Mapeando o ecossistema de microfrontends: um conjunto de dados de projetos de código aberto.

Samara Silvia Sabino**
samarasilvia.dev@gmail.com
Universidade Federal de Pernambuco, CIN
Recife, Pernambuco, BRA

Vinicius Cardoso Garcia
Universidade Federal de Pernambuco, CIN
Recife, BRA
vcg@cin.ufpe.br

Resumo

A arquitetura de Microfrontends (MFE) tem sido amplamente adotada na indústria como uma estratégia para escalar o desenvolvimento de interfaces web. No entanto, a realização de estudos nessa área é frequentemente limitada pela escassez de dados reais, levando pesquisadores a dependerem de projetos de demonstração que não refletem sistemas em produção. Este trabalho apresenta um estudo de mineração de repositórios com o objetivo de identificar, selecionar e catalogar projetos de **Microfrontend de código aberto (MFE-OSS) [5].**

Utilizando a API do GitHub e um conjunto de heurísticas técnicas baseadas em frameworks de integração (ex: Module Federation, Single-SPA), iniciamos com **2.793** repositórios candidatos. Um processo sistemático de filtragem, focado em atividade recente e maturidade colaborativa, reduziu a amostra para **1.205** projetos ativos. Em seguida, critérios de elegibilidade relacionados à documentação e ao idioma resultaram em **842** candidatos auditáveis. Por fim, após a remoção de falsos positivos e uma curadoria manual, o conjunto final foi consolidado em **455** projetos aprovados.

O dataset resultante documenta características estruturais, diversidade tecnológica e métricas sociotécnicas[1], fornecendo uma base sólida para investigações futuras sobre arquiteturas de Microfrontend.

Palavras-chave: Microfrontends, Frontend Distribuído, Mineração de Repositórios, Datasets Open-Source, Engenharia de Software.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY
ACM ISBN 978-1-4503-XXXX-X/2018/06.
DOI: https://doi.org/XXXXXXX.XXXXXXX

1 Introdução

A adoção de sistemas distribuídos tornou-se predominante na engenharia de software ao longo das últimas décadas, com os microsserviços consolidando-se como uma abordagem dominante no backend. No frontend, a popularização das Single Page Applications (SPAs), impulsionada por frameworks como React, Angular e Vue, trouxe ganhos expressivos de interatividade e experiência do usuário. Contudo, à medida que essas aplicações evoluíram, surgiram desafios relacionados à escalabilidade, manutenibilidade e ao trabalho paralelo entre equipes, fenômeno frequentemente descrito pela indústria e literatura técnica como o *monólito frontend* [3, 6].

Nesse contexto, os Microfrontends emergiram como uma alternativa arquitetural, propondo a decomposição da interface em unidades independentes. A abordagem, detalhada em obras seminais de **Mezzalira [4]** e **Geers [2]**, alinha-se aos princípios de autonomia e desacoplamento já consolidados no backend. Desde que foi listada como técnica essencial pelo ***ThoughtWorks Technology Radar [6]***, a arquitetura tem sido amplamente adotada na indústria. No entanto, a literatura técnica concentra-se majoritariamente em conceitos e padrões de design, oferecendo pouca evidência empírica sobre como os MFE são efetivamente implementados em sistemas reais.

Como consequência, permanecem em aberto questões fundamentais sobre os trade-offs dessa arquitetura, incluindo impactos na complexidade de integração, performance, organização dos repositórios e autonomia das equipes. Diferentemente do domínio de microsserviços, que já conta com conjuntos de dados consolidados para estudos empíricos, o ecossistema de Microfrontends carece de bases de dados sistemáticas e representativas.

Motivado por essa lacuna, este trabalho apresenta o **MFE-OSS Dataset**, um conjunto de dados curado de projetos de MFE de código aberto, obtido por meio de um processo de mineração multietapas de repositórios do GitHub. O dataset documenta padrões estruturais, diversidade tecnológica e características sociotécnicas do ecossistema atual, oferecendo uma base sólida para futuras análises sobre as práticas e desafios relacionados à implementação de arquiteturas distribuídas no frontend.

2 TRABALHOS RELACIONADOS

A construção deste dataset baseia-se em estudos recentes de catalogação e mineração de ecossistemas distribuídos. O trabalho de **d'Aragona et al.** [1] é pioneiro ao apresentar o primeiro dataset de larga escala focado em microsserviços, destacando a viabilidade de pipelines[2] de curadoria para análises empíricas reprodutíveis. Embora voltado ao backend, este estudo estabeleceu o padrão metodológico que orientou a filtragem de metadados, inspeção técnica e validação estrutural neste trabalho. Além disso, investigações sobre a evolução de projetos JavaScript e pacotes npm, como os estudos de **Wittern et al.** [7] e **Zhou et al.** [9], ressaltaram a importância de datasets que não apenas capturem tecnologias, mas também a modularização e dependências no ecossistema web. No entanto, nenhum desses trabalhos aborda as fronteiras arquiteturais de aplicações distribuídas no frontend, nem a organização de

*Both authors contributed equally to this research.

[1] Indicadores que combinam aspectos técnicos do código com sinais de colaboração, atividade e organização das equipes de desenvolvimento, incluindo volume de contribuições, distribuição de trabalho entre módulos e padrões de acoplamento entre microfrontends.

[2] Definição: um fluxo encadeado de etapas, geralmente automatizadas, no qual a saída de uma etapa serve como entrada da etapa seguinte, com o objetivo de transformar dados, código ou artefatos de forma sistemática, reprodutível e controlada.

Conference'17, Washington, DC, USA
2027. ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/XXXXXXX.XXXXXXX

projetos Microfrontend. Assim, o estudo em questão se diferencia ao aplicar uma metodologia de mineração sistemática — similar à utilizada no backend distribuído — a um domínio pouco explorado em estudos sistemáticos de caráter observacional, oferecendo uma base inicial para futuras contribuições.

3 METODOLOGIA

Nesta seção, descrevemos o processo de construção do MFE-OSS Dataset, incluindo a estratégia de mineração, os critérios de seleção, os mecanismos de validação e o protocolo de curadoria manual, conforme ilustrado na **Figura 1**.

3.1 Visão Geral

A construção do dataset seguiu um processo de mineração em múltiplos estágios, baseado em uma estratégia *code-first*, que prioriza evidências técnicas no código-fonte antes da aplicação de filtros baseados em metadados.

3.2 Estrutura e Atributos (Data Schema)

A Tabela 1 apresenta o esquema completo do projeto, detalhando os atributos coletados, seus tipos e descrições semânticas.

Tabela 1: Estrutura doMFE-OSSDataset (Data Schema)

Atributo	Tipo	Descrição
Identificação e Métricas Gerais		
repo_slug	String	Identificador único no GitHub.
commit_hash	String	Hash do commit analisado.
stars, forks	Inteiro	Métricas de popularidade.
open_issues	Inteiro	Sinaliza atividade contínua.
project_age	Inteiro	Idade do projeto em anos.
Métricas Sociotécnicas		
contributors	Inteiro	Número total de contribuidores únicos.
pull_requests	Inteiro	Total de pull requests abertos e fechados.
contributors_per_module	Real	Média de contribuidores por microfrontend.
cross_module_pr_ratio	Real	Proporção de PRs que alteram múltiplos módulos.
churn_rate	Real	Frequência média de alterações de código ao longo do tempo.
Arquitetura Técnica		
integration_technique	Categoria	Framework detectado (Module Federation, etc.).
repo_structure	Categoria	Monorepo ou Polyrepo.
build_tool	Categoria	Webpack, Vite, etc.
Aspectos Organizacionais		
project_purpose	Categoria	Demo, Produção, Template, POC.
project_origin	Categoria	Indústria, Comunidade, Individual.
api_style	Categoria	REST, GraphQL, eventos, N/A.
Complexidade e Granularidade		
microfrontends_count	Inteiro	Número de MFE identificados.
workspace_count	Inteiro	Pacotes internos no monorepo.

3.3 Estratégia de Mineração e Identificação (Code-Based Heuristics)

Como primeiro estágio operacional do processo de construção do conjunto de dados, foi definida uma estratégia de mineração orientada a código, responsável pela identificação inicial e coleta sistemática dos projetos candidatos. Nesse contexto, **a API do GitHub** serviu como fonte primária de dados. Para mitigar limitações de *rate limits* e evitar buscas superficiais, desenvolvemos um módulo integrado de mineração capaz de executar consultas paralelas e profundas nos arquivos dos projetos.

Inspirado pela abordagem de *d'Aragona et al. (2024)*, que inferiu arquiteturas de microsserviços pela análise de arquivos *docker-compose*, nossa estratégia concentrou-se na inspeção direta de artefatos de *build* e configuração. O objetivo foi identificar projetos que realmente implementam a arquitetura de Microfrontends, independentemente da terminologia utilizada na documentação ou no README.

As heurísticas de inclusão basearam-se em duas estratégias de varredura nos arquivos de configuração:

(i) **Identificação de Artefatos de Orquestração (IAO):** Busca por palavras-chave explícitas de frameworks (ex: *module-federation*, *single-spa*, *qiankun*) em manifestos como *package.json*, *webpack.config.js* e *vite.config.js*;
(ii) **Validação de Padrões Distribuídos (VPD):** Inspeção de arquivos de *build* para localizar configurações estruturais de *remotes*, *hosts* ou *shells*.

Essa varredura profunda resultou na identificação de um universo inicial de **2.793 repositórios candidatos**, contendo evidências explícitas sobre a adoção de tecnologias de Microfrontend.

3.4 Extração de Metadados

Uma vez mapeados os projetos com evidências técnicas de MFE, a segunda etapa consistiu na extração e análise de metadados para avaliar a **maturidade e atividade** dos projetos. Essa fase foi executada após a identificação técnica inicial dos projetos, visto que a coleta de métricas históricas — como varredura de *commits* e contagem de contribuidores — é custosa em termos de consumo da API.

3.5 Critérios de Seleção e Filtragem

Os critérios de inclusão e exclusão foram organizados em diferentes estágios do processo de mineração, abrangendo atividade, maturidade e validade técnica dos repositórios. A **Tabela 2** sintetiza os filtros aplicados ao longo da pipeline de construção do dataset.

A aplicação destes filtros reduziu o conjunto inicial de 2.793 para **1.205 projetos ativos**,eliminando POCs descartadas, *scaffolds*[3]

3.6 Filtragem de Qualidade e Validação Arquitetural

Após a filtragem preliminar de metadados, aplicou-se um *Quality Gate* voltado à viabilidade de análise, removendo repositórios sem documentação acessível ou cuja descrição estivesse em idiomas distintos do inglês. Esse filtro reduziu o conjunto para **842 candidatos auditáveis**.

Entretanto, reconhecemos que a presença de palavras-chave ou uma contagem mínima de commits não é suficiente para afirmar a existência de uma arquitetura de Microfrontends. Por esse motivo, conduzimos uma etapa de validação técnica aprofundada (*Sanity Check*), destinada a inspecionar a estrutura interna dos projetos

[3]Definição: São repositórios gerados automaticamente por ferramentas (CLI, templates, boilerplates) que contêm apenas a estrutura inicial do projeto, sem evolução funcional ou arquitetural significativa.

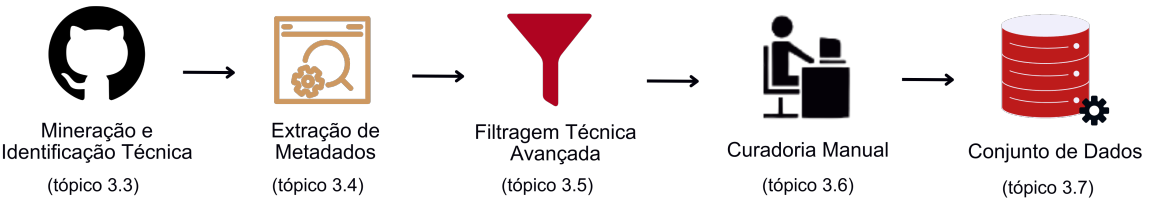

Figura 1: Visão Geral da pipeline de geração do conjunto de dados

e eliminar falsos positivos[4]. Essa validação foi baseada em três dimensões críticas de validade (I_6), sumarizadas na **Tabela 2**.

Este processo rigoroso removeu 387 falsos positivos (projetos sem evidência de orquestração real), consolidando a base final em **455 projetos validados** que seguiram para a etapa de curadoria manual.

3.6.1 Enriquecimento de Dados (Deep Mining). Para este conjunto final selecionado, executamos uma rotina de mineração profunda para extrair métricas históricas complexas:

- **Total Pull Requests:** Soma de PRs abertos e fechados, servindo como *proxy* de colaboração externa.
- **Closed Issues:** Contagem histórica de problemas resolvidos, utilizada como indicador de manutenção e suporte.
- **Churn Rate:** Cálculo da frequência média de alteração de código (Adições + Deleções / Semanas de vida), indicando a volatilidade e estabilidade técnica do projeto.

3.7 Protocolo de Curadoria Manual

Os **455 projetos** aprovados foram submetidos a um protocolo de **Validação Cruzada Circular** (*Circular Cross-Validation*), com o objetivo de reduzir vieses interpretativos e aumentar a confiabilidade dos dados. No protocolo é determinado que três pesquisadores com experiência acadêmica e industrial realizaram a catalogação inicial dos projetos, seguida por uma rotação sistemática de revisões por pares. Cada classificação foi auditada por um segundo pesquisador independente e, em casos de divergência persistente, a decisão final foi estabelecida por um terceiro membro atuando como juiz (*tie-breaker*).

3.8 Nomenclatura e Classificação de Dados

Para mapear o *estado da prática*[5], os dados foram organizados em uma nomenclatura híbrida, dividida em dois grupos:

3.8.1 Dimensões Automatizadas. Dados extraídos via script e validados manualmente:
1. Técnica de Integração: Define a orquestração central. Classificamos em: (i) *Module Federation* (uso nativo de Webpack 5/Vite); (ii) *Single-SPA* (meta-framework); e (iii) *Custom/Iframe* (soluções ad-hoc).
2. Estrutura do Repositório: Classificação baseada na organização de pastas em *Monorepo* (múltiplos apps na raiz, via Lerna/Nx) ou *Polyrepo* (módulo isolado).

3.8.2 Dimensões Qualitativas. Atributos preenchidos exclusivamente via curadoria humana:
1. Finalidade (*Purpose*): Baseado em d'Aragona et al. (2024), classificamos em: *Produção* (sistemas reais/comerciais); *Demo/Estudo* (aprendizado de conceitos); *Template* (estruturas base); e *Biblioteca/Tool* (ferramentas de suporte).
2. Origem (*Developed By*): Identifica a autoria em: *Indústria* (corporativos/consultorias); *Comunidade* (fundações/open-source); ou *Individual* (independentes).
3. Estilo de API: Inspeção manual para identificar *REST*, *GraphQL* ou *BFF*.
4. Contagem de MFE: Contagem manual para distinguir *remotes* funcionais de bibliotecas compartilhadas, garantindo precisão na métrica de complexidade.

4 CONJUNTO DE DADOS

O conjunto de dados inicial MFE-OSS resultante consolida informações de 50 projetos de código aberto. A **Figura 2** resume as métricas descritivas principais. Detalhes visuais sobre a distribuição de técnicas e origens encontram-se no **??**.

4.1 Relação entre Microfrontends e Métricas de Atividade

A **Tabela 3** resume a distribuição das métricas de atividade dos projetos conforme a quantidade de módulos de microfrontends identificados. As métricas incluem o número de *pull requests*, bem como *issues* abertas e fechadas, agregadas por faixas de quantidade de MFE.

4.2 Técnica de Integração

A **Figura 2** apresenta a proporção das técnicas mais utilizadas. O *Module Federation* (24.0%) e o *Single-SPA* (24.0%) consolidam-se como líderes, sendo que a maior parte dos projetos se enquadra em soluções customizadas ou outras categorias (35.6%).

4.3 Estrutura do Repositório

A análise estrutural (**Figura 2**) revela um ecossistema dividido: metade dos projetos opta por *Monorepos* (68.0%) e a outra metade mantém a estrutura de *Polyrepos* (32.0%).

[4]Para calibrar a precisão dos scripts de validação, realizamos preliminarmente uma inspeção manual em uma amostra aleatória de 5% dos repositórios candidatos, ajustando as heurísticas conforme os falsos positivos identificados nessa fase piloto.
[5]Definição: O estado atual de uma prática ou abordagem em uma área específica, considerando as metodologias, técnicas e ferramentas mais comuns ou amplamente utilizadas no momento.

4.4 Finalidade e Origem dos Projetos

A **Figura 2** ilustra a relação entre a finalidade do projeto e sua origem. Observa-se que desenvolvedores *individuais* focam massivamente em projetos de **Estudo/Demo** (projetos), enquanto a *indústria* prioriza **Produção** (projetos).

5 POTENCIAIS APLICAÇÕES

O MFE-OSS Dataset foi concebido para apoiar estudos baseados em dados sobre arquiteturas de microfrontends, oferecendo uma base estruturada e reprodutível para a análise do estado da prática em larga escala.

Sob uma perspectiva arquitetural e de projetos. O dataset pode ser utilizado para investigar a evolução de aplicações frontend distribuídas, focando em como decisões arquiteturais — como **monorepo vs. polyrepo** e técnicas de orquestração — impactam a escalabilidade, a complexidade do código e a colaboração entre equipes. Em vez de métricas em tempo real, o conjunto de dados possibilita uma análise retrospectiva, permitindo a identificação de boas práticas e pontos de melhoria para a evolução de sistemas distribuídos, complementando ferramentas tradicionais de monitoramento.

Além disso, sob a ótica de uma **investigação baseada em dados em andamento**, o dataset já vem sendo utilizado como base observacional em uma pesquisa de mestrado em curso, voltada à extração e análise de evidências de *dívida técnica auto-admitida (SATD)*[6] em larga escala de arquiteturas de MFE.

Outra aplicação relevante do dataset consiste na investigação da **complexidade do código**, da **testabilidade** e da **dinâmica de manutenção** em aplicações frontend distribuídas. A partir do dataset, é possível analisar como diferentes decisões arquiteturais — como técnicas de integração, organização dos repositórios e a **quantidade de módulos de microfrontends** — influenciam a modularidade do código, a adoção de testes automatizados e indicadores de manutenção, como o volume de **issues** e **pull requests** abertas. Essas análises permitem investigar a relação entre o grau de decomposição arquitetural e a carga de manutenção dos projetos, fornecendo evidências sobre seu impacto na manutenibilidade e na evolução dos sistemas ao longo do tempo.

Nesse sentido, o MFE-OSS Dataset fornece uma base observacional consistente para explorar a relação entre arquiteturas de microfrontends e práticas ágeis de desenvolvimento.

6 AMEAÇAS À VALIDADE

Discutimos as ameaças à validade deste estudo seguindo as diretrizes de Wohlin et al. [8], organizando-as em validade de construto, interna e externa.

Validade de Construto Esta dimensão refere-se à adequação entre as métricas adotadas e o fenômeno estudado. A principal ameaça decorre das heurísticas de identificação técnica, uma vez que a adaptação da metodologia de d'Aragona et al. (2024) exigiu a definição de novos indicadores específicos de Microfrontends, como *Module Federation* e *Single-SPA*. Existe o risco de que variações menos convencionais de implementação não tenham sido capturadas. Para mitigar essa ameaça, utilizamos um conjunto abrangente de palavras-chave derivadas dos frameworks mais utilizados na indústria, aliado a uma validação manual dos resultados. Além disso, para lidar com a volatilidade inerente a repositórios open-source, cada projeto do dataset registra o *commit hash* analisado, assegurando reprodutibilidade.

Validade Interna Esta categoria diz respeito à confiabilidade do processo analítico e ao risco de viés subjetivo na interpretação dos dados. A ameaça mais relevante deriva da **classificação manual dos projetos**, especialmente na distinção entre sistemas de "Demonstração" e de "Produção", bem como na categorização das autorias entre "Indústria", "Comunidade" e "Individual". Para reduzir esse risco, aplicamos o protocolo de **Validação Cruzada Circular** descrito na Seção 2.4, no qual cada projeto foi revisado por pelo menos dois pesquisadores em regime de rotação. Divergências foram avaliadas por um terceiro juiz, assegurando maior consistência nas decisões e reduzindo a influência de vieses individuais.

Validade Externa A validade externa diz respeito à possibilidade de generalização dos resultados. A principal ameaça decorre do foco exclusivo em repositórios públicos hospedados no GitHub, o que pode não refletir completamente práticas adotadas em sistemas proprietários. Como mitigação, o processo de curadoria priorizou projetos com evidências de maturidade técnica e atividade contínua, reduzindo o viés associado a exemplos didáticos ou protótipos simplificados. Ainda assim, os resultados devem ser interpretados como representativos do ecossistema open-source de microfrontends, e não como um retrato exaustivo de sistemas proprietários.

7 CONCLUSÃO

A arquitetura de Microfrontends consolidou-se como uma abordagem relevante para escalar o desenvolvimento web moderno; ainda assim, a literatura em engenharia de software apresenta uma escassez de estudos empíricos sistemáticos que avaliem seus benefícios, limitações e trade-offs. Este trabalho apresentou um mapeamento em larga escala de projetos OSS adotando MFE, resultando em um conjunto de dados curado voltado especificamente para esse ecossistema.

A partir de uma adaptação metodológica inspirada em investigações prévias sobre microsserviços, conduzimos uma mineração multietapas, aliando filtragem técnica e curadoria manual para assegurar rigor e consistência. O resultado é um conjunto de **455 projetos validados**, que representa implementações reais e heterogêneas, indo além de exemplos simplificados e refletindo práticas concretas da comunidade.

Em síntese, o **MFE-OSS Dataset** fornece uma base comparável, reprodutível e acessível para estudos empíricos sobre frontend distribuído. Ao disponibilizar publicamente esse recurso, buscamos apoiar novas análises, fomentar comparações mais precisas entre abordagens de orquestração e contribuir para uma compreensão mais fundamentada do estado da prática em Microfrontends.

[6] A *Self-Admitted Technical Debt (SATD)* refere-se a uma forma de dívida técnica em que a equipe de desenvolvimento reconhece explicitamente a necessidade de realizar melhorias ou correções no código no futuro, mas opta por adiar essas mudanças devido a restrições de tempo ou outras prioridades.

Figura 2: Caracterização do Ecossistema de MFE-OSS Dataset(50 Projetos).

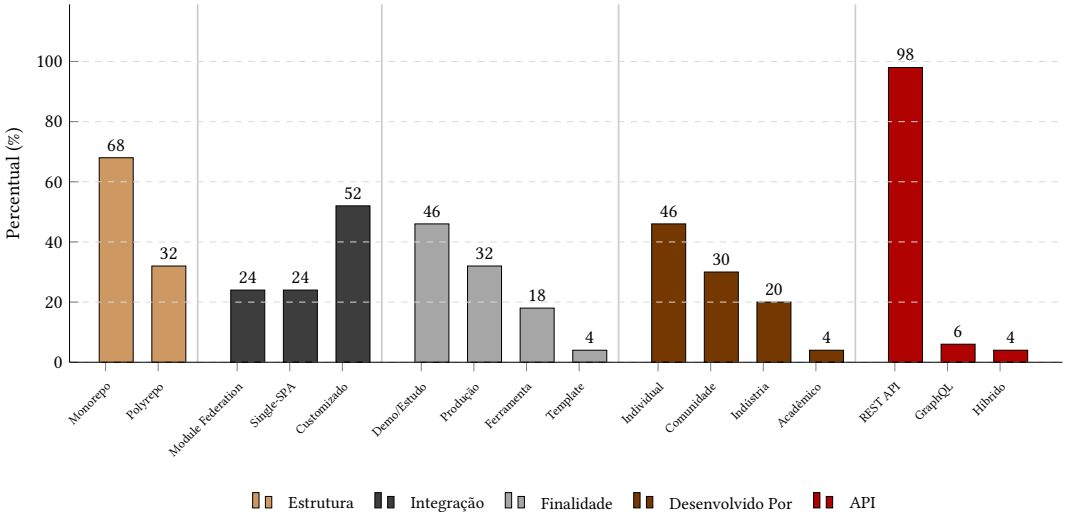

Tabela 2: Critérios Unificados de Seleção, Filtragem e Validação (I_1–I_6)

ID	Dimensão	Descrição e Objetivo do Critério
I_1	Atividade	Commit recente (2021–2025) para garantir compatibilidade com ferramentas modernas.
I_2	Colaboração	Pelo menos um contribuidor (inclui projetos individuais relevantes).
I_3	Maturidade	Mínimo de 10 commits (remove scaffolds vazios e testes descartados).
I_4	Documentação	README acessível para viabilizar a análise manual.
I_5	Idioma	Documentação predominante em inglês.
I_6	Validade Técnica	Validação estrutural profunda para remover falsos positivos (detalhada abaixo).
Detalhamento da Validação Técnica (I_6)		
$I_{6.1}$	Dependências	Verifica *package.json* por libs reais (*webpack, vite, single-spa*) para evitar menções apenas textuais.
$I_{6.2}$	Estrutura	Analisa pastas (ex: */packages*) para distinguir monorepos de monolitos simples.
$I_{6.3}$	Entry Points	Busca configs de build (*remotes/exposes*) que confirmam a arquitetura distribuída.

Tabela 3: Relação entre número de microfrontends, Pull Requests e Issues

Faixa MFE	PRs (média)	Issues Open	Issues Closed	Projetos
0–5	361.4	34.1	78.3	29
6–10	701.2	26.1	41.1	8
11–20	858.7	39.7	386.0	3
20+	1847.7	307.6	333.5	10

Referências

[1] Francesco d'Aragona, Riccardo Bruni, Jacopo Soldani, and Antonio Brogi. 2024. A Dataset of Microservices-based Open-Source Projects. In *Proceedings of the 21st International Conference on Mining Software Repositories (MSR '24)*. ACM. doi:10.1145/3643991.3644889

[2] Michael Geers. 2020. *Micro Frontends in Action.* Manning Publications.

[3] Cam Jackson. 2019. Micro Frontends. martinfowler.com. https://martinfowler.com/articles/micro-frontends.html Acessado em: 2025-09-24.

[4] Luca Mezzalira. 2021. *Building Micro-Frontends.* O'Reilly Media.

[5] Samara Silvia Sabino. 2025. Dataset and Artifacts of Mapping the Microfrontend Ecosystem. GitHub Repository. https://github.com/SamaraSilvia81/mfe-oss-dataset Código e dataset do trabalho de Mestrado. Acessado em: December 16, 2025.

[6] ThoughtWorks. 2016. Technology Radar: Micro Frontends. https://www.thoughtworks.com/radar/techniques/micro-frontends Acessado em: 2025-09-24.

[7] Erik Wittern, Philippe Suter, and Shriram Rajagopalan. 2016. A look at the dynamics of the JavaScript ecosystem. In *Proceedings of the 13th International Conference on Mining Software Repositories (MSR '16)*. ACM, 351–361. doi:10.1145/2901739.2901743

[8] Claes Wohlin. 2012. *Experimentation in Software Engineering.* Springer.

[9] Xiang Zhou, Xin Peng, Tao Xie, Jun Sun, Chao Ji, Wenyun Li, and Dan Ding. 2021. Fault Analysis and Debugging of Microservice Systems: Industrial Survey, Benchmark System, and Empirical Study. *IEEE Transactions on Software Engineering* 47, 2 (2021), 243–260. doi:10.1109/TSE.2019.2895256

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009