**Оглавление**

[Введение 3](#_Toc161917164)

[1 Постановка задачи 4](#_Toc161917165)

[1.1 Аналитический обзор аналогов 4](#_Toc161917166)

[1.2 Средства разработки 7](#_Toc161917167)

[1.3 Разработка функциональных требований 8](#_Toc161917168)

[2 Проектирование архитектуры приложения 10](#_Toc161917169)

[2.1 Обобщенная архитектура 10](#_Toc161917170)

[2.2 Диаграмма UML и блок-схема алгоритма 11](#_Toc161917171)

[2.3 Модель данных 14](#_Toc161917172)

[2.4 Таблицы базы данных 14](#_Toc161917173)

[3 Разработка web-приложения 19](#_Toc161917174)

[3.1 Микросервисы 19](#_Toc161917175)

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/).

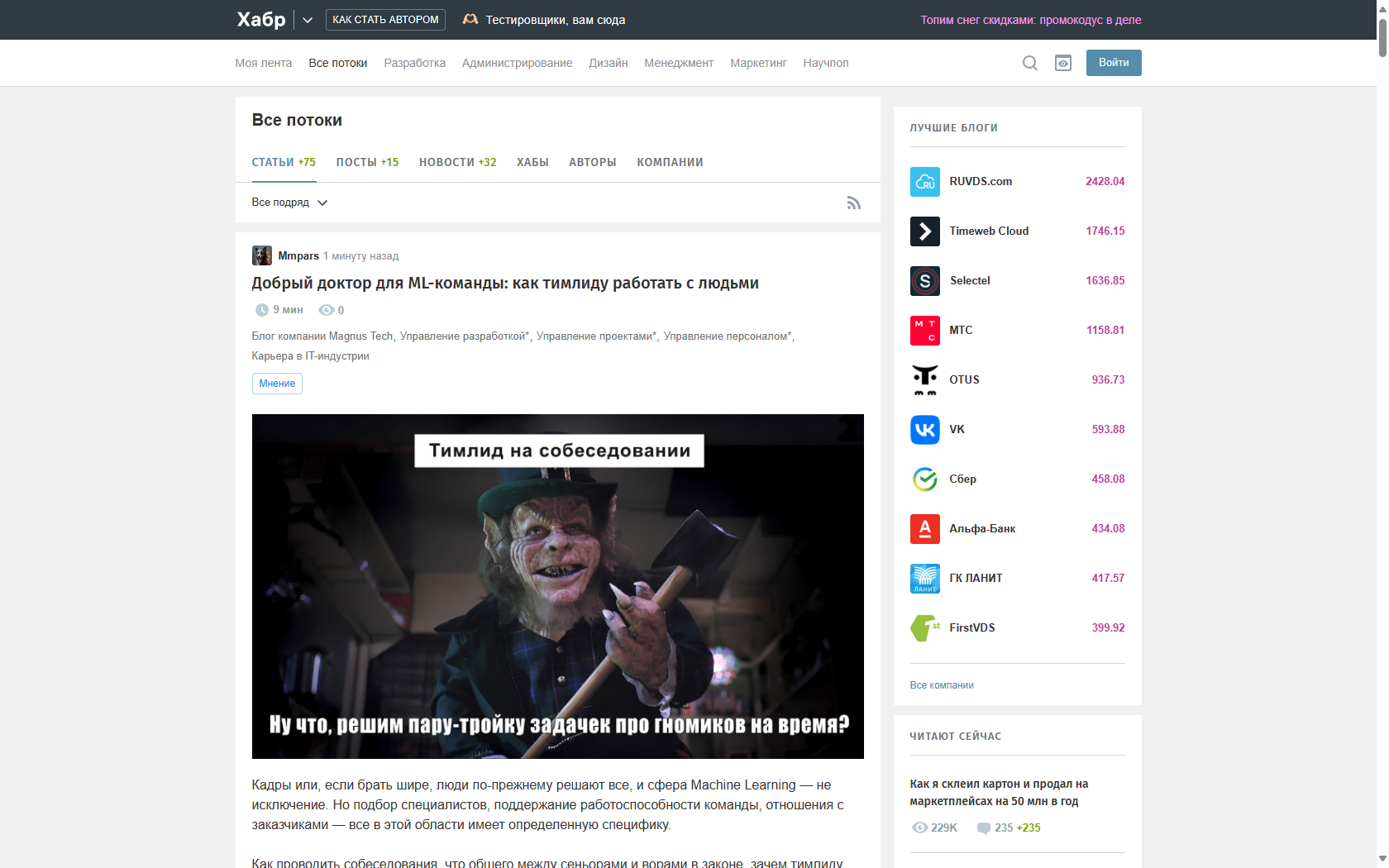


Рисунок 1.1 – Интерфейс [веб-приложения](http://unitoria.ru/blog/kadry/top-5-programm-dlya-kadrovogo-ucheta-v-2021-godu) Хабр

[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.

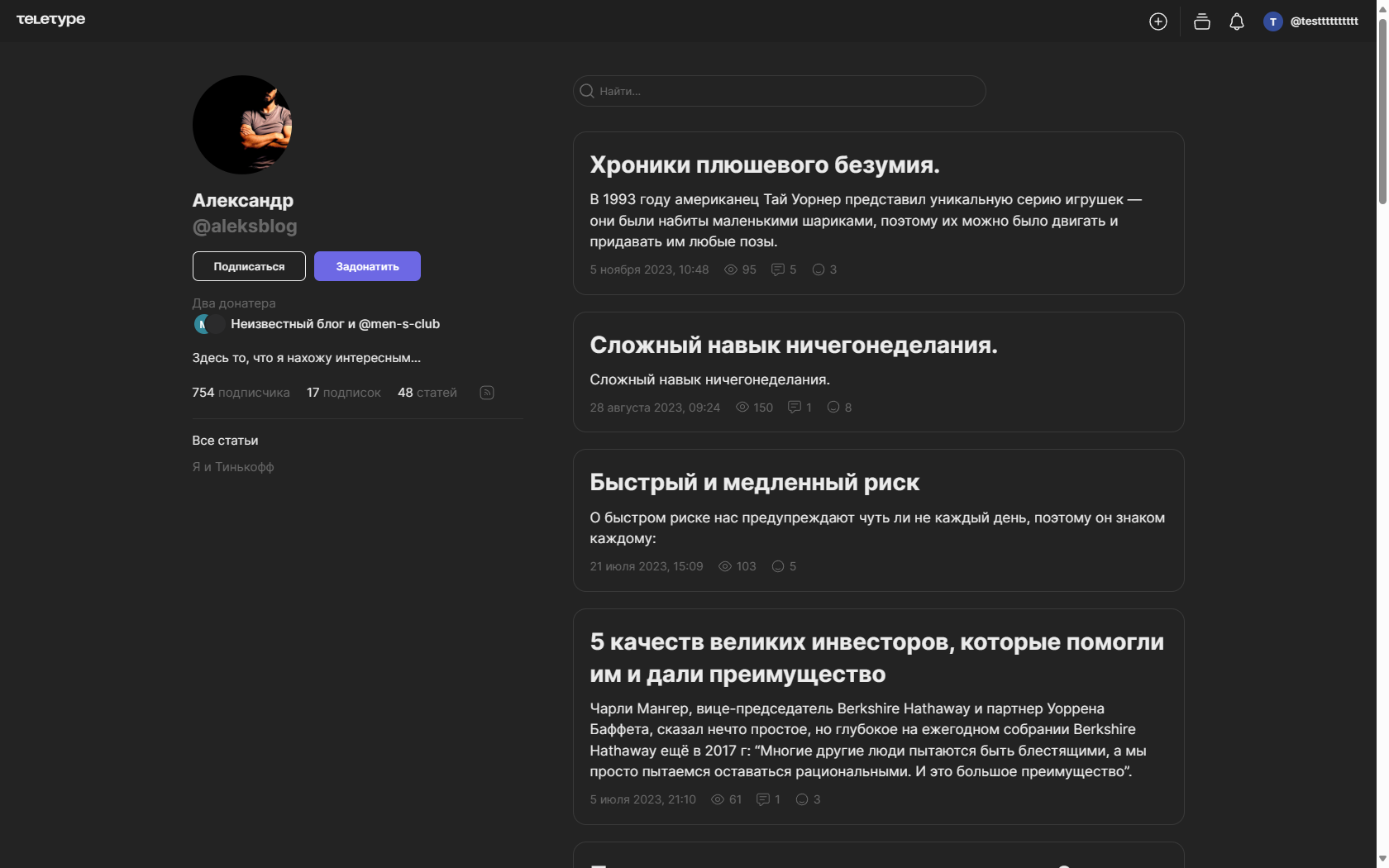


Рисунок 1.2 – Интерфейс [веб-приложения](http://unitoria.ru/blog/kadry/top-5-programm-dlya-kadrovogo-ucheta-v-2021-godu) Teletype

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.

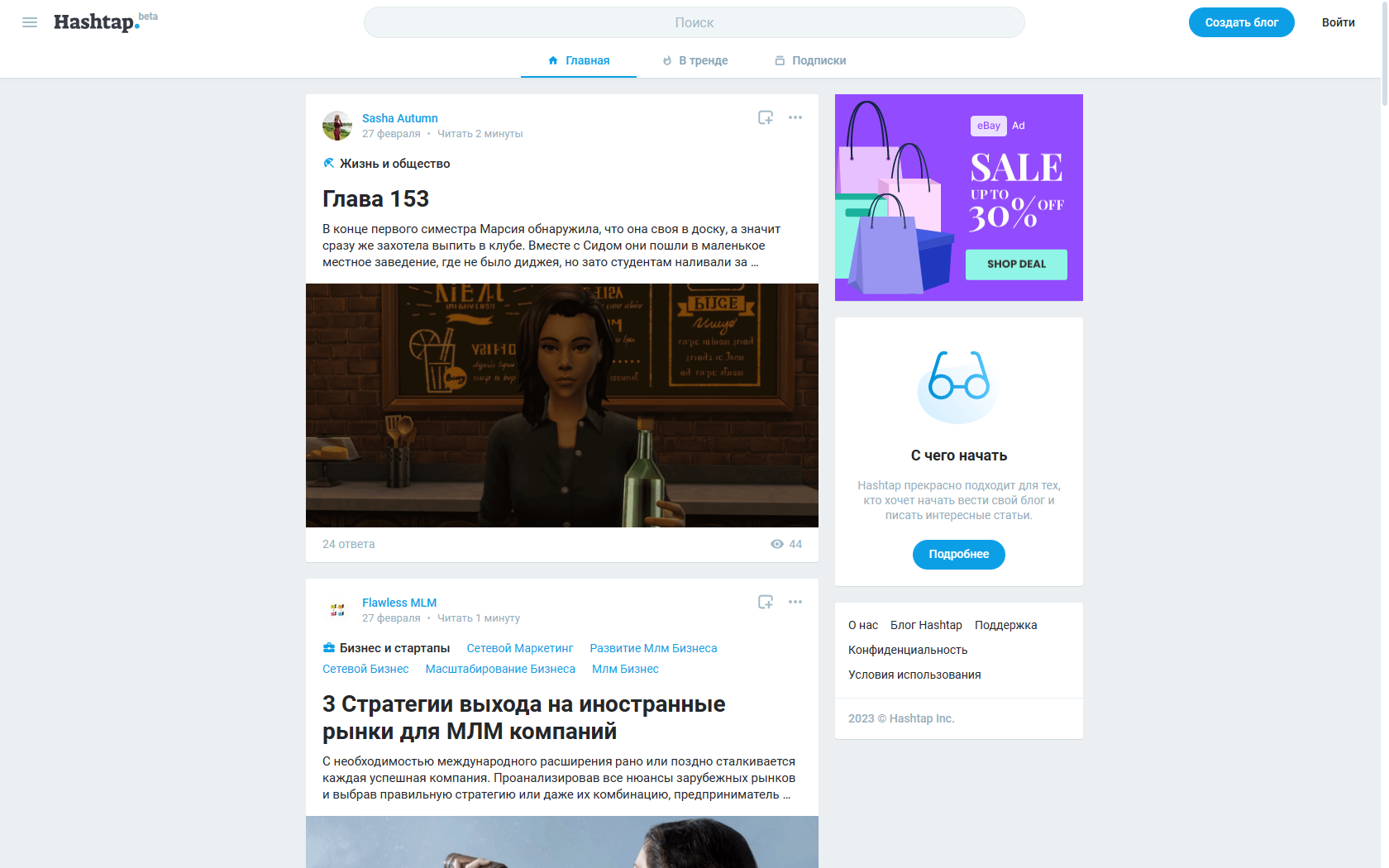


Рисунок 1.3 – Интерфейс [веб-приложения](http://unitoria.ru/blog/kadry/top-5-programm-dlya-kadrovogo-ucheta-v-2021-godu) Hashtap

[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).

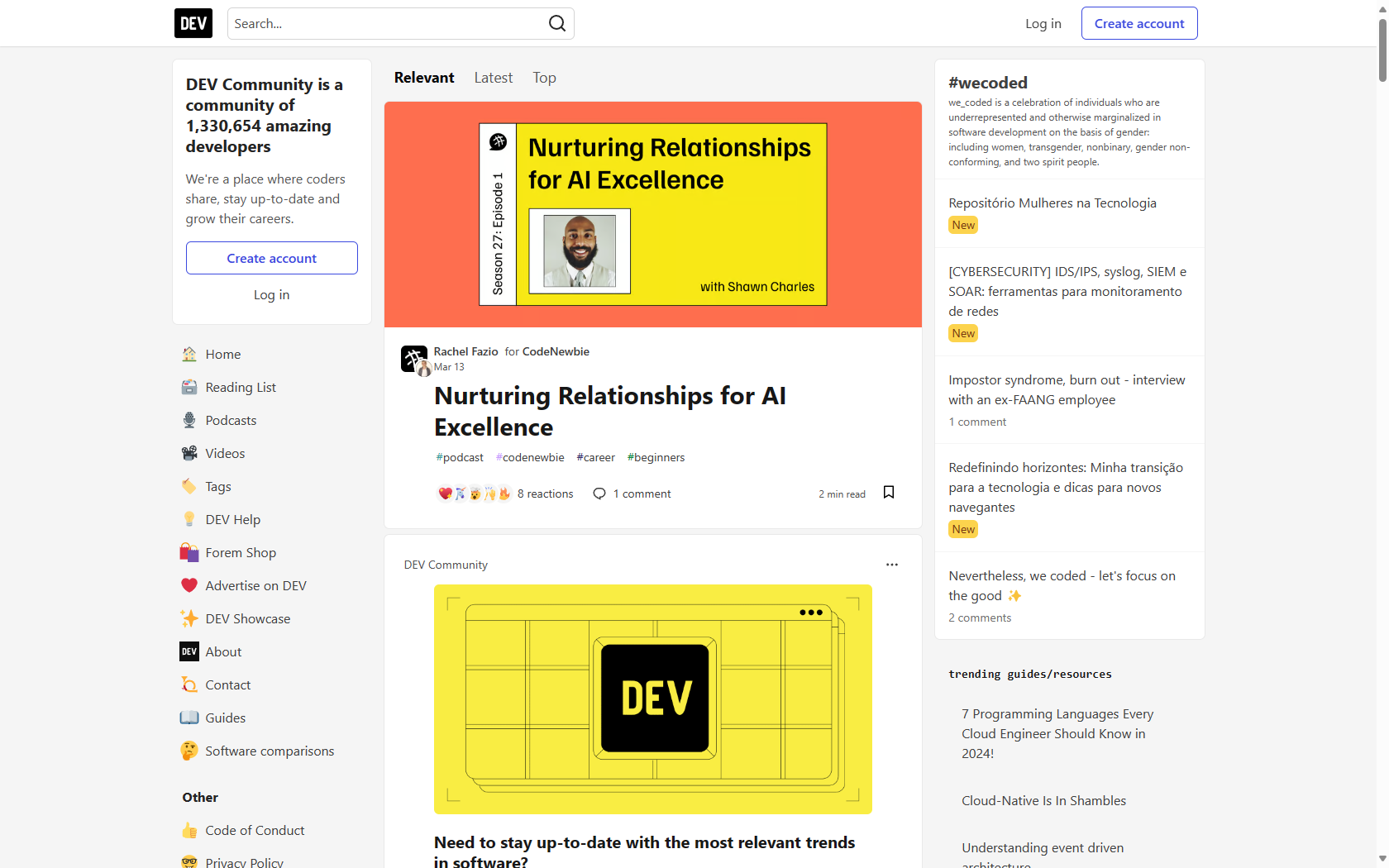


Рисунок 1.4 – Интерфейс [веб-приложения](http://unitoria.ru/blog/kadry/top-5-programm-dlya-kadrovogo-ucheta-v-2021-godu) [Dev.to](https://dev.to/t/webdev)

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.

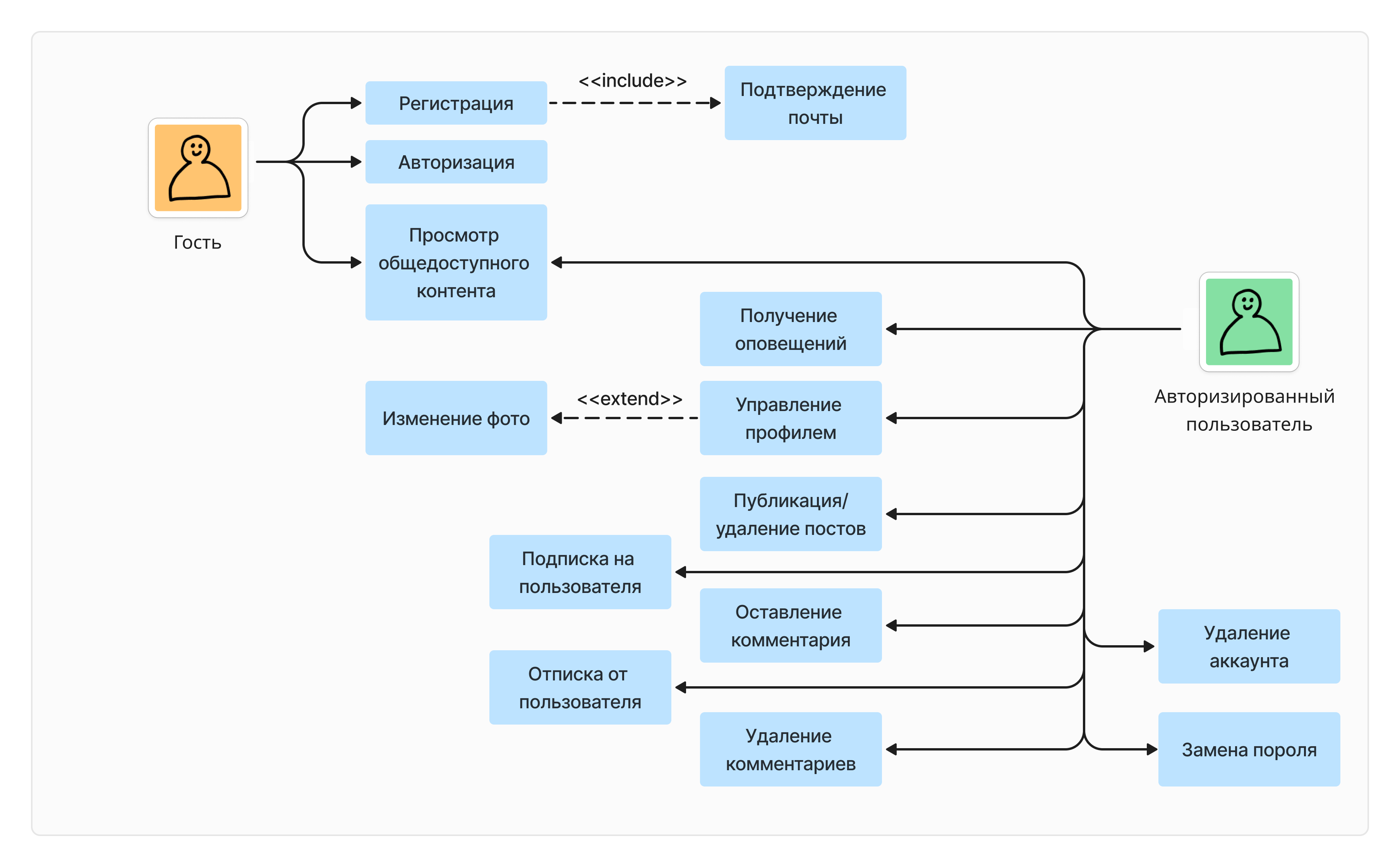


Рисунок 1.5 – Диаграмма вариантов использования

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 архитектуры 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.

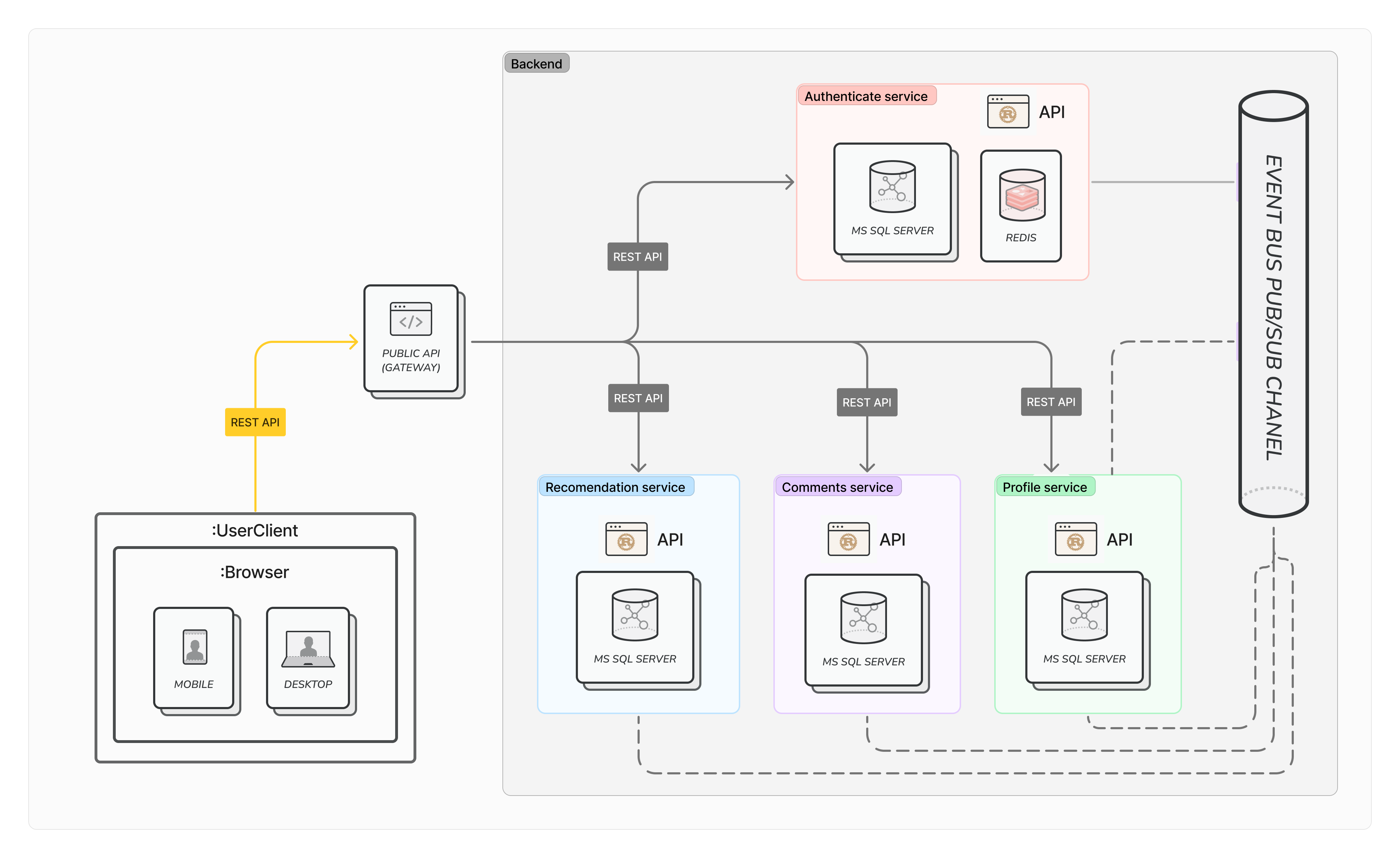


Рисунок 2.1 – Диаграмма размещения

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.

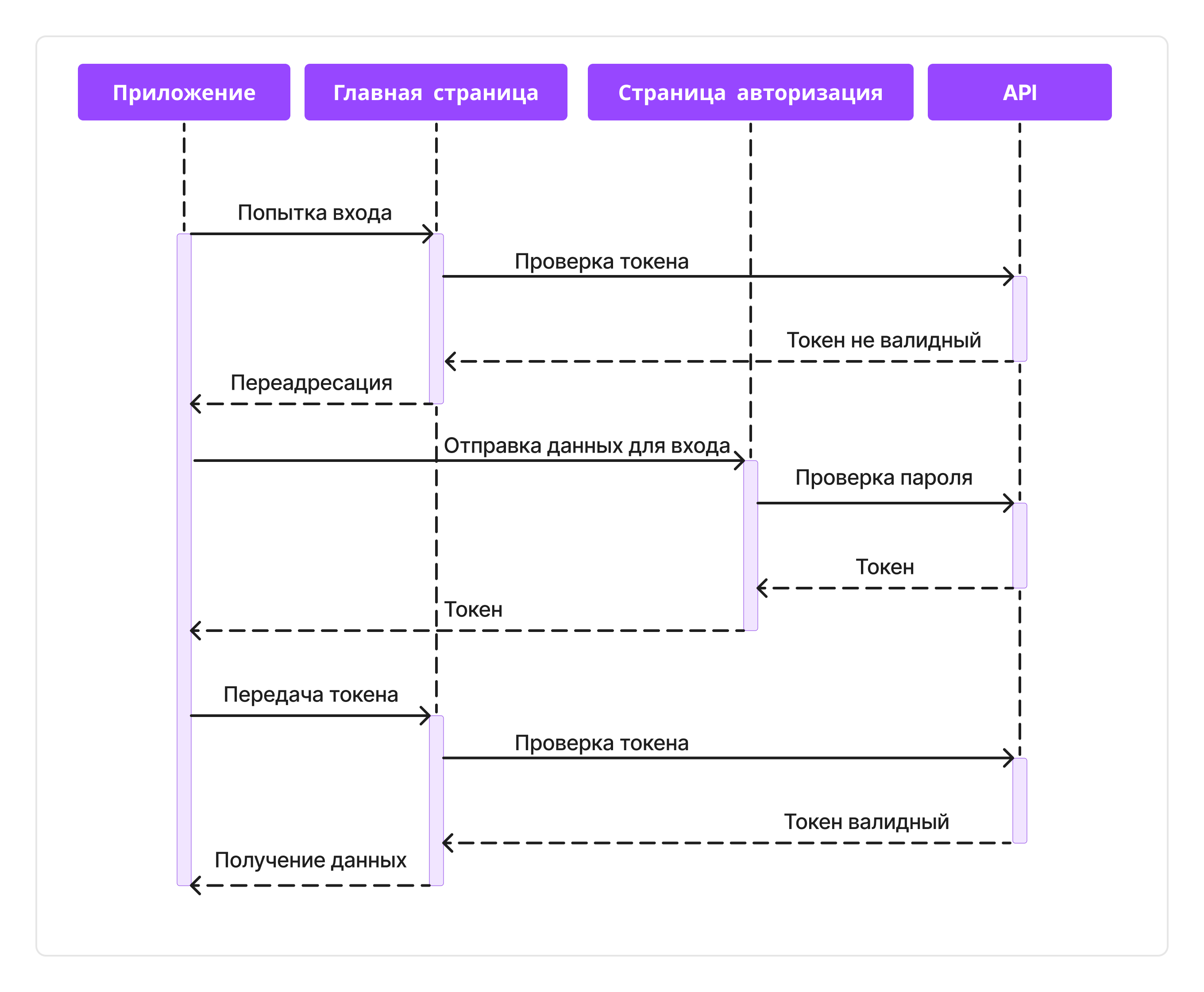


Рисунок 2.2 – Диаграмма последовательности

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.

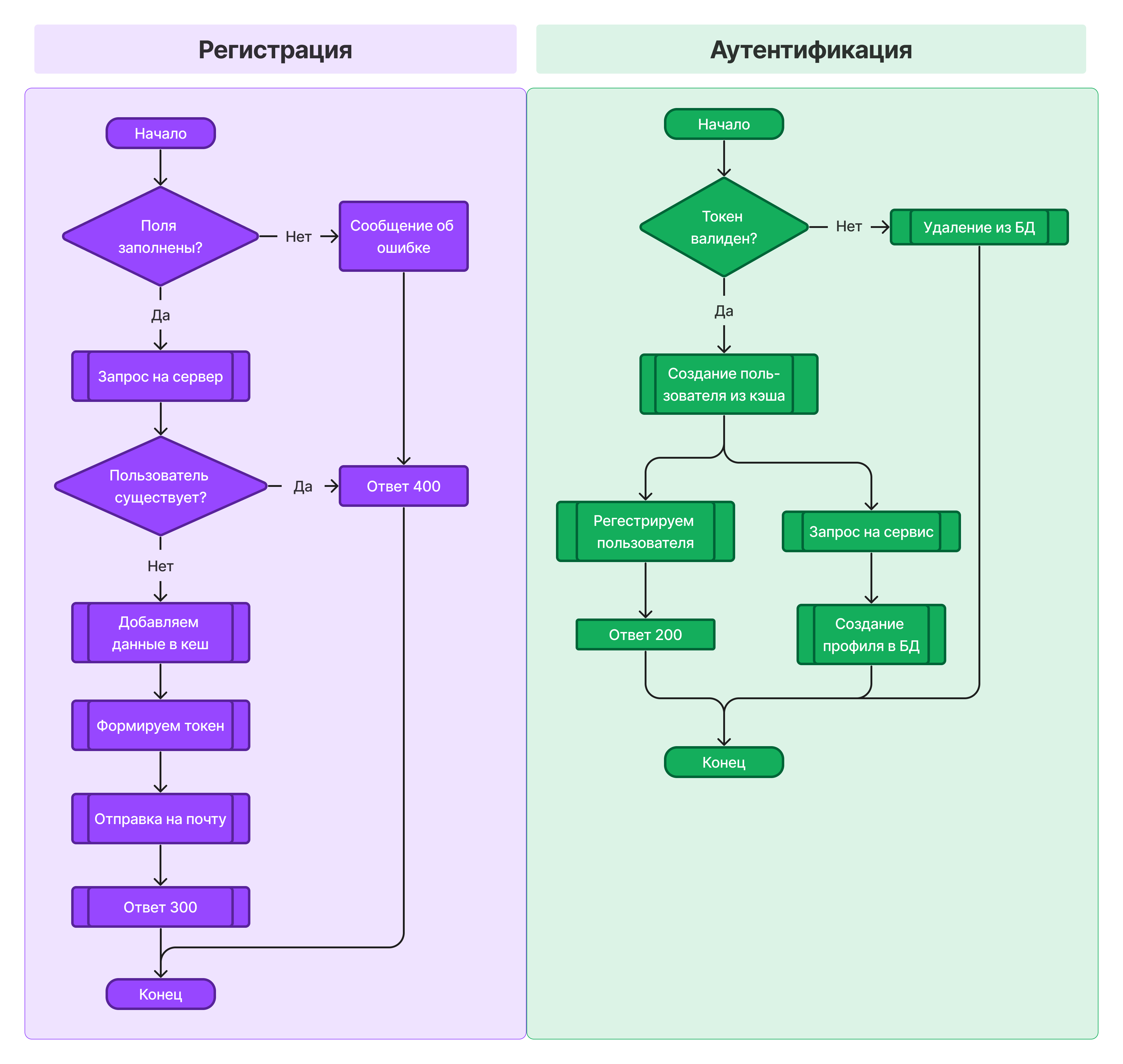


Рисунок 2.3 – Блок-схема авторизации

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.

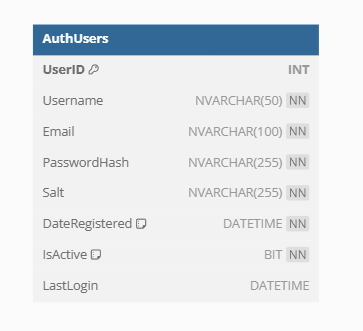


Рисунок 2.4 – Структура базы данных “ AUTHENTICATE\_SERVICE”

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.

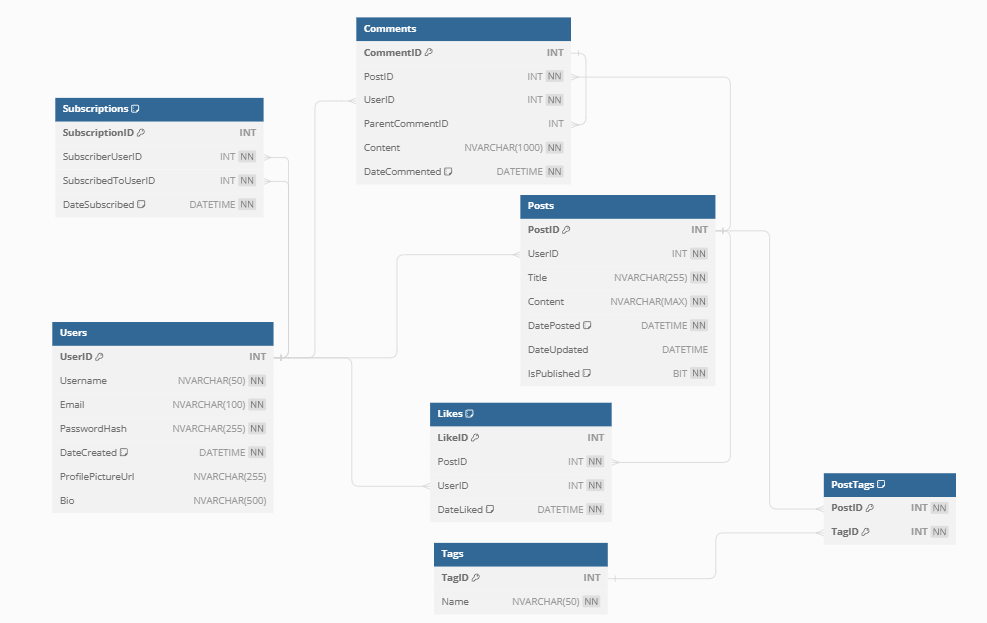


Рисунок 2.5 – Структура базы данных “PROFILE\_SERVICE”

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.

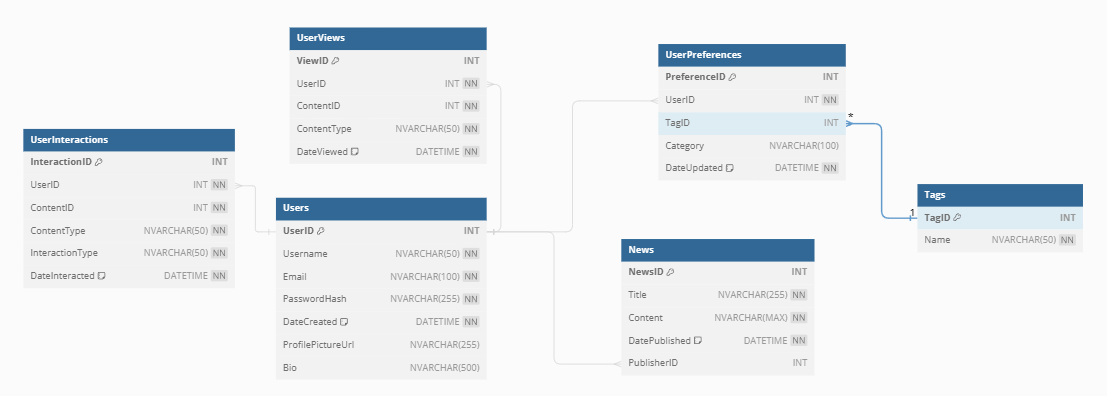


Рисунок 2.6 – Структура базы данных “RECOMMENDATION SERVICE”

1. 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.