UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Métricas em Microsserviços: Análise da ISO/IEC 25010 e Recomendações por LLM

Dissertação

Flávia Caroline dos Santos Galdino

São Cristóvão – Sergipe

2025

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Flávia Caroline dos Santos Galdino

Métricas em Microsserviços: Análise da ISO/IEC 25010 e Recomendações por LLM

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe como requisito para a obtenção do título de mestre em Ciência da Computação.

Orientador(a): Dr. Fabio Gomes Rocha
Coorientador(a): Dr. Michel dos Santos Soares

São Cristóvão – Sergipe

2025

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL
UNIVERSIDADE FEDERAL DE SERGIPE

G149m

Galdino, Flávia Caroline dos Santos
Métricas em microsserviços: análise da ISO/IEC 25010 e recomendações por LLM / Flávia Caroline dos Santos Galdino ; orientador Fabio Gomes Rocha. - São Cristóvão, 2025.
82 f.; il.

Dissertação (mestrado em Ciência da Computação) – Universidade Federal de Sergipe, 2025.

1. Computação. 2. Software - Desenvovimento. I. Fabio Gomes orient. II. Título.

CDU 004.4

Ata da Sessão Solene de Defesa da Dissertação do Curso de Mestrado em Ciência da Computação-UFS. Candidata: Flávia Caroline dos Santos Galdino

Em 13 dias do mês de janeiro do ano de dois mil e vinte cinco, com início às 16hs, realizou-se na Sala de Seminários do PROCC da Universidade Federal de Sergipe, na Cidade Universitária Prof. José Aloísio de Campos, a Sessão Pública de Defesa de Dissertação de Mestrado do candidata **Flávia Caroline dos Santos Galdino** , que desenvolveu o trabalho intitulado: **"Métricas em microsserviços: Análise da ISO/IEC 25010 e recomendação por LLM",** sob a orientação do Prof. Dr. **Fábio Gomes Rocha.** A Sessão foi presidida pelo Prof. Dr. **Fabio Gomes Rocha** (PROCC/UFS), que após a apresentação da dissertação passou a palavra aos outros membros da Banca Examinadora, o Dr. **Alejandro C. Frery (**VUW**),** o prof. Dr. **Glauco de Figueiredo Carneiro** (Procc/UFS) e, em seguida, Dr. **Michel dos Santos Soares** (PROCC/UFS) . Após as discussões, a Banca Examinadora reuniu-se e considerou o mestrando (a) Aprovada . Atendidas as exigências da Instrução Normativa 05/2019/PROCC, do Regimento Interno do PROCC (Resolução 67/2014/CONEPE), e da Resolução nº 04/2021/CONEPE que regulamentam a Apresentação e Defesa de Dissertação, e nada mais havendo a tratar, a Banca Examinadora elaborou esta Ata que será assinada pelos seus membros e pelo mestrando.

Cidade Universitária "*Prof. José Aloísio de Campos",* 13 de janeiro de 2025.

Documento assinado digitalmente
gov.br FABIO GOMES ROCHA
Data: 05/02/2025 19:21:19-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Fábio Gomes Rocha
(PROCC/UFS)
Presidente

Documento assinado digitalmente
gov.br MICHEL DOS SANTOS SOARES
Data: 17/01/2025 15:46:41-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Michel dos Santos Soares
(PROCC/UFS)
Coorientador

Prof. Dr. Alejandro C. Frery
(Victoria University - Nova Zelândia)
Examinador Externo

Documento assinado digitalmente
gov.br GLAUCO DE FIGUEIREDO CARNEIRO
Data: 17/01/2025 10:32:16-0300
Verifique em https://validar.iti.gov.br

Prof. Dr.Glauco de Figueiredo Carneiro
(PROCC/UFS)
Examinador Interno

Documento assinado digitalmente
gov.br FLAVIA CAROLINE DOS SANTOS GALDINO
Data: 17/01/2025 16:42:13-0300
Verifique em https://validar.iti.gov.br

Flávia Caroline dos Santos Galdino
Candidata

Resumo

Introdução: O padrão de microsserviços, que organiza um sistema em uma coleção de serviços independentes, promove flexibilidade no desenvolvimento, acelerando o ciclo de entrega e melhorando a qualidade do software. Esse modelo facilita a adaptação às mudanças, reduzindo o tempo de desenvolvimento e aumentando a satisfação das equipes. **Objetivo:** Este estudo busca coletar métricas de desempenho e qualidade da academia, por meio de uma revisão teórica e sistemática, e compreender a realidade da indústria por meio de uma pesquisa de opinião. O objetivo é identificar desafios e soluções para o gerenciamento de equipes de microsserviços, relacionando métricas de qualidade com os requisitos da norma ISO/IEC 25010, avançando as pesquisas científicas e apoiando o gerenciamento de tecnologias baseadas em microsserviços com métricas adaptáveis que integrem inteligência artificial. **Metodologia:** Foi adotado o método *Design Science Research*, no qual, no estudo de caso foram desenvolvidos dois artefatos: um catálogo de métricas e uma ferramenta de recomendações de métricas, baseados neste catálogo, para apoiar o gerenciamento de microsserviços. **Conclusão:** Este trabalho contribui para o entendimento das necessidades acadêmicas e da indústria, comparando ambas as áreas e propondo métricas para orientar pesquisadores, gestores e desenvolvedores. Ademais, oferece informações baseadas em dados para decisões nas implementação de microsserviços e serve como base para futuras pesquisas. As pesquisas futuras estarão focadas no aprofundamento de Large Language Models (LLM), no engajamento dos usuários em ferramentas baseadas em LLM, e na melhoria da LLM desenvolvida neste estudo, com potencial impacto tanto para a academia quanto para a indústria, otimizando a gestão de microsserviços e melhorando sua qualidade.

Palavras-chave: Métricas para Software. Microsserviços. Equipe de desenvolvimento.

Abstract

Introduction: The microservices pattern, which organizes a system into a collection of independent services, promotes flexibility in development, accelerating the delivery cycle and improving software quality. This model facilitates adaptation to changes, reducing development time and increasing team satisfaction. **Objective:** This study aims to collect performance and quality metrics from academia through a theoretical and systematic review and to understand the industry's reality through an opinion survey. The goal is to identify challenges and solutions for managing microservice teams, relating quality metrics to the ISO/IEC 25010 Standard requirements, advancing scientific research, and supporting the management of microservice-based technologies with adaptable metrics that integrate artificial intelligence. **Methodology:** The *Design Science Research* method was adopted, where, in the case study, two artifacts were developed: a metrics catalog and a metrics recommendation tool based on this catalog to support microservice management. **Conclusion:** This work contributes to understanding academic and industrial needs by comparing both areas and proposing metrics to guide researchers, managers, and developers. Furthermore, it provides data-driven insights for decision-making in microservice implementation and is a foundation for future research. Future studies will focus on advancing Large Language Models (LLM), user engagement in LLM-based tools, and improving the LLM developed in this study, potentially impacting academia and industry, optimizing microservice management, and enhancing its quality.

Keywords: Metrics for Software. Microservices. Development team.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

BLEU	*Bilingual Evaluation Understudy*
BP	*Brevity Penalty*
DSR	*Design Science Research*
GLUE	*General Language Understanding Evaluation*
GQM	*Goal-Question-Metric*
ICCSA	*International Conference on Computational Science and Its Applications*
LLM	*Large-Language Model*
PROCC	Programa de Pós-Graduação em Ciência da Computação
RAG	*Retrieval Augmented Generation*
SBSI	Simpósio Brasileiro de Sistemas de Informação
SQuAD	*Stanford Question Answering Dataset*
TF-IDF	*Term Frequency-Inverse Document Frequency*
UFS	Universidade Federal de Sergipe
VSM	*Vector Space Model*
WTDQS	*Workshop de Teses e Dissertações em Qualidade de Software*
SBQS	*Simpósio Brasileiro de Qualidade de Software*
J.UCS	*Journal of Universal Computer Science*

Sumário

1
Introdução

Almeida *et al.* (ALMEIDA; SILVA, 2020) destaca que sistemas monolíticos enfrentam desafios devido ao forte acoplamento entre seus componentes, o que torna difícil localizar e modificar partes específicas do código (MEGARGEL; SHANKARARAMAN; WALKER, 2020). Embora sistemas monolíticos se mostrem eficazes em muitos cenários, a gestão de sistemas mais complexos pode se tornar desafiadora, principalmente quando há necessidade de maior escalabilidade e flexibilidade. No entanto, tanto os sistemas monolíticos quanto os microsserviços apresentam vantagens e desafios próprios, e a escolha entre eles depende das exigências específicas de cada projeto. Ou seja, uma abordagem não é necessariamente superior à outra, mas sim mais adequada conforme o contexto e as necessidades de escalabilidade e manutenção do software. À medida que a complexidade dos sistemas cresce (SANTOS; SILVA, 2020) e a demanda por maior modularidade aumenta, novas abordagens arquiteturais, como os microsserviços, têm sido cada vez mais exploradas.

Os serviços, no contexto da arquitetura orientada a serviços (SOA) (BOUMAHDI et al., 2023) ou de microserviços (ZHOU et al., 2023), são unidades independentes que executam tarefas específicas e se comunicam entre si através de interfaces padronizadas, como APIs ou mensagens (RAJ; SADAM, 2021). A principal vantagem dessa abordagem é a modularidade (BOGNER et al., 2019), pois cada serviço pode ser desenvolvido, implantado e escalado de forma independente, facilitando a manutenção e a inovação em sistemas complexos (GENFER; ZDUN, 2022). No entanto, a comunicação entre os serviços pode gerar latência, e a gestão de transações distribuídas pode ser um desafio, especialmente em ambientes com muitos serviços interconectados (MOHAMED; EL-GAYAR, 2021).

A Arquitetura Orientada a Serviços (SOA) foi introduzida para organizar e integrar uma coleção de aplicações de software isoladas em um conjunto de serviços. Esses serviços se conectam e acessam uns aos outros por meio de interfaces e mensagens, permitindo que sistemas diferentes se integrem de maneira mais eficaz. O SOA foi adotado para desenvolver

software distribuídos com a promessa de melhor integração de sistemas legados e maior agilidade no desenvolvimento de software por meio da reutilização de código (GENFER; ZDUN, 2022; FRANcA et al., 2020). No entanto, o SOA pode enfrentar desafios relacionados à complexidade da integração e ao gerenciamento de transações em ambientes distribuídos, especialmente quando diferentes tecnologias estão envolvidas.

A evolução da SOA levou ao surgimento dos microsserviços que representam a segunda geração da arquitetura orientada a serviços (BOUMAHDI et al., 2023; RODRIGUEZ et al., 2023). Os microsserviços proporcionam uma modularidade maior, o que permite um desenvolvimento mais ágil e manutenção mais eficiente (RODRIGUEZ et al., 2023). Eles permitem que as equipes se concentrem em funcionalidades específicas, com cada serviço funcionando de maneira independente, facilitando a escalabilidade, o teste e a implantação (AGARWAL et al., 2021). A abordagem de microsserviços facilita a utilização de diferentes tecnologias e linguagens para diferentes partes do sistema, promovendo maior flexibilidade (BOGNER et al., 2019) e adaptação às mudanças (FRANcA et al., 2020). No entanto, a transição para microserviços pode ser desafiadora. Ela envolve complexidade adicional em termos de orquestração e comunicação entre serviços, o que pode resultar em dificuldades na gestão de falhas (BOUMAHDI et al., 2023). Garantir a consistência de dados e a segurança em uma arquitetura distribuída é mais complexo do que em sistemas monolíticos (KLEFTAKIS et al., 2022).

Na década de 2000, a Engenharia de Software evoluiu para se adaptar ao ritmo acelerado da evolução arquitetural (AGARWAL et al., 2021) e ao uso de métodos ágeis. Neste contexto, a Arquitetura Orientada a Serviços, ou *Service-Oriented Architecture* (SOA), foi introduzida para organizar e unir uma coleção de aplicações de software previamente isoladas em um conjunto de serviços. Esses serviços se conectam e acessam uns aos outros por meio de interfaces e mensagens (BOUMAHDI et al., 2023). O SOA foi adotado para desenvolver software distribuídos com a promessa de melhor integração de sistemas legados e maior agilidade no desenvolvimento de software por meio da reutilização de código, a fim de atender às demandas de flexibilidade e escalabilidade (GENFER; ZDUN, 2022; FRANcA et al., 2020).

A evolução da SOA levou ao surgimento dos microsserviços que representam a segunda geração da arquitetura orientada a serviços (BOUMAHDI et al., 2023). Os microsserviços possuem uma arquitetura inspirado no SOA, mas com melhorias relacionadas ao desempenho, e remoção de níveis desnecessários de complexidade. Nesta abordagem, os software distribuídos são compostos por módulos independentes, permitindo que os desenvolvedores foquem em serviços específicos para implementar melhorias ou correções de *bugs* de maneira mais eficiente. Este aspecto também economiza tempo e torna os testes mais objetivos (RODRIGUEZ et al., 2023).

Os microsserviços surgiram em 2015 como uma solução para a crescente demanda por escalabilidade, resiliência e flexibilidade na indústria de software (ROCHA; SOARES; RODRIGUEZ, 2023a). A granularidade refinada dos microsserviços e a ênfase na governança

de componentes os tornaram uma escolha de destaque para o desenvolvimento de software distribuído, particularmente na computação em nuvem (HASSAN; BAHSOON; BUYYA, 2022a). As métricas propostas nesta dissertação se alinham perfeitamente com os princípios dos microsserviços (ÜNLü et al., 2024), aprimorando seus benefícios e garantindo sua implementação efetiva (ERDEI; TOKA, 2023).

Sheikh e Ambhaikar (SHEIKH; AMBHAIKAR, 2021) destacam as vantagens que a arquitetura de microsserviços oferece, incluindo a facilidade de familiarização com o código-fonte de cada módulo (microsserviço). Por outro lado, Dong *et al.* (DONG et al., 2020) argumentam que a qualidade dos microsserviços está intrinsecamente ligada à coesão, escalabilidade e tempo de resposta, sendo este último um atributo essencial das métricas de desempenho. Genfer *et al.* (GENFER; ZDUN, 2022) enfatizam que o baixo acoplamento é fundamental para a modularidade dos microsserviços. Usar métricas para monitorar essas informações se torna imperativo para garantir a evolução do sistema. Nesse contexto, é essencial reconhecer a necessidade de desenvolver uma abordagem inovadora para o desenvolvimento de sistemas orientados a serviços (AGRAWAL, 2021).

A Figura 1 é uma representação da diferença entre um sistema monolítico e um sistema baseado na arquitetura de microsserviços.

Figura 1 – Arquiteturas de sistemas monolíticos de microsserviços (DRAGONI et al., 2017).

O objetivo geral desta pesquisa é desenvolver um catálogo de métricas aplicadas ao desenvolvimento de sistemas baseados em arquitetura de microsserviços, que possa ser utilizado para aprimorar a avaliação e a gestão da qualidade desses sistemas. Para alcançar esse objetivo,

foram estabelecidos os seguintes objetivos específicos:

1. Identificar métricas relevantes para arquitetura de microsserviços por meio de uma revisão teórica, revisão sistemática da literatura e análise bibliométrica.
2. Construir um catálogo estruturado de métricas que contemple diferentes perspectivas da qualidade de software, com base nos atributos definidos pela norma ISO/IEC 25010.
3. Desenvolver um sistema inteligente baseado em Modelos de Linguagem de Grande Escala (*Large-Language Models* – LLMs) e no Modelo de Espaço Vetorial (*Vector Space Model* – VSM), que permita consultas interativas e forneça recomendações personalizadas de métricas com base no catálogo.
4. Validar o catálogo e o sistema inteligente por meio de um estudo de caso, avaliando sua aplicabilidade e impacto no contexto prático.

A abordagem proposta envolve diversas etapas metodológicas. Inicialmente, foi realizada uma revisão teórica sobre a qualidade da arquitetura de microsserviços, complementada por uma revisão sistemática e análise bibliométrica. Um survey foi conduzido para compreender as necessidades da indústria, enquanto um estudo de caso foi desenvolvido com o objetivo de validar o catálogo de métricas proposto. Foi implementada nesta pesquisa uma ferramenta que utiliza Modelos de Linguagem de Grande Escala (LLMs) e o Modelo de Espaço Vetorial (VSM) para fornecer recomendações personalizadas baseadas no catálogo. A norma ISO/IEC 25010 será utilizada como referência para cruzar as métricas coletadas em diferentes etapas da pesquisa (GENFER; ZDUN, 2022).

Métricas bem definidas desempenham um papel essencial na operação e no funcionamento eficaz de sistemas baseados em microsserviços, orientando decisões relacionadas ao *design*, implantação (PULCINELLI; PEDROSO; BRUSCHI, 2023) e operação do software. Contudo, a identificação, seleção e interpretação de métricas adequadas apresentam desafios significativos (SCHRöER; WITTFOTH; GMEZ, 2021), devido às múltiplas dimensões e interdependências envolvidas. Essa complexidade reforça a relevância da abordagem aqui proposta, que busca estabelecer um método sistemático e inovador para a aplicação de métricas em ambientes de microsserviços.

Existem empresas que adotam a arquitetura de microsserviços, substituindo sistemas monolíticos e a arquitetura SOA tradicional (ALMEIDA; SILVA, 2020). Esta adoção está associada a menor investimento em hardware e melhor escalabilidade e manutenibilidade do software que adota a arquitetura de microsserviços (ZHOU et al., 2023). Migrar ou criar novos softwares, passando de arquiteturas mais antigas, como sistemas monolíticos e SOA, para microsserviços, pode trazer diversos benefícios (SANTOS; SILVA, 2020), incluindo melhor estruturação das equipes de desenvolvimento de software responsáveis por cada serviço (AUER

et al., 2021), tempo de manutenção reduzido e facilidade de adaptação às mudanças, evitando assim a estagnação tecnológica (RODRIGUEZ et al., 2023). Embora a adoção de microsserviços ofereça benefícios, sua complexidade requer um esforço significativo para gerenciar quando introduzida em sistemas (ZHOU et al., 2023).

À medida que os microsserviços adotam uma arquitetura nativa da nuvem, migrar um sistema para a nuvem se torna mais prático (SANTOS; PAULA, 2021). No entanto, sem uma metodologia de migração adequada, todo o esforço pode resultar em uma solução inadequada. Por esse motivo, é necessário que os desenvolvedores tenham habilidades não apenas no processo de implementação (ALMEIDA; SILVA, 2020), mas também em tecnologias e protocolos de nuvem. Os microsserviços herdaram de arquiteturas passadas, como os sistemas monolíticos e SOA (OUMOUSSA; SAIDI, 2024), a necessidade de serviços de suporte para descoberta de serviços e balanceamento de carga, sendo preciso a comunicação correta entre os serviços (MENG et al., 2020). O suporte prometido para múltiplas linguagens, que pode parecer vantajoso em teoria, frequentemente se torna insustentável na prática (ZHOU et al., 2023). A gestão de diferentes linguagens e suas dependências gera complexidade adicional, dificultando a manutenção, escalabilidade e aumentando a sobrecarga operacional e de integração entre os serviços (ÜNLü et al., 2024).

Grandes esforços podem ser necessários no uso de microsserviços, impactando diretamente suas equipes de desenvolvimento (ALMEIDA; SILVA, 2020). A falta de entendimento sobre desempenho da equipe, arquitetura e software em geral pode impactar negativamente a entrega do produto e levar à escassez de líderes de equipe (MEGARGEL; SHANKARARAMAN; WALKER, 2020) que podem analisar assertivamente as necessidades convergentes . Para esta análise mais assertiva (FRANcA et al., 2020), existem métricas como: métricas de desempenho, complexidade e granularidade (HASSAN; BAHSOON; BUYYA, 2022a; WEERASINGHE; PERERA, 2021).

Esta dissertação está organizada em seis capítulos, que estruturam de forma lógica o desenvolvimento da pesquisa. O Capítulo 2 apresenta o referencial teórico, abordando os conceitos e fundamentos que sustentam este trabalho. O Capítulo 3 descreve a metodologia adotada, detalhando as etapas e métodos utilizados para alcançar os objetivos propostos. O Capítulo 4 discute os resultados obtidos na construção da base de conhecimento, enquanto o Capítulo 5 explora os resultados relacionados aos artefatos desenvolvidos, incluindo o catálogo de métricas e o sistema inteligente proposto. Por fim, o Capítulo 6 apresenta as conclusões do trabalho, bem como os artigos científicos produzidos no âmbito desta pesquisa, destacando as principais contribuições e possibilidades de trabalhos futuros.

2

Referencial Teórico

Este capítulo apresenta o referencial teórico necessário para embasar o desenvolvimento deste trabalho. Inicialmente, são discutidos os conceitos fundamentais de Arquitetura Monolítica, Arquitetura Orientada a Serviços (SOA) e Microsserviços, destacando suas características, diferenças e a evolução tecnológica que levou à popularização de arquiteturas distribuídas. Em seguida, é explorada a ISO/IEC 25010, um padrão amplamente utilizado para avaliar a qualidade de sistemas de software, proporcionando uma visão estruturada sobre os atributos de qualidade. Para complementar, são apresentadas as Métricas utilizadas no contexto de sistemas distribuídos, bem como o papel de Large Language Models (LLMs) e técnicas de Geração Aumentada de Recuperação (RAG) no aprimoramento de processos de busca e geração de informações. Aborda-se o Modelo de Espaço Vetorial (VSM) e as Métricas de Avaliação de VSM, fundamentais para entender a aplicação de vetores em recuperação de informações e avaliação da similaridade semântica. Esses conceitos fornecem a base teórica necessária para compreender as abordagens e soluções propostas nesta dissertação. Por fim, será apresentado os trabalhos relacionados.

2.1 Arquitetura Monolítica

A arquitetura monolítica possui uma arquitetura em que todos os componentes estão integrados em uma única aplicação ou unidade coesa (KLEFTAKIS et al., 2022). Esta arquitetura permite que todas as funcionalidades e camadas do sistema estejam integradas em uma única base de código, assim, sistemas pequenos e simples se tornam mais fáceis de serem implementados (MEGARGEL; SHANKARARAMAN; WALKER, 2020). Como vantagens, a arquitetura monolítica possui pontos como a simplicidade do desenvolvimento e implantação, visto que tudo está contido em um único pacote. Uma única tecnologia para todo o sistema e, ainda, o monitoramento e a depuração são mais fáceis, pois todas as informações estão centralizadas (KLEFTAKIS et al., 2022).

No entanto, a arquitetura monolítica também apresenta desafios significativos, sendo que o software é, normalmente, executado em um único processo, gerando problemas de desempenho à medida que cresce em tamanho e complexidade. A escalabilidade horizontal é limitada, pois não é possível dimensionar componentes individuais separadamente (ROCHA; SOARES; RODRIGUEZ, 2023b). As atualizações e implantações também podem ser arriscadas, pois uma alteração em uma parte do sistema pode afetar todo o software (KLEFTAKIS et al., 2022). Esses desafios mostram a necessidade de arquiteturas mais flexíveis e escaláveis, como a Arquitetura Orientada a Serviços (SOA). Na SOA, o sistema é dividido em serviços independentes, o que facilita a escalabilidade, as atualizações e a modificação de componentes sem impactar o restante da aplicação (MUNIALO; MUKETHA; OMIENO, 2020).

2.2 Arquitetura Orientada a Serviços (SOA)

SOA é um modelo de comunicação composto por componentes independentes e distribuídos (TOUEIR; BROISIN; SIBILLA, 2011), que se comunicam por meio de mensagens (MUNIALO; MUKETHA; OMIENO, 2020), proporcionando interoperabilidade (ADESINA; DARAMOLA; AYO, 2010) e fácil manutenção (NURAINI; WIDYANI, 2014). Esta camada que constitui parte do software é fundamental porque as regras de negócio residem e são validadas nos serviços (HOJAJI; SHIRAZI, 2010).

No contexto da SOA, as regras de negócio são serviços web compostos, ou seja, de baixa granularidade, que executam uma sequência de passos para atingir um determinado objetivo (KHOSHKBARFOROUSHHA; JAMSHIDI; SHAMS, 2010). Uma das principais características da SOA é a reutilização, o que permite reduzir a complexidade de sistemas orientados a serviços e o volume de informações consumidas (FIEGLER; DUMKE, 2011). Este aspecto auxilia na construção de novas funcionalidades, aumentando a agilidade de desenvolvimento e trazendo diversos benefícios. Esta arquitetura possui uma estrutura voltada para a integração (ZHANG; XINKE, 2009a), que permite que diversos sistemas trabalhem em conjunto, interagindo de forma coesa entre si em uma ou mais aplicações (BALFAGIH; HASSAN, 2009).

Empresas que desejam utilizar plataformas com sua arquitetura orientada a serviços (RODRIGUEZ et al., 2023) adicionam ainda os benefícios de eficiência energética, uso eficiente de recursos, configurabilidade e neutralidade (GONZALEZ et al., 2011). É importante garantir a segurança das aplicações desenvolvidas em SOA dessas empresas para que fiquem protegidas contra possíveis danos (RAJ; SADAM, 2021).

Apesar de ganhar ampla aceitação (RAJ; SADAM, 2021), a arquitetura SOA ainda apresenta muitos desafios (JAWADDI; JOHARI; ISMAIL, 2022), como alta dependência entre os serviços, diferente dos microsserviços. Há apenas uma base de dados para todos os serviços, sendo empregada por diversos serviços e em diversos protocolos de mensagens, impactando seu tempo de resposta (SHEIKH; AMBHAIKAR, 2021). Tais problemas podem impactar negativamente

a adoção da SOA (RAJ; SADAM, 2021). Assim, buscando resolver diversos problemas que existiam no SOA e na arquitetura monolítica e sendo fortemente incluenciada pela computação em nuvem, surgem os microsserviços (ROCHA; SOARES; RODRIGUEZ, 2023b).

2.3 Microsserviços

A arquitetura de microsserviços surgiu da SOA dentro das comunidades de desenvolvimento ágil e recentemente (TUMMALAPALLI et al., 2022) vem recebendo atenção tanto da indústria quanto da academia (LI et al., 2022). Cada microsserviço deve ter uma única responsabilidade, o que facilita sua escalabilidade horizontal, pois cada módulo tem suas demandas, podendo ter seus recursos computacionais expandidos de forma independente (BOGNER et al., 2019).

Em projetos que adotam a arquitetura de microsserviços, a tolerância a falhas é essencial, tornando a aplicação robusta em comparação a um sistema monolítico (AGARWAL et al., 2021), pois, a infraestrutura permite maior escalabilidade individual (FOWLER, 2019; RODRIGUEZ VIRGINIA YANNIBELLI; MENEZES, 2023). Os microsserviços são mais fáceis de entender, escalar (SANTOS; PAULA, 2021), implantar e fornecem serviços escaláveis e fracamente acoplados (SHEIKH; AMBHAIKAR, 2021).

Os microsserviços fornecem flexibilidade em alterações de código, permitindo a modularização do código (SHEIKH; AMBHAIKAR, 2021). Hassan *et al.* (HASSAN; BAHSOON; BUYYA, 2022b) mencionam o termo "microservitização" como um valor resultante da flexibilidade dessa arquitetura, descrevendo-a como o isolamento de vários serviços específicos que podem ajudar a lidar com incertezas operacionais, facilitando o rastreamento de erros, a substituição do serviço, a oferta de manutenção ágil (KUMAR; BHATIA, 2015; CHHILLAR; GAHLOT, 2017) e, em uma análise mais profunda, redução de tempo, desenvolvedores (KESSEL; ATKINSON, 2015) e outros custos financeiros (HASSAN; BAHSOON; BUYYA, 2022b). Os microsserviços têm um propósito disruptivo e fornecem vários benefícios (LI et al., 2022), como manutenibilidade (KUMAR; BHATIA, 2015), independência (CERNY et al., 2023), reutilização (DONG et al., 2020) e escalabilidade. Entretanto, múltiplos fatores influenciam a qualidade do software, incluindo desempenho, segurança e manutenibilidade, que são requisitos da Norma ISO/IEC 25010. Ainda, a presença de usabilidade, compatibilidade e interoperabilidade tem um papel que pode auxiliar na qualidade dos serviços (FRANcA; JOYCE; SOARES, 2015). Portanto, é importante garantir a evolução do software com base nesses requisitos (KUMAR; BHATIA, 2015).

2.4 ISO/IEC 25010

A Norma ISO/IEC 25010 foi derivada da Norma ISO/IEC 9126, uma norma internacional de qualidade de produtos de software. A Norma ISO/IEC 9126 (ISO IEC, 1991) define atributos de qualidade consistindo em seis características: adequação funcional, confiabilidade, usabilidade, eficiência de desempenho, manutenibilidade e portabilidade (KUMAR; BHATIA, 2015), juntamente com 27 subcaracterísticas. Em 2011, a ISO/IEC 25010 introduziu duas novas características, segurança e compatibilidade, e reorganizou suas subcaracterísticas para aumentar a clareza (FRANcA; JOYCE; SOARES, 2015). A Norma ISO/IEC 25010:2023 adicionou a característica Segurança e cinco subcaracterísticas (ACHARYULU; SEETHARAMAIAH, 2015), visando abordar a integridade humana e ambiental. Para uma compreensão abrangente, as características e seus conceitos foram resumidos na Tabela 1 com base no site oficial[1]

Tabela 1 – Características e suas respectivas descrições.

Característica	Conceito
Adequação funcional	Garante que o produto execute as tarefas desejadas conforme especificado.
Eficiência de desempenho	Capacidade do produto de operar dentro de parâmetros de tempo e recursos definidos.
Compatibilidade	Capacidade do produto de interagir com outros e utilizar efetivamente recursos compartilhados.
Interoperabilidade	Capacidade do produto de se comunicar com os usuários para executar tarefas específicas.
Confiabilidade	Garante que o produto opere sem falhas em situações predefinidas.
Segurança	Protege dados e usuários contra acesso não autorizado e ataques maliciosos.
Manutenção	Facilita alterações no produto por mantenedores designados.
Segurança	Previne riscos à vida humana, saúde, propriedade ou meio ambiente sob circunstâncias específicas.

A Figura 2 representa a estrutura da Norma ISO/IEC 25010 com características e subcaracterísticas atualizadas.

[1] <https://www.iso.org/>

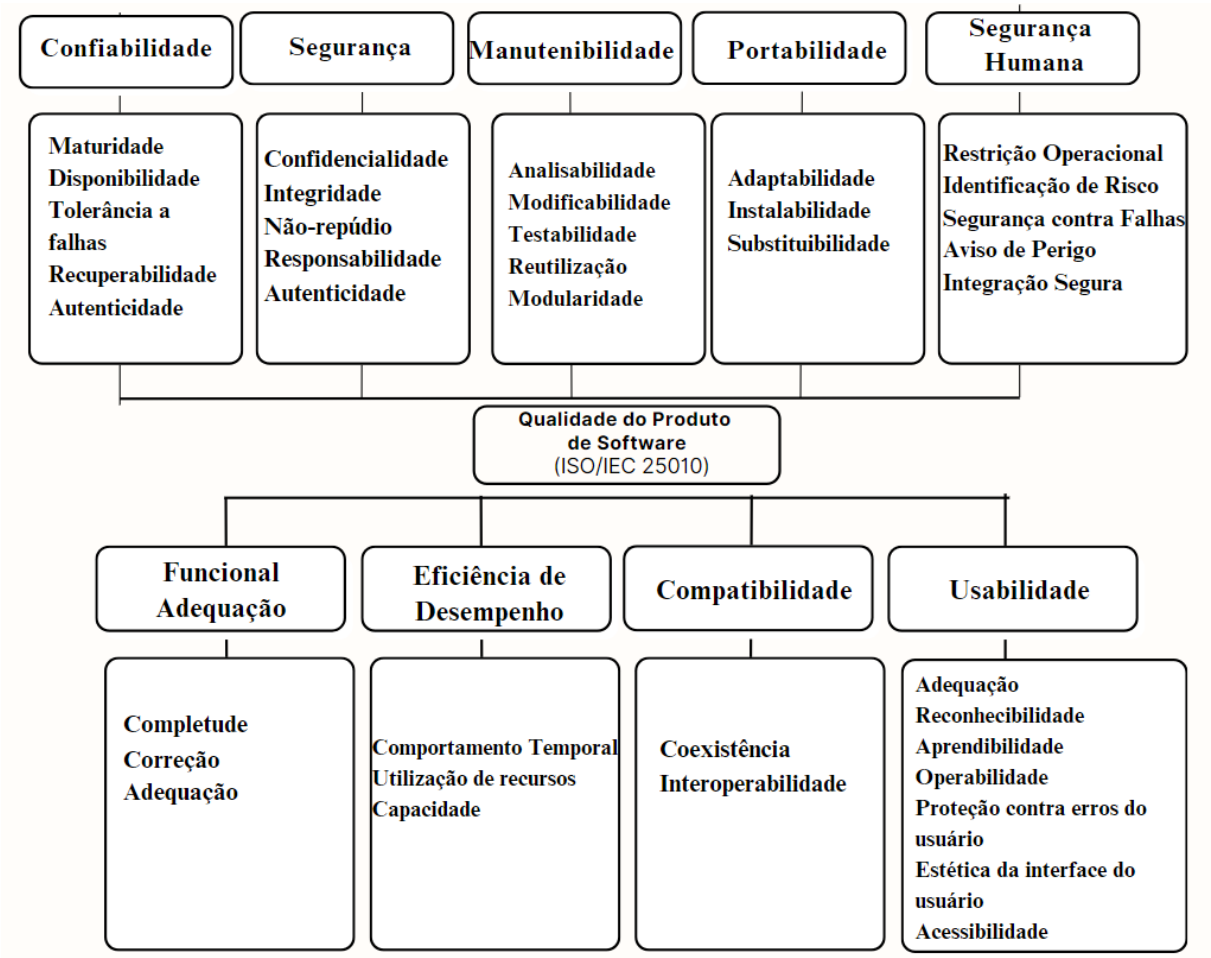

Figura 2 – Modelo de qualidade pela norma ISO/IEC 25010

Como detalhado na Figura 2, a norma ISO/IEC 25010 fornece uma estrutura abrangente para avaliar a qualidade de sistemas de software, incluindo microsserviços. A partir desta base, é possível aplicar métricas específicas, que permitem mensurar aspectos como desempenho, coesão e manutenibilidade, facilitando a avaliação prática e detalhada da qualidade de um software.

2.5 Métricas

Adotar métricas de software específicas é útil para identificar problemas logo no início e melhorar a eficiência operacional (ERDEI; TOKA, 2023). As métricas desempenham um papel essencial na tomada de decisões estratégicas, como a avaliação da complexidade do código-fonte, cobertura do código-fonte e densidade de defeitos. Fornecer dados objetivos sobre a qualidade e o desempenho do software permite que os gerentes avaliem o progresso do projeto, realoquem recursos conforme necessário e alinhem os esforços de desenvolvimento com as metas organizacionais de longo prazo (KESSEL; ATKINSON, 2015). Esta visão abrangente do ecossistema de software permite que as empresas identifiquem oportunidades de melhoria e

inovação.

Dentre os benefícios das métricas, destacam-se a capacidade de definir *benchmarks* e metas realistas para projetos de desenvolvimento (MEGARGEL; SHANKARARAMAN; WALKER, 2020). Ao definir padrões de desempenho mensuráveis, as equipes de desenvolvimento podem monitorar seu progresso ao longo do tempo e tomar medidas corretivas quando necessário (ZHOU et al., 2023). Esta característica aumenta a transparência e a responsabilidade dentro da equipe e ajuda a garantir que os projetos sejam concluídos no prazo e dentro do orçamento (HASSAN; BAHSOON; BUYYA, 2022a). Ao comparar o desempenho atual com resultados anteriores ou melhores práticas do setor, as empresas podem identificar áreas de oportunidade para melhoria contínua (ZHOU et al., 2023) e inovação (HASSAN; BAHSOON; BUYYA, 2022a).

Métricas de software são ferramentas e catalisadores para promover uma cultura de qualidade e excelência dentro de organizações (MEGARGEL; SHANKARARAMAN; WALKER, 2020) e academia (VALE et al., 2022). Ao enfatizar a importância da medição e análise objetivas, tais métricas inspiram uma abordagem proativa para o gerenciamento de qualidade, onde a prevenção de defeitos e a melhoria contínua são prioridades vitais (OUMOUSSA; SAIDI, 2024). Este aspecto reduz os custos associados à correção de defeitos após o desenvolvimento e fortalece a reputação da empresa e a satisfação do cliente a longo prazo (VALE et al., 2022). Ao adotar uma abordagem baseada em métricas para o desenvolvimento de software, as empresas podem alcançar maior eficiência, qualidade e sucesso organizacional (DAUD; KADIR, 2015). Ao integrar as diretrizes da Norma ISO/IEC 25010 com métricas de software, as empresas e a academia podem obter uma visão holística da qualidade de seus microsserviços, identificando áreas para melhoria e garantindo que atendam aos padrões estabelecidos.

Métricas de software fornecem métodos quantitativos para pontuar a qualidade do software (SOARES; FRANÇA, 2016), possibilitando monitorar seu desempenho, otimizando sua depuração, gerenciamento de recursos e consequentemente melhorando o desempenho organizacional (ZHOU et al., 2023). As métricas podem identificar lacunas e deficiências que, com este monitoramento, seriam possíveis de serem descobertas (VALE et al., 2022). No entanto, um software de baixa qualidade pode impactar negativamente os usuários que o utilizam (RODRIGUEZ et al., 2023) e, para as empresas, pode prejudicar significativamente sua reputação (MEGARGEL; SHANKARARAMAN; WALKER, 2020). Portanto, ao analisar as práticas da indústria e identificar pontos positivos e negativos (OUMOUSSA; SAIDI, 2024), perdas significativas podem ser evitadas, e os desenvolvedores podem ser motivados (ZHOU et al., 2023) ao destacar processos bem-sucedidos. No entanto, é necessário trabalhar em diversas métricas para rastrear diferentes ângulos possíveis de anomalias de serviço (VALE et al., 2022), ajudando na rastreabilidade e na qualidade dos microsserviços. Neste contexto, Modelos de Linguagem de Grande Escala, ou *Large-Language Models* (LLMs), podem ser valiosos ao sugerir métricas relevantes e analisar dados do sistema. Os LLMs são capazes de identificar padrões, recomendar ajustes nos microsserviços e facilitar o monitoramento, contribuindo para a melhoria

do desempenho e a detecção de anomalias de forma mais eficiente (HASSAN et al., 2024).

2.6 *Large Language Models* (LLMs)

LLMs são modelos de linguagem de grande escala de um tipo de modelo de aprendizado de máquina desenvolvido para entender e gerar linguagem natural de forma avançada para aprender representações contextuais de palavras, frases e documentos inteiros (HASSAN et al., 2024). Estes modelos são capazes de realizar uma ampla gama de tarefas linguísticas com precisão e adaptabilidade. A capacidade da LLM de entender e gerar texto com base em contextos complexos permite que execute funções como a análise de texto e a geração automática de conteúdo (OUYANG et al., 2023).

Na engenharia de software, LLMs são empregados para a geração automática de código, onde podem criar e sugerir blocos de código com base em descrições textuais fornecidas pelos desenvolvedores (MEGARGEL; SHANKARARAMAN; WALKER, 2020). Os LLMs facilitam a documentação automatizada, produzindo documentação técnica e explicativa de forma detalhada e consistente, e oferecem suporte ao desenvolvimento de software por meio de assistentes virtuais que ajudam a resolver problemas e a otimizar processos (HASSAN et al., 2024).

A arquitetura dos LLMs é composta por fluxos de entrada e saída que são essenciais para sua integração em processos de desenvolvimento e sistemas mais amplos. Estes fluxos aumentam a escalabilidade dos modelos em ambientes de produção, incluindo arquiteturas de microsserviços (VASWANI et al., 2017; BROWN et al., 2020). A capacidade dos LLMs de se integrar de forma eficaz em tais arquiteturas contribui para a evolução contínua das práticas e ferramentas de desenvolvimento de software, tornando-os parte do ecossistema tecnológico moderno (OUYANG et al., 2023). Uma abordagem avançada que utiliza esta integração é a Geração Aumentada de Recuperação, ou *Retrieval Augmented Generation* (RAG), que combina LLMs com técnicas de recuperação de documentos externos. Este aspecto melhora a precisão das respostas, permitindo uma interação mais eficiente entre sistemas de software e dados externos (SALTON; WONG; YANG, 1975).

2.7 Geração Aumentada de Recuperação (RAG)

RAG é uma abordagem de arquitetura avançada que combina LLMs com técnicas de recuperação de documentos externos para aprimorar a precisão das respostas geradas (SALTON; WONG; YANG, 1975). A RAG opera ao integrar um módulo de recuperação de documentos com um modelo gerador de linguagem, permitindo que o modelo acesse e utilize informações contextuais para responder a consultas de forma mais precisa, que possibilita uma interação com sistemas baseados em aprendizado de máquina. Interfaces projetadas facilitam a interação com o sistema RAG, permitindo que os usuários ajustem e refinem as consultas e resultados, otimizando

a geração de respostas (DUDLEY; KRISTENSSON, 2018).

A implementação do RAG requer técnicas avançadas de gerenciamento e recuperação de dados, para o acesso e utilização de grandes volumes de informações (SALTON; WONG; YANG, 1975). A capacidade de realizar buscas em grandes conjuntos de dados é característica do desempenho do RAG, permitindo que as informações recuperadas sejam relevantes e úteis para o modelo gerador (PAN; WANG; LI, 2024).

O RAG está fortemente associada ao Modelo de Espaço Vetorial, ou *Vector Space Model* (VSM), que fornece uma estrutura matemática para a indexação e recuperação de documentos (DUDLEY; KRISTENSSON, 2018). O VSM permite a representação de documentos e consultas como vetores. Esta abordagem faz parte do funcionamento do RAG, pois permite a recuperação de documentos com base na similaridade vetorial, suportando a combinação de informações externas com a geração de texto pelos LLMs (SALTON; WONG; YANG, 1975). No RAG, os LLMs geram respostas a partir de documentos relevantes recuperados, e a comparação da similaridade entre os vetores ajuda a identificar os documentos mais importantes. Esta abordagem, ao utilizar o VSM, torna as respostas mais precisas (SALTON; WONG; YANG, 1975).

2.7.1 Modelo de Espaço Vetorial (VSM)

O Vector Space Model (VSM) é uma técnica que representa documentos ou palavras como vetores em um espaço multidimensional, onde cada dimensão reflete termos ou características (SALTON; WONG; YANG, 1975). Amplamente usado em busca de informações e análise de similaridade textual, o VSM calcula a proximidade entre vetores, como por exemplo com a distância *cosine*. Uma das formas mais adotadas para análise de similaridade textual é a distância *cosine*, que mede a diferença entre dois vetores como o cosseno do ângulo entre eles. Quanto menor o valor do cosine, mais semelhantes são os documentos ou palavras representados (MANNING; RAGHAVAN; SCHüTZE, 2008). Em sistemas com microserviços, o VSM é útil para busca semântica e classificação (SALTON; WONG; YANG, 1975).

***Embeddings* ou vetores**: No contexto de bancos de dados vetoriais, vetores são representações que capturam propriedades, contextos e relações entre dados de forma compacta. Eles transformam informações complexas em formas mais simples, permitindo a análise e recuperação de dados(SALTON; WONG; YANG, 1975). Quando estes vetores são criados para representar características semânticas ou estruturais dos dados, eles são conhecidos como *embeddings* vetoriais (TORAMAN et al., 2023). Tais *embeddings* desempenham um papel fundamental em diversas aplicações de aprendizado de máquina e processamento de linguagem natural, pois permitem que os modelos compreendam e manipulem dados de maneira eficiente, proporcionando uma base para tarefas complexas, como recuperação de informações e análise semântica (ALIERO et al., 2023).

***Embeddings* de Entrada**: VSM mapeia documentos e consultas para um espaço vetorial,

permitindo a medição da similaridade entre eles. Esta técnica é útil para tarefas de recuperação de informação em sistemas de software, como a busca por código similar ou a identificação de documentação relevante (SALTON; WONG; YANG, 1975) conforme representado na Figura 3.

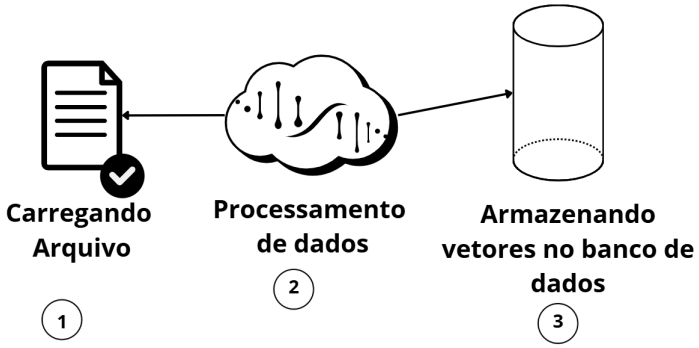

Figura 3 – *Embeddings* de Entrada

Etapa 1 - Carregando arquivos: Nesta etapa, os documentos são coletados e armazenados em seu formato bruto. Isto inclui textos não processados que serão analisados posteriormente (SALTON; WONG; YANG, 1975).

Etapa 2 - Processamento de dados: O processamento de dados textuais envolve etapas como remoção de *stop words* (palavras comuns removidas por não agregarem significado relevante, como "e" e "o") (CHANDA; PAL, 2023), normalização (transformação dos textos em um formato uniforme, como conversão para minúsculas) (ALIERO et al., 2023), tokenização (divisão do texto em unidades menores, ou tokens, para facilitar a análise) (TORAMAN et al., 2023) e *stemming* (redução das palavras às suas raízes para agrupar significados semelhantes) (SINGH; GUPTA, 2016), enfim, os textos são, então processados para que os embedings, representações numéricas dos textos, sejam gerados, para que possa ser armazenadas. O conceito de *score* é utilizado para avaliar a relevância dos textos, com métricas como a frequência de termos e modelos como *Term Frequency-Inverse Document Frequency* (TF-IDF), que destacam palavras significativas em contextos específicos. Estes *scores* servem para tarefas de classificação, recuperação de informações e recomendação de conteúdos, permitindo a identificação e priorização de informações que se aproximam em seus conceitos (MOURA; ROCHA; SOARES, 2024).

Etapa 3 - Armazenando vetores no banco de dados: Os textos processados são convertidos em vetores, e esses vetores são armazenados em um banco de dados vetoriais. Cada vetor representa um documento no espaço vetorial baseado em características como a frequência de termos ponderada (TF-IDF) (PAN; WANG; LI, 2024).

***Embeddings* de Saída:**

A consulta do usuário, deve ser processada, de forma similar ao embedings de entrada, de forma a gerar vetores, estes, são então utilizados para o calculo de similaridade no VSM,

normalmente por meio da distancia*cosine* (SALTON; WONG; YANG, 1975). No caso de alguns bancos vetorias, a busca é otimizada para lidar com grandes volumes de dados vetoriais e realizar consultas de forma eficiente, mesmo com grande complexidade de dados (PAN; WANG; LI, 2024).

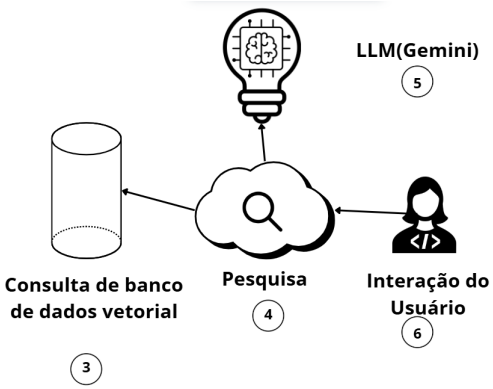

Figura 4 – *Embeddings* de Saída

Etapa 3 - Consulta de Banco de Dados Nesta fase, consultas são convertidas em vetores e comparadas com os vetores armazenados no banco de dados. A similaridade entre os vetores é calculada usando métricas como similaridade do *cosine* para determinar a relevância dos documentos (PAN; WANG; LI, 2024).

Etapa 4 - Pesquisa: A pesquisa está relacionada a A interface onde há interação dos usuários com a mesma, onde enviam consultas e recebem resultados. Esta etapa envolve a implementação de uma interface de usuário intuitiva e eficiente para interagir com o modelo vetorial (DUDLEY; KRISTENSSON, 2018).

Etapa 5 - LLM: A integração de LLMs pode melhorar a recuperação de informações ao fornecer entendimento semântico e respostas contextualmente relevantes. Os LLMs ajudam a interpretar e processar consultas complexas e a gerar respostas mais precisas (OUYANG et al., 2023).

Etapa 7 - Interação do Usuário: A interação com o usuário envolve a apresentação dos resultados de busca e a coleta de feedback para ajustar e melhorar o desempenho do sistema, incluindo a visualização dos resultados e a capacidade de refinar as consultas (PAN; WANG; LI, 2024).

Para entender melhor o impacto do uso do VSM, a Subseção a seguir apresentará cálculos para rastrear suas métricas. Tais cálculos fornecerão uma visão mais profunda da aplicação do modelo de espaço vetorial. Essas métricas ajudam a quantificar vários aspectos de precisão, substituto de avaliação bilíngue.

2.7.2 Métricas de Avaliação de VSM

Precisão: Salton *et al.* (SALTON; WONG; YANG, 1975) medem a proporção de documentos relevantes entre os documentos recuperados por meio da equação a seguir. onde R representa o conjunto de documentos relevantes, A o conjunto de documentos recuperados, e $|R \cap A|$ a interseção entre esses dois conjuntos, indicando o número de documentos que são tanto relevantes quanto recuperados. Onde, temos as seguintes variáveis:

- P: a intersecção entre os R e A, dividido por A, indicando o número de documentos que são, tanto relevantes quanto recuperados
- R: Soma de todos os documentos relevantes
- A: Soma de todos os documentos recuperados

:

$$P = \frac{|R \cap A|}{|A|}$$

Similaridade Cosine: É uma métrica usada para medir a semelhança entre dois vetores, frequentemente aplicada em tarefas de recuperação de informações, como a busca por documentos relevantes. Em vez de comparar os vetores pela sua magnitude (tamanho), a similaridade *cosine* avalia o ângulo entre eles, focando na direção. Isso significa que dois vetores com a mesma orientação, mesmo que tenham magnitudes diferentes, serão considerados semelhantes. Esse cálculo é amplamente utilizado em modelos baseados em vetores para identificar a relevância de documentos em relação a uma consulta, facilitando a comparação entre textos ou conjuntos de dados de forma eficiente(MANNING; RAGHAVAN; SCHüTZE, 2008).

Onde:

A e **B** são vetores que representam os documentos ou os termos.

A · **B** é o produto escalar (ou produto interno) entre os vetores **A** e **B**. Este é calculado como a soma dos produtos dos termos correspondentes nos dois vetores:

$$\mathbf{A} \cdot \mathbf{B} = \sum_{i=1}^{n} A_i B_i$$

$\|\mathbf{A}\|$ e $\|\mathbf{B}\|$ são as normas dos vetores, que representam o comprimento dos vetores **A** e **B**. A norma de um vetor é dada por:

$$\|\mathbf{A}\| = \sqrt{\sum_{i=1}^{n} A_i^2}$$

Assim, a similaridade *cosine* mede a orientação entre os dois vetores, não levando em conta a magnitude (tamanho) dos vetores, apenas a direção (MANNING; RAGHAVAN; SCHüTZE, 2008).

2.8 Trabalhos Relacionados

As ideias foram estruturadas por meio da Norma ISO/IEC 25010, analisando os artigos selecionados sobre quais métricas eram mais utilizadas para possibilitar um acompanhamento de microsserviços. Acoplamento, coesão, escalabilidade, complexidade e desempenho são exemplos de métricas que foram inter-relacionadas com as características de qualidade do padrão escolhido.

Li *et al.*, (LI et al., 2022) conduziram experimentos para analisar métricas de desempenho, coesão, acoplamento e redundância de código. Foi possível compreender a arquitetura das aplicações, por meio de grafos, onde foi extrair informações de sistemas e entidades monolíticas e identificar candidatos a se tornarem microsserviços. Os autores mencionaram pontos fortes a serem avaliados, como a independência de interferência humana e a descoberta de defeitos do sistema.

Hassan (HASSAN; BAHSOON; BUYYA, 2022b), realizaram experimentos que auxiliam na decisão de adaptação da granularidade ao reconhecer a escalabilidade. Eles contribuíram para um catálogo de métricas para microsserviços focado em escalabilidade. Este catálogo foi compilado com base no estado real da granularidade. Os autores afirmaram que precisarão de mais tempo para investigar se todas as métricas foram aplicadas corretamente, sem redundância, e apresentar como validação dessas métricas a aplicação do catálogo na indústria ou entrevistas.

Auer *et al.*, (AUER et al., 2021) também apoiaram a migração para microsserviços com base em métricas. No entanto, essas informações foram coletadas por meio de entrevistas. Os entrevistados tinham experiência em migração para microsserviços e responderam a um questionário informando quais métricas e características foram utilizadas. As informações coletadas na pesquisa tinham como objetivo auxiliar as empresas a entender se é necessário migrar para microsserviços ou manter a arquitetura monolítica.

Erdey (ERDEI; TOKA, 2023) gerenciaram recursos de forma eficiente em configurações de microsserviços nativos da nuvem, aproveitaram ao máximo os recursos de computação, economizaram despesas operacionais e evitaram alocações desnecessárias, além de aumentar o desempenho, a qualidade e a adaptabilidade dos serviços. Eles discutiram técnicas que poderiam ajudar nesses desafios, como modelos preditivos que oferecem otimização focada em propor melhorias em sistemas de nuvem.

Ren (REN; BARRETT; DAS, 2020), exploraram como os princípios e técnicas de gamificação podem ser efetivamente aplicados na Engenharia de Software para abordar vários desafios, desde o engajamento e motivação dos funcionários, até a qualidade do software e a satisfação do usuário final.

Zhou *et al.* (ZHOU et al., 2023) buscaram entender a implementação real e os desafios enfrentados ao adotar a arquitetura de microsserviços em ambientes industriais. Eles investigaram os desafios e dificuldades que as organizações enfrentam, o impacto nos processos de desenvolvimento de software, lições aprendidas e melhores práticas emergentes. O objetivo era fornecer uma visão aprofundada do estado atual da prática em relação às arquiteturas de microsserviços na indústria de software, destacando tendências, pontos problemáticos comuns e áreas para melhorias futuras.

Cerny *et al.* (CERNY et al., 2023) abordaram o desafio de identificar, categorizar e detectar padrões negativos (anti-padrões) e "maus cheiros" em arquiteturas de microsserviços. Eles se concentraram na abordagem de microsserviços como um paradigma de arquitetura de software que decompõe grandes aplicativos monolíticos em componentes menores e mais independentes. Embora reconheçam os benefícios desta abordagem, eles também reconheceram que ela pode introduzir novos desafios e problemas específicos. O objetivo da pesquisa dos autores era ajudar desenvolvedores e arquitetos de software a entender melhor essas questões e adotar práticas que melhorem a qualidade do software baseado em microsserviços.

Devlin *et al.* (DEVLIN et al., 2019) desenvolveram um modelo de linguagem inovador que utiliza treinamento bidirecional, permitindo uma melhor compreensão do contexto das palavras. Eles introduziram duas tarefas principais para o pré-treinamento: a previsão de palavras ocultas e a verificação de sequência lógica entre frases. O BERT (Bidirectional Encoder Representations from Transformers) alcançou resultados de ponta em *benchmarks* como *General Language Understanding Evaluation* (GLUE) e *Stanford Question Answering Dataset* (SQuAD), superando modelos anteriores. Ademais, demonstrou a eficácia da transferência de aprendizado, permitindo que conhecimentos pré-treinados fossem aplicados a tarefas específicas com ajuste fino. Essas inovações estabeleceram novos padrões em Processamento de Linguagem Natural.

Moura (MOURA; ROCHA; SOARES, 2024), desenvolveram uma ferramenta que utiliza técnicas de recuperação de informações para recomendar padrões de microsserviços. A ferramenta foi projetada para ajudar desenvolvedores a identificar e selecionar padrões apropriados com base em dados contextuais e requisitos específicos do projeto. A pesquisa mostrou que a aplicação desta abordagem melhora a eficiência do desenvolvimento, proporcionando recomendações mais relevantes e personalizadas. Ao integrar algoritmos de recuperação de informações com uma abordagem centrada no domínio, a ferramenta oferece suporte na tomada de decisões, promovendo a adoção eficaz de padrões de microsserviços e facilitando o design de arquiteturas desenvolvidas.

Esta pesquisa se diferencia das anteriores por focar na aplicação prática de métricas

específicas no desenvolvimento de microsserviços e seu impacto na qualidade do software, utilizando LLM, com técnica RAG e com banco de dados vetoriais, para sugerir métricas adaptadas a problemas reais enfrentados pelas equipes. Ao buscar um alinhamento entre teoria e prática, o estudo de caso avalia a eficácia de tais métricas em cenários do dia a dia, proporcionando recomendações que são validadas na prática. O registro das experiências dos usuários ao interagir com a ferramenta permite coletar dados sobre sua eficácia, criando um ciclo de feedback que aprimora continuamente a solução. Esta abordagem centrada no usuário oferece *insights* para melhorar a qualidade e eficiência no desenvolvimento de microsserviços.

3

Design Science Research e Execução

A metodologia deste projeto tem como objetivo orientar o desenvolvimento da pesquisa sobre métricas aplicadas a microsserviços. Para isso, foi adotado o *Design Science Research* (DSR), uma abordagem que integra pesquisa e inovação no campo dos Sistemas de Informação (HORITA E.A. et al., 2020; HEVNER et al., 2004). Esta abordagem possibilita consolidar o conhecimento proveniente de investigações anteriores sobre métricas para microsserviços, além de facilitar a criação de artefatos (HORITA E.A. et al., 2020) que visam aprimorar a avaliação da qualidade de softwares existentes. Os artefatos são elementos utilizados para documentar o conhecimento adquirido nas pesquisas, apoiando, assim, o desenvolvimento de novos produtos organizacionais e/ou computacionais (HEVNER et al., 2004).

A estrutura conceitual com os elementos essenciais para auxiliar no entendimento, criação e avaliação desta pesquisa é apresentada na Figura 5.

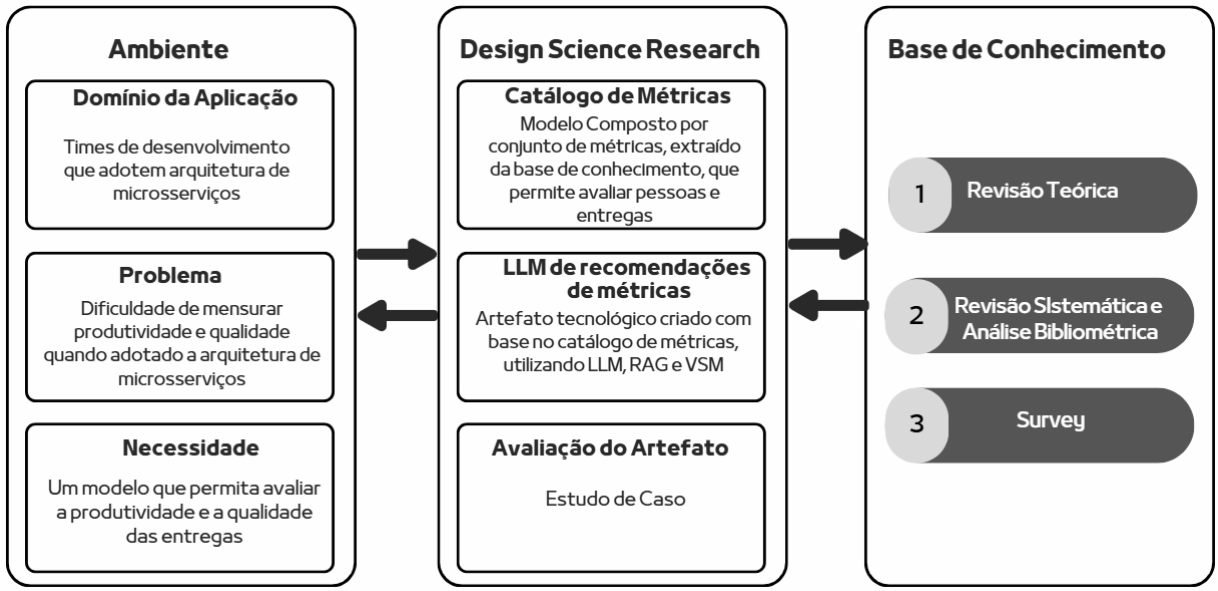

Figura 5 – Ciclos do *Design Science Research*

O ambiente é composto por: Domínio da aplicação, que representa o ambiente, determinando o fenômeno de interesse composto por pessoas, organizações, tecnologias, objetivos e atividades. O alvo desta pesquisa, como domínio de aplicação, são os times de desenvolvimento que adotem arquitetura de microsserviços. Este domínio fornece as necessidades de negócio que irão nortear a definição do problema de pesquisa, que é a dificuldade em mensurar produtividade e qualidade de software quando da adoção de microsserviços. Assim, surge a necessidade de se criar um modelo que permita avaliar a produtividade e a qualidade das entregas. Do outro lado do ciclo da pesquisa pode-se observar a base de conhecimento que deve prover toda fundamentação teórica para a condução das pesquisas que são compostas pelas revisões: intencional, sistemática e análise bibliométrica. Ainda, há a pesquisa de opinião. Esta base é elemento fundamental para compor o conhecimento aplicável durante a pesquisa.

Dessa forma, o DSR utiliza as necessidades contextuais do meio prático na definição de problemas, que foram solucionados empregando artefatos construídos e avaliados com conhecimento científico. As contribuições propostas a partir da DSR serão relevantes e importantes somente quando atenderem às necessidades do domínio de aplicação e também apoiarem na condução de novas pesquisas. Os trabalhos relacionados usaram aplicativos, experimentos e entrevistas para coletar informações sobre as métricas. Nesse sentido, por meio da metodologia proposta neste novo estudo, foram coletadas as estatísticas mais citadas, com base nas necessidades atuais para descobrir como garantir a qualidade dos microsserviços inter-relacionados com a Norma ISO/IEC 25010.

Para desenvolver a pesquisa, foram seguidos os passos metodológicos apresentados na Figura 6, que incluem uma revisão teórica e uma revisão sistemática com análise bibliométrica.

Para validar estes estudos, foi realizada uma pesquisa de opinião em empresas que utilizam microsserviços, analisando o ambiente de aplicação e compreendendo as necessidades específicas de cada contexto. Com base nesta análise, foi elaborado o primeiro artefato: um catálogo de métricas que auxilia gestores e equipes na seleção de métricas para gerenciar e monitorar a qualidade dos microsserviços. Em seguida, foi implementado o segundo artefato, uma ferramenta que utiliza inteligência artificial para sugerir métricas aos gestores e desenvolvedores, adaptando-se às necessidades da situação relatada. Foi realizado um estudo de caso onde o produto foi testado e validado na indústria, inicialmente nas empresas entrevistadas e expandido para desenvolvedores e gestores que trabalham com microsserviços.

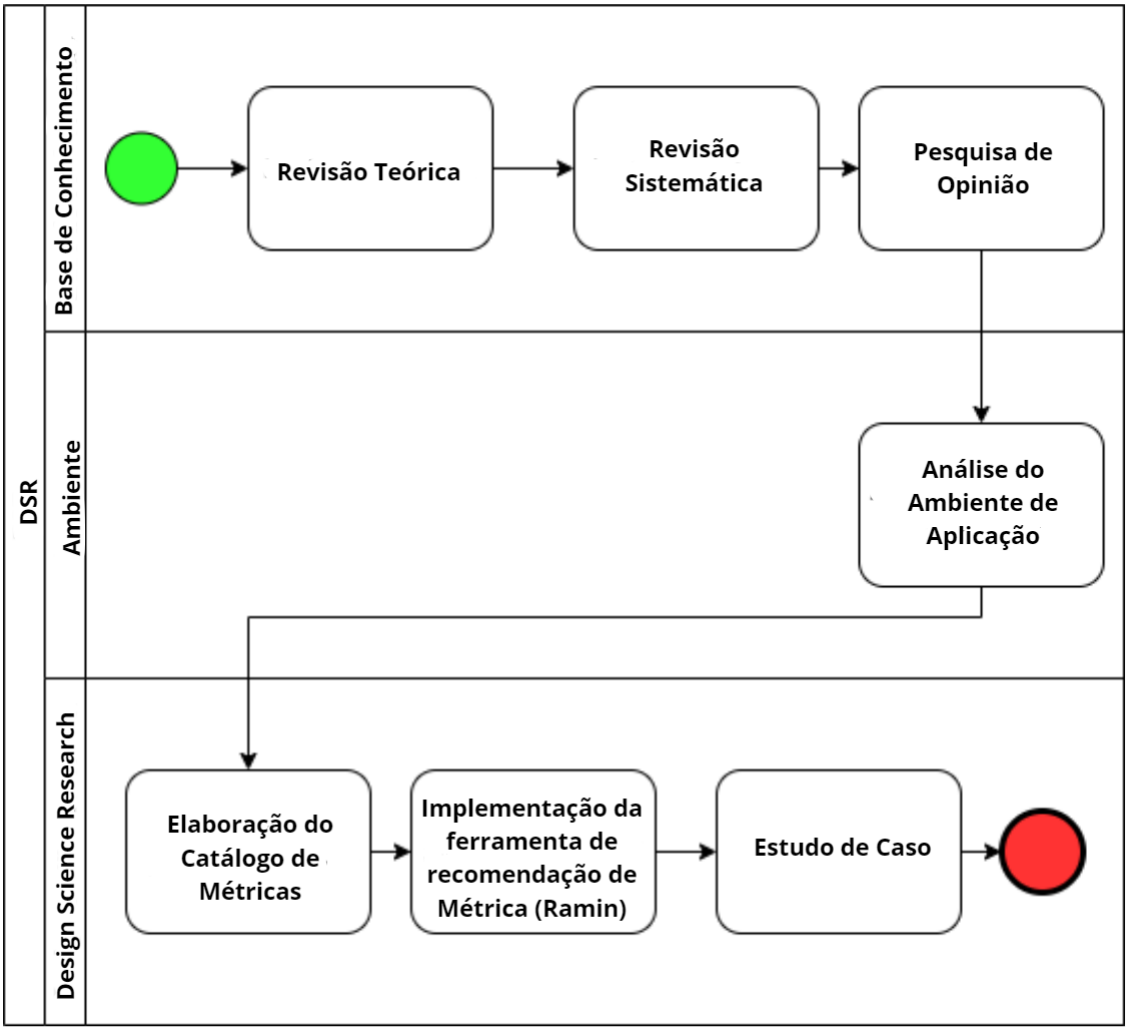

Figura 6 – Passos metodológicos

3.1 Revisão Teórica

Existem métricas para avaliação de aplicações SOA, porém a arquitetura de microsserviços necessita de alguns ajustes (PULPARAMBIL et al., 2018a; RAJ; SADAM, 2021) devido aos seus princípios como coesão, escalabilidade e baixo acoplamento. Inicialmente foi realizada uma

revisão com a intenção de identificar estas métricas. A partir da identificação das informações coletadas, esta revisão deu origem à revisão sistemática e suas métricas também são utilizadas na elaboração do survey (pesquisa de opinião).

A revisão teórica consistiu na coleta e análise de artigos focados em métricas, na Norma ISO/IEC 25010, e arquiteturas de software, com ênfase em microsserviços como arquitetura principal. O critério de seleção dos artigos envolveu a busca por estudos que abordassem a aplicação e avaliação de métricas em contextos de arquitetura de microsserviços, bem como a comparação com arquiteturas anteriores, como SOA e monolíticas. Adicionalmente, para garantir a relevância e a atualidade das informações sobre microsserviços, foram considerados apenas artigos a partir de 2015, ano em que essa arquitetura começou a ganhar destaque na indústria. A revisão procurou identificar e compilar evidências sobre a eficácia dessas métricas na avaliação da qualidade de software, assim como entender as transições entre as diferentes abordagens arquiteturais. Dessa forma, foram incluídos artigos que discutem tanto a Norma ISO/IEC 25010, que fornece uma base para a avaliação da qualidade de software, quanto aqueles que exploram a aplicação prática das métricas em diversos tipos de arquitetura, garantindo uma visão abrangente e contextualizada sobre o tema.

3.2 Revisão Sistemática e Análise Bibliométrica

Nesta etapa, foram realizadas atividades para estabelecer um protocolo de revisão: 1. Estabelecimento do objetivo; 2. Definição das questões de investigação; 3. Definição dos critérios PICO(População, Intervenção, Comparação e Resultado); 4. Definição da Estratégia de Pesquisa; 5. Definição dos critérios de inclusão e exclusão.

Inicialmente, o objetivo da revisão sistemática foi definido usando parte do modelo *Goal-Question-Metric* (GQM) (MASHIKO; BASILI, 1997; SOLINGEN et al., 2002) : **Analisar** adoção de métricas, **com o objetivo de** caracterizar, **no que diz respeito à** arquitetura como serviço e microsserviços; **do ponto de vista** dos pesquisadores; **no contexto** da pesquisa teórica e aplicada.

Foi proposto analisar e então caracterizar as métricas aplicadas a microsserviços na literatura científica. A caracterização nesta pesquisa está relacionada ao perfil bibliométrico das publicações. Duas perguntas de pesquisa e uma sub-questão são estabelecidas para este estudo:

Pergunta 1: Como são organizadas as publicações sobre o assunto?

Pergunta 2: Quais são as principais métricas apresentadas nos artigos?

Sub-Pergunta 2.1: Como as métricas estão sendo abordadas e como as métricas se relacionam com a Norma ISO/IEC 25010?

A seleção foi feita considerando as métricas mais citadas e é proposta aqui considerando o objetivo da pesquisa e fornecendo uma compilação sistemática de dados sobre métricas aplicadas

a microsserviços. Os critérios PICO são determinados (KITCHENHAM, 2004), (KEELE et al., 2007). Foi usado os atributos População, Intervenção e Resultado (PIO) para esta revisão conforme Tabela 2. Não foi utilizado Comparação, pois não é o objetivo desta revisão. Assim, esta pesquisa está dividida em três etapas: Planejamento, Condução e Análise dos Resultados.

Critérios PIO	**Descrição**
População **Intervention** **Outcome**	metric OR metrics microservices OR SOA OR software as a service applicability OR characteristics OR factors OR standard

Tabela 2 – Critérios PICO

Com base nos dados do critério PIO, uma *string* de busca usando operadores booleanos foi elaborada disponível na Tabela 3.

String de Busca
(‘microservice’ OR ‘microservices’ OR ‘SOA’ OR ‘software as a service’) AND (‘metric’ OR ‘metrics’) AND (‘type’ OR ‘applicability’ OR ‘characteristics’ OR ‘factors’ OR ‘standard’)

Tabela 3 – *String* de Busca

ACM Digital Library, *IEEE*, *Web of Science*, *ScienceDirect* e *Scopus* foram selecionados como fontes de dados para a pesquisa devido à sua relevância e abrangência na área de ciência e tecnologia. Essas plataformas são amplamente reconhecidas por fornecer acesso a artigos acadêmicos, conferências e outros materiais científicos de alta qualidade. Entre esses, *Scopus* e *Web of Science* foram especificamente utilizados para realizar buscas, visando à análise bibliométrica. Estas duas bases de dados foram escolhidas devido aos seus robustos mecanismos de busca e à capacidade de oferecer informações detalhadas sobre citações, coautorias e outras métricas relevantes para a análise quantitativa do impacto e da rede de colaboração acadêmica.

Execução

Foi possível obter diferentes pontos de vista sobre a pesquisa extraídos dos artigos selecionados, dividindo o resultado em duas partes. Primeiro, uma análise bibliométrica, onde os artigos mais relevantes foram classificados de acordo com requisitos escolhidos para esta seleção, como autores mais citados, nuvem de palavras dos tópicos mais discutidos e produção dos autores ao longo do tempo. Segundo, foram coletadas métricas no contexto de microsserviços que pudessem auxiliar na discussão de padrões de arquitetura de requisitos de software, apresentados neste artigo, para auxiliar pesquisadores que buscam conduzir trabalhos relacionados a este assunto respondendo às questões feitas no planejamento deste estudo.

A busca resultou em um total de 649 artigos, dos quais 75 foram selecionados, conforme ilustrado na Figura 7.

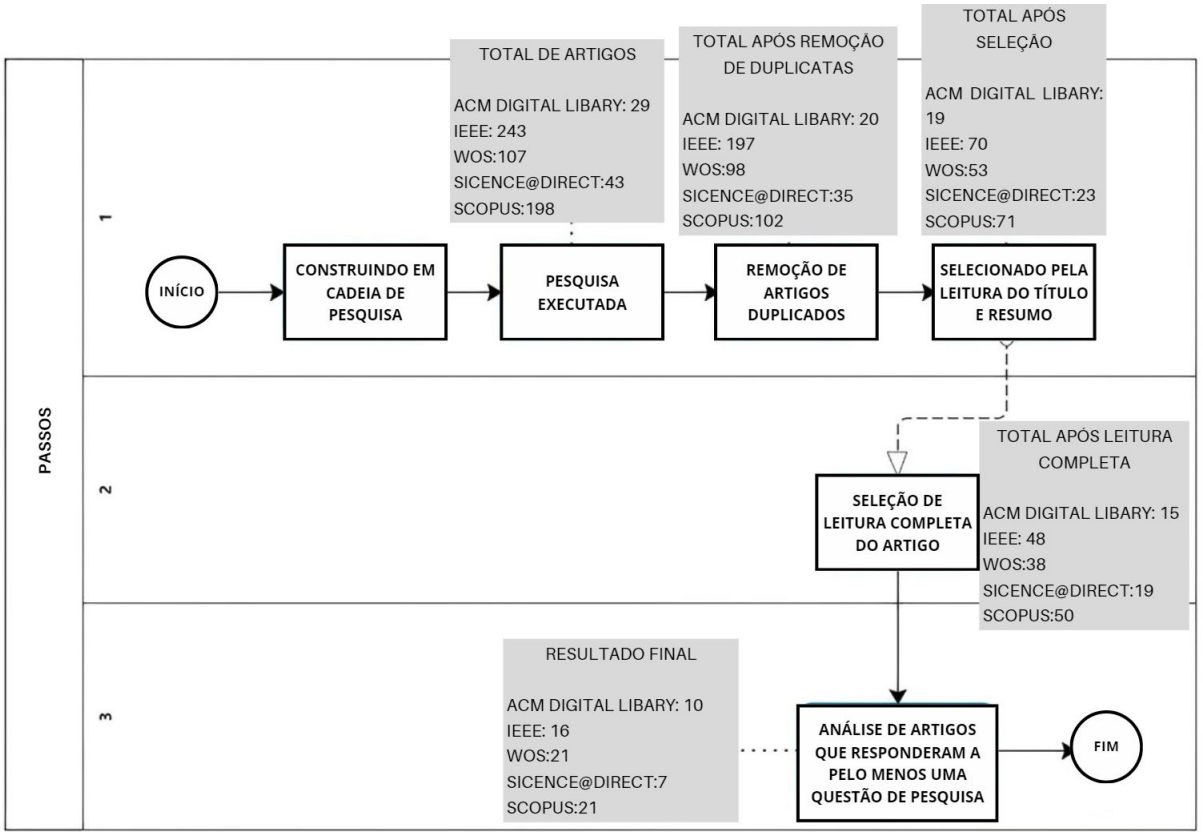

Figura 7 – Processo de seleção

3.3 Survey

Survey (pesquisa qualitativa de opinião) consiste em aplicar um questionário de investigação dividido em duas partes, uma com foco nos líderes/gestores de times e outra com foco nos desenvolvedores, sendo composto por questões objetivas com campo aberto caso a opção da realidade da empresa não esteja listada nos itens. O tempo estimado de resposta foi de 8 minutos. Os participantes foram quatro empresas que adotam em todos ou em parte dos projetos a arquitetura de microsserviços. Com o intuito de recolher informações específicas e detalhadas, o survey aplicado é categorizado como descritivo. As questões foram projetadas para investigar e analisar informações a respeito da gestão de métricas em ambientes que utilizam microsserviços, motivados pelo interesse em melhorias na qualidade do software gerido e desenvolvido por eles. As questões de pesquisa deste estudo são:

Para Gestores:

- Q1) Como a gerência avalia a qualidade dos serviços desenvolvidos?
- Q2) Como você mede as entregas?
- Q3) Como você mede a qualidade das entregas?

- Q4) Quais os maiores desafios na adoção da arquitetura de microsserviços?
- Q5) Como a arquitetura de microsserviços influenciou a qualidade do software?
- Q6) Quais ações os profissionais de software tomam para garantir a escalabilidade dos microsserviços?
- Q7) O que você considera uma prioridade para analisar o desempenho do software?
- Q8) Quais métricas você usa para gerenciar a responsividade de um serviço?
- Q9) Como você monitora a satisfação do usuário com o software usando a arquitetura de microsserviços?

Para Desenvolvedores:

- Q1) Qual sua maior motivação em desenvolver software com arquitetura de microsserviços?
- Q2) Quais são seus maiores desafios no desenvolvimento de microsserviços no dia a dia?
- Q3) Quais princípios norteiam seu desenvolvimento de microsserviços?
- Q4) Quais pontos devem ser melhor gerenciados nos microsserviços da sua empresa?
- Q5) Qual seria o ambiente de desenvolvimento ideal?
- Q6) Qual linguagem você adotou para desenvolvimento de microsserviços?

Execução Foram enviados por e-mail dois questionários para quatro empresas que utilizam microsserviços em sua arquitetura de desenvolvimento de software. Um para gestor[1] e outro para desenvolvedor[2]. O questionário recebeu respostas de 9 de maio a 15 de agosto de 2023. Foram coletadas ao total, 33 respostas, sendo 7 de gestores e 26 de desenvolvedores.

3.4 Estudo de Caso

O estudo de caso visa validar o artefato produzido com base nas etapas anteriores. Assim, o objeto deste estudo de caso é o artefato de *dashboard*, formalizado baseado no modelo GQM (SOLINGEN et al., 2002): **objetivo** avaliar, **com propósito** de efetividade, **na perspectiva** dos gestores e desenvolvedores, **no contexto** de microsserviços.

As questões de pesquisa aplicadas aos estudos foram:

Q1. As Métricas aplicadas a microsserviços conseguem garantir melhor acompanhamento de software com esta arquitetura?

[1] <https://docs.google.com/forms/d/1MoYaWkuWQpBal7zd75LBOyZnWA6mKU5-M_jtU0DJhyA/prefill>
[2] <https://docs.google.com/forms/d/1qsPFOBDFL49DCP5bMo-pMXslg5N16KP3wM_hWkoiNRQ/prefill>

Q2. As empresas que utilizam microsserviços acompanham suas métricas? Tal resultado está alinhado com a pesquisa acadêmica?

Durante o estudo de caso, foram coletados dados reais relacionados ao funcionamento, qualidade e gerenciamento dos microsserviços em empresa. Esses dados foram analisados em conjunto com as métricas selecionadas, permitindo auxiliar na aplicabilidade e eficácia no contexto específico da organização.

Para isto foi criado o Ramin, um LLM onde é apresentada uma interface que o usuário (desenvolvedores e gestores de microsserviços) podem tirar dúvidas ou relatar problemas sobre situações reais do cotidiano, em um campo único da interface. O software retorna métricas, recomendações, como utilizá-las e também é apresentada a pontuação de cada métrica. Desta forma o usuário terá sugestões de soluções que podem contribuir para refinamento das métricas, proporcionando um catálogo adaptado para a necessidade de cada empresa conforme é mostrado na Figura 8 na descrição do *Toy* 1.

Com o objetivo de testar o LLM antes de enviar para as empresas, foram criados 14 *Toys* problemas. Estes problemas são listados a seguir:

Toy 1: "A equipe de Devops está enfrentando dificuldades para corrigir *bugs* e adicionar novas funcionalidades em um dos seus microsserviços. A complexidade crescente do código está tornando a manutenção cada vez mais difícil e demorada. Qual métrica é possível usar para melhorar a manutenibilidade do microsserviço e simplificar a correção de *bugs* e adição de novas funcionalidades?"

A Figura 8 exibe como os dados são retornados pelo Ramin. Foi utilizado o *Toy* 1 para esta captura.

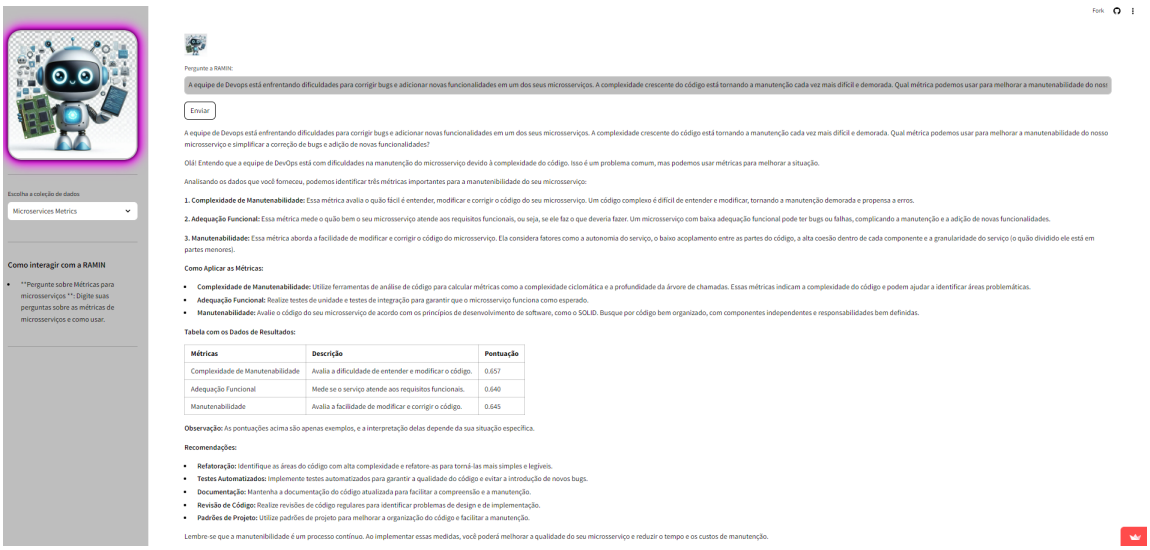

Figura 8 – Ramin

Toy 2: "A equipe no departamento de tecnologia está tendo dificuldades para adicionar

novos requisitos ao seu sistema de pagamentos, pois a estrutura atual está tornando essas mudanças complicadas e trabalhosas. Qual métrica deve ser utilizada para avaliar a capacidade de evolução da nossa arquitetura de microsserviços e facilitar a adição de novos requisitos?"

Toy 3: "A equipe da está enfrentando problemas ao atualizar um dos seus microsserviços, pois as atualizações estão causando efeitos colaterais inesperados em outras partes do sistema, indicando um acoplamento mais forte do que o desejado. Como é possível medir e melhorar o isolamento dos microsserviços para garantir que atualizações em um serviço não afetem os demais?"

Toy 4: "O time de desenvolvimento percebeu que um dos seus microsserviços está se tornando extremamente complexo, dificultando a compreensão e manutenção do código. Qual métrica pode ajudar a identificar e reduzir esta dificuldade crescente nos microsserviços para facilitar o desenvolvimento e a manutenção?"

Toy 5: "A equipe de integração está tentando conectar um novo microsserviço com um sistema legado, mas está encontrando dificuldades devido a problemas de comunicação entre os sistemas. Qual métrica pode ajudar a avaliar e melhorar a interoperabilidade entre novos microsserviços e sistemas legados?"

Toy 6: "A equipe de desenvolvimento está enfrentando problemas de sincronização e dificuldades para atualizar seus microsserviços devido a dependências cíclicas entre eles. Qual métrica pode ajudar a identificar e resolver problemas de dependências cíclicas em microsserviços?"

Toy 7: "O sistema de reservas está sofrendo com falhas em um microsserviço de gerenciamento de disponibilidade, o que está afetando a funcionalidade de toda a aplicação. Qual métrica seria útil para garantir que os microsserviços sejam mais tolerantes a falhas e não impactem o sistema como um todo?"

Toy 8: "A equipe está enfrentando dificuldades para monitorar e gerenciar a grande quantidade de microsserviços em seu sistema, o que está afetando a eficiência operacional. Qual métrica pode ajudar a melhorar a capacidade de gerenciamento e monitoramento dos microsserviços?"

Toy 9: "O microsserviço de relatórios da empresa está com o código altamente entrelaçado, dificultando a manutenção e a atualização. Qual métrica pode ajudar a modularizar melhor o código do microsserviço e facilitar a manutenção?"

Toy 10: "O time de desenvolvimento descobriu que o design de seu microsserviço de gerenciamento de frota não está refletindo corretamente as regras e processos do negócio, o que está causando problemas operacionais. Como é possível melhorar o alinhamento do design do microsserviço com as regras e processos do domínio de negócios?"

Toy 11: "A empresa gostaria de acompanhar a evolução do desenvolvimento do software

com arquitetura de microsserviços visando melhorar o desempenho e controlar o tráfego intenso. Qual métrica pode auxiliar para que possa ser tolerante a falhas, com a possibilidade de crescer horizontal (adicionando mais computadores) e verticalmente (adicionando mémoria, CPU ou disco para melhorar a escala dos serviços)?"

Toy 12: "Gostaria de acompanhar a evolução do desenvolvimento do software com arquitetura de microsserviços que está passando por manutenção em melhorias de maneira ágil. Qual métrica pode auxiliar neste acompanhamento?"

Toy 13: "A análise atual do software em desenvolvimento da empresa indica que a granularidade dos serviços é muito fina, resultando em um nível de serviços reduzido. Isto sugere uma falta de separação de responsabilidades, tornando o sistema menos coeso e autônomo, o que pode dificultar o desenvolvimento e a implantação independente dos serviços. Qual métrica pode solucionar esta situação?"

Toy 14: "Recentemente, na empresa em que trabalho, enfrentamos problemas com serviços sobrepostos e dependentes. Ao decompor domínios complexos e dividir serviços conforme contextos de negócio, vimos melhorias significativas. Como essa abordagem ajudou a melhorar a separação de responsabilidades, a coesão e a reduzir o acoplamento entre os serviços?"

Cada um desses problemas foi associado a métricas capazes de solucioná-los. A Tabela 4 ilustra essas associações de forma detalhada. Observa-se que todas as métricas estão relacionadas com a ISO/IEC 25010, seja por meio de suas características ou subcaracterísticas. A Tabela 4 evidencia que um único problema pode ser abordado por várias métricas, destacando assim a inter-relação entre elas.

Problema	**Métrica**
1	Descentralização, Manutenabilidade, Complexidade
2	Reusabililidade
3	Fraco Acoplamento
4	Coesão
5	Interoperabilidade
6	Sem dependências cíclicas
7	Tolerância de Falhas
8	Capacidade de Gerenciamento
9	Modularidade
10	Modelado em Torno do Domínio de Negócios
11	Escalabilidade
12	Evolutibilidade
13	Granularidade
14	Design Orientado a Domínio

Tabela 4 – Métricas como solução

3.4.1 Execução

Para avaliar a ferramenta, ela foi disponibilizada para testes em uma empresa do setor educacional e em outra da área de segurança da informação. Gestores e desenvolvedores tiveram acesso à ferramenta, assim como profissionais de outras empresas que trabalham com microsserviços. O Ramin disponbiliza um campo para que seja relatado o problema, e, para cada problema relatado, recomenda três métricas para auxiliar na solução. Ao todo, o Ramin foi utilizado por 11 profissionais diferentes. Os participantes não apenas utilizaram a ferramenta, mas também responderam a um formulário para indicar se as recomendações fornecidas atendiam ou não as suas necessidades. As perguntas do formulário estão listadas na Tabela 5.

Descrição da Pergunta
Q1 Em quais cenários específicos de arquitetura de microsserviços a ferramenta demonstra maior eficácia?
Q2 A ferramenta é eficaz para ajudar gestores e desenvolvedores a conhecer e aplicar métricas?
Q3 Existe algo que acredita ser necessário acrescentar como melhoria da ferramenta para melhor atender as necessidades no desenvolvimento e monitoramento de microsserviços?
Q4 Quais vantagens e benefícios a ferramenta oferece para projetos de software que utilizam arquitetura de microsserviços?
Q5 Existem desvantagens ou desafios associados ao uso da ferramenta? Quais?
Q6 Como você avalia a facilidade de uso da ferramenta?
Q7 Consideraria integrar a ferramenta em projetos no futuro? Por quê?
Q8 A ferramenta oferece métricas específicas que você acha necessárias para melhorar o desenvolvimento e a operação de microsserviços? Se não, quais seriam essas métricas?

Tabela 5 – Métricas para avaliação da ferramenta

Após os testes, foram agendadas entrevistas com cinco dos 11 respondentes do formulário, incluindo dois líderes técnicos, um engenheiro de dados, um arquiteto e um desenvolvedor, com o objetivo de coletar informações detalhadas sobre suas experiências com a ferramenta.

3.4.2 Funcionamento do Ramin

O Ramin apresenta um fluxo estruturado para o carregamento, processamento, armazenamento e recuperação de métricas baseados nos conceitos da Seção 2.6 (LLM) e 2.7 (RAG). Cada etapa é projetada para maximizar e gerenciar os dados, desde a inicialização com um catálogo até a interação do usuário. Os detalhes estão nas fases envolvidas e as tecnologias utilizadas. A Figura 9 representa o processo do Ramin.

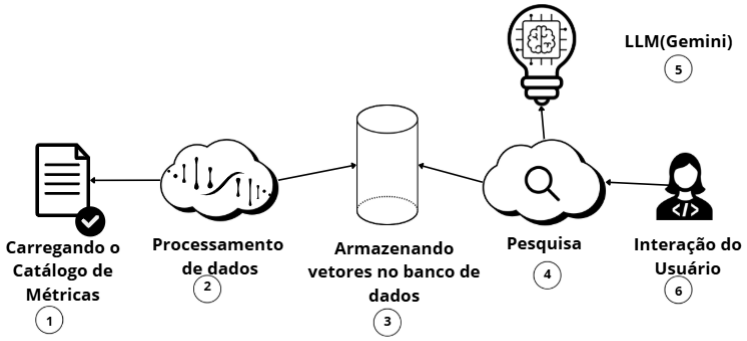

Figura 9 – Processo do Ramin

Carregamento do Catálogo de Métricas: O primeiro passo consiste em carregar um catálogo de métricas e seus detalhes em uma planilha dentro da ferramenta.

Processamento de Dados: No segundo passo, o sistema realiza o processamento dos dados do catálogo utilizando o modelo all-MiniLM-L6-V2, disponível na biblioteca *sentence transformers*. Este modelo é responsável por extrair e gerar vetores representativos dos dados do catálogo.

Armazenamento dos Vetores no Banco de Dados: O terceiro passo envolve o armazenamento dos vetores gerados no banco de dados. Para isso, é utilizado o Qdrant, um banco de dados *open source* que emprega a distância *cosine* para indexação e recuperação eficiente dos vetores.

Pesquisa: No quarto passo, o sistema realiza uma busca no banco de dados Qdrant para encontrar vetores relevantes com base na consulta fornecida.

Geração de Resposta com LLM *Gemini*: O quinto passo utiliza o modelo LLM *Gemini* para processar a consulta de busca. O LLM *Gemini* recebe o texto da consulta, os dados encontrados e um prompt padrão para gerar uma resposta apropriada.

Interação do Usuário: No sexto passo, o usuário interage com a interface do sistema, inserindo perguntas e recebendo respostas baseadas na busca e processamento realizados nos passos anteriores.

Esse fluxo detalha as etapas formais e técnicas envolvidas no carregamento, processamento, armazenamento e recuperação de dados na ferramenta, bem como na interação final com o usuário.

4

Resultados da Base de Conhecimento

Este capítulo apresentou os resultados obtidos dos conceitos que fundamentam o desenvolvimento deste trabalho. Primeiramente, a Revisão Teórica forneceu uma análise abrangente das métricas utilizadas na avaliação de sistemas distribuídos, incluindo tamanho de microsserviços, acoplamento, granularidade e desempenho, destacando seus métodos de cálculo e sua relevância no contexto da otimização de microsserviços. A Revisão Sistemática e Análise Bibliométrica apresentou os principais estudos e tendências da literatura, além de fornecer respostas às questões de pesquisa formuladas, permitindo a identificação de padrões emergentes. O Survey realizado junto à indústria forneceu resultados práticos sobre a adoção e aplicação dessas métricas no ambiente corporativo, possibilitando a comparação entre os conceitos teóricos e as práticas efetivamente adotadas pelas organizações. A análise desses resultados teóricos e práticos proporcionou uma base sólida para as soluções e abordagens discutidas ao longo da dissertação.

4.1 Revisão Teórica

A revisão teórica realizada focou na análise de métricas utilizadas para avaliar a eficácia e a eficiência de sistemas baseados em microsserviços. Visando identificar os métodos mais comuns de mensuração, bem como os cálculos frequentemente empregados na avaliação de características como desempenho, coesão, acoplamento e tamanho (ASIK; SELCUK, 2017).

4.1.1 Métricas de Tamanho do Microsserviço

Existem muitos métodos para medir o tamanho de um microsserviço, como por exemplo, quantidade de linhas de código, como também por meio da contagem de clientes, programas que têm como objetivo se conectar para enviar requisições aos serviços. Outra forma de medir o tamanho é por meio de recursos como contagem de recebidos para um dos serviços que foi

identificado um URI.

Tamanho (S): A medida do tamanho (S) é o resultado de a soma de duas métricas: Contagem de Clientes (CC) e Contagem de Recursos (RC). Um Recurso é definido como um serviço ou objeto de dados de rede identificado por URI. Um Cliente é um aplicativo que estabelece uma conexão para enviar solicitações. A Contagem de Clientes representa o número de serviços HTTP identificados pelo URI (FIELDING et al., 1999).

$$S = CC + RC$$

Conexão entre serviços (ISC): A comunicação entre microsserviços é realizada por mensagens via HTTP (FOWLER, 2019), sendo necessário que sejam totalmente compatíveis entre si de forma que suas URIs sejam válidas. Para analisar esta métrica, precisa-se contar os recursos não utilizados (URC) e os *endpoints* inacessíveis (UEC) (ASIK; SELCUK, 2017).

Contagem de recursos não utilizados (URC) é o número de URIs implementados, mas não usados por outros microsserviços ou aplicativos. No melhor cenário, a contagem de recursos não utilizados é 0 (ASIK; SELCUK, 2017).

Contagem de *endpoints* inacessíveis (UEC) é o número de URI implementado em um microsserviço que não existe ou não corresponde a nenhum critério (ASIK; SELCUK, 2017).

Essa métrica tem como objetivo fornecer manutenibilidade e confiabilidade, evitando más práticas.

$$ISC = URC + UEC$$

Más Práticas (BP): Embora não haja restrição quanto ao tamanho dos URIs (FIELDING et al., 1999), URIs longos (mais de 50 caracteres) dificultam a compreensão do código-fonte, URIs estáticos são desencorajados devido à dinâmica natureza dos microsserviços (ASIK; SELCUK, 2017). Espera-se que essa métrica seja o mais próximo possível de 0, pois evitar essas más práticas melhora a capacidade de manutenção e a confiabilidade do software.

Número de Serviços (NS): Esta métrica está relacionada ao tamanho, mas também ajuda a medir a complexidade da arquitetura de microsserviços. Basicamente, conta-se o número de serviços acoplados em um sistema, quanto maior este número, mais complexo é o sistema (ZHANG; XINKE, 2009b).

- NS= representa o número total de serviços no sistema.

- S[*]= é o conjunto de todos os serviços presentes no sistema de microsserviços.

$$NS = |S[*]|$$

As métricas a seguir estão relacionadas aos requisitos de qualidade da ISO/IEC 25010 (ISO/IEC 25010, 2011).

4.1.2 Métricas de Acoplamento

O acoplamento diz respeito ao nível de interconexão e conexão que os serviços têm entre si, em microsserviços. O grau de acoplamento deve ser baixo para que cada microsserviço seja independente e facilite a manutenção e reutilização do código-fonte (BOGNER; WAGNER; ZIMMERMANN, 2017).

Importância Absoluta do Serviço (AIS) : O número de clientes que invocam pelo menos uma operação da interface de um serviço (BOGNER; WAGNER; ZIMMERMANN, 2017).

Dependência Absoluta do Serviço (ADS): Se o serviço A chama o serviço B, então o serviço A depende do serviço B. Esta métrica é calculada a partir da soma das dependências que um serviço tem (BOGNER; WAGNER; ZIMMERMANN, 2017).

Interdependência de serviços no sistema (SIY): O número de pares de serviços dos quais um depende do outro (BOGNER; WAGNER; ZIMMERMANN, 2017). No melhor cenário, esse valor seria 0.

4.1.3 Métricas de Granularidade

Estas métricas medem o nível de serviços como reduzido.

Contagem ponderada de interface de serviço (WSIC) A definição proposta por Hirzala *et al.* (HIRZALLA; CLELAND-HUANG; ARSANJANI, 2008) para a métrica de número ponderado de interfaces expostas ou operações de um serviço considera diversos aspectos, como o número e a complexidade dos tipos de dados dos parâmetros em cada interface. Por padrão, o peso atribuído é 1. No entanto, Hirzala (HIRZALLA; CLELAND-HUANG; ARSANJANI, 2008) reconhece que essa métrica pode retornar o número de métodos e interfaces expostos de um serviço, com o peso variando conforme a complexidade da operação e o número de parâmetros.

4.1.4 Métricas de Coesão

A coesão em microsserviços melhora a qualidade do software, o que torna o sistema menos complexo. O ganho na qualidade do software vem do baixo impacto da manutenção do código-fonte, característica dos microsserviços (ZHANG; XINKE, 2009b).

Um nó, no contexto de sistemas orientados a serviços, fornece um ou mais serviços não móveis, e um serviço é um conjunto de operações.

Coesão de Rede no Sistema (NCS) Para *Zhang et al.* (ZHANG; XINKE, 2009b) é importante contar os laços entre nós em sistemas orientados a serviços.

- NCS: representa o número de conexões diretas entre nós de invocadores e nós de serviços em um sistema orientado a serviços.
- A: é a matriz ou tabela de relacionamentos que descreve as conexões entre os nós do sistema.
- $\sigma_{\text{Invoker Node}\neq\text{Service Node}}(A)$: aplica uma seleção ($\sigma$) na matriz A, filtrando apenas os pares em que o nó invocador (*Invoker Node*) e o nó de serviço (*Service Node*) são diferentes.
- $\pi_{\text{Invoker Node, Service Node}}$: realiza uma projeção ($\pi$) para obter os pares de nós relevantes após a seleção.

$$NCS = |\pi_{InvokerNode,ServiceNode}(\sigma_{InvokerNode\neq ServiceNode}(A))|$$

Coesão Relativa de Rede no Sistema (RNCS)

A métrica NCS entre dois nós específicos não considera a complexidade do serviço para determinar o grau de coesão, portanto, para esta métrica, o valor do NCS é dividido pelo número de serviços no sistema (ZHANG; XINKE, 2009b).

- NCS: representa o número de conexões diretas entre dois nós específicos no sistema.
- $RNCS$: é o valor normalizado da métrica NCS, calculado dividindo NCS pelo número total de serviços no sistema.
- $|N|$: é o número total de serviços presentes no sistema de microsserviços.

$$RNCS = \frac{NCS}{|N|}$$

Coesão de dados da interface de serviço (SIDC)

A alta coesão do sistema é alcançada quando todas as operações de serviço funcionam nos mesmos tipos de parâmetros de entrada. Para calcular esta métrica é necessário contar o total de operações de serviço que possuem os mesmos tipos de parâmetros de entrada e dividir pelo número de tipos de parâmetros discretos (PEREPLETCHIKOV; RYAN; FRAMPTON, 2007). Um resultado é um número entre 0 e 1, e quanto maior esse resultado, mais coeso é o sistema.

- $SIDC(s)$: é a métrica de coesão do sistema para o serviço s.
- $Common(Param(so \in SO(si_s)))$: representa o conjunto de tipos de parâmetros de entrada que são comuns a todas as operações (so) do serviço s.

- $SO(si_s)$: é o conjunto de operações de serviço associadas ao serviço s.
- $totalParamTypes$: é o número total de tipos de parâmetros discretos utilizados no sistema.

$$SIDC(s) = \frac{|Common(Param(so \in SO(si_s)))|}{totalParamTypes}$$

Coesão de Uso da Interface de Serviço(SIUC)

Para esta métrica, a alta coesão é um cenário com todas as operações de serviço invocadas por cada cliente (PEREPLETCHIKOV; RYAN; FRAMPTON, 2007). O número de todas as operações do usuário por cliente é dividido pelo número de clientes multiplicado pelo número de operações de serviço.

- $SIUC(s)$: é a métrica de coesão com base nas invocações de serviço por cliente para o serviço s.
- $INV(clientes, SO(si_s))$: representa o número total de invocações das operações de serviço ($SO(si_s)$) realizadas pelos clientes.
- $numclientes$: é o número total de clientes no sistema que interagem com o serviço s.
- $|SO(si_s)|$: representa a cardinalidade (ou tamanho) do conjunto de operações de serviço associadas ao serviço s.

4.1.5 Métricas de Desempenho

O desempenho figura como um atributo crítico de qualidade, sendo procurado tanto por desenvolvedores quanto por usuários. Todas as métricas de desempenho estão atreladas ao tempo, mais precisamente ao tempo de resposta. Weerasinghe *et. al.* (WEERASINGHE; PERERA, 2021) consideram as seguintes métricas:

Latência (L) - Indica quanto tempo leva para completar uma tarefa. Um tempo de resposta rápido, em microssegundos ou milissegundos, é indicativo de bom desempenho (WEERASINGHE; PERERA, 2021).

- L: representa a latência, ou seja, o tempo total necessário para completar uma tarefa.
- $t_{\text{Hora final}}$: é o instante de tempo em que a tarefa é concluída.
- $t_{\text{Hora inicial}}$: é o instante de tempo em que a tarefa foi iniciada.

$$L = t_{Horafinal} - t_{Horainicial}$$

***Throughput* (T)** - É o número de transações por segundo (TPS). Este é o número de tarefas concluídas em um período de tempo (WEERASINGHE; PERERA, 2021).

- TPS: é o *throughput*, ou seja, o número de transações por segundo.
- N: é o número total de tarefas ou transações concluídas.
- T: é o tempo total em segundos durante o qual as tarefas foram realizadas.

$$TPS = \frac{N}{T}$$

Capacidade (C) - O número máximo de transações que o software pode executar em um espaço de tempo com o máximo de recursos disponíveis (WEERASINGHE; PERERA, 2021).

4.2 Revisão Sistemática e Bibliométrica

Como resultado das buscas, obteve-se diferentes pontos de vista relacionados à pesquisa extraídos dos artigos selecionados, sendo então possível dividir o resultado em duas partes. Primeiramente, a análise bibliométrica, onde os artigos mais relevantes foram classificados de acordo com alguns requisitos escolhidos para esta seleção, como autores mais citados, nuvem de palavras dos assuntos mais comentados e produção dos autores ao longo do tempo. Na segunda parte, foram coletadas métricas no contexto de microsserviços que possam auxiliar na discussão de padrões de arquitetura de requisitos de software, apresentados na pesquisa com o intuito de auxiliar pesquisadores que buscam realizar trabalhos relacionados a este assunto respondendo as questões feitas em planejamento deste artigo.

A busca resultou em um total de 419 artigos, dos quais 51 artigos foram selecionados, conforme ilustrado na Figura 7 o processo de seleção e o número de artigos em cada fase, abrangendo dados detalhados e gerais em vários aspectos, como a relação entre trabalhos selecionados e aceitos em bancos de dados como *ACM Digital Library*, *IEEE*, *Web Of Science*, *ScienceDirect* e *Scopus*. O estudo investigou a distribuição de artigos escolhidos por base de dados e ano, analisou minuciosamente os autores e seus artigos publicados e explorou a correlação entre publicações e países de origem dos autores.

Um mapa temático foi elaborado para destacar os assuntos mais discutidos. Os resultados da pesquisa forneceram *insights* valiosos sobre o cenário atual de microsserviços e métricas do setor. Em seu escopo e profundidade, estes dados suplementares abrangentes enriqueceram ainda mais a compreensão sobre o tópico, fornecendo uma visão detalhada do comportamento e das tendências emergentes neste contexto. O resultado desta revisão sistemática e análise bibliométrica gerou um artigo submetido inicialmente ao ICCSA.

À medida que os artigos eram revisados, as respostas a cada pergunta da fase de planejamento eram gradualmente consolidadas.

4.2.1 Questão 1: Como as publicações sobre o tema são organizadas?

É possível avaliar a estabilidade do tema, pois há publicações contínuas desde 2006, mas com crescimento nos últimos anos. Outro ponto importante é que o tema é distribuído globalmente, com publicações de todos os continentes.

A Figura 10 mostra a distribuição dos artigos selecionados por base de dados e ano. Entre os anos de 2020 e 2022, houve crescimento quando comparado aos anos de 2013 a 2018. Neste contexto, é essencial ressaltar a importância do serviço, com mais de vinte ocorrências em cada base, conforme mostram as Figuras 11 e 12.

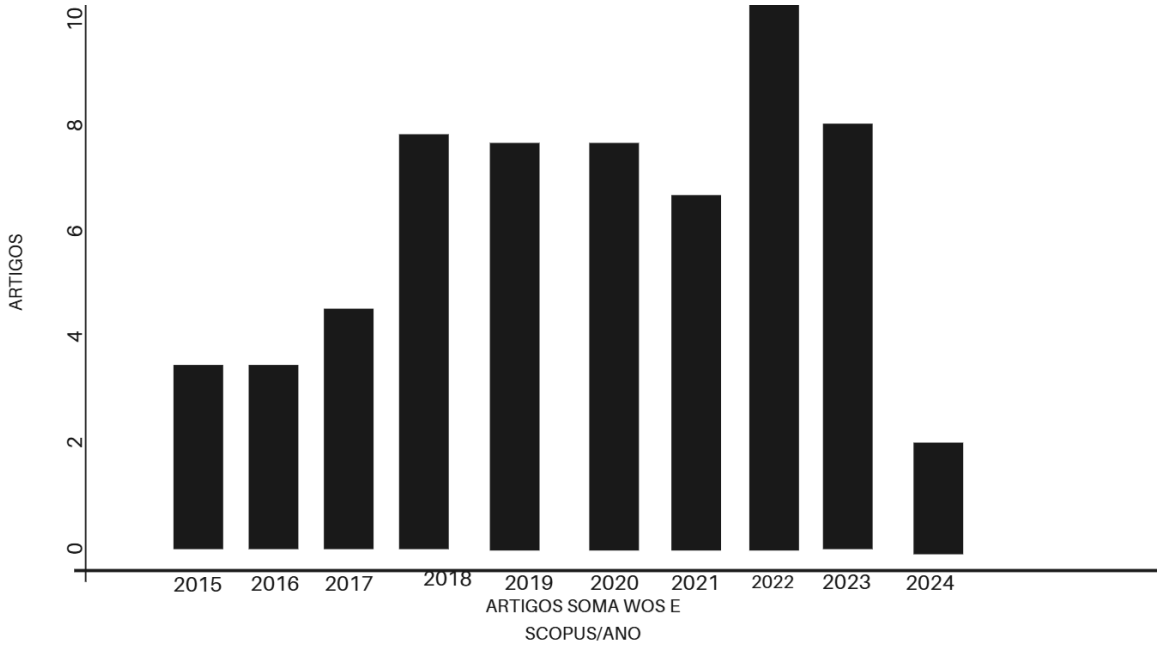

Figura 10 – Artigos Soma WOS e Scopus/Ano

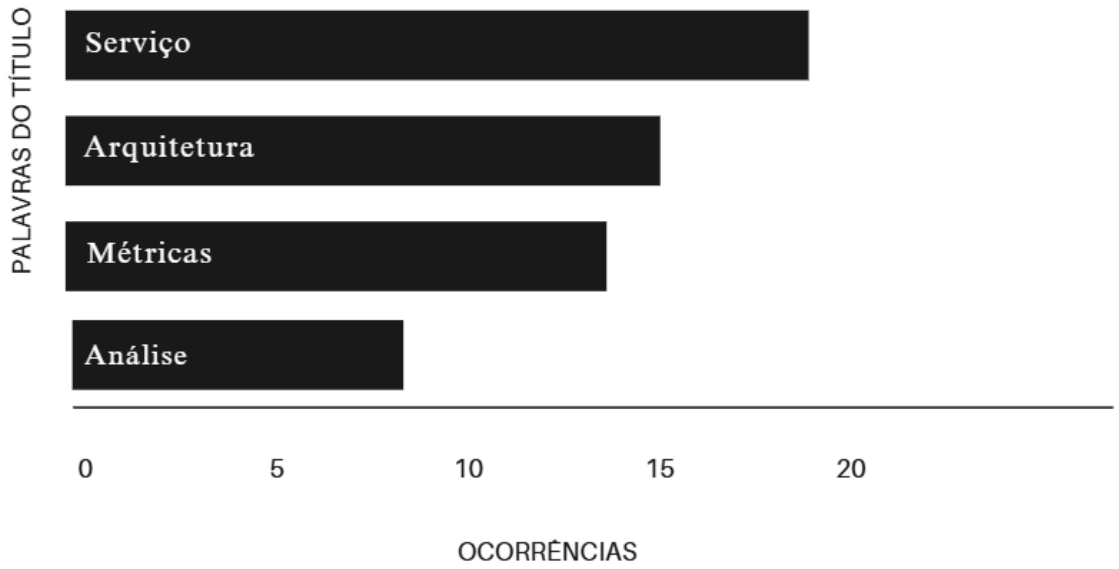

Figura 11 – Tópicos mais citados em artigos da *Scopus*

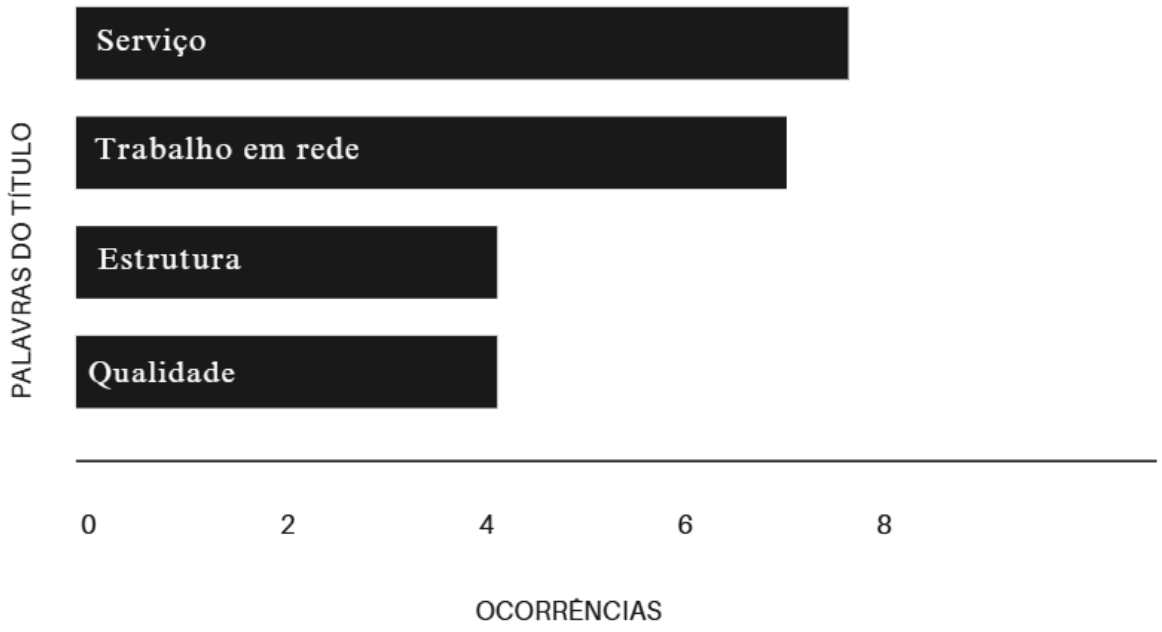

Figura 12 – Tópicos mais citados em artigos da *Web of Science*

As Figuras 13 e 14 apresentam o mapa temático extraíndo temas comuns usando o método *Multiple Correspondence Analysis* (MCA). A Figura 13 destaca temas centrais relacionados

à arquitetura de software, design de software de microsserviços, serviços web, qualidade de software, software como serviço (SaaS) e arquitetura orientada a serviços (SOA), além de abordar a importância dos serviços de informação. Esses temas estão organizados de forma a refletir as principais áreas de interesse dentro do campo de software e suas inter-relações. Já a Figura 14 foca em temas mais específicos, como arquitetura de software escalável, avaliação de análise dinâmica, roteamento móvel em redes e estrutura de nuvem de serviços, revelando nichos emergentes e especializados que complementam a visão geral obtida na Figura 13 . Juntas, as figuras fornecem uma visão clara de como os temas são agrupados em torno de conceitos-chave, como os nichos específicos, os temas essenciais e os emergentes, permitindo uma compreensão aprofundada das áreas de conhecimento mais relevantes e em desenvolvimento na pesquisa.

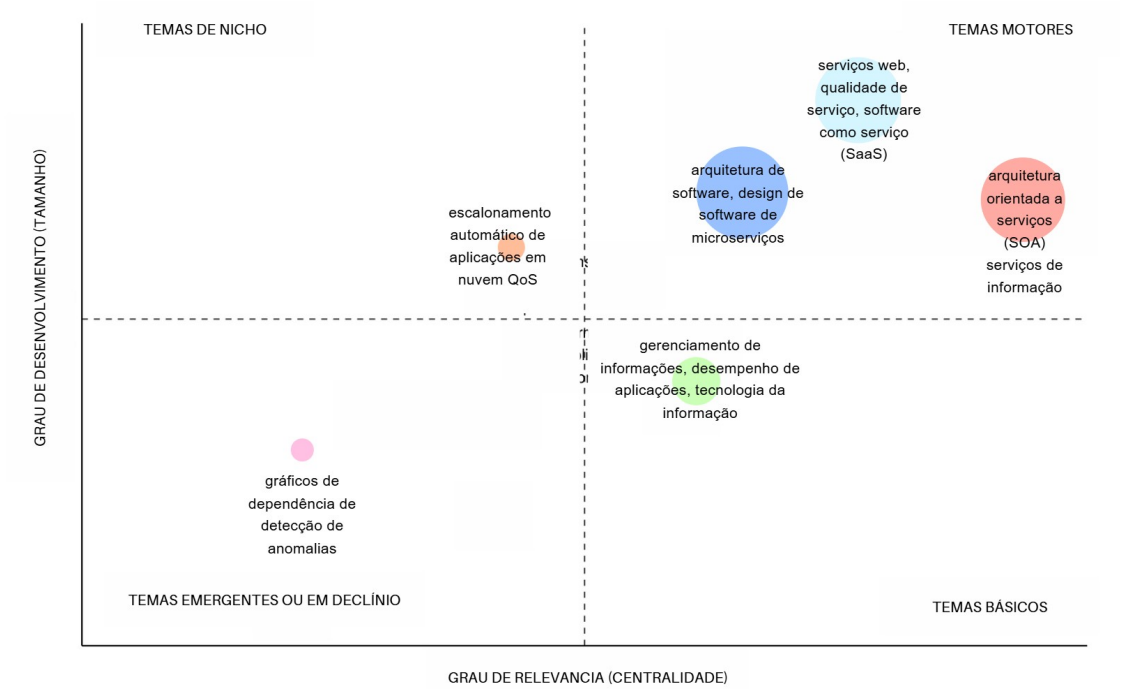

Figura 13 – Mapa temático do *Scopus*

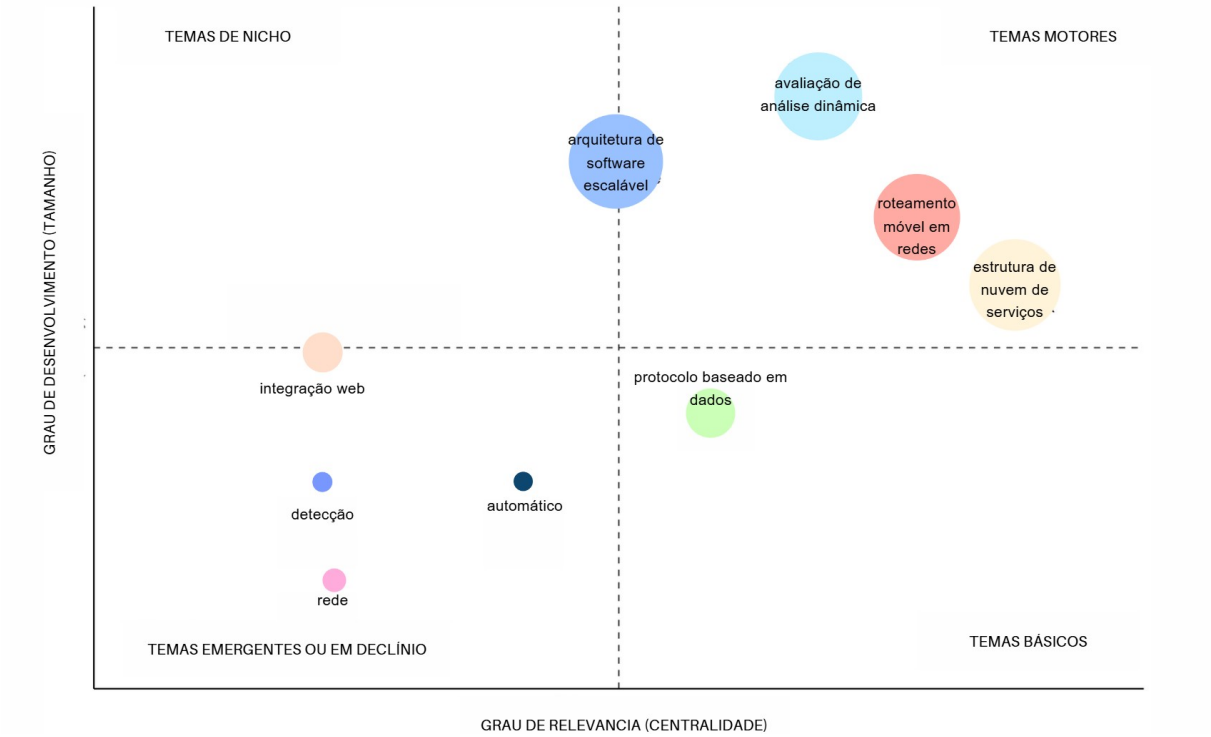

Figura 14 – Mapa temático da *Web of Science*

4.2.2 Questão 2: Quais são as principais métricas apresentadas nos artigos?

Quatro métricas principais foram identificadas com pelo menos vinte ocorrências entre os artigos: Desempenho, acoplamento, coesão e complexidade, conforme descrito na Tabela 6.

Tabela 6 – Referências relacionadas a cada uma das métricas

Métricas	**Referências**
Desempenho	(AGARWAL et al., 2021) , (LEE et al., 2009) , (GOTIN et al., 2018) , (BALFAGIH; HASSAN, 2009) , (DONG et al., 2020), (MENG et al., 2020) , (NORVAIsA; PACKEVIcIUS, 2017), (ADESINA; DARAMOLA; AYO, 2010), (ZHAO et al., 2006), (LEE, 2010), (HASSAN; BAHSOON; BUYYA, 2022b), (NURAINI; WIDYANI, 2014), (RIGUEIRA; BERNARDINO; PEDROSA, 2020), (MANIK, 2022), (MOHAMED; EL-GAYAR, 2021), (CEBOTARI; KUGELE, 2020), (MAYNARD; DIMITOGLOU, 2008), (ZHANG; XINKE, 2009a), (KHOSHKBARFOROUSHHA; JAMSHIDI; SHAMS, 2010), (HOJAJI; SHIRAZI, 2010), (ALAA, 2011), (KASSOU; KJIRI, 2012), (CAMARGO et al., 2016), (MUNIALO; MUKETHA; OMIENO, 2020), (SHEIKH; AMBHAIKAR, 2021)

Métricas	**Referências**
Acoplamento	(SHANMUGASUNDARAM; VENKATESAN; DEVI, 2012), (AGARWAL et al., 2021), (GOTIN et al., 2018), (ZHAO et al., 2006), (ASIK; SELCUK, 2017), (NURAINI; WIDYANI, 2014), (CEBOTARI; KUGELE, 2020), (MAYNARD; DIMITOGLOU, 2008), (ALAA, 2011), (MUAZ; RANA; HAMEED, 2021), (GONZALEZ et al., 2011), (PULPARAMBIL et al., 2018b), (ALMEIDA; SILVA, 2020), (SANTOS; PAULA, 2021) , (VALE et al., 2022) , (AUER et al., 2021), (RAJ; RAVICHANDRA, 2018), (TUMMALAPALLI et al., 2022), (DAUD; KADIR, 2015), (BHANDARI; GUPTA, 2020), (OLIVEIRA; VARGAS; RODRIGUES, 2018), (VALE et al., 2022) , (MUAZ; RANA; HAMEED, 2021), (GONZALEZ et al., 2011), (PULPARAMBIL et al., 2018b), (JAWADDI; JOHARI; ISMAIL, 2022), (VERA-RIVERA; GAONA; ASTUDILLO, 2021), (ALMEIDA; SILVA, 2020), (ZHANG; GRACANIN, 2008), (CHIBA et al., 2019), (VALE et al., 2022), (AUER et al., 2021), (SHAN et al., 2019), (AFIFY et al., 2013), (FIEGLER; DUMKE, 2011), (RAJ; RAVICHANDRA, 2018), (CHANG; LIN, 2008), (TUMMALAPALLI et al., 2022), (TOUEIR; BROISIN; SIBILLA, 2011), (DAUD; KADIR, 2015), (VALE et al., 2022),(OLIVEIRA; VARGAS; RODRIGUES, 2018), (VALE et al., 2022), (LI et al., 2022)
Coesão	(SHANMUGASUNDARAM; VENKATESAN; DEVI, 2012) , (MENG et al., 2020), (ADESINA; DARAMOLA; AYO, 2010), (OLIVEIRA; VARGAS; RODRIGUES, 2018), (ASIK; SELCUK, 2017), (RAJ; RAVICHANDRA, 2018), (VALE et al., 2022), (HOU et al., 2021), (MONTEIRO et al., 2023), (HOJAJI; SHIRAZI, 2010), (KLEFTAKIS et al., 2022), (TUMMALAPALLI et al., 2022), (ZHONG et al., 2023) ,(AFIFY et al., 2013) ,(MAYNARD; DIMITOGLOU, 2008), (KASSOU; KJIRI, 2012)
Complexidade	(SHANMUGASUNDARAM; VENKATESAN; DEVI, 2012) , (MENG et al., 2020), (ADESINA; DARAMOLA; AYO, 2010), (ZHAO et al., 2006), (ASIK; SELCUK, 2017), (MANIK, 2022), (CEBOTARI; KUGELE, 2020), (MAYNARD; DIMITOGLOU, 2008), (ZHANG; XINKE, 2009a), (KHOSHKBARFOROUSHHA; JAMSHIDI; SHAMS, 2010), (HOJAJI; SHIRAZI, 2010), (ALAA, 2011), (KASSOU; KJIRI, 2012), (CAMARGO et al., 2016), (MUNIALO; MUKETHA; OMIENO, 2020), (SHEIKH; AMBHAIKAR, 2021), (MUAZ; RANA; HAMEED, 2021), (GONZALEZ et al., 2011), (PULPARAMBIL et al., 2018b), (ALMEIDA; SILVA, 2020), (CHIBA et al., 2019), (SANTOS; PAULA, 2021), (VALE et al., 2022), (AUER et al., 2021), (FIEGLER; DUMKE, 2011), (RAJ; RAVICHANDRA, 2018), (TOUEIR; BROISIN; SIBILLA, 2011), (DAUD; KADIR, 2015), (BHANDARI; GUPTA, 2020), (OLIVEIRA; VARGAS; RODRIGUES, 2018), (VALE et al., 2022)

Sub-Pergunta 2.1: Como as métricas estão sendo abordadas e como as métricas se relacionam com a Norma ISO/IEC 25010?

Desempenho - O tempo de execução pode ser medido desde a requisição até a obtenção da resposta (CHANG; LIN, 2008). Para poder validar tais propriedades em microsserviços, o teste é um recurso essencial a ser aplicado, pois, a partir dele, é possível garantir que as regras de negócio sejam respeitadas e os gargalos encontrados. Em associação com a Norma ISO/IEC 25010, esta métrica está associada às características de qualidade, desempenho e eficiência. O tempo de resposta é uma medida que interfere no desempenho. Uma solução para que a velocidade seja alta é adotar um controle de comunicação que não pese, como a arquitetura estilo REST (VALE et al., 2022).

Coesão - Aplicações orientadas a serviços devem ter alta coesão e baixo acoplamento (ZHAO et al., 2006). Estas métricas são definidas com base no uso dos mesmos tipos de parâmetros e contendo padrões de interface dos consumidores de serviços, como operações invocadas, número e ordem (CEBOTARI; KUGELE, 2020). De acordo com a ISO/IEC 25010, esta métrica está associada às características de qualidade de portabilidade e eficácia.

Acoplamento - O relacionamento entre dois serviços é chamado de acoplamento (RAJ; RAVICHANDRA, 2018), mas se for necessário haver independência entre eles (ALAA, 2011), estes serviços devem ser fracamente acoplados (ZHANG; XINKE, 2009a). De acordo com a ISO/IEC 25010, esta métrica está associada às características de qualidade: desempenho, eficiência, portabilidade e eficácia.

Complexidade - Uma alta complexidade pode impactar negativamente na qualidade do software, principalmente em sua manutenção, pois quanto mais complexo, menor a manutenibilidade (MUAZ; RANA; HAMEED, 2021). Se for um microsserviço, a complexidade deve ser baixa porque o serviço deve ser atualizado, corrigido e implementado rapidamente. Portanto, medir a complexidade é primordial na arquitetura de microsserviços (VERA-RIVERA; GAONA; ASTUDILLO, 2021).

4.3 Survey

A pesquisa foi respondida de maio de 2023 a agosto de 2023 por 7 gestores e 26 desenvolvedores

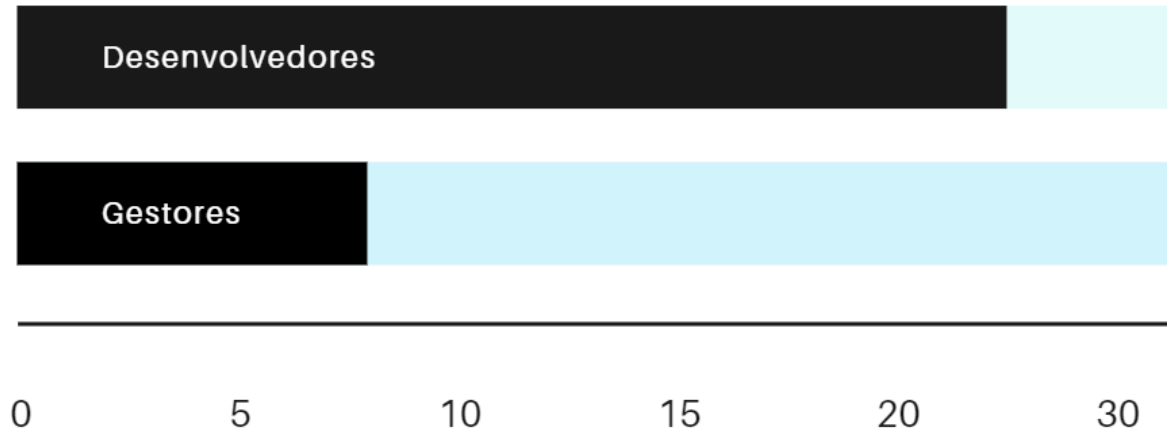

Figura 15 – Visão geral das votações de gestores e programadores

Os resultados da pesquisa de gerentes e desenvolvedores foram analisados individualmente, coletando as métricas mais votadas de cada grupo, permitindo a inter-relação dos votos quantitativos pela equipe e pela gerência, conforme mostrado na Tabela 9.

Métricas	**% Gerente**	**% Desenvolvedor**
Escalabilidade	61,5% afirmaram que a arquitetura de microsserviços aumentou a escalabilidade do sistema.	73,1% são motivados a programar com microsserviços porque podem escalar os módulos mais acessados.
Manutenção	61,5% afirmaram que melhora a agilidade nas entregas.	57,7% encontram facilidade em corrigir o código.
Descentralização	46.2 afirma que os microsserviços auxiliaram na independência entre funcionalidades.	76.9% afirmaram que este é um princípio orientador em suas implementações.
Granularidade	53.8% código mais limpo.	57.7% afirmaram que a granularidade é um princípio orientador em suas implementações de microsserviços.
Reusabilidade	53.8% entregas mais frequentes, 61.5% agilidade na manutenção.	61.5% afirmaram que este é um princípio orientador em suas implementações.
Evolutividade	76.9% melhorias constantes, priorização de demanda.	50% equipes menores focadas em resoluções estratégicas.
Acoplamento frouxo	61.5% dificuldade na integração com legados.	65.4% afirmaram que a dependência entre funcionalidades é algo que deve ser melhor gerenciado.
Desempenho e tempo de resposta	de 92,3%, uso de CPU como 61,5 %, rendimento com 53,8 %.	73% declararam que este é um princípio orientador em suas implementações.

Tabela 7 – Detalhes sobre votação de métricas

Esta comparação estruturada da Tabela 7 permitiu entender como os diferentes aspectos dos microsserviços são percebidos por aqueles que gerenciam projetos e aqueles que os implementam. As porcentagens para cada métrica ilustraram o grau de impacto atribuído a cada grupo.

A Tabela 8 apresenta seus principais focos e respectivas preocupações no desenvolvimento de software.

Métricas	**Gerente x Desenvolvedor**
Escalabilidade	Enquanto os desenvolvedores se concentram mais na implementação técnica e na eficiência geral do sistema, os gerentes estão mais preocupados com a capacidade do sistema de atender às demandas do usuário e manter uma boa experiência do cliente.
Manutenção	Enquanto os gerentes se concentram mais nas metas de negócios e na satisfação do cliente, os desenvolvedores estão mais preocupados com a qualidade técnica e a capacidade de manutenção do código.
Descentralização	Os gerentes se concentram mais nos objetivos e resultados de negócios, enquanto os desenvolvedores estão mais preocupados com questões técnicas e de engenharia de software.
Reutilização	Os gerentes reconhecem os benefícios de entregas mais frequentes e agilidade na manutenção do sistema. Enquanto isso, os desenvolvedores valorizam a reutilização como um princípio fundamental, promovendo eficiência e consistência no código.
Evolutividade	gerentes focam em resultados tangíveis da evolutividade, como melhorias contínuas e priorização efetiva da demanda, enquanto os desenvolvedores valorizam benefícios práticos como equipes menores focadas em resoluções estratégicas.
Acoplamento Fraco	gerentes estão mais preocupados com a dificuldade de integração com sistemas legados, enquanto os desenvolvedores estão mais preocupados com a dependência entre funcionalidades.
Desempenho	gerentes focam na otimização de métricas como tempo de resposta, uso de CPU e rendimento para garantir o desempenho do sistema na produção. Ao mesmo tempo, os desenvolvedores priorizam a implementação de princípios de desempenho desde o início do desenvolvimento, garantindo que os microsserviços sejam eficientes e responsivos.

Tabela 8 – Comparação das métricas mais votadas entre gerente e desenvolvedor

A Tabela 8 destaca como gerentes e desenvolvedores abordam microsserviços de diferentes

perspectivas — gerentes de um ponto de vista comercial e estratégico, e desenvolvedores de uma perspectiva técnica e focada na implementação.

Visão geral da análise

Além de entender as opiniões das partes envolvidas, que era o objetivo da pesquisa, foi possível comparar se as visões de líderes e subordinados eram consistentes, conforme mostrado na Figura 16.

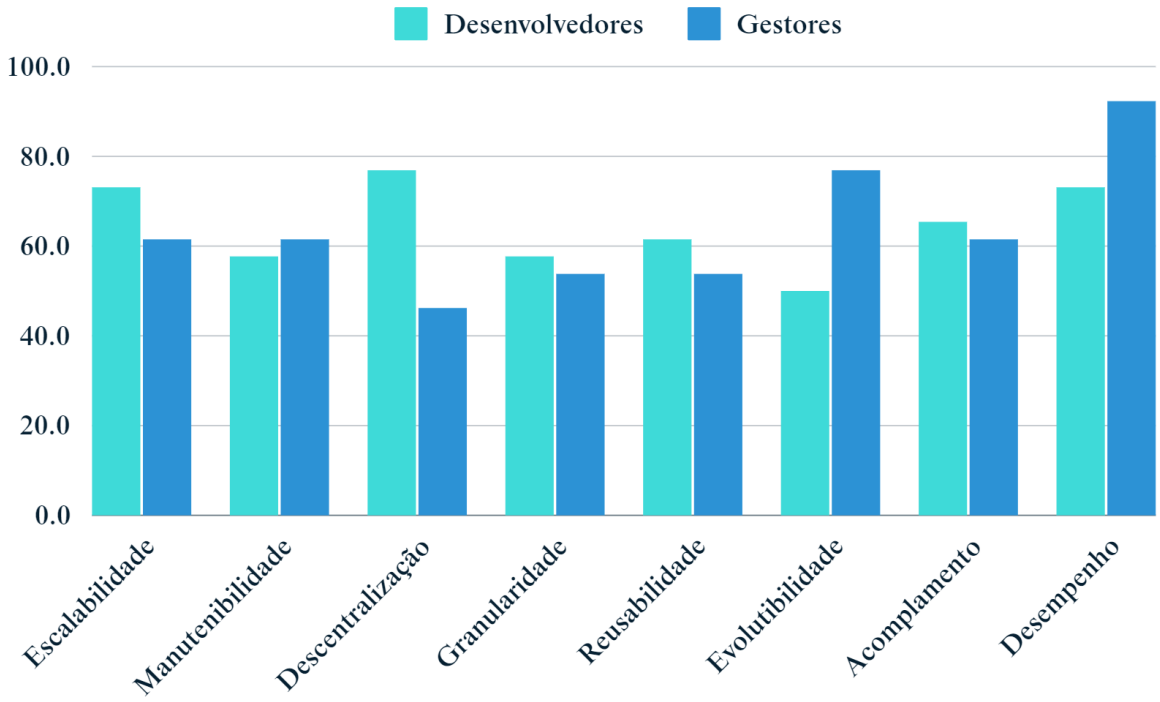

Figura 16 – Métricas mais votadas

Os resultados ressaltam a natureza multifacetada das motivações por trás da adoção de microsserviços. Enquanto alguns desenvolvedores priorizam a eficiência operacional e a manutenção do código, outros valorizam a flexibilidade arquitetural e a escalabilidade, como a Figura 17 representa. No entanto, é essencial reconhecer que estes fatores não são mutuamente exclusivos e geralmente são interconectados ao buscar sistemas robustos e escaláveis.

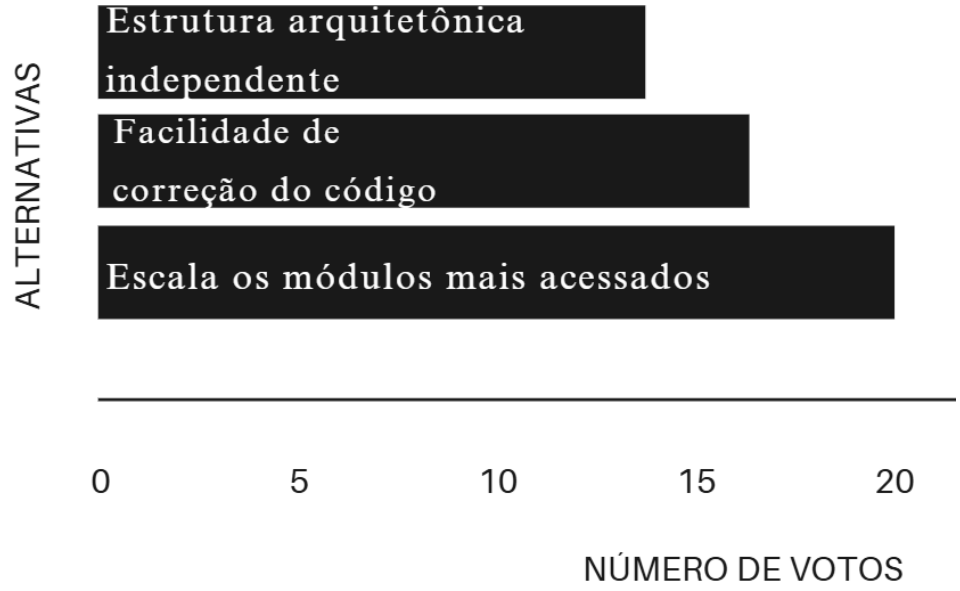

Figura 17 – As motivações mais significativas para desenvolver uma arquitetura de microsserviços

Cada métrica reflete diferentes aspectos do processo de desenvolvimento e o valor entregue ao cliente, enfatizando a importância de escolher métricas alinhadas aos objetivos do projeto e às necessidades dos *stakeholders*. A Figura 18 coletou dados sobre como os gerentes monitoram essas entregas.

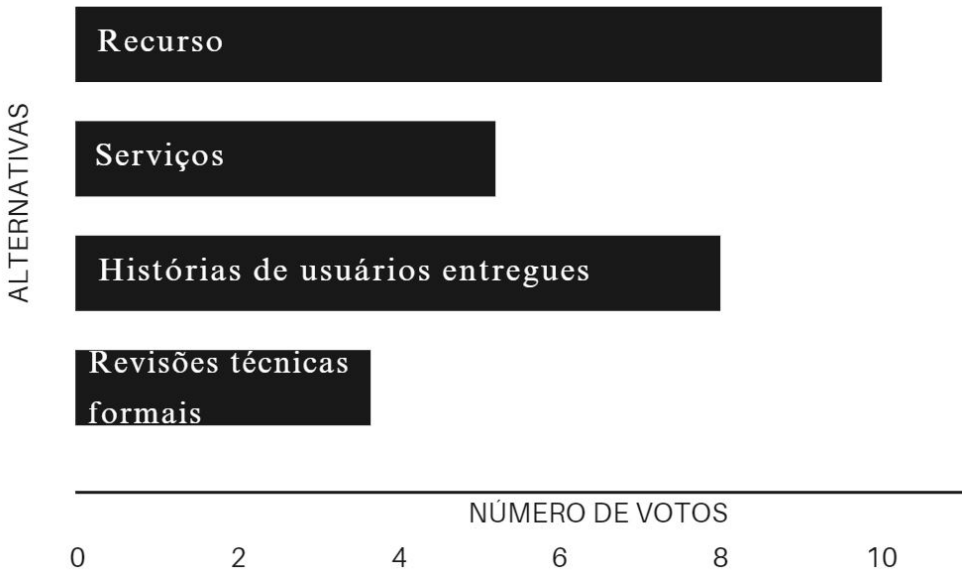

Figura 18 – Como os gestores medem suas entregas

5

Resultados dos Artefatos

Com fundamento na base de conhecimento, foram desenvolvidos dois artefatos para este trabalho: o Catálogo de Métricas e a ferramenta de recomendação Ramin. O catálogo organiza um conjunto de métricas voltadas para a avaliação de microsserviços, oferecendo uma estrutura acessível para sua consulta e aplicação. A ferramenta Ramin, por sua vez, foi projetada para recomendar as métricas mais adequadas, considerando as características e necessidades específicas de cada contexto, o que otimiza o processo de seleção. Juntos, esses artefatos tem o objetivo de proporcionar suporte, facilitando a análise e a melhoria de microsserviços mais eficiente.

5.1 Catálogo

As métricas associadas aos microsserviços podem ser interpretadas de maneiras distintas por diferentes autores, com alguns utilizando termos como escalabilidade, desempenho (DRAGONI et al., 2017), descentralização, modularidade (ASIK; SELCUK, 2017), manutenção e complexidade (BOGNER et al., 2019) para descrever aspectos fundamentais do design e operação desses sistemas. Em contraste, outros estudiosos, como exemplificado por (ENGEL et al., 2018), abordam os princípios de microsserviços, identificando 10 diretrizes baseadas em uma análise da literatura e entrevistas estruturadas. Estes princípios podem ser vistos como fatores ou características subjacentes às métricas mencionadas anteriormente.

A Tabela 9 apresenta 19 métricas selecionadas que estão alinhadas com os requisitos de qualidade da Norma ISO/IEC 25010 (ISO/IEC 25010, 2011), (FRANcA; JOYCE; SOARES, 2015), (SOARES; FRANÇA, 2016). Estas métricas cobrem uma variedade de critérios e foram fundamentais na construção do catálogo e do Repositório de Informações (RI). Foi realizada uma análise detalhada das métricas, incluindo uma descrição das características das normas subsequentes que as influenciam.

A Tabela 9 oferece uma explanação sobre cada métrica, abordando sua aplicação prática e as referências que justificam a escolha e o uso de tais métricas. Os dados apresentados na Tabela 9 foram utilizados para alimentar a LLM de recomendações de métricas, proporcionando *insights* valiosos para a avaliação e otimização de microsserviços.

Métrica	Características ISO/IEC 25010	Descrição	Como Aplicar
Escalabilidade (BOGNER et al., 2019), (AGARWAL et al., 2021), (SHARON et al., 2010)	Adequação Funcional	Otimiza o desempenho e gerencia o tráfego intenso. A escalabilidade está relacionada a tolerância a falhas e, em microsserviços, é individualizada, o que reduz custos em comparação com a escalabilidade de sistemas monolíticos, que requerem recursos para o sistema como um todo.	Um sistema é considerado escalável quando possui a capacidade de aumentar a sua capacidade de processamento, seja pela adição de novos recursos computacionais ou pela ampliação dos existentes, visando melhorar a capacidade dos serviços.
Descentralização (EL-SHARKAWY; KRAFCZYK; SCHMID, 2019)	Compatibilidade, Portabilidade	A lógica de negócios dos sistemas pode ser dividida em diversos microsserviços, com cada um deles sendo responsável pela definição e gestão independente de seu próprio modelo de domínio, regras de negócios e banco de dados.	Cada microsserviço deve ser projetado para operar de maneira autônoma, com sua própria lógica de negócios e um banco de dados específico, garantindo que possua uma responsabilidade clara e distinta para facilitar a descentralização.

Tabela 9 continuação da página anterior

Métrica	Características ISO/IEC 25010	Descrição	Como Aplicar
Granularidade (HASSAN; BAHSOON; BUYYA, 2022b), (CHHILLAR; GAHLOT, 2017)	Confiabilidade, Adequação Funcional	Avalia o nível de serviços como reduzido. A análise da granularidade de um serviço composto considera a quantidade de serviços que o integram.	A granularidade da aplicação esta intimamente ligado a separação de responsabilidade, ou seja, um sistema que tenha uma única responsabilidade tem uma granularidade mais adequada, sendo mais coeso e mais autônomos, podendo ser desenvolvido e implantado de forma independente.
Manutenibilidade (BOGNER et al., 2019) (CHHILLAR; GAHLOT, 2017), (KUMAR; BHATIA, 2015)	Manutenibilidade, Usabilidade	Devido à sua arquitetura baseada em um conjunto de serviços independentes, a estrutura de microsserviços torna a manutenção mais eficiente, o que pode contribuir para a redução do tempo de desenvolvimento, bem como para a melhoria da qualidade do software e o aumento da satisfação das equipes de desenvolvimento.	A manutenibilidade de um microsserviço está associada a fatores como autonomia, baixo acoplamento, coesão e granularidade apropriada, sendo esses elementos fundamentais para a avaliação e garantia da eficiência na manutenção do sistema.
Desempenho (WEERASINGHE; PERERA, 2021)	Desempenho	O desempenho para desenvolvedores e usuários, é medido pelo tempo de resposta, incluindo latência, capacidade e taxa de transferência.	$L = t_{EndTime} - t_{StartTime}$ $TPS = \frac{N}{T}$

Tabela 9 continuação da página anterior

Métrica	Características ISO/IEC 25010	Descrição	Como Aplicar
Reusabililidade (DONG et al., 2020)	Manutenibilidade, Portabilidade	Os serviços podem ser reutilizados por mais de um sistema e na construção de novos serviços sem precisar criar novas regras de negócios ou tabelas de banco de dados.	A reusabilidade em microsserviços é definida como a capacidade de um serviço ser utilizado em múltiplos contextos e aplicações sem necessidade de alterações, sendo necessário o planejamento modular e desacoplado, permitindo a integração de seus componentes em diferentes sistemas, com consequente redução de redundância.
Fraco Acoplamento (ZHANG; XINKE, 2009a)	Manutenibilidade	No contexto de sistemas de software, o relacionamento entre dois serviços é denominado "acoplamento". Para assegurar a independência entre os serviços, é fundamental que eles sejam fracamente acoplados.	O fraco acoplamento em microsserviços refere-se à independência entre os serviços, minimizando dependências e facilitando manutenção, evolução e integração em diferentes contextos.
Evolutibilidade (SHARON et al., 2010)	Compatibilidade, Segurança	A evolutibilidade é resultado de características como fraco acoplamento e descentralização, por ser divido em blocos independentes, os microsserviços passam por manutenções como melhorias e correções de maneira ágil.	A evolutibilidade dos microsserviços refere-se à capacidade de adaptação e evolução ao longo do tempo, sendo facilitada por documentação detalhada, testes automatizados, controle de versionamento e independência entre funcionalidades.

Tabela 9 continuação da página anterior

Métrica	Características ISO/IEC 25010	Descrição	Como Aplicar
Complexidade (COSCIA et al., 2012)	Manutenibilidade, Adequação Funcional, Desempenho	Auxiliam o desenvolvimento e a manutenção, melhoram a escalabilidade e reduzem o risco de falhas.	A complexidade em microsserviços é definida como o nível de dificuldade na construção, manutenção e integração dos serviços, sendo importante adotar práticas que minimizem a sobrecarga e garantam a clareza na comunicação entre os componentes.
Coesão (ZHANG; XINKE, 2009a)	*Safety*, Manutenibilidade	A coesão em microsserviços indica a eficiência e a inter-relação das funcionalidades, facilitando a manutenção e a evolução do serviço.	$RNCS = \frac{NCS}{\|N\|}$
Interoperabilidade (MEGARGEL; SHANKARARAMAN; WALKER, 2020; NERI et al., 2020)	Portabilidade, Segurança, Manutenibilidade, Compatibilidade	Microsserviços permitem a comunicação entre sistemas distintos através de mensagens HTTP, possibilitando o uso de diferentes linguagens e plataformas.	A implementação de interoperabilidade em microsserviços requer a adoção de padrões de comunicação como REST ou gRPC, definição de contratos de interface por meio de *OpenAPI* ou *Protocol Buffers*, e uso de mecanismos de descoberta de serviços como *Consul* ou *Kubernetes* para facilitar a localização dos serviços.

Tabela 9 continuação da página anterior

Métrica	Características ISO/IEC 25010	Descrição	Como Aplicar
Sem dependências cíclicas (ENGEL et al., 2018)	Confiabilidade, manutenibilidade, Eficiência de Desempenho, Compatibilidade	Para manter a independência, os microsserviços devem evitar dependências cíclicas, onde dois serviços dependem mutuamente.	É recomendado aplicar uma modularização clara e separação de responsabilidades. Implementando políticas de design que favoreçam a comunicação unidirecional e reduzam o acoplamento entre os microsserviços, e assim possuam escalabilidade e a manutenibilidade.
Tolerância de falhas (FOWLER, 2019)	*Safety*, Adequação Funcional, Compatibilidade, Confiabilidade	Microsserviços devem ser projetados para tolerar falhas em interações, permitindo maior robustez e escalabilidade individual em comparação a sistemas monolíticos.	Para identificar e resolver rapidamente problemas é utilizando monitoramento proativo e *logging* detalhado. É adotado práticas de design robusto, como isolamento de falhas para minimizar o impacto de falhas individuais no sistema como um todo.
Agnosticismo Organizacional (THÖNES, 2015), (DAUD; KADIR, 2015)	*Safety*, Usabilidade, flexibilidade	Uma vez definida a interface de serviço, o foco do desenvolvedor é em sua utilização, com arquitetura, banco de dados e tecnologia sendo irrelevantes.	Para aplicar esta métrica, pode-se separar das estruturas organizacionais serviços baseado em capacidades especificas. Além de adotar boas práticas de comunicação clara e padrões de projeto que contribuam com a colaboração entre equipes multifuncionais para melhorar a flexibilidade e escalabilidade dos serviços.

Tabela 9 continuação da página anterior

Métrica	Características ISO/IEC 25010	Descrição	Como Aplicar
Capacidade de gerenciamento (ENGEL et al., 2018)	Usabilidade, Eficiência de Desempenho	A distribuição modular dos microsserviços melhora o gerenciamento e proporciona maior clareza no design do sistema devido ao seu tamanho reduzido.	É aplicada através de monitoramento contínuo, automação de tarefas operacionais e uso de ferramentas de orquestração para garantir escalabilidade. Essas práticas permitem a detecção precoce de problemas e facilitam a manutenção e atualização dos serviços.
Design orientado a domínio (ENGEL et al., 2018)	Portabilidade, manutenibilidade	A decomposição de domínios complexos em domínios menores e o mapeamento de suas relações ajudam a compreender e reforçar a separação de responsabilidades.	É aplicado através da divisão dos serviços conforme os contextos delimitados do negócio, garantindo que cada serviço seja responsável por uma funcionalidade específica. Isso promove uma maior coesão e reduz o acoplamento entre os serviços
Modularidade (KESSEL; ATKINSON, 2015)	Manutenibilidade, Eficiência de Desempenho	Microsserviços consistem em componentes coesos com limites bem definidos, promovendo um sistema modular em vez de um único componente multifuncional.	É aplicada segmentando a aplicação em unidades independentes, cada uma com uma responsabilidade específica, permitindo desenvolvimento, implantação e escalabilidade autônomos. Isso facilita a manutenção e evolução do sistema, promovendo uma arquitetura mais flexível

Tabela 9 continuação da página anterior

Métrica	Características ISO/IEC 25010	Descrição	Como Aplicar
Modelado em torno do domínio de negócios Norma ISO/IEC 25010:2011	Usabilidade, Eficiência de Desempenho, confiabilidade, segurança	A arquitetura de microsserviços vai além da comunicação, subdividindo o sistema em serviços com funções específicas e distintas.	É aplicado através da criação de serviços que refletem os processos e regras da organização, garantindo alinhamento com os objetivos empresariais.E promovendo uma maior compreensão e eficiência na implementação de soluções tecnológicas.

Tabela 9 – Detalhe das métricas

5.2 Ramin

Ramin (Recursos Ágeis de Métricas Inteligentes) é uma ferramenta desenvolvida para otimizar a aplicação de métricas em sistemas de microsserviços, visando melhorar a qualidade do software. Por meio de técnicas como LLM, VSM e RAG, o Ramin sugere métricas personalizadas com base nos problemas relatados pelas equipes de desenvolvimento, permitindo a escolha das métricas mais adequadas para cada cenário específico, como escalabilidade, desempenho, modularidade e complexidade.

O Ramin oferece benefícios importantes, como a minimização da alucinação (erros gerados por modelos de linguagem que produzem informações incorretas ou irrelevantes). Isto é alcançado pela integração de uma base de dados estruturada e a personalização das respostas com base no conhecimento específico do contexto de desenvolvimento. Outro benefício significativo é a rastreabilidade da fonte da resposta do LLM, visto que todas as sugestões de métricas são derivadas diretamente do catálogo de padrões apresentado neste estudo, garantindo que as recomendações sejam transparentes e possam ser auditadas para verificar sua origem e adequação.

Ao utilizar o Ramin, as equipes podem obter informações sobre quais métricas devem ser aplicadas para otimizar processos, tomar decisões mais assertivas e melhorar a qualidade do software, alinhando-se aos requisitos de qualidade estabelecidos pelo Norma ISO/IEC 25010. A ferramenta visa agilizar o processo de seleção de métricas, proporcionando recomendações rápidas e práticas para a equipe, o que facilita a avaliação contínua e a melhoria dos microsserviços de maneira eficiente e ágil.

5.2.1 Testes com Problemas de Design de Industrial

Conforme descrição na Subseção 3.4.1 (Execução), foram realizados testes em uma empresa do setor educacional e outra do domínio da segurança da informação, enfrentando dificuldades relacionadas ao gerenciamento de microsserviços. Participaram profissionais de outras empresas que também trabalham com microsserviços. Neste contexto, 11 profissionais colaboraram no processo conforme descrito na Tabela 10.

Entrevistado	**Posição**	**Formação Acadêmica**	**Anos de experiência em desenvolvimento de software**	**Anos de experiência em desenvolvimento de microsserviços**
R1	Arquiteto de Software	Estudante de Mestrado	8	4
R2	Líder Técnico	Mestre	16	3
R3	Engenheiro de Dados	Mestre	30	6
R4	Líder Ténico	Mestre	7	4
R5	Desenvolvedor	Mestre	5	5
R6	Desenvolvedor	Bacharel em Ciência da Computação	3	3
R7	Desenvolvedor	Bacharel em Ciência da Computação	8	5
R8	Desenvolvedor	Graduado em Sistemas para Internet	3	2
R9	Desenvolvedor	Graduado em Sistemas para Internet	4	2
R10	Desenvolvedor	Mestre	4	2
R11	Desenvolvedor	Bacharel em Ciência da Computação	5	1

Tabela 10 – Respondentes do Formulário

Ao todo, foram identificados 35 problemas de gerenciamento de microsserviços, resultando em 105 recomendações de métricas, uma vez que para cada problema são sugeridas três métricas, conforme mencionado na Subseção de execução do Estudo de Caso. Depois de receber as recomendações, eles deram seu feedback em um formulário com perguntas específicas sobre a ferramenta. Para ilustrar alguns dos problemas identificados durante os testes e oferecer uma visão de suas descrições, são apresentados a seguir três dos problemas relatados pelos profissionais:

Problema de desenvolvimento industrial 1: Quais métricas eu deveria utilizar para

garantir que os microsserviços são performáticos?

Problema de desenvolvimento industrial 2: Quais métricas podem ser utilizadas para avaliar a eficácia da comunicação entre microsserviços em um ambiente onde frequentemente ocorrem falhas de integração?

Problema de desenvolvimento industrial 3: Como é possível identificar gargalos de desempenho em um sistema de microsserviços que está apresentando alto tempos de resposta, e quais métricas específicas podem ser utilizadas para monitorar e otimizar esses tempos?

81,8% informaram que a ferramenta é eficaz para auxiliar gestores e desenvolvedores a auxiliar métricas.

5.2.2 Questionário com gestores e desenvolvedores

Sobre a Q1, a análise das respostas sobre a eficácia da ferramenta em arquitetura de microsserviços revelou que ela é considerada útil no planejamento e levantamento de requisitos, com dois participantes destacando sua capacidade de alinhar a arquitetura às necessidades organizacionais. A ferramenta também é elogiada por sugerir métricas para microsserviços individuais, embora cinco respostas apontem que sua precisão diminui em cenários que envolvem múltiplos serviços. A visualização em tempo real para identificar gargalos de desempenho foi vista como um recurso valioso. Três participantes reconheceram a utilidade da ferramenta no desenvolvimento, manutenção e resolução de problemas, sugerindo que ela apoia a aprendizagem contínua. A eficácia na explicação de conceitos de monitoramento e na sugestão de ferramentas foi mencionada, mas alguns participantes pediram mais orientações práticas sobre como aplicar estas sugestões. A necessidade de maior profundidade nas orientações foi apontada como uma limitação a ser considerada.

Sobre a Q2, 81,8% “concorda” que a ferramenta é eficaz para ajudar gestores e desenvolvedores a conhecer e aplicar métricas e 18,8% “concorda um pouco”.

Em relação à Q3, a necessidade de permitir que os usuários avaliem a utilidade das respostas geradas pela inteligência artificial foi mencionada por 2 participantes, visando calibrar o algoritmo. A busca por métricas mais contextualizadas para cenários distribuídos foi apontada por 1 pessoa, assim como o pedido para incluir exemplos práticos que detalhem a aplicação dessas métricas. Melhorias na usabilidade foram sugeridas por 4 usuários, que solicitaram suporte a múltiplos idiomas, um vídeo explicativo na página inicial e uma interface mais interativa, como um formato de chat. 3 participantes notaram a generalidade das respostas, especialmente em contextos como *e-commerce*, pedindo informações mais específicas. 2 usuários mencionaram a importância de aprimorar a interpretação das perguntas para facilitar um diálogo mais contínuo. Essas melhorias visam aumentar a eficácia da ferramenta no desenvolvimento e monitoramento de microsserviços.

As respostas da Q4 sobre as vantagens da ferramenta para projetos de microsserviços

indicam que ela serve como um repositório confiável de informações, com 4 participantes destacando sua especialização em um nicho, ao contrário de uma abordagem generalizada. A velocidade nas respostas e a qualidade da base de conhecimento foram elogiadas por 3 usuários, ressaltando a facilidade de integração com múltiplos protocolos. A ferramenta é considerada um guia útil para desenvolvedores iniciantes (4 menções), oferecendo boas práticas e facilitando a tomada de decisão ao fornecer dados sobre uso de recursos e latência. Esta capacidade de gerar tabelas de métricas relevantes foi mencionada como um benefício para aqueles com pouca experiência em microsserviços.

Na Q5, uma necessidade apontada foi a falta de memória de histórico de conversas, mencionada por 1 usuário, que sugeriu que a ferramenta poderia reter informações para melhor atender às necessidades. A segurança dos dados também foi destacada como uma preocupação por 1 participante. Dois usuários indicaram que a ferramenta precisa de mais treinamento para oferecer respostas específicas a cenários e sugeriram integrações com ferramentas como *New Relic* ou *Datadog*. Um desafio identificado foi a sobrecarga de recursos, com 1 usuário mencionando que ferramentas de monitoramento podem consumir CPU e memória adicionais. Apesar de 3 participantes não terem identificado desvantagens, 3 outros mencionaram problemas de usabilidade, como a necessidade de um histórico de respostas e melhorias na apresentação das tabelas.

Sobre a Q6: 9,1% afirmaram ser muito difícil, 27,3% Muito fácil, e 63,6% Fácil de usar.

Em relação a Q7, quatro participantes expressaram a possibilidade de integração, mencionando que a ferramenta pode ser útil para operações, identificação de gargalos e melhoria contínua. Um usuário destacou que, se a ferramenta evoluir, poderá se tornar útil em projetos. Outro comentou que, dependendo do projeto, a integração poderia ser considerada, visto que a ferramenta atendeu suas expectativas. Por outro lado, dois usuários mostraram ceticismo. Um afirmou que, apesar da especialização da ferramenta, seu uso ainda é muito similar a outros assistentes, como o ChatGPT. Outro comentou que, no momento, está focado em Microfrontends, o que limita a aplicação da ferramenta em projetos. Uma resposta sugeriu que a ferramenta poderia ser integrada a sistemas de gerenciamento de projetos, melhorando a criação e aplicação de métricas de forma eficiente.

Sobre a Q8, as respostas sobre se a ferramenta oferece métricas específicas necessárias para melhorar o desenvolvimento e a operação de microsserviços mostraram um panorama misto. Seis participantes afirmaram que a ferramenta oferece métricas essenciais, destacando indicadores como taxa de erro, uso de recursos, latência e disponibilidade dos serviços. Um usuário comentou que a ferramenta traz boas ideias, especialmente para gestão, ajudando a formar uma visão mais completa do sistema. No entanto, dois usuários expressaram a necessidade de métricas adicionais. Um mencionou a importância de métricas relacionadas à correlação entre os serviços, visto que o desempenho é frequentemente calculada com todos os microsserviços envolvidos. Outro usuário destacou que, embora a ferramenta forneça um contexto geral, seria útil

um ajuste fino para evitar respostas genéricas. Ele mencionou um exemplo em que a ferramenta não atendeu sua expectativa ao não abordar a criação de *dashboards* no *New Relic*, focando em instalação em vez de aplicação.

5.2.3 Feedback via videoconferência com gestores e desenvolvedores

Além do formulário, foi realizada conversa por videoconferência com cinco dos 11 profissionais que responderam ao formulário. O objetivo deste encontro foi, de forma mais subjetiva, entender as necessidades de cada pessoa e avaliar se a ferramenta atende a estas demandas. A Tabela 11 mostra a posição de cada profissional.

Entrevistado	**Posição**	**Formação Acadêmica**	**Anos de experiência em desenvolvimento de software**	**Anos de experiência em desenvolvimento de microsserviços**
E1	Arquiteto de Software	Estudante de Mestrado	8	4
E2	Líder Técnico	Mestre	16	3
E3	Engenhreiro de Dados	Mestre	30	6
E4	Líder Técnico	Mestre	7	4
E5	Desenvolvedor	Mestre	5	5

Tabela 11 – Entrevistados

Opinião Geral sobre a Ferramenta

E1 considerou a ferramenta muito boa para auxiliar tanto gestores que não têm familiaridade com métricas quanto programadores. Sugeriu a criação de uma trilha de aprendizagem e a possibilidade de gerar PDFs. E2 afirmou que a ferramenta tem potencial e pode ser uma forma de tirar dúvidas e planejar, mesmo lembrando que, às vezes, esquece detalhes importantes. E3 destacou que a ferramenta é boa, mas sugere melhorar o *layout* e manter um histórico mais acessível. E4 também elogiou a ferramenta, classificando-a como muito útil. E5 achou a ferramenta bacana, ressaltando que o *layout* é parecido com o do ChatGPT, o que proporciona uma sensação de familiaridade. Ele também acredita que a ideia de abordar métricas cumpre seu papel, explicando e respondendo a questões relevantes.

Potencial de Uso e Aplicações Práticas

E1 mencionou que a ferramenta serviria para um gestor de arquitetura, ajudando a montar uma estrutura eficiente. E2 comentou que, embora existam muitas ferramentas de IA, esta é específica e pode ajudar a propor métricas, especialmente em situações como a queda nas vendas. E3 observou que o foco é excessivo na arquitetura e pouco nas métricas, o que pode ser um

ponto a ser ajustado. E4 indicou que utilizaria a ferramenta como guia em novos projetos e em futuras melhorias. E5 acredita que a ferramenta pode contribuir muito para a indústria, sugerindo que traga exemplos de aplicação e explique como aplicar as métricas. Ele também destacou a importância de realizar pesquisas.

Sugestões de Melhoria

E3 sugeriu que a ferramenta poderia melhorar o *layout*, com a possibilidade de limpar a tela ou manter um histórico acessível. E4 recomendou enriquecer a base de dados e trazer respostas mais completas. E5 sugeriu que a ferramenta poderia incluir exemplos práticos e abordagens de como aplicar as métricas no dia a dia.

Aceitação da Ferramenta como MVP

E1 acredita que a ferramenta pode evoluir de acordo com as sugestões dos usuários. E2 também expressou que adotaria a ferramenta para sua equipe, considerando suas características específicas. E3 disse que, inserindo suas sugestões, adotaria a ferramenta para utilização na equipe. E4 reforçou que adoraria a ferramenta, considerando-a um MVP com grande potencial de evolução. E5, por outro lado, mencionou que não utilizaria a ferramenta para gestão de microsserviços, preferindo focar na operação e no conhecimento das métricas para identificar possíveis gargalos, como a reutilização de recursos e a manutenção do desenvolvimento. Ele acredita que a ferramenta é boa para aprendizagem.

5.2.4 Resposta às perguntas

Os resultados responderam a duas perguntas-chave:

Q1. As Métricas aplicadas a microsserviços conseguem garantir melhor acompanhamento de software com esta arquitetura?

Sim, as métricas aplicadas a microsserviços serviram para um acompanhamento de software nesta arquitetura. O estudo recomendou 42 métricas que ajudaram a resolver problemas específicos de gerenciamento, abordando aspectos como desempenho, comunicação entre serviços e identificação de gargalos. Os testes em empresas de setores como educação e segurança da informação mostraram que os profissionais frequentemente enfrentam desafios na gestão de microsserviços, reforçando a importância de métricas específicas. O feedback dos participantes indicou que a aplicação destas métricas não só melhora a visibilidade das operações, mas também possibilita uma melhor tomada de decisão, contribuindo para a qualidade do software. A utilização de métricas adequadas permite que as equipes monitorem o desempenho de forma contínua, identificando áreas que necessitam de otimização e possibilitando que os microsserviços operem de forma eficiente.

Q2. As empresas que utilizam microsserviços acompanham suas métricas? Tal resultado está alinhado com a pesquisa acadêmica?

Sim, mas atualmente nenhum dos participantes relatou que utiliza uma ferramenta de recomendações com LLM. As empresas reconhecem sua importância para a eficiência operacional. A pesquisa revelou que 81,8% dos profissionais entendem que o acompanhamento de métricas é fundamental para a gestão eficaz de microsserviços. Durante as discussões, os cinco profissionais da videoconferência concordaram que as métricas permitem uma compreensão mais clara das interações entre os serviços e ajudam a identificar gargalos de desempenho. Eles enfatizaram a necessidade de métricas que possam ser contextualizadas em cenários específicos, como a correlação entre serviços e a latência, destacando que isso é importante para a manutenção e desempenho do software. Esses relatos estão alinhados com a literatura acadêmica, que, como relatado nos trabalhos relacionados, destaca a importância do monitoramento contínuo em arquiteturas de microsserviços. A capacidade de medir e analisar métricas pode contribuir para a melhoria contínua e para a identificação de problemas antes que se tornem críticos, confirmando a relevância dessas práticas no ambiente empresarial.

6

Conclusão

A pesquisa em métricas e indicadores aplicados a microsserviços tem se expandido, mas ainda são poucos os trabalhos que abordam especificamente estes temas dentro deste contexto. Indicadores são fundamentais para a transformação no desenvolvimento de sistemas, fornecendo dados que permitem decisões mais proativas. Neste estudo, além de uma revisão teórica, foi necessário realizar pesquisas com empresas que utilizam microsserviços em suas arquiteturas, o que contribui para o entendimento do estado da prática em termos de métricas e análises no contexto real de desenvolvimento de software.

Microsserviços, pesquisa acadêmica e indústria estão intimamente interligados, com uma interconexão crescente entre as métricas utilizadas nestes cenários. A descentralização e a independência entre os serviços permitem uma manutenção mais eficiente e uma evolução constante dos sistemas. A granularidade e a escalabilidade, características essenciais dos microsserviços, contribuem diretamente para a eficiência e o desempenho dos sistemas, permitindo múltiplos acessos e otimização de recursos. Esse ambiente altamente escalável impacta o desempenho de maneira significativa, criando a necessidade de métricas e indicadores específicos que possam ser utilizados para monitorar e melhorar continuamente essas arquiteturas.

A integração entre academia e indústria tem impulsionado a inovação no desenvolvimento de microsserviços, pois a academia oferece conhecimento teórico enquanto a indústria compartilha os desafios práticos do mundo real. A colaboração mútua entre estes dois campos promove uma relação simbiótica, onde as soluções acadêmicas são validadas e refinadas em contextos práticos, enquanto os dados do mundo real fornecem novas questões e inspirações para a pesquisa científica. Esse ciclo contínuo de troca acelera a inovação, resultando em avanços significativos no campo do desenvolvimento de microsserviços.

Neste contexto, surge a ferramenta Ramin, proposta por este estudo, que se destaca por usar um LLM, combinado com a técnica RAG, e bancos de dados vetoriais para sugerir métricas específicas para microsserviços. Ao contrário de outros trabalhos que apenas catalogam métricas

ou recomendam padrões de *design*, o Ramin oferece sugestões personalizadas, baseadas nas necessidades reais e nos desafios enfrentados pelas equipes de desenvolvimento. Este diferencial permite que as recomendações sejam mais eficazes, pois consideram o contexto único de cada microsserviço, facilitando sua implementação de maneira prática e direta.

A validação da ferramenta por profissionais da área tem demonstrado sua aplicabilidade no mundo real. Ao fornecer uma interface intuitiva e orientações detalhadas, o Ramin se posiciona como uma solução útil para melhorar a qualidade e o desempenho das arquiteturas de microsserviços. Sua adoção permite que as equipes de desenvolvimento tomem decisões mais informadas, minimizando os riscos de falhas e melhorando a produtividade no ciclo de vida do software. A ferramenta contribui para o desenvolvimento de sistemas mais robustos e alinhados às necessidades reais dos negócios e usuários finais.

Em cenários onde a complexidade dos microsserviços aumenta, o Ramin oferece uma abordagem inovadora que se adapta rapidamente às mudanças nas necessidades do mercado e dos usuários. A integração de inteligência artificial e personalização nas recomendações facilita a evolução dos microsserviços e o ajuste das soluções em tempo real, permitindo que as equipes de desenvolvimento enfrentem os desafios de um ambiente de software cada vez mais dinâmico. Esta abordagem não apenas preenche lacunas na literatura, mas também redefine práticas no campo da Engenharia de Software, promovendo a construção de sistemas mais escaláveis e alinhados às exigências do mercado.

6.1 Submissões

As submissões de artigos científicos consistem no envio de manuscritos para avaliação e possível publicação, sendo que, neste contexto, estão apresentados os artigos aprovados e aqueles que ainda aguardam resultado.

6.1.1 Aprovado

Métricas Aplicadas a Microsserviços -
Autores: Flávia Caroline dos Santos, Michel S. Soares, Fabio G. Rocha
Local: Workshop de Teses e Dissertações em Qualidade de Software (WTDQS) SBQS (2023)

6.1.2 Aguardando Resultado

A Systematic Review and Survey of Metrics for Microservices -
Autores: Flávia Caroline dos Santos, Shexmo Richarlison R. dos Santos, Michel S. Soares, Fabio G. Rocha
Local: Journal of Universal Computer Science (J.UCS) (2024)

A Catalog of Metrics Applied to Microservices -
Autores: Flávia Caroline dos Santos, Shexmo Richarlison R. dos Santos, Michel S. Soares, Fabio G. Rocha
Local: Empirical Software Engineering (2024)

6.2 Trabalhos Futuros

Em estudos futuros, serão exploradas formas de personalizar métricas para microsserviços, analisando a influência de variáveis como estágio de desenvolvimento, carga esperada e desafios operacionais (com base nas avaliações de Ramin) na escolha e eficácia dessas métricas. Também será aprofundado o uso de técnicas avançadas de inteligência artificial, como LLMs e RAG, para fornecer recomendações de métricas mais contextualizadas e adaptáveis, com base em dados e ajustes contínuos. Outrossim, será investigado como fornecer respostas dinâmicas aos usuários, incentivando sua maior interação com a ferramenta. Será analisada ainda a relação entre a aplicação de métricas adaptativas e a melhoria da qualidade do software e da produtividade em ambientes ágeis, com foco no impacto dessas métricas no desenvolvimento de microsserviços em cenários reais. Por fim, serão avaliados os efeitos dessas métricas em diferentes contextos industriais, validando seu valor e limitações em diversas organizações e projetos, proporcionando uma compreensão mais ampla de seu impacto na engenharia de software.

Referências

ACHARYULU, P. S.; SEETHARAMAIAH, P. A Measures and Metrics Framework for Software Safety. *SIGSOFT Softw. Eng. Notes*, Association for Computing Machinery, New York, NY, USA, v. 40, n. 1, p. 1–8, feb 2015. ISSN 0163-5948. Citado na página 17.

ADESINA, A.; DARAMOLA, J.; AYO, C. A SOA-based Framework for E-Procurement in Multi-Organisations. *International Journal of Electronic Finance*, Inderscience Enterprises Ltd., v. 4, p. 156–170, 2010. Citado 3 vezes nas páginas 15, 49 e 50.

AFIFY, Y. et al. A Semantic-based Software-as-a-Service (SaaS) Discovery and Selection System. *Proceedings - 2013 8th International Conference on Computer Engineering and Systems, ICCES 2013*, p. 642 – 651, 2013. Citado na página 50.

AGARWAL, S. et al. Monolith to Microservice Candidates using Business Functionality Inference. *2021 IEEE Int. Conf. on Web Services, ICWS 2021*, Institute of Electrical and Electronics Engineers Inc., p. 434 – 451, 2021. Citado 5 vezes nas páginas 10, 16, 49, 50 e 57.

AGRAWAL, N. Autonomic Cloud Computing Based Management and Security Solutions: State-of-the-Art, Challenges, and Opportunities. *Transactions on Emerging Telecommunications Technologies*, John Wiley and Sons Inc, v. 32, n. 12, p. 864–872, 2021. Citado na página 11.

ALAA, G. Service-oriented System Evolution Taxonomy and Metrics Derived from Complex Adaptive Systems Theory. *International Journal of Web Engineering and Technology*, Inderscience Publishers, v. 6, p. 44–61, 2011. Citado 3 vezes nas páginas 49, 50 e 51.

ALIERO, A. et al. Systematic Review on Text Normalization Techniques and its Approach to Non-Standard Words. *International Journal of Computer Applications*, v. 185, p. 975–8887, 09 2023. Citado 2 vezes nas páginas 21 e 22.

ALMEIDA, J.; SILVA, A. Monolith Migration Complexity Tuning Through the Application of Microservices Patterns. *Journal of Computer Science and Technology (JCST)*, p. 39–54, 09 2020. Citado 4 vezes nas páginas 9, 12, 13 e 50.

ASIK, T.; SELCUK, Y. E. Policy Enforcement upon Software based on Microservice Architecture. In: IEEE. *2017 IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA)*. [S.l.], 2017. p. 283–287. Citado 4 vezes nas páginas 40, 41, 50 e 56.

AUER, F. et al. From Monolithic Systems to Microservices: An Assessment Framework. *Information and Software Technology*, Elsevier, Amsterdam, The Netherlands, v. 137, p. 660–683, set. 2021. Citado 3 vezes nas páginas 13, 25 e 50.

BALFAGIH, Z.; HASSAN, M. Quality Model for Web Services from Multi-Stakeholders' Perspective. *Proceedings - 2009 International Conference on Information Management and Engineering, ICIME 2009*, Kuala Lumpur, p. 837–849, 2009. Citado 2 vezes nas páginas 15 e 49.

BHANDARI, G.; GUPTA, R. Fault Prediction in SOA-Based Systems Using Deep Learning Techniques. *International Journal of Web Services Research*, IGI Global, v. 17, p. 32–50, 2020. Citado na página 50.

BOGNER, J. et al. A Modular Approach to Calculate Service-Based Maintainability Metrics from Runtime Data of Microservices. *IEEE Access*, Springer, p. 489–496, 2019. Citado 6 vezes nas páginas 9, 10, 16, 56, 57 e 58.

BOGNER, J.; WAGNER, S.; ZIMMERMANN, A. Towards a Practical Maintainability Quality Model for Service-and Microservice-Based Systems. p. 195–198, 2017. Citado na página 42.

BOUMAHDI, F. et al. MDA4SOA+d: A New model Driven Architecture to Supporting Secision Making in SOA. *J. King Saud Univ. Comput. Inf. Sci.*, Elsevier Science Inc., USA, v. 35, n. 5, p. 432–446, may 2023. Citado 2 vezes nas páginas 9 e 10.

BROWN, T. B. et al. Language Models are Few-shot Learners. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2020. p. 54–79. Citado na página 20.

CAMARGO, A. de et al. An Architecture to Automate Performance Tests on Microservices. In: *Proceedings of the 18th International Conference on Information Integration and Web-Based Applications and Services*. New York, NY, USA: Association for Computing Machinery, 2016. (iiWAS '16), p. 422–429. Citado 2 vezes nas páginas 49 e 50.

CEBOTARI, V.; KUGELE, S. Playground for Early Automotive Service Architecture Design and Evaluation. *IEEE Intelligent Vehicles Symposium, Proceedings*, Institute of Electrical and Electronics Engineers Inc., p. 63 – 81, 2020. Citado 3 vezes nas páginas 49, 50 e 51.

CERNY, T. et al. Catalog and Detection Techniques of Microservice Anti-Patterns and Bad Smells: A Tertiary Study. *Journal of Systems and Software*, Elsevier, v. 206, p. 73–84, 2023. Citado 2 vezes nas páginas 16 e 26.

CHANDA, S.; PAL, S. The Effect of Stopword Removal on Information Retrieval for Code-Mixed Data Obtained Via Social Media. *SN Comput. Sci.*, Springer-Verlag, Berlin, Heidelberg, v. 4, n. 5, p. 63–82, jun 2023. Citado na página 22.

CHANG, S.; LIN, K.-J. A General QoS Error Detection and Diagnosis Framework for Accountable SOA. In: *Quality of Service in Service Computing: Models, Methodologies, and Applications*. Xi'an: [s.n.], 2008. p. 231–238. Citado 2 vezes nas páginas 50 e 51.

CHHILLAR, R. S.; GAHLOT, S. An Evolution of Software Metrics: A Review. In: *Proceedings of the International Conference on Advances in Image Processing*. New York, NY, USA: Association for Computing Machinery, 2017. (ICAIP '17), p. 139–143. Citado 2 vezes nas páginas 16 e 58.

CHIBA, T. et al. ConfAdvisor: A Performance-Centric Configuration Tuning Framework for Containers on Kubernetes. *Proceedings - 2019 IEEE International Conference on Cloud Engineering, IC2E 2019*, Institute of Electrical and Electronics Engineers Inc., p. 453–461, 2019. Citado na página 50.

COSCIA, J. L. O. et al. Predicting Web Service Maintainability Via Object-Oriented Metrics: A Statistics-Based Approach. *Journal of Systems and Software*, p. 29–39, 2012. Citado na página 60.

DAUD, N. N.; KADIR, W. W. Review on Structural Properties Metrics in SOA Design. *Jurnal Teknologi*, Penerbit UTM Press, v. 77, p. 223–238, 2015. Citado 3 vezes nas páginas 19, 50 e 61.

DEVLIN, J. et al. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. In: BURSTEIN, J.; DORAN, C.; SOLORIO, T. (Ed.). *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019. p. 4171–4186. Citado na página 26.

DONG, B. et al. Impact Analysis About Response time Considering Deployment Change of SaaS Software. *International Journal of Software Engineering and Knowledge Engineering*, World Scientific, v. 30, n. 07, p. 977–1004, 2020. Citado 4 vezes nas páginas 11, 16, 49 e 59.

DRAGONI, N. et al. Microservices: Yesterday, Today, and Tomorrow. *Present and Ulterior Software Engineering*, Springer, p. 195–216, 2017. Citado 3 vezes nas páginas 4, 11 e 56.

DUDLEY, J. J.; KRISTENSSON, P. O. A Review of User Interface Design for Interactive Machine Learning. Association for Computing Machinery, New York, NY, USA, v. 8, n. 2, p. 83–120, jun 2018. Citado 2 vezes nas páginas 21 e 23.

EL-SHARKAWY, S.; KRAFCZYK, A.; SCHMID, K. MetricHaven: More than 23,000 Metrics for Measuring Quality Attributes of Software Product Lines. Association for Computing Machinery, New York, NY, USA, p. 25–28, 2019. Citado na página 57.

ENGEL, T. et al. Evaluation of Microservice Architectures: A Metric and Tool-Based Approach. p. 74–89, 2018. Citado 3 vezes nas páginas 56, 61 e 62.

ERDEI, R.; TOKA, L. Minimizing Resource Allocation for Cloud-Native Microservices. *J. Netw. Syst. Manage.*, Plenum Press, USA, v. 31, n. 2, p. 105–123, feb 2023. Citado 3 vezes nas páginas 11, 18 e 25.

FIEGLER, A.; DUMKE, R. Growth- and Rntropy-Based SOA Measurement: Vision and Approach in a Large Scale Environment. *Proceedings - Joint Conference of the 21st International Workshop on Software Measurement, IWSM 2011 and the 6th International Conference on Software Process and Product Measurement, MENSURA 2011*, p. 735–741, 2011. Citado 2 vezes nas páginas 15 e 50.

FIELDING, R. et al. *RFC 2616: Hypertext Transfer Protocol–HTTP/1.1*. [S.l.], 1999. Citado na página 41.

FOWLER, M. *Microservices — martinfowler.com*. 2019. <https://martinfowler.com/articles/microservices.html>. [Accessed 24-Nov-2022]. Citado 3 vezes nas páginas 16, 41 e 61.

FRANcA, M. S.; JOYCE; SOARES, M. S. SOAQM: Quality Model for SOA Applications based on ISO 25010. In: *Proceedings of the 17th International Conference on Enterprise Information Systems - Volume 2*. Setubal, PRT: SCITEPRESS - Science and Technology Publications, Lda, 2015. (ICEIS 2015), p. 60–70. Citado 3 vezes nas páginas 16, 17 e 56.

FRANcA, M. S. et al. Architecture-Driven Development of an Electronic Health Record Considering the SOAQM Quality Model. *SN Comput. Sci.*, Springer-Verlag, Berlin, Heidelberg, v. 1, n. 3, apr 2020. Citado 2 vezes nas páginas 10 e 13.

GENFER, P.; ZDUN, U. Identifying Domain-Based Cyclic Dependencies in Microservice APIs Using Source Code Detectors. In: *Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. [S.l.]: Gesellschaft fur Informatik (GI), 2022. P-320, p. 29–31. Citado 4 vezes nas páginas 9, 10, 11 e 12.

GONZALEZ, A. et al. Context-aware Multimedia Service Composition Using Quality Assessment. *Proceedings - IEEE International Conference on Multimedia and Expo*, Barcelona, p. 42–55, 2011. Citado 2 vezes nas páginas 15 e 50.

GOTIN, M. et al. Investigating Performance Metrics for Scaling Microservices in CloudIoT-Environments. *ICPE 2018 - Proceedings of the 2018 ACM/SPEC International Conference on Performance Engineering*, Association for Computing Machinery, Inc, v. 2018-March, 2018. Citado 2 vezes nas páginas 49 e 50.

HASSAN, A. E. et al. Rethinking Software Engineering in the Era of Foundation Models: A Curated Catalogue of Challenges in the Development of Trustworthy FMware. *ACM Computing Surveys*, Association for Computing Machinery, New York, NY, USA, p. 294–305, 2024. Citado na página 20.

HASSAN, S.; BAHSOON, R.; BUYYA, R. Systematic Scalability Analysis for Microservices Granularity Adaptation Design Decisions. *SOFTWARE-PRACTICE & EXPERIENCE*, WILEY, 111 RIVER ST, HOBOKEN 07030-5774, NJ USA, v. 52, n. 6, p. 1378–1401, jun. 2022. Citado 3 vezes nas páginas 11, 13 e 19.

HASSAN, S.; BAHSOON, R.; BUYYA, R. Systematic Scalability Analysis for Microservices Granularity Adaptation Design Decisions. *Software - Practice and Experience*, John Wiley and Sons Ltd, v. 52, p. 324–334, 2022. Citado 4 vezes nas páginas 16, 25, 49 e 58.

HEVNER, A. et al. Design science in information systems research. *Management Information Systems Quarterly*, v. 28, p. 75–, 03 2004. Citado na página 28.

HIRZALLA, M.; CLELAND-HUANG, J.; ARSANJANI, A. A Metrics Suite for Evaluating Flexibility and Complexity in Service Oriented Architectures. In: SPRINGER. *International Conference on Service-Oriented Computing*. [S.l.], 2008. p. 41–52. Citado na página 42.

HOJAJI, F.; SHIRAZI, M. *Developing a More Comprehensive and Expressive SOA Governance Framework*. Chengdu: [s.n.], 2010. 105 - 123 p. Citado 3 vezes nas páginas 15, 49 e 50.

HORITA E.A., F. et al. Design Science in Digital Innovation: A Literature Review. In: . New York, NY, USA: Association for Computing Machinery, 2020. (SBSI'20), p. 200 –220. Citado na página 28.

HOU, C. et al. Diagnosing Performance Issues in Microservices with Heterogeneous Data Source. *IEEE Access*, p. 493–500, Sep. 2021. Citado na página 50.

ISO/IEC 25010. *ISO/IEC 25010:2011, Systems and Software Engineering — Systems and Software Quality Requirements and Evaluation (SQuaRE) — System and Software Quality Models*. 2011. Citado 2 vezes nas páginas 42 e 56.

JAWADDI, S. N. B. A.; JOHARI, M.; ISMAIL, A. A Review of Microservices Autoscaling with Formal Verification Perspective. *Software: Practice and Experience*, v. 52, p. 224–241, 08 2022. Citado 2 vezes nas páginas 15 e 50.

KASSOU, M.; KJIRI, L. SOASMM: A Novel Service Oriented Architecture Security Maturity Model. *Proceedings of 2012 International Conference on Multimedia Computing and Systems, ICMCS 2012*, Tangiers, p. 65–73, 2012. Citado 2 vezes nas páginas 49 e 50.

KEELE, S. et al. Guidelines for Performing Systematic Literature Reviews in Software Engineering. *IEEE Transactions on Software Engineering*, 2007. Citado na página 32.

KESSEL, M.; ATKINSON, C. Measuring the Superfluous Functionality in Software Components. In: *Software Engineering and Advanced Applications (SEAA 2017)*. New York, NY, USA: Association for Computing Machinery, 2015. (CBSE '15), p. 11–20. ISBN 9781450334716. Citado 3 vezes nas páginas 16, 18 e 62.

KHOSHKBARFOROUSHHA, A.; JAMSHIDI, P.; SHAMS, F. A metric for Composite Service Reusability Analysis. In: *Service-Oriented Computing – ICSOC 2005*. Cape Town: [s.n.], 2010. p. 76–82. Citado 3 vezes nas páginas 15, 49 e 50.

KITCHENHAM, B. Procedures for Performing Systematic Reviews. *Keele, UK, Keele University*, v. 33, p. 1–26, 2004. Citado na página 32.

KLEFTAKIS, S. et al. A Comparative Study of Monolithic and Microservices Architectures in Machine Learning Scenarios. *IEEE Access*, p. 352–357, Nov 2022. Citado 4 vezes nas páginas 10, 14, 15 e 50.

KUMAR, G.; BHATIA, P. K. Neuro-Fuzzy Model to Estimate & Optimize Quality and Performance of Component Based Software Engineering. *ACM SIGSOFT Softw. Eng. Notes*, v. 40, n. 2, p. 1–6, 2015. Citado 3 vezes nas páginas 16, 17 e 58.

LEE, J. et al. A Quality Model For Evaluating Software-as-a-Service in Cloud Computing. *Proceedings - 7th ACIS International Conference on Software Engineering Research, Management and Applications, SERA09*, Haikou, p. 75–99, 2009. Citado na página 49.

LEE, Y. QoS Metrics for Service Level Measurement for SOA Environment. *Proc. - 6th Intl. Conference on Advanced Information Management and Service, IMS2010, with ICMIA2010 - 2nd International Conference on Data Mining and Intelligent Information Technology Applications*, Seoul, p. 821 – 831, 2010. Citado na página 49.

LI, Z. et al. Microservice Extraction based on Knowledge Graph from Monolithic Applications. *Information and Software Technology*, Elsevier B.V., v. 150, p. 88–107, 2022. Citado 3 vezes nas páginas 16, 25 e 50.

MANIK, L. P. Performance Factors Effect on the Performance Metrics of the Enterprise Service Bus. *International Journal of Computing and Digital Systems*, v. 11, p. 107–115, 01 2022. Citado 2 vezes nas páginas 49 e 50.

MANNING, C. D.; RAGHAVAN, P.; SCHüTZE, H. Introduction to information retrieval. In: *Proceedings of the 2008 ACM Conference on Information and Knowledge Management*. [S.l.]: ACM, 2008. p. 313–324. Citado 3 vezes nas páginas 21, 24 e 25.

MASHIKO, Y.; BASILI, V. R. Using the GQM Paradigm to Investigate Influential Factors for Software Process Improvement. *Journal of Systems and Software*, Elsevier, v. 36, n. 1, p. 17–32, 1997. Citado na página 31.

MAYNARD, M.; DIMITOGLOU, G. A Service Oriented Architecture Complexity Metric, Based on Statistical Hypothesis Testing. *Proceedings of the 2008 International Conference on Software Engineering Research and Practice, SERP 2008*, p. 65–74, 2008. Citado 2 vezes nas páginas 49 e 50.

MEGARGEL, A.; SHANKARARAMAN, V.; WALKER, D. K. Migrating from Monoliths to Cloud-Based Microservices: A Banking Industry Example. *Journal of Cloud Computing: Advances, Systems and Applications*, Springer, p. 85–108, 2020. Citado 6 vezes nas páginas 9, 13, 14, 19, 20 e 60.

MENG, M. et al. AutoMAP: Diagnose Your Microservice-based Web Applications Automatically. In: *Web Conference 2020: Proceedings of the World Wide Web Conference (WWW 2020)*. New York, USA: Assoc Computing Machinery, 2020. Citado 3 vezes nas páginas 13, 49 e 50.

MOHAMED, H.; EL-GAYAR, O. End-to-End Latency Prediction of Microservices Workflow on Kubernetes: A Comparative Evaluation of Machine Learning Models and Resource Metrics. *Proceedings of the Annual Hawaii International Conference on System Sciences*, IEEE Computer Society, v. 2020-January, p. 211– 220, 2021. Citado 2 vezes nas páginas 9 e 49.

MONTEIRO, D. et al. Adaptive observability for forensic-ready microservice systems. *IEEE Transactions on Services Computing*, v. 16, n. 5, p. 3196–3209, Sep. 2023. ISSN 1939-1374. Citado na página 50.

MOURA Álex dos S.; ROCHA, F. G.; SOARES, M. S. Microservices Patterns Recommendation based on Information Retrieval. *Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, Journal of Universal Computer Science, v. 30, n. 11, p. 1455–1483, 2024. Citado 2 vezes nas páginas 22 e 26.

MUAZ, A.; RANA, M. E.; HAMEED, V. A Framework for Catering Software Complexity Issues Using Architectural Patterns. *Journal of Software: Evolution and Process*, p. 554–561, 11 2021. Citado 2 vezes nas páginas 50 e 51.

MUNIALO, S.; MUKETHA, G.; OMIENO, K. An Effort Estimation Method for Service-Oriented Architecture. *Journal of Engineering Science and Technology Review*, Eastern Macedonia and Thrace Institute of Technology, v. 13, p. 212–234, 2020. Citado 3 vezes nas páginas 15, 49 e 50.

NERI, D. et al. Design Principles, Architectural Smells and Refactorings for Microservices: A Multivocal Review. *SICS Software-Intensive Cyber-Physical Systems*, Springer, v. 35, n. 1, p. 3–15, 2020. Citado na página 60.

NORVAIsA, E.; PACKEVIcIUS, S. Service Oriented Architecture Evaluation Based on Maintainability Index. In: *Advances in Software Engineering*. [S.l.]: CEUR-WS, 2017. v. 1856, p. 704 – 721. Citado na página 49.

NURAINI, A.; WIDYANI, Y. Software with service oriented architecture quality assessment. *Proceedings of 2014 International Conference on Data and Software Engineering, ICODSE 2014*, Institute of Electrical and Electronics Engineers Inc., 2014. Citado 3 vezes nas páginas 15, 49 e 50.

OLIVEIRA, J.; VARGAS, M.; RODRIGUES, R. SOA reuse: Systematic Literature Review Updating and Research Directions. *ACM International Conference Proceeding Series*, Association for Computing Machinery, p. 612–624, 2018. Citado na página 50.

OUMOUSSA, I.; SAIDI, R. Evolution of Microservices Identification in Monolith Decomposition: A Systematic Review. *IEEE Access*, v. 12, p. 23389–23405, 2024. Citado 2 vezes nas páginas 13 e 19.

OUYANG, S. et al. *LLM is Like a Box of Chocolates: the Non-determinism of ChatGPT in Code Generation*. 2023. Citado 2 vezes nas páginas 20 e 23.

PAN, J. J.; WANG, J.; LI, G. Vector Database Management Techniques and Systems. *ACM Computing Surveys*, Association for Computing Machinery, New York, NY, USA, p. 597–604, 2024. Citado 3 vezes nas páginas 21, 22 e 23.

PEREPLETCHIKOV, M.; RYAN, C.; FRAMPTON, K. Cohesion Metrics for Predicting Maintainability of Service-Oriented Software. *Journal of Systems and Software*, p. 328–335, 2007. Citado 2 vezes nas páginas 43 e 44.

PULCINELLI, L. E. G.; PEDROSO, D. F.; BRUSCHI, S. M. Conceptual and Comparative Analysis of Application Metrics in Microservices. *IEEE Access*, p. 123–130, Oct 2023. Citado na página 12.

PULPARAMBIL, S. et al. Service Design Metrics to Predict IT-Based Drivers of Service Oriented Architecture Adoption. In: IEEE. *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. [S.l.], 2018. p. 1–7. Citado na página 30.

PULPARAMBIL, S. et al. Service Design Metrics to Predict IT-Based Drivers of Service Oriented Architecture Adoption. *2018 9th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2018*, Institute of Electrical and Electronics Engineers Inc., 2018. Citado na página 50.

RAJ, V.; RAVICHANDRA, S. Microservices: A Perfect SOA Based Solution for Enterprise Applications Compared to Web Services. *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology, RTEICT 2018 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., p. 1531–1536, 2018. Citado 2 vezes nas páginas 50 e 51.

RAJ, V.; SADAM, R. Evaluation of SOA-based Web Services and Microservices Architecture using Complexity Metrics. *SN Computer Science*, Springer, v. 2, p. 1–10, 2021. Citado 4 vezes nas páginas 9, 15, 16 e 30.

REN, W.; BARRETT, S.; DAS, S. Toward Gamification to Software Engineering and Contribution of Software Engineer. *O artigo **"Toward Gamification to Software Engineering and Contribution of Software Engineer"** foi publicado na **IEEE Transactions on Software Engineering**. Esta revista é uma das principais publicações sobre engenharia de software, abordando inovações e práticas na área, incluindo tópicos sobre gamificação.Se precisar de mais informações sobre o artigo ou ajuda para acessá-lo, estou à disposição!*, p. 1–5, 01 2020. Citado na página 26.

RIGUEIRA, F.; BERNARDINO, J.; PEDROSA, I. Extraction of Information from Log Files Using Python Programming and Tableau. *International Journal of Advanced Computer Science and Applications (IJACSA)*, p. 1–7, 2020. Citado na página 49.

ROCHA, F.; SOARES, M.; RODRIGUEZ, G. Patterns in Microservices-based Development: A Grey Literature Review. In: *Anais do XXVI Congresso Ibero-Americano em Engenharia de Software*. Porto Alegre, RS, Brasil: SBC, 2023. p. 61–76. Citado na página 10.

ROCHA, F.; SOARES, M.; RODRIGUEZ, G. Patterns in Microservices-based Development: A Grey Literature Review. *Journal of Software: Evolution and Process*, SBC, Porto Alegre, RS, Brasil, p. 61–76, 2023. Citado 2 vezes nas páginas 15 e 16.

RODRIGUEZ, G. et al. Understanding and addressing the allocation of microservices into containers: A review. *IETE Journal of Research*, Taylor & Francis, p. 1–14, 2023. Citado 4 vezes nas páginas 10, 13, 15 e 19.

RODRIGUEZ VIRGINIA YANNIBELLI, F. G. R. D. B. I. M. A. G.; MENEZES, P. M. Understanding and Addressing the Allocation of Microservices into Containers: A Review. *IETE Journal of Research*, Taylor & Francis, v. 0, n. 0, p. 1–14, 2023. Citado na página 16.

SALTON, G.; WONG, A.; YANG, C. S. A Vector Space Model for Automatic Indexing. *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 18, n. 11, p. 613–620, nov 1975. Citado 5 vezes nas páginas 20, 21, 22, 23 e 24.

SANTOS, A.; PAULA, H. Microservice Decomposition and Evaluation Using Dependency Graph and Silhouette Coefficient. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, p. 51–60, 09 2021. Citado 3 vezes nas páginas 13, 16 e 50.

SANTOS, N.; SILVA, A. R. A Complexity Metric for Microservices Architecture Migration. In: *IEEE 17TH INTERNATIONAL CONFERENCE ON SOFTWARE ARCHITECTURE (ICSA 2020)*. 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1264 USA: IEEE COMPUTER SOC, 2020. p. 169–178. Citado 2 vezes nas páginas 9 e 12.

SCHRöER, C.; WITTFOTH, S.; GMEZ, J. M. A Process Model for Microservices Design and Identification. In: *2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C)*. [S.l.: s.n.], 2021. p. 1–8. Citado na página 12.

SHAN, H. et al. -Diagnosis: Unsupervised and real-time Diagnosis of Small-Window Long-Tail Tatency in Large-Scale Microservice Platforms. *The Web Conference 2019 - Proceedings of the World Wide Web Conference, WWW 2019*, Association for Computing Machinery, Inc, p. 1453 – 1462, 2019. Citado na página 50.

SHANMUGASUNDARAM, G.; VENKATESAN, V. P.; DEVI, C. P. Research opportunities in service reusability of service oriented architecture. In: *2012 International Conference on Emerging Trends in Science, Engineering and Technology (INCOSET)*. [S.l.: s.n.], 2012. p. 396–403. Citado na página 50.

SHARON, I. et al. A Decision Framework for Selecting a Suitable Software Development Process. In: FILIPE, J.; CORDEIRO, J. (Ed.). *ICEIS 2010 - Proceedings of the 12th International Conference on Enterprise Information Systems, Volume 3, ISAS, Funchal, Madeira, Portugal, June 8 - 12, 2010*. [S.l.]: SciTePress, 2010. p. 34–43. Citado 2 vezes nas páginas 57 e 59.

SHEIKH, A.; AMBHAIKAR, A. Quality of Services Parameters for Architectural Patterns of IoT. *Journal of Information Technology Management*, University of Tehran, v. 13, 2021. Citado 5 vezes nas páginas 11, 15, 16, 49 e 50.

SINGH, J.; GUPTA, V. Text Stemming: Approaches, Applications, and Challenges. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 49, n. 3, p. 1762 – 1808, sep 2016. Citado na página 22.

SOARES, M. S.; FRANÇA, J. M. S. Characterization of the Application of Service-Oriented Design Principles in Practice: A Systematic Literature Review. *Journal of Software*, v. 11, n. 4, p. 403–417, 2016. Citado 2 vezes nas páginas 19 e 56.

SOLINGEN, R. V. et al. Goal Question Metric (GQM) Approach. *Encyclopedia of Software Engineering*, Wiley Online Library, 2002. Citado 2 vezes nas páginas 31 e 34.

THÖNES, J. Microservices. *IEEE Software*, IEEE, v. 32, n. 1, p. 116–116, 2015. Citado na página 61.

TORAMAN, C. et al. Impact of Tokenization on Language Models: An Analysis for Turkish. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, Association for Computing Machinery, New York, NY, USA, v. 22, n. 4, p. 3832 – 3853, mar 2023. Citado 2 vezes nas páginas 21 e 22.

TOUEIR, A.; BROISIN, J.; SIBILLA, M. Toward Configurable Performance Monitoring: Introduction to Mathematical Support for Metric Representation and Instrumentation of the CIM Metric Model. *5th Int. DMTF Academic Alliance Workshop on Systems and Virtualization Management: Standards and the Cloud*, Paris, 2011. Citado 2 vezes nas páginas 15 e 50.

TUMMALAPALLI, S. et al. Detection of Web Service Anti-Patterns Using Weighted Extreme Learning Machine. *Computer Standards Interfaces*, v. 82, p. 103–123, 2022. Citado 2 vezes nas páginas 16 e 50.

VALE, G. et al. Summary of The Artifact Accompanying the Article : "Designing Microservice Systems Using Patterns: An Empirical Study on Quality Trade-Offs". *IEEE Transactions on Software Engineering*, p. 57–57, 03 2022. Citado 3 vezes nas páginas 19, 50 e 51.

VASWANI, A. et al. Attention is all You Need. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2017. (NIPS'17), p. 6000–6010. Citado na página 20.

VERA-RIVERA, F. H.; GAONA, C.; ASTUDILLO, H. Defining and Measuring Microservice Granularity - A Literature Overview. *Peerj Computer Science*, PEERJ INC, London, England, v. 7, p. 3487 – 3498, SEP 8 2021. Citado 2 vezes nas páginas 50 e 51.

WEERASINGHE, L.; PERERA, I. An Exploratory Evaluation of Replacing ESB with Microservices in Service-Oriented Architecture. In: IEEE. *2021 International Research Conference on Smart Computing and Systems Engineering (SCSE)*. [S.l.], 2021. v. 4, p. 137–144. Citado 4 vezes nas páginas 13, 44, 45 e 58.

ZHANG, Q.; XINKE, L. Complexity Metrics for Service-Oriented Systems. In: *2009 Second International Symposium on Knowledge Acquisition and Modeling*. [S.l.: s.n.], 2009. v. 3, p. 375–378. Citado 6 vezes nas páginas 15, 49, 50, 51, 59 e 60.

ZHANG, Q.; XINKE, L. Complexity Metrics for Service-Oriented Systems. In: IEEE. *2009 second international symposium on knowledge acquisition and modeling*. [S.l.], 2009. v. 3, p. 375–378. Citado 3 vezes nas páginas 41, 42 e 43.

ZHANG, X.; GRACANIN, D. Service-Oriented-Architecture Based framework for Multi-User Virtual Environments. In: *2008 Winter Simulation Conference*. [S.l.: s.n.], 2008. p. 1139–1147. Citado na página 50.

ZHAO, W. et al. Towards Facilitating Development of SOA Application with Design Metrics. *Lecture Notes in Computer Science*, Chicago, IL, v. 4294 LNCS, p. 1863–1872, 2006. Citado 3 vezes nas páginas 49, 50 e 51.

ZHONG, T. et al. A Microservices Identification Method Based on Spectral Clustering for Industrial Legacy Systems. In: *2023 IEEE Globecom Workshops (GC Wkshps)*. [S.l.: s.n.], 2023. p. 1331–1337. Citado na página 50.

ZHOU, X. et al. Revisiting the Practices and Pains of Microservice Architecture in Reality: An industrial inquiry. *J. Syst. Softw.*, Elsevier Science Inc., USA, v. 195, n. C, p. 0164–1212, jan 2023. Citado 5 vezes nas páginas 9, 12, 13, 19 e 26.

ÜNLü, H. et al. Microservice-based Projects in Agile World: A Structured Interview. *Information and Software Technology*, v. 165, p. 107–134, 2024. Citado 2 vezes nas páginas 11 e 13.