет удобные методы для проверки и валидации данных запроса.

`loginValidation` - это массив функций-обработчиков, которые выполняют проверку входных данных при попытке входа в систему. В данном случае, проверяется формат поля `email` (должен быть валидным email-адресом) и длина поля `password` (должна быть не менее 5 символов).

registerValidation - это массив функций-обработчиков, которые выполняют проверку входных данных при регистрации нового пользователя. В данном случае, проверяется формат поля email (должен быть валидным email-адресом), длина поля password (должна быть не менее 5 символов), длина поля fullName (должна быть не менее 3 символов) и опционально ссылка на аватарку avatarUrl (обязательно должна быть валидной URL).

Если входные данные не проходят одну из проверок, будет сгенерировано сообщение об ошибке, которое будет возвращено клиенту с соответствующим статусом ответа.

### **3.1.3 Взаимодействие с БД**

Для того, чтобы взаимодействовать с базой данных, используется ODM Mongoose, предоставляющая удобные инструменты для работы с MongoDB.

При использовании Mongoose, можно легко определить модели данных, которые соответствуют коллекциям в базе данных. Каждая модель содержит схему, которая определяет структуру документов в коллекции. Схема может включать поля, их типы, валидацию, уникальность и другие свойства.

После подключения и определения моделей, можно использовать Mongoose в контроллерах для выполнения операций с базой данных. Например, можно создавать новых пользователей, получать список пользователей, обновлять и удалять пользователей с помощью соответствующих методов модели.

### **3.1.4 JWT. Аутентификация и авторизация**

JWT (JSON Web Token) - это открытый стандарт для создания токенов доступа, которые могут быть переданы и использованы между двумя сторонами в формате JSON. Он чрезвычайно широко используется для аутентификации и авторизации во многих веб-приложениях.

JWT состоит из трех частей: заголовка (header), полезной нагрузки (payload) и подписи (signature). Заголовок содержит информацию о типе токена и используемом алгоритме шифрования. Полезная нагрузка содержит информацию, которую вы хотите передать в токене, например, идентификатор пользователя или его роли. Подпись используется для проверки целостности токена и подтверждения его подлинности.

При регистрации и входе в систему код использовал библиотеку jsonwebtoken для создания и проверки JWT-токенов. В функции register после сохранения пользователя в базе данных с использованием bcrypt для хэширования пароля, токен генерируется с помощью метода jwt.sign(). В функции login также выполняется проверка пароля с использованием bcrypt, и при успешной аутентификации также генерируется и возвращается токен. JWT-токены имеют определенный срок действия, указанный в полезной нагрузке токена. По истечении срока действия токена требуется обновление или повторная аутентификация пользователя.

JWT-токены предоставляют ряд преимуществ, таких как простота использования, переносимость, возможность хранить в них информацию о пользователе и его правах, а также отсутствие необходимости хранить состояние сессии на сервере. Однако, для обеспечения безопасности, важно правильно настроить и защитить ключи подписи, а также учитывать потенциальные уязвимости, такие как утечка токенов или подделка токенов.

### 3.1.5 Обмен сообщениями с клиентом в реальном времени

Здесь рассмотрим, как обмениваться сообщениями с клиентом в режиме реального времени с помощью Socket.IO. Socket.IO является мощным инструментом, позволяющим устанавливать постоянное соединение между сервером и клиентом, чтобы обмениваться данными без необходимости постоянного обновления страницы.

Socket.IO - это библиотека, позволяющая реализовать веб-сокеты в Node.js и веб-браузерах. Веб-сокеты обеспечивают двустороннюю связь между сервером и клиентом, позволяя отправлять и получать данные в режиме реального времени. Прежде чем мы сможем начать обмениваться сообщениями с клиентом, нам нужно создать сервер Socket.IO на стороне сервера. Для этого мы можем использовать существующий сервер, созданный с помощью Express.js, или создать отдельный сервер. Когда клиент устанавливает соединение с сервером, событие `connection` срабатывает на сервере. Мы можем добавить обработчик этого события, чтобы выполнить действия при подключении нового клиента. Socket.IO позволяет обмениваться сообщениями между сервером и клиентом с помощью событий. События могут быть отправлены как с сервера на клиент, так и с клиента на сервер. На сервере мы принимаем сообщение от клиента и отправляем ответное сообщение обратно на клиент событием `message`. Таким образом, мы можем обмениваться сообщениями между клиентом и сервером в режиме реального времени.

### 3.2 Клиентская часть

В клиентской части приложения был использован фреймворк React с применением Redux для управления состоянием приложения. React является популярным и мощным инструментом для создания пользовательских интерфейсов, позволяющим разделить пользовательский интерфейс на компоненты и управлять ими с помощью виртуального DOM (Virtual DOM). Redux, в свою очередь, является библиотекой для управления состоянием приложения, обеспечивающей единообразное хранение данных и простую механику их изменения. Основные преимущества использования React включают:

- Компонентная архитектура: React позволяет разбить пользовательский интерфейс на небольшие и независимые компоненты, что упрощает разработку, тестирование и поддержку кода. Каждый компонент имеет свою внутреннюю логику и отображение, и может быть использован повторно в разных частях приложения.

- Виртуальный DOM: React использует виртуальный DOM, который является легковесным представлением реального DOM. Это позволяет React эффективно обновлять только необходимые части пользовательского интерфейса, минимизируя количество операций обновления и повышая производительность.

- Удобство работы с данными: React облегчает работу с данными и их передачу между компонентами. Он предлагает односторонний поток данных (от верхнего уровня к нижнему), что делает отслеживание изменений и управление состоянием проще и более предсказуемыми.

Redux, с другой стороны, предоставляет единое хранилище для состояния приложения и набор функций для его изменения. Он следует принципу

однонаправленного потока данных, что упрощает отслеживание изменений и обновление пользовательского интерфейса. Основные преимущества использования Redux включают:

- Централизованное хранение данных: Redux хранит все данные приложения в едином хранилище. Это делает управление и синхронизацию состояния между компонентами более простыми и предсказуемыми.

- Предсказуемость и отладка: Изменения состояния происходят только через чистые функции, называемые "редюсерами". Это делает состояние предсказуемым и облегчает отладку, так как все изменения происходят в одном месте.

- Расширяемость: Redux обеспечивает удобный механизм для добавления промежуточного программного обеспечения (middleware), которое может обрабатывать дополнительную логику и позволяет легко расширять функциональность приложения.

Сочетание React и Redux обеспечивает мощный инструмент для разработки сложных пользовательских интерфейсов с удобным управлением состоянием. React обеспечивает эффективное отображение пользовательского интерфейса, в то время как Redux предоставляет единое и предсказуемое хранилище данных. Это позволяет создавать масштабируемые и гибкие приложения, которые легко тестируются и поддерживаются.

### **3.2.1 Хранение состояния**

В Redux состояние приложения хранится в едином объекте, называемом "хранилищем" (store). Хранилище содержит все данные, необходимые для работы приложения, и предоставляет механизм для их обновления и получения.

В Redux состояние является неизменяемым, что означает, что его нельзя изменить напрямую. Вместо этого, для изменения состояния, создается новая копия состояния с помощью чистых функций, называемых "редюсерами" (reducers). Редюсеры принимают текущее состояние и действие (action) в качестве аргументов и возвращают новую копию состояния с обновленными данными. Для работы с Redux в приложении необходимо выполнить следующие шаги:

- Определение действий (actions): Действия представляют собой объекты, которые описывают, что произошло в приложении. Они содержат информацию, необходимую для обновления состояния. Например, действие может быть "добавить пост" и содержать данные нового поста.

- Создание редюсеров (reducers): Редюсеры являются чистыми функциями, которые принимают текущее состояние и действие, и возвращают новую копию состояния с обновленными данными. Редюсеры определяют, как будет изменяться состояние в ответ на действия. Обычно каждый редюсер отвечает за обновление определенной части состояния.

- Создание хранилища (store): Хранилище создается с использованием функции createStore из пакета Redux. Оно принимает корневой редюсер, который объединяет все редюсеры приложения, и опциональные параметры. Хранилище содержит текущее состояние и методы для его обновления и получения.

- Использование состояния в компонентах: для чтения состояния из хранилища и обновления состояния, компоненты могут использовать хуки useSelector и useDispatch из пакета react-redux. Хук useSelector позволяет получить доступ к



определенной части состояния, а хук `useDispatch` предоставляет метод для отправки действий в хранилище.

На листинге 3.5 листинг полного кода «хранилища».

```
import { configureStore } from '@reduxjs/toolkit';
import { postsReducer } from '../slices/posts';
import { authReducer } from '../slices/auth';
const store = configureStore({reducer:{
posts: postsReducer, auth:      authReducer, },});
export default store;
```

### Листинг 3.5 – Хранилище Redux

Функция `configureStore` из пакета `@reduxjs/toolkit` принимает объект с настройками, в котором определен редюсер для каждой части состояния. В данном случае, состояние разделено на две части: `posts` и `auth`. Каждый редюсер обрабатывает соответствующую часть состояния и определяет, как будет происходить обновление состояния в ответ на действия.

После создания хранилища, оно экспортируется по умолчанию и может быть подключено к компонентам `React` с помощью `Provider` из `react-redux`. Компоненты могут использовать хуки `useSelector` и `useDispatch` для чтения состояния и отправки действий в хранилище, соответственно.

Таким образом, данный код создает и настраивает хранилище `Redux`, где `postsReducer` и `authReducer` определяют, как обновлять состояние для функциональности постов и аутентификации.

#### 3.2.2 Маршрутизация

В приложении используется библиотека `react-router-dom` для реализации маршрутизации. Некоторые маршруты в приложении требуют авторизации пользователя, а также наличия определенной роли. При переходе на конкретный маршрут происходит проверка наличия токена в локальном хранилище. Если токен присутствует, пользователь считается авторизованным и получает доступ к определенному функционалу.

Часть реализации маршрутизации, представленная на листинге 3.6, позволяет по токenu получить идентификатор пользователя (`_id`) и использовать эту информацию для получения роли, полного имени и изображения профиля.

```
export const fetchAuth = createAsyncThunk('auth/fetchAuth', async
(params)=>{ const { data } = await axios.post('/auth/login', params);
return data;});
export const fetchRegister =
createAsyncThunk('auth/fetchRegister', async (params) => { const {
data } = await axios.post('/auth/register', params); return data;});
export const fetchAuthMe = createAsyncThunk('auth/fetchAuthMe',
async ()=>{ const { data } = await axios.get('/auth/me');
return data;});
```

### Листинг 3.6 – Подготовленные запросы проверяющие авторизацию

В данном коде определены три асинхронные функции, используемые для выполнения запросов к серверу, связанных с аутентификацией и регистрацией пользователей.

– `fetchAuth` – эта функция выполняет запрос к серверу для аутентификации пользователя. Она использует библиотеку `Axios` для отправки POST-запроса на путь `'/auth/login'` с переданными параметрами `params`. После получения ответа от сервера, извлекается поле `data` и возвращается как результат выполнения функции.

– `fetchRegister` – эта функция выполняет запрос к серверу для регистрации нового пользователя. Она также использует `Axios` для отправки POST-запроса на путь `'/auth/register'` с переданными параметрами `params`. После получения ответа от сервера, извлекается поле `data` и возвращается в качестве результата выполнения функции.

– `fetchAuthMe` – эта функция выполняет запрос к серверу для получения информации о текущем аутентифицированном пользователе. Она использует `Axios` для отправки GET-запроса на путь `'/auth/me'`. После получения ответа от сервера, извлекается поле `data` и возвращается в качестве результата выполнения функции.

Эти функции используются внутри `Redux`-редюсеров с помощью библиотеки `Redux Toolkit`. Они представляют асинхронные операции, которые могут быть диспетчеризованы из компонентов и могут обновлять глобальное состояние `Redux`-хранилища с помощью действий, сгенерированных `Redux Toolkit`.

#### 3.2.3 Обмен сообщениями с сервером в реальном времени

Для реализации двунаправленного соединения с сервером в режиме реального времени используется протокол связи поверх TCP-соединения - `Websocket`. Для работы с протоколом была использована библиотека, упрощающая работу с сокетами `socket.io-client`. При загрузке страницы чата, мы подключаемся к серверу по протоколу `WebSocket` и при успешном подключении подписываемся на событие `message` при возникновении которого, отображаем новое сообщение в компоненте чата. На листинге 3.7 клиентской части кода `Websocket`.

```
import io from "socket.io-client";
export const Chat = () => {
  const [chatMessages, setChatMessages] = useState([]);
  const [newMessage, setNewMessage] = useState("");
  const [socket, setSocket] = useState(null);
  const [userFullName, setUserFullName] = useState("");
  const [userAvatarUrl, setUserAvatarUrl] = useState("");
  useEffect(() => {const newSocket = io("https://localhost:4444");
    setSocket(newSocket);
    newSocket.on("chatMessage", (message) => {
      setChatMessages((prevMessages) => [...prevMessages, message]);});});
```

### Листинг 3.7 – Клиенская часть Websocket

В данном коде определен компонент Chat, который использует библиотеку `socket.io-client` для создания сокетного подключения и обмена сообщениями в режиме реального времени. Компонент Chat содержит следующие состояния:

- `chatMessages`: массив сообщений чата.
- `newMessage`: текущее введенное пользователем сообщение.
- `socket`: объект сокета для установки соединения с сервером.
- `userFullName`: полное имя пользователя.
- `userAvatarUrl`: URL-адрес аватара пользователя.

При монтировании компонента (`useEffect` с пустым массивом зависимостей), устанавливается новый экземпляр сокета, который подключается к адресу `"https://localhost:4444"`. Затем сокету присваивается новое значение с помощью `setSocket`. Затем устанавливается обработчик события `"chatMessage"`.

### 3.3 Развертывание web-приложения на Docker

После окончания разработки веб-приложения было принято решение развернуть данный проект на Docker, т.к. это существенно облегчает запуск данного приложения на других системах.

Docker является удобным способом развертывания приложений. Он позволяет создавать контейнеры, которые обеспечивают изолированное окружение для приложений, независимо от других приложений. Для удобного развертывания мы можем использовать `Dockerfile` – файл, содержащий инструкции по контейнеризации самого приложения. Необходимо добавить `Dockerfile` как для клиентской, так и для серверной части приложения.

Docker image представляет собой шаблон или образ для создания Docker-контейнеров. Контейнеры, в свою очередь, являются экземплярами Docker-образов, которые обеспечивают стандартизацию развертывания приложений и изолируют их от основной инфраструктуры.

Контейнеры позволяют упаковать приложение и все его зависимости (такие как библиотеки, системные утилиты и файлы настройки) в единый образ. Этот образ является портативным и может быть запущен на любой инфраструктуре, где установлен Docker. Контейнеры предоставляют изолированную среду выполнения для приложения, что означает, что они не влияют на основную операционную систему и другие контейнеры, работающие на том же хосте.

Благодаря контейнерам, приложение и его зависимости упаковываются вместе, что упрощает перенос приложения на другую инфраструктуру. Docker позволяет создавать, запускать и управлять контейнерами с помощью простых команд и инструментов. Это обеспечивает консистентное и повторяемое развертывание приложений в различных средах, включая локальные компьютеры разработчиков, серверы и облачные платформы.

После разработки получилось работающее приложение, представленное клиентской и серверной частями. Структура разработанного приложения представлена на рисунке 3.1.

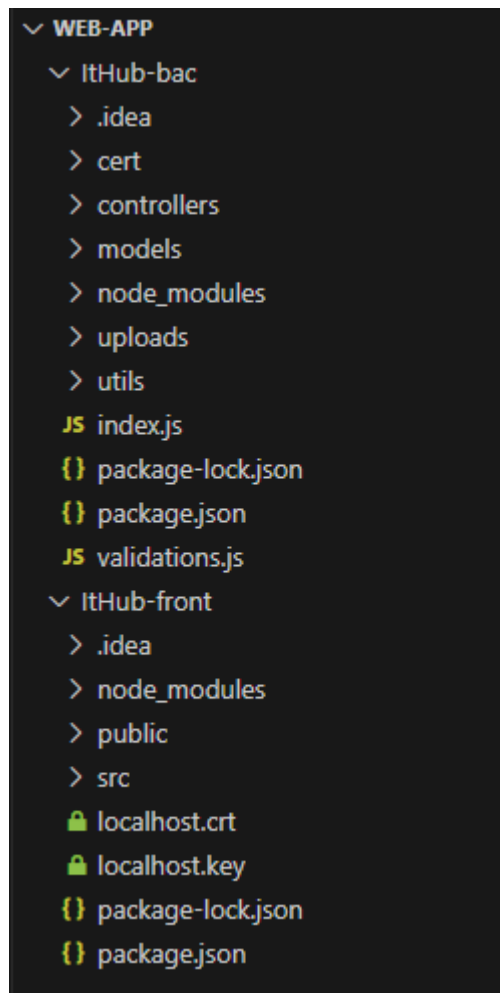


Рисунок 3.1 – Структура разработанного web-приложения

В папке React-приложения создадим Dockerfile с кодом представленным на листинге 3.8.

```
FROM node:18.16.0
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 4444
CMD [ "npm", "start" ]
```

Листинг 3.8 – Dockerfile клиентской части

В папке серверной части приложения создадим Dockerfile с кодом представленным на листинге 3.9.

```
FROM node:18.16.0
WORKDIR /usr/src/app
COPY package*.json ./
```

```
RUN npm install --silent
COPY . .
EXPOSE 3000
CMD ["npm", "start"]
```

### Листинг 3.9 – Dockerfile серверной части

В папке web-app, содержащей клиентскую и серверную части приложения, создадим файл docker-compose.yml. Файл docker-compose.yml содержит код представленный на листинге 3.10.

```
version: "3"
services:
  api:
    build: ./ithub-bac
    ports:
      - "4444:4444"
  client:
    build: ./ithub-front
    ports:
      - "3000:3000"
```

### Листинг 3.10 – Код файла docker-compose.yml

Запустить приложение можно через консоль в папке web-app при помощи команды, представленной на листинге 3.11.

```
docker compose up
```

### Листинг 3.10 – Команда запуска docker compose

Так же запустить приложение можно в Docker-desktop. Скрин данного проекта изображен на рисунке 3.2

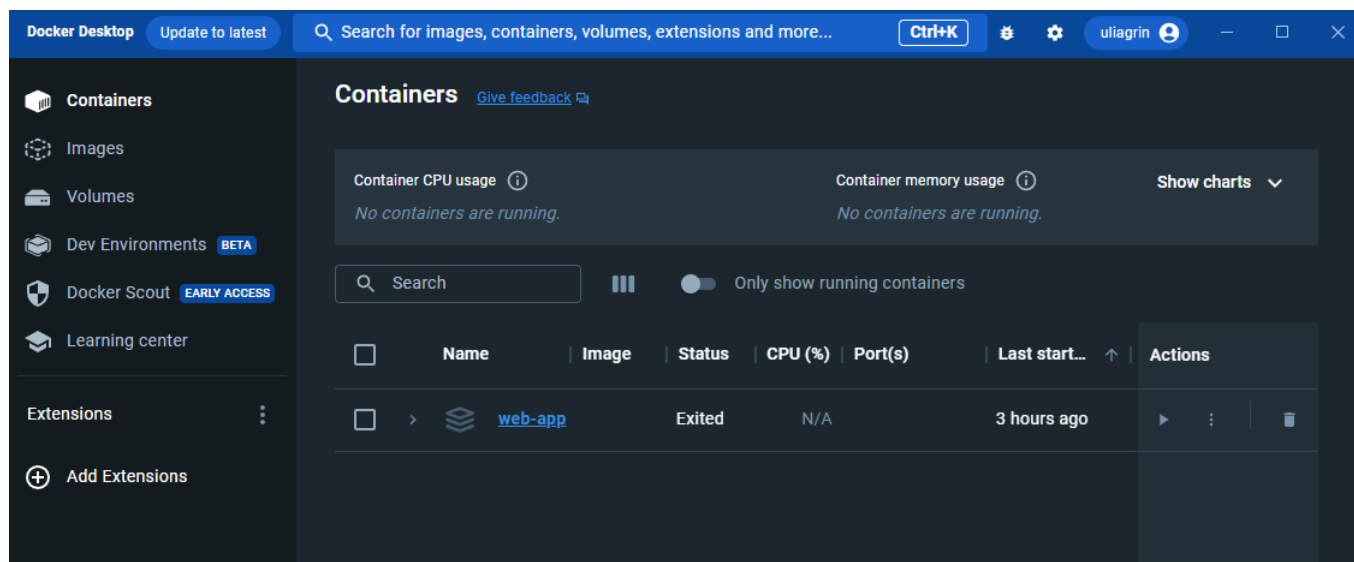


Рисунок 3.1 – Проект в Docker-desktop

После его запуска появится возможность обратиться к данному web-приложению.

По результату проведенной работы в данном разделе, были разработаны две части данного приложения, а именно frontend и backend части, каждая из которых является отдельный сервером. Между ними, как архитектурными единицами, существует одно яркое отличие, в то время как backend является полностью независимым от других сервисом сервером, frontend сервер же является полностью зависимым от backend части приложения, так как сам по себе он не несет никакой смысловой нагрузки и не способен обрабатывать никакую бизнес-логику. И в конце всего процесса был создан docker-контейнер.

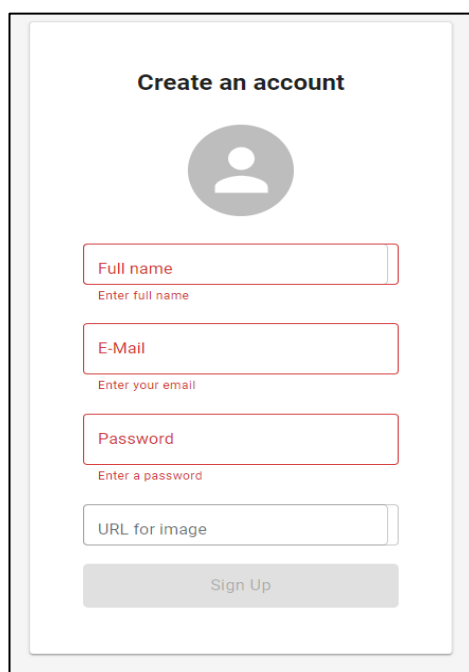
После разработки стоит приступить к следующей не менее важной части разработки любого приложения, к тестированию этого самого приложения.

## 4 Тестирование web-приложения

В этом разделе рассмотрим процесс тестирования завершеного и полноценного приложения с использованием клиентской части. Особое внимание будет уделено тестированию функций регистрации и авторизации пользователей.

При тестировании регистрации и авторизации пользователей, если вводятся некорректные данные, приложение будет выводить сообщение об ошибке для информирования пользователя о проблеме.

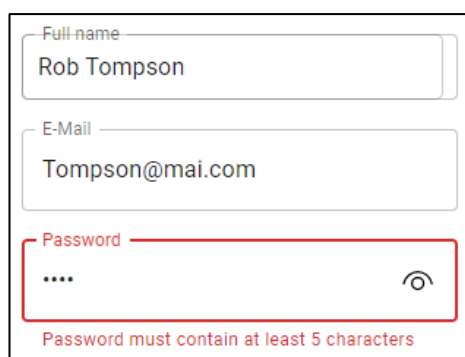
Пример такого сообщения представлен на рисунке 4.1 при оставлении любого из полей пустыми, происходит отображение ошибок.



The screenshot shows a 'Create an account' form with a user icon at the top. Below the icon are four input fields: 'Full name', 'E-Mail', 'Password', and 'URL for image'. Each of the first three fields has a red border and a red error message below it: 'Enter full name', 'Enter your email', and 'Enter a password' respectively. The 'URL for image' field is empty. At the bottom is a 'Sign Up' button.

Рисунок 4.1 – Ошибки валидации при пустых значениях

При вводе пароля меньше пяти символов получен соответствующее уведомление об ошибку валидации, представленное на рисунке 4.2. Длина пароля должна быть не менее 5 символов.

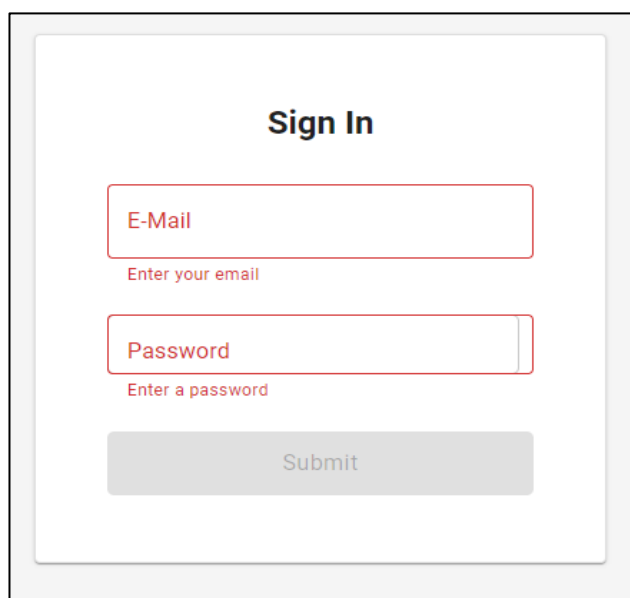


The screenshot shows a registration form with three input fields: 'Full name' (containing 'Rob Tompson'), 'E-Mail' (containing 'Tompson@mai.com'), and 'Password' (containing four dots). The 'Password' field has a red border and a red error message below it: 'Password must contain at least 5 characters'. There is also an eye icon to the right of the password field.

Рисунок 4.2 – Ошибка некорректного ввода пароля



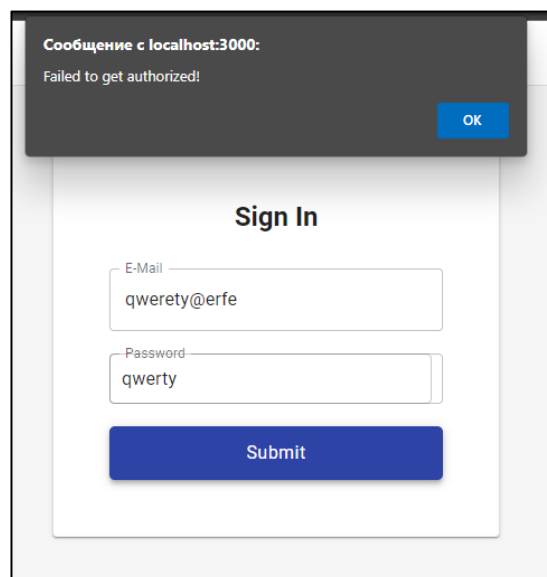
При вводе пустых значений при авторизации будет выведено сообщение об ошибке, представленной на рисунке 4.3.



The image shows a 'Sign In' form. At the top, the text 'Sign In' is centered. Below it are two input fields. The first is labeled 'E-Mail' and has the placeholder text 'Enter your email'. The second is labeled 'Password' and has the placeholder text 'Enter a password'. Below these fields is a grey 'Submit' button. The entire form is enclosed in a light grey border.

Рисунок 4.3 – Ошибка ввода пустых значений при авторизации

В случае заполнения полей корректными данными, также выполняется проверка на соответствие вводимого пароля и корректность почты. Пример ошибки несоответствия пароля приведен или почты на рисунке 4.3.

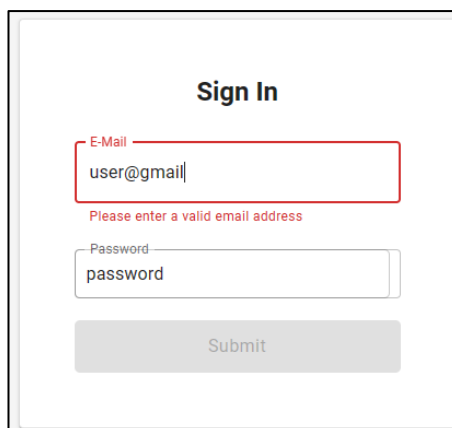


The image shows the same 'Sign In' form as in the previous figure, but with the 'E-Mail' field containing 'qwerety@erfe' and the 'Password' field containing 'qwerty'. A dark grey error message overlay is displayed at the top of the form. The message reads: 'Сообщение с localhost:3000: Failed to get authorized!'. There is a blue 'OK' button next to the message. The 'Submit' button is now blue.

Рисунок 4.3 – Ошибка ввода несуществующих данных

Если пользователь попытается ввести некорректный формат для почты, то получит соответствующую ошибку. В этом примере использован паттерн для проверки корректности электронного адреса. Он проверяет, содержит ли значение символ "@" и соответствует ли формату электронного адреса. Если пользователь вводит некорректный адрес, будет отображаться сообщение об ошибке "Please enter a valid email address". Это сообщение будет отображаться, если поле остается пустым

или введенный адрес не содержит символа "@". Случай такой ошибки наглядно представлен на рисунке 4.4, где видно текст ошибки.



The image shows a 'Sign In' form with two input fields: 'E-Mail' and 'Password'. The 'E-Mail' field contains the text 'user@gmail|' and is highlighted with a red border. Below it, a red error message reads 'Please enter a valid email address'. The 'Password' field contains the text 'password'. A 'Submit' button is located at the bottom of the form.

Рисунок 4.4 – Ошибка при вводе некорректного формата почты

На форуме представлена возможность оценить понравившееся обсуждение лайком. Работает по принципу один пользователь – один лайк. При нажатии на сердечко в первый раз увеличиться общее количество лайков обсуждения. Обработка данной ситуации представлена на рисунке 4.5.

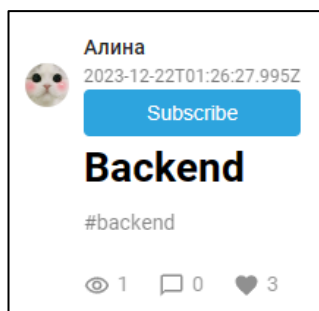


Рисунок 4.5 – Поставлен лайк от пользователя

Повторное нажатие по сердечку от того же пользователя приведет к отмене лайка, изменения можно будет увидеть в общем количестве лайков. Обработка данной ситуации представлена на рисунке 4.6.

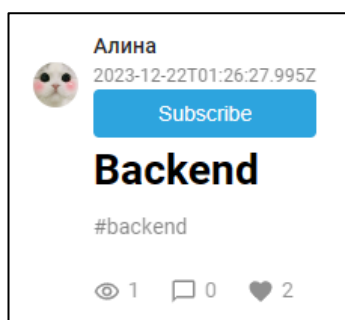


Рисунок 4.5 – Убран лайк от пользователя

В ходе проведения тестирования разрабатываемого приложения были представлены типичные ситуации, которые могут вызывать ошибки, и обнаруженные на них реакции. Результаты этого тестирования сыграли важную роль в выявлении и устранении множества ошибок и проблем, связанных с функциональностью и производительностью приложения. Это позволило значительно улучшить качество и работоспособность приложения.

Тем не менее, как и в случае с любой системой, это приложение нуждается в постоянном обновлении и развитии. Для достижения этой цели важно регулярно проводить тестирования, анализировать полученные данные и учитывать отзывы пользователей. Эти меры помогут непрерывно улучшать функциональность и общее качество приложения, обеспечивая положительный опыт пользователей. Однако, как и любая система, приложение нуждается в постоянном обновлении и совершенствовании. Для этого необходимо проводить регулярные тестирования, анализировать данные и отзывы пользователей, чтобы улучшать функциональность и повышать качество приложения.

Помимо обновления и развития приложения, необходимо также активно следить за изменениями в технологической среде и требованиями пользователей. Рынок приложений постоянно эволюционирует, и для того, чтобы оставаться конкурентоспособным, важно быть в курсе последних тенденций и инноваций. Это позволит разработчикам приложения внедрять новые функциональные возможности и улучшения, отвечающие потребностям пользователей и обеспечивающие превосходное пользовательское взаимодействие.

Одной из важных составляющих процесса совершенствования приложения является регулярный анализ данных и отзывов пользователей. Сбор и анализ данных о поведении пользователей в приложении помогает выявить слабые места, узнать о возможных проблемах или неудобствах, с которыми сталкиваются пользователи, и принять соответствующие меры для улучшения опыта использования.

## 5 Руководство пользователя

### 5.1 Возможности неавторизованного пользователя

Неавторизованному пользователю представлен ограниченный функционал форма, который сводится к просмотру всего что на нем представлено, кроме регистрации и авторизации. Главная страница форума от неавторизованного пользователя представлена на рисунке 5.1.

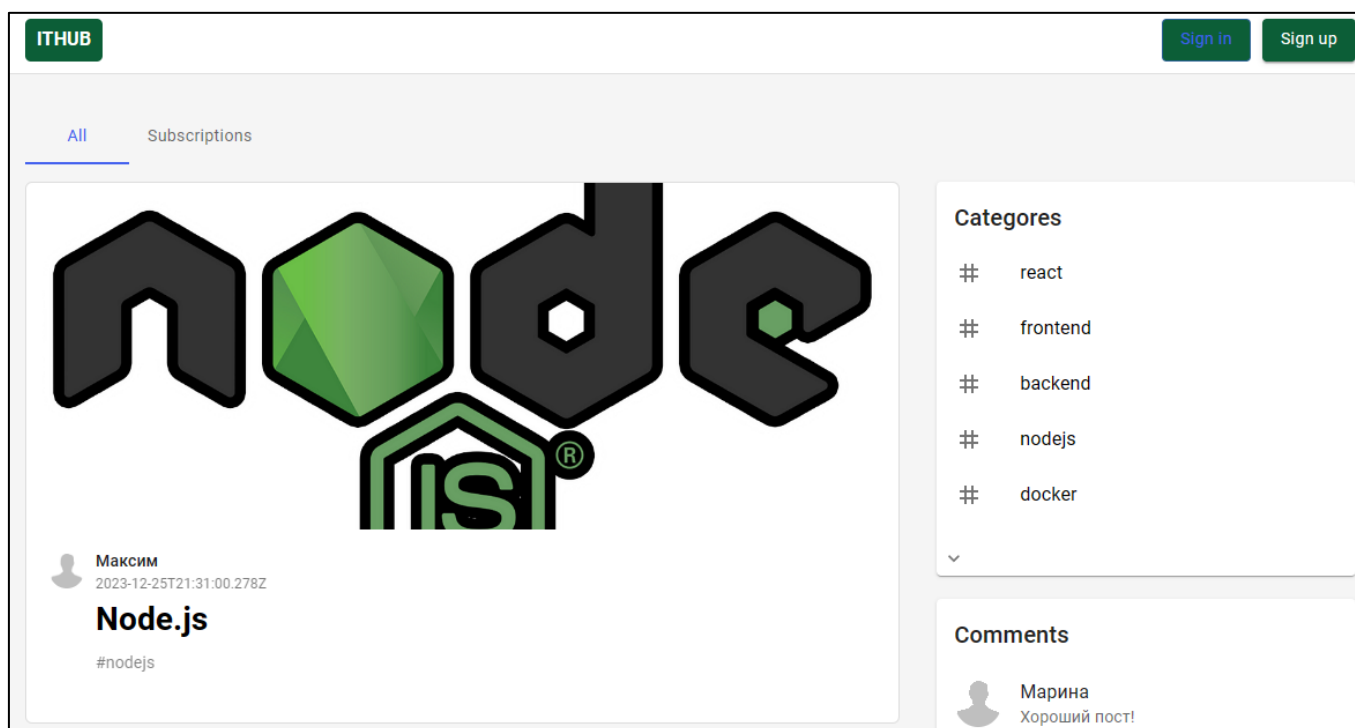


Рисунок 5.1 – Главная страница для неавторизованного пользователя

В правом верхнем углу располагаются кнопки предлагающие пройти авторизацию или регистрацию. На постах отключен вывод статистики о просмотрах, лайках и комментариях. Отсутствует возможность подписаться на авторов постов. В секции общего чата отсутствует поле для отправки сообщения в групповой чат, представлено на рисунке 5.2.

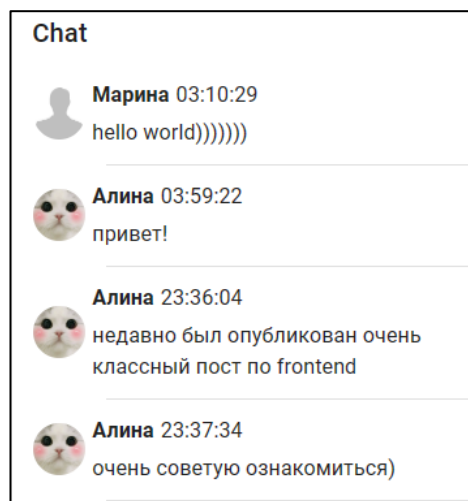


Рисунок 5.2 – Просмотр чата для неавторизованного пользователя

В секции обсуждения отсутствует возможность принять участие в обсуждении путем отправки комментария, представлено на рисунке 5.3

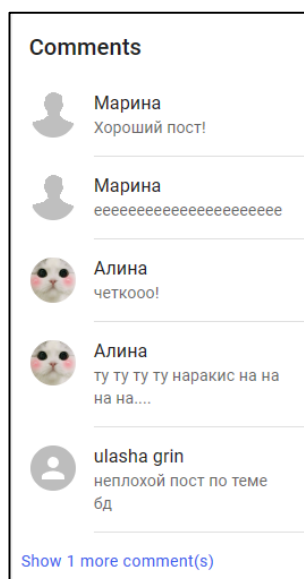


Рисунок 5.3 – Просмотр комментариев для неавторизованного пользователя

## 5.2 Регистрация пользователя

Для регистрации необходимо заполнить форму, ввести полное имя, почту пароль и url на картинку профиля по желанию. Форма продемонстрирована на рисунке 5.1.

The image shows a registration form titled "Create an account". At the top, there is a circular icon representing a user profile. Below the icon, there are four input fields: "Full name" with the text "user user", "E-Mail" with the text "user@test.com", "Password" with four dots, and "URL for image". At the bottom of the form is a button labeled "Sign Up".

Рисунок 5.4 – Форма регистрации

В случае успешной регистрации пользователь переадресовывается на главную страницу приложения, где будут представлены все возможности авторизованного пользователя. В них входят: отправления сообщений в групповой чат, подписка на авторов постов, участие в обсуждениях путем отправки комментария, оставление лайков, также видна статистика обсуждений. Еще одно преимущество авторизованного пользователя заключается в доступе панели поиска постов по их частичному названию, расположенной в верхней части экрана. Вид главной страницы авторизованного пользователя представлен на рисунке 5.5.

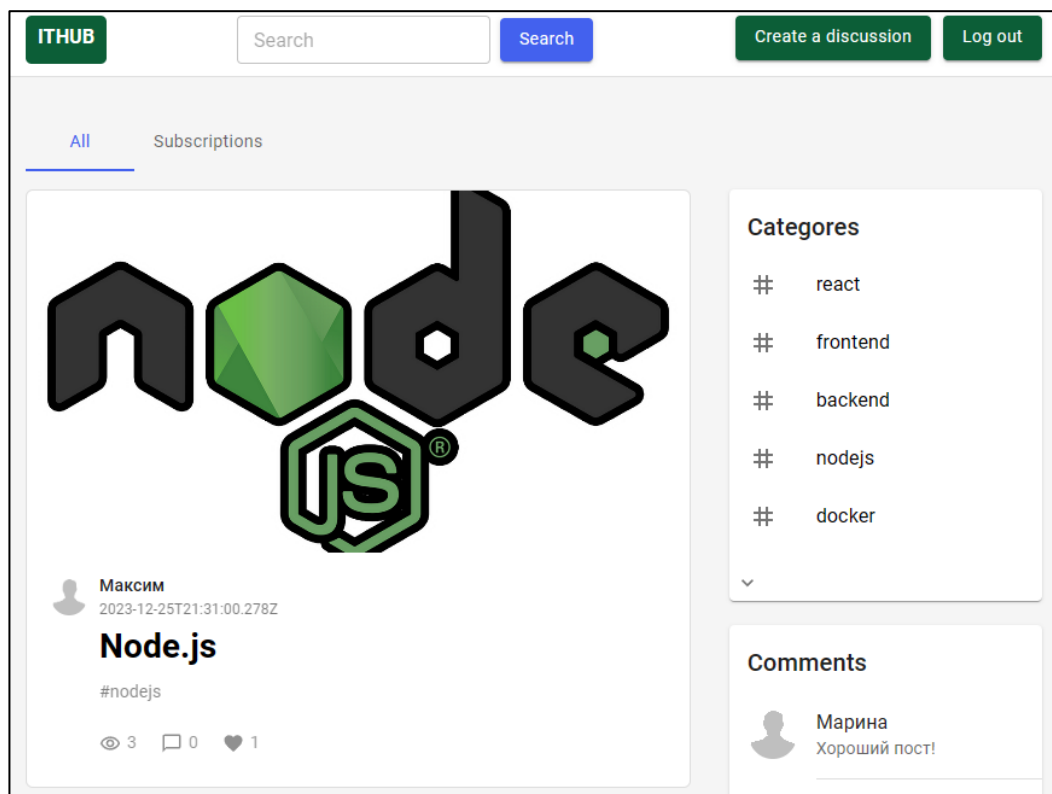


Рисунок 5.5 – Главная страница для авторизованного пользователя

### 5.3 Аутентификация и авторизация пользователя

На странице «login» пользователь может произвести аутентификацию, заполнив все необходимые поля формы. Произойдет проверка существования данного пользователя и проверка правильности ввода пароля.

В случае успешной аутентификации пользователя, он будет перенаправлен на главную страницу приложения и у него станут доступными другие страницы приложения. Форма авторизации представлена на рисунке 5.2.

The image shows a 'Sign In' form. It has a title 'Sign In' at the top. Below the title, there are two input fields. The first field is labeled 'E-Mail' and contains the text 'user@gmail.com'. The second field is labeled 'Password' and contains the text 'password'. Below these fields is a blue button with the text 'Submit'.

Рисунок 5.6 – Форма авторизации



После того, как пользователь авторизуется, ему выдается токен, который хранится в `localStorage`. Таким образом ограничиваются возможности пользователей в приложении.

У каждого пользователя определена роль. Всего их две: пользователь, администратор. Роль обычного пользователя позволяет использовать все основные возможности приложения. Администратору, в отличие от пользователя, дополнительно разрешено редактирование, удаление чужих постов и комментариев.

### 5.3 Возможности авторизованного пользователя

При успешной авторизации пользователь попадает на главную страницу приложения, на которой находится основной функционал форума. Вид главной страницы для авторизованного пользователя был представлен на рисунке 5.5. При нажатии на кнопку «Create discission» происходит переход на страницу создания статьи, функционал создания статьи представлен на рисунке 5.7. Добавим заголовок, загрузим картинку, добавим теги по теме и введем текст обсуждения.

Upload image

Удалить

# Node.js

nodejs

B

I

H

“

≡

≡

🔗

🖼️

👁️

📄

✂️

🔍

Node (или более формально Node.js) - кроссплатформенная среда исполнения с открытым исходным кодом, которая позволяет разработчикам создавать всевозможные серверные инструменты и приложения используя язык JavaScript. Среда исполнения предназначена для использования вне контекста браузера (т.е. выполняется непосредственно на компьютере или на серверной ОС). Таким образом, среда исключает API-интерфейсы JavaScript для браузера и добавляет поддержку более традиционных OS API-интерфейсов, включая библиотеки HTTP и файловых систем.

С точки зрения веб-серверной разработки Node имеет ряд преимуществ:

Save

Cancel

Рисунок 5.7 – Страница создание обсуждение

При нажатии на кнопку publish, обсуждение публикуется и его можно увидеть на главной странице приложения, представлена на рисунке 5.8.



Рисунок 5.8 — Добавленное обсуждение

При наведении на обсуждение появятся карандаш и крестик, отвечающие за редактирование и удаление соответственно, представлено на рисунке 5.9.



Рисунок 5.9 — Редактирование и удаление обсуждения

Для оставления комментария, нужно раскрыть статью, ввести текст и нажать на кнопку send, представлено на рисунке 5.10.

Comments

неплохой пост про node.js

Send

Рисунок 5.10 — Отправка комментария

Для удаления или редактирования нужно при наведении на свой комментарий нажать на появившийся значок карандаша или ведра для редактирования или удаления соответственно, наглядно представлено на рисунке 5.11.



Рисунок 5.11 — Редактирование и удаление комментария

После авторизации появилось поле для отправки сообщения в групповой чат, представлено на рисунке 5.12.

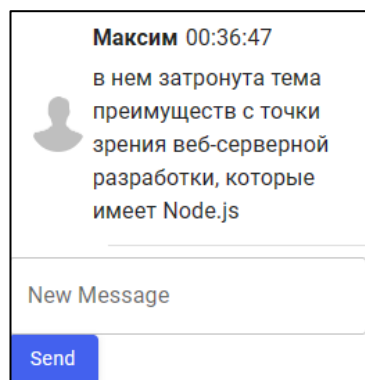


Рисунок 5.12 — Отправка сообщения в групповой чат

Отправленное сообщение сразу же отобразится у всех подключённых пользователей, благодаря реализации на WebSocket.

## 5.4 Возможности Администратора

Роль администратора добавлена для модерации отправляемого контента, в его возможности входит удаление и редактирование чужих постов, при наведении на них появляются инструменты для модерации, представлены на рисунке 5.13.



Рисунок 5.13 — Меню для редактирования и удаления чужого поста

Таким же образом администратор может удалять или редактировать чужие комментарии, представлено на рисунке 5.14.



Рисунок 5.14 — Меню для редактирования и удаления чужого комментария

Роль администратора на форуме также включает в себя модерацию отправляемого контента, что позволяет ему эффективно контролировать и поддерживать качество постов и комментариев. Администратор имеет возможность удаления и редактирования чужих постов и комментариев для поддержания порядка и соответствия правилам форума.

При наведении на чужой пост в интерфейсе администратора появляется меню для модерации, представленное на рисунке 5.13. Это меню обычно содержит опции, позволяющие администратору удалить или отредактировать пост, в зависимости от конкретных требований и настроек форума.

Аналогично, для чужих комментариев также доступно меню для редактирования и удаления, как показано на рисунке 5.14. Администратор может выбрать соответствующую опцию в меню, чтобы удалить комментарий или внести необходимые изменения.

Такие функции модерации позволяют администратору быстро и эффективно реагировать на нарушения правил форума, не допуская распространения нежелательного или нарушающего контента. Кроме того, администратор может использовать свои полномочия для создания безопасной и приятной среды для всех пользователей форума, предотвращая спам и другие негативные проявления.

## Заключение

В рамках данного курсового проекта было успешно разработано web-приложение форум «ItHub», которое отличается удобством использования и обеспечивает высокий уровень безопасности.

Перед началом разработки приложения был проведен аналитический обзор существующих прототипов приложений схожей тематики. Это помогло определить функциональные возможности, которые были реализованы в успешно разработанном приложении форум «ItHub».

Был разработан интуитивно понятный и простой в использовании интерфейс, который обеспечивает удобство взаимодействия пользователя с приложением.

В процессе разработки были использованы современные и популярные библиотеки, модули и фреймворки, такие как Express, Mongoose, Socket.io, React и Redux. Эти инструменты позволили полностью реализовать задуманный функционал приложения, согласно заданию.

Для обеспечения безопасности передачи данных приложение использует сгенерированные сертификаты, обеспечивая шифрованное HTTPS соединение между клиентом и сервером.

Важной составляющей разработки является взаимодействие приложения с базой данных MongoDB, которая размещена на сервере разработчиков. Это обеспечивает надежное и эффективное хранение данных приложения.

Особое внимание уделялось тестированию разработанного программного продукта. Тестирование позволило выявить и устранить возможные ошибки и недостатки в работе приложения, а также убедиться в правильности его функционирования и соответствии требованиям, указанным в техническом задании. Это позволило создать стабильное и надежное приложение, готовое к использованию.

В результате разработки были выполнены все поставленные задачи, в том числе приложение должно:

- обеспечивать возможность регистрации и авторизации;
- поддерживать роли администратора и пользователя;
- позволяет оставить комментарий под темой на определенную тематику;
- позволяет отвечать на комментарии под темами создавая тред;
- позволять пользователям создавать темы;
- реализовывать редактирование и удаление тем для их владельцев;
- предоставлять возможность поддерживать связь между пользователями форума при помощи чата;
- предоставлять возможность скрывать все темы для определенного подфорума;
- предоставлять возможность оценить тему лайком;
- предоставлять возможность подписаться на автора интересующих тем;
- предоставлять возможность поиска тематики по названию.

## Список используемых источников

- 1 Reddit [Электронный ресурс]. – Режим доступа: <https://www.reddit.com/> – Дата доступа: 06.10.2023.
- 2 4PDA [Электронный ресурс]. – Режим доступа: <https://4pda.to/> – Дата доступа: 08.10.2023.
- 3 Pikabu [Электронный ресурс]. – Режим доступа: <https://pikabu/> – Дата доступа: 08.10.2023.
- 4 Express документация [Электронный ресурс]. – Режим доступа: <https://expressjs.com/> – Дата доступа: 11.12.2023.
- 5 MongoDB документация [Электронный ресурс]. – Режим доступа: <https://www.mongodb.com/docs/> – Дата доступа: 13.12.2023
- 6 Mongoose документация [Электронный ресурс]. – Режим доступа: <https://mongoosejs.com/docs/> – Дата доступа: 22.12.2023.
- 7 React документация [Электронный ресурс]. – Режим доступа: <https://legacy.reactjs.org/docs/getting-started.html> Дата доступа: 24.12.2023.
- 8 Redux документация [Электронный ресурс]. – Режим доступа: <https://redux.js.org/introduction/getting-started> Дата доступа: 18.12.2023.

## ПРИЛОЖЕНИЕ А

### Листинг «Модели базы данных»

```
const CategorySchema = new mongoose.Schema(  
  {  
    category: {  
      type: String,  
      required: true  
    },  
  
  },  
  
  {  
    timestamps: true,  
  },  
);  
const CommentSchema = new mongoose.Schema(  
  {  
    text: {  
      type: String,  
      required: true  
    },  
  
    user: {  
      type: mongoose.Schema.Types.ObjectId,  
      ref: 'User',  
      required: true,  
    },  
    post: {  
      type: mongoose.Schema.Types.ObjectId,  
      ref: 'User',  
      required: true,  
    }  
  },  
  {  
    timestamps: true,  
  },  
);  
const LikeSchema = new mongoose.Schema(  
  {  
  
    user: {  
      type: mongoose.Schema.Types.ObjectId,  
      ref: 'User',  
      required: true,  
    },  
    post: {  
      type: mongoose.Schema.Types.ObjectId,  
      ref: 'User',  
      required: true,  
    }  
  }
```

```

    },
    {
      timestamps: true,
    },
  );
const MessageSchema = new mongoose.Schema(
  {
    username: {
      type: String,
      required: true,
    },
    message: {
      type: String,
      required: true,
    },
    timestamp: {
      type: String,
      required: true,
    },
    avatarUrl: {
      type: String,
      required: true
    }
  },
);
const PostSchema = new mongoose.Schema(
  {
    title: {
      type: String,
      required: true,
    },
    text: {
      type: String,
      required: true,
      unique: true,
    },
    tags: {
      type: Array,
      default: [],
    },
    viewsCount: {
      type: Number,
      default: 0,
    },
    user: {
      type: mongoose.Schema.Types.ObjectId,
      ref: 'User',
      required: true,
    },
  },

```



```

        imageUrl: String,
    },
    {
        timestamps: true,
    },
);
const RoleSchema = new mongoose.Schema(
    {
        role: {
            type: String,
            required: true
        },

    },
    {
        timestamps: true,
    },
);
const UserSchema = new mongoose.Schema(
    {
        fullName: {
            type: String,
            required: true,
        },
        email: {
            type: String,
            required: true,
            unique: true,
        },
        passwordHash: {
            type: String,
            required: true,
        },
        role:{
            type: mongoose.Schema.Types.String,
            default: 'user',
            ref: 'Role',
            required: false
        },
        subscribedTo: [{
            type: mongoose.Schema.Types.ObjectId,
            ref: 'User',
            required: false,
            default:[],
            uniq: true,
        }],
        avatarUrl: String,
    },
    {
        timestamps: true,
    },
);

```

## ПРИЛОЖЕНИЕ Б

### Листинг Socket-сервера

```
const io = new SocketIOServer(server, {
  cors: {
    origin: 'https://localhost:3000',
    methods: ['GET', 'POST'],
  },
});

const clients = new Map();
io.on('connection', (socket) => {
  console.log('New client connected');

  clients.set(socket.id, socket);

  socket.on('newPost', (data) => {
    console.log('Received new post from client:', data);

    socket.broadcast.emit('newPost', data);
  });

  socket.on('chatMessage', (data) => {
    console.log('Received chat message:', data);

    io.emit('chatMessage', data);
  });

  socket.on('disconnect', () => {
    console.log('Client disconnected');

    clients.delete(socket.id);
  });
});
```

## ПРИЛОЖЕНИЕ В

### Листинг файла package.json

```
{
  "name": "ithub-bac",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "type": "module",
  "scripts": {
    "start": "node index.js",
    "start:dev": "nodemon index.js"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "bcrypt": "^5.0.1",
    "cors": "^2.8.5",
    "express": "^4.18.2",
    "express-validator": "^6.14.1",
    "express-ws": "^5.0.2",
    "jsonwebtoken": "^8.5.1",
    "mongoose": "^6.3.5",
    "mongoose-autopopulate": "^1.0.1",
    "multer": "^1.4.5-lts.1",
    "nodemon": "^2.0.16",
    "socket.io": "^4.6.1",
    "socket.io-client": "^4.6.1",
    "ws": "^8.1.0"
  }
}
```

