

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий
Кафедра Информационные системы и технологии
Специальность 1–40 01 01 Программное обеспечение информационных технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОЙ РАБОТЕ НА ТЕМУ:**

Web-приложение «Издательская платформа»

Выполнил студент Пузиков Алексей Алексеевич
(Ф.И.О.)

Руководитель работы ассист. Нистюк Ольга Александровна
(учен. степень, звание, должность, Ф.И.О., подпись)

И.о. зав. кафедрой ст. преп. Блинова Е.А.
(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовая работа защищена с оценкой _____

Минск 2024

Содержание

Введение	4
1 Постановка задачи	5
1.1 Аналитический обзор аналогов	5
1.1.1 Интернет-ресурс «Хабр»	5
1.1.2 Интернет-ресурс «Teletype»	6
1.1.3 Интернет ресурс «Hashtap»	7
1.1.4 Интернет ресурс «Dev.to»	7
1.2 Постановка задачи	8
1.3 Средства разработки	9
2 Проектирование web-приложения	10
2.1 Архитектура приложения	10
2.2 Проектирование структуры базы данных	11
3 Разработка web-приложения	14
3.1 Обобщенная структура приложения	14
3.2 Разработка серверной части приложения	14
3.2.1 Микросервисы	16
3.2.2 Сервис аутентификации	17
3.2.3 Сервис уведомлений	18
3.2.4 Сервис профилей	20
3.2.5 Сервис новостей	20
3.2.6 API Gateway	21
3.3 Разработка клиентской части приложения	22
4 Тестирование web-приложения	25
4.1 Ошибка при вводе несуществующего маршрута	25
4.2 Ошибка при пустом значении полей	25
4.3 Ошибка при некорректных данных для входа	26
4.4 Ошибка при регистрации существующего логина	26
4.5 Иные ошибки	26
5 Руководство пользователя	27
5.1 Вход в аккаунт	27
5.2 Регистрация пользователя	27
5.3 Взаимодействие с профилем пользователя	28
5.4 Создание поста	29
5.5 Редактирование поста	29
5.6 Страница «People»	30
5.7 Страница новостей	30
5.8 Страница «Для вас»	31
5.9 Оставление комментариев	32
5.10 Страница технической поддержки	32
Заключение	34
Список используемых источников	35
Приложение А	36
Приложение Б	37

	3
Приложение В	38
Приложение Г	39
Приложение Д	41

Введение

В современном мире информационных технологий и цифровизации все больше людей полагаются на персонализированные рекомендации для обнаружения нового контента. От социальных сетей до новостных агрегаторов, персонализированные рекомендации стали ключевым элементом пользовательского опыта. В то же время, платформы для публикации контента предоставляют пользователям возможность делиться своими мыслями, идеями и творчеством с миром.

В рамках данного курсового проекта будет разработано веб-приложение, которое сочетает в себе сервис персональных рекомендаций и издательскую платформу. Цель – создать интуитивно понятное и удобное приложение, которое позволит пользователям публиковать свой контент в виде текста или изображений, комментировать посты других пользователей, кастомизировать свой профиль и получать персонализированный контент, основанный на их интересах и взаимодействиях. Для достижения этой цели необходимо выполнить следующие задачи:

- провести анализ предметной области и определить основные сущности и атрибуты системы;
- разработать архитектуру приложения, используя подход микросервисов;
- реализовать каждый микросервис, включая сервис аутентификации, уведомлений, профилей и новостей;
- разработать API Gateway для маршрутизации запросов к соответствующим микросервисам;
- создать клиентскую часть приложения используя React для взаимодействия с пользователем;
- наполнить базу данных тестовыми данными;
- разработать функции и методы для реализации бизнес-логики каждого микросервиса.

1 Постановка задачи

1.1 Аналитический обзор аналогов

В современном мире издательские платформы играют ключевую роль в распространении информации. Они предоставляют обычным пользователям публиковать свои мысли, статьи, работы, делая их доступными для широкой аудитории. В этом разделе будут приведены интернет-ресурсы, а также веб-приложения существующих издательских платформ.

1.1.1 Интернет-ресурс «Хабр»

Хабр – это одна из самых популярных платформ для IT-специалистов, где пользователи могут публиковать статьи, обмениваться знаниями и обсуждать различные темы, связанные с информационными технологиями.

Интерфейс интернет-ресурса «Хабр» представлен на рисунке 1.1.

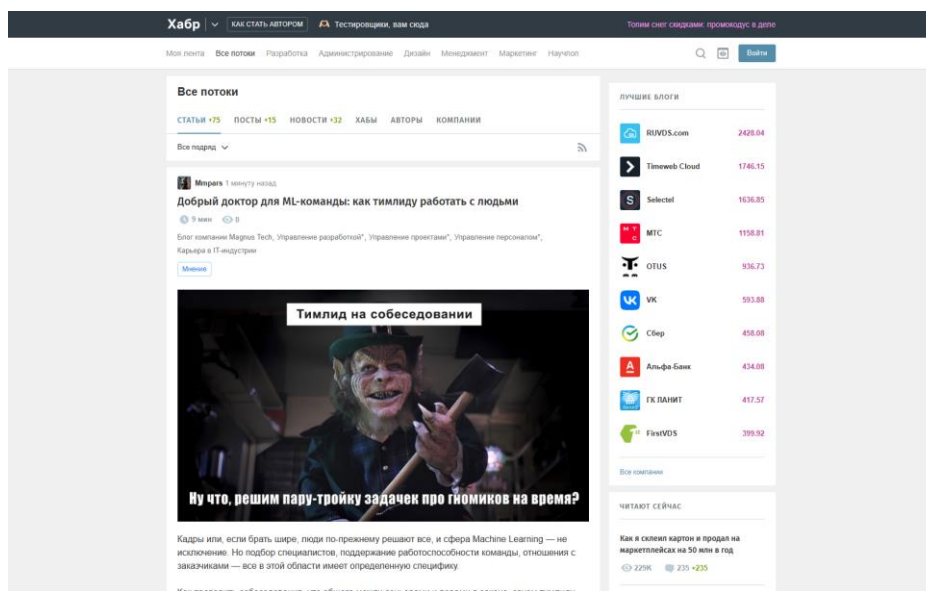


Рисунок 1.1 – Интерфейс интернет-ресурса «Хабр»

Достоинства:

- широкий спектр тем: хабр покрывает множество тем, связанных с IT, включая программирование, разработку веб-сайтов, информационную безопасность, аналитику данных и многое другое;
- сообщество экспертов: хабр имеет активное сообщество пользователей, которые являются экспертами в своих областях;
- персонализированный контент: пользователи могут настроить свою ленту новостей, чтобы видеть контент, который их интересует.

Недостатки:

- сложность навигации: для новых пользователей может быть сложно найти нужную информацию из-за большого количества контента;
- неравномерное качество контента: качество публикаций может сильно варьироваться, поскольку они создаются разными пользователями.

1.1.2 Интернет-ресурс «Teletype»

Teletype — это удобная платформа для публикации статей, которая тесно интегрирована с Telegram и позволяет легко делиться контентом через телеграм-каналы. Пользователи могут создавать статьи с картинками, ссылками и другими элементами, делая информацию доступной и привлекательной.

Интерфейс интернет-ресурса «Teletype» представлен на рисунке 1.2.

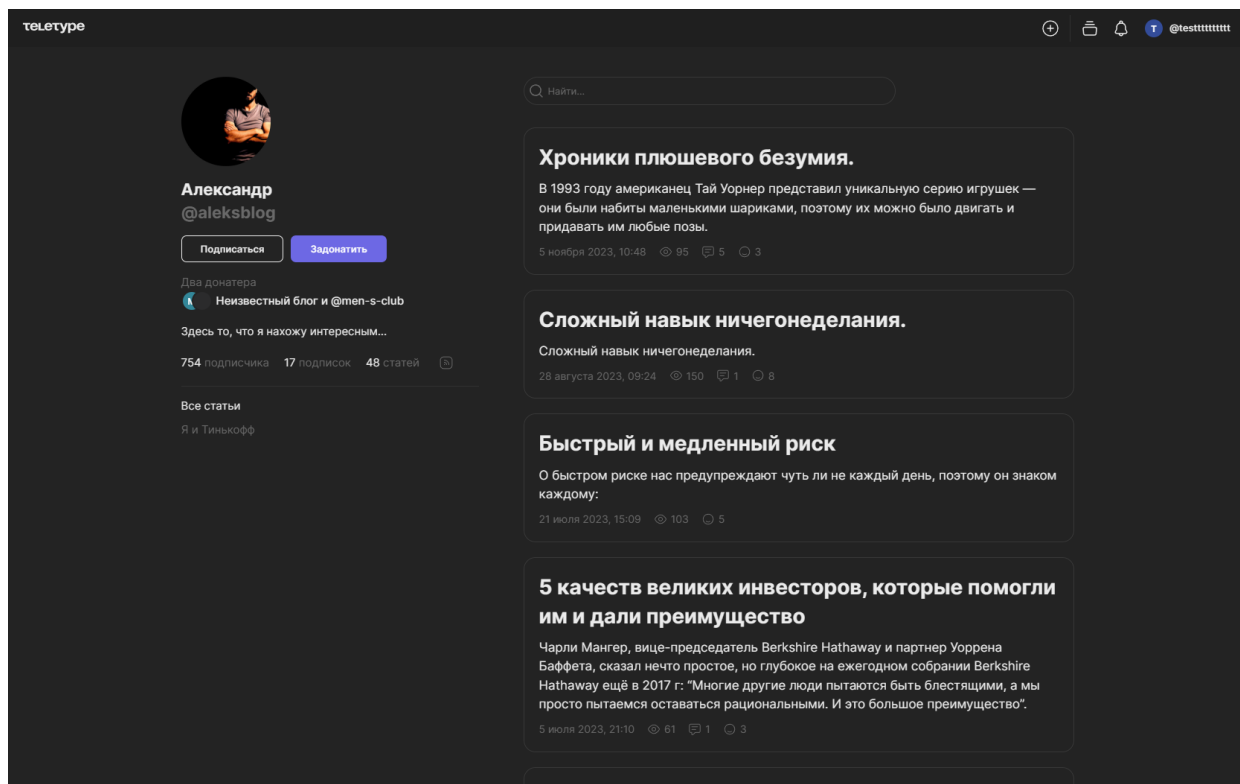


Рисунок 1.2 – Интерфейс интернет-ресурса «Teletype»

Достоинства:

- простота использования: Teletype предлагает интуитивно понятный интерфейс с простым редактором для создания статей;
- визуальное оформление: платформа предоставляет разнообразные инструменты для оформления текста, включая заголовки, курсив, жирный шрифт и вставку изображений;
- интеграция с Telegram: Teletype удобен для пользователей Telegram благодаря возможности быстрого перехода к статьям прямо из сообщений и каналов.

Недостатки:

- ограниченная функциональность: в отличие от более крупных платформ, Teletype может не предлагать некоторые продвинутые функции для публикации и управления контентом;
- зависимость от Telegram: для пользователей, которые не используют Telegram, платформа может быть менее привлекательной из-за её интеграции с этим мессенджером.

1.1.3 Интернет ресурс «Hashtap»

Сайт Hashtap представляет собой платформу, которая предлагает сервис персонализированных рекомендаций. Он также предлагает платформу email-маркетинга, которая автоматически отправляет персонализированные и триггерные email-письма.

Интерфейс интернет-ресурса «Hashtap» представлен на рисунке 1.3.

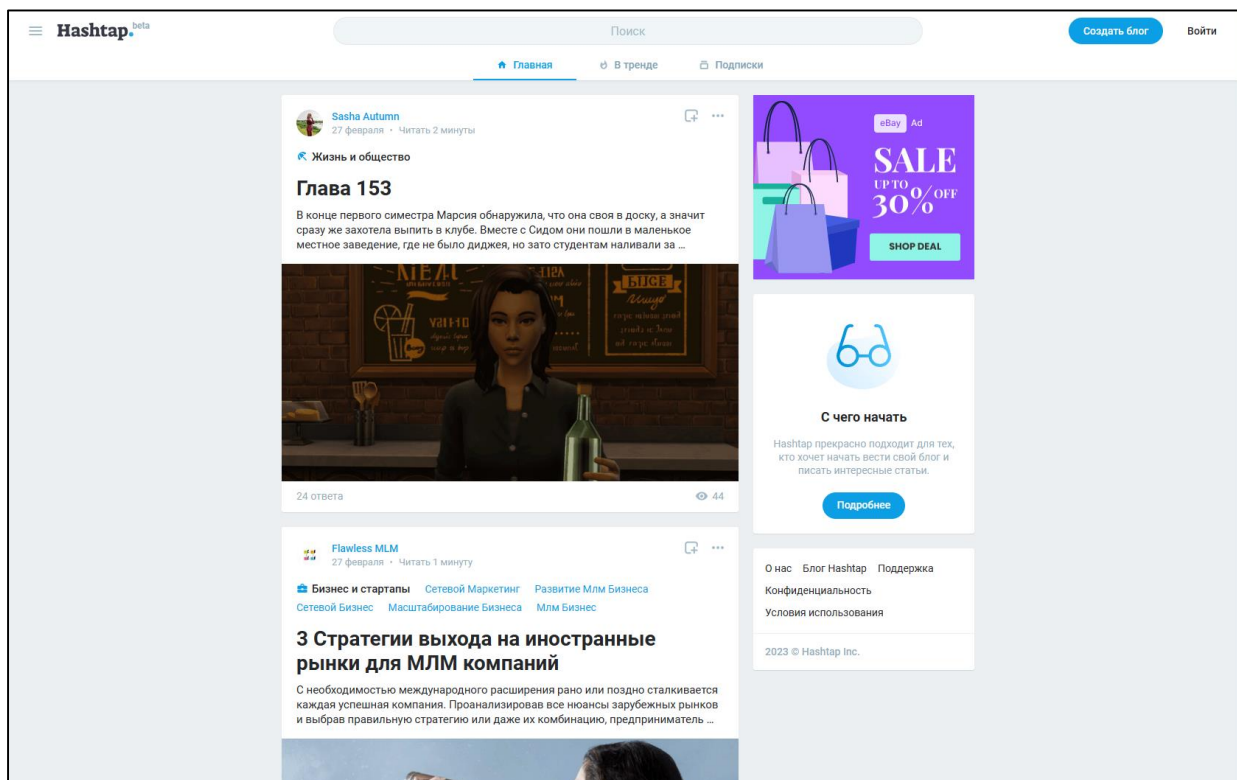


Рисунок 1.3 – Интерфейс интернет-ресурса «Hashtap»

Достоинства:

- персонализированные рекомендации: Hashtap предлагает персонализированные рекомендации, что может помочь пользователям обнаружить новый контент;
- email-маркетинг: Hashtap предлагает платформу email-маркетинга, которая автоматически отправляет персонализированные и триггерные email-письма;
- поддержка: Hashtap имеет команду поддержки, которая может помочь пользователям с вопросами и предложениями.

Недостатки:

- сложность навигации: Для новых пользователей может быть сложно найти нужную информацию из-за большого количества контента.

1.1.4 Интернет ресурс «Dev.to»

Dev.to – это популярное сообщество разработчиков, где они могут обмениваться знаниями, учиться и обсуждать различные темы, связанные с веб-разработкой.

Интерфейс интернет-ресурса «Dev.to» представлен на рисунке 1.4.

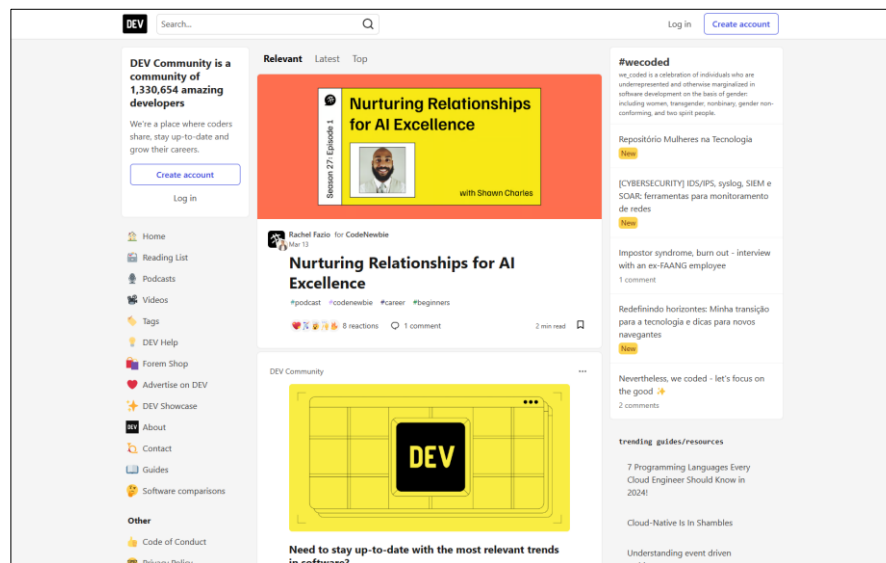


Рисунок 1.4 – Интерфейс интернет-ресурса «Dev.to»

Достоинства:

- активное сообщество: Dev.to имеет большое и активное сообщество разработчиков, которые регулярно публикуют и обсуждают новые идеи и концепции;
- широкий спектр тем: Dev.to покрывает множество тем, связанных с веб-разработкой, включая HTML, CSS, Javascript, React, Node.js и многое другое;
- обучающие материалы: Dev.to предлагает множество обучающих материалов и руководств, которые могут помочь разработчикам улучшить свои навыки.

Недостатки:

- неравномерное качество контента: качество публикаций может сильно варьироваться, поскольку они создаются разными пользователями;
- отсутствие модерации: некоторые пользователи могут заметить, что на сайте отсутствует строгая модерация, что может привести к появлению нерелевантного или низкокачественного контента.

1.2 Постановка задачи

Обзор аналогов позволяет проанализировать все преимущества и недостатки альтернативных возможностей и сформулировать список требований, предъявляемых к разрабатываемому в данном курсовом проекте программному средству. Программное средство должно реализовывать следующие функции:

- обеспечивать возможность регистрации и авторизации;
- отправлять уведомления о новом контенте, который может быть интересен пользователю;
- иметь интуитивно понятный и удобный пользовательский интерфейс;
- предоставлять возможность пользователям публиковать свой контент в виде текста или изображений;

- предоставлять возможность пользователям оставлять комментарии к постам;
- предоставлять пользователю возможность кастомизировать свой профиль;
- предоставлять пользователю возможность редактировать и удалять собственные посты;
- предоставлять персонализированный контент, основанный на интересах и взаимодействиях пользователя;
- обеспечивать защиту всех пользовательских данных.

Диаграмма вариантов использования представлена в приложении А.

1.3 Средства разработки

При разработке приложения будут использованы:

- интегрированная среда разработки JetBrains WebStorm;
- язык программирования Javascript;
- язык гипертекстовой разметки HTML совместно с CSS;
- CSS-фреймворк Bootstrap;
- технологии GRPC, RabbitMQ;
- библиотеки React, Koa;
- Базы данных MS SQL Server, Redis

Использование данных технологий имеет несколько преимуществ при разработке веб-приложения для публикации контента.

Во-первых, интегрированная среда разработки JetBrains WebStorm обеспечивает высокую производительность и удобство при разработке приложений.

Во-вторых, язык программирования Javascript — это высокопроизводительный и эффективный язык, который позволяет быстро и легко разрабатывать веб-приложения.

В-третьих, использование CSS-фреймворка Bootstrap позволяет быстро и легко создавать богатые пользовательские интерфейсы.

Четвертое преимущество заключается в том, что технологии GRPC и RabbitMQ а также библиотеки React и Koa позволяют создавать расширяемое и масштабируемое приложение, которое может обслуживать большое количество пользователей.

Базы данных MS SQL Server и Redis позволяют создавать надежные и быстрые хранилища данных, которые используются для хранения и обработки информации о пользователях, постах, лайках и других сущностях, необходимых для управления издательской платформой.

Исходя из описанных выше технологий, можно сделать вывод о том, что для реализации данного проекта были использованы современные инструменты и технологии, что позволило повысить эффективность работы. В целом, использование таких технологий позволило реализовать проект на высоком уровне и обеспечить его успешное выполнение.

В данной главе были сформулированы основные требования к приложению на основе преимуществ и недостатков некоторых аналогов.

2 Проектирование web-приложения

2.1 Архитектура приложения

В современном мире информационных технологий, где бизнес-требования постоянно меняются, адаптивность и масштабируемость приложений становятся ключевыми для успеха. Ответом на эти вызовы стало решение использовать микросервисную архитектуру для разработки веб-приложения. Микросервисы предоставляют значительные преимущества, такие как независимое масштабирование компонентов системы, что обеспечивает гибкость в управлении ресурсами и оптимизации производительности. Это позволяет командам разработчиков работать параллельно над разными частями системы, ускоряя разработку и деплоймент новых функций и упрощая процесс непрерывной интеграции и доставки. Кроме того, микросервисная архитектура повышает устойчивость приложения, позволяя эффективно управлять ошибками без риска сбоя всей системы.

Взаимодействие сервисов осуществляется благодаря брокеру сообщений RabbitMQ и технологии GRPC. Данные механизмы позволяют легко расширять и масштабировать сервисы, предоставляемые клиенту, не изменяя состояния других частей платформ, что обеспечивает безопасное внедрение новых компонентов и изменение существующих.

Клиентская часть проекта использует React.js, который представляет собой библиотеку Javascript для разработки динамических пользовательских интерфейсов. React.js позволяет создавать переиспользуемые компоненты, которые эффективно обновляются и реагируют на изменения данных. Это достигается за счет использования виртуального DOM (Virtual DOM), который позволяет React обновлять только необходимые элементы интерфейса без перерисовки всего дерева компонентов. Такой подход улучшает производительность взаимодействия с динамическими интерфейсами и значительно упрощает процесс разработки.

Взаимодействие между клиентской и серверной частями осуществляется посредством Web API. Сервер предоставляет API, определяющий доступные конечные точки и форматы данных для обмена информацией с клиентским приложением. Клиентское приложение, построенное на React.js, отправляет HTTP-запросы на сервер, используя соответствующие конечные точки, и принимает ответы в формате JSON. Это обеспечивает надежное взаимодействие между клиентом и сервером, позволяя передавать данные и обновления между ними и обеспечивая функциональность и интерактивность всего приложения.

API Gateway играет ключевую роль, выступая в качестве единой точки входа для всех клиентских запросов и обеспечивая управление трафиком, авторизацией, и мониторингом. Это упрощает взаимодействие клиента на React с микросервисами, скрывая сложность внутренней коммуникации и предоставляя единый интерфейс доступа к сервисам. В итоге, выбор микросервисной архитектуры для нашего приложения на Koa.js и клиента на React был продиктован желанием создать надежный, масштабируемый и легко управляемый программный продукт, а использование API Gateway только усиливает этот подход, оптимизируя взаимодействие между клиентом и сервером и улучшая пользовательский опыт.

Также в приложении используется протокол WebSockets. Использование WebSockets в web-приложении позволяет создать интерактивный и динамичный пользовательский опыт. WebSockets обеспечивают постоянное соединение между клиентом и сервером, что позволяет мгновенно передавать данные в обоих направлениях без необходимости постоянного обновления страницы.

Таким образом программное средство должно состоять из нескольких проектов, которые взаимодействуют с друг другом, по средствам инфраструктурного слоя, что позволяет легко и быстро изменить часть приложения, тем самым, не изменять другую часть. Также это позволит в любой момент изменить реализацию одного из модуля и это никак не повлияет на дальнейшую работу систему.

Диаграмма развертывания представлена в приложении Б.

2.2 Проектирование структуры базы данных

В данном разделе представлена концептуальная и логическая модель баз данных для проекта. На этапе разработки концептуальной модели было проведено детальное изучение бизнес-требований и основных процессов необходимых веб-приложению издательской платформы.

Так как архитектура приложения подразумевает что у каждого сервиса будет собственная база данных, то связи с этим все сущности рассредоточены между ними.

База данных сервиса аутентификации содержит единственную таблицу Users. Эта таблица содержит информацию о пользователях, которые зарегистрировались в системе. Её структура представлена в таблице 2.1.

Таблица 2.1 – Структура таблицы USERS

Столбец	Тип данных	Ограничение целостности
ID	INT	PRIMARY KEY IDENTITY
USERNAME	NVARCHAR(255)	NOT NULL
EMAIL	NVARCHAR(255)	NOT NULL
PASSWORD	NVARCHAR(255)	
SALT	NVARCHAR(255)	
IS_EMAIL_VERIFIED	BIT	DEFAULT 0
VERIFICATION_TOKEN	NVARCHAR(255)	
RESET_PASSWORD_TOKEN	NVARCHAR(255)	
RESET_PASSWORD_EXPIRES	DATETIME	

База данных сервиса уведомлений содержит единственную таблицу Notifications. Эта таблица содержит информацию о уведомлениях, отправленных пользователям. Её структура представлена в таблице 2.2.

Таблица 2.2 – Структура таблицы NOTIFICATIONS

Столбец	Тип данных	Ограничение целостности
ID	INT	PRIMARY KEY IDENTITY
AUTHOR	NVARCHAR(255)	NOT NULL
USER_ID	INT	NOT NULL
CONTENT	TEXT	NOT NULL
IMAGE	NVARCHAR(255)	DEFAULT 'default_profile.png'
PUBLISHED_AT	DATETIME	DEFAULT CURRENT_TIMESTAMP

База данных сервиса профилей содержит следующие таблицы: Users, Posts, Subscriptions, Comments, Support.

Users – содержит профильную информацию о пользователях. Её структура представлена в таблице 2.3;

Таблица 2.3 – Структура таблицы USERS

Столбец	Тип данных	Ограничение целостности
ID	INT	PRIMARY KEY IDENTITY
USERNAME	NVARCHAR(255)	
EMAIL	NVARCHAR(255)	
FIRSTNAME	NVARCHAR(255)	
LASTNAME	NVARCHAR(255)	
PHONE_NUMBER	NVARCHAR(255)	
PROFILE_PICTURE	NVARCHAR(255)	DEFAULT 'default_profile.png'
SUBSCRIBERS_COUNT	INT	DEFAULT 0
SUBSCRIPTIONS_COUNT	INT	DEFAULT 0
POSTS_COUNT	INT	DEFAULT 0

Posts – содержит информацию о публикациях пользователей. Её структура представлена в таблице 2.4;

Таблица 2.4 – Структура таблицы POSTS

Столбец	Тип данных	Ограничение целостности
ID	INT	PRIMARY KEY IDENTITY
AUTHOR	NVARCHAR(255)	NOT NULL
USER_ID	INT	NOT NULL
CONTENT	TEXT	NOT NULL
IMAGE	NVARCHAR(255)	DEFAULT 'default_profile.png'
PUBLISHED_AT	DATETIME	DEFAULT CURRENT_TIMESTAMP

Subscriptions – содержит информацию о подписках пользователей. Её структура представлена в таблице 2.5;

Таблица 2.5 – Структура таблицы SUBSCRIPTIONS

Столбец	Тип данных	Ограничение целостности
ID	INT	PRIMARY KEY IDENTITY
AUTHOR	NVARCHAR(255)	NOT NULL

Продолжение таблицы 2.5

Столбец	Тип данных	Ограничение целостности
USER_ID	INT	NOT NULL
CONTENT	TEXT	NOT NULL
IMAGE	NVARCHAR(255)	DEFAULT 'default_profile.png'
PUBLISHED_AT	DATETIME	DEFAULT CURRENT_TIMESTAMP

Comments – содержит информацию о комментариях к публикациям. Её структура представлена в таблице 2.6;

Таблица 2.6 – Структура таблицы COMMENTS

Столбец	Тип данных	Ограничение целостности
ID	INT	PRIMARY KEY IDENTITY
AUTHOR	NVARCHAR(255)	NOT NULL
USER_ID	INT	NOT NULL
CONTENT	TEXT	NOT NULL
IMAGE	NVARCHAR(255)	DEFAULT 'default_profile.png'
PUBLISHED_AT	DATETIME	DEFAULT CURRENT_TIMESTAMP

Support – содержит информацию о проблемах, с которыми пользователи обратились в техническую поддержку. Её структура представлена в таблице 2.7.

Таблица 2.7 – Структура таблицы SUPPORT

Столбец	Тип данных	Ограничение целостности
ID	INT	PRIMARY KEY IDENTITY
PROBLEM_MESSAGE	NVARCHAR(MAX)	NOT NULL
USER_ID	INT	NOT NULL
DATE	DATETIME	DEFAULT CURRENT_TIMESTAMP

В данной главе была спроектирована архитектура приложения, включая серверную и клиентскую часть, а также базы данных. Скрипт создания всех таблиц представлен в приложении Г.

3 Разработка web-приложения

3.1 Обобщенная структура приложения

Структура приложения представлена на рисунке 3.1.

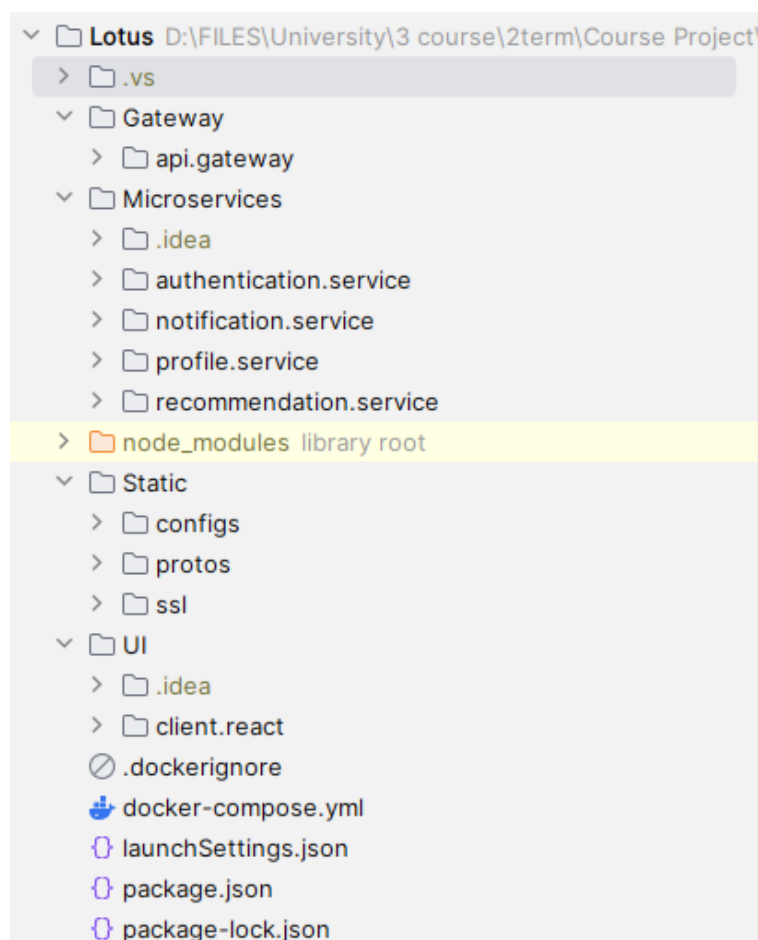


Рисунок 3.1 – Структура приложения

В папке Static хранятся статические данные, общие для всего приложения. В дочерней папке configs хранятся файлы конфигурации json, в которых содержатся данные, используемые в тех или иных файлах для конфигурации различных утилит приложения, таких как nodemailer для отправки сообщений на почту. В дочерней папке protos содержатся proto файлы – схемы данных Protobuf для использования технологии GRPC. В папке ssl находятся сертификаты для настройки https серверов.

В папке Microservices находятся собственно сами сервисы приложения, реализованные на Коа.

В папке UI находится клиент React, который подробно описан в главе 3.3.

3.2 Разработка серверной части приложения

Серверная часть приложения обеспечивает обработку клиентских запросов, выполняет запросы к базе данных, обеспечивает аутентификацию и авторизацию, а также поддерживает websocket сервер.

Структура серверного приложения на примере сервиса профилей представлена на рисунке 3.2.

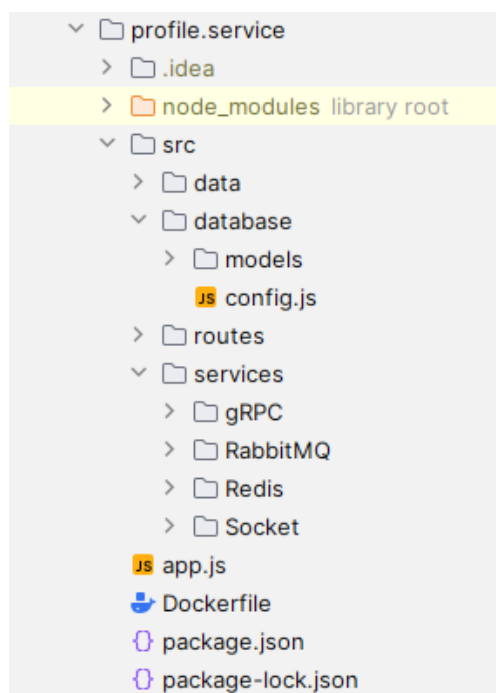


Рисунок 3.2 – Структура серверного приложения на примере сервиса профилей

Входной точкой каждого сервера является файл `app.js`. В нем происходит инициализация Кoa-сервера, websocket сервера (при наличии), а также установка сертификатов для обеспечения защищенного соединения с сервером.

В директории `database` находятся модели и файл конфигурации для настройки ORM-фреймворка `sequelize`. Модели описывают сущности базы данных и представляют интерфейс для взаимодействия с ней.

В директории `routes` находятся маршрутизаторы для каждой сущности. Пример маршрутизатора для сущности «User» представлен в листинге 3.1.

```
router.get('/api/user/subscriptions', koaJwt({ secret: SECRET_KEY
}), getUserSubscriptions);
router.get('/api/user/subscribers', koaJwt({ secret: SECRET_KEY }),
getUserSubscribers); router.post('/api/user/subscribe', koaJwt({
secret: SECRET_KEY }), subscribeUser);
router.post('/api/user/unsubscribe', koaJwt({ secret: SECRET_KEY }),
unsubscribeUser);
router.post('/api/user/support', koaJwt({ secret: SECRET_KEY }),
support);
```

Листинг 3.1 – Маршрутизатор для сущности «User»

С каждым маршрутом сопоставляется определенный метод, который непосредственно осуществляет обработку запроса. Пример метода `support`, который обрабатывает запрос пользователя в техническую поддержку представлен в листинге 3.2.

```

async function support(ctx) {
  const user_id = ctx.state.user.user_id;
  const username = ctx.state.user.username;
  const email = ctx.state.user.email;
  const { problem_message } = ctx.request.body;

  try {
    const newSupportMessage = await SUPPORT.create({
      USER_ID: user_id,
      PROBLEM_MESSAGE: problem_message
    });

    const data = {
      EMAIL: email,
      USERNAME: username,
      PROBLEM_MESSAGE: problem_message
    }
    sendToQueue("SupportEmailNotificationQueue", data);

    ctx.status = 201;
    ctx.body = { message: 'Support message saved successfully',
supportMessageId: newSupportMessage.ID };
  } catch (error) {
    console.error('Error saving support message:', error);
    ctx.status = 500;
    ctx.body = { message: 'Internal Server Error' };
  }
}

```

Листинг 3.2 – Метод support

Кроме конечных точек приложения, маршрутизаторы также используют промежуточную обработку запроса за счет использования `middleware`.

В приложении используется `middleware koaJwt`, который имеют стандартную реализацию в библиотеке «`koa-jwt`», и используется для аутентификации и авторизации. Этот `middleware` необходим для того, чтобы защитить некоторые маршруты от не аутентифицированного доступа.

3.2.1 Микросервисы

Приложение разделено на множество сервисов, что позволило сделать проект модульным. Исходя из этого, многие реализации являются независимыми, что позволяет легко менять либо подставлять другую реализацию.

GRPC — это открытый протокол для создания удаленного вызова процедур (RPC) с высокой производительностью, поддержкой различных языков и платформ. Он позволяет легко создавать и обновлять клиент-серверные приложения с использованием простых интерфейсов.

RabbitMQ - это система управления сообщениями, которая позволяет легко создавать и управлять очередями сообщений для обмена данными между

микросервисами. Он обеспечивает надежную и масштабируемую доставку сообщений между приложениями.

Вместе эти инструменты позволяют создать микросервисную архитектуру для веб-приложения издательской платформы. Микросервисы обрабатывают логику обработки различных функций, таких как аутентификация, авторизация, отправка и получение сообщений, хранение новостей и информации о профилях пользователей. Коммуникация между микросервисами осуществляется с помощью RabbitMQ и GRPC, что обеспечивает высокую производительность и масштабируемость.

3.2.2 Сервис аутентификации

Сервис аутентификации представляет из себя небольшое приложение, которое обрабатывает запросы для аутентификации и авторизации пользователей. В данном случае сервис обеспечивает процесс регистрации пользователей, генерацию уникального кода для верификации почты, процесс самой верификации электронной почты и авторизации пользователей.

Пользователь сначала должен зарегистрироваться, заполнив необходимую информацию в форме регистрации, такую как имя пользователя, адрес электронной почты и пароль. После этого пользователь должен подтвердить свой адрес электронной почты, чтобы завершить регистрацию. Для этого сервис генерирует уникальный код и отправляет его на указанный при регистрации адрес электронной почты пользователя. За саму отправку сообщения на электронную почту отвечает сервис уведомлений. Отправка письма с кодом подтверждения осуществляется посредством вызова GRPC метода со стороны сервиса аутентификации.

Исходный код генерации уникального кода для подтверждения электронной почты и вызов GRPC метода для отправки письма представлен в листинге 3.3.

```
async function identifyUser(ctx) {
  const { username, email, password } = ctx.request.body;
  const existingUser = await USER.findOne({ where: { EMAIL: email } });

  if (existingUser) {
    ctx.status = 400;
    ctx.body = { message: 'The user already exists' };
    return;
  }

  const code = Math.floor(100000 + Math.random() * 900000).toString();
  grpcVerifyEmail(username, email, `Please confirm your email address using the entered code: ${code}`);

  ctx.body = {
    code: code
  };
}
```

Листинг 3.3 – Код генерации уникального кода для подтверждения почты

После получения уникального кода, пользователь должен ввести его на странице подтверждения электронной почты. После чего пользователь может использовать свой логин и пароль для входа в систему. После успешной аутентификации пользователь получает сгенерированный JWT токен.

Исходный код генерации JWT токена для представлен в листинге 3.4.

```
const token = jwt.sign({
  user_id: newUser.ID,
  username,
  email
}, secretKey, { expiresIn: '1h' });

ctx.status = 201;
ctx.body = {
  message: 'The user has been successfully registered',
  username: newUser.USERNAME,
  token: token
};
```

Листинг 3.4 – Код генерации JWT токена

Этот код генерирует токен доступа для только что зарегистрированного пользователя. В качестве полезной нагрузки (Payload) в токене хранится идентификатор пользователя, его наименование в системе и адрес его электронной почты. Затем он генерирует токен с помощью этого списка утверждений и указанного времени жизни токена. Наконец, он преобразует токен в строку и возвращает его в теле ответа.

После регистрации нового пользователя сервис аутентификации извлекает информацию о нем и сохраняет ее в своей базе данных. Затем он генерирует сообщение и передает его в RabbitMQ для последующей обработки пользователя на других сервисах.

3.2.3 Сервис уведомлений

Сервис уведомлений, как можно понять из названия, отвечает за отправку и доставку уведомлений как в личный кабинет пользователя, так и на почту. В его основе лежит класс Mailer, который инкапсулирует в себе логику отправки сообщений. Класс Mailer представлен в приложении Д.

Для использования функционала сервиса уведомлений, другие сервисы, в зависимости от типа уведомлений, должны послать сообщение в ту или иную очередь сообщений брокера RabbitMQ. Сервис уведомлений прослушивает все очереди и, при наличии нового сообщения, отправляет уведомление пользователю.

В качестве примера разберем системные уведомления. В контексте сервиса уведомлений была разработана функция connectRabbitMQ, которая прослушивает очереди сообщений. При наличии нового сообщения в той или иной очереди отправляет сообщение на почту при помощи класса Mailer, а также сохраняет уведомление в базе данных. Код отправки системных уведомлений представлен в листинге 3.5.

```

const QUEUES = {
  NotifyUserRegisteredQueue: 'NotifyUserRegisteredQueue',
  UserNotificationQueue: 'UserNotificationQueue',
  SystemNotificationQueue: 'SystemNotificationQueue',
  EmailNotificationQueue: 'EmailNotificationQueue',
  LastEmailNotificationQueue: 'LastEmailNotificationQueue',
  SupportEmailNotificationQueue: 'SupportEmailNotificationQueue'
};

function connectRabbitMQ() {
  amqp.connect(`amqp://${RABBITMQ_HOST}:${RABBITMQ_PORT}`,
function(error0, connection) {
  if (error0) {
    throw error0;
  }

  connection.createChannel(function(error1, channel) {
    if (error1) {
      throw error1;
    }

    Object.values(QUEUES).forEach(queue => {
      channel.assertQueue(queue, { durable: false });
    });

    channel.consume(QUEUES.SystemNotificationQueue, async
function (msg) {
  const { EMAIL: email, USER_ID: user_id, USERNAME:
username, MESSAGE: message } = JSON.parse(msg.content.toString());
  Mailer.sendEmailSystemNotification(email, username,
message);

  await NOTIFICATION.create({
    AUTHOR: "System notification",
    CONTENT: message,
    USER_ID: user_id,
    IMAGE: "system_notification.png",
  });
}, { noAck: true });
  // Обработка иных данных...
});
});
}

```

Листинг 3.5 – Код отправки системных уведомлений

В этом коде, сначала, функция пытается установить соединение с сервером RabbitMQ. Далее функция создает канал в этом соединении. Затем функция проходит по всем очередям, указанным в объекте QUEUES, и утверждает их в созданном канале. Это означает, что если очередь не существует, она будет создана. Далее происходит потребление сообщения из очереди и его последующая обработка.

3.2.4 Сервис профилей

Сервис профилей предназначен для работы с профилем пользователя. Он предоставляет API, через которое пользователь может кастомизировать свой профиль, создавать, удалять и изменять посты, взаимодействовать с профилями других пользователей, оставлять комментарии, обращаться в техническую поддержку и т.д.

В основе сервиса профилей лежат сущности Users, Posts, Comments, Subscriptions и Support. После успешной регистрации пользователя, сервис аутентификации отправляет MQ-сообщение в очередь «UserRegisteredQueue». Сервис профилей, соответственно, создает соединение с сервером RabbitMQ. Далее он создает канал в этом соединении и утверждает указанную очередь в созданном канале. Далее происходит потребление сообщения из очереди и на основе переданных данных создает пользователя в своей базе данных. После чего пользователя можно считать полноценно зарегистрированным в системе.

3.2.5 Сервис новостей

Сервис новостей отвечает хранению и получению новостей. Он предоставляет API, через которое клиент может получить новости.

Сервис новостей представляет из себя систему, в основе которой лежит веб-парсер, который позволяет в автоматическом режиме опрашивать электронные веб-ресурсы новостей. Это достигается путем периодического запроса HTML-кода страницы новостей и его последующего анализа с помощью библиотеки cheerio. Парсер проходит по каждому HTML-элементу и извлекает текст из элементов с определённым селектором. Далее эти данные сохраняются в базу данных. После чего, клиент, при помощи http-запроса может получить список последних новостей. Код парсера новостей представлен на листинге 3.6.

```
function scheduleJob(topic) {
  cron.schedule('0 * * * *', async () => {
    const response = await
    axios.get(`https://ria.ru/${topic}/`);
    const $ = cheerio.load(response.data);
    $('.list-item__title').each(async (index, element) => {
      if (index >= 20) { return false; }
      const title = $(element).text();
      const link = $(element).attr('href');
      const image = $('.responsive_img.m-list-
img').eq(index).attr('src');
      const alt = $('.responsive_img.m-list-
img').eq(index).attr('alt');
      const date = $('.list-item__date').eq(index).text();
      await NEWS.create({TITLE: title, PARAGRAPH: alt,
ALT: alt, LINK: link, IMAGE: image, DATE: date, TOPIC_NAME:
topic});});
    });}
```

Листинг 3.6 – Код парсера новостей

3.2.6 API Gateway

Gateway – это компонент, который действует как единая точка входа в систему. Он обрабатывает внешние запросы и перенаправляет их к соответствующим микросервисам.

В контексте курсового проекта, был разработан API Gateway на базе Node.js и фреймворка Кoa, который выполняет ряд ключевых функций. Он настраивает прокси для перенаправления запросов к четырем различным микросервисам. Каждый из этих микросервисов имеет свой собственный порт и конечные точки API. Это достигается благодаря настройке маршрутов в Koa Router, который перенаправляет все запросы, начинающиеся с «/api/profile/», «/api/auth/», «/api/notify/» и «/api/news/», к соответствующим прокси-серверам. Пример перенаправление маршрутов представлен на листинге 3.7.

```
const proxyRequest = (ctx, target) => {
  return new Promise((resolve, reject) => {
    const agent = new https.Agent({ rejectUnauthorized: false });
    const options = { hostname: target.hostname, port:
target.port, path: ctx.url, method: ctx.method, headers:
ctx.request.header, agent: agent };
    const req = https.request(options, (res) => {
      let data = '';
      res.on('data', (chunk) => { data += chunk;});
      res.on('end', () => {
        ctx.status = res.statusCode;
        ctx.set(res.headers);
        ctx.body = data;
        resolve();});});
    router.all('/api/auth/(.*)', (ctx) => {
      return proxyRequest(ctx, { hostname: 'localhost', port:
31901 });
    });
  });
};
```

Листинг 3.7 – Пример перенаправление маршрутов

Разработанный API Gateway также включает обработку ошибок. Если во время обработки запроса происходит ошибка, она будет выброшена и обработана средствами Koa. Важной составляющей API Gateway является безопасность. Он использует JSON Web Tokens (JWT) для аутентификации пользователей. Это достигается с помощью промежуточного программного обеспечения jwtMiddleware, которое проверяет наличие JWT в заголовках авторизации входящих запросов.

Также, API Gateway настроен на использование HTTPS для обеспечения безопасности данных. Это достигается с помощью модуля https, который создает HTTPS-сервер с использованием сертификата SSL и ключа, хранящихся в файловой системе.

В общем и целом, API Gateway играет важную роль в микросервисной архитектуре. Он обеспечивает единую точку входа для всех микросервисов и обрабатывает различные аспекты безопасности и маршрутизации.

3.3 Разработка клиентской части приложения

Клиентская часть приложения предоставляет интерфейс для взаимодействия с серверным приложением. Клиентская часть представляет собой набор React-компонентов, и является одновременно http и websocket-клиентом.

Структура клиентского приложения представлена на рисунке 3.3.

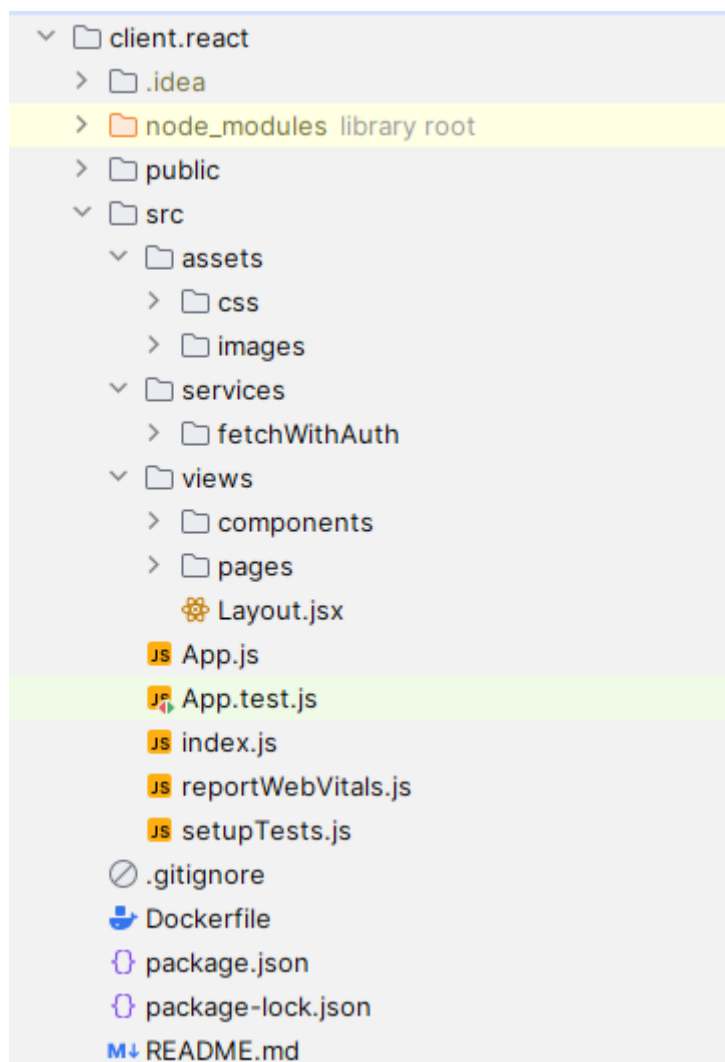


Рисунок 3.3 – Структура клиентского приложения

Директория `src` содержит весь исходный код приложения.

Директория `pages` содержит все React-страницы, к которым пользователь может получить доступ.

Директория `components` содержит все React-компоненты, повторно используемые в приложении.

Директория `services` содержит функции с дополнительным функционалом.

Директория `assets` содержит статические данные, используемые в приложении, такие как иконки, стили CSS и т.п.

В основном файле проекта `App.js` прописана маршрутизация приложения, реализованная при помощи библиотеки «react-router». Содержимое файла `App.js` представлены на листинге 3.8.

```

function App() {
  useEffect(() => AOS.init , []);

  return (
    <>
      <Routes>
        <Route path="/" element={<Layout/>}>
          <Route index element={<HomePage/>} />
          <Route path="home" element={<HomePage/>} />
          <Route path="news" element={<NewsPage/>} />
          <Route path="news/:topic" element={<NewsPage/>} />
          <Route path="news/:id" element={<SelectedNews/>} />
          <Route path="support" element={<SupportPage/>} />
          <Route path="profile/:username" element={<ProfilePage/>} />
          <Route path="profile/:username/edit"
element={<ProfileEditPage/>} />
          <Route path="profile/:username/change-password"
element={<ChangePasswordPage/>} />
          <Route path="/:username/:postid/comments"
element={<CommentsPage/>} />
          <Route path="/about" element={<AboutPage/>} />
          <Route path="/people" element={<PeoplePage/>} />
          <Route path="/saved" element={<SavedPage/>} />
          <Route path="/subscriptions"
element={<SubscriptionsPage/>} />
          <Route path="/notifications"
element={<NotificationsPage/>} />
        </Route>
        <Route path="login" element={<AuthenticationPage />} />
        <Route path="register" element={<RegistrationPage />} />
        <Route path="*" element={<NotFoundPage />} />
      </Routes>
    </>
  );
}
export default App;

```

Листинг 3.8 – Содержимое файла App.js

Также, файл App.js помимо инициализации всех остальных компонентов приложения он инициализирует websocket-клиента. Структура web-socket клиента представлена в листинге 3.9

```

useEffect(() => {
  socket.on(`new_comment_${CURRENT_USER_NAME}`, (data) => {
    setShowToast(true); setTime(data.time);
    setAuthor(data.author); setImage(data.image);
    setMessage(data.message);
  });
//...
}, []);

```

Листинг 3.9 – Инициализация websocket-клиента

Также, были использованы сторонние библиотеки и фреймворки для разработки визуального интерфейса приложения.

Bootstrap – это библиотека, которая предоставляет готовые компоненты и классы для быстрой и удобной стилизации веб-страниц. Мы использовали Bootstrap для создания адаптивного и красивого дизайна нашего приложения. Bootstrap позволяет легко создавать сетки, стилизовать формы, кнопки, карточки и многое другое. Это значительно ускоряет процесс разработки и обеспечивает единообразие стилей.

AOS (Animate On Scroll) – это небольшая библиотека, которая позволяет анимировать элементы, когда они появляются в области просмотра. Мы использовали AOS для добавления динамических анимаций в наше приложение. Это делает интерфейс более живым и интересным для пользователя.

4 Тестирование web-приложения

В данном разделе описано тестирование приложения с использованием клиентской части.

4.1 Ошибка при вводе несуществующего маршрута

Любое Web-приложение должно грамотно обрабатывать типичные ошибки пользователей, в том числе и ошибку 404 – Не найдено. Визуальное отображение в клиентском браузере ошибки при вводе несуществующего маршрута представлено на рисунке 4.1.

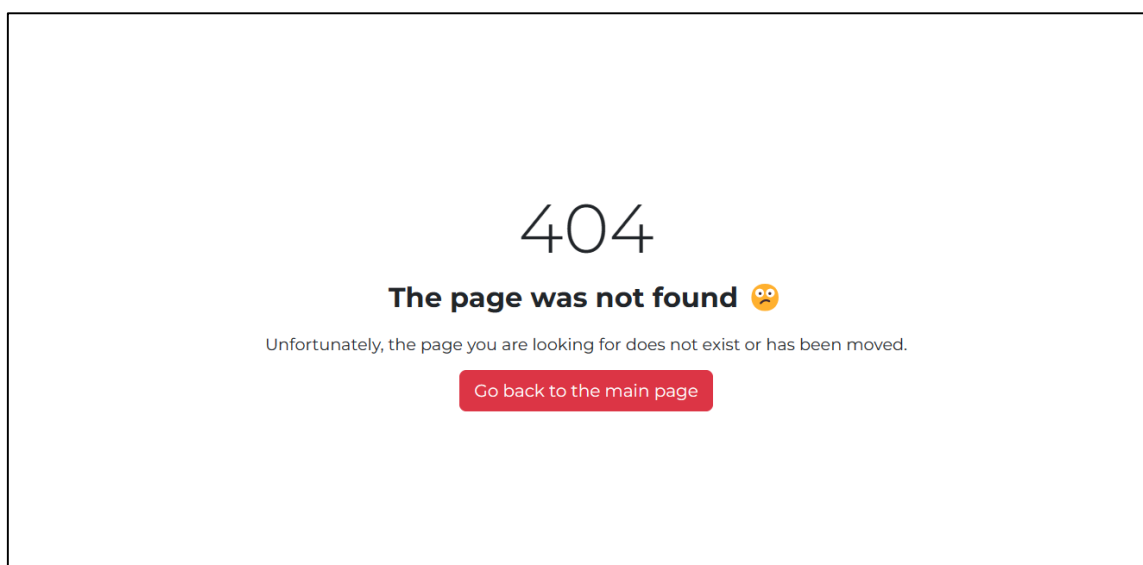


Рисунок 4.1 – Ошибка при вводе несуществующего маршрута

4.2 Ошибка при пустом значении полей

Во всем приложении, в местах, где этого требует логика, при вводе пустых значений в поля для ввода происходит проверка на наличия символов в поле. Ошибка отображена на рисунке 4.2.

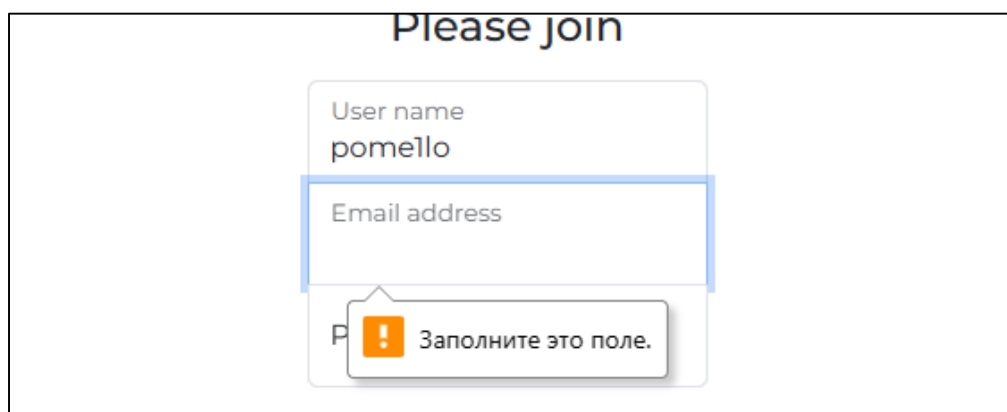


Рисунок 4.2 – Ошибка при пустом значении поля

4.3 Ошибка при некорректных данных для входа

При предоставлении некорректных данных для входа (неверный логин или пароль) в правом нижнем углу появляется уведомление о соответствующей ошибке. Ошибка представлена на рисунке 4.3.

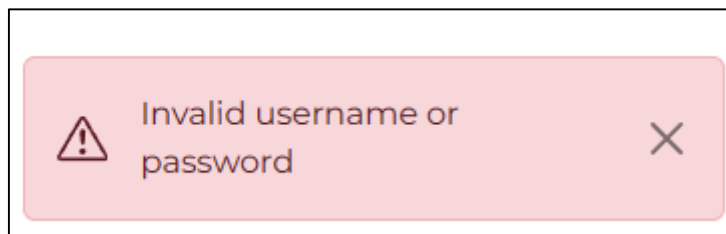


Рисунок 4.3 – Ошибка при некорректных данных для входа

4.4 Ошибка при регистрации существующего логина

При регистрации пользователя может возникнуть ситуация, при которой данная электронная почта или имя пользователя уже занято другим пользователем. Ошибка отображена на рисунке 4.4.

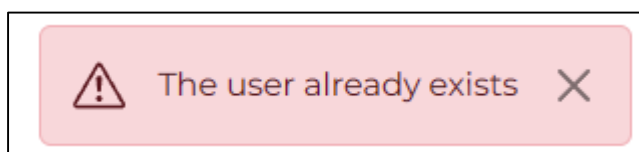


Рисунок 4.4 – Ошибка при регистрации существующего email

4.5 Иные ошибки

При возникновении иных ошибок на сервере она будет выведена в компактное уведомление в правом нижнем углу окна браузера. Ошибка представлена на рисунке 4.5.

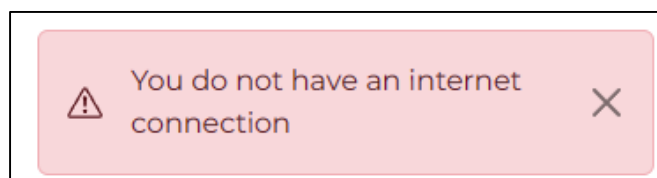


Рисунок 4.5 – Ошибка при регистрации существующего email

При попытке не аутентифицированного доступа к ресурсу пользователя перенаправит на страницу для входа.

Таким образом было проведено тестирование приложения. Были продемонстрированы типичные ситуации, вызывающие ошибки и реакция на них. Результаты тестирования позволили выявить и исправить множество ошибок и проблем, связанных с функциональностью и производительностью приложения.

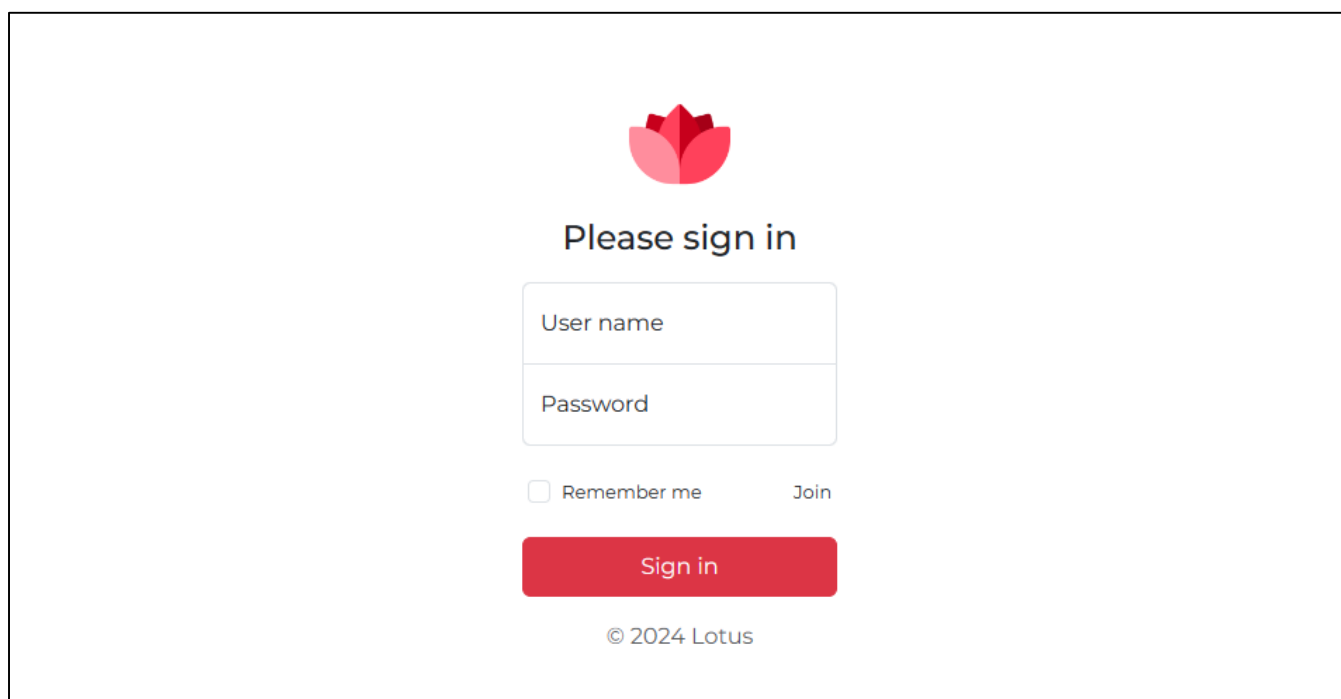
5 Руководство пользователя


Руководство пользователя является важным компонентом любого веб-приложения, поскольку оно предоставляет пользователю необходимую информацию о функционале приложения, его возможностях и способах использования. Руководство пользователя помогает пользователям быстро освоиться с интерфейсом и функциями приложения, снижает уровень путаницы и упрощает процесс взаимодействия с приложением.

Далее в руководстве пользователя будут рассмотрены ключевые функциональные возможности веб-приложения. Они включают в себя функционал логина, регистрации, создания и изменения постов, просмотр и персонализация страницы пользователя, просмотр новостей, просмотр страницы подписок и подписчиков.

5.1 Вход в аккаунт

После нажатия на кнопку «Login» пользователь попадает на страницу входа, где ему необходимо ввести электронную почту и пароль в поля для ввода, и нажать на кнопку «Sign in». Страница логина представлена на рисунке 5.1.





Please sign in

User name

Password

☐ Remember me [Join](#)

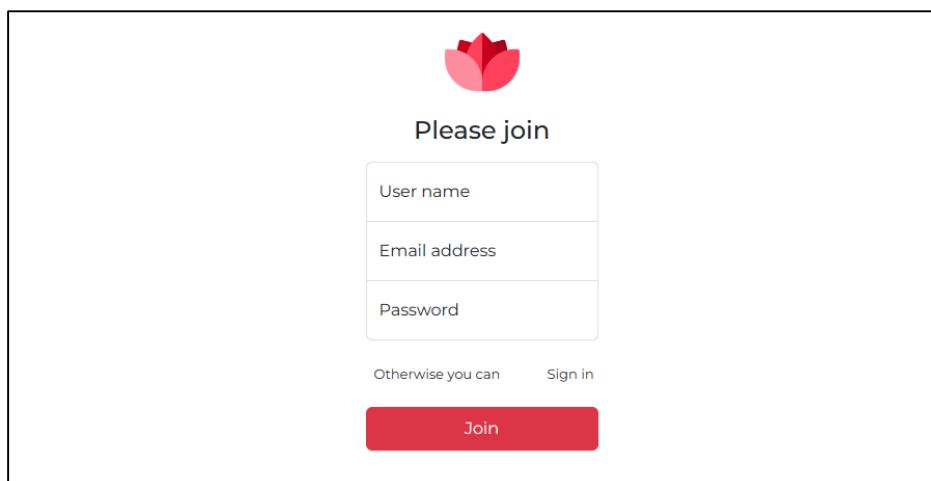
Sign in


© 2024 Lotus

Рисунок 5.1 – Страница входа

5.2 Регистрация пользователя

Для перехода на страницу регистрации на странице входа необходимо нажать кнопку «Join». Регистрация предоставляет возможность новым пользователям создать учетную запись, заполнив необходимую информацию со своими учетными данными. Далее необходимо нажать на кнопку «Join». Страница регистрации представлена на рисунке 5.2.




 Please join

User name
Email address
Password

[Otherwise you can](#) [Sign in](#)

[Join](#)

Рисунок 5.2 – Страница регистрации

5.3 Взаимодействие с профилем пользователя

Для дальнейшего взаимодействия пользователя с постами и своим профилем ему необходимо зайти на страницу своего профиля. Здесь отображается Username пользователя, количество постов, подписок и подписчиков, текст описание, а также список опубликованных им постов. С этой страницы пользователь может создать новый пост, перейти в настройки своего аккаунта, а также отредактировать персональную информацию. Страница профиля пользователя представлена на рисунке 5.3.

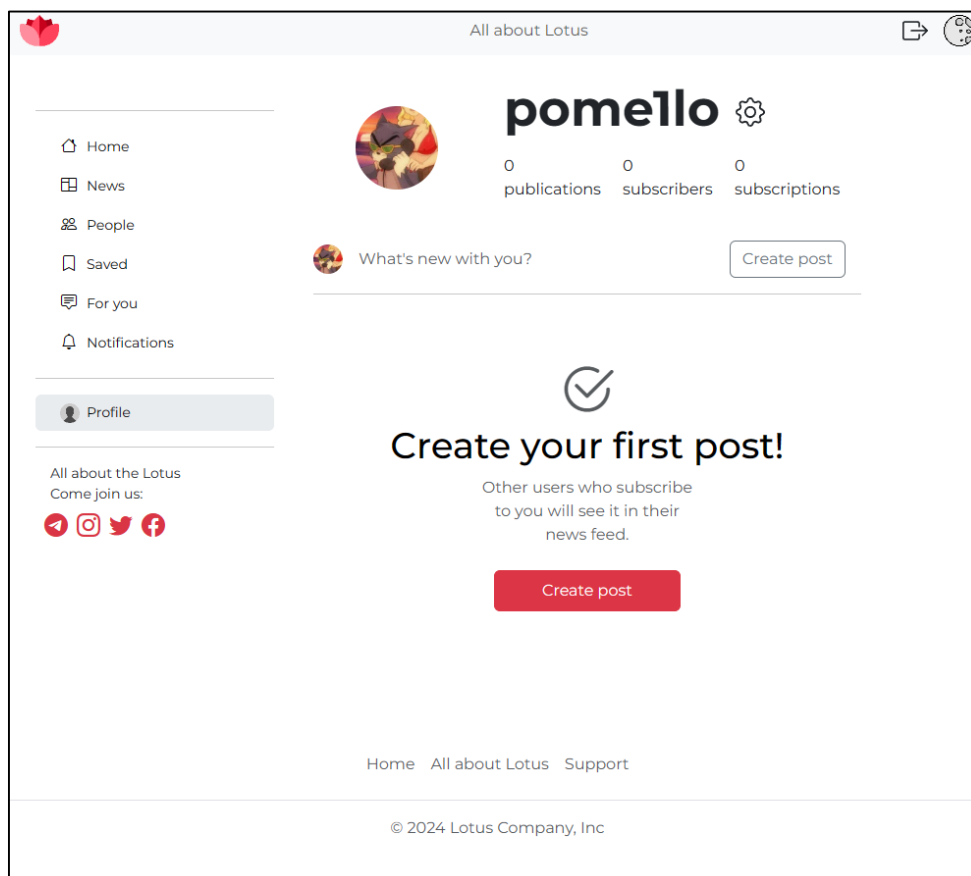


Рисунок 5.3 – Страница профиля пользователя

5.4 Создание поста

Для создания поста пользователю необходимо нажать на кнопку «Create post» и дождаться появления модального окна. В появившемся окне пользователь может создать новый пост. Ему необходимо ввести название поста, его содержимое и прикрепить картинку. Дата создания поста определяется автоматически. Нажатием на кнопку «Create» пользователь создает и одновременно публикует новый пост. Модальное окно для создания поста отображено на рисунке 5.4.

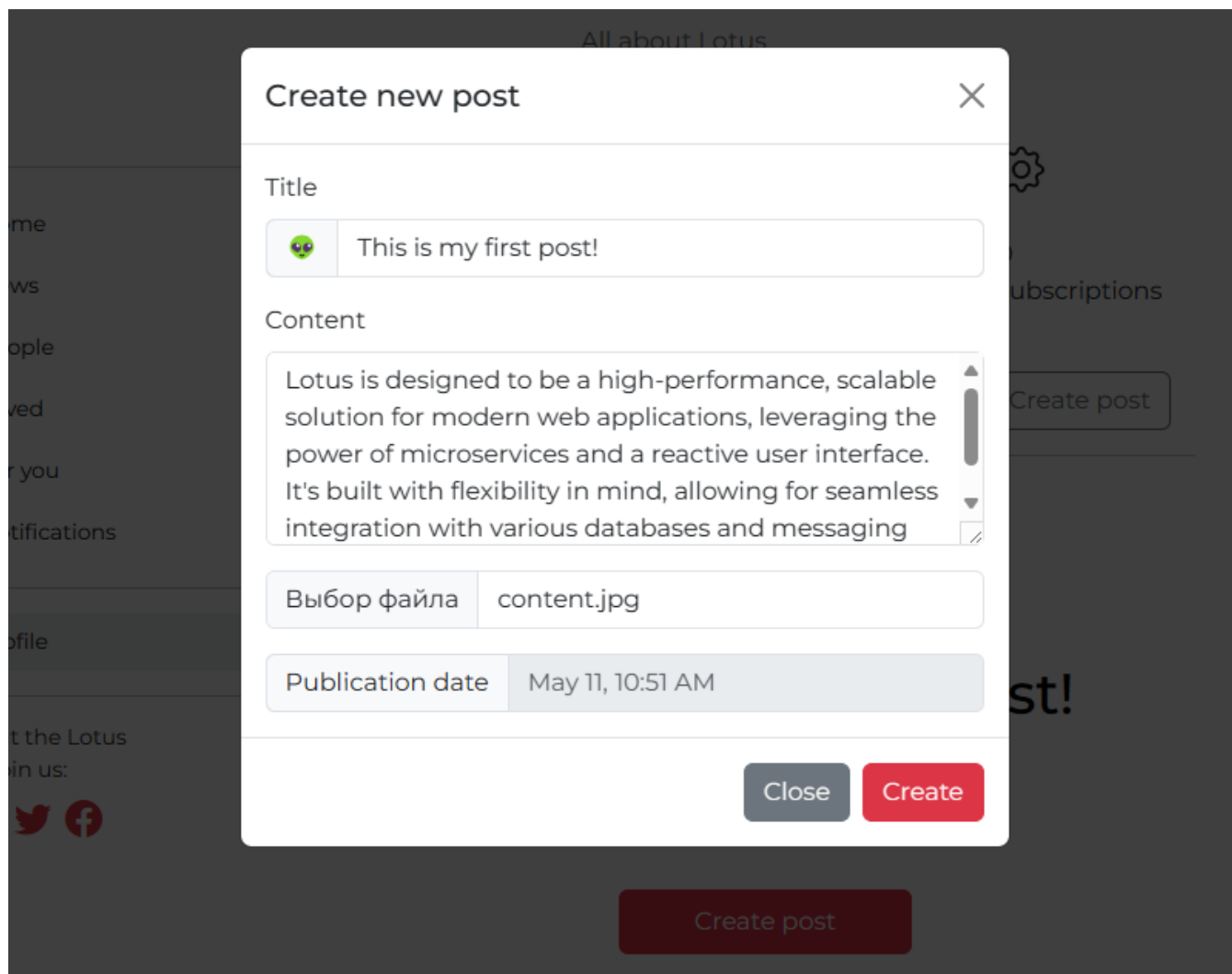


Рисунок 5.4 – Модальное окно для создания поста

5.5 Редактирование поста

На странице профиля пользователя в списке постов при нажатии на шестерёнку на одном из них появляется модальное окно редактирования поста. В нём пользователь может изменить заголовок, содержимое поста и прикрепить другую картинку. Также пользователь имеет возможность безвозвратно удалить пост при нажатии на кнопку «Delete post». При нажатии на кнопку «Edit post» пользователь применит изменения. Модальное окно для редактирования представлено на рисунке 5.5.

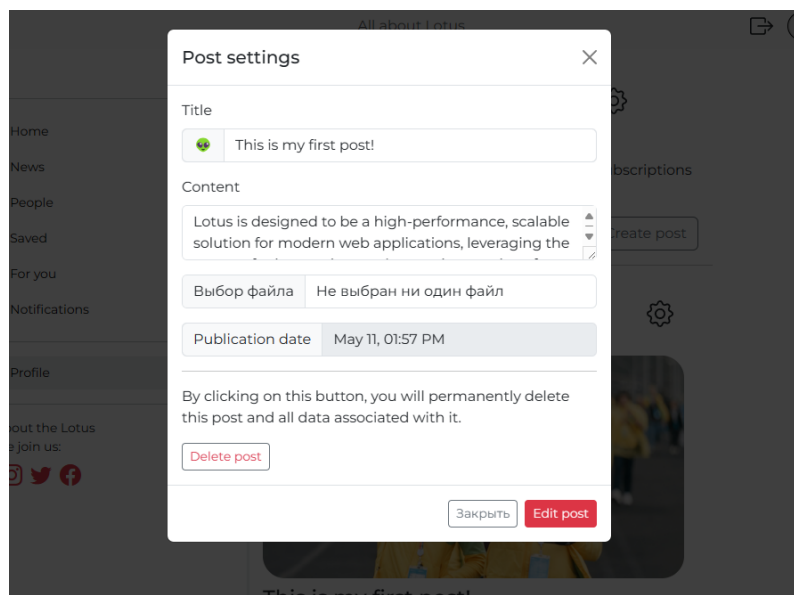


Рисунок 5.5 – Модальное окно для редактирования поста

5.6 Страница «People»

При переходе на страницу подписок и подписчиков пользователь может получить доступ к списку аккаунтов на которые он подписан и которые подписаны на него. При нажатии на аккаунт происходит перенаправление на профиль выбранного пользователя. Странице подписок и подписчиков представлена на рисунке 5.6.

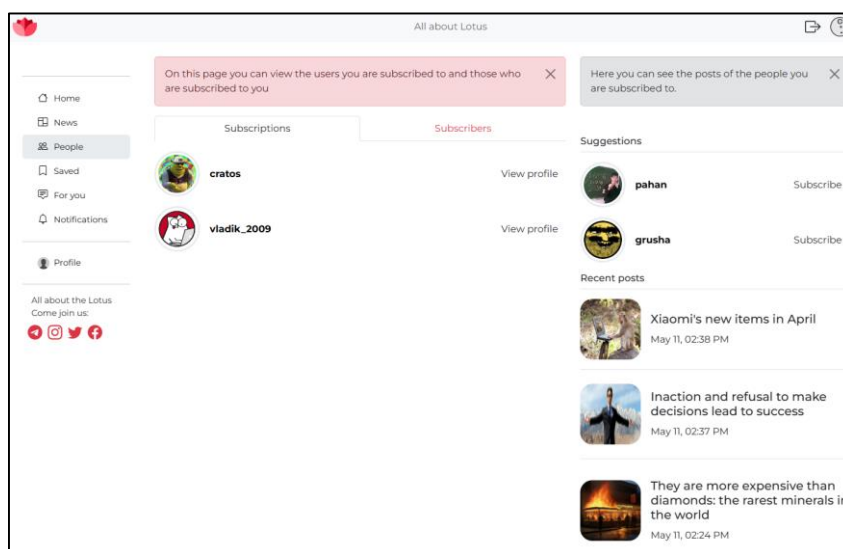


Рисунок 5.6 – Страница подписок и подписчиков

5.7 Страница новостей

При переходе на страницу новостей пользователь может просматривать последние новости, а также переключаться между интересующими его темами. Страница новостей представлена на рисунке 5.7.

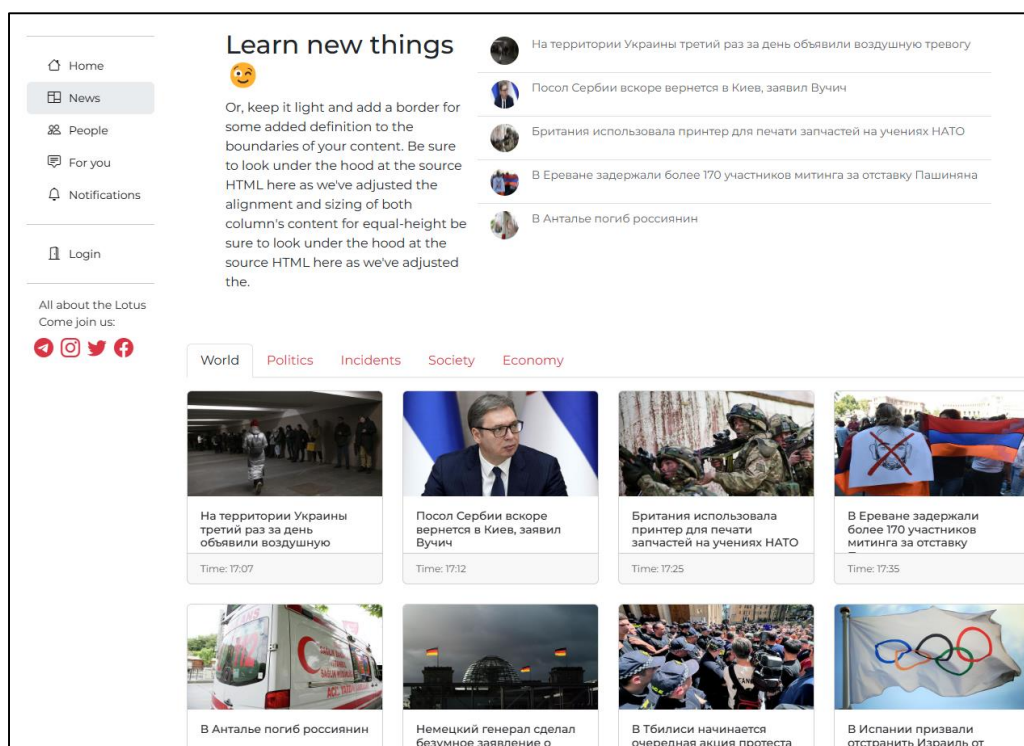


Рисунок 5.7 – Страница новостей

5.8 Страница «Для вас»

При переходе на страницу «Для вас» пользователь может просматривать последние посты пользователей, на которых он подписан. Страница «Для вас» представлена на рисунке 5.8.

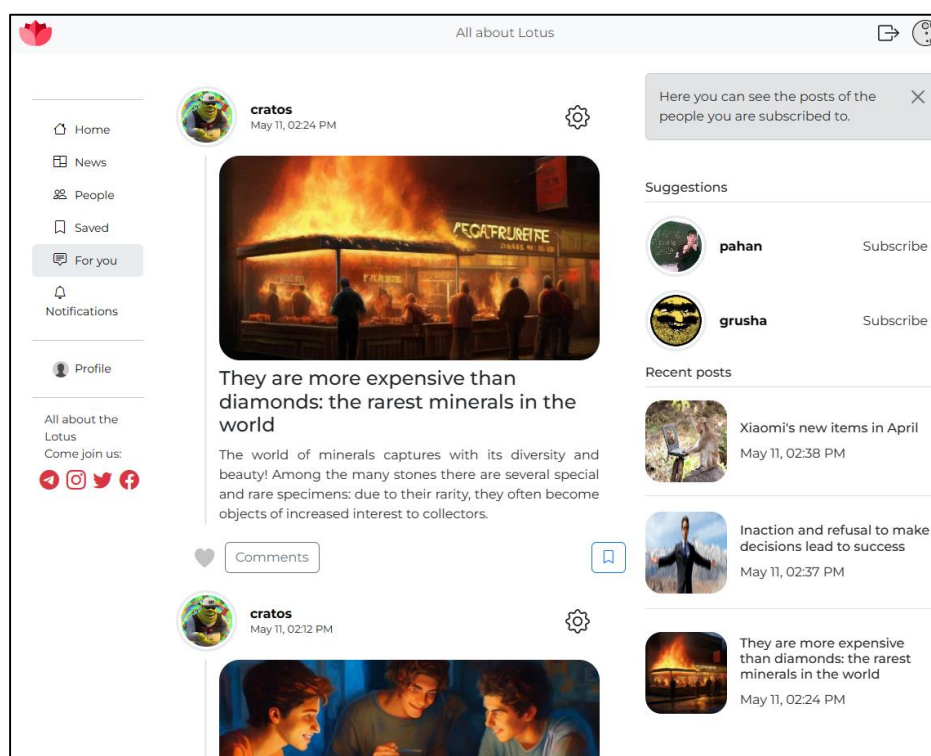


Рисунок 5.8 – Страница «Для вас»

5.9 Оставление комментариев

При клике на кнопку «Comments» под любым постом пользователя перенаправляет на страницу комментариев к этому посту. Здесь отображается картинка выбранного поста, его заголовок и текст, а также список комментариев других пользователей. Снизу находиться поле для ввода текста комментария и кнопка для отправки. Страница комментариев под одним из постов представлена на рисунке 5.9.

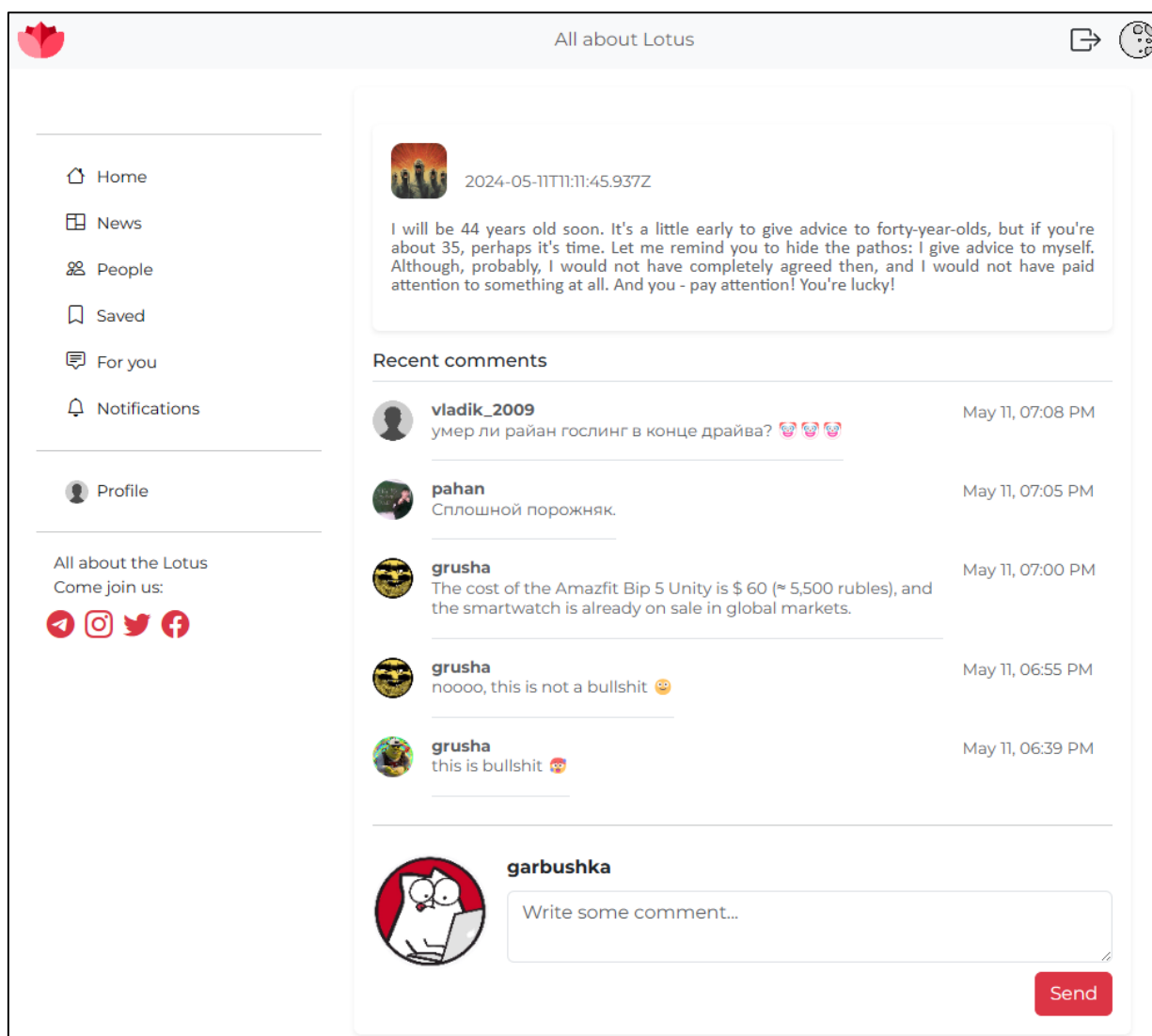


Рисунок 5.9 – Страница комментариев под одним из постов

5.10 Страница технической поддержки

При клике на кнопку «Support», в заголовке или подвале сайта, пользователь попадает на страницу связи с технической поддержкой. Здесь находится единственное поле для ввода, в которое пользователь может ввести свою проблему, и кнопка для отправки сообщения. Страница связи с технической поддержкой представлена на рисунке 5.10.

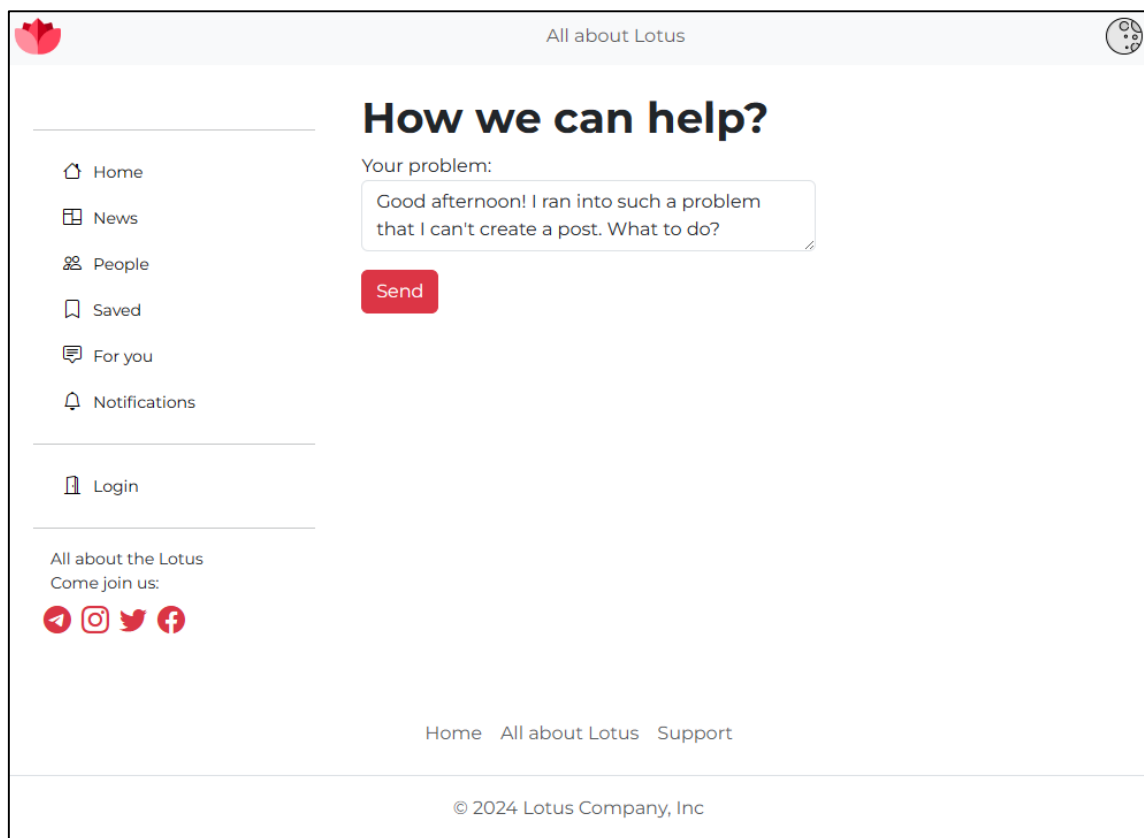


Рисунок 5.10 – Страница технической поддержки

Таким образом, в данной главе было разработано руководство по использованию, которое облегчает пользователю освоение функционала приложения.

Заключение

В ходе выполнения курсового проекта было разработано веб-приложение «Издательская платформа», сочетающее в себе функционал персонализированных рекомендаций и возможности издательской платформы. Проект успешно реализует поставленные задачи, предоставляя пользователям интуитивно понятный интерфейс для публикации контента, взаимодействия с другими пользователями и получения персонализированных рекомендаций.

В процессе разработки серверной части было реализовано API для работы со всеми необходимыми сущностями, также была реализована стратегия JWT для обеспечения безопасности.

Были реализованы следующие требования:

- обеспечивать возможность регистрации и авторизации;
- отправлять уведомления о новом контенте, который может быть интересен пользователю;
- иметь интуитивно понятный и удобный пользовательский интерфейс;
- предоставлять возможность пользователям публиковать свой контент в виде текста или изображений;
- предоставлять возможность пользователям оставлять комментарии к постам;
- предоставлять пользователю возможность кастомизировать свой профиль;
- предоставлять пользователю возможность редактировать и удалять собственные посты;
- предоставлять персонализированный контент, основанный на интересах и взаимодействиях пользователя;
- обеспечивать защиту всех пользовательских данных.

Для обеспечения безопасного обмена информации между клиентом и сервером был сгенерирован ssl-сертификат.

Важным этапом разработки было тестирование программного продукта, которое позволило выявить и исправить ошибки и недостатки в работе приложения. Тестирование позволило убедиться, что программа работает верно и соответствует требованиям.

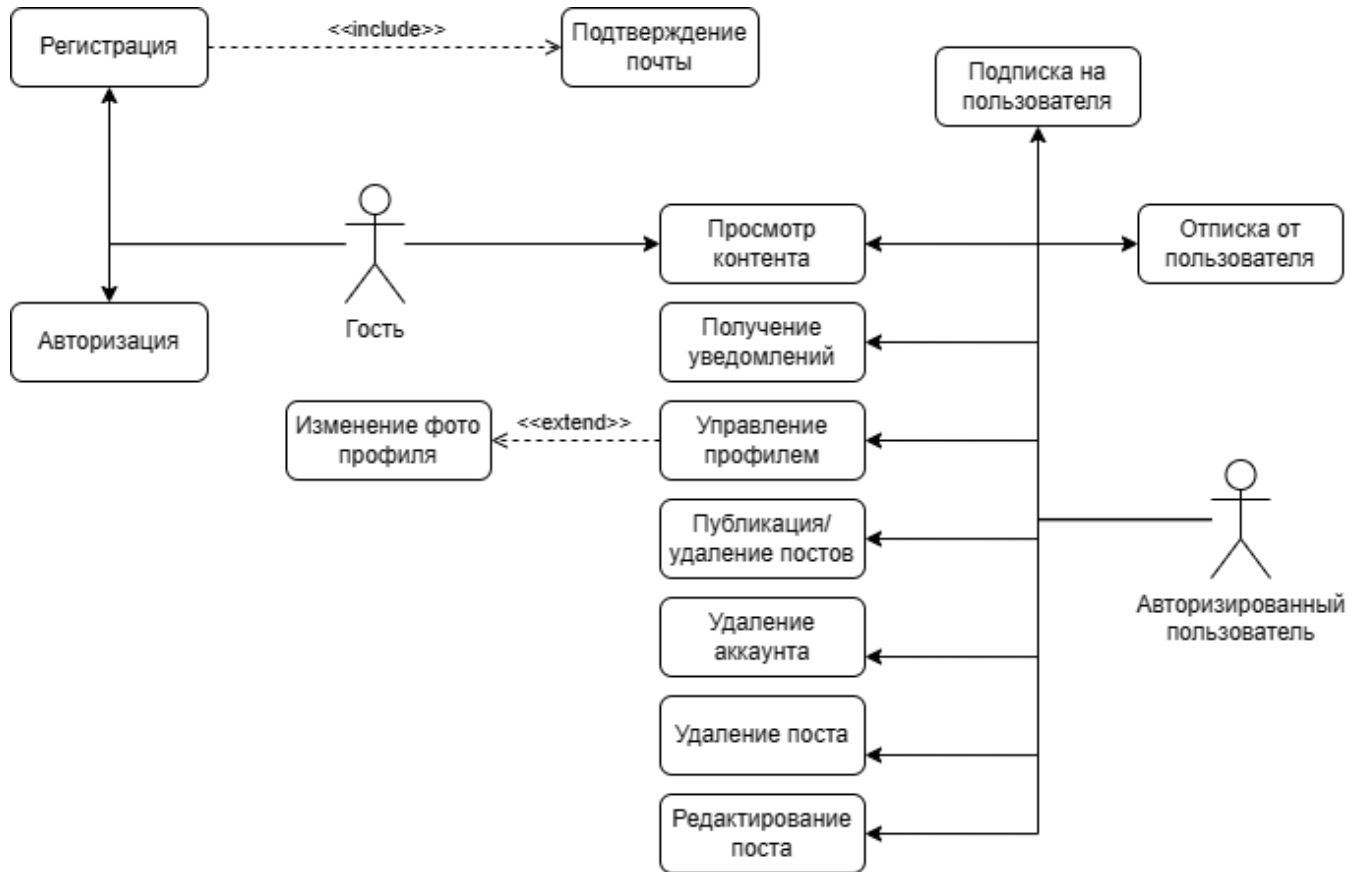
Проект демонстрирует эффективное применение современных технологий и подходов в разработке программного обеспечения и может служить основой для дальнейшего развития и внедрения новых функций.

Таким образом, цель данной работы была успешно достигнута. Созданное веб-приложение «Издательская платформа» успешно сочетает в себе функциональность и безопасность, предоставляя пользователям мощный инструмент для обмена и открытия нового контента. Это делает его ценным ресурсом как для авторов, так и для читателей, и открывает путь для дальнейших инноваций в области цифрового издательства.

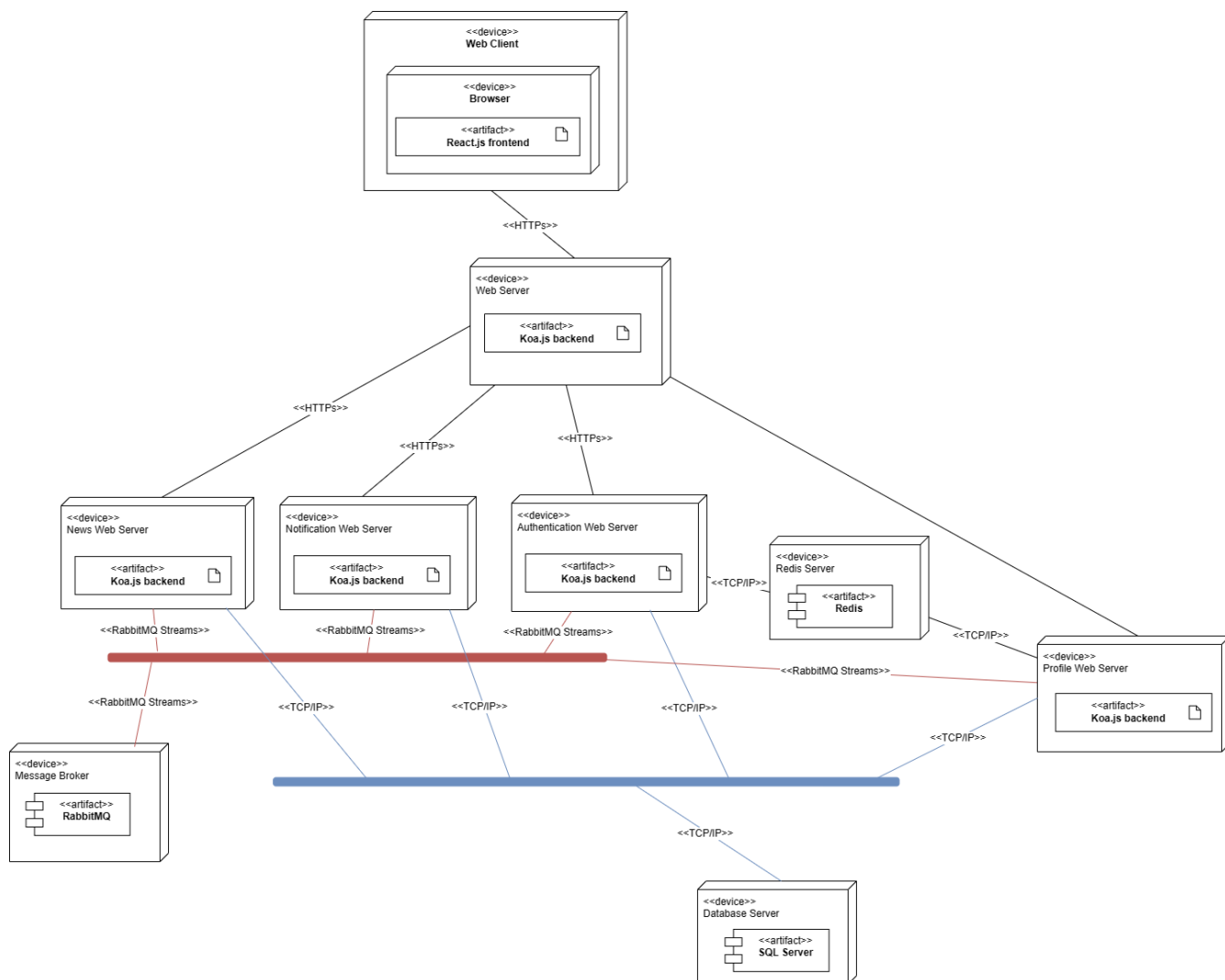
Список используемых источников

1. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: <https://metanit.com> – Дата доступа: 15.03.2024.
2. Koa.js [Электронный ресурс] / Режим доступа: <https://koajs.com>. – Дата доступа: 19.03.2024.
3. Stackoverflow.com [Электронный ресурс] / Режим доступа: <https://stackoverflow.com> – Дата доступа: 23.03.2024.
4. React.js документация [Электронный ресурс] / Режим доступа: <https://react.dev/learn>. – Дата доступа: 19.04.2024.
5. Socket.io документация [Электронный ресурс] / Режим доступа: <https://socket.io/docs/v4/>. – Дата доступа: 27.04.2024.

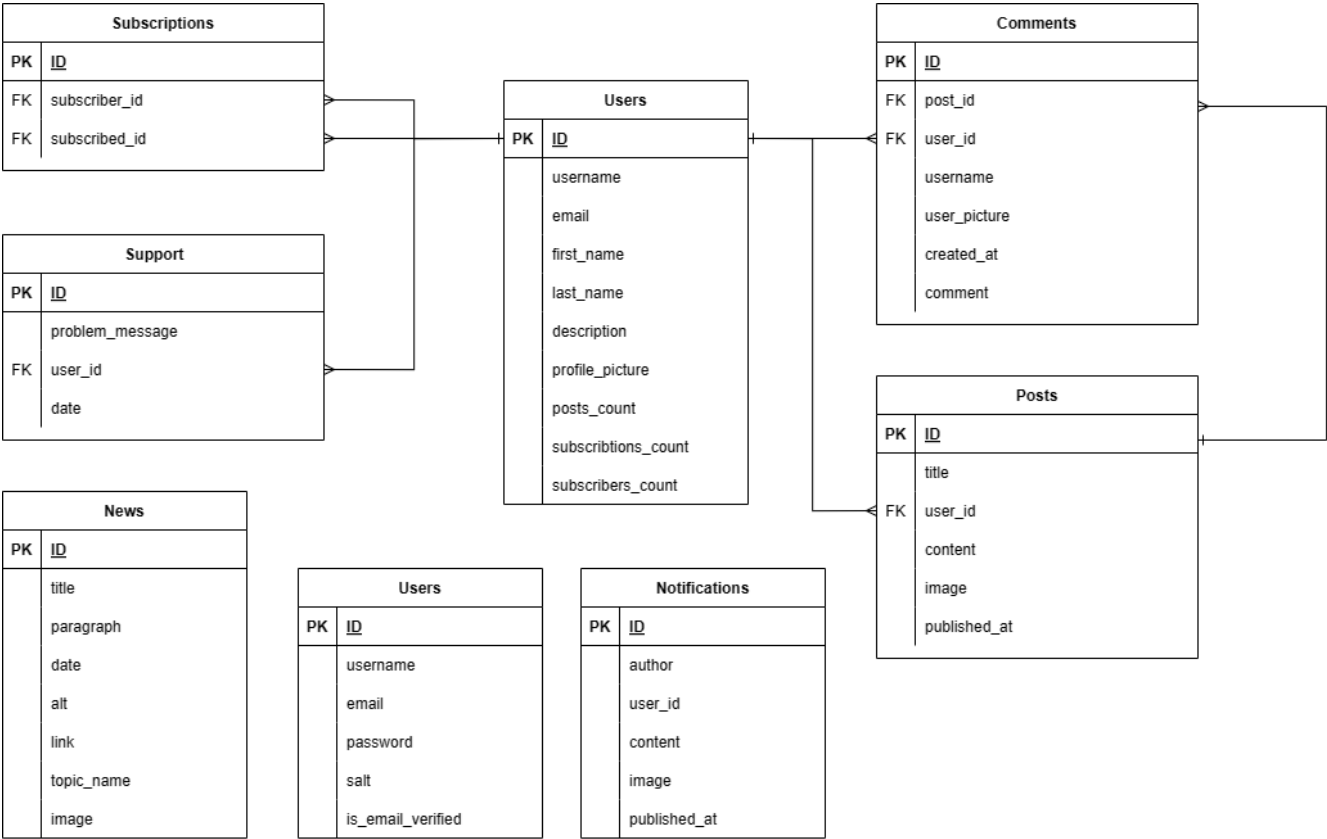
Приложение А



Приложение Б



Приложение В



Приложение Г

```
CREATE DATABASE AUTHENTICATION_SERVICE;
USE AUTHENTICATION_SERVICE;

CREATE TABLE USERS (
    ID INT PRIMARY KEY IDENTITY,
    USERNAME NVARCHAR(255) NOT NULL,
    EMAIL NVARCHAR(255) NOT NULL,
    PASSWORD NVARCHAR(255),
    SALT NVARCHAR(255),
    IS_EMAIL_VERIFIED BIT DEFAULT 0,
    VERIFICATION_TOKEN NVARCHAR(255),
);

CREATE DATABASE NOTIFICATION_SERVICE;
USE NOTIFICATION_SERVICE;

CREATE TABLE NOTIFICATIONS
(
    ID INT PRIMARY KEY IDENTITY,
    AUTHOR NVARCHAR(255) NOT NULL,
    USER_ID INT NOT NULL,
    CONTENT TEXT NOT NULL,
    IMAGE NVARCHAR(255) DEFAULT 'default_profile.png',
    PUBLISHED_AT DATETIME DEFAULT CURRENT_TIMESTAMP
);

CREATE DATABASE PROFILE_SERVICE;
USE PROFILE_SERVICE

CREATE TABLE USERS (
    ID INT PRIMARY KEY IDENTITY,
    USERNAME NVARCHAR(255),
    EMAIL NVARCHAR(255),
    FIRSTNAME NVARCHAR(255),
    LASTNAME NVARCHAR(255),
    PHONE_NUMBER NVARCHAR(255),
    PROFILE_PICTURE NVARCHAR(255) DEFAULT 'default_profile.png',
    SUBSCRIBERS_COUNT INT DEFAULT 0,
    SUBSCRIPTIONS_COUNT INT DEFAULT 0,
    POSTS_COUNT INT DEFAULT 0
);

CREATE TABLE POSTS (
    ID INT PRIMARY KEY IDENTITY,
    USER_ID INT FOREIGN KEY REFERENCES USERS(ID),
    TITLE NVARCHAR(255) NOT NULL,
    CONTENT NVARCHAR(MAX) NOT NULL,
    IMAGE NVARCHAR(255),
    PUBLISHED_AT DATETIME DEFAULT GETDATE()
);
```

```
CREATE TABLE SUBSCRIPTION (  
    ID INT IDENTITY(1,1) PRIMARY KEY,  
    SUBSCRIBER_ID INT,  
    SUBSCRIBED_TO_ID INT,  
    FOREIGN KEY (SUBSCRIBER_ID) REFERENCES USERS(ID),  
    FOREIGN KEY (SUBSCRIBED_TO_ID) REFERENCES USERS(ID)  
);  
  
CREATE TABLE COMMENTS (  
    ID INT PRIMARY KEY IDENTITY,  
    USER_ID INT,  
    POST_ID INT,  
    USERNAME NVARCHAR(255),  
    USER_PICTURE NVARCHAR(255) DEFAULT 'default_profile.png',  
    COMMENT NVARCHAR(MAX) NOT NULL,  
    CREATED_AT DATETIME DEFAULT GETDATE(),  
    FOREIGN KEY (USER_ID) REFERENCES USERS(ID),  
    FOREIGN KEY (POST_ID) REFERENCES POSTS(ID)  
);  
  
CREATE DATABASE NEWS_SERVICE;  
use NEWS_SERVICE  
  
CREATE TABLE NEWS (  
    ID INT PRIMARY KEY IDENTITY,  
    TITLE TEXT NOT NULL,  
    PARAGRAPH TEXT NOT NULL,  
    ALT TEXT NOT NULL,  
    LINK NVARCHAR(255) NOT NULL,  
    IMAGE NVARCHAR(255) NOT NULL,  
    DATE NVARCHAR(255) NOT NULL,  
    TOPIC_NAME nvarchar(255) not null  
);
```


Приложение Д

```

class Mailer {
  static async sendEmail(type, toMail, subject, content) {
    try {
      const isHtml = type === 'html';
      const response = await sendMail(toMail, subject,
content, isHtml);
      console.log("☞ the message has been sent successfully",
response);
    } catch (error) {
      console.error("Rejected: error sending the message\n" +
error);
    }
  }

  static sendEmailMessage(toMail, username, message) {
    this.sendEmail('text', toMail, 'Email verification', `Hello
${username}, ${message}`);
  }

  static sendSupportEmailMessage(mail, username, message) {
    this.sendEmail('text', "lotus.service.no.reply@gmail.com",
'Support', `| ${username} | ${mail} | ${message}`);
  }

  static sendEmailSystemNotification(toMail, message) {
    this.sendEmail('text', toMail, 'System notification',
message);
  }

  static sendEmailUserRegistered(toMail, username) {
    this.sendEmail('html', toMail, `Congratulations on the
successful registration of ${username}!`, "userRegistered.html");
  }
}

```