

Для выполнения этой задачи были внесены изменения в основной сервис ([https://github.com/plohoicod/otus\\_highload\\_5.1](https://github.com/plohoicod/otus_highload_5.1)), была добавлена таблица диалогов, REST методы для создания сообщений и REST методы для межсервисного взаимодействия.

Перед отправкой сообщения создается диалог с уникальным id для двух пользователей:

WHERE			ORDER BY		
	id	user1_id	user2_id	user_least	user_greatest
1	1	1000987	1000989	1000987	1000989

Далее этот найди будет использоваться для шардирования сообщений в следующем микросервисе - [https://github.com/plohoicod/otus\\_highload\\_5.2](https://github.com/plohoicod/otus_highload_5.2).

Алгоритм шардирования:


1. Считаем остаток от деления id диалога на 64(число выбрано случайно, можно выбрать и другое, подходящее под конкретную задачу). Так же можно посчитать хеш функцию, и после этого от нее взять остаток от деления(если ключ текстовый или составной к примеру)
2. У каждого шарда есть диапазон значений, в один из которых попадает результат.
3. Записываем результат в соответствующий шард
4. Получаем результат так-же, через остаток от деления и выбор шарда по диапазону

Данный алгоритм не требует перемещения всех элементов между шардами, если изменяется кол-во шардов. В текущей реализации конфигурация двух шардов сделана в приложении, в дальнейшем ее следует перенести в базу данных.

Автоматический решардинг не предусмотрен, но можно последовать следующему алгоритму:

Пишем данные всегда в новые шарды. Читаем сначала из нового шарда; если там нет нужных данных, то обращаемся к старым шардам. Если данные оказываются в старом шарде, то при желании можно их переносить в новый — и в конце концов данные перераспределятся между шардами. Так же можно параллельно запустить процесс автоматический перенос данных, к тем к которым не идут запросы.

Создаем сообщение в диалоге:

 localhost:8081/dialog/send Save Share

POST

localhost:8081/dialog/send

Send

Params Authorization Headers (9) Body Scripts Tests Settings Cookies

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

☒ JSON

Beautify

```
1 {
2   ... "userFrom": 1000987,
3   ... "userTo": 1000989,
4   ... "message": "123"
5 }
```

Оно появилось в том же шарде, что и прошлое сообщение:

WHERE				ORDER BY		
	id	chat_id	user_from_id	user_to_id	message	create_datetime
1	3	1	1000989	1000987	123	2024-08-02 11:11:39.006946
2	4	1	1000987	1000989	123	2024-08-02 11:33:39.023552

Получаем сообщения диалога, они отсортированы по времени:

localhost:8081/dialog/list?userFrom=1000987&userTo=1000989

GETlocalhost:8081/dialog/list?userFrom=1000987&userTo=1000989

Send

ParamsAuthorizationHeaders (9)BodyScriptsTestsSettingsCookies

Query Params

<input checked="" type="checkbox"/>	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	userFrom	1000987			
<input checked="" type="checkbox"/>	userTo	1000989			
	Key	Value	Description		

BodyCookiesHeaders (5)Test Results

Status: 200 OKTime: 538 msSize: 273 BSave as example

PrettyRawPreviewVisualizeJSON

```
1 [
2   {
3     "userFrom": 1000989,
4     "userTo": 1000987,
5     "message": "123"
6   },
7   {
8     "userFrom": 1000987,
9     "userTo": 1000989,
10    "message": "123"
11  }
12 ]
```