САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа 6

Выполнил:

Котовщиков Андрей

K3339

Проверил:
Добряков Д. И.

Санкт-Петербург

2022 г.

Задача

Реализовать межсерверное взаимодействие посредством очередей сообщений.

Ход работы

1. Подключение и настройка Kafka

В качестве брокера сообщений был выбран Apache Kafka. Для развертывания всей необходимой инфраструктуры был использован docker-compose (рисунок 1).

Рисунок 1 – Настройка kafka в docker-compose

2. Реализация межсервисного взаимодействия

2.1. Код потребителя (consumer)

В качестве потребителя (consumer) выступает микросервис для рассылки email писем, написанный на nodejs. Код представлен на рисунке 2. Сервис слушает kafka топик и читает сообщения, закодированные в JSON формат. Происходит деспирализация данных, из которых достается адрес получателя и код, который необходимо отправить на почту. Далее через smtp протокол происходит отправка письма с кодом.

```
const kafka = new Kafka({
    clientId: 'mail-sender',
    brokers: KAFKA_BROKERS.split(','),
});

const consumer = kafka.consumer({ groupId: 'mail-group' });

const transporter = nodemailer.createTransport({
    host: MAIL_HOST,
    port: Number(MAIL_PORT),
    secure: false,
    requireTLS: true,
    auth: {
        user: MAIL_USER,
        pass: MAIL_PASS,
    },
    tls: {
        minVersion: 'TLSv1.2',
    },
});

async function sendMail(to: string, code: string) {
    const info = await transporter.sendMail({
        from: `"${MAIL_FROM_NAME}" <${MAIL_USER}>`,
        to,
        subject: 'Ваш одноразовый код',
        text: `Ваш код: ${code}`,
        html: `<p>Ваш код: <b>${code}</b></p>`,
    });
     console.log(`Message sent: ${info.messageId}`);
}

async function run() {
    await consumer.connect();
    // @ts-ignore
    await consumer.subscribe({ topic: KAFKA_TOPIC, fromBeginning: false });
    console.log(`Subscribed to topic ${KAFKA_TOPIC}`);

    await consumer.run({
        eachMessage: async ({ message }) => {
            try {
                if (!message.value) return;
                const payload = JSON.parse(message.value.toString());
                const { email, code } = payload;
                console.log(`Received: ${email}, code=${code}`);
                await sendMail(email, code);
            } catch (err) {
                console.error('Error processing message', err);
            }
        },
    });
```

Рисунок 2 – Чтение сообщений из kafka

2.2. Код отправителя (producer)

В качестве отправителя данных (producer) выступает сервис с фильмами. При регистрации нового пользователя в системе генерируется специальный код. Затем в формате json кодируется сообщение с указанным при регистрации email адресом и сгенерированным кодом. В конце сообщение отправляется в топик kafka. Код отправки сообщения в брокер представлен на рисунке 3 и 4.

```
48  class ExternalMailSender(MailSender):
49      def __init__(self, broker: BrokerClient) -> None:
50          self._broker = broker
51
52      async def send_code(self, code: str, email: str) -> None:
53          broker_message = {
54              "email": email,
55              "code": code,
56          }
57
58          await self._broker.publish(
59              queue_name=config.MAILER_QUEUE_NAME,
60              message=json.dumps(broker_message),
61          )
```

Рисунок 3 – Логика отправки письма через брокер

```
34  class KafkaClient(BrokerClient):
35      def __init__(self, broker_url: str) -> None:
36          self._broker_url = broker_url
37          self._producer: Optional[AIOKafkaProducer] = None
38
39      async def connect(self) -> None:
40          self._producer = AIOKafkaProducer(bootstrap_servers=self._broker_url)
41          await self._producer.start()
42
43      async def publish(self, queue_name: str, message: str) -> None:
44          assert self._producer is not None, "Call connect before publish"
45          await self._producer.send_and_wait(topic=queue_name, value=message.encode())
46
47      async def close(self) -> None:
48          if self._producer is not None:
49              await self._producer.stop()
```

Рисунок 4 – Логика адаптера для взаимодействия с kafka

Вывод

В ходе выполнения домашней работы номер 5 была выстроена асинхронная коммуникация между микросервисами с помощью брокера сообщений kafka.