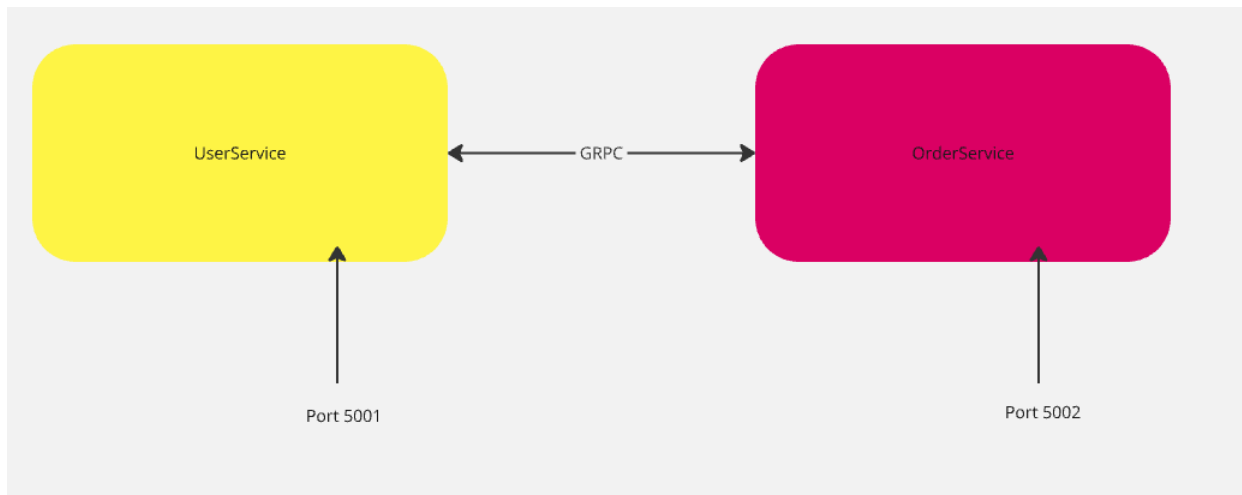


Разработка микросервисной архитектуры с использованием gRPC

Целью работы была разработка и развёртывание микросервисной архитектуры на основе Java, gRPC и Docker. Два микросервиса — UserService (управление пользователями) и OrderService (управление заказами) — были выделены в отдельные модули. Взаимодействие между ними происходит по gRPC-протоколу. Для удобного развёртывания использовались Docker и Docker Compose.



- **UserService:** Управляет информацией о пользователях (CreateUser, GetUser).
- **OrderService:** Создаёт заказы и при необходимости может обращаться к UserService за информацией о пользователе.

Основные функции:

- UserService: CreateUser, GetUser.
- OrderService: CreateOrder, (опционально GetOrder).

В файлах .proto определены сервисы и сообщения:

UserService.proto:

```
1  syntax = "proto3";
2
3  package com.example.user;
4
5  option java_multiple_files = true;
6  option java_package = "com.example.userservice.proto";
7  option java_outer_classname = "UserServiceProto";
8
9  service UserService {
10     rpc CreateUser(CreateUserRequest) returns (UserResponse);
11     rpc GetUser(GetUserRequest) returns (UserResponse);
12 }
13
14 // Запрос на создание пользователя
15 message CreateUserRequest {
16     string name = 1;
17     string email = 2;
18 }
19
20 // Ответ с данными пользователя
21 message UserResponse {
22     string user_id = 1;
23     string name = 2;
24     string email = 3;
25 }
26
27 // Запрос на получение данных пользователя
28 message GetUserRequest {
29     string user_id = 1;
30 }
```

OrderService.proto:

```
syntax = "proto3";

package com.example.order;

option java_multiple_files = true;
option java_package = "com.example.orderservice.proto";
option java_outer_classname = "OrderServiceProto";

service OrderService {
    rpc CreateOrder(CreateOrderRequest) returns (OrderResponse);
    rpc GetOrder(GetOrderRequest) returns (OrderResponse);
}

// Запрос на создание пользователя
message CreateOrderRequest {
    string user_id = 1;
    double amount = 2;
}

// Ответ с данными пользователя
message OrderResponse {
    string order_id = 1;
    string user_id = 2;
    double amount = 3;
}

// Запрос на получение данных пользователя
message GetOrderRequest {
    string order_id = 1;
}
```

Для каждого сервиса написан Dockerfile

```
# Используем Maven для сборки
1 FROM maven:3.9.4-eclipse-temurin-21 AS build
2 WORKDIR /app
3
4 # Копируем весь проект
5 COPY . .
6
7 # Переходим в папку order_service и собираем только этот модуль
8 WORKDIR /app/order_service
9 RUN mvn clean package -DskipTests
10
11 # Создаём лёгкий образ для запуска
12 FROM eclipse-temurin:21-jre
13 WORKDIR /app
14 COPY --from=build /app/order_service/target/order_service-1.0-SNAPSHOT.jar order_service.jar
15 EXPOSE 5002
16 ENTRYPOINT ["java", "-jar", "order_service.jar"]

# Используем Maven для сборки
1 FROM maven:3.9.4-eclipse-temurin-21 AS build
2 WORKDIR /app
3
4 # Копируем весь проект
5 COPY . .
6
7 # Переходим в папку user_service и собираем только этот модуль
8 WORKDIR /app/user_service
9 RUN mvn clean package -DskipTests
10
11 # Создаём лёгкий образ для запуска
12 FROM eclipse-temurin:21-jre
13 WORKDIR /app
14 COPY --from=build /app/user_service/target/user_service-1.0-SNAPSHOT.jar user_service.jar
15 EXPOSE 5001
16 ENTRYPOINT ["java", "-jar", "user_service.jar"]
```

docker-compose.yml:

```
version: '3.8'
1 services:
2   user_service:
3     build:
4       context: .
5       dockerfile: user_service/Dockerfile
6     ports:
7       - "5001:5001"
8
9   order_service:
10    build:
11      context: .
12      dockerfile: order_service/Dockerfile
13    ports:
14      - "5002:5002"
```

Тестирование через Postman (gRPC):

localhost:5001

UserService / CreateUser

Message

Authorization

Metadata

Service definition

Scripts

Settings

1 {

2 |

3 | "name": "Thomas",

4 | "email": "def@mail.ru"

4 }

Use Example Message

Response

Metadata (1)

Trailers

Test results

1 {

2 |

3 | "user_id": "4eafed85-706e-4348-b026-5d2e4daae696",

4 | "name": "Thomas",

4 | "email": "def@mail.ru"

5 }

localhost:5001

UserService / GetUser

Message

Authorization

Metadata

Service definition

Scripts

Settings

1 {

2 |

3 | "user_id": "123"

3 }

Use Example Message

Response

Metadata (1)

Trailers

Test results

1 {

2 |

3 | "user_id": "123",

4 | "name": "Test User",

4 | "email": "test@example.com"

5 }

localhost:5002

OrderService / GetOrder

Message

Authorization

Metadata

Service definition

Scripts

Settings

1 {

2 |

3 | "order_id": "132"

3 }

Use Example Message

Response

Metadata (1)

Trailers

Test results

1 {

2 |

3 | "order_id": "132",

4 | "user_id": "dummy_user_id",

4 | "amount": 99.99

5 }

