Caracteristicas de repositorios populares

Professor Danilo de Quadros Maia Filho
Alunas Ana Flávia de Carvalho Santos e Júlia Moreira Nascimento

Introdução

Este trabalho tem como objetivo mapear as principais características de sistemas open-source. Para isso, será analisada uma amostra composta pelos 1.000 repositórios do GitHub com maior número de estrelas, avaliando aspectos como frequência de lançamento de releases, participação de contribuidores externos e outras propriedades relevantes. A pesquisa busca responder às seguintes questões:

- **RQ01:** Sistemas populares são maduros/antigos?
  - Métrica: idade do repositório, calculado a partir da data de sua criação.
- **RQ02:** Sistemas populares recebem muita contribuição externa?
  - Métrica: total de pull requests aceitas.
- **RQ03:** Sistemas populares lançam releases com frequência?
  - Métrica: total de releases.
- **RQ04:** Sistemas populares são atualizados com frequência?
  - Métrica: tempo até a última atualização, calculado a partir da data de última atualização.
- **RQ05:** Sistemas populares são escritos nas linguagens mais populares?
  - Métrica: linguagem primária de cada um desses repositórios.
- **RQ06:** Sistemas populares possuem um alto percentual de issues fechadas?
  - Métrica: razão entre número de issues fechadas pelo total de issues.
- **RQ07:** Sistemas escritos em linguagens mais populares recebem mais contribuição externa, lançam mais releases e são atualizados com mais frequência?

Essas questões direcionam a investigação e permitem a formulação de hipóteses a serem testadas no decorrer do estudo. Para cada questão, são estabelecidas a hipótese nula (H_0), em que não há relação estabelecida entre as variáveis, e a hipótese alternativa (H_1), em que há uma relação que pode ser analizada entre as variáveis. As hipóteses são:

RQ01

- H_{01}: A popularidade de um sistema não está relacionada à sua idade.
- H_{11}: Sistemas populares tendem a ser mais recentes em comparação aos menos populares.

RQ02

- H_{02}: A popularidade de um sistema não influencia a quantidade de contribuições externas.
- H_{12}: Sistemas populares recebem mais contribuições externas (pull requests aceitas).

RQ03

- H_{03}: A popularidade de um sistema não está associada à frequência de lançamentos de releases.
- H_{13}: Sistemas populares lançam releases com maior frequência.

RQ04

- H_{04}:A popularidade de um sistema não está relacionada à frequência de atualização.

- H_{14}: Sistemas populares são atualizados com maior frequência.

RQ05

- H_{05}: A popularidade de um sistema não está associada à linguagem de programação utilizada.
- H_{15}:Sistemas populares são escritos predominantemente em linguagens amplamente utilizadas (ex.: Java, Python).

RQ06

- H_{06}: A popularidade de um sistema não impacta o percentual de issues fechadas.
- H_{16}: Sistemas populares apresentam um maior percentual de issues fechadas.

RQ07

- H_{07}: O uso de linguagens populares não influencia o volume de contribuições externas, a frequência de releases ou a frequência de atualizações.
- H_{17}: Sistemas escritos em linguagens populares recebem mais contribuições externas, lançam mais releases e são atualizados com maior frequência.

Metodologia

A metodologia adotada neste trabalho foi estruturada em três etapas principais:a coleta de dados, o tratamento das informações e a análise dos resultados.

Na primeira etapa, foi realizada a coleta de dados a partir da API GraphQL do GitHub, para obter informações sobre os 1.000 repositórios com maior número de estrelas. Essa consulta foi feita por meio de um script próprio, sem o uso de bibliotecas de terceiros, coletando todas as métricas necessárias para responder às questões de pesquisa definidas. Para estabelecer uma pesquisa com um tamanho de amostra válida, foi utilizada a técnica de paginação, permitindo o acesso a 10 repositórios por requisição até atingir o total de 1.000 projetos analisados. Os dados coletados foram armazenados em arquivos no formato *.csv*, de forma a possibilitar a análise dos dados.

Na segunda etapa, os dados foram tratados e organizados para permitir a avaliação das variáveis correspondentes de cada questão de pesquisa. Além disso, foram elaboradas hipóteses nulas (H_0) e alternativas (H_1) para cada questão, de forma a estruturar a investigação de maneira mais formal.

Na terceira etapa, foi feita a análise exploratória dos resultados, associando as métricas obtidas às questões de pesquisa (RQ01–RQ07). A análise buscou identificar se características como maturidade do repositório, frequência de contribuições externas, quantidade de releases, atualização recente, percentual de issues fechadas e uso de linguagens populares apresentam relação com o nível de popularidade dos sistemas.

Por fim, como extensão da análise, os resultados das RQs 02, 03 e 04 foram segmentados por linguagem de programação, permitindo investigar se repositórios escritos em linguagens mais populares apresentam comportamentos distintos em relação a contribuições externas, lançamentos de releases e frequência de atualizações.

Resultados

RQ01: Sistemas populares são maduros/antigos?

A análise da idade dos 1.000 repositórios mais populares revelou que a média é de 8,07 anos, enquanto a mediana é de 8 anos, como pode ser observado no gráfico abaixo. Esses valores indicam que a maioria dos sistemas populares não é necessariamente antiga, sugerindo que projetos recentes também podem atingir alta popularidade.

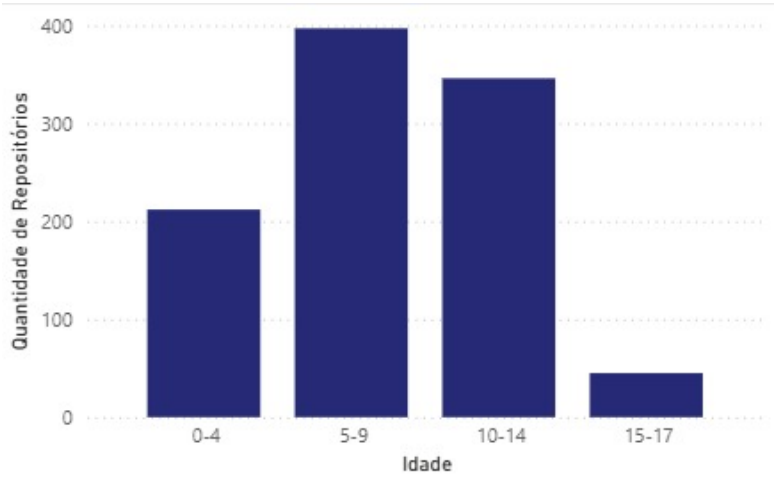

Figure 1: Gráfico da relação da quantidade de repositórios por idade.

O estudo com o escopo delimitado aos 10 repositórios mais famosos, foi observado uma idade média de 9,2 anos, o que é maior que a média dos 1.000 repositórios mais famosos. Esses dados reforçam que repositórios mais famosos já possuem certo nível de maturidade.

Posição	Repositório	Idade
1	freeCodeCamp/freeCodeCamp	11
2	codecrafters-io/build-your-own-x	7
3	sindresorhus/awesome	11
4	EbookFoundation/free-programming-books	12
5	public-apis/public-apis	9
6	kamranahmedse/developer-roadmap	8
7	jwasham/coding-interview-university	9
8	donnemartin/system-design-primer	8
9	996icu/996.ICU	6
10	vinta/awesome-python	11

Figure 2: Lista dos 10 repositórios mais famosos por idade.

RQ02: Sistemas populares recebem muita contribuição externa?

A análise do total de pull requests (PRs) aceitos nos 1.000 repositórios mais populares mostrou que a média é de 3.590,36 PRs, enquanto a mediana é de 702 PRs. O primeiro quartil (Q1), que indica o valor abaixo do qual se encontram 25% dos repositórios, é de 171 PRs, e o terceiro quartil (Q3), que representa o valor abaixo do qual se encontram 75% dos repositórios, é de 2.853 PRs. O repositório com menos contribuições externas possui 0 PRs aceitos, enquanto o mais colaborativo recebeu 86.068 PRs.

Posição	Repositório	PR's aceitos
1	freeCodeCamp/freeCodeCamp	25720
2	codecrafters-io/build-your-own-x	142
3	sindresorhus/awesome	679
4	EbookFoundation/free-programming-books	6820
5	public-apis/public-apis	1872
6	kamranahmedse/developer-roadmap	3690
7	jwasham/coding-interview-university	415
8	donnemartin/system-design-primer	200
9	996icu/996.ICU	1067
10	vinta/awesome-python	544

Figure 3: Lista dos 10 repositórios mais famosos por quantidade de PR.

Esses resultados revelam que, apesar da média elevada devido a alguns repositórios com número extremamente alto de contribuições, a mediana e os quartis mostram que a maioria dos sistemas populares recebe contribuições externas de forma mais moderada, evidenciando uma distribuição assimétrica à direita, com poucos outliers de valores muito altos.

RQ03: Sistemas populares lançam releases com frequência?

A análise do total de releases dos 1.000 repositórios mais populares revelou uma média de 109,42 releases, enquanto a mediana é de 35 releases. O primeiro quartil (Q1) é 0, indicando que 25% dos repositórios não lançaram nenhuma release, e o terceiro quartil (Q3) é 127, mostrando que 75% dos repositórios lançaram até 127 releases. O número mínimo de releases é 0, enquanto o máximo alcança 1.000 releases

Posição	Repositório	Releases
1	freeCodeCamp/freeCodeCamp	0
2	codecrafters-io/build-your-own-x	0
3	sindresorhus/awesome	0
4	EbookFoundation/free-programming-books	0
5	public-apis/public-apis	0
6	kamranahmedse/developer-roadmap	1
7	jwasham/coding-interview-university	0
8	donnemartin/system-design-primer	0
9	996icu/996.ICU	0
10	vinta/awesome-python	0

Figure 4: Lista dos 10 repositórios mais famosos por número de releases.

Esses dados evidenciam que, embora a média seja relativamente alta devido a alguns repositórios com número muito grande de releases, a mediana e os quartis indicam que a maioria dos sistemas populares lança releases de forma moderada, refletindo uma distribuição assimétrica com alguns outliers bastante expressivos.

RQ04: Sistemas populares são atualizados com frequência?

A análise do tempo desde a última atualização dos 1.000 repositórios mais populares revelou que a média é de 112,85 dias, enquanto a mediana é de 9 dias. O primeiro quartil (Q1) é 6 dias, indicando que 25% dos repositórios foram atualizados há menos de 6 dias, e o terceiro quartil (Q3) é 109 dias, mostrando que 75% dos repositórios foram atualizados há até 109 dias. Esses resultados evidenciam que, embora a média seja elevada devido a alguns repositórios que não foram atualizados recentemente, a mediana e os quartis indicam que a maioria dos sistemas populares recebe atualizações frequentes, caracterizando uma distribuição assimétrica à direita.

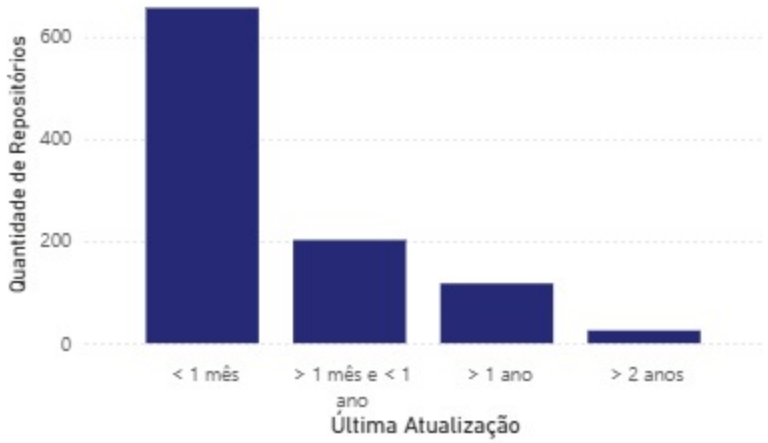

Figure 5: Gráfico relação quantidade de relesases e tempo da última atualização.

Com base nos dados dos 10 repositórios mais populares, podemos observar que há grande variação no tempo desde a última atualização: enquanto alguns projetos, como *kamranahmedse/developer-roadmap* e *freeCodeCamp/freeCodeCamp*, foram atualizados recentemente (em menos de 7 dias), outros, como *996icu/996.ICU* e *jwasham/coding-interview-university*, não recebem atualizações há vários meses. Isso evidencia que, mesmo entre os repositórios mais famosos, existe diversidade na frequência de manutenção e atualização.

Posição	Repositório	lastUpdate
1	freeCodeCamp/freeCodeCamp	6,13
2	codecrafters-io/build-your-own-x	27,92
3	sindresorhus/awesome	39,02
4	EbookFoundation/free-programming-books	9,17
5	public-apis/public-apis	98,13
6	kamranahmedse/developer-roadmap	5,86
7	jwasham/coding-interview-university	263,81
8	donnemartin/system-design-primer	97,33
9	996icu/996.ICU	327,28
10	vinta/awesome-python	40,11

Figure 6: Lista dos 10 repositórios mais famosos por tempo do último update.

RQ05: Sistemas populares são escritos nas linguagens mais populares?

A análise mostra que as linguagens Python, TypeScript e JavaScript concentram a maior parte dos repositórios populares, com destaque para o Python, presente em cerca de 22,5% dos projetos. Os repositórios mais reconhecidos também refletem essa tendência, embora haja casos em que a linguagem principal é N/A ou até menos comuns, como Markdown. Isso indica que, apesar de existir uma forte relação entre linguagens populares e sistemas populares, a popularidade de um projeto não depende exclusivamente da linguagem utilizada, mas também de fatores como proposta, utilidade e apoio da comunidade.

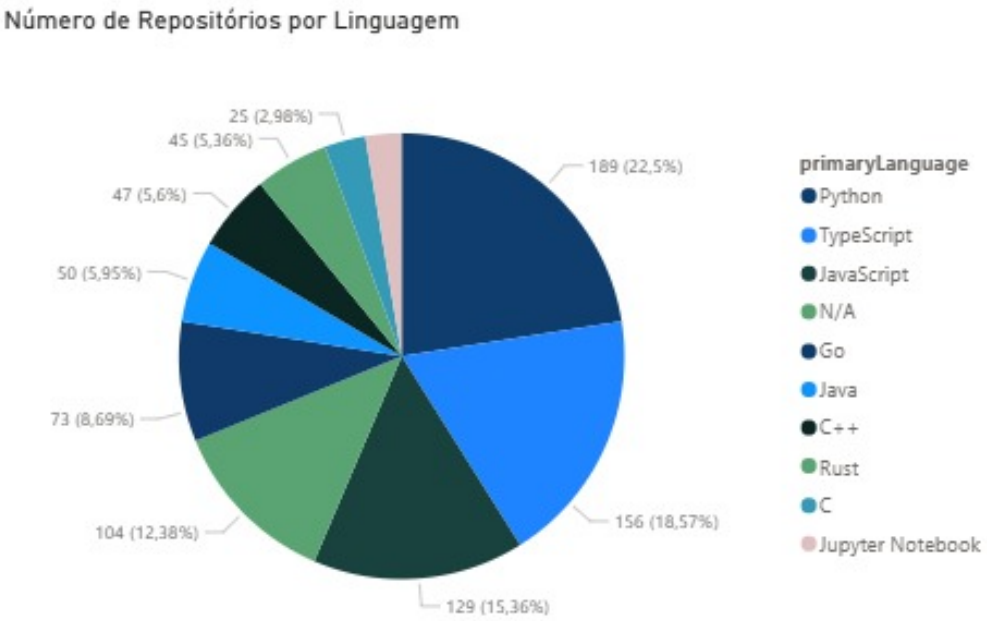

Figure 7: Gráfico de pizza com as linguagens mais famosas.

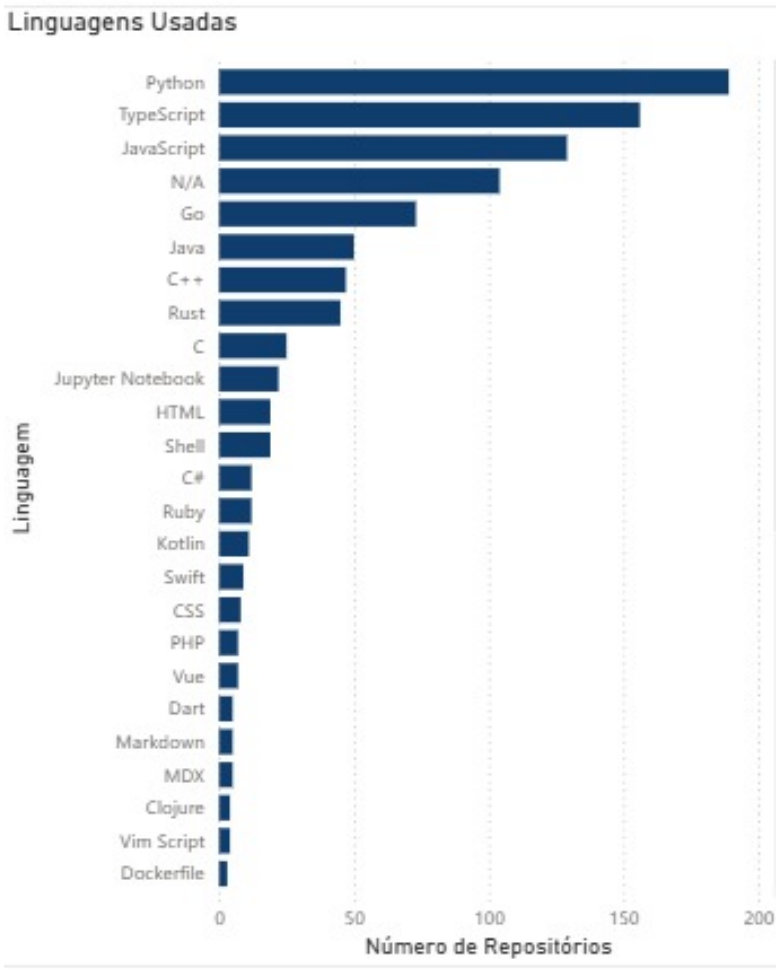

Figure 8: Gráfico de barras com análise das linguagens mais usadas.

Posição	Repositório	primaryLanguage
1	freeCodeCamp/freeCodeCamp	TypeScript
2	codecrafters-io/build-your-own-x	Markdown
3	sindresorhus/awesome	N/A
4	EbookFoundation/free-programming-books	Python
5	public-apis/public-apis	Python
6	kamranahmedse/developer-roadmap	TypeScript
7	jwasham/coding-interview-university	N/A
8	donnemartin/system-design-primer	Python
9	996icu/996.ICU	N/A
10	vinta/awesome-python	Python

Figure 9: Lista dos 10 repositórios mais famosos e sua linguagem.

RQ06: Sistemas populares possuem um alto percentual de issues fechadas?

Os resultados mostram que sistemas populares tendem a apresentar alto percentual de issues fechadas, como freeCodeCamp (99%), *EbookFoundation/free-programming-books* (97%) e *public-apis* (98,8%), indicando manutenção ativa e engajamento da comunidade. Contudo, há exceções relevantes: repositórios como *system-design-primer* (26,8%) e project-based-learning (37,8%) possuem baixo índice de fechamento, mesmo sendo bastante populares. Isso evidencia que, embora exista uma forte relação entre popularidade e gestão ativa de issues, ela não é absoluta, variando conforme o tipo de projeto e a organização da comunidade.

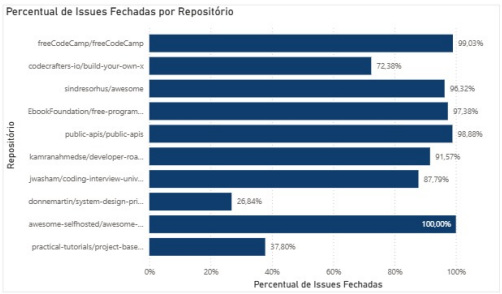

Figure 10: Gráfico de barras com percentual de issues fechadas por repositório.

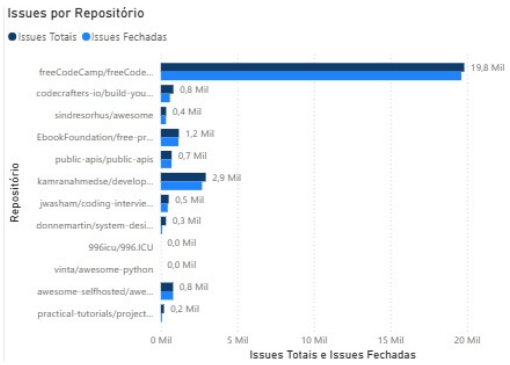

Figure 11: Gráfico de barras com relação de issues totais e issues abertas.

RQ07: Sistemas escritos em linguagens mais populares recebem mais contribuição externa, lançam mais releases e são atualizados com mais frequência?

Os resultados indicam que linguagens populares não são necessariamente as que mais recebem contribuições, lançam releases ou mantêm atualizações frequentes. Em pull requests aceitos, por exemplo, linguagens menos usuais como LLVM e Julia lideram, enquanto Python e JavaScript ficam atrás. No caso das releases, destacam-se PHP e TypeScript, mostrando cadência alta de entregas, mas não todas as linguagens populares seguem esse padrão. Já na frequência de atualização, projetos em Julia, LLVM e V apresentam intervalos curtos, contrastando com linguagens como Rust, onde há grandes períodos sem manutenção. Assim, percebe-se que esses fatores dependem mais do perfil do projeto e da comunidade do que da popularidade da linguagem em si.

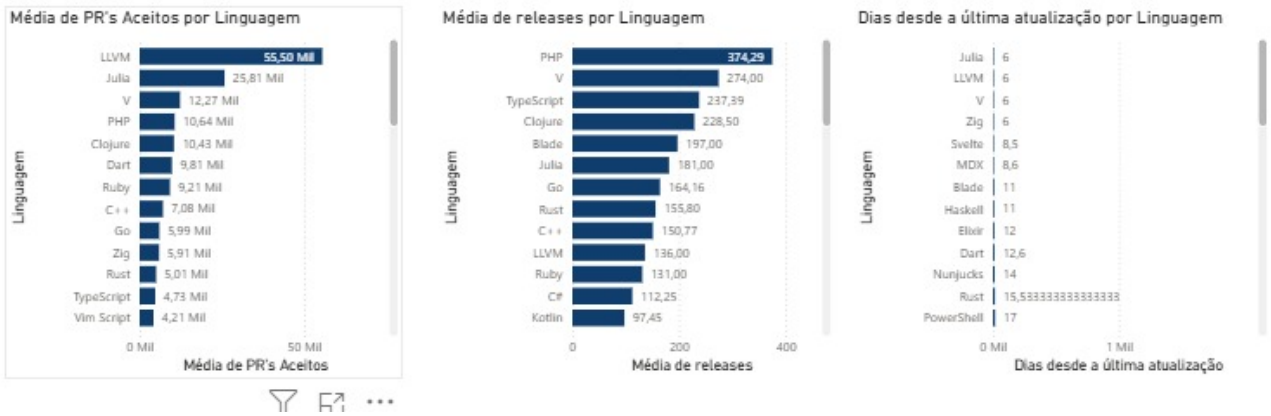

Figure 12: Gráficos de barra comparando médias de release por linguagens, média de PR's por linguagem e dias desde a última atualização por linguagem.

Discussão

Os resultados indicam que muitos repositórios populares no GitHub têm em média cerca de 8 anos de idade (mediana de 8 anos), porém existe variação significativa, com alguns dos mais famosos chegando a cerca de 9,2

anos, mostrando que tanto projetos maduros quanto mais recentes podem alcançar grande visibilidade [1]. Esse achado contraria a suposição de que apenas repositórios antigos se tornam populares—na verdade, relevância, proposta, qualidade e contexto social parecem ter papel mais decisivo do que simplesmente a longevidade.

Quanto às contribuições externas, a média de pull requests aceitos nos repositórios mais populares chegou a 3 590, enquanto a mediana foi de 702, e existiram casos extremos com até 86 068 PRs. Isso sugere uma distribuição assimétrica: poucos projetos concentram altíssimos volumes de colaboração, mas a maioria ainda recebe contribuições ajenas, ainda que em volume moderado [1]. Em paralelo, um estudo de 2022 sobre a correlação entre conteúdo no README e popularidade reforça que repositórios que fornecem diretivas claras para contribuição, com guias e referências, associam-se a níveis maiores de engajamento e, portanto, provavelmente atraem mais contribuições externas [2].

Em relação à frequência de releases, descobriu-se que a média por repositório popular foi de cerca de 109 lançamentos, enquanto a mediana ficou em torno de 35. Ainda, 25% dos repositórios não fizeram nenhum release, e o máximo ultrapassou mil releases [1]. Isso mostra que, mesmo entre projetos muito populares, muitos seguem um ritmo moderado de lançamentos sem necessariamente focar em releases frequentes.

Sobre atualizações, a média do tempo desde a última atualização foi de cerca de 113 dias, com mediana de apenas 9 dias — 25% dos repositórios populares foram atualizados nos últimos 6 dias, e 75% nos últimos 109 dias [1]. Ou seja, apesar da média elevada devido a alguns projetos pouco ativados, a maioria recebe manutenção recente, o que evidencia boa retenção de atividade e interesse.

No que diz respeito ao uso de linguagens de programação populares, há predominância de Python, TypeScript e JavaScript entre os repositórios de maior destaque; Python, por exemplo, aparece em cerca de 22,5% dos casos [1]. Ainda assim, a linguagem em si não determina a popularidade: fatores como utilidade, comunidade, documentação e exposição exercem papel substancial.

Quando se analisa o percentual de issues fechadas, projetos populares geralmente apresentam índices elevados — freeCodeCamp, por exemplo, alcançou 99%, free-programming-books 97%, public-apis 98,8% [1]. Contudo, há exceções relevantes como system-design-primer e project-based-learning, com taxas bem mais baixas (26,8% e 37,8%, respectivamente), evidenciando que o fechamento de issues está correlacionado com um conjunto de fatores, como o tipo de projeto e as práticas de gestão da comunidade.

Por fim, examinando se repositórios em linguagens populares recebem mais contribuições externas, mais releases e atualizações mais frequentes, os resultados mostram que essa relação não é consistente. Linguagens menos comuns, como LLVM e Julia, às vezes lideram em PRs aceitos; PHP e TypeScript se destacam em número de releases; Julia, LLVM e V mostram atualizações mais regulares, enquanto Rust tem intervalos de manutenção maiores [1]. Portanto, o comportamento dos repositórios está mais associado ao perfil do projeto e da comunidade, e não apenas à linguagem utilizada.

Por fim, examinando se repositórios em linguagens populares recebem mais contribuições externas, mais releases e atualizações mais frequentes, os resultados mostram que essa relação não é consistente. Linguagens menos comuns, como LLVM e Julia, às vezes lideram em PRs aceitos; PHP e TypeScript se destacam em número de releases; Julia, LLVM e V mostram atualizações mais regulares, enquanto Rust tem intervalos de manutenção maiores [1]. Portanto, o comportamento dos repositórios está mais associado ao perfil do projeto e da comunidade, e não apenas à linguagem utilizada.

Complementando essa análise, outro estudo específico sobre repositórios acadêmicos de inteligência artificial evidenciou que a presença de elementos como licença explícita, links para outros projetos e imagens no README exerce forte influência na popularidade, independentemente da linguagem de programação usada [3]. Isso reforça a ideia de que fatores sociais e de comunicação são decisivos para a atratividade de projetos open source.

Conclusão

Com base nos resultados analisados, é possível concluir que a popularidade de repositórios no GitHub não depende exclusivamente de fatores como idade, linguagem de programação ou frequência de lançamentos. Embora muitos dos projetos mais conhecidos apresentem certo nível de maturidade, há também casos recentes que rapidamente conquistam grande visibilidade, mostrando que a relevância do problema resolvido e o engajamento da comunidade são elementos-chave. Isso indica que a longevidade contribui para consolidar um projeto, mas não é condição necessária para alcançar destaque.

Outro ponto importante é que a dinâmica de colaboração e manutenção varia significativamente entre os repositórios. A aceitação de contribuições externas, a frequência de releases e a taxa de issues fechadas revelam que há um núcleo de projetos altamente ativos, enquanto outros mantêm popularidade mesmo com níveis mais baixos de atualização e governança comunitária. Isso reforça que a qualidade da documentação, clareza no processo de contribuição e alinhamento com as necessidades dos usuários podem ser tão ou mais determinantes do que métricas quantitativas tradicionais de atividade.

Por fim, a análise mostrou que o uso de linguagens populares como Python, JavaScript e TypeScript aparece com frequência entre os projetos de maior destaque, mas não garante por si só maior engajamento ou manutenção contínua. Repositórios escritos em linguagens menos comuns também podem se destacar quando oferecem soluções inovadoras e úteis para a comunidade. Dessa forma, pode-se afirmar que a popularidade em ecossistemas open source é multifatorial, resultado de uma combinação entre proposta do projeto, qualidade de documentação, facilidade de contribuição e mobilização da comunidade de usuários e desenvolvedores.

References

[1] Bruna Amorim dos Santos. Adoção de métricas ágeis integradas com a plataforma github. Master's thesis, ISCTE-Instituto Universitario de Lisboa (Portugal), 2022.

[2] Yuanrui Fan, Xin Xia, David Lo, Ahmed E Hassan, and Shanping Li. What makes a popular academic ai repository? *Empirical Software Engineering*, 26(1):2, 2021.

[3] Akhila Sri Manasa Venigalla and Sridhar Chimalakonda. An empirical study on correlation between readme content and project popularity. *arXiv e-prints*, pages arXiv–2206, 2022.