Relatório de Análise de Repositórios Populares do GitHub

Alunos: Pedro Reis e Gabriel Fernandes

Introdução

Este estudo tem como objetivo analisar as características comuns entre os 1.000 repositórios mais populares do GitHub, mensurados pelo número de estrelas, a fim de entender os padrões que podem levar ao sucesso de um projeto de código aberto.

Para guiar nossa análise, formulamos as seguintes hipóteses informais para cada questão de pesquisa (RQ):

- **RQ01 (Idade):** Esperamos que sistemas populares sejam maduros, com vários anos de desenvolvimento contínuo.
- **RQ02 (Contribuição Externa):** Acreditamos que projetos de sucesso recebem um volume muito alto de contribuições externas (Pull Requests).
- **RQ03 (Releases):** Nossa hipótese é que projetos populares lançam novas versões (releases) com boa frequência para entregar valor aos usuários.
- **RQ04 (Atualização):** Esperamos que esses sistemas sejam atualizados constantemente, com novas contribuições de código (pushes) ocorrendo quase diariamente.
- **RQ05 (Linguagens):** Acreditamos que as linguagens mais populares na indústria, como o ecossistema JavaScript (incluindo TypeScript) e Python, dominarão a lista.
- **RQ06 (Issues):** Projetos populares devem ter uma boa gestão de issues, resultando em uma alta porcentagem de issues fechadas.
- **RQ07 (Bônus):** Acreditamos que as métricas de atividade (contribuições, releases e atualizações) podem variar significativamente entre as diferentes comunidades de linguagens de programação.

Metodologia

Para responder às questões de pesquisa, adotamos a seguinte metodologia, dividida em três etapas principais:

1. **Coleta de Dados:**
   - Utilizamos um script em Python para interagir com a **API GraphQL do GitHub (v4)**.
   - A busca foi configurada para retornar os 1.000 repositórios com o maior número de estrelas (`query: "stars:>=1 sort:stars-desc"`).
   - Para evitar sobrecarga na API e erros de timeout, a coleta foi dividida em duas fases: uma busca inicial "leve" para obter a lista dos repositórios e, em seguida, requisições individuais para cada um a fim de obter métricas detalhadas.
   - As métricas coletadas para cada repositório foram: nome, estrelas, data de criação, data do último push, linguagem primária, contagem de pull requests aceitas, contagem de releases e contagens de issues totais e fechadas.
   - Os dados brutos foram salvos em um arquivo no formato **CSV** (`repositorios_graphql_completo.csv`).
2. **Processamento e Análise de Dados:**
   - O arquivo CSV foi carregado em um DataFrame utilizando a biblioteca **Pandas**.
   - Novas métricas foram calculadas a partir dos dados brutos, como a idade do repositório em dias, os dias desde o último push e a razão de issues fechadas.

- A análise focou no cálculo de valores de tendência central, especificamente a **mediana**, por ser uma medida mais robusta a outliers em distribuições assimétricas, comuns em dados de popularidade. Para dados categóricos (linguagens), realizamos a contagem de frequência.

3. **Visualização e Geração do Relatório:**
   - Utilizamos as bibliotecas **Matplotlib** e **Seaborn** para gerar visualizações gráficas (histogramas e gráficos de barras) para cada questão de pesquisa.
   - Um script final em Python foi responsável por consolidar todas as análises, textos e gráficos em um único relatório no formato **HTML**, que foi posteriormente convertido para **PDF**.

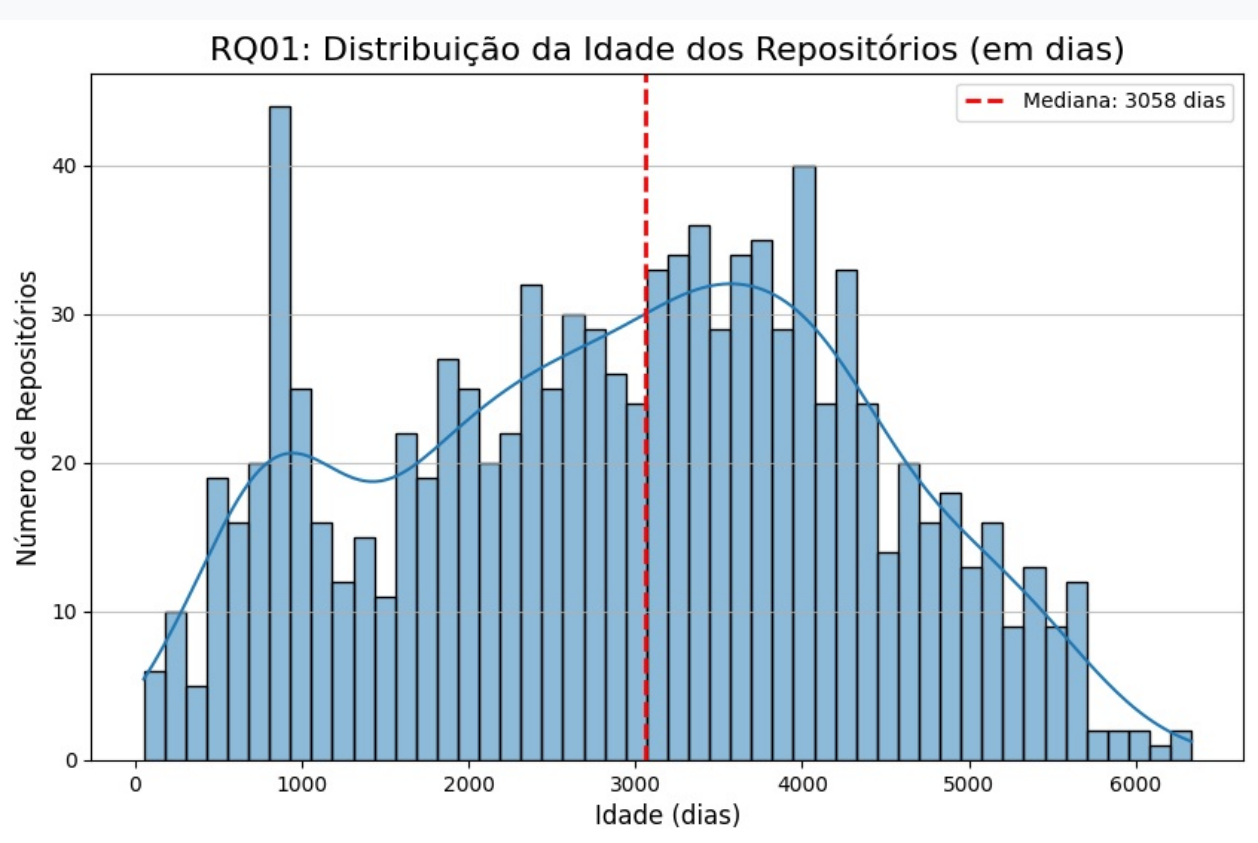
Resultados e Discussões

A seguir, apresentamos os resultados detalhados para cada questão de pesquisa.

RQ01: Sistemas populares são maduros/antigos?

Análise: A mediana da idade dos 1.000 repositórios mais populares é de **3058 dias** (aproximadamente **8.4 anos**).

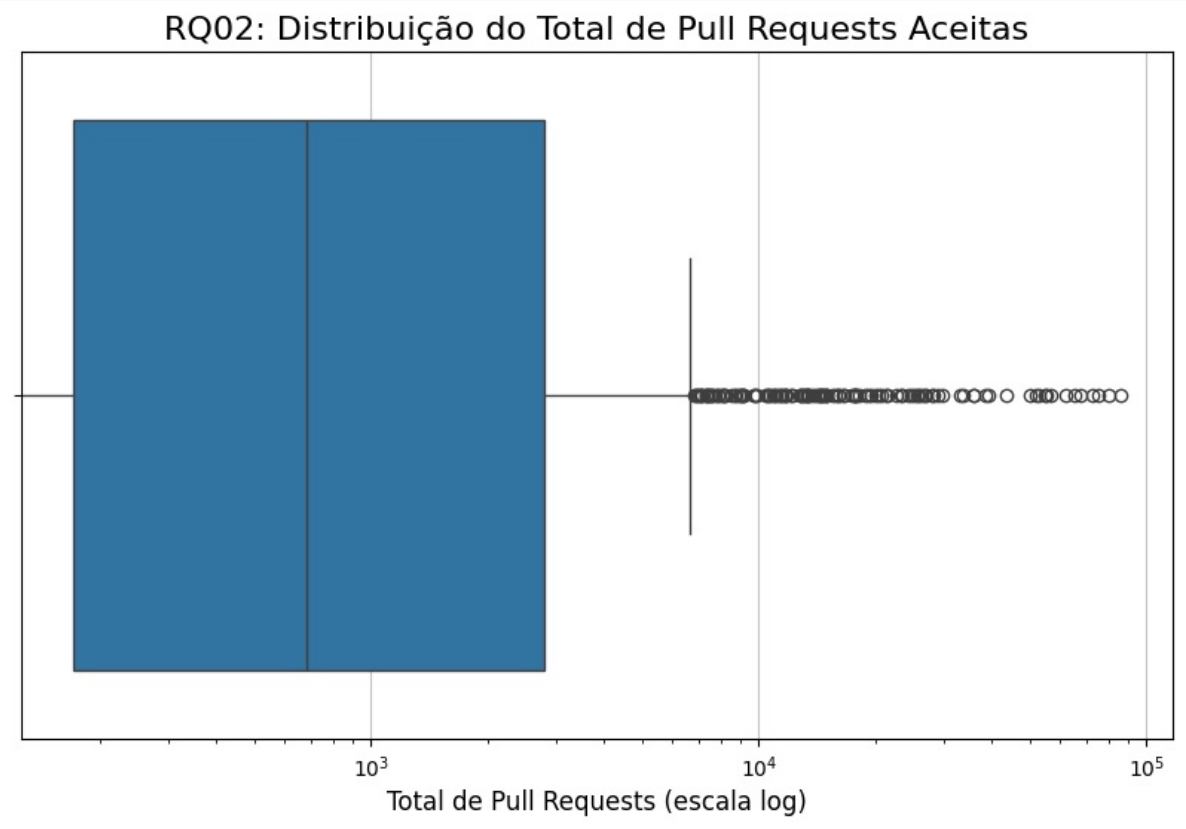
Discussão: O valor mediano de quase 8.4 anos confirma a hipótese de que a maioria dos repositórios populares não é recente, possuindo um tempo considerável de existência e desenvolvimento.



RQ02: Sistemas populares recebem muita contribuição externa?

Análise: A mediana do total de pull requests aceitas é de **683**.

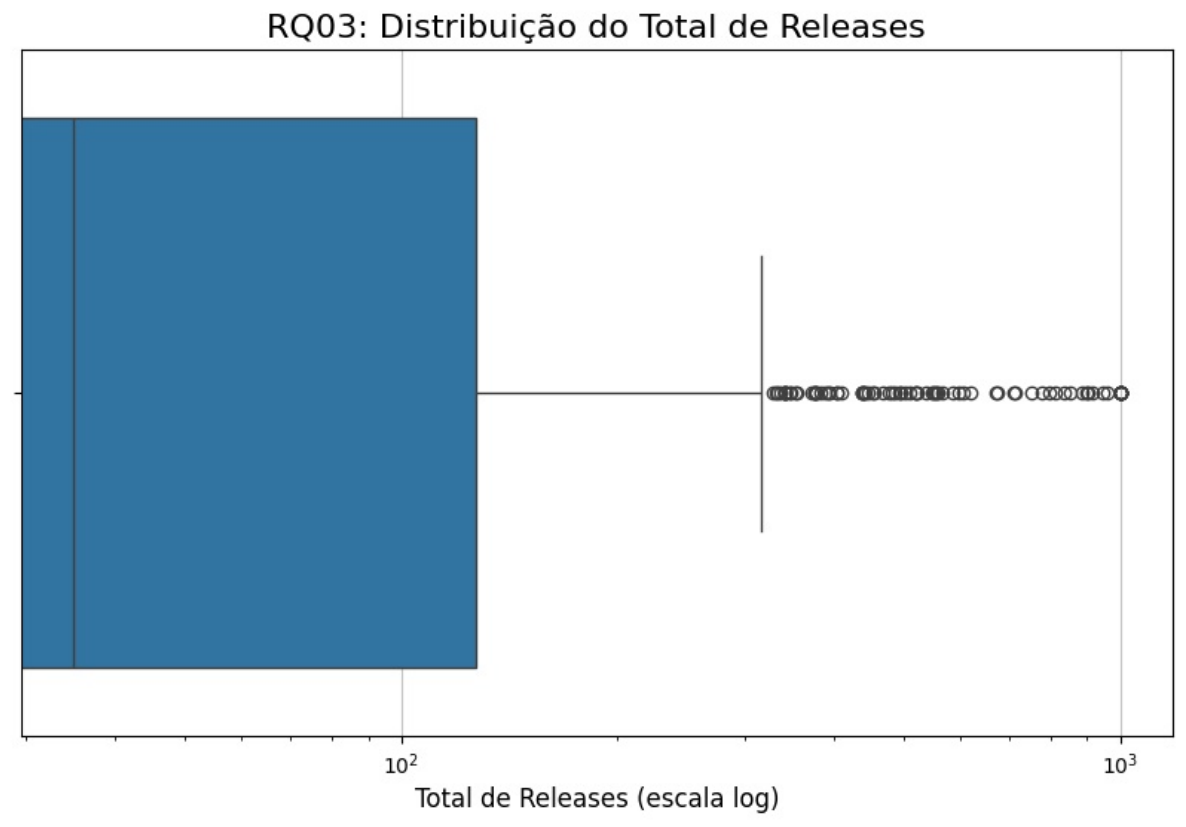
Discussão: Este valor mediano indica um volume significativo e constante de contribuições da comunidade. O gráfico de distribuição (boxplot) mostra que, embora a mediana seja alta, existem repositórios (outliers) que recebem um volume de contribuições ordens de magnitude maior, como frameworks e bibliotecas de uso massivo.



RQ03: Sistemas populares lançam releases com frequência?

Análise: A mediana do total de releases é de **35**.

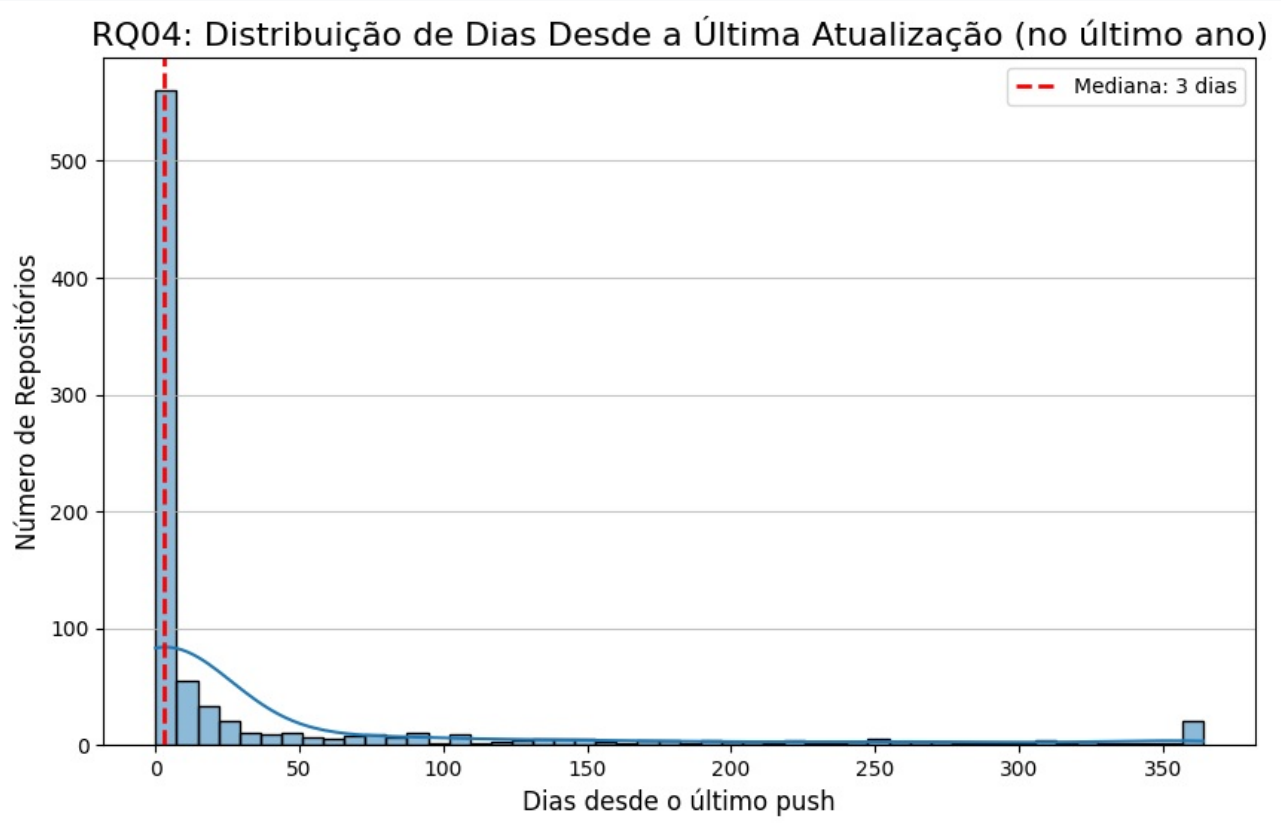
Discussão: A mediana sugere que a prática de versionamento formal via releases é bem estabelecida entre os projetos populares. Uma mediana de 35 releases ao longo da vida do projeto indica um ciclo de desenvolvimento maduro e organizado.



RQ04: Sistemas populares são atualizados com frequência?

Análise: A mediana de dias desde a última atualização é de **3 dias**.

Discussão: Uma mediana de 3 dias confirma que são projetos extremamente ativos, com metade dos repositórios tendo recebido código novo neste curto período.



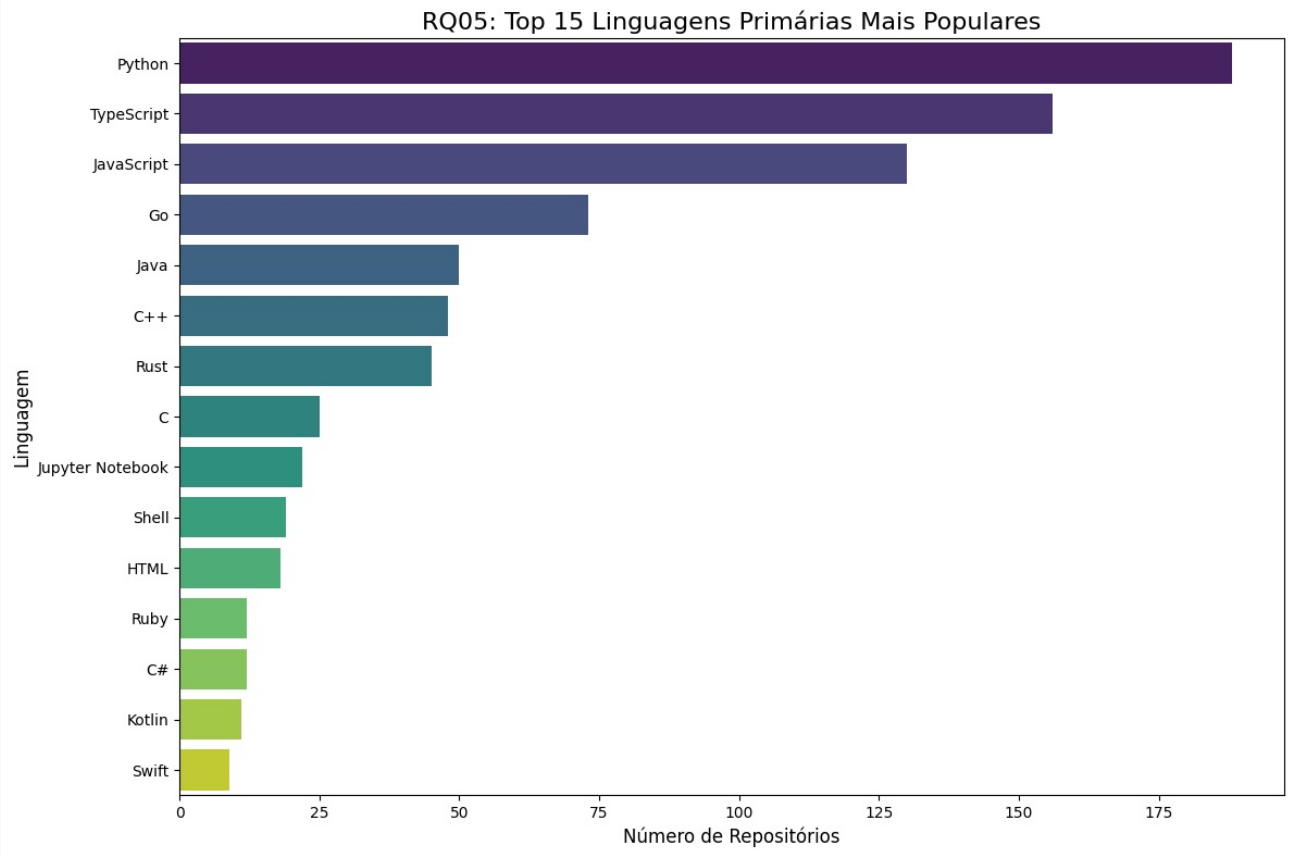
RQ05: Sistemas populares são escritos nas linguagens mais populares?

Análise: A contagem das 15 linguagens primárias mais frequentes é apresentada abaixo.

Discussão: Conforme esperado, o ecossistema JavaScript e Python são dominantes. A presença de Go, Rust e Kotlin reflete tendências modernas no desenvolvimento de software.

Contagem de Repositórios por Linguagem (Top 15)

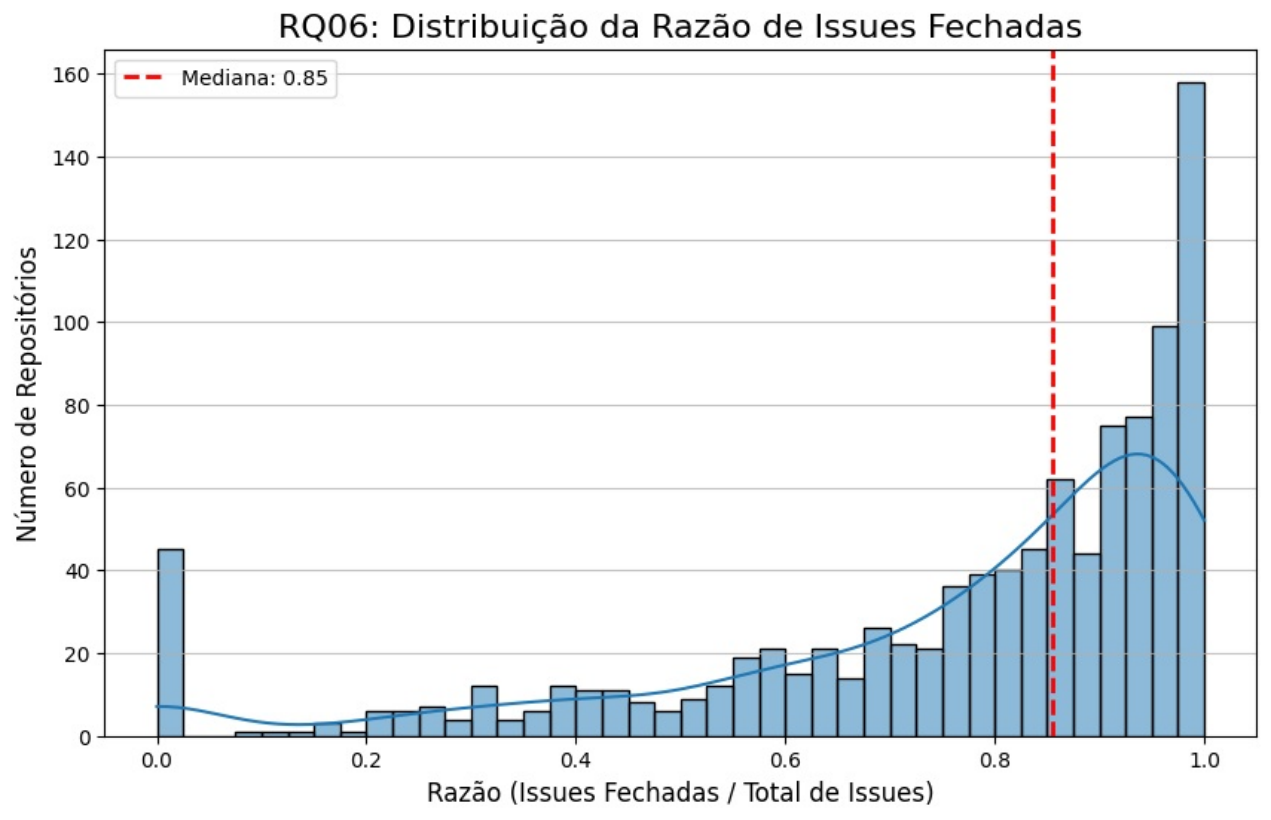
	count
linguagem_primaria	
Python	188
TypeScript	156
JavaScript	130
Go	73
Java	50
C++	48
Rust	45
C	25
Jupyter Notebook	22
Shell	19
HTML	18
Ruby	12
C#	12
Kotlin	11
Swift	9



RQ06: Sistemas populares possuem um alto percentual de issues fechadas?

Análise: A mediana da razão entre issues fechadas e o total de issues é de **0.85** (ou **85%**).

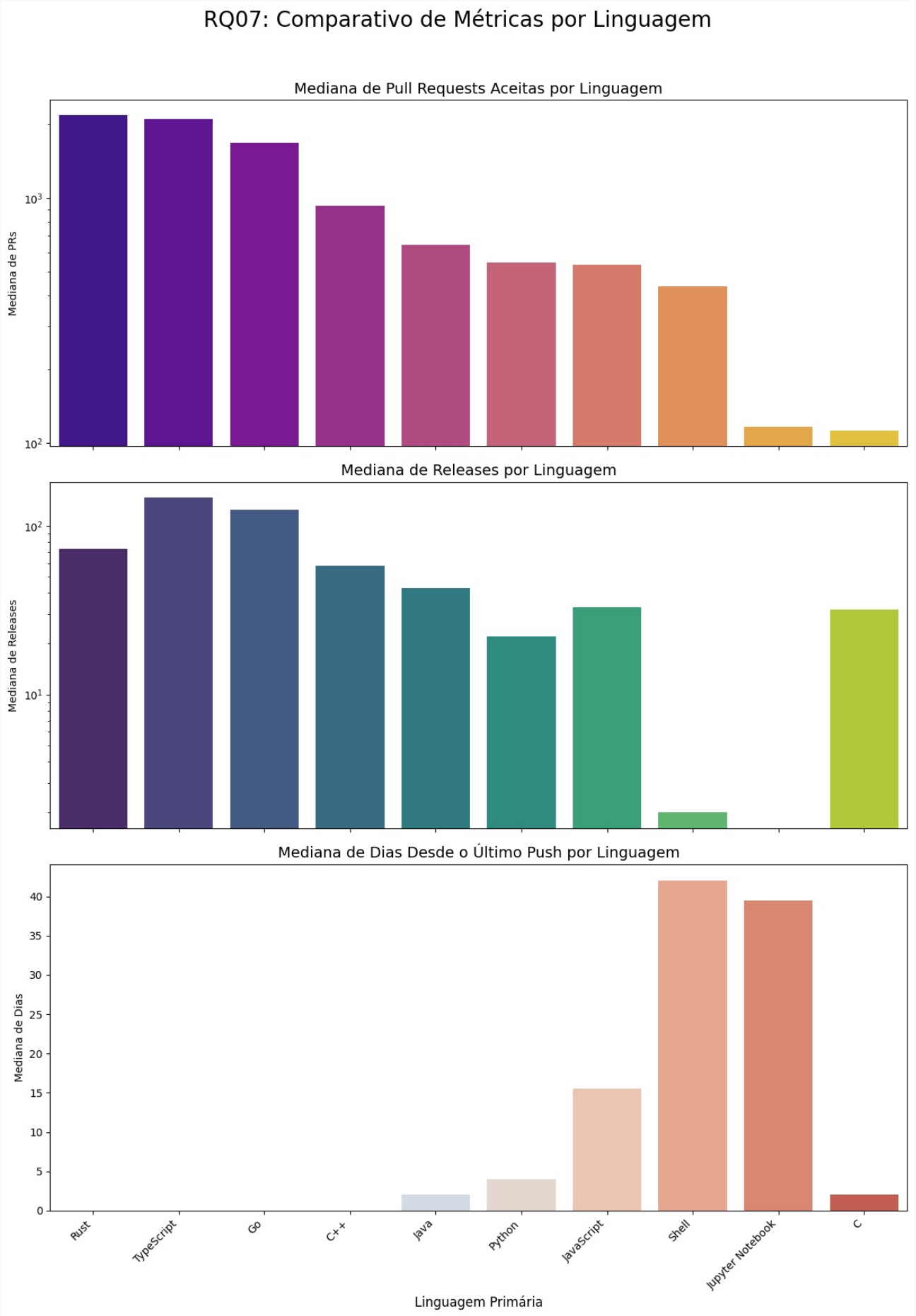
Discussão: Uma mediana de 85% é um indicador muito forte de saúde e boa manutenção do projeto. Mostra que a maioria dos projetos populares consegue gerenciar e resolver a grande maioria dos problemas e sugestões que recebem da comunidade.



RQ07 (Bônus): As métricas de atividade variam entre as linguagens mais populares?

Análise: O gráfico abaixo compara a mediana de Pull Requests, Releases e Dias desde o Último Push para as 10 linguagens mais populares no dataset.

Discussão: A análise revela nuances interessantes. Por exemplo, linguagens como Rust e Go, apesar de modernas, mostram um alto volume mediano de contribuições, refletindo comunidades vibrantes. A frequência de atualização (menor número de dias desde o último push) é consistentemente baixa em todas as linguagens populares, confirmando que todos são projetos ativos. As diferenças na mediana de releases podem indicar culturas de desenvolvimento distintas entre as comunidades de cada linguagem.


RQ07: Comparativo de Métricas por Linguagem
Mediana de Pull Requests Aceitas por Linguagem
Mediana de PRs
10^3
10^2
Mediana de Releases por Linguagem
Mediana de Releases
10^2
10^1
Mediana de Dias Desde o Último Push por Linguagem
Mediana de Dias
40
35
30
25
20
15
10
5
0
Rust
TypeScript
Go
C++
Java
Python
JavaScript
Shell
Jupyter Notebook
C
Linguagem Primária