САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа 4

Выполнил:

Викторова Анастасия

M3220d

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

Для начала настроила один из запросов (создание книги). Сделала для него соединение серверов через rabbitmq. Север авторизации отправляет сообщение с телом запроса в очередь, сервер книг принимает его и создает книгу.

Функция отправки сообщения:

lab4 > auth > src > rabbit > TS producer.ts > default

```ts
import amqplib, {Connection, Channel} from 'amqplib'

let channel: Channel, connection: Connection

async function connectRabbit(data: any) {
    try {
        const amqpServer = 'amqp://guest:guest@localhost:5672'
        connection = await amqplib.connect(amqpServer)
        channel = await connection.createChannel()

        await channel.assertQueue('request')

        // отправка запроса в сервис book
        channel.sendToQueue(
            'request',
            Buffer.from(
                JSON.stringify({
                    ...data,
                    date: new Date(),
                }),
            ),
        )
    } catch (error) {
        console.log(error)
    }
}


export default connectRabbit
```

Прием сообщения:

```
lab4 > books > src > rabbit > TS consumer.ts > connectRabbit
import BookService from '../services/book/Book'

let channel: Channel, connection: Connection
const bookService = new BookService

async function connectRabbit() {
    try {
        const amqpServer = 'amqp://localhost:5672'
        connection = await amqplib.connect(amqpServer)
        channel = await connection.createChannel()

        await channel.assertQueue('request')

        await channel.consume('request', async(msg: Message | null) => {
            console.log(`Сервис книг получил информацию о запросе ${Buffer.from(msg!.content)}`)

            const { route, body } = JSON.parse(msg!.content.toString())

            switch (route) {
                case 'create':
                    await bookService.create(body)
                    break
                default:
                    break
              }

            channel.ack(msg!);
        })
    } catch (error) {
        console.log(error)
    }
}

export default connectRabbit
```

Результат:

POST http://localhost:3000/bookcrossing/books

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
    "title": "{{$randomAdjective}} {{$randomNoun}}",
    "ownerId": 2,
    "description": "{{$randomLoremText}}"
}
```

Body Cookies Headers (7) Test Results (2/2) 200 OK 177 ms 255 B Sa

Pretty Raw Preview Visualize JSON

```
{
    "status": "created"
}
```

```
Сервис книг получил информацию о запросе {"route":"create","body":{"title":"wireless interface","ownerId":2,"
description":"Qui rerum error dolorem sit doloremque architecto voluptatem odit aliquam. Quo aut accusamus. C
upiditate non provident voluptas sequi sit fugit. Doloremque ex veritatis. Fuga libero in quas et ut repellat
 id. Qui dolorem unde."},"date":"2024-05-29T10:42:30.669Z"}
Executing (default): INSERT INTO `Books` (`id`,`title`,`ownerId`,`description`,`createdAt`,`updatedAt`) VALUE
S (NULL,$1,$2,$3,$4,$5);
Executing (default): SELECT `id`, `title`, `ownerId`, `description`, `createdAt`, `updatedAt` FROM `Books` AS
 `Book`;
```

Далее для каждого сервера создала файл Docker

lab4 > books > Dockerfile > ...

```
FROM node

COPY . /book_server

WORKDIR /book_server

EXPOSE 8001

RUN npm install

CMD [ "npm", "start" ]
```

А также общий файл docker-compose.yml

lab4 > docker-compose.yml

```
version: '3.8'

services:
  rabbitmq:
    container_name: rabbitmq
    image: rabbitmq
    restart: always

  auth:
    container_name: auth
    build: ./auth
    restart: always
    depends_on:
      - rabbitmq
    ports:
      - 8000:8000

  book:
    container_name: book
    build:
      context: ./books
    depends_on:
      - rabbitmq
    ports:
      - 8001:8001
    restart: always
```

Вывод

В ходе работы был изучен rabbitmq и приобретены навыки использования Docker