Laboratório 01 – Grupo 04

Eric Guimarães Caldas Jardim, Lucas Picinin Campos Lutti, Matheus Fontes Almeida Moreira Silva, Nando Augusto Veloso Tupinambá

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

{1349194, 1254051, 1322257, 1337014}@sga.pucminas.br

Resumo. *O trabalho consiste em extrair dados dos repositórios mais famosos do GitHub com o intuito de analisar informações importantes sobre o comportamento dos mesmos. Nesse sentido, como o projeto é parte da disciplina de Medição e Experimentação de Software (parte laboratorial), a ideia como desenvolvimento deste trabalho é aperfeiçoar o conhecimento de todos os componentes do grupo acerca de requisições no formato GraphQL, uma vez que, mesmo sendo um aspecto muito importante no ambiente tecnológico, a maioria dos presentes neste trabalho não têm contato direto com esse tipo de busca. Deste modo, a análise qualitativa dos dados extraídos da mineração dos requisitos permitiu que fossem mapeadas as principais dores e necessidades dos desenvolvedores nos dias atuais.*

1. Introdução

O projeto desenvolvido tem como premissa a descoberta de características em comum que interligam os repositórios mais populares do GitHUB. Nesse sentido, é fato que essa plataforma é a principal no mundo dos desenvolvedores e, por este fator, conhecer os temas dos itens mais buscados dentro da mesma permite que se tenha uma ideia maior de quais são as principais tendências no mundo da tecnologia. Sob esse viés, a coleta dos principais projetos no GitHUB irá se basear pelo número total de estrelas. Contudo, diversos itens serão analisados em cada repositório, como: data de criação, número de release, número de pull requests, linguagem principal, porcentagem de issues fechadas e data da última atualização. Todos esses parâmetros serão utilizados para que as perguntas definidas no GQM sejam respondidas com a maior precisão possível, trazendo à realidade os dados coletados e fazendo com que eles sejam úteis na análise de situações esporádicas e cotidianas.

Outrossim, mesmo que ainda na parte embrionária do projeto, torna-se necessário o levantamento de hipóteses sobre quais serão os resultados da análise dos dados coletados. Na visão do grupo, os principais repositórios encontrados serão, em sua maioria, temáticos de linguagens de programação mais utilizadas no desenvolvimentos de quaisquer sistemas como um todo. Ademais, pensa-se que a atualização rotineira de projetos faz com que cada vez mais usuários se interessem pelo mesmo e possam interagir com o repositório discutido.

Por fim, é notório a importância de uma análise minuciosa dos dados, que será feita utilizando a linguagem Python, com o intuito de criar dashboards que facilitem a visualização dos indícios coletados, evidenciando os diferentes parâmetros de pesquisa que foram utilizados para responder as perguntas norteadoras do projeto.

2. Metodologia (neste tópico deve ficar claro COMO foi realizado o seu trabalho)

As pesquisas quantitativas são aquelas onde são extraídos dados que podem ser manipulados e servem para análise dos indicativos matemáticos de alguma situação. De acordo com o livro Métodos de Pesquisa em Ciências Sociais, dos autores Chava Frankfort-Nachmias e David Nachmias, essa abordagem permite uma compreensão mais precisa e mensurável dos fenômenos estudados, fornecendo uma base sólida para inferências e generalizações sobre a população-alvo. Sob esse viés, o projeto desenvolvido, utilizando a metodologia detalhada abaixo, serviu para conhecer o padrão dos repositórios mais famosos do GitHUB, permitindo que, através dos dados coletados pelas métricas definidas, fosse possível entender o padrão dos desenvolvedores do mundo todo nos dias atuais.

O trabalho foi desenvolvido utilizando a metodologia SCRUM e, deste modo, todo o projeto foi confeccionado em três sprints, cada uma com a duração de uma semana. A sprint 1 foi onde foram levantadas as respostas e métricas para cada uma das perguntas do GQM. Nesse sentido, foram minerados 100 repositórios e suas informações respectivas que satisfizessem as métricas levantadas, além de confeccionar um script, utilizando a linguagem Python, que salvasse os dados de todos os projetos mapeados em um arquivo json específico. Já na sprint 2, o foco principal foi a confecção da paginação no código Python, permitindo que mais de 100 repositórios fossem minerados (nesse caso, houveram 1000 projetos que tiveram seus dados extraídos). Outrossim, o começo da criação do relatório também foi realizado nessa etapa do projeto. Por fim, a sprint 3 foi onde o grupo finalizou o relatório através da análise dos resultados obtidos através do código confeccionado, no qual o mesmo foi desenvolvido com as seguintes características:

- **Passo 1 -** Ler o arquivo json e obter os dados brutos iniciais
- **Passo 2 -** Converter os dados brutos em dados possíveis de serem analisados e manipulados. Os dados foram organizados em listas com a utilização da biblioteca pandas.
- **Passo 3 -** Análise dos dados. Os dados extraídos foram analisados seguindo as métricas levantadas inicialmente no projeto.
- **Passo 3.1 - Sistemas populares são maduros/antigos?**

  Métrica utilizada - data de criação dos repositórios

- **Passo 3.2 - Sistemas populares recebem muita contribuição externa?**

  Métrica utilizada - média de pull requests por repositório

- **Passo 3.3 - Sistemas populares são atualizados com frequência?**

  Métrica utilizada - data da última atualização dos repositórios

- **Passo 3.4 - Sistemas populares são escritos nas linguagens mais populares?**

  Métrica utilizada - Principal linguagem do repositório

- **Passo 3.5 - Sistemas populares possuem um alto percentual de issues fechadas?**

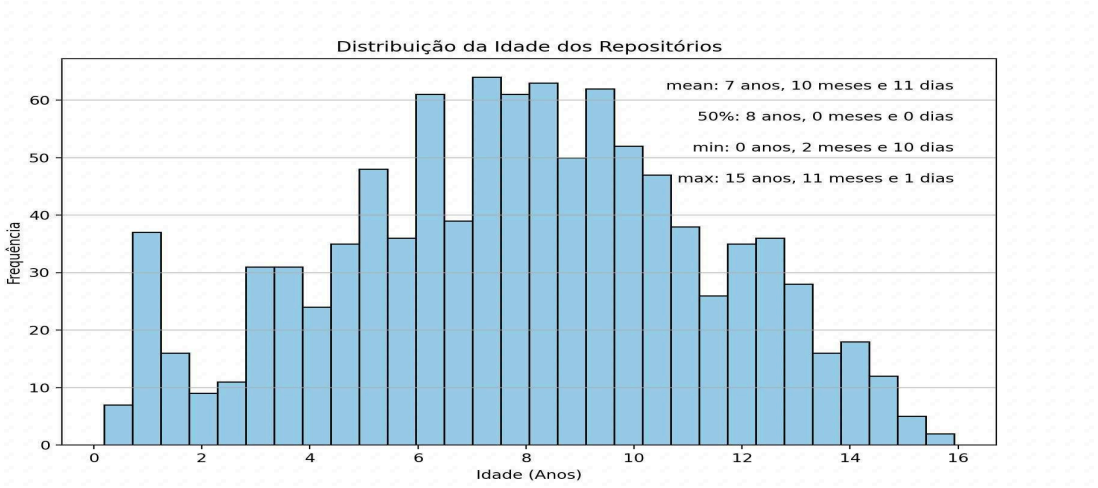
  Métrica utilizada - Média da razão entre issues abertas e issues concluídas

- **Passo 4 -** Visualização dos dados. A mesma é feita através da utilização da biblioteca Matplotlib, onde a mesma cria um gráfica de dispersão a cada métrica que for definida, permitindo uma análise mais fácil e certeira dos dados extraídos.

3. Resultados

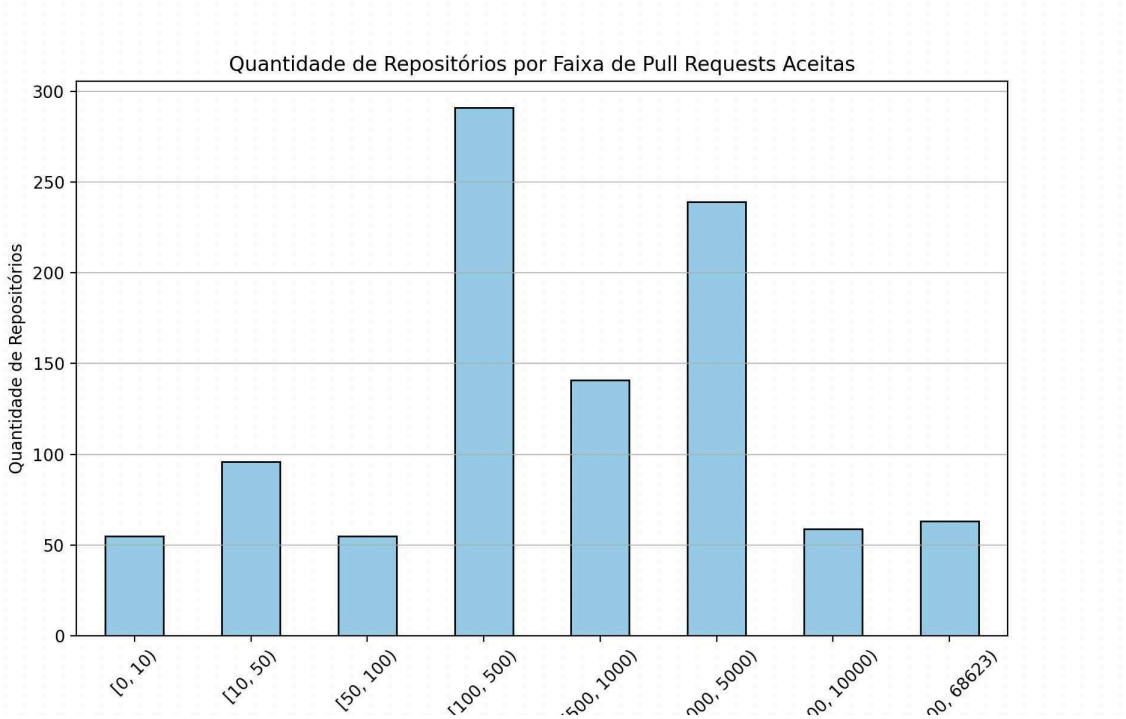
Q1

Análise descritiva das idades dos repositórios:
count (Número de entradas não nulas.): 1000.0
mean (Média aritmética dos dados.): 7 anos, 10 meses e 11 dias
std (Desvio padrão, indicando a dispersão dos dados.):
1254.245182091923 min (Menor valor encontrado.): 0 anos, 2 meses
e 10 dias
25% (Abaixo deste valor estão 25% dos dados (primeiro quartil).): 5 anos, 5 meses e
21 dias 50% (Mediana, o valor do meio dos dados.): 8 anos, 0 meses e 0 dias
75% (Abaixo deste valor estão 75% dos dados (terceiro quartil).): 10 anos, 3 meses e
12 dias max (Maior valor encontrado.): 15 anos, 11 meses e 1 dia



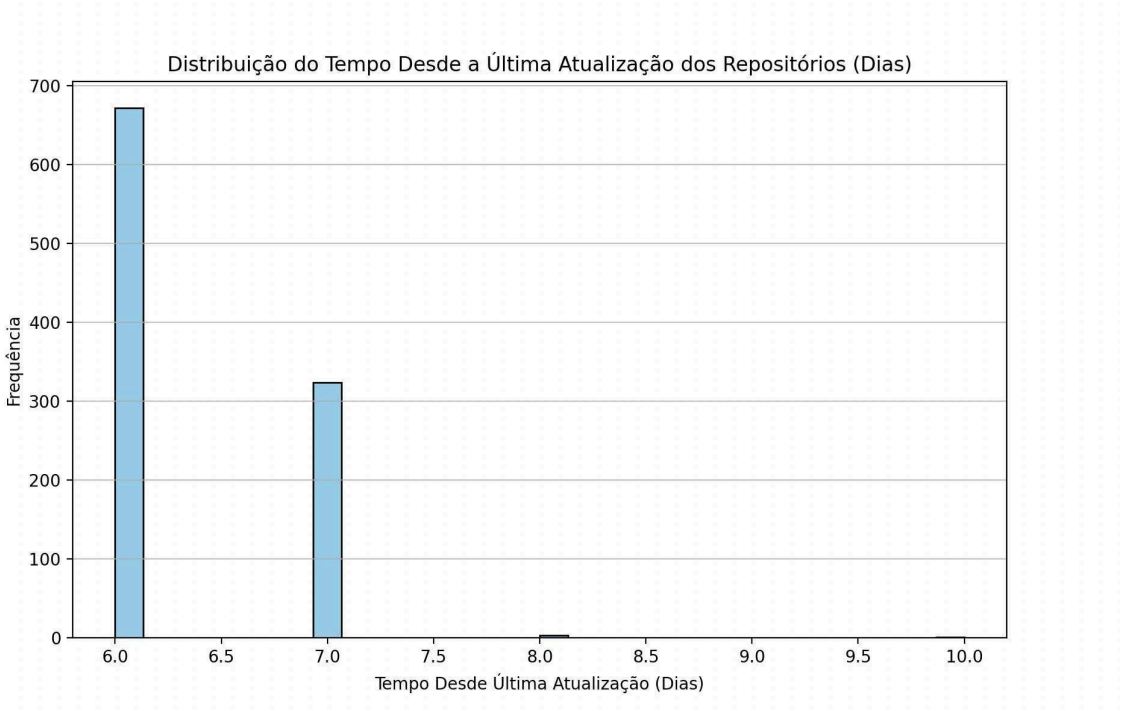
Q2

Análise descritiva do número de PRs aceitas nos repositórios:
count (Número de entradas não nulas.): 1000.0
mean (Média aritmética dos dados.): 2556.571
std (Desvio padrão, indicando a dispersão dos dados.): 6378.828271797264
min (Menor valor encontrado.): 0.0
25% (Valor abaixo do qual estão 25% dos dados (primeiro quartil).): 141.0
50% (Mediana, o valor do meio dos dados.): 523.5
75% (Valor abaixo do qual estão 75% dos dados (terceiro quartil).): 2064.0
max (Maior valor encontrado.): 68623.0



Q4

Análise descritiva do tempo desde a última atualização dos repositórios: count (Número de entradas não nulas.): 1000.0
mean (Média aritmética dos dados.): 0 anos, 0 meses e 6 dias
std (Desvio padrão, indicando a dispersão dos dados.): 0.4905962542505647 min (Menor valor encontrado.): 0 anos, 0 meses e 6 dias
25% (Abaixo deste valor estão 25% dos dados (primeiro quartil).): 0 anos, 0 meses e 6 dias
50% (Mediana, o valor do meio dos dados.): 0 anos, 0 meses e 6 dias
75% (Abaixo deste valor estão 75% dos dados (terceiro quartil).): 0 anos, 0 meses e 7 dias
max (Maior valor encontrado.): 0 anos, 0 meses e 10 dias



Q5

JavaScript	178
Python	135
TypeScript	13
Go	76
Java	66
C++	53
Rust	38
C	26
Shell	24
Jupyter Notebook	
HTML	16
Ruby	14
C#	13
PHP	10
Swift	10
Vue	9
CSS	8
Kotlin	7
Dart	5
MDX	5
Lua	4
Markdown	4
Less	3
Batchfile	3
Vim Script	3
Clojure	3
TeX	2
Scala	2
Zig	2
Dockerfile	2
Emacs Lisp	1
SCSS	1
Objective-C	1
Svelte	1
Assembly	1
Haskell	1
Nunjucks	1
Astro	1
Julia	1
Makefile	1
Elixir	1

Linguagens primárias mais populares nos repositórios:

4. Discussão das hipóteses

Idade dos Repositórios e Volume de Atividade:

Hipótese: Poderia-se esperar que repositórios mais antigos tenham um volume de atividade mais alto, medido pelo número de pull requests (PRs) aceitas.

Discussão: Embora a idade média dos repositórios seja de quase 8 anos, não há uma correlação direta entre a idade e o volume de PRs aceitas. Isso sugere que a idade por si só não é um preditor confiável da atividade do repositório. Outros fatores, como a relevância do projeto, a comunidade de desenvolvedores ativa e a manutenção do código, também podem influenciar a atividade do repositório.

Diversidade de Linguagens e Popularidade:

Hipótese: Linguagens de programação mais populares têm uma presença mais forte nos repositórios, refletindo uma preferência da comunidade de desenvolvedores.

Discussão: Os resultados mostram que JavaScript, Python e TypeScript são as linguagens mais populares nos repositórios analisados. Isso pode indicar a prevalência dessas linguagens na comunidade de desenvolvimento de software de código aberto, possivelmente devido à sua versatilidade, popularidade e recursos disponíveis. A presença de linguagens menos comuns também sugere uma diversidade de interesses e necessidades entre os desenvolvedores.

Atualização e Manutenção:

Hipótese: A frequência de atualização dos repositórios está correlacionada com sua relevância e atividade.

Discussão: A média de apenas 6 dias desde a última atualização indica um alto nível de atividade e manutenção nos repositórios analisados. Isso sugere que a comunidade de desenvolvedores está ativamente engajada na melhoria e atualização contínua do código. A rápida resposta a problemas e atualizações pode indicar um alto nível de comprometimento com a qualidade e a evolução do software.

Eficiência na Resolução de Issues:

Hipótese: Repositórios com altas taxas de issues fechadas indicam uma eficiência na gestão de problemas e na colaboração da comunidade.

Discussão: Os repositórios com os maiores percentuais de issues fechadas, como legacy-homebrew e pkg, demonstram uma capacidade eficaz de lidar com problemas e integrar contribuições da comunidade. Isso sugere uma cultura de colaboração e um processo de desenvolvimento robusto, que pode servir como exemplo para outros projetos.

5. Conclusões e trabalhos futuros

Com base nos resultados das métricas apresentadas, podemos concluir que a média de idade dos repositórios é de aproximadamente 7 anos, 10 meses e 11 dias, com uma ampla dispersão dos dados. A maioria dos repositórios tem menos de 10 anos de idade. Quanto ao número de pull requests (PRs) aceitas, a média é de aproximadamente

2556.57, mas há uma grande dispersão nos dados. A distribuição mostra que 75% dos repositórios têm menos de 2064 PRs aceitas.

Quanto ao tempo desde a última atualização dos repositórios, a média é de 6 dias, indicando que a maioria foi atualizada recentemente, com uma dispersão mínima. As linguagens primárias mais populares nos repositórios analisados são JavaScript, Python e TypeScript, seguidas por Go e Java. As menos populares incluem Svelte, Assembly e Nunjucks. Alguns repositórios, como legacy-homebrew, pkg e setup-ipsec-vpn, têm os maiores percentuais de issues fechadas, com uma razão de 100% em alguns casos.

Para trabalhos futuros, sugerimos explorar métricas de qualidade de código, analisar tendências temporais, investigar correlações entre as métricas mencionadas, realizar análises de performance e identificar boas práticas de desenvolvimento observadas nos repositórios com altas taxas de issues fechadas. Esses projetos futuros podem ser adaptados e expandidos conforme os objetivos específicos da pesquisa.

Referências

Boulic, R. and Renault, O. (1991) "3D Hierarchies for Animation", In: New Trends in Animation and Visualization, Edited by Nadia Magnenat-Thalmann and Daniel Thalmann, John Wiley & Sons ltd., England.

Dyer, S., Martin, J. and Zulauf, J. (1995) "Motion Capture White Paper", http://reality.sgi.com/employees/jam_sb/mocap/MoCapWP_v2.0.html, December.

Holton, M. and Alexander, S. (1995) "Soft Cellular Modeling: A Technique for the Simulation of Non-rigid Materials", Computer Graphics: Developments in Virtual Environments, R. A. Earnshaw and J. A. Vince, England, Academic Press Ltd., p. 449-460.

Knuth, D. E. (1984), The TeXbook, Addison Wesley, 15th edition.

Smith, A. and Jones, B. (1999). On the complexity of computing. In *Advances in Computer Science*, pages 555–566. Publishing Press.

Frankfort-Nachmias, C., & Nachmias, D. (1992). Métodos de Pesquisa em Ciências Sociais. Atlas.