Microserviços com foco em requisições e disponibilidade de dados.

D.V.Silva

Faculdade de Tecnologia de São José dos Campos - Professor Jessen Vidal

Av. Cesare Mansueto Giulio Lattes, 1350 - Eugênio de Melo, São José dos Campos/SP,

CEP.: 12247-014, Brasil.

Telefone: (12) 3905-2423

Com o crescimento e aumento da acessibilidade a tecnologia o número de requisições vem aumentando ao passar do tempo, exemplo disso são aparelhos móveis que está a todo momento conectado com as rede sociais.

No entanto grande parte das arquiteturas atuais tem como base a “arquitetura monolítica” onde as execuções são centralizadas em um único servidor, como o passar do tempo os sistemas exige diversas mudanças e isso vem dificultando a escalabilidade e gerenciamento dos dados gerando um déficit técnico.

[Método de migração de sistemas monolíticos legados para a arquitetura de microsserviços (unb.br)](https://repositorio.unb.br/bitstream/10482/41178/1/2021_TaylorRodriguesLopes.pdf)(dia 20/09/2022)

Baseados nessas dificuldades muitas empresas estão adotando um anova arquitetura denominada “microsserviços”, este vem sendo uma saída para a modernização dos sistemas.

Os microsserviços é composto por software dividido em pequenas partes, onde é desmembradas as suas funções, como demostrado na figura (xxxxxxx) isto facilita a manutenção e o desenvolvimento do software pois cada pequena parte pode ser tratada e desenvolvida separadamente sem prejudicar o funcionamento das outras.

[2021\_TaylorRodriguesLopes.pdf](file:///D:\Artigo\Artigos\2021_TaylorRodriguesLopes.pdf)(dia 04/10/2022)

Sendo assim este artigo vem abortar e os benefícios de se utilizar o microserviçoes desenvolvendo um projeto onde é composto de microserviços explicando suas funções e recolhendo dados demostrando os benefícios do processo., sendo feita uma comparação da arquitetura de monólito com a de microserviço. Sendo utilizada a tecnologia java como linguagem principal

[DevOps: A Software Architect's Perspective - Len Bass, Ingo Weber, Liming Zhu - Google Books](https://books.google.com.br/books?hl=en&lr=&id=fcwkCQAAQBAJ&oi=fnd&pg=PT13&dq=+DevOps:+A+Software+Architect%27s+Perspective&ots=KSzru4GRRa&sig=90qGop6bo8u2ry4JqE_R-H0Gxgk&redir_esc=y#v=onepage&q&f=false)

(acessado dia 18/10/2022)

**Introdução**

Uma das principais dificuldades em elaborar uma solução é uma alta demanda de requisição simultâneas que os sistemas podem consumir, assim sendo empresas e desenvolvedores precisam preparar o sistemas para o aumentar de escalabilidade, conforme a demanda do cliente final, pois com o passar do tempo as requisições simultâneas de uma solução(produto, sistema) está aumentando cada vez mais. Dentro desse cenário grandes empresas como IBM, Netflix entre outras estão migrando a sua arquitetura de monolítico para microserviços procurando melhorar a experiência do seu usuário final, esse processo de migração mostra que a arquitetura de microserviço gera vantagens comparada com a arquitetura de monolítico.

A arquitetura é parte crucial de uma solução, sendo assim o desenvolvedor leva em consideração diversas desvantagens ou vantagens como o tempo de ciclo de vida a facilidade de realizar manutenção no sistema e o tempo que leva para alterações.

//Confirmas se a parte de cima esta ok, foi baseada na parte abaixo.

Acessado 18/10/2022

Acesso 25/10/2022

[Método de migração de sistemas monolíticos legados para a arquitetura de microsserviços (unb.br)](https://repositorio.unb.br/bitstream/10482/41178/1/2021_TaylorRodriguesLopes.pdf)

08/11/2022

Além disso a arquitetura leva em consideração as expectativas do usuário final e possibilita a compreensão entre o relacionamento dos elementos.

[Método de migração de sistemas monolíticos legados para a arquitetura de microsserviços (unb.br)](https://repositorio.unb.br/bitstream/10482/41178/1/2021_TaylorRodriguesLopes.pdf)

A arquitetura de microserviços é baseada em um princípio de relacionamento único onde tarefas autônimas trabalham juntos. Cada tarefa possui sua função dentro dos ecossistemas da solução baseada nas regras de negócio que é proposta pelo cliente. Sendo assim a implementação de cada serviço é realizada de modo individual tendo sua comunicação via rede.

Sendo assim a arquitetura fica bem caracterizada como uma arquitetura Orientada a serviço (SOA) onde todas as aplicações trabalham em conjunto, porém de forma independente.

O microserviços traz entre outras facilidades a comunicação por API (Apllication Programming Interface) onde possibilita a facilidade de se trabalhar com linguagem de código diferente, além disso é recomendado a padronização de comunicação via HTTP (Hiper text Transfr Protocol) e Jason (*JavaScript Object Notation*) como forma de dados.

O microserviço traz muitas vantagens, como escalabilidade e para sistemas com variedade de serviço é possível escolher qual a melhor tecnologia será implementada em cada funcionalidade.

Além disso o microserviço traz a segurança de um sistema onde as falhas não afetam todos os serviços. Por trabalhar separado de forma individual, quando um determinado serviço entra em falha ele não interfere nas outras aplicações, isto proporciona uma segurança maior dos sistemas pois apenas um serviço é afetado. O fato de trabalha em pequenas partes e de forma autônima deixa arquitetura ainda mais interessante no que diz respeito a melhorias de implementação pois isto pode ser feito isoladamente a fada parte do sistema onde o microserviço é dedicado. Isto torna as melhorias e manutenção mais naturais e dinâmica.

//Confirmas se a parte de cima esta ok, foi baseada na parte abaixo.

08/11/2022

https://periodicos.ufsc.br/index.php/eb/article/view/1518-2924.2018v23n53p147

Baseado no Princípio da Responsabilidade Única de Robert C. Martin, Newman

(2015) propôs uma arquitetura de microsserviços, na qual serviços pequenos e autônomos

trabalham de forma cooperativa. Cada serviço possui uma única responsabilidade, de acordo

com a regra de negócio que ele implementa, por conseguinte, torna-se fácil evitar que o

código aumente de forma a prejudicar a sua manutenabilidade. Por ser uma entidade

dissociada, a implantação de cada um dos microsserviços deve ser realizada de forma

independente, como um serviço isolado em uma plataforma como serviço (PaaS), e toda a

sua comunicação deverá ser realizada via chamadas de rede (NEWMAN, 2015).

Pode-se dizer que micro serviços nada mais são do que serviços bem

definidos, pouco acoplados, seguindo os princípios da boa e velha

arquitetura orientada a serviços (SOA), com um grau de maturidade maior,

justamente por usa independência e capacidade de trabalhada em conjunto

com outros micro serviços ou de forma autônoma ao mesmo tempo

(MOREIRA; BEDER, 2015).

Cada um dos microsserviços expõe uma inteface de programação de aplicações, cujo

acrônimo API vem do inglês *Application Programming Interface*, que será utilizada por

outros serviços para comunicação. Visando possibilitar que cada microsserviços seja escrito

em diferentes linguagens de programação, Moreira e Beder (2015) recomendam que sejam

utilizados protocolos de comunicação padronizados, como o HTTP (*Hypertext Transfer*

*Protocol*), e formatos de dados multiplataforma, como o JSON (*JavaScript Object Notation*).

A utilização de microsserviços oferece um conjunto de vantagens que podem ser

identificadas como heterogeneidade tecnológica, resiliência, escalabilidade e facilidade de

implantação (NEWMAN, 2015; MOREIRA; BEDER, 2015). Em um sistema com múltiplos

serviços colaborativos, é possível definir, visando a performance, qual a tecnologia mais

adequada para implementar uma determinada funcionalidade. Em uma rede social, por

exemplo, seria possível elaborar um microsserviço na linguagem de programação Java que

armazenasse as interações entre os usuários em uma base de dados orientada a grafos. Por

outro lado, o serviço responsável pela gestão dos conteúdos postados seria em Ruby e

utilizaria um banco orientado a documentos.

**Figura 1**: Modelo de rede social

**Fonte**: Elaborada pelo autor

Resiliência diz respeito a habilidade de um sistema de se recuperar em uma falha.

Caso algum componente apresente uma falha, é possível isolá-lo sem comprometer a

disponibilidade do sistema. Segundo Moreira e Beder (2015), a Netflix é um excelente

exemplo de uso desta arquitetura: se o serviço de recomendações, por exemplo, falhar, o

153

serviço de *streaming* continua operando normalmente. Em uma aplicação monolítica, caso

ocorra algum erro, possivelmente a aplicação inteira permanecerá indisponível durante o

período de manutenção. No que tange a escalabilidade, Richardson (2014) afirma que em

aplicações que utilizam a arquitetura de microsserviços é possível definir quais serviços

precisam ser atualizados.

Cada microsserviço pode ser implantado de forma independente de outros,

permitindo que o código seja aplicado de forma mais rápida. Caso ocorra alguma falha no

processo de implantação, o serviço poderá ser isolado para que a transação seja encerrada e

as mudanças descartadas. Newman (2015) cita esta vantagem como chave para que grandes

empresas, como Netflix e Amazon, adotarem esta arquitetura de software.

Todavia, a utilização de microsserviços também oferece um conjunto de

desvantagens, que podem ser caracterizadas como complexidade de desenvolvimento,

chamadas remotas e gerenciamento de múltiplos bancos de dados e transações. No que

corresponde a primeira característica, pode-se destacar a dificuldade em estabelecer a

comunicação entre projetos distintos e realizar o gerenciamento das dependências, apesar

de existirem ferramentas específicas para isso, como o Maven por exemplo. No que tange as

chamadas remotas, é necessário definir o escopo de cada um dos serviços para que este

possa realizar a sua tarefa de forma independente, sem a necessidade de realizar chamadas

constantemente, haja vista que estas se tornam mais custosas devido ao tráfego na rede. No

que está relacionado ao gerenciamento de múltiplas bases de dados, as configurações e o

tratamento de falhas devem ser feitos de forma correta e a manutenção dos bancos poderá

apresentar um alto custo.