ANÁLISE DA ARQUITETURA DE MICROSERVIÇOS

MICROSERVICES ARCHITECTURE ANALYSIS

Júlia Leôncio Rodrigues – juliarodrigues606@gmail.com
Giuliano Scombatti Pinto – giuliano.pinto@fatectq.edu.br
Faculdade de Tecnologia de Taquaritinga (FATEC) – São Paulo – Brasil

RESUMO

O artigo tem como objetivo apresentar a arquitetura de MicroServiços, suas características, vantagens, desvantagens e *frameworks* que podem ser utilizados ao se desenvolver uma aplicação usando esse estilo arquitetônico. O artigo também apresenta noções do que é a arquitetura de software, qual a função de um arquiteto de softwares e um breve resumo da arquitetura monolítica. A fundamentação teórica do mesmo foi realizada por meio de uma pesquisa bibliográfica envolvendo artigos científicos publicados em revistas e sites, apresentações de simpósios, monografias de trabalhos de conclusão de curso, livros e ebooks. Após todas as pesquisas foi possível concluir que a arquitetura de MicroServiços pode ser vista como uma das soluções com os problemas de desenvolvimento da atualidade, pois ela responde bem a variedade de plataformas e linguagens disponíveis e o aumento do número de requisições ao sistema. No entanto a complexidade de desenvolvimento e documentação pode fazer com que outras arquiteturas mais simples sejam melhores aplicados ao software a ser desenvolvido.

Palavras-chave: MicroServiços. Arquitetura de Software. Arquitetura Monolítica.

ABSTRACT

The purpose of this article is to introduce the architecture of MicroServices, its characteristics, advantages, disadvantages and frameworks that can be used when developing an application using this architectural style. The article also introduce notions of what software architecture is, what the function of a software architect is, and a brief resume of the monolithic architecture. Theoretical foundation of the study was made out through a bibliographical research involving scientific articles published in magazines and websites, symposium presentations, undergraduate thesis, books and ebooks. After all the research it was possible to conclude that the MicroServices architecture can be seen as one of the solutions to the current development problems, because it responds well to the variety of platforms and languages available and the increase of the number of requests to the system. However the complexity of development and documentation may make other simpler architectures better applied to the software being developed.

Keywords: MicroServices. Software Architecture. Monolithic Architecture

1 INTRODUÇÃO

Segundo a OS Platform Statistics(2018) fornecida pela W3C, atualmente são utilizados 9 tipos de sistemas operacionais diferentes, sendo que esta plataforma caracteriza Mobile como sendo uma única categoria, a partir dela pode se derivar mais 2 principais sistemas operacionais (Android, IOS) - uma vez que o Windows para a plataforma móvel foi descontinuado - e por fim outros sistemas que são menos utilizados. Por meio destes dados é possível notar a atual dificuldade em desenvolver um software que seja compatível com todos os sistemas operacionais, já que cada um possui suas particularidades, como por exemplo, a linguagem de programação a ser utilizada.

O'Grady (2018) chama a atenção para 20 linguagens de programação mais utilizadas. A maioria dos desenvolvedores necessita definir dentre essas diversidades de linguagens e qual atenderá melhor as necessidades finais do cliente, sendo que cada uma delas possui seu ponto forte, que pode ser necessário em diferentes partes do projeto.

Outro problema apontado por Tripoli e Carvalho (2016) no atual ambiente de desenvolvimento é o contínuo aumento de usuários e números de requisições solicitadas ao sistema que será desenvolvido.

É nesse cenário que surge o termo que é visto como a solução para os problemas citados acima: A arquitetura de MicroServiços. Segundo Fowller e Lewis (2014) ela foi primariamente discutida em maio de 2011 dentro de um Workshop para arquitetos de software.

Apesar de ser um padrão de arquitetura relativamente novo, ele vem sendo muito discutido dentro do ambiente de desenvolvimento de sistemas, pelo fato de que muitas grandes empresas e startups vêm utilizando esse método de desenvolvimento. Tripoli e Carvalho (2016) apresentam também alguns exemplos de empresas que passaram a utilizar essa arquitetura, dentre elas: Amazon , Spotify e Netflix.

Deste modo o artigo tem como objetivo a apresentação deste padrão de arquitetura, suas vantagens e desvantagens e por fim, algumas das tecnologias que podem ser utilizadas juntamente com este paradigma. No entanto para que haja o entendimento de tal, será abordado no primeiro capitulo os pontos principais do que é a arquitetura de software. Após essa explanação, será apresentado o padrão referente a arquitetura de MicroServiços.

2 ARQUITETURA DE SOFTWARE

É difícil encontrar uma definição exata para o que é arquitetura de software. Fowler (2006) também cita esse problema, mas ele chega a seguinte conclusão sobre o termo: “Arquitetura se resume a coisas importantes”. Dentro da definição de arquitetura fornecida pela ISO/IEC/IEEE 42010 (2011) pode-se observar o significado dessas “coisas importantes”, que são entendidas como os elementos, as relações e os princípios. Os mesmos definem arquitetura de software da seguinte forma: “A organização fundamental de um sistema incorporado em seus **componentes,** suas **relações** entre si e com o meio ambiente, e os **princípios** que guiam seu design e evolução.”

Newman (2015) afirma que não se pode comparar o profissional arquiteto de outras áreas com o arquiteto de software, porque isso poderia significar que este profissional sabe o que está fazendo quando na maioria das vezes não é o que acontece. Ele aponta ao fato da quantidade de vezes que um sistema pode falhar, além das adaptações que o sistema deve ter assim que ele se encontra nas mãos do cliente final. Após essas declarações ele afirma a respeito do profissional arquiteto de software: “Eles precisam garantir que o sistema esteja apto para o uso agora, mas também uma plataforma para o futuro.”.

Sendo assim, para o auxílio desses profissionais que foram criados os padrões de arquitetura. Alexander (1979) define que “cada padrão é uma regra de três partes, que expressa uma relação entre um determinado contexto, um problema e uma solução.”.

O padrão de arquitetura é “um esquema de organização estrutural fundamental para sistemas de software. Ele fornece um conjunto de subsistemas predefinidos, especifica suas responsabilidades e inclui regras e diretrizes para organizar os relacionamentos entre eles (BUSCHMANN, F. et al.)” , ou seja ele fornece o modo como sistema será organizado em nível estrutural.

Martins (2007) compara o desenvolvimento de um sistema a construção de uma casa, e afirma que do mesmo modo que existem diferentes estilos arquitetônicos na construção civil, é possível observar também diferentes estilos arquitetônicos na Arquitetura de software. Ele afirma que cada estilo arquitetônico envolve “uma serie de elementos: banco de dados, comunicação entre componentes, integração e organização dos componentes”.

Martins(2007) também afirma: “Os princípios básicos de organização da arquitetura são a arquitetura em camadas, a arquitetura em colunas, a orientada a objetos e o monolítico.”

Mas para que se possa chegar ao objetivo final desse artigo que é a explanação da Arquitetura de MicroServiços é necessário que se compreenda o funcionamento da arquitetura monolítica.

2.1 Arquitetura Monolítica

Martins (2007) menciona que “Na arquitetura monolítica o software é construído num único modulo, incluindo a interface com o usuário, as regras de negocio, persistência de dados, comunicações com outros sistemas, dentre outras funções ”. Visto que todo o sistema é elaborado dentro de um único modulo o código se torna extenso, confuso, de difícil alteração e com pouca escalabilidade. No entanto como já mencionado no capitulo anterior é função de um arquiteto disponibilizar um sistema que funcione hoje e que servirá como base para as futuras necessidades do mesmo, ou seja, as futuras alterações. O próprio Martins (2007) conclui: “evidentemente não é a abordagem mais adequada.”. Fowller e Lewis (2014) explicam de forma sucinta o porquê de não ser o mais indicado: “Qualquer mudança no sistema consiste em publicar uma nova versão da aplicação *server-side*.”.

Isso não significa que uma aplicação monolítica não terá sucesso em solucionar os requisitos do sistema. Fowller e Lewis (2014) asseguram que “aplicações monolíticas podem ser bem-sucedidas”, mas o procedimento de republicação ou escalonamento se torna decepcionante.

A arquitetura de MicroServiços traz uma abordagem diferente que tem como objetivo sanar as dificuldades apresentadas pela arquitetura monolítica.

3 ARQUITETURA DE MICROSERVIÇOS

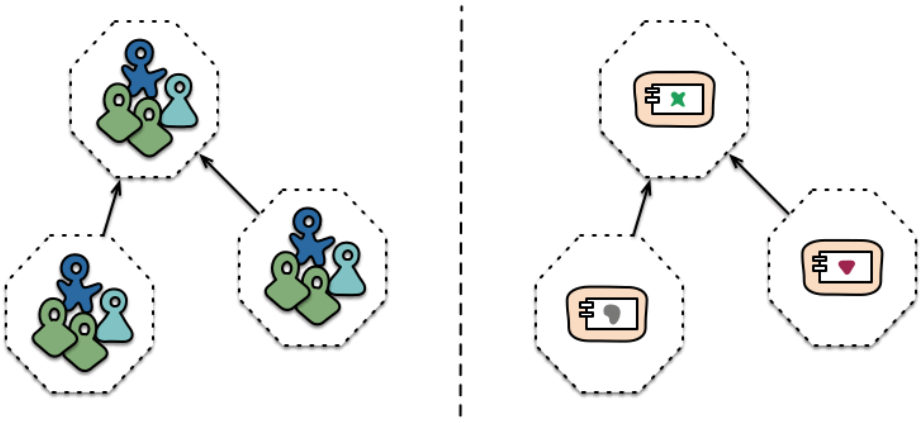
Fowller e Lewis (2014) descrevem a arquitetura de MicroServiços como sendo “ uma abordagem para desenvolver uma única aplicação como uma suíte de serviços, cada um rodando em seu próprio processo e se comunicando através de mecanismos leves”. Portanto na arquitetura de MicroServiços a aplicação é “quebrada” em diversos serviços que possuem

uma alta coesão e baixo acoplamento, ou seja eles tem pouca dependência ou até mesmo nenhuma de outros serviços da mesma aplicação.

Algumas das características da arquitetura de MicroServiços definidas por Fowler e Lewis (2014) são:

- Componentização por meio de serviços: Fowler e Lewis(2014) definem componentes como “uma unidade de software que é substituída ou atualizada de maneira independente.” e serviços como “componentes em processos diferentes que se comunicam através de mecanismos tais como requisições via webservices ou chamadas de códigos remotos”. Ou seja, utilizar a componentização por meio de serviços é dividir sua aplicação em componentes que se comunicarão entre si por meio das requisições (webservices ou chamadas de código remotas) de forma a transforma-los em um serviço. Ela traz benefícios, pois poderão ser atualizados os componentes individualmente de forma a não alterar toda a aplicação, podendo assim republicar apenas o componente alterado.
- Organização através da área de negócios: A equipe de desenvolvimento é criada para cada área de negocio deste modo ela é considerada *full stack.*

Figura 1 - Formação de equipes levando em consideração as regras de negócio



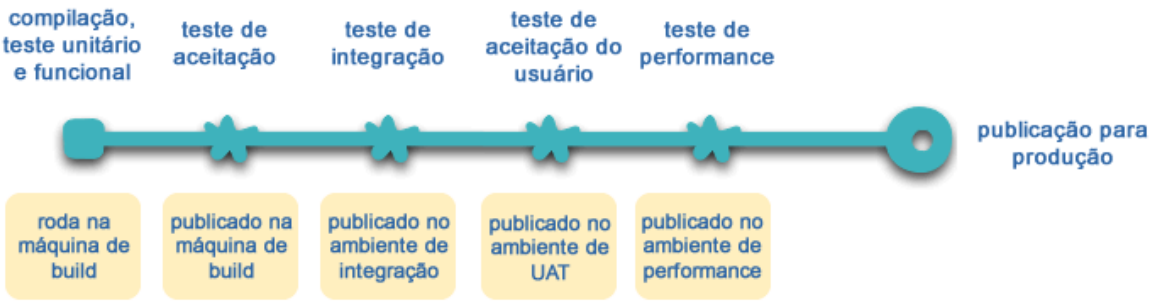
Fonte: Fowler e Lewis (2014)

É possível ver na figura 1 que cada equipe desenvolveu uma área de negócio inteira desde a base de dados até a sua interface.

- Produtos e não projetos: Fowler e Lewis(2014) justificam que “uma inspiração para isso é a noção da Amazon de “*you build, you run it*””. Ou seja, a equipe é responsável por entregar o serviço de forma que ele esteja funcionando independentemente.

- Endpoints inteligentes e fluxo de comunicação simples: Visto que os serviços dentro desta arquitetura possuem baixo acoplamento e alta coesão os mesmo autores indicam protocolos REST e barramento de mensagens simples para a comunicação entre serviços.
- Governança descentralizada: Cada componente pode ser desenvolvido de um forma diferente (até mesmo linguagem), também é possível utilizar *frame*works open sources para a solução de problemas comuns e as equipes são responsáveis por todos os aspectos do sistema que está sobre sua responsabilidade. Uma das empresas que disponibilizam projetos open sources como soluções comuns de problemas é a Netflix.
- Administração descentralizada de dados: Cada serviço ou componente pode possuir sua própria base de dados.
- Padrões amplamente testados e padrões impostos: Segundo Fowler e Lewis(2014) equipes de desenvolvimento que utilizam a arquitetura de MicroServiços costumam fazer " uso de padrões abertos tais como HTTP, ATOM e outros micro formatos" .
- Entrega e publicação contínua: Para que o componente seja publicado funcionando corretamente são realizados testes automatizados. A figura 2 contém as etapas de construção do serviço.

Figura 2 - Etapas de construção de um serviço



Fonte: Fowler e Lewis (2014)

- Projetado para a falha: Os autores enfatizam também que, como "os serviços podem falhar a qualquer momento, é importante ser capaz de detectar falhas rapidamente e, se possível, restaurar o serviço automaticamente." e para isso é necessário que seja feito monitoramento em tempo real.

Outro termo que é importante ter conhecimento ao se falar sobre MicroServiços é escalabilidade. Neto (2015) define escalabilidade como sendo "a habilidade que um sistema

computacional tem de lidar, de forma transparente, com um número crescente de usuários ao mesmo tempo”. No entanto, existe dois modos que podem fazer o sistema responder adequadamente ao numero de requisições impostas a ele - a escalabilidade vertical e a horizontal. O mesmo autor explica que na escalabilidade horizontal são utilizados diversos servidores e na escalabilidade vertical é aumentado o poder de processamento do servidor. No entanto, Neto (2015) ao falar sobre a escalabilidade vertical explica que uma das suas desvantagens é que “Normalmente esta modalidade apresenta maior investimento inicial e menor flexibilidade para introdução de novas tecnologias”.

Isso é relevante para a arquitetura de MicroServiços pois em consequência da componentização por meio de serviços e a administração descentralizada de dados, esta arquitetura facilita a escalabilidade horizontal.

3.1 Vantagens

Moreira e Beder (2015) citam entre as vantagens da arquitetura de MicroServiços: “ heterogeneidade tecnológica, resiliência, escalabilidade e facilidade de implantação.”. Resiliência no artigo refere-se à facilidade de solucionar as falhas que podem ser apresentadas durante o desenvolvimento e utilização da aplicação.

3.1 Desvantagens

As maiores desvantagens mencionadas por Moreira e Beder (2015) são a “complexidade de desenvolvimento, chamadas remotas e gerenciamento de múltiplos bancos de dados e transações”. Santos (2017) também chama atenção para a necessidade de um excelente planejamento e documentação bem fundamentada para que haja gerenciamento correto, pois a sua complexidade pode fazer com que o sistema se torne desorganizado e de difícil entendimento da aplicação como um todo.

4 PLATAFORMAS PARA IMPLEMENTAÇÃO DE MICROSERVIÇOS

Para que melhor se possa aproveitar das vantagens do uso da arquitetura de MicroServiços foram criados *frameworks* e plataformas que auxiliam na sua implantação, escalonamento e monitoração.

Uma das plataformas disponibilizadas atualmente é o Docker. Conforme mencionado na página oficial da plataforma de containers Docker(2018), com ele é possível "implantar e dimensionar independentemente cada MicroServiço, coordenar sua implantação através da orquestração Swarm ou Kubernetes e colaborar entre as equipes por meio de uma maneira consistente de definir os aplicativos". Cabe lembrar que Swarm e Kubernetes são *frameworks*, *open sources*. Eles podem ser utilizados juntamente com o Docker facilitando a experiência de implantação de componentes de uma aplicação de MicroServiço.

O Kubernetes além de ser um *framework* utilizado na implantação, também serve para auxiliar no escalonamento. Torre, Wagner e Rousos (2018) menciona que o *framework* Kubernetes também serve como orquestrador, ou seja, ele realiza o balanceamento do quanto cada serviço/container necessita de cada servidor como se eles fossem uma única maquina.

O Prometheus é um *framework open source* que é utilizado para o monitoramento, a partir dele é fácil encontrar falhas que possam estar ocorrendo.

> O Prometheus copia as métricas de trabalhos instrumentados, diretamente ou através de um gateway de envio intermediário para trabalhos de curta duração. Ele armazena todos os exemplos copiados localmente e executa regras sobre esses dados para agregar e registrar novas séries temporais a partir de dados existentes ou gerar alertas. (PROMETHEUS, 2018)

Além de registrar e alertar sobre o que se passa na aplicação, é disponibilizado pelo Prometheus uma interface gráfica para a apresentação dos dados coletados que cabe ao desenvolvedor a escolha de usa-la ou não.

5 PROCEDIMENTOS METODOLÓGICOS (OU MATERIAIS E MÉTODOS)

Para a realização deste artigo foi realizado uma pesquisa de base bibliográfica para a explanação dos termos citados e da arquitetura que serviu de base para a pesquisa.

A pesquisa bibliográfica, segundo Marconi e Lakatos (2008), "abrange toda bibliografia já tornada pública em relação ao tema de estudo". Deste modo foram utilizados

relatórios técnicos, apresentações de simpósio, teses de trabalhos de conclusão de curso, artigos apresentados em revistas e web sites, livros impressos e *ebooks* para a contextualização do trabalho.

4 CONCLUSÃO

De acordo com as principais características da arquitetura de MicroServiços pode-se notar como ela pode solucionar a maioria dos problemas atuais na área de desenvolvimento. Alguns dos exemplos de soluções que a arquitetura traz são:

- Número de requisições ao sistema: visto que a arquitetura de MicroServiços facilita o escalonamento horizontal é fácil dedicar mais ou menos *host* para determinado serviço.
- Variedades de plataformas: Como a arquitetura não é restrita a uma única linguagem promove a migração entre as plataformas.
- Equipes remotas: A equipe de desenvolvimento de aplicação não precisa estar em contato com todos os desenvolvedores, a divisão da aplicação em pequenos serviços permite que cada um tenha a necessidade de se comunicar apenas com os envolvidos no desenvolvimento o serviço que está designado.

Mas apesar de todas essas vantagens a complexidade da integração dos serviços e suas chamadas, o planejamento necessário e o extremo cuidado com a documentação para que não haja confusão durante o desenvolvimento, pode acabar não sendo a mais adequada conforme a aplicação. Algumas vezes pode ser que uma arquitetura mais simples, como por exemplo a monolítica, solucione melhor as necessidades do sistema.

REFERÊNCIAS

ALEXANDER, Christopher. **The timeless way of building**. Estados Unidos da America: Oxford University Press, 1979. 552 p.

BUSCHMANN, F. et al. **Patternoriented software architecture**: A-system-of-patterns. 1 ed. Alemanha: Wiley, 1996. 493 p.

DOCKER. Microservices. Disponível em: https://www.docker.com/solutions/microservices. Acesso em: 05 set. 2018.

ESTADOS UNIDOS DA AMERICA(EUA). **Systems and software engineering — Architecture description.** International Standard ISO/IEC/ IEEE 42010. 1ª. ed. 2011. p.1-37. ISBN 978-0-7381-7142-5

FOWLER, Martin; LEWIS, James Microservices a definition of this new architectural term. [S.L], jun. 2017. Disponível em: https://martinfowler.com/articles/microservices.html. Acesso em: 16 abr. 2018.

FOWLER, Martin. **Padrões de arquitetura de aplicações corporativas**. Porto Alegre: Bookman, 2006. 493 p.

MARCONI, Maria De Andrade; LAKATOS, Eva Maria. Técnicas de pesquisa: Planejamento e Execução de Pesquisas - Amostragens e Técnicas de Pesquisas - Elaboração, Análise e Interpretação de Dados. 7 ed. Brasil: Atlas, 2008. 296 p.

MARTINS, JOSE CARLOS CORDEIRO. **Técnicas para gerenciamento de projetos de software**. 1 ed. Rio de Janeiro: Brasport, 2007. 456 p.

MOREIRA, P. F. M.; BEDER, D. M. Desenvolvimento de Aplicações e Micro Serviços: Um estudo de caso. T.I.S. São Carlos, São Carlos, v. 4, n. 3, p. 209-215, nov./dez. 2015. ISSN 231 6-2872

NETO, Manoel Veras de Sousa. Virtualização: Tecnologia Central do Datacenter. 2 ed. Brasil: Brasport, 2015. 224 p.

NEWMAN, Sam. **Building microservices**. 1 ed. Estados Unidos da America: O'Reilly, 2015. 280 p.

PROMETHEUS. Overview. Disponível em: https://prometheus.io/docs/introduction/overview/. Acesso em: 05 set. 2018.

REDMONK. The redmonk programming language rankings: january 2018. Disponível em: http://redmonk.com/sogrady/2018/03/07/language-rankings-1-18/. Acesso em: 15 abr. 2018.

SANTOS, Lucas. Microserviços: dos grandes monólitos às pequenas rotas. Training Center, [S.L], jun. 2017. Disponível em: https://medium.com/trainingcenter/microservi%C3%A7os-dos-grandes-mon%C3%B3litos-%C3%A0s-pequenas-rotas-adb70303b6a3. Acesso em: 04 set. 2018.

TRIPOLI , C. S.; CARVALHO, R. P. Micros-Serviços: Características, benefícios e desvantagens em relação à arquitetura monolítica que impactam na decisão do uso desta arquitetura. **II Seminário de desenvolvimento em soa com cloud computing e conectividade Instituto nacional de telecomunicações – INATEL.** 2016. p.1-12. ISSN 2447-2352. Disponível em: http://www.inatel.br/biblioteca/pos-seminarios/seminario-de-desenvolvimento-em-soa-com-cloud-computing-e-conectividade/2016-3/9497-micros-servicos-caracteristicas-beneficios-e-desvantagens-em-relacao/file. Acesso em: 21/03/2018

W3SCHOOLS.COM. **Os platform statistics**. Acesso em: https://www.w3schools.com/browsers/browsers_os.asp. Acesso em: 09 abr. 2018.