Caracterização de bibliotecas de Componentes

Aylton B. de Almeida Junior[1], **Lucca Romaniello Benjamin**[2], **Pedro Paulo Andrade**[3]

[1]Engenharia de Software – Pontifícia Universiade Catolica de Minas Gerais (PUC MG)
Belo Horizonte – MG – Brazil

{abajunior, lbenjamin, pedro.faria.1115474} @sga.pucminas.br

1. Introdução

”Nos últimos anos, tornar as coisas mais utilizáveis se tornou uma responsabilidade de quase todos. Agora, designers visuais e desenvolvedores costumam fazer coisas como design de interação (decidir o que acontece a seguir quando o usuário clica, toca ou desliza) e arquitetura de informações (descobrir como tudo deve ser organizado)”. [Krug 2014].

Com base na citação de Steve Krug, percebemos que as interfaces visuais no mundo digital se tornam amplamente necessárias, especialmente para possibilitar que usuários completem das mais diversas tarefas em produtos digitais. Com o avanço da tecnologia e a crescente demanda por desenvolvimento de sistemas amigáveis aos usuários finais, se fazem necessárias ferramentas que auxiliem na criação dessas aplicações propostas.

Tendo em vista o ambiente de desenvolvimento de interfaces, é inegável dizer que dentre os ecossistemas de desenvolvimento existentes o da linguagem Javascript é o mais popular, mais especificamente o do *React*, esse que é o mais popular no mundo tendo como base o número de estrelas no Github. Com isso é comum que diversas opções existam para resolver um mesmo problema quando pensamos no desenvolvimento utilizando essa ferramenta, de forma que fazer a diferenciação entre quais bibliotecas deveriam ser usadas e quais podem causar problemas futuros pode ser bem complicado.

Esse trabalho nasce a partir dessa dor, de não saber qual biblioteca de componentes uma equipe deveria usar em sua aplicação. Nas pesquisas iniciais é perceptível que não existem muitos outros trabalhos com esse objetivo em mente, mesmo que esse seja um tópico relevante para o contexto atual de desenvolvimento de aplicações.

Ao pesquisarmos rapidamente pelo Github conseguimos encontrar facilmente cerca de 2.890 bibliotecas de componentes [Github], esse número mostra que existe a necessidade de fazer uma escolha mais consciente sobre qual biblioteca será utilizada. A escolha de uma biblioteca ruim pode gerar problemas de manutenção, desenvolvimento e até mesmo performance, seja por ela não receber muitas atualizações ou não ter uma comunidade muito ativa, dificultando o aprendizado dela.

Por conta disso esse estudo tem como objetivo comparar bibliotecas de componentes desenvolvidas para React, a fim de definir possíveis parâmetros que mostrem o motivo de uma equipe escolher uma biblioteca sobre a outra. Junto disso temos como objetivo secundário verificar se realmente existe uma necessidade de um estudo mais profundo sobre quais bibliotecas usar, ou se algo mais simples como a popularidade da biblioteca já não é suficiente pra a definir como uma boa opção.

Para alcançar nossos objetivos foram definidas três principais perguntas e suas métricas:

1. Qual a probabilidade de você ser respondido quando pergunta algo sobre a biblioteca para a comunidade?
    (a) Relação entre o número de questões respondidas pelo número total de perguntas, ambas encontradas no StackOverflow;
    (b) Relação entre o número de *Issues* fechadas com participação de mais de uma pessoa pelo número total de *Issues*;
2. Com que frequência a biblioteca recebe atualizações?
    (a) Relação entre número de *releases* e a idade do repositório em anos completos;
    (b) Diferença entre o tempo de fechamento e o de abertura de um *Pull Request* em dias completos;
3. A escolha de uma biblioteca mais popular tem um grande impacto no desempenho da aplicação?
    (a) *First contentful paint* em segundos
    (b) *Time to interactive* em segundos

As métricas serão recolhidas utilizando de três ferramentas: A API do Github, do StackExchange e a ferramenta Lighthouse.

O trabalho está organizado em 4 tópicos, sendo esses a Introdução, Trabalhos Relacionados, Metodologia e Resultados.

2. Trabalhos Relacionados

Nesta seção, estão descritos artigos relacionados ao tema proposto neste trabalho, ressaltando suas respectivas informações de forma geral, além de resultados e sua relação com este trabalho propriamente dito.

Which library should I use?. No trabalho de [López de la Mora and Nadi 2018], é introduzida a ideia da utilização de métricas para bibliotecas de *software* para ajudar desenvolvedores na tomada de decisão sobre qual biblioteca utilizar. Apesar do foco ser em bibliotecas para testes de *Application Programming Interfaces* (APIs) e não de componentes como no presente trabalho, métricas como popularidade, frequência de *releases* e *issues* e performance dos repositórios analisados no *GitHub* foram utilizadas para a análise. Desta forma, sua principal similaridade é de que os resultados obtidos não buscam responder à pergunta sobre qual biblioteca deve ser utilizada, e sim apresentar dados de utilização para que o próprio desenvolvedor escolha a biblioteca de acordo com sua devida necessidade.

End-user composition of interactive applications through actionable UI components. No trabalho de [Desolda et al. 2017], o principal objetivo é caracterizar funcionalidades que levam *frameworks* a minimizar o esforço para desenvolvimento de áreas de trabalho interativas por meio da reutilização de componentes. Sua principal contribuição se baseia numa análise técnica da arquitetura de determinadas plataformas de UI, que serviram para embasar a análise técnica realizada sobre as bibliotecas analisadas ao decorrer do trabalho.

A Case for Human-Driven Software Development. No trabalho de [Balland et al. 2013], foi desenvolvido um protótipo de *framework* de desenvolvimento,

focando no conceito de design centrado no usuário durante todo o processo. O trabalho facilitou o trabalho de stakeholders que não são do time técnico a se integrarem mais no processo de desenvolvimento, visto que artefatos como modelos de usuário e a interface em si do estudo de caso foram documentadas em conjunto. Assim, este trabalho permite maior entendimento sobre o ciclo de desenvolvimento de produtos digitais, facilitando a compreensão sobre o ponto de vista de desenvolvedores nesse tipo de projeto.

Understanding UI Integration. No trabalho de [Daniel et al. 2007], são dissecadas bibliotecas de interfaces e as tecnologias adotadas em sua utilização, abordando suas forças e fraquezas e levantando possíveis melhorias para a organização da camada de apresentação ao usuário. Em geral, este trabalho reforça a necessidade de uma possível padronização quanto ao desenvolvimento de interfaces em relação ao nível técnico e na forma como elas são estruturadas. Assim, sua principal relação com o presente trabalho se diz ao entendimento técnico e a necessidade de se adaptar à arquitetura que determinadas bibliotecas podem exigir em sua implementação, isto que pode influenciar a adoção ou não de uma biblioteca de componentes por parte dos desenvolvedores.

3. Metodologia

Para a realização da pesquisa será necessário recuperar dois grupos principais de dados, sendo o primeiro necessário para conseguir fazer a coleta do segundo. O primeiro grupo de dados é uma lista dos dois frameworks mais utilizados para desenvolvimento web utilizando como base o número de estrelas destes frameworks. Já o segundo se trata de 300 repositórios de bibliotecas UI destinadas a estes frameworks que são utilizadas para facilitar e dar agilizade durante o desenvolvimento de projeto. Após a busca de tais bibliotecas foram filtradas entre as top 10, middle 10, bottom 10 de acordo com o seu número de estrelas.

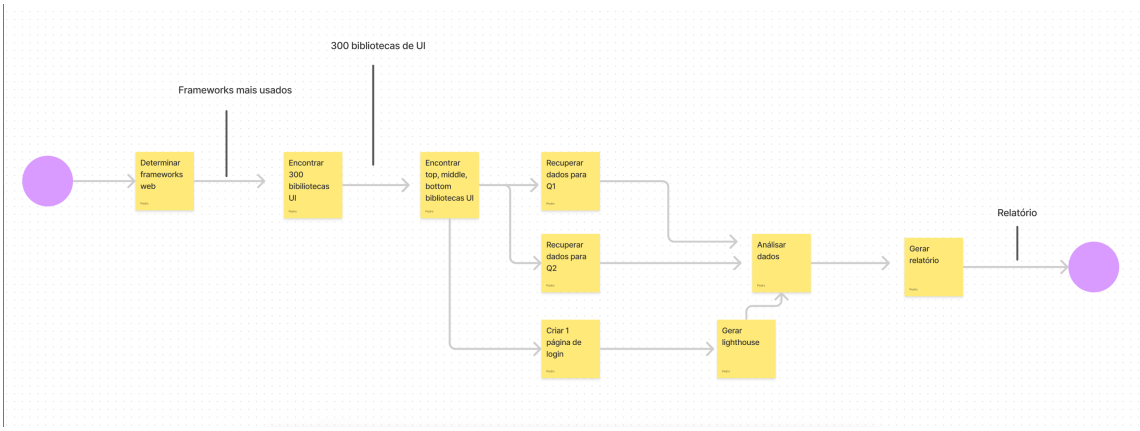

Figure 1. Imagem metodologia

3.0.1. *Dataset*

Com todas as buscas feitas e após aplicar o filtro de top, middle e bottom bibliotecas UI chegamos ao seguinte *dataset*

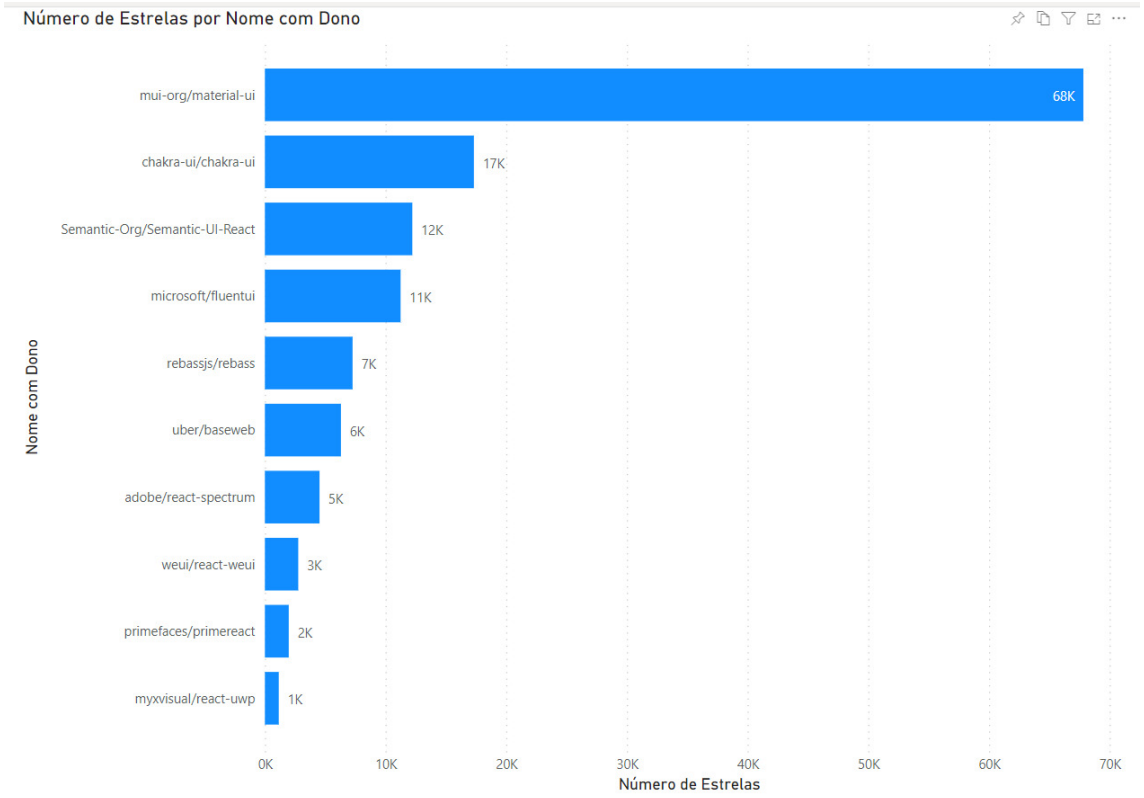

Figure 2. Imagem grupo das 10 bibliotecas UI com mais estrelas comparadas ao nosso dataset

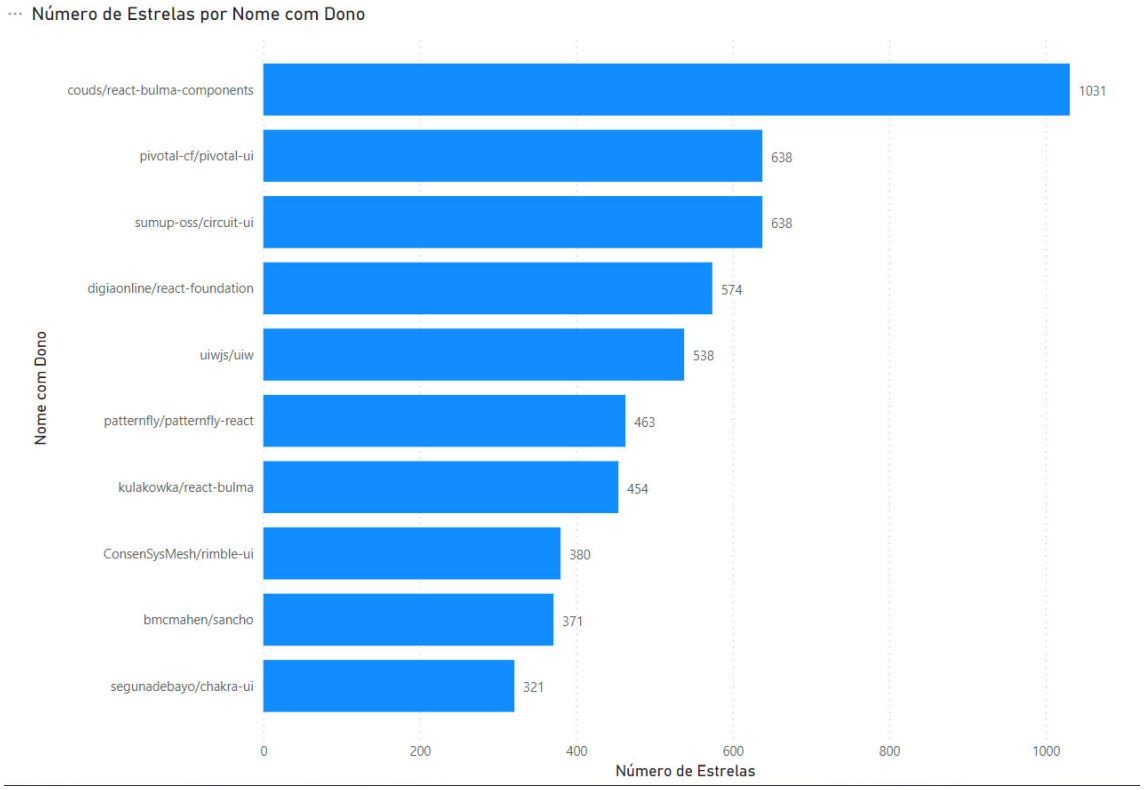

Figure 3. Imagem grupo das 10 bibliotecas UI intermediárias em questões de estrelas comparadas ao nosso dataset

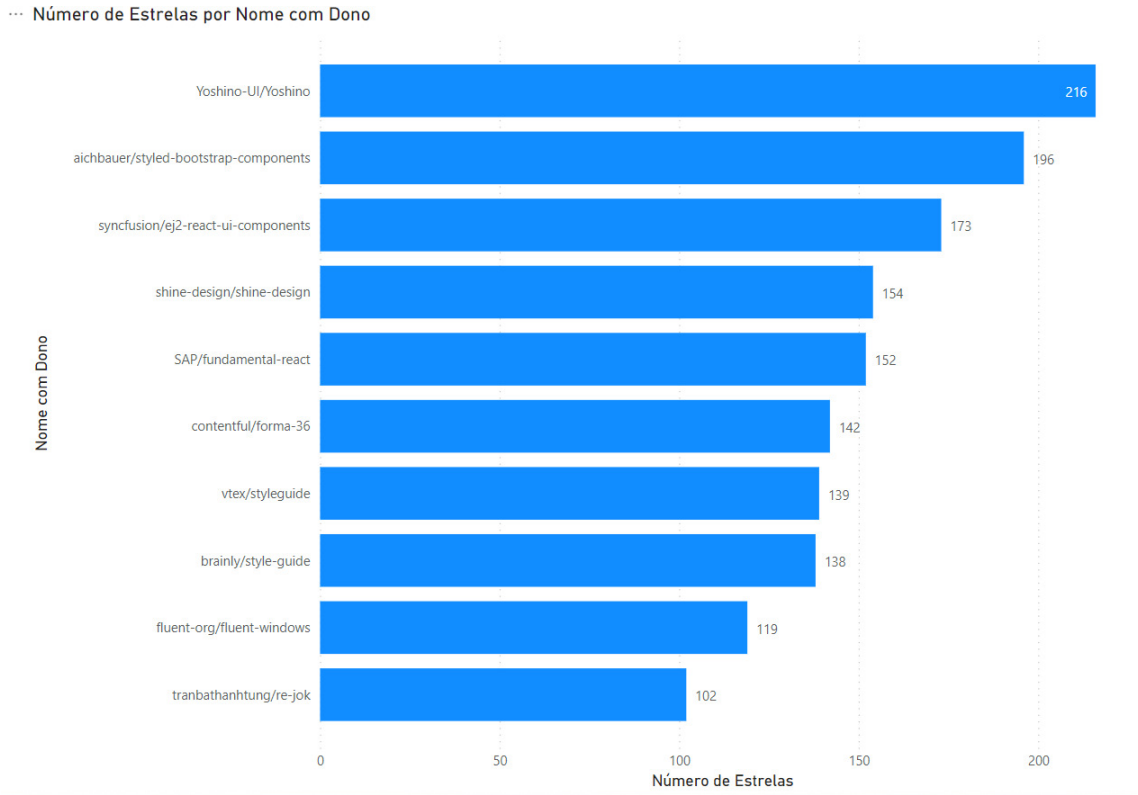

Figure 4. Imagem grupo das 10 bibliotecas UI com menos estrelas comparadas ao nosso dataset

3.1. Sobre a Questão 1

Primeiramente, assumimos que perguntas realizadas no site *StackOverflow* associadas à tais linguagens possuem um percentual de respostas maior do que aquelas menos populares. Dessa forma, utilizando o *Stack Exchange Data Explorer*, uma interface de consultas SQL na base dos sites pertencentes a essa stack (nesse caso, incluindo o *StackOverflow*), será calculado o percentual de questões respondidas para as bibliotecas analisadas.

Dentro do mesmo padrão será feito uma pesquisa das *issues* sobre o projeto criado no *GitHub* por meio da API oficial do *GitHub*, buscando todas as *issues*, entre elas abertas e fechadas, e comparando-as com o número de *issues* fechadas com participação de mais de um contribuinte.

3.1.1. Hipótese Nula e Alternativas

Hipótese Nula: Acredita-se que bibliotecas mais populares possuem uma taxa de resposta e *closed issues*, com mais de um participante, maior em comparação às bibliotecas em geral.

Hipótese Alternativa: Acredita-se a popularidade das bibliotecas de UI não interferem na taxa de respostas dentro to *StackOverflow* e na *closed issues* com mais de um participante maior ao comparar com as demais biblioteca

3.1.2. Variáveis dependentes e independentes

Variáveis dependentes: Índice de suporte feito pela comunidade de cada biblioteca UI

Variáveis independentes: Grupo de 30 bibliotecas UI separadas entre top 10, middle 10 e bottom 10 de acordo com sua popularidade.

3.2. Sobre a Questão 2

A primeira métrica da questão 2 é a relação entre o número de *Pull Requests* feitos por novos contribuidores pelo número total de *PRs*. Para isso serão recuperados todos os *Pull Requests* e seus autores, caso um autor possua apenas um *PR* ele é um considerado novo autor. Já a segunda métrica é o tempo médio para aceitação de *Pull Requests* feitos por novos contribuidores, esses que serão caracterizados da mesma forma que na métrica 1.

A fim de entender e caracterizar a frequência que as bibliotecas de UI recebem atualizações, serão analisados os repositórios destas bibliotecas por meio da API GraphQL disponibilizada pelo GitHub, esta que será consumida através de um programa desenvolvido na linguagem *Python*.

Para medir a frequência de atualização, serão coletadas duas métricas para cada repositório:

- Número de *releases* por idade do repositório (em anos completos);
- Tempo mediano entre a abertura e o fechamento de um *pull request* (em dias completos).

Vale destacar que para o cálculo do tempo mediano entre a abertura e o fechamento do *pull request*, todos os *pull requests* daquele repositório serão analisados.

Ambas essas métricas são acessíveis por meio da API GraphQL do Github. Dessa forma um *script* em Python será escrito, utilizando de cursores para ultrapassar o limite de 100 repositórios por *request*. Esse código tem como objetivo gerar um arquivo csv contendo uma sumarização dos dados para analise posterior.

3.2.1. Hipótese Nula e Alternativas

Hipótese Nula: Acredita-se que bibliotecas com mais popularidade possuem uma taxa de atualização maior em comparação às bibliotecas em geral.

Hipótese Alternativa: Acredita-se a popularidade das bibliotecas de UI não interferem na taxa de atualização ao comparar com as demais biblioteca

3.2.2. Variáveis dependentes e independentes

Variáveis dependentes: Índice de atualizações feita por cada comunidade de biblioteca UI

Variáveis independentes: Grupo de 30 bibliotecas UI separadas entre top 10, middle 10 e bottom 10 de acordo com sua popularidade.

3.3. Sobre a Questão 3

Com os *datasets* definidos será desenvolvido uma tela de login com cada biblioteca filtrada buscando entender o nível de performance de cada uma. Utilizando a ferramenta *Google Chrome Lighthouse*, um projeto *open-source* que informa a qualidade do site web, será possível calcular o desempenho de cada tela a partir de métricas como *First Contentful Paint* e *Time to interactive*.

3.3.1. Hipótese Nula e Alternativas

Hipótese Nula: Acredita-se que bibliotecas com mais popularidade possuem um melhor desempenho quando comparadas as demais bibliotecas

Hipótese Alternativa: Acredita-se a popularidade das bibliotecas de UI não interferem no resultado do teste de desempenho feito pelo *lighthouse*

3.3.2. Variáveis dependentes e independentes

Variáveis dependentes: Índice de desempenho de cada biblioteca UI

Variáveis independentes: Grupo de 30 bibliotecas UI separadas entre top 10, middle 10 e bottom 10 de acordo com sua popularidade.

4. Resultados

Nesta seção, serão apresentados os resultados obtidos a partir da execução da metodologia e, em seguida, os mesmos serão analisados.

4.1. Sobre a Questão 1

Issues e questionários respondidos dentro da comunidade são fatores extremamente relevantes no momento da escolha de uma biblioteca de UI, visto que, em diversas situações o desenvolvedor se depara com problemas que não está apto a resolver sozinho tornando necessário recorrerá comunidade que também utiliza desta biblioteca. Caso seja escolhida uma biblioteca que não possui um grande suporte isso pode acarretar em um atraso dentro do desenvolvimento do projeto. Com isso após definir o *dataset* conseguimos buscar todas as issues de cada projeto feito no *GitHub* e também foi feita a busca de todas as perguntas no *StackOverflow* para cada biblioteca de ui gerando os seguintes resultados.

4.1.1. Relação entre número de *issues* e o número de *issues* fechadas com mais de um participante

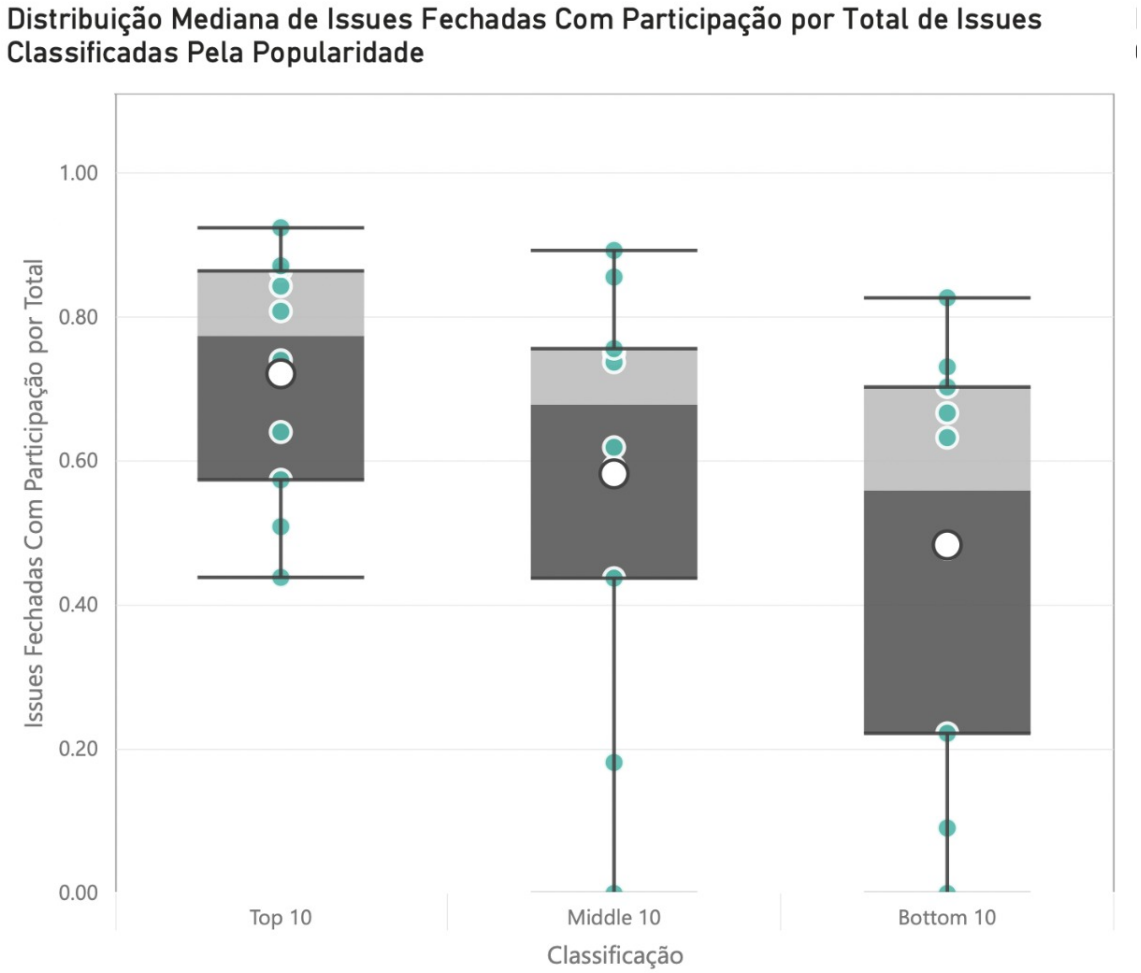

Figure 5. Distribuição Mediana de Issues Fechadas Com Participação por Total de Issues Classificadas Pela Popularidade

A partir do gráfico acima podemos perceber que o grupo 1 (top 10) obteve a maior mediana individual chegando a uma taxa de 92% que não fica muito distante do grupo 2 (middle 10) e 3 (bottom 10) que obtiveram como maior mediana entre seu grupos um valor de 89% e 83% respectivamente. Entretando podemos perceber que a existe uma grande diferença quanto as menores medianas de cada grupo, os quais o grupo middle 10 e bottom 10 tiveram como 0% a menor já o top 10 teve 44%.

Também é possível perceber um crescimento da mediana das taxas de *closed issue por total issue* onde no top 10 obteve uma mediana de 77% de *issues* fechadas com mais de um participante, este valor decresce no middle 10 que obteve uma mediana de 58% de *closed issues* chegando no bottom tem com a menor mediana de *closed issues* com mais de um participante obtendo apenas 48%.

Após a análise do gráfico é possível perceber que bibliotecas UI com mais popularidade possuem *closed issues* com colaboração de outras pessoas mais frequentes do que as menos populares. É possível também perceber o crescimento gradual a partir da popularidade.

4.1.2. Relação entre número total de perguntas no *StackOverflow* e o número de perguntas respondidas

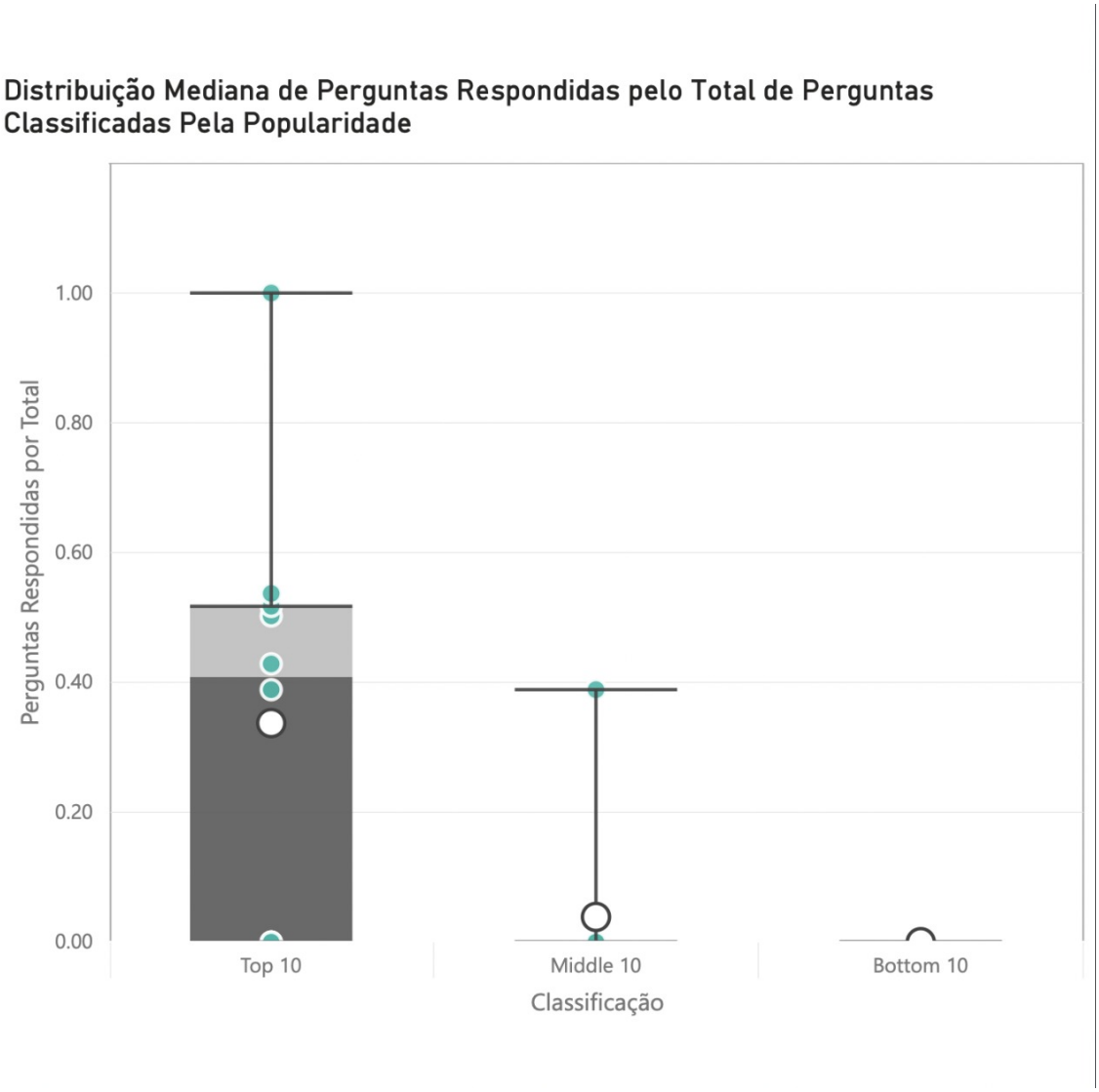

Figure 6. Distribuição Mediana de Perguntas Respondidas pelo Total de Perguntas Classificadas Pela Popularidade

Ao analisar o gráfico podemos concluir que apenas 1 das 30 bibliotecas obtiveram uma taxa de 100% de respostas dentro do *StackOverflow* e apenas o grupo top 10 com 40% de mediana de perguntas respondidas ficou acima dos 0%. O grupo middle 10 possui apenas uma biblioteca com perguntas no *StackOverflow* já o bottom 10 não teve perguntas.

4.2. Sobre a Questão 2

A frequência de atualização é um fator que pode ser considerado na adoção das bibliotecas em projetos de *front-end*. Enquanto frequentes atualizações podem indicar o cuidado com o projeto por parte da organização e uma comunidade ativa, é preciso identificar também se a biblioteca utilizada não está em fases iniciais e não atingiu uma maturidade considerável, por exemplo, o que pode afetar projetos que a utilizem. Também vale ressaltar que esses não são os únicos fatores a serem considerados para justificar sua adoção, mas sim os outros analisados no decorrer deste projeto.

Para cada grupo de repositórios selecionados no trabalho, foram coletadas as métricas do número de releases por idade do repositório (esta calculada em anos completos) e o tempo médio para um *pull request* ser aceito pela organização (calculado em dias completos).

Primeiramente, ao analisarmos os 30 repositórios de bibliotecas de componentes React em conjunto, sem categorizá-los por sua popularidade em número de estrelas no Github, foram obtidos os seguintes resultados:

- Mediana de 5,33 *releases* por ano de idade completo do repositório;
- Tempo médio de 3,6 dias completos entre a abertura e o fechamento de um *pull request*. Para fins de comparação e a fim de evitar valores destoantes no cálculo, pode-se destacar que o tempo mediano obtido foi de 2 dias completos.

Por outro lado, ao analisarmos os repositórios agrupados por suas devidas classificações, os resultados se mostram de outra forma, estes demonstrados a seguir. Para os gráficos de tempo médio de aprovação de *pull requests*, vale ressaltar que o eixo X não mostra todos os repositórios na imagem, apesar da figura do gráfico considerá-los.

4.2.1. Relação entre número de *releases* e a idade do repositório em anos completos

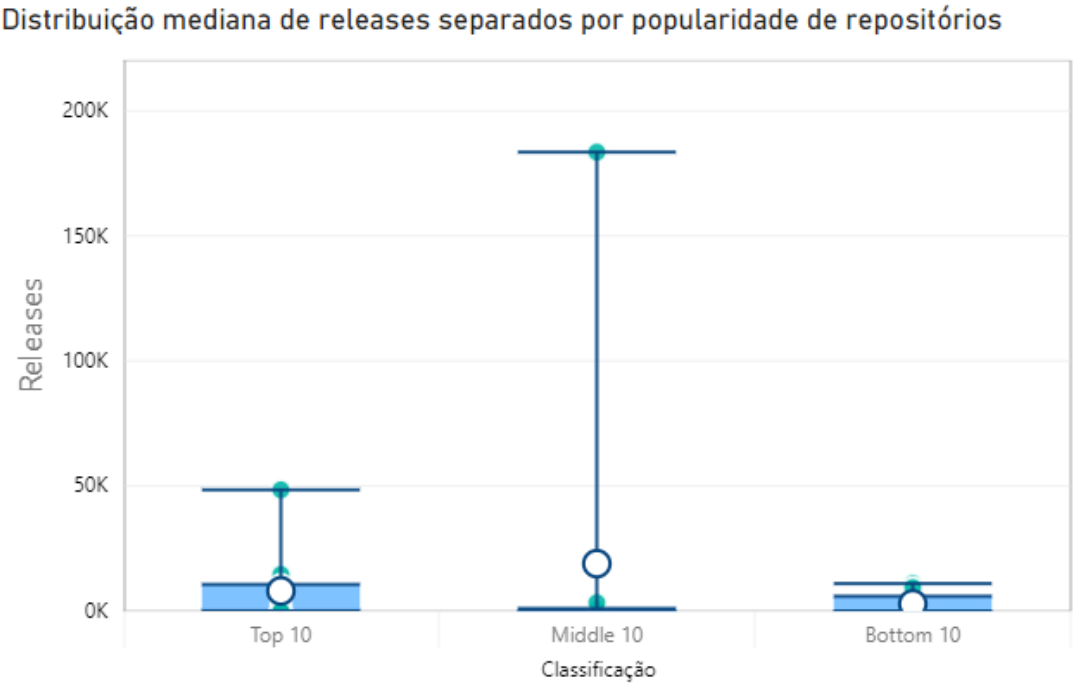

Figure 7. Distribuição mediana de releases separados por popularidade dos repositórios

A partir da análise do gráfico acima, demonstrando na Figura 7 e com auxílio de informações complementares da ferramenta, podem ser destacados os seguintes resultados:

1. Resultados do número mediano de releases de cada grupo
   (a) Os top 10 repositórios apresentam uma mediana de 1058 releases por ano de idade completo, sendo o maior entre os grupos;
   (b) Os repositórios intermediários apresentaram uma mediana de 373 releases por ano de idade, sendo o menor entre os grupos;
   (c) Os repositórios menos populares apresentaram uma mediana de 575 releases por ano de idade.
2. Resultados do interquartil de cada grupo
   (a) O interquartil dos top 10 repositórios apresentaram o valor de 10573 releases por ano de idade;
   (b) O interquartil dos repositórios intermediários apresentaram o valor de 1025 releases por ano de idade;
   (c) O interquartil dos repositórios menos populares apresentaram o valor de 5917 releases por ano de idade.

Com base nos resultados apontados acima, especialmente em relação ao número mediano de releases de cada grupo, percebemos que os repositórios intermediários apresentam um valor menor do que os outros grupos, o que em geral não era esperado. Os top

10 mantiveram o maior número de releases mas, ao contrário do esperado, os bottom 10 demonstraram ser mais assertivos quanto ao número de atualizações.

Em relação à análise dos interquartis, este que representa os valores apenas dos 50% repositórios (aqueles que estão no meio), pode ser apontado que o número de releases dos bottom 10 (5917) é muito maior do que os intermediários (1025). Ao compararmos apenas a mediana dos valores como foi analisado anteriormente, esta diferença parece ser baixa, mas isso indica que metade ao analisarmos metade dos repositórios de cada grupo, ela é muito mais significante e demonstra a diferença entre tais grupos.

4.2.2. Diferença entre o tempo de fechamento e o de abertura de um *Pull Request* em dias completos

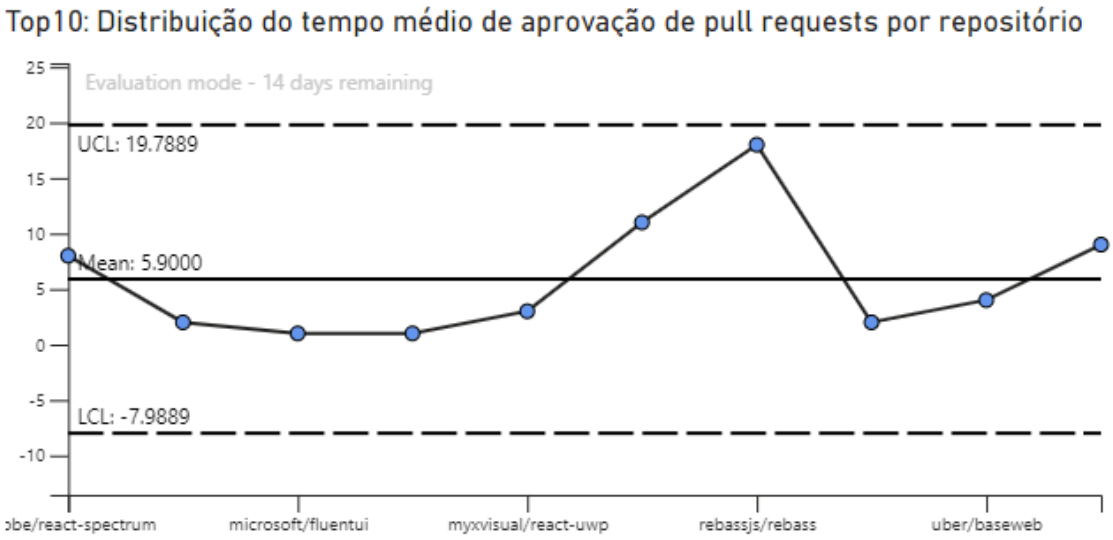

Figure 8. Distribuição do tempo médio de aprovação de PRs dos Top 10 repositórios

A partir da análise do gráfico acima, demonstrado na Figura 8, pode ser destacado que o tempo médio de aprovação dos repositórios mais populares foi de 5,9 dias completos, sendo maior do que a média geral. Também vale destacar que existem cerca de 2 valores que destoam de forma mais significante dentre os demais, que podem impactar neste resultado.

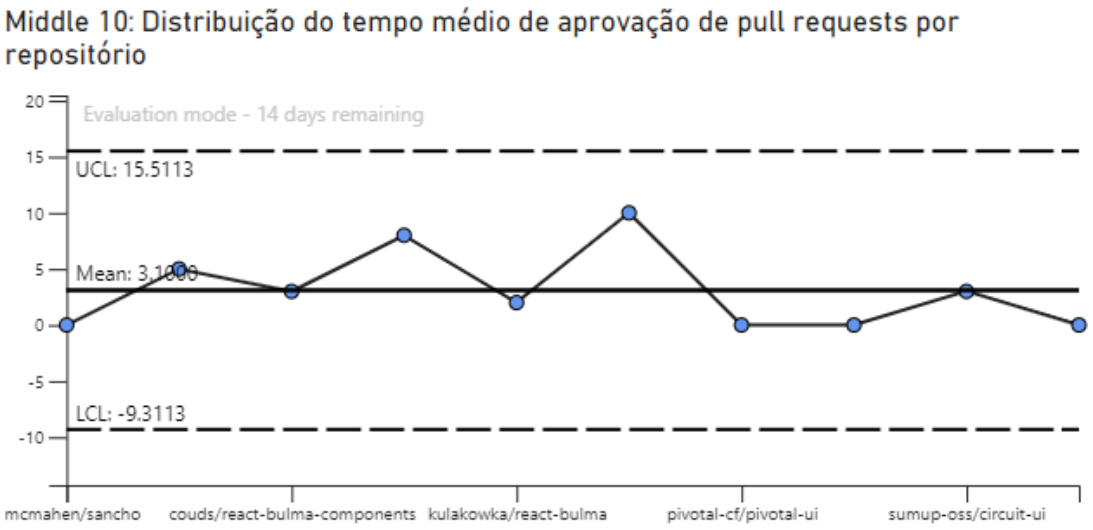

Figure 9. Distribuição do tempo médio de aprovação de PRs dos Middle 10 repositórios

Em relação aos repositórios intermediários, a partir da análise do gráfico acima, demonstrado na Figura 9, pode-se perceber que o tempo médio de aprovação se mantém com poucas variações em comparação aos repositórios mais avaliados descritos anteriormente. A média obtida foi de 3,1 dias completos para a aprovação de um *pull request*, o que se demonstra muito próximo da média geral de 3,6 dias, destacada no início desta seção.

Também é possível ressaltar a diferença do tempo de aprovação em comparação aos top 10 repositórios. Podemos assumir que isso se dá devido à um possível maior cuidado em relação ao impacto gerado pelos PRs na integração com o código existente. Por se tratarem de repositórios mais populares e utilizados por mais usuários, alterações devem ser realizadas com maior cautela, o que pode justificar que o tempo seja maior do que os repositórios intermediários.

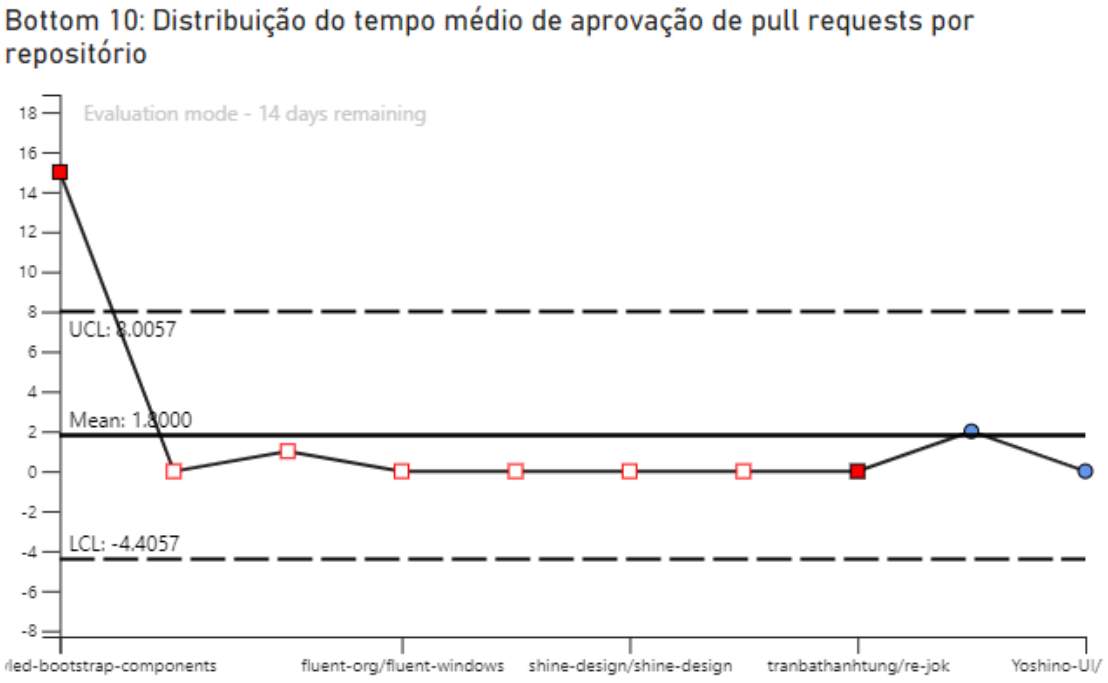

Figure 10. Distribuição do tempo médio de aprovação de PRs dos Bottom 10 repositórios

Com relação aos repositórios menos utilizados do dataset deste trabalho, o tempo médio de aprovação de um *pull request* obtido foi de 1,8 dias completos, conforme o gráfico da Figura 10. Entre os grupos de repositórios analisados, os repositórios bottom 10 se demonstram mais rápidos no tempo de aprovação e representam um número mais assertivo, visto que a distribuição do tempo para cada um dos repositórios é a mais próxima e constante dentro deste grupo.

Esse número poderia ser justificado por se tratarem de repositórios menos populares e, portanto, com menos contribuições. Além disso, assume-se que tais repositórios podem ser menores e menos complexos, o que facilitaria a contribuição da comunidade. Outro fator que pode impactar nesse número é que o projeto pode não ter atingido uma maturidade, logo, a revisão de *pull requests* exija menor esforço.

4.3. Sobre a Questão 3

Após o desenvolvimento de tela de login com as bibliotecas filtradas, os resultados obtidos a partir da ferramenta de análise de performance serão discutidos a seguir.

4.3.1. *First contentful paint*

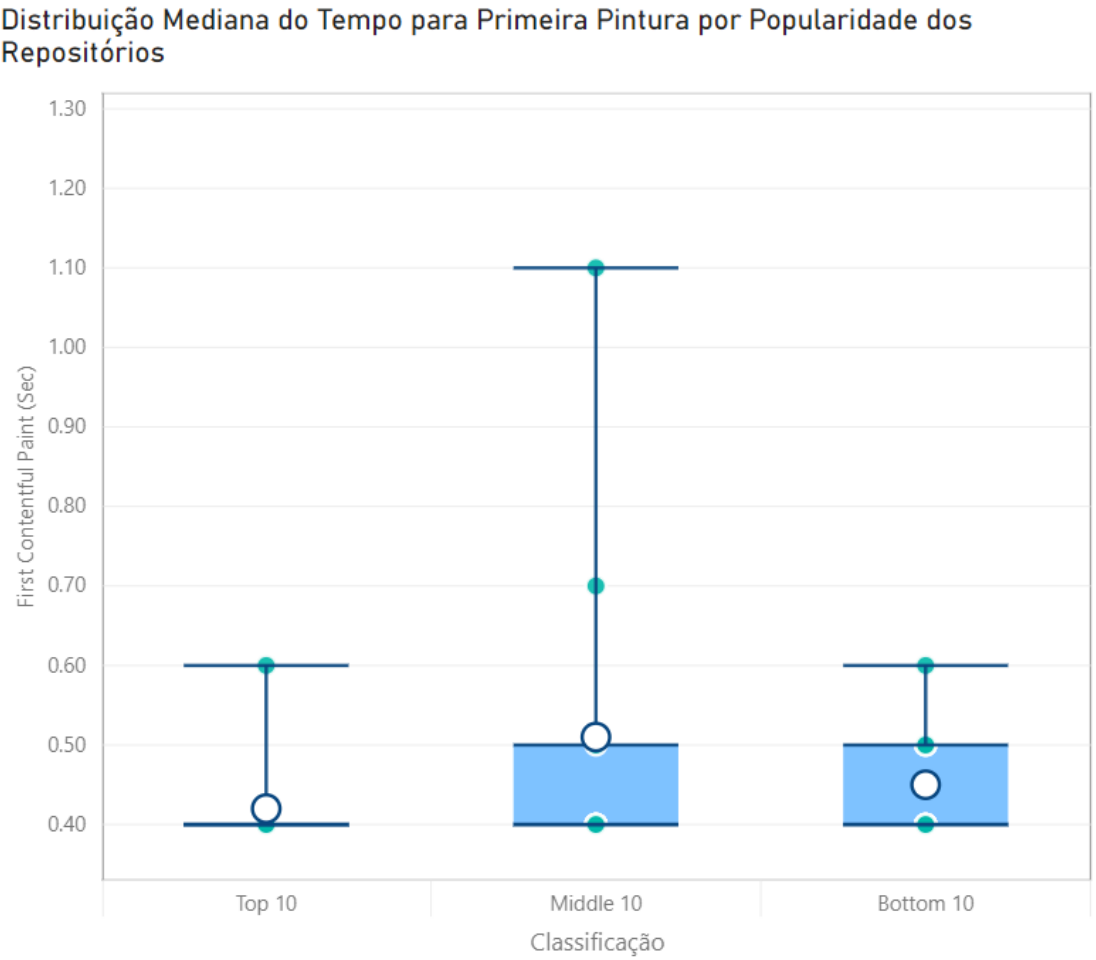

Figure 11. Distribuição mediana da primeira pintura por popularidade dos repositórios

Por meio da análise do gráfico acima representado pela Figura 11 e com auxílio de informações complementares da ferramenta, pode-se destacar que o valor mediano do tempo de primeira pintura é igual para todos os grupos de repositórios, sendo este valor de 0,4 segundos. Por outro lado, ao analisarmos a média para os top 10, middle 10 e bottom 10 repositórios, respectivamente, apresentaram o valor de 0,42, 0,51 e 0,45 segundos. Apesar dos números serem variados, acredita-se que tais variações seriam praticamente imperceptíveis para os usuários de plataforma que utilizem essas bibliotecas.

Dessa forma, apesar dos resultados variarem e o gráfico dar a impressão de que são muito diferentes, outros valores como a divisão de quartis também se mostraram muito próximos. A única diferença a ser destacada é que os repositórios intermediários possuem valores mais destoantes do restante, mas que aparentam não ter influência sob a análise geral. Assim, podemos assumir que a primeira pintura das telas desenvolvidas praticamente não afetaram a experiência do usuário (baseado somente nesta métrica) a ponto de justificar a não utilização de determinada biblioteca ou de um grupo de bibliotecas, por exemplo.

4.3.2. *Time to interactive*

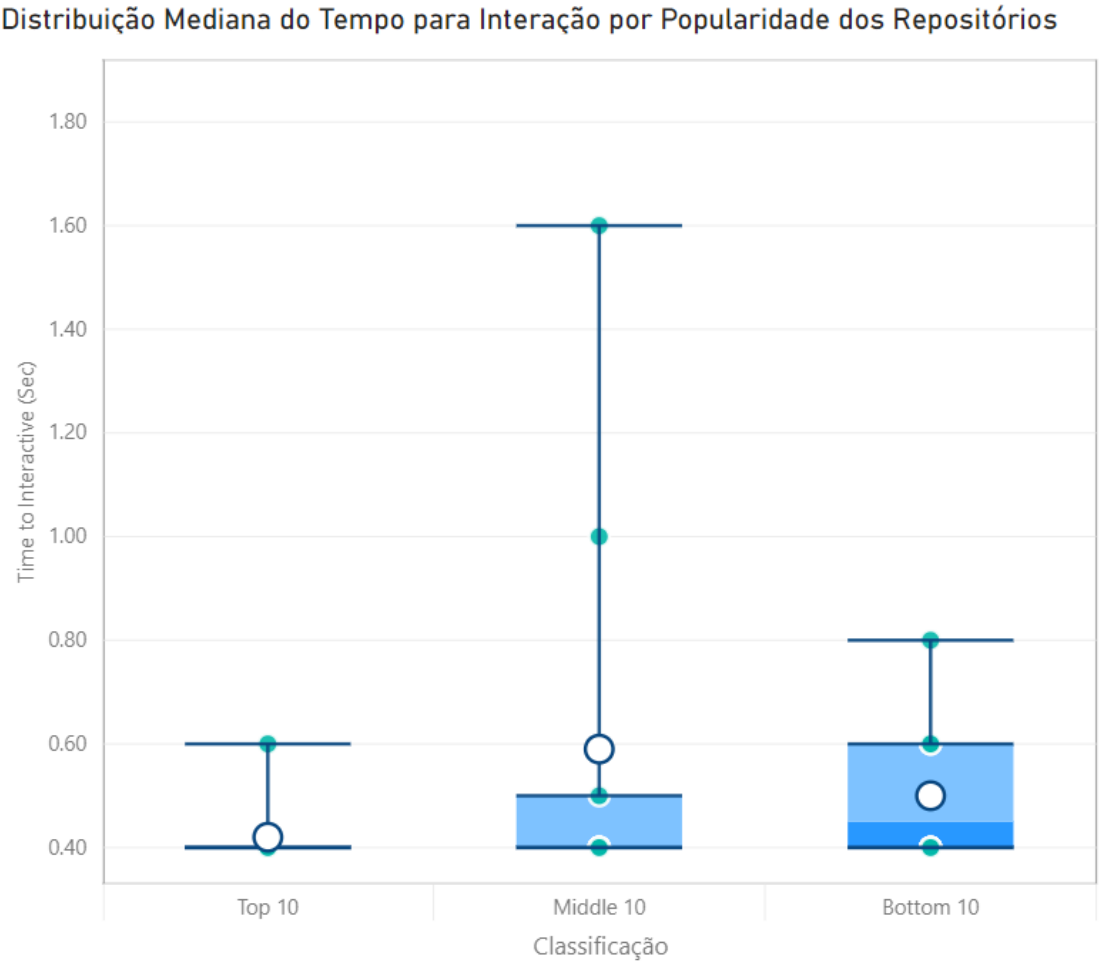

Figure 12. Distribuição mediana do tempo de interação por popularidade dos repositórios

Por meio da análise do gráfico acima representado pela Figura 12 e com auxílio de informações complementares da ferramenta, destaca-se o tempo mediano obtido para os top 10, middle 10 e bottom 10 repositórios, respectivamente, de 0,4, 0,4 e 0,45 segundos. Por outro lado, ao compararmos a média de cada grupo, obtivemos o resultado de 0,42, 0,59 e 0,50 segundos. Apesar de tais valores se mostrarem variados, ainda estão muito próximos (da mesma forma que ocorreu com a primeira pintura da tela), o que não necessariamente justificaria a adoção de determinado grupo de bibliotecas ou não por si só.

De outro modo, ao analisarmos ambas as métricas em conjunto, podemos perceber que os repositórios intermediários possuem a menor performance entre os demais, seguidos dos repositórios menos populares. Em geral, os repositórios mais populares se mostraram ter uma melhor performance, mesmo que a diferença entre o tempo contabilizado nas métricas analisadas sejam ainda menores que 1 segundo de forma genérica, o que para os usuários possa não ser perceptível de fato.

5. Conclusão (seção antiga)

A indústria de software também adota novas abordagens com a mudança de tecnologia e técnicas. A metodologia ágil é um dos métodos que auxiliam no sucesso de qualquer software. [Hayat et al. 2019]. Dessa forma, acredita-se que as bibliotecas de UI recebem atualizações com frequência (assumindo que esse intervalo de tempo é comum em times que atuam com metodologias ágeis, comuns no mercado de software).

Logo, assumimos que a constante atualização dessas bibliotecas dificultam manter o conhecimento relacionado à ela atualizado, sendo um fator que pode influenciar em sua utilização em algum projeto.

Sendo assim após toda a operação de recuperar os repositórios de bibliotecas relacionadas a UI do Github, buscar os repositórios criados em 2020 e filtrar todos estes dados entre os que utilizam alguma das bibliotecas e os que não utilizam de nenhuma biblioteca teremos dados suficientes para definir se o uso de bibliotecas UI se tornou uma característica relevante durante as criações de novos projetos.

6. Referências

Referências bibliográficas utilizadas pelo projeto: [Daniel et al. 2007], [Wikipedia], [Desolda et al. 2017], [Balland et al. 2013], [Org], [Inc. 2020], [Pfeiffer et al. 2016], [Krug 2014], [W. Wang 2020], [Sánchez et al. 2020], [Alafaireet 2006], [Badre and Jacobs 1999], [Khan et al. 2011], [Vinnakota 2016], [Erdős 2019], [Hayat et al. 2019]

References

Alafaireet, P. (2006). Graphic user interface: Needed design characteristics for successful physician use. In *2006 ITI 4th International Conference on Information Communications Technology*, pages 1–1.

Badre, A. and Jacobs, A. (1999). Usability, aesthetics, and efficiency: an evaluation in a multimedia environment. In *Proceedings IEEE International Conference on Multimedia Computing and Systems*, volume 1, pages 103–106 vol.1.

Balland, E., Consel, C., N'Kaoua, B., and Sauzéon, H. (2013). A case for human-driven software development. In *2013 35th International Conference on Software Engineering (ICSE)*, pages 1229–1232.

Daniel, F., Yu, J., Benatallah, B., Casati, F., Matera, M., and Saint-Paul, R. (2007). Understanding ui integration: A survey of problems, technologies, and opportunities. *IEEE Internet Computing*, 11(3):59–66.

Desolda, G., Ardito, C., Costabile, M. F., and Matera, M. (2017). End-user composition of interactive applications through actionable ui components. *Journal of Visual Languages Computing*, 42:46–59.

Erdős, F. (2019). Economical aspects of ux design and development. In *2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, pages 211–214.

Github. react-components.

Hayat, F., Rehman, A. U., Arif, K. S., Wahab, K., and Abbas, M. (2019). The influence of agile methodology (scrum) on software project management. In *2019 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pages 145–149.

Inc., G. (2020). The state of the octoverse.

Khan, M., Sulaiman, S., Said, A. M., and Tahir, M. (2011). Classification of usability issues for haptic systems. In *2011 7th International Conference on Emerging Technologies*, pages 1–4.

Krug, S. (2014). *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability*. New Riders Publishing;.

López de la Mora, F. and Nadi, S. (2018). Which library should i use?: A metric-based comparison of software libraries. In *2018 IEEE/ACM 40th International Conference on Software Engineering: New Ideas and Emerging Technologies Results (ICSE-NIER)*, pages 37–40.

Org, M. Material ui.

Pfeiffer, T., Hellmers, J., Schön, E., and Thomaschewski, J. (2016). Empowering user interfaces for industrie 4.0. *Proceedings of the IEEE*, 104(5):986–996.

Sánchez, V. R., Ayuso, P. N., Galindo, J. A., and Benavides, D. (2020). Open source adoption factors—a systematic literature review. *IEEE Access*, 8:94594–94609.

Vinnakota, T. (2016). A conceptual framework for complex system design and design management. In *2016 Annual IEEE Systems Conference (SysCon)*, pages 1–6.

W. Wang, J. Cheng, J. L. C. G. (2020). How do open source software contributors perceive and address usability? valued factors, practices, and challenges. *IEEE Software*, pages 0–0.

Wikipedia, a. e. l. Component-based software engineering.