**Отчет по теме: Очереди и отложенное выполнение**

1) Реализовать REST API:

Создание поста (метод /post/create из спецификации)

Используем существующий эндпойнт



2) Реализовать асинхронное API с websocket:

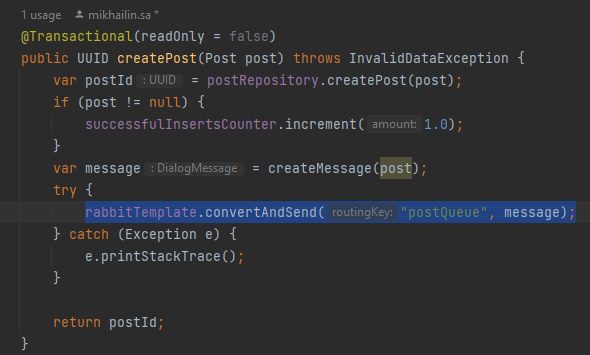
- добавляем эндпойнт **ws://localhost:8080/ws**, к которому подключается пользователь;

- при отправлении запроса по ссылке /app /post/feed/ пользователь получает из БД существующую ленты постов друзей. Лента отправляется в канал **/user/{user\_id}/post/feed/posted**.

Где {user\_id} – идентификационный номер подключенного к каналу пользователя.



* Параллельно при сохранении автором своего поста, мы отправляем этот пост в очередь RabbitMQ



- мы получаем сообщение из очереди и для всех друзей автора сохраняем пост в ленты и отправляем сообщение в канал **/user/{user\_id}/post/feed/posted**.



Использование {user\_id} в url позволяет обеспечить отправку только целевым пользователям это событие.

Тем самым достигается обновление ленты в реальном времени с отложенной материализацией.

3) Для масштабирования RabbitMQ необходимо развернуть дополнительные инстансы в докер-контейнерах и синхронизировать их.

Пример:

Задаем единый RABBITMQ\_ERLANG\_COOKIE и RABBITMQ\_USE\_LONGNAME=true

docker run -d --hostname node1.rabbit --net

clusternetwork --ip 192.168.0.10 --name

rabbitNode1 --addhost node2.rabbit:192.168.0.11

-p "15673:15672" -e "RABBITMQ\_USE\_LONGNAME=true"

-e RABBITMQ\_ERLANG\_COOKIE="cookie"

rabbitmq:3-management

После запуска нужно соединяем узлы в кластер:

docker exec rabbitNode2 rabbitmqctl stop\_app

docker exec rabbitNode2 rabbitmqctl join\_cluster

rabbit@node1.rabbit

docker exec rabbitNode2 rabbitmqctl start\_app

Включаем зеркалирование очередей

rabbitmqctl set\_policy ha-all ".\*" '{"ha-mode":"all","ha-sync-mode":"automatic"}'