Relatório Final: Análise de Repositórios Populares do GitHub

Alunos:

- Eric Rodrigues Diniz
- Ian Nascimento Rocha
- Pablo Guilherme Amâncio Pereira Magela Benevenuto

- Introdução

A popularidade de um repositório no GitHub pode ser influenciada por diversos fatores, como o tempo de existência, a quantidade de contribuições externas recebidas e a frequência de releases e atualizações. Esta análise busca identificar padrões nos repositórios populares da plataforma, avaliando sua maturidade, contribuição externa, ciclos de lançamento e manutenção ativa.

As hipóteses que fundamentam o estudo são:

I. **H1:** Repositórios mais antigos tendem a ter um maior número de contribuições e releases.
II. **H2:** Linguagens populares como Python e TypeScript permanecerão como as principais nos próximos 10 anos.
III. **H3:** A taxa de fechamento de issues continuará sendo um indicador relevante para medir a manutenção ativa de um projeto.

- Metodologia

Para responder às questões de pesquisa (RQs), realizamos uma coleta de dados via API GraphQL do GitHub, extraindo informações de 1000 repositórios populares. Os dados foram processados com **Pandas** e **Matplotlib**, possibilitando a análise exploratória e a geração de visualizações. A metodologia seguiu as etapas:

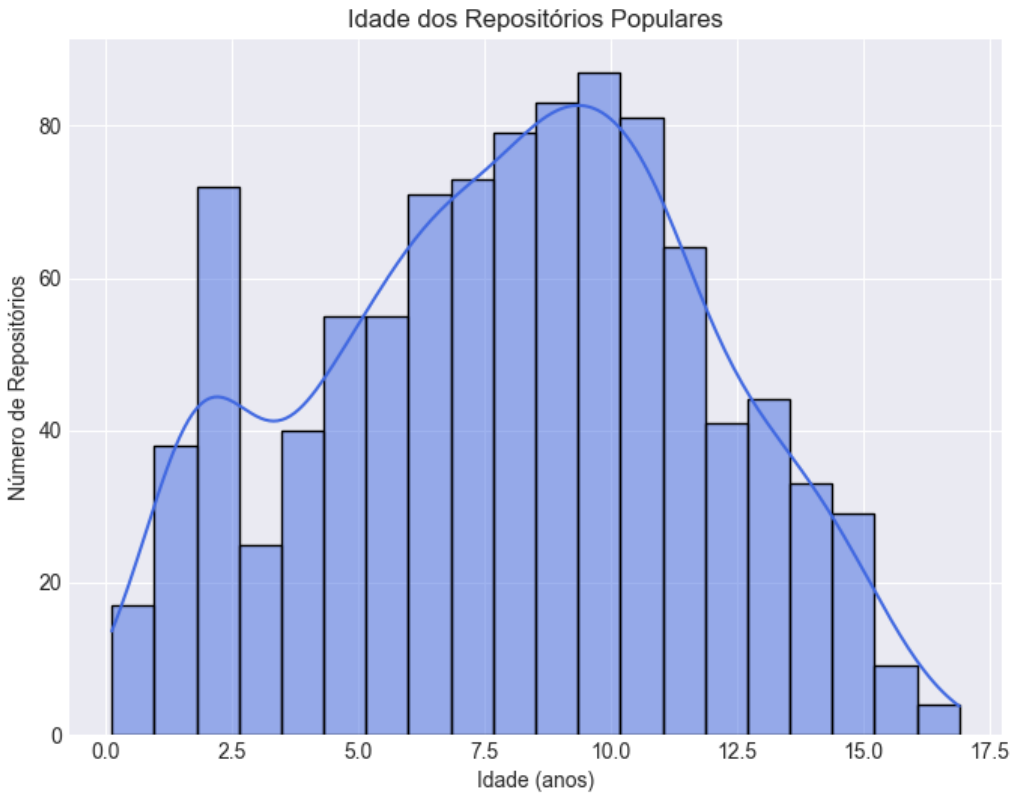
1. **Coleta de dados**: Consultas à API GraphQL para obter informações estruturadas.
2. **Análise exploratória**: Limpeza e organização dos dados.

3. **Geração de visualizações**: Construção de gráficos para apoiar a interpretação dos resultados.
4. **Discussão**: Comparar os resultados com as hipóteses formuladas.

- Resultados

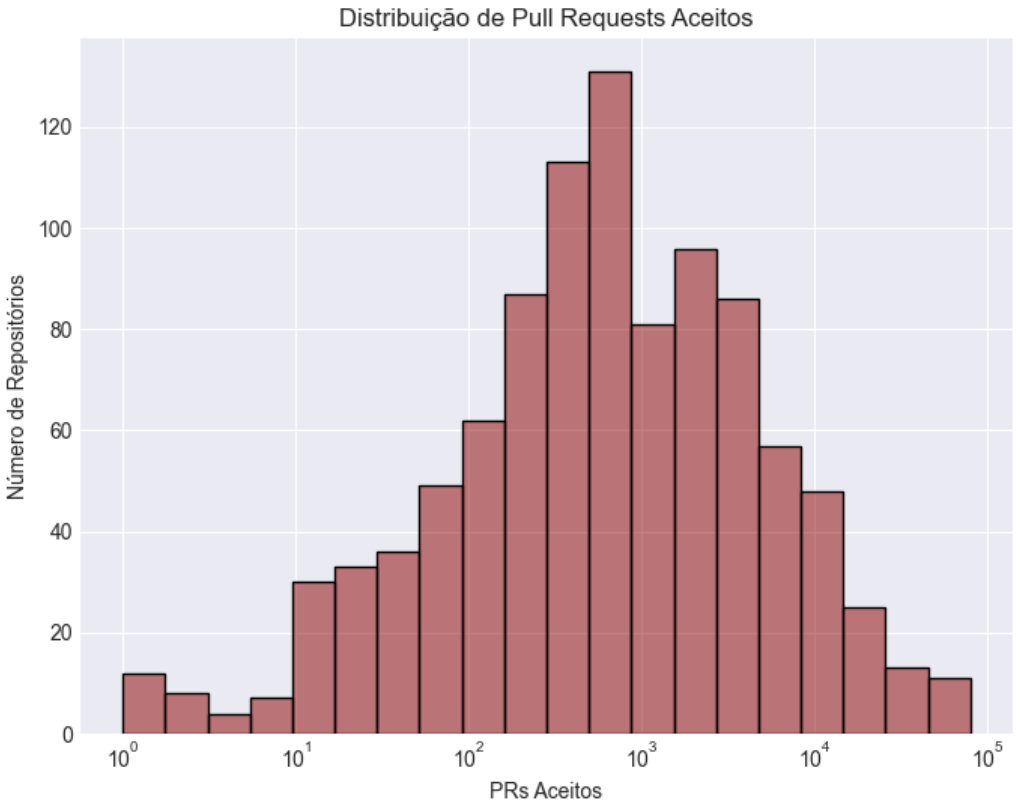
3.1. RQ01: Sistemas populares são maduros/antigos?

- A maioria dos repositórios tem entre **5 e 10 anos**, confirmando que muitos projetos populares têm um histórico consolidado.
- Projetos mais antigos continuam relevantes, mas em menor quantidade.



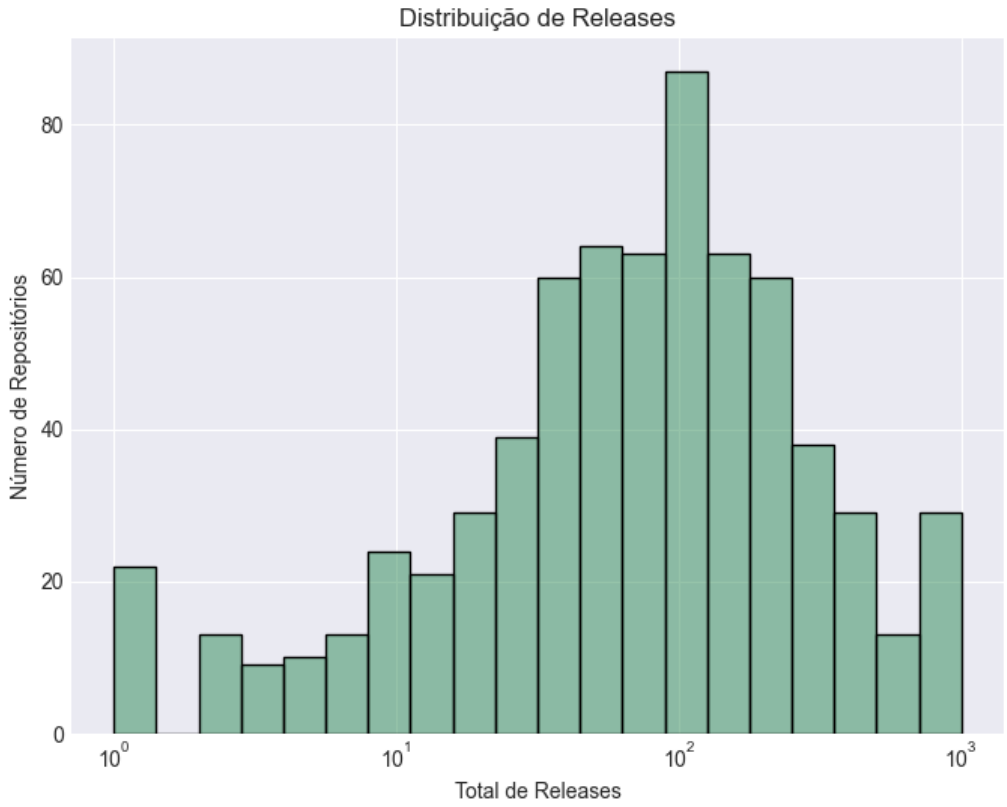
3.2. RQ02: Sistemas populares recebem muita contribuição externa?

- A maior parte dos repositórios apresenta um número **moderado** de pull requests aceitos.
- Poucos repositórios ultrapassam **10.000 pull requests aceitos**.
- **Analisando por linguagem**, repositórios em **Python e TypeScript tendem a ter maior contribuição externa**.



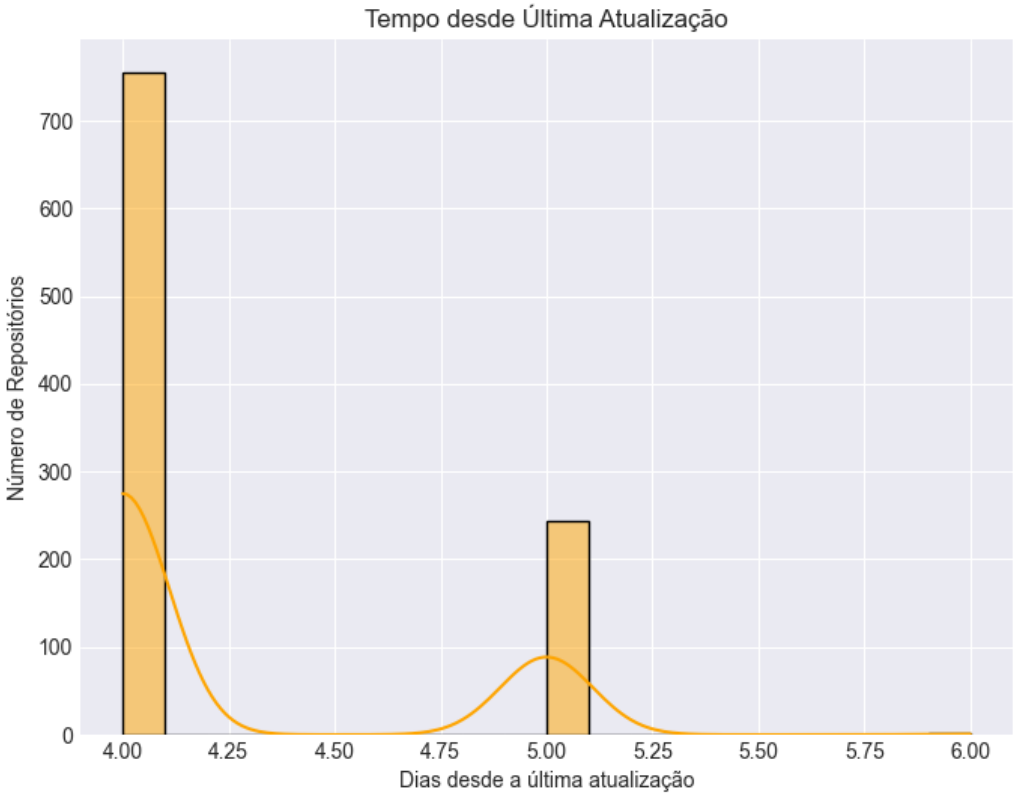
3.3. RQ03: Sistemas populares lançam releases com frequência?

- A distribuição de releases é **exponencial**, com a maioria dos projetos tendo um **número moderado de releases**.
- **Repositórios em TypeScript e JavaScript tendem a lançar releases mais frequentemente**.



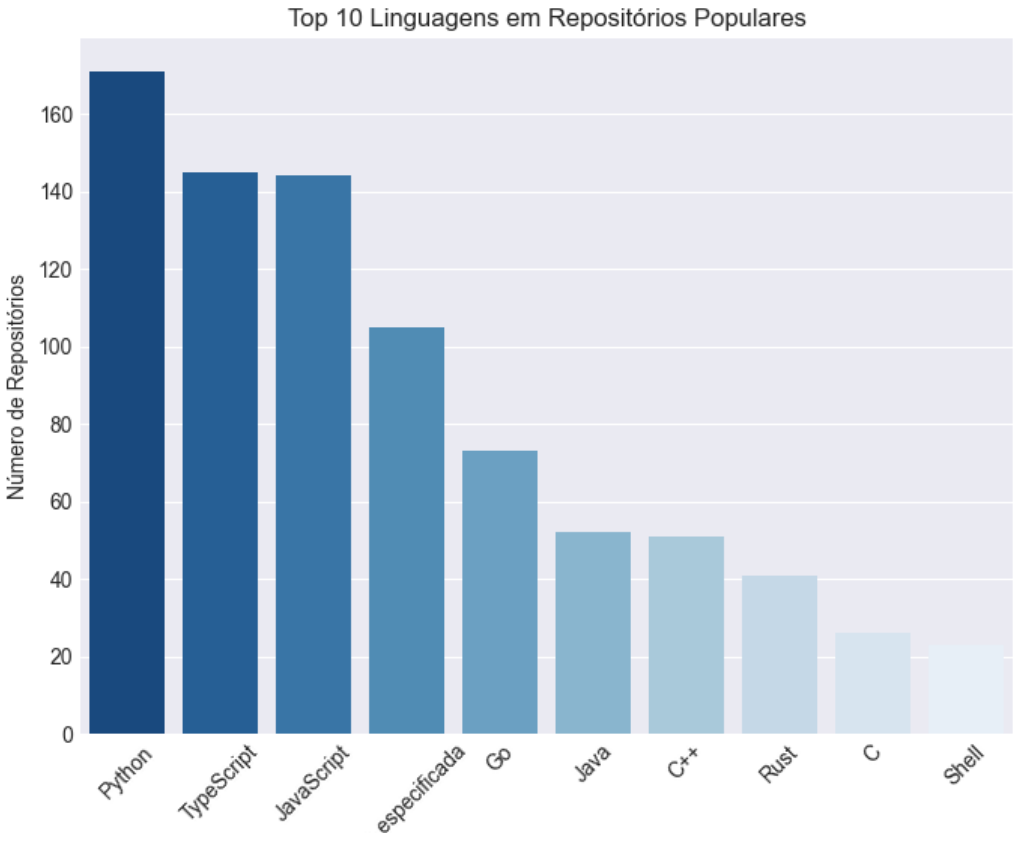
3.4. RQ04: Sistemas populares são atualizados com frequência?

- A grande maioria dos repositórios foi **atualizada recentemente**, indicando que projetos populares são frequentemente mantidos.
- **Projetos em Rust e Go apresentam tempos de atualização mais curtos em comparação com Java e C++**.



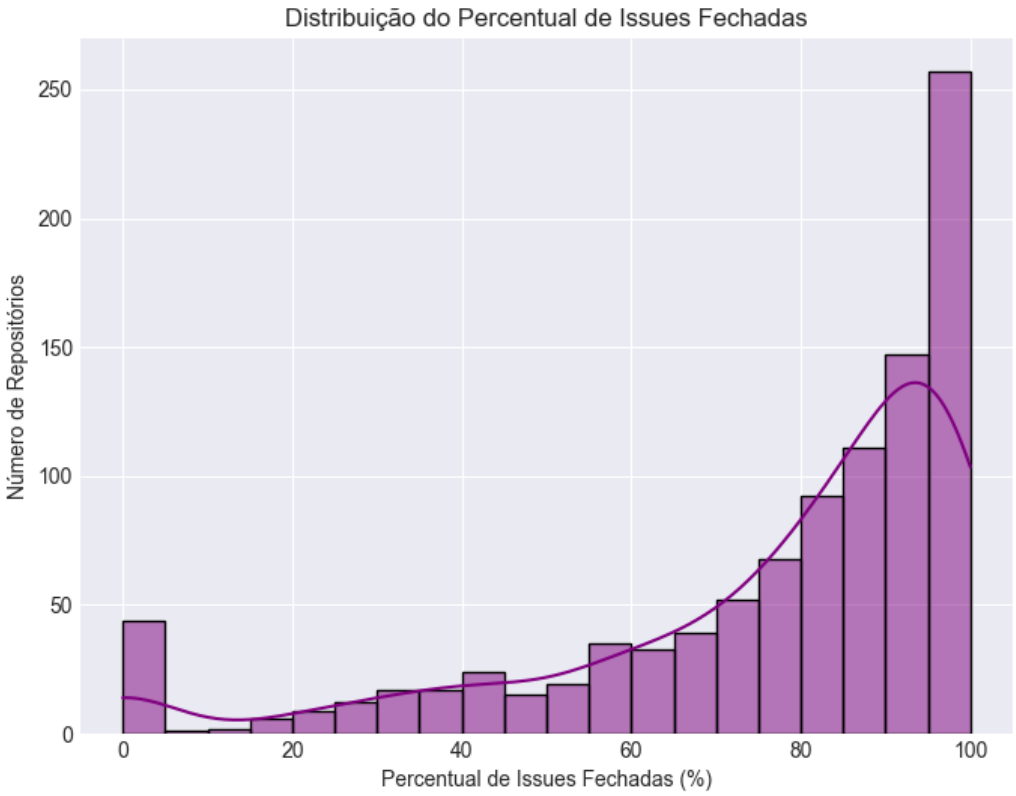
3.5. RQ05: Sistemas populares são escritos nas linguagens mais populares?

- As linguagens mais comuns foram **Python, TypeScript e JavaScript**, confirmando que essas tecnologias dominam o cenário open-source.



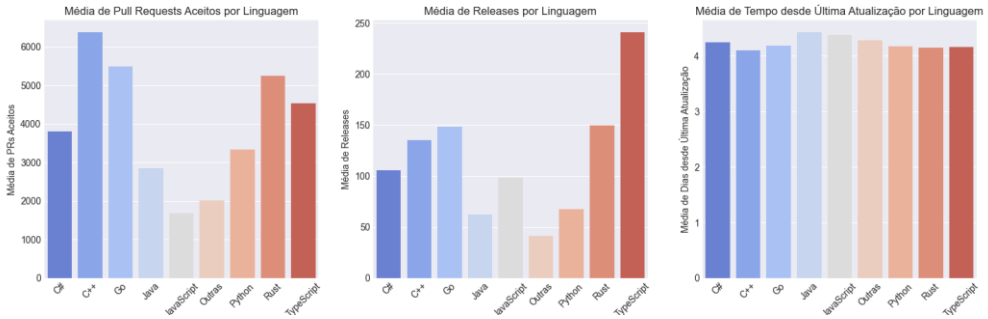
3.6. RQ06: Sistemas populares possuem um alto percentual de issues fechadas?

- Projetos **mais maduros tendem a ter uma taxa de fechamento de issues mais alta**.
- O **percentual médio de fechamento é superior a 70%**, indicando boa manutenção na maioria dos casos.



3.7. RQ07 (Bonus): Sistemas escritos em linguagens mais populares recebem mais contribuição externa, lançam mais releases e são atualizados com mais frequência?

- **Python e TypeScript** têm uma tendência maior a receber contribuições externas.
- **TypeScript e JavaScript** apresentam maior volume de releases.
- **Rust e Go** são mais frequentemente atualizados, sugerindo maior dinamismo na manutenção.



- Discussão

Os resultados confirmam parcialmente as hipóteses iniciais:

- **Idade dos repositórios**: A maioria dos projetos populares possui mais de 5 anos, sugerindo que a maturidade contribui para o sucesso.
- **Contribuições externas**: Embora algumas tecnologias recebam mais PRs, o impacto é variável.
- **Releases**: A frequência de lançamento depende do ecossistema da linguagem.
- **Atualização constante**: Projetos populares são regularmente atualizados, o que sugere manutenção ativa.
- **Linguagens populares**: Python e TypeScript seguem dominando, mas Rust e Go se destacam na manutenção.
- **Fechamento de issues**: Projetos bem-mantidos têm alta taxa de fechamento de issues, validando a hipótese.