Реализация WebSocket API и отложенной материализации ленты

Выполненные работы

1. Архитектурные изменения

- Реализовано отдельное WebSocket-приложение в собственном Docker-контейнере
- Настроено асинхронное взаимодействие через RabbitMQ между основным API и WebSocket-сервисом
- Реализована отложенная материализация ленты через очередь сообщений
- Обеспечена отправка уведомлений только целевым пользователям (друзьям автора поста)

2. Реализованная функциональность

- WebSocket API
- Канал /post/feed/posted для реального обновления ленты друзей
- Интеграция с существующей системой постов и друзей

3. Технические особенности

- Использование Direct Exchange в RabbitMQ для точной маршрутизации
- Поддержка множественных WebSocket-соединений для одного пользователя

Инструкция по запуску и тестированию

Предварительные требования

- Docker и Docker Compose
- Postman для тестирования API
- Веб-браузер с поддержкой WebSocket

Запуск системы

1. Клонируйте и запустите приложения:

```
bash

docker-compose up -d
```

Система запустит:

- postgres - база данных на порту 5432
- rabbitmq - брокер сообщений на портах 5672 и 15672
- sso - основное API на порту 5001
- websocket-service - WebSocket сервис на порту 8081

Тестирование функциональности

Коллекция Postman доступна по адресу https://www.postman.com/orange-satellite-666437/otus-highload/collection/8uv29vb/otus-highload

1. Используйте существующего пользователя:
    - В системе предустановлен пользователь с ID: b90940ae-ae81-4f6c-b4f2-6986d0b91d4c
2. Создайте нового пользователя: POST /user/register

Добавьте дружескую связь:

POST /friend/set?userId=b90940ae-ae81-4f6c-b4f2-6986d0b91d4c&friendId={ID_НОВОГО_ПОЛЬЗОВАТЕЛЯ}

3. Настройте WebSocket-клиент:
    - Откройте файл docs/queue/test.html в браузере
    - Обновите ID пользователя в адресе подключения:

```
// ЗАМЕНИТЕ ID на ID нового пользователя
const socket = new WebSocket('ws://localhost:8081/post/feed/posted?userId=ВАШ_НОВЫЙ_ID_ПОЛЬЗОВАТЕЛЯ');
```

4. Создайте тестовый пост: POST /post/create?userId=b90940ae-ae81-4f6c-b4f2-6986d0b91d4c

Ожидаемый результат

После выполнения запроса на создание поста:

1. В браузере вы должны увидеть всплывающее уведомление с содержимым поста
2. В консоли браузера отобразится полученное сообщение:

Особенности реализации

Масштабируемость

- WebSocket-сервис может масштабироваться независимо от основного API
- Использование статичных routing keys позволяет балансировать нагрузку
- Поддержка множественных соединений для одного пользователя

Обработка "эффекта Леди Гаги"

- Отложенная обработка через RabbitMQ предотвращает нагрузку при массовой подписке
- Асинхронная отправка уведомлений друзьям
- Балансировка нагрузки между экземплярами WebSocket-сервиса