Relatório Final - Laboratório 01

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

1Instituto de Ciências Exatas e Informática Pontifícia Universidade de Minas Gerais (PUC Minas)

Belo Horizonte – MG – Brasil

1. **Introdução e Hipóteses**

Este estudo tem como objetivo 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, re- fletindo 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 compatibili- dade 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 frequente- mente 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.

1. **Metodologia**

Para responder às questões da pesquisa, foi desenvolvido um código em Python que re- aliza 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 resulta- dos 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 com- preensão dos dados.

E´ importante destacar que a execução do código ocorreu em 15/02/2025. Assim, os resul-

tados obtidos refletem o estado dos repositórios nesse momento específico, podendo estar desatualizados ou não representar com precisão a realidade no momento da leitura.

1. **Resultados Obtidos**

1. **Conclusão**

**Referencias**