Análise de repositórios populares do Github

Alfredo Luis Vieira, Bruno Evangelista Gomes, Vinicius Salles

27 de agosto de 2025

1 Contextualização

Neste trabalho realizaremos a análise de repositórios populares do github, mais precisamente, os 1000 repositórios mais populares, esta análise visa identificar características destes repositórios, como número de issues fechadas, pull requests, releases e etc, e a partir destas métricas obtidas, levantar hipóteses acerca destes repositórios, identificar padrões, e analisar correlações entre os mesmos.

Este estudo é realizado a partir da realidade diversa dos repositórios situados no github, onde características como linguagem principal, contribuição, tamanho, total de estrelas e etc são muito variadas. O estudo se segue através de 6 perguntas, e métricas que podem respondê-las, sendo elas:

- Sistemas populares são maduros/antigos? (Métrica: idade do repositório, calculado a partir da data de sua criação).
- Sistemas populares recebem muita contribuição externa? (Métrica: total de pull requests aceitas).
- Sistemas populares lançam releases com frequência? (Métrica: total de releases).
- Sistemas populares são atualizados com frequência? (Métrica: tempo até a última atualização, calculado a partir da data de última atualização).
- Sistemas populares são escritos nas linguagens mais populares?(Métrica: linguagem primária de cada um desses repositórios).
- Sistemas populares possuem um alto percentual de issues fechadas?(Métrica: razão entre número de issues fechadas pelo total de issues).

2 Hipóteses

A partir do conhecimento existente antes da execução da análise, podemos inferir algumas hipóteses informais, que serão testadas, e reformuladas baseando-se na realidade identificada no estudo, sendo estas hipóteses informais são:

I) Sistemas mais populares (top 50 repositórios) possuem média de contribuição ativa por parte dos usuários acima da média dos outros repositórios analisados no estudo, esta hipótese se baseia na contribuição dos usuários com os repositórios, e que elas podem alavancar o desenvolvimento do projeto, basicamente, queremos descobrir se a participação ativa dos usuários, realmente alavanca o sistema.

II) As linguagens em alta no momento (senso stackOverflow) são refletidas nos repositórios mais populares do github, esta hipótese tenta relacionar as linguagens mais utilizadas pelos desenvolvedores, com os repositórios mais populares do github, ou seja, queremos entender se as tendências das linguagens se refletem em repositórios de projetos.

III) Repositórios com mais contribuição externa, possuem maior número de pull requests negadas, esta hipótese tem a ver com a questão do dilema quantidade x qualidade, partindo do pressuposto que nem sempre uma contribuição maior, tende a ser uma contribuição com mais qualidade,

IV) Repositórios com alto número de pull requests possuem maior número de issues, essa hipótese parte da ideia de quanto maior a contribuição, mais suscetível a erros o sistema está, devido ao alto nível de código inserido, e ao grande número de usuários atuando em conjunto, o que abre portas para possíveis falhas e bugs.

3 Metodologia

Neste estudo foi a aplicada a metodologia baseada na mineração de dados de repositórios do github, mais precisamente, os 1000 repositórios mais populares da plataforma, essa mineração foi realizada a partir de requisições junto a API GraphQL do próprio Github. Nessa API, buscamos as infomações relevantes para o estudo proposto. Em questões mais técnicas, utilizamos a Python como linguagem principal, utilizando bibliotecas da linguagem como o pandas. Vale citar tentativas de obtenção dos dados em massa junto a API, o que resultou em falha, pois a API foi implementada para retornar um erro após certo tempo de processamento de requisição, o que fez com que a consulta a mesma seja paginada, realizando um processamento em lote dos dados obtidos.

Para obter os resultados, foi realizado uma analise quantitativa dos dados, se baseando em dados estatisticos como média, mediana, razão entre valores e etc, estes resultados serão analisados mais pra frente.

4 Resultados obtidos

Nesta seção exibiremos os resultados obtidos, e uma breve discussão sobre eles, após a conclusão destas reflexões, uma reflexão geral será tratada na próxima seção.

4.1 Sistemas populares são maduros/antigos?

Nesta etapa da analise, observamos uma grande variedade de idades nos repositorios, abaixo, listaremos os repositórios mais jovens e mais velhos da análise:

4.1.1 Repositórios mais velhos

- rails/rails 208.4
- git/git 205.06
- jekyll/jekyll 202.12
- redis/redis 197.12
- jquery/jquery 196.70

4.1.2 Repositórios mais novos

- OpenCut-app/OpenCut 2
- google-gemini/gemini-cli 4.10
- openai/codex 4.27
- FoundationAgents/OpenManus 5.49
- x1xhlol/system-prompts-and-models-of-ai-tools 5.52

Nesta analise, podemos identificar uma média de 96.9 meses nestes repositórios, o que pode se considerar uma idade avançada, podemos assim, afirmar que sistemas populares são SIM maduros/antigos. Podemos tambem observar um desvio padrão de 46.8 meses, o que nos mostra que a idade dos repositorios é bem variada, apresentando valores que não estão muito concentrados em um intervalo.

4.2 Sistemas populares recebem muita contribuição externa?

Agora, devemos analisar se os sistemas analisados possuem um nivel alto de contribuição externa, ou seja, se os membros da comunidade realmente tendem a contribuir mais com sistemas populares, para fazer isso, calcularemos a média dos top 50 repositorios mais populares, e compararemos, coma média dos 50 repositórios observados com menos popularidade.

Podemos observar uma média de pull requests de 5371.90 nos 50 repositórios mais populares da lista, e uma média de 1741.23 pull requests nos 50 repositórios menos populares, ou seja, podemos dizer que a média aumentou em torno de 208%, ou seja, podemos identificar um grande aumento, e afirmar que sim, repositórios populares tendem a ter contribuições externas.

4.3 Sistemas populares lançam releases com frequência?

Para resolver-mos essa questão, tentaremos entender o total de releases de cada repositório, e o que esses dados em conjunto tem a dizer, inicialmente, devemos entender que 692 dos 1000 repositórios analisados possuiam releases, o que equivale a uma taxa de 69,2%, o que é uma taxa consideravelmente ok.

Para entendermos melhor a questão das releases, fizemos o seguinte calculo, primeito encontramos o intervalo médio de releases em dia de cada repositório, e depois encontramos a média geral deste intervalo, o que nos informa que em média, uma release é liberada a cada 108 dias, o que pode ser considerado uma frequencia média.

4.4 Sistemas populares são atualizados com frequência?

Com essa questão, queremos entender se os repositórios possuem atualizações recentes, o que pode nos dar indicios de que sua atualização é constante, após calcular uma média entre estes valores de cada respositório, encontramos uma média em torno de 3 a 5 dias, ou seja, em média.

4.5 Sistemas populares são escritos nas linguagens mais populares?

Neste estudo, encontramos repositórios com diversas linguagens principais, segue abaixo uma relação das linguagens encontradas e suas porcentagens:

- Python: 189 repositório(s) (18.9
- TypeScript: 156 repositório(s) (15.6
- JavaScript: 129 repositório(s) (12.9
- Go: 73 repositório(s) (7.3
- Java: 50 repositório(s) (5.0
- C++: 47 repositório(s) (4.7
- Rust: 45 repositório(s) (4.5
- C: 25 repositório(s) (2.5
- Jupyter Notebook: 22 repositório(s) (2.2
- Shell: 19 repositório(s) (1.9
- HTML: 19 repositório(s) (1.9
- Ruby: 12 repositório(s) (1.2
- C#: 12 repositório(s) (1.2
- Kotlin: 11 repositório(s) (1.1
- Swift: 9 repositório(s) (0.9
- CSS: 8 repositório(s) (0.8
- PHP: 7 repositório(s) (0.7
- Vue: 7 repositório(s) (0.7
- Markdown: 5 repositório(s) (0.5
- Dart: 5 repositório(s) (0.5
- MDX: 5 repositório(s) (0.5
- Clojure: 4 repositório(s) (0.4
- Vim Script: 4 repositório(s) (0.4
- Zig: 3 repositório(s) (0.3
- Dockerfile: 3 repositório(s) (0.3
- Assembly: 2 repositório(s) (0.2
- Batchfile: 2 repositório(s) (0.2
- TeX: 2 repositório(s) (0.2
- Lua: 2 repositório(s) (0.2
- Scala: 2 repositório(s) (0.2
- Makefile: 2 repositório(s) (0.2
- Roff: 2 repositório(s) (0.2
- Haskell: 2 repositório(s) (0.2
- Svelte: 2 repositório(s) (0.2
- Blade: 1 repositório(s) (0.1
- Nunjucks: 1 repositório(s) (0.1
- Julia: 1 repositório(s) (0.1
- PowerShell: 1 repositório(s) (0.1
- V: 1 repositório(s) (0.1
- LLVM: 1 repositório(s) (0.1
- Elixir: 1 repositório(s) (0.1
- Objective-C: 1 repositório(s) (0.1
- SCSS: 1 repositório(s) (0.1

Seguindo a pesquisa do stackoverflow do ano de 2025 (cujo os resultados estão inseridos no grafico abaixo), correlacionando com os dados encontrados nos repositórios, podemos identificar uma certa semelhança entre os lideres dos ranks em ambas as medições, como por exemplo, o Javascript, que

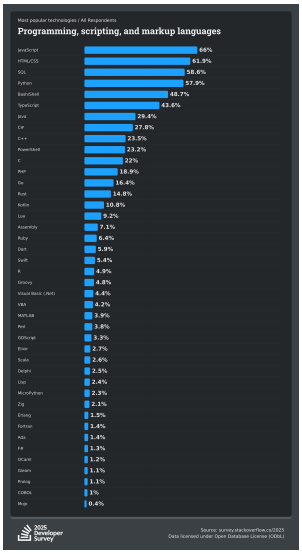

Figura 1: Resultados da pesquisa do stackoverflow de 2025

ficou em terceiro lugar nos repositórios que listamos, e que aparece em primeiro na pesquisa realizada em 2025, outro exemplo é o python, que está em primeiro nos levantamentos deste estudo, e em quarto na pesquisa realizada em 2025.

Podemos entender um certo padrão quando falamos de linguagens de programação utilizadas, principalmente quando falamos de tecnologias já consolidadas (como Javascript, python e Java), então podemos afirmar que existe sim uma tendencia de que as pesquisas relacionadas as linguagens favoritas, sigam caminhos semelhantes das linguagens principais dos repositórios mais populares,

4.6 Sistemas populares possuem um alto percentual de issues fechadas?

Nestes estudos, encontramos que os sistemas possuem uma média de 79% de suas issues fechadas, o que pode indicar que seus bugs, problemas relacionados a segurança ou qualquer outra coisa desta natureza, tendem a ser resolvidos, fato este que alinha muito com a pergunta 4.2, pois a maioria destas issues são resolvidas por membros externos, fazendo com que a atuação seja elevada.

5 Considerações finais

Este estudo se propôs a analisar as características dos 1000 repositórios mais populares do GitHub, com o objetivo de validar um conjunto de hipóteses e responder a questões sobre o que define um projeto de software popular. Ao final da análise quantitativa, é possível traçar um panorama claro que, em grande parte, confirma as expectativas iniciais, mas também revela nuances sobre o ciclo de vida e manutenção desses projetos.

A discussão a seguir confronta os resultados esperados, delineados em nossas hipóteses, com os valores efetivamente obtidos na mineração de dados.

A hipótese inicial (I), que sugeria que os repositórios mais populares teriam uma contribuição ativa superior à média, foi confirmada de forma contundente. Os resultados mostraram que os 50 repositórios no topo da lista possuem uma média de pull requests aceitas aproximadamente 208% maior que os 50 repositórios na base da lista. Este dado valida a premissa de que o engajamento da comunidade e a aceitação de contribuições externas são, de fato, um motor fundamental para a popularidade e o desenvolvimento de um projeto de grande escala.

Da mesma forma, a segunda hipótese (II), que previa uma correlação entre as linguagens em alta no mercado (segundo o senso do Stack Overflow) e as tecnologias predominantes nos repositórios populares, também foi corroborada. A análise revelou um domínio claro de linguagens como Python, TypeScript e JavaScript, que figuram consistentemente entre as mais utilizadas e amadas pela comunidade de desenvolvedores. Isso demonstra que a popularidade de um projeto está fortemente

ligada às tendências tecnológicas e às ferramentas preferidas pelos programadores, criando um ciclo de retroalimentação onde projetos populares reforçam o uso de linguagens populares.

As hipóteses III e IV, no entanto, exploravam relações mais complexas que merecem uma discussão aprofundada. A hipótese III levantava o dilema "quantidade vs. qualidade", sugerindo que um maior volume de contribuições externas poderia levar a um maior número de pull requests negadas. A hipótese IV, por sua vez, correlacionava o alto número de pull requests a uma maior incidência de issues. Embora os dados coletados não tenham se aprofundado na análise de PRs negadas ou na correlação direta entre PRs e a criação de issues, os resultados paralelos oferecem uma visão interessante. Com uma taxa média de 79% de issues fechadas, fica evidente que, independentemente do volume de novas falhas, há um esforço de manutenção muito eficaz nesses projetos. A alta taxa de contribuição externa (PRs) parece estar diretamente ligada à alta capacidade de resolução de problemas (issues fechadas), sugerindo que a comunidade não só contribui com funcionalidades, mas também é vital para a saúde e estabilidade do software.

Em suma, o estudo conclui que o perfil de um repositório popular no GitHub é o de um sistema maduro, com uma média de idade de quase 100 meses, mantido por uma comunidade ativa e engajada, e desenvolvido com tecnologias modernas e consolidadas. A expectativa de que a popularidade estaria atrelada ao engajamento comunitário e às tendências de linguagem foi plenamente atendida. Os dados sobre a frequência de releases e atualizações, ambos com média superior a 100 dias, indicam que a popularidade não exige uma atividade frenética, mas sim um ritmo de manutenção constante e confiável, capaz de gerir e resolver os problemas que surgem ao longo de seu ciclo de vida.