Este trabalho tem como objetivo analisar as principais características de repositórios populares do GitHub,

a partir de uma amostra dos 1000 projetos com maior número de estrelas.

HIPOTESES:

Nossa hipótese inicial são:

RQ 01. Sistemas populares são maduros/antigos?

Hipotese informal: Acredito que os sistemas mais populares sejam também os mais antigos, pois tiveram mais tempo para se desenvolver.

RQ 02. Sistemas populares recebem muita contribuição externa?

Hipotese informal: Projetos populares devem receber muitas contribuições externas, o que os torna mais acessíveis, conhecidos e alinhados às necessidades dos usuários.

RQ 03. Sistemas populares lançam releases com frequência?

Hipotese informal: Espera-se que os sistemas populares lancem releases frequentemente, acompanhando a evolução constante do software.

RQ 04. Sistemas populares são atualizados com frequência?

Hipotese informal: Repositórios populares devem ser atualizados com frequência para se manterem relevantes e não ficarem ultrapassados.

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

Hipotese informal: Os sistemas populares devem ser escritos em linguaguem que tambem são populares, ja que isso fornece uma base maior de contribuição e ferramentas para o sistema

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

Hipotese informal: Acredita-se que sistemas populares tenham um alto percentual de issues fechadas, pois contam com muitos colaboradores ativos.

RQ7: Sistemas escritos em linguagens mais populares recebem mais contribuição externa, lançam mais releases e são atualizados com mais frequência?

Metodologia:

A fonte dos dados foi a API GraphQL do GitHub, acessada por meio de um script próprio em Python. Para garantir conformidade com as restrições do trabalho, foram utilizadas apenas bibliotecas padrão: **urllib, json, csv, datetime e pathlib**.

A query GraphQL foi construída para recuperar os 100 repositórios públicos com maior número de estrelas (em ordem decrescente), extraindo os seguintes campos:

- **RQ01:** `createdAt`
- **RQ02:** `pullRequests(states: MERGED)`
- **RQ03:** `releases { totalCount }`
- **RQ04:** `updatedAt` e `pushedAt`
- **RQ05:** `primaryLanguage { name }`
- **RQ06:** `issues` e `closedIssues`

Além disso, foram coletados os atributos **id, name, owner, url e stargazerCount**, para fins de identificação e análise de popularidade.

Para a **RQ07**, realizou-se uma análise cruzada entre os atributos já coletados, relacionando a linguagem principal (`primaryLanguage`) com métricas de contribuição externa (`prsMerged`), versionamento (`releases`) e atualização (`dias_desde_ultima_atualizacao`).

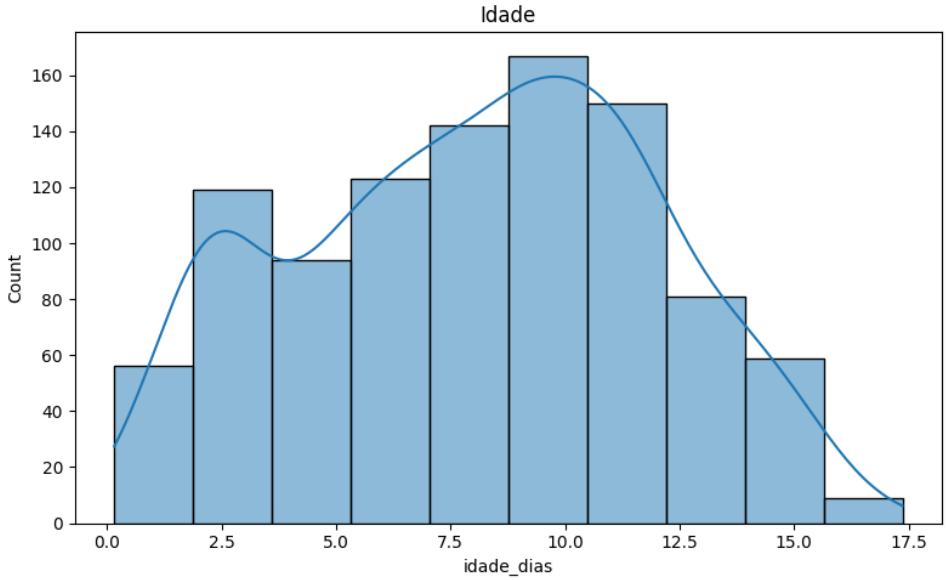
O tratamento dos dados seguiu o seguinte fluxo:

- Conversão de datas ISO para dias/anos.
- Cálculo de medianas para cada métrica (e médias em casos específicos, como tempo de atualização).
- Normalização dos dados em estruturas JSON e CSV.
- Geração de tabelas e gráficos para visualização exploratória.

RESULTADOS:

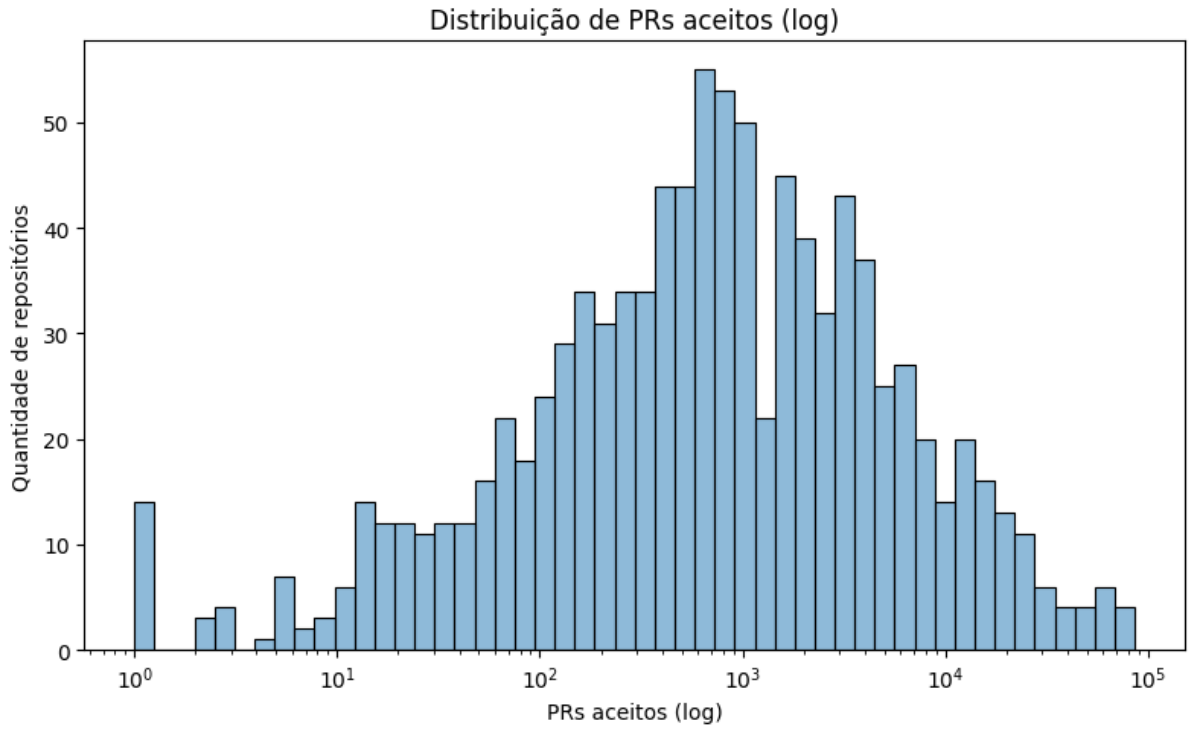
RQ1 – Idade dos repositórios:

A distribuição da idade mostra que a maior parte dos repositórios analisados possui entre 5 e 12 anos de criação, com pico em torno de 10 anos. Isso sugere que os repositórios da amostra são relativamente recentes, predominando projetos de curta duração ou ainda em fase inicial de desenvolvimento.



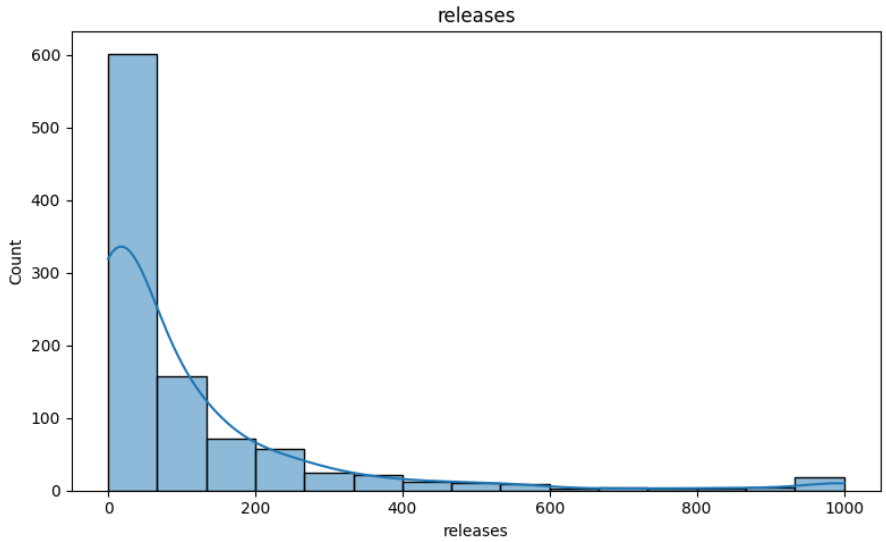
RQ2 – Pull Requests aceitos:

A análise em escala logarítmica mostra uma **grande dispersão** no número de *pull requests* aceitos. A distribuição revela que muitos repositórios acumulam entre **centenas e alguns milhares de PRs aceitos**, enquanto uma minoria concentra valores extremamente altos, chegando a dezenas de milhares. A mediana está próxima de mil PRs aceitos, indicando que metade dos projetos analisados recebeu pelo menos esse volume de contribuições externas. Isso evidencia que, embora a colaboração externa seja significativa em vários casos, ela é **desigual e concentrada em poucos projetos de grande porte**.



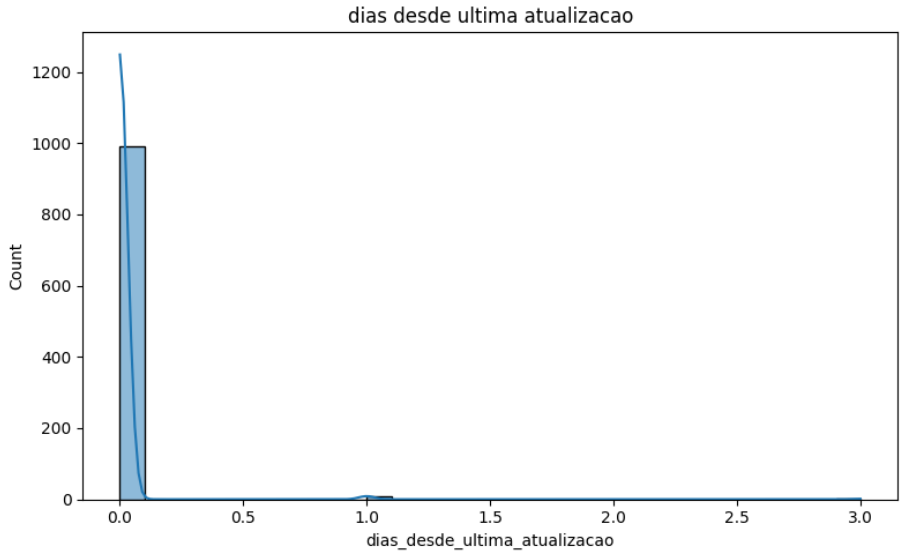
RQ3 – Releases:

A análise mostra que a maioria dos repositórios possui nenhuma ou poucas releases. O gráfico apresenta concentração próxima a zero, o que evidencia que a prática de formalizar versões de software é pouco adotada pela maior parte dos projetos.



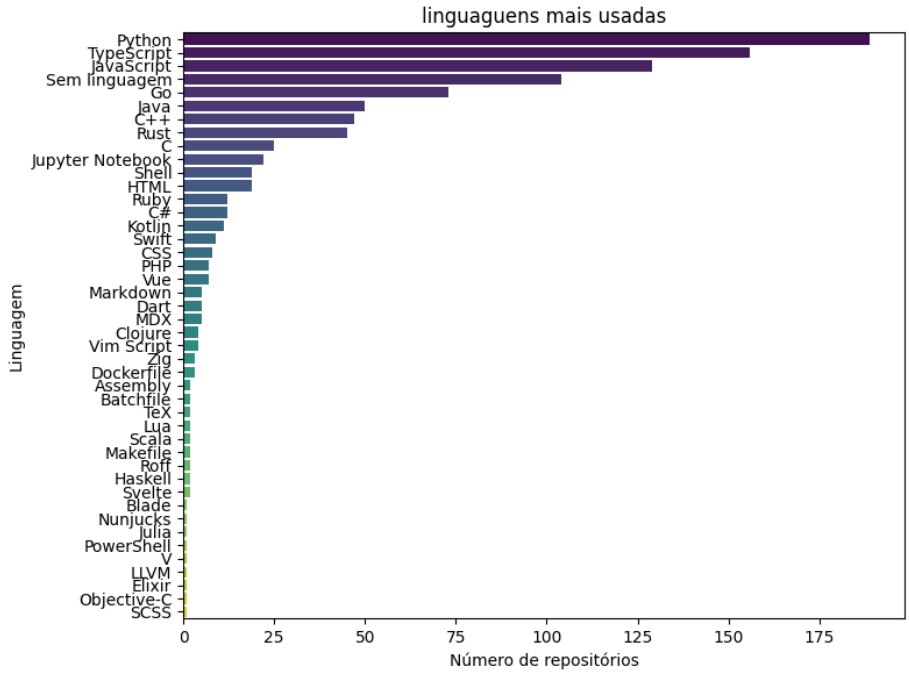
RQ4 – Dias desde a última atualização:

Observa-se que a grande maioria dos repositórios foi atualizada recentemente, com valores concentrados próximos de zero dias. Isso sugere que boa parte da amostra é composta por projetos ativos, ainda que nem todos recebam contribuições expressivas.



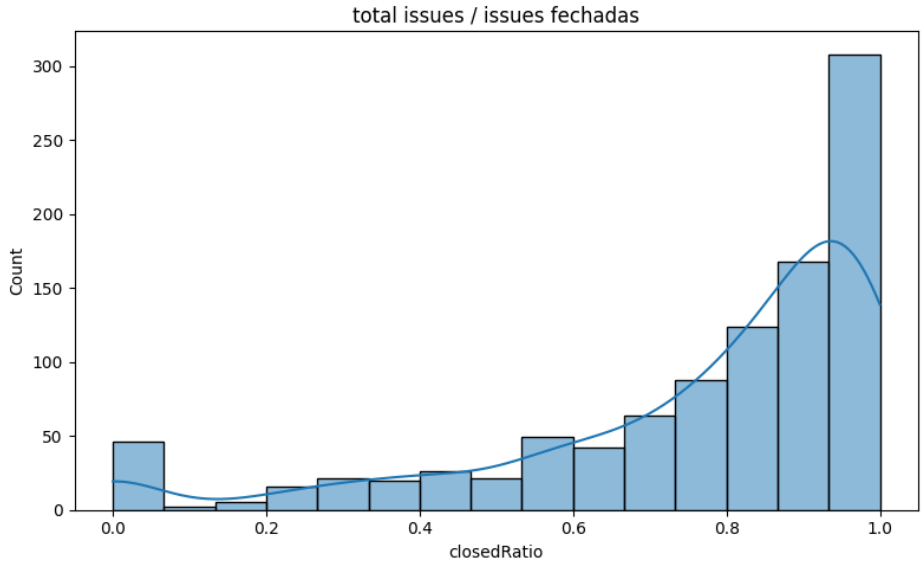
RQ5 – Linguagens mais utilizadas:

O ranking das linguagens indica predominância de Python, TypeScript e JavaScript, seguidos por Go, Java e C++. Essas linguagens concentram a maior parte dos repositórios, confirmando sua popularidade no ecossistema GitHub.



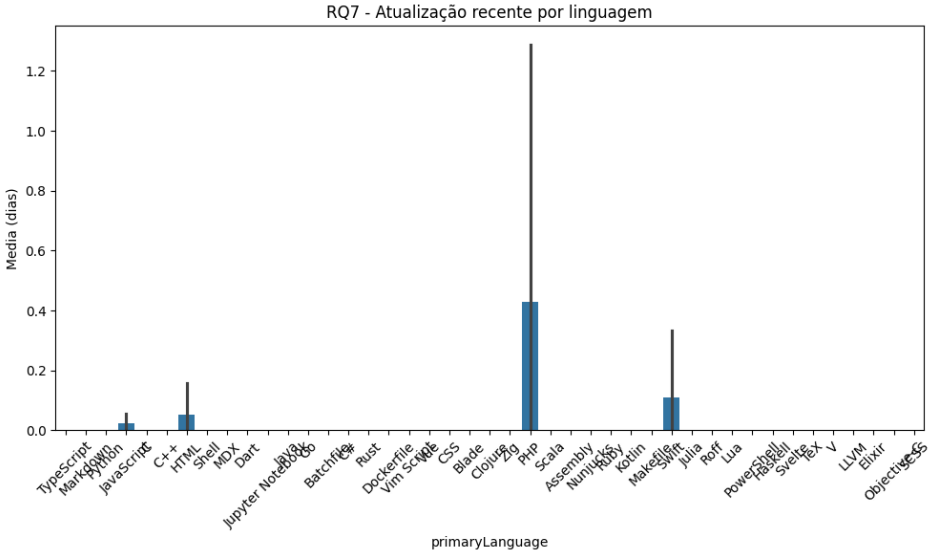
RQ6 – Relação entre issues totais e fechadas:

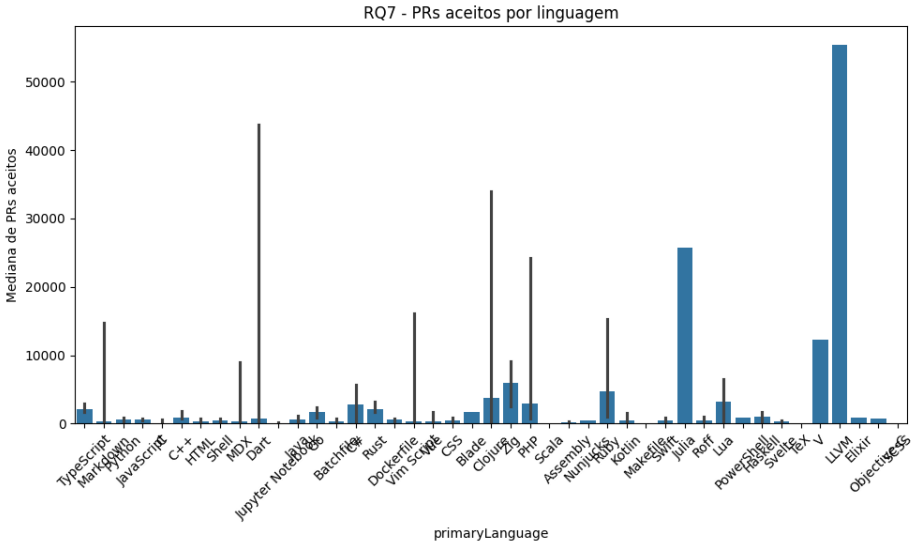
A distribuição da razão entre *issues* fechadas e o total de *issues* mostra que a maioria dos repositórios apresenta valores elevados, concentrando-se entre **0.7 e 1.0**, ou seja, entre 70% e 100% das *issues* registradas foram fechadas. Ainda existem casos isolados de repositórios com baixa taxa de fechamento, mas eles representam a minoria. Esses resultados indicam que os projetos populares, em geral, mantêm uma **boa gestão das issues**, evidenciando um esforço ativo de manutenção e resolução de problemas.

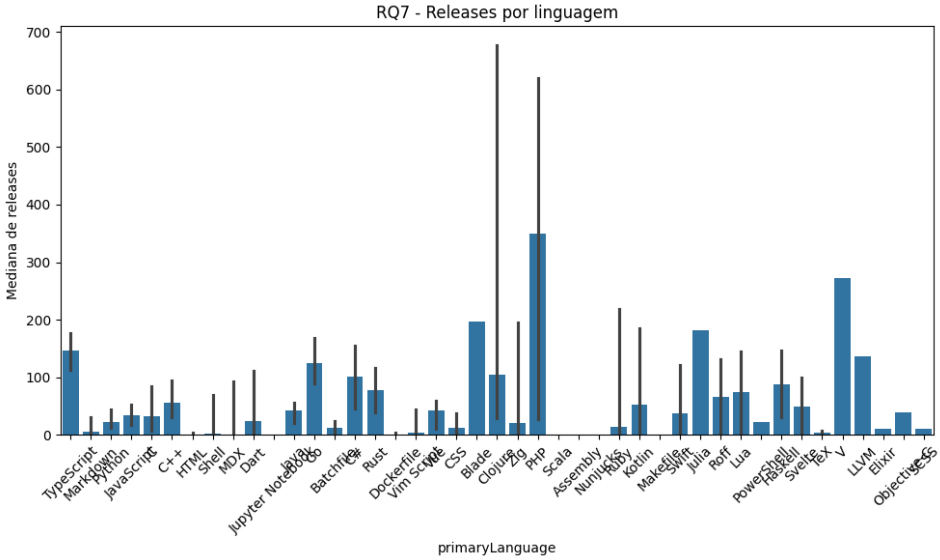


RQ7:

A análise cruzada entre linguagens e métricas de colaboração mostrou diferenças relevantes. No caso das contribuições externas (*pull requests* aceitos), observou-se que linguagens populares como **Python, JavaScript e TypeScript** concentram os maiores valores, com medianas na casa dos milhares, enquanto outras linguagens apresentam números bem menores. Em relação às *releases*, a maior parte das linguagens apresentou medianas baixas, reforçando que o lançamento formal de versões não é prática predominante, embora algumas linguagens (como **PHP, Scala e V**) tenham registrado valores mais elevados. Já quanto à atualização recente, os resultados mostraram que a maioria dos projetos é atualizada em períodos muito curtos, com médias próximas de zero dias, indicando forte atividade de manutenção independentemente da linguagem utilizada.


RQ7 - Atualização recente por linguagem
Media (dias)
primaryLanguage


RQ7 - PRs aceitos por linguagem
Mediana de PRs aceitos
primaryLanguage


RQ7 - Releases por linguagem
Mediana de releases
primaryLanguage

Conclusões:

RQ1:

A distribuição da idade mostra que a maior parte dos repositórios analisados possui entre **5 e 12 anos**, com pico em torno de 10 anos. Isso indica que os repositórios da amostra são relativamente antigos, predominando projetos já consolidados ao longo do tempo.

Na RQ1, a hipótese era de que os sistemas mais populares seriam também os mais antigos. Os resultados confirmam essa expectativa, pois os dados mostram que a maior parte dos repositórios analisados já possui vários anos de existência. Isso sugere que a popularidade tende a estar associada à maturidade temporal, provavelmente porque repositórios mais antigos tiveram mais tempo para acumular contribuições, consolidar comunidade e ganhar visibilidade.

RQ2:

Na RQ2, esperava-se que projetos populares recebessem muitas contribuições externas. Os resultados **confirmaram parcialmente essa hipótese**: de fato, diversos repositórios analisados apresentam centenas ou até milhares de *pull requests* aceitos, indicando participação ativa da comunidade. No entanto, a distribuição é bastante desigual, já que apenas uma minoria de projetos concentra valores extremamente altos, enquanto outros recebem contribuições em volume bem menor. Isso demonstra que a popularidade pode favorecer a colaboração, mas não garante que ela ocorra de forma uniforme entre todos os projetos.

RQ3:

Na RQ3, acreditava-se que sistemas populares lançariam releases com frequência. Os resultados **não confirmaram essa hipótese**, já que a grande maioria dos repositórios apresentou poucas ou nenhuma release formal. Isso sugere que, mesmo em projetos populares, o versionamento por meio de releases não é uma prática dominante, possivelmente substituída por modelos de entrega contínua ou distribuição direta do código.

RQ4:

Na RQ4, a expectativa era de que repositórios populares fossem atualizados frequentemente. Os resultados **confirmaram essa hipótese**, mostrando que a

maioria dos repositórios apresenta registros de atualização muito recentes. Esse achado reforça que a popularidade está fortemente ligada à atividade contínua, indicando que projetos ativos atraem e mantêm mais atenção da comunidade.

RQ5:

Na RQ5, supunha-se que sistemas populares seriam escritos em linguagens populares. Essa hipótese foi confirmada, já que Python, TypeScript e JavaScript apareceram como as mais frequentes na amostra, refletindo a alta adoção dessas linguagens no ecossistema.

RQ6:

Na RQ6, a hipótese era de que sistemas populares possuiriam um alto percentual de *issues* fechadas. Os resultados **confirmaram essa expectativa**, uma vez que a maior parte dos repositórios apresentou proporções elevadas de fechamento, em alguns casos próximas de 100%. Isso evidencia que muitos dos projetos analisados têm uma boa gestão de *issues* e contam com colaboradores ativos no processo de manutenção.

RQ7:

Na RQ7, a hipótese era de que sistemas escritos em linguagens mais populares receberiam mais contribuições externas, lançariam mais releases e seriam atualizados com maior frequência. Os resultados **confirmaram parcialmente essa expectativa**. De fato, linguagens amplamente utilizadas como Python, JavaScript e TypeScript concentram volumes significativamente maiores de contribuições externas e apresentam manutenção frequente, corroborando parte da hipótese. No entanto, a frequência de *releases* não segue o mesmo padrão: mesmo em linguagens populares, os lançamentos formais tendem a ser pouco expressivos, com exceções pontuais em linguagens como PHP e Scala. Dessa forma, pode-se concluir que a popularidade da linguagem está fortemente associada à colaboração e à atualização contínua, mas não necessariamente ao número de releases formais.