САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэкенд разработка

Отчет

Лабораторная работа 6

Выполнил:

Фролова Кристина

K3439

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

- подключить и настроить rabbitMQ/kafka;
- реализовать межсервисное взаимодействие посредством rabbitMQ/kafka.

Ход работы

В качестве брокера я выбрала Apache Kafka. В моём проекте он будет выступать в роли передатчика сообщений об изменениях агрегата User, чтобы сервисы обновляли собственную копию данных таблицы known_users, тем самым уменьшая межсервисное синхронное взаимодействие, например, на операции верификации пользователя. На рисунке 1 отображены доменные события, на рисунке 2 - класс продюсера, который отправляет ивенты в топик. На рисунке 3 продемонстрирован класс консюмера, который слушает события в топике и на основе типа ивента принимает решение об операции в реплицирующей таблице.

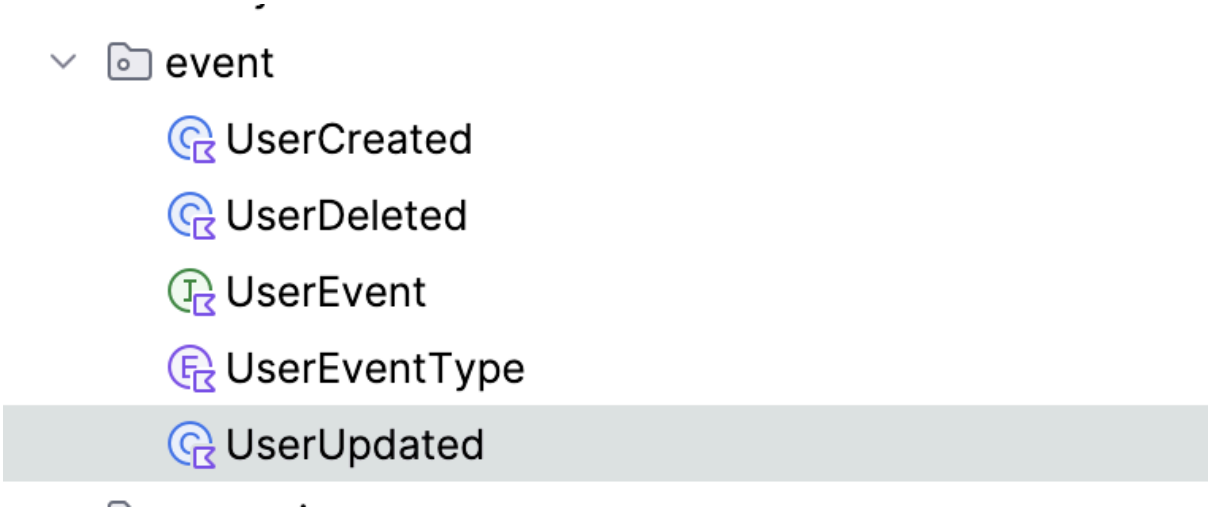
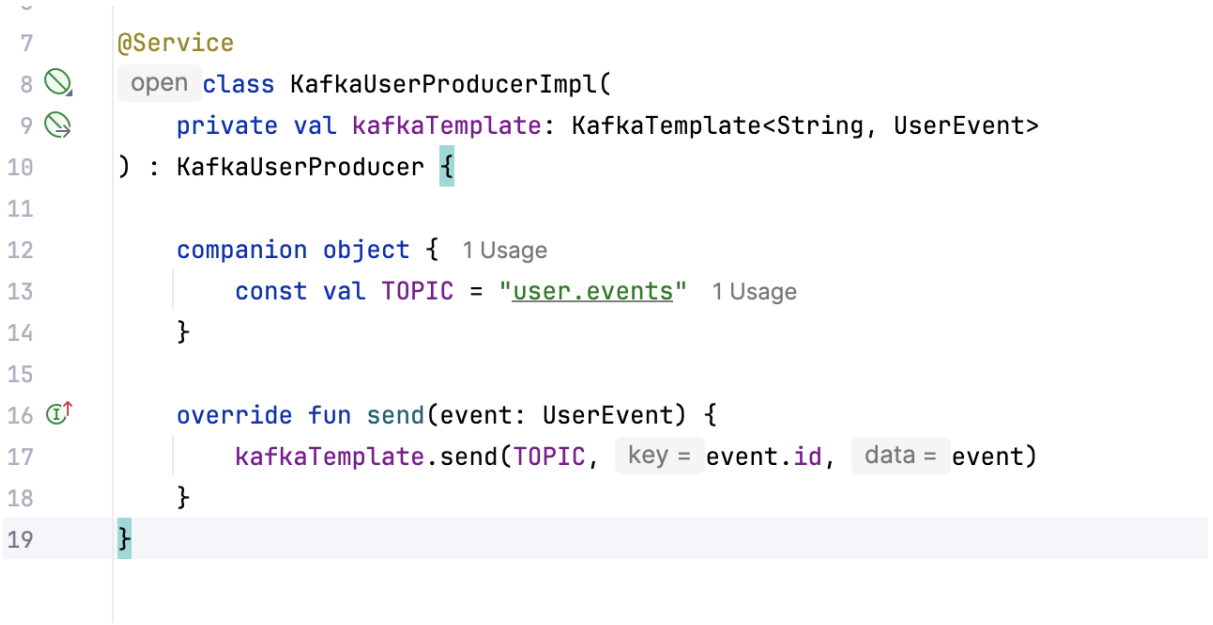


Рисунок 1 - Доменные события агрегата User



```
@Service
open class KafkaUserProducerImpl(
    private val kafkaTemplate: KafkaTemplate<String, UserEvent>
) : KafkaUserProducer {

    companion object {  1 Usage
        const val TOPIC = "user.events"  1 Usage
    }

    override fun send(event: UserEvent) {
        kafkaTemplate.send(TOPIC,  key = event.id,  data = event)
    }
}
```

Рисунок 2 - Класс продюсера

```
@Component
open class UserEventsConsumer(
    private val knownUsersService: KnownUsersService
) {

    @KafkaListener(
        topics = ["user.events"],
        groupId = "\${spring.kafka.consumer.group-id}"
    )
    open fun onUserEvent(event: UserEvent) {
        when (event.type) {
            UserEventType.CREATED -> knownUsersService.create(KnownUser( id = UUID.fromString( name = event.id), event.updatedAt))
            UserEventType.UPDATED -> knownUsersService.update(KnownUser( id = UUID.fromString( name = event.id), event.updatedAt))
            UserEventType.DELETED -> knownUsersService.delete( id = UUID.fromString( name = event.id))
        }
    }
}
```

Рисунок 3 - Класс консюмера

На рисунках 4 и 5 продемонстрирован пример работы очереди. При регистрации пользователя user-service отправляет сообщение в очередь, которая слушается chat-service. Получив ивент о создании пользователя, сервис создаёт во внутренней бд запись об этом пользователе. Таким

образом, если сервису понадобится верифицировать юзера, например, на существование. то ему не нужно будет обращаться к другому сервису.

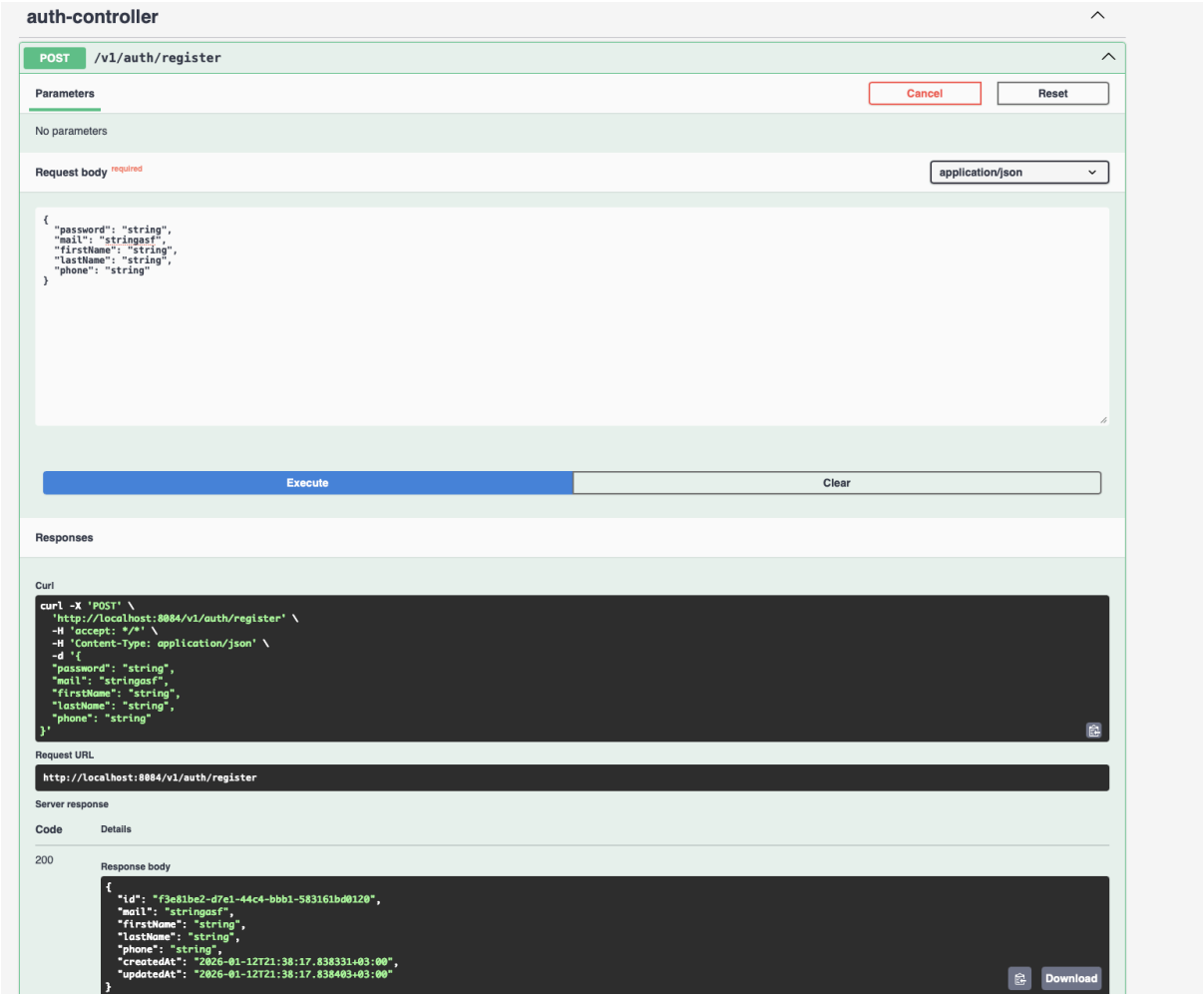


Рисунок 4 - Реализованный запрос в user-service на регистрацию

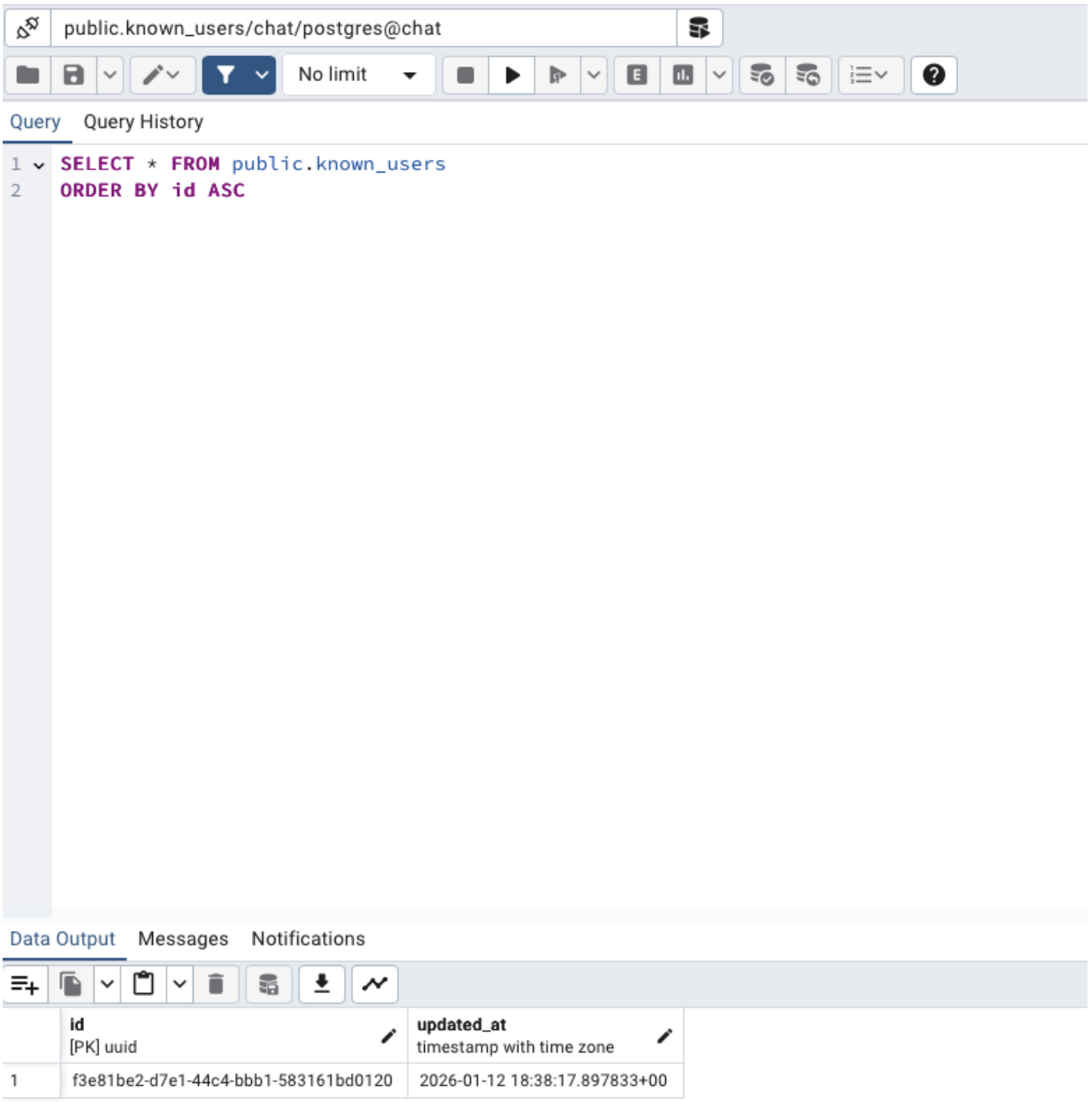


Рисунок 5 - known_users в chat-service

Вывод

В ходе работы была подключена и настроена очередь сообщений между сервисами.