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](#_Toc166678810)

[1 Постановка задачи 5](#_Toc166678811)

[1.1 Аналитический обзор аналогов 5](#_Toc166678812)

[1.1.1 Интернет-ресурс «Хабр» 5](#_Toc166678813)

[1.1.2 Интернет-ресурс «Teletype» 6](#_Toc166678814)

[1.1.3 Интернет ресурс «Hashtap» 7](#_Toc166678815)

[1.1.4 Интернет ресурс «Dev.to» 7](#_Toc166678816)

[1.2 Постановка задачи 8](#_Toc166678817)

[1.3 Средства разработки 9](#_Toc166678818)

[2 Проектирование web-приложения 10](#_Toc166678819)

[2.1 Архитектура приложения 10](#_Toc166678820)

[2.2 Проектирование структуры базы данных 11](#_Toc166678821)

[3 Разработка web-приложения 14](#_Toc166678822)

[3.1 Обобщенная структура приложения 14](#_Toc166678823)

[3.2 Разработка серверной части приложения 14](#_Toc166678824)

[3.2.1 Микросервисы 16](#_Toc166678825)

[3.2.2 Сервис аутентификации 17](#_Toc166678826)

[3.2.3 Сервис уведомлений 18](#_Toc166678827)

[3.2.4 Сервис профилей 20](#_Toc166678828)

[3.2.5 Сервис новостей 20](#_Toc166678829)

[3.2.6 API Gateway 21](#_Toc166678830)

[3.3 Разработка клиентской части приложения 22](#_Toc166678831)

[4 Тестирование web-приложения 25](#_Toc166678832)

[4.1 Ошибка при вводе несуществующего маршрута 25](#_Toc166678833)

[4.2 Ошибка при пустом значении полей 25](#_Toc166678834)

[4.3 Ошибка при некорректных данных для входа 26](#_Toc166678835)

[4.4 Ошибка при регистрации существующего логина 26](#_Toc166678836)

[4.5 Иные ошибки 26](#_Toc166678837)

[5 Руководство пользователя 27](#_Toc166678838)

[5.1 Вход в аккаунт 27](#_Toc166678839)

[5.2 Регистрация пользователя 27](#_Toc166678840)

[5.3 Взаимодействие с профилем пользователя 28](#_Toc166678841)

[5.4 Создание поста 29](#_Toc166678842)

[5.5 Редактирование поста 29](#_Toc166678843)

[5.6 Страница «People» 30](#_Toc166678844)

[5.7 Страница новостей 30](#_Toc166678845)

[5.8 Страница «Для вас» 31](#_Toc166678846)

[5.9 Оставление комментариев 32](#_Toc166678847)

[5.10 Страница технической поддержки 32](#_Toc166678848)

[Заключение 34](#_Toc166678849)

[Список используемых источников 35](#_Toc166678850)

[Приложение А 36](#_Toc166678851)

[Приложение Б 37](#_Toc166678852)

[Приложение В 38](#_Toc166678853)

[Приложение Г 39](#_Toc166678854)

[Приложение Д 41](#_Toc166678855)

Введение

В современном мире цифровых технологий и глобальной коммуникации всё больше людей участвуют в интеллектуальных играх, как в формате живого взаимодействия, так и онлайн. Эти игры не только развивают мышление, но и способствуют активному общению, социальному взаимодействию и обмену знаниями. В то же время, платформы с элементами социальной сети предоставляют пользователям возможность обмениваться впечатлениями, обсуждать свои достижения и результаты игр.

В рамках данного курсового проекта будет разработано программное средство, которое объединяет функционал для проведения интеллектуальных игр и элементы социальной сети. Цель — создать удобное и интуитивно понятное веб-приложение, которое позволит пользователям участвовать в интеллектуальных играх, отслеживать свою статистику, делиться результатами и общаться с другими участниками. Для достижения этой цели необходимо выполнить следующие задачи:

* провести анализ предметной области и определить основные сущности и атрибуты системы;
* разработать архитектуру приложения, используя трёхуровневую модель (слой доступа к данным, слой бизнес-логики, слой представления);
* реализовать ключевые модули системы, включая модули аутентификации, управления играми, профилей и статистики;
* разработать API Gateway для маршрутизации запросов к соответствующим микросервисам и обеспечения их взаимодействия;
* создать клиентскую часть приложения на основе React, обеспечив удобный интерфейс для взаимодействия с пользователем;
* наполнить базу данных тестовыми данными, такими как пользователи, игры и результаты;
* разработать функции и методы для реализации бизнес-логики системы, включая системы уведомлений, рейтинг игроков, комментарии и другие элементы социальной сети.

Данный проект направлен на улучшение опыта участников интеллектуальных игр и создание социальной среды, способствующей обмену знаниями и поддержанию интереса к играм.

1 Постановка задачи и обзор аналогичных решений

* 1. Аналитический обзор аналогов

В современном мире платформы для интеллектуальных игр и организации социальных взаимодействий играют важную роль в создании сообществ, обмене знаниями и проведении досуга. Такие платформы позволяют пользователям не только участвовать в играх, но и делиться результатами, обсуждать свои достижения и взаимодействовать с другими участниками, что создает активное и мотивирующее игровое сообщество. В данном разделе будут рассмотрены существующие интернет-ресурсы и веб-приложения, которые предоставляют возможности для организации интеллектуальных игр с элементами социальной сети.

* + 1. Интернет-ресурс «QuizUp»

QuizUp — это популярная платформа для проведения викторин, которая позволяет пользователям соревноваться в различных темах и сравнивать свои результаты с друзьями или глобальными участниками. Приложение привлекает своей простотой и большим выбором тем для игр, охватывающих широкий спектр интересов, от общих знаний до специализированных тем.

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

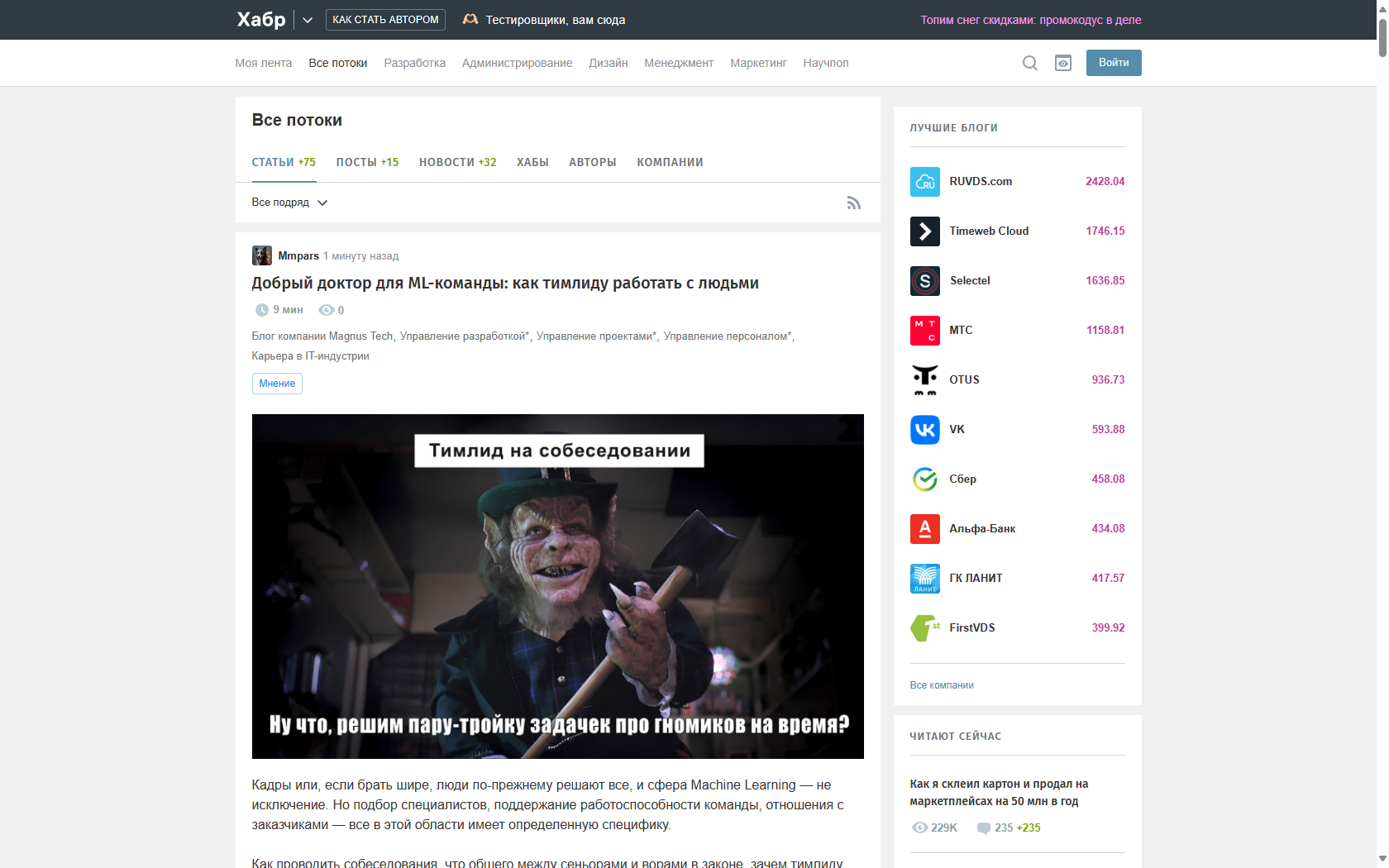


Рисунок 1.1 – Интерфейс интернет-ресурса «QuizUp»

Достоинства:

* Широкий выбор тем: QuizUp предлагает пользователям огромный выбор тем, начиная от кино и музыки и заканчивая наукой и историей.
* Глобальные рейтинги: Пользователи могут сравнивать свои результаты с другими участниками по всему миру, что создает дополнительную мотивацию для улучшения.
* Простой и удобный интерфейс: Интерфейс приложения интуитивно понятен и легко освоить даже новым пользователям.

Недостатки:

* Ограниченные социальные функции: Платформа не предоставляет возможностей для активного взаимодействия между пользователями вне игрового процесса.
* Отсутствие кастомизации игр: Пользователи не могут настраивать викторины под свои предпочтения, что ограничивает возможности для создания уникального контента.
  + 1. Интернет-ресурс «Kahoot!»

Kahoot! — образовательная платформа для создания викторин, которая часто используется в школах и университетах для проведения интерактивных занятий. Она ориентирована на группы участников, что делает её идеальной для обучения и совместных мероприятий.

Интерфейс интернет-ресурса «Kahoot!» представлен на рисунке 1.2.

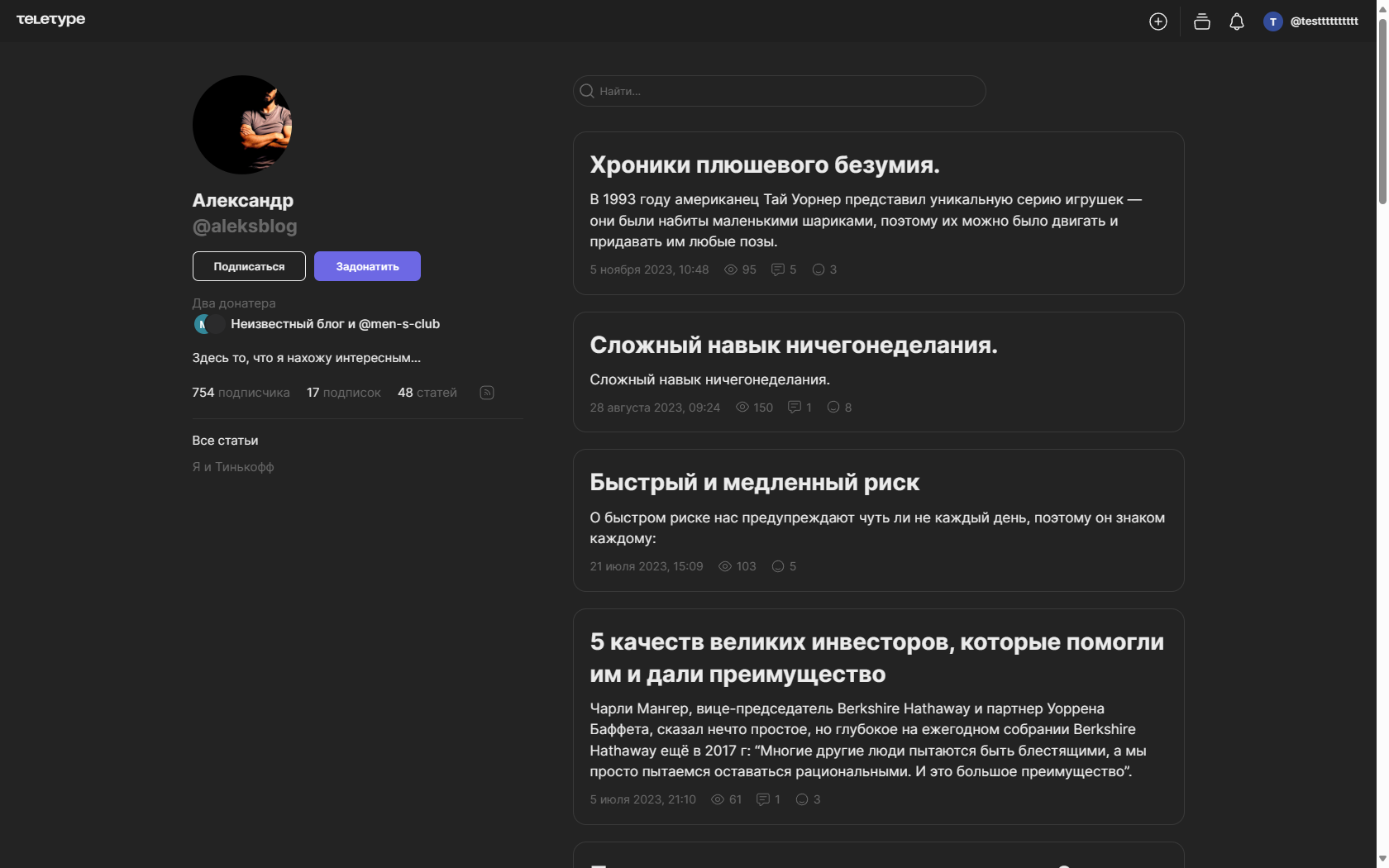


Рисунок 1.2 – Интерфейс интернет-ресурса «Kahoot!»

Достоинства:

* Поддержка живых игр: Kahoot! позволяет проводить викторины в режиме реального времени, что делает процесс участия более динамичным и интерактивным.
* Ориентированность на образование: Платформа широко используется в образовательных учреждениях для тестирования знаний и вовлечения учеников.
* Простота создания игр: Пользователи могут легко создавать свои викторины и адаптировать их под нужды учебного процесса.

Недостатки:

* Ограниченные возможности персонализации: Несмотря на возможность создания собственных викторин, пользователи не могут изменять многие аспекты интерфейса или игровых сценариев.
* Фокус на группе: Платформа больше подходит для групповых мероприятий, чем для индивидуального использования.
  + 1. Интернет ресурс «Jackbox Games»

Jackbox Games — это набор многопользовательских игр, которые можно играть на различных устройствах с участием большого количества людей. Игры отличаются оригинальными сценариями и креативными задачами, ориентированными на взаимодействие с друзьями.

Интерфейс интернет-ресурса «Jackbox Games» представлен на рисунке 1.3.

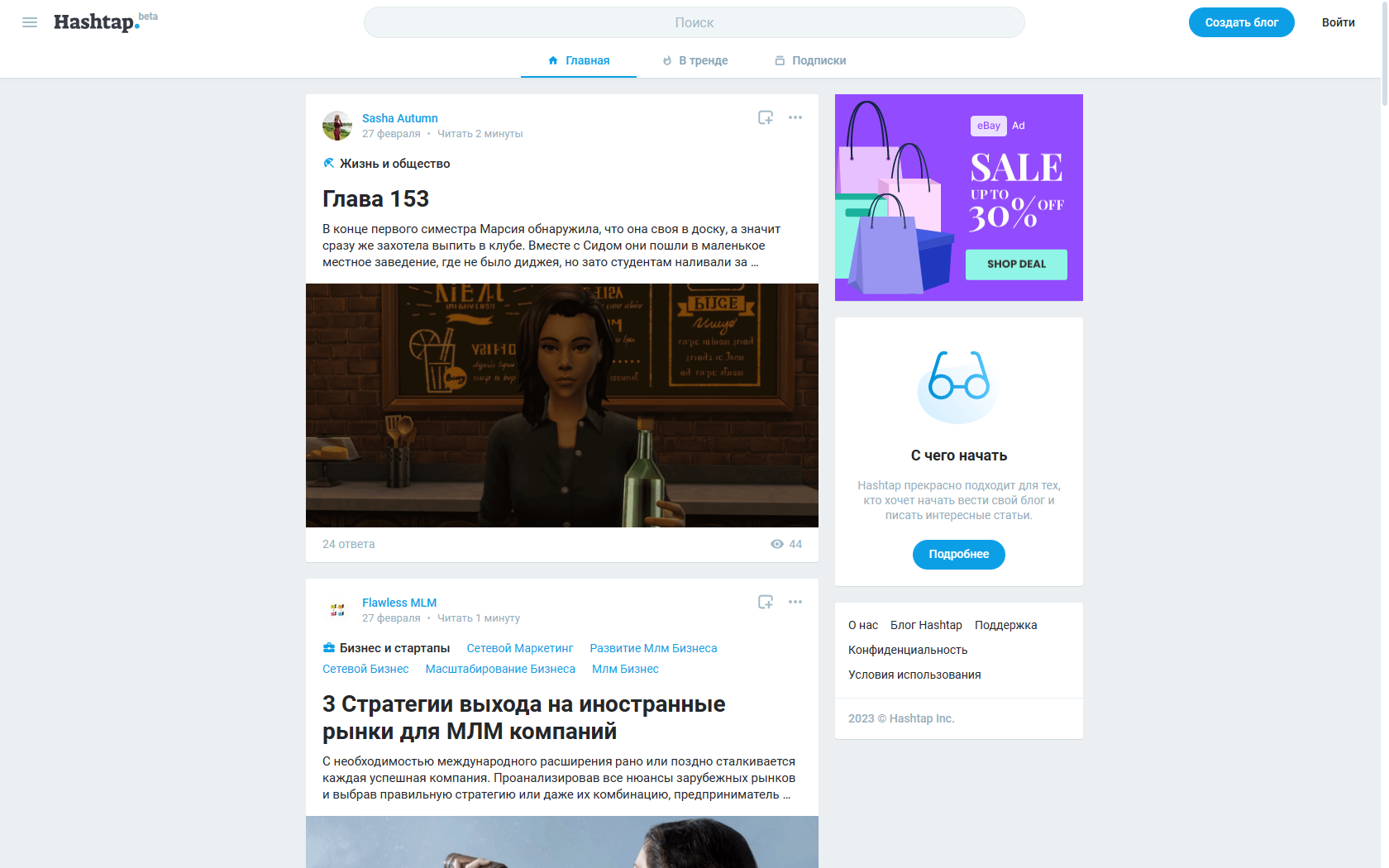


Рисунок 1.3 – Интерфейс интернет-ресурса «Jackbox Games»

Достоинства:

* Веселые и креативные игры: Jackbox Games предлагает оригинальные игровые сценарии, которые делают каждый игровой сеанс уникальным.
* Поддержка большого числа участников: Играть в Jackbox Games можно в больших компаниях, что делает его идеальным для вечеринок и социальных мероприятий.
* Мульти-платформенность: Игры можно запускать на любом устройстве, а участники подключаются через свои смартфоны.

Недостатки:

* Отсутствие социальных функций: Платформа не предоставляет возможностей для взаимодействия пользователей вне игрового процесса.
* Ограниченное количество игр: Несмотря на разнообразие игр в каждом наборе, их количество ограничено, и игроки могут быстро потерять интерес.
  1. Постановка задачи

Анализ аналогов позволяет выявить сильные и слабые стороны существующих платформ для проведения интеллектуальных игр и определить ключевые требования к разработке программного средства в рамках данного дипломного проекта. Программное средство должно выполнять следующие функции:

* обеспечивать возможность регистрации и авторизации участников для участия в играх;
* отправлять уведомления пользователям о предстоящих играх, обновлениях результатов и активности в социальной сети;
* иметь удобный и интуитивно понятный интерфейс для создания и управления играми;
* предоставлять пользователям возможность организовывать интеллектуальные игры с настройкой параметров (темы, количество участников и т.д.);
* обеспечивать функцию участия пользователей в играх и отображения статистики в режиме реального времени;
* предоставлять возможность пользователям комментировать игры и общаться в социальной сети;
* поддерживать персонализацию профилей пользователей, включая настройку аватаров и информации о себе;
* предоставлять пользователю возможность редактировать и удалять свои данные и комментарии;
* обеспечивать персонализированный контент и рекомендации для пользователей на основе их интересов и активности;
* гарантировать защиту всех пользовательских данных, соблюдая современные стандарты безопасности.

Диаграмма вариантов использования представлена в приложении А.

* 1. Средства разработки

Для разработки программного средства, предназначенного для сбора статистики и организации интеллектуальных игр с элементами социальной сети, будут использованы следующие инструменты и технологии:

* Интегрированная среда разработки JetBrains Rider – для создания и отладки кода на языке C#;
* Язык программирования C# – для разработки серверной части приложения;
* ASP.NET Core Web API – для создания REST-сервиса, обеспечивающего взаимодействие с клиентской частью и другими компонентами системы;
* React (TypeScript) – для разработки клиентской части приложения с использованием современного фреймворка для создания интерактивных пользовательских интерфейсов;
* Redis – для кеширования данных и увеличения производительности системы;
* nginx – для маршрутизации и управления трафиком, а также API Gateway;
* MS SQL Server – для хранения информации о пользователях, играх, статистике и других данных;
* Shadcn UI – библиотека компонентов для создания удобных и современных пользовательских интерфейсов.

Использование данных технологий предоставляет несколько важных преимуществ при разработке программного средства:

* JetBrains Rider обеспечивает удобную и мощную среду для разработки на C#, предоставляя инструменты для отладки, анализа кода и повышения производительности разработки.
* C# и ASP.NET Core Web API обеспечивают высокую производительность и расширяемость серверной части, что позволяет эффективно обрабатывать запросы пользователей и обеспечивать высокую нагрузку.
* React с TypeScript помогает создавать динамичные и отзывчивые пользовательские интерфейсы, позволяя пользователям взаимодействовать с системой в реальном времени.
* Redis обеспечивает быструю обработку данных и минимизирует задержки при частом доступе к информации, что особенно важно для системы с высокой нагрузкой, связанной с интеллектуальными играми.
* nginx и API Gateway способствуют масштабируемости системы и обеспечивают оптимальную маршрутизацию запросов к микросервисам.
* MS SQL Server используется как надежное и производительное хранилище данных, обеспечивая целостность информации и быструю работу с большими объемами данных.

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 современных условиях разработки информационных систем, где требования могут часто изменяться, важно, чтобы приложения были гибкими и масштабируемыми. Для программного средства, предназначенного для сбора статистики и организации интеллектуальных игр с элементами социальной сети, было решено использовать трехуровневую архитектуру с применением микросервисного подхода. Это решение обеспечивает несколько ключевых преимуществ, таких как независимость компонентов, возможность их автономного масштабирования, упрощение разработки и внедрения новых функций, а также повышение устойчивости системы к ошибкам.

Микросервисная архитектура позволяет каждому компоненту системы функционировать отдельно, что дает возможность параллельной разработки и тестирования отдельных функциональных модулей. Например, функциональные модули, такие как создание игры, сбор статистики и управление профилями пользователей, могут разрабатываться и обновляться независимо, не затрагивая работу других модулей. Это упрощает процесс интеграции и сокращает время на внесение изменений в систему.

Взаимодействие между сервисами будет осуществляться с использованием Redis для кеширования данных и RabbitMQ для обмена сообщениями. Эти технологии позволяют обеспечивать надежную и масштабируемую передачу данных между компонентами системы, а также поддерживать высокую производительность при работе с большими объемами данных.

Клиентская часть будет разработана на основе React с использованием TypeScript и библиотеки Shadcn UI для создания интуитивно понятного и современного пользовательского интерфейса. React позволяет эффективно управлять изменениями в интерфейсе благодаря использованию Virtual DOM, что улучшает производительность и обеспечивает более плавный пользовательский опыт.

Серверная часть будет построена на платформе ASP.NET Core Web API, что обеспечит гибкое и масштабируемое взаимодействие с клиентской частью через REST API. Сервер будет принимать HTTP-запросы, обрабатывать их и отправлять ответы в формате JSON, что гарантирует надежную связь между клиентом и сервером.

Одним из ключевых компонентов системы является API Gateway, который будет играть роль центральной точки для всех клиентских запросов. API Gateway будет управлять маршрутизацией запросов, обеспечивать безопасность, проводить аутентификацию и авторизацию пользователей, а также осуществлять мониторинг и управление трафиком. Это значительно упрощает взаимодействие клиентской части с микросервисами, скрывая от клиента сложную внутреннюю логику и обеспечивая единый интерфейс доступа к сервисам.

Программное средство также будет поддерживать WebSockets для реализации реального времени взаимодействия между пользователями и системой. Это обеспечит динамичное обновление информации (например, во время игр или при изменении статистики) без необходимости ручного обновления страницы.

Таким образом, программное средство будет состоять из нескольких взаимосвязанных проектов (слоев), каждый из которых будет выполнять свои функции, взаимодействуя друг с другом через инфраструктурный слой. Это обеспечит легкость в изменении или обновлении любого модуля системы без необходимости изменения других компонентов. Такая архитектура делает систему гибкой, масштабируемой и легко поддерживаемой в будущем.

Диаграмма развертывания программного средства представлена в приложении Б.

2.2 Проектирование структуры базы данных

Раздел

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.

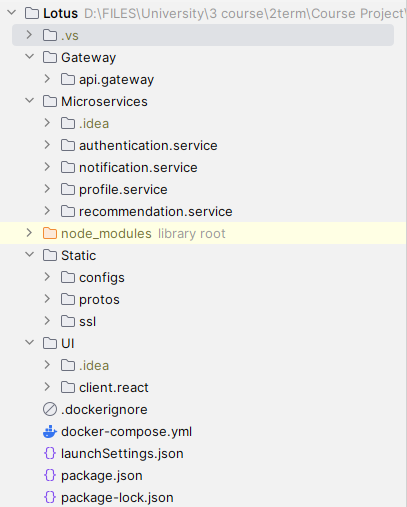


Рисунок 3.1 – Структура приложения

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 например 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 Koa.

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.

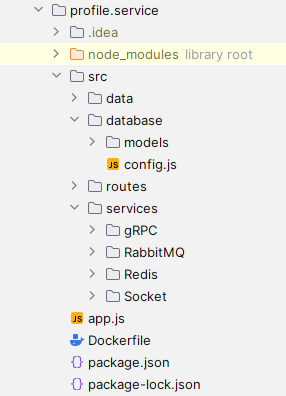


Рисунок 3.2 – Структура серверного приложения на примере сервиса профилей

Входной точкой каждого сервера является файл app.js. В нем происходит инициализация Koa-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 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 реализация 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 обрабатываю 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 отвевает сервис уведомлений. Отправка письма с кодом подтверждение 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 хранение и получение 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 Koa, 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.

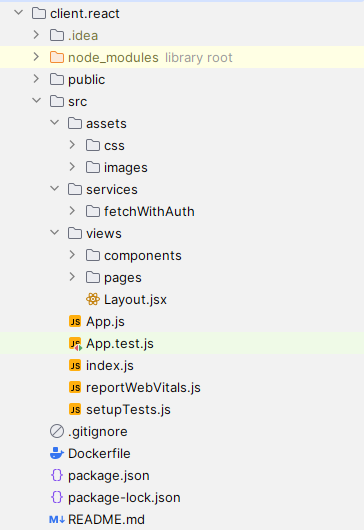


Рисунок 3.3 – Структура клиентского приложения

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 приложение, 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 websoket-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.

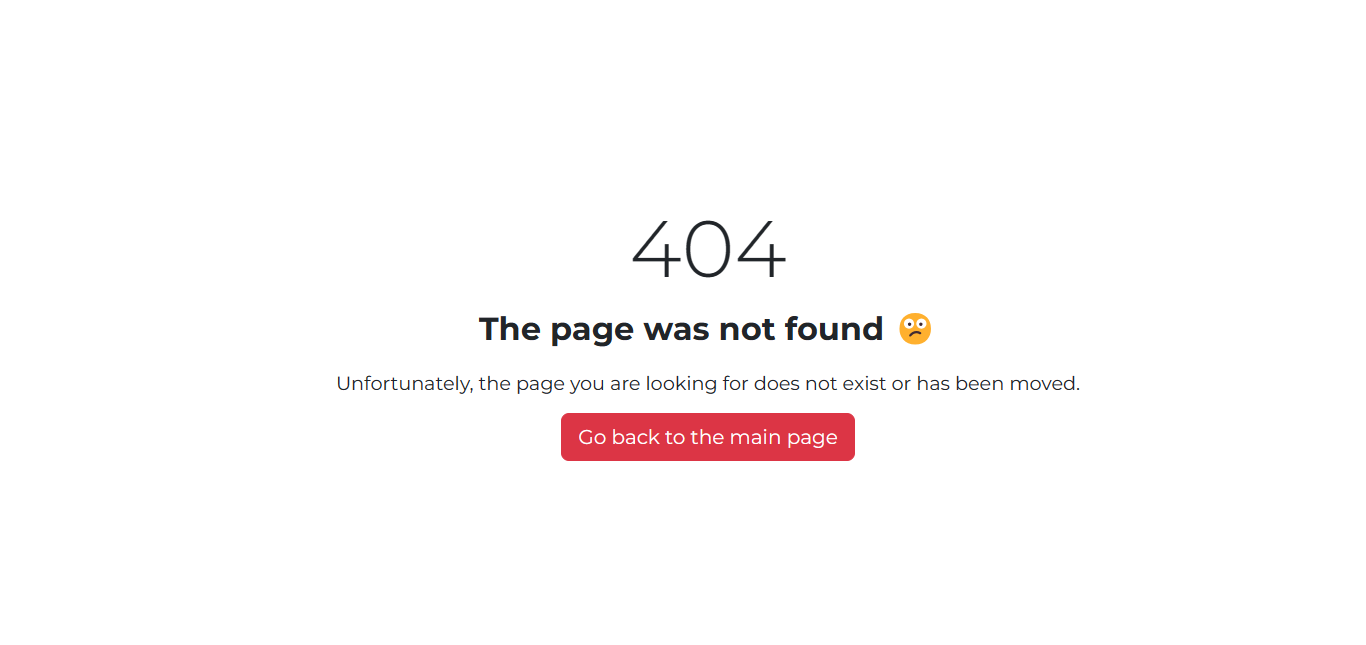


Рисунок 4.1 – Ошибка при вводе несуществующего маршрута

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.

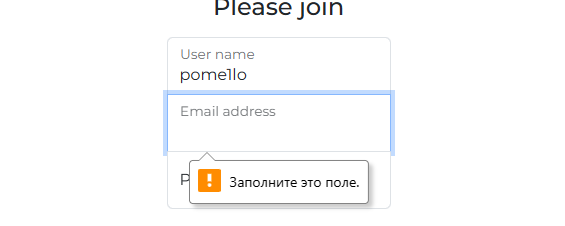


Рисунок 4.2 – Ошибка при пустом значении поля

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.

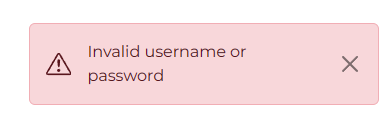


Рисунок 4.3 – Ошибка при некорректных данных для входа

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.

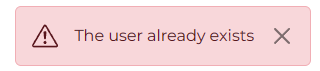


Рисунок 4.4 – Ошибка при регистрации существующего email

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.

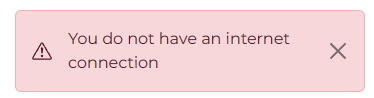


Рисунок 4.5 – Ошибка при регистрации существующего email

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.

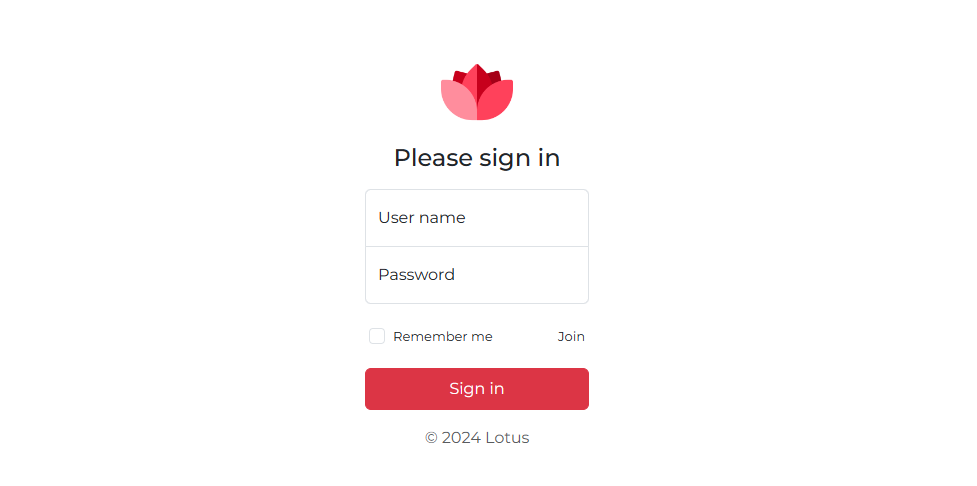


Рисунок 5.1 – Страница входа

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.

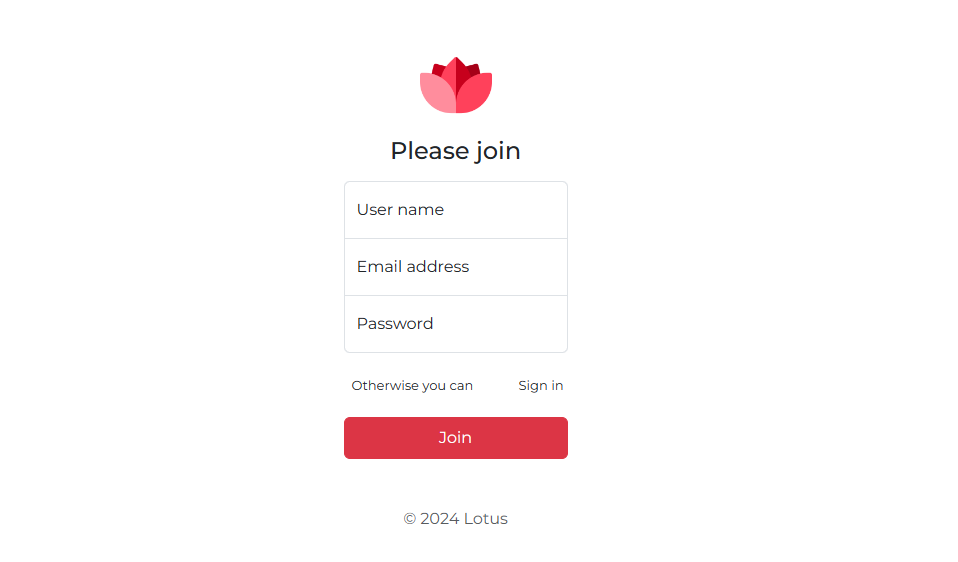


Рисунок 5.2 – Страница регистрации

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.

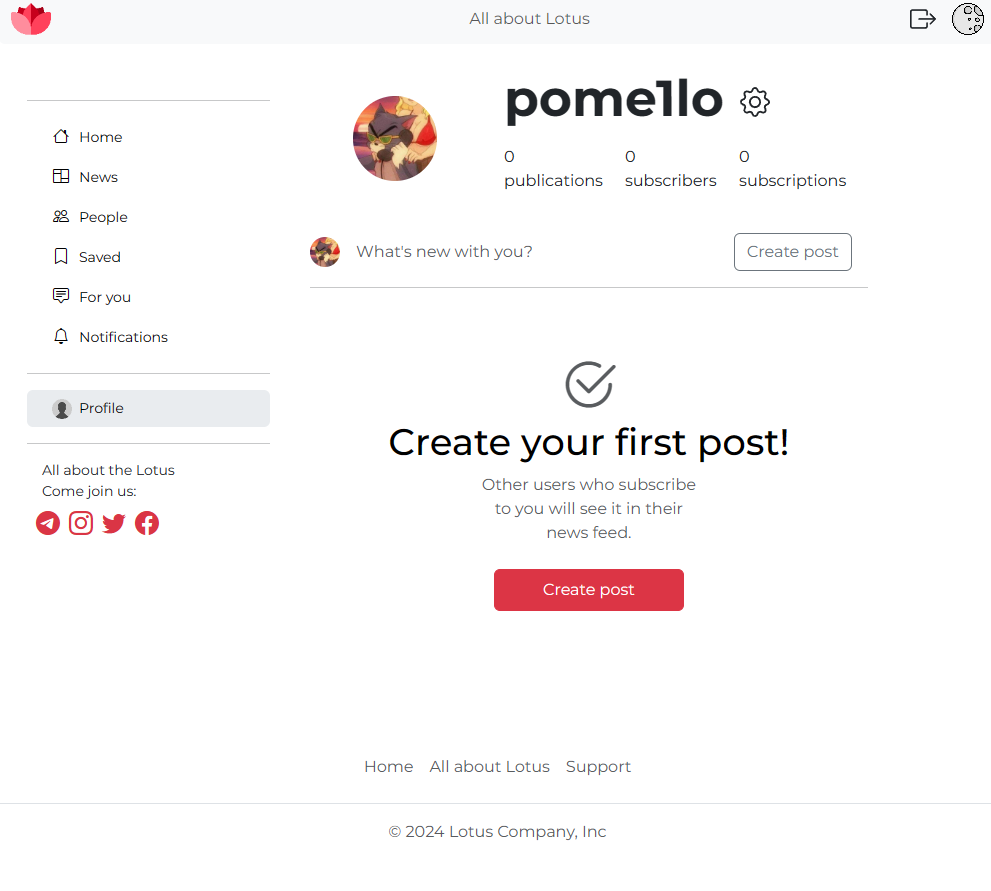


Рисунок 5.3 – Страница профиля пользователя

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.

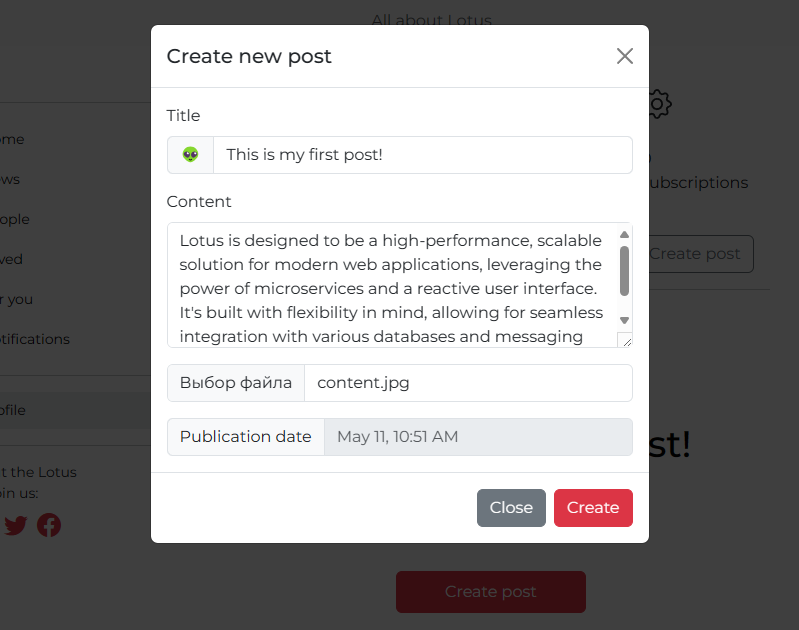


Рисунок 5.4 – Модальное окно для создания поста

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.

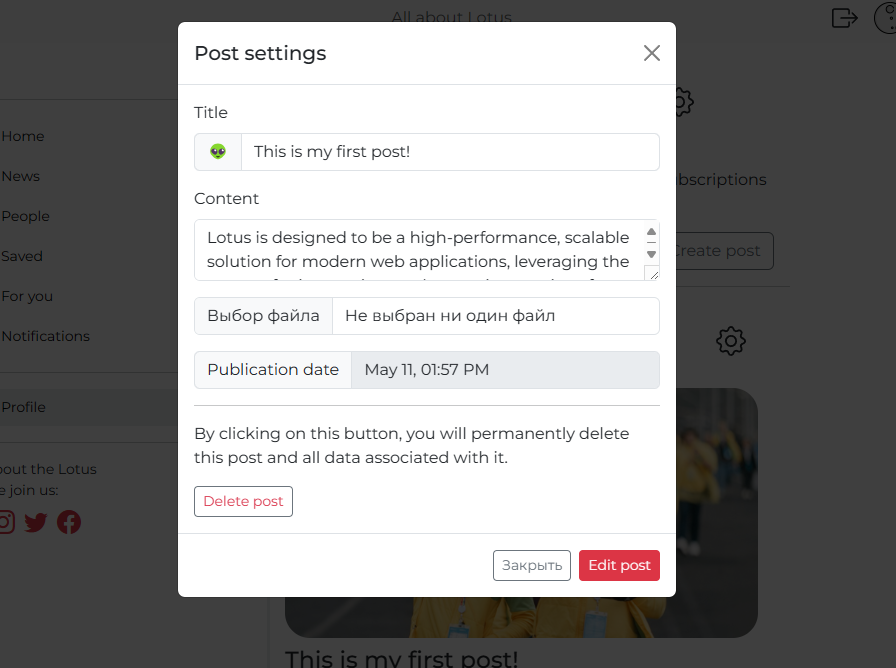


Рисунок 5.5 – Модальное окно для редактирования поста

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.

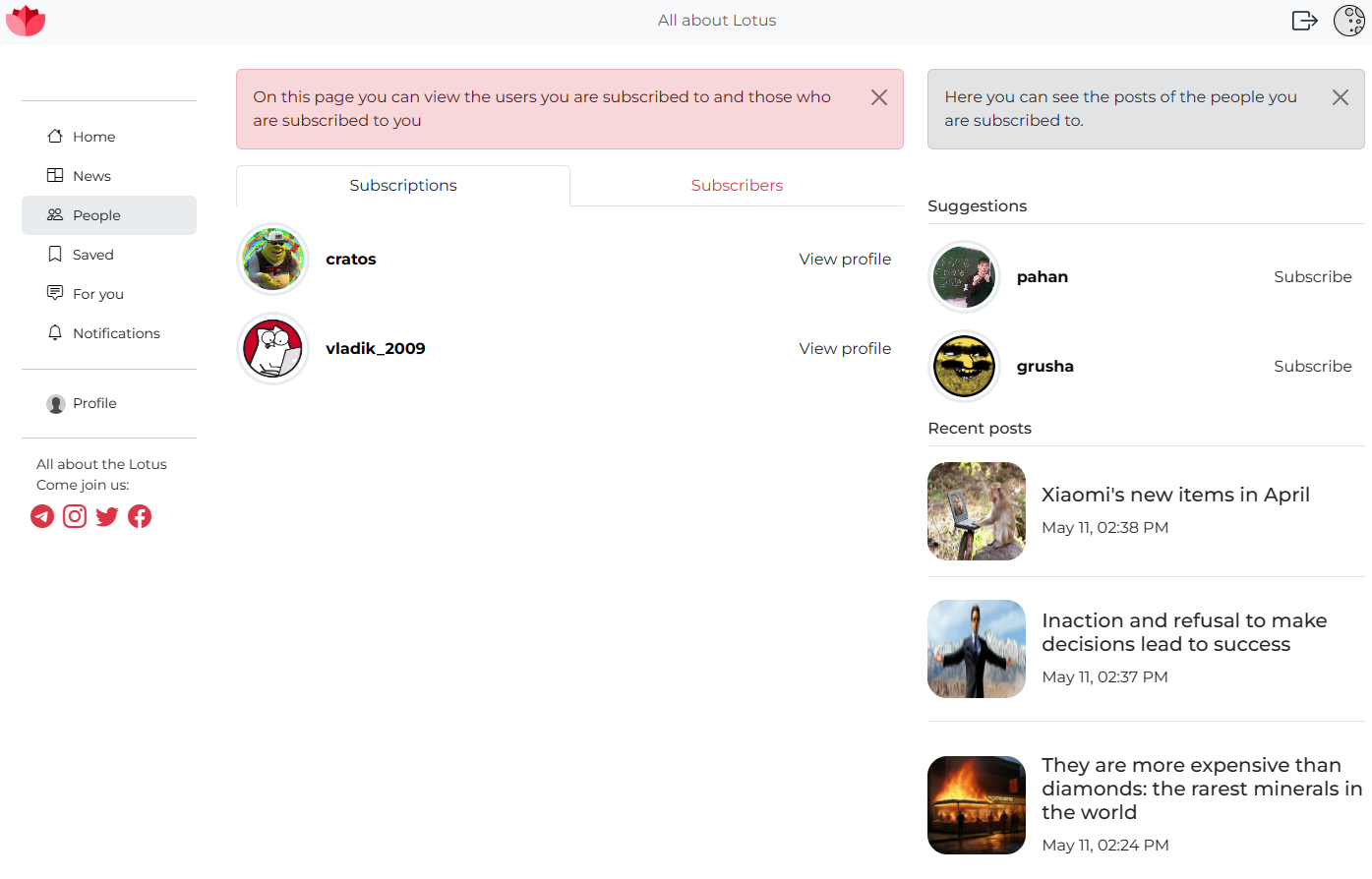


Рисунок 5.6 – Страница подписок и подписчиков

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.

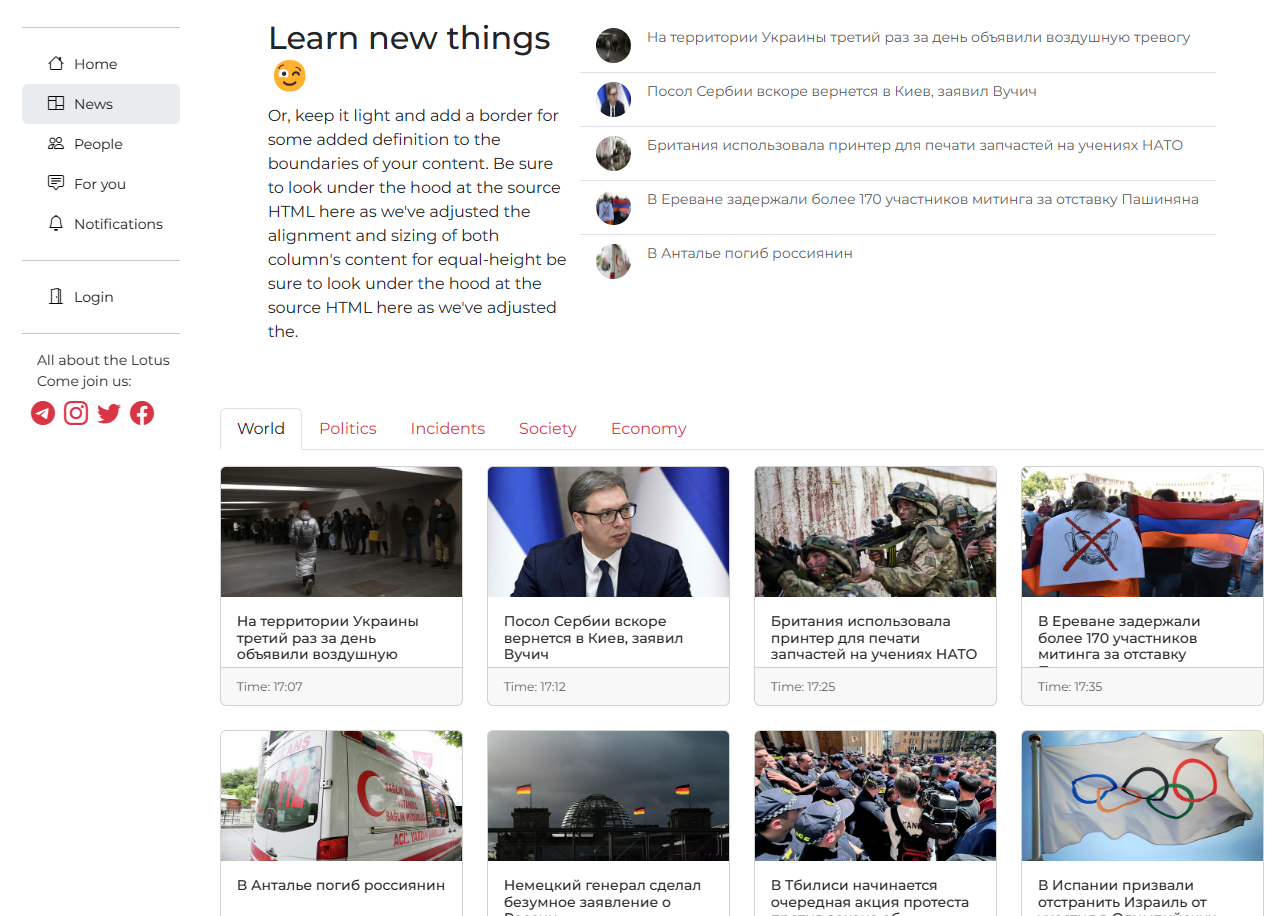


Рисунок 5.7 – Страница новостей

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.

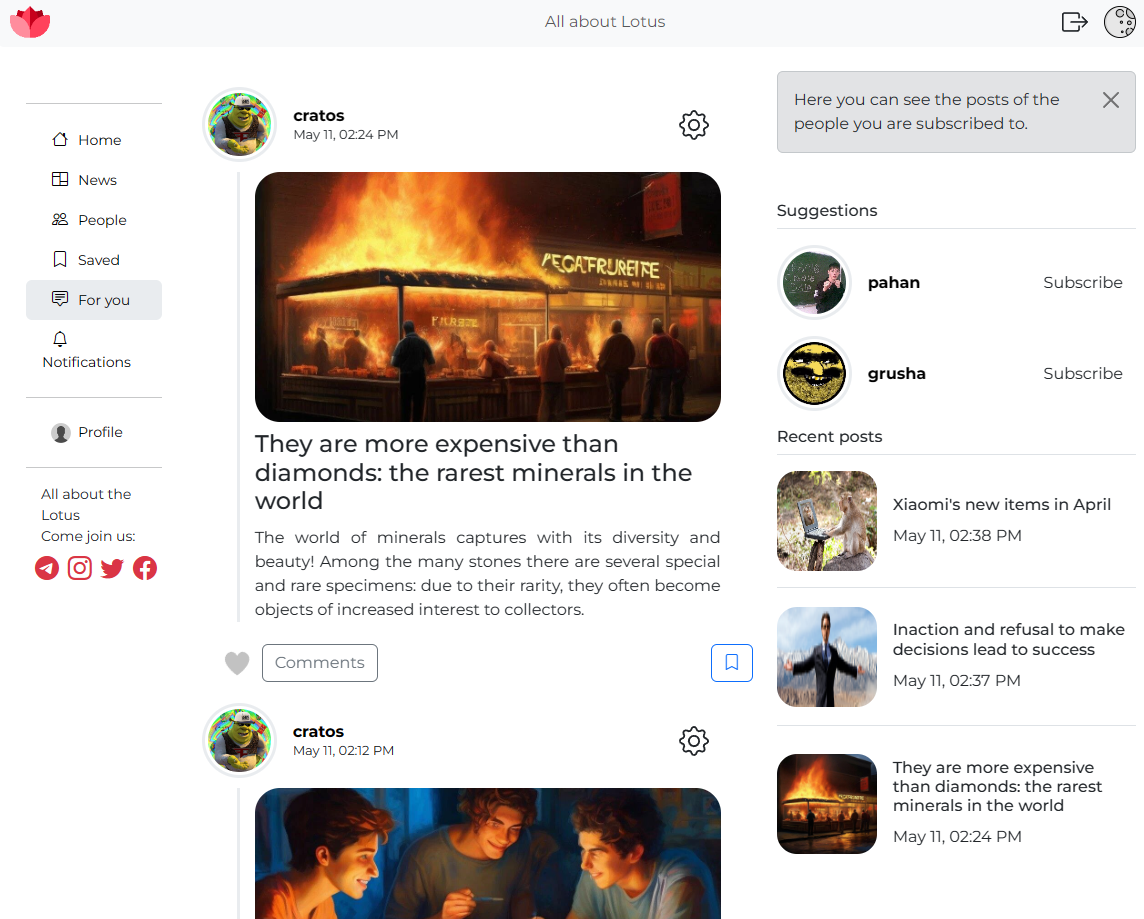


Рисунок 5.8 – Страница «Для вас»

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.

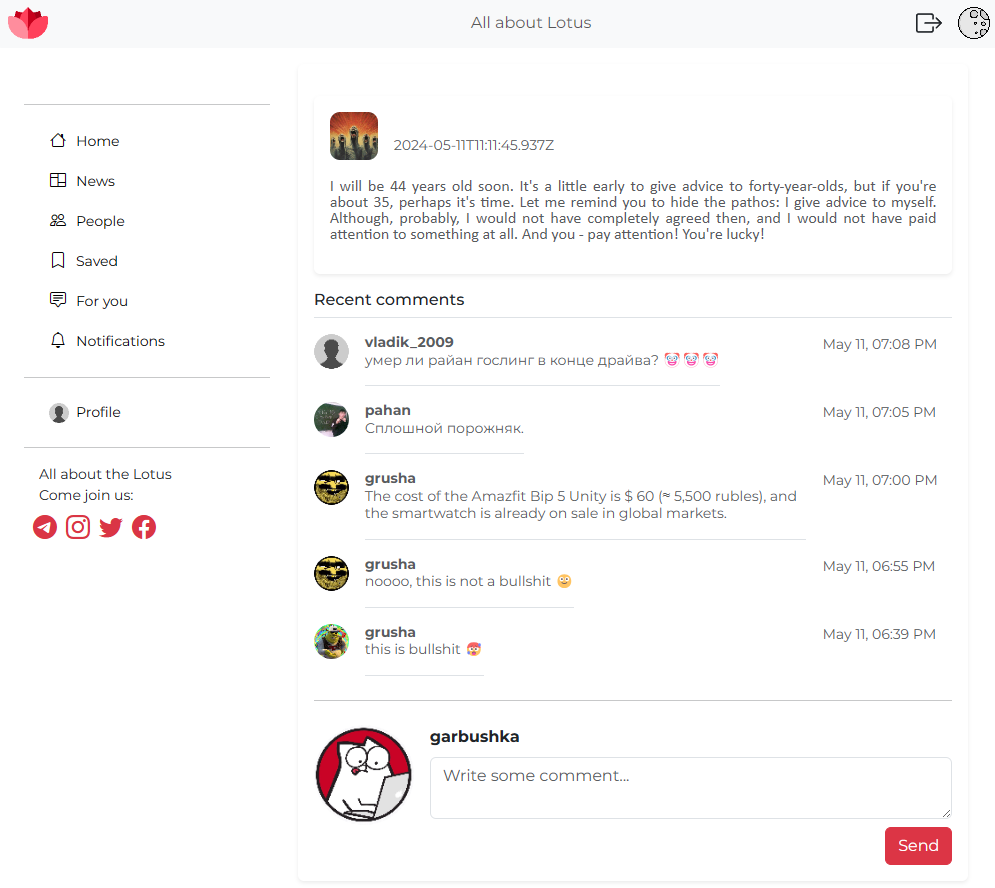


Рисунок 5.9 – Страница комментариев под одним из постов

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.

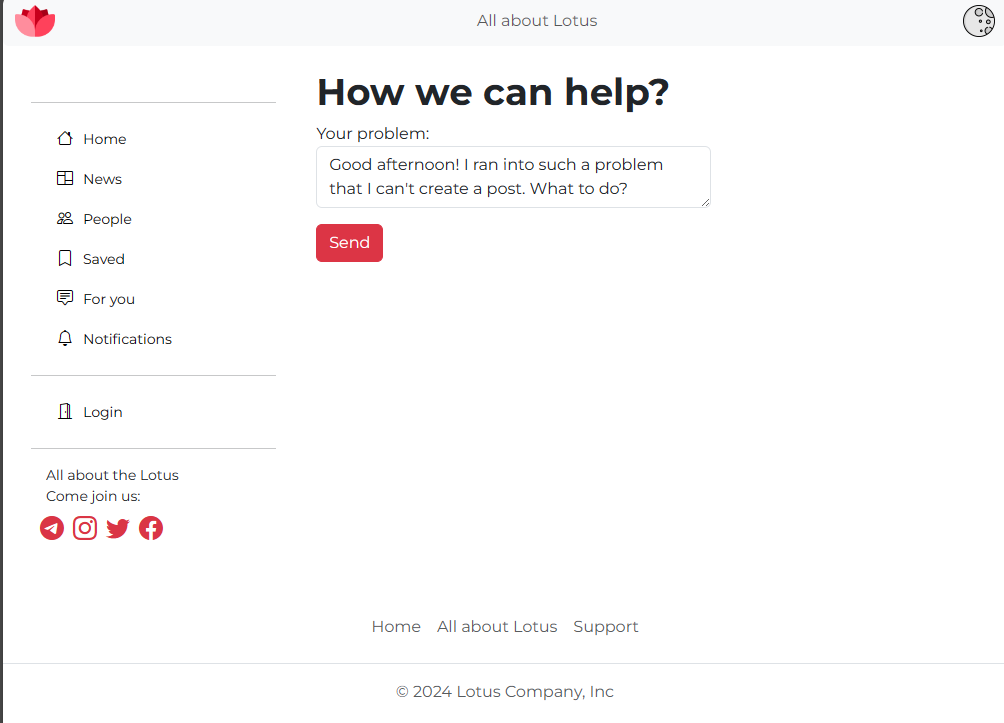


Рисунок 5.10 – Страница технической поддержки

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

Заключение

В ходе выполнения дипломного проекта было разработано программное средство «Программное средство сбора статистики и организации проведения интеллектуальных игр с элементами социальной сети». Проект успешно реализует поставленные задачи, предоставляя пользователям удобные инструменты для участия в интеллектуальных играх, сбора и анализа статистики, а также взаимодействия с другими пользователями в рамках игровой платформы.

В процессе разработки были реализованы ключевые функциональные модули, такие как:

* регистрация и авторизация пользователей;
* управление профилем пользователя и его настройками;
* создание и участие в интеллектуальных играх;
* сбор и отображение статистики по результатам игр;
* система комментариев и обсуждений;
* интеграция элементов социальной сети для взаимодействия между игроками.

Для обеспечения безопасности системы была реализована стратегия авторизации и аутентификации, а также внедрен механизм защиты данных с использованием SSL-сертификатов. Важным компонентом системы стала возможность масштабирования с применением микросервисной архитектуры, что позволяет системе обрабатывать большое количество пользователей и игр одновременно.

Важным этапом разработки стало тестирование программного продукта, которое позволило выявить и устранить ошибки, повысить производительность системы и убедиться, что все требования к проекту выполнены. Тестирование подтвердило, что система работает стабильно и предоставляет корректные данные в ходе взаимодействия с пользователями и сбора статистики.

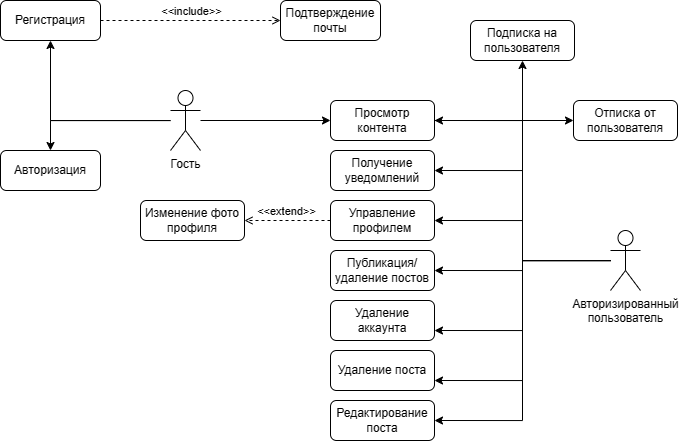
Использование современных технологий, таких как ASP.NET Core Web API, Redis, API Gateway и клиентская часть на React, позволило создать гибкую, надежную и удобную платформу. Программное средство демонстрирует эффективное применение этих технологий и подходов, что обеспечивает возможность его дальнейшего расширения и добавления новых функциональных возможностей.

Таким образом, цель проекта была достигнута. Созданное программное средство предоставляет пользователям функциональный и безопасный инструмент для организации интеллектуальных игр, обмена мнениями и анализа результатов. Это делает его ценным решением для всех участников интеллектуальных игр и открывает возможности для дальнейшего развития и улучшения системы.

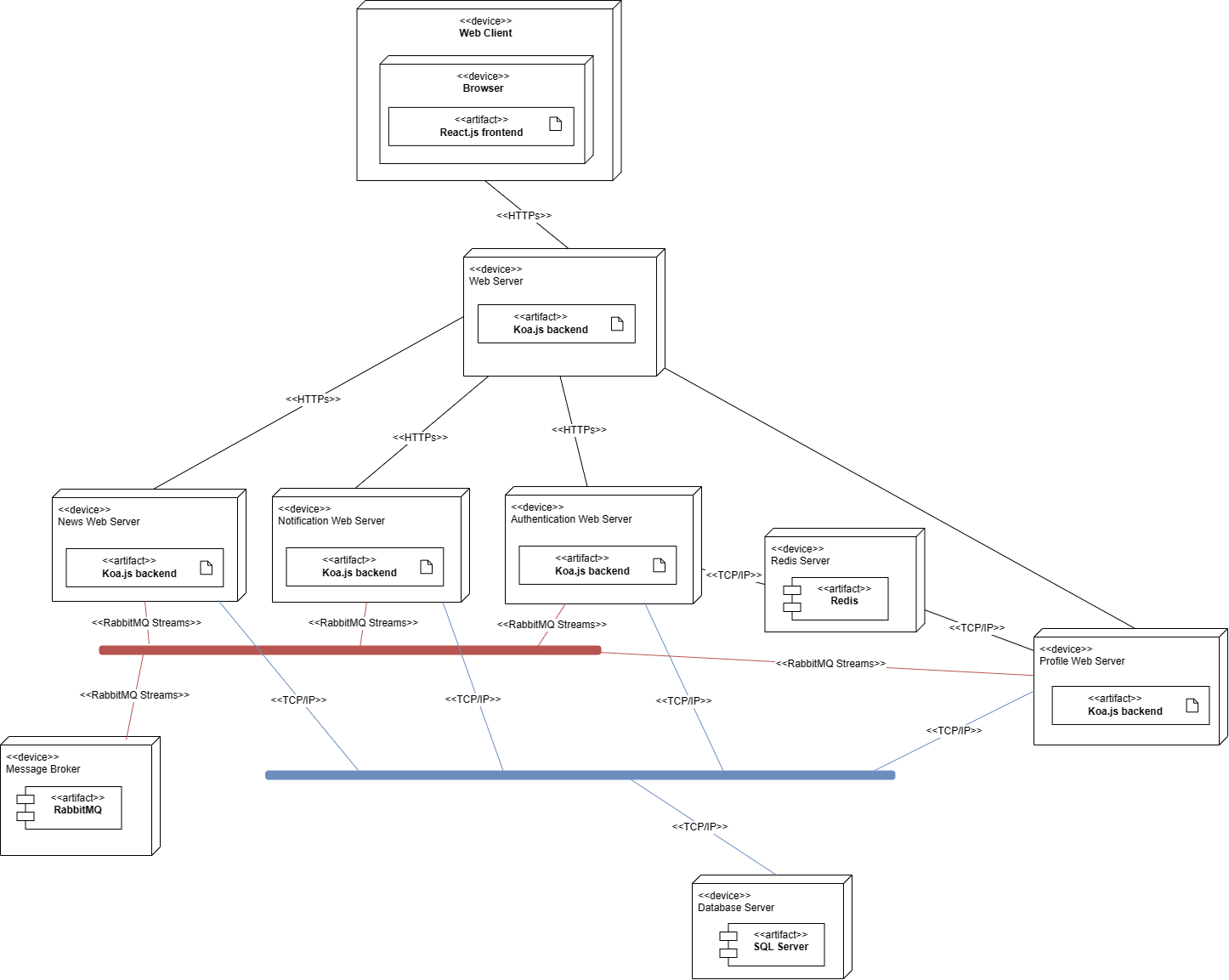
Список используемых источников

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.

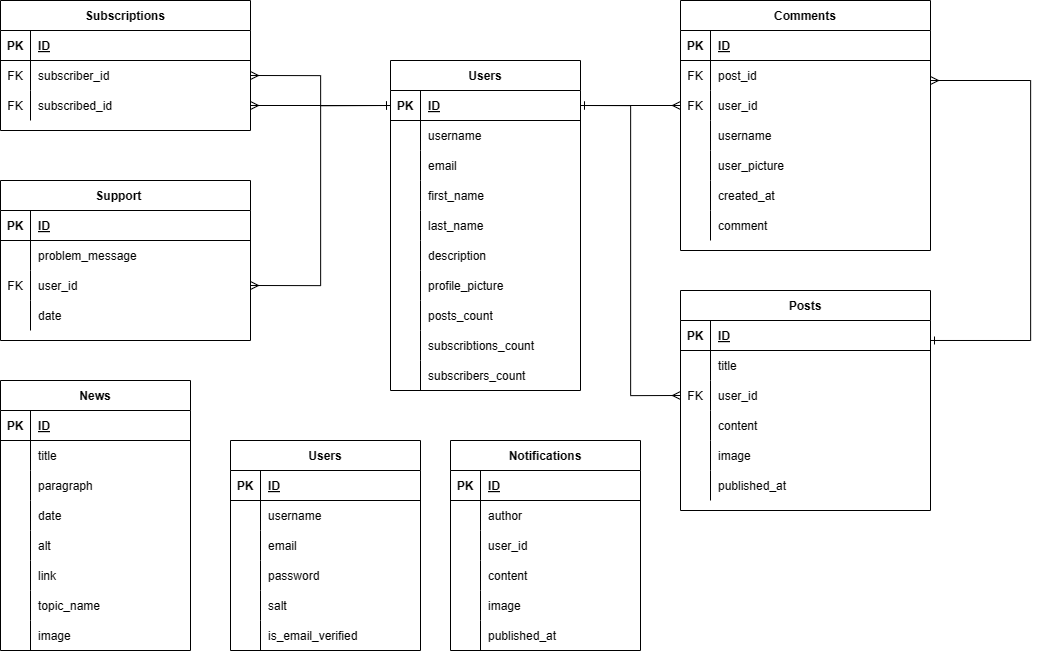
Приложение А



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");  }  } |