**Оглавление**

[Введение 4](#_Toc161677744)

[1 Постановка задачи 5](#_Toc161677745)

[1.1 Аналитический обзор аналогов 5](#_Toc161677746)

[1.2 Средства разработки 8](#_Toc161677747)

[1.3 Разработка функциональных требований 9](#_Toc161677748)

[2 Проектирование архитектуры приложения 11](#_Toc161677749)

[2.1 Обобщенная архитектура 11](#_Toc161677750)

[2.2 Диаграмма UML и блок-схема алгоритма 12](#_Toc161677751)

[2.3 Модель данных 15](#_Toc161677752)

[2.4 Таблицы базы данных 15](#_Toc161677753)

[2.5 Таблицы базы данных 20](#_Toc161677754)

[3 Разработка web-приложения 20](#_Toc161677755)

[3.1 Микросервисы 20](#_Toc161677756)

[3.2 Создание пользователе й, ролей, и таблиц базы данных 20](#_Toc161677757)

[3.3 Пользовательские процедуры 20](#_Toc161677758)

[3.4 Индексы базы данных 20](#_Toc161677759)

[3.5 Последовательности 20](#_Toc161677760)

[3.6 Триггеры 20](#_Toc161677761)

[3.7 Директории 20](#_Toc161677762)

[4 Описание процедур импорта и экспорта 20](#_Toc161677763)

[4.1 Экспорт в JSON 20](#_Toc161677764)

[4.2 Импорт из JSON 20](#_Toc161677765)

[5 Тестирование производительности 20](#_Toc161677766)

[6 Описание технологии и ее применения в базе данных 20](#_Toc161677767)

[6.1 Шифрование данных в Oracle 20](#_Toc161677768)

[6.2 Аутентификация 20](#_Toc161677769)

[6.3 Аудит SYS операций 20](#_Toc161677770)

[6.4 Защита слушателя 20](#_Toc161677771)

[7 Краткое описание приложения для демонстрации 20](#_Toc161677772)

[8 Руководство пользователя 20](#_Toc161677773)

[8.1 Пример работы приложения от лица пользователя 20](#_Toc161677774)

[8.2 Пример работы приложения от лица администратора 20](#_Toc161677775)

[Заключение 20](#_Toc161677776)

[Список используемых источников 20](#_Toc161677777)

[Приложение А 20](#_Toc161677778)

[Приложение Б 20](#_Toc161677779)

[Приложение В 20](#_Toc161677780)

[Приложение Г 20](#_Toc161677781)

[Приложение Д 20](#_Toc161677782)

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 сервисы авторизации и регистрации, уведомлений, контента, комментариев, профилей, рекомендаций и безопасности;
* разработать API Gateway для маршрутизации запросов к соответствующим микросервисам;
* создать клиентскую часть приложения на React для взаимодействия с пользователем;
* использовать Docker Compose для управления сервисами и обеспечения их совместной работы;
* наполнить базу данных тестовыми данными;
* разработать функции и методы для реализации бизнес-логики каждого микросервиса;

разработать запросы для получения необходимой информации из базы данных.

1. Постановка задачи
   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. Интерфейс веб-приложения представлен на рисунке 1.1.

[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](https://habr.com/ru/feed/).

Достоинства:

* [широкий спектр тем: Хабр покрывает множество тем, связанных с IT, включая программирование, разработку веб-сайтов, информационную безопасность, аналитику данных и многое другое](https://habr.com/ru/feed/);
* [сообщество экспертов: Хабр имеет активное сообщество пользователей, которые являются экспертами в своих областях](https://habr.com/ru/feed/);
* [персонализированный контент: Пользователи могут настроить свою ленту новостей, чтобы видеть контент, который их интересует](https://habr.com/ru/feed/).

Недостатки:

* [сложность навигации: Для новых пользователей может быть сложно найти нужную информацию из-за большого количества контента](https://habr.com/ru/feed/);
* [неравномерное качество контента: Качество публикаций может сильно варьироваться, поскольку они создаются разными пользователями](https://habr.com/ru/feed/).

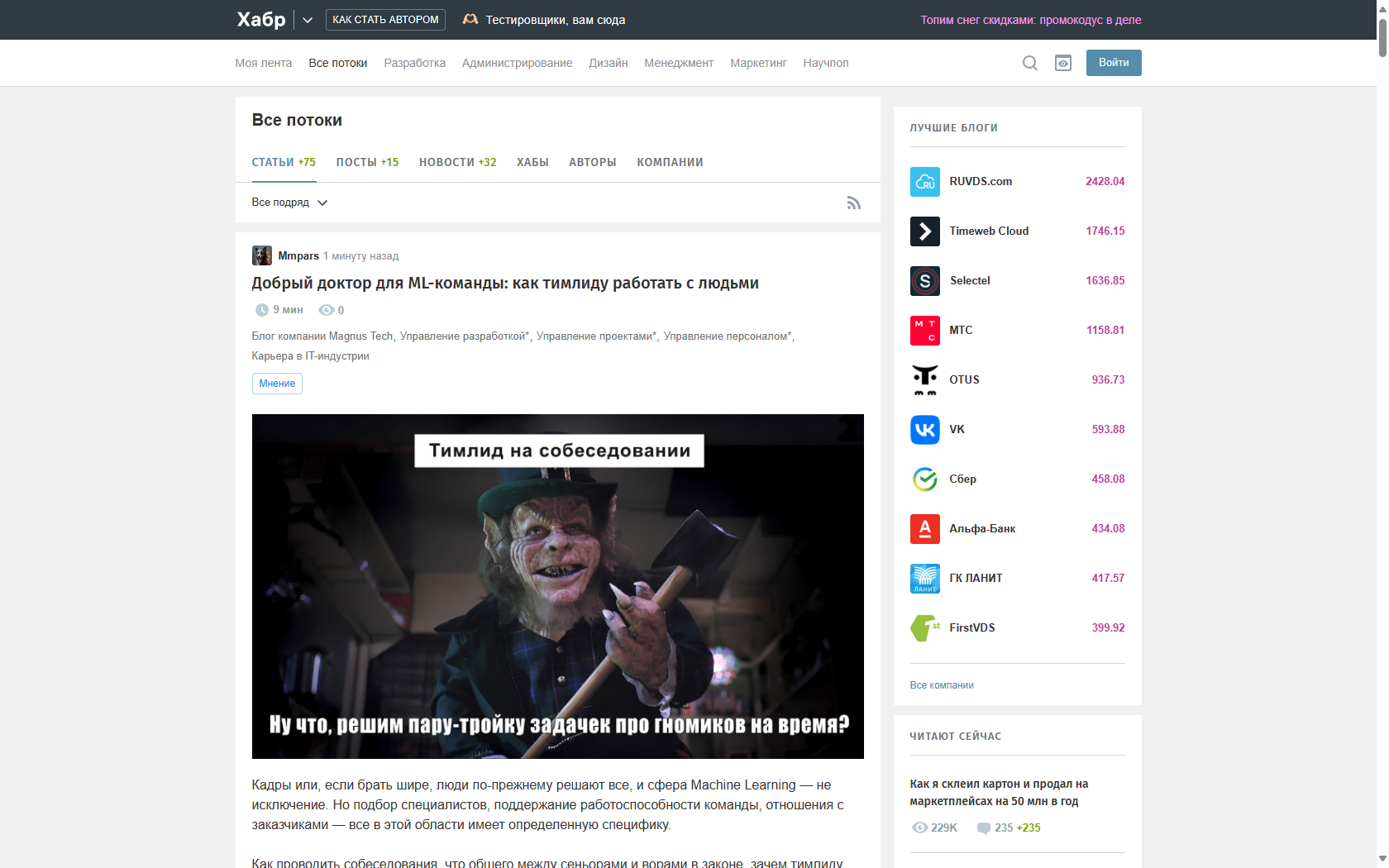


Рисунок 1.1 – Интерфейс [веб-приложения](http://unitoria.ru/blog/kadry/top-5-programm-dlya-kadrovogo-ucheta-v-2021-godu) Хабр

[Teletype — это удобная платформа для публикации статей, которая тесно интегрирована с Telegram и позволяет легко делиться контентом через телеграм-каналы](https://teletype.in/@davdeyonok/howtoteletype). [Пользователи могут создавать статьи с картинками, ссылками и другими элементами, делая информацию доступной и привлекательной](https://teletype.in/@davdeyonok/howtoteletype). Интерфейс веб-приложения представлен на рисунке 1.2.

Достоинства:

* [простота использования: Teletype предлагает интуитивно понятный интерфейс с простым редактором для создания статей](https://teletype.in/@alfiia/VVADFxVyE);
* [визуальное оформление: Платформа предоставляет разнообразные инструменты для оформления текста, включая заголовки, курсив, жирный шрифт и вставку изображений;](https://teletype.in/@davdeyonok/howtoteletype)
* [интеграция с Telegram: Teletype удобен для пользователей Telegram благодаря возможности быстрого перехода к статьям прямо из сообщений и каналов](https://teletype.in/@davdeyonok/howtoteletype).

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.

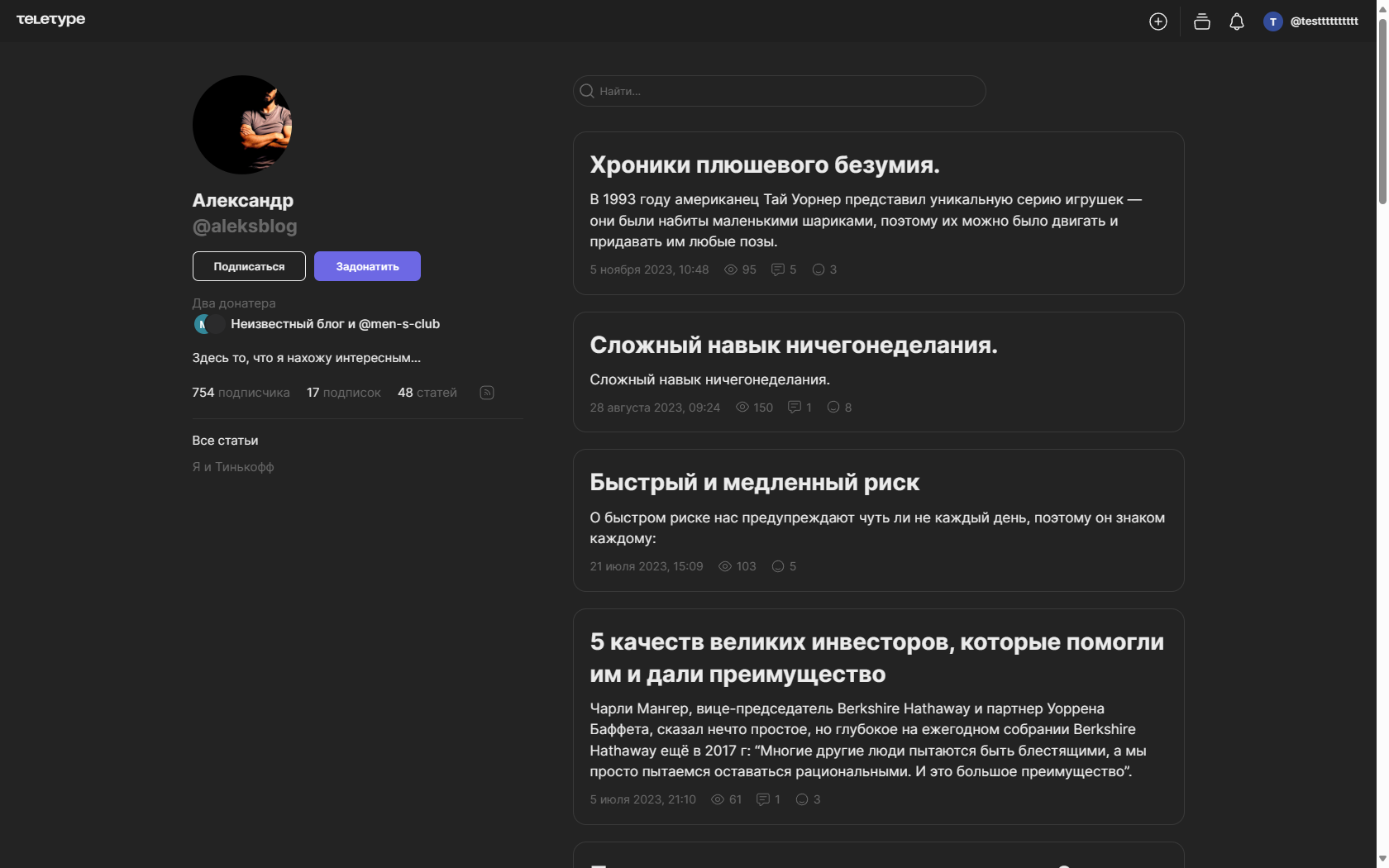


Рисунок 1.2 – Интерфейс [веб-приложения](http://unitoria.ru/blog/kadry/top-5-programm-dlya-kadrovogo-ucheta-v-2021-godu) Teletype

Сайт [Hashtap](https://www.hashtap.com/@support" \t "_blank)[представляет собой платформу, которая предлагает сервис персонализированных рекомендаций](https://www.hashtap.com/@support). [Он также предлагает платформу email-маркетинга, которая автоматически отправляет персонализированные и триггерные email-письма](https://www.hashtap.com/@support). Интерфейс веб-приложения представлен на рисунке 1.3.

Достоинства:

* [персонализированные рекомендации: Hashtap предлагает персонализированные рекомендации, что может помочь пользователям обнаружить новый контент](https://www.hashtap.com/@support);
* [email-маркетинг: Hashtap предлагает платформу email-маркетинга, которая автоматически отправляет персонализированные и триггерные email-письма](https://www.hashtap.com/@support);
* [поддержка: Hashtap имеет команду поддержки, которая может помочь пользователям с вопросами и предложениями](https://www.hashtap.com/@support).

Недостатки:

* [недостаток информации: На основе доступной информации, сложно определить недостатки Hashtap](https://www.hashtap.com/@support). Для более точной оценки могут потребоваться дополнительные данные, такие как отзывы пользователей или личный опыт использования платформы;
* [отсутствие активности на GitHub: Согласно информации на GitHub, Hashtap не имеет значительной активности в последнее время](https://www.hashtap.com/@support). Это может указывать на отсутствие обновлений или улучшений.

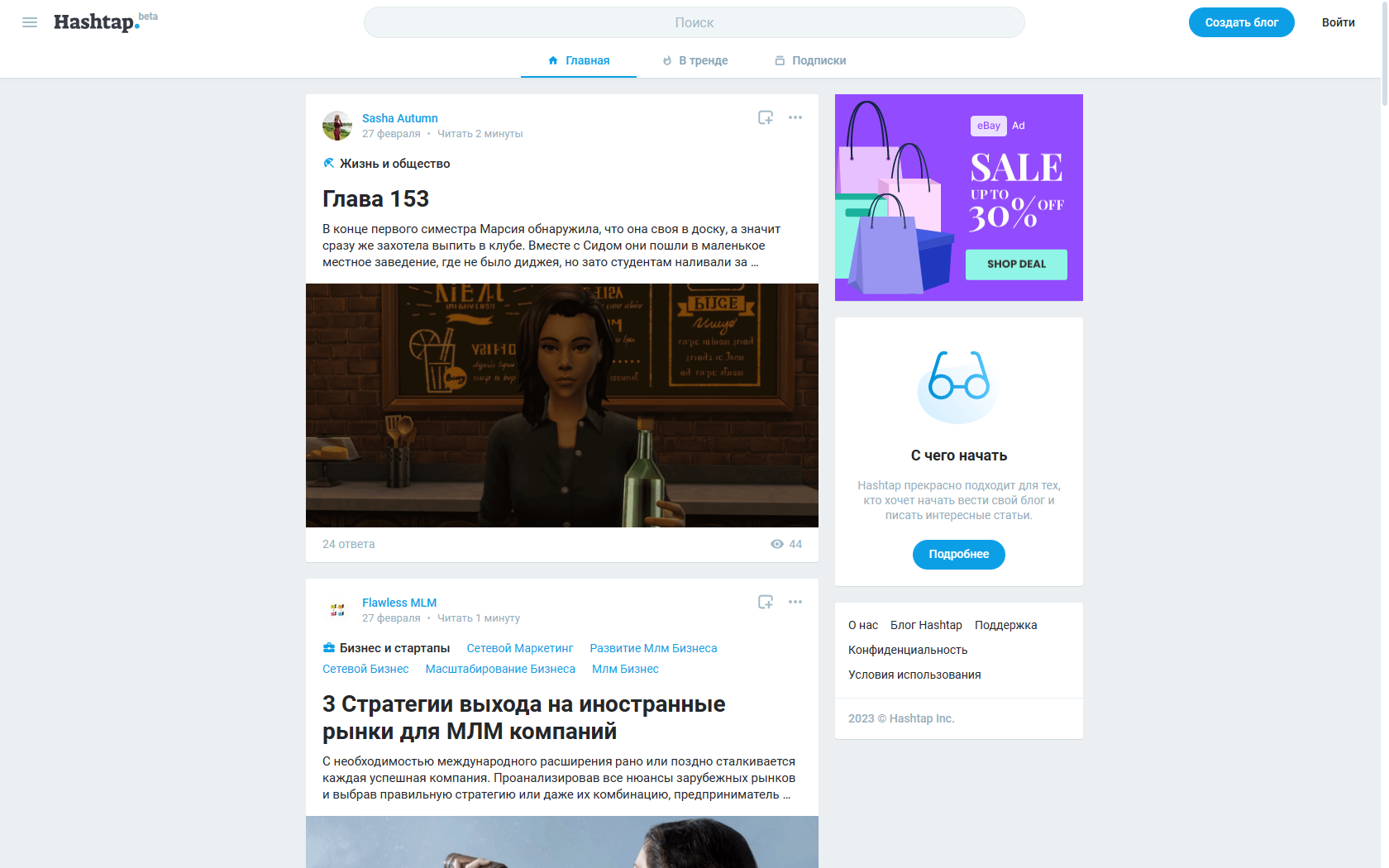


Рисунок 1.3 – Интерфейс [веб-приложения](http://unitoria.ru/blog/kadry/top-5-programm-dlya-kadrovogo-ucheta-v-2021-godu) Hashtap

[Dev.to - это популярное сообщество разработчиков, где они могут обмениваться знаниями, учиться и обсуждать различные темы, связанные с веб-разработкой](https://dev.to/t/webdev). Интерфейс веб-приложения представлен на рисунке 1.4.

Достоинства:

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

Недостатки:

* [неравномерное качество контента: Качество публикаций может сильно варьироваться, поскольку они создаются разными пользователями](https://dev.to/t/webdev);
* [отсутствие модерации: Некоторые пользователи могут заметить, что на сайте отсутствует строгая модерация, что может привести к появлению нерелевантного или низкокачественного контента](https://dev.to/t/webdev).

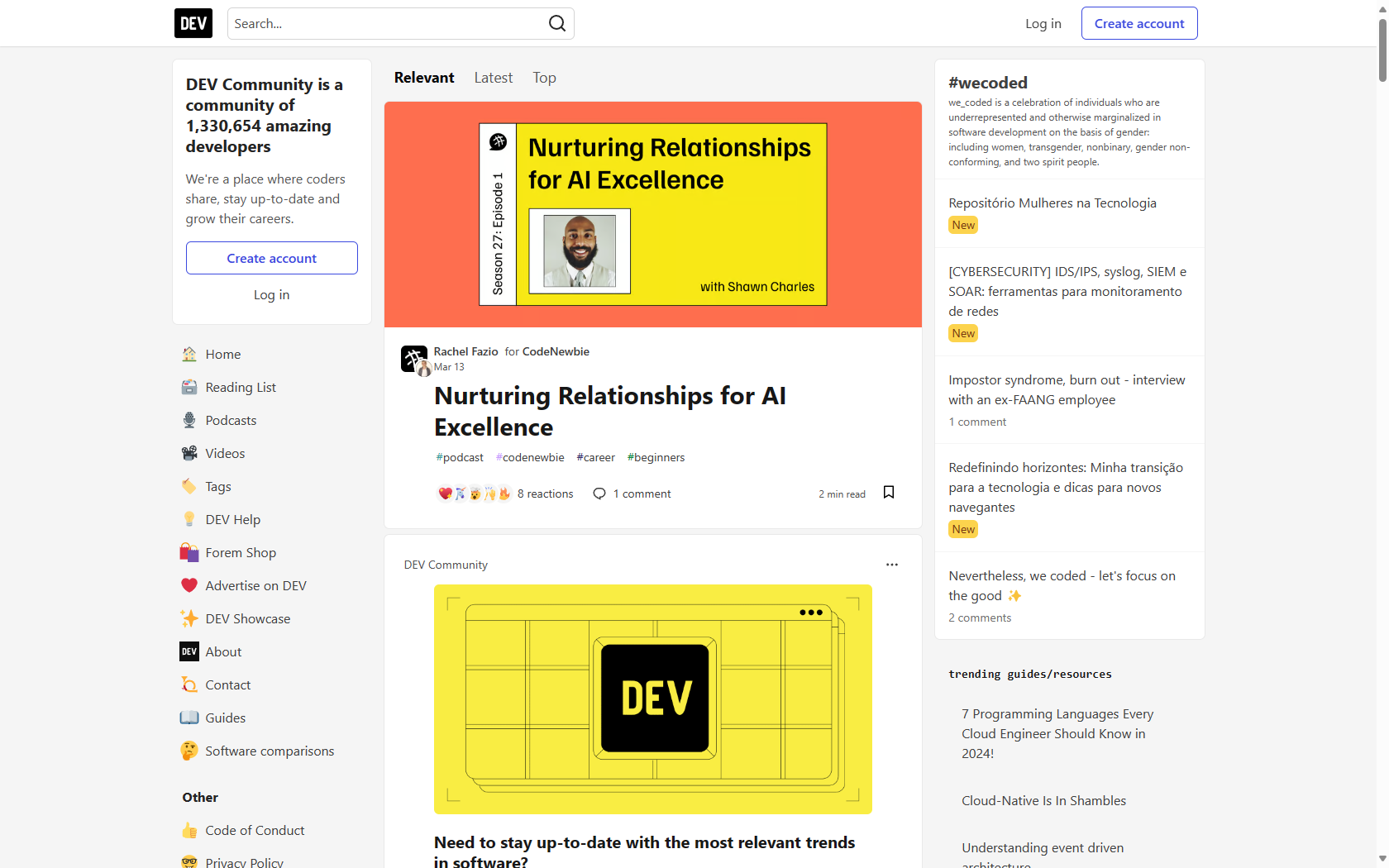


Рисунок 1.4 – Интерфейс [веб-приложения](http://unitoria.ru/blog/kadry/top-5-programm-dlya-kadrovogo-ucheta-v-2021-godu) [Dev.to](https://dev.to/t/webdev)

Таким образом, можно сделать вывод, что существующие аналоги издательских платформ имеют свои преимущества и недостатки, и не полностью удовлетворяют потребностям современных пользователей.

* 1. Средства разработки

При разработке приложения будут использованы

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

Использование данных технологий имеет несколько преимуществ при разработке приложения для управления образовательным процессом.

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 расширяемого языка разметки XAML позволяет 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.

* 1. Разработка функциональных требований

В целом, раздел, Разработка функциональных требований является ключевым для успешной разработки продукта или системы, так как он определяет основные функции и возможности продукта и помогает согласовать ожидания разработчиков и пользователей. Диаграмма вариантов использования представлена на рисунке 1.5.

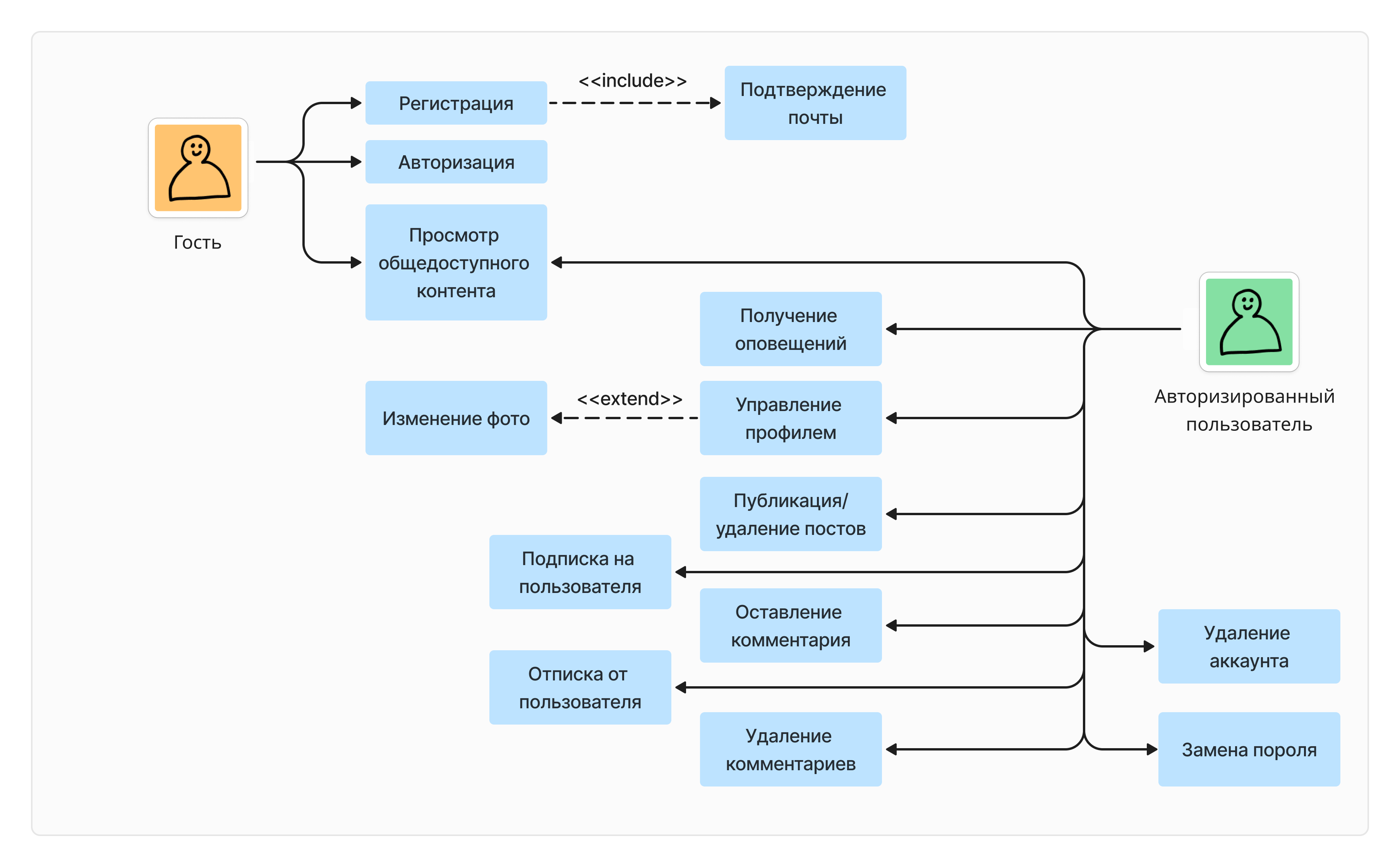


Рисунок 1.5 – Диаграмма вариантов использования

К основным функциональным требованиям относятся:

* регистрация и авторизация пользователей: Приложение должно предоставлять возможность пользователям регистрироваться и входить в систему. Это может включать в себя проверку подлинности по электронной почте и/или двухфакторную аутентификацию;
* публикация контента: Пользователи должны иметь возможность публиковать свой контент в виде текста или изображений. Это может включать в себя функции редактирования и форматирования текста;
* комментирование: Пользователи должны иметь возможность комментировать публикации других пользователей. Это может включать в себя функции ответа на комментарии и оценки комментариев;
* профили пользователей: Пользователи должны иметь возможность создавать и редактировать свои профили. Это может включать в себя функции загрузки аватара, редактирования информации о себе и настройки приватности;
* персонализированные рекомендации: Приложение должно предоставлять персонализированные рекомендации контента на основе интересов и взаимодействий пользователя.
* безопасность: Приложение должно обеспечивать безопасность пользовательских данных. Это может включать в себя шифрование паролей, защиту от атак и соблюдение нормативов по защите данных;
* API Gateway: Приложение должно иметь API Gateway для маршрутизации запросов к соответствующим микросервисам;
* Docker Compose: Все сервисы приложения должны быть управляемы через Docker Compose для обеспечения их совместной работы.

1. Проектирование web-приложения
   1. Обобщенная архитектура

Большое внимание уделяется качеству организации архитектуры приложения, в связи с разработкой схожей системы, было принято решение об использовании микро сервисной архитектурой, а также использование архитектурных паттернов:Gateway, MVC. Взаимодействие сервисов осуществляется благодаря брокерам сообщений: RabbitMQ, GRPC. Данный механизмы позволят легко расширять и масштабировать сервисы, предоставляемые клиенту, не изменяя состояния других частей платформ, что обеспечивает безопасное внедрение новых продуктов на рисунке 2.1.

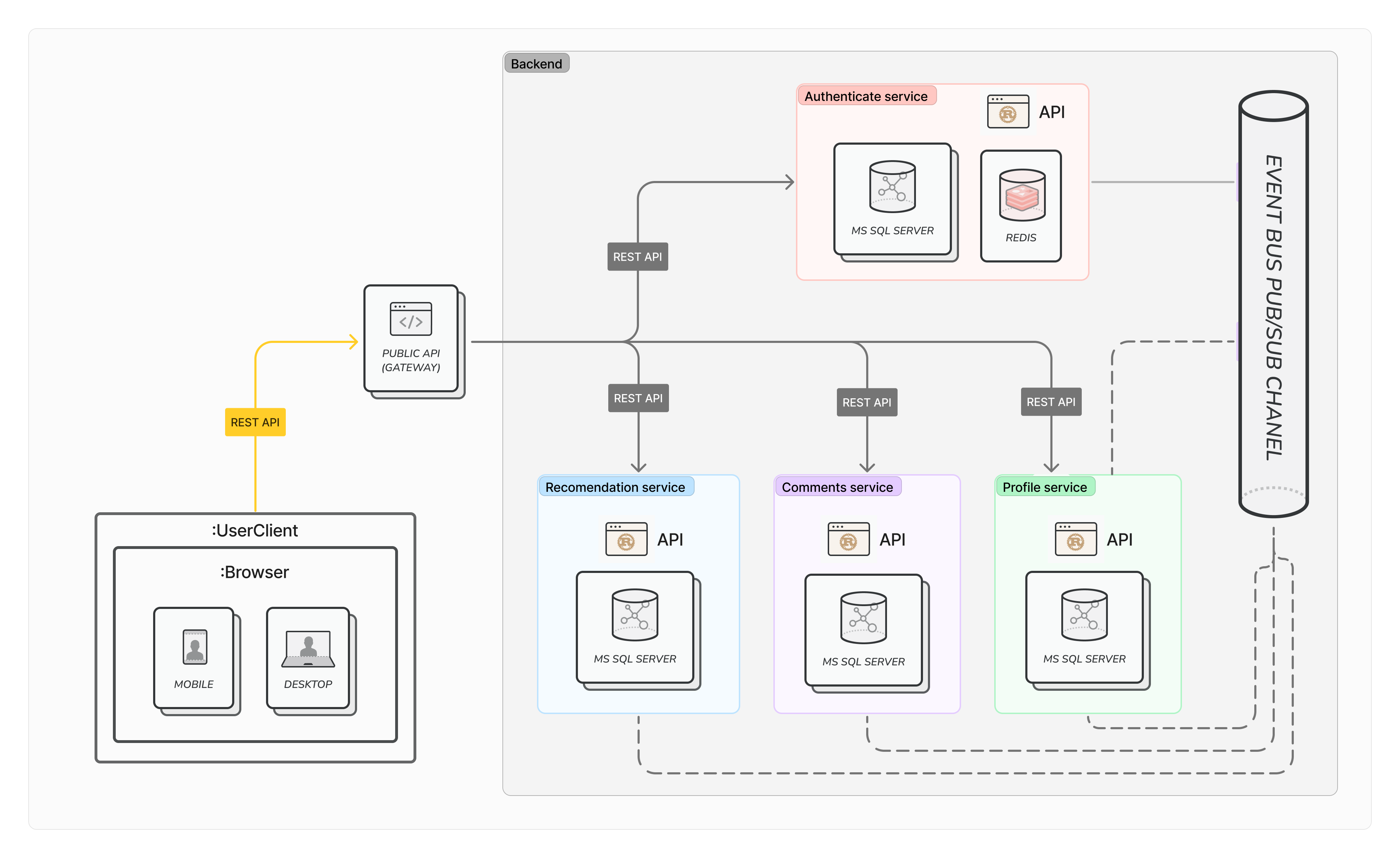


Рисунок 2.1 – Диаграмма размещения

Программное средство состоит из несколько 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 системы.

* 1. Диаграмма UML и блок-схема алгоритма

Диаграмма UML представляет собой графическое изображение структуры системы и взаимосвязей между ее элементами. Для веб-приложения необходимо построить диаграмму, которая будет отражать все основные компоненты системы и связи между ними. В данном случае система состоит из нескольких микросервисов: клиент на React, сервисов по авторизации и регистрации, сервис по управлению постами, сервис по работе с профилем, сервис по работе с рекомендациями и сервис по работе с комментариями.

Для создания диаграммы UML можно использовать различные виды диаграмм, такие как диаграмма вариантов использования (Use Case), диаграмма классов (Class Diagram), диаграмма последовательности (Sequence Diagram), диаграмма компонентов (Component Diagram) и другие.

Для моделирования взаимодействия объектов в языке UML используются соответствующие диаграммы последовательности.

Диаграмма описывает взаимодействие пользователя с системой для получения доступа к главной странице.

Сценарий начинается с того, что пользователь пытается получить доступ к главной странице системы. Система проверяет наличие токена, переданного в запросе, и выполняет проверку его валидности. Если токен не валиден, система перенаправляет пользователя на страницу авторизации.

На странице авторизации пользователь вводит свой email и пароль. Система выполняет проверку этих данных на соответствие данным, хранящимся в базе данных системы. Если введенные данные соответствуют данным в базе данных, система создает новый токен, который далее будет использоваться для аутентификации пользователя. Если же введенные данные не соответствуют данным в базе данных, система отображает сообщение об ошибке и пользователь остается на странице авторизации. Диаграмма последовательности проекта представлена на рисунке 2.2.

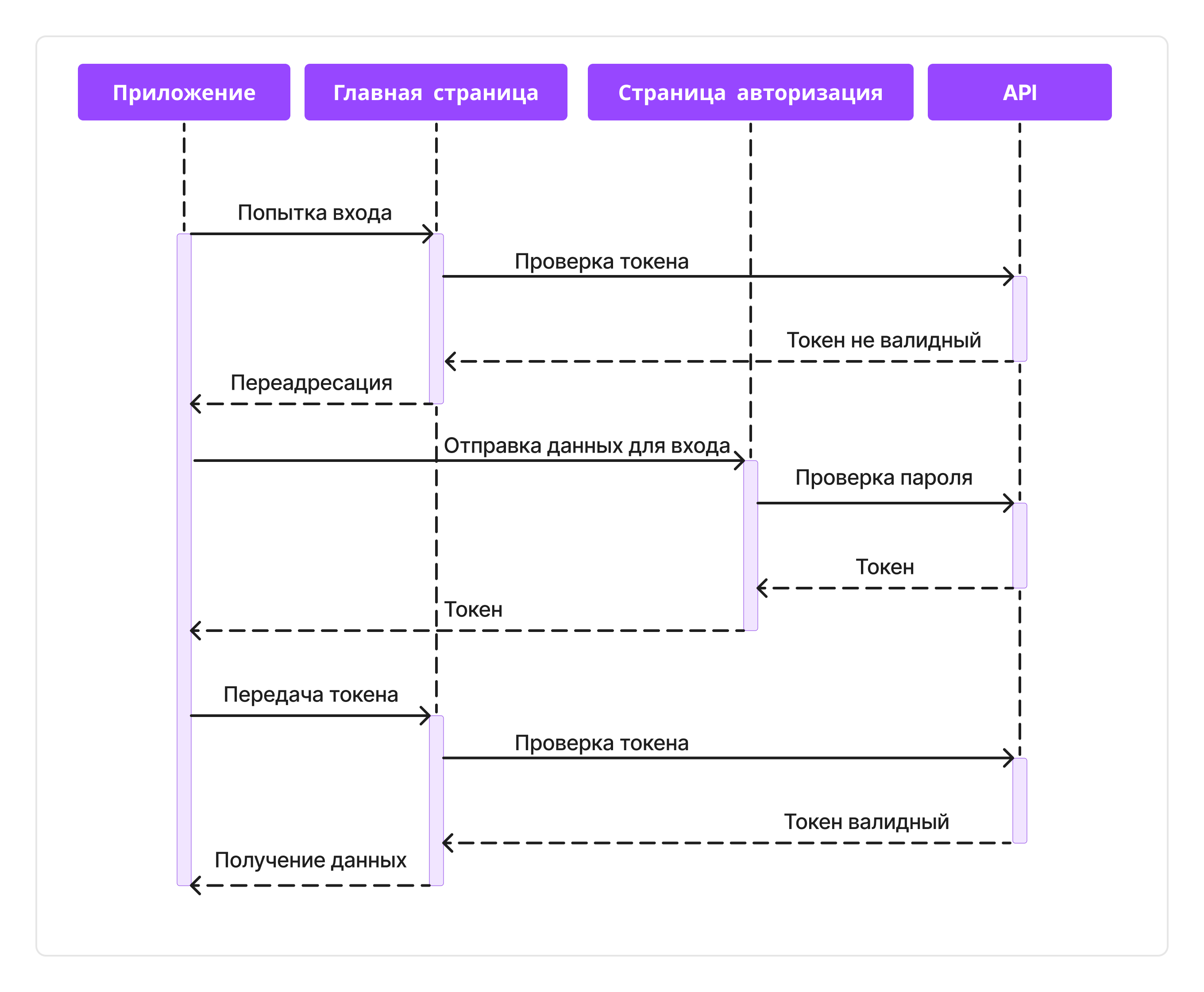


Рисунок 2.2 – Диаграмма последовательности

После успешной авторизации система перенаправляет пользователя обратно на главную страницу, и токен передается в запросе на получение страницы.

Система проверяет валидность токена и, если он валиден, открывает доступ к главной странице.

Блок-схема по регистрации пользователя начинается с заполнения пользователем формы регистрации, в которой он вводит необходимые данные, такие как имя, фамилию, email и пароль. После этого данные проходят процесс валидации, где проверяется их корректность и наличие всех необходимых полей. Если какие-либо данные не проходят валидацию, то процесс регистрации прерывается и пользователю возвращается соответствующий код ошибки, указывающий на то, какое поле заполнено неверно.

Если все данные валидные, то формируется запрос на сервер для проверки наличия такого пользователя в базе данных. Если пользователь с таким email уже существует, то регистрация прерывается и возвращается код ошибки, указывающий на то, что пользователь с таким email уже зарегистрирован. Если же такого пользователя нет, то на основании введенных данных формируется токен, который будет отправлен на указанный email. Блок-схема алгоритма авторизации на рисунке 2.3.

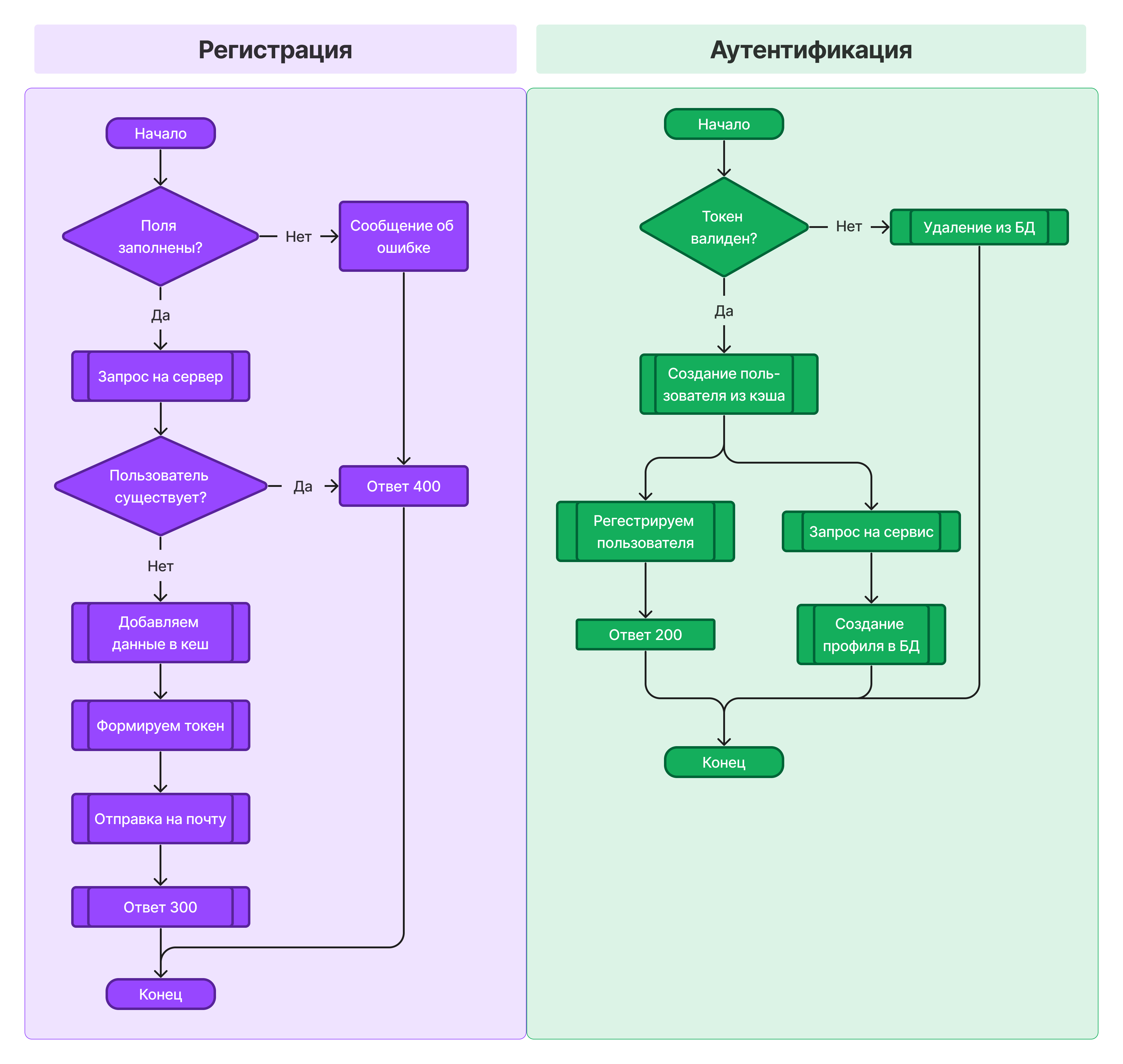


Рисунок 2.3 – Блок-схема авторизации

Чтобы продолжить регистрацию, пользователь должен зайти на свою почту и перейти по ссылке, которая расшифрует токен и получит данные из него. Данные, содержащиеся в токене, отправляются на другой микросервис для создания профиля пользователя.

После успешного создания профиля пользователю возвращается успешный статус код, указывающий на то, что процесс регистрации завершен. Пользователь может войти на сайт, используя свои учетные данные, и получить доступ к функционалу, предоставляемому системой.

* 1. Модель данных

В курсовом проекте используется MS SQL и Redis, которые являются популярными инструментами для хранения и обработки данных.

MS SQL используется как основная база данных для хранения информации, пользователей, профилей, постов и других данных, которые требуют структурированного хранения и связей между таблицами. Он обеспечивает надежность и целостность данных, обеспечивает масштабируемость и поддерживает широкий спектр функциональных возможностей для работы с данными.

Redis используется для хранения данных, которые требуют быстрого доступа, например, кеша данных, информации о токенов пользователей, временных данных, хранения промежуточных результатов вычислений и т.д. Redis является быстрым и эффективным инструментом для хранения данных в оперативной памяти, что позволяет обеспечить быстрый доступ к этим данным.

В целом, использование MS SQL и Redis совместно позволяет создать эффективное и масштабируемое приложение, которое быстро обрабатывает данные и обеспечивает быстрый доступ к ним, при этом обеспечивая надежность и целостность данных.

* 1. Таблицы базы данных

Каждая база данных и каждая таблица будет описана в далее в разделе. Скрипт создания таблиц представлен в приложении Б.

База данных AUTHENTICATE\_SERVICE для хранения идентифицирующих данных представлена на рисунке 2.4.

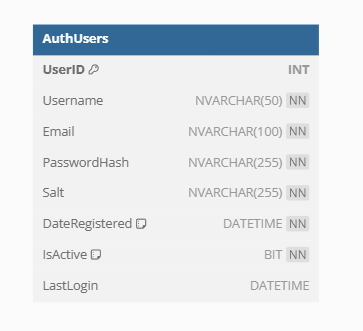


Рисунок 2.4 – Структура базы данных “ AUTHENTICATE\_SERVICE”

Таблица AuthUsers предназначена для хранения основной информации о пользователях, необходимой для процессов аутентификации и идентификации в веб-приложении издательской платформы. Она включает в себя следующие поля:

* UserID: Уникальный идентификатор пользователя в системе. Это поле является первичным ключом и автоматически инкрементируется с каждым новым пользователем;
* Username: Имя пользователя, используемое для входа в систему. Это поле должно быть уникальным среди всех пользователей;
* Email: Адрес электронной почты пользователя, который также используется для входа и восстановления доступа к аккаунту. Адрес почты должен быть уникальным;
* PasswordHash: Хеш пароля пользователя. В это поле записывается хешированный пароль для обеспечения безопасности данных;
* Salt: Случайно сгенерированная строка, используемая в сочетании с паролем пользователя для создания хеша пароля. Это повышает безопасность хранения паролей;
* DateRegistered: Дата и время регистрации пользователя в системе. Значение по умолчанию устанавливается в момент создания записи;
* IsActive: Логическое поле, указывающее, активен ли аккаунт пользователя. Значение 1 означает активный аккаунт, а 0 — деактивированный;
* LastLogin: Дата и время последнего входа пользователя в систему. Это поле может использоваться для отслеживания активности пользователя.

База данных `PROFILE\_SERVICE` предназначена для управления профилями пользователей и их взаимодействием с контентом на издательской платформе. Она включает в себя следующие таблицы:

1. Таблица Users:

* UserID (INT, PRIMARY KEY, IDENTITY): Уникальный идентификатор пользователя;
* Username (NVARCHAR(50), NOT NULL): Имя пользователя, используемое для идентификации в системе;
* Email (NVARCHAR(100), NOT NULL): Электронная почта пользователя;
* PasswordHash (NVARCHAR(255), NOT NULL): Хеш пароля пользователя;
* DateCreated (DATETIME, NOT NULL): Дата и время создания профиля пользователя;
* ProfilePictureUrl (NVARCHAR(255)): URL изображения профиля пользователя;
* Bio (NVARCHAR(500)): Биографическая информация о пользователе.

1. Таблица Posts:

* PostID (INT, PRIMARY KEY, IDENTITY): Уникальный идентификатор поста;
* UserID (INT, NOT NULL): Идентификатор пользователя, создавшего пост;
* Title (NVARCHAR(255), NOT NULL): Заголовок поста;
* Content (NVARCHAR(MAX), NOT NULL): Содержимое поста;
* DatePosted (DATETIME, NOT NULL): Дата и время публикации поста;
* DateUpdated (DATETIME): Дата и время последнего обновления поста;
* IsPublished (BIT, NOT NULL): Статус публикации поста.

1. Таблица Likes:

* LikeID (INT, PRIMARY KEY, IDENTITY): Уникальный идентификатор лайка;
* PostID (INT, NOT NULL): Идентификатор поста, к которому относится лайк;
* UserID (INT, NOT NULL): Идентификатор пользователя, поставившего лайк;
* DateLiked (DATETIME, NOT NULL): Дата и время, когда был поставлен лайк.

1. Таблица Comments:

* CommentID (INT, PRIMARY KEY, IDENTITY): Уникальный идентификатор комментария;
* PostID (INT, NOT NULL): Идентификатор поста, к которому относится комментарий;
* UserID (INT, NOT NULL): Идентификатор пользователя, написавшего комментарий;
* ParentCommentID (INT): Идентификатор родительского комментария для вложенных комментариев;
* Content (NVARCHAR(1000), NOT NULL): Текст комментария;
* DateCommented (DATETIME, NOT NULL): Дата и время написания комментария.

1. Таблица Subscriptions:

* SubscriptionID (INT, PRIMARY KEY, IDENTITY): Уникальный идентификатор подписки;
* SubscriberUserID (INT, NOT NULL): Идентификатор пользователя, который подписывается;
* SubscribedToUserID (INT, NOT NULL): Идентификатор пользователя, на которого подписываются;
* DateSubscribed (DATETIME, NOT NULL): Дата и время оформления подписки.

1. Таблица Tags:

* TagID (INT, PRIMARY KEY, IDENTITY): Уникальный идентификатор тега;
* Name (NVARCHAR(50), NOT NULL): Название тега.

1. Таблица PostTags:

* PostID (INT, NOT NULL): Идентификатор поста;
* TagID (INT, NOT NULL): Идентификатор тега.

Объединение PostID и TagID служит в качестве составного первичного ключа.

Каждая таблица содержит ряд полей, обеспечивающих хранение информации, необходимой для функционирования сервиса профилей. Внешние ключи используются для поддержания целостности данных между таблицами. Уникальные ограничения предотвращают дублирование информации, такое как повторные лайки или подписки. База данных PROFILE\_SERVICE представлена на рисунке 2.5.

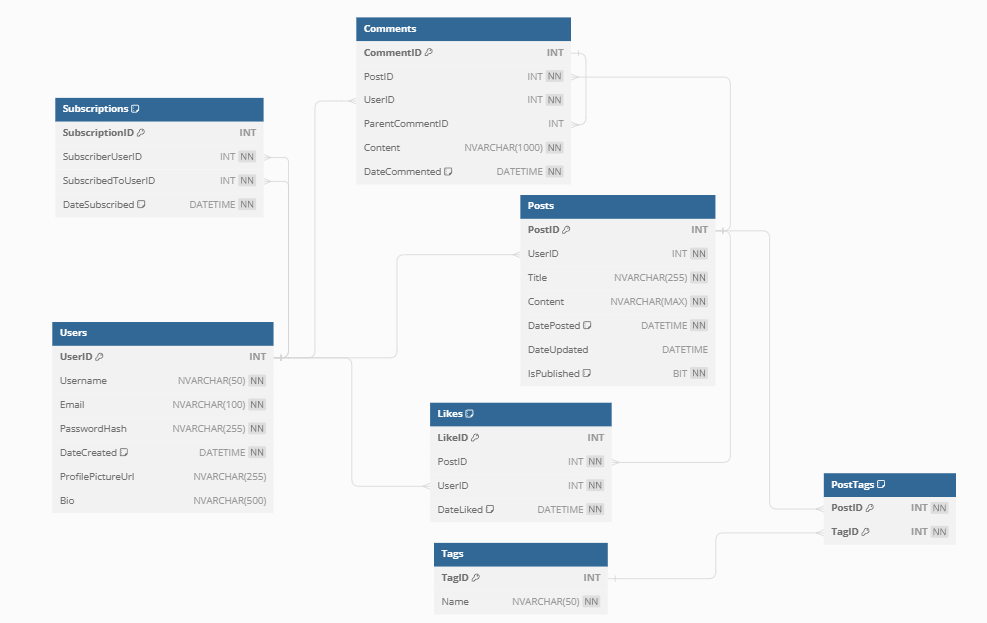


Рисунок 2.5 – Структура базы данных “PROFILE\_SERVICE”

База данных сервиса рекомендаций предназначена для хранения и обработки информации, необходимой для предоставления персонализированных рекомендаций пользователям веб-приложения издательской платформы. Она содержит следующие ключевые элементы:

1. Таблица UserPreferences:

* PreferenceID (INT, PRIMARY KEY, IDENTITY): Уникальный идентификатор предпочтения пользователя;
* UserID (INT, NOT NULL): Идентификатор пользователя, которому принадлежит предпочтение. Внешний ключ, связанный с таблицей пользователей;
* TagID (INT): Идентификатор тега, который предпочитает пользователь. Внешний ключ, связанный с таблицей тегов;
* Category (NVARCHAR(100)): Категория контента, которую предпочитает пользователь;
* DateUpdated (DATETIME, NOT NULL): Дата и время последнего обновления предпочтения.

1. Таблица News:

* NewsID (INT, PRIMARY KEY, IDENTITY): Уникальный идентификатор новостной статьи;
* Title (NVARCHAR(255), NOT NULL): Заголовок новостной статьи;
* Content (NVARCHAR(MAX), NOT NULL): Содержимое новостной статьи;
* DatePublished (DATETIME, NOT NULL): Дата и время публикации статьи;
* PublisherID (INT): Идентификатор издателя статьи. Внешний ключ, связанный с таблицей пользователей.

1. Таблица UserViews:

* ViewID (INT, PRIMARY KEY, IDENTITY): Уникальный идентификатор просмотра;
* UserID (INT, NOT NULL): Идентификатор пользователя, совершившего просмотр;
* ContentID (INT, NOT NULL): Идентификатор просмотренного контента (может быть PostID или NewsID);
* ContentType (NVARCHAR(50), NOT NULL): Тип просмотренного контента (например, 'Post' или 'News') ;
* DateViewed (DATETIME, NOT NULL): Дата и время просмотра.

1. Таблица UserInteractions:

* InteractionID (INT, PRIMARY KEY, IDENTITY): Уникальный идентификатор взаимодействия пользователя с контентом;
* UserID (INT, NOT NULL): Идентификатор пользователя, совершившего взаимодействие;
* ContentID (INT, NOT NULL): Идентификатор контента, с которым было совершено взаимодействие;
* ContentType (NVARCHAR(50), NOT NULL): Тип контента, с которым взаимодействовал пользователь;
* InteractionType (NVARCHAR(50), NOT NULL): Тип взаимодействия (например, 'Like', 'Bookmark');
* DateInteracted (DATETIME, NOT NULL): Дата и время взаимодействия.

Эти таблицы позволяют системе рекомендаций отслеживать интересы и взаимодействия пользователей с различными типами контента, что в свою очередь позволяет формировать персонализированные рекомендации на основе их предпочтений и истории просмотров. База данных RECOMMENDATION\_SERVICE представлена на рисунке 2.6.

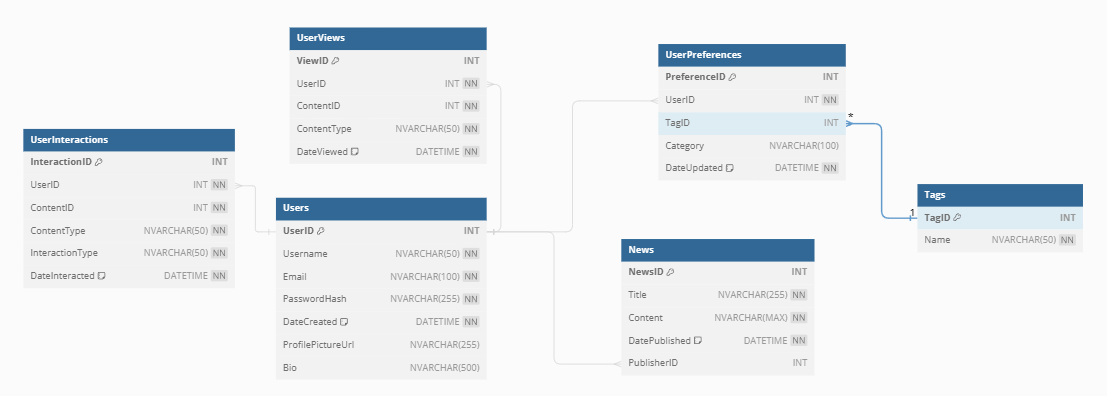


Рисунок 2.6 – Структура базы данных “RECOMMENDATION SERVICE”

* 1. Таблицы базы данных

Каждая таблица базы данных будет описана в далее в таблицах. Скрипт создания таблиц представлен в приложении Б.

Таблица 2.1 – EMPLOYEES

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| emp\_id | NUMBER(6) | Идентификатор сотрудника |
| first\_name | VARCHAR2(20) | Имя сотрудника |
| last\_name | VARCHAR2(25) NOT NULL | Фамилия сотрудника |
| email | VARCHAR2(25) NOT NULL | Электронная почта сотрудника |
| phone\_number | VARCHAR2(25) NOT NULL | Номер телефона сотрудника |
| job\_id | DATE NOT NULL | Идентификатор должности сотрудника |
| salary | VARCHAR2(10) NOT NULL | Зарплата сотрудника |
| commission\_pct | NUMBER(8,2) | Процент комиссии сотрудника |
| manager\_id | NUMBER(6) | Идентификатор менеджера сотрудника |
| department\_id | NUMBER(4) | Идентификатор отдела сотрудника |
| password\_hash | VARCHAR2(128) | Пароль |
| salt | VARCHAR2(128) | Соль для пароля |
| IsAuthentic | NUMBER(1) NOT NULL | Активирован ли аккаунт |

Эта таблица хранит информацию о сотрудниках.

Таблица 2.2 – JOBS

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| job\_id | VARCHAR2(10)PRIMARY KEY | Идентификатор должности |
| job\_title | VARCHAR2(35) NOT NULL | Название должности |
| min\_salary | NUMBER(6) | Минимальная зарплата |
| max\_salary | NUMBER(6) | Максимальная зарплата |

Эта таблица хранит информацию о должностях.

Таблица 2.3 – DEPARTMENTS

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| department\_id | NUMBER(4)PRIMARY KEY | Идентификатор отдела |
| department\_name | VARCHAR2(30) NOT NULL | Название отдела |
| manager\_id | NUMBER(6) | Идентификатор менеджера отдела |
| location\_id | NUMBER(4) | Идентификатор местоположения отдела |

Эта таблица хранит информацию об отделах.

Таблица 2.4 – LOCATIONS

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| location\_id | NUMBER(4) PRIMARY KEY | Идентификатор местоположения |
| street\_address | VARCHAR2(40) | Улица местоположения |
| postal\_code | VARCHAR2(12) | Почтовый индекс |
| City | VARCHAR2(30) NOT NULL | Город местоположения |
| state\_province | VARCHAR2(25) | Область местоположения |
| country\_id | CHAR(2) NOT NULL | Идентификатор страны |

Эта таблица хранит информацию местоположений отдел.

Таблица 2.5 – COUNTRIES

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| country\_id | CHAR(2) PRIMARY KEY | Идентификатор страны |
| country\_name | VARCHAR2(40) NOT NULL | Название страны |

Эта таблица хранит информацию о странах.

Таблица 2.6 – PROJECTS

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| project\_id | NUMBER(4) PRIMARY KEY | Идентификатор проекта |
| project\_name | VARCHAR2(50) NOT NULL | Название проекта |
| end\_date | DATE | Дата окончания проекта (может быть null) |
| start\_date | DATE NOT NULL | Дата начала проекта |

Продолжение таблицы 2.6

|  |  |  |
| --- | --- | --- |
| budget | NUMBER(12,2) | Бюджет проекта |
| department\_id | NUMBER(4) NOT NULL | Идентификатор отдела, ответственного за проект |

Эта таблица хранит информацию о проектах.

Таблица 2.7 – PARTICIPATION

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| emp\_email | VARCHAR2(25) | Электронная почта сотрудника |
| project\_id | NUMBER(4) NOT NULL | Идентификатор проекта, в котором участвует сотрудник |
| role | VARCHAR2(50) | Роль сотрудника в проекте |
| Hours | NUMBER(4) | Количество часов, затраченных сотрудником на проект |
| PRIMARY KEY (emp\_id, project\_id) |  | Составной первичный ключ из emp\_id и project\_id |

Эта таблица хранит информацию об участии сотрудников в проектах.

Таблица 2.8 – TASKS

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| task\_id | NUMBER(6) PRIMARY KEY | Идентификатор задачи |
| task\_name | VARCHAR2(100) NOT NULL | Название задачи |
| Description | VARCHAR2(500) | Описание задачи |
| Duration | NUMBER(4) NOT NULL | Продолжительность задачи в часах |
| project\_id | NUMBER(4) NOT NULL | Идентификатор проекта, к которому относится задача |

Эта таблица хранит информацию о задачах сотрудников.

Таблица 2.9 – VACATIONS

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| vacation\_id | NUMBER(6)PRIMARY KEY | Идентификатор отпуска |
| Reason | VARCHAR2(500) | Причина отпуска |
| emp\_email | VARCHAR2(25) | Электронная почта сотрудника |
| start\_date | DATE NOT NULL | Дата начала отпуска |
| end\_date | DATE NOT NULL | Дата окончания отпуска |

Эта таблица хранит информацию о отпусках сотрудников.

1. Разработка web-приложения
   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.

* 1. Создание клиентской части

ReactJS - это популярная JavaScript-библиотека для создания пользовательских интерфейсов, разработанная Facebook. Bootstrap – это бесплатный набор инструментов для создания сайтов и веб-приложений. Он включает в себя HTML и CSS-шаблоны дизайна для типографии, форм, кнопок, навигации и других компонентов интерфейса.

Основой React являются компоненты. Компоненты - это маленькие, повторно используемые куски кода, которые отвечают за отдельную часть UI. Они могут быть простыми, как кнопка или ввод, или более сложными, объединяя несколько других компонентов.

С помощью Bootstrap мы можем быстро и легко создавать красивые и отзывчивые интерфейсы. Bootstrap предлагает готовые компоненты, такие как навигационные панели, карусели, модальные окна и многое другое, которые мы можем использовать в нашем приложении.

Для создания маршрутизации в React-приложении мы используем React Router. Это позволяет нам создавать многопользовательские веб-приложения с несколькими страницами.

В нашем приложении мы также используем API для получения данных. Мы используем fetch или axios для отправки HTTP-запросов к серверу и получения данных.

1. Тестирование производительности

Для проверки производительности базы данных необходимо заполнить ее большим количеством различных данных и узнать время выполнения одного запроса. Для тестирования производительности мы добавляем в таблицу Employees 100000 строк. Скрипт вставки представлен на листинге 5.1.

|  |
| --- |
| DECLARE v\_counter NUMBER := 0; BEGIN  WHILE v\_counter <= 100000 LOOP  INSERT INTO employees (first\_name, last\_name, email, phone\_number, job\_id, salary, commission\_pct, manager\_id, department\_id, ISAUTHENTIC) VALUES (  DBMS\_RANDOM.STRING('A', 10),  DBMS\_RANDOM.STRING('A', 10),  'email' || TO\_CHAR(v\_counter) || '@example.com',  '1234567890',  'AD\_PRES',  DBMS\_RANDOM.VALUE(2000, 8000),  DBMS\_RANDOM.VALUE(0, 0.2),  null,  10,  0  );  v\_counter := v\_counter + 1;  END LOOP; END; / |

Листинг 5.1 – Вставка 100000 строк в таблицу

Для получения плана выполнения SQL-запроса в Oracle можно воспользоваться инструментом EXPLAIN PLAN используя функцию “Explain Plan”. [Результат будет отображаться в специальной вкладке “Plan” в формате, смешивающем древовидную структуру и таблицу.](https://www.jetbrains.com/help/datagrip/query-execution-plan.html)

Каждая строка плана выполнения представляет операцию, которую база данных выполнит, а также ее стоимость, учитывая различные факторы, такие как использование индексов, объем данных и другие. Каждая операция оценивается по ожидаемому количеству возвращаемых строк. В данном случае план выполнения состоит из двух строк:

SELECT STATEMENT: это верхний уровень плана выполнения, который описывает операцию SELECT в целом.

TABLE ACCESS FULL: это операция, которая читает все строки таблицы (без использования индексов), и возвращает их в качестве результата запроса. Стоимость этого плана выполнения равна 377 (без указания процента использования ресурсов ЦПУ) и означает, что база данных ожидает, что выполнение этого запроса займет примерно 377 единицы измерения работы (например, время CPU или количество операций ввода-вывода). Результат представлен на рисунке 5.1.

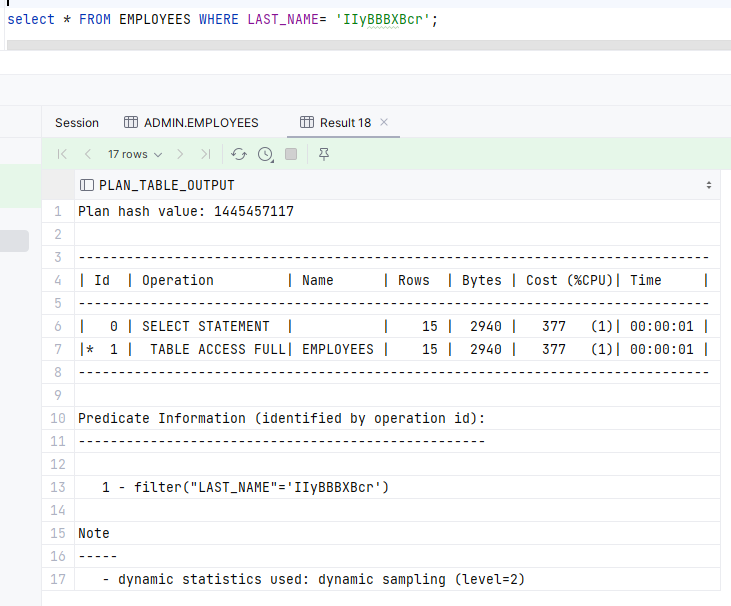


Рисунок 5.1 – Получения плана выполнения запроса

Проведенный анализ производительности базы данных с таблицей EMPLOYEES, содержащей большое количество строк, позволяет сделать следующие выводы. Первоначально, до создания индексов, запрос к таблице осуществлялся полным сканированием, что может привести к замедлению при работе с большим объемом данных.

Создание индексов на соответствующих столбцах таблицы существенно улучшило время выполнения запроса. Индексы позволяют эффективнее организовывать доступ к данным, снижая стоимость выполнения операций. Таким образом, база данных оказывается более подготовленной к обработке больших объемов данных, что является важным аспектом в условиях активной работы системы с множеством запросов и операций.

1. Тестирование web-приложения
   1. Тестирование клиента на React

Важным этапом тестирования является проверка системы маршрутизации. Это включает в себя подтверждение того, что каждый запрос корректно перенаправляется на соответствующий обработчик. Для этого были созданы тестовые сценарии, которые имитируют различные типы запросов (GET, POST, PUT, DELETE) и проверяют, что сервер правильно их обрабатывает и возвращает ожидаемые ответы.

Тестирование сервера

1. 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.

* 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 6.1.

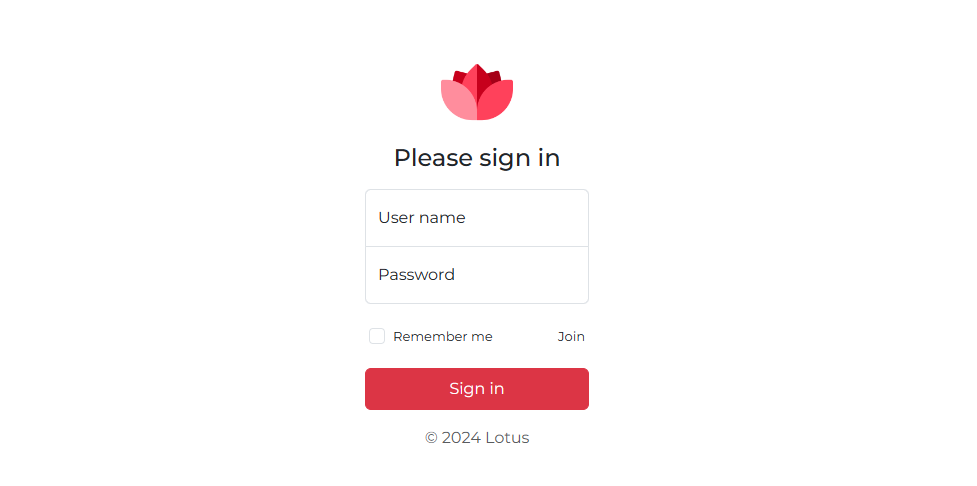


Рисунок 6.1 – Страница входа

* 1. 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 6.2.

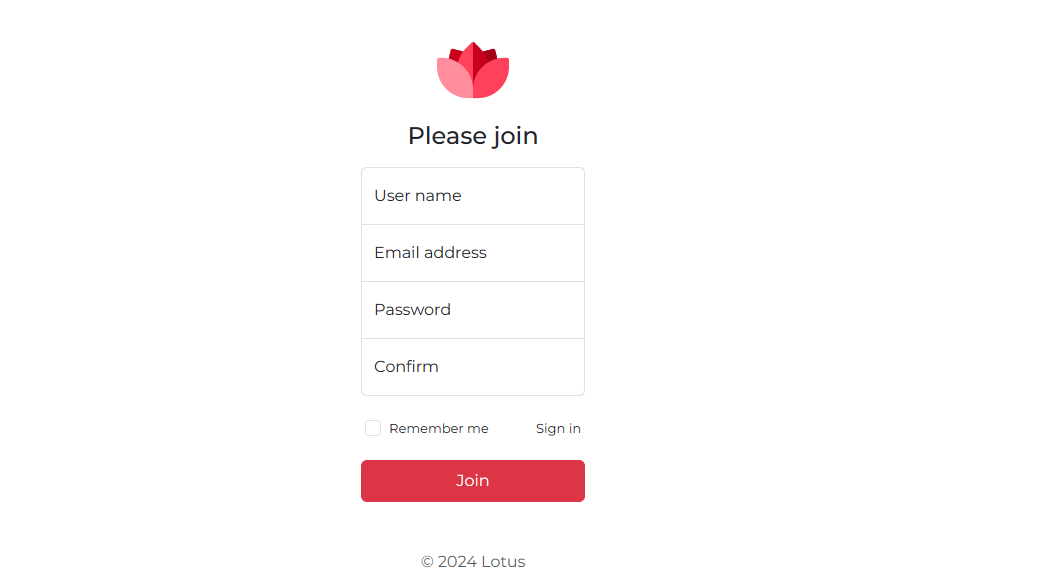


Рисунок 6.2 – Страница регистрации

* 1. 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 6.3.

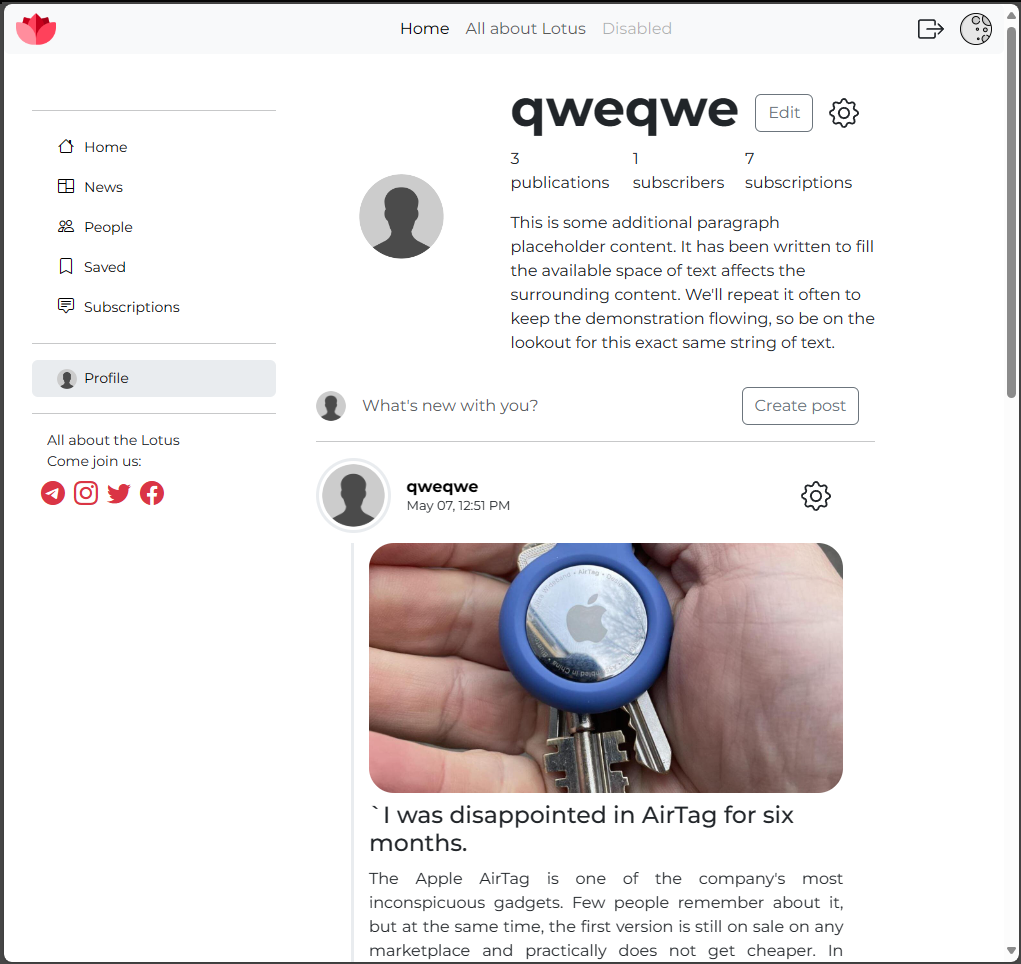


Рисунок 6.3 – Страница профиля пользователя

* 1. 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 6.4.

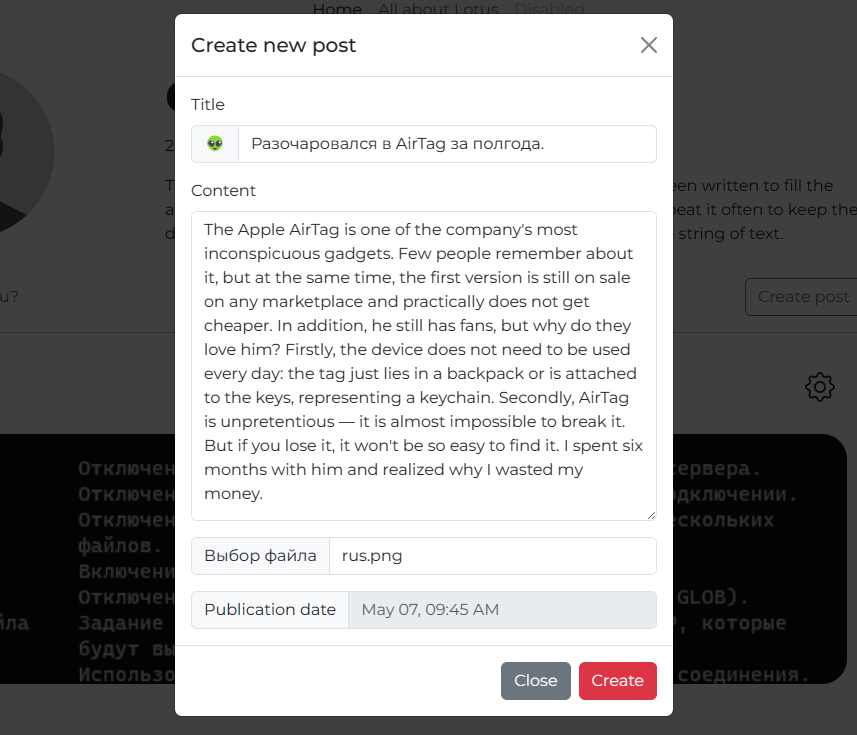


Рисунок 6.4 – Модальное окно для создания поста

* 1. 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 6.4.

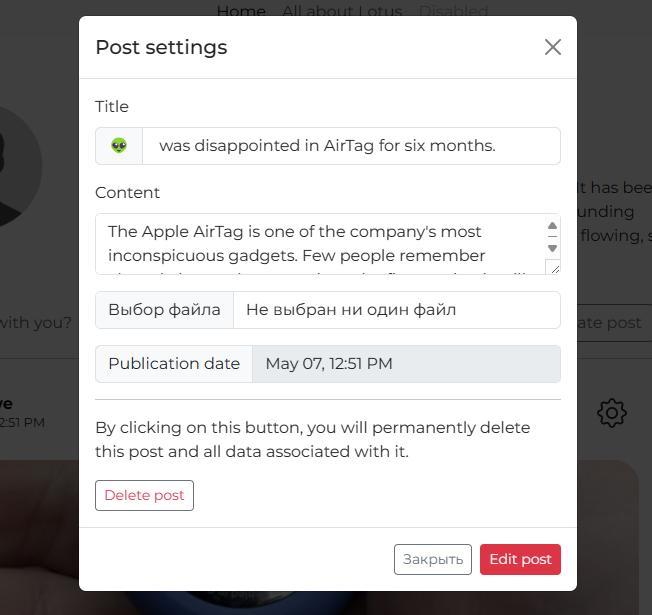


Рисунок 6.4 – Модальное окно для редактирования поста

* 1. 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 6.5.

Рисунок 6.5 – Страница подписок и подписчиков

* 1. Страница новостей

При переходе на страницу новостей пользователь может просматривать последние новости, а также переключаться между интересующими его темами. Страница новостей представлена на рисунке 6.6.

Рисунок 6.6 – Страница новостей

* 1. 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 6.7.

Рисунок 6.7 – Страница «Для вас»

* 1. 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 текс, 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 6.8.

Рисунок 6.8 – Страница комментариев под одним из постов

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.

* Реализация архитектуры: Архитектура приложения, основанная на микросервисах, обеспечивает гибкость и масштабируемость системы.
* Функциональность микросервисов: Каждый микросервис, включая авторизацию, уведомления, контент, комментарии, профили, рекомендации и безопасность, был реализован согласно требованиям задания.
* Клиентская часть и API Gateway: Клиентская часть на React и API Gateway для маршрутизации запросов были успешно разработаны и интегрированы с микросервисами.
* Тестирование и деплой: Приложение было тщательно протестировано и готово к деплою, что подтверждается наполнением базы данных тестовыми данными и стабильной работой всех сервисов.

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

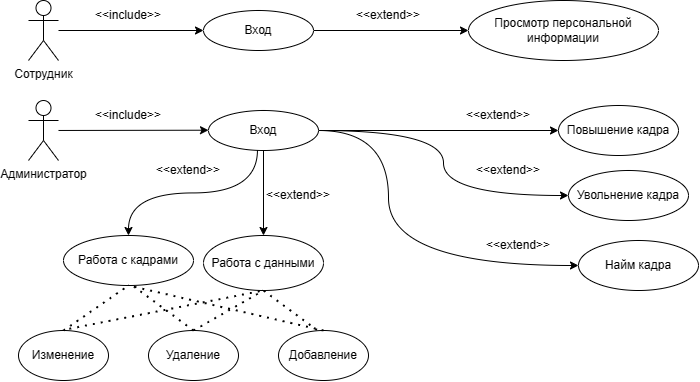
Таким образом, цель данной работы была успешно достигнута. Созданное веб-приложение «Издательская платформа» успешно сочетает в себе функциональность и безопасность, предоставляя пользователям мощный инструмент для обмена и открытия нового контента. Это делает его ценным ресурсом как для авторов, так 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: 15.03.2023.
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

Диаграмма вариантов использования



Приложение Б

Листинг создания таблиц

|  |
| --- |
| CREATE DATABASE AUTHENTICATION\_SERVICE;  USE AUTHENTICATION\_SERVICE;  CREATE TABLE USERS (  ID INT PRIMARY KEY IDENTITY,  GOOGLE\_ID NVARCHAR(255),  GITHUB\_ID NVARCHAR(255),  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  );  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 TABLE SAVED\_POSTS (  USER\_ID INT NOT NULL,  POST\_ID INT NOT NULL,  createdAt DATETIME NOT NULL,  updatedAt DATETIME NOT NULL,  CONSTRAINT PK\_SAVED\_POSTS PRIMARY KEY (USER\_ID, POST\_ID),  CONSTRAINT FK\_SAVED\_POSTS\_USER\_ID FOREIGN KEY (USER\_ID) REFERENCES USERS (ID),  CONSTRAINT FK\_SAVED\_POSTS\_POST\_ID FOREIGN KEY (POST\_ID) REFERENCES POSTS (ID)  ); |

Приложение В

Листинг создания триггеров

|  |
| --- |
| CREATE OR REPLACE TRIGGER emp\_id\_trg  BEFORE INSERT ON employees FOR EACH ROW  BEGIN  :NEW.salt := generate\_salt(:NEW.email);  SELECT emp\_id\_seq.NEXTVAL INTO :new.emp\_id FROM dual;  END;/  CREATE OR REPLACE TRIGGER department\_id\_trg  BEFORE INSERT ON departments FOR EACH ROW  BEGIN  SELECT department\_id\_seq.NEXTVAL INTO :new.department\_id FROM dual;  END;/  CREATE OR REPLACE TRIGGER location\_id\_trg  BEFORE INSERT ON locations FOR EACH ROW  BEGIN  SELECT location\_id\_seq.NEXTVAL INTO :new.location\_id FROM dual;  END;/  CREATE OR REPLACE TRIGGER project\_id\_trg  BEFORE INSERT ON projects FOR EACH ROW  BEGIN  SELECT project\_id\_seq.NEXTVAL INTO :new.project\_id FROM dual;  END;/  CREATE OR REPLACE TRIGGER task\_id\_trg  BEFORE INSERT ON tasks FOR EACH ROW  BEGIN  SELECT task\_id\_seq.NEXTVAL INTO :new.task\_id FROM dual;  END;/  CREATE OR REPLACE TRIGGER vacation\_id\_trg  BEFORE INSERT ON vacations FOR EACH ROW  BEGIN  SELECT vacation\_id\_seq.NEXTVAL INTO :new.vacation\_id FROM dual;  END;/ |

Приложение Г

Скрипт создания процедуры export\_json

|  |
| --- |
| CREATE OR REPLACE PROCEDURE export\_json AS  v\_file UTL\_FILE.FILE\_TYPE;  v\_data employees%ROWTYPE;  v\_json VARCHAR2(32767);  v\_count NUMBER := 0;  v\_total NUMBER := 0; BEGIN  -- Получаем общее количество записей  SELECT COUNT(\*) INTO v\_total FROM employees;   -- Открываем файл для записи  v\_file := UTL\_FILE.FOPEN('MY\_DIRECTORY', 'employees.json', 'w', 32767);   -- Записываем открывающую квадратную скобку  UTL\_FILE.PUT\_LINE(v\_file, '[');   -- Начинаем цикл по всем записям в таблице employees  FOR v\_data IN (SELECT \* FROM employees) LOOP  -- Преобразуем каждую запись в JSON  v\_json := JSON\_OBJECT(  'emp\_id' VALUE v\_data.emp\_id,  'first\_name' VALUE v\_data.first\_name,  'last\_name' VALUE v\_data.last\_name,  'email' VALUE v\_data.email,  'phone\_number' VALUE v\_data.phone\_number,  'job\_id' VALUE v\_data.job\_id,  'salary' VALUE v\_data.salary,  'commission\_pct' VALUE v\_data.commission\_pct,  'manager\_id' VALUE v\_data.manager\_id,  'department\_id' VALUE v\_data.department\_id,  'password\_hash' VALUE v\_data.PASSWORD\_HASH,  'isauthentic' value v\_data.ISAUTHENTIC  );   -- Увеличиваем счетчик  v\_count := v\_count + 1;   -- Записываем JSON в файл  IF v\_count < v\_total THEN  UTL\_FILE.PUT\_LINE(v\_file, v\_json || ',');  ELSE  UTL\_FILE.PUT\_LINE(v\_file, v\_json);  END IF;   -- Записываем JSON в файл -- UTL\_FILE.PUT\_LINE(v\_file, v\_json);  END LOOP;    -- Записываем закрывающую квадратную скобку  UTL\_FILE.PUT\_LINE(v\_file, ']');  -- Закрываем файл  UTL\_FILE.FCLOSE(v\_file); EXCEPTION  WHEN OTHERS THEN  -- Если произошла ошибка, закрываем файл  IF UTL\_FILE.IS\_OPEN(v\_file) THEN  UTL\_FILE.FCLOSE(v\_file);  END IF;  RAISE; END export\_json; / |

Приложение Д

Скрипт создания процедуры import\_json

|  |
| --- |
| CREATE OR REPLACE PROCEDURE import\_json IS  v\_file UTL\_FILE.FILE\_TYPE;  v\_data CLOB;  v\_line VARCHAR2(32767);  v\_json JSON\_ARRAY\_T;  v\_json\_obj JSON\_OBJECT\_T;  v\_emp\_rec employees%ROWTYPE; BEGIN  -- Открываем файл для чтения  v\_file := UTL\_FILE.FOPEN('MY\_DIRECTORY', 'employees.json', 'r');   -- Читаем данные из файла  BEGIN  LOOP  UTL\_FILE.GET\_LINE(v\_file, v\_line);  v\_data := v\_data || v\_line;  END LOOP;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  NULL; -- Игнорируем исключение, так как это означает, что мы достигли конца файла  END;   -- Закрываем файл  UTL\_FILE.FCLOSE(v\_file);   -- Парсим JSON  v\_json := JSON\_ARRAY\_T.parse(v\_data);   -- Извлекаем данные из JSON и вставляем их в таблицу  FOR i IN 0 .. v\_json.get\_size() - 1 LOOP  v\_json\_obj := JSON\_OBJECT\_T(v\_json.get(i));  v\_emp\_rec.emp\_id := v\_json\_obj.get\_number('emp\_id');  v\_emp\_rec.first\_name := v\_json\_obj.get\_string('first\_name');  v\_emp\_rec.last\_name := v\_json\_obj.get\_string('last\_name');  v\_emp\_rec.email := v\_json\_obj.get\_string('email');  v\_emp\_rec.phone\_number := v\_json\_obj.get\_string('phone\_number');  v\_emp\_rec.job\_id := v\_json\_obj.get\_string('job\_id');  v\_emp\_rec.salary := v\_json\_obj.get\_number('salary');  v\_emp\_rec.commission\_pct := v\_json\_obj.get\_number('commission\_pct');  v\_emp\_rec.manager\_id := v\_json\_obj.get\_number('manager\_id');  v\_emp\_rec.department\_id := v\_json\_obj.get\_number('department\_id');  v\_emp\_rec.password\_hash := v\_json\_obj.get\_string('password\_hash');  v\_emp\_rec.ISAUTHENTIC := v\_json\_obj.get\_string('isauthentic');   INSERT INTO employees VALUES v\_emp\_rec;  END LOOP;   COMMIT; EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  RAISE; END import\_json; / |