САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа 4

Выполнил:

Таякин Даниил

Группа К33392

Проверил:
Добряков Д. И.

Санкт-Петербург

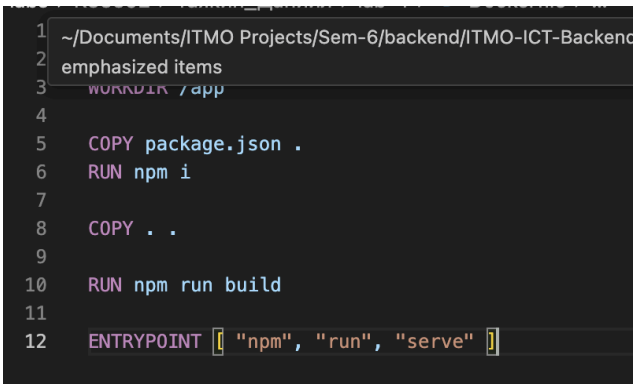
2024 г.

Задача

Необходимо упаковать ваше приложение в docker-контейнеры и обеспечить сетевое взаимодействие между различными частями вашего приложения, а также настроить общение микросервисов между собой посредством RabbitMQ.

Ход работы

1. Создадим Dockerfile для каждого сервиса.



2. Создадим сервис, который будет получать и обрабатывать сообщения из RabbitMQ. В данном случае, данный сервис будет записывать логи другого в базу данных.

```
channel.consume(
  queue,
  function (msg: Message | null) {
    if (!msg) return
    const log = JSON.parse(msg.content.toString()) as Log
    console.log(`Received: ${msg.content.toString()}`)
    db.run(
      `INSERT INTO logs (path, code, method) VALUES (?, ?, ?)`,
      [log.path, log.code, log.method],
      function (error: Error) {
        if (error) {
          console.log('Failed to save a log: %s', error)
        }
      }
    )
  },
  {
    noAck: true,
  }
)
```

3. Добавим middleware для основного сервиса, который будет отправлять сообщение в очередь после каждого запроса.

```
connection.createChannel(function (error1: Error, channel: Channel) {
  if (error1) {
    throw error1
  }

  const queue = 'logs_queue'
  var msg = JSON.stringify({
    code: res.statusCode,
    path: req.path,
    method: req.method,
  })

  channel.assertQueue(queue, {
    durable: false,
  })
  channel.sendToQueue(queue, Buffer.from(msg))

  console.log(' [x] Sent %s', msg)
})
```

4. Создадим compose.yaml со всей конфигурацией нашего backend.

```
version: "3.9"
name: backend

services:
  backend:
    build:
      context: ./lab-1
    environment:
      - "AUTH_URL=http://auth:9091"
      - "PORT=9090"
    ports:
      - "9090:9090"
    depends_on:
      - auth
      - rabbitmq

  auth:
    build:
      context: ./lab-3
    environment:
      - "PORT=9090"

  logs:
    build:
      context: ./lab-4
    depends_on:
      rabbitmq:
        condition:
          service_healthy

  gateway:
    image: caddy
    ports:
      - "80:80"
    volumes:
      - ./Caddyfile:/etc/caddy/Caddyfile
    depends_on:
      - auth
      - backend

  rabbitmq:
    image: "rabbitmq:3"
    healthcheck:
      test: rabbitmq-diagnostics -q ping
      interval: 5s
      timeout: 5s
      retries: 5
```

Вывод

В данной лабораторной работе удалось создать микросервис для работы с логами. Также все микросервисы были упакованы в docker контейнеры и запускаются при помощи docker-compose.