Universidade Federal de Pernambuco – UFPE

Centro de Informática – CIn

Microserviços – IF1007

Relatório de Projeto

Recife – 2018.1

Microerviços– IF1007

Especificação de Projeto

João Victor Oliveira Santos (jvos),

Rielson Leandro Silva de Lima (rlsl),

Rodrigo de Lima Oliveira (rlo).

Sumário

1. Introdução

Este projeto tem como objetivo fixar o conteúdo visto em sala, e dar um visão prática a respeito do desenvolvimento de aplicações que utilizam o conceito de microserviços. O projeto foi pensado e desenvolvido desde o início para utilizar os conceitos aprendidos, tais como fácil manutenção, escalabilidade, robustez, flexibilidade em lidar com diferentes implementações.

Como tema central, foi utilizado um sistema de delivery (alimentos), que conta com um sistema de cadastro, pedidos e pagamento por cartão de crédito ou dinheiro.

2. Descrição da Arquitetura

As tecnologias utilizadas foram SpringBoot, RabbitMQ, Eureka e Hystrix.

O Spring Boot é um projeto da Spring que veio para facilitar o processo de configuração e publicação de aplicações. A intenção é ter o projeto rodando o mais rápido possível e sem complicação.

A ideia do RabbitMQ é disponibilizar uma estrutura que facilite fluxos de mensagens, sobretudo em grandes aplicações, para a comunicação entre todos os processos.

O projeto Eureka, atua como um service registry, ele conhece todos os serviços e sabe o status de cada um (iniciando, em execução, fora do ar, etc.) e em que máquina, zona, região e IP que estão sendo executados.

O Hystrix implementa o padrão Circuit Breaker, que de forma bem rápida é um failover para chamadas entre micro serviços, ou seja, caso um micro serviço estiver fora do ar um método de fallback é chamado e aquela enxurrada de falhas é evitada.

A arquitetura se divide em basicamente em 3 microserviços principais. Cada um deles será apresentado nas secções seguintes.

2.1. DeliveryApp (config)

Tem como função configurar todos o sistema. Ele irá fornecer todas as configurações utilizadas pelos outros microserviços. Está conectado a todos.

2.2. Order-Service

Micro serviço responsável pela gestão de pedidos do App. Vale salientar, que pensando em escalabilidade e robustez, esse serviço ficou apenas responsável por pedidos.

A edição ou cancelamentos de pedidos, foi separado em um microserviço menor, que é responsável apenas pelas alterações realizadas nos pedidos. Assim, evita-se a sobrecarga no sistema de pedidos em picos.

2.3. Payments-Service

Serviço responsável pelos pagamentos realizados no App. Na mesma linha de raciocínio do Order-service, esse serviço conta com um serviço auxiliar que faz a distinção entre pagamento a vista e pagamento em cartão.

3. Conclusão

Ao utilizar o conceito de microserviços e ferramentas como o SpringBoot, Docker, Eureka, já citados anteriormente, o desenvolvimento de uma aplicação complexa se torna muito mais intuitivo. Essas ferramentas trazem robustez e facilidades que um sistema monolítico, onde todos os serviços rodam em um mesmo servidor/ambiente não possuem.

Além claro da facilidade em expandir a aplicação, implementando novas funcionalidades, sem que seja necessário modificar a lógica da aplicação em em produção, ou interromper o serviço.