Arquitetura de micro-serviços

Que tipo de problema essa arquitetura resolve?

Aplica solução para problemas de sobrecarga de requisições em um recurso específico, o que pode derrubar a aplicação ou trazer outros problemas de acesso ao serviço. Quando uma aplicação atinge milhares de usuarios os recursos para atender a toda essa demanda pode possuir gargalos.

Algumas soluções:

Dificuldade de Escalabilidade

Cada microserviço pode ser escalado individualmente, de acordo com a demanda de seus recursos específicos. Isso permite uma escalabilidade mais eficiente e orientada às necessidades, reduzindo custos e melhorando a performance.

Complexidade no Gerenciamento e Desenvolvimento

Os microservices dividem a aplicação em componentes menores e independentes, facilitando o desenvolvimento e a manutenção de cada serviço. Cada equipe pode ser responsável por um ou mais microservices, promovendo a autonomia e a agilidade no desenvolvimento.

Falta de Flexibilidade Tecnológica

Cada microserviço pode ser desenvolvido com a tecnologia mais adequada para sua função específica. Isso permite a utilização de diferentes linguagens de programação, bancos de dados, e frameworks para diferentes serviços, proporcionando mais flexibilidade.

Ciclos de Deploy e Atualização Lentos

Cada microserviço pode ser atualizado e implantado independentemente dos outros. Isso permite ciclos de deploy mais rápidos e menos arriscados, com impacto isolado em partes específicas do sistema.

Resiliência e Tolerância a Falhas

Microservices são desacoplados, o que significa que se um serviço falhar, os outros podem continuar funcionando. A arquitetura permite a implementação de estratégias de resiliência, como fallback, retries, circuit breakers, etc., para melhorar a disponibilidade e reduzir o impacto de falhas.

Dificuldade de Adaptação a Novas Funcionalidades

Novas funcionalidades podem ser desenvolvidas como novos microservices ou adicionadas a serviços existentes, com mínimo impacto em outras partes do sistema. Isso permite uma evolução mais fácil e rápida do sistema.

Confiabilidade de Equipes Grandes

Cada equipe pode ser responsável por um conjunto de microservices, o que promove a divisão de responsabilidades, permitindo que as equipes trabalhem de maneira mais independente, aumentando a produtividade e reduzindo conflitos.

Manutenção e Suporte a Código Legado

Microservices permitem que partes legadas do sistema sejam gradualmente substituídas por novos serviços, reduzindo a dependência de grandes blocos de código legado e facilitando a modernização contínua.

Performance e Latência

Cada microserviço pode ser otimizado e executado separadamente. Serviços mais críticos podem ter mais recursos ou melhores configurações, melhorando a performance global do sistema.

Integração com Sistemas Externos

Os microservices podem ser projetados para se comunicar facilmente com sistemas externos, e serviços específicos podem ser dedicados a gerenciar essas integrações, tornando-as mais simples de implementar e manter.

Gateway (Spring cloud gateway) faz a ponte entre as requisições e o acesso aos recursos do servidor. Nele fica a rota de acesso para os clientes.

Discovery Server (Eureka) gerencia as instancias de serviços.

Blanceamento de carga (LoadBalancer), distribui as requisições de forma equilibrada entre as instancias de serviços registradas no discovery server.

Diagrama

Descrição gerada automaticamente

[Documentação do Eureka](https://cloud.spring.io/spring-cloud-netflix/multi/multi_spring-cloud-eureka-server.html)

[Documentação Cloud Gateway](https://spring.io/projects/spring-cloud-gateway)