USO DE BANCO DE DADOS EM ARQUITETURA DE MICROSERVIÇOS

USE OF DATABASE IN MICROSERVICE ARCHITECTURE

OLIVEIRA, Taíssa C. S.[1]; LIMA, Iremar N.L.[2]

PALAVRAS-CHAVE
Arquitetura de sistemas, Microsserviços, Banco de Dados.

KEYWORDS
Systems architecture, Microservices, Database.

RESUMO
A arquitetura monolítica de sistemas tornou-se um padrão comum que gera complicações para grandes aplicações. Entretanto, a arquitetura de Microsserviços aparece como uma via alternativa para desenvolvimentos de aplicações complexas. Com o avanço desta tecnologia e sua ampla utilização em sistemas complexos este artigo se propõe a discutir características, vantagens e desvantagens da arquitetura de microsserviços hoje em dia. Em particular, o artigo analisa como esta tecnologia pode ser utilizada em conjunto com os modernos sistemas bancos de dados.

ABSTRACT
Monolithic systems architecture has become a common standard that creates complications for large applications. However, the Microservices architecture appears as an alternative route for the development of complex applications. With the advancement of this technology and its widespread use in complex systems, this article aims to discuss the characteristics, advantages and disadvantages of the microservice architecture today. In particular, the article examines how this technology can be used in conjunction with modern database systems.

1 Bacharel no curso de Sistemas de Informação do Centro Universitário Newton Paiva: taissa.cristina@hotmail.com
2 Professor Orientador: iremar.prof@newtonpaiva.br

1 INTRODUÇÃO

Nos últimos anos tem crescido consideravelmente a necessidade por empresas de diferentes setores pelo uso nos sistemas de informação da arquitetura de microsserviços. Isto se deve em grande parte devido ao fato da arquitetura de microsserviços aperfeiçoar a velocidade dos processos, dos testes, da escalabilidade e da gestão de equipes de desenvolvimento, dentre outros benefícios.

A arquitetura de microsserviços é uma aplicação que transforma um sistema monolítico, e o separa em pequenos serviços, que trabalham de forma individual. Os processos se comunicam entre si através da rede, tornando-se uma espécie de sistema distribuído. Microsserviços "são serviços que podem ser implantados de forma independente, e são modelados em torno de um domínio de negócios" (NEWMAN, 2020).

Este tipo de arquitetura apresenta importantes benefícios, os quais possibilitam múltiplas praticidades quanto à implantação, e um desenvolvimento mais rápido, permitindo o uso de diversas tecnologias, ao mesmo tempo.

Segundo Newman (2020), a natureza independente da arquitetura de microsserviços aumenta a escala e robustez dos sistemas, além da possibilidade de misturar e combinar diferentes tecnologias, linguagens e estilos de programação, plataformas de implantação ou banco de dados para encontrar a combinação correta e perfeita para trazer o melhor resultado ao cliente final do sistema.

Apesar dos diversos benefícios oferecidos por esse tipo de arquitetura, embora já adotados por muitas empresas como a Amazon, Nubank e Netflix entre outras, muitos ainda são os desafios para a implantação. Em particular, um problema importante diz respeito a como projetar o componente de banco de dados numa arquitetura de microsserviço.

Este artigo tem como objetivo entender como se dá a transição do projeto de banco de dados de um sistema que utiliza arquitetura monolítica para uma arquitetura de microsserviços. Na seção 2, será discutido a arquitetura de microsserviços. Na seção 3 é abordado como a arquitetura de microsserviços é integrada aos Sistemas Gerenciadores de Banco de Dados (SGBDs) e por fim, na seção 4 é discutido as estratégias para projetar o banco de dedos para sistemas de informação que utilizam a arquitetura de microsserviços.

2 A ARQUITETRUA DE MICROSSERVIÇOS

Segundo RedHat (2020), microsserviços é uma arquitetura de software inovadora, que consiste em desmembrar ou construir uma aplicação em pequenas partes, pequenos serviços, que podem ser implantados de forma independente, que trabalham juntos para realizar uma determinada tarefa.

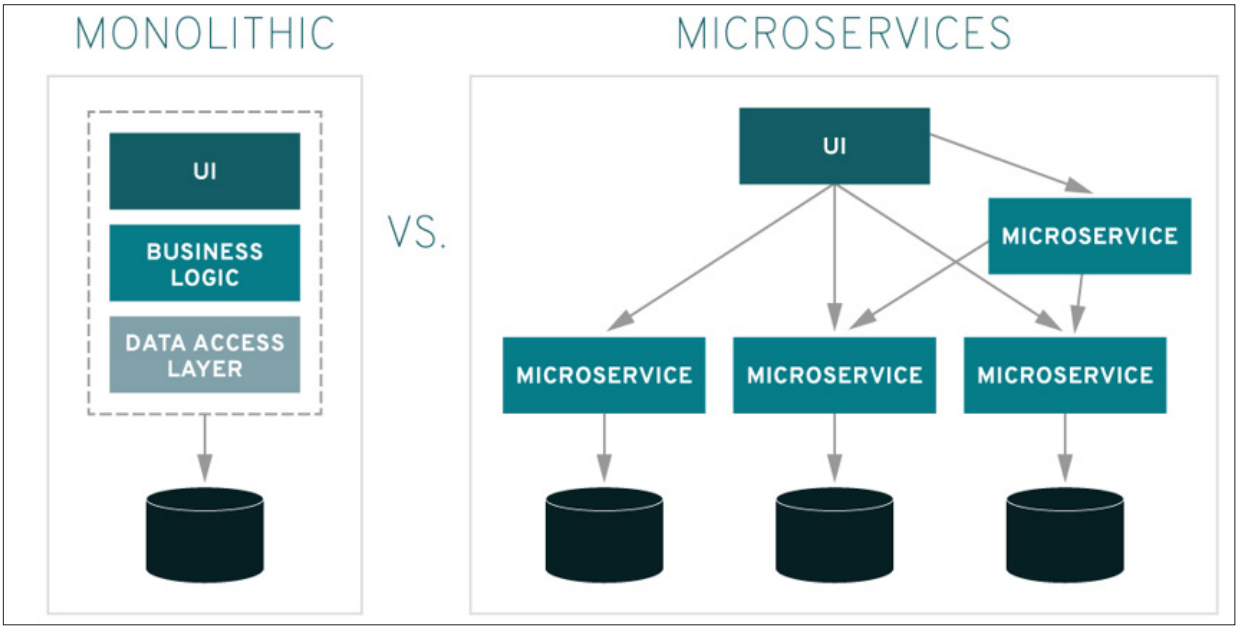
Como exemplo, ao acessarmos um site de compras pela internet, e utilizarmos a barra de pesquisa para procurar um determinado produto, calcular frete, adicionar ao carrinho ou até mesmo adicionar em uma lista de desejo, estamos utilizando um dos serviços, possíveis dessa arquitetura de microsserviços.

Segundo AWS (2020), "Microsserviços são uma abordagem arquitetônica e organizacional do desenvolvimento de software na qual o software consiste em pequenos serviços independentes que se comunicam usando APIs (*Application Programming Interface*) bem definidas".

Na arquitetura monolítica a aplicação de um sistema é caracterizada como um todo: um único serviço, que possui muitos processos dependentes e acoplados. Os sistemas tendem a crescer e tornam-se complexos, manutenções caras e lentas, que podem causar inconsistências na aplicação, além de se prender na tecnologia utilizada o que o torna não flexível. Qualquer mudança requer reinicialização do sistema (OPTUS SOFTWARE, 2017).

"O que diferencia a arquitetura de microsserviços das abordagens monolíticas tradicionais é como ela decompõe a aplicação por funções básicas." (REDHAT, 2020)

FIGURA 1: Arquitetura monolítica versus arquitetura microsserviços



Fonte: (REDHAT, 2020)

Em microsserviços, cada serviço é desenvolvido para solucionar um único problema, como se fosse um serviço especializado e autônomo. Se precisar de uma manutenção ou otimização, entre outras coisas, é possível realizar essa tarefa sem afetar os demais serviços. Sem que toda a aplicação seja reiniciada.

Abaixo, segue uma figura que diferencia uma arquitetura monolítica (*Monolithic*), da arquitetura de microsserviço (*Microservices*).

Segundo RedHat (2020), com o desenvolvimento distribuído que essa arquitetura permite, resulta em menos tempo de entrega e mais praticidade. A equipe de desenvolvimento pode ser dividida, de modo que vários microsserviços estejam sendo produzidos ao mesmo tempo. “Mais desenvolvedores trabalhando simultaneamente na mesma aplicação”.

Segundo a AWS (2020), os benefícios que a arquitetura de microsserviços provem são:

1. **Agilidade**: permite equipes de desenvolvimento pequenas e independentes com autonomia e rapidez.
2. **Escalabilidade flexível:** cada serviço é escalado de forma independente, permitindo o dimensionamento da infraestrutura de acordo com a necessidade de cada serviço.
3. **Fácil implantação:** permite integração e entregas contínuas de cada serviço.
4. **Liberdade tecnológica:** essa arquitetura não se prende a uma única tecnologia, como a arquitetura monolítica. A equipe responsável pelo serviço tem a possibilidade de optar pela melhor ferramenta para executar cada tarefa.
5. **Código reutilizável:** por ser dividido em pequenos módulos, ou serviços, permitindo que um serviço criado para uma função, seja reutilizado como componente para outros recursos.
6. **Resiliência:** na arquitetura monolítica a falha de um componente pode parar toda a aplicação. Com a independência dos serviços, em microsserviços, a falha de todo um componente degrada a funcionalidade, mas não para a aplicação.

Paralelamente, a arquitetura de microsserviços tem os seus pontos negativos e desafios. Segundo RedHat (2020), são os seguintes:

1. A organização que pretende migrar seu sistema monolítico para microsserviço, deve ter em mente mudanças não só na aplicação como no modo que as pessoas trabalham. Devem considerar mudanças organizacionais e culturais. Cada equipe terá um ritmo próprio de implantação e produção do serviço que estará responsável.
2. Os desafios na aplicação se prologam nas questões como, compilação: para identificar as dependências entre os serviços é necessário mais tempo.
3. Os testes, tornam-se mais difíceis e mais importantes como jamais foram.
4. O controle de versão é mais difícil, por possuir vários serviços independentes. A compatibilidade das versões desses serviços requer mais atenção, de modo que a compatibilidade de um serviço para outro não seja rompida.
5. A implantação na fase inicial, requer investimento na automação, pois o sistema é bastante complexo para automação manual.
6. A geração de logs precisa estar centralizada para gerenciar a escala para unificação dos serviços.
7. O monitoramento torna-se um desafio para identificar as fontes dos problemas, a depuração remota não é viável e a conectividade de maneira centralizada e integrada, há saltos na rede entre os serviços, adicionando latência e perda de dados. A arquitetura precisa acomodar essas falhas.

3 microsserviços e banco de dados

Segundo Grupo Mult (2018), a tarefa de integrar dados na arquitetura de microsserviços é umas das principais preocupações de quem opta por essa arquitetura de desenvolvimento. “Os bancos de dados devem funcionar de forma independente, mas consistente.”. A maneira de garantir que todos os serviços funcionem em sua eficiência máxima e os dados consistentes é gerenciá-los de forma eficiente.

Um dos desafios de integrar esses dados, é a garantia de independência de cada serviço, com o armazenamento separado. Se utilizarmos o mesmo banco de dados, podem surgir problemas como a interferência de desempenho entre os serviços, falta de escalabilidade, entre outros, uma opção inviável para essa arquitetura. O que nos leva a outro ponto: banco de dados separados podem levar a redundância de dados e problemas de consistências dos dados. “Se um dado é atualizado, essa atualização será replicada para todos os serviços?” (GRUPO MULT, 2018).

“O principal objetivo para gerenciar dados em microsserviços é garantir que os bancos de dados de cada microsserviço continuem funcionando de forma independente um do outro, mas que ainda exista consistência e integração entre eles” (GRUPO MULT, 2018).

Segundo o Grupo Mult (2018), ao migrar de um banco de dados monolíticos para microsserviços, normalmente é utilizado uma estratégia de quatro passos, para extrair os dados de um único bloco e encaminhá-los para cada microsserviço:

1. Criação do serviço e delimitação dos dados a serem utilizados por ele;
2. Desacoplamento das aplicações da arquitetura monolítica migrando-as para o microsserviço recém-criado;
3. Movimentação dos dados do banco de dados compartilhado para um novo banco de dados privado;
4. Repetição desse processo até que todas as aplicações sejam levadas do monolítico para os microsserviços.

Depois de separados, é preciso adotar medidas de integração para garantir a consistência e a persistência deles, segundo grupo Mult (2018).

4 Projeto de banco de dados na arquitetura de microsserviços

Nesta seção será apresentado como projetar um banco de dados para sistemas de informação utilizando arquitetura de microsserviços. Nesta etapa, estaremos discorrendo sobre algumas estratégias, métodos e padrões, a se seguir para realizar a migração de um banco de dados monolítico para o de microsserviços, os principais desafios e como realizar esse gerenciamento de dados da melhor maneira possível.

Segundo IBM (2020), a melhor abordagem para refatorar o seu código existente, para passar sua aplicação para arquitetura de microsserviços, é a incremental. Porém, o maior desafio dessa abordagem incremental será refatorar o banco de dados.

"Quando falamos sobre a construção de bancos de dados para microsserviços é importante reduzir ou eliminar o acoplamento no banco de dados - mas isso realmente significa acoplamento no nível do esquema do banco de dados" (IBM, 2020).

Ter o próprio banco de dados para cada serviço é uma das técnicas utilizadas para realizar essa migração. Segundo Epsagon (2020), cada serviço deve ter o seu próprio banco de dado. Em geral, é uma boa prática os microsserviços nunca acessarem o banco de dados uns dos outros.

Uma boa prática é utilizar o banco de dados NewSQL (ou NoSQL), onde torna-se possível "fornecer serviços de dados para várias implementações de microsserviços de maneira gerenciável, simplificando a arquitetura geral e aumentando o desempenho" (SingleStoreBlog, 2020).

Segundo SingleStoreBlog (2020), os novos bancos de dados SQL, em geral, apresentam atributos necessários e importantes para aplicações de microsserviços como: velocidade, escalabilidade, diversidades no tipo de dados, transações, análises e softwares nativos em nuvem. Estes, em geral, são os principais recursos que se destacam em um ambiente de Microsserviços.

Para exemplificar, a seguir é ilustrado um exemplo de microsserviços, cada um com seu banco de dados, disponibilizado pelo Epsagon (2020), de um pedido de compras de videogames.

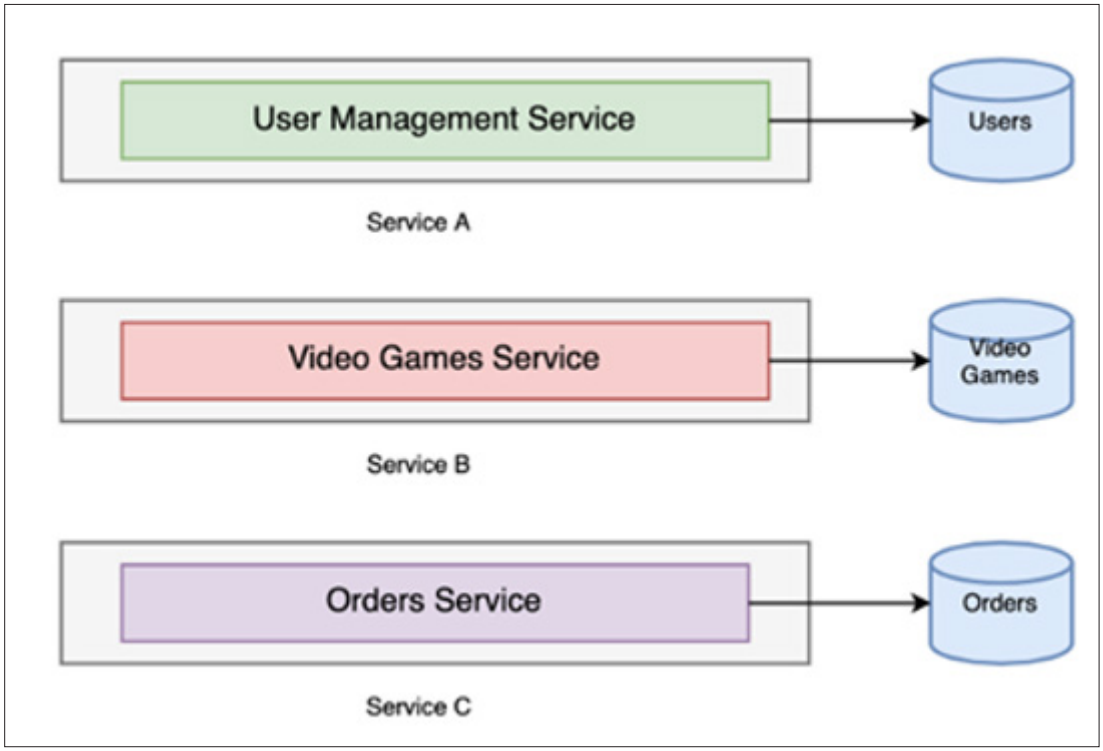
FIGURA 2 – Compra de videogames

Fonte: (EPSAGON, 2020)

Segundo Epsagon (2020), esta aplicação pode ser dividida em três serviços, e cada um tem o seu próprio banco de dados:

- Serviço de gerenciamento de usuários - Responsável pelo login e inscrição do usuário
- Serviço de videogames - Responsável pela lista de videogames disponíveis
- Serviço de pedidos - Responsável pela compra de videogames

FIGURA 3 - Banco de dados por microsserviço



Fonte: (EPSAGON, 2020)

Segundo Epsagon (2020), para que haja o melhor gerenciamento desses dados pelos microsserviços nessa aplicação, existem duas metodologias disponíveis para utilização:

- Comunicação Síncrona – quando o remetente envia uma solicitação direta e aguarda o processamento de algum tipo de resposta (normalmente envia solicitação por HTTP).
- Comunicação Assíncrona – não necessita esperar pela resposta, ela "pesquisa os resultados posteriormente ou grava uma função de retorno de chamada ou uma ação".

Outra técnica disponibilizada, segundo DataVersity (2019), é o padrão Saga. Ele é dividido em duas partes:

- Coreografia - onde a troca de comunicação se faz pela troca de eventos.
- Orquestração - onde existe um controle centralizado para invocar os participantes da saga usando as solicitações ou respostas

Segundo DataVersitiy (2019), esse padrão é predominante em aplicações com várias transações possíveis.

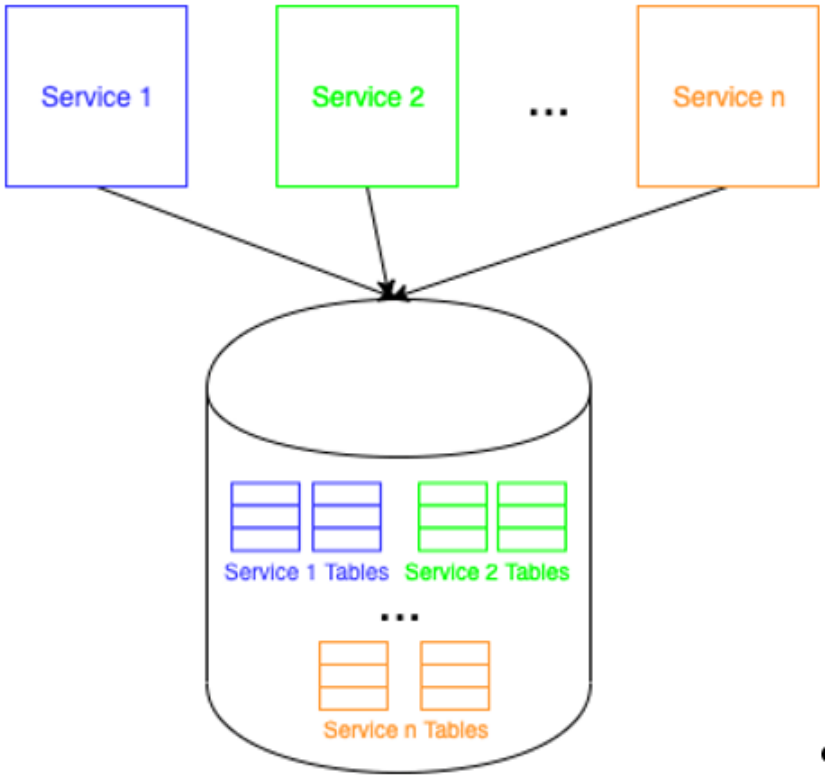
> "Uma série de transações locais onde cada transação publica um evento informando o status das consultas que estão sendo acionadas. Os outros serviços dependem do status dos serviços anteriores e, portanto, para as transações com status de falha anterior, a saga desfará automaticamente as transações posteriores." (DATAVERSITY, 2019).

Banco de dados compartilhado, segundo Dataversity (2019), é uma outra opção para quem não pode lidar com padrões de dados complexos e individuais. "O banco de dados compartilhado, como o nome indica, é comumente compartilhado pelos microsserviços para seus respectivos domínios de serviços" (DATAVERSITY, 2019).

Conforme afirmado por Dataversity (2012), as transações desse padrão são influenciadas pelas propriedades ACID (Atomicity, Consistency, Isolation, and Durability / atomicidade, consistência, isolamento e durabilidade) presentes atualmente nos sistemas gerenciadores de banco de dados Relacionais. Essas propriedades diminuem significativamente as vantagens de se utilizar a arquitetura de microsserviços, devido ao gerenciamento e a possível sobrecarga de dispor vários serviços em um mesmo banco de dados.

Segundo a DEV (2020), outra abordagem, disponível para utilização é optar por um único banco de dados lógico para configuração/armazenamento de todos os microsserviços. Cada microsserviço tem sua própria tabela, com referências e chaves estrangeiras para outras tabelas. Nessa abordagem a recuperação de um microsserviço específico torna-se mais complexo.

FIGURA 4 - Banco de Dados único, tabelas diferentes por microsserviços



Fonte: (DEV, 2020)

Segundo Microservice Architecture (2018), uma boa prática nesta arquitetura é o padrão de composição de APIs, onde são realizados os joins (junções) das tabelas usadas por cada microsserviço. Isto torna mais fácil implementar as consultas.

Segundo Dataversity (2019), existe ainda o padrão CQRS. Neste padrão existe a segregação dos dados voltados para otimização e tempo. Os dados são separados em camadas para cada microsserviço, por exemplo, camada de comandos, onde se inserem os dados, e camada de consultas, onde se recupera os dados armazenados.

E por último existe o padrão de fornecimento de eventos, como mostrado na figura 5. Nele é produzido eventos que já foram utilizados no aplicativo. Ele é capaz de armazenar e manter uma lista de eventos e os utilizará na execução de ações. São utilizados APIs para recuperar e adicionar eventos. Neste caso o padrão é mais indicado para aplicativos escaláveis, segundo Dataversity (2019).

FIGURA 5 - Padrão de Fornecimento de Eventosa

Fonte: (DataVersity, 2019)

Existem várias técnicas para realizar a migração de arquitetura monolítica para microsserviços. Porém, não existe uma forma única correta ou padrão a ser seguido à risca. A melhor forma de realizar essa migração é analisar toda a aplicação e dados e optar pela melhor forma da empresa e equipe agregar essa transição. E com avanço da tecnologia, hoje em dia temos API's, que possibilitam o gerenciamento de implementações desses dados de forma mais gerenciável, permitindo a simplificação e aumento do desempenho da arquitetura em geral. "A arquitetura de microsserviços desafia você a considerar que seu sistema pode não ter um único escopo." Software Engineering (2018).

CONCLUSÃO

Este artigo discutiu os conceitos e características da arquitetura de microsserviços, apresentou padrões de integração com os SGBDs, e discutiu estratégias de um sistema de banco de dados que utiliza a arquitetura de microsserviços nos sistemas de informação.

A arquitetura de um banco de dados utilizando microsserviços, tem ganhado popularidade nos últimos anos e se tornando um padrão no desenvolvimento de sistemas complexos que exigem alta escalabilidade. No entanto, existem vários desafios que precisam ser estudados, analisados e superados. Este artigo discutiu alguns deles apresentando algumas soluções encontradas na literatura.

REFERÊNCIAS

AWS**. O que são Microsserviços?.** 2020. Disponível em: <https://aws.amazon.com/pt/microservices/#:~:text=Microsservi%C3%A7os%20s%C3%A3o%20uma%20abordagem%20arquitet%C3%B4nica,pertencem%20a%20pequenas%20equipes%20autossuficientes.>. Acesso em 02/10/2020

DATAVERSITY. **ACID vs. BASE: The shifting pH of Database Transaction Processing**, 2012. Disponível em: <https://www.dataversity.net/acid-vs-base-the-shifting-ph-of-database-transaction-processing/>. Acesso em 18/11/2020.

DATAVERSITY. **Data Managment Patterns for Microservices Architecture,** 2019. Disponível em: <https://www.dataversity.net/data-management-patterns-for-microservices-architecture/#>. Acesso em 14/11/2020.

DEV. **Does your microservice deserve its own database?**, 2020. Disponível em: <https://dev.to/lbelkind/does-your-microservice-deserve-its-own-database-np2>. Acesso em 28/11/2020.

EPSAGON. **Monolithic to Microservices: Architecture and Data Management**, 2020. Disponível em: <https://epsagon.com/development/monolithic-to-microservices-architecture-data-management/>. Acesso em 18/11/2020.

Grupo Mult**. Como gerenciar e integrar dados em Microsserviços?** 2018. Disponível em https://www.grupomult.com.br/como-gerenciar-e-integrar-dados-em-microsservicos/. Acesso em 29/10/2020.

IBM. **Choosing the Right Database for Microsservices,** 2020. Disponível em: <https://www.ibm.com/cloud/blog/choosing-the-right-databases-for-microservices>. Acesso em 12/11/2020.

Microsservice Architecture. **Pattern: Database per Service**, 2018. Disponível em: <https://microservices.io/patterns/data/database-per-service.html>. Acesso em 28/11/2020.

NEWMAN, Sam; **Migrando Sistemas Monolíticos para Microsserviços**, Novatec, 1ª. Edição, 2020.

OPTUS SOFTWARE. **Micro Serviços: Qual a diferença para a Arquitetura Monolítica?**, 2017. Disponível em: <https://www.opus-software.com.br/micro-servicos-arquietura-monolitica/#:~:text=Como%20funciona%20a%20arquitetura%20monol%C3%ADtica,executada%20em%20uma%20mesma%20m%C3%A1quina.>. Acesso em 02/10/2020.

REDHAT**. O que são os Microsserviços?.** 2020 Disponível em: <https://www.redhat.com/pt-br/topics/microservices/what-are-microservices>. Acesso em 02/10/2020.

SingleStoreBlog. **A Balanced Approach to Database Use with Microservices**, 2020. Disponível em<https://www.singlestore.com/blog/a-balanced-approach-to-database-use-with-microservices/>. Acesso em 29/11/2020

SOFTWARE ENGINEERING. **How to keep relationship integrity with Microservice Architecture**, 2018. Disponível em: <https://softwareengineering.stackexchange.com/questions/381279/how-to-keep-relationship-integrity-with-microservice-architecture>. Acesso em 29/11/2020.