САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Реализация брокера сообщений на базе готового приложения

Выполнил:

Петухов Семён

Группа
К3439

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Цель
Реализация межсервисного взаимодействия посредством очередей сообщений

Задание:

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

Ход работы

В ходе работы была доработана система путём внедрения брокера сообщений в resume-service

Был добавлен модуль работы с брокером сообщений RabbitMQ

messageQueue.ts

```
import amqp from "amqplib";

const QUEUE_NAME = "resume.created";
const RABBIT_URL = process.env.RABBITMQ_URL || "amqp://rabbitmq";

let channel: amqp.Channel | null = null;

async function getChannel(): Promise<amqp.Channel> {
  if (channel) {
    return channel;
  }

  const connection = await amqp.connect(RABBIT_URL);
  const ch = await connection.createChannel();
  await ch.assertQueue(QUEUE_NAME, { durable: true });
  channel = ch;

  connection.on("error", (err: unknown) => {
    console.error("[resume-service] RabbitMQ connection error:", err);
    channel = null;
  });

  connection.on("close", () => {
    console.warn("[resume-service] RabbitMQ connection closed, channel reset");
    channel = null;
  });

  return ch;
}
```

```
export async function publishResumeCreatedEvent(payload: unknown): Promise<void>
{
  try {
    const ch = await getChannel();
    const buffer = Buffer.from(JSON.stringify(payload));
    ch.sendToQueue(QUEUE_NAME, buffer, { persistent: true });
    console.log("[resume-service] Published resume.created event");
  } catch (err: unknown) {
    console.error("[resume-service] Failed to publish resume.created event:",
err);
  }
}
```

Данный модуль инкапсулирует подключение к RabbitMQ и публикацию событий

Далее данный брокер был интегрирован в контроллер resume

```
await resumeRepo.save(resume);

        // Асинхронно публикуем событие в RabbitMQ о создании резюме
        void publishResumeCreatedEvent({
            id: resume.id,
            userId: resume.userId,
            full_name: resume.full_name,
            created_at: new Date().toISOString(),
        });

        return res.status(201).json(resume);
```

При успешном сохранении мы публикуем сообщение в брокере
Сервис-потребитель в notification-service обрабатывает события из очереди и оповещает сервис при создании resume

```
const amqp = require("amqplib");

const QUEUE = "resume.created";
const RABBIT_URL = process.env.RABBITMQ_URL || "amqp://rabbitmq";

async function start() {
  try {
    console.log(`[notification-service] Connecting to RabbitMQ at
${RABBIT_URL}...`);
    const connection = await amqp.connect(RABBIT_URL);
    const channel = await connection.createChannel();

    await channel.assertQueue(QUEUE, { durable: true });
    console.log(`[notification-service] Waiting for messages in queue
"${QUEUE}"`);

    channel.consume(
```

```
      QUEUE,
      (msg) => {
        if (!msg) return;
        const content = msg.content.toString();
        try {
          const data = JSON.parse(content);
          console.log("[notification-service] Received resume.created event:",
data);
        } catch {
          console.log("[notification-service] Received raw message:", content);
        }
        channel.ack(msg);
      },
      { noAck: false }
    );
  } catch (err) {
    console.error("[notification-service] RabbitMQ connection error:", err);
    setTimeout(start, 5000);
  }
}

start();
```

Поток работы

1. Клиент создаёт резюме → POST /resume
2. resume-service сохраняет резюме в PostgreSQL
3. resume-service публикует событие в RabbitMQ (очередь resume.created)
4. HTTP-ответ возвращается клиенту
5. notification-service получает событие из очереди
6. notification-service обрабатывает событие (логирование)
7. notification-service подтверждает обработку (ack)

Выводы

В результате был реализован брокер сообщений, который отправляет сообщение при создании resume и обрабатывает его в notification-service