Principais características de projetos open-source no Github

Matheus Nolasco 11, Maria Aryene Costa 21, Nome aluno 31, Nome aluno 41, Nome aluno 51

1 Instituto de Ciências Exatas e Informática

Pontifícia Universidade de Minas Gerais (PUC Minas)  
Belo Horizonte – MG – Brasil

{Matheus Nolasco Miranda Soares, Maria Aryene Costa dos Santos, aluno3, aluno4, aluno5}@sga.pucminas.br

**Resumo.** Escrever aqui o resumo. O resumo deve contextualizar rapidamente o trabalho, descrever seu objetivo e, ao final, mostrar algum resultado relevante do trabalho (até 10 linhas).

# 1. Introdução

Neste projeto pretende-se estudar e analisar um grupo de 1.000 repositórios populares e de *open source* no Github com o intuito de analisar como eles são desenvolvidos, com que frequência recebem contribuições externas, entre outras características. Esse projeto tem como objetivo responder as seguintes questões:

1. “Sistemas populares são maduros/antigos?”
2. “Sistemas populares recebem muita contribuição externa?”
3. “Sistemas populares lançam releases com frequência?
4. “Sistemas populares são atualizados com frequência?”
5. “Sistemas populares são escritos nas linguagens mais populares?”
6. “Sistemas populares possuem um alto percentual de issues fechadas?”

A partir de cada uma dessas perguntas, uma hipótese inicial foi formulada com o resultado esperado. Essas “Hipóteses Informais” são:

* Sistemas populares não são necessariamente maduros tendo em vista que a popularização de um sistema não está correlacionada a sua idade, mas sim, a outros fatores.
* Sistemas populares recebem muita contribuição externa pois, por serem populares, possuem muita visibilidade e utilização.
* Sistemas populares lançam release com frequência, pois, como possuem muitas contribuições, estão constantemente sendo atualizados.
* Sistemas populares não são necessariamente escritos nas linguagens mais populares, visto que, o que mais impacta na sua popularidade não é a tecnologia de sua utilização.
* Sistema populares não necessariamente possuem alto percentual de issues fechadas pois, por atraírem muitos contribuidores, também atraem desenvolvedores que começam alguma issue mas não finalizam ou demoram muito tempo para finalizá-las.

Para responder as perguntas da pesquisa e confirmar ou não as hipóteses, os dados coletados do github serão analisados em um arquivo CSV.

# 2. Metodologia

O trabalho apresentado foi conduzido utilizando a abordagem empírica quantitativa. De acordo com (Gomez & Reidl, 2010), a abordagem empírica engloba observação, medição e experimentação. Em complemento, a pesquisa quantitativa, para Mattar (2001), busca a validação das hipóteses mediante a utilização de dados estruturados, estatísticos, com análise de muitos casos representativos.

A coleta foi realizada Visual Studio Code consultando a API do GitHub via GraphQL. Segundo (Samer, 2012), a linguagem de consulta GraphQL simplifica as interações com servidores web, possibilitando consultas de API mais inteligentes que podem melhorar enormemente a eficiência das solicitações de dados.

A implementação do experimento foi dividida em três sprints. Na primeira sprint, foram coletados via GraphQL os 100 repositórios com maior número de estrelas no GitHub. O trabalho desenvolvido utilizou como roteiro as seguintes perguntas e os parâmetros do GitHub descritos abaixo:

RQ 01. Sistemas populares são maduros/antigos?

Métrica: idade do repositório [createdAt]

RQ 02. Sistemas populares recebem muita contribuição externa?

Métrica: total de pull requests aceitas [pullRequests(states: MERGED) { totalCount }]

RQ 03. Sistemas populares lançam releases com frequência?

Métrica: total de releases [releases][totalCount]

RQ 04. Sistemas populares são atualizados com frequência?

Métrica: tempo até a última atualização [name updatedAt]

RQ 05. Sistemas populares são escritos nas linguagens mais populares?

Métrica: linguagem primária de cada um desses repositórios [primaryLanguage] [name]

RQ 06. Sistemas populares possuem um alto percentual de issues fechadas?

Métrica: razão entre número de issues fechadas pelo total de issues

[issues(states: [CLOSED]) [totalCount]]

Na segunda sprint, a consulta foi realizada com 1.000 repositórios, sendo desenvolvidas nessa fase, as hipóteses informais supracitadas na introdução. Ainda na segunda fase, foi realizada a exportação dos dados dos 1.000 repositórios em .csv para análise posterior.

Na terceira e última sprint, foi desenvolvida a documentação do experimento, contendo metodologia, resultados, discussão e trabalhos futuros.

# 3. Resultados

Resultados do trabalho devem ser apresentados. Consiste da descrição técnica da solução desenvolvida. Use figuras e tabelas sempre que necessário. Todas as etapas descritas na metodologia devem ter seus resultados apresentados aqui. Detalhar os resultados obtidos para cada uma das perguntas. Ilustre, os resultados utilizando visualizações de dados que ajudem a demonstrar.

# 3. Discussão

Discuta os resultados obtidos e compare com as hipóteses que foram definidos no início do trabalho.

# 4. Conclusões e trabalhos futuros

A conclusão deve iniciar resgatando o objetivo do trabalho e os principais resultados alcançados. Em seguida, devem ser apresentados os trabalhos futuros.

Acrescentar aqui a tabulação da estatística de avaliação da aplicação (questionário de avaliação final da ferramenta).

# Referências

C. E. Anchundia and E. R. Fonseca C., "Resources for Reproducibility of Experiments in Empirical Software Engineering: Topics Derived From a Secondary Study," in IEEE Access, vol. 8, pp. 8992-9004, 2020, doi: 10.1109/ACCESS.2020.2964587.

MATTAR, F. N. Pesquisa de marketing. 3.ed. São Paulo: Atlas, 2001.

Samer Buna, GraphQL in Action , Manning, 2021.