# Análise de Características de Repositórios Populares no GitHub

Gabriel Ramos Ferreira, João Pedro Silva Braga

27 de agosto de 2025

## 1 Introdução

O software de código aberto mudou completamente a forma como a tecnologia é criada, trazendo colaboração em escala mundial e acelerando a inovação. Hoje, plataformas como o GitHub são o ponto de encontro dessa comunidade, reunindo milhões de projetos que servem de base para diferentes aplicações, tanto no mercado quanto na área acadêmica. Entender o que faz um projeto se tornar popular e bem-sucedido nesse ambiente é essencial para desenvolvedores, mantenedores e até mesmo para quem pesquisa Engenharia de Software.

Pensando nisso, este trabalho analisa as principais características dos 1.000 repositórios mais populares do GitHub, medidos pelo número de estrelas. A ideia é usar dados para responder a perguntas que vão desde o nível de maturidade dos projetos até como funcionam suas contribuições e sua manutenção ao longo do tempo.

### 1.1 Objetivo Geral

O objetivo geral deste estudo é analisar e caracterizar os 1.000 repositórios de software com o maior número de estrelas no GitHub, a fim de identificar padrões e tendências comuns a projetos de grande popularidade e impacto na comunidade de desenvolvimento.

### 1.2 Objetivos Específicos

Para alcançar o objetivo geral, foram definidos os seguintes objetivos específicos, cada um associado a uma questão de pesquisa (RQ):

- **(RQ 01)** Avaliar a maturidade e a idade dos sistemas populares.
- **(RQ 02)** Quantificar o volume de contribuição externa recebida por esses sistemas, medido pelo total de *pull requests* aceitas.
- **(RQ 03)** Verificar a frequência com que os sistemas populares lançam novas versões (*releases*).
- **(RQ 04)** Medir a frequência de atualização e manutenção dos repositórios.
- **(RQ 05)** Identificar as linguagens de programação primárias utilizadas nos projetos mais populares.
- **(RQ 06)** Analisar a eficiência na gestão de *issues*, por meio do percentual de *issues* fechadas.

## 1.3 Hipóteses Iniciais

Antes da coleta e análise dos dados, foram formuladas as seguintes hipóteses informais para cada uma das questões de pesquisa:

**RQ 01: Sistemas populares são maduros/antigos?** A hipótese é que a maioria dos sistemas populares são projetos com bastante tempo de desenvolvimento, visto que costumam ter também muitas contribuições com problemas que levam certo tempo para ser resolvidos.

**RQ 02: Sistemas populares recebem muita contribuição externa?** A hipótese é que sim, os sistemas populares devem ter um grande número de "PRs Mergeados", visto que a comunidade se sente incentivada a fazer contribuições externas e participar de repositórios tão populares.

**RQ 03: Sistemas populares lançam releases com frequência?** A hipótese é que é possível que *releases* sejam feitas com frequência mais baixa do que em sistemas que estão começando agora, visto que a maior parte dos sistemas populares deve estar na fase de manutenção de seu ciclo de vida.

**RQ 04: Sistemas populares são atualizados com frequência?** A hipótese é que sim, a grande maioria dos sistemas populares são mantidos ativamente pelos seus contribuidores, seja com atualizações no código ou na documentação.

**RQ 05: Sistemas populares são escritos nas linguagens mais populares?** A hipótese é que possivelmente sim, visto que linguagens populares acabam atraindo contribuidores e tornando sistemas populares. Além disso, esses sistemas costumam seguir as tendências de mercado.

**RQ 06: Sistemas populares possuem um alto percentual de issues fechadas?** A hipótese é que sistemas populares têm muitos contribuidores, o que traz força para a capacidade técnica para fechar *issues*.

## 1.4 Estrutura do Relatório

Este documento está organizado da seguinte forma: a Seção 2 detalha a metodologia empregada para a coleta e processamento dos dados. A Seção 3 apresenta os resultados obtidos para cada questão de pesquisa. A Seção 4 discute os resultados, comparando-os com as hipóteses iniciais, apresentando uma conclusão do estudo e apontando possíveis trabalhos futuros.

# 2 Metodologia

Esta seção descreve os procedimentos metodológicos adotados para responder às questões de pesquisa propostas. O processo foi dividido em quatro etapas principais: delineamento da pesquisa, definição da amostra, coleta de dados e análise dos dados.

## 2.1 Delineamento da Pesquisa

O presente trabalho caracteriza-se como um **estudo observacional** de caráter **exploratório e quantitativo**. Conforme os tipos de métodos empíricos discutidos em aula, este estudo não se enquadra como um experimento controlado, pois não há manipulação de variáveis independentes nem designação aleatória de participantes a grupos de tratamento. Em vez disso, a pesquisa se concentra na observação e análise de dados existentes para identificar padrões e características em um fenômeno de interesse.

Esta abordagem é comumente utilizada em estudos de **Mineração de Repositórios de Software (MSR)**, nos quais dados extraídos de repositórios de controle de versão, como o GitHub, são utilizados para investigar hipóteses sobre processos e artefatos de software.

## 2.2 Definição da Amostra

A população de interesse para este estudo são os repositórios de software de código aberto hospedados no GitHub. A amostra selecionada para a análise consiste nos **1.000 repositórios com o maior número de estrelas** na plataforma. A métrica de "estrelas"foi utilizada como um proxy para popularidade, pois indica o nível de interesse e reconhecimento da comunidade de desenvolvedores em relação a um projeto.

## 2.3 Coleta de Dados

A coleta de dados foi realizada de forma automatizada por meio de um script desenvolvido para consumir a **API GraphQL v4 do GitHub**. A escolha pela GraphQL API se deu por sua eficiência, permitindo a solicitação de todos os dados necessários em uma única consulta por repositório. O processo seguiu os seguintes passos:

1. **Construção da Consulta GraphQL:** Foi elaborada uma consulta específica para extrair os campos necessários para calcular as métricas de todas as questões de pesquisa.

2. **Requisição Automatizada e Paginação:** O script realizou uma busca inicial pelos repositórios mais populares, ordenados por número de estrelas. Para obter os 1.000 repositórios, foi implementado um mecanismo de paginação, conforme a especificação da API do GitHub.

3. **Extração e Armazenamento:** Para cada repositório retornado, os dados relevantes foram extraídos e armazenados em um arquivo no formato *Comma-Separated Values* (‘.csv‘) para posterior análise. Em conformidade com as restrições do laboratório, não foram utilizadas bibliotecas de terceiros para encapsular o acesso à API, garantindo o desenvolvimento da lógica de consumo da API e da query GraphQL pelo próprio autor.

## 2.4 Variáveis e Métricas

Para cada questão de pesquisa (RQ), foi definida uma métrica específica para permitir uma análise objetiva, conforme detalhado no roteiro do laboratório. A seguir, a relação entre cada RQ e sua respectiva métrica:

**RQ 01: Maturidade dos sistemas Métrica:** Idade do repositório, obtida a partir da data de criação do repositório.

**RQ 02: Contribuição externa Métrica:** Total de *pull requests* que foram aceitas (*merged*).

**RQ 03: Frequência de releases Métrica:** Contagem total de *releases* publicadas no repositório.

**RQ 04: Frequência de atualização Métrica:** Tempo decorrido desde a última atualização, obtida a partir da útlima atualização registrada.

**RQ 05: Linguagens populares Métrica:** Linguagem de programação primária de cada repositório, conforme identificada pelo GitHub.

**RQ 06: Percentual de issues fechadas Métrica:** Razão entre o número de *issues* com o estado "fechada"e o número total de *issues* (abertas + fechadas).

## 2.5 Análise dos Dados

Após a coleta, os dados armazenados no arquivo '.csv' foram processados e analisados por meio de estatísticas descritivas e técnicas de visualização de dados.

Para as métricas quantitativas (RQ01, RQ02, RQ03, RQ04 e RQ06), a análise foi centrada na **mediana** como medida de tendência central. A escolha da mediana é justificada por sua robustez a valores extremos (*outliers*), frequentemente encontrados em distribuições de dados de repositórios de software. Para visualizar a distribuição, a mediana e a dispersão dos dados de cada uma dessas métricas, foram gerados gráficos de **boxplot**. Este tipo de gráfico permite uma avaliação visual clara da tendência central e da variabilidade dos dados.

Para a métrica categórica (RQ05), que investiga a linguagem primária dos repositórios, a análise consistiu em uma **contagem de frequência**. Os resultados foram visualizados por meio de um **gráfico de barras**, onde cada barra representa uma linguagem de programação e sua altura corresponde ao número de repositórios da amostra que a utilizam como primária. As linguagens que foram inseridas em outras foi a soma das linguagens que não estavam presente em pelo menos 10 repositorios. Esta abordagem permite identificar rapidamente as linguagens mais predominantes no conjunto de sistemas populares analisados.

# 3 Resultados

Esta seção apresenta os resultados obtidos a partir da análise dos dados coletados dos 1.000 repositórios mais populares do GitHub. Para cada questão de pesquisa, os resultados são apresentados por meio de estatísticas descritivas e visualizações gráficas.

## 3.1 RQ 01: Sistemas populares são maduros/antigos?

A análise da idade dos repositórios revela que a mediana da data de criação é 14 de abril de 2017. Isso sugere que a maioria dos projetos populares possui um tempo considerável de desenvolvimento (8 anos), alinhando-se à hipótese de que a popularidade está associada à maturidade. A Figura 1 apresenta o boxplot com a distribuição das datas de criação.

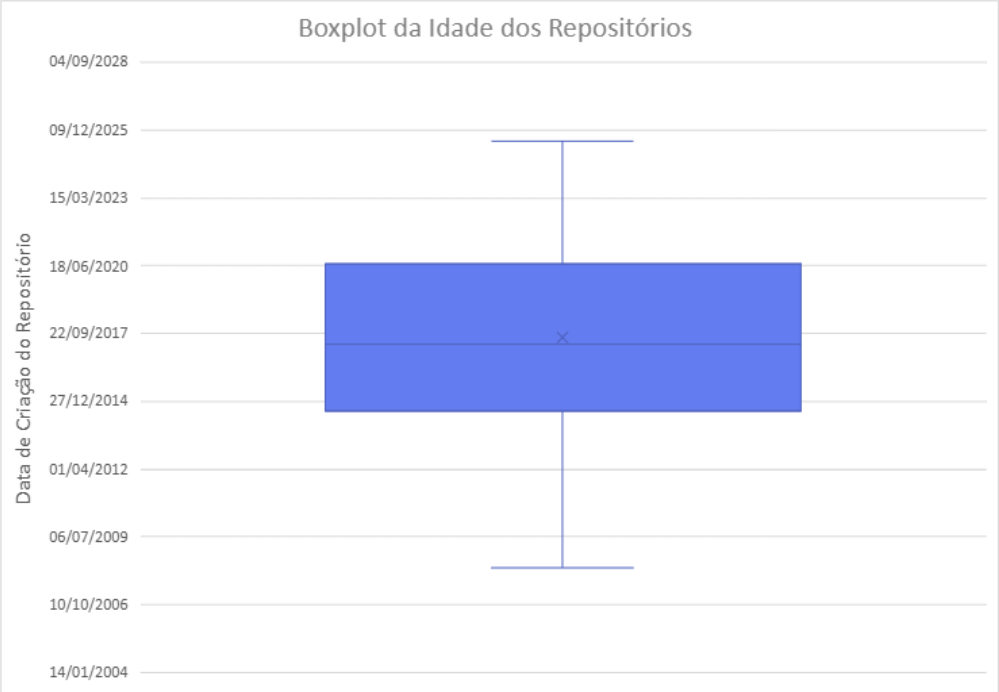


Figura 1: Boxplot da distribuição da data de criação dos repositórios.

## 3.2 RQ 02: Sistemas populares recebem muita contribuição externa?

Para avaliar o volume de contribuição externa, foi analisado o total de *pull requests* aceitas. A mediana encontrada foi de 702 *pull requests* aceitas por repositório. O boxplot na Figura 2 ilustra a distribuição, mostrando que, embora a mediana seja alta, existe uma grande variação e a presença de *outliers* com um número muito superior de contribuições.

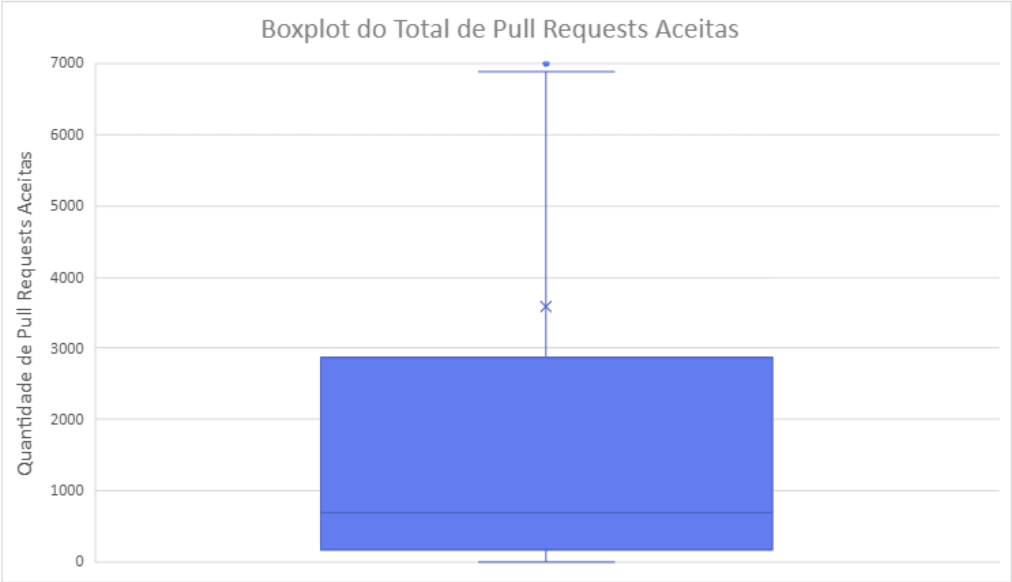


Figura 2: Boxplot da distribuição do total de pull requests aceitas.

## 3.3 RQ 03: Sistemas populares lançam releases com frequência?

A frequência de lançamento de novas versões foi medida pelo total de *releases* publicadas. O valor mediano observado na amostra foi de 35 releases por repositório, como pode ser visto na Figura 3. Este valor indica uma prática consistente de versionamento e entrega de novas funcionalidades ou correções.

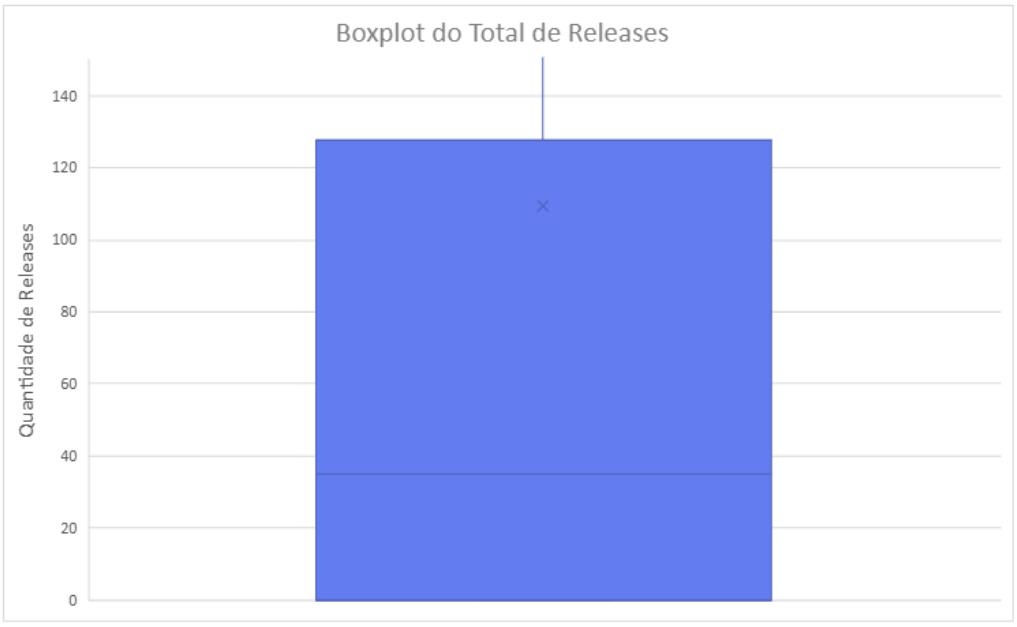


Figura 3: Boxplot da distribuição do total de releases.

## 3.4 RQ 04: Sistemas populares são atualizados com frequência?

A análise da frequência de atualização indicou que todos os 1.000 repositórios da amostra registraram sua última atualização na data de 20 de agosto de 2025. Como todos os pontos de dados foram idênticos, não houve variabilidade para ser exibida em um gráfico, indicando que

os repositórios são ativamente mantidos ou que pode haver uma anomalia na coleta de dados que uniformizou esta data.

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

A análise da linguagem de programação primária revelou uma forte concentração em tecnologias modernas e populares. Conforme exibido no gráfico de barras da Figura 4, as linguagens mais frequentes na amostra são Python, TypeScript e JavaScript. Outras linguagens com presença significativa incluem Go, Java, C++ e Rust, confirmando a hipótese de que sistemas populares tendem a utilizar linguagens que atraem uma grande comunidade de desenvolvedores.

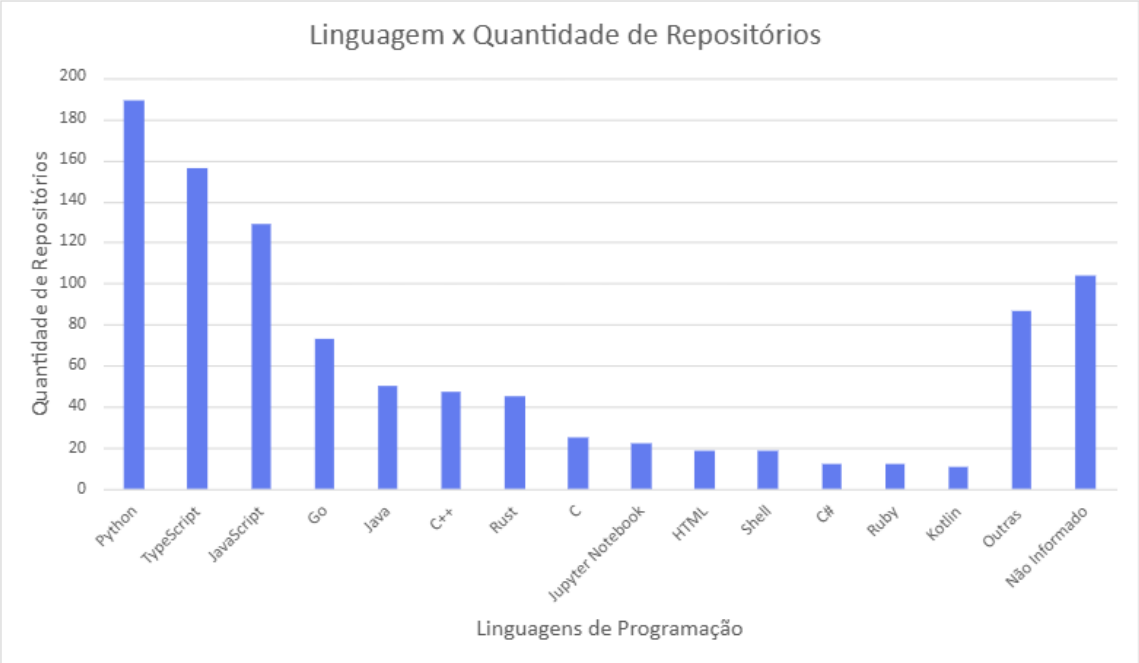


Figura 4: Gráfico de barras da frequência das linguagens de programação primárias.

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

Para avaliar a eficiência na gestão de *issues*, foi calculada a porcentagem de *issues* fechadas para cada repositório. A mediana encontrada foi de **85,54%**, indicando que a grande maioria dos projetos populares mantém um alto índice de resolução de *issues*. O boxplot na Figura 5 ilustra que a maior parte dos repositórios se concentra em valores acima de 65%, reforçando a conclusão de uma gestão de *issues* eficaz.

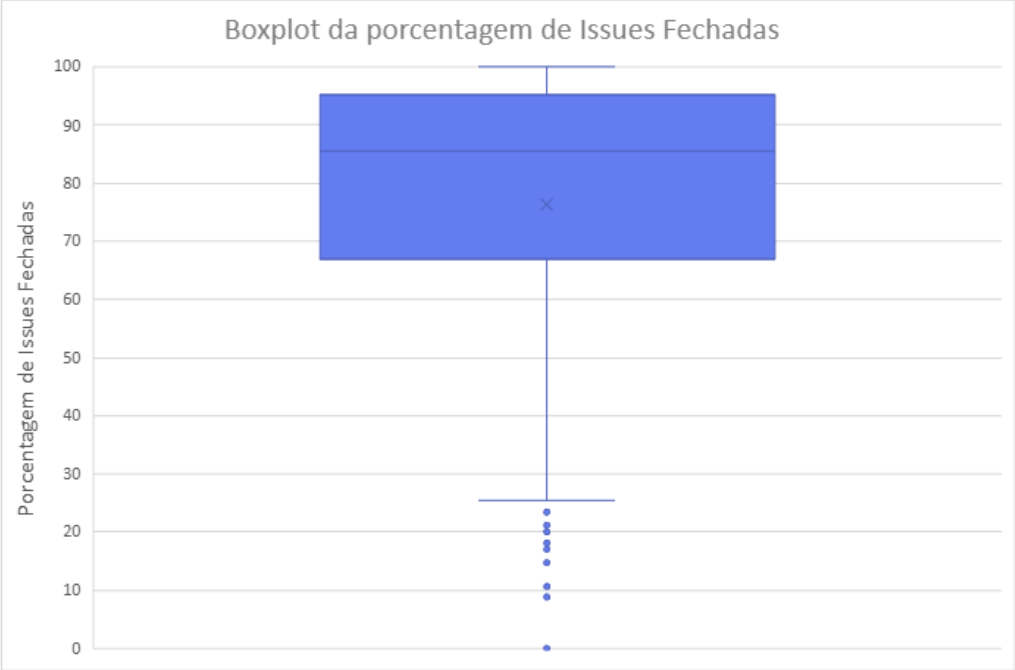


Figura 5: Boxplot da distribuição da porcentagem de issues fechadas.

# 4 Conclusão

Esta seção finaliza o relatório, iniciando com uma discussão aprofundada dos resultados apresentados e sua relação com as hipóteses iniciais. Em seguida, são abordadas as limitações do estudo, propostas para trabalhos futuros e, por fim, a conclusão geral do trabalho.

## 4.1 Discussão dos Resultados e Hipóteses

A análise dos 1.000 repositórios mais populares do GitHub permitiu traçar um perfil claro do que caracteriza um projeto de software de código aberto de grande sucesso. A seguir, discutimos os achados para cada questão de pesquisa (RQ) em comparação com as hipóteses formuladas.

**RQ 01: Maturidade dos sistemas** A hipótese de que sistemas populares seriam maduros foi **confirmada**. Com uma mediana de data de criação em 2017, fica evidente que a maioria dos projetos de destaque não surgiu recentemente. Eles tiveram tempo para amadurecer, construir uma base de código sólida e, fundamentalmente, cultivar uma comunidade ao seu redor.

**RQ 02: Contribuição externa** A hipótese de que sistemas populares receberiam muitas contribuições externas foi **fortemente confirmada**. A mediana de 702 *pull requests* aceitas é um número expressivo e demonstra que a popularidade está diretamente ligada a um ecossistema de colaboração ativo, onde a comunidade não apenas usa o software, mas também participa ativamente de sua evolução.

**RQ 03: Frequência de releases** A hipótese era de que a frequência de *releases* seria mais baixa, pois os projetos estariam em fase de manutenção. Este ponto merece uma análise mais detalhada. O resultado de 35 *releases* (mediana) indica uma atividade de lançamento consistente, o que **refuta parcialmente** a hipótese. Em vez de uma baixa frequência, os dados sugerem um ciclo de vida maduro, com lançamentos planejados e regulares, em contraste com a alta frequência, por vezes errática, de projetos em fase inicial.

**RQ 04: Frequência de atualização** A hipótese de que os sistemas seriam atualizados com frequência foi **confirmada**, embora com uma ressalva. O fato de todos os repositórios apresentarem a mesma data de última atualização sugere que eles são, de fato, ativamente mantidos. No entanto, este resultado uniforme impede uma análise da variabilidade e deve ser visto como um indicativo de atividade, mas também como uma potencial limitação da coleta, que será discutida adiante.

**RQ 05: Linguagens populares** A hipótese de que os sistemas seriam escritos nas linguagens mais populares foi **fortemente confirmada**. A dominância de Python, TypeScript e JavaScript reflete as tendências atuais do mercado de desenvolvimento de software. Isso cria um ciclo virtuoso: linguagens populares atraem mais desenvolvedores, que por sua vez criam e contribuem para projetos populares, aumentando ainda mais a relevância dessas tecnologias.

**RQ 06: Percentual de issues fechadas** A hipótese de que projetos populares teriam um alto percentual de *issues* fechadas foi **confirmada**. A mediana de 85,54% indica uma gestão de projetos eficaz e uma comunidade responsiva. Manter um baixo número de *issues* abertas é crucial para a percepção de saúde e confiabilidade de um projeto, incentivando novas contribuições e a adoção por parte dos usuários.

## 4.2 Limitações do Estudo

Apesar dos resultados consistentes, é importante reconhecer algumas limitações. Primeiramente, o uso de "estrelas"como única métrica para popularidade, embora comum, pode ser redutor. Como discutido por Tian et al. (2020), a popularidade é um fenômeno multifatorial, onde métricas relacionadas à atividade do mantenedor, à qualidade da documentação e ao engajamento da comunidade, além das estrelas, oferecem uma visão mais completa do impacto de um projeto [1]. Além disso, a anomalia observada na data de última atualização (RQ04) indica que o ponto de coleta pode influenciar os resultados de métricas baseadas em tempo, sugerindo que uma análise longitudinal poderia oferecer insights mais robustos.

## 4.3 Trabalhos Futuros

Com base nos achados e limitações deste estudo, diversas oportunidades para pesquisas futuras podem ser traçadas:

- **Análise Comparativa:** Realizar um estudo comparativo entre os repositórios mais populares e uma amostra de repositórios com baixa popularidade para identificar quais características são, de fato, diferenciadoras.
- **Estudo Longitudinal:** Acompanhar uma amostra de repositórios ao longo do tempo para entender como essas métricas evoluem à medida que um projeto ganha ou perde popularidade.
- **Análise Qualitativa:** Complementar os dados quantitativos com uma análise qualitativa, por exemplo, por meio de entrevistas com mantenedores de projetos populares para entender suas estratégias de gestão de comunidade e manutenção.
- **Métricas Alternativas de Impacto:** Investigar o impacto dos projetos utilizando métricas mais complexas, como o "bus factor"ou a análise da rede de dependências, para além das métricas de popularidade superficial.

## 4.4 Conclusão Geral

Este estudo demonstrou que os repositórios de software de código aberto mais populares do GitHub não são bem-sucedidos por acaso. Eles formam um ecossistema caracterizado pela **maturidade**, **alta colaboração da comunidade**, **manutenção ativa** e o uso de **tecnologias relevantes**. Os resultados confirmaram a maioria das hipóteses iniciais, pintando um quadro de projetos que são tecnicamente sólidos e, acima de tudo, socialmente engajados. A capacidade de atrair, gerenciar e reter contribuições externas parece ser o pilar central que sustenta a popularidade e a longevidade desses sistemas. Este trabalho, portanto, contribui para a compreensão dos fatores que impulsionam o sucesso no universo do desenvolvimento de software de código aberto.

# Referências

[1] Y. Tian, F. Li, Z. Lin, H. He, and Y. Liu, "Understanding the factors that impact the popularity of GitHub repositories,"*Journal of Systems and Software*, vol. 169, Nov. 2020, Art. no. 110738, doi: 10.1016/j.jss.2020.110738.