

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бекенд разработка

Отчет

Лабораторная работа 6

Выполнил:

Фадеев Д.А.

К3439

**Проверил:
Добряков Д. И.**

Санкт-Петербург

2025 г.

Задача

- Подключить и настроить RabbitMQ/Kafka;
- Реализовать межсервисное взаимодействие по средствам

RabbitMQ/Kafka.

Ход Работы

На рисунке 1 book-service отправляет через очередь уведомление, что на конкретный content_id создалась метадата, поэтому поле has_metadata становится true (post), false (post).

```

22
21 export class RabbitMQService {
20   private static instance: RabbitMQService;
19   private connection: amqp.Connection | null = null;
18   private channel: amqp.Channel | null = null;
17
16   private readonly queueName: string;
15   private readonly rabbitMqUrl: string;
14
13   private constructor() {
12     this.queueName = process.env.RABBITMQ_QUEUE || 'content_metadata_update';
11     this.rabbitMqUrl = `amqp://${process.env.RABBITMQ_USER || 'guest'}:${{
10       process.env.RABBITMQ_PASS || 'guest'
9       }@${process.env.RABBITMQ_HOST || 'localhost'}:${{
8       process.env.RABBITMQ_PORT || '5672'
7       }}`;
6     }
5
4   public static getInstance(): RabbitMQService {
3     if (!RabbitMQService.instance) {
2       RabbitMQService.instance = new RabbitMQService();
1     }
29   return RabbitMQService.instance;
1   }
2
3   public async connect(): Promise<void> {
4     if (this.connection && this.channel) {
5       return;
6     }
7     try {
8       this.connection = (await amqp.connect(this.rabbitMqUrl)) as unknown as amqp.Connection;
9       this.channel = (await this.connection.createChannel()) as unknown as amqp.Channel;
10      await this.channel.assertQueue(this.queueName, { durable: true });
11
12      this.connection.on('error', (err) => {
13        console.error('RabbitMQ Connection Error:', err);
14        this.connection = null;
15        this.channel = null;
16      });
17      this.connection.on('close', () => {
18        console.warn('RabbitMQ Connection Closed.');
19        this.connection = null;
20        this.channel = null;
21      });
22      this.channel.on('error', (err) => {
23        console.error('RabbitMQ Channel Error:', err);
24        this.channel = null;
25      });
26      this.channel.on('close', () => {
27        console.warn('RabbitMQ Channel Closed.');

```

Рисунок 1 – Producer на стороне book-service
Content-service отвечает за получение url контента (книг в случае реализации только book-service) и поиску контента по тегам. Swagger UI по Content-service представлен на рисунке 2.

```

8 export class RabbitMQConsumer {
7   private connection: amqp.Connection | null = null;
6   private channel: amqp.Channel | null = null;
5
4   private readonly queueName: string;
3   private readonly rabbitMqUrl: string;
2   private contentService: ContentService;
1
17  constructor() {
1     this.queueName = process.env.RABBITMQ_QUEUE || 'content_metadata_update';
2     this.rabbitMqUrl = `amqp://${process.env.RABBITMQ_USER || 'guest'}:${{
3       process.env.RABBITMQ_PASS || 'guest'
4     }@${process.env.RABBITMQ_HOST || 'localhost'}:${{
5       process.env.RABBITMQ_PORT || '5672'
6     }}`;
7     this.contentService = new ContentService();
8   }
9
10  public async startConsuming(): Promise<void> {
11    if (this.connection && this.channel) {
12      return;
13    }
14    try {
15      this.connection = (await amqp.connect(this.rabbitMqUrl)) as unknown as amqp.Connection;
16      this.channel = (await this.connection.createChannel()) as unknown as amqp.Channel;
17      await this.channel.assertQueue(this.queueName, { durable: true });
18
19      console.log(
20        `Content-service waiting for messages in queue: ${this.queueName}`
21      );
22
23      this.channel.consume(
24        this.queueName,
25        async (msg: amqp.ConsumeMessage | null) => {
26          if (msg) {
27            try {
28              const message: ContentMetadataUpdateMessage = JSON.parse(
29                msg.content.toString()
30              );
31              console.log(
32                'Content-service received message:',
33                message
34              );
35
36              await this.contentService.updateContent(message.content_id, {
37                has_metadata: message.has_metadata,
38              });
39            }

```

Рисунок 2 – Consumer на стороне content-service

На RabbitMQ можно посмотреть дашборд, что и было продемонстрировано на рисунке 3.

Name	User name	State	SSL / TLS	Protocol	Channels	From client	To client
172.31.0.7:53408 ?	guest	running	○	AMQP 0-9-1	1	0 B/s	0 B/s
172.31.0.8:56364 ?	guest	running	○	AMQP 0-9-1	1	0 B/s	0 B/s

[+/-](#)

[HTTP API](#)
 [Documentation](#)
 [Tutorials](#)
 [New releases](#)
 [Commercial edition](#)
 [Commercial support](#)
 [Discussions](#)

Рисунок 3 – Дашборд RabbitMQ

На рисунке 4 можно посмотреть на логи content-service, который делает значения поля `has_metadata` true или false в зависимости от сообщений из очереди.

```
Content 3c82bf84-6c2a-46c2-88be-25d2d8fe9a78 has_metadata updated to true
Content-service received message: {
  content_id: '3c82bf84-6c2a-46c2-88be-25d2d8fe9a78',
  has_metadata: false
}
Content 3c82bf84-6c2a-46c2-88be-25d2d8fe9a78 has_metadata updated to false
Content-service received message: {
  content_id: '3c82bf84-6c2a-46c2-88be-25d2d8fe9a78',
  has_metadata: true
}
Content 3c82bf84-6c2a-46c2-88be-25d2d8fe9a78 has_metadata updated to true
```

Рисунок 4 – логи Docker

Вывод

В ходе лабораторной работы выполнено требуемое соединение микросервисов по средствам MQ.