Entendendo o engajamento das comunidades front-end e back-end nos repositórios do GitHub

Altino Alves Júnior
Depto. de Engenharia de Software
PUC Minas
Belo Horizonte, Brasil
aajunior@sga.pucminas.br

Letícia de Souza Meireles
Depto. de Engenharia de Software
PUC Minas
Belo Horizonte, Brasil
leticia.meireles.124503@sga.pucminas.br

Lucas Alves Rossi Figueira
Depto. de Engenharia de Software
PUC Minas
Belo Horizonte, Brasil
larfigueira@sga.pucminas.br

Vítor Marcondes Morais Carmo
Depto. de Engenharia de Software
PUC Minas
Belo Horizonte, Brasil
vitor.carmo.1201734@sga.pucminas.br

Humberto T. Marques-Neto
Depto. de Engenharia de Software
PUC Minas
Belo Horizonte, Brasil
humberto@pucminas.br

Laerte Xavier
Depto. de Engenharia de Software
PUC Minas
Belo Horizonte, Brasil
laertexavier@pucminas.br

RESUMO

Frequentemente, a comunidade de desenvolvedores realiza comparações entre as linguagens de programação das *stacks* de *front-end* e *back-end*, o que levanta questões sobre manutenibilidade do código, engajamento da comunidade na resolução de *issues* e popularidade das linguagens. Neste contexto, propõe-se a investigar sobre o engajamento dos usuários nos repositórios mais populares do GitHub, das linguagens de programação mais populares nos últimos dois anos nessas *stacks*. A partir de métricas quantitativas, buscou-se identificar com quais linguagens e *stacks* os desenvolvedores mais interagem, utilizando técnicas de mineração nos quinhentos repositórios públicos mais populares das linguagens analisadas e que contenham os tópicos de *front-end* e *back-end*. Os resultados apontam que há mais interação com a *stack front-end* e uma preferência ao desenvolvimento com a linguagem JavaScript em ambas *stacks*.

KEYWORDS

GitHub, back-end, front-end, engajamento, mineiração de repositórios

1 INTRODUÇÃO

O *software* é um produto do trabalho humano cada vez mais presente na sociedade [3]. Nos últimos anos diversas ferramentas, métodos e técnicas têm sido desenvolvidas para serem utilizadas ao se aplicar Engenharia de *Software* na construção de seus artefatos [1]. Dentre esses métodos, ressalta-se o versionamento de código e *software*, ou seja, critérios e procedimentos necessários para o gerenciamento e controle das versões e mudanças de um código-fonte, visando maior segurança, armazenamento e histórico em relação a transição entre essas versões.

O uso de repositórios contribui com o versionamento de código, bem como uma forma de promover melhor integração e comunicação entre desenvolvedores e outros profissionais da área – por meio de revisões de código, por exemplo – contemplando também fluxos propostos pela Engenharia de *Software*, tais como de desenvolvimento, validação de requisitos, testes, etc. Além disso, sabe-se que um repositório pode estar alocado localmente em um computador ou em nuvem por meio de plataformas ou redes sociais específicas para código. Uma rede possível é a de desenvolvedores de código como o GitHub, onde participantes são desenvolvedores que podem criar, contribuir, compartilhar e buscar por repositórios de projetos de acordo com assuntos e Linguagens de Programação (LPs) em que foram desenvolvidos [9].

Outro fator associado às boas práticas da Engenharia de *Software* e projetos no GitHub são as *stacks*. Entende-se *stack* como um conjunto de tecnologias utilizadas para o desenvolvimento de software, que engloba linguagens de programação; *frameworks*; banco de dados e arquitetura das aplicações. Atualmente, destacam-se duas principais *stacks*: *back-end* e *front-end* – além do *mobile* que, muitas vezes, é incluída na *stack front-end*.

Compreende-se o *back-end* como o lado servidor da aplicação. Ele é responsável pelo armazenamento, gerenciamento de dados, bem como regras de negócios, APIs. Esta é uma camada que não interage diretamente com o usuário, visto que os dados são consumidos por meio do *front-end* de um aplicação. Em contrapartida, o *front-end* é a camada que o usuário visualiza e interage em uma aplicação, entregando uma experiência integrada ao *back-end*, consumindo-o por meio de requisições e envios de dados.

Assim, diante dessa segmentação de tecnologias, é comum observar discussões e *sites* que buscam esclarecer a diferença entre as *stacks* [10], principalmente para desenvolvedores iniciantes que estão ingressando na área. Até mesmo as organizações já alocam seus novos profissionais em equipes específicas a depender da *stack* [7]. Especializar-se em uma determinada área e conhecer o perfil da comunidade pode contribuir no momento de escolher em qual *stack* focar.

Dentro desse contexto, este trabalho pretende investigar o engajamento da comunidade de desenvolvimento no GitHub nas *stacks back-end* e *front-end*, buscando comparar, por meio das contribuições e popularidade dos repositórios, o comportamento e a participação dos desenvolvedores em cada um desses grupos. Entende-se como engajamento a característica de gerar novas ideias e informações num determinado contexto, evidenciando discussões entre equipes [5]. Neste trabalho, isso foi analisado, por exemplo, a partir da quantidade de *issues* e *pull requests* (PRs), bem como a participação é verificada pelos números de *reviews*, participantes e comentários dessas mesmas PRs.

Criou-se, portanto, um *dataset* composto por dados de 3.000 repositórios desenvolvidos nas quatro principais linguagens de programação do GitHub (Java, JavaScript, Python e TypeScript) [4], assim como *pull requests* e *issues* associadas a esses repositórios. Com isso, pretende-se responder às seguintes questões de pesquisa: ***RQ1.*** "Qual é a linguagem mais popular de *front-end* e qual a mais popular de *back-end*", ***RQ2.*** "Qual *stack* possui maior contribuição da sua respectiva comunidade?", ***RQ3.*** "Em qual *stack* há maior engajamento dos desenvolvedores?".

O presente artigo foi dividido da seguinte maneira: a seção 2 apresenta os trabalhos mais relacionados, na seção 3 a metodologia utilizada, a seção 4 descreve os resultados, a seção 5 apresenta as discussões, a seção 6 demonstra as ameaças à validade e, por fim, a seção 7 a conclusão.

2 TRABALHOS RELACIONADOS

Lu et al. [6] discutem sobre a necessidade de se escolher a linguagem de programação correta para o desenvolvimento de projetos de *software*. Foram analisados, na comunidade Gitee, os números de seguidores, *forks* e *collections* dos projetos e das linguagens de programação. Costa and Ponciano [2] propuseram uma investigação do comportamento dos programadores ao utilizarem repositórios hospedados no *GitHub* e como esses comportamentos influenciam no sucesso dos repositórios analisados. Esse estudo demonstra um padrão de comportamento dos desenvolvedores em relação aos tipos de repositórios, fazendo uma associação entre o perfil e repositórios, demonstrando a necessidade de se estudar também a interação dos programadores no contexto das *stacks*.

Montandon et al. [8] investiga as abordagens baseadas em aprendizado de máquina para identificar automaticamente os papéis técnicos dos desenvolvedores de código aberto. A pesquisa foi realizada com desenvolvedores rotulados em seis funções diferentes: *back-end*, *front-end*, *full-stack*, *mobile*, *DevOps* e *data science*. Os resultados mostraram que as linguagens de programação são os recursos mais relevantes para prever os papéis investigados. Este artigo relaciona ao presente projeto por compreender a importância da escolha da linguagem de programação para as qualidades de projetos no GitHub.

Diferentemente dos trabalhos correlatos, esta pesquisa busca compreender o comportamento de desenvolvedores dentro de uma comunidade de código aberto, analisando-os pela perspectiva da *stack* e linguagens com que trabalham.

3 METODOLOGIA

Este trabalho promove um estudo de caso exploratório. Para a realização deste experimento, o *dataset* contempla dados referentes a uma seleção de repositórios do GitHub, dispondo de determinadas informações – número de estrelas (*stargazers*) e a quantidade de *issues* abertas e fechadas, por exemplo – além de suas *pull requests*.

A coleta de dados ocorreu por meio de um *script* responsável pelo consumo da API GraphQL do GitHub. Dessa forma, foram feitas as seguintes etapas para a coleta de dados: 1) seleção de repositório para as linguagens definidas, bem como aqueles que possuíam tópico referente às *stacks*; 2) seleção de *pull requests* referentes aos repositórios coletados na etapa anterior; 3) seleção de repositórios que possuam tópico referente às *stacks* escolhidas independentemente do filtro por linguagem de programação utilizada. Vale ressaltar que, para cada uma das consultas estabeleceu-se alguns critérios para seleção, os quais serão abordados nas subseções seguintes.

3.1 Seleção de repositórios por *stack* e linguagem

A primeira coleta realizada são de repositórios de projetos que contém as linguagens de programação mais populares de acordo com o próprio GitHub, além daqueles que possuam o tópico referente às *stacks* escolhidas, conforme melhor explicado adiante.

A seleção das linguagens de programação dos repositórios coletados para execução desta pesquisa considera o *"The 2021 State of the Octoverse"* [4], também denominado como Relatório Octoverse, idealizado pelo GitHub referente ao ano de 2021 - versão mais recente até a realização deste experimento. Este documento divulga as tendências atuais no mundo do desenvolvimento, bem como resultados preditivos a respeito de desenvolvedores, equipes, organizações e comunidades presentes na plataforma. Dessa forma, tendo em vista os resultados, escolheu-se repositórios das quatro linguagens de programação mais populares no ano, sendo estas: JavaScript, Python, Java e TypeScript.

Outro fator levado em conta para a coleta dos repositórios são as "*stacks*". Portanto, para realização deste experimento consideraram-se as *stacks front-end* e *back-end*, as quais são associadas aos "tópicos" no GitHub (em inglês, *topics*). Os tópicos permitem aos usuários do GitHub pesquisar e encontrar repositórios de um assunto específico, áreas, grupos de afinidades e entre outras características importantes, localizando projetos os quais possam contribuir. Logo, utilizou-se os tópicos "*front-end*" e "*back-end*" para coleta dos repositórios. É importante ressaltar que o tópico "*front-end*" foi utilizado nas linguagens JavaScript e TypeScript devido à maior familiaridade do uso destas para aplicações de *front-end*, enquanto "*back-end*" para todas as presentes no experimento: Java, Python, JavaScript e TypeScript. Destaca-se ainda que, para a API do GitHub, a utilização ou não do hífen no nome das *stacks* na *query* é indiferente, retornando os mesmos repositórios. Por fim, a escolha de JavaScript e TypeScript para a *stack back-end* se dá pela crescente utilização das tecnologias com auxílios de *frameworks* ou bibliotecas como o Node.js nesse segmento.

Além disso, um outro fator foi a data de "*pushed*", ou seja, termo que se refere ao envio de alterações (*commits*) de uma *branch* e repositório Git local para um repositório remoto. Dessa forma, definiu-se a data a partir de 01 de janeiro de 2020, data em que as quatro linguagens de programação escolhidas ocupam o *ranking* como quatro das dez mais utilizadas na plataforma, conforme o Relatório *Octoverse* [4].

Por fim, é necessário considerar a quantidade de repositórios que serão coletados por linguagem/*stack*. Com o auxílio da API do GitHub utilizando GraphQL, foi possível filtrar e visualizar a quantidade de repositórios que atendiam os critérios definidos acima. Dessa forma, para *back-end* encontrou-se um total de 5.251 repositórios para todas as linguagens, sendo, respectivamente, para Java, JavaScript, Python e TypeScript os seguintes números: 578, 2.734, 791 e 1.148 repositórios. Enquanto *front-end* conta com um total

de 4.628 repositórios. Sendo, respectivamente, para JavaScript e TypeScript os seguintes números: 2.734 e 1.894.

Sendo assim, foram coletados 500 repositórios por linguagem e por *stack*. O critério para buscar apenas 500 repositórios baseia-se no fato de que, para realizar a pesquisa no GitHub, os filtros de *stack* limitaram o número de projetos retornados. Na linguagem Java, por exemplo, foram retornados apenas 578 repositórios. Além disso, os filtros foram necessários para que a busca realizada retornasse apenas repositórios que possuíssem os tópicos *back-end* ou *front-end*, dentro das linguagens analisadas (Java, Python, TypeScript e JavaScript). Ainda, ao tentar buscar os repositórios mais populares utilizando os filtros mencionados acima, foi retornada uma quantidade insignificante de projetos, o que levou à adoção da busca ordenada por estrelas para cada repositório.

3.2 Coleta de *pull requests*

A partir dos repositórios selecionados na etapa anterior, realizou-se a coleta dos *pull requests*, sendo atribuídos como uma forma de análise de contribuição. Em suma, o *pull request* é uma forma cooperativa de compartilhar uma ou mais mudanças em código de um repositório, assim, as mudanças ocorridas na *branches* secundárias serão publicadas na *branch* principal após uma ou mais revisões e/ou discussões entre membros do time em questão.

Sendo assim, o critério adotado foi a seleção de *pull requests* com status *merged* ou *closed*, que tiveram uma ou mais revisões e cujo intervalo entre a abertura e mudança de *status* foi de no mínimo uma hora. A motivação da definição é selecionar *pull requests* já trabalhadas, bem como mitigar revisões automáticas de código.

3.3 Seleção de repositórios por *stack*

Por fim, neste experimento executou-se um *script* responsável por coletar uma lista de repositórios por *stack*, ou seja, *front-end* e back-end.

Sendo assim, neste momento não se levou em conta as linguagens de programação utilizadas nos projetos em questão. Foi realizado um filtro da quantidade de repositórios cujos tópicos referem-se às *stacks*, obtendo-se 7.918 repositórios *back-end* e 12.637 *front-end*.

Assim, coletaram-se os 1.000 primeiros repositórios, ordenados por estrela de forma decrescente, ou seja, do maior para o menor número de estrelas. Define-se este valor visto uma limitação imposta no uso da API GraphQL do GitHub, a qual determina que para cada pesquisa do GitHub forneça até 1.000 resultados.

4 RESULTADOS

Esta seção objetiva analisar os resultados encontrados para cada uma das questões desta pesquisa.

4.1 RQ1 - Qual é a linguagem mais popular de *front-end* e qual a mais popular de *back-end*?

Conforme abordado na metodologia, foram coletados 1.000 repositórios classificados com o tópico *front-end* e outros 1.000 com o tópico *back-end* com o objetivo de averiguar, a partir da proporção de linguagens primárias dentre cada *stack*, qual é a linguagem mais popular dentre cada uma das *stacks* – caracterizando uma das métricas. Assim, é evidenciado na Figura 1 que, no caso da *stack back-end*, a proporção de repositórios da linguagem JavaScript é maior: 21,4% contra os 14,6% do TypeScript, 11,9% do Python e 5,8% do Java. É importante notar que 7,6% dos repositórios não possuíam linguagem primária, estando classficados no gráfico como "None". Já para a *stack front-end*, verifica-se que 34,2% dos repositórios são de JavaScript, enquanto 17,8% são de TypeScript e 12,2% não referem-se a nenhuma linguagem.

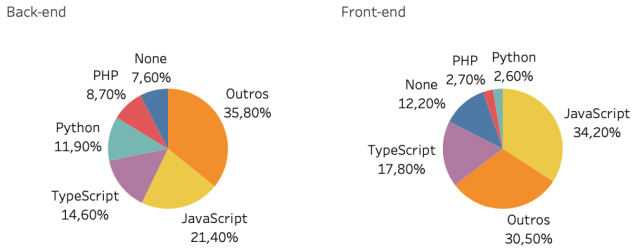

Figura 1: Proporção de linguagens nos repositórios por *stack*. Para cada *stack*, foram coletados 1000 repositórios.

Em termos da quantidade de estrelas por linguagem para cada uma das *stacks*, para a *stack front-end*, a linguagem JavaScript apresentou o total de 637.514 estrelas, contra 242.409 estrelas para repositórios sem uma linguagem definida e 176.747 estrelas para repositórios cuja linguagem primária é o TypeScript. Por outro lado, na *stack front-end*, JavaScript também se destacou, apresentando o total de 109.488 estrelas em repositórios dessa linguagem, contra 73.639 estrelas de repositórios que não possuem linguagem definida; 46.491 estrelas para repositórios da linguagem Go; 25.252 para TypeScript; 23.484 para PHP; 13.080 para Python e 9.674 estrelas para repositórios Java.

Tendo em vista o que foi exposto, pode-se considerar que a linguagem mais popular de *front-end* e *back-end*, a partir do modo de caracterização escolhido – que utilizou os "tópicos" do GitHub – foi JavaScript para ambos os casos.

4.2 RQ2 - Qual *stack* possui maior contribuição da sua respectiva comunidade?

A fim de analisar a primeira métrica, pode-se verificar a Figura 2, que evidencia, por meio de um gráfico boxplot, a distribuição de *pull requests* para cada *stack*. Nota-se que, para a *stack back-end*, a mediana de *pull requests* com a situação *merged* é de 109, enquanto para a *stack front-end* é de 140. Ainda, há menor distribuição de valores para a *stack back-end* quando comparada à *stack front-end*. Evidencia-se, assim, um maior engajamento por parte dos desenvolvedores *front-end*, tendo em vista que a mediana de PRs "*merged*" – que foram integradas, portanto, ao código principal – dessa *stack* é maior em relação a outra.

Para a segunda métrica, é possível analisar a Figura 3, que mostra, em gráficos distintos, a distribuição de *issues* com a situação *opened* (aberta) e *closed* (fechada) por *stack*. Verifica-se, portanto, que, para ambos os *status* de *issues*, a mediana dos repositórios *front-end* são maiores: 96 *issues* contra 63 *issues*, no caso daquelas com o *status closed*; e a mediana de 47 *issues* contra 32 para aquelas com o *status opened*.

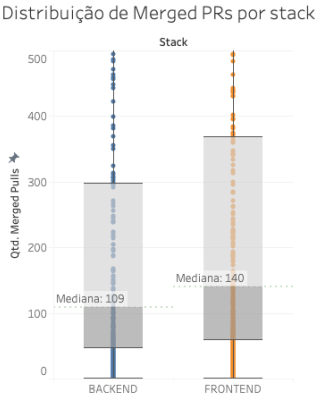

Figura 2: Distribuição, por *stack*, de *pull requests* que possuem a situação *merged*.

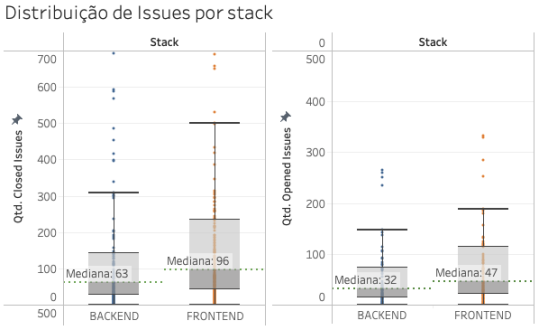

Figura 3: Distribuição, por *stack*, de *issues* que possuem a situação *open* e *closed*

Dado o que foi analisado, constata-se que os repositórios de *front-end* apresentam maior contribuição da sua respectiva comunidade, tanto em relação às *issues*, quanto em relação às *pull requests*.

4.3 RQ3 - Com qual *stack* há maior engajamento dos desenvolvedores?

Com o intuito de verificar em qual *stack* há maior engajamento por parte desenvolvedores, foram estabelecidas duas métricas: tempo mediano mensal para o fechamento das PRs por *stack* e mediana de comentários, participantes e *reviews* das PRs por *stack*.

Assim, conforme se vê na Figura 4, que representa o tempo de análise mediano mensal de *merged pull requests* por *stack* – em que a *stack front-end* está representada de laranja e a *back-end*, de azul – nota-se que, na maioria do meses, a mediana *stack front-end* apresenta-se menor, como foi o caso de junho de 2021, por exemplo, em que o tempo mediano para o *merge* de uma *pull request* foi de 20h, enquanto que, para a *stack back-end*, o tempo foi de 53,5 horas. Evidencia-se, portanto, um menor tempo na *stack front-end* para que as *pull requests* sejam adicionadas ao código principal, passando a ter o *status merged*, caracterizando, assim, um maior engajamento dessa comunidade.

Além disso, para a segunda métrica, pode-se analisar a Figura 5, que, por meio de um gráfico boxplot, mostra como se dá a distribuição de comentários, participantes e *reviews* para as *pull requests* de cada *stack*. Nota-se que, para todos os casos, a *stack front-end* apresenta maior mediana: no caso de comentários, 29,5 contra 26 do *back-end*; para os participantes, a mediana de 15 do *front-end* contra 9,5 do *back-end* e, em termos de *reviews*, 44,5 contra 35,5 do *back-end*. Evidencia-se, ainda, uma maior distribuição dos dados da *stack front-end* para todas as dimensões em relação à *stack back-end*, principalmente para o número de participantes.

Dessa forma, tendo-se em vista as análises feitas, constata-se que a *stack front-end* apresenta maior engajamento em termos das métricas selecionadas.

5 DISCUSSÃO

Considerando que este trabalho busca avaliar o engajamento e contribuição nas *stacks front-end* e *back-end*, no contexto dos repositórios do GitHub, nota-se, a partir dos resultados obtidos nas RQs, que a *stack front-end* apresentou, no geral, um maior nível de mobilização dos usuários.

É interessante notar que, até mesmo quando se analisou, na RQ1, a quantidade de estrelas e a proporção de repositórios por linguagem para cada uma das *stacks*, as linguagens JavaScript e TypeScript – escolhidas de forma secundária para a *stack back-end* por também serem bastante utilizadas nesse contexto – acabaram se sobressaindo a frente do Java e Python, que foram escolhidas as linguagens principais de *back-end*. Portanto, além do fato das linguagens JavaScript e TypeScript já serem bastante utilizadas pela comunidade da *stack back-end*, acredita-se também que o modo como foram obtidos os repositórios para cada *stack* (utilizando-se dos "tópicos" do GitHub) tenha deixado de fora uma quantidade significativa de repositórios que poderiam igualmente encaixarem-se nessa classificação de *stacks*.

Além disso, em relação às RQs 2 e 3, que buscaram avaliar o engajamento e contribuição para cada *stack*, o lado *front-end* também se destacou. Notou-se que a quantidade de *pull requests* que foram "mergeadas" e também o número de *issues* - tanto abertas, quanto fechadas – foi maior para a *stack front-end*, demonstrando uma maior quantidade de submissões de correções no código e relato de problemas. Ainda, em termos do tempo que se leva para uma *pull request* ser aceita, nota-se que a comunidade *front-end* mostra-se mais ativa para analisar e decidir pelo sucesso da contribuição feita por outro desenvolvedor. Acredita-se que o comportamento é notado pelo fato de repositórios dessa *stack* demandarem mais manutenção e modificações, pela constantes demandas de usuários em relação ao aspecto visual e, ainda, o fato de um projeto da *stack back-end* poder atender vários projetos *front-end*.

6 AMEAÇAS À VALIDADE

Pensando-se nas ameaças à validade do experimento, identificou-se riscos relacionadas a validade a conclusão, interna e externa. Em relação à validade à conclusão, tem-se a maneira como realizou-se a coleta de repositórios utilizando os tópicos do GitHub para identificar a *stack* do repositório, o que limitou o quantidade de repositórios coletados, bem como a presença de vários repositórios com zero estrelas. Além disso, foram buscados apenas 500

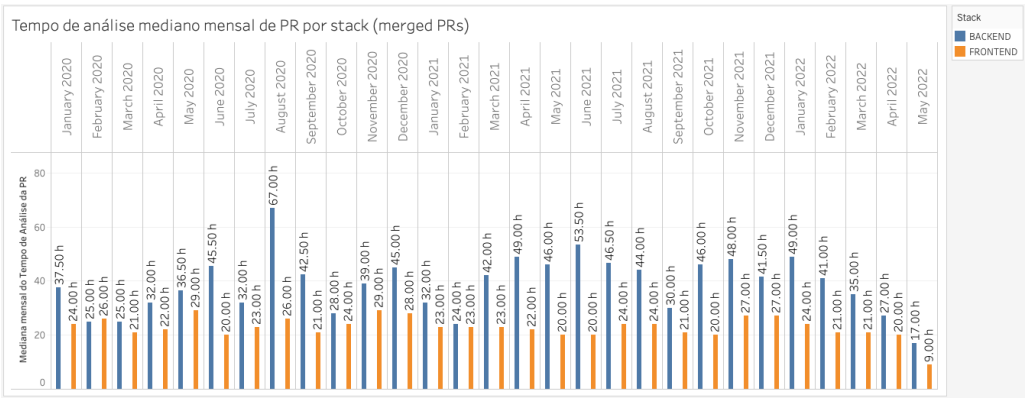

Figura 4: Tempo mediano mensal (em horas), por *stack*, para análise de *pull requests*. A linha laranja representa a *stack front-end* e a linha azul, a *stack back-end*.

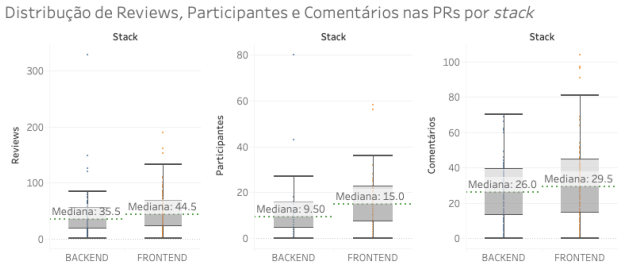

Figura 5: Distribuição, por *stack*, de comentários, participantes e *reviews* para as *pull requests*.

repositórios por linguagem que, conforme explicado na seção de Metodologia, justifica-se pelo baixo número de repositórios que seriam retornados caso fosse realizada a ordenação por estrelas, por exemplo. Pensando-se em validade externa, há uma complexidade maior para generalização caso opte por menos linguagens devido a quantidade limitada de repositórios, dessa forma, mitigou-se coletando repositórios de mais linguagens para formar uma amostra maior com 3.000 repositórios. Por fim, para ameaças de validade interna, conta-se com a possibilidade de falhas no *script* responsável pela coleta de dados, podendo causar erro ao salvar dados importantes, bem como a perda dessas informações. Dessa forma, mitigou-se por meio do armazenamento dos dados em um banco de dados, bem como os pontos de parada durante a execução do *script* para testar a sua precisão e funcionamento correto.

7 CONCLUSÃO

O uso de repositórios GitHub para o versionamento e armazenamento de códigos de um sistema de software tornou-se uma ferramenta essencial para desenvolvedores de todas as áreas. O experimento exploratório descrito ao longo do texto focou em analisar o engajamento da comunidade dentro da plataforma, dividindo a coleta de dados por linguagem de programação de acordo com as mais utilizadas e dividindo-as por *stacks* - *front-end* e *back-end*. O trabalho também respondeu, de maneira assertiva, as perguntas propostas na metodologia e determinou que:

RQ1. JavaScript é a linguagem de programação mais popular tanto no *back-end* quanto no *front-end*
RQ2. A *stack front-end* recebe maior contribuição dos usuários
RQ3. A *stack front-end* recebe maior engajamento dos desenvolvedores presentes na plataforma

Os resultados mostram uma maior popularidade do desenvolvimento *front-end* de acordo com a comunidade e uma preferência ao desenvolvimento com a linguagem JavaScript em ambos os *stacks*. Portanto, acredita-se que desenvolvedores iniciantes poderão ter maior amparo e *feedbacks* mais constantes caso optem por trabalhar com *front-end*, principalmente com JavaScript. No que se refere aos trablhos futuros, sugere-se que haja uma pesquisa sobre a relação entre a qualidade de um desenvolvimento back-end e um front-end em linguagens populares, como JavaScript e TypeScript.

REFERÊNCIAS

[1] Ariadne MBR Carvalho. 2001. *Introdução à engenharia de software*. Ed. da Unicamp.
[2] Victor Costa and Lesandro Ponciano. 2018. Minerando Padrões de Interação de Programadores com Repositórios na Plataforma GitHub.
[3] Jorge Henrique Cabral Fernandes. 2003. Qual a prática do desenvolvimento de software? *Ciência e Cultura* 55 (04 2003), 29 – 33. http://cienciaecultura.bvs.br/scielo.php?script=sci_arttext&pid=S0009-67252003000200021&nrm=iso
[4] GitHub. 2021. The 2021 State of the Octoverse. https://octoverse.github.com/
[5] Sherlock A. Licorish and Stephen G. MacDonell. 2017. Exploring software developers' work practices: Task differences, participation, engagement, and speed of task resolution. *Information Management* 54, 3 (2017), 364–382. https://doi.org/10.1016/j.im.2016.09.005
[6] Dongdong Lu, Jie Wu, Yongxiang Sheng, Peng Liu, and Mengmeng Yang. 2020. Analysis of the popularity of programming languages in open source software communities. In *2020 International Conference on Big Data and Social Sciences (ICBDSS)*. IEEE, 111–114.
[7] João Eduardo Montandon, Cristiano Politowski, Luciana Lourdes Silva, Marco Tulio Valente, Fabio Petrillo, and Yann-Gaël Guéhéneuc. 2021. What skills do IT companies look for in new developers? A study with Stack Overflow jobs. *Information and Software Technology* 129 (2021), 106429. https://doi.org/10.1016/j.infsof.2020.106429
[8] João Eduardo Montandon, Marco Tulio Valente, and Luciana L. Silva. 2021. Mining the Technical Roles of GitHub Users. *Information and Software Technology* 131 (2021), 106485.
[9] Lais MA Rocha, Thiago Henrique P Silva, and Mirella M Moro. 2016. Análise da Contribuição para Código entre Repositórios do GitHub. (2016).
[10] Liz Simmons. 2022. The Difference Between Front-End vs. Back-End | ComputerScience.org — computerscience.org. https://www.computerscience.org/bootcamps/resources/frontend-vs-backend/.