ANÁLISE DE UM ESTUDO DE CASO PARA COMPARAR AS ARQUITETURAS MONOLÍTICA E DE MICROSERVIÇOS

ANALYSIS OF A CASE STUDY TO COMPARE MONOLITHIC AND MICROSERVICE ARCHITECTURES

Mateus Pegrucci – mpegrucci1@gmail.com
Faculdade de Tecnologia (Fatec) – Taquaritinga – SP – Brasil

Fernando Tiosso – fernando.tiosso@fatectq.edu.br
Faculdade de Tecnologia (Fatec) – Taquaritinga – SP – Brasil

Helena Macedo Reis – helena.macedo@ufpr.br
Universidade Federal do Paraná (UFPR) – Jandaia do Sul – PR – Brasil

DOI: 10.31510/infa.v17i2.1029
Data de publicação: 18/12/2020

RESUMO

Este artigo propõe-se em avaliar duas arquiteturas de software, monolítica e microserviços, a fim de encontrar a que melhor se enquadra para um determinado propósito. Por meio de pesquisas bibliográficas, consulta de artigos e livros, algumas características foram propostas para avaliá-las junto ao desenvolvimento de um estudo de caso envolvendo dois projetos, cada qual utilizando um tipo de arquitetura, evidenciando pontos positivos e negativos dessa utilização com o objetivo de auxiliar o desenvolvedor de software em uma futura tomada de decisão mediante um contexto previamente estabelecido.

Palavras-chave: Arquitetura. Monolítica. Microserviços. Comparativo.

ABSTRACT

This article proposes to evaluate two software architectures, monolithic and microservices, in order to find the one that best fits for a given purpose. Through bibliographic research, consultation of articles and books, some characteristics were proposed to evaluate them together with the development of a case study involving two projects, each using a type of architecture, showing positive and negative points of this use with the objective to assist the software developer in future decision making through a previously established context.

Keywords: Architecture. Monolithic. Microservices. Comparative.

1 INTRODUÇÃO

Ao ser analisado o cenário atual tecnológico, principalmente no desenvolvimento de software, encontram-se diversas complexidades relacionadas à leitura, qualidade e

manutenção dos códigos escritos, visto que mudanças constantes em plataformas e ferramentas vêm ocorrendo em grande escala.

A arquitetura utilizada na construção do software pode contribuir para melhorar tais características. Segundo Shaw & Garlan (1996), uma arquitetura de software é responsável por definir a organização do sistema e manter os relacionamentos entre os componentes. Portanto, a arquitetura comporta-se como o esqueleto do sistema, por isso é essencial todo projeto ter uma arquitetura bem implementada para que o mantenha organizado e objetivo, visando facilitar seu entendimento por qualquer *stakeholder*[1] ao observá-lo.

Atualmente, existem vários tipos de arquitetura de software, e, dentre eles, podem ser destacados dois: monolítica e microserviços. Sabendo que cada arquitetura possui um padrão próprio junto a uma organização totalmente específica, este trabalho abordará as características fundamentais de tais arquiteturas.

A arquitetura monolítica é a mais utilizada no desenvolvimento de softwares no momento, onde todo o sistema é feito dentro de um único escopo, podendo ser dividido três partes: *front-end*, onde existe a interação com o usuário, *back-end*, onde é tratado toda a parte negocial bem como o processamento de requisições e uma base de dados (NHIMI, 2016).

Já a arquitetura de microserviços, vem sendo cada vez mais utilizada por conta do conceito de *Cloud Computing* (Computação em nuvem), possibilitando dividir o sistema em vários microserviços independentes que, no final, irão se tornar parte de um único sistema (BARTIÉ, 2002).

Visando demonstrar de forma mais prática esses tipos de arquitetura, este trabalho também apresentará o desenvolvimento de dois projetos, permitindo a comparação entre as duas arquiteturas de software: monolítica e microserviços.

Assim, com o objetivo de explorar essa conjuntura, o presente trabalho foi desenvolvido em 6 (seis) seções, sendo estas: a Seção 2 (dois) que apresenta a metodologia de pesquisa utilizada, a Seção 3 (três) que expõe a fundamentação teórica, descrevendo as arquiteturas monolítica e de microserviços, a Seção 4 (quatro) que apresenta um estudo de caso prático abordando diferenças entre as arquiteturas, a seção 5 (cinco) que mostra um quadro comparando as principais características de cada arquitetura e, por fim, a seção 6 (seis) que expressa as considerações finais do artigo.

[1] *Stakeholder:* qualquer indivíduo ou organização que, de alguma forma, é impactado pelas ações de determinada empresas (CONTENT, 2020).

2 METODOLOGIA DE PESQUISA

Primeiramente, o desenvolvimento deste trabalho contou com uma revisão bibliográfica que possibilitou adquirir conhecimentos fidedignos sobre o assunto abordado, visando embasar o conteúdo explanado pelo autor (GIL, 1999). Além disso, o trabalho também contou com a utilização do método de estudo de caso, que tem como objetivo validar alguns pontos, de forma prática, sobre a pesquisa realizada (YIN, 2005, p.32).

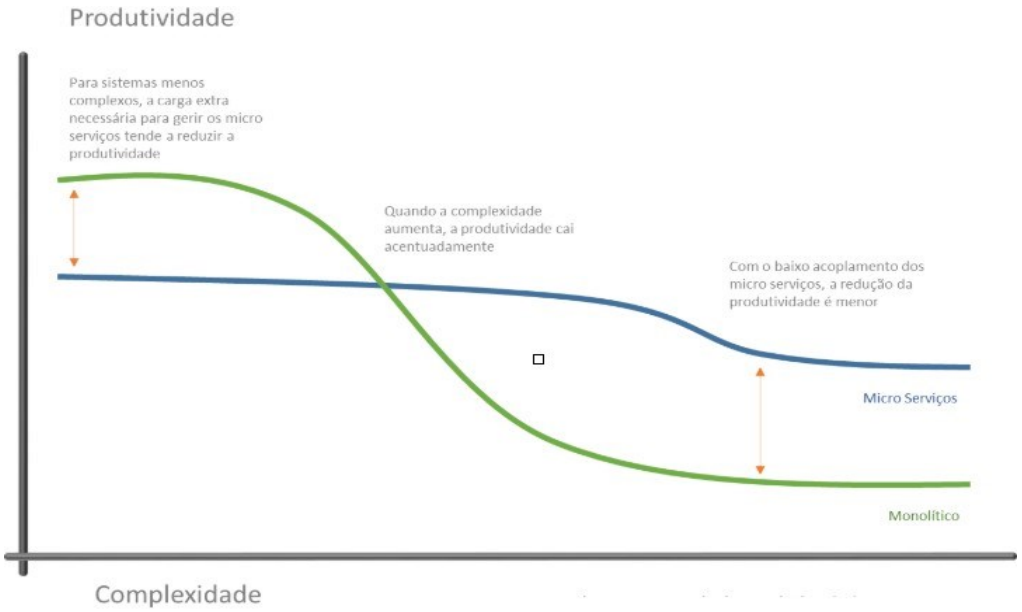
Portanto, como resultado da revisão bibliográfica realizada, a seção a seguir apresenta os conceitos inerentes aos tipos de arquitetura analisados e discutidos neste projeto.

3 ARQUITETURA MONOLÍTICA E DE MICROSERVIÇOS

Uma arquitetura compreende-se por um conjunto de classes agrupadas, originando componentes ou pacotes com funcionalidades definidas em um sistema, ressaltando a importância da sua escolha para uma determinada necessidade (WAZLAWICK, 2011).

Conforme mostra a Figura 1, sistemas que tendem ser mais complexos no acoplamento de seus módulos podem ter um comprometimento da produtividade quando adotada a arquitetura de monolítica. Por outro lado, a arquitetura de microserviços é mais estável para este cenário, porém, quando o sistema tem um baixo grau complexidade, a implementação desse sistema tende a ser mais complexo por conta da carga extra necessária para gerir os serviços disponíveis.

Figura 1 - Comparação da arquitetura monolítica com a arquitetura de microserviços envolvendo quesitos de produtividade e complexidade



Fonte: MACHADO (2017)

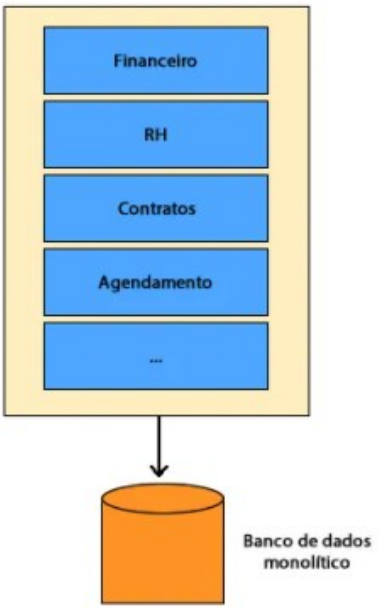
Com o objetivo de conhecer ambas as arquiteturas e entender suas aplicabilidades, uma breve explicação será realizada nos tópicos subsequentes.

3.1 ARQUITETURA MONOLÍTICA

A arquitetura monolítica, em termos gerais, disponibiliza suas funcionalidades no mesmo código fonte e/ou unidade de instalação, promovendo uma maior dependência entre os serviços no ambiente de execução e o compartilhamento de artefatos comuns entre os componentes do sistema (WOODS, 2015).

A Figura 2 mostra um exemplo da utilização da arquitetura monolítica. Nela, é possível observar que todas as funcionalidades de um determinado sistema estão dentro de um único projeto, que se comunica com apenas um banco de dados monolítico.

Figura 2 - Representação de uma arquitetura monolítica.



Fonte: ALMEIDA (2015)

Após a apresentação da arquitetura monolítica, faz-se necessário compreender a arquitetura de microserviços, exposta na seção seguinte.

3.2 ARQUITETURA DE MICROSERVIÇOS

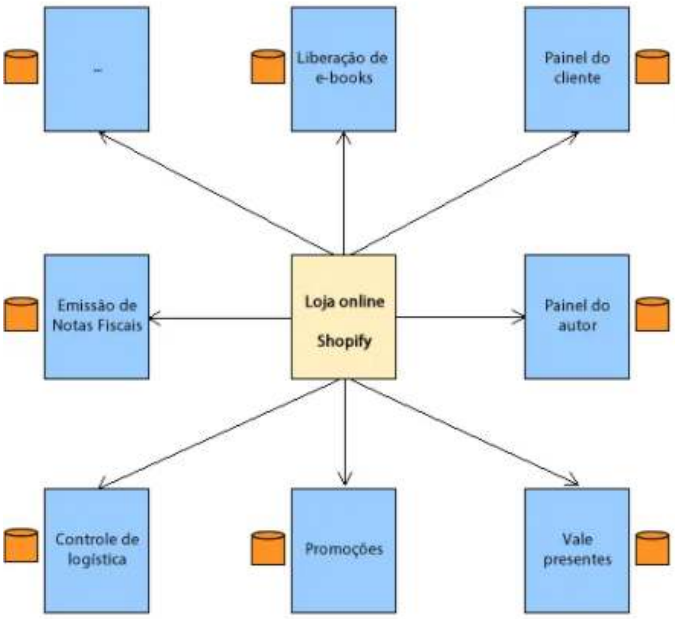
De acordo com Fowler (2014), o termo "*Microservice Architecture*" surgiu nos últimos anos com o intuito de contemplar aplicações de software independentes, sem a necessidade de depender de outra aplicação, podendo-se dizer, em características comuns, que essa arquitetura é dividida em vários projetos separados, mais comumente chamados de serviços.

A arquitetura de microserviços começou a ser mais evidenciada com a chegada da cultura DevOps, que se caracteriza por serviços que funcionam de forma independente que ao serem unificados originam uma única aplicação (MACHADO, 2017).

No entanto, arquiteturas baseadas em microserviços também utilizam bibliotecas compartilhadas, e a atualização de uma dessas bibliotecas pode requerer, também, a atualização dos componentes que as consomem (RICHARDSON, 2016).

A Figura 3 mostra um exemplo da utilização da arquitetura de microserviços. Nela, é possível observar todas as funcionalidades separadas em projetos (serviços) e que cada um tem seu respectivo banco de dados.

Figura 3 - Representação de uma arquitetura de microserviços.



Fonte: ALMEIDA (2015)

Após essa breve interação com as arquiteturas monolítica e de microserviços, propõe- se a execução de um estudo de caso comparando-as, com o objetivo de ressaltar suas características de forma prática, conforme descrevem as próximas seções.

4 ESTUDO DE CASO

O presente estudo de caso analisou, de forma prática, diferenças entre uma arquitetura monolítica e uma arquitetura de microserviços no desenvolvimento de dois projetos, cada qual utilizando uma das arquiteturas junto à linguagem de programação Java com Spring Framework.

Em ambos os projetos, desenvolveu-se um cadastro de funcionário demonstrando algumas de suas principais funcionalidades, como inserção, seleção, alteração e exclusão, mais comumente conhecidas pelo acrônimo CRUD (*Create, Retrieve, Update e Delete*). Na construção do Projeto 1, utilizou-se a arquitetura monolítica, enquanto o Projeto 2 fez uso da arquitetura de microserviços.

Ressalta-se que a camada de apresentação *(front-end)* das duas aplicações são análogas e, por isso, não serão discutidas neste experimento. Assim, os próximos parágrafos

apresentarão as diferenças entre as camadas que compõe o *back-end* de cada projeto.

A Figura 4 mostra a camada responsável por receber as requisições da camada de apresentação da arquitetura monolítica. Nela, observa-se a classe *PersonController,* mais conhecida como *Endpoint,* que se comunica com a camada de apresentação, ou seja, um caminho que permite o acesso de uma determinada funcionalidade, bem como as operações que poderão ser executadas, neste caso, a seleção, representada pela operação *getAll*.

Figura 4: Camada responsável por se comunicar com a camada de apresentação (*front-end*) na arquitetura monolítica



```
@RequestMapping(value="/person")
@RestController
public class PersonController {

    @Autowired
    private PersonService service;

    @GetMapping(value="/", produces=MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<List<Person>> getAll() {
        return ResponseEntity.ok(service.getAll());
    }

}
```

Fonte: próprio autor

Para a arquitetura monolítica, apenas esse tipo de classe é necessário para a comunicação entre a camada de apresentação (front-end) e a camada de negócio (back-end), pois todas as funcionalidades do sistema estão dentro de apenas um projeto, fazendo com que a camada de apresentação utilize apenas um único caminho de comunicação.

Já, na arquitetura de microserviços, há a necessidade de se ter um projeto principal responsável por receber cada requisição e encaminhá-la para seu respectivo microserviço, pois a camada de apresentação pode requisitar diferentes serviços utilizando diferentes caminhos. Por conseguinte, com o objetivo aprimorar a comunicação entre a camada de apresentação (*front-end*) e camada de negócio (*back-end*) da arquitetura de microserviços, duas novas camadas são acrescentadas no processo: *Client* e *Interface*.

A Figura 5 apresenta a interface *Client*, alocada junto ao projeto principal com o objetivo de encaminhar a requisição oriunda da camada de apresentação para o respectivo microserviço.

Figura 5: Camada responsável por se comunicar com a camada de apresentação (front-end) e redirecionar a requisição para o microserviço requisitado na arquitetura de microserviços

```
package br.com.microservices.client.funcionario

import ...

@FeignClient(name = "funcionario", url = "http://localhost:4200/funcionario")
public interface FuncionarioClient extends FuncionarioApi {

}
```

Fonte: próprio autor

Já a Figura 6, apresenta a interface responsável por expor o microserviço para que possa receber requisições da camada de apresentação, garantindo acesso aos seus *Endpoints*, bem como suas implementações por meio das assinaturas de contratos, assegurando o baixo acoplamento entre os objetos.

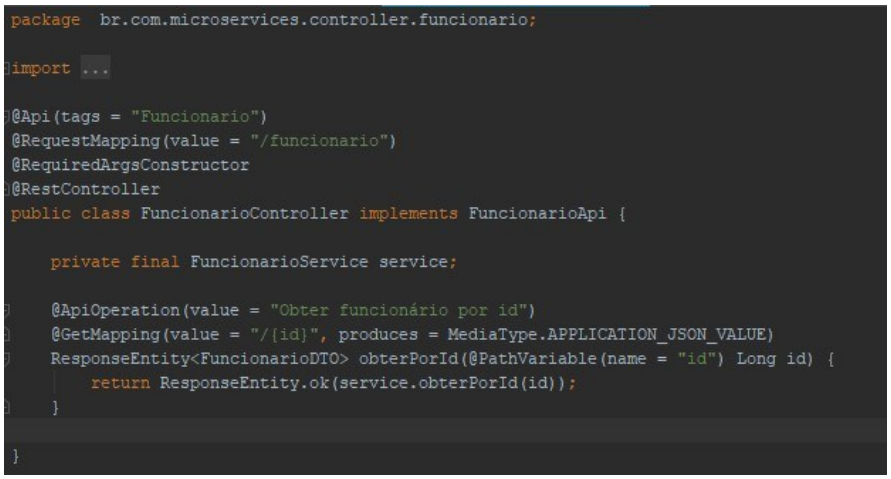
Figura 6: Camada que recebe a chamada do *Client* na arquitetura de microserviços

```
@Api(tags = "Funcionario")
@RequestMapping(value = "/funcionario")
public interface FuncionarioApi {

    @ApiOperation(value = "Obter funcionário por id")
    @GetMapping(value = "/{id}", produces = MediaType.APPLICATION_JSON_VALUE)
    ResponseEntity<FuncionarioDTO> obterPorId(@PathVariable(name = "id") Long id);

}
```

Fonte: próprio autor

A Figura 7 apresenta a camada responsável por receber as requisições que foram encaminhadas pelo *Client* de cada microserviço. A partir dessas classes, todos os procedimentos utilizados no padrão de desenvolvimento *Model-View-Controller* (MVC) são iguais nas duas arquiteturas, fazendo uso de uma classe onde realiza-se a implementação das lógicas negociais do sistema e outra classe que permite a comunicação da aplicação com o banco de dados.

Figura 7: Camada responsável por receber as requisições encaminhadas pelo *Client* de cada microserviço na arquitetura de microserviços



```
package br.com.microservices.controller.funcionario;

import ...

@Api(tags = "Funcionario")
@RequestMapping(value = "/funcionario")
@RequiredArgsConstructor
@RestController
public class FuncionarioController implements FuncionarioApi {

    private final FuncionarioService service;

    @ApiOperation(value = "Obter funcionário por id")
    @GetMapping(value = "/{id}", produces = MediaType.APPLICATION_JSON_VALUE)
    ResponseEntity<FuncionarioDTO> obterPorId(@PathVariable(name = "id") Long id) {
        return ResponseEntity.ok(service.obterPorId(id));
    }

}
```

Fonte: próprio autor

Como consequência do aprendizado proporcionado pela condução do estudo de caso exposto, foi possível sugerir algumas características que podem ser comparadas para as duas arquiteturas, visando facilitar a compreensão da sua utilização em um determinado cenário, como pode ser observado na seção a seguir.

5 QUADRO COMPARATIVO ENTRE AS ARQUITETURAS

Examinando-se mais detalhadamente os dois tipos de arquiteturas abordadas neste artigo, o Quadro 1 foi criado com o objetivo de evidenciar e comparar as características comuns dessas arquiteturas, pretendendo prover um maior entendimento das suas principais diferenças para assistir o processo de tomada de decisão, por parte de um desenvolvedor de software, no momento da definição de qual arquitetura utilizar em seu projeto.

Quadro 1 - Comparativo entre algumas características comuns das arquiteturas

Característica	**Arquitetura Monolítica**	**Arquitetura Microserviços**
Escalabilidade	Difícil escalabilidade, pelo fato da aplicação se encontrar dentro de um único projeto (unidade instalável).	Fácil escalabilidade por meios dos microserviços mais requisitados.

Heterogeneidade Tecnológica	Não factível, pois o desenvolvimento do sistema tem foco em um framework previamente definido.	Factível, pois o desenvolvimento do sistema conta com a disponibilidade das melhores tecnologias viáveis para o desenvolvimento de uma tarefa.
Resiliência	Baixa resiliência, pois no caso de uma falha, toda aplicação poderá ser impactada.	Alta resiliência, pois somente o microserviço que apresentou falha vai ser impactado.
Produtividade inicial	Alta, pois apresenta-se uma menor dificuldade no desenvolvimento inicial do projeto, por ter apenas uma aplicação para ser gerenciada.	Baixa, pois apresenta-se uma alta dificuldade no desenvolvimento inicial do projeto, por ter vários microserviços para serem gerenciados.
Agilidade nas entregas	Não há agilidade na entrega, pois como a aplicação se encontra dentro de um único projeto, é necessário que o projeto como um todo esteja funcionando para ser entregue as novas funcionalidades.	Há agilidade na entrega, pois como o projeto é dividido em vários microserviços, cada um deles é independente, então não é necessário que todos estejam funcionando para o projeto funcionar, podendo ser disponibilizadas novas funcionalidades mais rapidamente.
Manutenção	Difícil manutenção, devido ao forte acoplamento das unidades de código, tornando-se complexa e onerosa.	Fácil manutenção, pelo fraco acoplamento das unidades de código, tronando-se mais simples e menos impactante.
Aprendizagem	Fácil aprendizagem, pois todas as funcionalidades estão dentro de um único projeto.	Aprendizagem complexa, pois o projeto é dividido em vários microserviços independentes.

Dependência de componentes	Alta dependência, pelo fato de ser um único projeto, é possível a existência de funcionalidades fortemente acopladas, e caso aconteça alguma alteração ou inclusão nesse meio, a probabilidade de haver comportamentos inesperados é alta.	Baixa dependência, pois como o projeto é dividido em vários microserviços, a probabilidade de existirem funcionalidades fortemente acopladas e ocorrrem comportamentos inesperados é baixa.

Fonte: próprio autor

Assim, mediante o aprendizado proporcionado pelo estudo que embasou este projeto, algumas considerações puderam ser obtidas e serão descritas na seção a seguir.

6 CONCLUSÃO

Mediante o conhecimento proporcionado por meio das pesquisas bibliográficas, estudo de caso e comparação entre as principais características das arquiteturas monolítica e de microserviços, foi possível sintetizar algumas considerações.

Sugere-se que não existe melhor ou pior arquitetura de software para um cenário genérico, e sim qual atende de forma mais adequada às necessidades de um projeto. Desta forma, para compreender qual arquitetura utilizar, faz-se necessário realizar o levantamento do projeto e indicar a arquitetura que melhor atende suas necessidades, norteando-se por alguns aspectos, principalmente considerando as informações obtidas por meio da análise do Quadro 1, proposto neste artigo.

A arquitetura monolítica é recomenda para projetos de pequeno/médio porte, que não tenham uma grande porção de código e que não recebam atualizações constantes em suas funcionalidades. Além disso, a maior facilidade durante a fase inicial do aprendizado gera uma maior produtividade inicial, visto que o escopo do projeto é único. No entanto, é importante destacar a baixa resiliência do projeto que a utiliza, pois uma alteração ou uma falha pode impactá-lo como um todo devido ao escopo unificado do projeto, tornando a manutenção complexa e, por muitas vezes, onerosa, tanto em relação aos custos como, também, ao tempo para conclusão do trabalho.

Já a arquitetura de microserviços, pode ser recomendada para projetos de médio e

grande porte com alta probabilidade de extensão e crescimento, uma vez que seus serviços podem ser escalados mediante demanda, ou seja, um determinado microserviço que vem sendo mais requisitado pode ser escalado individualmente sem afetar a aplicação como um todo. Aliado a isso, o escopo individualizado proporcionado por cada microserviço proporciona a alta resiliência do projeto como um todo, pois atualizações e/ou falhas podem impactar funcionalidades de forma independente, promovendo maior flexibilidade nas operações de manutenção e entrega de novas funcionalidades.

Sendo assim, as lições aprendidas a partir desse estudo pretendem indicar meios para que os desenvolvedores de software possam tomar uma decisão de qual arquitetura utilizar mediante um cenário proposto, para que em um futuro próximo não haja a necessidade de migração de arquitetura em virtude da falta de conhecimento apresentada no processo de elaboração do sistema como um todo.

REFERÊNCIAS

ALMEIDA, Adriano. **Arquitetura de microserviços ou monolítica?**. Disponível em <https://blog.caelum.com.br/arquitetura-de-microservicos-ou-monolitica/> . Acessado em: 11 de Abril de 2020.

AMARAL, Odravison, **CARVALHO**, Marcus. **Arquitetura de Micro Serviços: uma Comparação com Sistemas Monolíticos**. Acessado em: 13 de Agosto de 2020.

BARTIÉ, A. **Garantia da Qualidade de Software** 5ª Edição. Elsevier, 2002.

CONTENT, R. R. **Stakeholders: o que são, quais os tipos e como gerenciá-los**. Disponível em < https://rockcontent.com/br/blog/stakeholder/ > . Acessado em: 07 de Dezembro de 2020.

FOWLER, M. **Microservices and the first law of distributed objects**.: Thoughtworks, 2015. Disponível em: . Acessado em: 22 de Setembro 2020.

FOWLER, M. **Microservice Trade-Offs**. Disponível em <https://martinfowler.com/articles/microservice-trade-offs.html />. Acessado em: 17 de Setembro de 2020.

GIL, A. C. **Métodos e Técnicas de Pesquisa Social**. 5. ed. São Paulo: Atlas, 1999.

KROTH, E. Arquitetura de software para reuso de componentes, Editora. Porto Alegre, 2000.

MACHADO, M. G. **Micro Serviços: Qual a diferença para arquitetura monolítica?**. Acessado em: 21 de Março de 2017.

NHIMI, F. **Princípios e Práticas em Arquitetura de Software**. Instituto de Gestão em Tecnologia da Informação, 2016

RICHARDSON, Chris. Smith, Floyd. **Microservices, From Design to Deployment**. NGINX 2016

SHAW, M. & **GARLAN**, D. **Software Architecture. Perspectives on an Emerging Discipline**. Editora Prentice Hall, 1996.

WAZLAWICK, R S. **Engenharia de Software: conceitos e práticas**. 2. Ed - Elsevier Editora, 2011

WOODS, Dan. **Building Microservices with Spring Boot**. Disponível em < https://www.infoq.com/articles/boot-microservices/?itm_source=infoq&itm_campaign=user_page&itm_medium=link>. Acessado em: 10 de Abril de 2020.

YIN, R. K. **Estudo de caso**: planejamento e métodos. 3. Ed – Porto Alegre: Bookman, 2005.