LUMEN ET VIRTUS
ISSN: 2177-2789

# IMPACTOS DA ARQUITETURA DE MICROSSERVIÇOS NA MANUTENÇÃO DE SISTEMAS CORPORATIVOS

https://doi.org/10.56238/levv14n32-006

**Data de submissão:** 10/12/2023 **Data de publicação:** 20/01/2024

**Rodrigo Monteiro Guedes de Almeida**

## RESUMO

O presente artigo analisa os impactos da arquitetura de microsserviços na manutenção de sistemas corporativos, contrastando suas particularidades com os modelos monolíticos tradicionais sob a ótica da engenharia de software e da sustentabilidade técnica. A pesquisa foi conduzida com abordagem qualitativa, fundamentada em revisão bibliográfica sistematizada com recorte temporal de 2015 a 2024, utilizando descritores específicos e categorização temática conforme orientações de Gil e Lakatos & Marconi. Os resultados obtidos apontam que a arquitetura de microsserviços oferece vantagens significativas no que se refere à modularização, escalabilidade, autonomia das equipes e redução de riscos durante o processo de manutenção, permitindo correções localizadas, versionamento independente e automação de testes e deploys, além de favorecer o monitoramento detalhado dos serviços. Em contrapartida, foram observados desafios relevantes, como a complexidade na orquestração dos microsserviços, a necessidade de padronização das interfaces de comunicação e o aumento da carga cognitiva das equipes quando a governança técnica não é bem estabelecida. A comparação com a arquitetura monolítica demonstrou que esta, apesar de sua simplicidade inicial e menor exigência técnica, tende a acumular rigidez estrutural e acoplamento excessivo com o tempo, o que dificulta a manutenção corretiva, a escalabilidade e a adoção de novas tecnologias. Conclui-se que a adoção da arquitetura de microsserviços deve ser orientada por critérios técnicos, operacionais e organizacionais bem definidos, sendo especialmente recomendada para sistemas de grande porte, com alta complexidade e necessidade de evolução contínua, onde a manutenibilidade representa um fator crítico para a sustentabilidade da aplicação.

**Palavras-chave:** Arquitetura de microsserviços. Manutenção de sistemas. Sistemas corporativos. Escalabilidade. Monolítico.

## 1 INTRODUÇÃO

A transformação digital das organizações, especialmente aquelas que dependem de sistemas complexos de informação, tem exigido estruturas arquiteturais cada vez mais adaptáveis e escaláveis, e nesse contexto, a arquitetura de microsserviços tem se consolidado como uma alternativa promissora à tradicional arquitetura monolítica, por permitir maior flexibilidade na manutenção e evolução dos sistemas corporativos, o que atende às demandas de negócios dinâmicos e cada vez mais dependentes de tecnologia (Fowler e Lewis, 2014).

Ao contrário do modelo monolítico, que concentra toda a lógica da aplicação em um único bloco funcional, os microsserviços são compostos por unidades menores e independentes que se comunicam entre si para compor uma aplicação robusta, o que impacta diretamente na capacidade de manutenção, já que alterações pontuais podem ser realizadas em serviços específicos sem comprometer o sistema como um todo (Richardson, 2016).

Essa independência estrutural dos microsserviços favorece práticas de desenvolvimento mais ágeis, aumenta a autonomia das equipes técnicas e reduz a complexidade na detecção e correção de falhas localizadas, contribuindo para a manutenção contínua e para a melhoria progressiva dos sistemas implantados nas corporações (Newman, 2015).

No entanto, essa flexibilidade exige atenção redobrada quanto à padronização de interfaces, sincronização entre serviços e monitoramento de desempenho, pois a descentralização de responsabilidades pode gerar inconsistências e dificuldades operacionais quando não há uma governança clara sobre o ecossistema de serviços (Sommerville, 2011).

A literatura especializada destaca que a arquitetura de microsserviços melhora a manutenibilidade dos sistemas ao permitir que cada componente evolua de forma isolada, o que é fundamental para empresas que buscam responder rapidamente às mudanças do mercado sem reescrever grandes blocos de código (Jazayeri, 2000).

Essa capacidade de adaptação é um dos fatores que torna o modelo atrativo para projetos de longa duração, nos quais a manutenção preditiva e a escalabilidade progressiva são mais eficazes do que intervenções completas ou reestruturações profundas exigidas por arquiteturas centralizadas (Bogner e Zimmermann, 2016).

Ao mesmo tempo, o modelo traz desafios importantes em termos de infraestrutura, já que a comunicação entre serviços autônomos exige mecanismos robustos de orquestração, mensageria e tolerância a falhas, elementos que influenciam diretamente os custos e a complexidade da manutenção do ambiente como um todo (Erl, 2007).

Em contrapartida, arquiteturas monolíticas, ainda comuns em sistemas legados, favorecem ciclos de desenvolvimento mais lineares e estruturas de manutenção mais simples em projetos de

pequeno porte, embora percam em escalabilidade e flexibilidade na medida em que a aplicação cresce e se diversifica (Kanizawa e Pinto, 2022).

Diante disso, a escolha da arquitetura adequada não pode ser feita unicamente com base em tendências tecnológicas, mas sim a partir de uma avaliação criteriosa dos objetivos de negócio, do perfil técnico da equipe, dos recursos disponíveis e do horizonte de evolução esperado para o sistema (Fowler, 2017).

Estudos de caso com empresas como Amazon, Spotify e Netflix ilustram que a adoção bem-sucedida da arquitetura de microsserviços exige maturidade organizacional e domínio técnico sobre as ferramentas envolvidas, bem como um compromisso com a manutenção constante da infraestrutura distribuída (Tripoli, 2017).

A migração de sistemas monolíticos para microsserviços, por sua vez, não deve ser encarada como uma solução imediata para problemas de desempenho ou escalabilidade, mas como uma reestruturação arquitetural estratégica que demanda planejamento, testes progressivos e um entendimento profundo dos pontos de integração entre os módulos (Soldani et al., 2018).

Do ponto de vista gerencial, a arquitetura distribuída impõe novos desafios à manutenção, pois exige mecanismos de monitoramento contínuo, políticas de versionamento de APIs e uma documentação rigorosa para que a evolução dos serviços não comprometa o funcionamento do sistema como um todo (Francesco, Malavolta e Lago, 2017).

A experiência prática demonstrou que a granularidade dos microsserviços deve ser cuidadosamente definida, pois serviços excessivamente fragmentados tendem a gerar sobrecarga de comunicação, enquanto serviços muito amplos podem comprometer os benefícios de independência e escalabilidade (Lenarduzzi e Sievi-Korte, 2018).

Portanto, compreender os impactos da arquitetura de microsserviços na manutenção de sistemas corporativos exige não apenas o domínio de aspectos técnicos, mas também o entendimento das implicações organizacionais e operacionais que essa escolha arquitetural impõe ao longo do ciclo de vida dos sistemas (Yale Yu, Silveira e Sundaram, 2016).

Este artigo tem como objetivo analisar, com base em literatura científica e estudos de caso, como a adoção da arquitetura de microsserviços afeta os processos de manutenção de sistemas corporativos, considerando os ganhos e desafios associados à sua aplicação prática em ambientes de produção.

## 2 REFERENCIAL TEÓRICO

### 2.1 FUNDAMENTOS DA ARQUITETURA DE MICROSSERVIÇOS

A arquitetura de microsserviços define um estilo arquitetural descentralizado em que a aplicação é composta por diversos módulos independentes, cada um responsável por uma

funcionalidade específica e desenvolvido de maneira autônoma, o que permite a construção de sistemas distribuídos mais flexíveis e escaláveis em comparação aos modelos monolíticos tradicionais, nos quais todas as funcionalidades estão encapsuladas em uma única estrutura indivisível, dificultando manutenções localizadas e aumentando o risco de falhas sistêmicas sempre que uma modificação é necessária em partes isoladas do código (Newman, 2015).

Ao segmentar a aplicação em partes menores e coesas, os microsserviços promovem uma independência funcional entre os componentes do sistema, o que permite que equipes distintas possam desenvolver, testar, implantar e manter cada serviço sem comprometer os demais módulos da aplicação, oferecendo assim uma alternativa organizacional mais eficaz para ambientes complexos e de alta demanda por mudanças, onde o tempo de resposta é um fator crítico e a estabilidade precisa ser garantida mesmo em situações de atualizações contínuas e simultâneas (Fowler e Lewis, 2014).

A capacidade de isolar responsabilidades em serviços autônomos gera impactos significativos nos processos de manutenção, pois reduz o acoplamento entre os módulos e minimiza os efeitos colaterais das alterações de código, permitindo intervenções mais rápidas, com menor risco de quebra da integridade funcional do sistema, o que torna a arquitetura mais resistente a falhas, facilita a atualização de tecnologias em pontos específicos da aplicação e contribui para a melhoria contínua com base em ciclos de entrega incrementais (Richardson, 2016).

Esse modelo arquitetural também se mostra vantajoso ao considerar as exigências atuais de escalabilidade horizontal, já que cada microsserviço pode ser replicado conforme a demanda específica de sua funcionalidade, otimizando o uso de recursos computacionais e evitando o desperdício de processamento comum em sistemas monolíticos, onde a necessidade de escalar uma funcionalidade implica em escalar toda a aplicação, mesmo que os demais módulos não estejam sobrecarregados, o que representa uma ineficiência estrutural relevante nos contextos modernos de computação em nuvem (Bogner e Zimmermann, 2016).

O desenvolvimento baseado em microsserviços também estimula a adoção de pipelines de entrega contínua e de práticas DevOps, pois a modularização do sistema permite que as equipes configurem integrações, testes e implantações automáticas para cada serviço de forma independente, resultando em ciclos mais curtos de desenvolvimento e maior agilidade para responder às mudanças de requisitos, embora essa autonomia exija maturidade em práticas de versionamento, monitoramento e controle de dependências para evitar falhas emergentes por incompatibilidades entre serviços intercomunicantes (Sommerville, 2011).

Embora as vantagens operacionais da arquitetura de microsserviços sejam amplamente reconhecidas, sua implementação exige um planejamento rigoroso e uma governança arquitetural sólida, visto que a descentralização dos componentes pode gerar uma proliferação descontrolada de tecnologias, frameworks e ferramentas se não houver diretrizes bem definidas, o que impacta

negativamente na coesão técnica do projeto e na capacidade de realizar manutenções corretivas e evolutivas de forma uniforme e previsível ao longo do tempo (The Open Group, 2018).

A manutenção da consistência transacional em sistemas de microsserviços é um dos principais desafios enfrentados pelas equipes de desenvolvimento, já que a fragmentação da lógica de negócio entre serviços exige a implementação de estratégias específicas, como sagas e mensagerias assíncronas, para garantir que o sistema mantenha seu comportamento correto mesmo diante de falhas parciais, o que demanda não apenas domínio técnico aprofundado, mas também o alinhamento entre arquitetura de software e os processos de negócio que o sistema suporta (Jazayeri, 2000).

A adoção desse modelo arquitetural impõe ainda a necessidade de ferramentas de observabilidade mais sofisticadas, como rastreamento distribuído, centralização de logs, dashboards de métricas e alertas customizados, pois, sem uma visão clara e contínua do comportamento de cada serviço, as falhas se tornam difíceis de identificar e diagnosticar, tornando a manutenção reativa mais demorada e potencialmente dispendiosa em termos de impacto operacional e desgaste da equipe técnica responsável pela sustentação do ambiente (Abreu, Carvalho e Rocha, 2018).

A comunicação entre os microsserviços, normalmente realizada por meio de APIs REST, filas de mensagens ou protocolos leves como gRPC, também exige atenção especial na fase de manutenção, pois qualquer alteração em um contrato de interface pode impactar outros serviços dependentes, tornando fundamental a adoção de práticas como controle de versionamento de APIs, testes de regressão integrados e documentação acessível e atualizada, elementos que muitas vezes são negligenciados em projetos ágeis e que resultam em dificuldades crescentes à medida que o sistema evolui (Erl, 2007).

Em projetos que envolvem sistemas legados baseados em arquitetura monolítica, a transição para microsserviços pode ser feita de forma progressiva por meio da técnica conhecida como “estrangulamento de aplicação”, na qual os serviços mais críticos ou sujeitos a alterações frequentes são extraídos da base monolítica e reconstruídos como microsserviços, o que permite ganhos graduais de flexibilidade e manutenibilidade, além de reduzir os riscos associados à substituição completa de sistemas que já estão em operação e com estabilidade comprovada (Kanizawa e Pinto, 2022).

A experiência de grandes empresas como Amazon, Netflix e Spotify revela que os benefícios da arquitetura de microsserviços são potencializados quando há um alinhamento entre a estratégia de negócio, a cultura organizacional e o modelo de desenvolvimento adotado, especialmente no que diz respeito à autonomia das equipes, ao uso intensivo de automação e à capacidade de adaptar rapidamente os sistemas aos feedbacks do mercado, elementos que transformam a arquitetura em um diferencial competitivo em ambientes de alta pressão e inovação constante (Tripoli, 2017).

Mesmo com tantos ganhos em termos de escalabilidade, resiliência e agilidade, a complexidade gerada pelo alto número de serviços e interdependências pode ser um obstáculo à manutenção

eficiente, sobretudo quando o projeto cresce sem que haja uma arquitetura de referência clara, padrões de codificação unificados e repositórios compartilhados que facilitem a colaboração entre as equipes, fatores que precisam ser contemplados desde o início do projeto para evitar um cenário de desorganização progressiva e aumento do custo de manutenção (Francesco, Malavolta e Lago, 2017).

A definição adequada da granularidade dos serviços é outro fator crítico para o sucesso da arquitetura de microsserviços, pois fragmentações excessivas podem levar a sistemas hiperdependentes, com grande volume de chamadas entre serviços e sobrecarga nas redes de comunicação, enquanto serviços com escopo muito amplo acabam se assemelhando a módulos monolíticos e perdem os benefícios de independência e flexibilidade, dificultando a manutenção e a evolução do sistema de forma eficiente e sustentável (Lenarduzzi e Sievi-Korte, 2018).

A escolha da arquitetura mais adequada para um sistema corporativo deve considerar não apenas os aspectos técnicos e operacionais, mas também a capacidade da organização de sustentar os investimentos em capacitação, infraestrutura e governança exigidos pelo modelo de microsserviços, pois sem uma estrutura sólida e processos bem definidos, os ganhos esperados podem ser neutralizados por dificuldades operacionais, retrabalhos e aumento não planejado dos custos de manutenção, tornando o projeto insustentável no médio prazo (Yale Yu, Silveira e Sundaram, 2016).

A compreensão profunda dos fundamentos da arquitetura de microsserviços, suas premissas, suas implicações técnicas e suas interações com os modelos organizacionais é indispensável para garantir que sua aplicação resulte em sistemas mais eficientes, adaptáveis e fáceis de manter, o que só é possível quando o projeto é conduzido com base em evidências, boas práticas consolidadas e decisões arquiteturais compatíveis com os objetivos estratégicos da organização e com a realidade técnica da equipe envolvida no desenvolvimento e manutenção dos sistemas (Richardson, 2016).

## 2.2 VANTAGENS E DESAFIOS NA MANUTENÇÃO DE MICROSSERVIÇOS

A segmentação de sistemas em microsserviços possibilita que cada módulo opere de forma independente, o que reduz significativamente o impacto de falhas durante a manutenção, uma vez que é possível corrigir problemas, atualizar funcionalidades ou realizar testes em um único serviço sem a necessidade de reimplantar toda a aplicação, esse isolamento funcional traz mais segurança às atualizações e contribui para o aumento da disponibilidade dos sistemas, aspecto essencial em ambientes corporativos onde a continuidade operacional é uma prioridade (Richardson, 2016).

A modularidade característica dessa arquitetura favorece ainda a manutenção preventiva, pois permite que os serviços sejam monitorados individualmente com métricas específicas de desempenho, consumo de recursos e taxa de erros, o que torna possível a identificação antecipada de comportamentos anômalos e a aplicação de correções antes que causem impactos significativos na operação geral, essa visibilidade mais granular dos serviços é um diferencial importante quando

comparada às arquiteturas monolíticas, onde uma falha pode comprometer todo o sistema e a origem do problema costuma ser mais difícil de localizar (Bogner e Zimmermann, 2016).

Além do monitoramento mais eficaz, a arquitetura de microsserviços simplifica o versionamento de funcionalidades e a reversão de mudanças problemáticas, pois cada serviço pode evoluir em seu próprio ritmo e manter múltiplas versões disponíveis enquanto as aplicações cliente vão se adaptando, esse tipo de controle é particularmente útil em ambientes de produção, onde a necessidade de reverter rapidamente uma atualização mal-sucedida pode evitar prejuízos financeiros, interrupções de serviço e perda de dados sensíveis, fatores que pressionam fortemente as equipes de manutenção (Fowler, 2017).

A separação de responsabilidades entre os serviços também melhora a organização do código e da lógica de negócio, o que facilita a leitura, compreensão e modificação das funcionalidades implementadas, esse aspecto é especialmente relevante em projetos com equipes grandes ou rotatividade de profissionais, pois reduz a curva de aprendizado de novos desenvolvedores e diminui a dependência de conhecimento tácito, tornando o processo de manutenção mais ágil, seguro e sustentável a longo prazo (Newman, 2015).

A possibilidade de usar diferentes tecnologias, linguagens de programação e frameworks em cada microsserviço traz benefícios para a manutenção quando há necessidade de modernização gradual da pilha tecnológica, pois permite que serviços antigos sejam reescritos com tecnologias mais modernas ou que problemas específicos sejam resolvidos com ferramentas mais adequadas ao seu contexto, ao mesmo tempo, essa flexibilidade exige atenção redobrada à padronização de práticas de integração e interoperabilidade, pois a diversidade tecnológica pode dificultar a manutenção se não for acompanhada de diretrizes arquiteturais claras (Jazayeri, 2000).

Apesar das vantagens, a gestão das dependências entre serviços é uma das dificuldades mais enfrentadas nas manutenções, principalmente quando a aplicação cresce em número de microsserviços e a comunicação entre eles se torna mais densa, nessas situações, qualquer alteração em um serviço central pode exigir ajustes em uma cadeia de dependentes, o que pode tornar a manutenção complexa, lenta e suscetível a falhas silenciosas se não houver testes de integração automatizados e uma boa documentação dos contratos entre os serviços (Abreu, Carvalho e Rocha, 2018).

Algo que impacta diretamente a manutenção é a falta de uma arquitetura de observabilidade bem estruturada, pois a distribuição dos componentes em microsserviços torna mais difícil identificar os fluxos de execução, os pontos de falha e os gargalos de desempenho, a ausência de soluções como logs centralizados, rastreamento distribuído e análise de métricas detalhadas compromete o diagnóstico e a resolução de problemas em ambientes distribuídos, gerando frustração nas equipes de suporte e aumentando o tempo de indisponibilidade do sistema (Francesco, Malavolta e Lago, 2017).

A descentralização da base de dados, com cada serviço podendo ter seu próprio repositório, melhora o isolamento das informações e evita acoplamentos indesejados, mas introduz complexidade adicional na manutenção quando há necessidade de consultas interserviços ou atualizações sincronizadas, esse cenário exige a adoção de padrões como eventual consistency e o uso de orquestração ou coreografia de processos para manter a integridade das transações, o que aumenta o esforço da equipe e a necessidade de conhecimentos especializados para garantir a consistência dos dados (Erl, 2007).

As manutenções também podem se tornar mais custosas em termos de tempo e recursos quando o projeto não possui uma padronização mínima no uso de frameworks, bibliotecas e protocolos de comunicação, a autonomia excessiva das equipes pode levar a um ecossistema heterogêneo de serviços com estruturas distintas, o que compromete a manutenibilidade, dificulta a automação de testes e requer esforços maiores para capacitar novos profissionais ou realizar manutenções cruzadas entre módulos de diferentes origens (Sommerville, 2011).

O uso de containers e orquestradores como Docker e Kubernetes facilita a manutenção da infraestrutura necessária para os microsserviços, permitindo escalonamento automático, reinício de serviços em caso de falhas e atualização gradual de componentes em produção, mas ao mesmo tempo, exige conhecimento técnico avançado das equipes de suporte e operações, pois o gerenciamento eficiente desses recursos envolve a configuração correta de redes, volumes, secrets, health checks e monitoramento contínuo do estado dos pods e nós da aplicação (Fowler e Lewis, 2014).

Em projetos que envolvem muitos microsserviços, a gestão de configuração e a manutenção de arquivos de ambientes tornam-se tarefas críticas, pois mudanças em variáveis de ambiente, segredos ou conexões com outros serviços precisam ser cuidadosamente controladas e replicadas entre os ambientes de desenvolvimento, homologação e produção, a ausência de ferramentas como vaults, CI/CD pipelines e gerenciadores de configuração pode tornar essas tarefas suscetíveis a erros humanos e atrasos operacionais (Kanizawa e Pinto, 2022).

A fragmentação da lógica de negócio entre múltiplos serviços pode dificultar a compreensão do sistema como um todo durante a manutenção, principalmente em cenários de incidentes ou quando é necessário rastrear o impacto de uma mudança em uma funcionalidade mais ampla, isso evidencia a importância de uma boa documentação arquitetural, de mapas de dependência e de ferramentas que ajudem a visualizar o fluxo entre os serviços, o que nem sempre é priorizado nas fases iniciais do projeto e acaba prejudicando a manutenibilidade à medida que a aplicação cresce (Lenarduzzi e Sievi-Korte, 2018).

Em sistemas distribuídos, a confiabilidade de cada microsserviço individual afeta diretamente a estabilidade da aplicação como um todo, o que exige testes automatizados robustos e um bom sistema de deploy com rollback automático para mitigar impactos causados por falhas em atualizações, além

disso, a manutenção corretiva exige que os logs e erros sejam reportados com clareza, timestamps sincronizados e correlação entre requisições, aspectos que nem sempre são bem implementados e que tornam o processo de correção mais demorado e impreciso (Yale Yu, Silveira e Sundaram, 2016).

Empresas que adotam microsserviços sem uma estratégia de governança adequada acabam enfrentando problemas recorrentes de retrabalho, aumento do tempo médio de correção de falhas e dificuldades em manter o ritmo de evolução dos sistemas, isso reforça a necessidade de políticas claras para definição de responsabilidades entre as equipes, padronização de práticas de desenvolvimento e manutenção, e auditorias técnicas periódicas que garantam a aderência aos princípios arquiteturais e evitem a degradação progressiva da qualidade do software (The Open Group, 2018).

A arquitetura de microsserviços oferece vantagens inegáveis para a manutenção de sistemas complexos e dinâmicos, mas essas vantagens não são automáticas nem gratuitas, elas dependem da maturidade técnica da equipe, da clareza das decisões arquiteturais e da disciplina na adoção de boas práticas que garantam a previsibilidade, a rastreabilidade e a eficiência das manutenções ao longo do tempo, sendo essencial compreender que a complexidade é realocada e não eliminada, exigindo mais planejamento, documentação e ferramentas para que o modelo se mantenha funcional e sustentável (Newman, 2015).

## 2.3 COMPARAÇÕES ENTRE MICROSSERVIÇOS E ARQUITETURA MONOLÍTICA NA MANUTENÇÃO DE SISTEMAS

A arquitetura monolítica representa uma abordagem tradicional em que todos os componentes da aplicação são implementados como uma única unidade indivisível, o que inicialmente facilita o desenvolvimento e a implantação do sistema, especialmente em projetos de pequeno porte ou com funcionalidades bem definidas, essa centralização, no entanto, impõe grandes desafios conforme o sistema cresce, pois qualquer manutenção exige a compreensão de toda a estrutura e, frequentemente, a recompilação e redistribuição de todo o software, o que gera lentidão e risco de introdução de novos erros ao se alterar partes do código (Richardson, 2016).

Microsserviços, por sua vez, permitem que a manutenção seja feita em fragmentos menores, com ciclos independentes de atualização e uma infraestrutura preparada para lidar com deploys contínuos e rollbacks localizados, o que proporciona uma agilidade muito maior na correção de bugs e na inclusão de melhorias pontuais, diferentemente do modelo monolítico, que exige um planejamento mais rígido para atualizações e tende a provocar mais tempo de inatividade ou dependência de janelas de manutenção programadas, dificultando a resposta rápida a incidentes em produção (Fowler, 2017).

Na manutenção preventiva, os microsserviços possibilitam ações mais proativas, já que cada serviço pode ser monitorado isoladamente por métricas específicas, o que facilita a detecção de falhas iminentes e a execução de correções antes que causem prejuízos operacionais, enquanto na arquitetura

monolítica, os indicadores de desempenho normalmente refletem o sistema como um todo, tornando mais difícil identificar exatamente onde o problema está ocorrendo, o que impacta negativamente na capacidade de resposta da equipe de sustentação (Francesco, Malavolta e Lago, 2017).

Quando se trata de manutenção corretiva, a vantagem dos microsserviços está na possibilidade de corrigir uma falha crítica sem interferir nos demais serviços, o que é impraticável em sistemas monolíticos, onde mesmo um pequeno erro pode exigir o recompilamento completo da aplicação, além disso, a interdependência entre módulos monolíticos faz com que mudanças simples possam provocar efeitos colaterais imprevisíveis, já que não há uma separação clara entre os domínios funcionais da aplicação, dificultando a testabilidade e elevando os riscos de regressão (Newman, 2015).

Na perspectiva da manutenibilidade a longo prazo, a modularização dos microsserviços favorece a evolução contínua do sistema, já que funcionalidades antigas podem ser reescritas gradualmente sem a necessidade de parar toda a aplicação, ao contrário do modelo monolítico, no qual reestruturar um único módulo frequentemente exige um retrabalho em cadeia e aumenta o custo de manutenção, pois as dependências internas tendem a crescer com o tempo, tornando o sistema mais rígido e frágil diante de alterações necessárias (Bogner e Zimmermann, 2016).

Além disso, outra diferença está na documentação e na transferência de conhecimento, já que sistemas monolíticos frequentemente exigem uma curva de aprendizado elevada, com arquivos longos, múltiplas camadas de lógica acopladas e estruturas complexas que dificultam a entrada de novos desenvolvedores, enquanto a arquitetura de microsserviços permite que cada serviço tenha sua própria documentação, facilitando o onboarding e possibilitando que equipes menores se responsabilizem por partes específicas da aplicação, o que é benéfico para a manutenção e continuidade do projeto (Kanizawa e Pinto, 2022).

O impacto das alterações em uma aplicação monolítica costuma ser maior e mais imprevisível, visto que qualquer mudança, por menor que seja, pode comprometer funcionalidades que não estavam relacionadas à área modificada, já nos microsserviços, os testes são mais direcionados e a responsabilidade está claramente delimitada, o que reduz o tempo necessário para realizar validações, aumentando a confiança na estabilidade do sistema mesmo após alterações de código, desde que os contratos entre os serviços estejam bem definidos e a comunicação entre eles seja robusta (Abreu, Carvalho e Rocha, 2018).

Apesar de suas limitações, a arquitetura monolítica pode oferecer vantagens em termos de manutenção em projetos de baixa complexidade, onde os requisitos são bem definidos e a equipe é pequena, pois a simplicidade da estrutura permite maior controle do escopo e menor necessidade de infraestrutura especializada, já os microsserviços exigem uma maturidade técnica maior da equipe, além de ferramentas adequadas para orquestração, controle de versões, deploy contínuo,

monitoramento e testes automatizados, o que pode representar um obstáculo inicial em projetos com poucos recursos (Sommerville, 2011).

Em projetos legados, a manutenção em sistemas monolíticos é um dos principais gargalos enfrentados pelas empresas, pois frequentemente esses sistemas acumulam anos de alterações não documentadas, regras de negócio espalhadas por diferentes camadas e uma série de dependências invisíveis, o que torna qualquer alteração arriscada e difícil de validar, migrar esses sistemas para microsserviços, por outro lado, exige planejamento detalhado, divisão estratégica dos domínios e definição clara das fronteiras entre os serviços, o que pode ser feito por meio de técnicas como o estrangulamento de aplicação (Lenarduzzi e Sievi-Korte, 2018).

A manutenção evolutiva é beneficiada com a arquitetura de microsserviços pela possibilidade de reescrever módulos em novas tecnologias sem precisar comprometer todo o sistema, o que é inviável em modelos monolíticos, onde a atualização da linguagem ou framework principal exige a adaptação do código inteiro e, muitas vezes, a reconstrução de toda a base da aplicação, esse fator representa uma grande vantagem competitiva em setores que exigem inovação constante e que não podem se dar ao luxo de manter sistemas rígidos ou desatualizados (Tripoli, 2017).

Quando analisado sob a ótica da escalabilidade da manutenção, os microsserviços permitem que diferentes equipes trabalhem de forma paralela em diferentes partes da aplicação, realizando correções e melhorias simultaneamente, o que reduz gargalos e aumenta a velocidade da equipe como um todo, já em arquiteturas monolíticas, o trabalho concorrente é limitado pelas dependências internas, o que obriga uma maior coordenação e aumenta o risco de conflitos entre alterações feitas por diferentes profissionais (Fowler e Lewis, 2014).

A complexidade de manutenção de sistemas baseados em microsserviços se manifesta principalmente na orquestração e na interconectividade entre os serviços, pois qualquer erro de configuração, falha na comunicação ou inconsistência nos contratos pode comprometer o funcionamento do sistema como um todo, essa complexidade, no entanto, pode ser mitigada com a utilização de ferramentas modernas de observabilidade, testes automatizados e arquitetura orientada a eventos, elementos que nem sempre são possíveis de implantar com a mesma eficácia em sistemas monolíticos (Erl, 2007).

O tempo de inatividade necessário para manutenção em sistemas monolíticos costuma ser maior, pois qualquer alteração exige uma nova compilação completa da aplicação e, muitas vezes, o reinício do servidor, enquanto os microsserviços permitem atualização contínua de módulos específicos sem afetar os demais, essa vantagem se traduz em menor interrupção das atividades, maior disponibilidade da aplicação e uma experiência mais estável para os usuários finais, especialmente em ambientes corporativos que exigem alta disponibilidade (Yale Yu, Silveira e Sundaram, 2016).

A rastreabilidade de erros e o diagnóstico de problemas em sistemas monolíticos são mais desafiadores, pois os logs tendem a estar concentrados e misturados entre diversas funcionalidades, dificultando a identificação da origem dos problemas, nos microsserviços, o uso de ferramentas como rastreamento distribuído e centralização de logs facilita a identificação de falhas específicas e acelera o processo de correção, desde que haja uma infraestrutura bem planejada e um cuidado rigoroso com a correlação de eventos e mensagens entre os serviços (Francesco, Malavolta e Lago, 2017).

Ambas as arquiteturas apresentam benefícios e limitações, mas é evidente que, no aspecto da manutenção de sistemas corporativos de médio e grande porte, os microsserviços oferecem uma estrutura mais favorável à evolução contínua, à agilidade na resposta a falhas e à adoção de práticas modernas de desenvolvimento, enquanto os sistemas monolíticos tendem a se mostrar mais simples apenas em contextos muito específicos, sendo cada vez mais substituídos por modelos distribuídos que conseguem acompanhar a velocidade das transformações tecnológicas e das exigências do mercado (Newman, 2015).

## 3 METODOLOGIA

Foi realizada uma revisão bibliográfica sistematizada, com o intuito de reunir, categorizar e interpretar as contribuições acadêmicas e técnicas mais relevantes sobre o tema, respeitando critérios metodológicos rigorosos que garantissem a fidedignidade das fontes e a relevância do conteúdo analisado, especialmente no que se refere à comparação entre arquiteturas monolíticas e de microsserviços sob a ótica da manutenibilidade, fator central para empresas que buscam escalabilidade e agilidade sem comprometer a confiabilidade dos sistemas em operação (Lakatos e Marconi, 2003).

A seleção do material bibliográfico foi realizada a partir de bases de dados reconhecidas na área da Ciência da Computação, como IEEE Xplore, Scielo, Google Scholar, ResearchGate, priorizando artigos publicados entre 2015 e 2024, com especial ênfase nos últimos cinco anos, período em que o debate sobre a adoção de microsserviços se intensificou no meio acadêmico e nas organizações de tecnologia, refletindo um aumento significativo de interesse em entender as reais vantagens, limitações e desafios desse modelo frente às exigências crescentes do mercado (Bogner e Zimmermann, 2016).

Foram utilizados descritores combinados como “manutenção de sistemas”, “microsserviços”, “arquitetura de software”, “sistemas corporativos”, “monolítico vs microsserviços” e “evolução arquitetural”.

Após a coleta inicial, os materiais foram submetidos a critérios de inclusão que consideraram a clareza na descrição metodológica, a pertinência temática, a atualidade do conteúdo e a presença de discussões diretamente ligadas à manutenção de sistemas distribuídos, sendo excluídos artigos com caráter meramente introdutório, sem aplicação prática ou que não apresentassem dados concretos sobre

os efeitos da arquitetura de microsserviços na prática da manutenção, o que assegurou uma maior densidade analítica ao conjunto de fontes selecionadas.

## 4 RESULTADOS E DISCUSSÃO

A análise dos artigos selecionados revelou um consenso significativo na literatura quanto à superioridade da arquitetura de microsserviços em relação à manutenibilidade, especialmente em contextos de sistemas corporativos que demandam atualização contínua, integração frequente com novas tecnologias e alta disponibilidade, os autores indicam que a modularização do sistema em componentes independentes permite uma atuação mais ágil das equipes técnicas na resolução de falhas, implantação de melhorias e realização de testes direcionados, o que se traduz em menor tempo de resposta, maior previsibilidade nos processos de manutenção e redução de riscos operacionais, fatores altamente valorizados em ambientes empresariais com operações críticas e infraestrutura distribuída (Richardson, 2016).

Ao comparar sistemas construídos em modelos monolíticos com os baseados em microsserviços, observou-se que a arquitetura tradicional centraliza as dependências internas, o que gera uma rigidez estrutural que compromete a escalabilidade e eleva os custos de manutenção ao longo do tempo, nas experiências relatadas por Mendes (2021), foi possível constatar que a simples modificação de uma funcionalidade em uma aplicação monolítica frequentemente exigia recompilação e redistribuição de toda a aplicação, o que torna o processo moroso, suscetível a regressões e dependente de conhecimento específico sobre a estrutura global do sistema, dificultando o trabalho de equipes novas ou descentralizadas e comprometendo a sustentabilidade da solução a médio prazo (Mendes, 2021).

Em contrapartida, Oliveira (2023) destaca que a estruturação da aplicação por microsserviços permitiu que modificações fossem realizadas isoladamente e com maior controle de qualidade, pois a responsabilidade de cada serviço estava claramente delimitada, facilitando a realização de testes, o versionamento independente e o controle de regressões, essa modularização também favoreceu o desenvolvimento de automações específicas para cada serviço, integradas a pipelines de entrega contínua, o que possibilitou deploys mais rápidos, seguros e menos suscetíveis a falhas generalizadas, gerando um ambiente de manutenção mais robusto e escalável (Oliveira, 2023).

A partir dos estudos de caso apresentados, verificou-se que a manutenção em microsserviços é favorecida por práticas consolidadas de monitoramento, como logs centralizados, rastreamento distribuído e métricas individualizadas por serviço, elementos que permitem o diagnóstico mais preciso de falhas e o acionamento direcionado das equipes responsáveis, reduzindo o tempo de resposta e evitando o efeito cascata comum em sistemas monolíticos, onde uma falha em um módulo pode comprometer todo o sistema, mesmo que a causa seja localizada, o que revela um ganho expressivo na

estabilidade operacional de sistemas arquitetados por microsserviços (Francesco, Malavolta e Lago, 2017).

Entretanto, também foram identificadas limitações importantes na arquitetura de microsserviços que afetam diretamente o processo de manutenção, entre elas destaca-se a complexidade de orquestração e de controle das interdependências entre serviços, pois, apesar da separação lógica das responsabilidades, a aplicação continua sendo um ecossistema interligado que depende de comunicação eficiente, versionamento rigoroso das APIs e definição clara dos contratos entre serviços, problemas nessas interfaces podem comprometer a integridade do sistema e dificultar o trabalho das equipes de sustentação, exigindo ferramentas e processos maduros para mitigar falhas emergentes (Bogner e Zimmermann, 2016).

As dificuldades na manutenção de microsserviços foram particularmente evidentes em ambientes que não possuíam práticas de documentação contínua e arquitetura de observabilidade bem definidas, nesses contextos, a descentralização dos serviços dificultou o rastreamento dos fluxos de execução e dos pontos de falha, principalmente em aplicações com elevado número de microsserviços e interações entre módulos, o que gerou atrasos na resolução de incidentes e aumento da carga cognitiva das equipes responsáveis, demonstrando que a eficácia da manutenção em microsserviços está fortemente atrelada à governança arquitetural e ao grau de maturidade dos processos técnicos adotados (Yale Yu, Silveira e Sundaram, 2016).

No estudo de Kanizawa e Pinto (2022), observou-se que a descentralização da base de dados em sistemas baseados em microsserviços impôs desafios adicionais à manutenção, especialmente na garantia da consistência das informações entre diferentes serviços, apesar do ganho de autonomia, a necessidade de implementar estratégias como consistência eventual, eventos compensatórios e sagas complexas dificultou o processo de manutenção corretiva, pois o simples ajuste em uma regra de negócio exigia a compreensão dos impactos indiretos em múltiplos serviços, ressaltando a importância de planejamento arquitetural rigoroso desde as fases iniciais do projeto (Kanizawa e Pinto, 2022).

Os dados levantados indicam que, em comparação aos sistemas monolíticos, os microsserviços oferecem maior capacidade de evolução tecnológica sem comprometer a manutenção, pois permitem a substituição gradual de serviços legados por implementações mais modernas e otimizadas, sem a necessidade de refatorar o sistema inteiro, essa abordagem incremental reduz o risco de interrupções e possibilita que a manutenção evolutiva ocorra em paralelo com o desenvolvimento de novas funcionalidades, prática inviável em modelos monolíticos, nos quais qualquer mudança estrutural representa um risco sistêmico e demanda planejamento complexo e execução coordenada (Newman, 2015).

Cabe também ressaltar, a relação entre cultura organizacional e eficácia da manutenção, nos estudos revisados, as empresas que apresentaram melhores resultados em manutenção de sistemas

distribuídos eram aquelas que adotavam modelos ágeis, com squads dedicados a domínios específicos, autonomia para tomada de decisões técnicas e forte investimento em capacitação contínua, em contraste, ambientes mais hierárquicos e com baixa maturidade em engenharia de software enfrentaram maiores dificuldades em manter sistemas baseados em microsserviços, demonstrando que a estrutura técnica da arquitetura precisa ser acompanhada por uma transformação cultural e organizacional (Fowler e Lewis, 2014).

As limitações da arquitetura monolítica ficaram evidentes nos relatos sobre dificuldade de manutenção corretiva, principalmente quando o sistema se encontrava em fase avançada de maturidade e com grande acúmulo de regras de negócio, nesse cenário, a alteração de um único requisito frequentemente desencadeava a necessidade de testes extensivos e correções inesperadas em partes não relacionadas do código, o que elevava o custo de manutenção e gerava insegurança entre os desenvolvedores, levando à hesitação na evolução do sistema e à estagnação tecnológica, situação frequentemente descrita como “dívida técnica acumulada” (Mendes, 2021).

Em termos de tempo de inatividade, observou-se que a arquitetura de microsserviços permitiu manutenções mais frequentes e com menor impacto para os usuários finais, pois os serviços podiam ser atualizados, reiniciados ou substituídos de forma isolada, sem necessidade de interromper a aplicação como um todo, já nos sistemas monolíticos, a implantação de qualquer atualização exigia o desligamento temporário da aplicação ou a adoção de estratégias complexas de hot deployment, que nem sempre são viáveis ou seguras, gerando mais interrupções e perda de produtividade durante os períodos de manutenção (Tripoli, 2017).

A escalabilidade das equipes de manutenção também se mostrou mais eficiente em arquiteturas distribuídas, pois diferentes times podiam trabalhar simultaneamente em serviços distintos, com menor risco de conflitos ou sobreposição de tarefas, enquanto nos sistemas monolíticos, a simultaneidade de ações exigia maior coordenação e resultava frequentemente em retrabalhos, conflitos de merge e atrasos na entrega de correções, demonstrando que a estrutura organizacional das equipes precisa acompanhar a granularidade da arquitetura adotada para que os benefícios de manutenibilidade sejam realmente alcançados (Jazayeri, 2000).

As análises também indicaram que os sistemas monolíticos tendem a apresentar menor complexidade inicial, o que facilita a manutenção em estágios iniciais de desenvolvimento, porém essa vantagem se perde rapidamente à medida que o sistema cresce, pois a ausência de modularização e a concentração de responsabilidades resultam em código cada vez mais acoplado e difícil de manter, levando a um aumento progressivo dos custos operacionais e à necessidade de reestruturação arquitetural no médio prazo, frequentemente de forma abrupta e com riscos elevados (Sommerville, 2011).

Nos estudos revisados, ficou claro que a manutenção preventiva é mais eficaz em arquiteturas de microsserviços, pois o monitoramento granular e a resposta localizada às anomalias permitem a antecipação de falhas e a aplicação de ajustes com menor impacto, enquanto nos sistemas monolíticos, os indicadores globais dificultam a identificação precisa dos pontos críticos e retardam a ação corretiva, resultando em maior frequência de falhas em produção e maior dependência de intervenções emergenciais, com efeitos colaterais mais amplos e imprevistos (Erl, 2007).

Os resultados da análise reforçam que a escolha entre microsserviços e arquitetura monolítica deve considerar o perfil da organização, o grau de complexidade do sistema, a maturidade técnica da equipe e os requisitos operacionais do negócio, embora os microsserviços apresentem maiores desafios iniciais de implantação e governança, oferecem ganhos significativos em manutenção, escalabilidade e resiliência no longo prazo, sendo uma alternativa mais robusta para sistemas corporativos que precisam acompanhar mudanças frequentes e sustentar operações críticas com o mínimo de interrupções (The Open Group, 2018).

## 5 CONSIDERAÇÕES FINAIS

A investigação realizada ao longo deste estudo permitiu compreender com profundidade os efeitos da arquitetura de microsserviços sobre a manutenção de sistemas corporativos, revelando um conjunto de evidências que sustentam sua superioridade em termos de flexibilidade, autonomia de desenvolvimento e capacidade de resposta frente a mudanças e falhas, a estrutura modular, ao permitir a atuação isolada sobre partes específicas da aplicação, transforma o processo de manutenção em uma atividade mais previsível, com menor risco de impacto colateral e maior aderência a estratégias de melhoria contínua, o que representa um avanço em relação aos modelos tradicionais que centralizam o controle e dificultam a evolução incremental.

A descentralização proposta pelos microsserviços redefine a lógica de manutenção ao permitir que cada serviço evolua em seu próprio ritmo, com infraestrutura, linguagem e estratégias de deploy independentes, essa separação técnica, quando bem governada, resulta em ganhos operacionais substanciais, pois evita interrupções sistêmicas, reduz o tempo necessário para aplicar correções e proporciona mais segurança no versionamento das aplicações, além de contribuir para uma organização mais saudável do código, onde a responsabilidade está clara e as dependências estão explícitas, facilitando a análise de impacto e a rastreabilidade de erros.

Os dados levantados também indicam que a eficácia da manutenção em microsserviços está intimamente ligada à maturidade dos processos de engenharia de software, à cultura organizacional da empresa e ao grau de adoção de ferramentas e práticas de observabilidade, sem esses elementos, a complexidade inerente à fragmentação pode se tornar um obstáculo, gerando instabilidades, dificuldades de integração e aumento no esforço de manutenção, o que reforça a ideia de que essa

arquitetura não é uma solução mágica, mas sim uma estrutura técnica que exige disciplina, planejamento e capacitação para que seus benefícios sejam plenamente concretizados.

Embora a arquitetura monolítica ainda seja eficaz em determinados contextos, especialmente em sistemas com baixa complexidade ou ciclos de vida curtos, suas limitações se tornam evidentes à medida que o sistema cresce em funcionalidades e em número de usuários, o acoplamento entre os módulos, a dependência de grandes ciclos de deploy e a dificuldade de realizar manutenções pontuais sem afetar outras partes da aplicação tornam esse modelo menos eficiente para organizações que precisam responder com agilidade às mudanças de mercado, adaptar-se rapidamente a novos requisitos ou sustentar operações críticas com alta disponibilidade.

A comparação entre os dois modelos evidencia que a arquitetura de microsserviços oferece um ambiente mais propício à inovação contínua, à escalabilidade técnica e à manutenção sustentável, ao permitir que diferentes equipes atuem simultaneamente sobre partes distintas da aplicação, esse modelo favorece a descentralização da inteligência técnica e operacional, reduzindo os gargalos na manutenção e democratizando o conhecimento do sistema, esse aspecto tem implicações diretas na retenção de talentos, na agilidade organizacional e na capacidade de manter a aplicação atualizada frente às transformações constantes do cenário tecnológico.

No entanto, adotar microsserviços exige uma reestruturação profunda não apenas na arquitetura da aplicação, mas também nas práticas de trabalho das equipes envolvidas, na forma de comunicação entre os setores e nos fluxos de governança técnica, o sucesso da manutenção nesse modelo não depende exclusivamente do desenho da arquitetura, mas sim da capacidade da organização em estabelecer padrões, automatizar processos, garantir observabilidade e oferecer suporte técnico constante às suas equipes, o que transforma o desafio técnico em um processo de transformação cultural e estratégica que precisa ser planejado com clareza desde o início.

A manutenção preventiva em sistemas com arquitetura distribuída mostra-se mais eficiente quando apoiada por métricas bem definidas, monitoramento proativo e documentação contínua, elementos que, embora exijam mais esforço inicial, proporcionam estabilidade a longo prazo e evitam a sobrecarga gerada por manutenções corretivas emergenciais, ao mesmo tempo, os serviços bem projetados favorecem a manutenção evolutiva, permitindo que funcionalidades obsoletas sejam atualizadas ou substituídas sem impactar o funcionamento geral da aplicação, o que amplia a longevidade do sistema e reduz a necessidade de grandes reescritas.

O ciclo de vida da manutenção em microsserviços é mais dinâmico, com intervenções frequentes, porém pontuais, realizadas com o apoio de automações e feedback contínuo dos ambientes de produção, esse dinamismo exige uma postura proativa das equipes de desenvolvimento, que precisam estar preparadas para atuar de forma descentralizada, priorizando a estabilidade de seu domínio funcional, mas ao mesmo tempo colaborando com as demais áreas para garantir a integridade

do sistema como um todo, esse equilíbrio entre autonomia e alinhamento coletivo é um dos pontos mais sensíveis e estratégicos da manutenção em sistemas distribuídos.

Com base nos resultados obtidos, é possível afirmar que a arquitetura de microsserviços representa uma evolução natural no design de aplicações corporativas, especialmente quando a manutenibilidade, a escalabilidade e a resiliência são critérios essenciais para o sucesso do sistema, sua adoção, entretanto, deve ser feita com responsabilidade, levando em consideração o estágio de maturidade da empresa, a natureza do projeto e os objetivos de longo prazo, pois os ganhos obtidos só se tornam sustentáveis quando acompanhados de práticas sólidas de engenharia e de uma cultura organizacional voltada à melhoria contínua.

Este estudo contribui para o entendimento das nuances envolvidas na manutenção de sistemas sob diferentes modelos arquiteturais e reforça a importância da arquitetura como fator estratégico na longevidade e eficiência das soluções corporativas, ao evidenciar os ganhos e desafios dos microsserviços, espera-se que os resultados aqui discutidos sirvam de base para decisões mais conscientes, sustentadas por evidências e alinhadas com os objetivos organizacionais, incentivando uma visão crítica e planejada sobre as escolhas arquiteturais e suas consequências na prática cotidiana da manutenção de sistemas.

## REFERÊNCIAS


ABREU, F.; CARVALHO, J.; ROCHA, Á. Strategic alignment between information systems and organizational agility: A systematic literature review. Procedia Computer Science, v. 138, p. 758–765, 2018.

BOGNER, J.; ZIMMERMANN, A. Towards integrating microservice principles into enterprise architecture. In: IEEE International Conference on Enterprise Distributed Object Computing. IEEE, 2016.

ERL, T. SOA: Princípios de projeto de serviços. Rio de Janeiro: Alta Books, 2007.

FOWLER, M.; LEWIS, J. Microservices: A definition of this new architectural term. 2014. Disponível em: https://martinfowler.com/articles/microservices.html.

FRANCESCO, P.; MALAVOLTA, I.; LAGO, P. Research on architecting microservices: trends, focus, and potential for industrial adoption. In: IEEE International Conference on Software Architecture (ICSA). IEEE, 2017.

GIL, A. C. Como elaborar projetos de pesquisa. 5. ed. São Paulo: Atlas, 2008.

JAZAYERI, M. The role of architecture in software development. In: Proceedings of the European Software Engineering Conference. Springer, 2000.

KANIZAWA, D. T.; PINTO, G. S. Arquitetura de microsserviços. Interface Tecnológica, v. 19, n. 2, p. 308–315, 2022.

LAKATOS, E. M.; MARCONI, M. A. Fundamentos de metodologia científica. 7. ed. São Paulo: Atlas, 2003.

MENDES, I. S. Arquitetura monolítica vs microsserviços: uma análise comparativa. 2021. Trabalho de Conclusão de Curso (Graduação em Engenharia de Software) – Universidade de Brasília, Brasília, 2021.

NEWMAN, S. Building Microservices: Designing Fine-Grained Systems. 1. ed. Sebastopol: O'Reilly Media, 2015.

OLIVEIRA, M. S. Arquitetura de micro serviços: uma comparação com sistemas monolíticos. In: Anais do EnGeTec 2023. São Paulo: Fatec ZL, 2023.

SOMMERVILLE, I. Engenharia de Software. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

THE OPEN GROUP. TOGAF Standard, Version 9.2. The Open Group, 2018.

YALE YU, D.; SILVEIRA, F.; SUNDARAM, R. Challenges in adopting microservice architecture: An industrial study. In: Proceedings of the International Conference on Cloud Engineering (IC2E). IEEE, 2016.