PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
ENGENHARIA DE SOFTWARE - CAMPUS LOURDES
LABORATÓRIO DE EXPERIMENTAÇÃO DE SOFTWARE

Ana Carolina Corrêa, Caio Elias,
Henrique Diniz e Maria Clara Santos

Análise de Características de Repositórios Populares do GitHub

04 de setembro de 2024
Belo Horizonte

LISTA DE FIGURAS

SUMÁRIO

1. Introdução

Este estudo tem como objetivo analisar as principais características dos repositórios populares no GitHub, buscando identificar padrões em aspectos como idade, contribuição externa, frequência de releases, atualizações, linguagem de programação utilizada e manutenção de issues. Para orientar essa análise, utilizamos como base as seguintes questões de pesquisa:

1. Sistemas populares são maduros/antigos?
2. Sistemas populares recebem muita contribuição externa?
3. Sistemas populares lançam releases com frequência?
4. Sistemas populares são atualizados com frequência?
5. Sistemas populares são escritos nas linguagens mais populares?
6. Sistemas populares possuem um alto percentual de issues fechadas?

Para responder a essas perguntas, coletamos dados de 1000 repositórios populares utilizando a API GraphQL do GitHub. Este relatório apresenta os resultados dessa investigação, acompanhados de hipóteses iniciais, metodologia empregada e uma discussão sobre os achados mais relevantes.

2. Hipóteses

- **Maturidade dos Sistemas (RQ 01):**

Acreditamos que os repositórios populares tendem a ser mais antigos, o que sugere que a popularidade pode estar associada a uma longa história de desenvolvimento e melhorias contínuas. Espera-se que a idade média desses repositórios seja superior a cinco anos.

- **Contribuição Externa (RQ 02):**

Presumimos que repositórios populares recebem um número significativo de contribuições externas na forma de pull requests aceitas. Isso pode indicar que a comunidade em torno desses projetos é ativa e engajada, contribuindo para sua popularidade.

- **Frequência de Releases (RQ 03):**

Supomos que os repositórios populares lançam releases com frequência, refletindo um ciclo de desenvolvimento ágil e um esforço contínuo para aprimorar e atualizar o software. Projetos populares podem ser caracterizados por uma média de pelo menos uma release a cada três meses.

- **Frequência de Atualizações (RQ 04):**

Esperamos que repositórios populares sejam frequentemente atualizados, sugerindo uma manutenção ativa. Isso pode se manifestar em uma média de atualizações a cada mês ou em intervalos ainda menores.

- **Linguagem de Programação (RQ 05):**

Acreditamos que a maioria dos repositórios populares são escritos nas linguagens de programação mais utilizadas globalmente, como JavaScript, Python, e Java. Essas linguagens podem contribuir para a popularidade dos projetos devido à sua ampla base de usuários.

- **Manutenção de Issues (RQ 06):**

Supomos que repositórios populares têm um alto percentual de issues fechadas, indicando uma boa gestão de problemas e solicitações. Acreditamos que mais de 75% das issues em repositórios populares estejam fechadas.

3. Metodologia

Para realizar a análise dos repositórios populares no GitHub, seguimos uma abordagem sistemática que envolveu a coleta e processamento de dados utilizando a API GraphQL do GitHub. Abaixo, descrevemos as etapas principais do processo:

1. Coleta de Dados:

Seleção dos Repositórios: Utilizamos a API GraphQL do GitHub para buscar e coletar dados de 1000 repositórios populares. A popularidade foi definida como repositórios com um número significativo de estrelas atribuídas por usuários. Para garantir uma amostra representativa, buscamos repositórios com mais de uma estrela, aplicando filtros específicos na query.

Paginação: Devido às limitações da API, que restringe o número de resultados retornados por consulta, implementamos a técnica de paginação para garantir que todos os 1000 repositórios fossem coletados. Em cada consulta, um número limitado de repositórios (20 por vez) foi coletado, utilizando o cursor fornecido pela API para continuar a busca a partir do último repositório retornado.

Atributos Coletados: Para cada repositório, coletamos os seguintes dados:

- Idade do Repositório: Data de criação do repositório.
- Contribuição Externa: Número total de pull requests aceitas.
- Frequência de Releases: Número total de releases publicadas. ▪ Frequência de Atualizações: Data da última atualização do repositório.
- Linguagem de Programação: Linguagem principal utilizada no repositório.
- Manutenção de Issues: Número total de issues abertas e fechadas.

2. Processamento e Análise dos Dados:

Cálculo de Métricas: Para cada questão de pesquisa, calculamos a mediana das métricas coletadas. Por exemplo, a idade dos repositórios foi calculada em anos, enquanto a frequência de releases e a contribuição externa foram analisadas em termos absolutos e relativos.

Análise de Linguagens de Programação: Classificamos e contamos a frequência das linguagens de programação utilizadas nos repositórios populares, identificando quais são as mais comuns entre os projetos analisados.

Razão de Issues Fechadas: Calculamos a razão entre o número de issues fechadas e o total de issues para avaliar a eficácia da manutenção nos repositórios.

3. Ferramentas Utilizadas:

Python: A coleta e análise dos dados foram realizadas utilizando scripts em Python. Bibliotecas como requests foram usadas para interagir com a API do GitHub, enquanto pandas e statistics foram empregadas para processamento e análise dos dados.

API GraphQL do GitHub: A API GraphQL foi escolhida por sua flexibilidade e capacidade de retornar exatamente os dados necessários, facilitando a coleta de informações precisas e detalhadas dos repositórios.

4. Salvamento e Organização dos Dados:

Todos os dados coletados foram armazenados em arquivos CSV para facilitar a manipulação e análise subsequente. Os resultados intermediários e as análises foram organizados em tabelas, que são apresentadas nas seções seguintes deste relatório.

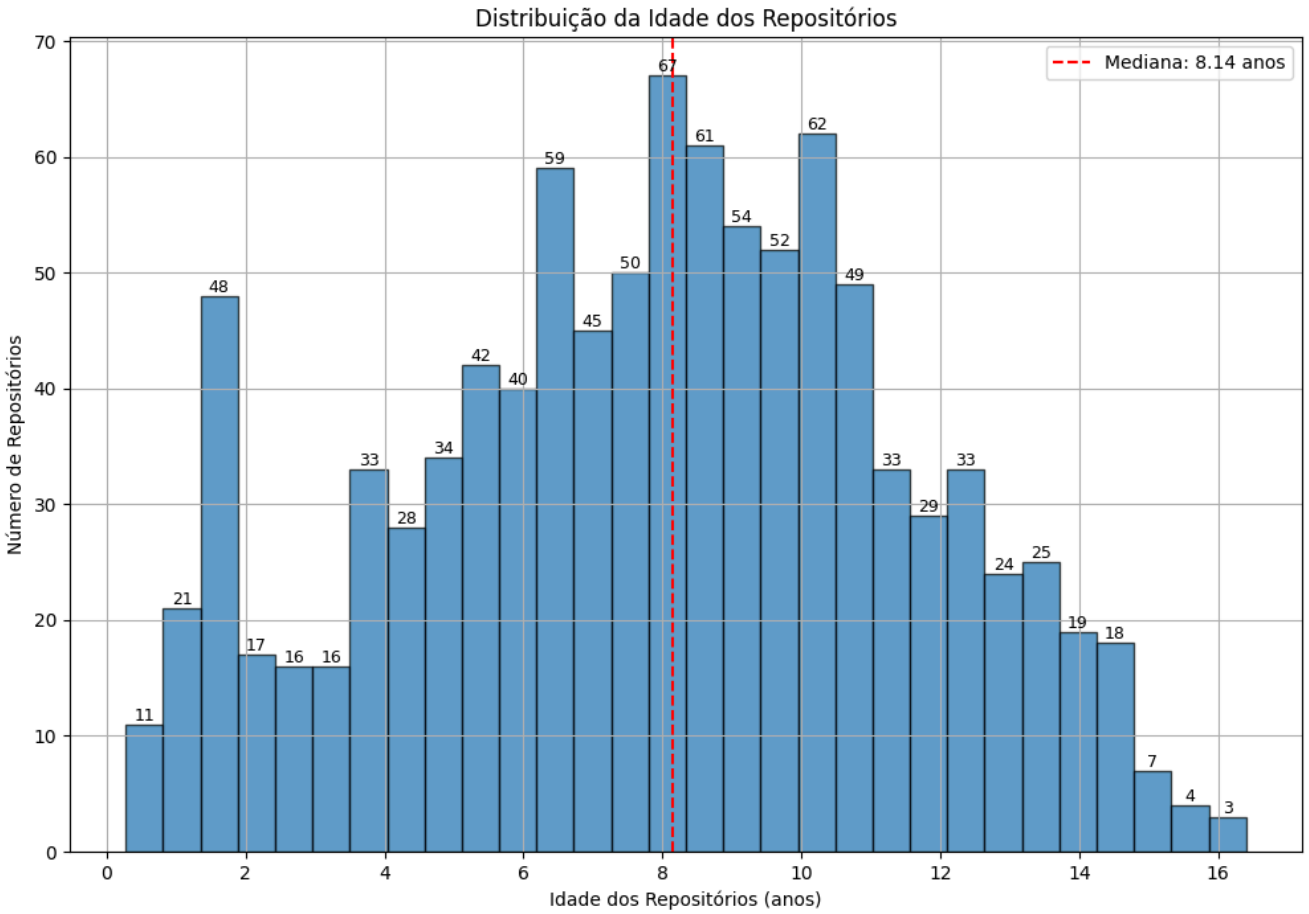
4. Resultados

A análise dos 1000 repositórios populares do GitHub gerou insights valiosos sobre as características desses projetos. A seguir, apresentamos os resultados obtidos para cada uma das questões de pesquisa formuladas.

RQ 01: Sistemas populares são maduros/antigos?

A mediana da idade dos repositórios analisados foi de 8,1 anos. Este resultado sugere que os repositórios populares tendem a ser relativamente maduros, com uma história de desenvolvimento que se estende por vários anos. Isso corrobora a hipótese de que projetos com maior tempo de existência têm mais oportunidades de se tornarem populares, devido ao tempo para amadurecer e ganhar tração na comunidade de desenvolvedores.

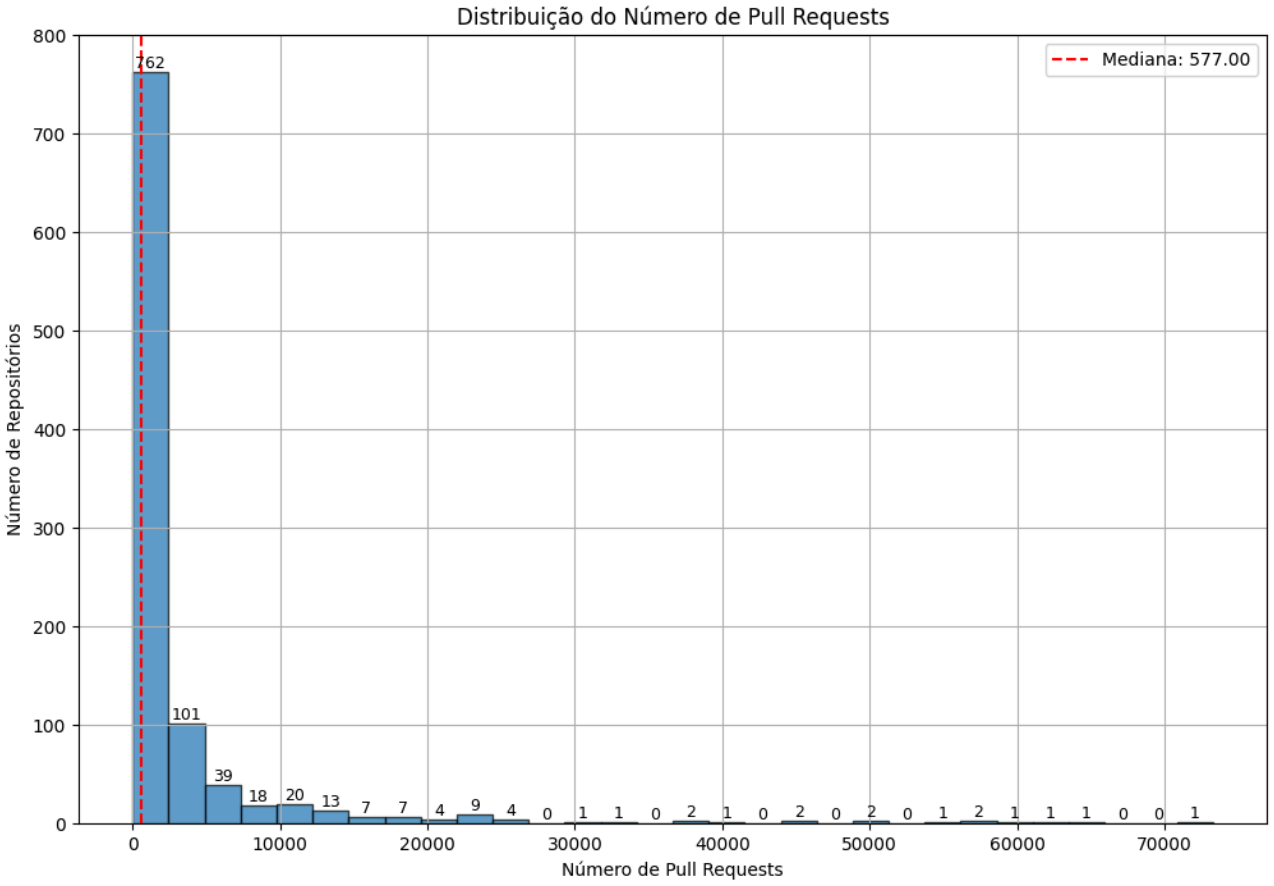
Figura 01 - Distribuição da idade dos repositórios



RQ 02. Sistemas populares recebem muita contribuição externa?

A mediana do total de pull requests aceitas foi de 577. Esse número indica um alto nível de contribuição externa, sugerindo que os repositórios populares atraem muitas colaborações da comunidade. Esse dado reforça a ideia de que projetos populares tendem a ser sustentados por uma comunidade ativa e engajada.

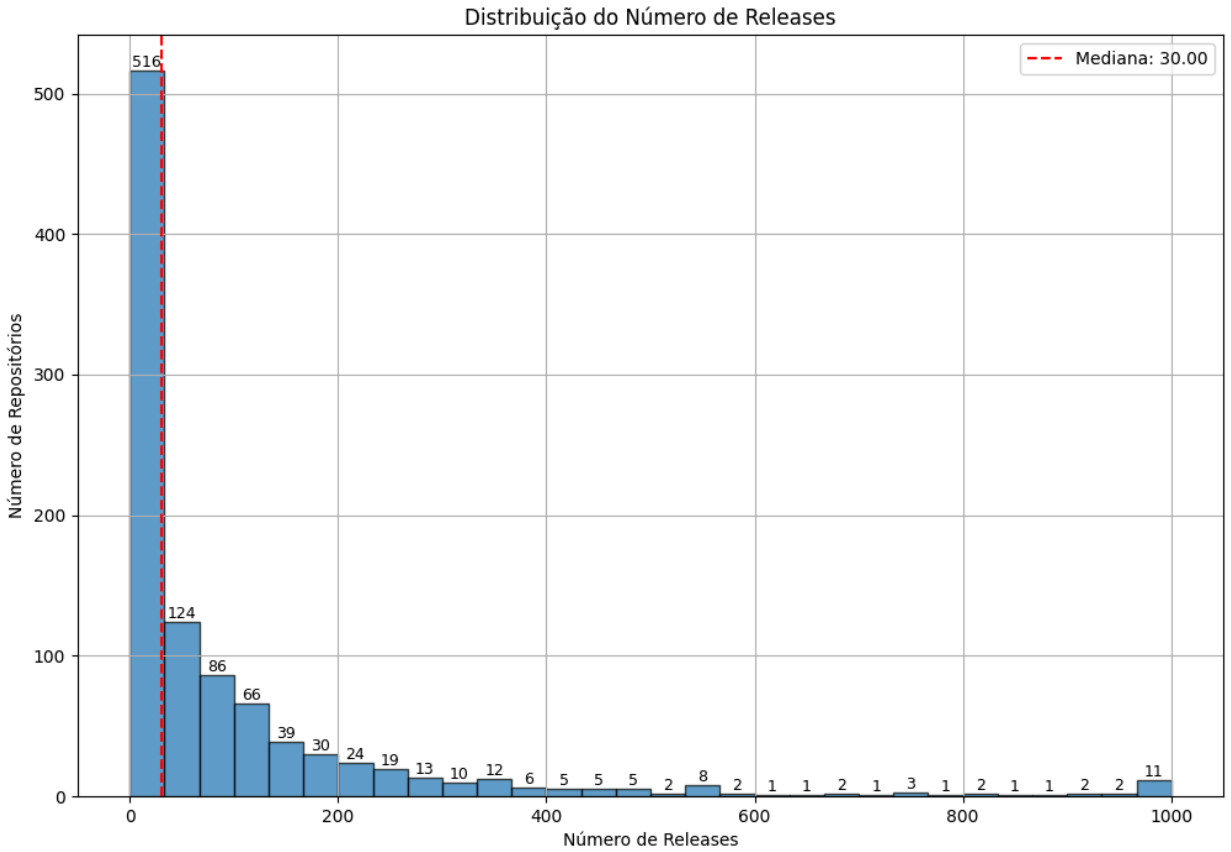
Figura 02 - Distribuição do número de pull requests



RQ 03. Sistemas populares lançam releases com frequência?

A mediana do total de releases foi de 30. Isso sugere que os repositórios populares mantêm um ciclo de lançamento de novas versões razoavelmente frequente, o que pode ser um indicador de um desenvolvimento contínuo e um compromisso com a melhoria e a manutenção do software.

Figura 03 - Distribuição do número de releases



RQ 04. Sistemas populares são atualizados com frequência?

A mediana do tempo desde a última atualização foi de 0 dias, indicando que muitos dos repositórios populares são atualizados constantemente. Esse dado é consistente com a prática de desenvolvimento ágil e contínuo, onde as atualizações e melhorias são feitas regularmente, mantendo o software relevante e em constante evolução.

RQ 05. Sistemas populares são escritos nas linguagens mais populares?

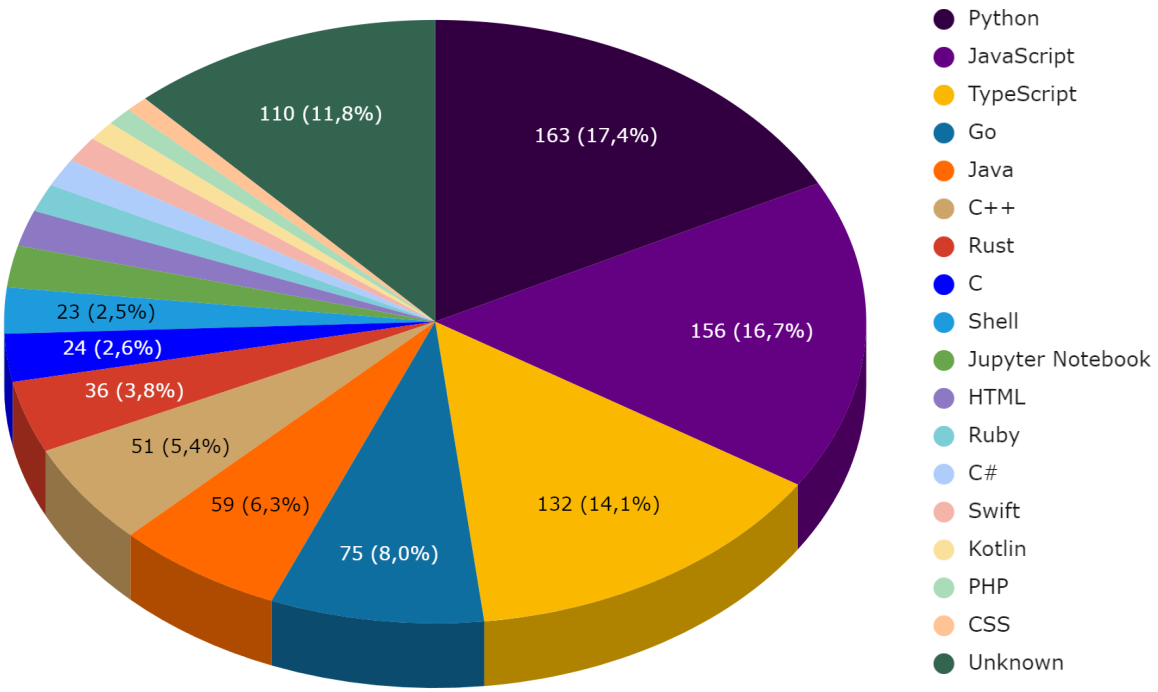
As linguagens de programação mais comuns entre os repositórios populares analisados foram:

1. **Python**: 163 repositórios
2. **JavaScript**: 156 repositórios
3. **TypeScript**: 132 repositórios
4. **Go**: 75 repositórios
5. **Java**: 59 repositórios

6. **C++**: 51 repositórios
7. **Rust**: 36 repositórios
8. **C**: 24 repositórios
9. **Shell**: 23 repositórios
10. **Jupyter Notebook**: 21 repositórios
11. **HTML**: 18 repositórios
12. **Ruby**: 14 repositórios
13. **C#**: 14 repositórios
14. **Swift**: 13 repositórios
15. **Kotlin**: 10 repositórios
16. **PHP**: 9 repositórios
17. **CSS**: 8 repositórios

Esses resultados mostram que linguagens amplamente adotadas e versáteis, como Python, JavaScript e TypeScript, dominam entre os projetos populares. O número significativo de repositórios marcados como "Unknown" (110) pode refletir a presença de projetos que não especificam uma linguagem principal ou que utilizam uma variedade de linguagens.

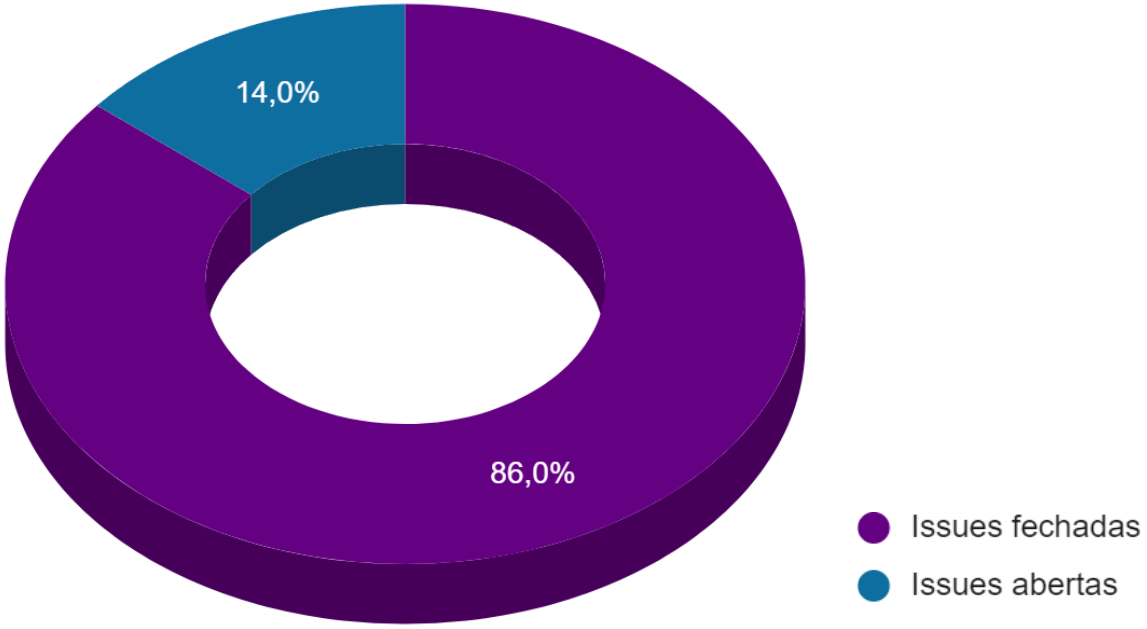
Figura 04 - Linguagens mais usadas



RQ 06. Sistemas populares possuem um alto percentual de issues fechadas?

A mediana da razão de issues fechadas foi de 0,86. Isso indica que, em média, 86% das issues nos repositórios populares são fechadas, sugerindo uma boa gestão e manutenção dos problemas relatados. Esse alto percentual de issues fechadas pode ser visto como um sinal de um projeto bem mantido e com uma equipe de desenvolvedores responsiva.

Figura 05 - Issues



5. Discussão

Os resultados obtidos confirmam muitas de nossas expectativas iniciais. Em primeiro lugar, a idade média dos repositórios (de 8,1 anos), sugere que a maturidade é um fator importante para a popularidade.

Além disso, a alta mediana de pull requests aceitas (577) e releases (30) indica que esses repositórios não apenas atraem colaborações externas significativas, mas também mantêm um ciclo de desenvolvimento contínuo. Isso é essencial para manter o software atualizado e funcional, o que por sua vez contribui para a sua popularidade.

Outro ponto relevante é a constante atualização dos repositórios, refletida na

mediana de 0 dias desde a última atualização. Essa assiduidade na prática de desenvolvimento ágil é crucial para a manutenção do interesse da comunidade e da relevância do projeto no GitHub.

Em relação às linguagens de programação, vemos que Python, JavaScript e TypeScript dominam os repositórios populares, o que era esperado devido à sua versatilidade e ampla adoção. A presença significativa de repositórios marcados como "Unknown" (110) pode indicar projetos que utilizam múltiplas linguagens ou que não especificam claramente a linguagem principal, refletindo a diversidade tecnológica desses projetos.

Em resumo, nossos achados sublinham a importância de fatores como maturidade, colaboração ativa, e manutenção rigorosa para o sucesso de repositórios no GitHub. No entanto, é importante reconhecer que aspectos como visibilidade e
utilidade também desempenham papéis cruciais na determinação da popularidade, embora não tenham sido o foco desta análise.

6. Conclusão

A análise dos 1000 repositórios populares no GitHub mostrou que a maturidade dos projetos, a colaboração ativa da comunidade e a manutenção contínua são fatores determinantes para alcançar e manter a popularidade na plataforma. Projetos mais antigos, com ciclos de desenvolvimento regulares e um número significativo de contribuições externas, tendem a se destacar entre os demais.

Além disso, uma gestão eficaz de issues e a prática constante de atualizações reforçam a importância de uma manutenção constante para a longevidade e a relevância desses repositórios. Essas observações oferecem uma visão clara dos elementos essenciais para o sucesso de projetos open-source, destacando a importância de práticas de desenvolvimento bem estabelecidas e de uma comunidade engajada.