INTERFACE TECNOLÓGICA

MIGRAÇÃO DE SOFTWARE MONOLÍTICO PARA MICRO SERVIÇOS: uma revisão sistemática da literatura

MIGRATION FROM MONOLITHIC ARCHITECTURE TO MICROSERVICE ARCHTECTURE: a systematic review of literature

Mariana Trevisoli Gervino – marigervino@gmail.com

Faculdade de Tecnologia de Taquaritinga (FATEC) –SP –Brasil

Universidade Federal de São Carlos (UFSCar) – São Carlos – São Paulo – Brasil

DOI: 10.31510/infa.v17i1.700

RESUMO

Com a necessidade de escalabilidade dos sistemas, times multi territoriais e agilidade no processo de desenvolvimento para, assim, possibilitar entregas mais rápidas e de qualidade, há uma crescente demanda pela migração de softwares construídos usando arquitetura monolítica para arquitetura de micro serviços. Esta migração é difícil e cheia de riscos pois os sistemas monolíticos são fortemente acoplados. A presente revisão sistemática sintetiza trabalhos científicos para um estudo e maior compreensão sobre esta migração de arquiteturas. Na pesquisa, foram identificados dezesseis trabalhos, dos quais dez foram analisados, sintetizando no presente trabalho motivos para a migração acontecer, ferramentas e práticas utilizadas no processo e os desafios encontrados durante a migração.

Palavras-chave: Migração. Transição. Arquitetura Monolítica. Arquitetura de Micro serviços. Micro serviços.

ABSTRACT

The constant increase of the need for scalability of softwares, multi territorial teams and agility in the development process to enable faster and high quality deliveries, there is a growing demand for the migration from monolithic architecture to microservice architectute. The migration process is difficult and risky due to monolithic softwares are tightly coupled. This systematic review synthesizes scientific works for a study and better understanding of this architecture migration. During the research phase, 16 papers were performed, 10 of which analyzed, thus, summarizing in this paper reasons which leads companies to migrate from monolithic architecture, tools and practices used during the migration process and challenges faced.

Keywords: Migration. Transition. Monolithic Architecture. Microservice Architecture. Micro-services.

1 INTRODUÇÃO

Softwares que possuem arquiteturas monolíticas são de difícil escalabilidade e implementação de novas funcionalidades pois são uma única aplicação interdependente e que divide a mesma base de dados entre todas as funcionalidades (Carvalho et al., 2019). Portanto, muitas empresas, como a Netflix, Soundcloud e Amazon (Eski e Buzluca, 2018), seguem adotando a migração para a arquitetura de micro serviços para contornar as dificuldades da monolítica, permitindo a escalabilidade, fácil manutenção e implementação de novas funcionalidades e entrega contínua.

A presente revisão sistemática busca entender, através da análise de trabalhos científicos, motivos que levam à migração da arquitetura monolítica para micro serviços, ferramentas e práticas usadas e os desafios enfrentados durante este processo, para entender também como evitá-los.

O presente trabalho está organizado da seguinte forma: na seção 2 são apresentados conceitos teóricos sobre arquiteturas monolítica e de micro serviços e entrega e integração contínuas de software. Na seção 3 é apresentada a metodologia de pesquisa utilizada. Na seção 4 são apresentados os resultados da revisão sistemática. Na seção 5, todos estudos analisados são sintetizados. Por fim, na seção 6 é apresentada a conclusão do presente estudo.

2 FUNDAMENTAÇÃO TEÓRICA

Softwares construídos usando arquitetura monolítica são sistemas que possuem todo seu código encapsulado em uma única aplicação. Essa arquitetura possui um alto grau de acoplamento, significando que seus componentes são interdependentes.

De acordo com Kazanavičius e Mazeika (2019), esta arquitetura é interessante para uma aplicação que está no início da vida, não muito grande, que não possui equipes multi territoriais, tornando fácil a implementação, o desenvolvimento e a configuração da aplicação pois se trata de um único sistema, onde todas funcionalidades estão fortemente acopladas. Conforme essas aplicações tem a necessidade de expansão, expandem também a dificuldade e o risco em dar manutenção ao código existente e adicionar novas funcionalidades, visto que o sistema é interdependente (Kalske, Makitalo e Mikkonen, 2018). Como o desenvolvimento é arriscado

pois uma parte do código pode prejudicar o funcionamento do sistema como um todo, a entrega de software em produção é muito espaçada. Outra dificuldade está na necessidade de times que dividem o mesmo espaço físico, visto que todas equipes desenvolvem no mesmo sistema que é fortemente acoplado, o ideal é não possuir equipes multi territoriais.

Figura 1 – Organização de entrega em um sistema de arquitetura monolítica
Fonte: Kalske, Makitalo e Mikkonen (2018)

A figura 1 mostra uma forma de entrega muito comum em uma arquitetura monolítica e, a partir dela, podemos entender o motivo das dificuldades listadas acima. Com a inter dependência entre todas as partes do sistema, cada funcionalidade nova ou manutenção em alguma parte, impacta diretamente em todas as partes do sistema, diminuindo a confiabilidade e integridade. Isso impacta nas entregas que, para diminuir os riscos, é cada vez mais espaçada. Como uma só aplicação, o código (mesmo que siga os padrões de código limpo - o *Clean Code*) não é totalmente modularizado, o que impacta na curva de aprendizado da equipe de desenvolvimento no sistema, na necessidade em equipes trabalhando juntas e não multi territoriais e na adoção de novas tecnologias na aplicação.

Para superar as limitações da arquitetura monolítica, está cada vez mais sendo usada a arquitetura de micro serviços, que consiste em conjuntos de serviços interdependentes que se comunicam entre si, organizados sobre a capacidade de negocio, com *deploy* automatizado e controle decentralizado de linguagens de dados (Fowler e Lewis, 2014).

O baixo acoplamento que a interdependência de serviços traz permite que times multi territoriais trabalhem com equipes focadas em serviços, com entrega de serviços independentes em produção que pode usar suas próprias tecnologias e também sua própria base de dados. Esta interdependência faz com que os times tenham a liberdade de escolher a tecnologia, não se prendendo a uma apenas, adicionem novas funcionalidades e detêm manutenção no código mais facilmente.

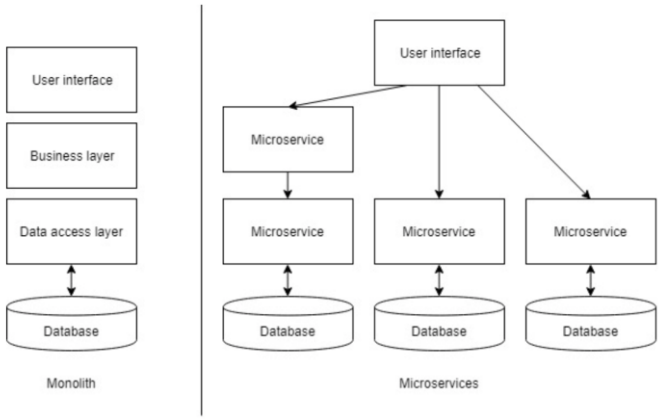

Figura 2 – Comparação entre arquiteturas monolítica e de micro serviços
Fonte: Kazanavičius e Mazeika (2019)

A figura 2 exibe uma comparação entre as arquiteturas, na qual podemos ver que a arquitetura monolítica tem a mesma base de dados, o que significa que a escalabilidade do sistema é difícil e modificações na base de dados devem ser coordenadas entre todas as equipes de desenvolvimento, por possuir uma só (Kalske, Makitalo e Mikkonem, 2018). Ainda de acordo com Kazanavičius e Mazeika (2019), a arquitetura de micro serviços possibilita a adoção de uma base de dados por serviço, abrindo o leque de tecnologias, dando maior suporte a escalabilidade da aplicação e facilitando modificações.

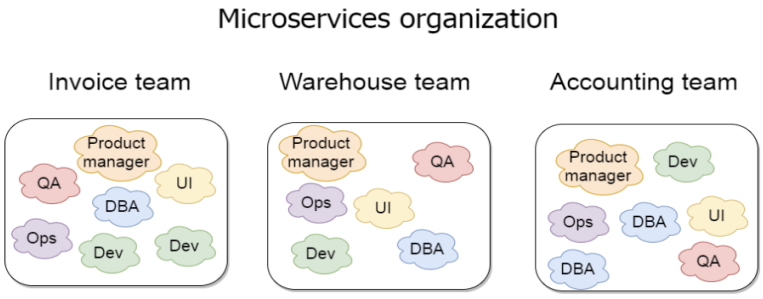

Figura 3 – Organização das equipes em uma arquitetura de micro serviços
Fonte: Kalske, Makitalo e Mikkonem (2018)

A figura 3 mostra a organização das equipes e sua mobilidade na arquitetura de micro serviços, mostrando como essa organização auxilia na distribuição dos times em mais de um

território, a garantir a rapidez e qualidade das entregas, visto que as equipes são focadas em serviços, facilitando assim a entrega e integração contínuas. Um pipeline de integração contínua consiste também em práticas de DevOps para que o código seja desenvolvido e testado de maneira ágil, tenha o deploy automatizado e a entrega aconteça diariamente (Kalske, Makitalo e Mikkonen, 2018).

3 MÉTODO DE PESQUISA

A presente pesquisa foi feita através da análise de trabalhos científicos sobre o tema, que foram sintetizados nesta Revisão Sistemática de Literatura e que tem por base as diretrizes de Kitchenham e Charters (2009), sendo as diretrizes: formulação das questões de pesquisa; busca nas bases científicas através dos parâmetros de busca pré estabelecidos; avaliação da qualidade através de critérios de seleção e rejeição dos trabalhos; seleção dos trabalhos; extração dos trabalhos; interpretação dos resultados.

Na fase de busca foi usada uma das maiores bases de dados de trabalhos da Ciência da Computação, o Scopus. Durante as fases de avaliação de qualidade e seleção dos trabalhos foi usado o software Start[1].

4 RESULTADOS

Esta sessão apresenta detalhadamente como foi planejada a busca, como foram filtrados os trabalhos resultantes desta busca para a análise e como foi feita a extração de dados destes trabalhos.

4.1 Planejamento

A presente pesquisa teve como foco a análise e seleção de literatura científica para um processo atual e cada vez mais crescente que é a transição de sistemas com arquitetura monolítica para arquitetura de micro serviços, que é um processo trabalhoso e difícil. Donos de

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

sistemas monolíticos, apesar da necessidade, se encontram muitas vezes relutantes em aceitar a migração por conta das dificuldades e riscos que a mesma implica (Sarkar, Vashi e PP, 2018).

Frente a essa crescente necessidade, o presente estudo é feito sobre artigos científicos que relatam essa migração, os processos utilizados e os desafios encontrados.

As perguntas planejadas para o desenvolvimento dessa revisão sistemática foram: Como fazer a migração de um sistema monolítico ou legado para micro serviços?; Quais processos e ferramentas são mais indicados durante a migração de sistemas monolíticos ou legados para micro serviços?; Como otimizar a migração de sistemas monolíticos ou legados para micro serviços, em busca de minimizar os desafios que vem com essa migração?

4.2 Execução - identificação dos estudos

De acordo com as perguntas planejadas para o tema do artigo, foi feita uma *String* de busca com palavras chave na base de dados digital Scopus.

Palavras chave: {transition OR Migration} AND { microservice OR architecture} AND {“monolithic architecture” OR “legacy system” OR “legacy software”}.

O resultado dessa busca foi um corpo literário formado por 16 artigos, dos quais 10 foram aceitos e 6 foram rejeitados.

4.3 Execução - seleção dos estudos

Antes da busca ser feita, foram desenvolvidos critérios de aceite e de rejeição dos artigos trazidos na busca, resultando em um total de 6 artigos rejeitados, o que equivale a 37.5% de todo o corpo literário retornado na busca.

Os critérios de inclusão (CI) para os artigos da busca são:

CI1. Descrever a transição de um software ou sistema legado ou monolítico para micro serviço;

CI2. Ser um estudo secundário sobre ferramentas de apoio ao processo de transição de um software ou sistema legado ou monolítico para micro serviço;

CI3. Estar disponível nos idiomas Inglês ou Português;

CI4. Estar disponível nas plataformas de busca utilizadas.

Os critérios de exclusão (CE) para artigos retornados na busca são:

CE1. Não estar disponível nas plataformas utilizadas;

CE2. Não se tratar de uma transição entre um software ou sistema monolítico ou legado para micro serviços;

CE3. Não estar disponível nos idiomas Inglês ou Português.

A seleção se deu com base no título e introdução dos trabalhos retornados após a busca através da combinação das palavras chave no Scopus.

4.4 Execução - extração dos dados

A extração dos dados se iniciou após a etapa de seleção dos trabalhos ser concluída, através da análise dos 10 trabalhos que atenderam ops critérios de aceite e buscando extrair de cada um deles respostas não que diz respeito ao motivo da migração, como essa migração foi feita e os desafios encontrados durante a migração.

5 DISCUSSÃO

A análise dos trabalhos retornados teve enfoque no entendimento do motivo da migração de arquitetura monolítica para de micro serviços, ferramentas e práticas usadas e os desafios encontrados no decorrer da migração.

Quadro 1 – Síntese de dados dos artigos aceitos referentes a migração

Referências	Motivo da migração de monolíto para micro serviços	Ferramentas/Práticas usadas	Desafios encontrados
Balaie, Heydarnoori e Jamshidi (2016)	Dificuldade em implementar novas funcionalidades como serviços.	Implementação de *Pipeline* de integração contínua e práticas de DevOps.	Repensar arquitetura e a necessidade de metodologia de migração bem planejada.

Carvalho e outros (2019)	Dificuldade de expansão e adição de novas funcionalidades	Aponta a importância da variabilidade de análises e de extrações dos microserviços.	Baixo conhecimento disseminado sobre a extração de código do sistema monolítico e refatoração para micro serviços.
Cojocaru, Uta e Oprescu (2019)	Novos membros da equipe precisam de tempo para entender o sistema e releases limitadas pela dificuldade na adição de novas funcionalidades.	Ferramenta semi automática de extração de código guiada por critérios de acoplamento. Funcionalidades são extraídas do monolíto, convertidas para micro serviços e conectadas ao monolíto novamente para validação.	Identificar os limites da interface e busca por granularidade apropriada para a implementação automática.
da Silva, Carneiro e Monteiro (2019)	Fortemente acoplado, dificultando manutenção, correções, adições ou atualizações de bibliotecas.	Metodologia DDD; modularização; desacoplamento das camadas do sistema; padrão *Strangler* e primeiro padrão SOLID.	Aumento de complexidade; custos para a implementação.
Eski e Buzluca (2018)	Baixa escalabilidade, confiabilidade, agilidade e produtividade. Dificuldade de implantação e do tempo para mercado (*time-to-market*).	Extração automática de código do sistema monolítico para micro serviços.	Dificuldade da equipe de desenvolvimento entender completamente o sistema monolítico e em reutilizar código.

Ghayyur e outros (2018)	Dificuldade na adaptabilidade às mudanças tecnológicas, tempo para o mercado (*time-to-market*) e estruturação dos times de desenvolvimento em torno dos serviços.	Práticas e *pipeline* de Integração Contínua.	Repensar a arquitetura; testes mais desafiadores; custo da migração; desacoplamento dos bancos de dados; monitoramento de performance sob desenvolvimento contínuo.
Kalske, Makitalo e Mikkonen (2018)	Alto acoplamento de micro serviços, tornando mais difícil e demorada a manutenção e a escalabilidade.	Alta cobertura de testes; mudança da arquitetura da organização para cultura DevOps; monitoramento e controle de *logs*.	Comunicação entre serviços, orquestração dos serviços em produção; desafios técnicos de infraestrutura; migração é demorada; gerenciamento de dados.
Kazanavicius e Mazeika (2019)	Dificuldades em dar manutenção ao sistema e em adicionar novas funcionalidades.	Apresenta matriz de estudos de caso de migrações feitas a partir da reconstrução do sistema e a partir da refatoração do sistema.	Migração não acontece rapidamente, os custos da decomposição do sistema antigo para a nova arquitetura são altos.
Richter e outros (2017)	Dificuldade na implementação de novas funcionalidades e manutenção de bugs.	Diminuir o acoplamento dividindo o sistema em domínios conectados; Docker; serviços em nuvem.	Testes e monitoramento são necessários; preservar a consistência dos dados; *overhead* de comunicação.

Sarkar, Vashi e PP (2018)	Dificuldade de escalabilidade sobre demanda.	Usaram container-ização - Docker.	Sistema Operacional Windows (Docker funciona melhor em Unix).

Fonte: Própria autora

O quadro 1 mostra todos os dados de cada artigo aceito e analisado. Sintetizando os motivos para a transição do sistema monolítico para outra arquitetura apresentados no quadro acima, os principais são, por ordem de maior citação para menor: dificuldade em adicionar novas funcionalidades (9 citações); dificuldade para escalar (4 citações); dificuldade na correção e entregas de *bugs* (4 citações); tempo de entrega, ou *time-to-market*, alto (3 citações); dificuldade na integração e na curva de aprendizado com o sistema da equipe de desenvolvimento (3 citações). A partir das dificuldades listadas, entendemos a importância e a urgência na migração de arquiteturas.

De acordo com os trabalhos analisados, as principais práticas para a migração são: a implementação de um *pipeline* de integração contínua com práticas de DevOps. Todos artigos que citam testes, Docker, containerização, monitoramento e controle de *logs* estão falando de sub-processos de um *pipeline* de Entrega Contínua; refatoração ou reconstrução do sistema, definindo limites no sistema monolítico e quebrando-o em funções e classes menores usando, por exemplo, o *Single Responsibility Principle*, que é um dos princípios do SOLID e que diz que uma classe deve ter apenas uma responsabilidade e boas práticas de programação usando DDD - Design Orientado á Domínio (Martin, 2002); aumento dos testes e integração do código migrado para micro serviços aos poucos. Dois trabalhos citam ferramentas de automação para a identificação de código no sistema monolítico e refatoração automática para um desacoplamento do código e modularização. Também é citado nestes artigos a importância da supervisão e análise humanas nesse processo.

Quanto aos desafios encontrados os trabalhos citam a dificuldade em encontrar as fronteiras de funcionalidades no código monolítico para refatorar e modularizar para micro serviços; lidar com a comunicação entre os serviços e orquestração dos mesmos; os custos em modificar a arquitetura e infraestrutura necessárias para a migração e também os custos de desenvolvimento e testes; a migração não é feita em um curto espaço de tempo e exige planejamento prévio da migração; implementação de testes mais robustos e monitoramento e tratamento adequado e cuidadoso dos dados.

5 CONCLUSÃO

Como foi apresentado no presente trabalho, a migração de uma arquitetura monolítica para de micro serviços é trabalhosa e custosa, porém se faz indicada quando o sistema cresce e há a necessidade de times multi territoriais, entrega rápida e modificações crescentes no sistema. Além de permitir a democratização da adoção de tecnologias que funcionam melhor para certas partes do sistema e para certos times de desenvolvimento. Existem ferramentas que automatizam partes deste trabalho porém não é recomendado a utilização dessas ferramentas sem um acompanhamento e análise humanas.

As lições aprendidas a partir desta revisão sistemática são para sempre manter a qualidade e maior modularização de código, mesmo que o sistema usado seja monolítico, para num futuro evitar o máximo os percalços de uma migração.

REFERÊNCIAS

BALAIE, A., HEYDARNOORI, A., JAMSHIDI, P. **Microservices architecture enables devops**: an experience report on migration to a cloud-native architecture. IEE Software, 2016. Vol. 33, pp. 1-1, 05.

CARVALHO, L., GARCIA, A., ASSUNÇÃO, WKG., BONIFÁCIO, R., TIZZEI, LP., COLANZI, TE. **Extraction of configurable and reusable microservices from legacy systems**: an exploratory study. 23rd Internation Systems and Software Product Line Conference, 2019. Vol. A.

COJOCARU, MD., UTA, A., PRESCU, AM. **Attributes assessing the quality of micro services automatically decomposed from monolithic applications**. International Symposium on Parallel and Distributed Computing (ISPDV), 2019. pp. 84-93.

DA SILVA, H., CARNEIRO, G., MONTEIRO, M. **Towards a roadmap for the migration of legacy software systems to a microservice based architecture**. Proceedings of the 9th International Conference on Cloud Computing and Services Science (CLOSER), 2019.

ESKI, S., BUZLUCA, F. **An automation extraction approach** - transition to micro services architecture from monolithic application. International Symposium on Parallel and Distributed Computing, 2018.

FOWLER, M., LEWIS, J. **Microservices**. Disponível em: http://martinfowler.com/articles/microservices.html. Acesso em 16 dez. 2019.

GHAYYUR, S., RAZZAQ, A., ULLAH, S., AHMED, S. **Matrix clustering based migration of system application to microservice architecture**. Internation Journal of Advanced Computer Science and Application, 2018. Vol. 9.

KALSKE, M., MAKITALO, N., MIKKONEN, T. **Challenges when moving from monolith to microversice architecture**, 2018. 02 pp. 32-47

KAZANAVICIUS, J., MAZEIKA, D. **Migrating legacy software to microservices architecture**. Open Conference of Electrical, Electronic and Information Sciences (eStream), 2019.

KITCHENHAM, B., BRERETON, P., BUDGEN, D., TURNER, M., BAILEY, J., LINKMAN, S. **Systematic literature reviews in software engineering** - A systematic literature review. Information and Software Technology, 2009. V. 51, pp.7-15.

MARTIN, R. C. **The single responsibility principle**. The principles, patterns, and practices of Agile Software Development, 2002. 149:154.

RICHTER, D., KONRAD, M., UTECHT, K., POLZE, A. **Higly-available applications on unreliable infrastructure**: microservice architecture in practice. International Conference on Software Quality, Reliability and Security, 2017.

SARKAR, S., VASHI, G., PP, A. **Towards transforming an industrial automation system from monolithic to microservices**. ABB Corp. Research India, 2018.