Toward a Catalog of Observability Anti-Patterns: A Literature Review

Double-blind
Double-blind
Double-blind, Double-blind
Double-blind
Double-blind

Double-blind
Double-blind
Double-blind, Double-blind
Double-blind
Double-blind

Double-blind
Double-blind
Double-blind, Double-blind
Double-blind
Double-blind

ABSTRACT

Modern software systems are increasingly built on microservices architectures to achieve greater scalability, modularity, and support for continuous delivery. While this architectural style streamlines development, it also introduces considerable operational complexity, making system failures more frequent and harder to diagnose. To address these challenges, observability has become a fundamental practice, enabling system introspection through logs, metrics, and traces. However, despite its critical role, observability is often implemented improperly due to the absence of standardized guidelines, leading to ineffective monitoring, alert fatigue, and decreased efficiency in incident response. Although existing research addresses observability tools, concepts, and challenges, no prior work has specifically focused on identifying and analyzing observability anti-patterns. As a consequence, design and implementation decisions are frequently made without a clear understanding of common pitfalls, resulting in suboptimal solutions. This paper presents the findings of a systematic literature review combined with a gray literature review, culminating in a catalog of 37 observability anti-patterns. Each anti-pattern is succinctly described, highlighting how it typically manifests and offering actionable refactoring strategies to mitigate its effects. Our results show that the catalog serves as a valuable resource for helping teams recognize and address recurring observability issues in microservices-based systems.

KEYWORDS

Monitoring, Observability, Microservices, Catalog, Anti-patterns

1 INTRODUÇÃO

Os sistemas de software modernos aumentaram em escala e complexidade, tornando mais difícil a integração de novas funcionalidades. A arquitetura de microsserviços surgiu como uma solução chave, possibilitando a transição de bases de código monolíticas complexas para aplicações mais escaláveis [11, 39]. Embora os microsserviços melhorem o desempenho e reduzam os esforços de desenvolvimento, eles também introduzem novos desafios [9, 37]. À medida que as aplicações nativas da nuvem se tornam mais distribuídas e imprevisíveis, os riscos de falha aumentam, tornando o diagnóstico especialmente difícil em sistemas de grande escala [11]. O monitoramento tradicional carece de visibilidade completa e da capacidade de detectar proativamente anomalias, impulsionando a necessidade de uma abordagem mais avançada: a observabilidade [19].

Observabilidade pode ser definida como a capacidade de inferir o estado de um sistema complexo com base em suas saídas [17]. Também se refere à propriedade de um sistema que pode ser observado e comparado com o estado esperado ao longo do ciclo de vida do desenvolvimento de software [18]. Ao utilizar observabilidade, um sistema se beneficia de maior visibilidade sobre seu fluxo de execução, auxiliando na depuração e na detecção de comportamentos inesperados. Além disso, ajuda a identificar gargalos e ineficiências, contribuindo para a melhoria dos tempos de resposta e para uma utilização mais eficiente dos recursos, entre outras vantagens.

Apesar desses benefícios, muitas equipes acabam adotando anti-padrões ao implementar a observabilidade [7, 19, 24]. Esses anti-padrões envolvem práticas que, embora inicialmente pareçam úteis, acabam comprometendo a eficácia da observabilidade e podem levar a interpretações equivocadas, ineficiências e atrasos na resolução de problemas. Um exemplo comum ocorre quando um sistema de monitoramento gera alertas em excesso, muitas vezes devido a regras redundantes, à ausência de mecanismos de agregação ou a notificações excessivamente granulares. Isso resulta na fadiga de alertas, fazendo com que engenheiros se tornem insensíveis às notificações e eventualmente as ignorem, inclusive aquelas que indicam problemas críticos.

Diversos estudos analisaram soluções de observabilidade. Niedermaier *et al.* (2019) realizaram um estudo baseado na indústria, por meio de entrevistas, para entender os desafios e as melhores práticas no campo da observabilidade e do monitoramento de sistemas distribuídos [30]. Li *et al.* (2022) conduziram uma pesquisa sobre microsserviços e observabilidade, destacando a complexidade dos sistemas industriais baseados em microsserviços operando em infraestruturas em nuvem intrincadas, onde instâncias de serviço são criadas e destruídas dinamicamente [24]. Usman *et al.* (2022) realizaram uma pesquisa abrangente sobre ambientes de TI distribuídos e observabilidade em microsserviços, com o objetivo de explorar desafios, requisitos, boas práticas e soluções atuais para observação de sistemas distribuídos [42]. Kosinska *et al.* (2023) apresentaram um estudo de mapeamento sistemático sobre observabilidade de aplicações nativas da nuvem, mostrando abordagens de engenharia, maturidade, eficiência, ferramentas e direções futuras de pesquisa [19]. Gomes *et al.* (2024) apresentou uma taxonomia de observabilidade para aplicações baseadas em microsserviços, destacando as principais características para a classificações das soluções de observabilidade [13]. Até onde sabemos, nenhum estudo mapeou práticas inadequadas no uso da observabilidade. Para preencher essa lacuna, é necessária uma revisão abrangente dos anti-padrões de observabilidade.

Este estudo examina a literatura existente sobre observabilidade e introduz um catálogo sistemático de anti-padrões de observabilidade. O objetivo é fornecer uma abordagem clara e eficaz para identificar e lidar com práticas prejudiciais associadas à observabilidade. Este catálogo oferece um guia prático para ajudar as equipes a tornarem seus sistemas mais confiáveis e eficientes. A pesquisa

foi conduzida por meio de uma revisão sistemática da literatura, que permitiu mapear os principais estudos e práticas existentes sobre observabilidade e seus anti-padrões. Para complementar essa análise, também foi realizada uma revisão de literatura cinzenta, a fim de enriquecer a compreensão dos desafios enfrentados na implementação da observabilidade. Como resultado, foram identificados 37 anti-padrões de observabilidade, classificados segundo categorias funcionais, abrangendo seis áreas principais: métricas, *logs*, *traces*, alertas, *dashboards* e geral. Eles também foram avaliados quanto à sua relevância com base na literatura. Por fim, os resultados de uma validação com 58 especialistas mostrou que o nosso estudo é 94,82% relevante. As principais contribuições deste trabalho são:

- Uma revisão sistemática da literatura, permitindo mapear os principais estudos e práticas relacionadas à observabilidade e seus anti-padrões;
- Uma revisão de literatura cinzenta, enriquecendo a compreensão sobre os desafios enfrentados na implementação da observabilidade em contextos práticos;
- A documentação das más práticas da observabilidade em formato de um catálogo sistemático contendo 37 anti-padrões que comprometem a efetividade da observabilidade;
- A implementação de um portal de acesso aberto contendo todas as informações sobre os anti-padrões identificados; e
- A validação do catálogo por especialistas na área de observabilidade, garantindo a relevância tanto do estudo quanto dos anti-padrões identificados.

O restante deste artigo está organizado da seguinte forma: A Seção 2 contextualiza sobre observabilidade e anti-padrões. A Seção 3 descreve a metodologia utilizada. O catálogo de anti-padrões de observabilidade é apresentado na Seção 4. A Seção 5 discute a validação do catálogo por profissionais. A Seção 6 apresenta as ameaças à validade do trabalho. Por fim, a Seção 7 conclui o estudo e apresenta sugestões para pesquisas futuras.

2 FUNDAMENTAÇÃO TEÓRICA

Esta Seção apresenta os conceitos, definições e características-chave relacionadas às áreas e subáreas que fundamentam esta pesquisa, as quais são de suma importância para a sua compreensão. Serão abordados os conceitos de Observabilidade e Anti-padrões.

2.1 Observabilidade

2.1.1 ***Introdução.*** Em um ambiente de produção, enfrentar os desafios da operação de microsserviços exige a capacidade de mantê-los funcionando corretamente e identificar rapidamente qualquer falha ou degradação na qualidade do serviço. Tanto provedores de serviços em nuvem quanto usuários precisam monitorar constantemente o uso, o desempenho e a disponibilidade dos recursos [23, 35]. Em situações de falhas ou interrupções, eles dependem dos dados de monitoramento para diagnosticar problemas e, por fim, realizar os reparos necessários. A transição para o paradigma Nativo em Nuvem introduziu maior complexidade operacional, tornando mais desafiador observar e compreender plenamente a integridade, o desempenho e o comportamento das aplicações. Assim, surgiu uma nova estratégia conhecida como Observabilidade [10, 28, 29, 45].

No guia da *Cloud Native Computing Foundation* (CNCF) [8], um dos passos fundamentais é a Observabilidade e Análise [19]. O conceito de observabilidade tem origem na Teoria de Controle [17], que define um sistema como observável se seu estado atual puder ser determinado em um tempo finito apenas por meio de suas saídas. Medir o desempenho geral de um microsserviço é crucial para garantir o cumprimento dos parâmetros de *Quality of Service* (QoS) estabelecidos no Service Level Agreement (SLA). Para alcançar esses parâmetros, o sistema deve expor adequadamente seu estado por meio de técnicas de instrumentação. A observabilidade é frequentemente composta por métricas, *logs* e *traces* e é uma característica essencial em ambientes nativos em nuvem [20], pois seu principal objetivo é tornar a aplicação mais compreensível em vários níveis. Niedermaier et al. (2019) afirmam que, em ambientes em nuvem, a observabilidade indica o grau em que a infraestrutura, as aplicações e suas interações podem ser monitoradas [30].

As motivações para equipar aplicações baseadas em microsserviços com observabilidade derivam da fase de execução dessas aplicações [19]. Elas estão relacionadas ao provisionamento de recursos com foco no rastreamento em tempo real, na medição da sobrecarga do sistema e no escalonamento, incluindo aspectos como escalabilidade. Do ponto de vista da gestão, as motivações mais relevantes incluem a detecção da causa raiz (análise de falhas, previsão de falhas, gerenciamento de falhas, análise de dependências), o gerenciamento de SLA (carga, disponibilidade, QoS e violações) [34], bem como a gestão de ponta a ponta (desempenho, tempo de resposta, análise de latência e rastreamento ponta a ponta), entre outros [36].

2.1.2 ***Diferença entre Monitoramento e Observabilidade.*** Embora o monitoramento tenha desempenhado um papel central na TI ao longo das últimas décadas, sua eficácia tem sido questionada devido a vários fatores, incluindo o surgimento de metodologias ágeis de desenvolvimento, implantações nativas em nuvem e novas práticas DevOps. Esses fatores mudaram a forma como todo o ecossistema de TI - infraestruturas, sistemas e aplicações - precisa ser monitorado para garantir uma resposta rápida a incidentes. A simples aplicação de ferramentas tradicionais de monitoramento já não é suficiente por diversas razões, como a presença de componentes conteinerizados, ambientes mais dinâmicos e ciclos acelerados de implantação de aplicações. A observabilidade complementa o monitoramento ao ajudar a identificar onde e por que uma falha ocorreu e o que desencadeou o problema. No entanto, ela não substitui o monitoramento nem elimina sua necessidade; ambos trabalham em conjunto.

O monitoramento envolve o rastreamento da integridade de um sistema usando um conjunto predefinido de métricas e *logs*, focando na detecção de falhas previamente conhecidas. No entanto, em um sistema distribuído, muitas mudanças são dinâmicas, frequentemente ocorrem em paralelo e podem acontecer de maneira imprevisível. Isso pode levar a problemas que não se encaixam nas categorias previamente definidas pelo monitoramento, escapando da sua detecção. A observabilidade, portanto, complementa o monitoramento ao fornecer uma análise mais profunda sobre as falhas e suas causas.

2.1.3 ***Definições de Observabilidade.*** Ao longo dos anos, vários autores ofereceram definições adicionais do conceito de observabilidade, complementando as ideias apresentadas anteriormente.

De acordo com Usman *et al.* (2022), a observabilidade é um conjunto crescente de práticas combinadas com ferramentas que vão além do mero monitoramento, fornecendo *insights* sobre o estado interno de um sistema por meio da análise de suas saídas externas, predominantemente conhecidas como dados de telemetria, como métricas, *logs* e *traces* [42]. Esta abordagem oferece uma visão geral de alto nível da saúde do sistema, bem como *insights* detalhados sobre os modos de falha subjacentes. Além disso, um sistema com alto nível de observabilidade fornece um contexto abrangente para seu funcionamento interno, permitindo a identificação de falhas mais profundas no sistema.

Para Kratzke e Stage (2022), a observabilidade representa a base para o desenvolvimento de software orientado por dados [21]. Enquanto isso, para Marie *et al.* (2021) [25], a observabilidade é um conceito emergente que pode ser considerado um conjunto mais amplo do que o monitoramento, estendendo seus limites e aplicando técnicas de análise de dados. De certa forma, a observabilidade pode ser percebida como uma extensão necessária do monitoramento para a operação de aplicações nativas de nuvem.

O presente trabalho adota a definição de Usman et al. (2022) por sua abrangência e por enfatizar o uso de saídas externas, em alinhamento com [17].

2.1.4 ***Benefícios da Observabilidade.*** Aplicações baseadas em microsserviços podem melhorar o desempenho quando seus ambientes de execução são equipados com capacidades avançadas de observabilidade [19]. As principais vantagens de um ecossistema de observabilidade incluem:

- **Aumento da Visibilidade**: A observabilidade fornece uma compreensão abrangente das atividades do sistema, incluindo o que acontece, onde acontece, como se desenrola e quando ocorre. Ela permite a identificação de serviços ativos, avaliação de desempenho, localização de componentes e detecção de condições inesperadas;
- **Detecção Precoce de Problemas**: A observabilidade facilita a identificação precoce de problemas, permitindo sua resolução antes que afetem a funcionalidade do sistema. Isso se aplica não apenas a questões relacionadas ao desempenho, mas também a aspectos arquiteturais e de design do sistema;
- **Melhoria dos Processos de *DevOps* e Fluxo de Desenvolvimento**: A observabilidade oferece *insights* valiosos sobre o comportamento do sistema, que podem orientar o processo de desenvolvimento de software. Observações em tempo real do sistema em execução podem ajudar nas decisões de design, potencialmente levando a redesenhos ou ajustes arquiteturais;
- **Escalabilidade Aprimorada**: A observabilidade apoia a escalabilidade, garantindo que subsistemas, especialmente os relacionados ao monitoramento, possam acomodar as crescentes demandas de dados conforme o sistema se expande;
- **Habilitação de Automação**: Os dados de observabilidade podem impulsionar processos de automação, como escalabilidade e *failover* automático, contribuindo para um controle mais autônomo do sistema;
- **Dados Valiosos para Análise**: Os dados coletados por meio da observabilidade podem ser utilizados para análises avançadas, incluindo avaliação estatística e aprendizado de máquina, oferecendo *insights* mais profundos sobre o comportamento da aplicação e seu ambiente de execução;
- **Melhoria na Experiência do Usuário**: Aplicações com observabilidade eficaz tendem a oferecer melhores experiências para o usuário, levando a maior satisfação do cliente e apoiando o alcance de objetivos de negócios.

2.1.5 ***Tipos de Dados de Observabilidade.*** Como mencionado anteriormente, existem três tipos principais de dados frequentemente referidos como os pilares da observabilidade: *Traces*, Métricas e *Logs*.

Traces fornecem uma visão abrangente do fluxo de uma solicitação por meio de uma aplicação [34]. *Traces* são essenciais para entender o caminho completo que uma solicitação percorre na aplicação. Um *trace* distribuído consiste em uma série de eventos gerados em diferentes pontos de um sistema, conectados por um identificador único. Esse identificador é propagado por todos os componentes envolvidos em qualquer operação necessária para concluir a solicitação, permitindo que cada operação associe dados de evento com a solicitação original. Cada *trace* representa uma solicitação única por meio de um sistema, que pode ser tanto síncrona quanto assíncrona. Solicitações síncronas ocorrem sequencialmente, com cada unidade de trabalho concluída antes de prosseguir. Solicitações assíncronas podem iniciar uma série de operações que podem ocorrer simultaneamente e de forma independente. Cada operação registrada em um *trace* é representada por um *span*, uma única unidade de trabalho executada no sistema.

Métricas fornecem informações sobre o estado de um sistema em execução para desenvolvedores e operadores. Os dados coletados por meio de métricas podem ser agregados ao longo do tempo para identificar tendências e padrões nas aplicações, representados graficamente por meio de várias ferramentas e visualizações. O termo métricas tem uma ampla gama de aplicações, pois pode capturar desde métricas de sistema de baixo nível, como ciclos de CPU, até detalhes de nível mais alto, como o número de camisetas azuis vendidas no momento. Além disso, as métricas são essenciais para monitorar a saúde de uma aplicação e decidir quando alertar sobre um possível problema. As métricas são frequentemente úteis por si só, mas quando correlacionadas com outros tipos de dados, como *logs* ou *traces*, podem fornecer informações ainda mais úteis sobre o estado do sistema.

Logs fornecem um registro das atividades que ocorreram e podem ser usados para depurar sistemas e investigar problemas. No contexto da observabilidade, os *logs* frequentemente complementam *traces* e métricas, fornecendo uma visão mais detalhada de eventos específicos que ocorreram em um sistema. Os *logs* são especialmente úteis para investigar problemas em tempo real, pois fornecem uma visão detalhada do que estava acontecendo no sistema em um momento específico. Isso pode ser particularmente útil quando falhas ou problemas de desempenho ocorrem no sistema, pois permite que os desenvolvedores vejam exatamente o que estava acontecendo no sistema no momento do problema.

2.2 Anti-padrões

Como mencionado anteriormente, a observabilidade é um pilar fundamental para o gerenciamento e a manutenção de sistemas modernos, especialmente aqueles distribuídos e que operam em grande

escala. No entanto, a implementação inadequada dessas práticas pode resultar em efeitos adversos, gerando sobrecarga de dados e dificultando a compreensão do comportamento do sistema [43]. Esse cenário de falsa percepção de controle ocorre justamente quando anti-padrões de observabilidade são adotados [27]. Em engenharia de software, um anti-padrão refere-se a uma solução aparentemente eficaz que, com o tempo, revela-se prejudicial, trazendo impactos negativos inesperados [5]. Essas falhas decorrem da busca por soluções imediatas, da falta de experiência ou do desconhecimento sobre as consequências de determinadas escolhas.

Os anti-padrões de observabilidade comprometem a visibilidade do sistema, tornando mais difícil diagnosticar falhas, realizar manutenções e otimizar o desempenho das aplicações [41]. Em vez de aprimorar a compreensão sobre o estado do sistema, essas práticas geram ruído, dificultando a extração de informações relevantes. Assim como ocorre em outras áreas da engenharia de software, sua adoção geralmente resulta da necessidade de respostas rápidas ou da falta de consciência sobre os efeitos adversos dessas decisões. Quando mal implementada, a observabilidade não apenas falha em seu propósito, mas também se torna um obstáculo à gestão eficiente do sistema [22].

No nível operacional, um dos maiores erros que pode ser cometido é a falta de alinhamento entre os alertas e os objetivos de serviço [47]. O sistema pode gerar alertas a todo momento, mas se esses alertas não são baseados em metas claras de desempenho ou indicadores de serviço, eles acabam se tornando um ruído, sem valor real para a equipe. Uma grande quantidade de alertas irrelevantes leva à fadiga dos engenheiros, que acabam ignorando notificações importantes por já estarem acostumados com a constante avalanche de alarmes. O problema se agrava quando os alertas são configurados sem uma análise detalhada do impacto real que uma falha pode ter no serviço. Nesse cenário, alertas podem ser acionados para problemas que, na prática, não afetam a experiência do usuário, enquanto problemas realmente críticos ficam ocultos, sem o devido destaque. Um exemplo prático ocorre quando uma equipe de Site Reliability Engineering (SRE) configura alertas para cada pequena variação na utilização de CPU em servidores individuais, gerando um volume massivo de notificações. Como essas flutuações são normais e nem sempre indicam degradação real do serviço, os engenheiros passam a desconsiderar os avisos. Quando, finalmente, um incidente grave compromete a experiência dos usuários, a falha não é detectada de imediato, pois o alerta crítico se perde em meio a um histórico de notificações irrelevantes. Isso retarda a resposta ao problema e impacta negativamente a confiabilidade do sistema.

3 METODOLOGIA UTILIZADA

Para o desenvolvimento do catálogo de anti-padrões de observabilidade, que foi inspirado em [40], foi realizada uma Revisão Sistemática da Literatura (RSL), com o objetivo de identificar e sintetizar o conhecimento acadêmico disponível sobre o tema. Adicionalmente, foi conduzida uma Revisão de Literatura Cinzenta (RLC), abrangendo documentação técnica, blogs, vídeos e outras fontes não acadêmicas, a fim de capturar percepções práticas e desafios enfrentados pela indústria. A combinação dessas abordagens proporcionou uma perspectiva abrangente e bem fundamentada, equilibrando rigor científico com aplicabilidade prática no contexto da observabilidade. O protocolo de revisão está disponibilizado na Seção Disponibilidade de Artefatos.

3.1 Revisão Sistemática da Literatura

Em relação à RSL, a pesquisa seguiu os seguintes passos: Definição do escopo, Busca, Filtragem, Classificação e Extração de informações. Na fase de definição do escopo, foram formuladas perguntas norteadoras para obter resultados mais precisos sobre o tema em estudo. Ao final, apresentamos os resultados obtidos.

Definição do Escopo: O primeiro passo é dedicado à definição do escopo da RSL, o que inclui a formulação das perguntas de pesquisa que orientam o estudo. Essas perguntas estabelecem os limites da pesquisa, determinando a área a ser mapeada. Em seguida, são utilizadas strings de busca para selecionar artigos relevantes. Vale ressaltar que, antes da formulação da pergunta, houve um consenso entre os autores, com base em estudos anteriores (omitido pelo duplo-anônimo), para analisar tópicos úteis para orientar pesquisas sobre anti-padrões de observabilidade. Assim, este estudo propõe duas **Questões de Pesquisa (QP)**:

- *QP1 - Quais são os anti-padrões de observabilidade mais comuns?* Esta pergunta tem como objetivo identificar os anti-padrões de observabilidade que ocorrem com maior frequência no cotidiano de equipes de desenvolvimento e operações em ambientes microsserviços.
- *QP2 - Como os anti-padrões de observabilidade identificados podem ser mitigados ou evitados por meio de boas práticas?* Esta pergunta busca explorar soluções para mitigar ou prevenir anti-padrões de observabilidade por meio da adoção de boas práticas, e que não é formalizado em observabilidade.

Busca: Nesta etapa, foi aplicada uma string de busca em quatro mecanismos de pesquisa: **IEEE Xplore, ACM, Scopus** e **Web of Science**. Esses mecanismos são reconhecidos pela comunidade acadêmica e científica. O uso de múltiplas fontes garante a consideração da literatura mais abrangente e relevante disponível, cobrindo desde publicações de conferências (IEEE Xplore e ACM) até periódicos científicos e revisões (Scopus e WoS). Foram utilizadas palavras-chave relacionadas ao tema "Anti-padrões de Observabilidade"para gerar a string de busca. Após a análise dos termos, identificou-se que a string deveria conter "Observability". Os termos "Monitoring" e "Tracing" foram utilizados como sinônimos para abranger a ideia geral de observabilidade. Além disso, incluíram-se as palavras "Microservice" e "Micro-service", pois é a principal arquitetura de software quando se fala da utilização de observabilidade. O termo "Anti-pattern" também foi incluído, juntamente com sinônimos como "antipattern, bad smell, pitfall" e "bad practice". Após a busca e a recuperação dos documentos, foi realizado um processo de filtragem, conforme ilustrado na Figura 1. Destaca-se ainda que a string foi aplicada utilizando a busca avançada oferecida pelos mecanismos e ajustada para considerar o texto completo dos artigos.

Filtragem: A estratégia de filtragem também considerou critérios diretos de exclusão: (i) Documento não está escrito em inglês; (ii) Documento possui menos de seis páginas; (iii) Não é um artigo científico; e (iv) Estudo fora do escopo, com base no título. A abordagem adotada verificou se os artigos selecionados se enquadravam

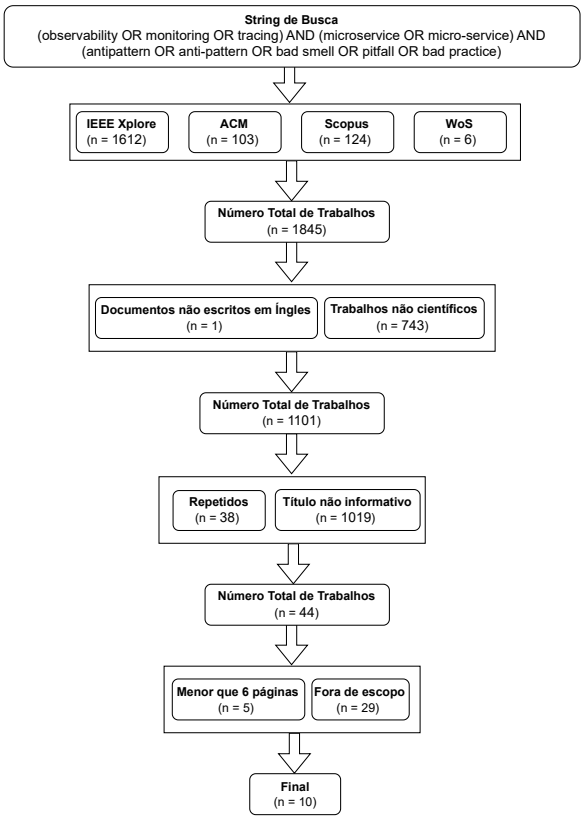

Figura 1: Fluxograma demonstrando as etapas do processo de filtragem.

nos critérios de exclusão. Caso o artigo não atendesse a nenhum dos motivos explícitos de exclusão, ele era automaticamente incluído e avançava para as próximas etapas. A Figura 1 apresenta o fluxograma utilizado para aplicar os critérios de exclusão e ilustra todas as fases de filtragem realizadas. Os artigos analisados foram publicados entre 2019 e 2024. No total, as buscas retornaram 1.845 artigos. O fluxograma mostra o número de artigos excluídos e seus respectivos motivos em cada fase de filtragem.

A primeira etapa de filtragem foi baseada em critérios como "Não é artigo científico" e "Documento não está escrito em inglês", removendo 744 artigos. A segunda etapa eliminou estudos duplicados e aqueles cujo título não era informativo, excluindo 1.057 artigos fora do escopo. A terceira etapa analisou os 44 artigos restantes, aplicando os critérios que removeram estudos com menos de seis páginas ou fora do escopo, de acordo com o texto completo, restando 10 artigos que serviram de base para o catálogo de anti-padrões de observabilidade.

Classificação: Na quarta etapa, focamos na Classificação, na qual analisamos os artigos e identificamos os anti-padrões de observabilidade presentes neles. Esse processo permitiu extrair informações relevantes e agrupar os anti-padrões conforme suas categorias funcionais, contribuindo para o desenvolvimento do catálogo proposto. Inicialmente, definimos as principais categorias. Especificamente, estabelecemos seis categorias: **métricas, *logs*, *traces*, alertas, *dashboards*** e **geral**. As três primeiras refletem os tipos de dados de observabilidade. Uma categoria foi criada para "alertas" e "*dashboards*" porque foram identificados anti-padrões relacionados a alertas gerados por ferramentas, bem como *dashboards* desenvolvidos para exibir os dados. Por fim, a categoria "geral" abrange aspectos relacionados a processos, uso de ferramentas e outros fatores. Adicionalmente, os anti-padrões foram classificados com base no número de artigos que discutem cada um, destacando sua relevância.

Extração de Informações: Na quinta etapa, foi realizada a extração de informações com base na leitura integral dos artigos selecionados, mapeando os dados conforme as categorias previamente definidas. O processo de classificação foi conduzido de forma colaborativa por todos os autores deste trabalho. O primeiro autor, aluno de doutorado, foi responsável pela leitura completa dos estudos, enquanto os coautores, orientadores, acompanharam os achados e ofereceram direcionamentos na construção das classificações. Além disso, orientaram o processo de extração das informações e participaram de discussões para o preenchimento do *template* utilizado na descrição dos anti-padrões de observabilidade (ver Seção 4.1).

3.2 Revisão de Literatura Cinzenta

A RLC foi realizada com o objetivo de identificar informações relevantes que não estavam disponíveis em fontes acadêmicas tradicionais. Ela seguiu os mesmos passos da RSL. Para isso, foram feitas buscas em três plataformas: Google Search[1], StackOverflow[2] e YouTube[3], utilizando os mesmos termos da RSL. No Google, as buscas foram feitas até a décima página de resultados, onde foi encontrado um vídeo relevante sobre o tema. No StackOverflow, não foram encontrados resultados significativos. No YouTube, foram identificados dois vídeos pertinentes. No total, a RLC resultou em 3 vídeos, 9 blogs técnicos e 2 sites oficiais. Além disso, dois artigos sobre observabilidade foram incluídos, ambos sendo pesquisas industriais. Esses resultados complementam as informações obtidas na RSL, oferecendo uma visão mais ampla sobre os anti-padrões de observabilidade.

3.3 Fontes de Anti-padrões de Observabilidade

A Tabela 1 apresenta a distribuição das fontes de anti-padrões de observabilidade, considerando a RSL e a RLC. Essa tabela é dividida de acordo com o tipo de fonte (Artigo Científico, Vídeo, Blog Técnico e Site Oficial), título da fonte, ano e referência.

[1]https://www.google.com/
[2]https://stackoverflow.com/
[3]https://www.youtube.com/

Tabela 1: Fontes dos anti-padrões de observabilidade.

Tipo	Título	Ano	Ref
Artigo Científico	On Observability and Monitoring of Distributed Systems – An Industry Interview Study	2019	[30]
	On the Study of Microservices Antipatterns: a Catalog Proposal	2020	[40]
	A Survey of Software Log Instrumentation	2021	[6]
	A Survey on Automated Log Analysis for Reliability Engineering	2021	[16]
	Enjoy your observability: an industrial survey of microservice tracing and analysis	2021	[24]
	A Survey on Observability of Distributed Edge & Container-Based Microservices	2022	[42]
	Visualizing Microservice Architecture in the Dynamic Perspective: A Systematic Mapping Study	2022	[14]
	Characterizing and Mitigating Anti-patterns of Alerts in Industrial Cloud Systems	2022	[47]
	Logging Practices in Software Engineering: A Systematic Mapping Study	2022	[15]
	Serverless: From Bad Practices to Good Solutions	2022	[38]
	Catalog and detection techniques of microservice anti-patterns and bad smells: A tertiary study	2023	[5]
	Toward the Observability of Cloud-Native Applications: The Overview of the State-of-the-Art	2023	[19]
Vídeo	10 tips to BAD Observability by Sameer Mhaisekar	2024	[26]
	Bad API observability: 5 anti-patterns in 5 minutes	2024	[3]
	AWS: Your Ally Against Observability Anti-Patterns | Indika Wimalasuriya | Conf42 Observability 2024	2024	[46]
Blog Técnico	Observability Mythbusters: Observability Anti-Patterns	2022	[43]
	Mistakes to avoid in Observability	2022	[27]
	Bad Observability	2023	[41]
	Everything You Know About Observability is Wrong	2023	[22]
	OpenTelemetry Collector Anti-Patterns	2024	[44]
	How to Resolve Bad Observability Data Quality	2024	[32]
	How to Solve 3 Data Flow Issues in Observability	2024	[33]
	Top 10 Mistakes People Make When Building Observability Dashboards	2024	[31]
	7 API Observability Anti-Patterns to Avoid	2024	[7]
Site Oficial	Anti-patterns for continuous monitoring	2025	[1]
	AWS Observability Best Practices	2025	[2]

4 UM CATÁLOGO DE ANTI-PADRÕES DE OBSERVABILIDADE

Esta Seção apresenta o catálogo de anti-padrões de observabilidade desenvolvido com base na metodologia proposta, evidenciando as classificações realizadas. A Tabela 2 lista os 37 anti-padrões identificados, acompanhados de suas respectivas categorias e classificações de relevância, bem como um resumo de suas descrições e exemplos. Informações complementares, incluindo as fontes utilizadas, a lista completa de anti-padrões com suas respectivas categorias e relevâncias, encontram-se disponíveis na Seção Disponibilidade de Artefatos.

4.1 Template de Documentação dos Anti-padrões de Observabilidade

O modelo adotado para a documentação dos anti-padrões de observabilidade foi adaptado a partir do template disponibilizado na C2 Wiki [4]. Tal estrutura propicia uma abordagem sistemática para a descrição dos anti-padrões, promovendo a uniformização da apresentação e facilitando a compreensão e mitigação dos problemas relacionados à observabilidade em arquiteturas baseadas em microsserviços. O template contempla os seguintes campos essenciais:

- **Nome**: Denominação concisa e descritiva do anti-padrão.
- **Categoria**: Classificação do anti-padrão com base em categorias predefinidas (métricas, *logs*, *traces*, alertas, *dashboards* ou geral).
- **Contexto**: Descrição do cenário em que o anti-padrão ocorre, incluindo as condições típicas que favorecem seu surgimento.
- **Problema**: Principais implicações do anti-padrão, com destaque para seus impactos sobre a observabilidade.
- **Exemplo**: Ilustração prática, real ou hipotética, que evidencia a manifestação do anti-padrão.
- **Solução Recomendada**: Conjunto de boas práticas e diretrizes para a mitigação ou prevenção do anti-padrão.
- **Consequências**:
  - **Positivas**: Benefícios decorrentes da aplicação das soluções recomendadas.
  - **Negativas**: Possíveis dificuldades na adoção das soluções recomendadas, como o aumento da complexidade ou a elevação de custos operacionais.

4.2 Classificação de Relevância

Após a consolidação dos anti-padrões de observabilidade a partir das fontes levantadas na revisão de literatura, procedeu-se à análise do número de citações de cada anti-padrão com o intuito de aferir sua relevância. Na sequência, foi aplicado um processo de ranqueamento com base em quartis, assegurando uma classificação estatisticamente fundamentada que refletisse a distribuição das frequências de citação. Os valores de corte foram estabelecidos da seguinte forma: primeiro quartil (Q1) com 2 citações, segundo quartil (Q2) com 3 citações, terceiro quartil (Q3) com 4 citações e quarto quartil (Q4) com 5 ou mais citações. Com base nessa segmentação, a classificação de relevância foi definida da seguinte forma: (i) Baixa relevância: 1 ou 2 citações; (ii) Relevância média: 3 ou 4 citações; e (iii) Alta relevância: de 5 a 8 citações.

4.3 Portal do Catálogo

Foi implementado um portal do catálogo, contendo informações completas sobre todos os anti-padrões, disponível no seguinte endereço: https://observability-antipatterns.github.io/. A Figura 2 apresenta uma parte desse portal. Ele reúne todos os anti-padrões identificados, em formato de grade, e suas respectivas descrições, estruturadas conforme o template previamente apresentado. No portal, são oferecidos filtros tanto por categoria quanto por relevância (adicionados posteriormente à avaliação com especialistas). Além disso, oferece um campo de busca que auxilia na identificação de soluções relacionadas a problemas específicos. Esse mecanismo de busca verifica o termo digitado em todos os campos do template dos anti-padrões, com o objetivo de proporcionar maior precisão e efetividade na recuperação da informação por parte do usuário.

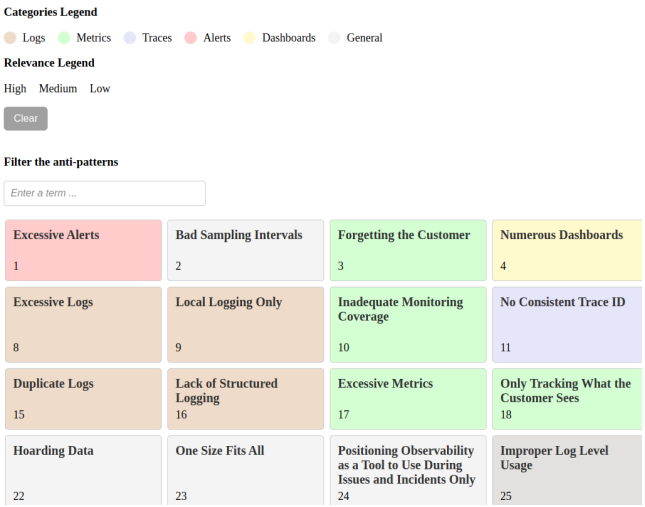

Figura 2: Parte do Portal do Catálogo

4.4 Exemplo: Alertas Excessivos

Em virtude das limitações de espaço, optou-se por apresentar, como exemplo, um anti-padrão classificado como alta relevância, citado em oito fontes distintas, abrangendo artigos científicos, vídeos, blogs técnicos e sites oficiais. O anti-padrão selecionado é denominado Alertas Excessivos, o qual se revela particularmente problemático em ambientes baseados em microsserviços, nos quais a complexidade e o elevado volume de métricas tendem a gerar uma sobrecarga de alertas, dificultando a triagem eficiente e comprometendo a eficácia dos mecanismos de monitoramento.

- **Nome**: Alertas Excessivos
- **Categoria**: Alertas
- **Contexto**: Alertas excessivos ocorrem quando um sistema de monitoramento gera um volume elevado de notificações, dificultando a distinção entre problemas críticos e eventos menores. Esse problema geralmente resulta de regras redundantes, granularidade excessiva ou falta de mecanismos de agregação. Como consequência, os engenheiros podem se tornar insensíveis às notificações, incluindo aquelas que indicam falhas significativas. A ausência de políticas bem definidas para filtragem e priorização de alertas agrava esse anti-padrão.
- **Problema**: O excesso de alertas pode causar fadiga de alerta, levando os engenheiros a desconsiderarem notificações devido ao elevado volume. Isso resulta no aumento do tempo médio de resolução, pois alertas críticos podem se perder no excesso de notificações. Além disso, há desperdício de tempo operacional com alertas de baixa prioridade ou redundantes, o que desvia a atenção de incidentes realmente críticos. Como resultado, falhas significativas podem passar despercebidas, comprometendo a confiabilidade do sistema e impactando a experiência dos usuários finais.
- **Exemplo**: Uma equipe de SRE configura alertas para cada pequena variação no tempo de resposta, resultando em centenas de notificações diárias. Com o tempo, os engenheiros começam a ignorar essas notificações devido ao alto volume. Quando um problema real ocorre, causando uma degradação significativa no desempenho do sistema, ele passa despercebido até que a falha se agrave, afetando milhares de usuários antes que qualquer ação corretiva seja tomada.
- **Soluções Recomendadas**:
  - Implementar mecanismos de deduplicação, ou seja, eliminar alertas repetidos ou duplicados, mesmo que o evento seja detectado mais de uma vez em um curto período, para reduzir a redundância de alertas;
  - Implementar mecanismos de agregação, que consiste em um processo de agrupar múltiplos alertas relacionados em um único alerta mais geral, para reduzir a redundância de alertas. Isso é útil quando um conjunto de alertas está relacionado a uma mesma causa ou problema. Em vez de gerar vários alertas isolados para cada métrica ou falha, um alerta agregado pode fornecer uma visão consolidada do problema, o que torna a resposta mais eficiente;
  - Definir níveis claros de priorização, classificando alertas como informativos, de aviso e críticos, e consequentemente, estabelecendo uma hierarquia que categorize os alertas com base na gravidade e no impacto que eles têm no sistema ou no serviço monitorado. Isso ajuda as equipes a se concentrarem nas notificações mais importantes e a responder de maneira mais eficaz a incidentes. Como exemplo, imagine que você tenha um sistema de monitoramento que verifica a disponibilidade de servidores e bancos de dados. Se a utilização de CPU de um servidor atingir 85%, o alerta seria classificado como "Aviso". Caso o servidor fique offline, seria classificado como "Crítico". Por fim, se uma tarefa de rotina for executada com sucesso, o alerta seria "Informativo". Essa classificação ajuda a equipe a entender rapidamente a urgência de cada situação e a priorizar suas ações;
  - Utilizar técnicas de alerta dinâmico, que se baseiam em comportamentos adaptáveis, ajustados às condições reais e variáveis do sistema ao longo do tempo, em vez de valores fixos ou predefinidos. Isso torna os alertas mais flexíveis e relevantes, respondendo de forma precisa às flutuações do sistema e evitando a geração de alertas desnecessários (falsos positivos). Esses alertas podem incorporar a

definição de *baseline* (referência de comportamento esperado) e a detecção de anomalias para identificar desvios significativos;
  - A equipe de monitoramento deve revisar regularmente as regras de alerta para garantir que estão alinhadas com as necessidades atuais do sistema e da equipe. Isso envolve remover ou ajustar alertas que já não fazem sentido devido a mudanças no sistema ou na infraestrutura; e
  - Implementar limitação de taxa para evitar notificações excessivas em curtos períodos de tempo. A limitação de taxa tem como objetivo controlar a frequência de notificações geradas por alertas em um curto intervalo de tempo, evitando que múltiplos alertas sobre o mesmo problema sejam enviados para a equipe de operações. Como exemplo, suponha que um sistema de microsserviços esteja gerando alertas a cada segundo devido a falhas temporárias de rede. Para evitar que a equipe seja inundada com notificações, a limitação de taxa pode ser configurada para enviar apenas uma notificação a cada 30 segundos, mesmo que o problema continue ocorrendo nesse intervalo. Isso garante que a equipe não receba um volume excessivo de alertas repetitivos e possa focar melhor na resolução de problemas importantes.
- **Consequências**:
  - **Positivas**:
    - **Redução da fadiga de alerta**: A implementação dessas soluções resulta em uma diminuição significativa da sobrecarga de notificações, permitindo que os engenheiros se concentrem nos alertas que realmente necessitam de atenção e ação;
    - **Maior eficiência operacional**: Com um número reduzido de alertas irrelevantes, as equipes conseguem otimizar o tempo, priorizando problemas mais críticos e realizando ações corretivas de forma mais rápida e eficaz; e
    - **Maior confiabilidade do sistema de monitoramento**: Ao melhorar a qualidade e a relevância das notificações, o sistema de monitoramento se torna mais confiável, garantindo que as alertas importantes sejam detectados e tratados de forma adequada, aumentando a confiabilidade do sistema como um todo.
  - **Negativas**:
    - **Investimento inicial significativo**: A configuração de mecanismos de deduplicação, agregação e limitação de taxa de alertas pode demandar um esforço inicial considerável, incluindo a adaptação de ferramentas e a definição de novas regras de alerta;
    - **Necessidade de ferramentas avançadas de monitoramento**: Implementar essas estratégias pode exigir o uso de ferramentas mais sofisticadas, com recursos de correlação inteligente de eventos e análise dinâmica, o que pode representar um desafio em termos de custo e integração; e
    - **Necessidade de ajustes contínuos**: O sistema de alertas precisará de manutenção regular e ajustes contínuos para garantir que a configuração permaneça eficaz, ajustando-se a mudanças no sistema e nos padrões de comportamento, o que pode gerar um trabalho contínuo de monitoramento e revisão.

4.5 Resultados Gerais

De acordo com o estudo, anti-padrões de observabilidade foram identificados em diversas fontes, tanto acadêmicas quanto da indústria. Os anti-padrões mais frequentemente citados foram Alertas Excessivos, Intervalos de Amostragem Inadequados e Esquecer o Cliente, cada um com oito citações, o que os posiciona como os mais relevantes. Além disso, destaca-se o anti-padrão *Dashboards* em Excesso, citado exclusivamente por fontes não acadêmicas, totalizando sete citações. Esse dado evidencia sua relevância prática e a necessidade de uma atenção especial por parte dos usuários. A categoria com o maior número de anti-padrões foi a categoria Geral, que representa 37,83% (14 de 37) do total identificado. As categorias *Logs* e Métricas apareceram em seguida, cada uma com 21,62% (8 de 37) do total.

Por fim, tais resultados contribuem para a resposta à **QP1**, ao demonstrar que o catálogo mapeou os anti-padrões de observabilidade mais recorrentes, os quais demandam maior atenção por parte dos usuários. Ademais, a **QP2** é também contemplada por meio do catálogo, uma vez que foram apresentadas soluções voltadas à mitigação ou prevenção dos anti-padrões de observabilidade, por meio da aplicação de boas práticas, promovendo, assim, uma observabilidade mais eficaz e confiável.

5 AVALIAÇÃO COM ESPECIALISTAS

Com o intuito de avaliar o catálogo, elaboramos um formulário de avaliação baseado na metodologia de Duarte et al. [12]. O formulário é composto por uma seção introdutória, na qual apresentamos os autores da pesquisa, os objetivos do formulário e do catálogo, a descrição das seções e das questões, bem como a definição do público-alvo e o tempo estimado para responder. Também disponibilizamos aos avaliadores o link de acesso ao portal contendo o catálogo. Nosso formulário avaliou qualitativamente o catálogo em termos de clareza, legibilidade, relevância, utilidade e avaliação geral. Ele foi estruturado em duas partes: a primeira voltada à autoavaliação dos participantes e a segunda com questões qualitativas objetivas sobre o catálogo. As respostas a essas questões seguem a escala de Likert (concordo totalmente, concordo, neutro, discordo e discordo totalmente). Se caso o participante escolher "discordo" ou "discordo totalmente", era pedido para ele justificar o porque da escolha. Além disso, uma pergunta aberta optativa foi fornecida para os participantes, no sentido de sugerir melhorias para o catálogo ou realizar alguma observação.

Distribuímos o formulário por meio do Google Forms, selecionando os participantes tanto por contatos pessoais dos autores quanto em grupos especializados em observabilidade e redes de profissionais da área. A seleção via contatos pessoais foi direcionada a profissionais com experiência prática com observabilidade. Adicionalmente, o primeiro autor realizou contatos diretos a profissionais da área via LinkedIn. Essa abordagem incluiu reuniões por videoconferência para apresentar o estudo e solicitar apoio na disseminação do formulário. Posteriormente, ainda pelo LinkedIn, outros profissionais foram abordados, especialmente SREs e DevOps, todos com vivência prática em observabilidade. Como nosso

Tabela 2: Lista contendo os anti-padrões de observabilidade do catálogo.

ID	Nome	Categoria	Relevância	Fonte
1	Alertas excessivos	Alerta	Alta	[47], [46], [27], [41], [22], [31], [1], [2]
2	Intervalos de amostragem inadequados	Geral	Alta	[6], [16], [24], [26], [46], [27], [41], [32]
3	Esquecer o cliente	Métrica	Alta	[30], [26], [3], [46], [27], [41], [7], [2]
4	Dashboards em excesso	Dashboard	Alta	[26], [46], [43], [27], [41], [22], [31]
5	Falta de informações contextuais	Geral	Alta	[30], [42], [46], [22], [31], [2]
6	Não entender seu ecossistema	Geral	Alta	[30], [42], [19], [46], [43], [41]
7	Uso excessivo de ferramentas	Geral	Alta	[26], [46], [43], [27], [41], [7]
8	Logs excessivos	Log	Alta	[6], [16], [15], [26], [46], [2]
9	Log apenas local	Log	Média	[40], [14], [5], [2]
10	Cobertura de monitoramento inadequada	Métrica	Média	[40], [5], [44], [1]
11	Ausência de ID de rastreamento consistente	Trace	Média	[14], [26], [46], [41]
12	Falta de diretrizes de instrumentação para novos serviços	Geral	Média	[46], [43], [27]
13	Negligência na manutenção regular	Geral	Média	[19], [33], [31]
14	Observabilidade apenas para produção	Geral	Média	[30], [3], [7]
15	Logs duplicados	Log	Média	[15], [6], [16],
16	Falta de logging estruturado	Log	Média	[6], [46], [2]
17	Métricas excessivas	Métrica	Média	[38], [26], [22]
18	Monitorar apenas o que o cliente vê	Métrica	Média	[27], [33], [2]
19	Subestimar a importância dos traces	Trace	Média	[46], [43], [27]
20	Ter apenas dashboards baseados em gráficos de séries temporais	Dashboard	Baixa	[14], [27]
21	Inconsistência entre ambientes	Geral	Baixa	[46], [41]
22	Acumular dados desnecessariamente	Geral	Baixa	[27], [41]
23	Tentar aplicar uma abordagem única para todos os casos	Geral	Baixa	[3], [7]
24	Tratar a observabilidade como uma ferramenta apenas para incidentes	Geral	Baixa	[27], [31]
25	Uso inadequado de níveis de log	Log	Baixa	[15], [2]
26	Usar logs de acesso à API para depuração	Log	Baixa	[3], [7]
27	Má compreensão das métricas	Métrica	Baixa	[26], [41]
28	Ignorar métricas derivadas	Métrica	Baixa	[1], [2]
29	Começar o trace no API Gateway é superestimado	Trace	Baixa	[3], [7]
30	Transações sintéticas preguiçosas	Geral	Baixa	[41]
31	Imposição de ferramentas	Geral	Baixa	[41]
32	Negligência na retenção de dados	Geral	Baixa	[31]
33	Falha ao monitorar logs de erro	Log	Baixa	[33]
34	Uso inadequado de logs para coleta de métricas	Log	Baixa	[2]
35	Confiar apenas no monitoramento de APIs	Métrica	Baixa	[7]
36	A métrica burra e gigante	Métrica	Baixa	[41]
37	Spans de trace muito longos	Trace	Baixa	[46]

objetivo era contar apenas com especialistas na avaliação, desconsideramos as respostas de participantes com menos de um ano de experiência na área (4 participantes). Ao todo, 58 avaliadores com mais de um ano de experiência em observabilidade participaram da avaliação.

5.1 Perfil dos Avaliadores

A primeira parte do questionário teve como objetivo analisar o perfil dos avaliadores. Assim, 24,13% dos avaliadores são SRE, 25,86% são DevOps, 27,58% Desenvolvimento de Software, o restante varia entre Gestão de TI, Infraestrutura, Pesquisa acadêmica, Qualidade de Software, Segurança da Informação e Arquitetura de software. Além disso, pelo menos 80% dos avaliadores atuam no mercado (colocamos esse campo como opcional), em empresas nacionais e internacionais, que atuam nos mais variados ramos, desde telecomunicação, serviços financeiros, desenvolvimento de software, recursos humanos, entre outros. Com relação à experiência com observabilidade, 37,93% dos avaliadores afirmaram ter entre 1 e 3 anos de experiência. Cerca de 29,31% indicaram ter entre 3 e 5 anos de atuação na área. Por fim, 32,75% afirmaram possuir mais de 5 anos de experiência trabalhando com observabilidade.

5.2 Resultados Obtidos

A Figura 3 resume os resultados das perguntas qualitativas em escala de Likert respondidas pelos participantes. Somando-se as respostas positivas (por exemplo, "concordo" e "concordo totalmente"), obteve-se uma taxa de aprovação de 93,1% para a avaliação geral, 94,82% para relevância, 96,55% para utilidade, 84,48% para legibilidade e 82,75% para clareza. Tanto para legibilidade quanto para clareza, houve críticas negativas (respostas "discordo") de dois participantes, os quais sugeriram tornar o portal mais intuitivo para facilitar o acesso às informações. Essa sugestão foi incorporada com a adição de novos filtros, tanto por categoria quanto por relevância, visando otimizar a navegação. Os resultados obtidos evidenciam uma forte aceitação por parte dos participantes em relação ao catálogo de anti-padrões de observabilidade. As altas taxas

de aprovação, especialmente nos critérios de utilidade e relevância, indicam que o conteúdo atende às expectativas do público-alvo e possui aplicabilidade prática. A avaliação geral reforça essa percepção positiva, e as críticas pontuais demonstram um engajamento construtivo por parte dos usuários.

Com relação à pergunta aberta, alguns participantes deixaram comentários adicionais que reforçam os resultados. Um deles afirmou que o catálogo possui um excelente conteúdo de pesquisa, sendo relevante, abrangente e com alto nível de qualidade, detalhamento e profundidade. Outro destacou que o material está muito bom e aborda diversas dores enfrentadas diariamente por profissionais de SRE. Também foi ressaltado que, para quem deseja iniciar na área de monitoramento de aplicações, é essencial conhecer os possíveis erros desde o início do desenvolvimento, a fim de preveni-los. Um participante comentou: "o tema é superinteressante, frequentemente me deparo com situações de *logs* sem informações úteis sobre a realidade da exceção, ou até mesmo com a ausência completa desses registros. Entendo completamente a importância do estudo". Por fim, houve quem enxergasse o catálogo como uma verdadeira "bússola" para empresas em diferentes níveis de maturidade na adoção da observabilidade, o que evidencia o potencial do material como referência prática e estratégica para o setor. Os comentários reforçam o valor do catálogo, destacando sua profundidade, aplicabilidade no cotidiano de profissionais da área e importância para iniciantes no campo da observabilidade.

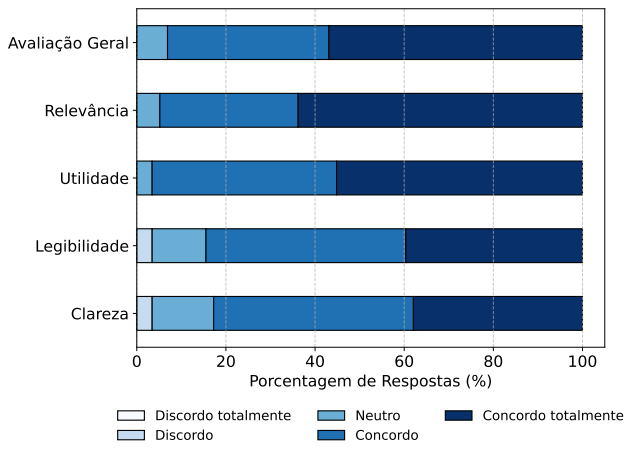

Figura 3: Avaliação do Catálogo

6 AMEAÇAS À VALIDADE

Este trabalho apresenta algumas ameaças à sua validade, baseado em [13], relacionadas principalmente a: (i) identificação das fontes, (ii) extração dos dados relevantes, (iii) síntese dessas informações e (iv) avaliação dos especialistas.

Com relação à primeira categoria de ameaças, é possível que algumas fontes relevantes não tenham sido identificadas durante o processo de seleção. No entanto, para mitigar esse risco, foram conduzidas uma RSL e uma RLC, utilizando um conjunto diversificado de termos de busca. No caso da RSL, a busca foi realizada em bases de dados acadêmicas consolidadas e amplamente utilizadas na área de computação. Para a RLC, exploraram-se diversas fontes, incluindo blogs técnicos, sites oficiais e conteúdos em vídeo, assegurando uma cobertura mais ampla de perspectivas práticas. Ademais, foi definido um protocolo de revisão com o intuito de mitigar vieses durante o processo de seleção.

Quanto à extração dos dados, as informações recuperadas tanto da RSL quanto da RLC foram estruturadas com base nas questões de pesquisa estabelecidas no protocolo de revisão. Em situações de dúvida ou ambiguidade, os revisores discutiram entre si até alcançar um consenso quanto às informações extraídas.

Quanto a ameaça associada à síntese dos dados foi mitigada por meio da utilização de categorias para agrupar os trabalhos identificados. Além disso, estatísticas descritivas foram empregadas para sumarizar os resultados da revisão da literatura.

Por fim, avaliação dos especialistas, apesar da quantidade limitada de participantes, buscou-se incluir participantes com experiência prévia em observabilidade, desconsiderando com menos de 1 ano de experiência, de modo a garantir a relevância e a qualidade das contribuições fornecidas.

7 CONCLUSÃO E TRABALHOS FUTUROS

A mensuração do desempenho geral de sistemas baseados em microsserviços é essencial para garantir a escalabilidade e a resiliência dessas aplicações. A observabilidade desempenha um papel fundamental nesse contexto, ao fornecer uma visão clara do estado do sistema por meio de *logs*, métricas e *traces*. Este trabalho apresentou a importância da observabilidade em ambientes de microsserviços, destacando práticas inadequadas e anti-padrões que podem comprometer sua eficácia. A partir de uma RSL e de uma RLC, foi desenvolvido um catálogo contendo 37 anti-padrões de observabilidade, com o objetivo de auxiliar na identificação e na resolução desses problemas. Ademais, o estudo foi validado por profissionais, o que garante a relevância do catálogo. Como trabalhos futuros, estão previstas as seguintes ações: (i) realização de uma nova rodada de revisão do catálogo com um número maior de especialistas, visando aprimorar a clareza e a completude da documentação; (ii) testes em cenários reais, aplicando o catálogo em uma aplicação industrial para identificar e solucionar problemas, avaliando as melhorias proporcionadas, como ganho de performance e maior facilidade na detecção de problemas, além de avaliar sua eficiência em termos de tempo de implementação; e (iii) a condução de um teste de usabilidade do portal do catálogo, com o objetivo de avaliar a facilidade de navegação e a efetividade na descoberta dos anti-padrões por parte dos usuários.

DISPONIBILIDADE DE ARTEFATOS

A planilha contendo os dados referentes ao estudo realizado pode ser acessada em https://anonymous.4open.science/r/sbes2025-0137.

AGRADECIMENTOS

Omitido devido à revisão duplo-anônimo.

REFERÊNCIAS

[1] AWS. 2025. Anti-patterns for continuous monitoring. (2025). https://docs.aws.amazon.com/wellarchitected/latest/devops-guidance/anti-patterns-for-continuous-monitoring.html Accessed: March 10, 2025.

[2] AWS. 2025. AWS Observability Best Practices. (2025). https://aws-observability.github.io/observability-best-practices/signals/logs Accessed: March 10, 2025.
[3] Budhaditya Bhattacharya. 2024. Bad API observability: 5 anti-patterns in 5 minutes. (2024). https://devopsdays.org/events/2024-houston/program/budhaditya-bhattacharya-ignite/ Accessed: March 10, 2025.
[4] C2 Wiki. 2024. Anti-Pattern Template. https://wiki.c2.com/?AntiPatternTemplate Accessed: March 9, 2025.
[5] Tomas Cerny, Amr S Abdelfattah, Abdullah Al Maruf, Andrea Janes, and Davide Taibi. 2023. Catalog and detection techniques of microservice anti-patterns and bad smells: A tertiary study. *Journal of Systems and Software* 206 (2023), 111829.
[6] Boyuan Chen and Zhen Ming Jiang. 2021. A survey of software log instrumentation. *ACM Computing Surveys (CSUR)* 54, 4 (2021), 1–34.
[7] S. Chevre. 2024. 7 API Observability Anti-Patterns to Avoid. (2024). https://devops.com/7-api-observability-anti-patterns-to-avoid/ Accessed: March 10, 2025.
[8] Shuiguang Deng, Hailiang Zhao, Binbin Huang, Cheng Zhang, Feiyi Chen, Yinuo Deng, Jianwei Yin, Schahram Dustdar, and Albert Y Zomaya. 2023. Cloud-Native Computing: A Survey from the Perspective of Services. *arXiv preprint arXiv:2306.14402* (2023).
[9] Paolo Di Francesco, Ivano Malavolta, and Patricia Lago. 2017. Research on architecting microservices: Trends, focus, and potential for industrial adoption. In *2017 IEEE International conference on software architecture (ICSA)*. IEEE, 21–30.
[10] António Pedro dos Santos Carvalho. 2022. *Observabilidade e telemetria em arquiteturas de micro-serviços*. Master's thesis. Universidade do Porto (Portugal).
[11] Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin, and Larisa Safina. 2017. Microservices: yesterday, today, and tomorrow. *Present and ulterior software engineering* (2017), 195–216.
[12] Paulo Duarte, Rainara Carvalho, and Windson Viana. 2024. A Catalog of Interoperability Solutions for Ambient Assisted Living. In *Anais do XVIII Simpósio Brasileiro de Componentes, Arquiteturas e Reutilização de Software* (Curitiba/PR). SBC, Porto Alegre, RS, Brasil, 61–70. https://doi.org/10.5753/sbcars.2024.3859
[13] Francisco Gomes, Paulo Rego, and Fernando Trinta. 2024. Rumo a uma Taxonomia de Observabilidade para Aplicações Baseadas em Microsserviços. In *Anais do XXXVIII Simpósio Brasileiro de Engenharia de Software* (Curitiba/PR). SBC, Porto Alegre, RS, Brasil, 234–245. https://doi.org/10.5753/sbes.2024.3386
[14] Mia E Gortney, Patrick E Harris, Tomas Cerny, Abdullah Al Maruf, Miroslav Bures, Davide Taibi, and Pavel Tisnovsky. 2022. Visualizing microservice architecture in the dynamic perspective: A systematic mapping study. *IEEE Access* 10 (2022), 119999–120012.
[15] Shenghui Gu, Guoping Rong, He Zhang, and Haifeng Shen. 2022. Logging practices in software engineering: A systematic mapping study. *IEEE transactions on software engineering* 49, 2 (2022), 902–923.
[16] Shilin He, Pinjia He, Zhuangbin Chen, Tianyi Yang, Yuxin Su, and Michael R Lyu. 2021. A survey on automated log analysis for reliability engineering. *ACM computing surveys (CSUR)* 54, 6 (2021), 1–37.
[17] Rudolf E Kalman. 1960. On the general theory of control systems. In *Proceedings First International Conference on Automatic Control, Moscow, USSR*. 481–492.
[18] Suman Karumuri, Franco Solleza, Stan Zdonik, and Nesime Tatbul. 2021. Towards observability data management at scale. *ACM SIGMOD Record* 49, 4 (2021), 18–23.
[19] Joanna Kosińska, Bartosz Baliś, Marek Konieczny, Maciej Malawski, and Sławomir Zieliński. 2023. Towards the Observability of Cloud-native applications: The Overview of the State-of-the-Art. *IEEE Access* (2023).
[20] Joanna Kosińska and Krzysztof Zieliński. 2020. Autonomic management framework for cloud-native applications. *Journal of Grid Computing* 18 (2020), 779–796.
[21] Nane Kratzke. 2022. Cloud-native observability: the many-faceted benefits of structured and unified logging—a multi-case study. *Future Internet* 14, 10 (2022), 274.
[22] T. LaRock. 2023. Everything You Know About Observability is Wrong. (2023). https://www.selector.ai/blog/everything-you-know-about-observability-is-wrong/ Accessed: March 10, 2025.
[23] Joshua Levin and Theophilus A Benson. 2020. ViperProbe: Rethinking microservice observability with eBPF. In *2020 IEEE 9th International Conference on Cloud Networking (CloudNet)*. IEEE, 1–8.
[24] Bowen Li, Xin Peng, Qilin Xiang, Hanzhang Wang, Tao Xie, Jun Sun, and Xuanzhe Liu. 2022. Enjoy your observability: an industrial survey of microservice tracing and analysis. *Empirical Software Engineering* 27 (2022), 1–28.
[25] Nicolas Marie-Magdelaine. 2021. *Observability and resources managements in cloud-native environnements*. Ph. D. Dissertation. Bordeaux.
[26] Sameer Mhaisekar. 2024. 10 tips to BAD Observability. (2024). https://www.youtube.com/watch?v=ljyMJ1i-zxI Accessed: March 10, 2025.
[27] D. Mittal. 2022. Mistakes to avoid in Observability. (2022). https://notes.drdroid.io/mistakes-to-avoid-in-observability Accessed: March 10, 2025.
[28] Moeen Ali Naqvi, Sehrish Malik, Merve Astekin, and Leon Moonen. 2022. On Evaluating Self-Adaptive and Self-Healing Systems using Chaos Engineering. In *2022 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*. IEEE, 1–10.
[29] Chujie Ni. 2023. *Adopting Observability-Driven Development for Cloud-Native Applications : Designing End-to-end Observability Pipeline using Open-source Software*. Master's thesis. KTH, School of Electrical Engineering and Computer Science (EECS).
[30] Sina Niedermaier, Falko Koetter, Andreas Freymann, and Stefan Wagner. 2019. On observability and monitoring of distributed systems–an industry interview study. In *Service-Oriented Computing: 17th International Conference, ICSOC 2019, Toulouse, France, October 28–31, 2019, Proceedings 17*. Springer, 36–52.
[31] J. O'Donnell. 2024. Top 10 Mistakes People Make When Building Observability Dashboards. (2024). https://logz.io/blog/top-10-mistakes-building-observability-dashboards/ Accessed: March 10, 2025.
[32] A. Patel. 2024. How to Resolve Bad Observability Data Quality. (2024). https://read.srepath.com/p/how-to-resolve-bad-observability Accessed: March 10, 2025.
[33] A. Patel. 2024. How to Solve 3 Data Flow Issues in Observability. (2024). https://read.srepath.com/p/how-to-solve-3-data-flow-issues-in Accessed: March 10, 2025.
[34] William Pourmajidi, Lei Zhang, John Steinbacher, Tony Erwin, and Andriy Miranskyy. 2023. A Reference Architecture for Observability and Compliance of Cloud Native Applications. *arXiv preprint arXiv:2302.11617* (2023).
[35] Mario Scrocca, Riccardo Tommasini, Alessandro Margara, Emanuele Della Valle, and Sherif Sakr. 2020. The Kaiju project: enabling event-driven observability. In *Proceedings of the 14th ACM International Conference on Distributed and Event-Based Systems*. 85–96.
[36] Anas Shatnawi, Bachar Rima, Zakarea Alshara, Gabriel Darbord, Abdelhak-Djamel Seriai, and Christophe Bortolaso. 2023. Telemetry of Legacy Web Applications: An Industrial Case Study. (Nov. 2023). https://doi.org/10.36227/techrxiv.24449092 working paper or preprint.
[37] Andy Singleton. 2016. The economics of microservices. *IEEE Cloud Computing* 3, 5 (2016), 16–20.
[38] Davide Taibi, Ben Kehoe, and Danilo Poccia. 2022. Serverless: from bad practices to good solutions. In *2022 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. IEEE, 85–92.
[39] Johannes Thönes. 2015. Microservices. *IEEE software* 32, 1 (2015), 116–116.
[40] Rafik Tighilt, Manel Abdellatif, Naouel Moha, Hafedh Mili, Ghizlane El Boussaidi, Jean Privat, and Yann-Gaël Guéhéneuc. 2020. On the Study of Microservices Antipatterns: a Catalog Proposal *(EuroPLoP '20)*. Association for Computing Machinery, New York, NY, USA, Article 34, 13 pages. https://doi.org/10.1145/3424771.3424812
[41] S. Townshend. 2023. Bad Observability. (2023). https://squaredup.com/blog/bad-observability/ Accessed: March 10, 2025.
[42] Muhammad Usman, Simone Ferlin, Anna Brunstrom, and Javid Taheri. 2022. A survey on observability of distributed edge & container-based microservices. *IEEE Access* 10 (2022), 86904–86919.
[43] A. Villela. 2022. Observability Mythbusters: Observability Anti-Patterns. (2022). https://faun.pub/observability-mythbusters-observability-anti-patterns-2b3062405b54 Accessed: March 10, 2025.
[44] A. Villela. 2024. OpenTelemetry Collector Anti-Patterns. (2024). https://geekingoutpodcast.substack.com/p/opentelemetry-collector-anti-patterns Accessed: March 10, 2025.
[45] Anton Widerberg and Erik Johansson. 2021. Observability of Cloud Native Systems: : An industrial case study of system comprehension with Prometheus & knowledge transfer. , 111 pages.
[46] Indika Wimalasuriya. 2024. AWS: Your Ally Against Observability Anti-Patterns | Conf42 Observability 2024. (2024). https://www.youtube.com/watch?v=Mn4eqXiZoHI Accessed: March 10, 2025.
[47] Tianyi Yang, Jiacheng Shen, Yuxin Su, Xiaoxue Ren, Yongqiang Yang, and Michael R Lyu. 2022. Characterizing and mitigating anti-patterns of alerts in industrial cloud systems. In *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 393–401.