Arquitetura de Microsserviços: Uma Revisão Multivocal

Larissa Zanata Morais, André F. R. Cordeiro, Edson OliveiraJr

[1]Departamento de Informática – Universidade Estadual de Maringá (UEM)
Maringá – PR – Brazil

ra99495@uem.br, cordeiroandrefelipe@gmail.com, edson@din.uem.br

Abstract. *Microservices architecture has emerged as a robust and efficient approach to software development, enabling the construction of large systems composed of smaller, independent, and interoperable services. In this context, this paper presents a multi-voiced review of microservices architecture, aiming to identify the main topics covered, explore the challenges and proposed solutions, examine the technologies used, and identify research opportunities. This review contributes to a deeper and more comprehensive understanding of microservices architecture, providing insights for software engineering practitioners and researchers.*

Resumo. *Arquitetura de microsserviços tem se destacado como uma forma robusta e eficiente para o desenvolvimento de software, permitindo a construção de grandes sistemas compostos de serviços menores, independentes e interoperáveis. Nesse contexto, o presente trabalho apresenta uma revisão multivocal sobre arquitetura de microsserviços, com o intuito de identificar os principais tópicos abordados, explorar os desafios e soluções propostas, examinar as tecnologias utilizadas e identificar oportunidades de pesquisa. Esta revisão contribui para uma compreensão mais profunda e abrangente da arquitetura de microsserviços, fornecendo percepções para profissionais e pesquisadores na área de engenharia de software.*

1. Introdução

Com o crescimento das necessidades de escalabilidade, flexibilidade e manutenção nas aplicações, as arquiteturas de software têm passado por transformações. Durante anos, as arquiteturas monolíticas foram adotadas em razão de sua simplicidade e coesão [Society 2018]. O aumento crescente na complexidade evidenciou a existência de dificuldades significativas nas aplicações, principalmente em relação à escalabilidade e manutenção [Fowler 2023]. A transição para aplicações baseadas em microsserviços representa uma resposta a essas dificuldades [Bonfiglioli 2021].

A arquitetura de microsserviços influencia o projeto e o desenvolvimento das aplicações, além de favorecer a escalabilidade e a adaptabilidade [Lima and Fontoura 2023]. A arquitetura permite que os desenvolvedores escalem elementos individuais de maneira autônoma, proporcionando maior eficiência no uso dos recursos computacionais. Esse modelo permite que as aplicações se adaptem com maior rapidez às novas demandas [Fowler 2023].

A adaptabilidade pode ser observada durante a divisão das aplicações em serviços modulares. Dessa maneira, os desenvolvedores podem atualizar ou modificar elementos

individuais, sem a necessidade de reestruturar toda a aplicação [Lima and Fontoura 2023]. Essa abordagem modular reduz os riscos associados a mudanças e facilita o desenvolvimento alinhado com as práticas de Integração Contínua (*Continuous Integration* - CI) e Implantação Contínua (*Continuous Deployment* - CD). Tais práticas são fundamentais em ambientes de desenvolvimento caracterizados pela rápida evolução de demandas [Fowler 2023].

Mesmo com os benefícios relevantes associados, é importante destacar que a implementação de uma arquitetura de microsserviços também apresenta desafios. A descentralização dos elementos exige uma coordenação cuidadosa para garantir que os serviços interajam de forma eficaz. A comunicação entre os serviços é outro aspecto crítico, principalmente em ambientes distribuídos [Kubernets 2024].

Diante das informações apresentadas, entende-se que a arquitetura de microsserviços oferece possibilidades relevantes para o desenvolvimento de software, bem como apresenta novos desafios. Estudos multivocais sobre tal arquitetura não foram encontrados na literatura. Considerando essa ausência, este artigo apresenta uma revisão multivocal realizada sobre arquiteturas de microsserviços.

Este trabalho está organizado da seguinte maneira: a Seção 2 apresenta a fundamentação teórica; a Seção 3 apresenta a metodologia aplicada; a Seção 4 apresenta os resultados obtidos; a Seção 5 apresenta a discussão dos resultados; a Seção 6 discute as ameaças à validade; e a Seção 7 apresenta as considerações finais.

2. Arquitetura de Software e Microsserviços

Segundo a IEEE *Computer Society* [Society 2018], uma Arquitetura de Software é uma estrutura para representar uma aplicação. Essa estrutura é construída considerando diferentes decisões sobre a organização dos elementos que compõem o software. Além de influenciar diretamente a qualidade, entende-se que a arquitetura é essencial para lidar com a complexidade das aplicações, favorecendo atributos como desempenho e manutenibilidade [Society 2018].

Diferentes modelos de arquitetura são descritos na literatura [Pressman and Maxim 2021]. A arquitetura de microsserviços representa um desses modelos e é caracterizada por elementos de software independentes, que realizam tarefas específicas. Essa abordagem representa uma evolução das Arquiteturas Orientadas a Serviços [Amazon 2024].

De acordo com Samad, a arquitetura baseada em microsserviços surgiu como uma resposta às grandes demandas do mercado por softwares escaláveis, rápidos e flexíveis [Samad 2021]. Para Maxime, as principais características dos microsserviços são as capacidades de descentralização e desacoplamento, em que cada serviço é responsável por uma única funcionalidade [Maxime 2023]. Tecnologias como *Docker* e *Kubernetes* têm sido utilizadas para conteineres e orquestração, respectivamente [Docker 2024, Kubernets 2024].

A transição de arquiteturas monolíticas para arquiteturas baseadas em microsserviços tem sido um foco crescente de pesquisa devido às vantagens oferecidas em termos de modularidade, escalabilidade e flexibilidade. Essa transição envolve desafios significativos, como a remodularização contínua do software, a integração de novas

abordagens e a gestão da dívida técnica [Bushong et al. 2021].

Para lidar com a complexidade da migração de sistemas monolíticos para microserviços, diferentes soluções têm sido desenvolvidas. Ferramentas de gerenciamento de dependências têm sido utilizadas para detectar "cheiros" de microsserviços [Walker et al. 2020].

3. Metodologia de Pesquisa

Esta revisão multivocal considerou orientações apresentadas na literatura, com o objetivo de identificar materiais relevantes para os objetivos da revisão [Garousi et al. 2019]. Para favorecer a reprodução da metodologia, foram consideradas as etapas de definição dos objetivos e das questões de pesquisa, definição e execução dos processos de busca, seleção, extração e análise. Mais informações sobre cada etapa são apresentadas nesta seção.

3.1. Objetivos e Questões de Pesquisa

O objetivo desta revisão é identificar e analisar as principais tendências, desafios e oportunidades de pesquisa existentes sobre arquiteturas de microsserviços. As seguintes questões de pesquisa (QP) foram consideradas: **QP1**: Quais são as principais tendências e avanços no estudo e aplicação das arquiteturas baseadas em microsserviços?; **QP2**: Quais desafios são enfrentados pelos profissionais na adoção e implementação de microsserviços?; **QP3**: Quais soluções foram propostas para mitigar os desafios identificados na adoção e implementação de microsserviços?; e **QP4**: Quais ferramentas e tecnologias são utilizadas nas arquiteturas baseadas em microsserviços?

3.2. Processo de Busca

A busca pelos materiais foi realizada em diferentes locais. que incluem a Biblioteca Digital da Sociedade Brasileira de Computação, a IEEE Computer Society, o MDPI, a rede ResearchGate e as empresas Amazon, Docker e Kubernetes. Tais locais foram considerados em razão do conhecimento preliminar, antes da revisão, sobre a existência de materiais relevantes nesses locais. Ao final, a string de busca utilizada foi:

("microservices architecture" **OR** "microservices")
AND
("challenges" **OR** "solutions" **OR** "tools" **OR** "trends")

Após a definição, a string de busca foi executada. Ao final da execução, foram retornados 283 materiais.

3.3. Processo de Seleção

A seleção inicial foi realizada considerando um processo estabelecido com o objetivo de garantir a relevância dos materiais selecionados. A contribuição para responder às questões de pesquisa foi o único critério de seleção adotado. O processo considerou as leituras do título, do resumo e das palavras-chave de cada material retornado. Ao final da seleção, foram considerados 24 materiais. O processo foi realizado por uma autora. Inicialmente, foi estabelecido que, em caso de dúvidas sobre a seleção, o artigo deveria ser mantido.

Os materiais incluem dissertações, teses, artigos científicos, relatórios técnicos e fontes de referência técnica de empresas de tecnologia, como *Amazon*, *Docker* e *Microsoft*. Os materiais selecionados estão apresentados na Tabela 1. Esses materiais abordam assuntos relacionados às questões de pesquisa. Para cada material selecionado, foi estabelecido um identificador (ID), bem como o registro do título e do ano de publicação.

Tabela 1. Conjunto final de estudos.

ID	Título	Ano
E1	Estratégias de *deployment* em arquitetura de microsserviços	2023
E2	Desenvolvendo aplicações utilizando a arquitetura de microsserviços	2021
E3	*On Microservice Analysis and Architecture Evolution: A Systematic Mapping Study*	2021
E4	*DevOps, CI/CD and Containerization: 44 Images Explaining a Winning Trio*	2023
E5	*What is Docker*	2024
E6	Arquitetura de Microsserviços	2021
E7	Aplicando Arquitetura de Microsserviços no Desenvolvimento de Software	2021
E8	Uma Arquitetura de Microsserviços Centrada na Observabilidade Multinível para Espaços Inteligentes Baseados em Visão Computacional	2021
E9	Segurança em Micro Serviços: Boas Práticas e Estratégias	2023
E10	*Kubernetes Documentation*	2019
E11	*A Comprehensive Study of the Transition from Monolithic to Microservices-Based Software Architectures*	2023
E12	*Automated Database Schema Evolution in Microservices*	2023
E13	Considerações de dados para microsserviços	2024
E14	*Circuit Breakers, Discovery, and API Gateways in Microservices*	2016
E15	Arquitetura de Microsserviços	2022
E16	Adesão da arquitetura de microsserviços nas grandes corporações para desenvolvimento ou migração de aplicações	2023
E17	Avaliação experimental de uma arquitetura de microsserviços para o gerenciamento de notas fiscais eletrônicas	2022
E18	Micro Serviços: 3 métodos de comunicação entre serviços	2019
E19	*Auto-scaling Policies to Adapt the Application Deployment in Kubernetes*	2020
E20	*Architectural Transition: Unveiling the Shift from Monolithic to Microservices in Digital Experience Platforms*	2021
E21	Proposta Introdutória de Processo Para Seleção de Tecnologias para Comunicação entre Microserviços	2018
E22	Uma abordagem de conformidade arquitetural para arquitetura de microsserviços	2019
E23	Arquitetura de microsserviços: quando vale a pena migrar?	2020
E24	Arquitetura de microsserviços: um estudo de caso	2023

Mais informações sobre os estudos podem ser encontradas em https://zenodo.org/records/17238803.

3.4. Processo de Extração

Após a seleção dos materiais, foi realizada a etapa de extração dos dados. Inicialmente, foi elaborado um formulário de extração contendo as informações necessárias para responder as questões de pesquisa (tendências e avanços, desafios enfrentados, soluções para os desafios e ferramentas e tecnologias). Cada questão de pesquisa representa um campo do formulário.

3.5. Processo de Análise

Os dados extraídos dos materiais foram utilizados para responder às questões apresentadas na Seção 3.1. Os resultados e discussões são apresentados nas Seções 4 e 5, respecti-

vamente.

4. Resultados

Esta seção apresenta os resultados do estudo em função das questões de pesquisa estabelecidas.

4.1. QP1 - Tendências e Avanços

Diferentes avanços e tendências foram observados. Tais resultados estão registrados na Tabela 2.

Considerando os resultados apresentados na Tabela 2, é possível observar diferentes tendências e avanços no estudo e aplicação das arquiteturas baseadas em microsserviços. Por exemplo, o estudo **E5** destaca que a utilização de *containers* facilita a criação de ambientes isolados para cada serviço, promovendo uma maior consistência entre as fases de desenvolvimento, teste e produção. O estudo **E9**, por sua vez, enfatiza que a combinação de DevOps e microsserviços permite a implementação de sistemas mais robustos e resilientes, uma vez que facilita a detecção e correção de falhas de forma contínua e em tempo real.

Outras tendências e avanços estão relacionados com escalabilidade horizontal (**E6**), comunicação por *Application Programming Interface* (API) (**E7** e **E8**), gestão de falhas e resiliência (**E15**), orquestração com kubernetes (**E10**), bancos de dados distribuídos (**E12**), segurança (**E2**), erros de observabilidade (**E8**), conformidade arquitetural (**E22**), desafios de migração (**E23**), gestão de dependências (**E24**), erros no *deployment* (**E1**), particionamento de domínios (**E2**) e estratégias de segurança (**E9**).

4.2. QP2 - Desafios Enfrentados

Uma possível categorização dos principais desafios enfrentados pelos profissionais é apresentada na Tabela 3.

A partir da Tabela 3, é possível observar quais desafios são enfrentados pelos profissionais na adoção e implementação de microsserviços. Entre os desafios registrados estão a escolha de tecnologias de comunicação e a gestão de falhas. O estudo **E21**, sobre a escolha de tecnologias, discute como a seleção de tecnologias deve ser planejada, considerando fatores como a latência, a compatibilidade entre os serviços e as necessidades de escalabilidade. O estudo **E13**, sobre a gestão de falhas e resiliência, descreve que, em arquiteturas de microsserviços, falhas em um serviço podem impactar todo o sistema, sendo fundamental a implementação de mecanismos de resiliência, como *circuit breakers* e *retries* automáticos. Tais técnicas minimizam o impacto das falhas e permitem que o sistema continue operando de forma eficiente, mesmo diante de problemas em um dos serviços.

Outros desafios enfrentados estão relacionados com a comunicação entre serviços (**E18** e **E21**), a escalabilidade (**E15**, **E17** e **E19**), a consistência dos dados (**E15**) e o monitoramento e rastreamento (**E3**, **E14** e **E19**).

4.3. QP3 - Soluções para os Desafios

As soluções propostas para os desafios enfrentados estão registradas na Tabela 4.

Tabela 2. Principais tendências e avanços na área.

Tendência/Avanço	Descrição	ID
Uso de Containers e Orquestração	Containers criam ambientes isolados para cada serviço; Kubernetes facilita a implementação e gerenciamento eficiente de microsserviços	E5
Escalabilidade Horizontal	Permite que cada microsserviço seja escalado de forma independente, atendendo a variações de demanda sem impactar o sistema como um todo	E6
Comunicação por APIs	Tecnologias como REST e GraphQL garantem integração eficiente e consultas otimizadas entre microsserviços	E7, E8
Práticas DevOps e CI/CD	Automação de testes, integração e *deploy* aumenta a rapidez e a qualidade do ciclo de vida dos microsserviços	E9
Gestão de Falhas e Resiliência	Padrões como Circuit Breaker e observabilidade ajudam na recuperação automática de falhas, mantendo a continuidade dos sistemas	E15
Orquestração com *Kubernetes*	Facilita balanceamento de carga, implantação, escalabilidade e gerenciamento de falhas em sistemas distribuídos	E10
Bancos de Dados Distribuídos	Bancos como Cassandra e MongoDB garantem acesso eficiente e escalável aos dados em arquiteturas descentralizadas	E12
Segurança	Implementação de autenticação, autorização por *tokens* e criptografia para proteger dados e comunicação entre microsserviços	E2
Erros de Observabilidade	A falta de ferramentas robustas de monitoramento multinível dificulta o diagnóstico de falhas em sistemas inteligentes baseados em visão computacional, prejudicando a escalabilidade e manutenção	E8
Conformidade Arquitetural	A ausência de padrões bem definidos na comunicação entre serviços causa inconsistências em contratos, impactando a interoperabilidade e a confiabilidade dos sistemas	E22
Desafios de Migração	Decisões de migração sem análises detalhadas de viabilidade podem aumentar desnecessariamente a complexidade e os custos, comprometendo os objetivos do projeto	E23
Gestão de Dependências	Falhas no gerenciamento de versões de serviços prejudicam a integração e evolução contínua, levando a problemas de compatibilidade no sistema	E24
Erros no *Deployment*	A falta de automação e pipelines bem estruturados no processo de entrega contínua aumenta a introdução de falhas em produção e afeta a estabilidade do sistema	E1
Particionamento de Domínios	Dividir funcionalidades sem análise criteriosa resulta em problemas de coesão e acoplamento excessivo, reduzindo a flexibilidade da solução	E2
Estratégias de Segurança	A ausência de autenticação e autorização granulares expõe dados sensíveis e compromete a integridade de sistemas distribuídos e de alta complexidade	E9

Tabela 3. Desafios enfrentados.

Desafio	ID	Quantidade
Comunicação entre serviços	E18, E21	2
Escolha de tecnologias de comunicação	E21	1
Escalabilidade	E15, E17, E19	3
Consistência dos dados	E15	1
Gestão de falhas e resiliência	E13	1
Monitoramento e rastreamento	E3, E14, E19	3

Tabela 4. Soluções propostas.

Solução	ID	Quantidade
Comunicação entre serviços	E2, E6, E11	3
Gestão de dados distribuídos	E12	1
Gestão de falhas e resiliência	E7, E13	2
Orquestração de microsserviços	E5, E10, E15, E17, E19	5
Segurança	E16	1
Automação e DevOps	E9	1

Ao analisar a Tabela 4, é possível observar as soluções propostas para os desafios mais comuns enfrentados pelos profissionais. Entre as soluções estão a gestão de dados distribuídos e a melhoria dos aspectos de segurança.

A gestão de dados distribuídos, abordada no estudo **E12**, é descrita com o objetivo de garantir que cada microsserviço tenha acesso eficiente às suas próprias informações, sem que haja redundância ou inconsistência. Para melhorar a gestão, a prática de ”banco de dados por serviço”tem sido adotada. Nesse contexto, cada microsserviço possui seu próprio banco de dados, descentralizando o armazenamento e evitando conflitos. Essa abordagem é combinada com o uso de bancos de dados distribuídos, que são projetados para suportar grandes volumes de dados e escalabilidade horizontal, permitindo que o sistema lide com aumentos no tráfego sem prejudicar o desempenho.

A segurança em sistemas de microsserviços, abordada no estudo **E16**, é considerada um desafio relevante. O estudo destaca que a solução mais comum para enfrentar esse desafio é a implementação de autenticação e autorização baseadas em *tokens*, que permite que os microsserviços verifiquem a identidade do usuário de maneira segura e eficiente. A criptografia de dados também é fundamental para garantir a privacidade e a integridade da comunicação entre os microsserviços. Além disso, o uso de ferramentas de monitoramento e *logs* permite a detecção precoce de incidentes de segurança.

Outras soluções propostas consideram a comunicação entre serviços (**E2**, **E6** e **E11**), gestão de dados distribuídos (**E12**), gestão de falhas e resiliência (**E7** e **E13**), orquestração de microsserviços (**E5**, **E10**, **E15**, **E17** e **E19**), automação e devops (**E9**).

4.4. QP4 - Ferramentas e Tecnologias

As principais ferramentas e tecnologias utilizadas estão descritas na Tabela 5.

5. Discussão dos Resultados

A análise dos resultados obtidos na adoção da arquitetura de microsserviços revelou benefícios importantes em relação à escalabilidade, flexibilidade e eficiência no desenvolvimento de software. A utilização de ferramentas como Kubernetes [1] e Docker [2] foi essencial para alcançar maior agilidade no gerenciamento de aplicações, permitindo a distribuição dinâmica de cargas de trabalho e a implantação rápida de novos serviços.

Outro ponto relevante foi a adoção de ferramentas de monitoramento, como Pro-

[1]https://kubernetes.io/pt-br/
[2]https://www.docker.com/

Tabela 5. Ferramentas e tecnologias utilizadas.

Ferramenta/Tecnologia	Descrição	ID
Kubernetes	Automação de implantação, escala e gerenciamento de aplicativos baseados em contêineres	E15, E19, E17
Docker	Criação, implantação e execução de aplicativos em contêineres isolados e portáveis	E16, E20, E15
Prometheus, *Grafana*	Coleta de métricas e visualização de dados para análise de desempenho e monitoramento contínuo	E19
Jenkins, *GitLab* CI	Automação de testes, builds e implantações em um fluxo de desenvolvimento contínuo	E4
Helm, *ConfigMaps* (*Kubernetes*)	Gerenciamento de configurações e controle de versões para ambientes de microsserviços	E19
ELK Stack (*Elasticsearch*, *Logstash*, *Kibana*)	Ferramentas para coleta, análise e visualização de logs gerados pelos serviços	E3
MongoDB, *PostgreSQL*, *Redis*	Bancos de dados otimizados para diferentes necessidades de armazenamento e consulta	E17

metheus [3] e Grafana [4], que permitiram uma visão detalhada do desempenho do sistema. A coleta contínua de métricas possibilitou a identificação precoce de gargalos. Essas ferramentas são importantes para manter a operação estável em ambientes distribuídos e complexos, nos quais pequenos problemas podem gerar impactos significativos. A integração e entrega contínuas (CI/CD), implementadas por meio de Jenkins [5] e GitLab [6], também se destacaram na redução do tempo necessário para implementar mudanças. O pipeline automatizado eliminou etapas manuais, reduzindo a probabilidade de erros e acelerando o ciclo de desenvolvimento.

Os resultados ainda apontaram que a comunicação entre os microsserviços, quando estruturada com protocolos adequados como gRPC [7] e mensagens assíncronas via RabbitMQ [8] ou Kafka [9], garantiu a troca eficiente de informações. Esse modelo diminuiu a latência e aumentou a capacidade de processar grandes volumes de dados de maneira distribuída. No entanto, desafios relacionados à complexidade da configuração inicial foram identificados, especialmente em equipes sem experiência prévia nesse tipo de arquitetura.

Por outro lado, alguns desafios relacionados à migração de sistemas legados para microsserviços foram destacados. A transição exigiu um planejamento robusto, incluindo a escolha de padrões arquiteturais adequados e a reestruturação do código existente. A complexidade dessa migração apresentou custos iniciais elevados, mas os benefícios a longo prazo, como a facilidade de manutenção e atualização, compensaram esse investimento inicial. Por fim, os resultados também evidenciaram a importância de uma cultura organizacional alinhada ao uso dessa arquitetura. A adoção de práticas ágeis e a capacitação contínua das equipes foram essenciais para o sucesso da implementação. Essa

[3]https://www.opservices.com.br/monitoramento-prometheus/
[4]https://www.opservices.com.br/grafana/
[5]https://www.jenkins.io/
[6]https://about.gitlab.com/
[7]https://grpc.io/
[8]https://www.rabbitmq.com/
[9]https://www.redhat.com/pt-br/topics/integration/what-is-apache-kafka

integração entre tecnologia e estratégia organizacional garantiu não apenas a funcionalidade do sistema, mas também sua evolução conforme as demandas do mercado.

6. Ameaças à Validade

A condução deste estudo pode ter sido influenciada por algumas limitações metodológicas que impactam a validade dos resultados. A seguir, apresentam-se as principais ameaças identificadas.

Validade de Construção. A definição das questões de pesquisa e dos critérios de extração pode não ter capturado todos os aspectos relevantes da literatura sobre microsserviços. A seleção dos materiais foi realizada com base em títulos, resumos e palavras-chave, o que pode ter levado à exclusão de materiais relevantes que não utilizavam termos representativos considerados na string de busca. Para minimizar tal risco, decidiu-se que um material deveria ser mantido em situações de dúvida.

Validade Interna. O processo de inclusão envolveu julgamentos subjetivos. Apesar do uso de um formulário padronizado, existe o risco de viés na interpretação durante a extração e categorização dos dados. Por isso, os materiais foram mantidos em situações de dúvida.

Validade Externa. Os resultados refletem apenas o conjunto de estudos identificados e selecionados, não sendo possível garantir que todos os trabalhos relevantes disponíveis na literatura cinzenta ou em bases de dados não consultadas tenham sido contemplados. A predominância de um tipo de material também pode limitar a aplicabilidade das conclusões. Os autores reconhecem e aceitam essa ameaça, uma vez que este estudo pode ser considerado um ponto de partida para estudos secundários mais específicos, como uma Revisão Sistemática de Literatura [Mahood et al. 2014].

Validade de Conclusão. A validade de conclusão pode ter sido afetada pelo número reduzido de materiais encontrados, o que restringe a generalização dos resultados. Dessa maneira, entende-se que os resultados devem ser interpretados como um panorama atual e indicativo, mas não definitivo, sobre os avanços e tendências, desafios, soluções e tecnologias relacionadas à arquitetura de microsserviços.

7. Considerações Finais

Este trabalho apresentou uma revisão multivocal da literatura sobre arquiteturas de microsserviços, abordando as evoluções dos estudos, desafios, soluções e tecnologias associadas. As principais tendências identificadas estão relacionadas a práticas de desenvolvimento, integração contínua, flexibilidade e escalabilidade, com destaque para o uso crescente de contêineres, orquestradores e ferramentas de monitoramento. Embora avanços relevantes tenham sido observados em áreas como gerenciamento e segurança, persistem desafios importantes, como a necessidade de padronização e a complexidade na migração de sistemas monolíticos. Entre os benefícios, ressalta-se o impacto positivo no desenvolvimento ágil e na escalabilidade, além da consolidação de boas práticas que orientam profissionais e pesquisadores.

A revisão também apontou limitações e oportunidades de pesquisa, como a escassez de estudos empíricos em larga escala e a necessidade de abordagens interdisciplinares que integrem microsserviços a áreas emergentes, como inteligência artificial, internet das

coisas e computação em borda. Sugere-se, assim, a realização de investigações sobre ferramentas automatizadas de monitoramento, metodologias que facilitem a transição de arquiteturas monolíticas e análises de custo-benefício da adoção de microsserviços em projetos complexos, oferecendo um panorama útil tanto para o meio acadêmico quanto para a prática profissional. Existe ainda a possibilidade de realizar estudos secundários mais específicos sobre a temática de microsserviços, tais como as tendências e os desafios enfrentados.

Referências

Amazon (2024). O que são microsserviços? `https://aws.amazon.com/pt/microservices/`.

Bonfiglioli, C. P. (2021). Desenvolvendo aplicações utilizando a arquitetura de microsserviços. Acesso em: 2024-12-06.

Bushong, V., Abdl-Fattah, A. S., Maruf, A. A., Das, D., Lehman, A., Jaroszewski, E., Coffey, M., Cerny, T., Frajtak, K., Tisnovsky, P., and Bures, M. (2021). On microservice analysis and architecture evolution: A systematic mapping study. *MDPI Open Access Journals*, page 27.

Docker (2024). What is docker. `https://www.docker.com/resources/what-container`.

Fowler, M. (2023). A comprehensive study of the transition from monolithic to microservices-based software architectures. Acesso em: 20 dez. 2024.

Garousi, V., Felderer, M., and Mäntylä, M. V. (2019). Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Information and software technology*, 106:101–121.

Kubernets (2024). Kubernetes documentation. `https://kubernetes.io/docs/home/`.

Lima, G. P. D. S. and Fontoura, J. R. D. A. (2023). Adesão da arquitetura de microsserviços nas grandes corporações para desenvolvimento ou migração de aplicações. Acesso em: 2024-12-07.

Mahood, Q., Van Eerd, D., and Irvin, E. (2014). Searching for grey literature for systematic reviews: challenges and benefits. *Research synthesis methods*, 5(3):221–234.

Maxime, A. (2023). Automated database schema evolution in microservices. *Conference: 49th International Conference on Very Large Data Bases (VLDB 2023) PhD Workshop*, page 34.

Pressman, R. S. and Maxim, B. R. (2021). *Engenharia de software-9*. McGraw Hill Brasil.

Samad, A. (2021). Architectural transition: Unveiling the shift from monolithic to microservices in digital experience platforms. *Massachusetts Institute of Technology*, page 17.

Society, I. C. (2018). What is software architecture in software engineering?

Walker, A., Das, D., and Cerny, T. (2020). Automated code-smell detection in microservices through static analysis: A case study. *Applied Sciences*, 10:7800.