Migração de Aplicações Monolíticas para Microsserviços com Uso de Inteligência Artificial

Leonardo Henrique da Rocha Araujo[1]**(doutorando),**
Guillermo Horacio Rodriguez[1]**(orientador),**
Rodrigo Pereira dos Santos[2]**(co-orientador),**
Fábio Gomes Rocha[3]**(co-orientador)**

[1]Doctorado en Ciencias de la Computación
Instituto Superior de Ingeniería de Software de
Tandil (ISISTAN) – Universidad Nacional del Centro de la Provincia
de Buenos Aires (UNICEN) – Tandil – Buenos Aires – Argentina
Ingresso: 04/2019 – Qualificação: 07/2021 – Previsão Defesa: 12/2023

{leonardo.araujo,guillermo.rodriguez}@isistan.unicen.edu.ar

[2]Programa de Pós-Graduação em Informática
Universidade Federal do Estado do Rio de Janeiro (UNIRIO)

rps@uniriotec.br

[3]Departamento de Ciência da Computação
Universidade Tiradentes (UNIT) – Aracaju, SE – Brasil

fabio_rocha@itp.org.br

Abstract. *The microservices architecture has been increasingly adopted, with the objective of providing a scalable, versatile software system with continuous integration. However, there are still legacy systems with the monolithic architecture. In order to take advantage of the benefits of microservices' architecture, a migration is necessary. The migration of monolithic systems to a microservice architecture faces a few challenges, such as the need for microservices expert developers and time taken to complete the process. This research aims to propose an alternative to a more efficient migration with a higher level of automation, assisting developers regardless of their expertise in microservice architecture.*

Resumo. *A arquitetura de microsserviços vem sendo adotada cada vez mais com o objetivo de prover um sistema de software escalável, versátil e com integração contínua. Porém, ainda há uma enorme quantidade de sistemas legados com a arquitetura monolítica. Para um aproveitamento das vantagens da arquitetura de microsserviços, é necessária uma migração. A migração de sistemas monolíticos para arquitetura de microsserviços possui dificuldades, tais como carência de desenvolvedores especialistas em microsserviços e tempo despendido para a migração. Esta pesquisa visa propor uma alternativa para uma migração mais eficiente, com maior nível de automatização, auxiliando programadores independente de sua experiência em microsserviços.*

Palavras-chave: *Sistema Monolítico, Microsserviços, Migração de arquitetura*

Evento CBSoft: *WTDSoft*

1. Caracterização do Problema

A arquitetura de microsserviços vem sendo apresentada como uma alternativa para a arquitetura monolítica, visto que aplicações monolíticas apresentam uma série de desvantagens, como: alta complexidade de código, dificuldade de realizar manutenção e implantação (*deployment*) da aplicação e problemas de desempenho ao escalar essas aplicações [Fritzsch et al. 2018]. No entanto, na adoção de microsserviços, também há problemas, como a complexidade em seu design e necessidade de especialistas em microsserviços para desenvolvimento [Di Francesco et al. 2019].

Dessa forma, o primeiro problema notado é: como é feito o desenvolvimento de microsserviços de maneira eficiente? Além disso, observando situações de migrações de aplicações monolíticas para microsserviços em empresas, pode-se notar que algumas dificuldades na hora de adotar essa arquitetura são a complexidade para desacoplamento, orquestração e separação de dados [Taibi et al. 2017].

Uma das características mais importantes para que um sistema possa aproveitar as vantagens de uma arquitetura de microsserviços — escalabilidade, independência e simplicidade — é ter uma granularidade adequada [Fritzsch et al. 2018]. Granularidade é o tamanho da funcionalidade do microsserviço [Shadija et al. 2017]. Consequentemente, desenvolver uma aplicação baseada em microsserviços se torna uma tarefa que requer desenvolvedores especializados nessa arquitetura para um bom aproveitamento. Considerando que é uma arquitetura recente, cujos estudos com maior volume começaram a partir de 2015 [Di Francesco et al. 2019], encontrar desenvolvedores experientes é um desafio.

Após a aplicação ser desenvolvida, mesmo com a arquitetura de microsserviços, é necessário fazer seu *deployment* para disponibilizar o sistema aos usuários. Ainda que haja formas de utilização de *containers* (processos isolados com uso reduzido de recursos) para distribuição de microsserviços, observa-se uma quantidade escassa de estudos relacionados à integração adequada entre ambos [Pahl et al. 2017]. Dessa forma, parte do objetivo do estudo é automatizar o *deployment*, configurando os *containers* e finalizando o processo, uma vez que os microsserviços estejam implementados.

Logo, a questão norteadora da pesquisa é: que conjuntos de passos são necessários para efetuar uma migração de um sistema monolítico para arquitetura de microsserviços, de forma eficiente e com algum grau de automação? Para isso, será definido um modelo teórico de migração completa, desde uma base de informação do sistema (seja o código fonte ou documentação) até a implantação dos microsserviços. Também será desenvolvido um método automático de implantação dos microsserviços. Dessa forma, mesmo após mudanças no sistema, o modelo poderá gerar novos microsserviços, caso seja necessário, e implantá-los automaticamente novamente.

Portanto, o objetivo dessa pesquisa é desenvolver um modelo teórico e, em seguida, uma ferramenta capaz de auxiliar na realização de migrações de aplicações monolíticas para a arquitetura de microsserviços do ponto de vista de desenvolvedores de sistemas com pouca ou nenhuma experiência nessa área. Esse objetivo será alcançado por meio dos seguintes objetivos específicos:

- investigar métodos, técnicas e ferramentas que auxiliam a migração de sistemas monolíticos para a arquitetura de microsserviços;
- definir critérios importantes na migração e fatores que impactam nesse processo;

- elaborar um modelo teórico de migração, definindo o processo desde a etapa inicial (código-fonte ou documentação do sistema) até o *deployment* dos microsserviços;
- desenvolver a ferramenta que realize a migração, automatizando o *deployment* dos microsserviços gerados.

2. Fundamentação teórica

A arquitetura de sistemas começou a ser elaborada com um foco em prover aplicações de forma rápida e econômica [Hofmeister et al. 2000]. Nesse sentido, o modelo monolítico cumpre o objetivo, já que algumas de suas vantagens são implementação simples e capacidade proporcionar funções em um curto espaço de tempo.

Um sistema monolítico tem, por definição, suas funcionalidades agrupadas em apenas uma aplicação [Ponce et al. 2019]. Entretanto, esse modelo de arquitetura conta com uma série de desvantagens: dificuldade de entendimento do código-fonte, produtividade reduzida por um algoritmo extremamente complexo, maior dependência entre suas funções, limitação no uso de diferentes tecnologias e problemas de escalabilidade e desenvolvimento contínuo [Al-Debagy e Martinek 2018].

Para buscar uma solução para esses problemas, a arquitetura de microsserviços foi desenvolvida. Seu objetivo é prover um conjunto de serviços relativamente pequenos, consistentes, isolados e podendo ser distribuídos independentemente, cada um com um propósito claro e único [Taibi et al. 2017]. Esse paradigma não só visa atender estruturas mais simples, de pequenas empresas, como também dar suporte a demandas de escala global, como cidades inteligentes [Khatoun e Zeadally 2016].

Dessa forma, para fornecer sistemas com a arquitetura de microsserviços, é necessário realizar a migração de arquitetura. Uma das alternativas é com o auxílio da inteligência artificial (IA). A área de inteligência artificial tem como foco o desenvolvimento de modelos matemáticos para realizar tarefas como classificação e separação de dados [Russell e Norvig 2009]. Considerando que as aplicações de IA são desenvolvidas para realizar tarefas que consomem muito tempo e demandam muito conhecimento do usuário em uma certa área, seu uso nessa situação é um facilitador aos desenvolvedores efetuando a migração. Um exemplo de uso na área de engenharia de software é a utilização de técnicas como classificação e predição de resultados por meio de redes neurais para estimar custo em sistemas [Harman 2012].

Uma das áreas da inteligência artificial é o processamento de linguagem natural. Seu objetivo é criar modelos que entendam as línguas utilizadas pelos seres humanos, para assim utilizá-las em tarefas como tradução de textos, criação de tópicos em documentos [Russell e Norvig 2009]. Uma de suas aplicações mais populares é realizar modelagem de tópicos. Os tópicos definidos são conjuntos de palavras com pesos diferentes e estão distribuídos em um documento. Uma das métricas utilizadas para avaliação de tópicos é chamada de coerência, que quantifica tópicos semanticamente significantes [Chang et al. 2009]. Portanto, o uso de modelagem de tópicos pode auxiliar na criação de tópicos que representariam microsserviços em um sistema, utilizando análise semântica.

3. Metodologia

O pesquisa foi dividida em várias etapas, ilustradas na Figura 1, que mostra os estudos a serem realizados e um círculo para representar seu resultado. Inicialmente, foi realizada uma revisão informal da literatura, buscando informações sobre a arquitetura de microsserviços e sua migração. Com isso, foi identificada a dificuldade para migrar para essa arquitetura, com os fatores principais sendo o tempo despendido ao realizar a migração e a falta de experiência dos desenvolvedores. Dessa forma, foi delineado o objetivo de auxiliar a migração de forma que reduza essas dificuldades.

Com o objetivo de analisar a viabilidade de usar modelagem de tópicos para dividir serviços de uma aplicação em grupos, que seriam análogos aos microsserviços, foi realizado um experimento. Foram utilizadas documentações de serviços leves (categoria em que microsserviços se encontram). Foi utilizada uma base de dados de aplicações com especificações OpenAPI, um padrão de documentação de *Application Programming Interfaces* (API) feita para ser lida por seres humanos e computadores. Assim, foi possível separar diversos serviços leves que possuíam uma relação semântica. Esse estudo exploratório foi realizado seguindo as diretrizes de experimentos em engenharia de software [Wohlin et al. 2012]. Foi possível constatar que documentações de serviços leves podem gerar tópicos que classificam serviços em tópicos com um valor de coerência acima do mínimo sugerido para considerá-los semanticamente correlacionados [Röder et al. 2015].

Em seguida, para encontrar formas de migrações que serão futuramente comparadas com o método desenvolvido nesta pesquisa, identificar formas de avaliação para o método e encontrar documentações específicas de microsserviços, está sendo realizado um mapeamento sistemático da literatura. O objetivo desse mapeamento sistemático é catalogar e analisar a eficiência de abordagens — sejam elas métodos, ferramentas ou técnicas — utilizadas para auxiliar na migração de aplicações monolíticas para a arquitetura de microsserviços, especialmente abordagens que utilizam IA.

O mapeamento sistemático está sendo realizado seguindo as diretrizes para estudos desse tipo [Kitchenham e Charters 2007]. Para isso, foi elaborado um protocolo de mapeamento sistemático, contendo o objetivo mencionado acima, questões de pesquisa, palavras chaves, *strings* de busca, bases de estudos utilizadas, critério PICOC (*Population*, *Intervention*, *Comparison*, *Outcomes*, *Context*) e critérios de inclusão e exclusão.

Posteriormente, com os dados coletados, será realizada uma pesquisa de opinião com especialistas em microsserviços. Essa pesquisa de opinião buscará definir quais são os critérios utilizados para uma migração e quais os fatores que impactam esse processo, além de determinar outras métricas de desempenho nessa tarefa além daquelas existentes na literatura. Esse procedimento será realizado para garantir que o modelo teórico de migração considere os fatores adequados, minimizando efeitos indesejados na migração e definindo situações em que seu uso trará melhor benefício.

Com o modelo de tópicos de documentações de serviços leves, ferramentas de migração e métricas de avaliação e critérios que causam a migração e a impactam, a próxima etapa será a construção do modelo teórico de migração. Esse modelo definirá todas as etapas, desde o início da migração, quando uma documentação e/ou código de um sistema ou parte dele será apresentado, até o *deployment* dos microsserviços.

Uma vez que o modelo teórico esteja completo, será iniciado o processo de

implementação da ferramenta que executará a migração. Para garantir um melhor desempenho do algoritmo de modelagem de tópicos, será realizado um estudo utilizando a técnica de análise de texto chamada *word embeddings*, i.e., uma técnica que realiza a transformação de palavras em vetores para demonstrar a relação entre diversas palavras por meio da distância dos seus vetores [Schnabel et al. 2015]. Com o uso de *word embeddings*, é possível definir o quão próxima uma palavra está de outra por seu significado, ou até mesmo se uma palavra é derivada de outra, assim aumentando a coerência dos tópicos.

Seguindo a abordagem Goal Question Metric (GQM) [Van Solingen et al. 2002] o objetivo da ferramenta foi definido: auxiliar em migrações de arquitetura monolítica para microsserviços na perspectiva de desenvolvedores com pouca ou nenhuma experiência em microsserviços. Para isso, foram definidas as perguntas e métricas a seguir.

- Pergunta: Qual é o custo de tempo da migração?
- Métrica: Tempo despendido pelos desenvolvedores para realizar a migração.
- Pergunta: Qual é o nível de qualidade dos microsserviços gerados?
- Métrica: Avaliação de atributos de qualidade (acoplamento, manutenção, coesão e complexidade) nos microsserviços gerados [Cardarelli et al. 2019].

Uma vez que a ferramenta esteja desenvolvida, um estudo de viabilidade será realizado. A ferramenta será apresentada a desenvolvedores, com o intuito de analisar sua facilidade de uso. Caso seja necessário, serão feitos ajustes na ferramenta. Em seguida, a ferramenta será avaliada em questão de utilidade. Para isso, um caso real de migração será testado com e sem a ferramenta. Em um sistema de Gestão Eletrônica de Documentos de uma instituição pública do setor econômico, serão realizadas migrações de uma parte de um sistema. Os desenvolvedores serão divididos em grupos e as ferramentas encontradas no mapeamento sistemático serão divididas entre esses grupos, junto com a ferramenta desenvolvida na etapa anterior. Serão medidos o tempo gasto nos dois cenários e outras métricas identificadas pela pesquisa de opinião feita anteriormente. Em seguida, serão definidos outros grupos para testar as ferramentas que ainda não tiverem sido testadas.

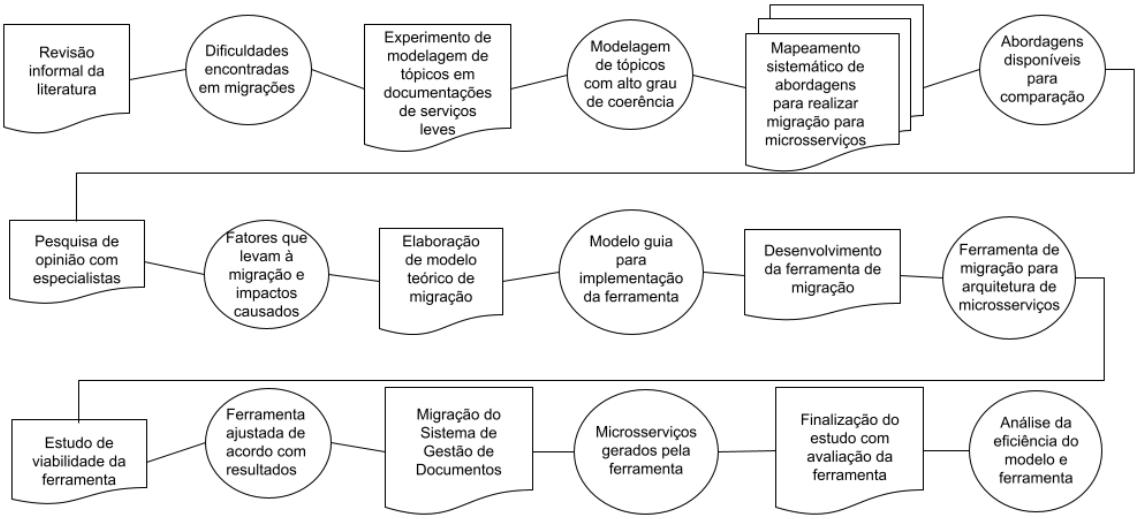

Figura 1. Ilustração da pesquisa. Formatos de documentos representam estudos a serem feitos e os círculos representam os resultados obtidos.

Também será aplicado um questionário para os participantes, que responderão sobre sua experiência ao utilizar a ferramenta, buscando uma avaliação qualitativa. A

meta é definir quais foram as melhorias percebidas pela equipe de desenvolvimento ao utilizá-la. Para a análise dos microsserviços criados, será realizada uma avaliação de atributos de qualidade. Grau de acoplamento, manutenção, coesão e complexidade serão quantificados e avaliados na comparação das duas abordagens [Cardarelli et al. 2019].

4. Resultados Preliminares

Com a finalidade de dividir serviços leves em categorias, de forma análoga ao objetivo de definir os microsserviços em uma aplicação, foi executado um experimento de modelagem de tópicos na base OpenAPI. Foi escolhida essa base de dados, visto que aplicações de microsserviços são geralmente desenvolvidas como API [Baresi et al. 2017]. Nesse estudo, tópicos foram modelados com os três algoritmos mais utilizados — *Latent Dirichlet Allocation* (LDA), *Non-negative Matrix Factorization* (NMF) e *Latent Semantic Indexing* (LSI) — nos três componentes que definem serviços OpenAPI — *Summary* (descrição curta), *Description* (descrição completa) e *Summary* + *Description* (ambos agrupados).

Pelos resultados encontrados, o componente que gera tópicos com maior grau de coerência é a descrição completa do serviço. Os algoritmos NMF e LSI identificam tópicos com valor de coerência similar e significativamente maior que o LDA. Por fim, os serviços similares foram agrupados, por meio da modelagem de tópicos com os parâmetros de definem tópicos com maior valor de coerência (componentes e algoritmos) para gerar um modelo de categorias de serviços leves. Além disso, o modelo gerado com os melhores parâmetros oferece um valor de coerência alto suficiente para considerar os tópicos gerados como tópicos semanticamente correlacionados.

Está sendo realizado um mapeamento sistemático de métodos de migração de aplicações monolíticas para arquitetura de microsserviços, como parte da etapa inicial. Nesse mapeamento, além de métodos de migração tradicionais, estão sendo analisados métodos que utilizam IA, com a finalidade de encontrar uma ferramenta similar àquela que será desenvolvida, visando uma comparação em outra etapa do estudo.

O mapeamento sistemático já apontou alguns modelos para identificação de microsserviços. Foram encontrados diferentes modelos que transformam código-fonte de sistemas em grafos e, em seguida, realizam um agrupamento entre classes relacionadas, para gerar candidatos a microsserviços. Esses modelos serão utilizados ao final para comparação com o modelo desenvolvido pela pesquisa.

Foram identificados no mapeamento estudos com propostas de avaliação de microsserviços. Essa avaliação é feita definindo os graus dos atributos de qualidade dos microsserviços, que depois serão computados por uma ferramenta descrita em outro estudo [Cardarelli et al. 2019]. Com isso, será possível comparar a qualidade dos microsserviços gerados pelo modelo da pesquisa com os gerados pelos outros modelos.

5. Contribuições Esperadas

As contribuições científicas e técnicas esperadas são:

- Contribuições Científicas
  - Identificação das dificuldades gerais em uma migração de arquitetura: por meio da pesquisa de opinião com especialistas, será possível definir, além

de situações que levam à migração, quais são as maiores dificuldades atualmente para os desenvolvedores de sistemas realizarem migrações de arquitetura monolítica para microsserviços.
  - Modelo teórico de realização de migração de um sistema monolítico para arquitetura de microsserviços, com identificação automática de candidatos a microsserviços, implementação manual por parte dos desenvolvedores e *deployment* automático.
- Contribuição Técnica
  - Ferramenta de migração de aplicações de arquitetura monolítica para a arquitetura de microsserviços: a ferramenta permitirá que desenvolvedores realizem a migração de uma aplicação monolítica para arquitetura de microsserviços com pouca ou nenhuma experiência em microsserviços, além de criar um ambiente com esses microsserviços implantados em *containers*, facilitando futuros *deployments*.

6. Trabalhos Relacionados

Os estudos que se relacionam à proposta de doutorado sugerem soluções para alguns problemas mencionados, ou métricas a serem utilizadas para avaliação de microsserviços. Uma das alternativas propostas [Kazanavičius e Mažeika 2019] é realizar a migração analisando e, manualmente, identificando pontos de dependência entre o banco de dados de um sistema e seu código-fonte para assim dividir funções em microsserviços independentes. A desvantagem desse método é que, por ser manual, despende muito tempo e, em casos de bancos de dados com muitas dependências entre tabelas, não gera microsserviços com um grau de granularidade adequado.

Há opções de métodos automatizados [Mazlami et al. 2017], por meio da proposta de criação de um grafo que representa o sistema monolítico por relações entre classes que foram modificadas, histórico de modificações e desenvolvedores que as realizaram. Uma vez que esse grafo é construído, representando o monólito, partes são agrupadas para gerar microsserviços candidatos. Assim, sua abordagem permite criar microsserviços automaticamente, mesmo sem conhecimento prévio.

Uma alternativa similar [Eski e Buzluca 2018] propõe a análise de código estático e do repositório, buscando as modificações realizadas. Com isso, é criado um grafo de dependência entre classes. Dessa forma, classes alteradas em conjunto com frequência estão ligadas. Depois disso, é feito o agrupamento dessas classes por meio do algoritmo *Fast Community Graph Clustering*, por conta de seu resultado em modularização de sistemas orientados a objetos em comparação com outros algoritmos. A desvantagem dessas alternativas é o requerer código fonte em um repositório com histórico de modificações.

Na etapa de avaliação, microsserviços gerados por esses modelos serão comparados com os microsserviços gerados pelo modelo desenvolvido nesta pesquisa. Uma alternativa [Cardarelli et al. 2019] propõe a avaliação de modelos por meio de atributos de qualidade do sistema. Para isso, é necessário definir o grau dos atributos de qualidade acoplamento, manutenção, coesão e complexidade e assim calcular a qualidade geral dos microsserviços utilizando o *Evaluation Engine* proposto pelos autores.

Em geral, os estudos realizados sugerem microsserviços baseados apenas na relação entre classes do código-fonte. Entretanto, o modelo sugerido pelos autores ana-

lisa também as relações semânticas, o que pode resultar em uma granularidade mais adequada, aumentando a eficiência da solução. Ainda é possível uma abordagem híbrida, em que é utilizada a opção com melhor resultado dentre as encontradas no mapeamento sistemático em conjunto com a solução proposta nesta pesquisa. Outro fator não considerado nos estudos anteriores é o *deployment* dos microsserviços gerados, visto que um processo automático é mais conveniente e mais rápido, sendo também menos propenso a erros, considerando a possível falta de experiência na arquitetura de microsserviços do desenvolvedor utilizando a ferramenta.

Referências

Al-Debagy, O. e Martinek, P. A comparative review of microservices and monolithic architectures. In: *Proceedings of the 2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI)*. Budapest, Hungary, pp. 149–154, 2018.

Baresi, L., Garriga, M., e De Renzis, A. Microservices identification through interface analysis. In: *Proceedings of the European Conference on Service-Oriented and Cloud Computing (ESOCC)*. Oslo, Norway, pp. 19–33, 2017. Springer.

Cardarelli, M., Iovino, L., Di Francesco, P., Di Salle, A., Malavolta, I., e Lago, P. An extensible data-driven approach for evaluating the quality of microservice architectures. In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. New York, United States, pp. 1225–1234, 2019.

Chang, J., Gerrish, S., Wang, C., Boyd-Graber, J. L., e Blei, D. M. Reading tea leaves: How humans interpret topic models. In: *Proceedings of Advances in neural information processing systems*. Vancouver, BC, Canada, pp. 288–296, 2009.

Di Francesco, P., Lago, P., e Malavolta, I. Architecting with microservices: A systematic mapping study. *Journal of Systems and Software (JSS)*, 150:77–97, 2019.

Eski, S. e Buzluca, F. An automatic extraction approach: transition to microservices architecture from monolithic application. In: *Proceedings of the 19th International Conference on Agile Software Development: Companion*. Porto, Portugal, pp. 1–6, 2018.

Fritzsch, J., Bogner, J., Zimmermann, A., e Wagner, S. From monolith to microservices: A classification of refactoring approaches. In: *Proceedings of the International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*. Villebrumier, France, pp. 128–141, 2018.

Harman, M. The role of artificial intelligence in software engineering. In: *Proceedings of 2012 First International Workshop on Realizing AI Synergies in Software Engineering (RAISE)*. Zurich, Switzerland, pp. 1–6, 2012.

Hofmeister, C., Nord, R., e Soni, D. *Applied software architecture*. Addison-Wesley Professional, Boston, MA, USA, 5ª edição, 2000.

Kazanavičius, J. e Mažeika, D. Migrating legacy software to microservices architecture. In: *Proceedings of the 2019 Open Conference of Electrical, Electronic and Information Sciences (eStream 2019)*. Vilnius, Lithuania, pp. 1–5, 2019.

Khatoun, R. e Zeadally, S. Smart cities: concepts, architectures, research opportunities. *Communications of the ACM*, 59(8):46–57, 2016.

Kitchenham, B. e Charters, S. Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE-2007-01, Inglaterra, 2007.

Mazlami, G., Cito, J., e Leitner, P. Extraction of microservices from monolithic software architectures. In: *Proceedings of the 2017 IEEE International Conference on Web Services (ICWS)*. Honolulu, USA, pp. 524–531, 2017.

Pahl, C., Brogi, A., Soldani, J., e Jamshidi, P. Cloud container technologies: a state-of-the-art review. *IEEE Transactions on Cloud Computing*, 7(3):677–692, 2017.

Ponce, F., Márquez, G., e Astudillo, H. Migrating from monolithic architecture to microservices: A rapid review. In: *Proceedings of the 2019 38th International Conference of the Chilean Computer Science Society (SCCC)*. Concepcion, Chile, pp. 1–7, 2019.

Röder, M., Both, A., e Hinneburg, A. Exploring the space of topic coherence measures. In: *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. WSDM '15, New York, NY, USA, page 399–408, 2015.

Russell, S. e Norvig, P. *Artificial intelligence: a modern approach*. Pearson, New York, NY, USA, 3ª edição, 2009.

Schnabel, T., Labutov, I., Mimno, D., e Joachims, T. Evaluation methods for unsupervised word embeddings. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Llanguage Processing*. pp. 298–307, 2015.

Shadija, D., Rezai, M., e Hill, R. Microservices: Granularity vs. performance. In: *Companion Proceedings of the 10th International Conference on Utility and Cloud Computing (UCC '17)*. New York, USA, pp. 215–220, 2017.

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

Van Solingen, R., Basili, V., Caldiera, G., e Rombach, H. D. Goal question metric (GQM) approach. *Encyclopedia of software engineering*, 2002.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., e Wesslén, A. *Experimentation in software engineering*. Springer Science & Business Media, New York, NY, USA, 2012.