UNIVERSIDADE FEEVALE

RONEI DOS SANTOS BERLEZI

##### ANÁLISE SOBRE A CONSISTÊNCIA DE DADOS EM APLICAÇÕES BASEADAS NA ARQUITETURA DE MICROSSERVIÇOS

(Título Provisório)

###### Anteprojeto de Trabalho de Conclusão

Novo Hamburgo

2018

RONEI DOS SANTOS BERLEZI

##### ANÁLISE SOBRE A CONSISTÊNCIA DE DADOS EM APLICAÇÕES BASEADAS NA ARQUITETURA DE MICROSSERVIÇOS

(Título Provisório)

Anteprojeto de Trabalho de Conclusão de Curso, apresentado como requisito parcial

à obtenção do grau de Bacharel em

Ciência da Computação pela

Universidade Feevale

Orientador: Rodrigo Rafael Villarreal Goulart

Novo Hamburgo

2018

# RESUMO

Com o crescente aumento da complexidade dos sistemas corporativos, surgiu um novo termo, Microsserviços. Este termo surgiu nos últimos anos para descrever uma maneira específica de desenvolver aplicações com o isolamento de responsabilidades sobre o sistema como um todo. A arquitetura tradicional monolítica é o caminho natural do desenvolvimento de um sistema, toda a lógica para tratar uma requisição é feita em um único processo utilizando tecnologias especificas. É possível escalar um aplicativo monolítico horizontalmente, de forma a executar várias instâncias. Esta arquitetura possui limitações, pois cada vez que um *deploy* é realizado, toda a aplicação deve ser reiniciada. Microsserviços, por ser uma arquitetura de pequenos projetos com um só foco, traz inúmeras vantagens, como independência de tecnologia para construir os módulos, possibilidade de mudança rápida em um único serviço sem necessidade de restestar e gerar um novo *deploy* para toda a aplicação, alta escalabilidade, etc. No contexto de banco de dados em sistemas monolíticos, transações utilizam o conjunto de propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade) para garantir a integridade dos dados, já na arquitetura de microsserviços, os serviços devem estar desacoplados e, portanto, cada um deve ter seu próprio repositório de dados, apesar de algumas implementações utilizarem banco de dados relacional para isso, a grande maioria das aplicações baseadas em microsserviços utilizam bancos NoSQL. A implementação de transações que utilizam dados de múltiplos serviços acaba aumentando a complexidade do sistema, ao mesmo tempo que muitos dos bancos de dados não suportam transações distribuídas. Sendo assim, este trabalho tem como objetivo sintetizar os métodos de garantir a consistência de dados dentro de um sistema baseado na arquitetura de microsserviços.

Palavras-chave: microsserviços. Transações distribuídas. NoSQL. contexto de dados.

SUMÁRIO

[RESUMO 3](#_Toc526795375)

[MOTIVAÇÃO 5](#_Toc526795376)

[METODOLOGIA 9](#_Toc526795377)

[CRONOGRAMA 10](#_Toc526795378)

[BIBLIOGRAFIA 11](#_Toc526795379)

# MOTIVAÇÃO

Vivemos em uma constante mudança de tecnologias e abordagens no âmbito do desenvolvimento de aplicações, fazendo com que reavaliemos como construímos e entregamos soluções aos nossos clientes. Um exemplo disso é a adoção de várias empresas pela utilização da arquitetura baseada em microsserviços. Esta arquitetura é composta por pequenos serviços autônomos que trabalham juntos, definidos como pequenos módulos responsáveis por fazer uma única coisa bem feita, de forma que alterações não afetam os demais, e ao mesmo tempo permite a geração de um novo *deploy* por si próprio sem a necessidade de alteração nos consumidores (NEWMAN, 2015).

Microsserviços ganharam popularidade na indústria nos últimos anos. Esta arquitetura pode ser considerada um refinamento e simplificação de *Service-Oriented Architecture (SOA)* (AMARAL et al., 2016).

SOA emergiu para combater os desafios de grandes aplicações monolíticas através da quebra do sistema em uma série de serviços que trabalham de forma colaborativa para fornecer um conjunto de funcionalidades. A ideia em si é muito sensata, mas apesar dos esforços da comunidade, houve uma falta de consenso em como aplicar o SOA. Muitos dos problemas envolviam protocolos de comunicação, como o SOAP (*Service-Oriented Architecture Protocol*), e principalmente a falta de orientação sobre a granularidade de serviços (NEWMAN, 2015).

A arquitetura de microsserviços veio como uma variação de SOA, este conceito começou a ser adotado pela comunidade em 2014 como uma resposta direta a alguns dos desafios de escalabilidade, tanto tecnicamente quanto organizacional de grandes aplicações monolíticas (CARNELL, 2017),.

A arquitetura monolítica, que é definida como “aplicações que utilizam uma base de código única e geram um único artefato de *deploy”* (KALSKE, 2017), é um modelo que pode haver diversos times trabalhando no desenvolvimento de uma única grande e complexa solução de software. Os repositórios de códigos crescem conforme o projeto evolui, frequentemente a compreensão começa a se tornar mais difícil com o avanço da complexidade da aplicação, fazendo com que a implementação de novas funcionalidades ou até mesmo a correção de erros seja uma tarefa árdua para os desenvolvedores. Uma das vantagens da arquitetura monolítica ocorre na fase de desenvolvimento inicial, onde a quantidade de código é menor, que torna o entendimento da aplicação mais fácil e a tarefa de *deploy* acaba por ser mais simples, por exemplo em uma aplicação Java Web, composta por diversos serviços e componentes, que acaba por gerar um único pacote. Em contrapartida, quando o código passa a crescer, a arquitetura pode se deteriorar, afetando o processo de desenvolvimento. Para solucionar estes problemas em grandes bases de código, o modelo de arquitetura de microsserviços é apresentado.

Os benefícios apresentados pela arquitetura de microsserviços incluem bases de códigos simplificadas para serviços individuais, habilidade de atualizar e escalar serviços de forma isolada, possibilidade de escrita de serviços em diferentes linguagens e a utilização de diferentes camadas de dados (AMARAL et al., 2016).

Um dos maiores desafios de sistemas baseados em microsserviços é a divisão dos serviços, cada um deles deve ser simples o suficiente a ponto de possuir poucas responsabilidades. Esta fragmentação da aplicação pode transformar o gerenciamento de dados no gargalo do sistema (MESSINA et al., 2013).

Existem dois padrões predominantes na implementação de banco de dados, por serviço e compartilhado (KALSKE, 2017). A utilização de um banco de dados compartilhado como forma de integração entre os serviços pode fazer sentido em um primeiro momento, é uma forma rápida e fácil de implementação, mas ao utilizar esta técnica estamos permitindo que recursos externos sejam ligados diretamente ao modelo. Ao modificar a modelagem de dados, deve ser feita uma análise em todos os serviços afim de evitar que outros consumidores não quebrem (NEWMAN, 2015).

Deve-se evitar a comunicação indireta entre microsserviços pela utilização de banco de dados compartilhado. Uma das ramificações deste principio é que usualmente cada microsserviço irá armazenar seus dados em um banco isolado (BROWN; WOOLF, 2016). Com a utilização do padrão de banco de dados por serviço, o acesso a informações de outros contextos ocorre por forma de chamadas aos serviços responsáveis. Ao descentralizar os dados, existem implicações durante o gerenciamento das atualizações. Transações garantem que toda a informação seja armazenada no banco, ou nenhuma. Esta abordagem é comumente utilizada em arquiteturas monolíticas. Transações distribuídas são notoriamente difíceis de implementar e por consequência as arquiteturas de microsserviços enfatizam a coordenação sem transação entre os serviços, com reconhecimento explicito de consistência eventual e de que os problemas serão resolvidos por operações de compensação (LEWIS; FOWLER, 2014).

Uma transação distribuída é um processo complexo com muitas partes móveis que podem falhar. Todas estas partes são executadas em máquinas diferentes ou até mesmo em *data centers* diferentes, o processo de concluir uma transação pode se tornar muito longo e inseguro. O objetivo deste trabalho é, portanto, sintetizar as possíveis abordagens para manter a consistência de dados sobre uma arquitetura baseada em microsserviços, analisando o contexto de dados e padrões propostos para garantir uma transação distribuída, como o modelo de consistência eventual e compensação, o protocolo *Two-Phase-Commit* e o padrão SAGA.

OBJETIVOS

Objetivo geral

O objetivo geral é avaliar e sintetizar os diversos métodos de garantia de consistência de dados dentro de uma arquitetura baseada em microsserviços.

Objetivos específicos

* Comparar métodos de transações de arquiteturas monolíticas e microsserviços
* Analisar a fragmentação de dados baseado em contexto
* Identificar métodos e padrões de transações
* Identificar atributos para comparação entre os métodos de transação distribuída
* Comparar padrões de transação distribuída
* Identificar e avaliar os meios de escalar dados e serviços
* Avaliar métodos de consistência de dados dentro do ambiente *Amazon Web Services (AWS)*

# METODOLOGIA

O presente projeto de pesquisa é de natureza básica qualitativa, pois procura elucidar e entender os métodos para garantir a consistência de dados em aplicações baseadas em microsserviços. Quanto aos objetivos, o trabalho será realizado através de pesquisa exploratória, elaborando uma pesquisa bibliográfica sobre um referencial teórico, deve-se investigar e observar o comportamento dos dados sobre uma arquitetura distribuída, identificando os principais meios de garantia de consistência de dados em sistemas baseados na arquitetura de microsserviços.

Será feito um levantamento de atributos para um estudo comparativo entre os padrões e métodos de transações propostos na literatura. Para a validação dos modelos de estudo será utilizado a plataforma *Amazon Web Services* a fim de avaliar e comparar as implementações dos métodos de consistência de dados.

O experimento a ser realizado compreende no desenvolvimento de um sistema baseado em microsserviços como uma prova de conceito, onde o mesmo apresentará variações na arquitetura através da utilização de diferentes padrões e métodos de transações, leitura de dados e replicação de informações propostos na literatura. No que tange a persistência de dados, pretende-se utilizar um banco de dados NoSQL com instâncias específicas por microsserviço, abordando o conceito de escalamento tanto de armazenamento quanto por serviços.

# CRONOGRAMA

Trabalho de Conclusão I

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Etapa | Meses | | | |
|  | Ago | Set | Out | Nov |
| Elaboração anteprojeto | X | X |  |  |
| Entrega anteprojeto |  | X |  |  |
| Comparativo transações monolíticas e distribuídas |  | X | X |  |
| Estudo sobre a fragmentação de dados |  |  | X |  |
| Estudo sobre dimensionamento de serviços e dados |  |  | X |  |
| Estudo de modelos de transações distribuídas |  |  | X | X |
| Comparação dos modelos |  |  | X | X |
| Entrega do trabalho |  |  |  | X |

Trabalho de Conclusão II

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Etapa | Meses | | | |
|  | Mar | Abr | Mai | Jun |
| Definição das tecnologias para validação dos modelos | X |  |  |  |
| Identificação de métricas para comparação |  | X |  |  |
| Implementação dos modelos de transação |  | X | X |  |
| Coleta de resultados e sintetização |  |  | X | X |

# BIBLIOGRAFIA

AMARAL, M. et al. Performance evaluation of microservices architectures using containers. **Proceedings - 2015 IEEE 14th International Symposium on Network Computing and Applications, NCA 2015**, p. 27–34, 2016.

BROWN, K.; WOOLF, B. Implementation patterns for microservices architectures. **Pattern Language of Patterns**, p. 7, 2016.

CARNELL, J. **Spring Microservices in Action**. [s.l.] Manning, 2017.

KALSKE, M. Transforming monolithic architecture towards microservice architecture. 2017.

LEWIS, J.; FOWLER, M. **Microservices**. Disponível em: <https://martinfowler.com/articles/microservices.html>. Acesso em: 8 set. 2018.

MESSINA, A. et al. The Database-is-the-Service Pattern for Microservice Architectures. v. 8060, n. January 2018, 2013.

NEWMAN, S. **Building Microservices**. [s.l.] O’Reilly Media, Inc., 2015.