UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
MBA EM GESTÃO DA TECNOLOGIA DA INFORMAÇÃO E COMUNICAÇÃO

RAPHAEL HENRIQUE FERREIRA DE ANDRADE

MODELO DE MIGRAÇÃO DO MONOLÍTICO AO MICROSSERVIÇO

MONOGRAFIA

CURITIBA
2019

RAPHAEL HENRIQUE FERREIRA DE ANDRADE

MODELO DE MIGRAÇÃO DO MONOLÍTICO AO MICROSSERVIÇO

Monografia apresentada como requisito parcial para obtenção do título de Especialista em Gestão da Tecnologia da Informação e Comunicação da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Msc. Alexandre Miziara

CURITIBA

2019

Folha destinada à inclusão da **Ficha Catalográfica** por meio de solicitação ao Departamento de Biblioteca da UTFPR e posteriormente inserida nesse espaço: verso da Folha de Rosto (folha anterior).

Espaço para a ficha catalográfica sob responsabilidade exclusiva do Departamento de Biblioteca da UTFPR.

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Curitiba
Diretoria de Pesquisa e Pós-Graduação
VIII CURSO DE ESPECIALIZAÇÃO EM GESTÃO DE TECNOLOGIA DA INFORMAÇÃO E COMUNICAÇÃO

TERMO DE APROVAÇÃO

MODELO DE MIGRAÇÃO DO MONOLÍTICO AO MICROSSERVIÇO

Por

Raphael Henrique Ferreira de Andrade

Esta monografia foi apresentada às **19h** do dia **22/05/2019** como requisito parcial para a obtenção do título de Especialista no CURSO DE ESPECIALIZAÇÃO EM GESTÃO DE TECNOLOGIA DA INFORMAÇÃO E COMUNICAÇÃO, da Universidade Tecnológica Federal do Paraná, **Câmpus Curitiba**. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho:

1		Aprovado
2		Aprovado condicionado às correções Pós-banca, postagem da tarefa e liberação do Orientador.
3		Reprovado

_____

Prof. MSc. Bernadete M.V.F. Rosa
UTFPR – Examinador

_____

Prof. MSc. Alexandre Jorge Miziara
UTFPR – Orientador

_____

Prof. MSc. Alexandre Jorge Miziara
UTFPR – Coordenador do Curso

O documento original encontra-se arquivado no DAELN

AGRADECIMENTOS

Agradeço a minha família e amigos pelo apoio constante na conclusão desse trabalho, especialmente a minha esposa Aline Zerwes, aos meus pais e irmãos. Agradeço aos professores que me ensinaram ao longo do curso e aos professores Alexandre Miziara e Bernadete Voichcoski, pela orientação da monografia e a todos os colegas do curso na UTFPR.

“Não sou nada.
Nunca serei nada.
Não posso querer ser nada.
À parte disso, tenho em mim todos os sonhos
do mundo.”
Fernando Pessoa

RESUMO

De Andrade, Raphael H. F. **MODELO DE MIGRAÇÃO DO MONOLÍTICO AO MICROSSERVIÇO**. 2019. 35 f. Monografia (MBA em Gestão da Tecnologia da Informação e Comunicação) - Universidade Tecnológica Federal do Paraná. Curitiba, 2019

A definição de um padrão de arquitetura para sistemas é alvo de discussão de acadêmicos e instituições a fim criar formas que facilitam a manutenção, evolução e entendimento dos sistemas componentes. Microsserviços é um padrão recente que tem conseguido abordar todos esses objetivos de maneira eficaz e com conceitos baseados na simplicidade e automação. Este estudo visa realizar um estudo das técnicas e modelos de um sistema de informação de arquitetura na arquitetura de microsserviços e elucidar uma maneira de evoluir a arquitetura monolítica em um sistema existente, para o padrão de microsserviços, cobrindo desde a concepção até a montagem de equipes de trabalho. Para tal, foi realizada uma revisão sistemática da literatura nas principais bases de conhecimento científico para identificar quais são as peculiaridades relevantes nesse processo, da extração até a concepção dos novos microsserviços que constituem o novo sistema, baseando-se em estudos de casos e também em algoritmos estruturados para decomposição de funcionalidades dos sistemas monolíticos. O processo de revisão teve como foco identificar e explicar atividades e etapas que serão realizadas durante a conversão da arquitetura e também definir de maneira clara o que deve compreender um sistema na arquitetura de microsserviços, além de seus benefícios. Como resultado, este trabalho listou os principais pontos que devem ser executados para maximizar a chance de sucesso da migração de arquitetura para microsserviços. Baseando-se nas técnicas e etapas apresentas e no fato de que todas as atividades que exigem algum tipo de ferramenta possuem alternativas baseadas em código aberto. A implantação dessa nova arquitetura requer baixo investimento em infraestrutura e também em treinamento, tornando mais plausível e acessível para quem deseja realizar a implantação.

Palavras-chave: Microsserviço. Monolítico. Arquitetura de *software*. Migração.

ABSTRACT

De Andrade, Raphael H. F. **Monolithic to microservice migration model**. 2019. 35 p. Monograph (MBA in Information Technology and Communication Management - Federal University of Technology) - Paraná. Curitiba, 2019.

The definition of a standard system architecture is the goal of various academic and industry discussions in order to create ways to facilitate the maintenance, evolution and understanding of system's components. Microservices is a recent pattern that has covered all these goals in an effective way with concepts based on simplicity and task automation. This paper aims to explore technics and models of information systems on the microservices architecture and enumerate the best practices and tasks to convert a monolithic architecture from an existing software to microservices, going from the conception to the team work construction. To achieve this objective a systematic review on existing studies on the main scientific databases was performed with the following target, identify what are the peculiarity and common tasks with relevance on this process, from extraction until the conception and development of the new services that will be part of the new architecture, based on case studies and also structured algorithms to decompose the monolithic system's features. The review process focused on identifying and explaining the activities and steps which will be required during the conversion of the architecture and also to define in a clear and simple way in what should comprehend a microservices system architecture, besides its benefits. As a result, this paper listed the main point that should be covered to magnify the success chances of the microservices migration. Taking the steps and activities presented as a base and the fact that all activities that requires any type of external tool, have open source alternatives. The implementation of this new architecture requires low infrastructure and training investment, making it feasible and achievable to whoever desires to implement it.

Keywords: Microservices. Monolithic. Software Architecture. Migration

LISTA DE FIGURAS

Figura 1 – Fluxo de seleção de estudos..... 21
Figura 2 – Exemplo de diagrama de fluxo de dados puro..... 24
Figura 3 – Exemplo de diagrama de fluxo de dados decomponível e candidatos a microsserviços..... 24
Figura 4 – Desenho, compilação, implantação e execução..... 29
Figura 5 – Aplicações baseadas em estados..... 30

LISTA DE SIGLAS

API – *Application Programming Interface*

HTTP – *Hypertext transfer protocol*

DevOps – *Development and operations*

SUMÁRIO

1 INTRODUÇÃO 13
1.1 OBJETIVOS 14
1.1.1 Objetivo geral 14
1.1.2 Objetivos específicos 14
2 CONTEXTO E A REALIDADE INVESTIGADA 16
3 REFERENCIAL TEÓRICO 17
3.1 ARQUITETURA DE SISTEMAS 17
3.2 MICROSSERVIÇO 17
3.3 DEVOPS 17
3.4 REVISÃO DE LITERATURA 18
4 METODOLOGIA 19
5 RESULTADOS E CONTRIBUIÇÃO TECNOLÓGICA E/OU SOCIAL 22
5.1 PONTO DE PARTIDA 22
5.2 ORGANIZAÇÃO DE EQUIPES DE TRABALHO 25
5.3 IMPLEMENTAÇÃO – ARQUITETURA 26
5.3.1 Gerenciamento de código – repositório único 26
5.3.2 Dependências externas e configurações de sistemas 27
5.3.3 Do código a execução 28
5.3.4 Do Estado ao Sem Estado 29
5.3.5 Desenvolvimento e Operação 30
CONSIDERAÇÕES FINAIS 33
REFERÊNCIAS 34

1 INTRODUÇÃO

Nos últimos anos, a comunidade de engenharia de *software* tem visto uma tendência em direção a computação em nuvem (CITO et al, 2015). As mudanças nos aspectos de infraestrutura, colocaram uma demanda por uma arquitetura que elevasse o benefício das oportunidades dadas pela infraestrutura em nuvem e que facilitasse encarar os desafios de construir aplicações nativas em nuvem. Um estilo de arquitetura que tem atraído considerável atenção das companhias neste contexto – como explicado por Fowler, 2014 e Thones, 2015 – é a arquitetura de microsserviços.

Recentes implementações de microsserviços em ambientes em nuvem, tem elevado o conceito de arquitetura orientada a serviço a novos limites e tem sido embasado em objetivos como: velocidade, comunicação mútua, fácil adaptação e componentes de fácil adaptação (SILL, 2016).

A definição mais aceita do termo microsserviço é a elaborada por Martin Fowler e James Lewis: Microsserviço é o estilo de arquitetura com uma abordagem de desenvolver uma única aplicação composta por pequenos serviços cada um rodando seu processo próprio e comunicando-se entre si sobre mecanismos de baixo custo, na maioria dos casos utilizando um recurso de uma API via HTTP. Esses serviços são construídos se agrupando em capacidades de negócio e implantados independentemente e de forma automatizada. Há um mínimo de gerenciamento centralizado entre eles e podem ser construídos usando diferentes linguagens e fazendo uso de diferentes tecnologias de banco de dados (LEWIS et al, 2018).

Microsserviços são a solução - talvez a única solução - para o problema de construir e gerenciar, de maneira eficiente, sistemas de *software* complexos. Para pequenas e médias empresas eles podem entregar redução de custo, aumentar qualidade, agilidade e reduzir o *Time to Market* (tempo que um produto ou serviço leva da concepção até o lançamento para o público consumidor. Para grandes empresas eles fundamentalmente mudam as regras do jogo (SINGLETON, 2016).

Desde que aplicações corporativas começaram a ser usadas em larga escala o padrão de arquitetura que se tornou padrão de forma orgânica foi o modelo centralizado, onde um grande módulo era responsável por servir a todas as requisições e funcionalidades, este modelo passou a ser chamado recentemente de monolítico, fazendo referência ao monólito, uma grande pedra pesada.

Aplicações monolíticas convergem para um ponto único ao longo do tempo dentro e médias e grandes empresas, elas ficam muito grandes e concentradas, tornando difícil e

complexo o gerenciamento de mudanças, refatorações e modernização. Além disso, o custo para implantação de pequenas alterações se assemelha muito ao custo de grandes mudanças e melhorias, ao passo que a correção de pequenos defeitos na aplicação pode levar horas e requerer o engajamento de vários profissionais trabalhando em conjunto. Esses fatores fazem com que o tempo de resposta ao mercado, de grandes companhias, fique grande e de alto custo, ocasionando em perda de mercado pela demora em lançar um produto ou melhoria para seus clientes.

Assim sendo, não é surpresa que grandes companhias como *Google, Ebay* e *Netflix*, - mais recentemente outras como Mastercard, Itaú e empresas menores também – tem realizado sérios esforços para mover suas aplicações monolíticas legadas para uma arquitetura orientada a microsserviços (MAZLAMI et al, 2017). O problema comum de todas as companhias é que, identificar quais componentes das aplicações legadas podem ser convertidas, de forma coesa, em unidades de microsserviço independente, é uma tarefa tediosa e manual que requer análises em várias dimensões na arquitetura do sistema e cai fortemente no conhecimento e experiência de quem os está fazendo (MAZLAMI et al, 2017).

O intuito deste estudo é analisar as formas de realizar essa migração de forma coesa e coordenada, apresentando cenários e sugestões para a realização bem sucedida da modernização dos sistemas legados na arquitetura de microsserviços, tendo como contexto empresas classificadas como de porte grande e médio dentro do cenário brasileiro e demonstrar quais benefícios essas empresas podem alavancar com essa migração.

1.1 OBJETIVOS

1.1.1 Objetivo geral

Este estudo visa realizar um estudo das técnicas e modelos de um sistema de informação de arquitetura monolítica na arquitetura de microsserviços, de forma a elucidar uma maneira de evoluir a arquitetura monolítica em um sistema existente, para o padrão de microsserviços, cobrindo desde a concepção até a montagem de equipes de trabalho.

1.1.2 Objetivos específicos

- Analisar através de uma revisão sistemática de literatura os estudos que propõem modelos e técnicas de migração de arquitetura monolítica para microsserviços.

- Baseado nos estudos dos modelos existentes, gerar um conjunto de boas práticas alinhadas com a realidade das empresas de médio e grande porte no Brasil, para realizar a migração da arquitetura monolítica para a arquitetura de microsserviços. Esse conjunto de boas práticas, pode ser visualizado no item 5 desse trabalho.

2 CONTEXTO E A REALIDADE INVESTIGADA

No Brasil, são consideradas empresas de médio porte, aquelas que possuem entre 50 e 99 funcionários (100 a 499 para a indústria) e grandes empresas aquelas que possuem mais de 100 funcionário (500 para a indústria), de acordo com o SEBRAE.

Em 2014 um estudo do IBGE constatou que há por volta 1 milhão de pessoas empregadas no setor de serviços de tecnologia da informação e comunicação (IBGE, 2014).

Neste contexto este estudo visa estudar os aspectos abaixo:

- Existentes formas implantação e migração da arquitetura de microsserviços e trazê-las para a realidade dessas empresas de forma que fique sintetizadas as etapas necessárias para essa implantação ou migração.
- Elucidar os benefícios provenientes da agilidade e adaptabilidade que essa nova arquitetura pode trazer para empresas que produzem ou consomem *softwares* em qualquer uma das áreas de negócio.

Além dos benefícios que essa nova arquitetura pode oferecer para quem a implanta, a motivação maior para este estudo é a quantidade de artigos publicados e por consequência a quantidade de conhecimento existente no Brasil nesse tema.

3 REFERENCIAL TEÓRICO

3.1 Arquitetura de sistemas

A organização dos módulos internos, a comunicação com componentes externos, interfaces de rede e diretamente com o consumidor, os padrões de comportamento e ferramentas adicionais utilizadas, assim como restrições e propriedades físicas e lógicas. Todos esses elementos e características definidos é o que caracteriza a arquitetura de um sistema. Portanto, por arquitetura de sistemas se entende a forma como um sistema é concebido, organizado e também utilizado.

3.2 Microsserviço

Nascido do modelo de arquitetura orientada a serviços (SOA), microsserviços são pequenas aplicações com uma única responsabilidade de negócio que pode ser implantada e escalada e testada independentemente (XABIER et al, 2018). Essa decomposição de uma aplicação monolítica, aplicações cuja regras de negócios não podem ser gerenciadas independentemente, mesmo que agrupadas em módulos, alinhada com técnicas de automatização de implantação, testes e compilação gera uma agilidade e simplicidade no momento da atualização de funcionalidades que compreendem o sistema.

3.3 *DevOps*

O termo *DevOps* caracteriza a integração de dois mundos dentro das áreas de tecnologia da informação, desenvolvimento e operações, através da automação do desenvolvimento, implantação e monitoramento da infraestrutura. Além de técnicas e ferramentas para automação trabalhando juntas, é uma mudança organizacional, onde, ao invés de times separados e distribuídos, equipes multifuncionais trabalham juntas para a entrega contínua de funcionalidades (EBERT et al, 2016).

O emprego de ferramentas para automatizar atividades relativas as etapas do desenvolvimento, implantação e suporte de sistemas torna-se mandatório para a correta adoção de *DevOps.* Portanto, a escolha dessas ferramentas é de extrema importância para esse processo. Além do benefício da execução em menor tempo das atividades, existe a segurança, do ponto de vista de qualidade, já que há uma garantia de que todas as execuções terão sempre o mesmo resultado e qualidade, garantindo máxima aderência e confiança.

3.4 Revisão de literatura

Uma das ferramentas mais utilizadas para suportar uma pesquisa baseada em evidências em outros domínios de conhecimento é a geração de uma revisão sistemática de literatura, usada para agregar experiências em uma determinada extensão de diferentes estudos para alcançar um objetivo de pesquisa específico (BUDGEN et al, 2006).

A revisão sistemática de literatura é uma das formas mais consolidadas de revisar e interpretar todas as pesquisas disponíveis sobre um determinado tema ou questão científica (KINCHENHAM, 2004).

Há alguns fatores que diferenciam uma revisão sistemática de uma revisão não sistemática, o principal deles são alguns protocolos que são seguidos ao longo de sua execução. Dentre eles o primeiro é: um ponto de partida. Uma pergunta científica que norteia as buscas realizadas ao longo do processo de revisão.

A estratégia de busca deve ser documentada, afim de possibilitar ao leitor que possa refazer a busca e validar quão rigoroso foi o processo de escolha dos estudos analisados.

Uma revisão sistemática engloba a atividade de explicitar os fatores que são usados para inclusão e exclusão de estudos que serão analisados na íntegra.

Outro ponto ainda documentado em uma revisão sistemática, é a informação que se objetiva obter através da leitura dos estudos selecionados e qualquer outro critério de qualidade que será usado, afim de considerar ou não, o documento do estudo em questão.

4 METODOLOGIA

Para a realização deste trabalho foi utilizado a revisão sistemática de literatura à fim de explorar de maneira metodológica e organizada o conhecimento existente nas áreas relacionadas. Foi efetuada uma busca na base de dados Scielo, a qual compreende os principais publicadores de pesquisa científica no Brasil e também da américa latina. O resultado de artigos disponíveis encontrados foi de zero publicações. Também foi efetuada uma busca no portal eletrônico Periódicos Capes e foram encontrados 9 resultados, até a data de 10 de janeiro de 2019. Desses 9 resultados, 8 foram descartados baseados na leitura do título, pois a área de pesquisa não era relacionada com a computação. O resultado restante, um apenas, foi descartado após a leitura do resumo, pois o alvo do estudo era a utilização de sistemas de informação em microempresas, por último a busca foi realizada na base de trabalhos científicos IEEE Xplore e foram encontrados 35 artigos que referenciam o tema pesquisado.

O processo da revisão conduzida neste estudo englobou três atividades elementares: planejamento da revisão, condução da revisão e o relatório do resultado da revisão. Na etapa de planejamento foi determinado o termo que seria usado como norte para a as pesquisas nas bases e também qual o período que seria usado, além de outros parâmetros, como bases de dados científicas a serem verificadas, critérios de inclusão e exclusão. Na tabela 1, estão discriminados os parâmetros e seus valores aplicados nessa revisão.

Tabela 1 - Parâmetros utilizados na seleção de artigos.

Parâmetros	Valores utilizados
Expressão de consulta	((*monolithic*) *AND* (*microservice OR micro-service*)) *OR* ((*microservice OR micro-service*) *AND* (*architecture*)) e ((monolítico) *AND* (microsserviço *OR* micro-serviço)) *OR* ((microsserviço *OR* micro-serviço) *AND* (arquitetura))
Período utilizado	2008 até 2018
Bases de dados	IEEE Xplore, Scielo e Periódicos Capes
Tipos de documentos analisados	Artigos de jornais e revistas e capítulos de livros
Critérios de inclusão – Assunto	Arquitetura de sistemas, arquitetura de *software*, arquitetura de microsserviços e engenharia de *software*

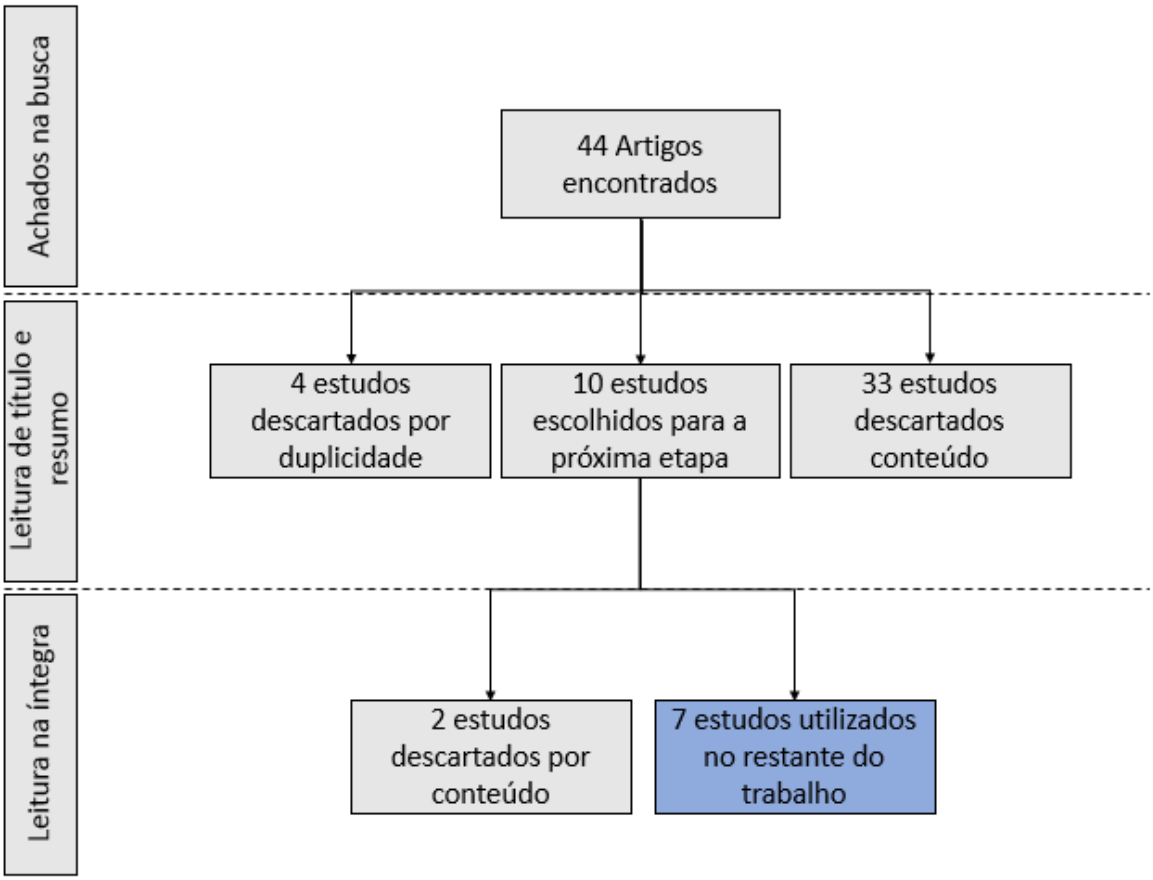
Fonte: O autor (2019)

As buscas efetuadas nas bases descritas retornaram um total de seis artigos, sendo todos eles encontrados somente na base *IEEE Xplore.* Esse processo foi conduzido no mês de janeiro de 2019.

Afim de ser considerado no estudo final, o artigo ou capítulo de livro selecionado deve abordar a migração ou comparação entre arquitetura de *software* monolítica e a arquitetura de microsserviços. A leitura dos estudos que abordam a comparação entre as duas estratégias foi considerada pela existência da possibilidade de o conteúdo trazer aspectos importantes de ambos os padrões de arquitetura, que podem ser levados em consideração no momento de definição de qual das técnicas serão utilizadas para a implementação ou migração do *software* existente ou pretendido, para novos sistemas. Além disso, outro aspecto importante que foi considerado, é a existência de evidências nas conclusões geradas pelo autor no momento da consolidação do estudo.

Na etapa de condução da revisão sistemática da literatura, todos os artigos encontrados pelos descritores utilizados foram selecionados para leitura do título e resumo. Após esta etapa, alguns estudos foram desconsiderados para a etapa de aprofundamento do estudo devido ao fato de não apresentarem os pré-requisitos apresentados anteriormente. Os achados remanescentes prosseguiram para leitura na íntegra e, por conseguinte, foram avaliados ao longo da etapa de execução da etapa de condução, conforme mostrado figura 1.

Figura 1 – Fluxo de seleção de estudos.



Fonte: O Autor (2019)

Após a etapa de condução e a leitura na íntegra dos nove artigos selecionados, os que mostraram relevância e coerência, baseando-se no objetivo deste estudo, fizeram parte do relatório final. Por conseguinte, foram utilizados como base para uma espécie de arcabouço para realizar a migração de arquitetura de aplicações monolíticas para microsserviços.

Para a extração de informação, dois pontos principais foram levados em consideração: aspectos citados por mais de um estudo e também aspectos novos trazidos pelos artigos desde que comprovados as suas eficácias diante do estudo proposto pelo autor na ocasião, ou seja, dados concretos que embasassem as conclusões apresentadas.

Por último, foram considerados aspectos apresentados que fossem ao encontro da realidade das empresas de médio e grande porte no Brasil, como poder de investimento, realidade econômica atual e também estratégia para inovação tecnológica a médio e longo prazo. Todos esses pontos serão apresentados na etapa de resultados deste estudo.

5 RESULTADOS E CONTRIBUIÇÃO TECNOLÓGICA E/OU SOCIAL

Para que o processo de leitura e análise dos estudos, tendo como norte a identificação de um conjunto de práticas, ferramentas e atividades, possam auxiliar instituições ou empresas que planejam a modernização do seu portfólio de sistemas objetivou levar em consideração atividades sem custo adicional para o dia a dia dessas instituições, como por exemplo, licenças de *softwares*, contratação adicional de força de trabalho e também custos com consultorias externas.

Portanto, ao longo das etapas apresentadas algumas ferramentas tecnológicas auxiliares serão indicadas por possuírem livre acesso ao código fonte ou ao uso empresarial, pois isso pode ser um fator decisivo das partes envolvidas e interessadas na migração de uma aplicação monolítica para microsserviço. Levando-se em consideração que esta tecnologia ainda está em estado incipiente principalmente no cenário brasileiro, essas ferramentas facilitarão a implantação dessa arquitetura.

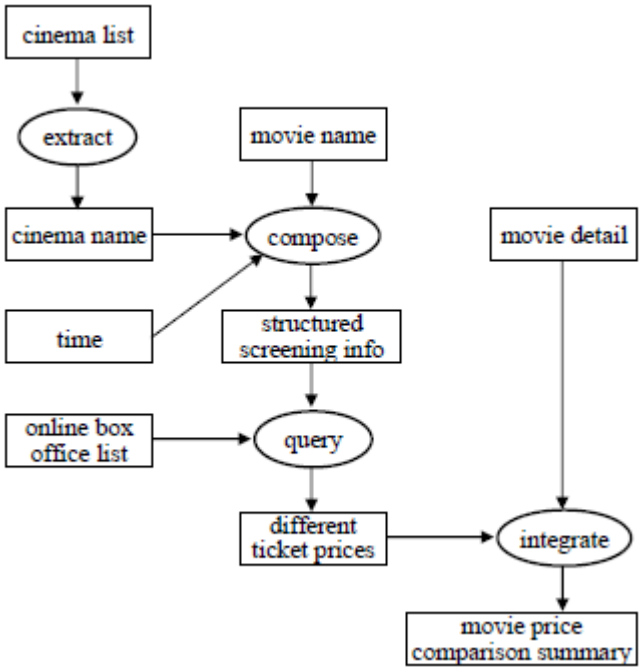
5.1 PONTO DE PARTIDA

Um dos principais problemas apresentados no momento de planejar e criar um mapa a longo prazo para a conversão de uma arquitetura para a outra, microsserviço, é identificar por onde deve-se começar essa conversão. Um dos principais pontos a serem considerados nesta etapa é uma análise puramente focada na funcionalidade e sua aplicabilidade do ponto de vista de regras de negócios existentes na aplicação. Essa estratégia é comumente chamada de migração guiada pelo negócio. Nela, funcionalidades completas e/ou compartilhada por várias outras, são selecionadas como pré-candidatas a novos microsserviços. Posteriormente esses pré-candidatos podem ser acoplados a outros serviços ou serem considerados a mais uma etapa de quebra em dois ou mais microsserviços, tanto por serem grandes demais para apenas um serviço ou serem igualmente requisitados por várias funcionalidades (BUCCHIARONE et al, 2018).

Após esse levantamento e armazenamento de funcionalidades candidatas, é importante realizar o refinamento do relacionamento entre as partes componentes da aplicação. Com a realização desse estudo do sistema, é possível identificar como é o fluxo da informação da entrada, processamento, transformação e saída para outros componentes ou usuários. Um dos métodos que possibilitam essa visão das funcionalidades e o fluxo de dados entre elas, é um diagrama de fluxo de dados que pode ser usado para representar sistemas monolíticos da perspectiva de processos de negócio (CHEN et al, 2017).

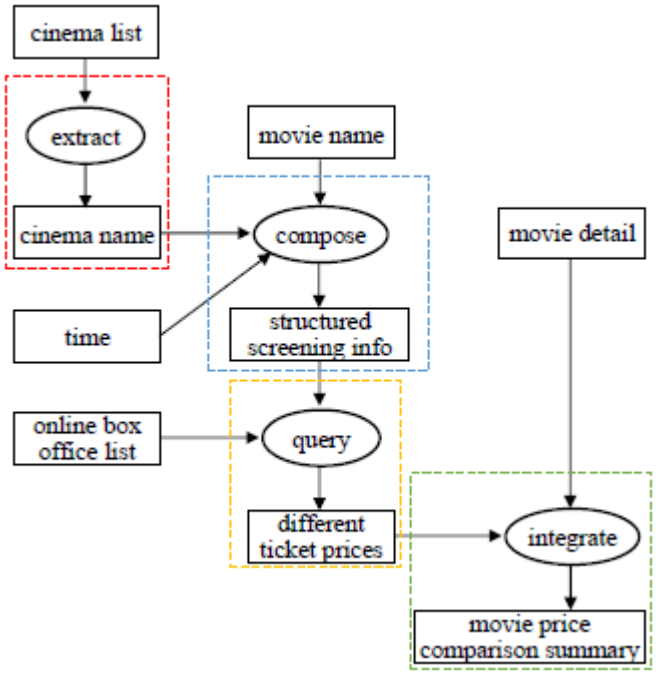
A composição desse diagrama de fluxo de dados pode ser dividida em três etapas que consistem em: construção do diagrama puro, transformação em um diagrama decomponível e decomposição do diagrama em candidatos a microsserviços. A forma como o diagrama pode ser decomposto é definido a partir dos relacionamentos estabelecidos entre as funcionalidades e a forma como o fluxo se agrupa. A Figura 2 representa um exemplo simples de um sistema de *streaming* de vídeo, enquanto na Figura 3 é possível observar os candidatos a microsserviços selecionados para o exemplo da Figura 2.

Figura 2 – Exemplo de diagrama de fluxo de dados puro



Fonte: Chen (2017)

Figura 3 – Exemplo de diagrama de fluxo de dados decomponível e candidatos a microsserviços.



Fonte: Chen (2017)

As etapas citadas são processos manuais e que exigem um nível de conhecimento da aplicação, não somente do ponto de vista de arquitetura e modularização, mas também de processos de negócios dos quais a aplicação em questão faz parte. Para o mapeamento das funcionalidades e também o desenho dos diagramas, é possível utilizar pacotes específicos para escritórios, como planilhas eletrônicas e processadores de texto ou programas específicos para desenho de fluxogramas. Dentre as opções sem custo disponíveis, o *LibreOffice* oferece suporte para planilhas eletrônicas e também para desenho de fluxograma em seu pacote.

Outra abordagem mais complexa para a análise do sistema monolítico existente e posteriormente a extração de possíveis modelos de decomposição em microsserviços é a análise do código existente. Neste cenário, é ideal que o código seja mantido com o auxílio de uma ferramenta de controle de versão de código. Um estudo conduzido por Mazlami, Cito e Leitner (2017) da Universidade de Zurich, propõe algoritmo de extração de serviços novos a serem implementados na nova arquitetura. Esse estudo apresenta formas de analisar o código existente, baseando-se nas relações entre as classes e módulos existentes, nas alterações que o código fonte sofreu, analisando a sua linha do tempo e também no time que introduziu essas alterações.

O intuito desse algoritmo é extrair do código fonte um grafo de relacionamento entre as classes com arestas que possuem peso entre as ligações dos vértices (classes) que definem o acoplamento entre elas. Posteriormente a extração do grafo do sistema monolítico é realizada uma etapa de agrupamento que pode é realizada com base no peso das arestas. O objetivo do agrupamento é deixar vértices ligados por arestas de peso maiores e eliminar as arestas de peso menor. Para a determinação do peso das arestas algumas abordagens são recomentadas, como ligação lógica entre as classes, que seria uma dependência direta entre elas; ligação semântica, onde leva-se em consideração o contexto de negócio que essa classe está inserida; ligação através de contribuidores em comum que considera que times similares trabalham comumente em partes isoladas dos sistemas. Portanto considera-se que classes alteradas pelos mesmos desenvolvedores possuem uma interseção de contexto.

Todas essas abordagens citadas acima são maneiras de estabelecer um estudo prévio da aplicação existente, afim de extrair candidatos por onde iniciar a conversão da arquitetura, de monolítica para microsserviços. Além de um ponto de partida elas podem gerar um entendimento mais profundo do sistema em estudo. Após essa etapa concluída é possível um planejamento e priorização de quais serviços são mais importantes e quais possuem mais riscos no momento da reescrita dos mesmos, facilitando assim, uma decisão do ponto de vista técnico e de negócio, sendo possível maximizar o ganho obtido com a migração gradativa do sistema.

5.2 ORGANIZAÇÃO DE EQUIPES DE TRABALHO

Considerando um microsserviço como um sistema com uma única responsabilidade e totalmente independente do ponto de vista técnico e também de negócio, alguns princípios adicionais também ajudam a definir essa arquitetura, alguns deles como: Modelagem através de conceitos de negócio, descentralização em todos os âmbitos (CHEN et al, 2017).

Esses princípios devem também ser considerados durante a reorganização dos times que trabalharão em cada implementação de microsserviço. Uma prática importante é que cada serviço deve ser implementado e gerenciado inteiramente por um único time, objetivando o desenvolvimento com um desenho de arquitetura de forma modular e com baixo acoplamento (BUCCHIARONE et al, 2018).

Organizacionalmente, o ambiente de trabalho deve estar preparado para mudanças, pois a implementação de microsserviços traz consigo uma nova cultura de trabalho. As equipes também devem estar preparadas para reestudar o sistema constantemente e estarem prontas para o surgimento de novas equipes não baseadas somente em desenvolvimento, mas também, em

automação e de operações, trabalhando juntamente com os arquitetos e desenvolvedores (LARRUECA et al, 2018).

Microsserviços são, por conceito, desenvolvidos e mantidos de forma autônoma, facilitando assim, que metodologias ágeis sejam aplicadas como metodologia de gerenciamento projeto e para que isso se torne realidade, as equipes devem ser familiarizadas com métodos ágeis, assim como a gerência dessas equipes, que devem facilitar que essas metodologias sejam aplicadas.

5.3 IMPLEMENTAÇÃO – ARQUITETURA

Após as realizações do trabalho de identificação de quais partes do sistema serão convertidas em microsserviços e quais times serão responsáveis pelo seu desenvolvimento, ainda antes de iniciar a escrita dos códigos, é necessário conceber a estrutura que será responsável pela orquestração desses microsserviços.

Uma empresa chamada *Heroku* fundada em São Francisco no estado da Califórnia, Estados Unidos da América, tornou público o seu processo de modernização de sua plataforma de sistemas e arquitetura e também de desenvolvimento de construção de uma plataforma como serviço no ambiente em nuvem. Esse processo ficou conhecido como *The Twelve Factor-App* (WIGGINS, 2017) e consiste em doze fatores que são considerados imprescindíveis no momento da concepção de uma aplicação baseada em arquitetura de microsserviços.

Esses fatores são hoje a base de qualquer instituição que está planejando o início da sua jornada em direção a microsserviços

5.3.1 Gerenciamento de código – repositório único

Quando se trata das inúmeras tarefas de um time de desenvolvimento, dentre elas o gerenciamento de código e de artefatos são, por muitas vezes, negligenciados. Porém, a correta aplicação de disciplina e organização nesses detalhes podem significar a diferença entre uma implantação que dura um mês ou um dia apenas (HOFFMAN, 2016).

O gerenciamento e o controle da versão do código é o primeiro ponto da arquitetura de um sistema, no qual deve-se focar a atenção, e é de suma importância que versões da aplicação sejam rapidamente ligadas a uma revisão direta do código fonte, sendo assim fácil de controlar, alterar, corrigir ou evoluir uma versão específica do sistema com base na necessidade de mercado e também com base nos objetivos do time de desenvolvimento.

A ferramenta de controle de versão mais utilizada e também mais completa atualmente é disponibilizada através do código aberto e possibilita criar repositórios para realizar o controle versão do código em qualquer tipo de ambiente, seja em nuvem, servidores próprios, sejam eles livres ou proprietários, essa ferramenta chama-se *Git (*https://git-scm.com/). Além disso, há algumas interfaces adicionais que podem ser incorporadas ao *Git* para uma fácil visualização da árvore de evolução do código, consulta a revisões e gerenciamento de ramificações do código. Dentre elas, o *Gitlab* (https://about.gitlab.com/) é uma dessas interfaces que facilitam o uso no dia a dia provendo um sistema acessível pelo navegador com uma interface amigável para o usuário.

5.3.2 Dependências externas e configurações de sistemas

Aplicações baseadas em nuvem são uma evolução do clássico modelo de aplicação empresarial e com isso, essas aplicações devem evoluir para poder tirar vantagem da agilidade do ambiente em nuvem. Isso significa que enquanto concebidas as aplicações não devem esperar que um servidor, uma aplicação container, irá prover tudo o que é necessário para a sua execução. Esses requisitos elas devem trazer consigo, acopladas ao seu artefato, tudo o que é necessário para a sua implantação e execução, devem também trazer as suas dependências (HOFFMAN, 2016).

Essa necessidade já inclina o time de desenvolvimento e arquitetura a escolher entre algumas das linguagens de programação que não se baseiam em modelos de aplicações container (servidores de aplicação), pois para essas linguagens o artefato executável é dependente de uma infraestrutura, na qual ela irá se basear para receber o restante de suas dependências.

Em conjunto com o correto gerenciamento das dependências as configurações também requerem devida atenção e devem ficar o mais longe possível do código. Em suma, configurações devem ser tratadas como extremamente voláteis e, portanto, de fácil alteração. Portanto é muito importante haver um entendimento claro da diferença entre dependência e configuração e assim poder manter uma o mais perto possível (dependência) e a outra o mais longe possível (configuração).

Das linguagens atuais que suportam esses paradigmas para configuração e dependência alguma delas são, *Golang* (desenvolvida pela equipe da *Google Inc.*), *Ruby* e *Java* com *Spring Boot.* Todas elas já possuem estrutura pronta para uso, para trabalhar com o correto gerenciamento de dependência e configuração.

5.3.3 Do código a execução

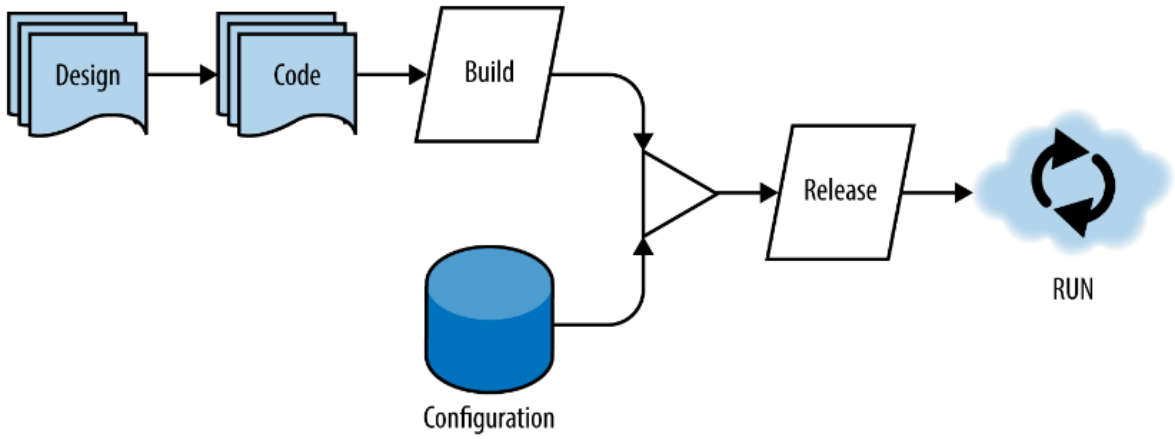
É considerado que uma aplicação vá do código fonte até a sua execução através de três estágios: compilação, implantação e execução. No momento da compilação, o código fonte é convertido em um único artefato interpretável diretamente pelo servidor, ou por alguma camada de máquina virtual, em caso de linguagens baseadas em interpretação. A implantação, é a combinação do artefato gerado pela compilação com as configurações do ambiente no qual a aplicação está sendo executada, ao passo que um mesmo código fonte executando com configurações diferentes são duas implantações diferentes. Por fim, a execução é o estágio final, onde a implantação é disponibilizada para o usuário final, seja ele outras aplicações ou uma pessoa física.

É de extrema importância que esses três estágios sejam estritamente separados, ao passo que seja impossível uma alteração de código enquanto a aplicação esteja executando (WIGGINS, 2017). Essa separação é um dos fatores predominantes para a construção de microserviços. Além dessa separação, é também crucial, que a realização de uma etapa prévia de *design* para um padrão de arquitetura seja seguido entre os vários microsserviços que irão compor a aplicação como um todo (HOFFMAN, 2016). Figura 4.

Não há dados de um número médio de microsserviços que compõem uma aplicação com essa arquitetura, porém, a quantidade de artefatos sendo movimentados para o funcionamento da aplicação como um todo é considerável, a automação de tarefas é inevitável ou o gerenciamento desses serviços se toram inviáveis (BUCCHIARONE et al, 2018).

Para a automação dos estágios de compilação, implantação e execução há ferramentas que auxiliam, possibilitando a customização de roteiros de execução para poder atender as necessidades da linguagem utilizada para o desenvolvimento bem como do ambiente de execução. Seja ele em nuvem ou infraestrutura tradicional. Em especial para automatização da compilação e geração de artefatos, os exemplos são *Maven (*https://maven.apache.org/), *Rake (*https://rubyonrails.org/*)* e *Gradle (*https://gradle.org/). Todos esses possuem acesso e código livre.

Figura 4 – Desenho, Compilação, Implantação e Execução.



Fonte: Hoffman (2016)

5.3.4 Do Estado ao Sem Estado

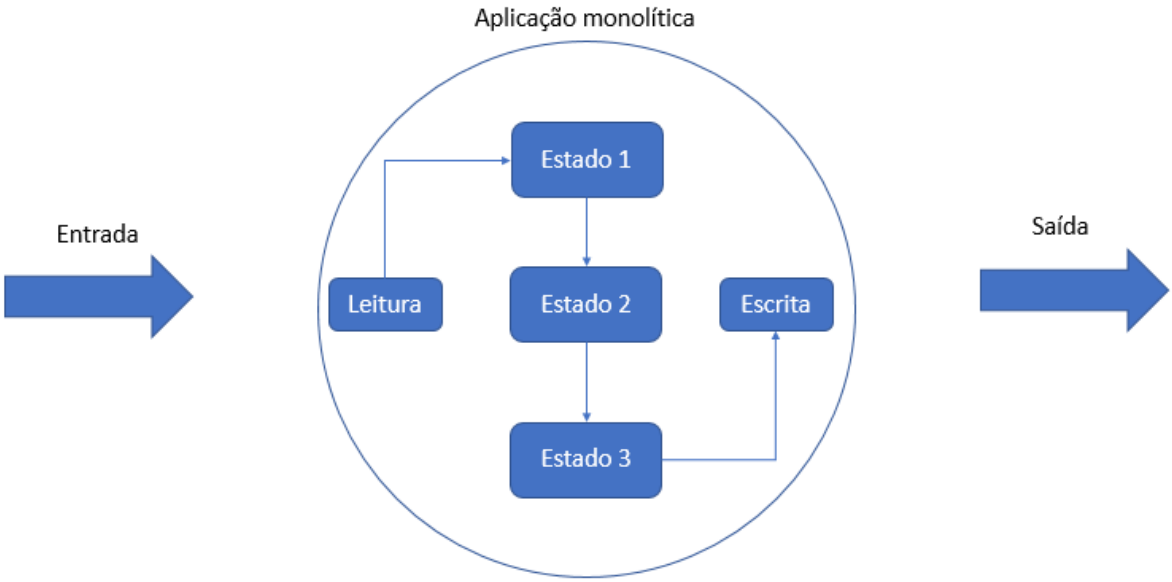
Uma das características de aplicações monolíticas é o fato de manter um estado o qual caracteriza uma determinada ação que lhe foi enviada. Esse estado, geralmente caracteriza a transição ou transformação de uma informação que foi recebida até a sua saída. Figura 5.

Essas informações do estado atual e do estado o qual a aplicação já esteve são informações voláteis dentro da aplicação, ao passo que, ao ser desligada ou interrompida esse estado se perde e com ele as informações que os caracterizava. Esse paradigma deriva das linguagens orientadas a objetos e o modo que os sistemas são desenhados ao seu redor, cada classe possui uma quantidade de propriedade e os valores dessas propriedades caracterizam o atual estado daquele objeto definido pela sua classe. Mesmo em aplicações que possuem mais de uma instância em execução, para balanceamento de carga, essa estratégia se mantém e cada instância é responsável pelo seu próprio estado. Mesmo que se tratando da mesma aplicação sendo executada com os mesmos parâmetros.

Essa arquitetura baseada em estado não é um problema em sistemas monolíticos, pois elas são desenhadas para ficarem em execução pelo maior tempo possível, sem interrupções. As interrupções ocorrem em caso de alguma necessidade de alteração por correção ou por evolução, em caso de não atender algum requisito necessário. Porém essa estratégia não funciona com microsserviços, que idealmente são aplicações que não mantém estado. Isso por que para explorar todos os benefícios da computação em nuvem, especialmente, tais como

elasticidade por demanda, balanceamento de carga, alta disponibilidade e alta disponibilidade através de redundância, cada serviço deve ter a habilidade de ser interrompido, iniciar e estar disponível mesmo sem ter acesso às informações requisitadas ou enviadas antes a sua existência (FURDA *et al*, 2018).

Figura 5 - Aplicações baseadas em estados.



Fonte: O autor (2019).

Uma aplicação sem estado não deve presumir sobre o contexto de ações anteriores baseadas em memória enquanto, está processando uma requisição. Porém não é proibido que a aplicação crie dados provisórios enquanto está executando. Mas esses dados devem ser descartados após a requisição ser concluída, pois a arquitetura dentre os serviços deve assumir que outra instância desse mesmo serviço pode receber a nova requisição do mesmo cliente ou até mesmo que, a atual instância, pode ser desligada a qualquer momento devido á escalabilidade horizontal (crescer em números ao invés de tamanho).

5.3.5 Desenvolvimento e Operação

Segundo a maior ferramenta de indexação e busca da internet, da empresa *Google Inc*, microsserviços de *DevOps* são conceitos em crescimento em taxa igual, desde 2014 (BALALAIE *et al*, 2016). Embora técnicas de *DevOps* possam ser aplicadas também a

aplicações monolíticas, a arquitetura de microsserviços facilita uma implantação efetiva, acima de tudo, por promover a importância de times de pequeno porte.

Embora já abordado em alguns aspectos anteriores, a automação no processo da integração entre o desenvolvimento e a operação e suporte das aplicações se torna ainda mais crucial quando no contexto da manutenção da aplicação durante todo o processo, da compilação ao monitoramento.

Na linha do tempo, para chegar de um ponto ao outro, algumas etapas requerem um pouco mais atenção, dentre elas é a automatização da compilação e geração de artefatos, já citada na seção 6.3.3 desde estudo. Além dela, a integração contínua de cada código novo criado com os já existentes facilita a análise do ponto de vista de qualidade, já que ela antecipa a identificação de possíveis problemas que os novos códigos e serviços podem gerar. Das opções disponíveis, sem a necessidade de adquirir licença ou pagar por tempo de uso ou consumo de recursos, as que tem sido mais utilizada são: *Jenkins (*https://jenkins.io/), *Concourse (*https://concourse-ci.org/*)* e *Drone (*https://drone.io/).

Para o processo de implantação, mais de uma ferramenta é necessária, já que é preciso realizar a combinação de diferentes fatores para que a aplicação seja totalmente implantada em um ambiente, seja ele de produção ou para testes. É o gerenciamento das configurações pertinentes a cada sistema e ambiente no qual ele será executado, para o gerenciamento automatizado das configurações e seus respectivos ambientes, ferramentas como *Chef (*https://www.chef.io/*)* e *Puppet (*https://puppet.com/*)* possuem todas as funcionalidades necessárias para essas atividades, e possuem livre acesso.

Uma das funções pertinentes às tradicionais equipes de operações, em se tratando de sistemas de computadores, é manter a aplicação funcionando máxima disponibilidade e aderência ao que lhe é proposto. Para tal, é necessário manter constante monitoramento da aplicação e também das informações presentes nos registros de execução gerados pela aplicação da forma tradicional, estes são armazenados em arquivos de texto e podem ser consultados ou indexados para uma posterior consulta mais rápida e eficiente.

No cenário de uma aplicação monolítica, mesmo que com mais de uma instância, é razoávelmente fácil consultar, os registros gerados pelas instâncias da mesma aplicação em busca de alguma informação ou erro gerado por algum comportamento inesperado. Porém, quando se muda o contexto para microsserviços, essa consulta se torna praticamente impossíel, se seguirmos os mesmos padrões de sistemas monolíticos. O primeiro motivo que torna isso inviável é a quantidade de pontos de geração de registros. Por exemplo uma aplicação que foi

reescrita com 10 microsserviços e cada microsserviço está rodando em duas instâncias, isso gera um total de 20 arquivos para se examinar. Outro ponto para ter em mente, é que microsserviços são desenhados para serem escalados horizontalmente a qualquer momento, isso pode siginificar aumentar ou diminuir o número de instências. Portanto há a possibilidade de que a instância que gerou aquele registro não esteja mais ativa e com isso toda a sua informação de registro de execução também.

De forma geral, registros de execução de uma aplicação, são sequências de eventos geradas em modo texto, onde cada linha representa alguma ação tomada em um determinado estado. Na arquitetura de microsserviçoes os registros devem ser gerados na saída padrão e não em arquivos, e capturados em tempo real (WIGGINS, 2019). Porém, a aplicação geradora não deve ser responsável por gravá-los em um local que possa ser consultado de forma estruturada posteriormente, pois caso essa instância seja desligada, perde-se o rastreio até a fonte. Para capturar essas informações existem algumas ferramentas que fazem com que seja possível capturar e indexar esses registros de uma maneira que seja possível consultá-las de forma agregada. *Graylog* é um dos poucos exemplos de agregadores de registros de execução que possuem livre acesso, porém essa ferramente possue todas as funcionalidades necessárias para gerenciar a consulta aos registros num ambiente de microsserviços.

Todos esses aspectos precisam estar alinhados por todas as partes componentes de um sistema baseado em microseriços e portanto, equipes de operação de desenvolvimento devem estar dispostas juntas desde o início, do desenho à implementação. A decisão de como será a estratégia deixa de ser uma questão de arquitetura, código ou tecnologia e sim, para a ser, uma coisa só, desenho, desenvolvimento e operação.

CONSIDERAÇÕES FINAIS

Conclui-se que ao decidir realizar a migração de sua arquitetura de sistemas para o padrão de microsserviços, qualquer instituição, está decidida a ter um diferencial competitivo, com maior qualidade e maior agilidade no lançamento de produtos e serviços, esses são os maiores benefícios que esse modelo de arquitetura visa. Esses benefícios são alcançados se todos os aspectos acima elucidados forem atendidos em sua completude, pois trabalhar com microsserviços, requer muita disciplina dos times responsáveis pela sua concepção e operação. Isso se dá, em parte, pelo alto número de processos e sistemas, executando de forma concomitante e tendo dependência entre si para que a funcionalidade ou serviço esteja completamente disponível e aderente aos seus requisitos funcionais.

Além disso, algumas mudanças no pensamento e cultura da organização serão necessárias para a obtenção de sucesso, já que um dos primeiros impactos que a adoção dos microsserviços trazem é justamente a mudança no organograma da empresa. E também, outras atividades apresentadas devem ser constantemente executadas para garantir a manutenibilidade do sistema. A execução de constantes avaliações e monitoramento do sistema, usando as métricas individuais para identificar possíveis pontos de lentidão devido ao tráfego ou mal funcionamento de algum dos componentes, além de reavaliar possíveis quebras de serviços em serviços menores concomitante com o constante treinamento das equipes de trabalho para garantir melhoria contínua;

Baseando-se nas técnicas e etapas apresentas e no fato de que todas as atividades que exigem algum tipo de ferramenta possuem alternativas baseadas em código aberto, a implantação da arquitetura de microsserviços requer baixo investimento em infraestrutura e também em treinamento, pois existe vasta documentação de como desenvolver e manter esse tipo de sistema, o que facilita a justificativa dessa migração frente aos benefícios que ela traz.

REFERÊNCIAS

BALALAIE, Armin; HEYDARNOORI, Abbas; JAMSHIDI, Pooyan. Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture. **IEEE Software**, v. 33, n. 3, p. 42-52, Mai-Jun. 2016.

BUCCHIARONE, Antonio et al. From Monolithic to Microservices: An Experience Report from the Banking Domain. **IEEE Software**, v. 35, n. 3, p. 50-55, Mai-Jun. 2018.

BUDGEN, David; BRERETON, Pearl. Performing systematic literature reviews in software engineering. In: 28TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING. EUA, N.Y. **Proceedings of the 28th international conference on software engineering (icse '06)**. EUA, N.Y: IEEE Xplore, 2016.

CHEN, Rui; LI Shanshan; LI, Zheng. From Monolith to Microservices: A Dataflow-Driven Approach. In: 24th Asia-Pacific Software Engineering Conference (APSEC). China, Nanjing, 2017. **Proceedings of the 24th Asia-Pacific Software Engineering Conference (APSEC)**. China, Nanjing, IEEE Xplore, 2017.

CITO, Jürgen; LEITNER, Philipp; FRITZ, Thomas; GALL Harald C. The making of cloud applications: an empirical study on software development for the cloud. In: 10TH JOINT MEETING ON FOUNDATIONS OF SOFTWARE ENGINEERING (ESEC/FSE 2015). ACM, New York, NY, USA, Ago. 2015. **Proceedings of 10th joint meeting on foundations of software engineering (esec/fse 2015)**. USA, N.Y. ACM, 2015

EBERT, Christof; GALLARDO, Gorka; HERNANTES, Josune; SERRANO, Nicolas. DevOps. **IEEE Software**, v. 33, n. 3, p. 94-100, Mai-Jun. 2016.

FOWLER, Martin. **Microservices: a definition of this new architectural term**. Disponível em: http://martinfowler.com/articles/microservices.html. Acesso em: 12/Dez/2018.

FURDA, Andrei; FIDGE, Colin; ZIMMERMANN, Olaf; KELLY, Wayne; BARROS, Alistar. Migrating Enterprise Legacy Source Code to Microservices: On Multitenancy, Statefulness, and Data Consistency. **IEEE Software**, v. 35, n. 3, p 63-72, Mai-Jun. 2018.

HOFFMAN, Kevin. **Beyond the Twelve-Factor App.** 1ª edição. São Francisco, Califórnia. O'Reilly Media, Inc.

IBGE. **Brasil em síntese. Pessoal ocupado, por segmento de serviço.** Disponível em: https://brasilemsintese.ibge.gov.br/servicos/pessoal-ocupado-por-segmento-de-servico.html. Acesso em: 09/Jan/2019.

KINCHENHAM, Barbara**. Procedures for performing systematic reviews**; Keel, Staffs, United Kingdom. Julho 2004. 27 p.

LARRUECA, Xabier; SANTAMRIA Izaskun; PALACIOS-COLOMO, Ricardo; EBERT, Christof. Microservices, **IEEE Software**, v. 35, n. 3, p. 96-100, Mai-Jun 2018.

LEWIS, James; FOWLER, Martin. **What are microservices?** Disponível em: https://martinfowler.com/microservices/#what. Acesso em 07/Jan/2019

MAZLAMI, Genc; CITO Jürgen; LEITNER, Philipp. Extraction of Microservices from Monolithic Software Architectures, **IEEE International Conference on Web Services (ICWS),** Honolulu, HI, 2017, p. 524-531.

PELLEGRINI, F. Rezende.; FOGLIATTO, Flavio. Estudo comparativo entre modelos de Winters e de Box-Jenkins para a previsão de demanda sazonal. **Revista Produto & Produção**, v. 4, número especial, p.72-85, Abr. 2000.

SILL, Alan. The Design and Architecture of Microservices, **IEEE Cloud Computing**, v. 3, n. 5, p. 76-80, Set-Out. 2016

SINGLETON, Andy. The Economics of Microservices. **IEEE Cloud Computing**, v. 3, número. 5, p. 16-20, Set-Out. 2016.

THONES, Johannes. Microservices. **IEEE Software**, v. 32, n. 1, p. 116–116, Jun. 2015.

WIGGINS, Adam. **The Twelve-Factor App - Logs**. Disponível em: https://12factor.net/logs. Acesso em 10/Jan/2019.

WIGGINS, Adam. **The Twelve-Factor App - V. Build, release, run**. Disponível em: https://12factor.net/build-release-run. Acesso em 10/Jan/2019.

WIGGINS, Adam. **The Twelve-Factor App**. Disponível em: https://12factor.net/. Acesso em 07/Jan/2019.