Relatório de Análise Empírica: Características de Repositórios Open-Source Populares no GitHub

CURSO: ENGENHARIA DE SOFTWARE

DISCIPLINA: LABORATÓRIO DE EXPERIMENTAÇÃO DE SOFTWARE

PROFESSOR: João Paulo Carneiro Aramuni

AUTORES:

- Gabriel Afonso Infante Vieira
- Rafael de Paiva Gomes
- Rafaella Cristina de Sousa Sacramento

1. Introdução

1.1. Contextualização

O desenvolvimento de software contemporâneo é profundamente influenciado pelo modelo de código aberto (open-source). Este paradigma representa uma abordagem colaborativa para a criação de tecnologia, onde o código-fonte é disponibilizado publicamente, permitindo que uma comunidade global de desenvolvedores contribua para a sua evolução. No epicentro deste ecossistema está o GitHub, a principal plataforma para o desenvolvimento e a análise de projetos de software, hospedando milhões de repositórios e desenvolvedores.

1.2. Definição do Problema e Justificativa

Dentro do GitHub, a "popularidade" de um repositório é frequentemente mensurada pelo seu número de "estrelas" (stars), um indicador de visibilidade e interesse da comunidade. Embora seja uma métrica amplamente utilizada, as características intrínsecas que definem e sustentam a popularidade desses projetos de elite não são completamente compreendidas de forma empírica.

Este estudo se justifica pela necessidade de fornecer um perfil quantitativo dos projetos de código aberto mais populares. A compreensão dessas características pode oferecer um roteiro para novos projetos sobre como cultivar comunidades ativas, além de fornecer um

retrato das tendências e da evolução do cenário de software de código aberto.

1.3. Questões de Pesquisa e Hipóteses Iniciais

Para guiar esta investigação, foram formuladas as seguintes questões de pesquisa (RQs), cada uma acompanhada de uma hipótese informal:

- **RQ01: Sistemas populares são maduros/antigos?**
  - **Hipótese (H1):** Sim. Espera-se que a popularidade se construa com o tempo. Projetos mais antigos tendem a acumular maior popularidade devido à sua maturidade e estabilidade.
- **RQ02: Sistemas populares recebem muita contribuição externa?**
  - **Hipótese (H2):** Sim. Espera-se que repositórios populares recebem muitas contribuições externas, o que impulsiona sua evolução.
- **RQ03: Sistemas populares lançam releases com frequência?**
  - **Hipótese (H3):** Sim. Espera-se que projetos com mais *releases* tendem a demonstrar maior organização e planejamento no ciclo de desenvolvimento.
- **RQ04: Sistemas populares são atualizados com frequência?**
  - **Hipótese (H4):** Sim. Espera-se que a manutenção contínua é um fator determinante para a permanência de um projeto entre os mais populares.
- **RQ05: Sistemas populares são escritos nas linguagens mais populares?**
  - **Hipótese (H5):** Sim. Espera-se que linguagens mais populares concentram naturalmente maior número de projetos relevantes no GitHub.
- **RQ06: Sistemas populares possuem um alto percentual de issues fechadas?**
  - **Hipótese (H6):** Sim. Espera-se que projetos que mantêm alta taxa de fechamento de *issues* transmitem maior confiabilidade à comunidade.

2. Metodologia

2.1. Fonte de Dados e Amostragem

A fonte de dados foi a API GraphQL v4 do GitHub. A amostra consiste nos 1.000 repositórios com o maior número de estrelas (stargazerCount), utilizando esta métrica como um proxy para popularidade. A coleta foi realizada de forma automatizada.

2.2. Protocolo de Coleta de Dados

Para a extração dos dados, foi desenvolvido um script em **C#** que realiza requisições automáticas e autenticadas à API. A consulta GraphQL foi estruturada para buscar os repositórios ordenados por estrelas e extrair os seguintes campos: name, description, url, stargazerCount, forkCount, createdAt, updatedAt, primaryLanguage, issues(totalCount), issues(states: CLOSED), pullRequests(totalCount) e releases(totalCount). Para coletar os 1.000 repositórios, foi implementado um mecanismo de paginação baseado em cursores. Os dados extraídos foram armazenados em um arquivo **repositorios_populares.csv**.

2.3. Definição e Cálculo das Métricas

As métricas para cada questão de pesquisa foram calculadas da seguinte forma:

- **Idade do Repositório (RQ01):** Diferença entre a data da coleta e o campo createdAt, expressa em anos.
- **Total de Pull Requests (RQ02):** Valor de totalCount do campo pullRequests.
- **Total de Releases (RQ03):** Valor de totalCount do campo releases.
- **Tempo até a Última Atualização (RQ04):** Diferença entre a data da coleta e o campo updatedAt, expressa em dias.
- **Linguagem Primária (RQ05):** Valor do campo primaryLanguage.name.
- **Razão de Issues Fechadas (RQ06):** Calculada pela fórmula Issues Fechadas / Issues Totais. O resultado é um valor entre 0 e 1.

2.4. Análise Estatística e Ferramental

A análise dos dados foi conduzida utilizando o próprio script em C# e os dados exportados. Para todas as métricas numéricas, a **mediana** foi escolhida como a principal medida de tendência central, pois é robusta a valores extremos (*outliers*), fornecendo uma representação mais fiel do repositório "típico" da amostra. A visualização dos dados foi realizada com a biblioteca **ScottPlot**, gerando gráficos de barras para apresentar os resultados.

3. Resultados

Esta seção apresenta os dados sumarizados obtidos na análise.

3.1. RQ01: Maturidade dos Sistemas Populares

- **Mediana da Idade dos Repositórios: 8 anos**.

Este valor indica que metade dos repositórios mais populares do GitHub possui uma maturidade considerável. Os gráficos abaixo ilustram os 10 repositórios mais antigos e os 10 mais novos entre os analisados.

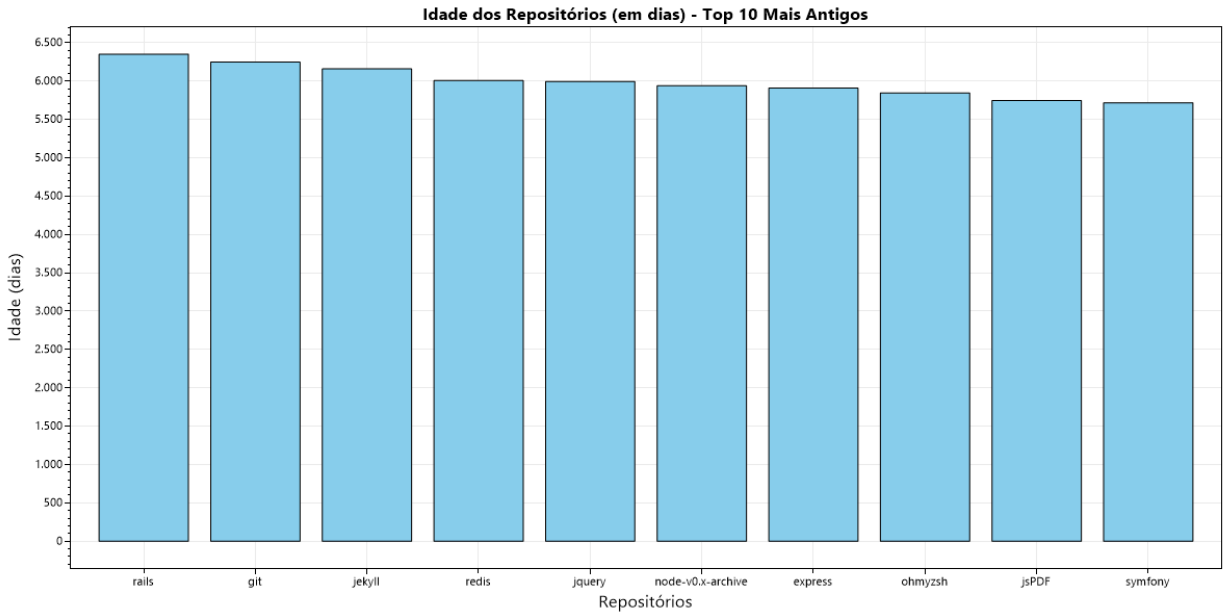

Figura 1: Top 10 repositórios mais antigos (em dias).

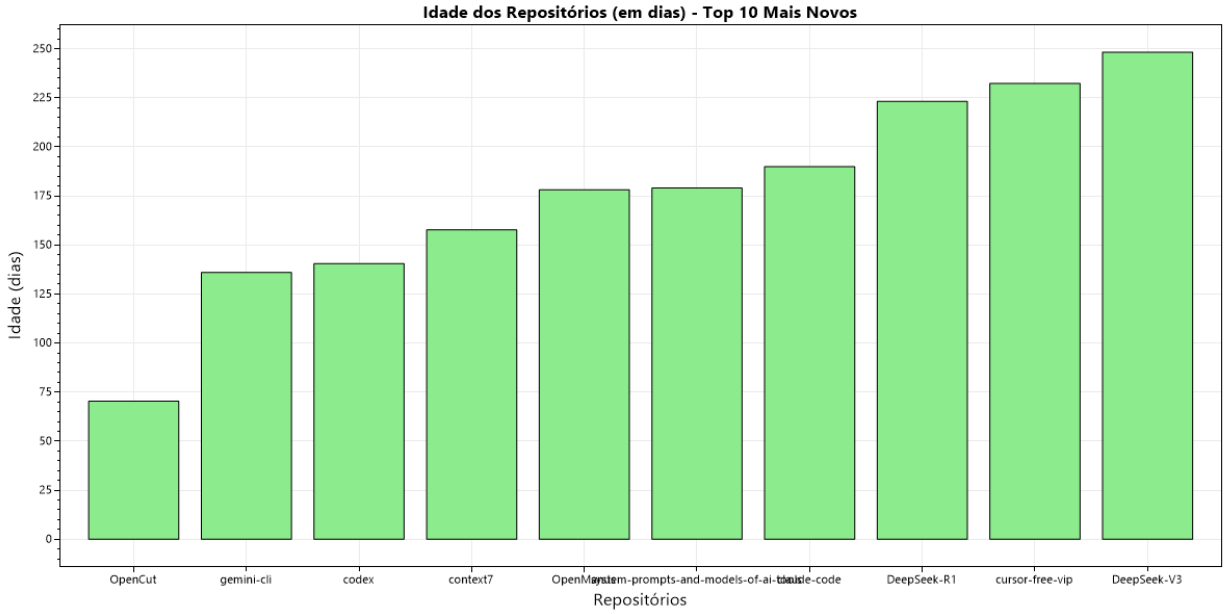

Figura 2: Top 10 repositórios mais novos (em dias).

3.2. RQ02: Volume de Contribuição Externa

- **Mediana do Número de Pull Requests: 1.147**.

Este número sugere um alto nível de engajamento e colaboração externa.

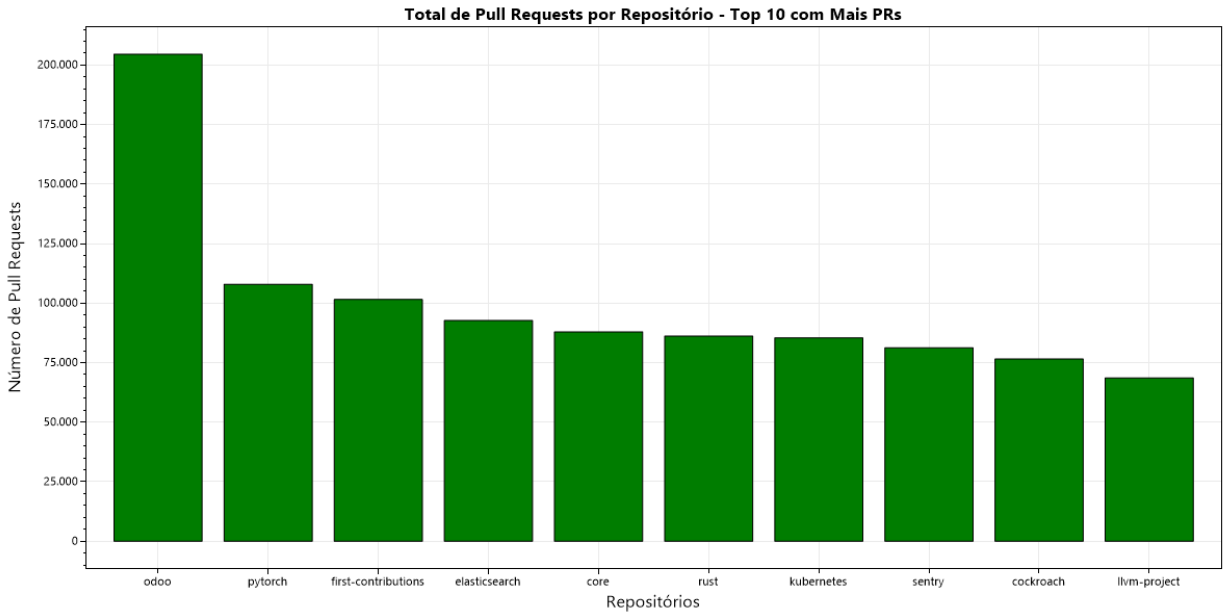

Figura 3: Top 10 repositórios com maior número de Pull Requests.

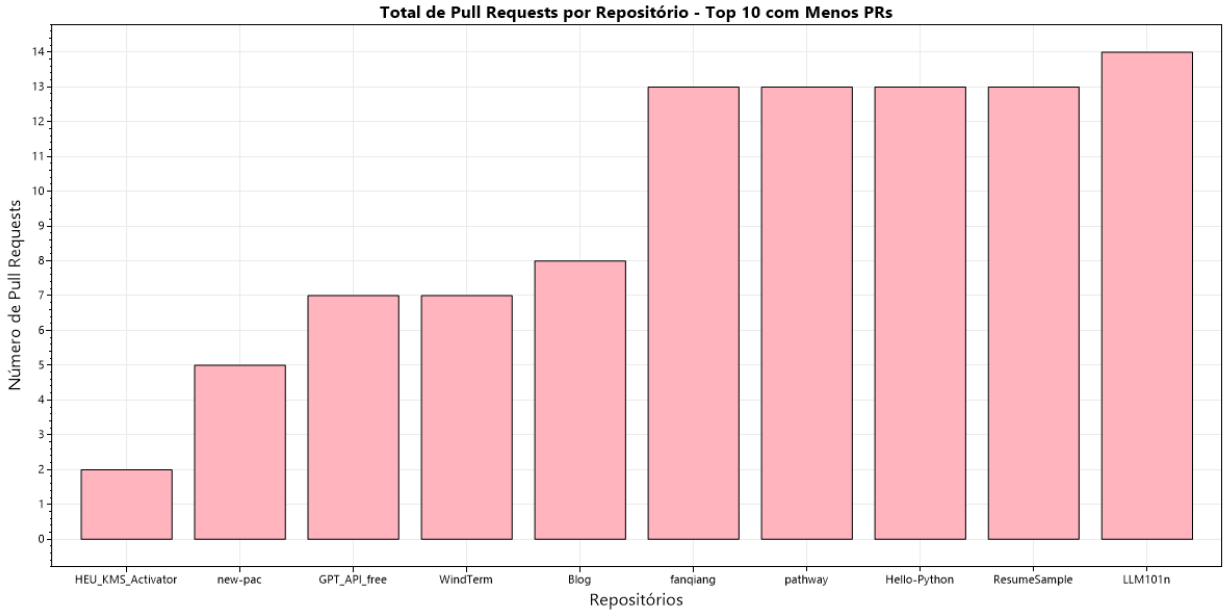

Figura 4: Top 10 repositórios com menor número de Pull Requests.

3.3. RQ03: Frequência de Lançamento de Versões

- **Mediana do Número de Releases: 35,5**.

Um projeto popular mediano já publicou mais de 35 versões oficiais, indicando uma cadência regular de entregas formais.

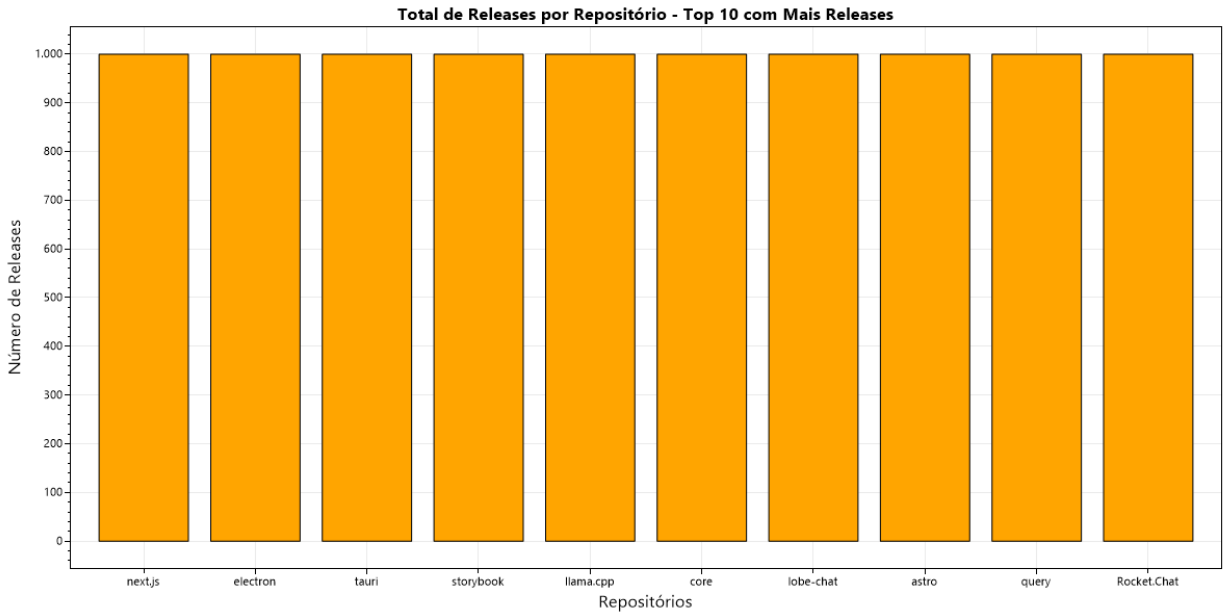

Figura 5: Top 10 repositórios com maior número de Releases.

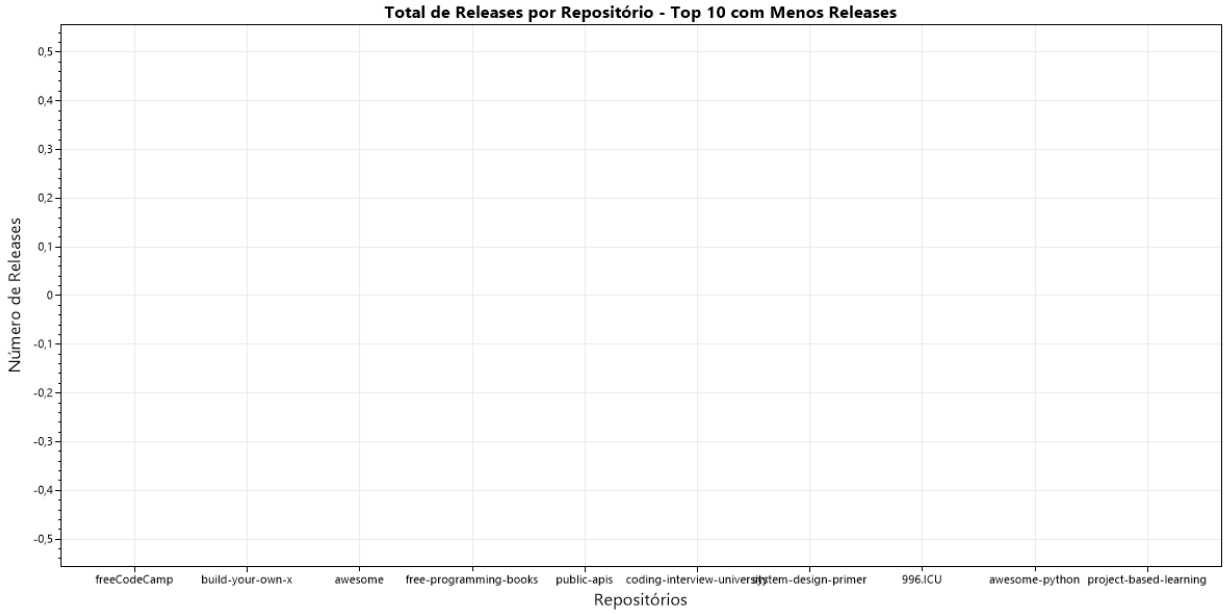

Figura 6: Top 10 repositórios com menor número de Releases.

3.4. RQ04: Frequência de Atualizações

A manutenção ativa é uma característica chave. Os gráficos abaixo mostram os repositórios atualizados há mais e menos tempo, reforçando que a maioria dos projetos populares é atualizada com alta frequência.

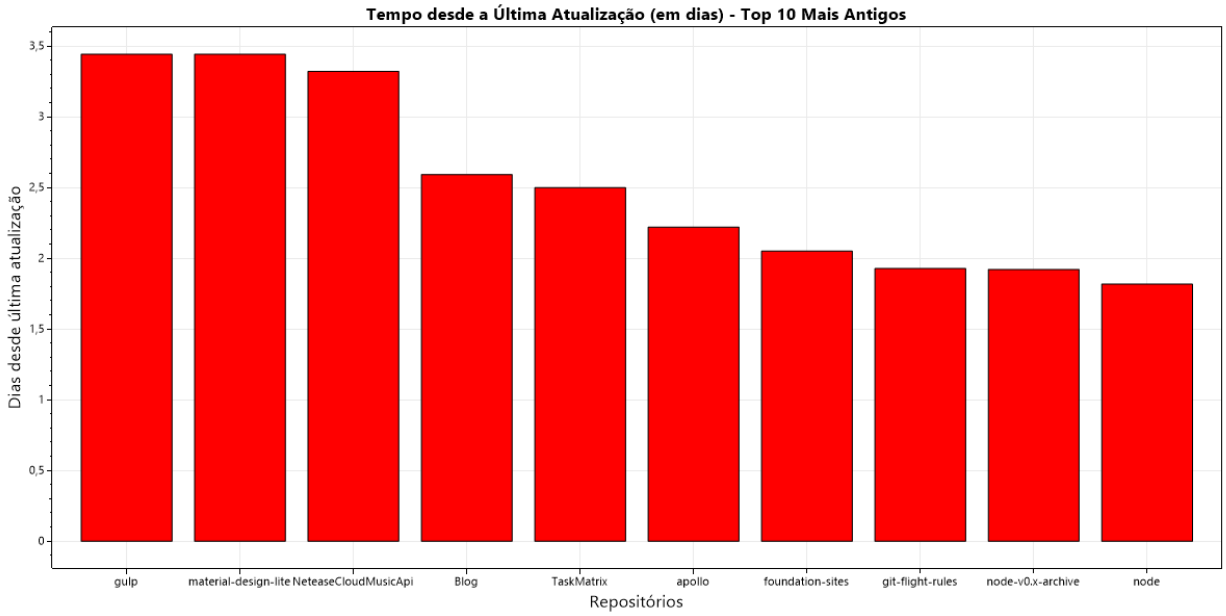

Figura 7: Top 10 repositórios com maior tempo desde a última atualização (em dias).

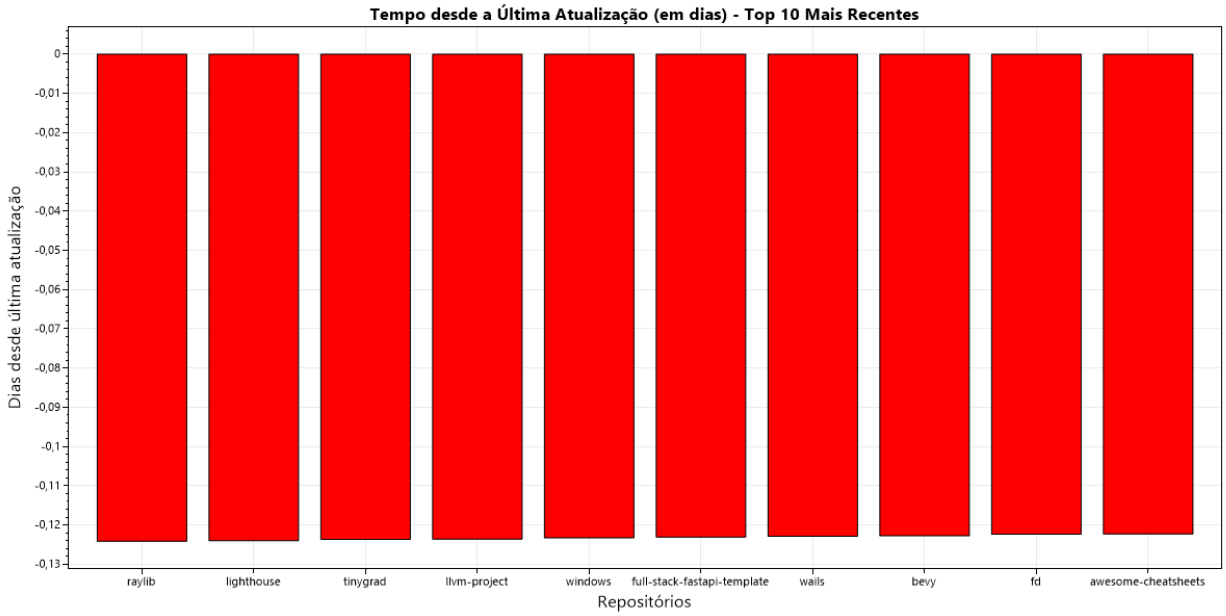

Figura 8: Top 10 repositórios com menor tempo desde a última atualização (em dias).

3.5. RQ05: Ecossistema de Linguagens de Programação

- **Linguagem Predominante: Python**.

A análise da linguagem primária confirma a dominância de certas tecnologias. O gráfico abaixo detalha a distribuição das 10 linguagens mais utilizadas.

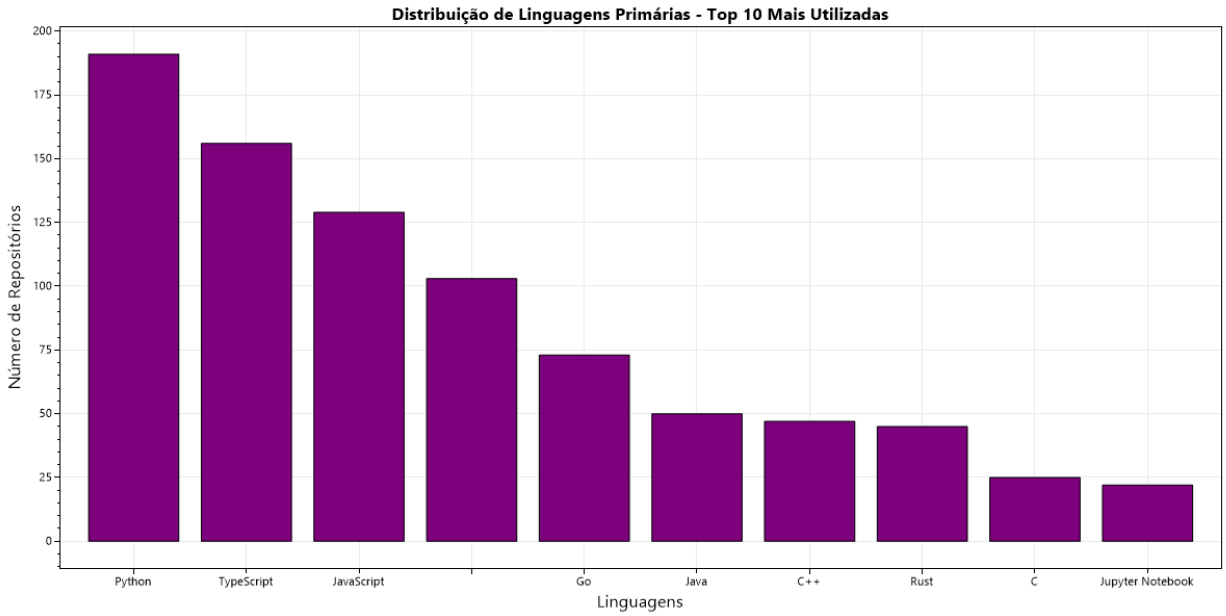

Figura 9: Distribuição das 10 linguagens primárias mais utilizadas.

3.6. RQ06: Eficiência na Gestão de Issues

- **Mediana da Razão de Issues Fechadas: 0.86**

  Este resultado,sugere que a maioria dos projetos populares gerencia seus problemas de forma eficaz.

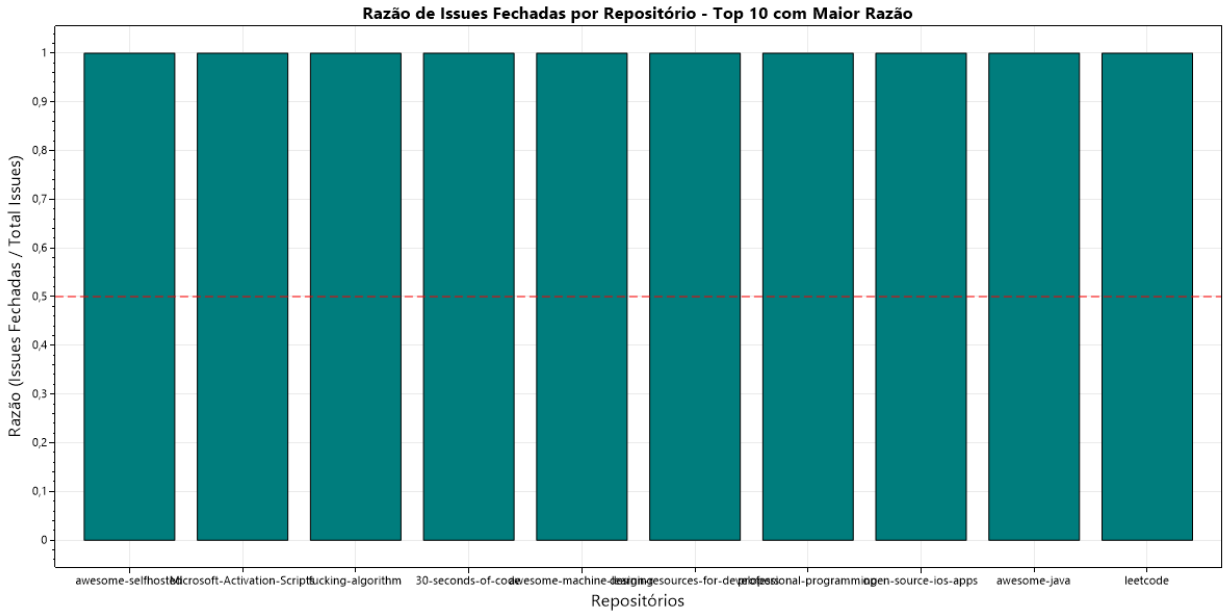

Figura 10: Top 10 repositórios com a maior razão de issues fechadas.

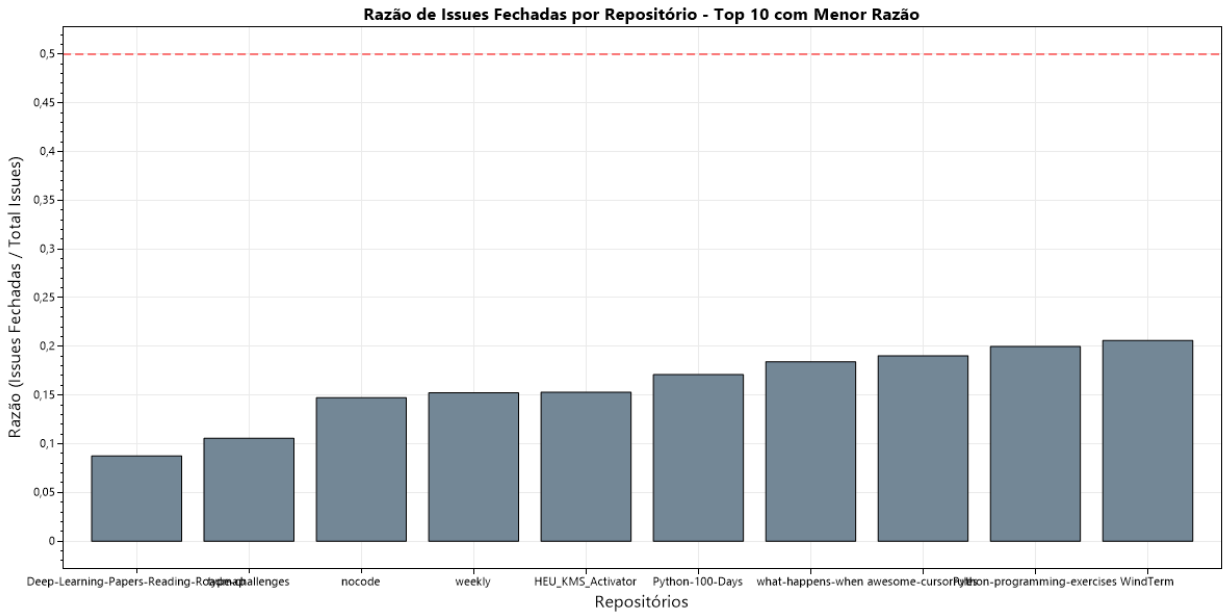

Figura 11: Top 10 repositórios com a menor razão de issues fechadas.

4. Discussão

4.1. Maturidade como Pilar da Popularidade (RQ01)

A mediana de **8 anos** confirma robustamente a hipótese H1. A popularidade não é um fenômeno efêmero, mas sim um atributo construído sobre a longevidade. Um projeto que persiste por quase uma década demonstra resiliência e utilidade contínua, fatores cruciais para a construção de confiança e de um ecossistema de suporte (tutoriais, plugins, especialistas) ao redor da tecnologia.

4.2. A Comunidade como Motor (RQ02)

A mediana de **1.147 pull requests** apoia fortemente a hipótese H2, evidenciando que a colaboração em massa é uma característica central. Este número não reflete apenas o engajamento, mas também a existência de processos de governança e engenharia de software sofisticados. Aceitar milhares de contribuições exige revisões de código rigorosas, testes automatizados e um fluxo de trabalho escalável para gerenciar a comunidade.

4.3. A Cadência das Entregas (RQ03 e RQ04)

A mediana de **35,5 releases** (H3) e a observação de atualizações constantes (H4) revelam as práticas de desenvolvimento modernas. Os releases formais indicam um versionamento estável, enquanto a alta frequência de atualizações (commits) reflete uma atividade de desenvolvimento quase em tempo real, alinhada com práticas de Integração e Entrega Contínua (CI/CD).

4.4. A Hegemonia das Linguagens Populares (RQ05)

A predominância de **Python**, seguida por linguagens como JavaScript e TypeScript (conforme Figura 9), confirma a hipótese H5. Este resultado reflete as grandes tendências da indústria: a centralidade da web (JavaScript/TypeScript) e a ascensão da Inteligência Artificial e Ciência de Dados (Python). Os repositórios mais populares são, muitas vezes, as próprias ferramentas que capacitam os domínios tecnológicos mais relevantes da atualidade.

4.5. Gestão de Issues e o Paradoxo da Popularidade (RQ06)

Uma alta mediana na razão de issues fechadas confirma a hipótese H6 de que projetos populares são bem gerenciados. No entanto, isso revela o "paradoxo da popularidade": esses projetos atraem um volume torrencial de interações. Um grande número de issues abertas pode não ser um sinal de negligência, mas sim um sintoma direto de um altíssimo nível de engajamento da comunidade, que supera a capacidade finita dos mantenedores.

4.6. Limitações do Estudo

É importante reconhecer as limitações desta análise. Primeiramente, o uso de "estrelas" como único proxy para popularidade não captura outras dimensões de impacto. Em segundo lugar, a "linguagem primária" simplifica a realidade de projetos poliglotas. Por fim, este estudo representa um *snapshot* em um ponto específico no tempo.

5. Conclusão

Este estudo realizou uma análise empírica dos 1.000 repositórios mais populares do GitHub, traçando um perfil claro do que constitui um projeto de sucesso no ecossistema open-source.

5.1. Síntese dos Resultados

Um repositório popular mediano é:

- **Maduro:** Com cerca de **8 anos** de existência.
- **Colaborativo:** Tendo integrado mais de **1.147 contribuições** externas.
- **Ativo:** Com uma cadência regular de **releases** (mediana de 35,5) e atualizações constantes.
- **Tecnologicamente Relevante:** Escrito em linguagens que dominam as tendências da indústria, como **Python**.
- **Bem Gerenciado:** Com uma alta taxa de resolução de *issues*.

Estes resultados reforçam que o sucesso de um projeto de código aberto transcende a qualidade técnica do código, dependendo fundamentalmente da longevidade, da capacidade de gerenciar uma comunidade e da manutenção de uma cadência de desenvolvimento ativa.