Migration of Monolithic Systems to Microservices: A Systematic Mapping Study

Migração de Sistemas Monolíticos Para Microsserviços: Um Estudo de Mapeamento Sistemático

Abstract. *The popularity of microservices architecture has been increasing considerably because of its capacity to alleviate monolithic architecture issues, namely: limited scalability in terms of modules, hard maintenance, and technological dependence, among others. Due to these issues, it is crucial to migrate to microservices architecture. The relevance of this work stems from the fact that there is an urgent need of methodologies and techniques to address migration towards microservices, such as decomposition from the monolithic, communication between services, DevOps implantation, etc. We revised research works recently published aiming to analyze migration cases of applications, identification of microservices techniques, factors that promote the migration, tools that are being used during the migration, problems and benefits of the migration. Results showed that this article contributed considerably to shed some light on migration processes, microservice identification techniques, factors that drive migration, tools that are used during the migration, migration problems and benefits.*

Resumo. *A popularidade da arquitetura de microsserviços tem aumentado consideravelmente devido à sua capacidade de solucionar problemas da arquitetura monolítica, por exemplo: escalabilidade limitada em termos de módulos, manutenção árdua e dependência tecnológica, entre outros. Devido a esses problemas, é crucial migrar para a arquitetura de microsserviços. A relevância deste trabalho decorre do fato de haver uma necessidade urgente de metodologias e técnicas para abordar a migração para microsserviços, como decomposição do monolítico, comunicação entre serviços, implantação de DevOps, etc. Revisamos trabalhos de pesquisa recentemente publicados com o objetivo de analisar casos de migração de aplicações, identificação de técnicas de microsserviços, fatores que promovem a migração, ferramentas que estão sendo usadas durante a migração, problemas e benefícios da migração. Os resultados mostraram que este artigo contribuiu consideravelmente para esclarecer alguns processos de migração, técnicas de identificação de microsserviços, fatores que impulsionam a migração, ferramentas usadas durante a migração, problemas e benefícios da migração.*

1. Introdução

A Arquitetura Monolítica fundamenta-se no encapsulamento integral das funcionalidades acerca de uma única aplicação (Baresi, Garriga e Renzis, 2017; Chen, Li e Li, 2017; Taibi, Lenarduzzi e Pahl, 2017) que é posta em execução por meio de um único processo (Koschel, Astrova e Dötterl, 2017). Essa tem sido a arquitetura de maior predominância no âmbito de desenvolvimento de software (Baresi, Garriga e Renzis, 2017). A ampla utilização da arquitetura monolítica não é à toa, uma vez que oferece uma solução que pode ser implementada de maneira simples (Chen, Li e Li, 2017;

Koschel, Astrova e Dötterl, 2017). Além disso, conforme Chen et al. (2017) e Koschel et al. (2017) é facilmente dimensionada empregando instâncias da aplicação em companhia de um balanceador de cargas e, a implantação também é descomplicada, visto que há somente um arquivo para ser implantado (Chen, Li e Li, 2017; Koschel, Astrova e Dötterl, 2017).

Os problemas da arquitetura monolítica surgem ao passo que a aplicação aumenta (Fritzisch, Bogner, Zimmermann e Wagner, 2018; Kosherl, Astrova e Dötterl, 2017), eles causam diversos impactos como: alta acoplagem de códigos, compreensão dificultosa, manutenção árdua, lentidão ao decorrer do processo de compilação, acréscimo de esforço de comunicação entre envolvidos no projeto, etc. (Chen, Li e Li, 2017). Além do mais, alguns empecilhos são herdados da própria natureza da arquitetura, por exemplo: desprovimento de escalabilidade a nível de módulo (Yugopuspito, Panduwinata e Sutrismo, 2017), dependência tecnológica (Fritzisch, Bogner, Zimmermann e Wagner, 2018) e implantação contínua dificultosa (Chen, Li e Li, 2017).

Desta forma, objetivando prover soluções para as deficiências da arquitetura monolítica, surgiu a arquitetura de microsserviços. Essa é uma arquitetura nativa da nuvem (Fritzisch, Bogner, Zimmermann e Wagner, 2018; Taibi, Lenarduzzi, Pahl e Janes,2017), por esse motivo é notório a presença da propriedade de escalabilidade e disponibilidade, além de outras propriedades que a nuvem proporciona (Balalaie, Heydarnoori e Jamshidi, 2015) nesta arquitetura. Apesar da constatação da escalabilidade em arquiteturas nativas da nuvem, para que de fato a aplicação torne-se dimensionável é necessário que a mesma esteja incorporada em um modelo escalável, por exemplo, a arquitetura de microsserviços que nasce com esta característica (Balalaie, Heydarnoori e Jamshidi, 2015).

A arquitetura de microsserviços é um aglomerado de boas práticas (Koschel, Astrova e Dötterl, 2017; Yugopuspito, Panduwinata e Sutrismo, 2017; Thönes, 2015) que abrange várias características, a título de exemplo pode-se citar: DevOps, heterogeneidade tecnológica, escalabilidade, serviços autônomos, conteinerização etc. Os principais componentes desta arquitetura são os microsserviços que, por sua vez, não possuem uma definição exata na literatura, porém diversos autores os definem como sendo pequenos serviços autônomos (Baresi, Garriga e Renzis, 2017) que são capazes de desempenhar uma tarefa, claramente definida, de modo eficiente (Taibu, Lenarduzzi, Pahl e Janes, 2017; Taibu, Lenarduzzi e Pahl, 2017; Tugopuspito, Panduwinata e Sutrismo, 2017) por via de processos individuais (Balalaie, Heydarnoori e Jamshidi, 2015; Koschel, Astrova e Dötterl, 2017).

As aplicações construídas de acordo com a arquitetura de microsserviços possuem diversos aspectos considerados como positivos, tendo como principais a escalabilidade a nível de módulo, manutenção e implantação facilitada (Chen, Li e Li, 2017). Diante das vantagens apresentadas e outras que serão abordadas ao decorrer deste estudo, diversas empresas como Netflix, SoundCloud e Amazon optaram por migrar da arquitetura monolítica para microsserviços (Chen, Li e Li, 2017).

Apesar disso, a migração não é classificada como um procedimento simples, pois envolve questões complexas que necessitam de estratégias para serem solucionadas (Balalaie, Heydarnoori e Jamshidi, 2015; Christoforou, Garriga, Andreou e Baresi, 2017). Dessa maneira, a migração é considerada um tópico de alta sensibilidade na

atualidade (Dragoni, Dustdar, Larsen e Mazzara, 2015). Assim, o presente estudo visa o agrupamento de diversas informações acerca da migração de aplicações monolíticas para microsserviços, baseando-se em estudos bibliográficos recentemente publicados. Assim sendo, os dados apresentados neste trabalho podem proporcionar uma ampla visão para cidadãos interessados no tema, além de servir como base para futuros estudos.

As informações para a realização deste estudo foram coletadas por meio de um mapeamento sistemático de literatura. Assim sendo, os dados extraídos dos materiais bibliográficos são relacionados a processos de migração, técnicas de identificação de microsserviços, fatores que impulsionam a migração, ferramentas que são utilizadas ao decorrer da migração, problemas e benefícios da migração.

O restante deste estudo está organizado da seguinte forma, na Seção 2, são expostos os trabalhos relacionados. Na Seção 3, é descrito o mapeamento sistemático de literatura. Na Seção 4, é retratado um exemplo de caso. Na Seção 5, são apresentados os resultados, seguido na seção 6 pelas limitações do trabalho. E por fim, na Seção 7, são apresentados a conclusão e trabalhos futuros.

2. Trabalhos Relacionados

Esta seção tem como proposta abordar os trabalhos de revisão sistemática acerca da migração de aplicações monolíticas para microsserviços.

Fritzshch et al. (2018) abordam a dissociação do monolítico que consiste em fragmentar a aplicação monolítica em pequenos serviços. Essa é uma tarefa significativa e de nível elevado em termos de complexidade. Além disso, pode-se considerar como sendo a tarefa mais importante para a migração. Assim, os autores realizam uma revisão sistemática a fim de coletar técnicas de identificação de microsserviços, disponíveis na literatura, que auxiliem no processo de dissociação do monolítico. Assim sendo, dez técnicas foram encontradas.

Em função disso, Fritzshch et al. (2018) propõe uma espécie de guia de decisões com finalidade de prestar assistência em ocasiões de migração, onde é possível optar pela técnica mais viável a partir de um dado cenário.

No estudo de Christoforou et al. (2010) foi abordado os fatores que impulsionam a realização da migração. Diante disso, uma revisão sistemática é efetuada visando coletar um conjunto de impulsionadores que posteriormente são refinados por especialistas da área. No final, os autores propõem um tipo de sistema de apoio a decisões, baseado em um gráfico de nós, visando auxiliar na migração para microsserviços.

Diferentemente dos trabalhos apresentados, o presente estudo não se limita apenas a expor técnicas de identificação de microsserviços e fatores que impulsionam a migração, este trabalho também tem como proposta apresentar com base nos materiais bibliográficos utilizados: metodologias de migração, ferramentas utilizadas na migração, problemas e benefícios da migração. Em função disso, os resultados expostos poderão ser utilizados para o fornecimento de uma ampla visão em relação à migração, contribuindo com a disseminação do conhecimento para indivíduos interessados na mesma, além de servir como base para posteriores estudos.

3. Mapeamento Sistemático de Literatura

O presente trabalho tem como finalidade apresentar uma ampla visão referente à migração de aplicações na arquitetura monolítica para microsserviços, a partir da análise de materiais bibliográficos, investigando metodologias de migração, técnicas para a identificação de microsserviços, impulsionadores da migração, ferramentas utilizadas na migração e problemas e benefícios da migração.

A estratégia adotada para a elaboração deste estudo foi o mapeamento sistemático de literatura que visa apresentar a frequência de publicações por categoria incluída em um contexto específico (Silva, 2018). Para auxiliar na execução do protocolo de pesquisa foi utilizada a ferramenta denominada StArt (Zmboni, Tommazo, Hernandes e Fabbri, 2010).

Portanto, o objetivo da pesquisa foi formalizado empregando porção do modelo GQM (Goal-Question-Metric) segundo, Basili e Weiss (1984): **Analisar** casos de migração de aplicações monolíticas para microsserviços, **com o propósito de** caracterizá-los **com respeito** ao microsserviço **do ponto de vista** de pesquisadores **no contexto de** pesquisas teóricas e aplicadas. Em prol da realização dos estudos foram estabelecidos quatro pesquisadores: Pe1, Pe2, Pe3 e Pe4. Os mesmos exerceram interação ao decorrer de todas as etapas.

Com finalidade de atender o protocolo foram determinados critérios circunstanciados pelo PICO (população, intervenção, comparação e resultados) (VSchugerl, Rilling, Witte e Charland (2009).

Como esse mapeamento visa a garantia da qualidade foram apenas utilizados os atributos de população, intervenção e resultados, de outra maneira, um extrato do PICO exibido pela Tabela 1.

Tabela 1. Critérios PICO

Critério PICO	Descrição
População	*Microservices, Monolith*
Intervenção	*Decomposition, Migration*
Comparação	Não se aplica
Resultados	*Process / Technique / Model / Method / Approach / Features / Characteristics / Factors / Tools*

Desta forma, foram elaboradas uma questão e quatro sub-questões de pesquisa como seguem:

Q1 - Como são realizadas as migrações de sistemas monolíticos para microsserviços?

Sub-Q1 - Quais técnicas têm sido utilizadas para identificar pontos passíveis de utilização de microsserviços?

Sub-Q2 - Quais fatores têm impulsionado a migração de aplicações monolíticas para microsserviços?

Sub-Q3 - Quais ferramentas têm sido utilizadas na migração de sistemas monolíticos para microsserviços?

Sub-Q4 - Quais são os problemas e benefícios da migração?

3.1. Seleção das Fontes

O estabelecimento das fontes a serem empregadas para a execução do presente estudo se deu por meio de uma série de critérios expostos na Tabela 2. As fontes selecionadas foram: ACM, IEEE, Scopus, Web of Science, Science Direct e arXiv. No total, seis fontes foram eleitas.

Tabela 2. Critérios para seleção das fonts bibliográficas

ID	Descrição
CSF01	Abrangência de suas máquinas de busca
CSF02	Disponibilidade de acesso ao conteúdo
CSF03	Por serem bases bibliográficas
CSF04	Por serem motores de busca

3.2. Estratégia de Busca

A estratégia de busca visa localizar cláusulas que atendam as questões principais sugeridas na pesquisa. Primeiramente foi executada a busca de maneira automatizada nas fontes já estabelecidas.

Para a elaboração deste estudo nenhuma unidade de tempo mínima foi determinada em relação aos materiais bibliográficos a serem analisados. Logo, quando a *string* de busca foi posta em execução nas bases de dados bibliográficas, citadas no parágrafo anterior, as mesmas responderam com resultados transparentes aos pesquisadores.

A estratégia de busca que é dividida nas etapas a seguir:

(i). Declaração dos critérios, extração das cláusulas de busca e realização das buscas. Sendo selecionadas um total de 38 artigos.
(ii). Remoção de artigos duplicados. Restando 24 artigos.
(iii). Execução da primeira análise qualitativa dos resultados tendo como base os critérios de inclusão e exclusão. Tendo como resultado um total de 11 artigos.
(iv). Realização da segunda análise qualitativa com base na análise dos resultados dos artigos. Mantendo os 11 artigos
(v). Extração dos dados para a resolução das questões de pesquisa propostas, dos 11 apenas 9 artigos respondia às questões.

3.3. Termos de Pesquisa

Os termos de pesquisa foram reunidos de acordo com as seguintes palavras-chaves: Microservices, Microservice, Microservice architecture, monolith, monoliths, decomposition, migration, transform e move. Essas foram utilizadas em uma *string* de busca, cujo é um conjunto de palavras-chaves combinadas com operadores booleanos, para a realização da pesquisa.

A *string* de busca foi utilizada para a realização desta pesquisa: (((“Microservices” OR “Microservice” OR “Microservice architecture”) AND (“monolith” OR “monoliths”)) AND (“decomposition” OR “migration” OR “transform” OR “move”)).

O motivo pelo qual os termos foram definidos no idioma inglês se dá pelo fato de grande parte das revistas e conferências o adotarem no tema abordado.

Dois testes foram aplicados sobre essa *string*, a fim de refiná-la visando o retorno de melhores resultados.

3.4. Critérios de Seleção do Estudo

Para a seleção dos materiais bibliográficos, resultantes da busca, foi definido uma série de critérios de inclusão e exclusão. É importante ressaltar que não houve restrição de tempo de busca. Assim sendo, a Tabela 3 apresenta os critérios de inclusão e exclusão adotados nesta revisão.

Tabela 3. Critérios de seleção de estudos

Critérios de inclusão	
ID	**Critério**
IC01	O artigo analisa migração de sistemas monolíticos para microsserviços.
IC02	O artigo apresenta técnica para identificação de microsserviços.
IC03	O artigo analisa um ou mais dos seguintes fatores: benefícios, problemas e impulsionadores da migração.
Critérios de exclusão	
ID	**Criteria**
EC01	Artigos duplicados.
EC02	Artigos que não estão escritos em inglês.
EC03	Artigos publicados apenas como resumos ou prefácio de periódicos e eventos.
EC04	Falta de disponibilidade do artigo para download.
EC05	Teses, blogs, sites, dissertações e monografias de conclusão de curso.

3.5. Procedimento de Seleção de Estudos

O procedimento de seleção de estudos foi realizado por quatro pesquisadores, denominados Pe1, Pe2, Pe3 e Pe4. Durante esse procedimento títulos e resumos dos artigos foram analisados com o intuito de verificar se são convenientes para este estudo.

A partir da análise realizada foi possível classificar os artigos, com o auxílio da ferramenta StArt[1], em aceito, rejeitado ou duplicado. Contemplamos 11 (29%) aceitos, 13 (34%) rejeitados e 14 (37%) duplicados.

Posteriormente, foram feitas as leituras dos artigos aceitos e partir disso foi possível encontrar respostas para os questionamentos feitos no protocolo da pesquisa, é importante salientar que todos os artigos aceitos estão de acordo com os critérios de inclusão e contribuíram para solucionar as questões.

Assim, o ciclo utilizado para este estudo, pode ser visto na figura 1.

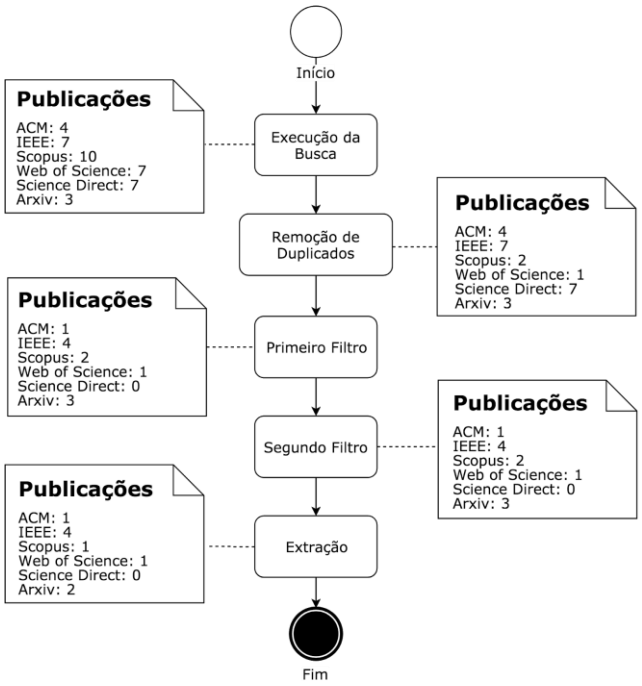

Figura 1. Etapas do processo de busca e seus respectivos resultados – Estratégia de Busca

4. Exemplo de Caso

Buscando analisar os achados, propomos como exemplo um SGE (Sistema de Gerenciamento de Eventos) que conta com diversas funcionalidades que visam facilitar o gerenciamento de eventos. Dessa maneira, o sistema abrange operações que estão relacionadas a colaboradores, atividades, equipamentos, eventos, convidados, custos, atrações, equipes, dentre outras.

A figura 2, expõe o SGE na arquitetura monolítica na qual encapsula totalmente as suas funcionalidades em uma única aplicação.

---

[1] http://lapes.dc.ufscar.br/tools/start_tool

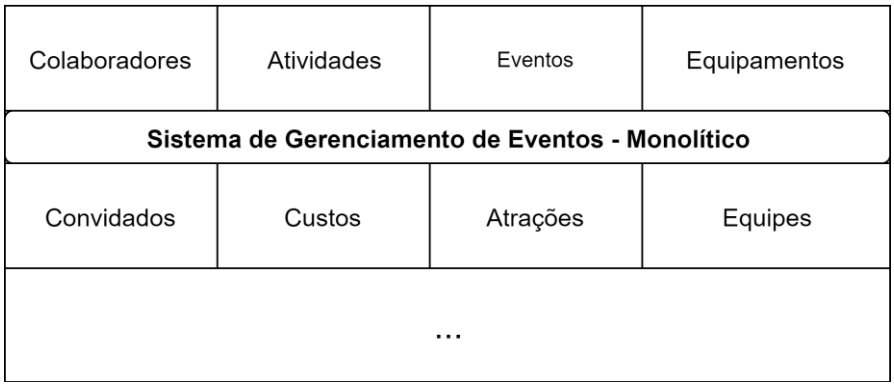

Figura 2. SGE na arquitetura monolítica

A figura 3, retrata o SGE na arquitetura de microsserviços onde as operações são distribuídas em serviços que dispõem de dimensão reduzida, contexto limitado e independência.

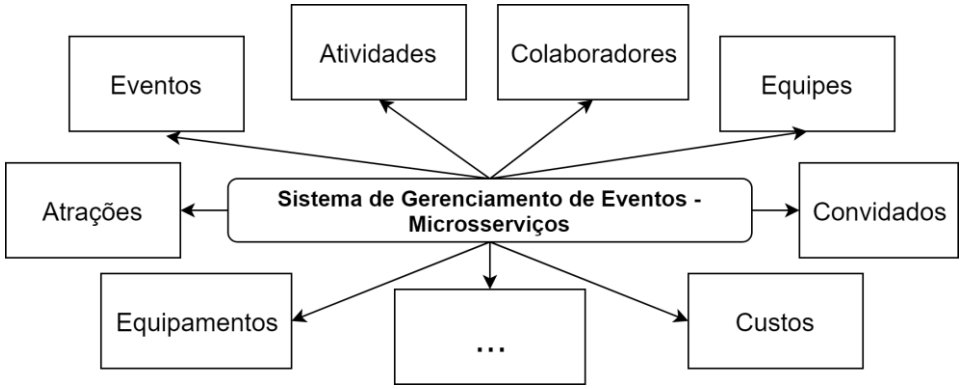

Figura 3. EMS on microservice architecture

O exemplo de caso apresentado será posteriormente mencionado nos resultados com o propósito de explanar diversos aspectos.

5. Resultados

Os artigos eleitos para a realização deste estudo são apresentados na Tabela 4. Foram apurados nove artigos dos quais sete foram publicados em evento, somente um em pré-print e apenas um em periódico. Esses são considerados recentes, sendo o mais antigo publicado no ano de 2016.

Tabela 4. Artigos eleitos após o mapeamento

ID	Artigo	Autores	Ano de publicação	Local de publicação
1	Migrating to cloud-native architectures using microservices: an experience report	Balalaie, Armin and Heydarnoori, Abbas and Jamshidi, Pooyan	2016	European Conference on Service-Oriented and Cloud Computing

2	Microservices identification through interface analysis	Baresi, L. and Garriga, M. and De Renzis, A.	2017	European Conference on Service-Oriented and Cloud Computing
3	From Monolith to Microservices: A Dataflow-Driven Approach	R Chen and S. Li and Z. Li	2017	Asia-Pacific Software Engineering Conference (APSEC), 2017 24th
4	Microservices: Migration of a mission critical system	Dragoni, Nicola and Dustdar, Schahram and Larsen, Stephan T. and Mazzara, Manuel	2017	arXiv
5	From monolith to Microservices: Lessons learned on an industrial migration to a web oriented architecture	J. Gouigoux and D. Tamzalit	2017	Software Architecture Workshops (ICSAW), 2017 IEEE International Conference on
6	Making the move to microservice architecture	A. Koschel and I. Astrova and J. Dötterl	2017	Information Society (i-Society), 2017 International Conference on
7	Microservices in agile software development: A workshop-based study into issues, advantages, and disadvantages	Taibi, Davide and Lenarduzzi, Valentina and Pahl, Claus and Janes, Andrea	2017	XP '17 Proceedings of the XP2017 Scientific Workshops
8	Processes, motivations, and issues for migrating to microservices architectures: an empirical investigation	Taibi, Davide and Lenarduzzi, Valentina and Pahl, Claus	2017	IEEE Cloud Computing
9	Microservices architecture: Case on the migration of reservation-based parking system	P. Yugopuspito and F. Panduwinata and S. Sutrisno	2017	Communication Technology (ICCT), 2017 IEEE 17th International Conference on

Os resultados destinam-se a responder uma questão principal e quatro sub questões que são:

Q1 - Como são realizadas as migrações de sistemas monolíticos para microsserviços?

RQ1 - Os artigos de Gouigoux e Tamzalit (2017) e Yugopuspito, Panduwinata e Sutrismo (2017), relatores de caso real de migração, não descrevem claramente um processo de migração a ser seguido. Já Balalaie, Heydarnoori e Jamshidi (2015) e Dragoni, Dustdar, Larsen e Mazzara (2015) também relatores de caso real de migração, ambos propõem um processo de migração aplicado a partir do seu próprio cenário, não sendo classificados esses como processos generalizados.

O artigo de Taibi, Lenarduzzi e Pahl (2017) propõe três processos de migração generalizados que foram expostos a partir de entrevistas realizadas com diversos profissionais da área. Os processos foram denominados P1, P2, e P3. Os dois primeiros processos preconizam o fim do sistema monolítico e o início de um novo sistema baseado em microsserviços. Diferentemente o P3 sugere que os microsserviços sejam paulatinamente implantados a partir do surgimento da necessidade de novos recursos, ou alterações importantes em recursos já existentes, substituindo tais recursos por microsserviços. Ambos os processos possuem cinco etapas nomeadas de maneira equivalente, de forma ordenada são: análise da estrutura do sistema, arquitetura do sistema, priorização do desenvolvimento de recursos, codificação e testes. Apesar de possuírem nomes semelhantes, as etapas possuem peculiaridades a depender do processo. A descrição das etapas dos processos citados é apresentada no fluxograma da Figura 4.

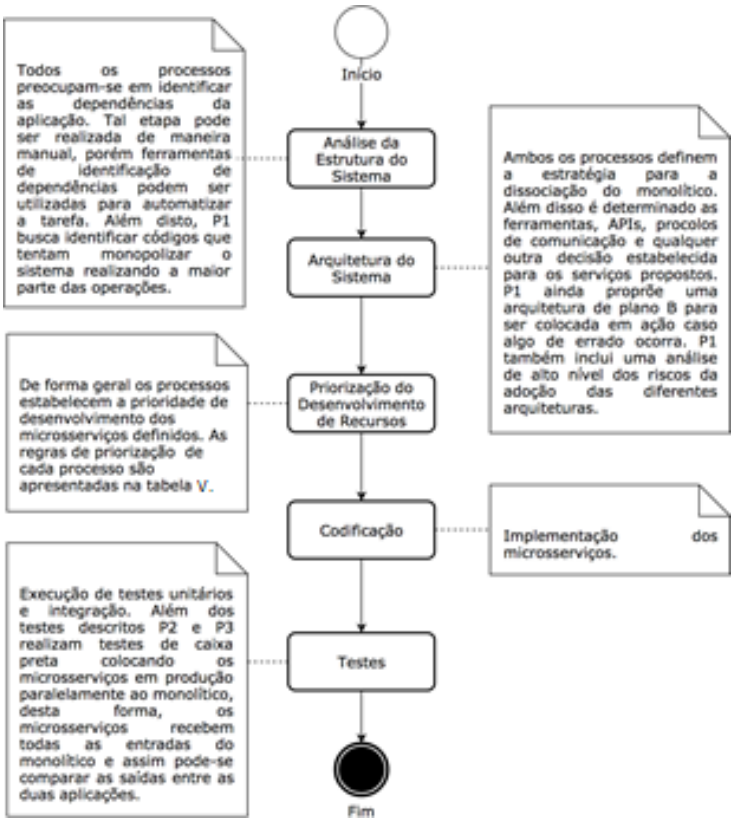

Figura 4. Etapas e descrições dos processos de migração citados

A Tabela 5 retrata o critério de cada processo para determinar a prioridade de desenvolvimento dos microsserviços propostos.

Tabela 5. Critério de definição de prioridade de cada processo

Processo	Critério de Prioridade
P1	A prioridade é julgada a partir da importância que a funcionalidade tem para o cliente, quanto maior a importância, maior a prioridade de desenvolvimento.
P2	A prioridade é baseada em serviços existentes, que contém uma maior quantidade de *bugs* ou que precisam de mais trabalho para tentar reduzir imediatamente as necessidades do cliente, sempre considerando serviços com menos dependências. Caso a prioridade do cliente seja equivalente em ambos serviços, o serviço que possui maior quantidade de dependências é mais priorizado.
P3	Apenas novos serviços são desenvolvidos como microsserviços, sendo colocados ao redor do monolítico. A longo prazo, os microsserviços irão gradualmente eliminando cada característica do monolítico.

Sub-Q1 – Quais técnicas têm sido utilizadas para identificar pontos passíveis de utilização de microsserviços?

RSub-Q1 – Duas técnicas foram constatadas, análise de interfaces e diagrama de fluxo de dados, conforme a Tabela 6 (Baresi, Garriga e Renzis, 2017; Chen, Li e Li, 2017).

Tabela 6. Técnicas de identificação de microsserviços

Técnica	Entrada	Saída
Análise de interfaces	Vocabulário de referência e especificação OpenAPI da aplicação	Microsserviços candidatos
Diagrama de fluxo de dados	Diagrama de fluxo de dados purificado	Microsserviços candidatos

Análise de Interfaces

A técnica de análise de interfaces (Baresi, Garriga e Renzis, 2017) realiza a identificação por meio de um cálculo de similaridade semântica baseado em co-ocorrências. O cálculo utiliza uma função de adequação baseada em DISCO (DIStributionally related words using CO-occurrences) sendo executada de forma automatizada.

Essa é aplicada por meio da execução de um algoritmo de decomposição, que recebe como entrada os apetrechos informados na Tabela VI. Em seguida, para cada operação descrita na especificação, ocorre uma associação a um conceito de referência proveniente do vocabulário, este procedimento é realizado através de um algoritmo de avaliação semântica que é chamado internamente no algoritmo de decomposição. Após o mapeamento operação/conceito, as operações que compartilham do mesmo conceito são agrupadas formando um microsserviço candidato.

A técnica proposta proporciona microsserviços com granularidade fina. Algumas limitações afetam a técnica, tais como: o mapeamento de operações para

poucos conceitos de referência, resulta em microsserviços de granularidade mais grossa; A técnica é baseada em interfaces bem relatadas que disponibilizam nomes expressivos e que seguem as convenções de nomenclatura de programação, porém nem sempre é possível ter controle sobre isso e alguns cenários são complicados de lidar, por exemplo, identificadores como *param*, *response* e *request*.

Diagrama de Fluxo de Dados

A identificação de microsserviços por meio de uma abordagem orientada a diagrama de fluxo de dados (Chen, Li e Li, 2017) é realizada em um processo de três etapas. É relatada como uma técnica semi-automática onde a primeira etapa é executada de forma manual, enquanto as demais etapas são executadas automaticamente por meio de um algoritmo.

Inicialmente é produzido de forma manual um diagrama de fluxo de dados purificado (DFD) de acordo com uma lógica de negócio. Em seguida, o algoritmo proposto realiza a condensação do DFD purificado em um DFD decomponível utilizando uma função denominada GetDecomposableDFD (PDF). Por fim, realizando a chamada da função GetMicroservice (DDF), é retornado o conjunto de microsserviços candidatos.

A técnica possui superioridade ao *Service Cutter* (Gysel, Kölberner, Giersche e Zimmermann, 2016) em aspectos como: racionalidade, objetividade, facilidade de operar e facilidade de entender. Entretanto, diversas limitações foram relevadas, tais como: a combinação de operações é realizada através dos nomes das mesmas, portanto deve-se nomear as operações de maneira adequada; É possível que os microsserviços resultantes, refinados em termos de operação de dados baseando-se em uma lógica de negócio, sejam avaliados por especialistas antes de serem desenvolvidos, pois pode haver a necessidade de integrá-los para formar um microsserviço de granularidade mais grossa em tempo de *design*. Tendo em vista que a técnica é direcionada para decomposição em tempo de projeto, ainda há uma falta de avaliação dos microsserviços candidatos e seus recursos de tempo de execução como, requisitos não funcionais e escalabilidade.

Sub-Q2 – Quais fatores têm impulsionado a migração de aplicações monolíticas para microsserviços?

RSub-Q2 – Quinze fatores foram identificados, porém quatro se destacam por serem citados em mais de um trabalho, são eles:

Escalabilidade

Os microsserviços dispõem de independência. Portanto, o dimensionamento dos mesmos é facilmente realizado. Tendo como exemplo o SGE, figura 3, baseado em microsserviços, supondo que o microsserviço relacionado a custos esteja recebendo variadas requisições e consequentemente necessitando de recursos computacionais, pode-se escalar o mesmo de forma individual por meio de um gerenciador de *containers*.

No caso do monolítico, figura 2, as operações relacionadas aos custos não podem ser dimensionadas de forma individual, já que o dimensionamento a nível de

módulo no monolítico não é possível, ou seja, toda a aplicação deverá ser dimensionada (Balalaie, Heydarnoori e Jamshidi, 2015; Dragoni, Dustdar, Larsen e Mazzara, 2015; Taibi, Lenearduzzi e Pahl, 2017).

Manutenção

A independência, dimensão reduzida e o contexto limitado são características pertencentes aos microsserviços que propicia a facilidade de manutenção.

Diante disso, pode-se inferir que uma modificação é descomplicada, ou seja, é possível dirigir-se diretamente ao microsserviço que receberá a alteração e aplicá-la sem maiores preocupações com o impacto que será causado nos demais microsserviços que compõem a aplicação, já que há um baixo grau de acoplamento entre os mesmos.

De maneira diferente, tendo como exemplo, o SGE monolítico que é composto por diversas operações e possui um vasto tamanho, aplicar alterações sobre o mesmo não será nada fácil, já que há um maior acoplamento com a possibilidade de maiores impactos (Dragoni, Dustdar, Larsen e Mazzara, 2015; Taibi, Lenearduzzi e Pahl, 2017).

Independência Tecnológica

A aplicação monolítica é construída empregando tecnologias que não poderão ser alteradas durante ou após o seu desenvolvimento, por exemplo: o SGE monolítico não poderá experimentar outra linguagem de programação para a resolução de problemas a não ser a própria em que a aplicação foi desenvolvida.

De maneira oposta, como os microsserviços são independentes, os mesmos podem ser desenvolvidos com diferentes tecnologias, por exemplo: o SGE baseado na arquitetura de microsserviços possui o microsserviço de convidados (Java), atrações (C#), eventos (Python), etc.

Logo, os microsserviços não inferem em dependência tecnológica, permitindo assim a experimentação de novas tecnologias para a resolução de problemas (Dragoni, Dustdar, Larsen e Mazzara, 2015; Taibi, Lenearduzzi e Pahl, 2017).

Implantação Facilitada

A facilidade de implantação se dá por meio da independência, por exemplo: caso haja uma alteração no microsserviço de equipes do SGE, surgindo assim a necessidade de implantação, o mesmo pode ser implantado de forma individual sem a necessidade de implantar todos os outros microsserviços novamente. Já o SGE monolítico exigiria a reimplantação de toda aplicação (Balalaie, Heydarnoori e Jamshidi, 2015; Dragoni, Dustdar, Larsen e Mazzara, 2015).

Além desses fatores há também: delegação de responsabilidades (Taibi, Lenarduzzi e Pahl, 2017), suporte DevOps (Taibi, Lenarduzzi e Pahl, 2017), influência (Taibi, Lenarduzzi e Pahl, 2017), tolerância a falhas (Taibi, Lenarduzzi e Pahl, 2017), delegação de responsabilidades de software (Taibi, Lenarduzzi e Pahl, 2017), interface web altamente ergonômica, interoperabilidade de baixo custo, garantia de evolução a longo prazo (Gouigoux e Tamzalit, 2017), baixo acoplamento (Dragoni, Dustdar, Larsen e Mazzara, 2015), governança de dados descentralizada (Balalaie, Heydarnoori e Jamshidi, 2015) e reutilização (Balalaie, Heydarnoori e Jamshidi, 2015).

Sub-Q3 – Quais ferramentas têm sido utilizadas na migração de sistemas monolíticos para microsserviços?

RSub-Q3 – Um total de 16 ferramentas foram identificadas, sendo o Docker e o Kubernetes as mais abordadas. Por exemplo, o Docker é um gerenciador de *containers* cuja função é proporcionar mecanismos para o processo de gerenciamento de *containers*. Outro exemplo é o Kubernetes, o qual é definido como um orquestrador e sua função é automatizar a implantação, gerenciar a escalabilidade e administrar aplicativos em *containers*.

Com o objetivo de dividir as ferramentas em categorias, dez categorias foram elaboradas, tais como: gerenciador de *containers* (C01), orquestração (C02), identificação de dependências (C03), sistema de gerenciamento de banco de dados (C04), sistema de mensageria (C05), monitoramento (C06), servidor de integração contínua (C07), repositório de código (C08), repositório de artefatos (C09) e banco de dados (C10).

As categorias são descritas como segue. C01 proporciona mecanismos para o processo de gerenciamento de containers. C02 concede infraestrutura automatizada. C03 realiza análise do projeto identificando as possíveis dependências contidas no mesmo. C04 estabelece que as dependências podem ser tanto a nível de código quanto a nível de banco de dados. C05 proporciona o gerenciamento do banco de dados, geralmente disponibilizando uma interface gráfica para que o usuário realize a manipulação dos dados, por exemplo: inserindo, atualizando ou consultando. C06 possibilita a comunicação entre aplicações através de mensagens. Aliás, monitora os contêineres e informa quando algum comportamento não esperado ocorre, e realiza análise de logs além de retratar os estados dos *containers*, por exemplo: memória, CPU, etc. C07 fornece suporte para o funcionamento da entrega contínua. C08 permite o armazenamento de código-fonte. C09 proporciona o armazenamento de artefatos. C10 possibilita o armazenamento de dados.

Dessa forma cada ferramenta foi associada a uma categoria para saber qual é sua função e aplicação no desenvolvimento de software baseado em microsserviços. A Tabela 7 expõe as ferramentas identificadas com a categoria associada.

Tabela 7. Ferramentas utilizadas na migração

ID do Artigo	Categoria	Ferramenta
1 ,4, 5, 9	C01	Docker
1, 4	C02	Kubernetes
4	C02	Marathon
8	C03	SchemaSpy e Structure 101
4	C04	Redis, PostgreSQL, Cassandra
4	C05	RabbitMQ
4	C06	Icinga
4	C06	cAdvisor

4	C06	Elasticsearch
1	C07	Jenkins
4	C07	GoCD
1	C08	Gitlab
1	C09	Artifactory
2	C10	DISCO

Assim, no caso do exemplo SGE da figura 3, a utilização do Docker, neste modelo, disponibilizaria à infraestrutura de *containers* para que cada um dos serviços funcione de forma individualizada (Balalaie, Heydarnoori e Jamshidi, 2015; Dragoni, Dustdar, Larsen e Mazzara, 2015; Gouigoux e Tamzalit, 2017; Yugopuspito, Panduwinata e Sutrismo, 2017). Portanto, o Kubernetes permitirá, então, automatizar o gerenciamento desta infraestrutura (Balalaie, Heydarnoori e Jamshidi, 2015; Dragoni, Dustdar, Larsen e Mazzara, 2015), proporcionando assim, escalabilidade automática para o serviço. Por exemplo, caso o serviço passe a ter uma demanda por recursos, o Kubernetes se responsabiliza por ampliar o número de *containers* ou recursos para o serviço; entretanto, caso diminua a demanda estes recursos adicionais são automaticamente desativados (C02). Diferenciando do sistema monolítico, figura 2, que a escalabilidade demanda hardware, virtualização e clusterização de recursos, tornando o processo mais complexo e com maior custo, já que envolve o sistema como um todo e não apenas um módulo.

Além disso, no modelo de microsserviços, figura 3, cada módulo, por ser menor, permite a adoção de ferramentas que permite a análise de dependências durante a implementação de forma simplificada (Taibi, Lenarduzzi e Pahl, 2017), integrado a repositórios de artefatos (Balalaie, Heydarnoori e Jamshidi, 2015) que permite disponibilizar recursos ao microsserviço, lembrando que as dependências entre os módulos são reduzidas, se comparado com o modelo monolítico (Taibi, Lenarduzzi e Pahl, 2017), figura 2. Outro ponto importante é que nos microsserviços, a comunicação se torna mais eficiente por meio dos serviços de mensageria (Dragoni, Dustdar, Larsen e Mazzara, 2015), melhorando a comunicação entre os módulos. Por fim, os microsserviços integrados aos *containers* se beneficiam do modelo de implantação ágil, necessitando de serviços de integração contínua integrado a repositórios de códigos (Balalaie, Heydarnoori e Jamshidi, 2015; Dragoni, Dustdar, Larsen e Mazzara, 2015).

Sub-Q4 – Quais são os problemas e benefícios da migração?

RSub-Q4 – No total foram identificados 19 problemas, no entanto 3 deles se destacam por serem abordados em mais de 2 trabalhos, são eles:

Dissociação do Monolítico

A dissociação do monolítico é a segmentação da aplicação em serviços que dispõem de dimensão reduzida, contexto limitado e independência. Essa atividade é crucial para a realização da migração, é relevante compreender as operações do sistema para que desta maneira seja possível agrupá-las de modo que forme microsserviços coesos, a título de exemplo pode-se citar a figura 3 que diz respeito ao SGE baseado em

microsserviços que retrata claramente a segmentação que resulta em diversos microsserviços.

A segmentação inadequada pode provocar diversos problemas, como: acoplamento elevado, dimensionamento impossível no nível de módulo, etc.

Sendo assim, é perceptível que a dissociação do monolítico é uma tarefa que deve possuir grande atenção no processo de migração. (Baresi, Garriga e Renzis, 2017; Chen, Li e Li, 2017; Fritzish, Bogner, Zimmermann e Wagner, 2018; Koschel, Astrova e Dötterl, 2017; Taibu, Lenarduzzi, Pahl e Janes, 2017; Taibi, Lenarduzzi e Pahl, 2017).

Comunicação

Em uma aplicação monolítica a comunicação é exercida por meio da invocação de métodos, visto que as operações estão totalmente encapsuladas em uma única aplicação.

Logo, o SGE monolítico, figura 2, pode invocar operações relacionadas aos convidados durante a execução de tarefas relacionadas ao evento de forma simples.

Ao contrário do monolítico, a arquitetura de microsserviços apossa-se de diversos serviços que abrangem variadas funcionalidades conforme o seu contexto, como por exemplo: o SGE, figura 3, contém o microsserviço de convidados que abrange operações relacionadas apenas a convidados.

À vista disso, a comunicação na arquitetura de microsserviços não pode ser realizada por meio da invocação de métodos, surgindo a necessidade de traçar meios que tornem a comunicação entre os serviços possível, como: API Restful, etc. (Gouigoux e Tamzalit, 2017; Koschel, Astrova e Dötterl, 2017; Taibi, Lenarduzzi e Pahl, 2017; Yugopuspito, Panduwinata e Sutrismo, 2017).

DevOps

A existência da cultura DevOps é extremamente significativa tratando-se de aplicações fundamentadas na arquitetura de microsserviços, porém não é algo simples e prático de se fazer. (Gouigoux e Tamzalit, 2017; Koschel, Astrova e Dötterl, 2017; Taibu, Lenarduzzi, Pahl e Janes, 2017; Taibi, Lenarduzzi e Pahl, 2017).

Além desses problemas há também: estimativa de esforço (Taibu, Lenarduzzi, Pahl e Janes, 2017; Taibi, Lenarduzzi e Pahl, 2017), monitoramento (Koschel, Astrova e Dötterl, 2017; Taibu, Lenarduzzi, Pahl e Janes, 2017), granularidade (Baresi, Garriga e Renzis, 2017; Gouigoux e Tamzalit, 2017), desenvolvedores experientes (Balalaie, Heydarnoori e Jamshidi, 2015; Taibi, Lenearduzzi e Pahl, 2017), versionamento (Balalaie, Heydarnoori e Jamshidi, 2015; Taibu, Lenarduzzi, Pahl e Janes, 2017), heterogeneidade tecnológica (Balalaie, Heydarnoori e Jamshidi, 2015; Koschel, Astrova e Dötterl, 2017), localização de serviços (Yugopuspito, Panduwinata e Sutrismo, 2017), migração de banco de dados e divisão de dados (Taibi, Lenarduzzi e Pahl, 2017), conversão de biblioteca (Taibi, Lenarduzzi e Pahl, 2017), mente das pessoas (Taibi, Lenarduzzi e Pahl, 2017), tempo de retorno sobre o investimento (Taibi, Lenarduzzi e Pahl, 2017), gestão estatal (Taibu, Lenarduzzi, Pahl e Janes, 2017), escalabilidade (Koschel, Astrova e Dötterl, 2017), interação cliente-servidor (Koschel, Astrova e Dötterl, 2017), coesão (Baresi, Garriga e Renzis, 2017) e implantação no ambiente de desenvolvimento (Balalaie, Heydarnoori e Jamshidi, 2015).

De forma semelhante 19 benefícios foram identificados, contudo apenas 3 destacaram-se por serem citados em mais de 2 trabalhos, são eles: escalabilidade (Baresi, Garriga e Renzis, 2017; Chen, Li e Li, 2017; Fritzish, Bogner, Zimmermann e Wagner, 2018; Koschel, Astrova e Dötterl, 2017; Taibu, Lenarduzzi, Pahl e Janes, 2017; Taibu, Lenarduzzi e Pahl, 2017), implantação facilitada (Gouigoux e Tamzalit, 2017; Koschel, Astrova e Dötterl, 2017; Taibi, Lenearduzzi e Pahl, 2017; Yugopuspito, Panduwinata e Sutrismo, 2017) e manutenção (Gouigoux e Tamzalit, 2017; Koschel, Astrova e Dötterl, 2017; Taibu, Lenarduzzi, Pahl e Janes, 2017; Taibi, Lenearduzzi e Pahl, 2017). Os benefícios destaques são apresentados e descritos na RSub-Q2.

Além dos benefícios destaques há também: independência tecnológica (Taibu, Lenarduzzi, Pahl e Janes, 2017; Taibi, Lenearduzzi e Pahl, 2017), contexto limitado (Koschel, Astrova e Dötterl, 2017; Taibu, Lenarduzzi, Pahl e Janes, 2017), ciclo de lançamento curto (Baresi, Garriga e Renzis, 2017; Gouigoux e Tamzalit, 2017), aumento de desempenho (Balalaie, Heydarnoori e Jamshidi, 2015; Taibi, Lenearduzzi e Pahl, 2017), capacidade de substituição (Balalaie, Heydarnoori e Jamshidi, 2015; Taibu, Lenarduzzi, Pahl e Janes, 2017), reusabilidade (Balalaie, Heydarnoori e Jamshidi, 2015; Koschel, Astrova e Dötterl, 2017), autonomia (Yugopuspito, Panduwinata e Sutrismo, 2017), fácil de compreender (Taibi, Lenarduzzi e Pahl, 2017), tempo de retorno sobre o investimento, testabilidade (Taibi, Lenearduzzi e Pahl, 2017), inteligibilidade (Taibu, Lenarduzzi, Pahl e Janes, 2017), redução de bugs (Koschel, Astrova e Dötterl, 2017), baixo acoplamento (Koschel, Astrova e Dötterl, 2017), governança descentralizada (Baresi, Garriga e Renzis, 2017) e disponibilidade (Balalaie, Heydarnoori e Jamshidi, 2015).

Os materiais bibliográficos selecionados para a realização deste estudo são recentes, sendo o mais antigo publicado em 2016. Tendo em vista os trabalhos selecionados para a elaboração deste estudo, nota-se uma falta de trabalhos publicados por meio de periódico, dos quais somente um possui esse como veículo de publicação.

É importante mencionar a ausência de um processo de migração generalizado funcionalmente comprovado. Visto que os processos apresentados por (Taibi, Lenearduzzi e Pahl, 2017) não foram submetidos a experimentos.

A quantidade de técnicas, expostas neste estudo, que auxiliam na dissociação do monolítico é consideravelmente mínima se comparado ao estudo de Fritzish, Bogner, Zimmermann e Wagner (2018) que aborda dez técnicas, disponíveis na literatura, inclusive a técnica apresentada em (Baresi, Garriga e Renzis, 2017). Entretanto a abordagem orientada a diagrama de fluxo de dados (Chen, Li e Li, 2017) não foi mencionada, dado isso, pode-se inferir que houve uma contribuição deste estudo para com a apresentação de técnicas utilizadas para dissociar o monolítico.

7. Conclusão e Trabalhos Futuros

Neste mapeamento sistemático, as resoluções das questões problemáticas foram concebidas por meio da análise de estudos referentes ao atual cenário da migração de sistemas monolíticos para microsserviços. Diante disso, é importante ressaltar que as resoluções apresentadas são baseadas em dados atuais.

Além disso, o presente estudo discorreu a respeito da migração de maneira clara, tratando processos de migração, técnicas de identificação de microsserviços, fatores que impulsionam a migração, ferramentas que são utilizadas na migração, problemas e

benefícios da migração. Portanto, uma ampla visão em relação ao procedimento de migração foi concedida.

A formação de categorias relativas às ferramentas identificadas é de grande relevância em termos de organização, além de propor uma melhoria ofertando um vocabulário comum para ser utilizado no ambiente de desenvolvimento entre os participantes da migração.

Em relação aos problemas, benefícios e impulsionadores da migração, pode-se dizer que este artigo contribuiu fortemente, proporcionando uma ampla visão sobre o estado da arte dos mesmos.

Assim, atendendo o objetivo de agrupar as informações sobre migrações de monoliticos para microsserviços, destacamos que definir um processo e as prioridades deve preceder a própria migração (Gouigoux e Tamzalit, 2017; Yugopuspito, Panduwinata e Sutrismo, 2017; Taibu, Lenarduzzi e Pahl, 2017). Além disso, duas técnicas que permite analisar sistemas monoliticos com o objetivo de identificar microsserviços foram encontradas, a análise de interfaces (Baresi, Garriga e Renzis, 2017) e o diagrama de fluxo de dados (Chen, Li e Li, 2017), requerendo estudos experiêntais sobre as técnicas, para comprovar sua eficiência. Tendo definido um processo, suas prioridades e a técnica para a migração, este trabalho também identificou os fatores que impusionam a migração de sistemas monoliticos para microsserviços, apesar de encontrarmos 15 fatores, quatro se destacam por ter sido apresentado em mais de um estudo, assim escalabilidade (Balalaie, Heydarnoori e Jamshidi, 2015; Dragoni, Dustdar, Larsen e Mazzara, 2015; Taibi, Lenearduzzi e Pahl, 2017), manutenibilidade (Dragoni, Dustdar, Larsen e Mazzara, 2015; Taibi, Lenearduzzi e Pahl, 2017), independencia tecnológica (Dragoni, Dustdar, Larsen e Mazzara, 2015; Taibi, Lenearduzzi e Pahl, 2017) e implantação simplificada (Balalaie, Heydarnoori e Jamshidi, 2015; Dragoni, Dustdar, Larsen e Mazzara, 2015; Taibi, Lenearduzzi e Pahl, 2017); Gouigoux e Tamzalit, 2017) e, por fim, foram identificadas as principais ferramentas adotadas nos estudos, totalizando 16, porém destaca-se o uso de conteiners com Docker (Balalaie, Heydarnoori e Jamshidi, 2015; Dragoni, Dustdar, Larsen e Mazzara, 2015; Gouigoux e Tamzalit, 2017; Yugopuspito, Panduwinata e Sutrismo, 2017) e o Kubernets (Balalaie, Heydarnoori e Jamshidi, 2015; Dragoni, Dustdar, Larsen e Mazzara, 2015). Além dos pontos positivos e conjunto de procedimentos, técnicas e ferramental analisados nos trabalhos, também foram identificados problemas para a migração, do total de dezenove problemas três se destacam, sendo a dissociação do monolitico a mais drastica (Baresi, Garriga e Renzis, 2017; Chen, Li e Li, 2017; Fritzish, Bogner, Zimmermann e Wagner, 2018; Koschel, Astrova e Dötterl, 2017; Taibu, Lenarduzzi, Pahl e Janes, 2017; Taibi, Lenearduzzi e Pahl, 2017) requerendo estudos para criar modelos para esta tarefa, seguido da comunicação entre microsserviços (Gouigoux e Tamzalit, 2017; Koschel, Astrova e Dötterl, 2017; Taibi, Lenearduzzi e Pahl, 2017; Yugopuspito, Panduwinata e Sutrismo, 2017) e a aplicação da cultura DevOps (Gouigoux e Tamzalit, 2017; Koschel, Astrova e Dötterl, 2017; Taibu, Lenarduzzi, Pahl e Janes, 2017; Taibi, Lenearduzzi e Pahl, 2017) requerendo assim mais estudos experimentais sobre os problemas.

Os estudos sobre migração entre sistemas monoliticos para microsserviços são relativamente novos, a análise mostra que a maioria dos trabalhos apresentam soluções específicas, fato que exige mais pesquisas fundamentais, práticas reutilizáveis,

esperimentais e lições aprendidas. Também podemos destacar que uma parte significativa dos estudos analisados, está intrinsecamente ligada a contêineres baseados em nuvem para implantação dos microsserviços.

Levando em consideração a ausência de um processo de migração generalizado funcionalmente comprovado, é relevante planejar como trabalho futuro prosseguir a pesquisa aplicando experimentos aos processos apresentados em Taibi, Lenearduzzi e Pahl (2017). A partir disso, será possível analisar de acordo com os resultados se os processos são funcionais ou não funcionais. Aliás, como trabalho futuro, estamos planejando acrescentar nossa pesquisa incorporando mais trabalhos que incluam literatura cinzenta, como blogs e sites, entre outras fontes.

References

Balalaie, A., Heydarnoori, A., & Jamshidi, P. (2015, September). Migrating to cloud-native architectures using microservices: an experience report. In European Conference on Service-Oriented and Cloud Computing (pp. 201-215). Springer, Cham.

Baresi, L., Garriga, M., & De Renzis, A. (2017, September). Microservices identification through interface analysis. In European Conference on Service-Oriented and Cloud Computing (pp. 19-33). Springer, Cham.

Basili, V. R., & Weiss, D. M. (1984). A methodology for collecting valid software engineering data. IEEE Transactions on software engineering, (6), 728-738.

Chen, R., Li, S., & Li, Z. (2017, December). From monolith to microservices: a dataflow-driven approach. In 2017 24th Asia-Pacific Software Engineering Conference (APSEC) (pp. 466-475). IEEE.

Christoforou, A., Garriga, M., Andreou, A. S., & Baresi, L. (2017, November). Supporting the decision of migrating to microservices through multi-layer fuzzy cognitive maps. In International Conference on Service-Oriented Computing (pp. 471-480). Springer, Cham.

Dragoni, N., Dustdar, S., Larsen, S. T., & Mazzara, M. (2017). Microservices: Migration of a mission critical system. arXiv preprint arXiv:1704.04173.

Fritzsch, J., Bogner, J., Zimmermann, A., & Wagner, S. (2018, March). From monolith to microservices: a classification of refactoring approaches. In International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment (pp. 128-141). Springer, Cham.

Gouigoux, J. P., & Tamzalit, D. (2017, April). From monolith to microservices: Lessons learned on an industrial migration to a web oriented architecture. In 2017 IEEE International Conference on Software Architecture Workshops (ICSAW) (pp. 62-65). IEEE.

Gysel, M., Kölbener, L., Giersche, W., & Zimmermann, O. (2016, September). Service cutter: A systematic approach to service decomposition. In European Conference on Service-Oriented and Cloud Computing (pp. 185-200). Springer, Cham.

Koschel, A., Astrova, I., & Dötterl, J. (2017, July). Making the move to microservice architecture. In 2017 International Conference on Information Society (i-Society) (pp. 74-79). IEEE.

Silva, Í. D. (2018). Almanaque para popularização de ciência da computação.

Taibi, D., Lenarduzzi, V., & Pahl, C. (2017). Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation. IEEE Cloud Computing, 4(5), 22-32.

Taibi, D., Lenarduzzi, V., Pahl, C., & Janes, A. (2017, May). Microservices in agile software development: a workshop-based study into issues, advantages, and disadvantages. In Proceedings of the XP2017 Scientific Workshops (pp. 1-5).

Thönes, J. (2015). Microservices. IEEE software, 32(1), 116-116.

VSchugerl, P., Rilling, J., Witte, R., & Charland, P. (2009, September). A quality perspective of software evolvability using semantic analysis. In 2009 IEEE International Conference on Semantic Computing (pp. 420-427). IEEE.

Yugopuspito, P., Panduwinata, F., & Sutrisno, S. (2017, October). Microservices architecture: case on the migration of reservation-based parking system. In 2017 IEEE 17th International Conference on Communication Technology (ICCT) (pp. 1827-1831). IEEE.

Zamboni, A. B.; Thommazo, A. D.; Hernandes, E. C. M.; Fabbri, S. C. P. F. StArt (State of the Art through Systematic Review). http://lapes.dc.ufscar.br/tools/start_tool.2010.