Lab01S03 - Relatório Final

Características de Repositórios Populares no GitHub

Alunos: Estêvão Rodrigues, Henrique Lobo
Curso: Engenharia de Software
Disciplina: Laboratório de Experimentação de Software
Professor: Danilo de Quadros Maia Filho
Turno: Noite
Universidade: PUC Minas

1. Introdução

Este estudo investiga as características dos 1.000 repositórios com maior número de estrelas no GitHub. O objetivo é entender padrões de desenvolvimento em sistemas open-source populares através de seis questões de pesquisa.

2. Questões de Pesquisa e Hipóteses Informais

RQ01: Sistemas populares são maduros/antigos?

Métrica: Idade do repositório (dias desde criação)

Hipótese Informal: Acreditamos que repositórios populares são relativamente antigos (3+ anos), pois precisam de tempo para ganhar reconhecimento da comunidade e acumular estrelas.

RQ02: Sistemas populares recebem muita contribuição externa?

Métrica: Total de pull requests aceitas

Hipótese Informal: Esperamos que projetos populares tenham muitas pull requests aceitas, indicando forte colaboração da comunidade e desenvolvimento aberto.

RQ03: Sistemas populares lançam releases com frequência?

Métrica: Total de releases

Hipótese Informal: Projetos populares devem ter várias releases, demonstrando ciclos de desenvolvimento organizados e entregas estruturadas para os usuários.

RQ04: Sistemas populares são atualizados com frequência?

Métrica: Tempo até a última atualização (dias)

Hipótese Informal: Repositórios populares devem ser atualizados recentemente (últimos 30-90 dias), mostrando manutenção ativa e desenvolvimento contínuo.

RQ05: Sistemas populares são escritos nas linguagens mais populares?

Métrica: Linguagem primária de cada repositório

Hipótese Informal: Esperamos que as linguagens mais comuns sejam JavaScript, Python, Java, TypeScript e C++, refletindo as tendências atuais de desenvolvimento.

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

Métrica: Razão entre issues fechadas e total de issues

Hipótese Informal: Projetos populares devem ter alta taxa de issues resolvidas (>70%), indicando boa manutenção e responsividade da equipe.

3. Metodologia

3.1 Coleta de Dados

- Fonte: GitHub REST API
- Amostra: 1.000 repositórios com maior número de estrelas
- Ferramenta: Script Python com bibliotecas padrão
- Critério: Busca por stars:>1 ordenada por estrelas (decrescente)

3.2 Paginação Implementada

- 10 páginas de 100 repositórios cada
- Rate limiting otimizado (0.1s entre requisições)
- Search API para contagem eficiente de PRs e issues

3.3 Dados Coletados

Para cada repositório foi coletado: nome, proprietário, URL, número de estrelas, Data de criação e última atualização, Linguagem primária, Total de PRs aceitas, releases, issues (abertas/fechadas), Número de forks e descrição

4. Resultados obtidos

4.1 Arquivos Gerados

- repositorios_1000.csv - Dados principais em formato CSV
- repositorios_1000.json - Backup dos dados em JSON
- main.py - Script de coleta otimizado
- Backups progressivos a cada 100 repositórios processados

4.2 Metricas e discussão das hipoteses

- **RQ01 – Idade dos repositórios:**
  A mediana da idade ficou em **~3049 dias (~8,3 anos)**. Isso sugere que repositórios populares tendem realmente a ser **antigos/maduros**, o que confirma parcialmente a hipótese de que sistemas populares precisam de tempo para consolidar sua reputação.

- **RQ02 – Pull requests aceitas:**
  A mediana encontrada foi **0**. Isso mostra que boa parte dos repositórios populares não tem um volume relevante de PRs aceitas. Pode ser porque muitos projetos não trabalham com o mesmo fluxo de contribuição de software tradicional. Isso coloca a hipótese em xeque talvez colaboração externa seja relevante apenas em **certos tipos** de projetos.

- **RQ03 – Releases publicadas:**
  A mediana foi **35 releases**. Isso indica que há uma parcela significativa de projetos com **entregas organizadas e estruturadas**, validando em parte a hipótese. Porém, também há muitos casos com zero releases (documentações, coleções de recursos), o que puxa a distribuição para baixo.

- **RQ04 – Atualizações recentes:**
  A mediana está em **6 dias desde a última atualização**, ou seja, mais da metade dos projetos foram atualizados muito recentemente. Isso é um indicador forte de **manutenção ativa**. A hipótese de que projetos populares são mantidos de forma constante se sustenta.

- **RQ05 – Linguagens de programação:**
  A análise das linguagens primárias mostra que a maioria dos repositórios populares é escrita em **TypeScript, Python, JavaScript, C++ e Go**, confirmando em parte a hipótese inicial.

- **RQ06 – Taxa de issues fechadas:**
  A mediana da razão foi **0**, indicando que pelo menos metade dos projetos não têm issues registradas (ou não utilizam o tracker de issues). Isso sugere que a hipótese de “alta taxa de issues fechadas” só vale para projetos que efetivamente **usam** o sistema de issues. Entre esses, há exemplos com taxas muito altas (>90%), mas o dado geral não confirma a hipótese.

5. Conclusão

A análise dos 1.000 repositórios mais populares do GitHub mostra que a popularidade em software open-source não depende exclusivamente de práticas clássicas de engenharia de software, mas revela padrões interessantes:

Maturidade é um fator relevante: os repositórios analisados têm, em média, mais de oito anos, o que confirma a ideia de que o reconhecimento da comunidade demanda tempo.

Contribuição externa não é universal. Muitos projetos populares não possuem pull requests ativas, o que indica que o sucesso de um repositório pode estar mais ligado ao seu conteúdo ou utilidade (listas, catálogos, frameworks) do que a um modelo de colaboração intensa.

Organização e entregas aparecem em parte dos projetos, com mediana de 35 releases, mas não são regra entre todos.

Manutenção ativa é fortemente confirmada, com a maioria dos repositórios atualizados recentemente, mostrando vitalidade e interesse contínuo da comunidade ou dos mantenedores.

Linguagens predominantes refletem de fato as tendências do mercado (TypeScript, Python, JavaScript, C++ e Go), validando a hipótese inicial.

Gestão de issues se mostra um ponto heterogêneo: enquanto alguns projetos possuem taxas de fechamento altíssimas, muitos sequer utilizam o sistema de issues, sugerindo diferentes estratégias de gestão e suporte.

Em resumo, os resultados apontam que projetos populares tendem a ser antigos, ativos e desenvolvidos em linguagens consolidadas, mas nem sempre seguem as práticas esperadas de colaboração, versionamento e acompanhamento de issues. Isso evidencia a diversidade de modelos de sucesso no GitHub, onde tanto frameworks complexos quanto simples listas de recursos podem atingir grande relevância global.