INTERFACE TECNOLÓGICA

MICROSERVIÇOS: um estudo de caso apontando suas potencialidades

MICROSERVICES: a case study pointing out its potential

Leonardo Luis Domingos – l_ldomingos@hotmail.com
Universidade de Araraquara – UNIARA – Araraquara – SP – Brasil

Renata Mirella Farina – mirellafarina@yahoo.com.br
Universidade de Araraquara – UNIARA – Araraquara – SP – Brasil

DOI: 10.31510/infa.v17i2.851
Data de publicação: 18/12/2020

RESUMO

O presente artigo tem como objetivo apontar as potencialidades e as vantagens da utilização da arquitetura de microserviços frente a arquitetura monolítica. A metodologia utilizada refere-se a um estudo de caso com abordagem qualitativa. O estudo de caso apresenta a migração de um sistema de *e-commerce*, construído inicialmente com a arquitetura monolítica, para uma arquitetura de microserviços. De acordo com os resultados descritos conclui-se que uma aplicação em arquitetura de microserviços acaba por refletir em diversos benefícios, como a facilidade de manutenção, escalabilidade dos serviços, baixo acoplamento e independência, de modo que um problema apresentado passa a não afetar o restante da aplicação, demonstrando, assim, as vantagens por optar pela utilização do microserviços.

Palavras-chave: Arquitetura de microserviços. Arquitetura monolítica. Estudo de caso.

ABSTRACT

This article aims to point out potentialities and advantages of using microservice architecture compared to monolithic architecture. The methodology used refers to a case study with a qualitative approach. The case study presents the migration of an e-commerce system, initially built with monolithic architecture, to a microservice architecture. According to the results, it is concluded that an application in microservices architecture ends up reflecting on several benefits, such as ease of maintenance, scalability of services, low coupling and independence. In this way a problem does not affect the rest application, thus demonstrating the advantages of choosing microservices.

Keywords: Microservices architecture. Monolithic architecture. Case study

1 INTRODUÇÃO

A grande e a constante evolução da área de tecnologia da informação (TI) vem proporcionando a criação de *softwares* (que chamaremos neste artigo muitas vezes de aplicações) nas organizações cada vez maiores, com mais funcionalidades e com tecnologias que englobam novos produtos e serviços.

Um problema comum com a rápida evolução desta área é que algumas aplicações com o passar do tempo acabam tornando-se obsoletas ou há o surgimento de novas ferramentas que resolvem de forma melhor o problema para o qual tal aplicação foi criada. Dessa forma, é muito importante que as organizações responsáveis por estas aplicações estejam atentas às mudanças do mercado e ao surgimento de novas e melhores tecnologias para aprimorar ou até mesmo evoluí-las.

Entretanto, a adição de novas funcionalidades ou recursos para estas aplicações podem acarretar outros problemas que muitas vezes as organizações não levam em conta, como, por exemplo, o aumento do consumo de *hardware*, já que com uma maior complexidade envolvida no *software*, é esperado a utilização de mais recursos físicos. Quando lidamos com aplicações de grande porte, comumente chamadas de monolitos, é preciso saber dimensionar toda aplicação como uma única peça; porém, quando lidamos com serviços menores, em que cada um tem uma única responsabilidade, é possível escalar apenas aquele serviço que precisa ser dimensionado, possibilitando ainda executar outras partes do sistema em um *hardware* com menor capacidade de processamento (NEWMAN, 2015).

Dessa forma o artigo visa a apresentação da arquitetura monolítica e da arquitetura de microserviços, ambas utilizadas para o desenvolvimento de *softwares*. A partir de um estudo de caso é demonstrado os problemas que uma grande empresa de varejo enfrenta por causa do aumento de acessos em seu site, especialmente em seu *e-commerce*, construído atualmente com a arquitetura monolítica e sua tentativa de migração para a arquitetura de microserviços. Por fim, são apresentados quais os ganhos, os riscos e os problemas que podem ocorrer com essa evolução do sistema.

2 TECNOLOGIAS E FERRAMENTAS UTILIZADAS

Para o desenvolvimento do presente trabalho é necessário o conhecimento de algumas das principais características que diferem a arquitetura de microserviços da arquitetura

monolítica, bem como o conhecimento de alguns termos técnicos. Dessa forma, nas páginas abaixo essas características são apresentadas e explicadas para uma melhor compreensão do leitor.

2.1 Arquitetura de microserviços

A arquitetura de microserviços é um modelo que estrutura toda aplicação em uma coleção de serviços que tem como características facilidade de manutenibilidade e testabilidade, baixo acoplamento entre os serviços, implantação independente, comumente chamado de *"deploy" e* organização em torno dos recursos de negócios, que podem ser construídos por times pequenos e utilizar diferentes tecnologias entre eles (RICHARDSON, 2015).

Segundo Familiar (2015), o termo "micro" presente em microserviços é uma referência ao escopo da funcionalidade que aquele serviço provê. Idealmente, um microserviço deve ser responsável por uma parte específica de todo o sistema, tendo seu próprio banco de dados e fornecendo um recurso da aplicação a partir de uma API (*Application Programum Interface)*, que pode ser entendida como "pontes" para conectar os sistemas. Dessa forma, facilita a troca de informações por mensagem entre os serviços, de forma que cada API define quais serão seus *endpoints* visíveis da aplicação.

Os microserviços têm várias propriedades e comportamentos que os diferenciam das demais arquiteturas, afetando diversos aspectos do desenvolvimento, desde a estrutura do time que irá construí-lo até a organização do código fonte e o controle da integração até a implantação do sistema. Algumas dessas propriedades e comportamentos relatam o fato dos microserviços serem autônomos e isolados, isto é, são unidades independentes de funcionalidade que têm dependências de outros serviços fracamente acopladas e são projetados, desenvolvidos, testados e liberados independentemente. Outras propriedades que caracterizam os microserviços são a elasticidade, resiliência e responsividade, porque eles são reutilizados em muitas soluções diferentes e, portanto, devem ser capazes de dimensionar adequadamente dependendo do cenário que serão utilizados. Além disso, eles devem ser tolerantes a falhas e fornecer um prazo razoável para recuperação se algo der errado. Por fim, eles precisam ser responsivos, fornecendo desempenho razoável independente do cenário de execução (FAMILIAR, 2015).

Familiar (2015) aponta ainda outras características essenciais dos microserviços, como a necessidade de terem contratos de dados para definirem como a interação com o serviço é alcançado e serem orientados à mensagem e programáveis, dado que eles dependem de APIs. Os contratos de dados definem qual a estrutura que essas mensagens devem ser enviadas e recebidas, como por exemplo, o formato *JSON,* um modelo de dados que armazena e transmite informações no formato de texto, utilizando um padrão "chave: valor". Na figura abaixo é apresentado uma mensagem no formato JSON, sendo as informações em verde as chaves ou atributos e as em amarelo seu valor.

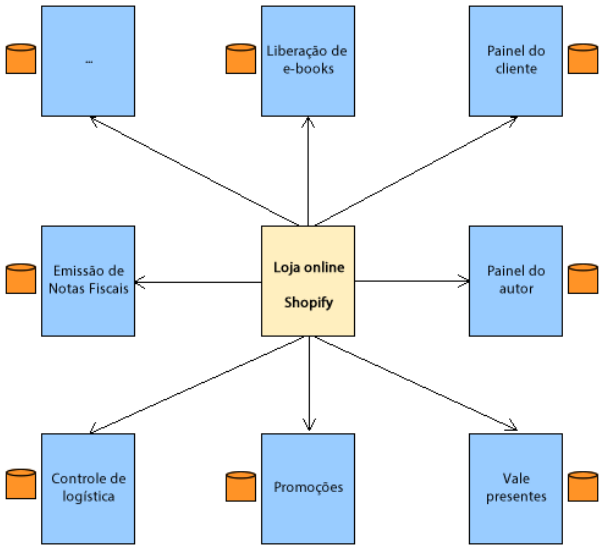
Figura 1 - Exemplo de uma mensagem enviada no formato JSON.

Fonte: Elaborado pelos autores, 2020

Por fim, duas outras características essenciais dos microserviços são a capacidade deles serem adaptáveis a diferentes configurações, como por exemplo, prover seus serviços para mais de uma API e diferentes contratos de dados, além de serem totalmente automatizados durante todo seu ciclo de vida, desde o design até a implementação do serviço (FAMILIAR, 2015).

A partir da Figura 2 é possível observar uma representação esquemática de uma aplicação desenvolvida com a arquitetura de microserviços, em que cada funcionalidade da aplicação é representada por um microserviço específico, com sua própria base de dados, que se comunica com todos os outros microserviços da aplicação.

Figura 1 - Representação esquemática de uma aplicação construída com a arquitetura de microserviços



Fonte: Almeida, 2015.

De acordo com Almeida (2015), um dos grandes benefícios da arquitetura de microserviços é a possibilidade de diminuir a complexidade dentro da aplicação, porque cada serviço tem uma base de código menor em comparação à arquitetura monolítica. Desse modo torna-se compreensível tanto para os membros do time, que são responsáveis por seu desenvolvimento, quanto para outras pessoas que possam ter contato com o código.

Outro ponto benéfico para a arquitetura de microserviços é que, diferentemente da arquitetura monolítica, a aplicação não tem um único ponto de falha. Dessa forma, se algum microserviço apresentar algum tipo de problema, o restante da aplicação não será afetado por isso, e apenas o serviço que apresenta tal problema não estará disponível para o usuário (ALMEIDA, 2015).

Familiar (2015) aponta ainda que outra grande vantagem do microserviço é o fato de ser aberto, uma vez que ele deve ser desenvolvido para expor sua funcionalidade a partir de uma API endereçável na rede, possibilitando que a equipe de desenvolvimento tenha autonomia para escolher as tecnologias envolvidas com aquele microserviço, como a linguagem de programação, por exemplo.

Apesar de resolver problemas apresentados por outras arquiteturas, o microserviço não pode ser vistos como uma solução para todas as aplicações e nem para todas as situações, pois ele também apresenta desvantagens e complexidade em sua implementação.

Uma das principais dificuldades apontadas por Familiar (2015) é a organização para o suporte dos microserviços, uma vez que torna-se necessário uma equipe madura em relação aos conhecimentos de operações, já que é preciso gerenciar vários serviços que se comunicam de diversas formas diferentes.

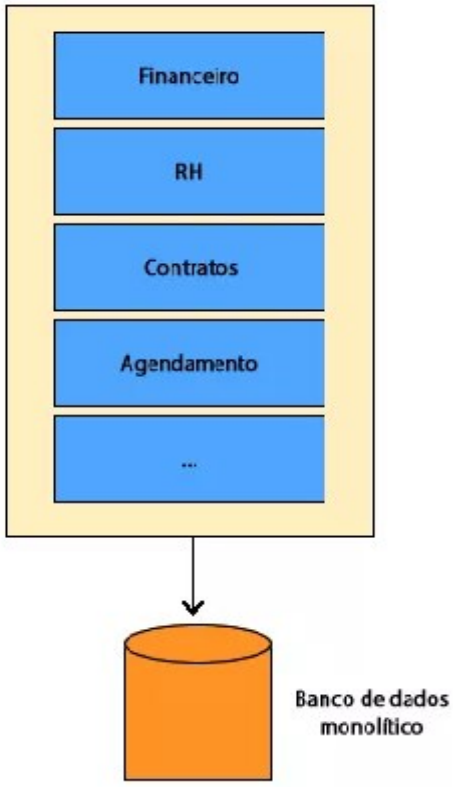
Outro ponto levantado pelo autor é a complexidade que envolve os testes dos microserviços, porque além dos testes unitários, TDD (*Test-driven Development*), que são realizados antes mesmo da escrita do código da aplicação, entre outros, ainda é preciso testar a implementação de cada serviço, a comunicação com a sua base de dados e realizar os testes de protocolo. Neles as APIs recebem as requisições, usualmente o HTTP (*Hypertext Transfer Protocol*), utilizado amplamente na internet para a comunicação entre os sistemas de informação de hipermídia, distribuídos e colaborativos.

2.2 Arquitetura monolítica

O desenvolvimento de aplicações utilizando a arquitetura monolítica pode ser compreendido como um empacotamento de todos os componentes do servidor em uma única unidade. Normalmente ela é construída em três partes principais: i) a *client-side*, também conhecida como a interface de usuário, que consiste em páginas utilizando HTML e Javascript, por exemplo; ii) o banco de dados, que consiste em tabelas inseridas em um sistema de gerenciamento de banco de dados comum, e; iii) o *server-side,* que é a aplicação em si, ou seja, a parte responsável por suportar as requisições HTTP, executar todo o domínio lógico, acessar o banco de dados, popular as informações das páginas apresentadas aos usuários, entre outras funções (LEWIS & FOWLER, 2014).

A Figura 3 exemplifica a representação de uma aplicação construída utilizando a arquitetura monolítica, na qual todas as funções do sistema estão dentro de uma única aplicação e acessam o mesmo banco de dados.

Figura 2 - Representação esquemática de uma aplicação construída com a arquitetura monolítica.



Fonte: Almeida, 2015.

Segundo Richardson (2014), a arquitetura monolítica é um padrão utilizado no desenvolvimento de aplicações corporativas, porque seu funcionamento é compatível com aplicações normalmente pequenas. Ainda segundo o autor, tanto as *IDES*, que são ferramentas utilizadas para o apoio do desenvolvimento de software durante a escrita do código, como outras ferramentas utilizadas durante o desenvolvimento são direcionadas para a construção de uma única aplicação e não diversos microserviços. Ele ainda ressalta que, quando se deseja testar uma aplicação monolítica é preciso apenas executar o aplicativo e para a sua implementação basta copiar a unidade de implantação para uma máquina com um servidor adequado (RICHARDSON, 2014).

Quando as aplicações passam a escalar e se tornam grandes demais, com muitas funcionalidades, a arquitetura monolítica começa a apresentar seus principais problemas, pois, como aponta Richardson (2014), uma grande aplicação monolítica dificulta o entendimento e a manutenção do código para os desenvolvedores, além de ser um obstáculo implementar novas funcionalidades, uma vez que é preciso fazer o *build* e implantar a aplicação inteira,

atividade que pode ser complexa e arriscada e que consome muito tempo e exige o envolvimento de muitos desenvolvedores e diversos ciclos de testes.

Outro problema da arquitetura monolítica levantada por Almeida (2015), é que tais aplicações apresentam um único ponto de falha, ou seja, se ter, por exemplo, uma funcionalidade para cadastrar clientes e ela ficar fora do ar por algum tipo de problema, todo o restante da aplicação também será afetado, não sendo possível acessar nenhuma de suas funcionalidades.

3 PROCEDIMENTOS METODOLÓGICOS

O presente artigo é caracterizado por um estudo de caso com abordagem qualitativa, tendo como objetivo apresentar os problemas enfrentados por uma empresa em plataforma digital de vendas, atualmente construída com a arquitetura monolítica, levando-a a iniciar o processo de migração para a arquitetura de microserviços.

Segundo Patel (2020), um estudo de caso pode ser compreendido como uma pesquisa ampla e profunda de um ou poucos objetos de estudo, que permitem o conhecimento amplo e detalhado de uma temática. Seu principal objetivo é servir de base e de referência para estudo e investigações de outras pessoas sobre a mesma temática.

A escolha dessa metodologia vai ao encontro do objetivo estipulado, apresentando de forma clara e simplificada a comparação entre as duas arquiteturas descritas e suas vantagens e desvantagens. Dessa maneira, contribui para a tomada de decisão da migração de uma arquitetura para outra.

Para o entendimento do problema apresentado neste trabalho, a abordagem qualitativa foi adotada a fim de compreender a partir de dados narrativos como o produto se comporta ao invés de medi-lo. Assim, foi possível conseguir uma grande quantidade de dados que auxilia no entendimento e na resolução do problema, como, por exemplo, qual a principal necessidade do produto tanto para seu dono, como para seu consumidor. Estes dados foram obtidos a partir de entrevistas realizadas de maneira individual e também com grupos focais que atuam diretamente no produto de software estudado.

3.1 O caso Global Varejo

Diante a atual situação mundial vivenciada por causa da pandemia do novo Coronavírus, readaptações diárias tornam-se necessárias para a diminuição dos impactos ocasionados, sendo a tecnologia uma aliada ainda mais poderosa e fundamental para esse propósito.

A pandemia vem afetando tanto pessoas quanto grandes empresas e corporações, não sendo diferente para a companhia Global Varejo, empresa do comércio varejista brasileira que comercializa diversos produtos, desde móveis até eletrônicos.

Um dos impactos ocasionados na empresa foi a diminuição das vendas em suas lojas físicas, sendo necessário voltar todos os esforços para a sua plataforma de comércio digital. Com o crescente número de acessos em seu site, diversos problemas passaram a surgir por causa da arquitetura utilizada nesaa aplicação, como recursos constantemente indisponíveis, impossibilitando os clientes de efetuarem suas compras, além de tráfego de informações com lentidão.

A plataforma digital da Global Varejo foi construída há algum tempo baseada na arquitetura monolítica, que naquele momento era uma das arquiteturas mais conhecidas e utilizadas em diversas aplicações. Porém, diante de toda evolução tecnológica, diversas melhorias surgiram e ganharam espaço, como a arquitetura de microserviços que passou a ser amplamente utilizada em sistemas de *e-commerce*.

Grande parte dos problemas enfrentados pela plataforma digital estão relacionados à arquitetura da aplicação, como a dificuldade em escalar o serviço por conta da necessidade de copiar toda a base do código a cada vez que se deseja aumentar sua capacidade, além da existência de um único ponto de falha, fazendo com que toda a aplicação fique indisponível caso algum recurso do sistema esteja enfrentando problemas. Vale ressaltar ainda que os problemas mencionados anteriormente estão intrinsecamente ligados, já que uma sobrecarga no sistema devido a falta de escalonamento pode levar à queda de um serviço da aplicação.

Como forma de solucionar esses problemas a Global Varejo passou a analisar a migração de parte de sua aplicação para a arquitetura de microserviços com o intuito de melhorar seus serviços disponibilizados para o usuário final, mas sem interromper suas atividades.

O processo de migração de aplicações monolíticas para microserviços não tem um método único e fechado, porque existem particularidades que não podem ser previstas. Sendo assim, tal processo um grande desafio. Como afirma Freire e col. (2019) é uma tarefa complexa decidir o que migrar, quando migrar e como migrar do monolítico para o microserviço, principalmente em aplicações que já estão em produção.

Assim, para a migração da plataforma digital da Global Varejo foram adotados alguns aspectos da metodologia proposta por Freire e col. (2019). Tal metodologia adota uma abordagem híbrida, na qual o sistema monolítico é mantido em produção enquanto os microserviços vão sendo desenvolvidos para substitui-lo, permitindo que a aplicação seja retomada para o estado inicial, caso algum problema seja apresentado por um microserviço que substituiu parte do monolítico.

De maneira geral, essa metodologia consiste em utilizar os conceitos de *Reflection* (capacidade de obter metadados sobre o próprio programa compilado) e Programação Orientada à Aspectos (AOP), permitindo desviar as chamadas realizadas a algum módulo do monolítico para os microserviços desenvolvidos. Desse modo, essas chamadas são transformadas em requisições REST (*Representational State Transfer)*, um conjunto de restrições utilizadas para que as requisições HTTP atendam as diretrizes definidas na arquitetura.

4 RESULTADOS E DISCUSSÃO

Para a migração do sistema monolítico para o de microserviços, algumas partes foram escolhidas. Priorizaram-se os serviços mais essenciais da aplicação, como o fluxo de compras, que deve adicionar um produto ao carrinho do cliente, validar os dados de pagamento e finalizar a venda. Como a própria metodologia utilizada e proposta por Freire e col. (2019) sugerem, essa migração deve ocorrer de forma desacoplada, reversa, com zero *dowtime* e micro-banco, ou seja, não é preciso modificar ou desligar o código do monolítico. Cada microserviço passa a ter sua própria base de dados e, caso identificado algum erro de desenvolvimento no microserviço, épossível reverter a aplicação para o monolítico.

Com os microserviços prontos para uso, as requisições das chamadas feitas ao monolítico passam a ser interceptadas pela API, utilizando a programação orientada a Aspecto, sendo encaminhadas ao microserviço correspondente via REST, como prevê a metodologia proposta por Freire e col. (2019). Além disso, ainda de acordo com a

metodologia, o micro-banco do microserviço é criado assim que a primeira requisição for realizada a ele.

Com as funcionalidades do fluxo de compras sendo direcionadas aos microserviços e não mais ao monolítico, foi possível notar uma melhora no consumo de CPU (Unidade Central de Processamento) ao comparar as duas arquiteturas com um mesmo número de requisições. Enquanto o consumo de CPU do monolítico chega máximo, o microserviço é capaz de continuar atendendo normalmente as requisições (FREIRE e col., 2019).

Dessa forma, como parte das funções são removidas do monolítico e direcionadas ao microserviço, é possível receber um número maior de requisições tanto nos serviços não migrados do monolítico, como nos microserviços, diminuindo assim, a probabilidade de a aplicação gerar falhas diante inúmeros acessos dos clientes.

Outro ponto de possível avaliação é a possibilidade em escalar os microserviços conforme necessário, uma vez que esse serviço precisa apenas ser replicado, sendo possível aumentar sua capacidade quando desejado, como, por exemplo, em uma determinada ação ou época do ano que o número de vendas é maior.

Os estudos realizados por Freire e col. (2019), mostram que utilizar essa metodologia de migração não altera o tempo de resposta das requisições, porque elas precisam primeiramente chegar até o monolítico para serem interceptadas pela API e direcionadas para o microserviço correspondente. A partir de testes realizados na aplicação da Global Varejo foi possível observar este mesmo comportamento, isso porque, a chamada realizada ao microserviço ocorre por AOP, ou seja, primeiramente a requisição é recebida pelo monolítico para então ser desviada ao microserviço responsável.

Por fim, outro ponto que merece destaque enquadra-se no fato da indisponibilidade do microserviço não afetar todo o restante da aplicação, já que apenas ele não ficará disponível. Além disso, caso ocorra esta indisponibilidade, a metodologia de migração proposta por Freire e col. (2019), permite religar esse serviço no monolítico caso seja desejado, até que o problema seja resolvido no microserviço.

5 CONSIDERAÇÕES FINAIS

Levando em conta o objetivo estabelecido que visava migrar um sistema com arquitetura monolítica para arquitetura de microserviços, demonstrando as vantagens e desvantagens desse processo, observamos a não existência de uma metodologia fechada e que

pode ser considerada ideal para todas as situações. Porém, a proposta de Freire e col.(2019) apresentada neste artigo provê uma boa saída para essa situação, não sendo preciso alterar e desligar o monolítico para que os microserviços funcionem corretamente.

Uma aplicação em arquitetura de microserviços reflete diversos benefícios, como escalabilidade dos serviços, baixo acoplamento e independência, de modo que um problema apresentado passa a não afetar o restante da aplicação. No entanto, é preciso uma alta organização e conhecimento da equipe responsável por seu desenvolvimento, uma vez que gerenciar diversos serviços que se comunicam de formas diferentes e lidar com testes complexos não é tarefa fácil.

Construir aplicações com a arquitetura de microserviço não é uma tarefa fácil e necessita de conhecimentos prévios, dessa forma, uma boa estratégia pode ser iniciar o desenvolvimento com uma arquitetura monolítica, tendo em vista o não conhecimento do tamanho e da complexidade que tal aplicação pode atingir. Porém, ao decorrer do processo, observando a grandeza em que encontra-se, o ideal seria ocorrer a migração de arquitetura, facilitando essa tarefa.

Com o crescimento exponencial de novas tecnologias e a consolidação de tantas outras, é preciso atentar-se as melhores soluções para manter sua competitividade no mercado, garantindo assim, a entrega do melhor produto e experiência ao usuário final.

A principal dificuldade encontrada no caso abordado foi a migração da arquitetura, uma vez que nos deparamos com a falta de uma teoria fechada e ideal para tal, além da criticidade da parte do sistema migrado, uma vez que o *e-commerce* encontra-se como a principal fonte de vendas da empresa, não podendo apresentar falhas que prejudiquem os lucros.

REFERÊNCIAS

ALMEIDA, A. Arquitetura de microserviços ou monolítica?. Caelum, 2015. Disponível em: <https://blog.caelum.com.br/arquitetura-de-microservicos-ou-monolitica/>. Acesso em: 8 abr. 2020.

FAMILIAR, B. Microservices, IoT, and Azure: Leveraging DevOps and Microservice Architecture to Delicer SaaS Solutions: 1.ed. New York: Apress Media, 2015.

FREIRE, F. A.; SAMPAIO, A.; MEDEIROS, O.; MENDONÇA, N. , Migração de Sistemas Monolíticos para Microserviços: Uma Abordagem Híbrida sem Downtime., In: WORKSHOP EM CLOUDS E APLICAÇÕES (WCGA), 17. , 2019, Gramado. Anais do XVII Workshop em Clouds e Aplicações. Porto Alegre: Sociedade Brasileira de Computação, sep. 2019 . p. 1-14. DOI: https://doi.org/10.5753/wcga.2019.7590.

NEWMAN, S. Building Microsservices: Designing fine-grained systems. 1. ed. O'Reilly Media, 2015.

PATEL, N. Estudo de caso: O que é, exemplos e como fazer. **NEILPATEL**, 2020. Disponível em: < https://neilpatel.com/br/blog/como-fazer-um-estudo-de-caso/>. Acesso em: 10 jun. 2020.

RICHARDSON, C. Microservices: Decomposing Applications for Deployability and Scalability. InfoQ, 2014. Disponível em: <https://www.infoq.com/articles/microservices-intro/?itm_source=infoq&itm_campaign=user_page&itm_medium=link>. Acesso em: 15 abr. 2020.

RICHARDSON, C. Introduction to Microservices. NGINX, 2015. Disponível em: <https://www.nginx.com/blog/introduction-to-microservices/>. Acesso em: 8 abr. 2020.