UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FRANCISCO GUTENBERG DA SILVA FILHO

TAXONOMIA DE FALTAS PARA APLICAÇÕES BASEADAS EM MICROSSERVIÇOS

FORTALEZA

2023

FRANCISCO GUTENBERG DA SILVA FILHO

TAXONOMIA DE FALTAS PARA APLICAÇÕES BASEADAS EM MICROSSERVIÇOS

Dissertação apresentada ao Curso de do Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Engenharia de Software.

Orientadora: Profa. Dra. Valéria Lelli Leitão Dantas.
Coorientadora:
Profa. Dra. Rossana Maria de Castro Andrade.

FORTALEZA

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S58t Silva Filho, Francisco Gutenberg da.
Taxonomia de Faltas para Aplicações Baseadas em Microsserviços / Francisco Gutenberg da Silva Filho. – 2023.
104 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2023.
Orientação: Profa. Dra. Valéria Lelli Leitão Dantas.
Coorientação: Profa. Dra. Rossana Maria de Castro Andrade.

1. Teste de Software. 2. Microsserviço. 3. Engenharia de Software. I. Título.

CDD 005

FRANCISCO GUTENBERG DA SILVA FILHO

TAXONOMIA DE FALTAS PARA APLICAÇÕES BASEADAS EM MICROSSERVIÇOS

Dissertação apresentada ao Curso de do Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Engenharia de Software.

Aprovada em: 29/11/2023

BANCA EXAMINADORA

Profa. Dra. Valéria Lelli Leitão Dantas (Orientadora)
Universidade Federal do Ceará (UFC)

Profa. Dra. Rossana Maria de Castro Andrade (Coorientadora)
Universidade Federal do Ceará (UFC)

Prof. Dr. Windson Viana de Carvalho
Universidade Federal do Ceará (UFC)

Prof. Dr. Marcio Espíndola Freire Maia
Universidade Federal do Ceará (UFC)

Prof. Dr. Ismayle de Sousa Santos
Universidade Estadual do Ceará (UECE)

AGRADECIMENTOS

Primeiramente a Deus, por sempre iluminar meus caminhos e me manter forte na busca pelos meus objetivos. Gratidão!

A minha esposa, que tem me motivado e apoiado todos os dias. Obrigado por estar ao meu lado em todos os momentos. Te amo!

À minha família, expresso minha gratidão por todo o apoio incansável, pelo incentivo constante e pela educação valiosa que me concederam ao longo do meu caminho. Agradeço por acreditarem em mim e por serem a fonte inestimável de amor e suporte.

À minha orientadora, Profa. Dra Valéria Lelli Leitão Dantas, por desde o início ter me apoiado, incentivado e acreditado no meu potencial.

À todo o corpo docente e de servidores do Centro de Ciências da UFC que efetivamente contribuíram com minha formação, em especial os professores do Laboratório do Grupo de Pesquisa Grupo de Redes de Computadores, Engenharia de Software e Sistemas (GREat).

A todos os amigos que fiz durante esta jornada e que estiveram comigo nos momentos mais importantes deste ciclo.

"Seja a mudança que você quer ver no mundo."

(Mahatma Gandhi)

RESUMO

As aplicações baseadas em microsserviços suportam um estilo arquitetural que permite a organização de aplicações distribuídas como um conjunto de serviços independentes para alcançar escalabilidade e capacidade de manutenção. Essas aplicações têm sido amplamente utilizadas na indústria, no entanto, sua arquitetura pode trazer desafios relacionados à tolerância a falhas, prevenção de falhas, detecção de falhas e atividades de tratamento de falhas. No contexto de microsserviços, a injeção de faltas é mais complexa, pois esses aplicativos contêm principalmente processos assíncronos e, nesse caso, a injeção de faltas se torna ainda mais dispendiosa. Nessa direção, o uso de uma taxonomia de faltas é benéfico para o planejamento e execução de atividades de teste relacionadas, por exemplo, as técnicas de teste baseadas em faltas. Vários estudos na literatura têm tratado da complexidade dos testes de aplicações baseadas em microsserviços, tais como, a investigação das causas de falhas de microsserviços ou o desenvolvimento de *frameworks* que permitam aos usuários injetar faltas relacionadas com microsserviços. Dessa forma, o objetivo deste trabalho é definir uma taxonomia de faltas para aplicações baseadas em microsserviços a fim de melhor apoiar o desenvolvimento e testes dessas aplicações. Neste trabalho, foi primeiramente conduzido um mapeamento sistemático para identificar faltas relacionadas com microsserviços. Os resultados e contribuições deste trabalho englobam diversos aspectos. Primeiramente, apresenta-se um catálogo abrangente contendo 136 faltas, das quais 33 são distribuídas em 11 categorias. Em seguida, com base no catalogo foi definida uma taxonomia preliminar utilizando um esquema de classificação de Requisitos Não Funcionais e Características de Microsserviços que que classifica 117 dessas faltas. A evolução desta taxonomia culmina em uma versão final, que inclui 106 faltas, também categorizadas por seis RNFs e correlacionadas com 11 características da arquitetura de microsserviços. O esquema de classificação da taxonomia baseado em 6 RNFs (*e.g.,* Desempenho, Segurança, Confiabilidade) e a correlação das faltas com 11 características (*e.g.,* Tempo de Execução, Armazenamento de dados) inerentes à arquitetura de microsserviços. Por fim, o trabalho oferece um relato abrangente do processo de criação e avaliação da taxonomia, fornecendo uma compreensão sobre sua aplicabilidade.

Palavras-chave: teste de software; microsserviço; engenharia de software; injeção de faltas.

ABSTRACT

Microservice-based applications support an architectural style that allows the organization of distributed applications as a set of independent services to achieve scalability and maintainability. They have been widely used in the industry, however, this architecture can bring challenges regarding fault Tolerance, Fault Prevention, Fault Detection and Fault Handling activities. In the context of microservices, fault injection is more complex since those applications contains mostly assynchronous process. Therefore, a fault injection becomes even more expensive. Therefore, the above activities that require tasks such as fault injection and debugging become even more expensive. This way, the use of a fault taxonomy can be beneficial for performing software fault activities. Several studies in the literature have been handling the complexity of testing microservice-based applications such as investigating the root causes of microservices failures or providing *frameworks* that allow users injecting faults related to microservices. In this work, we conducted a systematic mapping study to catalog faults related to microservice-based applications to better support their development and testing and then create a fault taxonomy. The results and contributions of this study encompass various facets. Initially, a comprehensive catalog is presented, comprising 136 faults, of which 103 are distributed across 11 categories. Subsequently, a preliminary taxonomy is proposed, classifying 117 of these faults in accordance with Non-Functional Requirements (NFRs) and correlating them with specific characteristics of microservices architecture. The evolution of this taxonomy culminates in a final version that includes 106 faults, also categorized by NFRs and correlated with characteristics of microservices architecture. Additionally, the taxonomy's classification scheme is presented, predicated on 6 NFRs (e.g., Performance, Security, Reliability), alongside the correlation of faults with 14 characteristics (e.g., Runtime, Data Storage) inherent to microservices architecture. Lastly, the work furnishes a comprehensive account of the taxonomy creation and evaluation process, affording insights into its applicability.

Keywords: software testing; microservice; software engineer.

LISTA DE FIGURAS

Figura 1 – Arquitetura de Microsserviços vs.Arquitetura Monolítica (HAT, 2018) . . . 18
Figura 2 – Metodologia de Pesquisa 25
Figura 3 – Visão geral dos resultados da Revisão Multivocal da Literatura 32
Figura 4 – Resultados por base de dados na Rodada 1 - Leitura Resumo 33
Figura 5 – Resultados por base de dados na Rodada 2 - Leitura Completa 34
Figura 6 – Resultados do Snowballing 34
Figura 7 – Criação do Catálogo de Faltas 35
Figura 8 – Visão geral das atividades de definição da Taxonomia 41
Figura 9 – Respostas por Questão - Grupo Focal 75
Figura 10 – Respostas por Questão - Método *Delphi* 79

LISTA DE TABELAS

Tabela 1 – Comparação dos Trabalhos Relacionados 24
Tabela 2 – String de Busca 27
Tabela 3 – Critérios de Inclusão 27
Tabela 4 – Critérios de Exclusão 27
Tabela 6 – 103 Faltas não classificadas 36
Tabela 5 – 33 Faltas Identificadas e Classificadas na Literatura 40
Tabela 7 – Agrupamento de Faltas do Catálogo de Faltas 42
Tabela 8 – Distribuição de Faltas na Taxonomia Final 50
Tabela 9 – Comportamento Temporal (22 Faltas) 51
Tabela 10 – Utilização de Recursos (06 Faltas) 55
Tabela 11 – Capacidade (07 Faltas) 56
Tabela 12 – Recuperabilidade (08 Faltas) 57
Tabela 13 – Disponibilidade (04 Faltas) 59
Tabela 14 – Autenticidade (10 Faltas) 60
Tabela 15 – Integridade (6 Faltas) 62
Tabela 16 – Confidencialidade (17 Faltas) 63
Tabela 17 – Disponibilidade (02 Faltas) 66
Tabela 18 – Analisabilidade (14 Faltas) 66
Tabela 19 – Interoperabilidade (5 Faltas) 69
Tabela 20 – Completude Funcional (2 Faltas) 70
Tabela 21 – Distribuição de Faltas na Taxonomia Final e Preliminar 71
Tabela 22 – Perfil dos Especialistas 73
Tabela 23 – Respostas do questionário de Avaliação utilizando Método *Delphi* 80
Tabela 24 – Faltas relacionadas à *RNFs* na literatura 85
Tabela 25 – Publicações Relacionadas ao Tema 89
Tabela 26 – Publicações Não Relacionadas ao Tema 89

SUMÁRIO

1 INTRODUÇÃO 12
1.1 Objetivos e Questões de Pesquisa 13
1.2 Método da Pesquisa 14
1.3 Contribuições 15
1.4 Organização da Dissertação 15
2 FUNDAMENTAÇÃO TEÓRICA 17
2.1 Aplicações Baseadas em Microsserviços 17
2.1.1 Webservice 19
2.2 Teste de Software 19
2.2.1 Teste Baseado em Faltas 20
2.3 Taxonomia de Faltas 20
2.4 Considerações Finais 21
3 TRABALHOS RELACIONADOS 22
3.1 Trabalhos identificados 22
3.2 Considerações Finais 24
4 METODOLOGIA DE PESQUISA 25
4.1 Planejamento 26
4.1.1 Definição de Objetivos 26
4.1.2 Definição de Questões de Pesquisas 26
4.1.3 Seleção de bases de dados 26
4.1.4 Definição de Filtros e String de Busca 27
4.1.5 Definição de Critérios de Parada e Seleção 27
4.1.6 Protocolo de Revisão 28
4.2 Seleção de Artigos 28
4.3 Extração e Análise 29
4.4 Definição da Taxonomia 29
4.5 Avaliação da Taxonomia 30
4.6 Considerações Finais 30
5 CATÁLOGO DE FALTAS 32
5.1 Resultados da Revisão Multivocal da Literatura 32

5.2 Catálogo de Faltas . . . 35
5.3 Considerações Finais . . . 39
6 TAXONOMIA DE FALTAS PARA APLICAÇÕES BASEADAS EM MICROSSERVIÇOS . . . 41
6.1 Esquema de Classificação . . . 41
6.1.1 Requisitos Não Funcionais . . . 44
6.1.1.1 Desempenho . . . *44*
6.1.1.2 Segurança . . . *44*
6.1.1.3 Confiabilidade . . . *45*
6.1.1.4 Manutenibilidade . . . *45*
6.1.1.5 Compatibilidade . . . *45*
6.1.1.6 Funcionalidade . . . *46*
6.1.2 Características de Microsserviços . . . 46
6.2 Taxonomia Preliminar de Faltas para Microsserviços . . . 48
6.3 Taxonomia de Faltas para Microsserviços . . . 49
6.4 Faltas de Desempenho . . . 50
6.5 Faltas de Confiabilidade . . . 57
6.6 Faltas de Segurança . . . 60
6.7 Faltas de Manutenibilidade . . . 66
6.8 Faltas de Compatibilidade . . . 68
6.9 Faltas de Funcionalidade . . . 70
6.10 Taxonomia Preliminar versus Taxonomia Final . . . 70
6.11 Considerações Finais . . . 71
7 AVALIAÇÃO DA TAXONOMIA . . . 73
7.1 Perfil dos Especialistas . . . 73
7.2 Grupo Focal . . . 73
7.3 Método Delphi . . . 77
7.4 Considerações Finais . . . 81
8 DISCUSSÕES E AMEAÇAS À VALIDADE . . . 83
8.1 Ameaças à Validade . . . 85
8.1.1 Ameaças Internas . . . 85
8.1.1.1 Ameaças Relacionadas à Identificação de Estudos Primários . . . *85*

8.1.1.2 Ameaças Relacionadas à Extração de Dados 86
8.1.2 Ameaças Externas 86
8.1.2.1 Ameaças Relacionadas à Avaliação da Taxonomia 86
8.2 Considerações Finais 87
9 CONCLUSÃO E TRABALHOS FUTUROS 88
9.1 Resultados Alcançados 88
9.2 Produção Bibliográfica 89
9.2.1 Publicações Relacionadas ao Tema 89
9.2.2 Publicações Não Relacionadas ao Tema 89
9.3 Trabalhos Futuros 90
REFERÊNCIAS 91
APÊNDICE A – PROTOCOLO DE REVISÃO 95
APÊNDICE B – TAXONOMIA PRELIMINAR DE FALTAS 98
APÊNDICE C – TEMPLATE DO CHECKLIST - GRUPO FOCAL . . 102
APÊNDICE D – TEMPLATE DO QUESTIONÁRIO - MÉTODO DELPHI 104

1 INTRODUÇÃO

O avanço da tecnologia e a transformação digital revolucionaram diversos setores industriais, como o comércio eletrônico, as finanças e a educação. A transformação digital tem desempenhado um papel fundamental na aceleração do desenvolvimento de arquiteturas de aplicativos baseados em microsserviços na indústria de Tecnologia da Informação (TI). Essas arquiteturas oferecem suporte para o progresso rápido e escalável da transformação digital em diferentes áreas e departamentos de empresas e organizações (HERBST *et al.*, 2018). Essa mudança significativa permitiu a adaptação de serviços e soluções para o ambiente digital, impulsionando a eficiência, a agilidade e a inovação em um cenário altamente competitivo. Portanto, a adoção de arquiteturas de microsserviços tornou-se essencial para atender às demandas da era digital.

Os microsserviços representam uma abordagem arquitetônica inovadora com o propósito de acelerar os ciclos de implantação, aprimorar a manutenção e aumentar a escalabilidade de aplicativos de software, como discutido por Newman (NEWMAN, 2015). Entretanto, a adoção dessa arquitetura não está isenta de desafios. As atividades relacionadas à Injeção de Faltas, Tolerância a Faltas, Prevenção de Faltas, Tratamento de Faltas e Detecção de Faltas, conforme apontado por Chen *et al.* (2020), se destacam como elementos cruciais que merecem atenção. Lidar com esses desafios torna-se essencial para garantir a robustez e a confiabilidade de sistemas baseados em microsserviços, uma vez que as faltas podem impactar significativamente a experiência do usuário e a integridade do software. Portanto, o entendimento e a abordagem adequada dessas questões tornam-se fundamentais no contexto das arquiteturas de microsserviços.

Os desafios de lidar com faltas em microsserviços são intrincados devido à natureza distribuída e modular dessa arquitetura. A identificação e isolamento de faltas são difíceis, pois uma falta em um serviço pode se manifestar como uma falha em outro, exigindo sistemas robustos de logging e tracing. A correção de faltas é desafiadora, pois atualizar um microsserviço sem causar regressões em outros requer uma gestão cuidadosa de versões e um processo de integração contínua bem definido. Garantir a consistência de dados e a integridade transacional é crucial, já que uma falta em um microsserviço pode levar a estados inconsistentes, necessitando de mecanismos como compensações transacionais e sagas. A comunicação entre serviços é outra fonte de faltas, onde problemas de rede ou incompatibilidades de API podem introduzir comportamentos indesejados e falhas.

A atividade de Injeção de Faltas consiste em inserir faltas em um código com o objetivo de avaliar a capacidade de um sistema de continuar operando na presença de faltas, ou seja, a tolerância a faltas em um sistema. Assim, a atividade de Tolerância a Faltas visa prevenir faltas que levam a falhas em um sistema. Ela tem duas fases: Detecção de Faltas e Recuperação do sistema. A atividade de Tratamento de Faltas pode ser aplicada após a detecção de uma falta. (HöLLER *et al.*, 2015).

Nesse sentido, com diferentes graus de formalização, as técnicas de teste baseadas em faltas (GAO *et al.*, 2015) elaboram casos de teste para revelar categorias de faltas prováveis ou predefinidas. Tais técnicas têm sido aplicadas para lidar com a complexidade dos testes de aplicativos baseados em microsserviços (AUé *et al.*, 2018) (CHEN *et al.*, 2020). Além disso, esses aplicativos possuem algumas particularidades, como processos assíncronos (ZHOU *et al.*, 2021) que tornam a atividade de injeção de faltas mais complexa. Por exemplo, Aue et al. (AUé *et al.*, 2018) investigaram a ocorrência de falhas com base na frequência e impacto dos consumidores de API, e suas descobertas mostraram que as falhas de integração de API podem estar relacionadas a 11 causas raizes gerais, como entrada de usuário inválida ou ausente e dados de solicitação inválidos ou expirados.

1.1 Objetivos e Questões de Pesquisa

Dada a crescente relevância das arquiteturas baseadas em microsserviços e a complexidade inerente à injeção de faltas nesses contextos, o propósito deste trabalho é a criação de uma taxonomia de faltas. Essa taxonomia desempenha um papel multifacetado, servindo como um artefato de suporte a diversas atividades essenciais no domínio de desenvolvimento de software. Entre os principais papéis que se beneficiarão dessa taxonomia, destacam-se:

- Apoiar Arquitetos de Software na concepção da arquitetura de um aplicativo em desenvolvimento ou migração de legado;
- Apoiar Desenvolvedores de Software a pensar sobre possíveis faltas e suas formas de tratamento antes mesmo de codificar;
- Apoiar Engenheiros de Teste no planejamento e construção de testes baseados em faltas;
- Apoiar Engenheiros DevOps na construção de um pipeline de integração contínua com injeção de faltas; e
- Apoiar Pesquisadores na contribuição para o estado da arte tanto na academia quanto na indústria.

Nesse contexto, a criação de uma taxonomia de faltas para aplicações que se baseiam em microsserviços apresenta um potencial significativo para beneficiar tanto a comunidade de pesquisadores quanto os profissionais da indústria envolvidos nas diversas etapas do Ciclo de Vida de Desenvolvimento de Software (do inglês *Software Development Life Cycle* - SDLC). O SDLC abrange todo o espectro de atividades desde a concepção e *design* de software até a implementação, testes, operação e manutenção.

Com o intuito de guiar eficazmente o desenvolvimento e resultados deste estudo, foram investigar as seguintes questões de pesquisa:

QP1 Quais faltas ocorrem em aplicações baseadas em microsserviços?

QP2 Como as faltas são classificadas em aplicações baseadas em microsserviços?

QP2 Quais RNFs estão relacionados às faltas identificadas em aplicações baseadas em microsserviços?

1.2 Método da Pesquisa

A definição da taxonomia de faltas proposta neste trabalho teve seu início com a realização de uma Revisão Multivocal da Literatura (do inglês, *Multivocal Literature Review* - MLR), com base nos procedimentos propostos por Garousi *et al.* (2019) e (KITCHENHAM, 2004). Essa abordagem foi adotada com o intuito de identificar as faltas que possivelmente estão relacionadas a aplicativos que se baseiam em microsserviços. No decorrer desse processo, foram identificadas um total de 151 ocorrências de faltas e falhas no contexto investigado. Dessas, 33 faltas foram devidamente classificadas em 11 categorias distintas, com base em uma seleção criteriosa de 31 estudos relevantes. Essas categorias abrangem uma ampla variedade de aspectos, incluindo o Gerenciamento de Instâncias, Configuração de Ambiente, Gerenciamento de Chamadas, e diversas outras áreas críticas.

A avaliação da taxonomia de faltas foi realizada por meio de dois métodos: Grupo Focal e Método Delphi. Seis especialistas em Microsserviços e Engenharia de Software foram convidados a participar, sendo que todos possuíam pelo menos seis anos de experiência nas áreas relevantes, incluindo trabalhos anteriores com Requisitos Não Funcionais. A avaliação do Grupo Focal qualitativa envolveu a reunião de especialistas para discussões aprofundadas sobre a taxonomia, proporcionando informações detalhadas sobre percepções e experiências. Após o refinamento com base no Grupo Focal, uma segunda avaliação foi conduzida utilizando o Método Delphi, no qual um painel de especialistas anônimos respondeu a uma série de questionários

para alcançar consenso e previsões. Essas avaliações garantiram a qualidade e relevância da taxonomia.

1.3 Contribuições

Este trabalho apresenta contribuições para o entendimento e a abordagem de faltas em aplicações baseadas em microsserviços, fornecendo insumos para profissionais e pesquisadores. As principais contribuições incluem:

- **Catálogo de Faltas:** um catálogo abrangente contendo 136 faltas, das quais 33 são distribuídas em 11 categorias. Este catálogo é uma base fundamental para a compreensão das faltas comuns em aplicações baseadas em microsserviços.
- **Taxonomia de Faltas:** a versão final da taxonomia, com 106 faltas categorizadas por RNFs e relacionadas com características de arquitetura de microsserviços. Esta taxonomia refinada representa um avanço significativo na compreensão e na classificação das faltas em aplicações baseadas em microsserviços.
- **Esquema de Classificação:** um esquema de classificação da taxonomia de acordo com seis RNFs e suas oito subcaracterísticas, além da correlação das faltas com 11 características inerentes à arquitetura de microsserviços. Esse esquema oferece uma visão abrangente das faltas em relação aos requisitos e características específicas da arquitetura de microsserviços.
- **Relato do Processo de Avaliação:** um relato detalhado do processo de criação e avaliação da taxonomia, fornecendo uma compreensão abrangente sobre sua aplicabilidade e relevância no contexto de sistemas baseados em microsserviços.

Em suma, as contribuições apresentadas neste trabalho não apenas abordam lacunas cruciais na compreensão de falhas em sistemas baseados em microsserviços, mas também fornecem ferramentas valiosas para profissionais e pesquisadores no campo.

1.4 Organização da Dissertação

O restante deste trabalho está organizado da seguinte forma:

Capítulo 2 - Fundamentação Teórica: esse capítulo estabelece os fundamentos, contextualizando o leitor com as características distintivas da arquitetura de microsserviços. Explora conceitos essenciais e define terminologias pertinentes, proporcionando uma base sólida

para a compreensão mais aprofundada apresentada nos capítulos subsequentes.

Capítulo 3 - Trabalhos Relacionados: destaca a relevância do trabalho atual em relação às pesquisas prévias. Uma revisão crítica da literatura revela avanços existentes, lacunas no conhecimento e posiciona este trabalho em um contexto mais amplo.

Capítulo 4 - Metodologia: oferece uma visão detalhada da metodologia empregada na pesquisa. Essencial para proporcionar transparência quanto às abordagens de coleta e análise de dados utilizadas, visa garantir a validade e confiabilidade dos resultados.

Capítulo 5 - Catálogo de Faltas: apresenta um catálogo abrangente de falhas em microsserviços, detalhando 136 faltas, das quais 33 são distribuídas em 11 categorias.

Capítulo 6 - Taxonomia de Faltas: propõe uma taxonomia preliminar e final, categorizando 117 e 106 faltas, respectivamente, com base em Requisitos Não Funcionais (RNFs) e características da arquitetura de microsserviços.

Capítulo 7 - Avaliação da Taxonomia: apresenta como a taxonomia foi avaliada e os resultados obtidos na avaliação.

Capítulo 8 - Discussão: analisa detalhadamente os resultados obtidos, identificando e abordando ameaças à validade para uma compreensão mais profunda das implicações práticas e teóricas do trabalho. Além disso, foram respondidas as questões de pesquisa neste capitulo.

Capítulo 9 - Conclusões e Trabalhos Futuros: apresenta as conclusões finais, recapitula os principais resultados e delineia caminhos para pesquisas futuras.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo é dedicado à apresentação dos fundamentos teóricos que sustentam o contexto e os conceitos centrais desta dissertação. Compreender esses fundamentos é essencial para uma apreciação abrangente da arquitetura de microsserviços e dos desafios relacionados a ela, bem como para o desenvolvimento de estratégias eficazes de teste e prevenção de faltas nesse ambiente dinâmico.

2.1 Aplicações Baseadas em Microsserviços

A arquitetura de microsserviços é um estilo arquitetônico introduzido com a finalidade de possibilitar a organização de aplicativos distribuídos como uma coleção de serviços possivelmente sem estado, com o intuito de alcançar escalabilidade, separação de preocupações e facilidade de manutenção (NEWMAN, 2015). Empresas renomadas, tais como Netflix[1], Amazon[2] e Facebook[3], empregam extensivamente a arquitetura de microsserviços para disponibilizar múltiplas atualizações por minuto (SAVOR *et al.*, 2016). Algumas dessas empresas estão inclusive adotando o paradigma de computação sem servidor, levando a funcionalidade sem estado a um extremo. A arquitetura de microsserviços é frequentemente associada à tecnologia baseada em contêineres no contexto de processos de implantação contínua, permitindo a atualização, implantação e operação de serviços de forma rápida e iterativa (CHEN, 2018).

A Figura 1 ilustra a comparação entre arquiteturas monolítica e baseada em microsserviços, destacando as principais diferenças entre essas abordagens. No modelo monolítico, toda a aplicação é construída como uma única unidade, onde os componentes estão fortemente interligados, o que pode resultar em desafios na escalabilidade e manutenção, além de dificultar a implementação de mudanças em partes específicas do sistema. Por outro lado, a arquitetura de microsserviços divide a aplicação em serviços menores e independentes, que podem ser desenvolvidos, implantados e escalados de maneira separada, oferecendo maior flexibilidade e agilidade. Essa abordagem, embora promova maior escalabilidade e resiliência, também exige um gerenciamento mais complexo e a coordenação entre os diferentes serviços.

As aplicações de microsserviços têm angariado popularidade devido ao seu design fino e aos serviços fracamente acoplados que implementam funcionalidades limitadas, propor-

[1] https://www.netflix.com/
[2] https://www.amazon.com/
[3] https://www.facebook.com/

cionando maior flexibilidade de escalabilidade. Essas aplicações permitem a implantação de serviços individuais em Máquinas Virtuais (VMs) fisicamente separadas. Cada microsserviço atua como um processo independente e autônomo, comunicando-se com outros microsserviços por meio de Interfaces de Programação de Aplicativos (APIs). Com frequência, cada processo desse tipo é encapsulado em um contêiner individual, visando obter maior flexibilidade e gerenciabilidade da aplicação, embora também seja admissível a execução de múltiplos serviços em um único contêiner. Uma das principais vantagens das aplicações de microsserviços é a capacidade de dimensionar um serviço específico, em vez de lançar múltiplas instâncias de toda a aplicação.

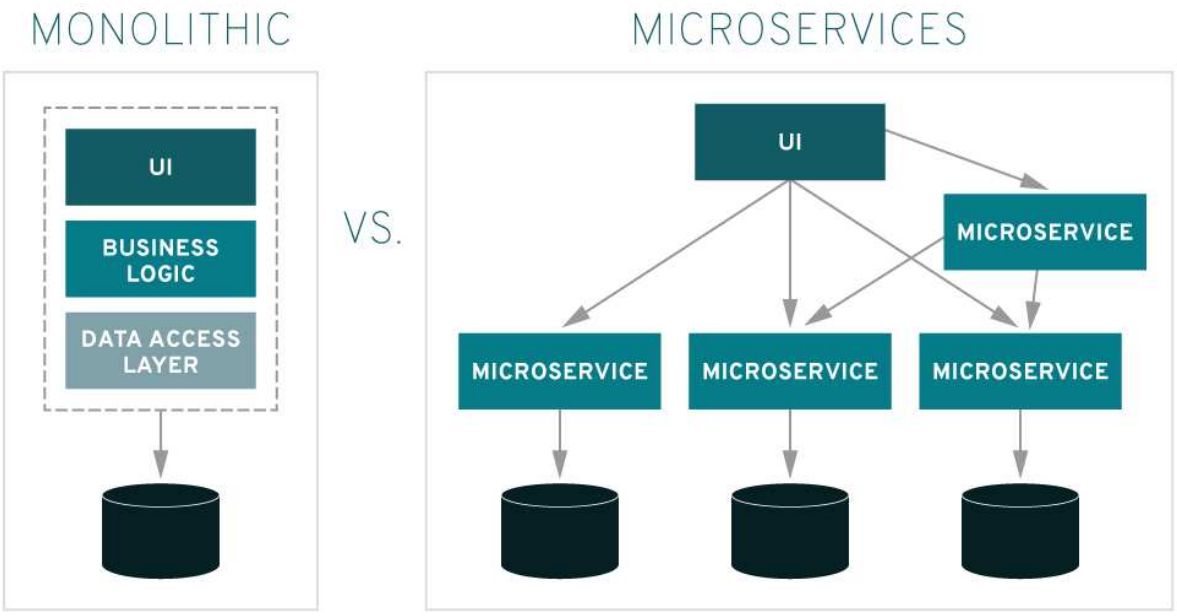

Figura 1 – Arquitetura de Microsserviços vs.Arquitetura Monolítica (HAT, 2018)

A adoção desse estilo arquitetônico proporciona diversas vantagens às aplicações que o implementam. Por exemplo, ao decompor o aplicativo em microsserviços independentes, torna-se possível escalar e desenvolver a arquitetura de forma independente. Cada serviço pode, portanto, possuir sua própria pilha tecnológica (linguagem de programação, *frameworks* e ferramentas), tornando a manutenção do serviço menos dependente de tecnologias ou linguagens de programação específicas. No entanto, é importante destacar que essa arquitetura também implica desafios a serem enfrentados por profissionais da indústria e pesquisadores.

A descentralização característica da arquitetura de microsserviços introduz complexidades que encarecem algumas atividades. Por exemplo, a injeção de falhas (CHEN *et al.*, 2020) e a depuração (ZHOU *et al.*, 2021) tornam-se atividades mais complexas. Em uma arquitetura

monolítica, há apenas um bloco central onde é possível injetar ou depurar falhas. No entanto, na arquitetura de microsserviços, existem vários serviços independentes que se comunicam de forma assíncrona, exigindo abordagens distintas para testes e solução de problemas.

2.1.1 Webservice

O *webservice*, uma das muitas tecnologias que suportam a entrega de serviços pela web ou HTTP, pode ser implementado utilizando o conceito de microsserviços. Após a implementação da arquitetura de microsserviços, o aplicativo é subdividido em serviços, cada um com uma responsabilidade claramente definida. Esses serviços são executados como processos isolados e, frequentemente, são distribuídos em servidores distintos. Embora operem de maneira independente, a troca de informações entre esses processos é, muitas vezes, crucial para o funcionamento do aplicativo. Portanto, os webservices desempenham um papel fundamental na arquitetura de microsserviços, facilitando a comunicação entre os diversos serviços do aplicativo.

É relevante ressaltar que o conceito de microsserviço, devido à sua flexibilidade, encontra aplicações em uma variedade de contextos. Dentre as aplicações que podem implementar a arquitetura de microsserviços, destacam-se: aplicações web (Webservices) (PRASAD *et al.*, 2017) e aplicativos móveis (HIGASHINO, 2017); aplicações em nuvem (WU *et al.*, 2018); e aplicações de Internet das Coisas (PALLEWATTA *et al.*, 2019).

2.2 Teste de Software

A atividade de teste, dentro do ciclo de vida de desenvolvimento de software, desempenha um papel fundamental, uma vez que é responsável por identificar e priorizar erros em um software, com o objetivo primordial de garantir a qualidade do produto final. Além disso, os resultados dos testes também são verificados para identificar possíveis anomalias ou avaliar os atributos não funcionais do sistema (SOMMERVILLE, 2011). No entanto, a execução desta atividade pode se tornar complexa e dispendiosa, especialmente em sistemas complexos, como as aplicações baseadas em microsserviços. Isso ocorre devido à presença de múltiplos processos e comunicações assíncronas entre os diversos microsserviços que compõem tais aplicações.

2.2.1 Teste Baseado em Faltas

O teste baseado em faltas, conhecido como *Fault-based Testing*, tem como objetivo demonstrar a ausência de faltas pré-definidas em um sistema sob teste (MORELL, 1990). Essa abordagem de teste tem se mostrado eficaz em cenários onde a lógica e as regras do sistema podem ser especificadas ou modeladas em expressões booleanas (WEYUKER *et al.*, 1994).

Existem duas técnicas comuns de teste baseado em faltas: o Teste Baseado em Erros (*Error Guessing*) e o Teste de Mutação. No primeiro, os casos de teste são projetados para antecipar as faltas mais prováveis com base em experiência e histórico de faltas em projetos anteriores (BOURQUE *et al.*, 2014). O objetivo é garantir a detecção de tipos comuns de faltas introduzidas pelos desenvolvedores por meio de um conjunto limitado de entradas de teste. Quando um sistema não apresenta falhas ao ser testado com esse conjunto, pode-se inferir que está livre dessas faltas específicas (DANIEL; SIM, 2012).

Por outro lado, o teste de mutação é uma abordagem computacionalmente custosa, mas eficaz, baseada em faltas (CHAN *et al.*, 2005). Essa técnica envolve a aplicação de operadores de mutação que introduzem mudanças sintáticas nas declarações do sistema, criando mutantes, ou seja, cópias defeituosas do código original. No contexto do teste de mutação, a injeção de faltas é frequentemente necessária, tornando-se uma atividade desafiadora e onerosa (PINHEIRO *et al.*, 2018). É nesse contexto que uma taxonomia de faltas desempenha um papel relevante, ajudando a orientar a injeção de faltas e aprimorar as estratégias de teste de mutação.

2.3 Taxonomia de Faltas

O conceito de taxonomia remonta a Carolus Linnaeus (LINDLEY, 1836) e pode ser entendido como um esquema de classificação ou categorização. No contexto desta dissertação, a taxonomia de faltas tem como objetivo fornecer uma estrutura organizada para a classificação de faltas em aplicações baseadas em microsserviços, contribuindo para diversos estudos relacionados à prevenção, tolerância e tratamento de faltas.

Uma taxonomia de faltas é composta por categorias que representam conjuntos de valores característicos de uma falta, podendo também incluir relações e dependências entre essas categorias. Diversas taxonomias de faltas têm sido propostas em diferentes contextos, como a "Fault Taxonomy for Event-Based Systems"(HUMMER *et al.*, 2012) e a "Fault Taxonomy for Wireless Networks"(BHAGYAVATI, 2005).

No âmbito deste trabalho, a distinção entre os termos "falta"e "falha"é fundamental. Considera-se uma "falta"como uma condição anormal que pode causar a "falha"de um elemento ou item do sistema. Uma "falha"ocorre quando o sistema não executa uma função requerida dentro dos requisitos especificados, resultando em um comportamento inesperado. Além disso, os termos "bug", "defeito", "anomalia", "erro"e "defeito"são considerados sinônimos de "falta".

A taxonomia de faltas proposta neste estudo tem como objetivo contribuir para uma compreensão mais profunda das faltas em aplicações baseadas em microsserviços, auxiliando na melhoria das estratégias de teste, prevenção e tratamento de faltas nesse contexto específico.

2.4 Considerações Finais

Neste capítulo, foi estabelecida a base conceitual essencial para esta dissertação. Explorou-se a arquitetura de microsserviços, compreendendo seus princípios, desafios e integração com tecnologias-chave, como contêineres. Foi investigado o papel crítico do teste de software, enfocando abordagens baseadas em faltas, como o teste de mutação. Introduziu-se a taxonomia de faltas como uma ferramenta valiosa para categorização e organização de faltas. Foram definidos termos fundamentais, como "falta", "erro", e "falha.".

3 TRABALHOS RELACIONADOS

Este capítulo apresenta uma revisão dos trabalhos relacionados à temática desta dissertação. O objetivo é analisar estudos prévios que abordaram questões relacionadas à classificação e taxonomia de faltas em sistemas de software, com um foco especial em aplicações baseadas em microsserviços. Embora não tenham sido encontrados estudos que tenham explorado a catalogação ou taxonomia de faltas especificamente para aplicações baseadas em microsserviços, alguns trabalhos relevantes na literatura discutem falhas ou smells em sistemas similares.

3.1 Trabalhos identificados

O trabalho de (PONCE *et al.*, 2022) aborda a importância crucial da segurança em sistemas de microsserviços, uma vez que muitas empresas de tecnologia da informação estão adotando essas arquiteturas para impulsionar seus negócios. O estudo concentra-se na identificação de 'security smells' que podem potencialmente afetar aplicativos baseados em microsserviços, resultando em possíveis vazamentos de segurança e necessidade de refatorações para mitigar os impactos desses 'security smells'. A abordagem metodológica adotada envolveu uma revisão multivocal abrangente da literatura branca e cinza sobre o tema da segurança em microsserviços. Durante essa análise, foram minuciosamente examinados 58 estudos primários publicados no período de 2011 até o final de 2020. Como resultado desse esforço, dez 'security smells' foram identificados e organizados em uma taxonomia, cada um associado às propriedades de segurança que podem ser violadas e às refatorações recomendadas para atenuar seus efeitos prejudiciais. Os resultados deste estudo oferecem um valor pragmático considerável para profissionais que trabalham com microsserviços, fornecendo orientações concretas para abordar questões de segurança em suas atividades cotidianas. Além disso, essas descobertas servem como um ponto de partida valioso para pesquisadores que desejam explorar novas direções de pesquisa relacionadas à segurança em microsserviços, contribuindo assim para o avanço contínuo da área tecnológica.

Por outro lado, o trabalho de (SIDDIQUI *et al.*, 2023) explora as implicações da Internet das Coisas (IoT), que viabiliza a conectividade entre objetos físicos e uma ampla gama de aplicações em diferentes domínios. Esse nível de conectividade em larga escala apresenta desafios funcionais e não funcionais significativos, os quais podem ser atenuados por meio de

arquiteturas e técnicas de software adequadas. Nesse contexto, a arquitetura de microsserviços surge como um paradigma recente projetado para lidar com muitos desses desafios. Este estudo apresenta uma revisão abrangente do estado da arte no que diz respeito à arquitetura baseada em microsserviços para sistemas IoT. No âmbito deste artigo, é analisado como essa arquitetura é atualmente empregada para aprimorar as características não funcionais de sistemas IoT, com foco particular na confiabilidade e disponibilidade. É importante destacar que esta revisão representa o primeiro mapeamento abrangente das arquiteturas de microsserviços aplicadas à IoT, identificando tanto suas virtudes quanto suas limitações. Para conduzir este estudo, foram formuladas perguntas de pesquisa específicas e estabelecida uma taxonomia destinada a sistemas IoT baseados em microsserviços. As respostas às questões de pesquisa proporcionam uma visão do estado da arte, enquanto a associação dos artigos pesquisados com a taxonomia definida oferece uma compreensão mais detalhada do que já foi realizado. A análise resultante realça o potencial promissor da arquitetura de microsserviços para sistemas IoT, especialmente no que tange à melhoria da confiabilidade e disponibilidade. No entanto, também evidencia algumas fragilidades nas soluções existentes, apontando para a necessidade de pesquisas adicionais para lidar de maneira mais eficaz com a confiabilidade e disponibilidade nos sistemas IoT.

Gill e Buyya (GILL; BUYYA, 2020) propuseram uma taxonomia de gerenciamento de falhas para aplicações em nuvem. Esta taxonomia tem como objetivo identificar técnicas de confiabilidade existentes que merecem atenção e devem ser investigadas com cuidado. As técnicas identificadas foram comparadas com base em características comuns e propriedades de gerenciamento de falhas (*e.g.,* implementadas em soluções comerciais e de código aberto). No entanto, os autores se concentram no gerenciamento de falhas para aplicações em nuvem e, portanto, não propõem uma taxonomia geral para microsserviços.

Hambatova et al. (HUMBATOVA *et al.*, 2020) propuseram uma taxonomia de faltas para sistemas de Aprendizado Profundo (DL). Os autores analisaram 1059 artefatos do Github e edições de projetos que usam frameworks populares para sistemas DL. Os autores também adicionaram à sua taxonomia, por meio de uma entrevista estrutural, problemas identificados por pesquisadores e profissionais em suas experiências. A taxonomia foi validada por meio de uma pesquisa com 21 desenvolvedores, na qual se observou que 13 das 15 categorias de faltas, ou a maioria delas, foram experimentadas por pelo menos 50% dos participantes. A principal diferença entre a taxonomia de (HUMBATOVA *et al.*, 2020) e esta taxonomia é que nosso trabalho se concentra em aplicações baseadas em microsserviços, enquanto aquele se concentra

em sistemas DL.

A Tabela 1 apresenta a comparação dos estudos mencionados. Pode-se observar que todos os estudos apresentaram algum tipo de classificação ou taxonomia, abordando diversos contextos, como Microsserviços, Aplicações em Nuvem, *Deep Learning* e Internet das Coisas. No entanto, vale ressaltar que, até o momento, não foi encontrado nenhum estudo com foco específico em taxonomia de faltas para aplicações baseadas em microsserviços, o que constitui o objetivo central deste trabalho.

Tabela 1 – Comparação dos Trabalhos Relacionados

Estudo	**Tipo**	**Contexto**
(PONCE *et al.*, 2022)	Taxonomia de Smells de Segurança	Microsserviço
(GILL; BUYYA, 2020)	Taxonomia de Gerenciamento de Falhas	Aplicações em Nuvem
(HUMBATOVA *et al.*, 2020)	Taxonomia de Falhas	*Deep Learning*
(SIDDIQUI *et al.*, 2023)	Taxonomia de Conceitos	Internet das Coisas
Este Trabalho	Taxonomia de Faltas	Microsserviços

3.2 Considerações Finais

Este capítulo analisou trabalhos relevantes sobre classificação e taxonomia em diferentes contextos, como segurança em microsserviços, confiabilidade em nuvem, *Deep Learning* e Internet das Coisas (IoT). Observou-se que, embora existam taxonomias para várias arquiteturas, não há um estudo específico sobre taxonomia de faltas para aplicações baseadas em microsserviços, o que destaca a importância e a novidade deste trabalho. Os trabalhos relacionados oferecem uma base teórica e metodológica que serve como referência para o desenvolvimento de uma taxonomia que categoriza e classifica faltas em microsserviços.

4 METODOLOGIA DE PESQUISA

A metodologia foi dividida em 18 atividades, conforme apresentadas na Figura 2. Para apoiar a execução dessas atividades, foram utilizadas as seguintes ferramentas:

- Google Sheets[1] para criar e gerenciar as planilhas relacionadas ao estudo (por exemplo, Formulário de Extração de Dados, Resultados do Mapeamento Sistemático e Taxonomia de Faltas);
- Parsifal[2] para gerenciar a revisão na literatura (importação e seleção de estudos);
- Trello[3] para gerenciar as atividades conduzidas durante a pesquisa; e
- Anonymous Open Science[4] para armazenar os artefatos da pesquisa.

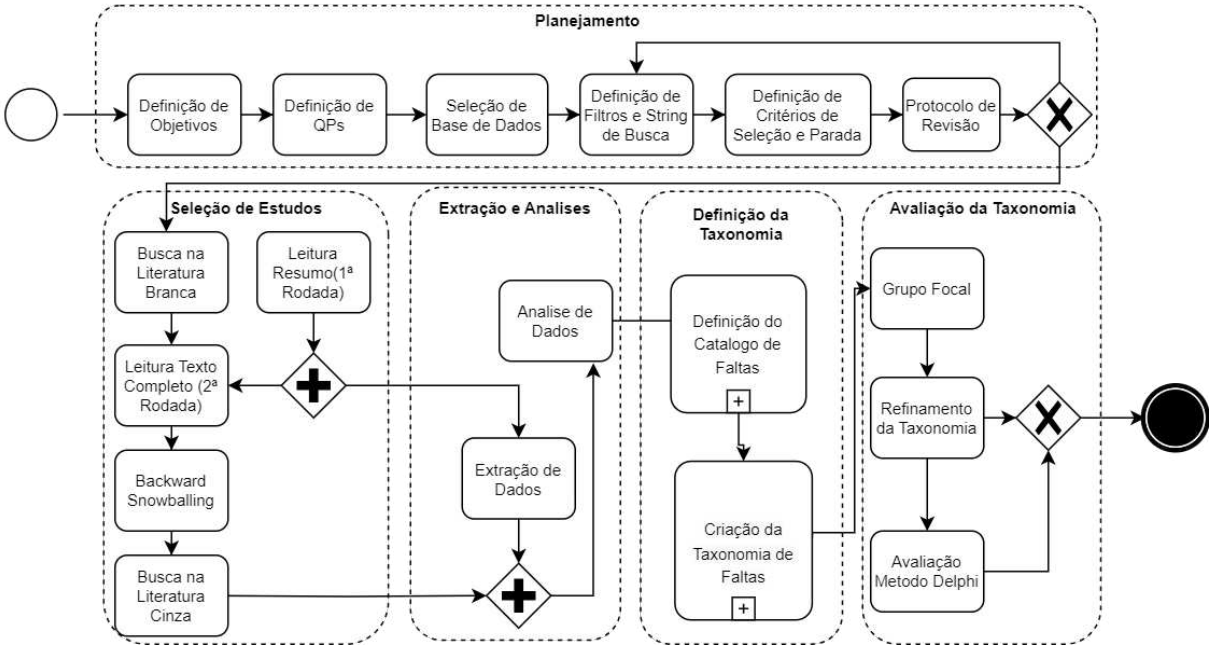

Figura 2 – Metodologia de Pesquisa

Nas próximas subseções são descritas as atividades conduzidas em cada fase da metodologia adotada para a criação da proposta neste trabalho.

[1] https://www.google.com/sheets/about/
[2] https://parsif.al/
[3] https://trello.com/
[4] https://anonymous.4open.science/

4.1 Planejamento

Nesta seção são descritas as atividades referentes ao planejamento detalhado adotado para a Revisão Multivocal da Literatura. As tarefas essenciais desse planejamento são exploradas nas subsubseções subsequentes, fornecendo uma visão geral das estratégias e abordagens utilizadas para guiar eficazmente o processo de revisão.

4.1.1 Definição de Objetivos

Foi conduzida a revisão multivocal da literatura com o objetivo de identificar estudos que discutissem as faltas relacionadas a microsserviços. A revisão visou fornecer uma compreensão ampla e abrangente das faltas em microsserviços, servindo como base essencial para a criação da taxonomia detalhada apresentada nesta dissertação.

4.1.2 Definição de Questões de Pesquisas

Para atingir os objetivos estipulados, as seguintes Questões de Pesquisa (QPs) foram definidas:

- **QP1** Quais faltas ocorrem em aplicações baseadas em microsserviços?
- **QP2** Como as faltas são classificadas em aplicações baseadas em microsserviços?
- **QP3** Quais requisitos não funcionais (RNFs) estão relacionados às faltas identificadas em aplicações baseadas em microsserviços?

4.1.3 Seleção de bases de dados

Nesta atividade, as bases de dados foram selecionadas. Inicialmente as pesquisas foram realizadas no ACM DL[5]. Em seguida, a base da Scopus[6] também foi adicionada, uma vez que ele indexa outras bases de dados, como o IEEE Xplore[7]. Para a Literatura Cinza (LC), decidiu-se utilizar o Google[8] e Stack Overflow[9].

[5] https://dl.acm.org/
[6] https://www.scopus.com/home.uri
[7] https://ieeexplore.ieee.org/Xplore/home.jsp
[8] https://www.google.com/
[9] https://stackoverflow.com/

Tabela 2 – String de Busca

(webservice OR microservice) AND (fault OR failure OR defect OR bug OR error OR anomaly) AND (classification OR taxonomy OR characterization OR categorization OR catalog)

4.1.4 Definição de Filtros e String de Busca

Nesta atividade, uma *string* de busca foi definida com base nas questões de pesquisa (ver Tabela 2) e esta foi executada nas bases de dados ACM DL e Scopus, com o objetivo de identificar estudos que abordem faltas em aplicações baseadas em microsserviços.

Além disso, foram definidos filtros para a base de dados da Scopus, por ela indexar várias outras bases e trazer resultados de diversas áreas, para refinar os resultados da pesquisa bibliográfica.

– Scopus: o filtro de pesquisa utilizado para os estudos foi por "Resumo"e por "Ciência da Computação", que é a nossa área de interesse.

Para a revisão na literatura cinza, foram adicionadas duas novas fontes de pesquisa: Google e Stackoverflow, comumente utilizadas na indústria.

4.1.5 Definição de Critérios de Parada e Seleção

Os critérios foram divididos em Critérios de Inclusão (CI) e Critérios de Exclusão (CE), ambos apresentados na Tabela 3 e na Tabela 4, respectivamente.

Tabela 3 – Critérios de Inclusão

ID	Critério
CI1	Artigos que discutem faltas em microsserviços
CI2	Artigos que discutem classificação de faltas em microsserviços
CI3	Artigos que são escritos em inglês

Tabela 4 – Critérios de Exclusão

ID	Critério
CE1	Artigos que não discutem explicitamente sobre microsserviços
CE2	Artigos que são resumos
CE3	Artigos cujo texto completo não está disponível
CE4	Artigos que são versões resumidas de outros

Para selecionar os artigos da LC, foi decidido aplicar o *Effort Bounded* de Garousi *et*

al. (2019), que consiste em incluir apenas um número específico de resultados de mecanismos de busca, ou seja limitando em esforço a pesquisa. No caso deste estudo, foram incluídos apenas os 140 primeiros resultados das bases de dados, visto que os resultados posteriores não eram relevantes.

4.1.6 Protocolo de Revisão

O protocolo apresenta o planejamento do estudo de mapeamento sistemático, descrevendo a *string* de busca, as bases de dados, os critérios de seleção e a forma de extração dos dados. O protocolo foi revisto por outros pesquisadores que participaram deste estudo. Durante essa revisão, foram feitos ajustes na *string* de busca proposta e refinados o filtro e os critérios definidos.

Conforme mencionado no início deste capítulo, a ferramenta Parsifal foi utilizada para gerenciar a revisão e, portanto, um relatório gerado pela ferramenta está disponível no Apêndice A.

4.2 Seleção de Artigos

Uma vez concluída a atividade de Planejamento, foi iniciada a atividade de Seleção de Estudos. Após a pesquisa na Literatura Branca, a seleção foi dividida em duas rodadas: Leitura do Resumo e Leitura do Trabalho Completo. A primeira consistiu em analisar todos os artigos através do Resumo e selecioná-los com base nos critérios de seleção descritos na Subseção 4.1.5. Com base nos resultados da primeira rodada, foi iniciada a segunda rodada e os artigos selecionados foram lidos na sua totalidade e, em seguida, foi realizada uma nova seleção com base nos critérios de seleção.

Com o intuito de ampliar a investigação dessa pesquisa, o passo seguinte foi a condução do *Backward Snowballing*, incluindo os três anos anteriores ao período atual. Neste passo, foram analisadas todas as referências de cada estudo selecionado, e os dois passos anteriores (Leitura de Resumos e Leitura de Artigos Completos) foram realizados novamente. Além disso, foi utilizada o Checklist de avaliação de qualidade da literatura cinza para engenharia de software proposta por Garousi *et al.* (2019) para a seleção das fontes de LC.

4.3 Extração e Análise

Simultaneamente à Seleção de Artigos, foi realizada a extração de dados dos artigos selecionados. Para a extração de dados foi criado um formulário com os seguintes itens:

- Referência
- Base de Dados
- Conferência
- Título do Artigo
- Autores
- Filiação
- Tipo de Estudo
- Objetivo da Pesquisa
- Questões de Pesquisa
- Metodologia
- Oportunidades de Pesquisa (Resultados)
- Tipo de Avaliação
- Desafios

4.4 Definição da Taxonomia

Quando a Extração de dados foi concluída, as faltas foram identificadas e catalogadas. Para catalogar as faltas, foi criada uma planilha na qual foram descritas as seguintes informações: ID, Categoria, Falha, Descrição, Requisito Não Funcional, Tag, Referência.

O catálogo de faltas serviu como artefato de entrada para esta atividade: Definição da taxonomia. Assim, primeiramente foi analisado o catálogo de faltas para definir a taxonomia desenvolvida, com foco nas categorias de faltas, na descrição da falta e no RNF com a qual uma falta pode estar relacionada. Na análise que foi efetuada, o objetivo foi compreender a falta em si. Além disso, pretendeu-se identificar faltas semelhantes relatadas em diferentes estudos e faltas que tivessem mais de uma classificação ou faltas que não tivessem classificação na literatura ou na indústria.

Dessa maneira, foi analisada a descrição de cada falta para melhor delinear a classificação da taxonomia. Em paralelo, buscou-se na literatura características de arquiteturas de microsserviços que pudessem apoiar o esquema de classificação da taxonomia. Como resul-

tado, foi selecionado o estudo de Martin Garriga (2018), por apresentar uma taxonomia de características de microsserviços elaborada a partir de uma revisão sistemática.

4.5 Avaliação da Taxonomia

Após a Taxonomia ter sido definida, ela passou por uma atividade de avaliação que consistiu de 2 etapas: Grupo Focal Krueger e Casey (2014) e Avaliação Utilizando o Metodo Delphi Dalkey e Helmer (1969). Primeiramente foi aplicado o Grupo Focal, que é uma técnica qualitativa de pesquisa que envolve a reunião de um grupo de participantes com conhecimento ou experiência relevantes sobre o tópico de pesquisa para explorar suas opiniões, perspectivas e experiências em um ambiente de discussão moderado. Após a aplicação desta técnica, os resultados foram analisados e houve um refinamento inicial da taxonomia. Uma nova avaliação foi feita após o refinamento, desta vez, utilizando o Método Delphi, que é uma abordagem de pesquisa que envolve uma série de rodadas de questionários e *feedback* de especialistas em um tópico específico.

A escolha de utilizar um Grupo Focal e o Método Delphi para avaliar a taxonomia é fundamentada em suas respectivas vantagens e aplicabilidades. Os grupos focais permitem a reunião de participantes com diversas perspectivas, promovendo discussões interativas que enriquecem a análise da taxonomia. Isso é particularmente relevante ao se considerar a necessidade de explorar a validade e aplicabilidade da taxonomia, bem como identificar potenciais lacunas ou áreas de melhoria. Além disso, o feedback qualitativo detalhado obtido por meio dos grupos focais pode aprimorar a taxonomia de maneira mais abrangente. Por outro lado, o Método Delphi foi escolhido para consolidar a opinião de especialistas, garantindo que a estrutura e as categorias da taxonomia sejam validadas. O anonimato dos especialistas no Método Delphi promove discussões imparciais e a obtenção de consenso, enquanto a iteração controlada permite refinamentos com base no feedback especializado, contribuindo assim para uma avaliação completa e confiável da taxonomia. Ambos os métodos se alinham de forma estratégica aos objetivos da avaliação da taxonomia.

4.6 Considerações Finais

Neste capítulo, foram delineadas as etapas seguidas para definir a taxonomia de faltas em microsserviços. A abordagem multivocal baseada nas diretrizes de Garousi e Kitchenham foi

adotada, e um planejamento detalhado conduziu as atividades de revisão bibliográfica e seleção de artigos. Ferramentas como Google Sheets, Parsifal, Trello e Anonymous Open Science foram utilizadas para apoiar o processo. As questões de pesquisa e os critérios de seleção foram definidos, seguidos pela seleção de bases de dados e a criação de uma *string* de busca. Os critérios de inclusão e exclusão foram estabelecidos, juntamente com um protocolo de revisão revisado. A atividade de seleção de artigos envolveu duas rodadas de leitura de resumos e leitura de trabalhos completos, seguidas pelo Backward Snowballing. A extração de dados e a definição da taxonomia foram as etapas subsequentes, permitindo a criação de um catálogo de faltas. A taxonomia foi avaliada por meio de um Grupo Focal e do Método Delphi.

Este capítulo serviu como base metodológica para o desenvolvimento da taxonomia de faltas em microsserviços, fornecendo insights sobre como as faltas foram identificadas e classificadas. As etapas de avaliação também asseguraram a validação e a aplicabilidade da taxonomia. Os próximos capítulos apresentarão em detalhes os resultados dessa pesquisa, destacando as faltas identificadas e a taxonomia final, fornecendo uma contribuição para a compreensão e aprimoramento da qualidade de microsserviços.

5 CATÁLOGO DE FALTAS

Neste capítulo, a criação do catálogo de faltas e os resultados da revisão multivocal da literatura que apoiaram esta criação são apresentados.

5.1 Resultados da Revisão Multivocal da Literatura

Esta Seção apresenta a síntese dos resultados da Revisão Multivocal da Literatura realizada para a identificação de faltas. A Figura 3 apresenta os resultados organizados em 3 etapas: Pesquisa na base de dados, Snowballing e Literatura Cinza. Essas etapas são apresentadas a seguir.

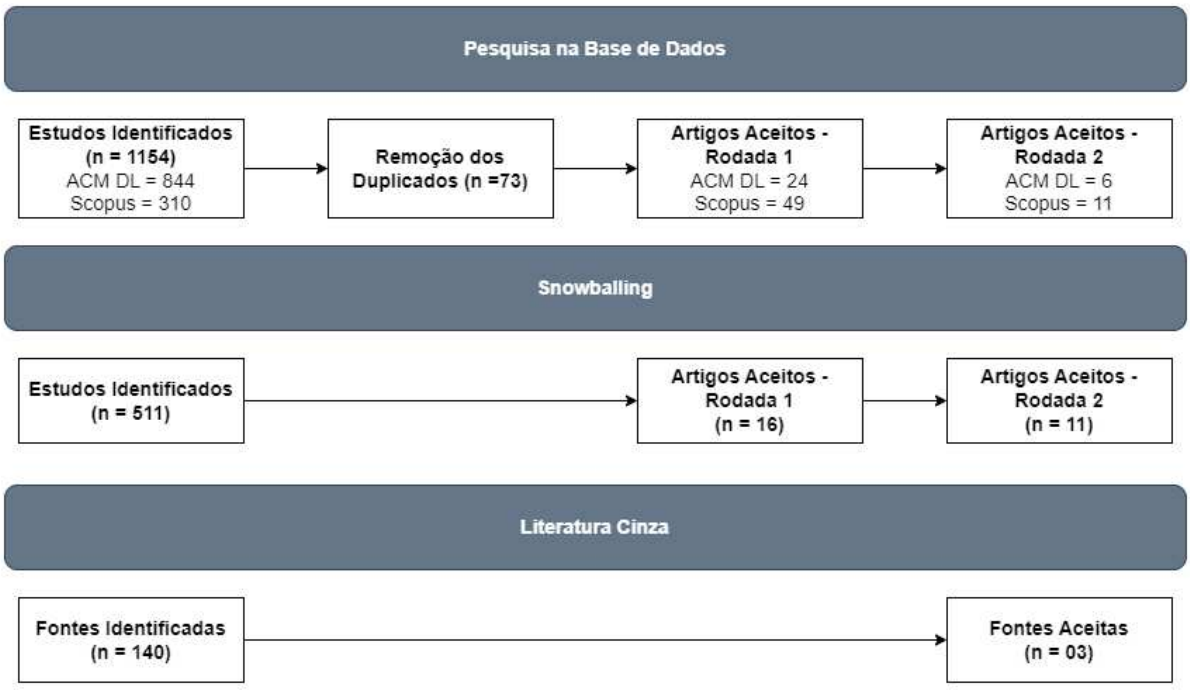

Figura 3 – Visão geral dos resultados da Revisão Multivocal da Literatura

- **Pesquisa na base de dados** conforme discutido na Subseção 4.2, a etapa de Pesquisa na Base de Dados se deu em duas rodadas, Leitura do Resumo (Rodada 1) e Leitura do Texto Completo (Rodada 2.) No entanto, inicialmente, foram identificados 1154 artigos, dos quais 844 foram retornados da ACM DL e 310 da Scopus. Destes 1154 artigos, 73 eram duplicados e portanto foram removidos, restando 1081 artigos únicos. Destes 1081 artigos, ao serem lidos os resumos, foram aceitos, com base nos critérios de inclusão e exclusão, 73 artigos, sendo 24 da ACM DL e 49 da Scopus. Finalmente, ao serem lidos os 73 artigos

por completo, foram aceitos, também com base nos critérios de inclusão e exclusão, 17 artigos, sendo 6 da ACM DL e 11 da Scopus.

- **Snowballing** nesta etapa, foi realizado um backward snowballing, onde foram analisadas todas as referencias dos 17 artigos selecionados na etapa anterior, totalizando 511 referências(estudos identificados). Assim como na etapa anterior, esta etapa foi realizada também em duas rodadas, portanto, inicialmente foi lido o resumo dos 511 artigos e foram selecionados 16 artigos, dos quais 11 foram selecionados após serem lidos por completo.
- **Literatura Cinza (LC)** conforme discutido na subseção 4.1.5, foi utilizada técnica *Effort Bounded*[1] de Garousi *et al.* (2019), foram incluídos 140 fontes da literatura cinza, das quais foram aceitas três fontes.

As Figuras 4 e 5 mostram, por rodada e base de dados, o número de artigos retornados pela busca, o número de artigos aceitos, rejeitados e duplicados.

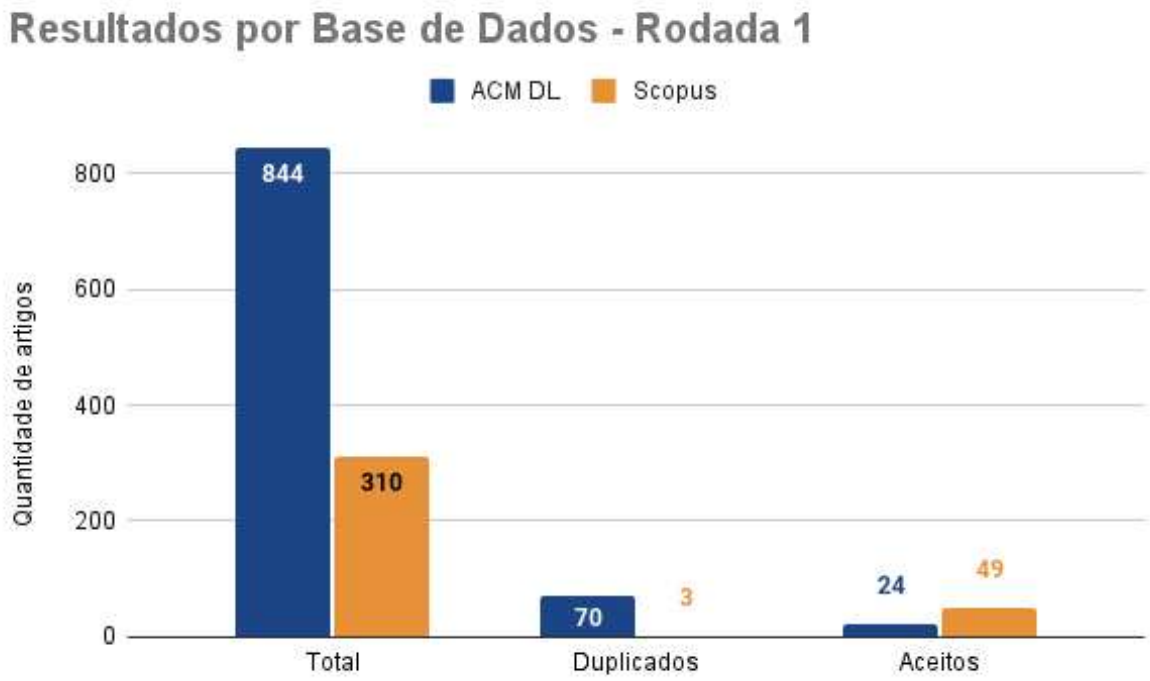

Figura 4 – Resultados por base de dados na Rodada 1 - Leitura Resumo

Pode-se observar que na Rodada 1 (Leitura do Resumo), a taxa de aceitação foi mais elevada na Scopus, onde 49 estudos dos 310 iniciais foram aceitos, representando 15,80%,

[1] O termo "Effort Bounded"de Garousi se refere a uma abordagem que limita o esforço necessário para realizar uma tarefa específica ou alcançar um objetivo, garantindo eficiência e economia de recursos.

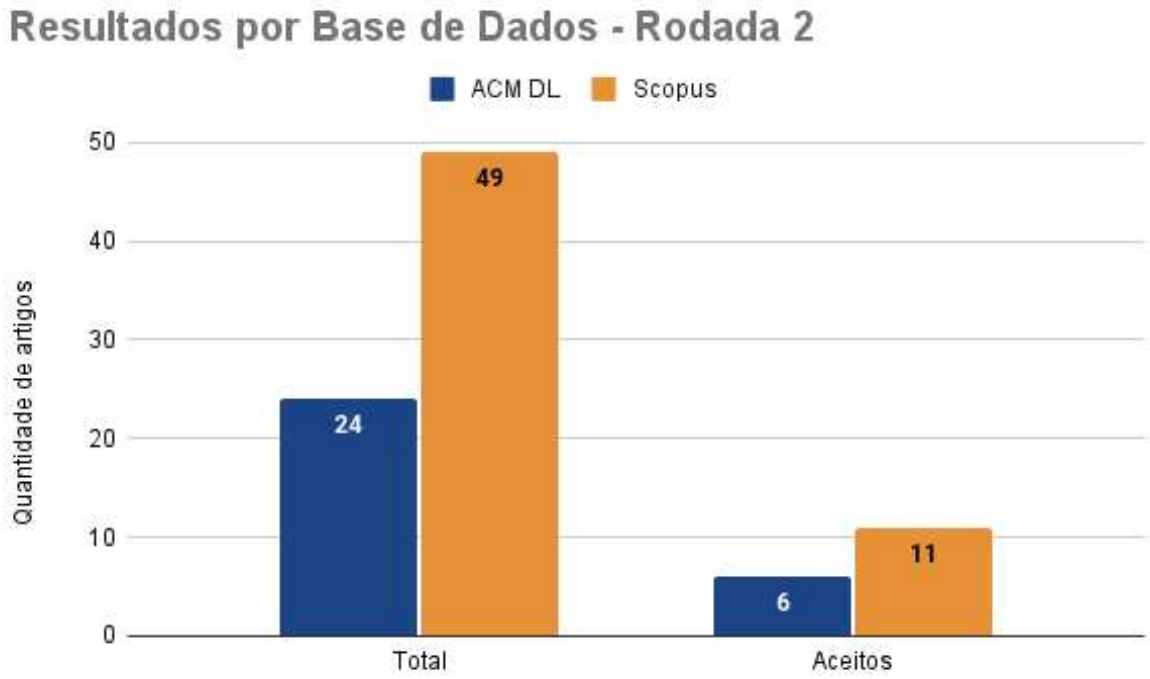

Figura 5 – Resultados por base de dados na Rodada 2 - Leitura Completa

enquanto na Rodada 2 (Leitura do Texto Completo), a taxa de aceitação foi mais elevada na ACM DL, na qual 6 estudos dos 24 iniciais foram aceitos, representando 25%.

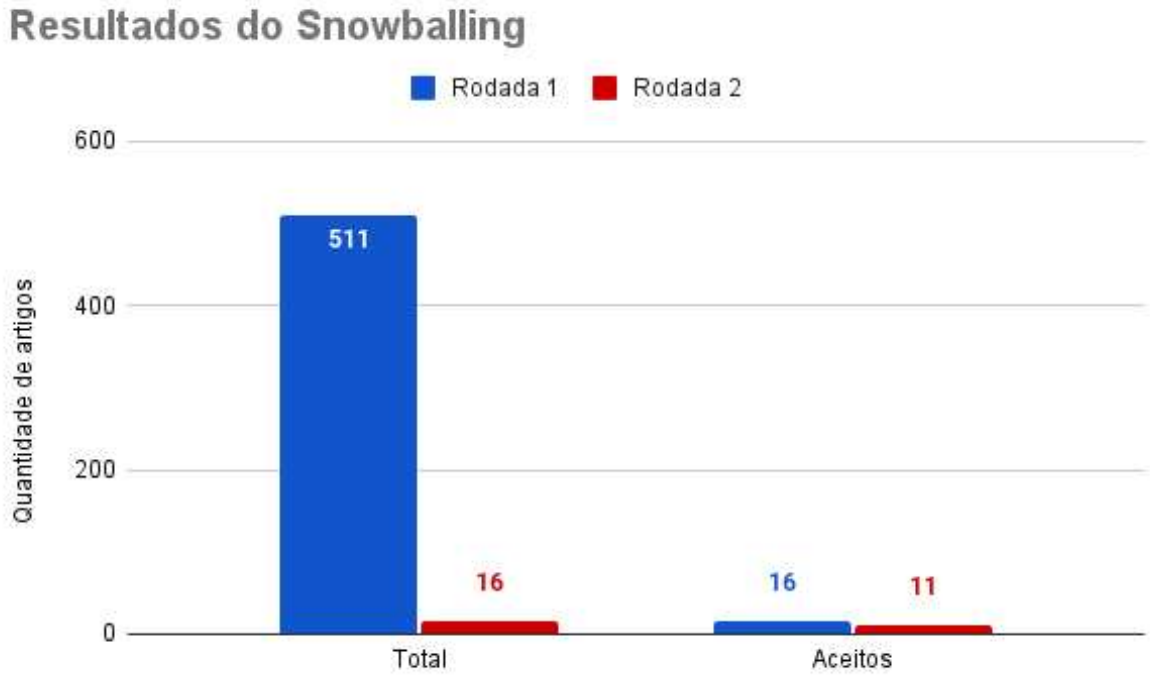

Figura 6 – Resultados do Snowballing

Em relação aos resultados do *Snowballing*, a Figura 6 mostra a quantidade de artigos aceitos em cada rodada. Comparando os resultados da etapa *Backward Snowballing*, pode-se observar que a taxa de aceitação foi muito mais elevada na Rodada 2 (11 de 16 artigos) do que na Rodada 1 (16 de 511). Além disso, a taxa de aceitação final do Snowballing, onde 11 estudos dos 16, na rodada de leitura completa, foram aceitos, representando 68,7% foi superior à dos artigos selecionados nas bases de dados.

Na etapa de Pesquisa na Base de Dados, foram selecionados 17 artigos. Com base nesse conjunto, foi realizado o *Backward* Snowballing e, após a triagem, foram adicionados mais 11 artigos, totalizando 28 artigos. Em relação à LC, conforme mostrado na Figura 3, foram identificadas 140 fontes e, após a aplicação dos critérios de parada, três fontes foram selecionadas e a taxa de aceitação foi de 2,14%.

5.2 Catálogo de Faltas

Durante a extração de dados realizada nos 31 estudos (28 da literatura branca e 3 da literatura cinza), foi identificado um total de 151 ocorrências de falhas e faltas, das quais 128 foram extraídas da Literatura Branca, 21 da Literatura Cinza e 2 de ambos. A lista completa dos estudos selecionados, as suas falhas e faltas estão disponíveis no repositório deste estudo[2].

O catálogo de faltas, compilado a partir da análise dos estudos, é a base para a criação da taxonomia de faltas que será discutida no Capítulo 6. Este catálogo permite uma estrutura de catalogação e compreensão das faltas em aplicações baseadas em microsserviços. A Figura 7 apresenta as tarefas conduzidas para a criação do Catálogo de Faltas.

Figura 7 – Criação do Catálogo de Faltas

As tarefas realizadas para criação do catálogo de faltas são descritas abaixo:

- **Identificação das Faltas e Falhas** inicialmente foi feita uma identificação de 151 ocorrências entre faltas e falhas, onde foram atribuídos identificadores (IDs) e identificado se havia algum tipo de categorização prévia nessa ocorrências.

[2] https://github.com/gutenbergf/Fault-Taxonomy-for-Microservice-Based-Applications

- **Análise de Faltas** foi realizada uma análise das 151 ocorrências para identificar quais delas eram realmente faltas, já que a string de busca utilizada neste estudo também considerou o termo “falha”. Com base nesta análise, 15 falhas foram removidas do catálogo, uma vez que são falhas em vez de faltas.
- **Catalogação de Faltas** nesta tarefa, foram reatribuídos ids apenas nas faltas (excluídas as falhas) e às suas respectivas categorias (se houvesse). Além disso, este catálogo foi apresentado em formato de planilha[3].

Como resultado de todas as tarefas realizadas na criação do catálogo de faltas, chegou-se à versão final do catálogo que apresenta 136 faltas distintas. É importante notar que, inicialmente, havia sido identificadas um total de 151 ocorrências entre faltas e falhas. No entanto, após análise , o catálogo foi refinado, excluindo 15 das ocorrências que originalmente eram consideradas faltas, mas que se revelaram ser, na verdade, falhas. A lista das faltas já classificadas na literatura (33) nas 11 categorias e as faltas não classificadas (103) são apresentadas nas Tabelas 5 e 6.

No que diz respeito à classificação das faltas, foram identificadas 33 faltas organizadas em 11 categorias e 103 faltas sem qualquer classificação. Além disso, nem todos os estudos apresentaram a mesma classificação. Por exemplo, a falta *CPU Hog* é citada por 4 estudos (WANG *et al.*, 2020)(AHMED *et al.*, 2017)(MARIANI *et al.*, 2018)(WANG *et al.*, 2016) mas apenas Wang et al. (WANG *et al.*, 2020) a classificou numa categoria denominada *Physical Resources*.

Ainda sobre as categorias encontradas na literatura, pode-se observar que boa parte das categorias se mostraram bem específicas, seja por só terem uma falta associada, como por exemplo as categorias *Implementation* (C1), *Instance Management* (C2), *Environment Configuration* (C3) e *Invocations Management* (C4) ou por ter faltas de um contexto muito específico, como por exemplo as categorias *Internal Change Anomaly* (C6), *External Change Anomaly* (C7) e *TOF Bug* (C11)

Tabela 6 – 103 Faltas não classificadas

ID Falta	**Falta**	**ID Falta**	**Falta**
F34	Invalid User Input Fault	F86	Memory Anomaly
Continua na próxima página			

[3] https://github.com/gutenbergf/Fault-Taxonomy-for-Microservice-Based-Applications

Tabela 6 – Continuação da Página Anterior

ID Falta	**Falta**	**ID Falta**	**Falta**
F35	Missing User Input Fault	F87	Availability Anomaly
F36	Expired request data Fault	F88	Functional Fault
F37	Invalid request data Fault	F89	Non-Functional Fault
F38	Missing Request data Fault	F90	Interaction Fault
F39	Insufficient permissions Fault	F91	Internal Fault
F40	Double processing Fault	F92	Environment Fault
F41	API Configuration Fault	F93	CPU error
F42	Missing server data Fault	F94	Memory errors
F43	API Internal Fault	F95	Disk errors
F44	Third party Fault	F96	Network packet loss and latency errors
F45	Point-wise	F97	Permanent Fault
F46	Collective anomalie	F98	Transient Fault
F47	Contextual anomalie	F99	Intermittent Fault
F48	Delete Fault	F100	CPU Load
F49	Disconnect Fault	F101	IO Load
F50	Data Type Probing	F102	Commit Leaks
F51	Malicious Control	F103	CPU cap problem
F52	Malicious Operation	F104	Memory cap problem
F53	Scan	F105	I/O interference problem
F54	Spying	F106	Packet loss problem
F55	Wrong Setup	F107	Flag setting bug
F56	Missing authentication	F108	Infinite wait bug
F57	Authentication bypass	F109	Infinite loop bug
F58	Relying on single factor authentication	F110	Infinite read bug
F59	Insufficient session management	F111	Thread shutdown bug

Continua na próxima página

Tabela 6 – Continuação da Página Anterior

ID Falta	Falta	ID Falta	Falta
F60	Downgrade authentication	F112	Disk I/O fault
F61	Insufficient crypto key management	F113	Network fault
F62	Missing authorization	F114	Process Crash Fault
F63	Missing access control	F115	Message Corruption Fault
F64	No re-authentication	F116	Deadlock Fault
F65	Unmonitored execution	F117	AccessDeniedException
F66	No context when authorizing	F118	IncompleteSignature
F67	Not revoking authorization	F119	InvalidAction
F68	Insecure data storage	F120	InvalidClientTokenId
F69	Insufficient credentials management	F121	InvalidParameterCombination
F70	Insecure data exposure	F122	InvalidParameterValue
F71	Use of custom/weak encryption	F123	InvalidQueryParameter
F72	Not validating input/data	F124	MalformedQueryString
F73	Insufficient auditing	F125	MissingAction
F74	Uncontrolled resource consumption	F126	MissingAuthenticationToken
F75	Computing resources exhaustion	F127	MissingParameter
F76	Network Delay Fault	F128	NotAuthorized
F77	Network Abortion Fault	F129	OptInRequired
F78	Conversation Error	F130	RequestExpired
F79	Metadata Fault	F131	ServiceUnavailable
F80	Data Handling Routines Fault	F132	ThrottlingException
F81	Latency Anomaly	F133	ThrottlingException
F82	Throughput Anomaly	F134	CPU Pressure

Continua na próxima página

Tabela 6 – Continuação da Página Anterior

ID Falta	Falta	ID Falta	Falta
F83	Power Anomaly	F135	Disk IO Jitter
F84	CPU Anomaly	F136	Network Latency
F85	I/O Anomaly		

Apesar destas 103 faltas não serem classificadas em categorias, existem algumas faltas que podem sugerir intuitivamente algum tipo de classificação apenas pelo seu nome, como por exemplo, CPU (F93), Memory (F94) e Disk Errors (F95) poderiam indicar que são relacionadas à desempenho. Do mesmo modo, Malicious Control (F51) e Malicious Operation (F52) poderiam indicar que são relacionadas à segurança.

5.3 Considerações Finais

Neste capítulo, foi apresentada a criação do Catálogo de Faltas para aplicações baseadas em microsserviços. Os resultados da Revisão Multivocal da Literatura foram discutidos, incluindo a busca em bases de dados, a etapa de snowballing e a inclusão de literatura cinza. Foram identificados 151 ocorrências de faltas e falhas nos estudos selecionados. Após uma análise inicial, foram identificadas 136 faltas distintas no catálogo, excluindo 15 ocorrências que se revelaram falhas. Destas 136 faltas, 33 foram classificadas em 11 categorias diferentes, enquanto as restantes 103 faltas permaneceram sem classificação.

É importante ressaltar que a classificação das faltas em categorias nem sempre foi consistente entre os estudos e algumas categorias eram muito específicas, contemplando apenas uma falta. As faltas não classificadas podem sugerir intuitivamente alguma classificação com base em seus nomes e descrições, como faltas relacionadas a desempenho ou segurança.

Este capítulo forneceu uma base para a construção da taxonomia e um entendimento das faltas em aplicações baseadas em microsserviços. No próximo capítulo, será apresentada a Taxonomia de Faltas para aplicações baseadas em microsserviços, usando o Catálogo de Faltas como base. Essa taxonomia fornecerá uma estrutura organizada para compreender e categorizar de forma mais assertiva as faltas em aplicações de microsserviços.

Tabela 5 – 33 Faltas Identificadas e Classificadas na Literatura

ID Categoria	Categoria	ID Falta	Fault
C1	Implementation	F01	Monolithic Fault
C2	Instance Management	F02	Multi-Instance Fault
C3	Environment Configuration	F03	C2 Configuration Fault
C4	Invocations Management	F04	Asynchronous Interaction Fault
C5	Workload	F05	Deadlock Bug
		F06	Infinit Wait
		F07	Unexpected Fault
		F08	Illegal Access Anomaly
C6	Internal Change Anomaly	F09	Application Change Anomaly
C7	External Change Anomaly	F10	System Change Anomaly
		F11	Remote Service Change Anomaly
C8	Hardware Usage	F12	High CPU usage due to mediation
		F13	High Memory usage due to mediation
		F14	High disk I/O due to mediation
C9	Performance	F15	Increased numbers of users
		F16	Long response time in back-end services
		F17	Increased message size
		F18	CPU Hog
		F19	Network Congestion
		F20	Memory Leak
		F21	IO Interference
		F22	Insufficient CPU Allocation
		F23	Insufficient Memory Allocation
		F24	Performance degradation
		F25	Data Flushing
		F26	Drop Table
C10	VM Instance	F27	Resource Usage Fault
		F28	Memomy Usage Fault
		F29	Application Bug
		F30	S.O Kernel Bug
		F31	Random System Termination Fault
C11	TOF Bug	F32	Crash-Regular Bug
		F33	Crash-Recovery Bug

6 TAXONOMIA DE FALTAS PARA APLICAÇÕES BASEADAS EM MICROSSERVIÇOS

Para a criação da taxonomia de faltas, foi definido um esquema de classificação baseado em RNFs e características de microsserviços e, posteriormente, a análise do catálogo de faltas com o intuito de mapear as faltas para esse esquema bem como reagrupar faltas similares.

Os detalhes para a criação da taxonomia são apresentados nas próximas subseções.

6.1 Esquema de Classificação

A Figura 8 apresenta uma visão geral das atividades da etapa de "Definição da Taxonomia". Após, o catálogo de faltas, com 103 faltas classificadas e 33 faltas não classificadas, foi criado. Com base nesse catálogo, foi visto que algumas faltas propostas por diferentes estudos eram semelhantes. Então, foi analisado o significado dessas faltas e observou-se que elas poderiam ser agrupadas. Por exemplo, 20 faltas estão relacionadas a componentes de hardware, tais como CPU, disco e memória. Foram identificadas cinco faltas relacionadas com Memória, oito faltas relacionadas com CPU e sete faltas relacionadas com Disco.

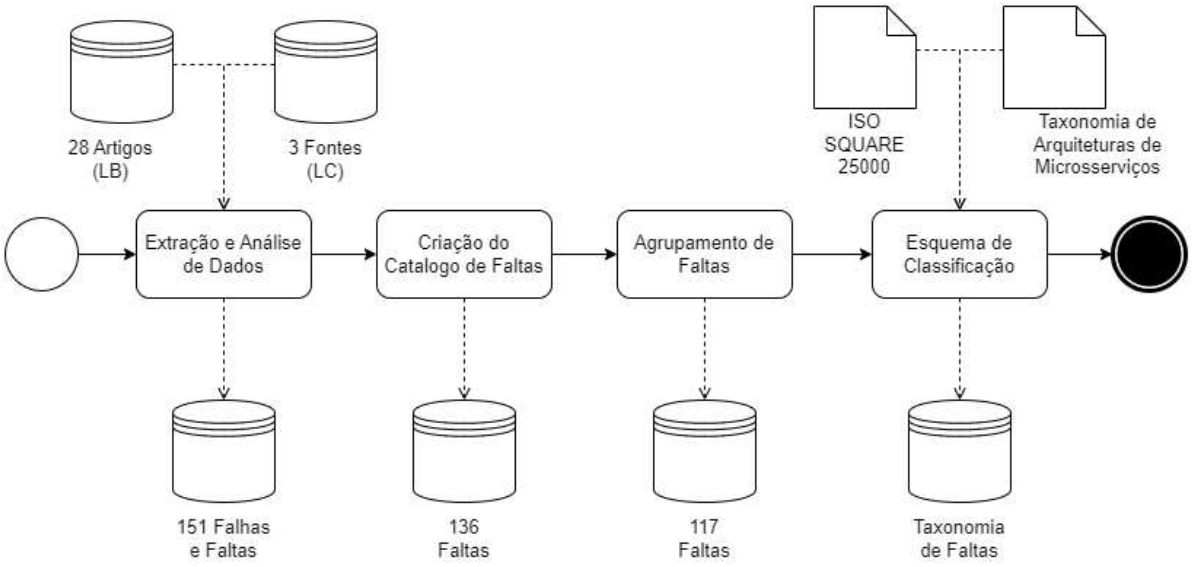

Figura 8 – Visão geral das atividades de definição da Taxonomia

Dessa forma, 26 das 136 faltas do catálogo foram agrupadas em sete faltas, resultando em 7 faltas, conforme apresentado na Tabela 7. Pode-se observar que duas faltas relativas à Memória denominadas na literatura *Memory Anomaly* (MA *et al.*, 2020) e *High Memory Usage*

(GEETHIKA *et al.*, 2019) estavam relacionadas, por esse motivo, estas faltas foram agrupadas em uma única falta, a qual foi denominada como *Memory Usage*. Ambas as faltas estão relacionadas com o consumo de memória pelas aplicações, mas foram propostas por autores diferentes e com nomenclaturas diferentes. Portanto, quando se completou o agrupamento destas 26 faltas do catálogo, o total de faltas diferentes era de 117 (136 - 26 + 7).

O passo seguinte foi a definição do "Esquema de Classificação"que consistiu em categorizar cada falta de acordo com Requisitos Não Funcionais e Característica de Microsserviço.

Tabela 7 – Agrupamento de Faltas do Catálogo de Faltas

Falta Original	**Falta Atual**	**Justificativa**
Memory Anomaly	Memory Usage Fault	As faltas se referem a Uso de Memória
High Memory Usage		
Insufficient Memory Allocation	Memory Allocation Fault	As faltas se referem a Alocação de Memória
Memory errors		
Memory cap		
CPU Hog	CPU Hog Fault	As faltas se referem a Uso de CPU
CPU Anomaly		
High CPU Usage		
CPU Pressure		
Insufficient CPU Allocation	CPU Allocation Fault	As faltas se referem a Alocação de CPU
CPU cap problem		
CPU error		
CPU Load		
Disk errors	Disk Usage Fault	As faltas se referem a Uso de Disco
Disk I/O		
High disk I/O Usage		
IO Load		
IO Interference		
Disk IO Jitter		
I/O Anomaly		
Packet loss/Latency errors	Network Fault	As faltas se referem a Perda de Pacotes
Packet loss problem		
Network Congestion		
Network Latency		
Infinite Loop Bug	Deadlock Fault	As faltas se referem a Deadlock
Deadlock		

Como supracitado, 103 das 136 faltas catalogadas não tinha sido classificada ante-

riormente nos estudos que foram extraídas. Assim, a classificação das 117 faltas foi feita com base em seis requisitos não funcionais da norma ISO 25000 (International Organization for Standardization (ISO), 2005) e 14 características de microsserviços apresentadas por Garriga (2018).

A seleção dos RNFs nesta pesquisa foi conduzida com base em dois critérios essenciais. Primeiramente, foram considerados trabalhos anteriores que estabeleceram uma relação explícita entre as faltas identificadas com RNFs correspondentes. Essa relação direta entre as faltas identificadas e os RNFs ofereceu uma base sólida para a seleção. Neste caso, foram revisitados os estudos para identificar se alguma dessas faltas apresentava algum tipo de correlação com os RNFs. Durante esta análise, foi possível identificar que três RNFs (*Segurança, Desempenho e Confiabilidade*) estavam relacionados com 66 das 117 faltas, como é apresentado na Seção 8. Por esse motivo, estes RNFs foram inicialmente adicionados no conjunto de categorias da taxonomia.

Além disso, as 117 faltas foram reanalisadas de acordo com a sua descrição e propósito. O objetivo desta análise era: (i) verificar se a correlação das 66 faltas com os RNFs é adequada ou se outros RNFs deveriam ser mais apropriados; e (ii) classificar as demais faltas (51 de 117 faltas) que não foram correlacionadas com os RNFs nos estudos, os quais foram encontradas.

Durante a primeira análise (i), 10 das 66 faltas foram reclassificadas de acordo com outros RNFs. Por exemplo, a falta *Degradação de Desempenho* (PF02) que causa a degradação da base de dados, que por usa vez ocorre quando há uma queda no desempenho geral do sistema de gerenciamento de banco de dados (SGBD) e, por consequência, nas consultas e operações realizadas nesse banco de dados, foi inicialmente relacionada com *Confiabilidade* na literatura. No entanto, esta falta é considerada mais relacionada com *Desempenho* e foi reclassificada.

Na segunda análise (ii), uma abordagem mais contextual e descritiva foi adotada. Nessa análise, o contexto e a descrição das faltas foram avaliados, classificando-as com base nas diretrizes fornecidas pela ISO/IEC 25010 (International Organization for Standardization (ISO), 2005). Essa abordagem permitiu a identificação de outros três RNFs relevantes mesmo em situações em que a conexão direta não era aparente, garantindo a abordagem abrangente de uma variedade de características de qualidade. Esses RNFs foram: Manutenibilidade, Compatibilidade e Funcionalidade.

Nas próximas subseções, os Requisitos Não Funcionais e às Características de

Microsserviços são apresentadas.

6.1.1 Requisitos Não Funcionais

A estrutura de RNFs serviu como base para a classificação das faltas. Como os RNFs podem ser divididos em subcategorias, na taxonomia também utilizou-se no esquema de classificação das subcaterísticas de cada RNF conforme ISO 25010 (International Organization for Standardization (ISO), 2005).

Os seis RNFs selecionados para o esquema da taxonomia e suas respectivas subcaraterísticas são detalhados a seguir.

6.1.1.1 Desempenho

Desempenho, no contexto da qualidade do software, está relacionado à eficiência e à quantidade de recursos utilizados pelo software para realizar suas tarefas. Esse RNF avalia a capacidade do software de realizar suas funções de maneira eficaz, mantendo um uso eficiente dos recursos, como CPU, memória e largura de banda, garantindo assim um uso otimizado dos recursos disponíveis. As subcaracterísticas de Desempenho utilizadas na taxonomia foram:

- *Comportamento Temporal:* o grau em que os tempos de resposta e de processamento e as taxas de *throughput* de uma aplicação satisfazem os requisitos;
- *Utilização de Recursos:* o grau em que as quantidades e os tipos de recursos utilizados por uma aplicação satisfazem os requisitos; e
- *Capacidade:* grau em que os limites máximos de um parâmetro de aplicação ou de sistema satisfazem os requisitos.

6.1.1.2 Segurança

Segurança é a capacidade de um produto ou sistema de proteger informações e dados contra vulnerabilidades, garantindo a confidencialidade, integridade e disponibilidade dessas informações, além de prevenir ameaças internas e externas, assegurando um ambiente seguro e confiável. As subcaracterísticas de Segurança utilizadas na taxonomia foram:

- Autenticidade: o grau em que é possível validar a identidade de um utilizador ou recurso solicitado.
- Integridade: o grau em que uma aplicação impede o acesso não autorizado ou a modificação

de programas ou dados.

- Confidencialidade: o grau em que uma aplicação garante que os dados são acessíveis apenas a quem está autorizado.
- Disponibilidade: o grau em que as ações ou acontecimentos podem ser provados de modo que não possam ser repudiados mais tarde.

6.1.1.3 Confiabilidade

Confiabilidade é a capacidade do software de manter um desempenho consistente e previsível ao longo do tempo e em condições específicas, garantindo que o sistema seja confiável e funcione conforme esperado, mesmo sob diversas circunstâncias. As subcaracterísticas de Confiabilidade utilizadas na taxonomia foram:

- *Recuperabilidade*: o grau em que, em caso de interrupção ou falta, uma aplicação pode recuperar os dados afetados e restaurar o estado desejado do sistema; e
- *Disponibilidade*: o grau em que uma aplicação está operacional e acessível quando necessário para utilização.

6.1.1.4 Manutenibilidade

Manutenibilidade é a capacidade do produto de software de ser facilmente modificado, permitindo a incorporação de melhorias, extensões de funcionalidade ou correções de defeitos, garantindo que o software permaneça resiliente e adaptável ao longo do tempo. A subcaracterística de Manutenibilidade utilizada na taxonomia foi:

- *Analisabilidade*: o grau de eficácia e eficiência com que é possível avaliar o impacto de uma alteração prevista numa aplicação a uma ou mais partes.

6.1.1.5 Compatibilidade

Compatibilidade é a medida da capacidade de um produto, sistema ou componente para interoperar e compartilhar informações efetivamente com outros produtos, sistemas ou componentes, enquanto executa suas funções essenciais no mesmo ambiente de hardware ou software. Isso garante a operação harmoniosa e a colaboração eficiente em sistemas interconectados. A subcaracterística de Compatibilidade utilizada na taxonomia foi:

- *Interoperabilidade*: o grau em que duas ou mais aplicações podem trocar e utilizar

informações.

6.1.1.6 Funcionalidade

Funcionalidade representa o conjunto de recursos e funcionalidades que atendem às necessidades explicitamente definidas e também às necessidades implícitas, cumprindo assim os requisitos essenciais para a finalidade à qual o produto se destina, assegurando uma experiência eficaz e abrangente para os usuários. A subcaracterística de Funcionalidade utilizada na taxonomia foi:

– *Completude Funcional*: o grau em que o conjunto de características cobre todas as tarefas e objetivos do usuário.

6.1.2 Características de Microsserviços

Para definir as características de microsserviços foi utilizada a taxonomia de características de microsserviços proposta por Martin (GARRIGA, 2018) que foi construída a partir de uma revisão sistemática. Nessa taxonomia, as características *Segurança, Manutenibilidade, e Confiabilidade* são mencionadas, por esta razão, todas as faltas classificadas no mesmo RNFs nesta taxonomia foram diretamente correlacionadas com estas características de microsserviço. Assim, nesta seção são apresentadas apenas 11 características:

– **MC01 - Suporte Arquitetônico** refere-se à descrição das obrigações e restrições que devem ser atendidas pelo sistema de microsserviços. Também envolve as estratégias para aplicar essas obrigações e restrições em um ambiente contextual dinâmico. Essa característica está relacionada à definição das diretrizes arquitetônicas e princípios que governam o design e a composição de microsserviços para atender a requisitos funcionais e não funcionais específicos. O suporte arquitetônico garante que os microsserviços em um sistema alinhado à visão arquitetônica global, promovendo consistência e coesão em suas interações e comportamentos no contexto dinâmico.

– **MC02 - Implementação** no contexto de microsserviços envolve uma consciência aguçada da complexidade do programa. Com milhares de microsserviços sendo executados de forma assíncrona em uma rede de computadores distribuída, o desenvolvimento desses serviços pode ser desafiador. Programas difíceis de entender tornam-se difíceis de escrever e modificar. Além disso, a implementação deve suportar a evolução contínua, um requisito frequentemente ditado pelo domínio da aplicação. Essa característica destaca a importância

de criar microsserviços que sejam não apenas robustos do ponto de vista funcional, mas também inteligíveis, modificáveis e adaptáveis às necessidades em constante evolução do sistema.

- **MC03 - Modelos de Interação** referem-se aos padrões ou métodos que governam como diferentes microsserviços se comunicam e trocam dados. Isso abrange vários aspectos dos fluxos de comunicação entre microsserviços em um sistema distribuído. Essa característica envolve a definição de como os microsserviços interagem, compartilham informações e coordenam suas ações. Modelos de interação comuns incluem solicitação e resposta síncronas, mensagens assíncronas, publicação e assinatura, e arquiteturas orientadas a eventos. Uma compreensão dos modelos de interação é essencial para a concepção de estratégias eficazes de comunicação entre microsserviços.
- **MC04 - Troca de Dados** refere-se aos protocolos e mecanismos usados para representar e transmitir dados entre microsserviços. Isso envolve a definição de como os dados são estruturados, codificados e transferidos entre as fronteiras de serviço. Microsserviços frequentemente precisam trocar dados para realizar suas funções ou compartilhar informações sobre o estado do sistema. Mecanismos eficazes de troca de dados são essenciais para garantir uma comunicação fluida e interoperabilidade entre microsserviços.
- **MC05 - Armazenamento de Dados** em arquiteturas de microsserviços difere dos sistemas legados com múltiplos serviços integrados. Em microsserviços, é imperativo identificar divisões lógicas em bancos de dados e aproveitar tecnologias apropriadas para desacoplar os dados de forma limpa. Isso garante que cada microsserviço tenha acesso aos dados necessários, mantendo a integridade, consistência e escalabilidade dos dados. O armazenamento de dados em microsserviços abrange o uso de várias soluções de armazenamento, como bancos de dados relacionais, bancos de dados NoSQL, mecanismos de cache e sistemas de dados distribuídos.
- **MC06 - Descoberta de Serviços** envolve a capacidade dos clientes de fazer solicitações a um conjunto extenso e em constante mudança de instâncias de serviço transitórias. É crucial para localizar e se conectar aos microsserviços disponíveis de maneira flexível e adaptativa. Mecanismos de descoberta de serviço possibilitam uma comunicação eficiente e interação entre microsserviços em um ambiente dinâmico e em constante evolução.
- **MC07 - Implantação** abrange as estratégias e considerações para hospedar e implantar microsserviços. Embora a nuvem tenha se tornado a plataforma padrão para microsserviços,

existem opções alternativas de implantação dentro e fora da nuvem. Essa característica aborda questões relacionadas ao local e a forma os serviços são hospedados, escalados e gerenciados para garantir sua disponibilidade e confiabilidade.

– **MC08 - Plataforma** refere-se à adaptabilidade da plataforma de microsserviços para acomodar personalizações com base em privacidade, segurança ou restrições de negócios. Essa característica permite que os microsserviços sejam adaptados a requisitos e restrições específicas dentro do ambiente operacional do sistema.
– **MC09 - Tempo de Execução** em microsserviços exige atenção adicional devido ao grande número de componentes independentes, arquivos de log e interações. Esses fatores podem afetar a latência e a confiabilidade. Alguns microsserviços podem precisar ser co-localizados e executados juntos para otimizar suas interações. O Tempo de Execução também envolve mecanismos de monitoramento proativo e adaptação automática para lidar com problemas de comportamento do sistema.
– **MC10 - Virtualização** abrange uma variedade de técnicas e tecnologias que fornecem diferentes níveis de abstração de plataforma, isolamento e compartilhamento de recursos. Essas técnicas permitem uma alocação eficiente de recursos para microsserviços, facilitando sua implantação e escalonamento. Métodos de virtualização incluem tecnologias de contêineres como Docker, máquinas virtuais (VMs) e infraestrutura baseada em nuvem, oferecendo recursos escaláveis e abstraídos para que os microsserviços operem eficazmente.
– **MC11 - Resiliência** envolve o tratamento eficaz de faltas nos níveis de serviço e infraestrutura de baixo nível. Ela exige a implementação de mecanismos para lidar com faltas, garantir a persistência de dados e permitir processos de recuperação. Técnicas de resiliência são essenciais para manter a disponibilidade do sistema, tolerância a faltas e robustez, especialmente em ambientes de microsserviços distribuídos e dinâmicos.

6.2 Taxonomia Preliminar de Faltas para Microsserviços

Antes da versão final da taxonomia, foi elaborada ma versão preliminar (ver Apêndice B) com 117 faltas classificadas com base nos seis RNFs e 14 características de microsserviços. Essa taxonomia foi estruturada e classificada com base no esquema de classificação definido na seção anterior, o qual se apoia na categorização das faltas de acordo com os seis Requisitos Não Funcionais (RNFs) estabelecidos e suas respectivas subcaracterísticas. Além disso, cada falta foi

cuidadosamente relacionada com uma característica específica de microsserviços.

Observando a taxonomia preliminar, é possível constatar uma distribuição heterogênea das faltas entre os seis RNFs identificados. Notavelmente, o RNF de Desempenho emergiu como a categoria com o maior número de faltas, totalizando 39 ocorrências. Isso destaca a importância crítica atribuída ao desempenho nas aplicações de microsserviços, indicando a necessidade de medidas substanciais para otimizar essa característica. Por outro lado, o RNF de *Funcionalidade* registrou o menor número de faltas, com apenas duas. No entanto, isso não deve ser interpretado como negligência, mas sim como um reflexo das especificidades das aplicações de microsserviços. O RNF de *Segurança* também se destacou, com 35 faltas registradas, sublinhando a preocupação inerente à segurança no contexto dessas aplicações. O RNF de *Confiabilidade* e o RNF de *Manutenibilidade* também desempenharam um papel significativo, com 14 e 19 faltas, respectivamente. Enquanto isso, o RNF de *Compatibilidade* apresentou 8 faltas, indicando a importância de garantir que as aplicações de microsserviços sejam compatíveis com diversas configurações e ambientes.

Essa análise ressalta a necessidade de priorizar a melhoria da *Desempenho* , *Segurança*, *Confiabilidade* e *Manutenibilidade* nas aplicações de microsserviços, enquanto também reconhece a importância de manter uma funcionalidade eficaz e assegurar a compatibilidade em um ambiente diversificado. No contexto da taxonomia preliminar, a estruturação dessas faltas por categoria facilita a identificação e resolução de questões específicas em aplicações de microsserviços, contribuindo para aprimorar sua qualidade e desempenho de acordo com os requisitos não funcionais estabelecidos.

6.3 Taxonomia de Faltas para Microsserviços

Nesta seção, é apresentada a versão final da Taxonomia de Faltas para Aplicações Baseadas em Microsserviços. O desenvolvimento dessa taxonomia foi um processo iterativo que envolveu uma avaliação por especialistas na área. Inicialmente, uma taxonomia preliminar foi criada como parte deste estudo, destinada a categorizar e organizar as faltas comuns encontradas em aplicações baseadas em microsserviços. No entanto, reconhecendo a importância da validação e refinamento, uma avaliação abrangente foi conduzida por meio de um grupo focal composto com a participação de especialistas experientes em engenharia de software e microsserviços. As contribuições fornecidas pelos especialistas permitiram uma evolução e aprimoramento da taxonomia. Posteriormente, a taxonomia na sua versão final também foi avaliada utilizando o

Método Delphi. Ambas as avaliações são apresentadas no Capítulo 7.

A Taxonomia de Faltas para Aplicações Baseadas em Microsserviços está acessível e disponível de forma aberta em um Wiki Digital hospedada no GitHub[1]. Esse repositório serve como um recurso central para a comunidade de engenharia de software e profissionais que desejam explorar e utilizar a taxonomia para a classificação e compreensão de faltas em ambientes de microsserviços. A versão final da Taxonomia de Faltas possui 106 faltas classificadas em seis RNFs e 14 características de microsserviços. Com relação a distribuição das faltas na versão final da taxonomia, a quantidade de faltas por RNF é apresentada na Tabela 8. Nela, pode-se observar que as categorias com maior quantidade de faltas são Desempenho e Segurança e as com menor quantidade são Compatibilidade e Funcionalidade. Além disso, para a versão final da taxonomia foi introduzido para cada falta, um exemplo de como a falta ocorre, i.e. falha.

Tabela 8 – Distribuição de Faltas na Taxonomia Final

RNF	**Número de Faltas**
Desempenho	38
Confiabilidade	12
Segurança	35
Manutenibilidade	14
Compatibilidade	5
Funcionalidade	2

Nas seções a seguir, são apresentadas as faltas e suas respectivas categorias e relacionamentos com as características de microsserviços, vale ressaltar que para cada RNF, são apresentadas as faltas separadas de acordo com a subcaracterística que ela está relacionada.

6.4 Faltas de Desempenho

Com relação ao RNF de Desempenho , foram definidas 38 faltas que foram classificadas em: Comportamento Temporal (22 faltas), Utilização de Recursos (06 Faltas) e Capacidade (08 Faltas). Além disso, as 35 faltas foram também relacionadas com 7 características de microsserviços, são elas: Tempo de Execução, Armazenamento de Dados, Resiliência, Implementação, Descoberta de Serviços, Modelos de Interação e Virtualização.

As Tabelas 9, 10 e 11 apresentam as faltas de Desempenho classificadas em Comportamento Temporal, Utilização de Recursos e Capacidade.

[1] https://github.com/gutenbergf/Fault-Taxonomy-for-Microservice-Based-Applications

Tabela 9 – Comportamento Temporal (22 Faltas)

ID	Falta	Descrição da Falta	Exemplo de Falta	Característica
PF01	Infinit Wait	Esta falta ocorre quando uma sequência de instruções continua indefinidamente sem intervenção externa.	Um valor de contador não é atualizado corretamente, causando a paralisação dos threads de processamento de solicitações.	Tempo de Execução
PF02	Performance Degradation	A performance do sistema deteriora ao longo do tempo devido à fragmentação da heap, exigindo uma compactação completa.	Configurar as conexões do banco de dados para 10, 15 e 20 leva à degradação do desempenho do banco de dados.	Armazenamento de Dados
PF03	Process Crash Fault	Um processo para de funcionar corretamente e encerra inesperadamente.	Encerrar um processo leva a um desligamento inesperado.	Resiliência
PF04	Deadlock Fault	Duas ou mais operações desejam acessar recursos bloqueados entre si.	Interceptação de chamadas Lib relacionadas a processos e bloqueios, levando a uma situação de impasse.	Implementação
PF05	Flag setting bug	Quando recursos experimentais estão habilitados, podem ocorrer problemas.	Excluir uma porta de escuta e reiniciar o servidor causa um erro de chamada bloqueante devido a uma tag não limpa.	Resiliência

PF06	Infinit Read Bug	O computador repete continuamente as mesmas ações de leitura.	Os arquivos do sistema não verificam o estouro, causando tentativas repetidas de leitura, eventualmente resultando em timeout.	Tempo de Execução
PF07	Thread Shutdown Bug	O desligamento inesperado de thread pode ocorrer durante o registro.	O mecanismo de desligamento de thread não funciona, levando a uma variável atômica esperando indefinidamente.	Resiliência
PF08	Point Wise	Detecção de pontos de dados desviados, frequentemente usada para detecção de fraudes.	Identificação de transações anormais com cartão de crédito comparando padrões de gastos.	Tempo de Execução
PF09	Collective Anomalie	Detectado por padrões recorrentes, como atrasos na cadeia de suprimentos.	Atraso recorrente na entrega de produtos em rotas de envio exige investigação e análise.	Implementação
PF10	Contextual Anomalie	Detectado ao considerar o contexto anterior, como pedidos sazonais incomuns.	Aumento repentino de pedidos de roupas de inverno durante o verão, considerando padrões de compra do cliente.	Implementação
PF11	Disponibilidade Anomaly	Desvio nas métricas de disponibilidade de serviço, exigindo investigação.	Queda repentina na porcentagem de disponibilidade do aplicativo web devido a tempos de resposta e erros.	Resiliência

PF12	Service Interruption Fault	Interrupção no serviço devido a desconexão.	Perda de conexão com a internet durante uma chamada de vídeo online.	Resiliência
PF13	Data Loss Fault	Falta na instância do microsserviço resultando em perda de dados.	Falha no serviço do banco de dados leva à exclusão acidental de dados.	Resiliência
PF14	Network Transmission Delay	Tempo necessário para transmitir um pacote é excessivo.	Falta de atraso HTTP pelo istio devido a problemas de rede.	Tempo de Execução
PF15	Network Transmission Abortion	Tempo necessário para transmitir um pacote é abortado.	Falta de aborto HTTP pelo istio causando problemas de comunicação entre microsserviços.	Tempo de Execução
PF16	Power Anomaly	Ajuste do consumo de energia com base na taxa de transferência e latência.	Dispositivos de rede ajustam o consumo de energia para um desempenho ótimo.	Tempo de Execução
PF17	Throughput Anomaly	Latência média anormal afetando o desempenho do throughput.	O tempo de resposta de um serviço aumenta de 50ms para 500ms, afetando o throughput.	Tempo de Execução
PF18	Latency Anomaly	Anomalia com base na latência média da chamada de serviço.	A latência do serviço aumenta de 50ms para 500ms devido à congestão de rede.	Tempo de Execução
PF19	Long Response Time Fault	Atrasos estendidos nos serviços de backend devido a congestão de rede ou aumento da carga.	O tempo de resposta do API-Gateway passa de 100ms para 5 segundos devido à sobrecarga do backend.	Descoberta de Serviços

PF20	Remote Service Change Anomaly	Consultas adhoc competindo por recursos devido ao uso interno de construtores de consultas.	Múltiplos funcionários executando consultas adhoc intensivas em recursos impactando o desempenho do aplicativo.	Armazenamento de Dados
PF21	Packet Loss Fault	Erros de rede ou infraestrutura levando à perda de dados.	Comunicação entre microsserviços experimenta perda de dados devido a problemas de rede.	Resiliência
PF22	Deadlock Database Bug	Duas conexões de banco de dados tentando travar tabelas, causando um impasse.	Conexões simultâneas de banco de dados tentando travar tabelas, levando a um impasse.	Armazenamento de Dados

Tabela 10 – Utilização de Recursos (06 Faltas)

ID	Falta	Descrição da Falta	Exemplo de Falta	Característica
PF23	Network Fault	Saturação da rede devido a códigos maliciosos ou a um grande número de solicitações, levando à negação de serviço.	Ataque DDoS inunda a rede, tornando um microsserviço indisponível.	Resiliência
PF24	CPU Allocation	Alocação insuficiente de CPU devido a limite baixo de CPU VM.	O desempenho do microsserviço cai devido a um limite baixo de CPU.	Tempo de Execução
PF25	Memory Allocation	Alocação insuficiente de memória devido a um limite baixo de memória VM.	Falhas frequentes devido à alocação baixa de memória na VM.	Tempo de Execução
PF26	Memory Usage	Microsserviço consome memória excessiva, afetando outros serviços.	O API-Gateway fica instável devido a um microsserviço alta demanda por memória.	Resiliência
PF27	Computing Resources Exhaustion	Um processo intensivo em recursos consome recursos computacionais.	Um processo em segundo plano usa a maior parte da CPU, impactando outros serviços.	Resiliência
PF28	Disk Usage Fault	Acesso intenso ao disco causando repetições ou falhas.	Microsserviço escreve em um disco muito acessado, causando repetições e atrasos.	Resiliência
PF29	Resource Usage	Desaceleração do microsserviço devido a recursos insuficientes na VM.	O desempenho do microsserviço é afetado por recursos inadequados de CPU e memória.	Resiliência
PF30	CPU Hog	Falta desencadeiada pelo alto uso de CPU em um microsserviço.	Loop do microsserviço consome 75% e 90% do tempo da CPU, afetando o desempenho.	Resiliência
PF31	Memory Leak	Objetos não liberados adequadamente, levando ao aumento do uso de memória.	O uso de memória aumenta constantemente devido a objetos não liberados, causando problemas de desempenho.	Resiliência

Tabela 11 – Capacidade (07 Faltas)

ID	Falta	Descrição da Falta	Exemplo de Falta	Característica
PF32	Message Size Fault	Desempenho cai ou fica instável à medida que o tamanho da mensagem aumenta.	O desempenho do API-Gateway diminui ao processar grandes mensagens.	Tempo de Execução
PF33	Commit Leaks	Gerenciamento incorreto dos tamanhos de commit.	Microsserviço confirma transações em lotes grandes, causando consumo de recursos e inconsistências.	Tempo de Execução
PF34	Increased Number of Users	API-Gateway instável devido ao alto acesso de usuários concorrentes.	O API-Gateway fica instável durante as horas de pico com um grande número de usuários.	Tempo de Execução
PF35	Monolithic Fault	Faltas em invocações assíncronas executadas ou retornadas inesperadamente.	Microsserviços A e B se comunicam de forma assíncrona, causando problemas de resposta.	Modelos de Interação
PF36	Multi-Instance Fault	Falta de coordenação entre diferentes instâncias de um microsserviço.	Múltiplas instâncias de microsserviço mantêm estados locais inconsistentes, causando comportamento inesperado.	Modelos de Interação
PF37	Configuration Fault	Configurações incorretas ou inconsistentes de microsserviço ou ambiente.	Falta de comunicação entre microsserviços devido a configurações incompatíveis.	Virtualização
PF38	Asynchronous Interaction Fault	Coordenação inadequada de sequências de invocações assíncronas.	Microsserviço recebe respostas assíncronas fora de ordem, levando a resultados incorretos.	Modelos de Interação

Com base nas tabelas acima, pode-se perceber que a característica de microsserviço Resiliência foi associada com uma maior quantidade de faltas de performance (15 de 38) e Virtualização foi a que menos teve (1 falta). Além disso, comparada com a Taxonomia Preliminar, a taxonomia de faltas final possui uma falta a menos, isso porque a Falta PF12 *Delay Fault* da preliminar foi agrupada na falta já existente *Network Transmission Delay* da taxonomia final. Por outro lado, as faltas PF13 (*Disconect Fault*) e PF14 (*Delete Fault*) da preliminar foram renomeadas em *Data Loss Fault* e *Service Interruption Fault*, PF12 e PF13 da taxonomia final.

6.5 Faltas de Confiabilidade

No que diz respeito ao RNF de Confiabilidade, 12 faltas foram cuidadosamente subcategorizadas em duas categorias distintas: Recuperabilidade e Disponibilidade. Além disso, essas 12 faltas foram relacionadas com apenas uma característica de microsserviço: Confiabilidade. Informações detalhadas sobre essas faltas são apresentadas nas Tabelas 12 e 13, onde as faltas foram organizadas em suas respectivas subcategorias.

Tabela 12 – Recuperabilidade (08 Faltas)

ID	Falta	Descrição da Falta	Exemplo de Falta
RF01	Application Bug	Um bug de aplicativo que estressa a instância.	Um exemplo dessa falta é quando um bug de software no código de um microsserviço faz com que ele consuma recursos excessivos de CPU e memória, resultando em desempenho degradado.
RF02	Random System Termination Fault	Encerramento aleatório do sistema para avaliação da resiliência e recuperabilidade dos sistemas em produção.	Um exemplo poderia ser um framework de teste que termina aleatoriamente um microsserviço para avaliar sua resiliência e mecanismos de recuperação.
RF03	Crash-Recovery Bug	Operações conflitantes são do nó de falha NCrash e do nó de recuperação NRecovery.	Um exemplo pode ser quando, durante um evento de failover, um nó de falha e um nó de recuperação acessam simultaneamente recursos compartilhados, levando à corrupção de dados.

RF04	Message Corruption Fault	Essa falta envolve a adulteração intencional de mensagens trocadas entre componentes ou serviços dentro de uma aplicação baseada em microsserviços. Isso inclui alterar o conteúdo das mensagens, seja invertendo bits individuais, introduzindo ruído aleatório ou modificando os payloads das mensagens, para simular erros de comunicação ou corrupção inesperada de dados.	Um exemplo pode envolver uma falha injetada que intercepta uma mensagem de rede e inverte um bit em seu conteúdo, causando corrupção de dados durante a transmissão.
RF05	Permanent Fault	Uma falta permanente, por outro lado, tem um período ativo infinito ou seu período ativo excede uma quantidade pré-definida de tempo, além do qual a reparação não pode mais ser aplicada.	Um exemplo pode ser uma falha de hardware em um módulo de memória de um servidor que causa travamentos intermitentes que não podem ser reparados.
RF06	Transient Fault	Uma falta é transitória se seu período ativo for finito e for seguido por um período dormente infinito, assim a reparação é bem-sucedida dentro de uma quantidade máxima de tempo.	Um exemplo é quando um microsserviço experimenta slowdowns ocasionais devido a problemas transitórios de conectividade de rede.
RF07	Intermittent Fault	Uma falta intermitente oscila entre uma operação com falha e uma operação sem falha, ela aparece ocasionalmente e parece não determinística, pois suas condições de ocorrência não são facilmente reproduzíveis.	Um exemplo pode envolver uma falha intermitente em um microsserviço que o faz funcionar bem para algumas solicitações e falhar para outras de forma imprevisível.

RF08	Invalid Action Fault	A ação ou operação solicitada é inválida.	Um exemplo pode ser uma chamada de API para excluir um recurso que falha porque o ID do recurso fornecido está incorreto ou não existe.

Tabela 13 – Disponibilidade (04 Faltas)

ID	Falta	Descrição da Falta	Exemplo de Falta
RF09	Request Expired	A solicitação chegou ao serviço mais de 15 minutos após o carimbo de data na solicitação ou mais de 15 minutos após a data de expiração da solicitação (como em URLs pré-assinadas), ou o carimbo de data na solicitação está mais de 15 minutos no futuro.	Um exemplo pode envolver um cliente enviando uma solicitação a um serviço, mas o carimbo de data na solicitação está mais de 15 minutos atrás do tempo atual do servidor, resultando na rejeição da solicitação.
RF10	Service Unavailable	A solicitação falhou devido a uma falha temporária do servidor.	Um exemplo é quando um servidor enfrenta uma carga elevada, resultando em indisponibilidade temporária e falhas nas solicitações.
RF11	Throttling Exception	A solicitação foi negada devido à limitação de requisições.	Um exemplo pode envolver um serviço que impõe limites de taxa e nega solicitações que excedem a taxa de requisições permitida.
RF12	Validation Error	A entrada não atende às restrições especificadas por um serviço AWS.	Um exemplo pode ser um cliente enviando uma solicitação de API com uma entrada que viola as regras de validação definidas pelo serviço AWS, resultando em uma resposta de erro de validação.

Com relação as faltas de Confiabilidade, na taxonomia preliminar tinham 14 faltas e na final possui 12 faltas, duas faltas a menos. Isso ocorreu porque a falta RF10 (*Unexpected Fault*) da taxonomia preliminar foi agrupada na PF03 (*Process Crash Fault*) da taxonomia final e a falta RF02 (*S.O Kernel Bug*) da preliminar foi removida da taxonomia final por ser muito genérica.

6.6 Faltas de Segurança

Nesta seção, as 35 faltas de Segurança foram subdivididas em quatro subcategorias distintas: Autenticidade , Integridade, Confidencialidade e Disponibilidade. Além disso, essas 35 faltas foram correlacionadas com a característica de microsserviço de Segurança. Detalhes completos sobre essas faltas são apresentados nas Tabelas 14, 15, 16 e 17.

Tabela 14 – Autenticidade (10 Faltas)

ID	Falta	Descrição da Falta	Exemplo de Falta
SF01	Missing authentication	Ausência de um mecanismo de autenticação no sistema	Por exemplo, uma aplicação web que permite o acesso a dados sensíveis sem exigir que os usuários façam login.
SF02	Authentication bypass	Existência de um ponto de entrada com mecanismo de autenticação que pode ser contornado	Por exemplo, uma aplicação web em que a adição de um parâmetro específico à URL concede acesso sem autenticação.
SF03	Relying on single Factor Authentication	Os mecanismos de autenticação dependem do uso de senhas	Por exemplo, um sistema que requer apenas um nome de usuário e senha para autenticação, sem fatores adicionais como autenticação de dois fatores (2FA).
SF04	Downgrade authentication	Possibilidade de autenticar com um mecanismo de autenticação mais fraco (ou obsoleto)	Por exemplo, um sistema que ainda permite o uso de métodos de autenticação desatualizados e inseguros, mesmo quando opções mais seguras estão disponíveis.

SF05	Message Corruption Fault	Corrupção de mensagem refere-se a falhas em dados de computador que ocorrem durante a escrita, leitura, armazenamento, transmissão ou processamento, introduzindo alterações não intencionais na mensagem original	Por exemplo, uma comunicação de rede onde os pacotes de dados são alterados em trânsito, resultando em mudanças não intencionais no conteúdo da mensagem.
SF06	No re-authentication	Ausência de reautenticação durante operações críticas	Por exemplo, um sistema que não exige reautenticação quando um usuário tenta realizar ações sensíveis, mesmo após um login inicial.
SF07	Missing authorization	Ausência de um mecanismo de autorização no sistema	Por exemplo, uma aplicação web que não verifica se um usuário tem as permissões necessárias para acessar determinados recursos ou realizar ações específicas.
SF08	No context when authorizing	Ausência de verificações condicionais para controle de acesso	Por exemplo, um sistema que concede acesso a recursos sem considerar informações ou condições contextuais adicionais.
SF09	Not revoking authorization	Ausência de um processo para revogar o acesso do usuário	Por exemplo, um sistema que não remove os privilégios de acesso de um usuário, mesmo após a alteração da função ou permissões do usuário.
SF10	Missing Authentication Token	A solicitação não contem um ID de chave de acesso válida (registrada) ou certificado.	Por exemplo, uma solicitação de API da AWS que não possui o token de autenticação necessário, tornando impossível verificar a identidade do solicitante.

Tabela 15 – Integridade (6 Faltas)

ID	Falta	Descrição da Falta	Exemplo de Falta
SF11	Insecure data storage	Armazenamento de dados sensíveis de forma clara ou mecanismos de controle de acesso fracos.	Por exemplo, um banco de dados que armazena senhas de usuários em texto simples, sem criptografia ou controles de acesso adequados.
SF12	Insecure data exposure	Exposição de dados sensíveis durante o transporte em texto simples	Por exemplo, um aplicativo da web que envia dados sensíveis do usuário por uma conexão HTTP não criptografada, tornando-os vulneráveis a interceptação.
SF13	Not validating input/data	Ausência de verificações de validação ao receber dados de entidades externas.	Por exemplo, um aplicativo que não valida dados de entrada de usuários, permitindo a injeção de código ou dados maliciosos.
SF14	Data Type Probing	Essa anomalia ocorre quando o tipo de dados do valor enviado ou recebido é alterado	Por exemplo, um sistema que aceita dados em um formato e os converte para outro sem validação adequada, potencialmente resultando em vulnerabilidades relacionadas ao tipo de dados.
SF15	Data Flushing	Truncar uma tabela causa lentidão nas inserções devido a um bug no mecanismo de armazenamento para grandes conjuntos de dados. O banco de dados não marca os dados truncados como excluídos e constantemente aloca novos blocos	Por exemplo, um sistema de banco de dados que encontra um bug ao truncar grandes tabelas, causando lentidão nas inserções de tabelas e má gestão da alocação de armazenamento.
SF16	Drop Table	A exclusão dos índices de algumas tabelas desacelera as operações de consulta	Por exemplo, um administrador de banco de dados que erroneamente exclui índices em tabelas críticas, resultando em desempenho mais lento nas consultas.

Tabela 16 – Confidencialidade (17 Faltas)

ID	Falta	Descrição da Falta	Exemplo de Falta
SF17	Insufficient permissions Fault	Falhas de permissões insuficientes são causadas por consumidores de API que tentam usar um endpoint ou fazer uso de um recurso, enquanto não têm permissão para fazê-lo.	Por exemplo, um usuário sem privilégios administrativos tentando acessar uma seção restrita a administradores em um site.
SF18	Insufficient crypto key management	As chaves não são gerenciadas com segurança ao longo de seu ciclo de vida	Por exemplo, chaves criptográficas sendo armazenadas em um local inseguro sem controles de acesso adequados, tornando-as vulneráveis a roubo.
SF19	Insufficient session management	As sessões não são gerenciadas com segurança ao longo de seu ciclo de vida	Por exemplo, um aplicativo da web que não invalida ou expira corretamente as sessões do usuário, permitindo que um atacante assuma a sessão de um usuário.
SF20	Insufficient credentials management	As credenciais não são gerenciadas com segurança ao longo de seu ciclo de vida	Por exemplo, armazenar senhas de usuários em texto simples ou em formatos fracos, tornando-as suscetíveis a roubo ou quebra.
SF21	Use of custom/weak encryption	Uso de chaves pequenas, algoritmos de criptografia obsoletos	Por exemplo, usar algoritmos criptográficos desatualizados ou chaves de criptografia fracas que podem ser facilmente quebradas por atacantes.

SF22	Missing access control	Ausência de controle de acesso no sistema	Por exemplo, um sistema que não restringe o acesso a dados ou recursos sensíveis, permitindo que usuários não autorizados os visualizem ou modifiquem.
SF23	Insufficient auditing	O acesso a recursos ou operações críticas não é registrado.	Por exemplo, falha ao registrar tentativas de login, acesso a arquivos sensíveis ou ações administrativas, dificultando a detecção e investigação de incidentes de segurança.
SF24	Malicious Control	Essa anomalia ocorre quando alguém tenta controlar o tráfego de rede de forma maliciosa	Por exemplo, um atacante tentando manipular o tráfego de rede para interceptar ou redirecionar dados para fins maliciosos.
SF25	Malicious Operation	Essa anomalia ocorre quando atividades fictícias ou operações ocultas distorcem as atividades originais.	Por exemplo, um atacante criando operações enganosas ou maliciosas para desviar a atenção de atividades maliciosas reais.
SF26	Scan	Essa anomalia ocorre quando alguém tenta replicar o cliente ou servidor para obter acesso a credenciais e dados do usuário	Por exemplo, um atacante escaneando uma rede ou sistema em busca de vulnerabilidades e fraquezas para explorar.
SF27	Spying	Essa anomalia ocorre quando ocorre escuta ilegal, ou seja, alguém monitorando nossa operação regular e descobrindo informações importantes	Por exemplo, uma parte não autorizada interceptando e monitorando comunicações para reunir informações sensíveis.

SF28	Wrong Setup	Essa anomalia ocorre quando alguém tenta manipular nossa configuração ou configuração errada feita pelo próprio usuário	Por exemplo, um atacante alterando configurações ou ajustes do sistema para interromper operações ou obter acesso não autorizado.
SF29	Illegal Access Anomaly	Usar a senha ou usuário errado para acessar ou autenticar-se no banco de dados causa anomalias de acesso ilegal	Por exemplo, tentar acessar um sistema seguro com uma conta de usuário incorreta ou não autorizada.
SF30	Not Authorized	Essa falta indica que o usuário não possui as permissões necessárias para realizar uma ação específica ou acessar um recurso específico. Significa uma restrição de segurança que impede a operação solicitada devido à autorização insuficiente.	Por exemplo, um usuário tentando realizar uma ação ou acessar um recurso para o qual não possui as permissões necessárias.
SF31	Access Denied Exception	Essa falta ocorre quando um usuário tenta uma ação para a qual tem direitos de acesso insuficientes. Indica uma restrição de segurança que nega a operação solicitada devido à falta de autorização adequada.	Por exemplo, um sistema retornando um erro de "Acesso Negado"quando um usuário tenta realizar uma ação para a qual não está autorizado.
SF32	Invalid Client TokenId	O certificado ou ID da chave de acesso fornecido não existe nos registros.	Por exemplo, um cliente fornecendo um token ou chave inválido ou inexistente para autenticação.

SF33	Subscription Authorization Required	O ID da chave de acesso precisa de uma assinatura para o serviço.	Por exemplo, um usuário tentando acessar um serviço premium sem ter a assinatura ou autorização necessária.

Tabela 17 – Disponibilidade (02 Faltas)

ID	Falta	Descrição da Falta	Exemplo de Falta
SF34	Uncontrolled resource consumption	Consumo não controlado de recursos de componentes internos	Por exemplo, um componente de software que consome memória ou recursos da CPU excessivamente, levando à instabilidade ou falha do sistema.
SF35	Unmonitored execution	Consumo não controlado de recursos devido a interações com entidades externas	Por exemplo, um aplicativo da web que sofre um ataque de negação de serviço distribuído (DDoS), tornando-se irresponsivo devido a uma inundação de solicitações recebidas.

6.7 Faltas de Manutenibilidade

Em relação ao RNF de Manutenibilidade, 17 faltas foram subdividas em uma subcategoria: Analisabilidade. Além disso, essas 17 faltas foram relacionadas com apenas uma característica dos microsserviços: Manutenibilidade. Os detalhes completos sobre as faltas de Manutenibilidade são apresentadas na Tabela 18, onde as faltas estão organizadas de acordo com a sua respectiva subcategoria.

Tabela 18 – Analisabilidade (14 Faltas)

ID	Falta	Descrição da Falta	Exemplo de Falta
MF01	Invalid User Input Fault	Falta de Entrada de Usuário Inválida refere-se a solicitações que falham porque um usuário final forneceu uma entrada que não pode ser usada para concluir a ação pretendida	Inserir caracteres não numéricos em um campo que espera um número.

ID	Falta	Descrição da Falta	Exemplo de Falta
MF02	Missing User Input Fault	Falta de Entrada de Usuário Ausente está fortemente relacionada à entrada de usuário inválida. Neste caso, no entanto, o usuário final deixa de preencher informações necessárias, o que causa o fracasso da solicitação subsequente.	Não fornecer um nome de usuário e senha ao tentar fazer login.
MF03	Expired request data Fault	Falta de dados de solicitação expirados ocorrem quando a solicitação não é tratada a tempo	Tentar confirmar uma reserva após a data de check-in ter passado.
MF04	Invalid request data Fault	Falta de dados de solicitação inválidos são causadas por entradas que não podem ser manipuladas pela API	Inserir caracteres alfabéticos em um campo que espera uma data.
MF05	Missing server data Fault	Falta de dados ausentes do servidor ocorrem quando o consumidor da API solicita dados que costumavam existir, mas não existem mais, pois foram atualizados, removidos ou desativados no passado.	Solicitar informações sobre um produto que foi descontinuado.
MF06	Metadata Fault	Esta falta é explorada para considerar o cenário de uma atualização que, além de modificar o comportamento do serviço, introduz uma falha nos metadados	Alterar a descrição de um produto, mas não atualizar seus metadados de categoria.
MF07	Data Handling Routines Fault	Esta falta é explorada para considerar o cenário de uma atualização que, além de modificar o comportamento do serviço, introduz uma falha nas Rotinas de Manipulação de Dados	Alterar a estrutura do banco de dados sem atualizar as rotinas de manipulação de dados correspondentes.

ID	Falta	Descrição da Falta	Exemplo de Falta
MF08	Incomplete Signature	A assinatura da solicitação não está em conformidade com os padrões da AWS.	Enviar uma solicitação AWS com uma assinatura de autenticação ausente ou incompleta.
MF09	Invalid Parameter Combination	Parâmetros que não devem ser usados juntos foram usados juntos.	Tentar filtrar produtos por categoria e marca quando a API não suporta essa combinação.
MF10	Invalid Parameter Value	Um valor inválido ou fora de alcance foi fornecido para o parâmetro de entrada.	Tentar definir um preço negativo para um produto.
MF11	Invalid Query Parameter	A string de consulta está malformada ou não segue os padrões.	Fornecer uma data mal formatada em um parâmetro de consulta.
MF12	Malformed Query String	A string de consulta contém um erro de sintaxe.	Usar caracteres especiais que não estão devidamente codificados na string de consulta.
MF13	Missing Action	A solicitação está sem uma ação ou um parâmetro obrigatório.	Tentar enviar uma solicitação sem especificar a ação desejada.
MF14	Missing Parameter	Um parâmetro obrigatório para a ação especificada não é fornecido.	Tentar criar uma nova conta de usuário sem fornecer um endereço de e-mail.

Na taxonomia final, o RNF de Manutenibilidade possui 14 faltas, cinco a menos que na taxonomia preliminar. Isso ocorreu, em sua maioria, porque existiam algumas faltas redundantes na taxonomia preliminar, como as faltas MF02 e MF06, em que ambas eram *Missing User Inpult Fault* e as faltas MF04 e MF08, as quais representavam a mesma falta, *Invalid request data Fault*.

6.8 Faltas de Compatibilidade

Relativamente ao RNF de Compatibilidade, cinco faltas foram classificadas em uma subcategoria: Interoperabilidade. Além disso, essas cinco faltas foram correlacionadas com duas

características essenciais de microsserviços, sendo elas: Modelos de Interação (quatro faltas) e Implementação (uma falta). As informações detalhadas sobre as faltas de Compatibilidade são apresentadas na Tabela 19.

Tabela 19 – Interoperabilidade (5 Faltas)

ID	Falta	Descrição da Falta	Exemplo de Falta	Característica
CF01	Double processing Fault	Falta de processamento duplicado são causadas por consumidores de API que enviam uma solicitação mais de uma vez.	Um consumidor de API envia acidentalmente a mesma solicitação de pedido de compra duas vezes, resultando em pedidos duplicados.	Modelo de Interação
CF02	API Configuration Fault	Falta de configuração da API são causadas pela configuração incorreta da conta do consumidor de API.	O consumidor de API configura suas configurações de autenticação, resultando em erros de acesso não autorizado.	Implementação
CF03	Missing Request Data Fault	Falta de dados de solicitação ausentes são semelhantes a Falta de dados de solicitação inválidos. No entanto, neste caso, o consumidor de API negligencia enviar informações necessárias para a ação pretendida.	Um consumidor de API esquece de incluir o endereço do destinatário ao fazer uma solicitação de pedido, causando erros de processamento.	Modelo de Interação
CF04	Conversation Error	As faltas de conversação surgem de erros de conversação: desvios do comportamento normativo especificado por um determinado script de conversação.	Durante uma interação com um chatbot, a entrada do usuário se desvia do fluxo de conversa esperado, levando à confusão.	Modelo de Interação
CF05	Interaction Fault	Essas faltas são frequentemente causadas pela falta ou coordenação incorreta de interações entre microserviços.	Um microserviço falha em se comunicar com outro microserviço, resultando no processamento incompleto de pedidos.	Modelo de Interação

Na taxonomia final, o RNF compatibilidade possui apenas cinco faltas, três a menos que na taxonomia preliminar, que possuía oito faltas. Isso ocorreu porque algumas faltas estavam redundantes, por exemplo, CF05 e MF05 da taxonomia preliminar eram a mesma falta, *Missing*

request data. Esta por sua vez foi reanalisada e classificada apenas no RNF de Manutenibilidade.

6.9 Faltas de Funcionalidade

Com relação ao RNF de Funcionalidade, 02 faltas foram classificadas em uma subcategoria: Completude Funcional. Além disso, essas duas faltas foram correlacionadas com apenas uma característica de microsserviço: Implementação. Detalhes completos sobre essas faltas são apresentados na Tabela 20.

Tabela 20 – Completude Funcional (2 Faltas)

ID	Falta	Descrição da Falta	Exemplo de Falta	Característica
FF01	Functional Fault	Resultam no mau funcionamento dos serviços do sistema, gerando erros ou produzindo resultados incorretos	Um site de comércio eletrônico falha ao aplicar um código de desconto durante o *checkout*, fazendo com que os clientes sejam cobrados pelo valor errado.	Implementação
FF02	Internal Fault	As causas raiz das falhas internas estão na implementação interna de microservices individuais	Um microserviço de processamento de pagamento calcula incorretamente as taxas de transação, resultando em discrepâncias financeiras no sistema.	Implementação

6.10 Taxonomia Preliminar versus Taxonomia Final

A diferença entre a taxonomia preliminar e a final é notável e demonstra um processo de refinamento e aprimoramento. Inicialmente, na taxonomia preliminar, havia um total de 117 faltas classificadas em seis *Requisitos Não Funcionais (RNFs)* e relacionadas a 14 características de microsserviços. No entanto, após análise e discussões em um grupo focal com especialistas, identificou-se que algumas faltas eram redundantes ou ambíguas.

Como resultado desse processo de revisão e aperfeiçoamento, o conjunto de faltas foi reduzido para 106, eliminando as redundâncias e ambiguidades. Por exemplo, as faltas *Delay Fault* (PF12) e a *Network Transmission Delay* (PF15) da taxonomia preliminar foram agrupadas em uma nova falta, chamada *Delay Fault*. A redução no número de faltas foi acompanhada por uma otimização nas categorias de classificação das faltas.

A Tabela 21 apresenta a forma como as faltas foram distribuídas nas taxonomias preliminar e final.

Tabela 21 – Distribuição de Faltas na Taxonomia Final e Preliminar

Categoria	**Quantidade Preliminar**	**Quantidade Final**
Falta de Desempenho	39	38
Falta de Confiabilidade	14	12
Falta de Segurança	35	35
Falta de Manutenibilidade	19	14
Falta de Compatibilidade	8	5
Falta de Funcionalidade	2	2

Conforme observado na tabela, a redução no número de faltas ocorreu principalmente nas categorias de *Manutenibilidade* e *Compatibilidade*. Essa redução ocorreu devido à reclassificação e reagrupamento de algumas faltas, conforme descrito nas subseções 6.1 a 6.9. Essas mudanças tornaram a taxonomia final mais concisa e clara, eliminando redundâncias e ambiguidades que foram identificadas durante o processo de revisão.

O processo de refinamento demonstra a importância da contribuição dos especialistas no grupo focal e método delphi para a criação de uma taxonomia mais precisa e útil. A taxonomia final oferece um esquema de classificação para a organização das faltas em aplicações de microsserviços, facilitando a compreensão e o gerenciamento de desafios nessas aplicações.

6.11 Considerações Finais

O capítulo apresentou a criação da taxonomia de faltas para aplicações baseadas em microsserviços. Inicialmente, foi descrito o processo de desenvolvimento da taxonomia, que envolveu a análise e categorização de faltas relacionadas a requisitos não funcionais e características de microsserviços. Foi destacada a importância da colaboração de especialistas por meio de um grupo focal para refinar a taxonomia preliminar e reduzir redundâncias e ambiguidades. Por outro lado, O método Delphi foi essencial para avaliar e melhorar a taxonomia visto que permitiu reunir as opiniões dos especialistas de forma estruturada, ajudando a refinar e validar a taxonomia com base em feedbacks iterativos.

A taxonomia final foi apresentada, classificando as faltas em seis categorias de *RNFs*: *Desempenho* , *Confiabilidade*, *Segurança*, *Manutenibilidade*, *Compatibilidade* e *Funcionalidade*. Cada categoria foi subdividida em subcategorias correspondentes, e as faltas foram relacionadas

às características de microsserviços apropriadas. Além disso, a taxonomia final teve um total de 106 faltas, refletindo a eliminação de faltas redundantes e ambíguas durante o processo de refinamento.

A transição da taxonomia de faltas da versão preliminar para a final envolveu melhorias substanciais. Após uma avaliação minuciosa por especialistas, foram eliminadas faltas redundantes e ambíguas, resultando em uma taxonomia mais precisa e eficaz. A versão preliminar apresentava 117 faltas, enquanto a versão final foi refinada para conter 106 faltas. As melhorias incluíram uma redistribuição das faltas entre as categorias de *RNFs*. A categoria *Desempenho* passou de 39 para 36 faltas, *Confiabilidade* de 14 para 12, *Segurança* permaneceu com 35, *Manutenibilidade* diminuiu de 19 para 14, *Compatibilidade* foi reduzida de 8 para 5, e *Funcionalidade* permaneceu com 2 faltas. Essas mudanças refletem um refinamento, aprimorando a utilidade da taxonomia na análise de faltas em aplicações de microsserviços.

A taxonomia de faltas é uma ferramenta para a compreensão e classificação de faltas em aplicações de microsserviços. Ela fornece uma estrutura organizada para analisar e abordar as faltas em diferentes contextos, contribuindo para o aprimoramento da qualidade e desempenho das aplicações de microsserviços. O próximo capítulo deste trabalho apresenta como foi realizada a avaliação da taxonomia.

7 AVALIAÇÃO DA TAXONOMIA

Para avaliar a taxonomia preliminar de faltas e obter feedbacks para a evolução da mesma, foram realizadas duas avaliações: (1) Grupo Focal e (2) Método Delphi. Dessa maneira, foram convidados especialistas em Microsserviços e/ou Engenharia de Software com amplo conhecimento teórico e prática sobre o tema.

7.1 Perfil dos Especialistas

Para melhor delinear o perfil dos especialistas, foi enviado um formulário com perguntas sobre sua formação acadêmica e tempo de experiência na área. Foram convidados seis especialistas, cujos perfis são apresentados na Tabela 22. A tabela apresenta o id de cada especialista, sua titulação acadêmica, ocupação atual, área de experiência e tempo de experiência na respectiva área.

Tabela 22 – Perfil dos Especialistas

ID	Titulação	Ocupação	Áreas de Experiência	Anos
E1	Mestre	Doutorando	Engenharia de Software	6
E2	Doutor	Professor e Líder Técnico	Engenharia de Software	10
E3	Doutor	Pesquisador	Engenharia de Software	10
E4	Mestre	Doutorando e Professor da Graduação	Engenharia de Software e Microsserviços	11
E5	Mestre	Doutorando e Gerente de Projeto	Engenharia de Software	8
E6	Bacharel	Mestrando e Desenvolvedor	Arquitetura e Engenharia de Software	13

Todos os especialistas possuem pelo menos seis anos de experiência nas áreas de Microsserviços e/ou Engenharia de Software, além disso, todos já haviam trabalhado com avaliação de Requisitos Não Funcionais. É importante mencionar que o E6 não conseguiu participar do Método *Delphi* por questões pessoais, participando apenas do Grupo Focal

7.2 Grupo Focal

Um Grupo Focal é uma técnica de pesquisa qualitativa que envolve a reunião de um pequeno grupo de participantes, geralmente de 6 a 12 pessoas, para discutir tópicos específicos de forma aprofundada. Essas discussões são moderadas por um facilitador e têm o objetivo de coletar informações detalhadas sobre as percepções, opiniões e experiências dos participantes

em relação ao assunto em questão. De acordo com Krueger e Casey (2014), os grupos focais são amplamente utilizados na pesquisa social para explorar questões complexas e obter percepções valiosas. Eles são especialmente úteis para compreender as atitudes dos participantes e suas interações sociais, o que os torna uma ferramenta valiosa em estudos qualitativos.

Para a estruturação do grupo focal foi elaborado um *checklist* de 24 questões (22 objetivas e 2 subjetivas) divididas em cinco seções, as quais são apresentadas a seguir:

- **Organização:** questões relacionadas ao esquema de classificação da taxonomia;
- **Compreensão:** questões relacionadas ao conteúdo da taxonomia;
- **Completude:** questões relacionadas à integralidade e totalidade das informações apresentadas na taxonomia;
- **Wiki:** questões relacionadas à forma como a taxonomia foi apresentada na Wiki Digital; e
- **Priorização:** questões relacionadas à forma como poderiam ser feitas priorizações com base nas características e *RNFs* da taxonomia.

O *template* do *checklist* com as seções e suas respectivas questões são apresentadas no Apêndice C. Vale ressaltar que as questões Q23 e Q24 são questões subjetivas e as demais objetivas.

Após a elaboração do *checklist*, foi enviado um e-mail para os especialistas com o link da Wiki Digital com a Taxonomia Preliminar de Faltas para leitura prévia e por parte dos especialistas e com o convite para um encontro presencial para a realização do Grupo Focal, que veio a ocorrer no dia 22/09/2022, no Laboratório do Grupo de Pesquisa Grupo de Redes de Computadores, Engenharia de Software e Sistemas (GREat) da UFC. O grupo focal teve a presença dos seis especialistas e um facilitador (autor deste trabalho) e teve 2 horas de duração. Durante o encontro, o facilitador fez uma breve apresentação sobre a taxonomia e então iniciou a aplicação do *checklist* , passando item por item e mediando o debate entre os especialistas. Ao fim do encontro, foram geradas 6 folhas de repostas (uma para cada especialista) e uma gravação em áudio contendo todas as discussões do encontro.

Após o encontro, através das folhas de respostas dos especialistas, cada questão foi analisada. O objetivo foi entender o nível de concordância entre eles para cada item do *checklist*. A Figura 9 sumariza as folhas de repostas.

Para analisar se houve concordância entre as respostas dos especialistas, foram observados os seguintes critérios:

1. se metade ou mais dos participantes (>=3) acrescido de um, ou seja, quatro ou mais

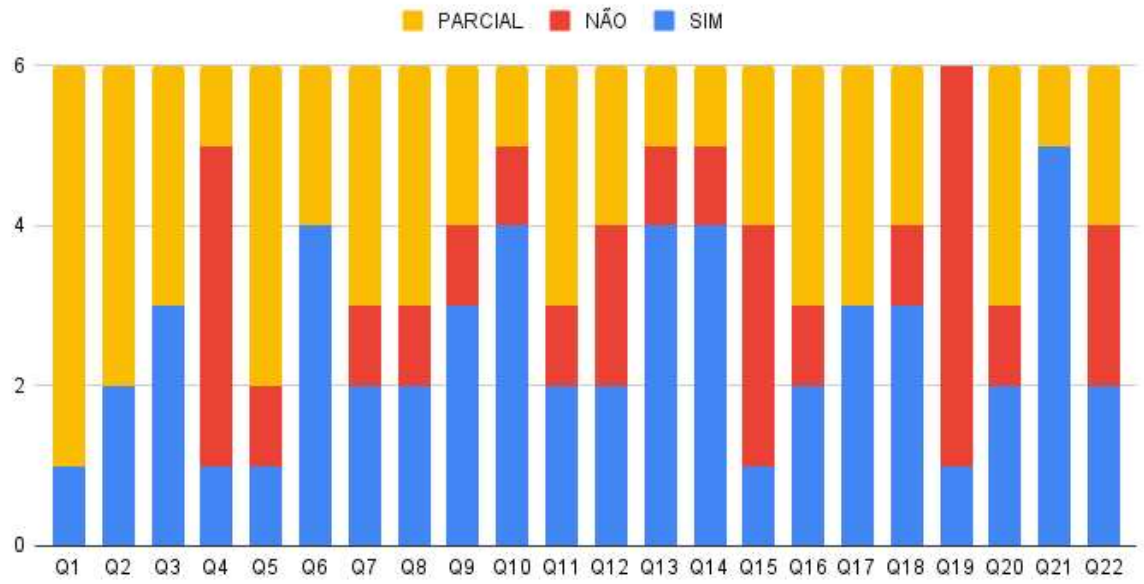

Figura 9 – Respostas por Questão - Grupo Focal

participantes responderam a questão com a mesma resposta (SIM, NÃO ou PARCIAL), houve concordância.

2. se menos da metade dos participantes (< 3) acrescido de um, ou seja, três ou menos participantes responderam a questão com a mesma resposta, não houve concordância.

Ao observar a Figura 9 e os critérios definidos acima, pode-se inferir que houve concordância em 10 das 22 questões objetivas (Q1,Q2,Q4,Q5,Q6,Q10,Q13,Q14,Q19,Q21), sendo as questões Q1 (5 Parcial), Q19 (5 Não) e Q21 (5 Sim) as que houveram o maior número de especialistas (4) concordando. Para melhor exemplificar: através dessas concordância, é possível perceber que os especialistas concordam que:

– **Algumas faltas são bem descritas e outras não:** pode-se perceber, através da Q1 (As faltas estão bem descritas?), que existiam faltas que estavam bem descritas e outras que não, isso porque 5 dos 6 especialistas responderam a questão como PARCIAL;
– **Não existe alguma falta relacionada à microsserviço que não foi apresentada na taxonomia:** através da Q19 (Existe alguma falta relacionada à microsserviço que não foi apresentada na taxonomia?), pode-se perceber que não havia outra falta de conhecimento dos especialistas, que era relacionada a microsserviços mas que não foi apresentada na taxonomia, isso porque 5 dos 6 especialistas responderam a pergunta como NÃO; e
– **As seções da Wiki podem ser reutilizadas de forma fácil:** pode-se perceber, através da Q21 (A wiki possui todas as seções necessárias para seu uso?) que a Wiki Digital possui

todas as seções que são necessáriar para sua utilização, isso porque 5 dos 6 especialistas responderam a questão como SIM.

Por outro lado, nas outras 10 questões objetivas (Q3, Q7, Q8, Q9, Q11, Q12, Q15, Q16, Q17, Q18, Q20 e Q22), não houve concordância entre os especialistas. Ao analisar estas questões de forma individual, pode-se perceber que os motivos que levaram à discordância entre os especialistas, foram:

- **Algumas faltas estavam ambíguas:** pode-se perceber que alguns especialistas encontraram faltas ambíguas ou com informações inconsistentes, como foi citado pelo E2 ao responder a Q3 (As faltas estão claras e não ambíguas?): *"Algumas das descrições são ambíguas e difíceis de entender as diferenças entre elas e como ocorrem propriamente dito."*
- **A definição de algumas características de microsserviços não estavam claras:** alguns especialistas acharam que alguma das características estavam definidas de forma muito resumida, como foi citado pelo E1 ao responder a Q8 (A definição de cada característica de microsserviço da taxonomia está clara?): *"Seria interessante ampliar a descrição MC03 e MC04"*;
- **O relacionamento entre as faltas e as características de microsserviços não estavam bem definidos:** alguns especialistas tiveram uma percepção de que algumas faltas poderiam estar relacionadas com mais de uma característica, como foi citado pelo E1 ao responder a Q11 (O relacionamento entre as faltas e as características de microsserviços está bem definido? (de forma geral)) : *Algumas faltas parecem ter mais relação com mais de uma característica*; e
- **Algumas faltas não estavam classificadas corretamente:** Assim como no relacionamento das faltas com as características de microsserviços, alguns especialistas também tiveram a percepção de que, por vezes, algumas faltas poderiam estar classificadas em mais de uma categoria (ou RNF), como foi comentado pelo E2 ao responder a Q10 (O relacionamento entre as faltas e categorias está bem definido? (de forma geral)): *"Não. Existem faltas que precisam ser atualizadas ou reclassificadas. Além disso, há a possibilidade de uma falta estar relacionada de forma primária a uma categoria, mas impacta em outras. Isso pode ter gerado uma confusão na hora da classificação."*.

Finalmente, nas questões subjetivas (Q23 e Q24), foram identificados aspectos relevantes à arquitetura de microsserviços, destacando a percepção dos participantes sobre

características e requisitos não funcionais (RNF). A análise dessas respostas proporcionou *insights* sobre as preferências e o entendimento dos especialistas em relação à concepção de uma arquitetura baseada em microsserviços.

Com relação à Q23 (Cite três características de microsserviços que você priorizaria caso fosse construir uma arquitetura de microsserviços.), alguns especialistas não conseguiram indicar quais características priorizariam, pois haviam características que foram definidas de forma sucinta. Por outro lado, alguns especialistas indicaram, por exemplo:

- E1: Resiliência, Troca de Dados e Modelos de Interação
- E3: Troca de Dados, Descoberta de Serviços e Implementação
- E5: Modelos de Interação, Tempo de Execução e Resiliência

Com base nas respostas acima, pode-se perceber que cada especialista que respondeu a pergunta, indicou características diferentes, no entanto, algumas características foram indicadas por mais de 1 especialista, como por exemplo: Resiliência, Troca de Dados e Modelos de Interação, indicada por 2 especialistas.

Por outro lado, com relação à Q24 (Cite três RNF de microsserviços que você priorizaria caso fosse construir uma arquitetura de microsserviços.), um especialista indicou que todos os *RNFs* são importantes, outro não respondeu e quatro restantes indicaram *RNFs*, conforme descrito abaixo:

- E1: Desempenho, Confiabilidade e Segurança
- E3: Funcionalidade, Desempenho e Confiabilidade
- E4: Desempenho, Confiabilidade, Segurança
- E5: Confiabilidade, Funcionalidade, Compatibilidade

Dessa forma, foi possível perceber que de modo geral, os *RNFs* Confiabilidade e Desempenho seriam mais prioritários na construção de uma arquitetura de microsserviços, isso porque 4 dos 6 especialistas indicaram que priorizariam estes *RNFs*.

Com base nos resultados da avaliação, foi então feito um refinamento da taxonomia com correções e melhorias, para que a mesma pudesse ser avaliada posteriormente.

7.3 Método Delphi

Após o refinamento da taxonomia preliminar ser refinada com base nos resultados obtidos no Grupo Focal, uma nova avaliação da taxonomia (versão final) foi feita, desta vez utilizando o Método *Delphi*.

O Método *Delphi* é uma técnica de pesquisa que envolve a coleta de opiniões e julgamentos de um painel de especialistas, geralmente de forma anônima. Os especialistas respondem a uma série de rodadas de questionários ou pesquisas com o objetivo de alcançar um consenso ou prever futuros eventos. O Método *Delphi* é frequentemente utilizado em situações em que não há uma resposta definitiva e é necessário obter a opinião de especialistas. É uma abordagem valiosa para a tomada de decisões e a previsão de tendências. De acordo com (DALKEY; HELMER, 1969) em seu artigo "An Experimental Application of the *Delphi* Method to the Use of Experts", o Método *Delphi* pode ser uma ferramenta eficaz para explorar questões complexas e obter *insights* de especialistas em diversas áreas.

Conforme apresentado anteriormente, participaram do Método *Delphi* cinco Especialistas, os mesmos que participaram do grupo focal, com exceção do E6 (ver Tabela 22). Inicialmente, foi desenvolvido um questionário composto por 15 questões (ver Apêndice D),das quais 11 eram objetivas (Q1, Q2, Q3, Q4, Q5, Q6, Q8, Q9, Q10, Q12 e Q13) utilizando a escala *Likert* como método de coleta de dados e 4 (Q7, Q11, Q14 e Q15) eram subjetivas. Diferentemente do *checklist* do grupo focal, neste questionário, novas questões foram elencadas e as respostas foram coletadas com base na escala likert. O questionário foi projetado para avaliar as opiniões, atitudes e percepções dos respondentes em relação a taxonomia de faltas.

A escolha da escala *Likert* baseou-se em sua ampla aceitação na pesquisa social (LIKERT, 1932) e em sua capacidade de medir de forma eficaz atitudes e percepções dos participantes, fornecendo dados valiosos para análise e interpretação dos resultados. Este questionário foi enviado aos especialistas para que eles respondessem de forma anônima, com base nas suas experiências e conhecimentos. O critério de parada utilizado para finalizar as rodadas de questionário foi: caso mais da metade das questões objetivas (5) obtenham valores da moda para >= 4 (concordo), não é necessário realizar uma nova rodada de questionário, pois entende-se que houve uma concordância entre os especialistas na maioria das perguntas.

A escolha da moda como medida nesta avaliação foi pelo fato dela ser uma medida estatística que representa o valor mais frequente em um conjunto de dados. Ao contrário da média, que calcula a média aritmética de todos os valores, a moda identifica o valor que ocorre com maior frequência. A principal vantagem da moda é sua capacidade de identificar o valor mais comum em um conjunto de dados, sendo especialmente útil quando os dados apresentam valores discrepantes ou distribuição assimétrica

Para que fosse possível calcular as modas, foi atribuído os seguintes valores para cada

categoria de resposta possível {Concordo Totalmente = 5, Concordo = 4, Nem concordo, Nem discordo = 3, Discordo = 2, Discordo Totalmente = 1}. Ou seja, se em um questão 3 especialistas afirmarem que "Concordo Totalmente"e dois especialistas afirmarem que "Não concordo, nem discordo", a moda da questão é 5 já que a maioria dos especialistas (3) responderam concordo totalmente (5). A Figura 10 apresenta um sumário das repostas coletadas dos especialistas por meio do questionário.

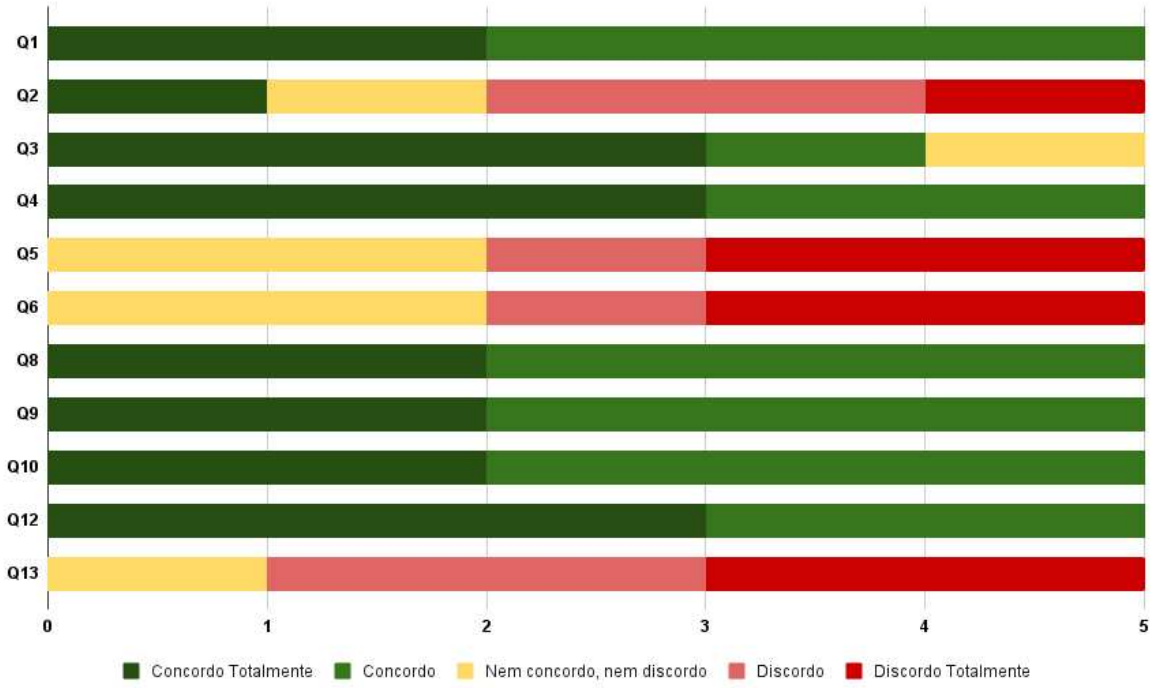

Figura 10 – Respostas por Questão - Método *Delphi*

Com base na Figura 10, pode-se perceber que na maior parte das questões (Q1, Q3, Q4, Q8, Q9, Q10, Q12), a maioria dos especialistas concordaram ou concordaram totalmente sobre as questões. Por exemplo, na Q1 (Na sua opinião, a descrição das categorias e subcategorias de faltas é clara e compreensível?), 60% (3 dos 5 especialistas) concordaram totalmente e 40% (2 dos 5) especialistas concordaram com o que era indagado na questão, o que resultou em uma moda igual a 5.

Por outro lado, nas outras quatro questões (Q2,Q5, Q6 e Q13) do questionário, houve divergência entre os especialistas, alguns concordaram, outros se mostraram neutros e outros discordaram do que a questão afirmava. Por exemplo, na questão Q2 (Existem termos ou conceitos que você considera ambíguos ou difíceis de entender na taxonomia?), um especialista concordou totalmente, um se mostrou neutro e 3 discordaram ou discordaram totalmente.

A Tabela 23 apresenta as respostas das 11 questões objetivas do questionário, de acordo com a escala likert.

Tabela 23 – Respostas do questionário de Avaliação utilizando Método *Delphi*

Questão	**Concordo Totalmente**	**Concordo**	**Nem Concordo, Nem Discordo**	**Discordo**	**Discordo Totalmente**
Q1	2	3	0	0	0
Q2	1	0	1	2	1
Q3	3	1	1	0	0
Q4	3	2	0	0	0
Q5	0	0	2	1	2
Q6	0	0	2	1	2
Q8	2	3	0	0	0
Q9	2	3	0	0	0
Q10	2	3	0	0	0
Q12	3	2	0	0	0
Q13	0	0	1	2	2

Com base nas repostas obtidas, ao observar a distribuição das respostas, é possível perceber que a opção ”Concordo Totalmente” recebeu a maioria das respostas, 18, seguido de ”Concordo”, que recebeu 17 respostas. Por outro lado, ao calcular a moda das respostas para cada pergunta, é possível perceber que houve uma concordância geral, ou seja, as modas para a maioria das perguntas estão acima de 3 (a categoria "Nem Concordo, Nem Discordo"), o que sugere uma tendência de concordância geral entre os respondentes em relação às afirmações. Isso indica que a maioria dos respondentes concorda ou concorda totalmente com as afirmações apresentadas. Além disso, As perguntas Q1, Q3, Q8, Q9, Q10 e Q12 têm modas superiores a 4 (a categoria "Concordo"), o que indica uma alta concordância em relação a essas questões. Isso sugere que a maioria dos respondentes concorda ou concorda totalmente com essas afirmações.

De todo modo, houveram questões úteis de se explorar, pois existem perguntas com variação nas respostas (como Q2, Q5, Q6 e Q13). As questões Q5, Q6 e Q13 tiveram modas inferiores a 3 (a categoria "Nem Concordo, Nem Discordo"ou abaixo), o que pode indicar uma tendência de discordância ou mesmo falta de opinião clara em relação a essas perguntas.

Com relação as questões subjetivas (Q7, Q11, Q14 e Q15), as seguintes percepções foram coletadas:

– **Q7 (Você pode fornecer um exemplo de como a taxonomia poderia ser aplicada em um cenário real de desenvolvimento ou teste?)**: *"Creio que a taxonomia pode auxiliar desenvolvedores a gerenciar as possíveis falhas relacionadas a aplicações arquitetadas*

como micro serviços. Por exemplo, antes de iniciar o desenvolvimento, o time pode usar a taxonomia para ponderar os riscos e criar um plano de ação caso alguma falta ocorra."

- **Q11 (Você acredita que alguma falta foi classificada incorretamente? Se sim, pode fornecer um exemplo?)**: *"Acredito que foi tudo classificado corretamente"*
- **Q14 (Você tem sugestões específicas para melhorar a taxonomia? Isso pode incluir adições, modificações ou esclarecimentos.)**: *"Acredito que mais exemplos reais podem ser dado, no contexto de microsserviço. Por exemplo: Por um bug, o sistema fica em loop, ou por indisponibilidade do serviço algo acontece. Acredito que dar exemplos mais práticos (cenários reais) facilite a identificação por parte de quem estiver lendo"*
- **Q5 (há algum feedback adicional que você gostaria de compartilhar sobre a taxonomia ou este processo de avaliação?)**: *"Gostei muito da Taxonomia, mas senti falta de um manual simples de uso dessa taxonomia para um projeto, ou para auxiliar a construção de planos de teste. Fica como proposta futura."*

Com sete questões (Q1, Q3, Q4, Q8, Q9, Q10 e Q12) alcançando moda ≥ 4, o experimento foi encerrado na primeira rodada do método Delphi. Essa decisão reflete o consenso significativo entre os especialistas, garantindo eficiência no processo e resultando em informações valiosas.

7.4 Considerações Finais

Este capitulo apresentou duas avaliações da proposta deste trabalho. A avaliação da taxonomia preliminar de faltas por meio do Grupo Focal e do Método *Delphi* proporcionou *insights* para a evolução da taxonomia. A participação de especialistas com experiência em Microsserviços e Engenharia de Software enriqueceu a análise, contribuindo para uma compreensão mais aprofundada das percepções e opiniões sobre a taxonomia.

O perfil dos especialistas, apresentado na Tabela 22, revela uma diversidade de experiências e conhecimentos, enriquecendo a abordagem do estudo. Mesmo diante das divergências em algumas questões, a participação ativa e as contribuições dos especialistas foram fundamentais para o refinamento da taxonomia. A utilização do Grupo Focal, técnica qualitativa, permitiu explorar aspectos subjetivos e identificar pontos específicos de ambiguidade, dificuldade de compreensão e possíveis melhorias na taxonomia. As respostas obtidas nas questões objetivas indicaram áreas de concordância e discordância entre os especialistas, direcionando ajustes necessários. Por outro lado, o Método Delphi, ao empregar uma abordagem mais quantitativa,

reforçou a tendência de concordância geral entre os especialistas. A análise das respostas das questões objetivas revelou uma maioria de respostas positivas, indicando um alinhamento consistente em relação às afirmações apresentadas. Nas questões subjetivas, destacaram-se percepções sobre a aplicabilidade prática da taxonomia em cenários reais de desenvolvimento e teste. Além disso, a sugestão de fornecer exemplos práticos e a solicitação de um manual de uso para a taxonomia evidenciam oportunidades de aprimoramento para tornar a taxonomia mais acessível e aplicável. Diante do consenso atingido nas seis questões do Método Delphi, optou-se por encerrar o experimento na primeira rodada. Essa decisão reflete a eficácia do método em alcançar convergência de opiniões, validando a taxonomia refinada como uma ferramenta robusta para análise de faltas em arquiteturas de microsserviços.

As considerações dos especialistas, tanto em aspectos positivos quanto em sugestões de melhorias, orientarão futuras etapas de desenvolvimento da taxonomia. A interação entre teoria e prática proporcionada pelos especialistas é crucial para a evolução contínua da taxonomia, contribuindo para sua relevância e utilidade na indústria de desenvolvimento de software. Por fim, a avaliação da taxonomia representa um passo significativo em direção à sua maturidade, integrando a expertise dos especialistas à estrutura conceitual proposta. As contribuições desses especialistas forneceram uma validação valiosa, consolidando a taxonomia como uma ferramenta para a identificação e categorização de faltas em arquiteturas de microsserviços.

8 DISCUSSÕES E AMEAÇAS À VALIDADE

Neste capítulo de discussões, os resultados deste estudo são examinados, com foco nas Questões de Pesquisa (QPs) investigadas ao longo desta dissertação de mestrado. Com base nos achados da pesquisa, as QPs são analisadas, proporcionando uma visão aprofundada das descobertas e suas implicações para o campo de estudo em questão. Abaixo são discutidas cada Questão de Pesquisa de forma individual

QP1. Quais faltas ocorrem em aplicações baseadas em microsserviços?

Com base na identificação das faltas realizada no estudo, um conjunto de 136 faltas relacionadas à aplicações baseadas em microsserviços foi identificado. Deste conjunto, 33 faltas foram classificadas previamente na literatura em 11 categorias, no entanto, a maioria das faltas (103) não havia sido classificada. Além disso, 24 das 136 faltas foram citadas por mais de um estudo. Por exemplo, faltas relacionadas ao desempenho, como *CPU Hog* e *Memory Leak* foram as mais citadas, por 5 e 4 estudos (em um total de 28), respectivamente. Uma vez que essas faltas foram mencionadas por vários estudos, esse resultado pode indicar que elas ocorrem com mais frequência. No entanto, essa suposição não foi validada neste estudo, uma vez que está fora de seu escopo, e, portanto, outros estudos empíricos podem ser conduzidos para confirmar esse resultado. De todo modo, à medida que o estudo foi avançando e a taxonomia de faltas foi sendo desenvolvida, percebeu-se que algumas faltas eram similares e poderiam ser agrupadas. Na taxonomia preliminar de faltas foram apresentadas 117 faltas, isso porque 26 faltas do catálogo foram agrupadas em sete faltas, por exemplo as faltas do catálogo: *Infinite Loop* (F110) e *Deadlock Fault* (F116) foram agrupadas em uma nova falta chamada *Deadlock Fault* que fez parte da taxonomia, por ambas as faltas apresentarem a mesma definição. Por outro lado, com após a avaliação da taxonomia preliminar com os especialistas, percebeu-se que outras faltas também poderiam ser agrupadas, seja por estarem redundates ou ambiguas, por exemplo as faltas *Delay Fault* (PF12) e a *Network Transmission Delay* (PF15) da taxonomia preliminar foram agrupadas em uma nova falta, chamada *Network Transmission Delay* (PF14). Portanto, ao fim do estudo, foi possível inferir que existem pelo menos 106 faltas (taxonomia final) que podem estar relacionadas à aplicações baseadas em microsserviços.

QP2. Como as faltas são classificadas em aplicações baseadas em microsserviços?

A partir do MLR, identificou-se que 33 das 136 faltas já haviam sido classificadas em 11 categorias, que eram: 12 faltas em *Performance*; 4 em *Workload*; 5 em *VM Instance*;

3 em *Hardware Usage*; 2 em *External Change Anomaly*; 2 em *TOF bug*; e apenas 1 falta em cada uma das categorias *Implementation*, *Instance Management*, *Environment Configuration*, *Invocations Management* e *Internal Change Anomaly*. Por exemplo, encontrou-se apenas a falta *Asynchronous Interaction Fault* (F04 do catálogo) na categoria *Invocations Management* citada por (ZHOU *et al.*, 2019). Vale ressaltar que as faltas classificadas citadas em mais de um estudo não necessariamente foram classificadas em todos os estudos, como mencionado no exemplo da falta "CPU Hog"na subseção 5.

Com base nessas evidências, foi observado que algumas das categorias de faltas identificadas eram bastante específicas, chegando a conter apenas uma falta em sua composição. Essa especificidade levantou a necessidade de uma reavaliação e refinamento das categorias, com o objetivo de torná-las mais bem definidas ou, possivelmente, de incluí-las em uma estrutura de classificação como subcategorias mais especializadas.

Portanto, para estabelecer a taxonomia, optou-se pela estratégia de categorizar as faltas com base nos Requisitos Não Funcionais, como Desempenho, e, ao mesmo tempo, de subdividir as faltas com base em suas subcaracterísticas, como a Capacidade. Essa abordagem permitiu uma organização mais precisa e minuciosa das faltas, levando em consideração sua natureza altamente específica. Além disso, como parte desse processo, estabeleceu-se uma correlação entre as faltas da taxonomia e 14 características inerentes à arquitetura de microsserviços, como a característica Tempo de Execução, ampliando ainda mais a compreensão do panorama das faltas nesse contexto específico.

QP3. Quais requisitos não funcionais estão relacionados às faltas identificadas em aplicações baseadas em microsserviços?

Foram observadas que 66 das 117 faltas na taxonomia preliminar estavam relacionadas a algum Requisito Não Funcional. Inicialmente, três *RNFs* foram identificados: Desempenho, Segurança e Confiabilidade, e, portanto, essas 66 faltas foram distribuídas entre esses três *RNFs*, conforme apresentado na Tabela 24. As outras 51 faltas não apresentaram correlação com nenhum *RNF* em seus estudos de origem.

Para definir a taxonomia de forma mais abrangente, as 117 faltas foram analisadas uma a uma, a fim de determinar a qual *RNF* estavam relacionadas, conforme discutido na subseção 6.1. Como resultado dessa etapa de análise, mais três *RNFs* foram incluídos no conjunto: Manutenibilidade, Compatibilidade e Funcionalidade. É importante mencionar que essas 66 faltas que já haviam sido relacionadas com *RNF* na literatura também foram analisadas

Tabela 24 – Faltas relacionadas à *RNFs* na literatura

RNF	ID Falta
Segurança	SF01, SF02, SF03, SF04, SF05, SF06, SF07, SF08, SF09, SF10, SF11, SF12, SF13, SF14, SF17, SF18, SF19, SF20, SF21, SF22, SF23, SF24, SF25, SF26, SF27, SF28, SF34 e SF35
Desempenho	PF01, PF02, PF04, PF05, PF06, PF07 RF10, PF20, PF21, PF22, PF24, PF25, PF26, PF27, PF29, PF30, PF31, PF32, PF33, PF34, PF35, PF36, PF37, PF38, PF39, SF15 e SF16
Confiabilidade	PF01, PF02, PF04, RF10, PF24, PF25, PF29, PF31, PF32, SF15, SF16

e quando necessário foi feita uma reclassificação de RNF, por exemplo, a falta da taxonomia preliminar *Performance Degradation* (PF02) havia sido inicialmente relacionada com o *RNF* Confiabilidade na literatura, mas neste estudo, na etapa de análise, foi visto que ela deveria ser classificada em Desempenho.

O conjunto final de *RNFs* apresentados na taxonomia para categorizar as 117 faltas consiste em: Desempenho, *Segurança*, *Confiabilidade*, *Manutenibilidade*, Compatibilidade e Funcionalidade. Essa abordagem possibilitou uma organização mais precisa e detalhada das faltas, considerando sua associação específica a cada um dos *RNFs* considerados.

8.1 Ameaças à Validade

Nesta seção, são discutidas as potenciais ameaças à validade deste estudo. A validade de um estudo refere-se à capacidade de medir e avaliar com precisão o que o estudo se propõe a investigar. É essencial abordar essas ameaças de forma abrangente, uma vez que podem influenciar a credibilidade e a confiabilidade dos resultados obtidos. A seguir, são discutidas as ameaças à validade que podem ter impacto nos resultados deste trabalho.

8.1.1 Ameaças Internas

A análise das ameaças internas à validade deste estudo é fundamental para compreender a robustez da identificação de estudos primários e o processo de extração de dados. As estratégias adotadas visaram minimizar o viés e assegurar a integridade dos resultados.

8.1.1.1 Ameaças Relacionadas à Identificação de Estudos Primários

Estas ameaças referem-se à possibilidade de não identificar alguns estudos relevantes durante o processo de seleção de artigos. Para mitigar essa ameaça, foram adotadas estratégias

como o uso de palavras-chave distintas, análises abrangentes na *string* de busca e a escolha de bases de dados amplamente reconhecidas. Além disso, um protocolo de revisão foi estabelecido, definindo diretrizes claras e critérios rigorosos. Apesar dos esforços para minimizar o viés, persiste a possibilidade de omissão de estudos relevantes devido a limitações inerentes ao processo de seleção.

8.1.1.2 Ameaças Relacionadas à Extração de Dados

Estas ameaças estão associadas à coleta e extração de dados dos artigos revisados. Desafios surgem de interpretações variadas nos artigos, levando a incertezas na extração. Para mitigar isso, foi estabelecido um processo de discussão e colaboração entre revisores. A abordagem colaborativa contribuiu para reduzir a incerteza, mas a subjetividade pode persistir em algumas situações, enfatizando a importância da documentação detalhada das decisões durante a extração.

8.1.2 Ameaças Externas

A análise das ameaças externas à validade concentra-se na avaliação da taxonomia, destacando os cuidados tomados para mitigar possíveis viés e garantir a confiabilidade dos resultados.

8.1.2.1 Ameaças Relacionadas à Avaliação da Taxonomia

A avaliação da taxonomia por meio de Grupo Focal e Método *Delphi* introduz ameaças à validade. No Grupo Focal, o viés pode ocorrer devido à influência indevida. Para mitigar, foram adotadas práticas rigorosas, como liderança imparcial e documentação transparente. No Método *Delphi*, a ameaça de consenso prematuro é minimizada incentivando justificativas detalhadas dos especialistas, mesmo em consensos rápidos. Essa abordagem visa garantir robustez na avaliação da taxonomia, promovendo transparência e imparcialidade. Além disso, no Método *Delphi*, há uma ameaça potencial de consenso prematuro, em que o consenso é alcançado rapidamente, sem uma exploração adequada das perspectivas e argumentos dos especialistas. Para minimizar essa ameaça, os especialistas foram incentivados a fornecer justificativas detalhadas para suas decisões, mesmo que o consenso tenha sido alcançado na primeira rodada. Isso permitiu que todas as perspectivas fossem consideradas e que a decisão fosse baseada em uma

análise mais completa das opiniões dos especialistas. Essa abordagem de coleta de justificativas detalhadas e consideração abrangente das perspectivas dos especialistas, mesmo em uma única rodada, foi adotada para garantir a robustez do processo de avaliação da taxonomia, promovendo a transparência e a imparcialidade.

Por outro lado, a utilização da moda para análise de concordância em vez de métodos estatísticos mais robustos, como o Fleiss Kappa, representa uma ameaça à validade da interpretação da concordância entre os avaliadores. Para mitigar essa ameaça à validade, foi adotada uma estratégia dupla de avaliação, envolvendo não apenas a análise de concordância por meio da moda, mas também a utilização de dois métodos distintos: o Grupo Focal e o Método Delphi. Apesar da escolha da moda como métrica principal, a adoção de procedimentos claros e comunicação constante entre os avaliadores buscou assegurar uma interpretação consistente dos dados.

8.2 Considerações Finais

Neste capítulo, as questões de pesquisa foram abordadas com o objetivo de identificar as faltas em aplicações de microsserviços e desenvolver uma taxonomia para categorizá-las. Com base nos resultados obtidos, foi possível identificar 136 faltas no catálogo, das quais 33 foram categorizadas em 11 áreas distintas, englobando diversos Requisitos Não Funcionais (NRFs), incluindo Desempenho, Segurança, Confiabilidade, Manutenibilidade, Compatibilidade e Funcionalidade. No entanto, a maioria das faltas (103) permaneceu não classificada, o que evidencia a complexidade e diversidade das faltas em aplicações de microsserviços.

Adicionalmente, as ameaças à validade que surgiram durante o estudo foram abordadas com estratégias rigorosas para mitigação, visando garantir a credibilidade e a precisão dos resultados. Este estudo proporciona uma base sólida para a compreensão das faltas em aplicações de microsserviços e suas categorizações, ao mesmo tempo em que reconhece as limitações e delineia direções futuras. Espera-se que essas descobertas contribuam para a melhoria da qualidade e confiabilidade de sistemas de software baseados em microsserviços.

9 CONCLUSÃO E TRABALHOS FUTUROS

Nesta dissertação foi apresentada a metodologia e resultados da elaboração de uma Taxonomia de Faltas para Aplicações Baseadas em Microsserviços. A pesquisa abrangeu uma revisão multivocal da literatura, a identificação e refinamento de faltas específicas associadas a microsserviços, bem como a classificação com base em requisitos não funcionais e características de microsserviços.

9.1 Resultados Alcançados

Este trabalho apresentou contribuições para o campo de aplicações baseadas em microsserviços, culminando em um conjunto abrangente de elementos que fortalecem a compreensão e a análise de faltas nesse contexto. Primeiramente, um catálogo de faltas foi elaborado, contendo 136 faltas identificadas, das quais 103 eram distribuídas em 11 categorias. Essa compilação oferece uma visão abrangente das possíveis faltas que podem surgir em ambientes de microsserviços.

Além disso, o trabalho propõe uma taxonomia de faltas, inicialmente com 117 faltas categorizadas de acordo com Requisitos Não Funcionais e relacionadas às características específicas da arquitetura de microsserviços. Essa taxonomia evolui para uma versão final, contemplando 106 faltas, ainda categorizadas por RNFs e correlacionadas com as características da arquitetura de microsserviços.

O esquema de classificação da taxonomia, organizado de acordo com 6 Requisitos Não Funcionais e suas 8 subcaracterísticas, é uma contribuição que oferece uma abordagem estruturada para analisar e classificar as faltas identificadas. Além disso, a correlação das faltas da taxonomia com 14 características inerentes à arquitetura de microsserviços enriquece a compreensão das inter-relações entre as diferentes faltas e aspectos arquiteturais.

Finalmente, o relato detalhado do processo de criação e avaliação da taxonomia proporciona uma visão transparente do método utilizado. Isso não apenas valida a robustez da taxonomia, mas também oferece percepções sobre sua aplicabilidade e limitações. Ao documentar as etapas do processo de avaliação, este trabalho contribui para a replicabilidade e a compreensão do desenvolvimento e utilidade da taxonomia proposta. No conjunto, essas contribuições consolidam este trabalho como um recurso para pesquisadores e profissionais que exploram aplicações baseadas em microsserviços.

9.2 Produção Bibliográfica

Durante o período do mestrado, o autor contribuiu com pesquisas em conferências e simpósios acadêmicos, resultando em publicações relacionadas e não relacionadas ao tema central da dissertação. A seguir, são apresentadas essas publicações, organizadas em duas tabelas.

9.2.1 Publicações Relacionadas ao Tema

A Tabela 25 a seguir lista as publicações do autor diretamente relacionadas ao tema da dissertação: a taxonomia de faltas para aplicações baseadas em microsserviços.

Tabela 25 – Publicações Relacionadas ao Tema

Nº	**Título**	**Evento**	**Referência**
1	"Towards a Fault Taxonomy for Microservices-Based Applications"	XXXVI Brazilian Symposium on Software Engineering (SBES) 2022	(SILVA *et al.*, 2022)

Esta publicação destaca o trabalho realizado na elaboração da taxonomia de faltas para aplicações baseadas em microsserviços, representando um dos principais resultados da pesquisa durante o mestrado.

9.2.2 Publicações Não Relacionadas ao Tema

A Tabela 26 a seguir enumera publicações do autor que, embora relevantes para a área de Engenharia de Software, não estão diretamente relacionadas ao tópico da taxonomia de faltas em aplicações de microsserviços.

Tabela 26 – Publicações Não Relacionadas ao Tema

Nº	**Título**	**Evento**	**Referência**
1	"Correlations among Software Testability Metrics"	XIX Simpósio Brasileiro de Qualidade de Software (SBQS), 2020	(SILVA *et al.*, 2020)
2	"Gamification in Remote Teaching of SE Courses: Experience Report"	XXXIV Simpósio Brasileiro de Engenharia de Software (SBES) 2020	(LELLI *et al.*, 2020)

Essas publicações abordaram tópicos variados na área de Engenharia de Software, como métricas de testabilidade de software e a aplicação de gamificação no ensino de Engenharia

de Software. É importante observar que, embora essas publicações não estejam diretamente relacionadas ao tema da dissertação, elas representam a diversidade de interesses e contribuições do autor na área de Engenharia de Software ao longo do mestrado.

9.3 Trabalhos Futuros

No que se refere aos trabalhos futuros, há diversas direções de pesquisa que prometem enriquecer ainda mais o conhecimento nessa área. A exploração prática da aplicação da Taxonomia de Faltas no desenvolvimento e manutenção de sistemas baseados em microsserviços pode oferecer insights valiosos sobre sua utilidade em cenários do mundo real. Além disso, pesquisas podem focar em abordagens de automação para a detecção de faltas da taxonomia, incluindo a investigação de técnicas de aprendizado de máquina e monitoramento contínuo.

A expansão da taxonomia para englobar requisitos não funcionais emergentes e características em evolução dos microsserviços é uma consideração importante para mantê-la relevante no futuro. Estudos comparativos com outras taxonomias existentes podem proporcionar uma compreensão mais detalhada de suas áreas de destaque e especialização.

Por fim, a aplicação da taxonomia em estudos de caso em ambientes de produção pode demonstrar sua eficácia e utilidade prática. Essas oportunidades de pesquisa apontam para um caminho promissor na expansão do conhecimento no campo das faltas em aplicações baseadas em microsserviços e no aprimoramento da qualidade e confiabilidade dos sistemas que adotam essa arquitetura.

REFERÊNCIAS

AHMED, J.; JOHNSSON, A.; MORADI, F.; PASQUINI, R.; FLINTA, C.; STADLER, R. Online approach to performance fault localization for cloud and datacenter services. In: **2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)**. [*S. l.*: *s. n.*], 2017. p. 873–874.

AUé, J.; ANICHE, M.; LOBBEZOO, M.; DEURSEN, A. van. An exploratory study on faults in web api integration in a large-scale payment company. In: **Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice**. New York, NY, USA: Association for Computing Machinery, 2018. (ICSE-SEIP '18), p. 13–22. ISBN 9781450356596. Disponível em: https://doi.org/10.1145/3183519.3183537.

BHAGYAVATI. Taxonomy of faults in wireless networks. In: **Symposium, 2005 Wireless Telecommunications**. [*S. l.*: *s. n.*], 2005. p. 120–125.

BOURQUE, P.; FAIRLEY, R. E.; SOCIETY, I. C. **Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0**. 3rd. ed. Washington, DC, USA: IEEE Computer Society Press, 2014. ISBN 0769551661.

CHAN, W.; CHEUNG, S.; TSE, T. Fault-based testing of database application programs with conceptual data model. In: **Fifth International Conference on Quality Software (QSIC'05)**. [*S. l.*: *s. n.*], 2005. p. 187–196.

CHEN, H.; CHEN, P.; YU, G. A framework of virtual war room and matrix sketch-based streaming anomaly detection for microservice systems. **IEEE Access**, v. 8, p. 43413–43426, 2020.

CHEN, L. Microservices: Architecting for continuous delivery and devops. In: **2018 IEEE International Conference on Software Architecture**. [*S. l.*: *s. n.*], 2018. p. 39–397.

DALKEY, N.; HELMER, O. An experimental application of the delphi method to the use of experts. **Management Science**, INFORMS, v. 15, n. 9, p. 458–467, 1969.

DANIEL, P.; SIM, K. Y. Dynamic fault visualization tool for fault-based testing and prioritization. In: **2012 International Conference on Advanced Computer Science Applications and Technologies (ACSAT)**. [*S. l.*: *s. n.*], 2012. p. 301–306.

GAO, Z.; CECATI, C.; DING, S. X. A survey of fault diagnosis and fault-tolerant techniques—part ii: Fault diagnosis with knowledge-based and hybrid/active approaches. **IEEE Transactions on Industrial Electronics**, v. 62, n. 6, p. 3768–3774, 2015.

GAROUSI, V.; FELDERER, M.; MäNTYLä, M. V.; KUUTILA, M.; OIVO, M. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. **Information and Software Technology**, v. 106, p. 101–121, 2019.

GARRIGA, M. Towards a taxonomy of microservices architectures. In: ____. [*S. l.*: *s. n.*], 2018. p. 203–218. ISBN 978-3-319-74780-4.

GEETHIKA, D.; JAYASINGHE, M.; GUNARATHNE, Y.; GAMAGE, T. A.; JAYATHILAKA, S.; RANATHUNGA, S.; PERERA, S. Anomaly detection in high-performance api gateways. In: **2019 International Conference on High Performance Computing Simulation (HPCS)**. [*S. l.*: *s. n.*], 2019. p. 995–1001.

GILL, S. S.; BUYYA, R. Failure management for reliable cloud computing: A taxonomy, model, and future directions. **Computing in Science Engineering**, v. 22, n. 3, p. 52–63, 2020.

HAT, R. **What are Microservices**. 2018. http://aiweb.techfak.uni-bielefeld.de/content/bworld-robot-control-software/.

HERBST, N.; BAUER, A.; KOUNEV, S.; OIKONOMOU, G.; EYK, E. V.; KOUSIOURIS, G.; EVANGELINOU, A.; KREBS, R.; BRECHT, T.; ABAD, C. L.; IOSUP, A. Quantifying cloud performance and dependability: Taxonomy, metric design, and emerging challenges. Association for Computing Machinery, New York, NY, USA, v. 3, n. 4, 2018. ISSN 2376-3639.

HIGASHINO, M. Application of mobile agent technology to microservice architecture. In: . New York, USA: Association for Computing Machinery, 2017. ISBN 9781450352994.

HUMBATOVA, N.; JAHANGIROVA, G.; BAVOTA, G.; RICCIO, V.; STOCCO, A.; TONELLA, P. Taxonomy of real faults in deep learning systems. In: **Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering**. New York, NY, USA: Association for Computing Machinery, 2020. (ICSE '20), p. 1110–1121. ISBN 9781450371216. Disponível em: https://doi.org/10.1145/3377811.3380395.

HUMMER, W.; INZINGER, C.; LEITNER, P.; SATZGER, B.; DUSTDAR, S. Deriving a unified fault taxonomy for event-based systems. In: **Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems**. New York, NY, USA: Association for Computing Machinery, 2012. (DEBS '12), p. 167–178. ISBN 9781450313155. Disponível em: https://doi.org/10.1145/2335484.2335504.

HöLLER, A.; KAJTAZOVIC, N.; RAUTER, T.; RöMER, K.; KREINER, C. Evaluation of diverse compiling for software-fault detection. In: **2015 Design, Automation Test in Europe Conference Exhibition (DATE)**. [*S. l.*: *s. n.*], 2015. p. 531–536.

International Organization for Standardization (ISO). **ISO/IEC 25000:2005, Software Engineering - Software Product Quality Requirements and Evaluation (SQuaRE)**. 2005.

KITCHENHAM, B. Procedures for performing systematic reviews. **Keele, UK, Keele University**, 2004.

KRUEGER, R. A.; CASEY, M. A. **Focus Groups: A Practical Guide for Applied Research**. 5th. ed. [*S. l.*]: SAGE Publications, 2014.

LELLI, V.; ANDRADE, R. M. C.; FREITAS, L. M.; SILVA, R. A. S.; FILHO, F. G. S.; GOMES, R. F.; SEVERO, J. S. de O. Gamification in remote teaching of se courses: Experience report. In: **Proceedings of the XXXIV Brazilian Symposium on Software Engineering**. New York, NY, USA: Association for Computing Machinery, 2020. (SBES '20), p. 844–853. ISBN 9781450387538. Disponível em: https://doi.org/10.1145/3422392.3422497.

LIKERT, R. A technique for the measurement of attitudes. **Archives of Psychology**, v. 140, p. 1–55, 1932.

LINDLEY, J. **A Natural System of Botany, or, A systematic view of the organization, natural affinities, and geographical distribution, of the whole vegetable kingdom: together with the uses of the most important species in medicine, the arts, and rural or domestic economy**. [*S. l.*]: Longman, Rees, Orme, Brown, Green, and Longman, 1836.

MA, M.; XU, J.; WANG, Y.; CHEN, P.; ZHANG, Z.; WANG, P. Automap: Diagnose your microservice-based web applications automatically. In: . New York, NY, USA: Association for Computing Machinery, 2020. ISBN 9781450370233. Disponível em: https://doi.org/10.1145/3366423.3380111.

MARIANI, L.; MONNI, C.; PEZZé, M.; RIGANELLI, O.; XIN, R. Localizing faults in cloud systems. In: **2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)**. [*S. l.*: *s. n.*], 2018. p. 262–273.

MORELL, L. A theory of fault-based testing. **IEEE Transactions on Software Engineering**, v. 16, n. 8, p. 844–857, 1990.

NEWMAN, S. **Building Microservices**. 1st. ed. [*S. l.*]: O'Reilly Media, Inc., 2015. ISBN 1491950358.

PALLEWATTA, S.; KOSTAKOS, V.; BUYYA, R. Microservices-based iot application placement within heterogeneous and resource constrained fog computing environments. In: . New York, NY, USA: Association for Computing Machinery, 2019. ISBN 9781450368940. Disponível em: https://doi.org/10.1145/3344341.3368800.

PINHEIRO, P.; VIANA, J. C.; FERNANDES, L.; RIBEIRO, M.; FERRARI, F.; FONSECA, B.; GHEYI, R. Mutation operators for code annotations. In: **Proceedings of the III Brazilian Symposium on Systematic and Automated Software Testing**. New York, NY, USA: Association for Computing Machinery, 2018. (SAST '18), p. 77–86. ISBN 9781450365550. Disponível em: https://doi.org/10.1145/3266003.3266006.

PONCE, F.; SOLDANI, J.; ASTUDILLO, H.; BROGI, A. Smells and refactorings for microservices security: A multivocal literature review. **Journal of Systems and Software**, v. 192, 2022. Cited by: 14; All Open Access, Green Open Access. Disponível em: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85132709336&doi=10.1016%2fj.jss.2022.111393&partnerID=40&md5=09b2abaf1ac557d5ef96d15d43319ee3.

PRASAD, G. V. S.; CHIMALAKONDA, S.; CHOPPELLA, V.; REDDY, Y. R. An aspect oriented approach for renarrating web content. In: **Proceedings of the 10th Innovations in Software Engineering Conference**. New York, NY, USA: Association for Computing Machinery, 2017. (ISEC '17), p. 56–65. ISBN 9781450348560. Disponível em: https://doi.org/10.1145/3021460.3021466.

SAVOR, T.; DOUGLAS, M.; GENTILI, M.; WILLIAMS, L.; BECK, K.; STUMM, M. Continuous deployment at facebook and oanda. In: **2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)**. [*S. l.*: *s. n.*], 2016. p. 21–30.

SIDDIQUI, H.; KHENDEK, F.; TOEROE, M. Microservices based architectures for iot systems - state-of-the-art review. **Internet of Things (Netherlands)**, v. 23, 2023. Cited by: 1. Disponível em: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85165286125&doi=10.1016%2fj.iot.2023.100854&partnerID=40&md5=f0aa3fe08785a3d87fbdcfc7cc821598.

SILVA, F.; LELLI, V.; SANTOS, I.; ANDRADE, R. Towards a fault taxonomy for microservices-based applications. In: **Proceedings of the XXXVI Brazilian Symposium on Software Engineering**. New York, NY, USA: Association for Computing Machinery, 2022. (SBES '22), p. 247–256. ISBN 9781450397353. Disponível em: https://doi.org/10.1145/3555228.3555245.

SILVA, F. G. S. F.; LELLI, V.; SANTOS, I. d. S.; ANDRADE, R. M. C. Correlations among software testability metrics. In: **Proceedings of the XIX Brazilian Symposium on Software Quality**. New York, NY, USA: Association for Computing Machinery, 2020. (SBQS '20). ISBN 9781450389235. Disponível em: https://doi.org/10.1145/3439961.3439972.

SOMMERVILLE, I. **Engenharia de software**. 9th. ed. [*S. l.*]: PEARSON BRASIL, 2011. ISBN 9788579361081.

WANG, T.; ZHANG, W.; XU, J.; GU, Z. Workflow-aware automatic fault diagnosis for microservice-based applications with statistics. **IEEE Transactions on Network and Service Management**, v. 17, n. 4, p. 2350–2363, 2020.

WANG, T.; ZHANG, W.; YE, C.; WEI, J.; ZHONG, H.; HUANG, T. Fd4c: Automatic fault diagnosis framework for web applications in cloud computing. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, v. 46, n. 1, p. 61–75, 2016.

WEYUKER, E.; GORADIA, T.; SINGH, A. Automatically generating test data from a boolean specification. **IEEE Trans. Software Eng.**, v. 20, p. 353–363, 1994.

WU, N.; ZUO, D.; ZHANG, Z. An extensible fault tolerance testing framework for microservice-based cloud applications. In: . New York, NY, USA: Association for Computing Machinery, 2018. ISBN 9781450365345.

ZHOU, X.; PENG, X.; XIE, T.; SUN, J.; JI, C.; LIU, D.; XIANG, Q.; HE, C. Latent error prediction and fault localization for microservice applications by learning from system trace logs. In: . New York, NY, USA: Association for Computing Machinery, 2019. ISBN 9781450355728.

ZHOU, X.; PENG, X.; XIE, T.; SUN, J.; JI, C.; LI, W.; DING, D. Fault analysis and debugging of microservice systems: Industrial survey, benchmark system, and empirical study. **IEEE Transactions on Software Engineering**, v. 47, n. 2, p. 243–260, 2021.

APÊNDICE A – PROTOCOLO DE REVISÃO

Revisão Multivocal da Literatura - Taxonomia de Faltas para Microserviços

Gutenberg, valerialelli

Planning

Research Questions

1. RQ01. Quais faltas ocorrem em aplicativos baseados em microsserviços?
2. RQ02. Como as faltas são classificadas em aplicativos baseados em microsserviços?
3. RQ03. Quais requisitos não funcionais (RNFs) estão relacionados às faltas identificadas em aplicativos baseados em microsserviços?

Keywords and Synonyms

Microservice	Webservice
Fault	Failure, Defect, Bug, Error, Anomaly
Classification	Taxonomy, Characterization, Categorization, Catalog

Search String

(webservice OR microservice) AND (fault OR failure OR defect OR bug OR error OR anomaly) AND (classification OR taxonomy OR characterization OR categorization OR catalog)

Sources

- ACM Digital Library (http://portal.acm.org)
- Scopus (http://www.scopus.com)

Selection Criteria

Inclusion Criteria:

- C1. Artigos que discutem faltas em microservices
- C2. Artigos que discutem a classificação de faltas em microservices
- C3. Artigos que são escritos em Inglês

Exclusion Criteria:

- CE1. Artigos que não apresentam explicitamente discussões sobre microservices
- CE2. Artigos que são resumos

- CE3 Artigos cujo texto completo não está disponível
- CE4. Artigos que são versões reduzidas de outros

Conducting

Digital Libraries Search Strings

Imported Studies

- **ACM Digital Library:** 844
- **Scopus:** 309

APÊNDICE B – TAXONOMIA PRELIMINAR DE FALTAS

TAXONOMIA PRELIMINAR DE FALTAS

Faltas de Performance

ID	Subcategoria	Falta	Caracteristica de Microserviço
PF01	Temporal Behavior	Infinit Wait	Runtime
PF02		Performance Degradation	Data Storage
PF03		Process Crash Fault	Resilience
PF04		Deadlock Fault	Implementation
PF05		Flag setting bug	Resilience
PF06		Infinit Read Bug	Runtime
PF07		Thread Shutdown Bug	Resilience
PF08		Point Wise	Runtime
PF09		Collective anomalie	Implementation
PF10		Contextual anomalie	Implementation
PF11		Availability Anomaly	Resilience
PF12		Delay Fault	Resilience
PF13		Disconnect Fault	Resilience
PF14		Delete Fault	Resilience
PF15		Network Transmission Delay	Runtime
PF16		Network Transmission Abortion	Runtime
PF17		Power Anomaly	Runtime
PF18		Throughput Anomaly	Runtime
PF19		Latency Anomaly	Runtime
PF20		Long Response Time Fault	Service Discovery
PF21		Remote Service Change Anomaly	Data Storage
PF22		Packet Loss Fault	Resilience
PF23		Deadlock Database Bug	Data Storage
PF24	Resource Utilization	Network Fault	Resilience
PF25		CPU Allocation Fault	Runtime
PF26		Memory Allocation Fault	Runtime
PF27		Memory Usage Fault	Resilience
PF28		Computing Resources Exhaustion	Resilience
PF29		Disk Usage Fault	Resilience
PF30		Resource Usage Fault	Resilience
PF31		CPU Hog	Resilience
PF32		Memory Leak	Resilience
PF33	Capacity	Message Size Fault	Runtime
PF34		Commit Leaks	Runtime
PF35		Increseade Number of User	Runtime
PF36		Monolithic Fault	Interaction Models
PF37		Multi-Instance Fault	Interaction Models
PF38		Configuration Fault	Virtualization
PF39		Asynchronous Interaction Fault	Interaction Models

Faltas de Reliability

ID	Subcategoria	Falta	Caracteristica de Microserviço
RF01		Application Bug	Reliability
RF02		S.O Kernel Bug	Reliability

RF03	Recoverability	Random System Termination Fault	Reliability
RF04		Crash-Recovery Bug	Reliability
RF05		Message Corruption Fault	Reliability
RF06		Permanent Fault	Reliability
RF07		Transient Fault	Reliability
RF08		Intermittent Fault	Reliability
RF09		Invalid Action Fault	Reliability
RF10	Disponibility	Unexpected Fault	Reliability
RF11		Request Expired	Reliability
RF12		Service Unavailable	Reliability
RF13		Throttling Exception	Reliability
RF14		Validation Error	Reliability

Faltas de Maintainability

ID	Subcategoria	Falta	Caracteristica de Microserviço
MF01	Analysability	Invalid User Input Fault	Maintainability
MF02		Missing User Input Fault	Maintainability
MF03		Expired request data Fault	Maintainability
MF04		Invalid request data Fault	Maintainability
MF05		Missing Request data Fault	Maintainability
MF06		Missing User Input Fault	Maintainability
MF07		Expired request data Fault	Maintainability
MF08		Invalid request data Fault	Maintainability
MF09		Missing server data Fault	Maintainability
MF10		Metadata Fault	Maintainability
MF11		Insecure data exposure	Maintainability
MF12		Data Handling Routines Fault	Maintainability
MF13		Incomplete Signature	Maintainability
MF14		Invalid Parameter Combination	Maintainability
MF15		Invalid Parameter Value	Maintainability
MF16		Invalid Query Parameter	Maintainability
MF17		Malformed Query String	Maintainability
MF18		Missing Action	Maintainability
MF19		Missing Parameter	Maintainability

Faltas de Compatibility

ID	Subcategoria	Falta	Caracteristica de Microserviço
CF01	Interoperability	Double processing Fault	Interaction Model
CF02		API Configuration Fault	Implementation
CF03		API Internal Fault	Resilience
CF04		Thirdy Party Fault	Deployment
CF05		Missing Request Data Fault	Interaction Model
CF06		Conversation Error	Interaction Model
CF07		Interaction Fault	Interaction Model
CF08		Environment Fault	Deployment

Faltas de Functionality

ID	Subcategoria	Falta	Caracteristica de Microserviço
FF01	Functional	Functional Fault	Implementation

FF02	Completeness	Internal Fault	Implementation

Faltas de Security

ID	Subcategoria	Falta	Caracteristica de Microserviço
SF01	Authenticity	Missing authentication	Security
SF02		Authentication bypass	Security
SF03		Relying on single FA	Security
SF04		Downgrade authentication	Security
SF05		Message Corruption Fault	Security
SF06		No re-authentication	Security
SF07		Missing authorization	Security
SF08		No context when authorizing	Security
SF09		Not revoking authorization	Security
SF10		Missing Authentication Token	Security
SF11	Integrity	Insecure data storage	Security
SF12		Insecure data exposure	Security
SF13		Not validating input/data	Security
SF14		Data Type Probing	Security
SF15		Data Flushing	Security
SF16		Drop Table	Security
SF17	Confidenciality	Insufficient permissions Fault	Security
SF18		Insufficient crypto key management	Security
SF19		Insufficient session management	Security
SF20		Insufficient credentials management	Security
SF21		Use of custom/weak encryption	Security
SF22		Missing access control	Security
SF23		Insufficient auditing	Security
SF24		Malicious Control	Security
SF25		Malicious Operation	Security
SF26		Scan	Security
SF27		Spying	Security
SF28		Wrong Setup	Security
SF29		Illegal Access Anomaly	Security
SF30		Not Auhtorized	Security
SF31		Access Denied Exception	Security
SF32		Invalid Client TokenId	Security
SF33		OptIn Required	Security
SF34	Disponibility	Uncontrolled resource consumption	Security
SF35		Unmonitored execution	Security

APÊNDICE C – TEMPLATE DO CHECKLIST - GRUPO FOCAL

Compreensão		Sim	Não	Parcial	Obs
Q1	As faltas estão bem descritas?				
Q2	É possível saber como as faltas ocorrem?				
Q3	As faltas estão claras e não ambíguas?				
Q4	Existem faltas duplicadas?				
Q5	O esquema de classificação possui todas as informações necessárias para seu uso?				
Q6	Cada falta da taxonomia foi claramente relacionada com as características de microserviços?				
Q7	Cada falta foi claramente (apresentação da taxonomia) relacionada com as categorias (RNF)?				
Compreensão					
Q8	A definição de cada característica de microsserviço da taxonomia está clara?				
Q9	A definição de cada RNF (Categoria) e subcaracterísticas da taxonomia está clara?				
Q10	O relacionamento entre as faltas e categorias está bem definido? (de forma geral)				
Q11	O relacionamento entre as faltas e as características de microserviços está bem definido? (de forma geral)				
Q12	Todas as faltas de Performance são relacionadas com Performance?				
Q13	Todas as faltas de Reliability são relacionadas com Reliability?				
Q14	Todas as faltas de Functionality são relacionadas com Functionality?				
Q15	Todas as faltas de Maintainability são relacionadas com Maintainability?				
Q16	Todas as faltas de Security são relacionadas com Security?				
Q17	Todas as faltas de Compatibility são relacionadas com Compatibility?				
Completude					
Q18	Todas as faltas descritas na taxonomia são relacionadas à microserviços?				
Q19	Existe alguma falta relacionada à microserviço que não foi apresentada na taxonomia?				
Wiki					
Q20	As seções da wiki estão apresentadas de forma clara?				
Q21	As seções da wiki podem ser reutilizadas de forma fácil?				
Q22	A wiki possui todas as seções necessárias para seu uso?				
Priorização					
Q23	Cite três características de microsserviços que você priorizaria caso fosse construir uma arquitetura de microsserviços.				
Q24	Cite três RNF que você priorizaria caso fosse construir uma arquitetura de microsserviços.				

APÊNDICE D – TEMPLATE DO QUESTIONÁRIO - MÉTODO DELPHI

ID	Questão
Q1	Na sua opinião, a descrição das categorias e subcategorias de faltas é clara e compreensível?
Q2	Existem termos ou conceitos que você considera ambíguos ou difíceis de entender na taxonomia?
Q3	Você acredita que a estrutura da taxonomia facilita a identificação e classificação de faltas em aplicações microservices?
Q4	As categorias e subcategorias incluídas na taxonomia abrangem adequadamente os tipos de faltas que podem ocorrer em aplicações baseadas em microsserviços?
Q5	Existe alguma falta que você acredita que está ausente na taxonomia, mas deveria ser incluída?
Q6	Você acha que esta taxonomia pode ser útil para profissionais que trabalham com aplicações baseadas em microsserviços?
Q7	Você pode fornecer um exemplo de como a taxonomia poderia ser aplicada em um cenário real de desenvolvimento ou teste?
Q8	Você concorda com as definições fornecidas para cada falta na taxonomia?
Q9	Os exemplos dados para ilustrar cada falta são relevantes e úteis em sua compreensão?
Q10	As categorias e subcategorias usadas para classificar as faltas são apropriadas e representam com precisão as características das faltas?
Q11	Você acredita que alguma falta foi classificada incorretamente? Se sim, pode fornecer um exemplo?
Q12	As características de microsserviços definidas na taxonomia são abrangentes e precisas?
Q13	Você considera que algum aspecto importante das características de microsserviços foi omitido?
Q14	Você tem sugestões específicas para melhorar a taxonomia? Isso pode incluir adições, modificações ou esclarecimentos.
Q15	Há algum feedback adicional que você gostaria de compartilhar sobre a taxonomia ou este processo de avaliação?