Análise de Repositórios Populares no GitHub

Lucas Cabral Soares
Lucas Hemétrio
Maria Eduarda Amaral Muniz

1. Introdução

Os repositórios mais populares do GitHub são frequentemente utilizados como referência para boas práticas de desenvolvimento de software, manutenção e colaboração aberta. No entanto, características como idade do repositório, número de contribuições externas, frequência de releases e qualidade da gestão de issues podem variar significativamente. Este estudo analisa 1000 repositórios mais populares do GitHub para identificar padrões nessas características.

1.1. Hipóteses

Antes da análise dos dados, formulamos algumas hipóteses sobre os sistemas populares:

- **RQ 01:** A maioria dos repositórios populares tem pelo menos 5 anos, refletindo maturidade.
- **RQ 02:** Repositórios populares recebem muitas contribuições externas (PRs aceitos).
- **RQ 03:** Sistemas populares lançam releases frequentemente, possivelmente uma vez por mês.
- **RQ 04:** A maioria dos repositórios é atualizada regularmente, com intervalo máximo de algumas semanas.
- **RQ 05:** JavaScript, Python e TypeScript são as linguagens mais comuns.
- **RQ 06:** Repositórios populares apresentam uma taxa de fechamento de issues superior a 80%.

2. Metodologia

Para responder às questões de pesquisa, utilizamos uma abordagem baseada na coleta, processamento e análise de dados de repositórios populares do GitHub. A seleção dos repositórios foi feita considerando os 1000 projetos com maior número de estrelas, garantindo que nossa análise se concentrasse nos sistemas mais reconhecidos e amplamente utilizados na plataforma.

A metodologia adotada pode ser dividida em três etapas principais: coleta dos dados, processamento das métricas e análise dos resultados.

2.1 Coleta dos Dados

Os dados foram extraídos utilizando a GitHub GraphQL API, o que permitiu coletar informações detalhadas sobre cada repositório, incluindo idade, número de pull requests aceitos, releases, tempo desde a última atualização, linguagem primária e gestão de issues. Para garantir uma amostra representativa, utilizamos uma consulta paginada para coletar até 1000 repositórios de forma eficiente. Os dados foram armazenados em um arquivo JSON para facilitar o processamento posterior.

2.2 Processamento e Cálculo das Métricas

Após a coleta, os dados foram processados e organizados em um arquivo CSV, permitindo uma melhor estruturação para análise estatística. As métricas utilizadas para responder às questões de pesquisa foram calculadas da seguinte forma:

- **Idade do repositório**: diferença entre a data de criação e a data de coleta dos dados.
- **Contribuições externas**: total de pull requests aceitos no repositório.
- **Frequência de releases**: total de releases registradas no histórico do projeto.
- **Frequência de atualizações**: tempo decorrido desde a última atualização do repositório.
- **Linguagens mais utilizadas**: categorização da linguagem primária de cada repositório.
- **Gestão de issues**: cálculo da razão de fechamento de issues, dada por:

$$Razão\ de\ Issues\ Fechadas\ = (\frac{Total\ de\ Issues\ Fechadas}{Total\ de\ Issues})\ x\ 100$$

Os dados foram analisados com a biblioteca Pandas, que permitiu calcular estatísticas descritivas como média, mediana, quartis e desvio padrão para cada métrica.

2.3 Visualização e Análise dos Dados

Para facilitar a interpretação dos dados, os resultados foram visualizados por meio de gráficos gerados com Matplotlib. Cada questão de pesquisa foi associada a um gráfico específico:

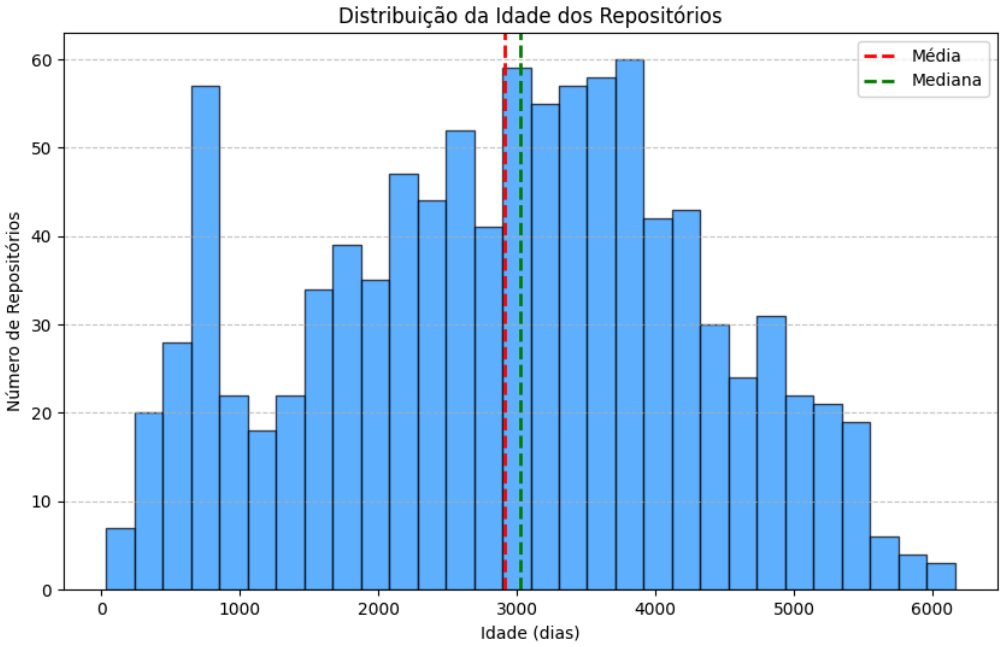
- **Distribuição da idade dos repositórios** (RQ 01)
- **Distribuição do número de pull requests aceitos** (RQ 02)
- **Distribuição do número de releases** (RQ 03)
- **Distribuição do tempo desde a última atualização** (RQ 04)

- **Frequência das linguagens mais utilizadas** (RQ 05)
- **Distribuição da razão de fechamento de issues** (RQ 06)

Com base nas estatísticas e nos gráficos, os resultados foram interpretados e comparados com nossas hipóteses iniciais, permitindo uma discussão detalhada sobre as tendências e padrões observados nos repositórios populares do GitHub.

3. Resultados e Discussão

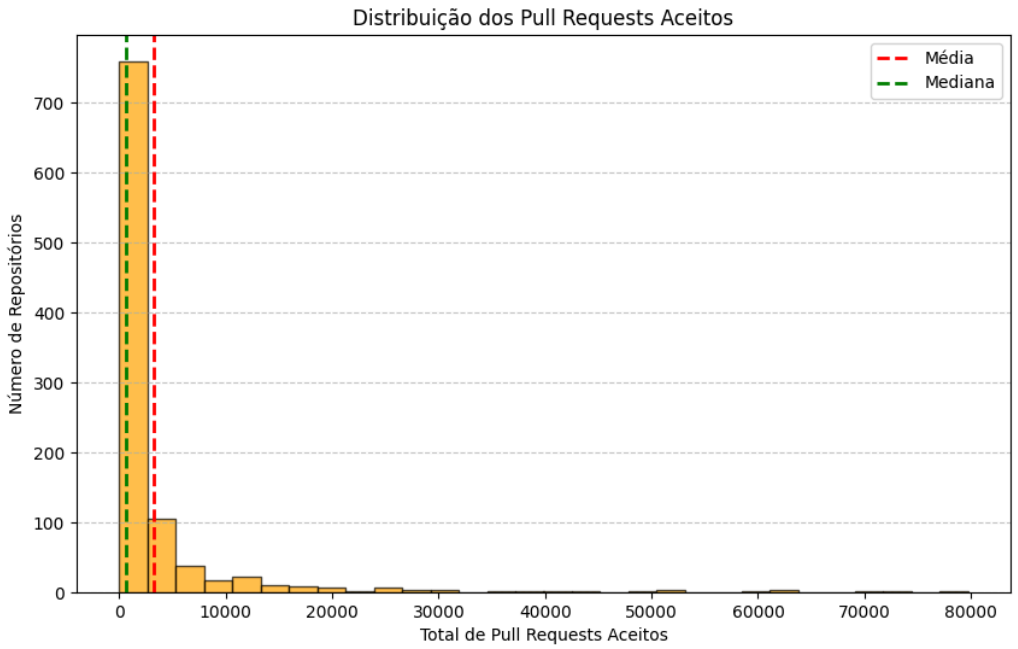
3.1 RQ 01: Sistemas populares são maduros/antigos?



A análise revelou que a idade média dos repositórios populares é de 2915 dias (~8 anos), enquanto a mediana está em 3023 dias (~8,3 anos). O repositório mais antigo tem 6163 dias (~16,9 anos), e o mais novo apenas 35 dias (~1 mês). Isso sugere que, embora a maioria dos repositórios populares seja madura e consolidada, alguns projetos conseguem alcançar popularidade rapidamente, mesmo com pouco tempo de existência.

Comparando com nossa hipótese inicial, que previa que a maioria dos repositórios teria pelo menos 5 anos de existência, os resultados confirmam essa expectativa. A mediana de 8 anos reforça que os projetos populares tendem a ser mais antigos, mas a presença de repositórios muito recentes entre os mais populares mostra que projetos inovadores conseguem crescer rapidamente no GitHub.

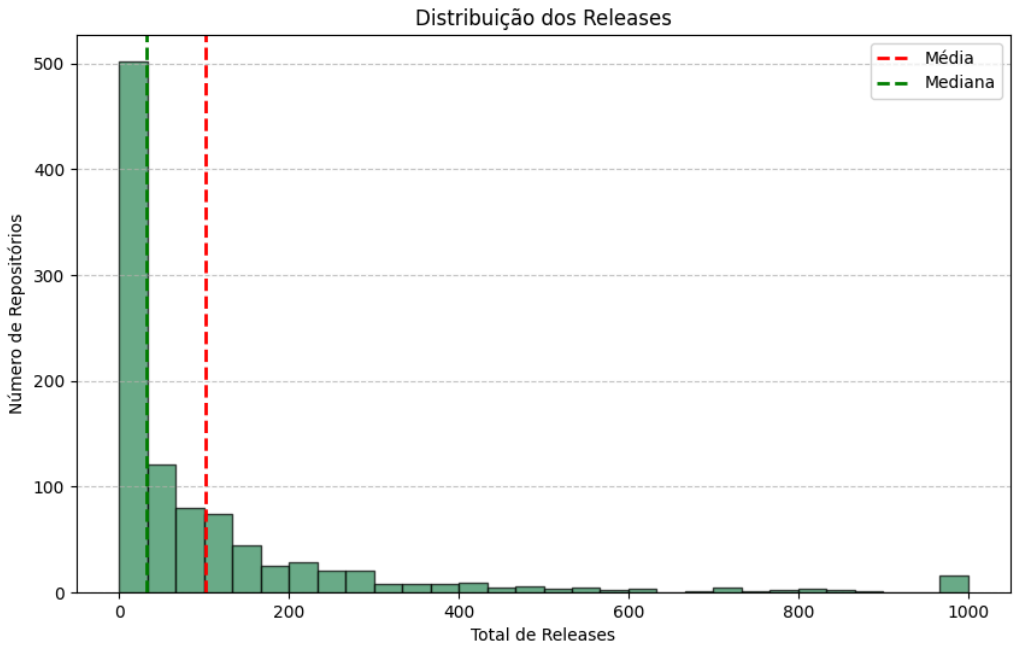
3.2 RQ 02: Sistemas populares recebem muita contribuição externa?



A análise mostrou que o número médio de Pull Requests (PRs) aceitos por repositório é 3231, mas a mediana é consideravelmente menor, com 619 PRs aceitos. Há uma grande disparidade entre os repositórios: alguns chegam a ter mais de 79.000 PRs aceitos, enquanto outros não possuem nenhum. Essa variação indica que, enquanto certos projetos são altamente colaborativos, outros podem ser desenvolvidos majoritariamente por um grupo fechado de mantenedores.

Nossa hipótese inicial sugeria que repositórios populares recebem muitas contribuições externas, o que foi parcialmente confirmado. Embora alguns repositórios tenham recebido um grande número de contribuições, a mediana de 619 PRs indica que a maioria recebe contribuições em um volume moderado. Além disso, a existência de repositórios sem PRs aceitos sugere que a popularidade de um projeto não está diretamente ligada ao nível de colaboração externa.

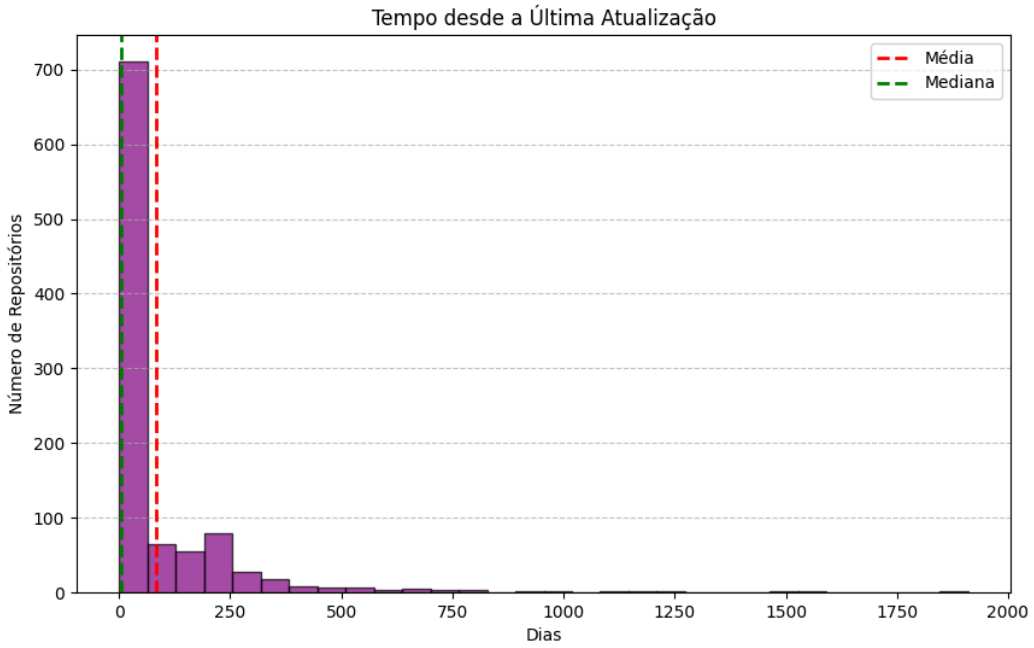
3.3 RQ 03: Sistemas populares lançam releases com frequência?



O número médio de releases por repositório é 101, mas a mediana é bem menor, com 33 releases. Um fato relevante é que 25% dos repositórios não possuem releases registrados, o que indica que nem todos os projetos populares utilizam versionamento formal ou preferem um fluxo contínuo de atualizações.

Nossa hipótese previa que sistemas populares lançariam releases regularmente, possivelmente uma vez por mês. No entanto, os resultados refutam essa ideia, pois muitos projetos não seguem esse padrão. A presença de um número significativo de repositórios sem releases sugere que nem todos os projetos priorizam um ciclo de lançamento formal, utilizando outras formas de distribuição de código.

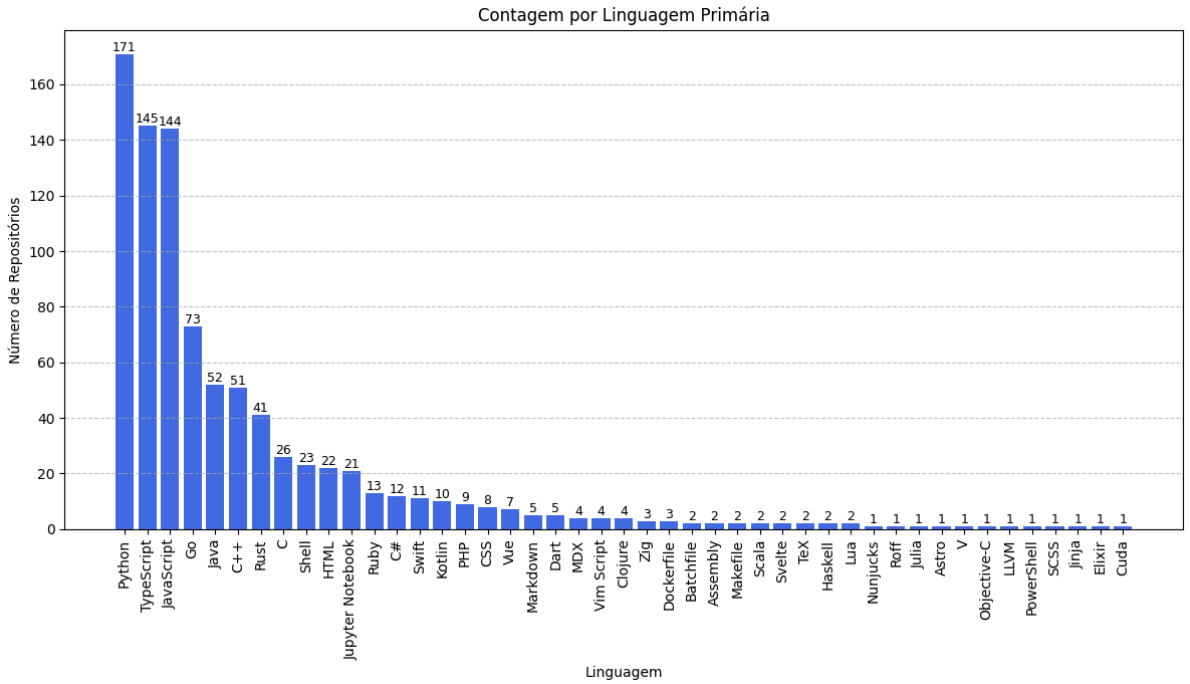
3.4 RQ 04: Sistemas populares são atualizados com frequência?



Os dados indicam que a média de tempo desde a última atualização dos repositórios é de 84 dias (~3 meses), enquanto a mediana é de apenas 5 dias. Isso significa que metade dos repositórios analisados foi atualizada nos últimos 5 dias, o que sugere um alto nível de atividade para muitos projetos. No entanto, outros repositórios não são atualizados há anos, com um caso extremo de 1910 dias (~5,2 anos) sem atualizações.

Nossa hipótese inicial previa que a maioria dos repositórios populares seria atualizada regularmente, com intervalo máximo de algumas semanas. Os resultados parcialmente confirmam essa expectativa: enquanto muitos repositórios são frequentemente mantidos, há outliers que não recebem atualizações há anos, o que pode indicar abandono ou estabilidade do software.

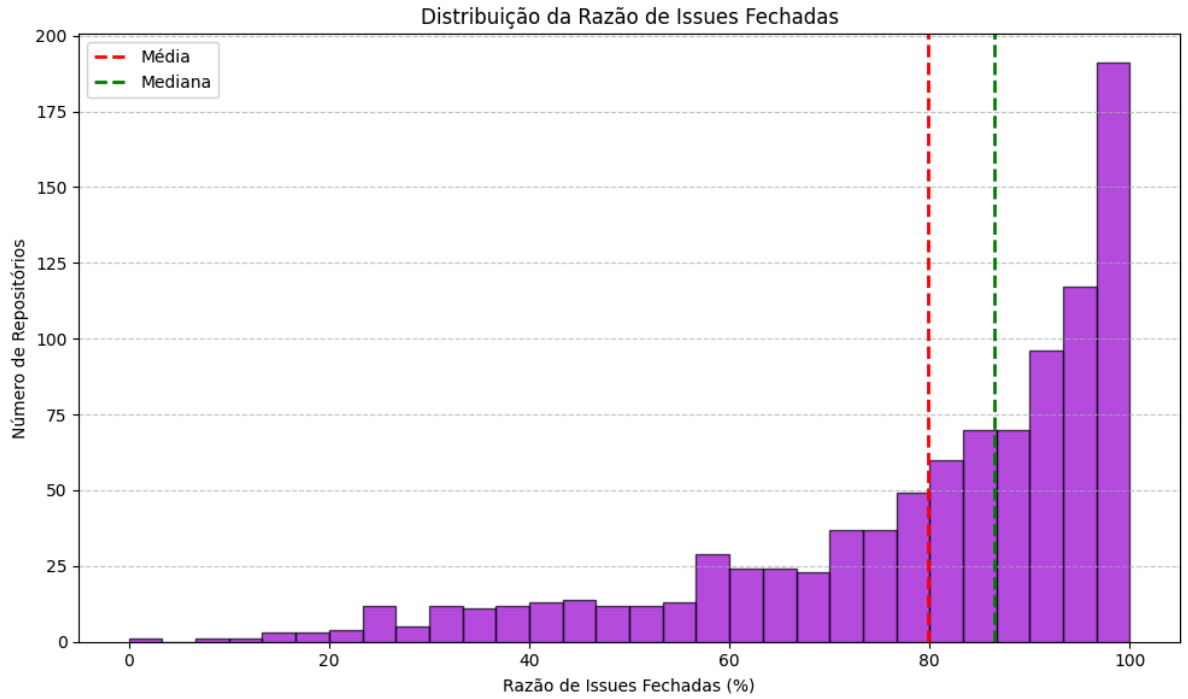
3.5 RQ 05: Sistemas populares são escritos nas linguagens mais populares?



A análise das linguagens primárias confirmou que Python, TypeScript e JavaScript dominam entre os repositórios populares, com 171, 145 e 144 repositórios, respectivamente. Outras linguagens com presença significativa incluem Go (73) e Java (52). Esses dados reforçam a tendência de que linguagens amplamente utilizadas no mercado também dominam os projetos open-source populares.

Nossa hipótese previa que JavaScript, Python e TypeScript estariam entre as linguagens mais comuns nos repositórios populares, o que foi confirmado. A forte presença dessas linguagens sugere que elas são amplamente adotadas tanto em projetos comerciais quanto em open-source, facilitando a popularização de repositórios escritos nessas tecnologias.

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



A análise da gestão de issues mostrou que, em média, 79,84% das issues abertas nos repositórios populares foram fechadas. Esse percentual sugere que a maioria dos projetos gerencia bem seus problemas e feedback da comunidade, mantendo um fluxo ativo de resolução de problemas. No entanto, existem repositórios com baixas taxas de fechamento, possivelmente devido ao acúmulo de backlog ou falta de manutenção ativa.

Nossa hipótese previa que a taxa média de fechamento de issues seria superior a 80%, o que foi parcialmente confirmado. O resultado médio ficou muito próximo da expectativa, mas a presença de projetos com taxas de fechamento mais baixas sugere que nem todos os repositórios populares mantêm uma gestão eficiente de issues.

4. Conclusão

Os dados analisados confirmam que a maioria dos repositórios populares no GitHub são antigos, com mais de cinco anos de existência, mas também mostram que novos projetos podem rapidamente ganhar popularidade. Em relação à colaboração externa, encontramos uma grande variação: enquanto alguns projetos aceitam milhares de pull requests, outros não recebem contribuições externas significativas. Isso indica que a popularidade de um repositório não está necessariamente ligada ao seu nível de colaboração aberta.

A análise dos releases mostrou que, embora muitos repositórios façam lançamentos frequentes, um quarto dos projetos não segue um versionamento formal, sugerindo que alguns utilizam outras formas de distribuição ou mantêm um fluxo contínuo de desenvolvimento. Já no quesito frequência de atualização, verificamos que a maioria dos repositórios é atualizada regularmente, mas existem projetos populares que não recebem atualizações há anos, o que pode indicar abandono ou estabilidade do software.

Em relação às linguagens de programação, os resultados confirmam a hipótese de que JavaScript, Python e TypeScript dominam os repositórios populares, refletindo sua ampla adoção no desenvolvimento de software moderno. Finalmente, a análise da gestão de issues mostrou que a maioria dos projetos mantém um bom gerenciamento, com 79,84% das issues fechadas. Embora essa taxa esteja próxima da expectativa de 80%, a variação entre os projetos sugere que nem todos possuem uma estrutura eficiente de triagem e resolução de problemas.