PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Instituto de Ciências Exatas e de Informática

Análise de alternativas para implementação de microsserviços: um estudo prático sobre refatoração de sistemas monolíticos.*

Analysis of alternatives for the implementation of microservices: a practical study on refactoring of monolithic systems.

Pablo Ricardo Pereira[1]
Pedro Alves de Oliveira[2]

Resumo

O presente artigo tem como objetivo comparar os pontos positivos e negativos ao se aplicar microsserviços e arquitetura orientada a serviços. Duas arquiteturas de software serão utilizadas e comparadas, a arquitetura orientada a serviços SOA utilizando microsserviços em relação a uma arquitetura monolítica. Por meio da implementação de uma funcionalidade em um sistema utilizando a arquitetura monolítica e reproduzindo essa funcionalidade com a utilização de microsserviços na arquitetua SOA é possível compará-las em termos de desempenho, escalabilidade, correção, refatoração, entre outras comparações realizadas. Com os resultados obtidos são demonstrados os ganhos alcançandos ao utilizar-se de microsserviços em uma arquitetura orientada a serviços SOA, servindo como fonte de informação e argumentação na escolha e utilização dessa arquitetura. A comparação de duas arquiteturas em conjunto com o desenvolvimento da aplicação, permite demonstrar as vantagens e desvantagens entre as duas arquiteturas, além de possibilitar a aplicação do conhecimento teórico e tácito adquirido durante a graduação, servindo futuramente como material científico para novos estudos.

Palavras-chave: Microsserviços. SOA. Monolítico. LaTeX. Abakos. Periódicos.

*Artigo apresentado ao Instituto de Ciências Exatas e Informática da Pontifícia Universidade Católica de Minas Gerais como pré-requisito para obtenção do título de Bacharel em Sistemas de Informação.

[1]Aluno Graduando em Sistemas de informação, Brasil – pablorpmg@gmail.com.
[2]Orientador, Instituto de Ciências Exatas e de Informática da PUC Minas, Brasil – pedro.aol@gmail.com.

Abstract

The objective of this article is to compare the positives and negatives when applying microservices and service oriented architecture. Two software architectures will be used and compared, the SOA service-oriented architecture using microservices in relation to a monolithic architecture. By implementing a functionality in a system using the monolithic architecture and reproducing this functionality with the use of microservices in the SOA architecture, it is possible to compare them in terms of performance, scalability, correction, refactoring and other comparisons. The obtained results demonstrate the gains achieved when using micro-services in an SOA-oriented architecture, serving as a source of information and argumentation in the choice and use of this architecture. The comparison of two architectures together with the development of the application, allows to demonstrate the advantages and disadvantages between the two architectures, besides allowing the application of the theoretical and tacit knowledge acquired during the graduation, serving in future as scientific material for new studies.

Keywords: Microservices. SOA. Monolithic.. LaTeX. Abakos. Periodics.

1 INTRODUÇÃO

Uma etapa essencial no desenvolvimento de software é a definição da arquitetura que será utilizada. Durante a escolha da arquitetura define-se as características de implementação do sistema a ser desenvolvido, tornando-se assim uma tarefa chave para estabelecer os requisitos de escalabilidade, complexidade, restrições tecnológicas, manutenibilidade entre outros.

Dentre as arquiteturas existentes para desenvolvimento de software, a estrutura monolítica constitui uma opção que não atende a sistemas distribuidos, pois sua lógica é formada por trechos de códigos que são chamados de páginas ou módulos que se interligam através de chamadas, conhecidas como *links*. Na arquitetura monolítica todo o sistema fica concentrado somente em um *host*, dificultando assim sua abertura para acesso externo.

A arquitetura orientada a serviços (SOA), por sua vez tem a estrutura orientada a disponibilizar serviços. Ao utilizar microsserviços para aplicar a arquitetura SOA, cada microsserviço tem um objetivo e papel específico na constituição do sistema. Os microsserviços devem funcionar separados e, quando utilizados em conjunto, devem constituir o sistema.

Ambas as arquiteturas podem ser utilizadas em desenvolvimento de sistemas e cada opção arquitetural tem suas vantagens e desvantagens. Este artigo tem como objetivo servir como uma fonte de conhecimento teórico e prático na comparação entre a arquitetura monolítica em relação a microsserviços aplicados na arquitetura SOA.

2 DESENVOLVIMENTO

A elaboração deste artigo constitui-se de pesquisa qualitativa, experimental e comparativa. Foi desenvolvido um estudo teórico junto a provas de conceito, para comparar a utilização de microsserviços aplicados na arquitetura orientada a serviços, em relação a arquitetura monolítica.

2.1 O que é arquitetura de software

De acordo Gheezzi et al. (2002) citados por Souza (SOUZA et al., 2015), a Arquitetura de Software é um conjunto de componentes relacionados que satisfaz requisitos funcionais e não funcionais de um sistema ou uma abstração situada entre o problema que pretende solucionar a implementação técnica.

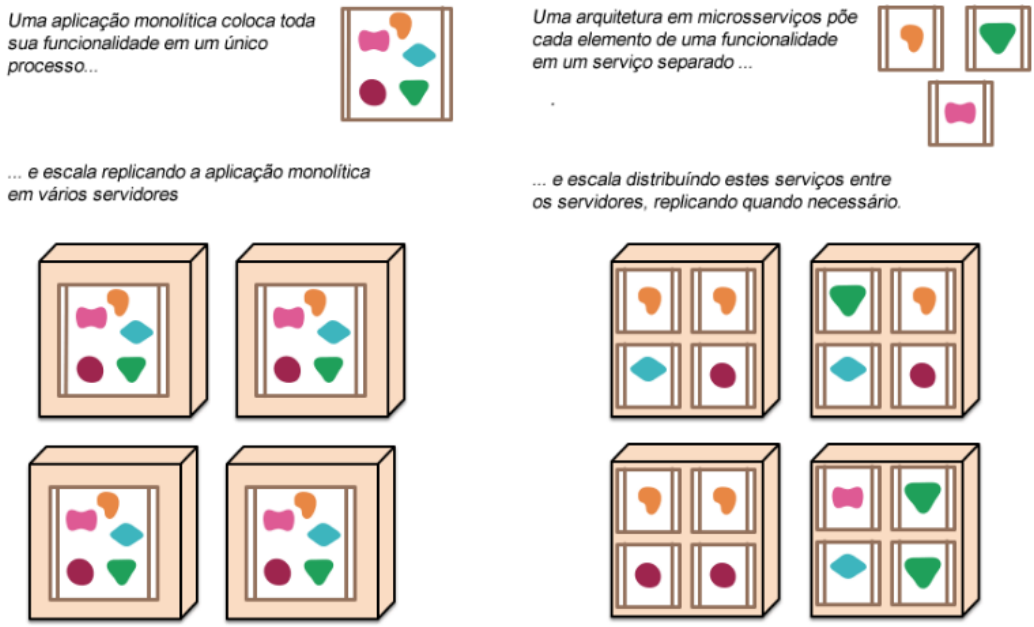
Arquitetura é algo que está baseado na sua interpretação individual, é um entendimento conjunto do projeto baseado nas práticas dos desenvolvedores e constantemente se exibe nos elementos de maior relevância do software e o modo como eles se comunicam (FOWLER, 2006). Ainda segundo o autor, arquitetura está relacionada as decisões inicias por ser difícil realizar alterações no futuro. Caso as alterações futuras não apresentem dificuldades, elas não

dizem respeito a arquitetura, no fim a arquitetura refere-se aos recursos importantes.

2.2 Arquitetura monolítica e arquitetura orientada a serviço

De acordo com Lewis e Fowler (2014), um servidor monolítico tem uma abordagem mais natural onde sua estrutura de funcionamento permite usar os mecanismos fundamentais que são executados em um só processo, podendo assim dividir a aplicação em funções e classes. Sistemas monolíticos podem dar certo, mas problemas são mais comuns. Uma única alteração necessita que todo o sistema seja refeito, manter sua estrutura modular de manutenção e alteração fica mais difícil, tornando-se necessário mais recursos para escalonar o sistema (LEWIS; FOWLER, 2014). A diferença entre as duas arquiteturas pode ser observada na Figura 1.

Figura 1 – Aplicações monolíticas e microsserviços



Fonte: Lewis e Fowler (2014)

Para Erl (2009) o modelo arquitetural de sistemas orientado a serviços tem por objetivo demonstrar que os serviços são o caminho para se atingir a estratégia de sistemas orientados a serviços e tem como objetivo a melhoria do desenvolvimento de forma eficiente e produtiva da organização.

Newman (2015) conceitua SOA como:

> A arquitetura orientada a serviços (SOA) é uma abordagem de design em que vários serviços colaboram para fornecer algum conjunto final de recursos. Um serviço aqui normalmente significa um processo de sistema operacional separado. Comunicação entre estes serviços ocorre através de chamadas através de uma rede, em vez de chamadas de método dentro de um limite de processo. SOA surgiu como uma abordagem para combater os desafios das grandes aplicações monolíticas. É uma abordagem que tem como objectivo promover a reutilização de software; dois ou mais aplicativos do usuário final, por exemplo, poderia utilizar os mesmos serviços. Ele tem o objetivo de torná-lo mais

> fácil de manter ou reescrever o software, como teoricamente podemos substituir um serviço com outro, sem ninguém o saber, enquanto a semântica do serviço não mudam muito. (NEWMAN, 2015, p. 8, tradução nossa)[3].

Segundo Sommerville (2011) SOA consiste em uma metodologia de construção para sistemas de informação que possui serviços isolados, os quais podem estar distribuídos em diferentes máquinas. Cada serviço tende a ter funcionalidade curta e especifica. Sommerville (2011) define a arquitetura SOA como uma maneira de produzir-se software subdividido em segmentos autossuficientes e desacoplados fisicamente.

2.3 SaaS, Serviço, Microsserviço

Conforme Sommerville (2011) software como serviço (SaaS) é a forma de se entregar o funcionamento de software para o utilizador, ao passo que SOA é uma metodologia de desenvolvimento de software. Ainda segundo o autor a aplicação arquitetural orientada a serviço necessita se comportar como serviço. Por sua vez, SaaS não necessita ser desenvolvido utilizando os coneitos de SOA, mas ao se utilizar SOA na produção de SaaS, permite-se a integração com outros serviços.

De acordo com Erl (2009) serviços são materialmente autônomos e de natureza dissemelhante que estruturam o alcance do propósito estratégico relacionados a computação orientada a serviço. Para Erl (2009) um serviço possui seu âmbito funcional próprio e uma coleção de aplicabilidade relativa a seu âmbito. Por este moivo serviços são apropriados para serem utilizados por softwares externos, usualmente na forma de Interface de Programação de Aplicativos API.

Segundo Newman (2015) microsserviços são componentes separados e simplificados que são desenvolvidos de forma isolada e distribuída, mas que trabalham juntos, permitindo diminuir problemas relacionados a sua utilização. Os microsserviços utilizam a mesma abordagem dos serviços autônomos e devem centralizar em uma única aplicação, permitir sua modificação e implementação de forma autônoma uns aos outros sem interferência de quem os utilizam, impossibilitando sua expansão desnecessária.

Fowler (2017) define mircrosserviço como um pequeno serviço responsável por uma única funcionalidade que deve ser executada de forma correta, constituindo assim um pequeno módulo que pode ser trocado, implantado e desenvolvido independentemente de outros microsserviços, sozinho não há um significado de existência mas em conjunto compõe um software executável que seria formado por volumoso sistema autônomo.

[3] Service-oriented architecture (SOA) is a design approach where multiple services collaborate to provide some end set of capabilities. A service here typically means a completely separate operating system process. Communication between these services occurs via calls across a network rather than method calls within a process boundary. SOA emerged as an approach to combat the challenges of the large monolithic applications. It is an approach that aims to promote the reusability of software; two or more end-user applications, for example, could both use the same services. It aims to make it easier to maintain or rewrite software, as theoretically we can replace one service with another without anyone knowing, as long as the semantics of the service don't change too much.

2.3.1 Arquitetura de microsserviços

Lewis e Fowler (2014) elucidam que:

> Para começar explicando o que é o padrão de microsserviços, podemos compara-lo ao padrão monolítico: uma aplicação monolítica é feita como uma única unidade. Aplicações empresariais são geralmente construídas em três partes principais: uma interface para o cliente (tais como páginas HTML e Javascript rodando em um navegador no computador do usuário), um banco de dados (várias tabelas em um mesmo lugar, geralmente um sistema de banco de dados relacional) e uma aplicação server-side. A aplicação server-side irá manipular as requisições HTTP, executar toda lógica de domínio, receber e atualizar os dados da base de dados e por fim, selecionar e popular os blocos HTML para enviar ao navegador. Esta aplicação server-side é monolítica – uma única unidade lógica executável. (LEWIS; FOWLER, 2014, p. 1, tradução nossa)[4].

Arquitetura de microsserviços tem como finalidade desenvolver um grupo de funcionalidades autônomas permitindo que cada serviço seja autossuficiente, divergindo das formas tradicionais como a monolítica, onde deve-se processar todo mecanismo que se concentra em uma única aplicação e possui copias em servidores, os microsserviços constituem um ecossistema onde cada um tem sua função no conjunto de microsserviços que formam o sistema (FOWLER, 2017).

2.4 Front-End e Back-End

De acordo com Wales (2017) podemos dividir o o desenvolvimento em tipos, dentre eles temos o desenvolvimento *front-end* e *back-end*, no *front-end* cuidamos da parte visual apresentada para o usuário, nele que preocupamos com o desenvolvimento que envolvendo a interface, usabilidade e interatividade. Normalmente utiliza-se *frameworks* e bibliotecas de componentes como o Bootstrap que permitem economizar tempo na entrega, melhorar a experiência e garantir responsividade ao usuário.

Ainda segundo Wales (2017) o desenvolvimento *back-end* seria a estrutura lógica que possibilita o funcionamento, desenvolvendo assim a ligação do servidor, com a aplicação e banco de dados, uma analogia seria que o *front-end* cuida da parte visual, e o *back-end* cuida da parte funcional do sistema.

[4]To start explaining the microservice style it's useful to compare it to the monolithic style: a monolithic application built as a single unit. Enterprise Applications are often built in three main parts: a client-side user interface (consisting of HTML pages and javascript running in a browser on the user's machine) a database (consisting of many tables inserted into a common, and usually relational, database management system), and a server-side application. The server-side application will handle HTTP requests, execute domain logic, retrieve and update data from the database, and select and populate HTML views to be sent to the browser. This server-side application is a monolith - a single logical executable

3 METODOLOGIA

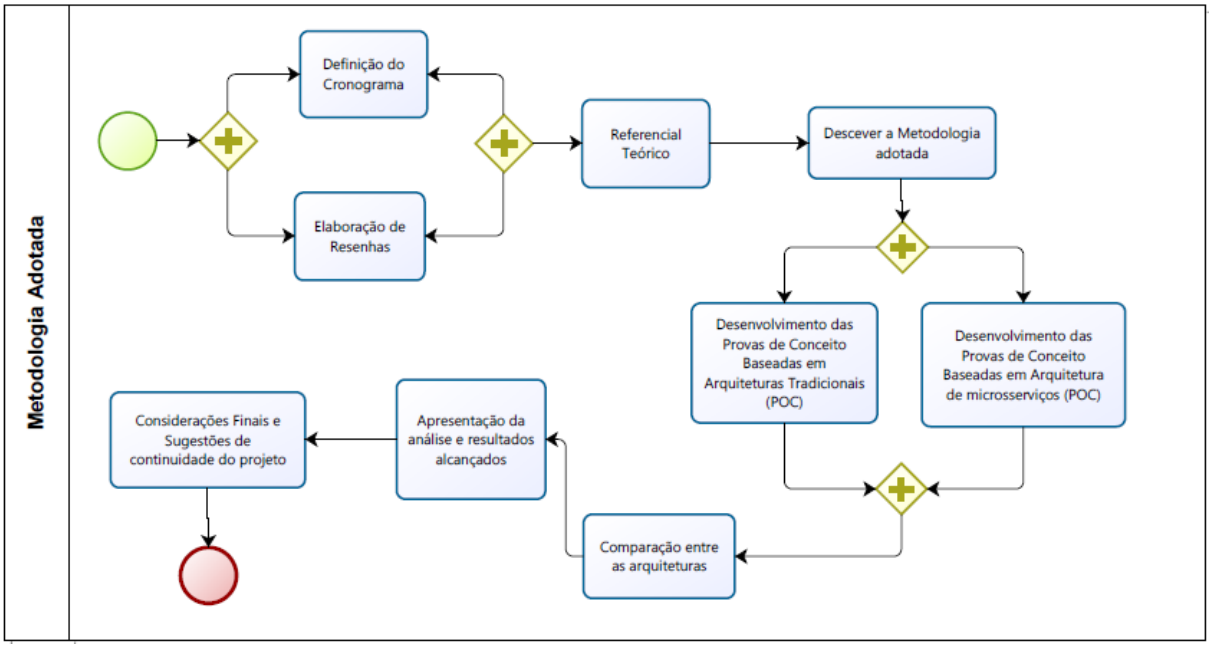
Para desenvolver esta a pesquisa acadêmica foi aplicada a metodologia da pesquisa qualitativa, onde a preocupação não recai na representação numérica, mas sim na especialização e entendimento de uma instituição, grupo social, etc. (GOLDENBERG, 2004).

Em conjunto com a pesquisa qualitativa, foram desenvolvidas provas de conceitos (PDC) para comprovar as vantagens de se utilizar microsserviços na arquitetura SOA em comparação com a arquitetura monolítica, para isso será utilizado de experimentos e implementações junto com o conteúdo teórico apresentado.

Inicialmente foi definido um cronograma e elaboradas resenhas junto à realização da pesquisa, estabelecendo-se um referencial teórico que permita demonstrar as vantagens de se utilizar uma arquitetura baseada em microsserviços em relação à arquitetura tradicional. A arquitetura de microsserviços gera ganhos de produtividade e velocidades de desenvolvimento, possibilita de maneira mais factível a troca de tecnologias legadas, proporcionando o aumento do tamanho da aplicação, seja adicionar recursos ou máquinas (FOWLER, 2017).

Por fim são apresentados os resultados, limitações, ganhos alcançados e sugestões de continuidade do projeto. Para tornar mais claro a metodologia adotada é apresentado na Figura 2 um fluxograma criado utilizando a notação para modelagem de processos de negócios *"Business Process Modeling Notation"* (BPMN).

Figura 2 – Metodologia Aplicada



Fonte: Elaborado pelo autor.

4 ESTUDO PRÁTICO

Como estudo prático foram desenvolvidos dois sistema para alocação dos alunos de TCC, o primeiro sistema foi construído utilizando arquitetura monolítica, onde toda lógica e o banco de dados ficaram localizados dentro da máquina, seguindo o conceito de arquitetura monolítica.

A apartir do sistema monolítico foi realizado uma refatoração e criado um segundo sistema que utiliza microsserviços, foram separados o *front-end* do *back-end*, utilizado versionamento do desenvolvimento, hospedado o banco de dados e integrado o versionamento com o site de hospedagem.

4.1 Tecnologias Utilizadas

Para desenvolver o estudo prático foram utilizadas algumas tecnologias no *front-end* e *back-end*, sendo descritas as mais relevantes para o estudo da refatoração da arquitetura monolítica para uma arquitetura de microsserviços, dentre elas serão destacadas o html, css, javascript, algumas bibliotecas como jquery, a combinação do objeto *XMLHttpRequest* utilizando ajax e PHP que permite realizar a ligação do servidor, com a aplicação e banco de dados.

Além das tecnologias utilizadas na codificação, foi utilizado tecnologia de versionamento e banco de dados, no versionamento foi utilizando o git com a plataforma do github, e integrada a provedora de hospedagem na web hostinguer, o sistema de gerenciamento de banco de dados foi o mysql administrado pelo phpmyadmin.

4.2 PHP

De acordo com Leone (2017) linguagem PHP possibilita construir sistemas e sites de forma interativa, pois permite ser escrito de forma simples e ser embutido dentro do código HTML, mas é executado no servido viabilizando assim que possa ser feito qualquer tarefa, ele é responsável pela parte do *back-end*.

4.3 MySQL e phpMyAdmin

Para utilizar o banco de dados, foi necessário o MySQL como Sistema Gerenciador de Banco de Dados (SGBD) e o phpMyAdmin como programa para administrar o SGBD. Segundo Ricardo (2013) o MySQL possui fácil utilização, pode ser utilizado com uma variedade de sistemas operacionais e possui sua utilização com código aberto, se tornando essencial para trabalhar com uma grande quantidade de dados.

4.4 JavaScript, JQuery e Ajax

De acordo com Carvalho (2016) JavaScript é uma linguagem de programação utilizada na parte do *front-end*, ela permite realizar validações, acionar eventos e modificar de forma assíncrona a visualização e interação com o usuário, podendo também ser utilizada no *back-end* ao se utilizar programas como node.js. Ainda segundo Carvalho (2016) JQuery é uma biblioteca com funcionalidades do JavaScript que simplifica a maneira de se utilizar JavaScript, ajudando a precipitar futuros recursos e tornar o desenvolvimento mais rápido, por sua vez o ajax possiblita visualização e interações assíncronas dos dados.

4.5 Html, CSS e Bootstrap

Eis (2011) define HTML como uma linguagem de marcação de texto, utilizada em websites e que permite mostrar informações que se deve exibir, já o CSS é uma linguagem de estilo, possibilitando caracterizar a aparência ou seja o *front-end* do sistema.

Segundo Costa (2014) bootstrap é um framework que utiliza JavaScript, CSS e HTML, facilitando a criação de sites responsivos, adicionando funcionalidades ao código ao utilizar os seus componentes, ajudando a desenvolver uma interface mais diversificada, bonita e responsiva, além de ajudar a economizar tempo no desenvolvimento.

4.6 GuitHub e Hostinguer

Segundo Schmitz (2015) o git é um programa que permite gerenciar, versionar e colaborar o código de desenvolvimento, auxiliando o manter em ordem, se necessário recuperar e desenvolver. Por sua vez o github é um sistema que disponibliza mais funcionalidades que podem ser utilizadas no git, e permite hospedar o desenvolvimento para que outras pessoas possam colaborar.

O hostiguer foi o servidor escolhido para armazenar e hospedar o banco de dados e realizar a integração com o guithub, ao atualizar o código no guithub, é possível pressionando um botão no painel do hostinguer, atualizar o código que está hospedado, facilitando assim o versionamento e atualização do sistema.

4.7 Comparação entre o estudo prático monolítico e microsserviço

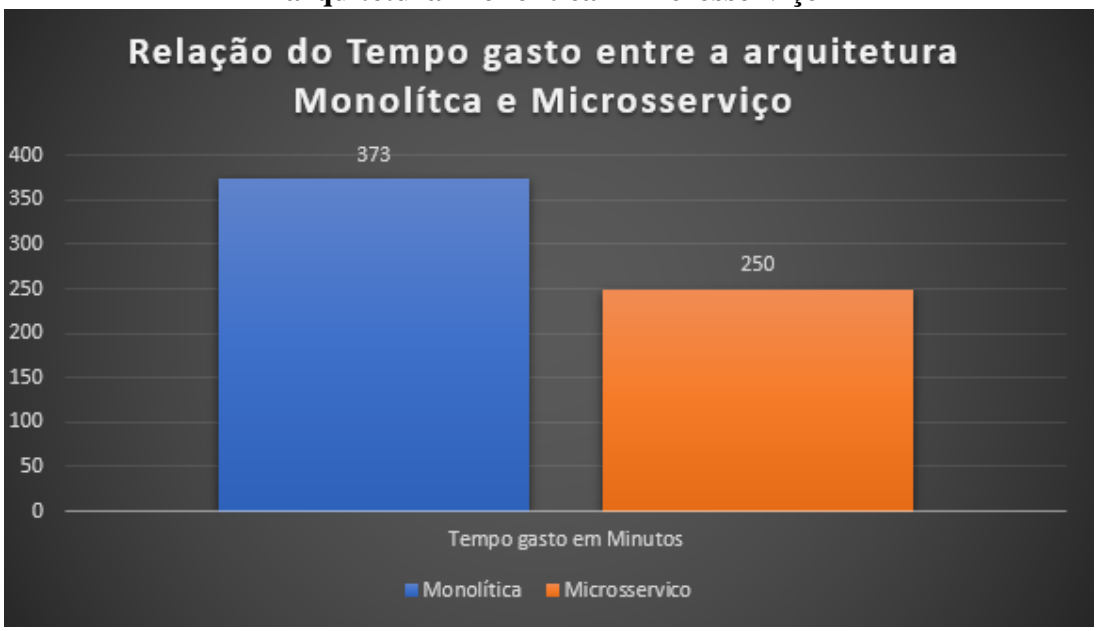
O desenvolvimento de um estudo prático foi essencial para transformar o conhecimento explícito sobre arquitetura monolítica e de microsserviços em um conhecimento tácito, além de permitir comparar as diferenças de ganhos e perdas entre as duas arquiteturas.

Em realação a arquitetura monolítica uma vantagem é que não tem muitas camadas no desenvolvimento do sistema, e seu desenvolvimento segue uma mesma tecnologia, o que facilita a questão de conhecer uma pequena gama de linguagem para programação. Por não ter muitas camadas e não utilizar linguagens diferente, se torna mais simples de desenvolver.

Em contra partida na arquitetura monolítica o código se tornou muito grande e acoplado, uma pequena falha faz com que todo o sistema pare, ficou difícil de dar manutenção, de utilizar novas tecnologias, bibliotecas e linguagens, aumentando o tempo de desenvolvimento e tornando o sistema extremamente acoplado.

Utilizando microsserviços foi reduzido o tempo de desenvolvimento, uma das razões dessa diferença se deve ao menor acoplamento que ocorre na arquitetura monolítica, como exemplo da redução do tempo, o Gráfico 1 mostra a imagem da comparação em minutos gasto para desenvolver a funcionalidade de cadastrar, pesquisar e editar professores. Na arquitetura monolítica demorou 6 horas e 13 minutos para ser desenvolvida a funcionalidade, já na arquitetura com microsserviço foi desenvolvido a mesma funcionalidade utilizando 4 horas e dez minutos, ou seja, obtive uma redução de aproximadamente 32.97% do tempo que foi gasto na arquitetura de monolítica.

Gráfico 1 - Tempo gasto em minutos para se desenvolver a tela de professores na arquitetura Monolítica x Microsserviço



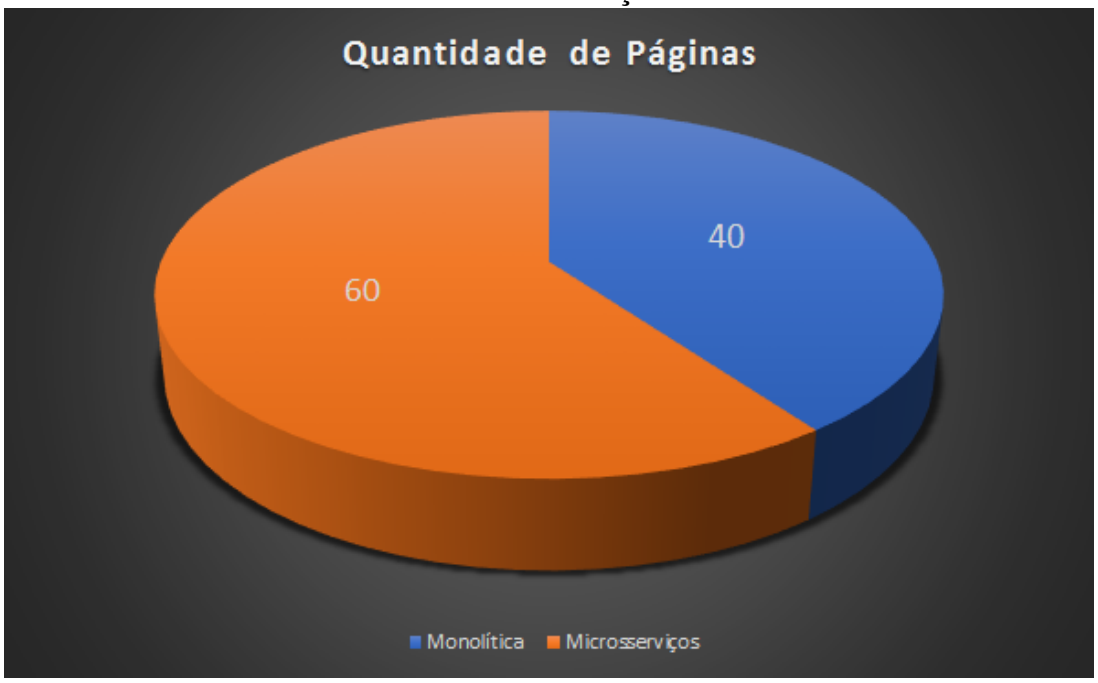
Fonte: Elaborado pelo autor.

Outro fator a se destacar é a diferença da quantidade de páginas entre as duas arquiteturas, a arquitetura de microsserviços atualmente conta com aproximadamente 60 páginas, em

relação a monolítica que possui cerca de 40 páginas, ambas compondo o front-end e back-end, sendo retiradas da contagem páginas de customizações *css* e configuração do banco de dados e servidor. A diferença pode ser notada no Gráfico 2.

A arquitetura de microsserviços gerou mais páginas em relação a monolíticas, mas tornou mais fácil sua manutenibilidade, versionamento e *deploy* do código para o ambiente de produção. Já a monolítica possui uma quantidade menor de páginas o que reduziu o tamanho do sistema, contudo se tornou mais acoplada, uma vez realizado o *deploy*, é necessário subir todo o conteúdo daquela página para o ambiente de produção.

Gráfico 2 - Quantidade de páginas aproximada da arquitetura Monolítica x Microsserviço



Fonte: Elaborado pelo autor.

Entre as vantagens de microsseviços foi possível notar que ao utilizar essa arquitetura, consigo dividir em partes o desenvolvimento e funcionalidades do sistema, o que permite dividir o software em funcionalidades específicas sem impactar as demais. Tornado assim o desenvolvimento e gerenciamento mais fácil por funcionalidades, onde pode ser criado microsserviços específicos.

Também viabiliza utilizar novas tecnologias, facilitando que problemas possam ser solucionados de forma mais simples, utilizando tecnologias mais adequadas. Outra vantagem é o baixo acoplamento do código que permite desenvolver trecho de código que executam funções específicas, tornado mais fácil sua manutenibilidade e evolução.

5 CONCLUSÃO

Discussão dos resultados obtidos na pesquisa. É onde se colocam as observações do autor. Poderá também apresentar sugestões de novas linhas de estudo.

A conclusão deve estar de acordo com os objetivos do trabalho.

A conclusão não deve apresentar citações ou interpretações de outros autores.

REFERÊNCIAS

CARVALHO, Johel. **JavaScript vs jQuery - Tudo o Que Você Precisa Saber**, 2016. Disponível em: <http://programadorobjetivo.co/jquery-vs-javascript-tudo-o-que-voce-precisa-saber/>. Acesso em: 17 de out. 2018.

COSTA, Gabriel. **O QUE É BOOTSTRAP?**, 2014. Https://www.tutorialwebdesign.com.br/o-que-e-bootstrap/>. Acesso em: 17 de out. 2018.

EIS, Diego. **O básico: O que é HTML?**, 2011. Disponível em: <https://tableless.com.br/o-que-html-basico/>. Acesso em: 17 de out. 2018.

ERL, Thomas. **SOA: princípios do design de serviços**. 1. ed. [S.l.]: O'Reilly Media, 2009.

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

FOWLER, Susan. **Microsserviços prontos para produção**. São Paulo: Novatec, 2017.

GHEEZZI, Carlos; JAZAYERI, Mehdi; MANDRIOLI, Dino. **Fundamentals of software engineering**. 2. ed. New Jersey: Prentice Hall, 2002.

GOLDENBERG, Mirían. **A arte de pesquisar: como fazer pesquisa**. 8. ed. Rio de Janeiro: Record, 2004.

LEONE, Leonello. **O que é PHP e porque você precisa aprender HOJE!**, 2017. Disponível em: <https://becode.com.br/o-que-e-php/>. Acesso em: 14 de out. 2018.

LEWIS, James; FOWLER, Martin. **Microservices**, 2014. Disponível em: <https://martinfowler.com/articles/microservices.html>. Acesso em: 3 de jun. 2018.

NEWMAN, Sam. **Building microservices: designing fine-grained systems**. Sebastopol: O'Reilly Media, 2015.

RICARDO, José. **Introdução ao MySQL**, 2013. Disponível em: <https://www.devmedia.com.br/introducao-ao-mysql/27799>. Acesso em: 17 de out. 2018.

SCHMITZ, Daniel. **Tudo que você queria saber sobre Git e GitHub, mas tinha vergonha de perguntar**, 2015. Disponível em: <https://tableless.com.br/tudo-que-voce-queria-saber-sobre-git-e-github-mas-tinha-vergonha-de-perguntar/>. Acesso em: 17 de out. 2018.

SOMMERVILLE, Ian. **Engenharia de software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

SOUZA, Fábio et al. **Estudo de viabilidade do uso de arquitetura orientada a microsserviços para maximizar o reaproveitamento de código**, 2015. Disponível em: <http://revistas.ung.br/index.php/computacaoaplicada/article/view/2407/2190>. Acesso em: 3 de jun. 2018.

WALES, Michael. **3 opções de carreira para desenvolvedores web: front-end vs. back-end vs. full stack**, 2017. Disponível em: <https://br.udacity.com/blog/post/front-end-vs-back-end-vs-full-stack-desenvolvedor-web>. Acesso em: 14 de out. 2018.