Arquitetura microserviços voltada para escalabilidade

Matheus Augusto Picioli, Henrique Dezani, Carlos Magnus Carlson Filho

e-mail: matheus.picioli@fatec.sp.gov.br; henrique.dezani@fatec.sp.gov.br;
carlos.carlson@fatec.sp.gov.br

Resumo

Este artigo explica a escalabilidade de aplicação ao utilizar a arquitetura de microserviços. Explica conceitos de escalabilidade horizontal e vertical, bem como suas aplicações exemplificadas. A partir destes conceitos, explica cenários de aplicação em que essa arquitetura se encaixa, com foco em aumentar o entendimento sobre a arquitetura, e auxilia na decisão de uso com explicações de como pode ser utilizada, e quais seus benefícios.

Palavras-chave: Microserviços. Arquitetura de aplicações. Escalabilidade. DevOps. Padrões de projeto.

Abstract

This article explains scalability applications using microservice architecture. Understand concepts of horizontal scalability and vertical scalability, as well your applications with examples. Starting of this concepts, understanding applicable scenarios with this architecture matches, increasing understanding of then, assisting on use decision with examples of how use architecture and your benefits.

Keywords: *Microservice. Software architecture. Scalability. DevOps. Design patterns.*

Introdução

Com o crescimento da tecnologia e, consequentemente o uso de sistemas, redes sociais, serviços de *streaming* e aplicativos em geral, os usuários se tornaram mais exigentes com performance. Usuários da Amazon, por exemplo, não querem perder uma promoção de tempo limitado, porque o servidor está sobrecarregado e não consegue processar sua compra dentro

do prazo da promoção. Então foi identificado um problema de escalabilidade na arquitetura monolito.

Essa arquitetura por sua vez, consiste em ter um único código fonte para a aplicação com todas as funcionalidades, o que gera um código extenso, complexo e pesado.

Em contrapartida, a utilização de uma arquitetura em que uma parte da aplicação tem autonomia total sobre ela mesma sem a necessidade de espera de processamento por outra parte do sistema, torna-se muito mais fácil aumentar sua escala por conseguir lidar com essas partes de forma individual e isolada.

Com um sistema que possui suas funcionalidades de forma isolada e com autonomia total própria, então a infraestrutura para esse serviço é muito mais controlada, pois também se torna individual e tem autonomia própria.

Justificativa

A escalabilidade que muitas empresas fazem hoje é a escalabilidade vertical, que consiste em aumentar o seu único servidor cada vez mais. Mas isso traz um problema relevante para a aplicação, se um bug ocorre em determinada parte do sistema, esse *bug* pode interromper a aplicação inteira, por estar em um único servidor.

Também pode gerar um custo elevado sem necessidade. Na aplicação de um banco, a funcionalidade de geração de boleto será muito mais custosa para o servidor do que a visualização do boleto pelo cliente. Como essas funcionalidades compõem uma aplicação de sistema monolito elas estão no mesmo servidor, que escala verticalmente, sem a possibilidade de ter a funcionalidade de visualização do boleto fazendo com que concorram por recursos, e torne a funcionalidade lenta.

Objetivos

O objetivo deste artigo é explicar quais problemas a arquitetura soluciona e seus pontos chaves, para determinar quais cenários faz-se um bom uso dessa arquitetura. Visto que sua complexidade é alta, há cenários em que o uso da arquitetura não se faz valer.

Fundamentação Teórica

Uma das primeiras empresas a sentir necessidade do conceito de serviços, foi a Amazon em 2002 quando perceberam que seu sistema monolito estava muito grande, e de difícil manutenção. Então começaram uma migração para o conceito de *loosely coupled services* (serviço fracamente acoplados), que eles criaram. *Loosely coupled services* são pequenas partes da aplicação acopladas e que podiam ser perdidas e restauradas a qualquer momento que afeta apenas os serviços acoplados em questão, e mantém todo o restante em funcionamento.

A nova arquitetura fazia uso do conceito de escalabilidade horizontal, pois era composta de pequenos códigos fontes que atuavam de forma mais isolada.

Essa migração de arquitetura começou a ter reflexo na facilidade em dar manutenção e criar esses serviços, em comparação ao monolito que tinham até então.

Com os *loosely coupled services*, como chamavam seus primeiros serviços, a Amazon percebeu um aumento na velocidade de entrega de *features* (funcionalidades). Isso se deve ao fato que, códigos isolados de serviços, são mais curtos, e tendem a ser mais simples.

Trabalhos Similares

Amaral e Carvalho (2017) fazem uma comparação de desempenho entre uma aplicação com arquitetura monolítico e arquitetura em microserviço. Este é um exemplo de um lado negativo da arquitetura microserviço, como depende de uma série de fatores para se fazer funcional, o processamento nesta arquitetura tende a ser mais custoso.

Ribeiro (2017) traz uma comparação visual entre as duas arquiteturas e como elas estão conectadas ao seus bancos de dados, detalha a parte técnica e explica com detalhes o que deve ser feito em cada; além de abordar a questão principal deste artigo: escalabilidade.

Desenvolvimento

Foi realizado um estudo dos conceitos atuais da arquitetura de microserviços para seu entendimento conceitual sólido. Com base na sua origem e necessidade de seu conceito até a

atualidade, com entendimento das mudanças e tomadas de decisões feitas até então, a fim de garantir sua aplicabilidade original. Com entendimento da arquitetura e cenários que ela atende, foi comparado o porquê de não ser amplamente utilizada.

A arquitetura de microserviços atende o problema exemplificado na justificativa, pois suas funcionalidades podem ser transformadas em serviços e passam a ter autonomia para funcionarem de forma isolada. Logo pode-se ter um servidor mais modesto que irá conter apenas o serviço de visualização do boleto pelo cliente, e um servidor intermediário que irá conter o serviço de geração de boleto.

Além dos custos desses servidores serem mais baixos se somados e comparados ao servidor mais robusto, o impacto de um *bug* na geração de boleto não impede o usuário de consultar um boleto já gerado anteriormente.

Mas não há somente pontos positivos, visto que a arquitetura de microserviços ganha facilidade para sua escala, seu ponto negativo é a complexidade. Ao utilizar microserviços é necessário manter dados sincronizados em diversos banco de dados, e isso não é uma tarefa simples. Além de ferramentas que antes o monolito não exigia, como um *message broker*[1], por exemplo.

São diversos fatores que implicam em sua construção e manutenibilidade, que reforçam o ponto de que seu uso se faz essencial em situações de escala para performance, visto toda a complexidade que está agregada.

Um exemplo de empresa que precisa de escalabilidade e controle individual de seus serviços sem um afetar ao outro, é a Netflix[2].

Dada a necessidade do uso de uma arquitetura que atendesse mais especificamente a escalabilidade foi encontrado também a arquitetura SOA (*service-oriented architeture*), que define uma forma de criar componentes por interfaces padrões. Essas interfaces possuem comunicações padrões que podem ser facilmente incorporadas em novas aplicações a fim de evitar que o novo software precise de muito conhecimento das responsabilidades desse serviço a cada nova integração. Essa arquitetura foi utilizada pela Uber com o foco na facilidade de integração desses serviços em novos códigos com foco na dinâmica da empresa.

[1] Mensageria é um conceito que define que sistemas distribuídos, possam se comunicar por meio de troca de mensagens (evento), sendo estas mensagens "gerenciadas" por um Message Broker (servidor/módulo de mensagens), Brito (2019).
[2] A Netflix utiliza vastamente essa arquitetura para escalabilidade e também pra DevOps https://netflixtechblog.com/how-we-build-code-at-netflix-c5d9bd727f15.

Nas figuras 1 e 2 abaixo é mostrada a comparação visual entre as duas arquiteturas, monolito e microserviço:

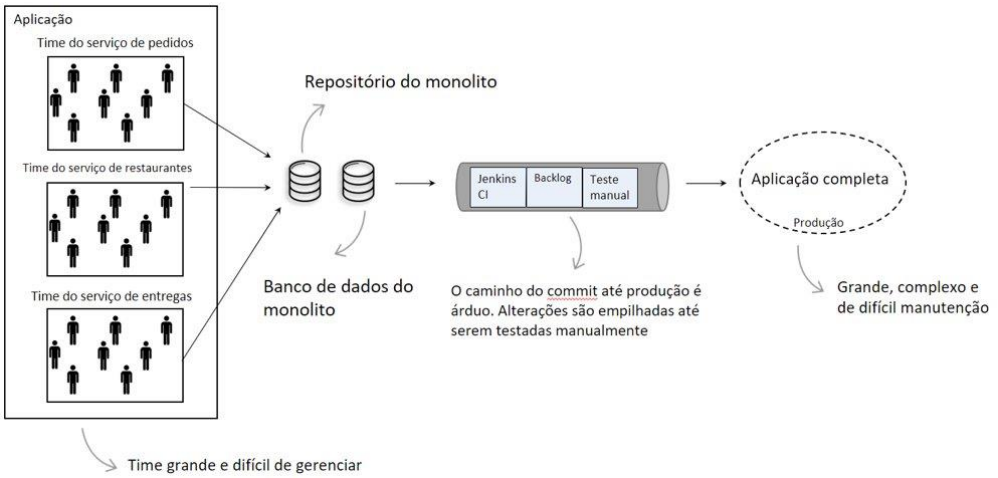

Figura 1: Arquitetura de um sistema com arquitetura monólito.
Fonte: Adaptado de Richardson, 2017, p. 45

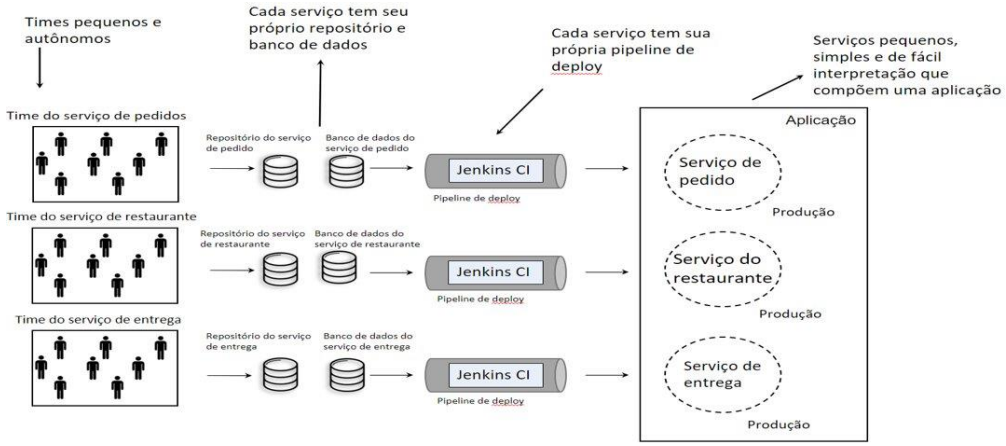

Figura 2: Arquitetura de um sistema com arquitetura de microserviço**.**
Fonte: Adaptado de Richardson, 2017, p. 45

Conclusões

No artigo são explicadas características de como a arquitetura se comporta, como ela pode resolver determinados problemas e seus pontos negativos. Além de comparações com a arquitetura monolítica e arquitetura orientada a serviços (SOA).

A arquitetura microserviço é essencial quando envolve o contexto de escalabilidade, mas sua complexidade é tão alta quanto seus benefícios. E é necessário entender se há a necessidade de escalabilidade. E o mais importante, disseminar o conhecimento da necessidade dela para que todos possam entendê-la, e assim trabalharem em prol de se manter uma arquitetura consistente.

A partir daqui poderão ser desenvolvidos artigos que explicam o que é *DevOps*, e como essa arquitetura combina perfeitamente com essa nova profissão (e cultura) emergente.

Agradecimentos

Agradecimentos especiais ao meu co-orientador Carlos Magnus, e pela participação de Maycon Cruz na banca.

Referências

AMARAL, ODRAVISON; CARVALHO, MARCUS. Arquitetura de Micro Serviços: uma comparação com sistemas monolíticos, 2017. Disponível em: https://repositorio.ufpb.br/jspui/bitstream/123456789/3235/1/OAJ14062017.pdf. Acesso em: 03 abril 2022.

BLOG, Netflix Technology (org.). **How We Build Code at Netflix**. 2016. Disponível em: https://netflixtechblog.com/how-we-build-code-at-netflix-c5d9bd727f15. Acesso em: 07 maio 2022.

ÖZKAYA, Mehmet. **Monolithic Architecture Is Still Worth at 2021?** 2021. Disponível em: https://medium.com/design-microservices-architecture-with-patterns/monolithic-architecture-is-still-worth-at-2021-98bfc112dc24. Acesso em: 11 abr. 2022.

REDHAT (org.). **O que é arquitetura orientada a serviços (SOA)?** Disponível em: https://www.redhat.com/pt-br/topics/cloud-native-apps/what-is-service-oriented-architecture. Acesso em: 28 maio 2022.

RIBEIRO, Bruno Rafael Costa. **Estudo comparativo entre arquiteturas monolíticas e de micro serviços**. 2017. 1 v. Tese (Mestrado) - Curso de Sistemas e Tecnologias de Informação

Para As Organizações, Escola Superior de Tecnologia e Gestão de Viseu, Viseu, 2017. Disponível em: https://repositorio.ipv.pt/bitstream/10400.19/4810/1/Dissertacao_MSTIO_BrunoRibeiro_ESCOLA.pdf. Acesso em: 11 abr. 2022.

RICHARDSON, Chris. **Microservices Patterns: With Examples in Java**. São Franciso: Manning Publications, 2018. 520 p.

TALEB, Feras. **Microservices Architecture: To Be Or Not To Be**. 2019. Disponível em: https://feras.blog/microservices-architecture-to-be-or-not-to-be/. Acesso em: 18 abr. 2022.