Adoção de Boas Práticas e Integração Contínua em Repositórios GitHub com Linguagens Funcionais

Rullian Andriel Pianca, Jailton Coelho

Instituto Federal de Educação, Ciência e Tecnologia do Paraná (IFPR)
Campus Telêmaco Borba – Telêmaco Borba, PR – Brasil

andrielrullian@gmail.com, jailton.coelho@ifpr.edu.br

Abstract. *Functional programming languages, such as Haskell, Elixir, and F#, have gained prominence in both academic and industrial contexts for enabling concise, composable, and parallelizable code. Although their adoption in open-source projects has increased, little is known about how well these projects follow established software engineering practices. This study investigates whether repositories written in functional languages adopt (i) maintenance guidelines recommended by the open-source community and (ii) continuous integration practices. An analysis of 496 GitHub repositories revealed widespread use of introductory documentation (91.9%) and licenses (89.7%), but low adoption of contribution guidelines (34.1%) and codes of conduct (17.7%). Continuous integration was present in 85.1% of the projects.*

Resumo. *Linguagens de programação funcionais, como Haskell, Elixir e F#, têm ganhado destaque por promoverem código mais conciso e paralelizável. Embora seu uso tenha crescido em projetos de código aberto, pouco se sabe sobre a adoção de boas práticas de manutenção em repositórios com estas linguagens. Este estudo investiga se projetos com linguagens funcionais adotam (i) diretrizes de manutenção recomendadas pela comunidade open source e (ii) práticas de integração contínua. A partir da análise de 496 repositórios no GitHub, observou-se alta presença de documentação introdutória (91,9%) e licença (89,7%), mas baixa adoção de guias de contribuição (34,1%) e códigos de conduta (17,7%). Já a integração contínua aparece em 85,1% dos projetos.*

1. Introdução

Linguagens de programação funcionais vêm ganhando destaque nos últimos anos, tanto em ambientes acadêmicos quanto industriais [Castagna et al. 2023]. Baseadas em conceitos como imutabilidade, funções puras e composição funcional, essas linguagens promovem programas mais concisos e paralelizáveis [Hudak 1989]. Haskell, OCaml, F# e Elixir são exemplos populares, com aplicações em sistemas embarcados, aplicações distribuídas e processamento de dados em larga escala. Um caso notável é a linguagem Elixir [Valim 2023], utilizada por empresas como Discord e PepsiCo [Castagna et al. 2023].

Paralelamente, práticas modernas de engenharia de software são fundamentais para o sucesso de projetos de código aberto. Adoção de diretrizes de contribuição, documentação estruturada e ferramentas de automação, como integração contínua, são apontadas como essenciais para manter a qualidade e facilitar a

colaboração [Steinmacher et al. 2015, Elazhary et al. 2019]. O GitHub, por exemplo, recomenda explicitamente essas práticas e fornece suporte técnico para sua implementação [GitHub 2025b, GitHub 2025a].

Estudos recentes têm investigado a adoção dessas práticas em diferentes contextos e linguagens [Prana et al. 2019, Elazhary et al. 2019, Vasilescu et al. 2015, Ray et al. 2014]. No entanto, até o momento, não foram encontrados estudos com foco específico em repositórios que utilizam linguagens funcionais. Essa ausência representa uma lacuna importante, considerando as particularidades dessas linguagens e seu crescente uso na indústria e na academia.

Este estudo busca investigar se projetos de software desenvolvidos com linguagens funcionais adotam diretrizes de manutenção recomendadas pela comunidade open source e fazem uso de práticas de integração contínua. Com base nessa motivação, este estudo propõe responder às seguintes questões de pesquisa:

QP1. Projetos desenvolvidos com linguagens de programação funcionais adotam boas práticas de manutenção recomendadas pela comunidade open source?

QP2. Projetos desenvolvidos com linguagens de programação funcionais adotam práticas de integração contínua?

Para responder a essas questões de pesquisa, foram analisados 496 repositórios GitHub que utilizam uma das dez linguagens funcionais mais populares, a partir da lista dos 100 mil repositórios com mais estrelas.

2. Procedimentos Metodológicos

Inicialmente, foi construída uma lista com os 100.000 repositórios mais populares do GitHub, ordenados pelo número de estrelas, coletados em maio de 2025. A decisão de limitar a amostra a esses repositórios visa concentrar a análise em projetos mais relevantes. Com base na edição de 2023 da pesquisa anual do Stack Overflow[1], foram selecionadas as dez linguagens de programação funcionais mais populares, incluindo linguagens puramente funcionais ou com forte suporte ao paradigma. As linguagens consideradas no estudo foram: Elixir, Clojure, Haskell, F#, Erlang, OCaml, PureScript, Idris, Agda e Lisp.

A partir da lista dos 100.000 repositórios mais populares, identificaram-se 937 repositórios cuja linguagem principal corresponde a uma das linguagens funcionais mencionadas. Em seguida, foram removidos 438 repositórios que não apresentaram nenhum *commit* nos últimos seis meses, com o objetivo de manter apenas projetos com atividade recente. Além disso, foram excluídos outros 3 repositórios por estarem arquivados. Segundo a documentação do GitHub [2], projetos arquivados são repositórios marcados como inativos, nos quais não é possível realizar alterações sem antes reativá-los.

Após essas etapas, a amostra final consistiu em 496 repositórios. Para caracterizar o conjunto de dados analisado, a Figura 1 apresenta a distribuição do número de estrelas (*stars*), contribuidores e idade (meses) dos repositórios, com remoção dos *outliers*. As medianas são de 1.000 estrelas, 43 contribuidores e 109 meses de idade por projeto. A Figura 2 apresenta a distribuição do número de repositórios por linguagem funcional.

[1] [https://survey.stackoverflow.co/2023/

[2] [https://docs.github.com/en/repositories/archiving-a-github-repository/

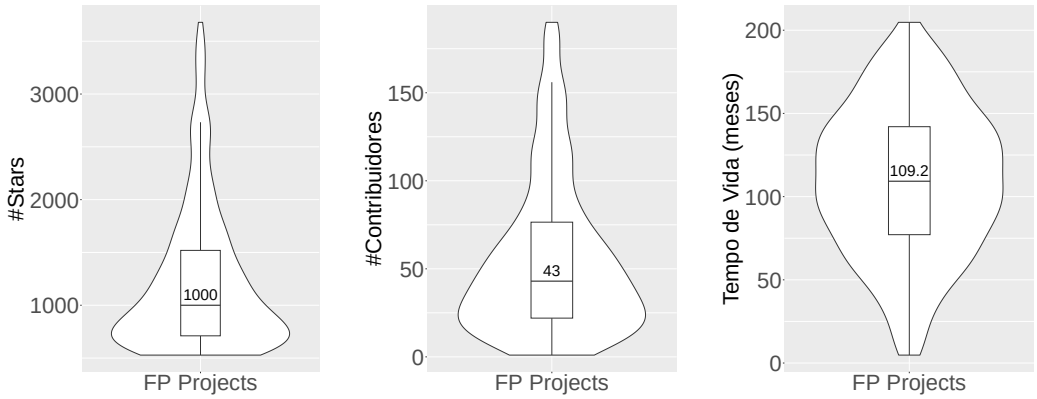

Figura 1. Distribuição do número de estrelas (*stars*), contribuidores e tempo de vida (meses) dos repositórios, com remoção de *outliers*.

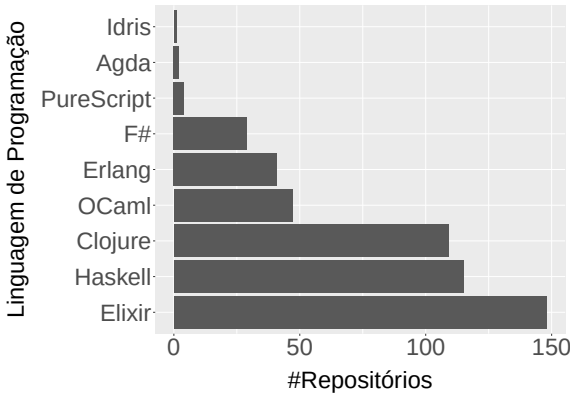

Figura 2. Quantidade de repositórios por linguagem funcional.

3. Resultados

QP1. Projetos desenvolvidos com linguagens de programação funcionais adotam boas práticas de manutenção recomendadas pela comunidade open source?

Para responder a essa questão, foram analisadas diretrizes recomendadas para projetos open source, conforme os *Open Source Guides* [GitHub 2025c] e a documentação oficial do GitHub [GitHub 2025b]. Foram consideradas sete práticas: documentação introdutória, código de conduta, guia de contribuição, templates de issues e pull requests, wiki e licença de uso. A verificação da presença dessas práticas foi realizada de forma automatizada, por meio da detecção dos respectivos arquivos na raiz de cada repositório.

- **README:** arquivo que introduz o projeto, seu propósito, como instalar e usar. Repositórios com README podem apresentar maior engajamento da comunidade [Wang et al. 2023].
- **Código de Conduta:** documento que define normas de comportamento para contribuidores. Sua presença pode estar associada a comunidades mais inclusivas e acolhedoras [Tourani et al. 2017].
- **Guia de Contribuição:** especifica como contribuir com o projeto, quais padrões

archiving-repositories

seguir e como executar testes. Sua adoção pode ajudar a entrada de novos contribuidores [Steinmacher et al. 2015].
- **Template de Issues:** fornece estrutura para relato de problemas ou sugestões.
- **Template de Pull Request:** guia para descrever submissões de código.
- **Wiki:** espaço adicional para documentação extensiva, como tutoriais, arquitetura do sistema e decisões técnicas.
- **Licença:** estabelece as permissões e restrições de uso do software.

A Tabela 1 resume a frequência com que essas diretrizes foram adotadas nos 496 repositórios analisados. Observa-se ampla presença de documentação essencial: 91,9% dos repositórios possuem documentação introdutória (README) e 89,7% apresentam uma licença de uso. Apenas 34,1% dos repositórios incluíam um guia de contribuição, e 17,7% adotavam um código de conduta. Templates para abertura de issues (5,8%) e pull requests (10,1%) foram diretrizes menos adotadas. A presença de wiki foi registrada em 66,5% dos casos. Por fim, entre os 445 (89,7%) repositórios que especificaram uma licença, a MIT foi a mais popular, presente em 150 deles, seguida pela Apache License 2.0, encontrada em 74 repositórios, e a Eclipse Public License 1.0 em 43.

Tabela 1. Adoção de diretrizes de contribuição nos repositórios analisados

Diretriz	Repositórios	Percentual
README	456	91.9%
Código de Conduta	88	17.7%
Guia de Contribuição	169	34.1%
Template de Issues	29	5.8%
Template de Pull Request	50	10.1%
Wiki	330	66.5%
Licença	445	89.7%

QP2. Projetos desenvolvidos com linguagens de programação funcionais adotam práticas de integração contínua?

Com o objetivo de investigar a adoção de boas práticas de engenharia de software, analisou-se o uso de ferramentas de integração contínua (CI/CD) na lista de 496 repositórios. Os repositórios que utilizam Agda, Idris e PureScript foram removidos da amostra porque tinham poucos repositórios ativos para análise estatística. A verificação foi realizada por meio da presença de arquivos de workflow localizados no diretório `.github/workflows/`, os quais indicam a configuração de pipelines automatizados por meio do GitHub Actions. No total, foram encontrados 415 repositórios com esses arquivos.

Estudos anteriores, como os de Vasilescu et al. [Vasilescu et al. 2015] e Rausch et al. [Rausch et al. 2017], utilizaram uma abordagem similar para identificar o uso de CI/CD em projetos do GitHub. Para validar o algoritmo, 79 repositórios, escolhidos aleatoriamente, foram inspecionados manualmente (com 95% de confiança e 10% de margem de erro), confirmando que todos continham arquivos de workflow. A Tabela 2 detalha o percentual de adoção de CI/CD por linguagem funcional. Os repositórios em Agda (2), Idris (1) e PureScript (4) apresentaram 100% de adoção de CI/CD.

Tabela 2. Percentual de repositórios com CI/CD por linguagem funcional.

Linguagem	Total	CI/CD
Clojure	109	70,6%
Elixir	148	93,2%
Erlang	41	90,2%
F#	29	75,9%
Haskell	115	89,6%
OCaml	47	80,9%

Os resultados revelam uma relevante adoção de integração contínua entre os projetos analisados. Linguagens como Elixir (93,2%), Haskell (89,6%) e Erlang (90,2%) apresentam percentuais elevados. Já linguagens como Clojure (70,6%), F# (75,9%) e OCaml (80,9%) apresentam taxas intermediárias.

4. Ameaças à validade

Quanto à **validade de construção**, como o número total de projetos com linguagens funcionais no GitHub é desconhecido, adotou-se como amostra os 100 mil repositórios públicos mais populares, restringindo-se àqueles cuja linguagem principal pertence a um conjunto de dez linguagens funcionais mais populares. Para garantir a **validade interna**, a identificação das práticas analisadas foi automatizada com base em arquivos e diretórios padronizados. A acurácia da detecção de CI/CD foi verificada por meio da inspeção manual de 79 repositórios. Quanto à **validade externa**, os resultados não se aplicam a projetos privados ou inativos. Ainda assim, os 496 repositórios analisados fornecem uma amostra representativa de projetos ativos e populares com linguagens funcionais.

5. Conclusão

Foi investigada a adoção de boas práticas de manutenção e de integração contínua em projetos desenvolvidos com linguagens de programação funcionais, utilizando uma amostra de 496 repositórios hospedados no GitHub. Em relação à adoção de boas práticas de manutenção (**QP1**), observou-se que a maioria dos projetos analisados inclui uma descrição geral do sistema (91,9%) e uma licença de uso (89,7%). No entanto, a presença de diretrizes formais de contribuição (34,1%), código de conduta (17,7%), templates de issues (5,8%) e pull requests (10,1%) é menos frequente. Quanto à adoção de integração contínua (**QP2**), Elixir apresentou a maior taxa de adoção (93,2%), seguida por Erlang (90,2%) e Haskell (89,6%).

Como trabalhos futuros, pretende-se realizar uma comparação com repositórios de linguagens não-funcionais populares, como JavaScript e Python, a fim de investigar se as práticas observadas se diferenciam entre paradigmas de programação. Além disso, será avaliada a existência de correlação entre a adoção de boas práticas e métricas como número de estrelas, quantidade de *commits* e número de contribuidores.

Disponibilidade de Artefato

Os dados desta pesquisa estão disponíveis publicamente em https://doi.org/10.5281/zenodo.16789408.

Agradecimentos

Essa pesquisa é apoiada pelo IFPR.

Referências

Castagna, G., Duboc, G., and Valim, J. (2023). The design principles of the elixir type system. *arXiv preprint arXiv:2306.06391*.

Elazhary, O., Storey, M.-A., Ernst, N., and Zaidman, A. (2019). Do as i do, not as i say: Do contribution guidelines match the github contribution process? In *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 286–290. IEEE.

GitHub (2020–2025c). Open source guides. `https://opensource.guide/`. Acessado em 17 de junho de 2025.

GitHub (2025a). GitHub Actions: Automate your workflow. `https://github.com/features/actions`. Acessado em: 2025-06-10.

GitHub (2025b). Github community guidelines. `https://docs.github.com/pt/site-policy/github-terms/github-community-guidelines`. Acessado em 17 de junho de 2025.

Hudak, P. (1989). Conception, evolution, and application of functional programming languages. *ACM Computing Surveys (CSUR)*, 21(3):359–411.

Prana, G. A. A., Treude, C., Thung, F., Atapattu, T., and Lo, D. (2019). Categorizing the content of github readme files. *Empirical Software Engineering*, 24:1296–1327.

Rausch, T., Hummer, W., Leitner, P., and Schulte, S. (2017). An empirical analysis of build failures in the continuous integration workflows of java-based open-source software. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, pages 345–355. IEEE.

Ray, B., Posnett, D., Filkov, V., and Devanbu, P. (2014). A large scale study of programming languages and code quality in github. In *Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering*, pages 155–165.

Steinmacher, I., Silva, M. A. G., Gerosa, M. A., and Redmiles, D. F. (2015). A systematic literature review on the barriers faced by newcomers to open source software projects. *Information and Software Technology*, 59:67–85.

Tourani, P., Adams, B., and Serebrenik, A. (2017). Code of conduct in open source projects. In *2017 IEEE 24th international conference on software analysis, evolution and reengineering (SANER)*, pages 24–33. IEEE.

Valim, J. (2023). Elixir programming language. `https://elixir-lang.org`. Accessed: 2025-06-15.

Vasilescu, B., Yu, Y., Wang, H., Devanbu, P., and Filkov, V. (2015). Quality and productivity outcomes relating to continuous integration in github. In *Proceedings of the 2015 10th joint meeting on foundations of software engineering*, pages 805–816.

Wang, T., Wang, S., and Chen, T.-H. P. (2023). Study the correlation between the readme file of github projects and their popularity. *Journal of Systems and Software*, 205:111806.