Relatório Final - Laboratório 01

Gabriel Alejandro Figueiro Galindo [1], **Marcelo Aguilar Araújo D'Almeida** [1], **Philippe Roberto Dutra Chaves Vieira**[1]

[1]Instituto de Ciências Exatas e Informática
Pontifícia Universidade de Minas Gerais (PUC Minas)
Belo Horizonte – MG – Brasil

1. Introdução e Hipóteses

Este estudo pretende analisar as principais características de sistemas open-source populares, incluindo sua maturidade, frequência de atualização, envolvimento da comunidade e outras particularidades relevantes. Os repositórios populares no GitHub representam uma amostra significativa do ecossistema de desenvolvimento open-source, sendo amplamente adotados. Por isso, eles serão utilizados como base para a coleta de dados. Com base nas questões de pesquisa propostas, as seguintes hipóteses informais foram elaboradas:

- RQ 01: Esperamos que a maioria dos repositórios populares sejam relativamente antigos, pois repositórios bem estabelecidos tiveram mais tempo para acumular estrelas e reconhecimento da comunidade.
- RQ 02: Repositórios populares devem receber muitas contribuições externas, refletindo o engajamento da comunidade open-source e sua relevância.
- RQ 03: Projetos populares provavelmente lançam releases com frequência, uma vez que precisam corrigir bugs, adicionar funcionalidades e manter compatibilidade com outras tecnologias.
- RQ 04: Esperamos que repositórios populares sejam atualizados com frequência, pois a manutenção ativa é um fator essencial para sua popularidade.
- RQ 05: A maioria dos repositórios populares devem estar escritos em linguagens amplamente utilizadas, como JavaScript, Python e Java, pois estas são frequentemente empregadas em projetos open-source de grande escala.
- RQ 06: Repositórios populares provavelmente possuem uma alta taxa de issues fechadas, indicando uma boa manutenção e resolução eficiente de problemas.
- RQ 07: Sistemas escritos em linguagens mais populares provavelmente recebem mais contribuição externa, lançam mais releases e são atualizados com mais frequência.

Após a coleta de dados, essas hipóteses serão analisadas e discutidas para verificar sua validade.

2. Metodologia

Para responder às questões da pesquisa, foi desenvolvido um código em Python que realiza a coleta de dados por meio da API GraphQL do GitHub. A busca é feita utilizando uma query que recupera repositórios com mais de 1.000 estrelas, processando os resultados em lotes de 20 repositórios por requisição. O código implementa paginação por meio dos parâmetros endCursor e hasNextPage, garantindo a obtenção de todos os repositórios relevantes. Como a API do GitHub possui um limite máximo de 1.000 itens retornados

por consulta, não há necessidade de definir um limite adicional no código. Os dados extraídos de cada repositório incluem:

- Nome do repositório;
- Nome do proprietário;
- Data de criação;
- Data da última atualização;
- Linguagem principal do repositório;
- Número total de releases;
- Quantidade de pull requests aceitas;
- Número total de issues fechadas e abertas;
- Quantidade de estrelas recebidas.

Após a coleta, os dados são processados para calcular métricas adicionais, como: idade do repositório (anos desde a criação), tempo desde a última atualização (dias) e percentual de issues fechadas em relação ao total. Depois do processamento, os dados são salvos em um arquivo CSV, onde eles são organizados em uma tabela com cabeçalhos apropriados e valores numéricos e temporais formatados, garantindo uma melhor legibilidade e compreensão dos dados.

O arquivo em CSV é então importado para um arquivo Excel, por onde (com o uso de Power Query) são realizadas manipulações, com estes dados, no intuito de gerar gráficos. É importante destacar que a execução do código ocorreu em 15/02/2025. Assim, os resultados obtidos refletem o estado dos repositórios nesse momento específico, podendo estar desatualizados ou não representar com precisão a realidade no momento da leitura.

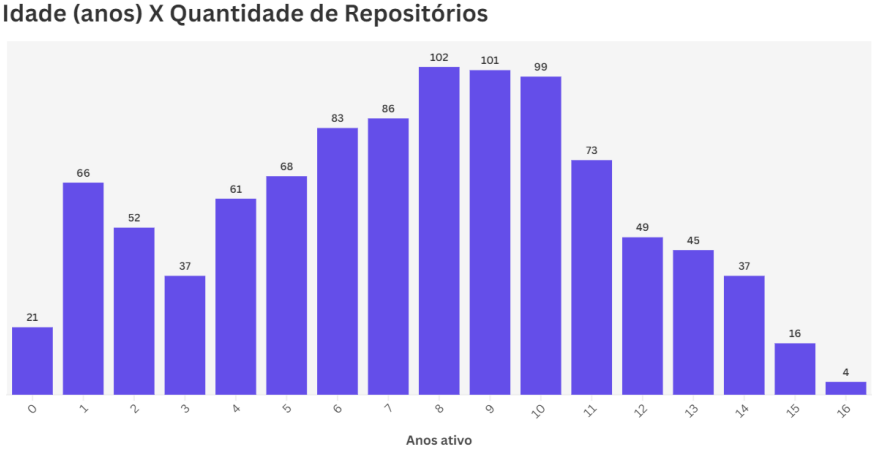
3. Resultados Obtidos

Com os dados analisados pelo Power Query do Excel, conseguimos visualizar de forma gráfica as métricas ressaltadas por cada umas das questões de pesquisa (as RQs da seção 1). Abaixo estão a interpretação dos resultados obtidos.

RQ01:

Em termo de idade de repositório, como previmos, podemos identificar que repositórios mais antigos são relativamente mais populares (com a grande maioria de repositórios tendo de 6 a 11 anos).

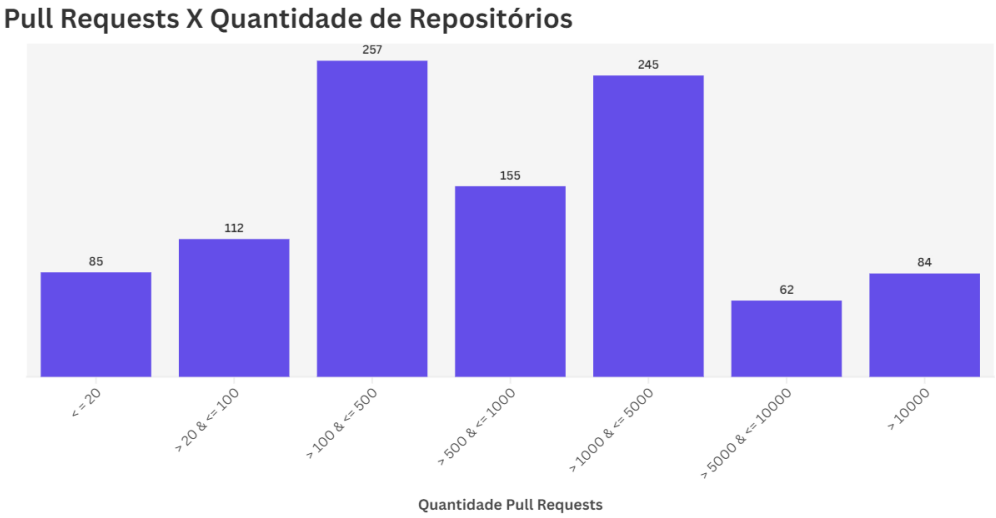
Porem, o que foi surpreendente foi o fato de 17,6% dos repositórios possuem idade igual ou inferior a 3 anos, representando que repositórios mais novos (que tratam de problemas mais atuais) também são reconhecidos e buscados pela comunidade.



RQ02:

Para os dados de quantidade de Pull Requests(participações externas), nossa previsão foi fora da realidade.

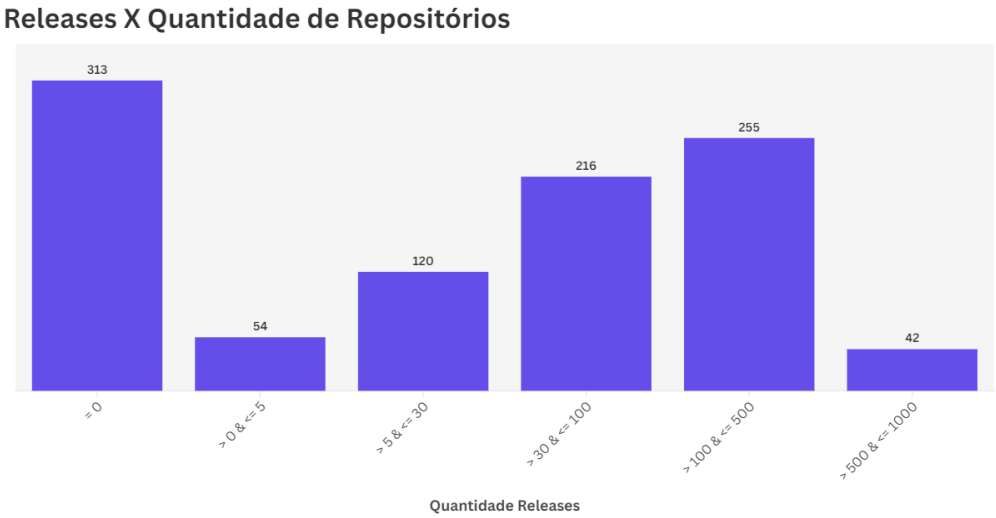
Foi constatado que 45,4% da nossa amostra, representa repositórios com até 500 Pull Requests; o que difere da hipótese que levantamos, onde dizemos que os repositórios mais populares recebem mais contribuições externas devido a um alto engajamento da sociedade.



RQ03:

É correto dizer que os repositórios mais populares tendem a lançar novas versões (Releases) com frequência, e isso fica evidente com o fato que 51,3% da nossa amostra lançaram de 31 a 1000 versões diferentes.

Porem, o oposto também foi evidenciado, quando 31,3% dos dados foram de repositórios com nenhuma nova release lançada (isto é, as modificações são feitas na versão original do projeto).

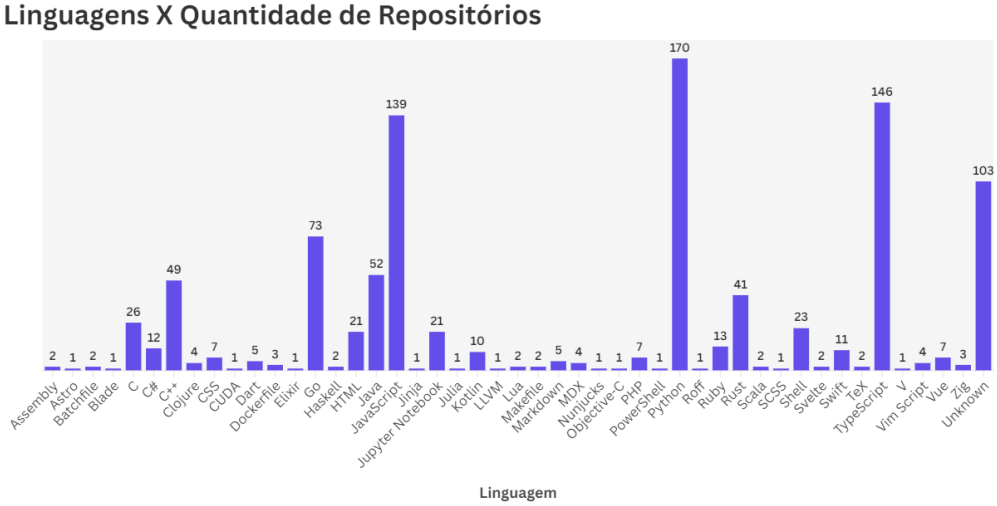


RQ04:

Para esta métrica, os dados obtidos condisseram perfeitamente com a hipótese levantada. Ficou definido que 98,8% dos repositórios possuem atualizações diárias, e que os demais estão a no máximo 2 dias sem receber modificações.

RQ05:

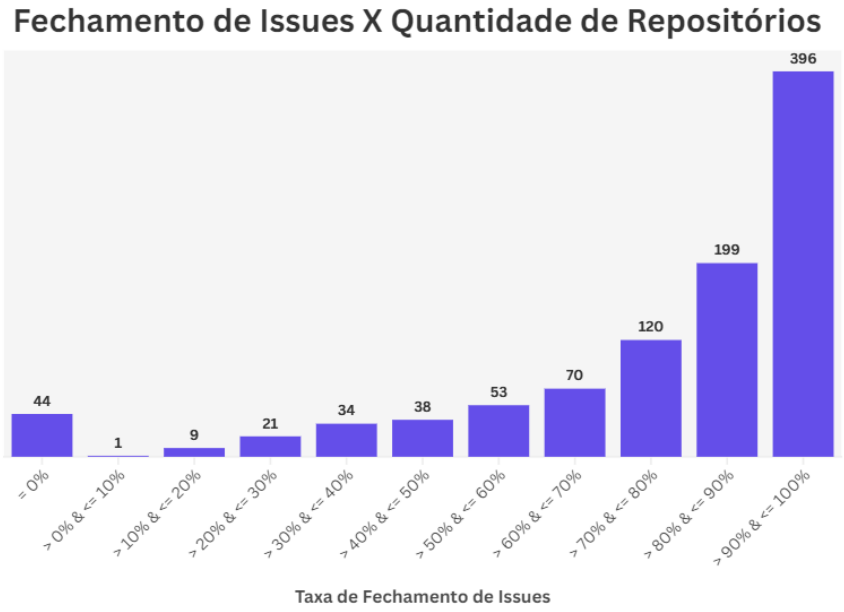
Assim como previmos, repositórios populares são escritos com linguagens populares. Ficou definido que as 5 linguagens com o maior número de repositórios são Python, TypeScript, JavaScript, GO e Java (nesta ordem); representando 58,8% da amostra.



RQ06:

No quesito de fechamento de issues, assim como previsto, os repositórios mais populares têm tendencia a fechar a maioria de suas issues; onde 71,5% da amostra possuem médias de fechamento de issues superiores a 70%

Um fato interessante foram os 44 repositórios que possuem 0% de fechamento de issues, mesmo muitos destes tendo issues em aberto.



RQ07:

Para a análise desta métrica foram filtrados os repositórios, agrupados por linguagem, que atendessem aos dados abaixo:

- Dias sem atualização = 0;
- Nº de releases superior ou igual à 100;
- Nº de pull requestes superior ou igual à 3000;

Com estes filtros ficou evidente o fato da linguagem ser popular não necessariamente significa que o repositório recebe muitas contribuições e é bem mantido (atualizações e correções frequentes). O que importa é quem mantém o repositório.

Isso pode ser verificado com as linguagens Python e JavaScript; onde, mesmo cada uma tendo mais de 140 repositórios na amostra, menos de 20 destes passaram pelos filtros definidos.

Por outro lado, os repositórios singulares de Julia, LLVM e V passaram pelo filtro.

4. Conclusão

Por este trabalho e sua análise, ficou evidente que é, sim, possível prever certos resultados e gerar hipóteses que se provarão verdadeiras após a análise; mesmo, em poucos casos, a realidade tendo sido diferente da hipótese.
Ficou definido que os repositórios mais populares são aqueles que, no geral, se mantêm frequentemente atualizados e com muitas contribuições, além de acompanharem as tendências do mercado (isto é, utilizam as linguagens de programação que estão "em alta").

References

Dados: https://api.github.com/graphql
Gráficos: https://flourish.studio