Comparação de bibliotecas de Componentes: Definindo parâmetros de comparação

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

Abstract. *This article aims to characterize Javascript component libraries in order to define relevant parameters to compare them. This comparison has the goal to define whether there is a relevant difference between the most popular and the least popular libraries. This, it is expected that at the end of the article we will be able to know if using a more popular library brings some advantages to the development of the application over using a not so popular one.*

Resumo. *Este artigo tem como objetivo caracterizar bibliotecas de componentes Javascript a fim de definir parâmetros relevantes para compara-las. Essa comparação tem como objetivo definir se existe uma diferença relevante entre as bibliotecas mais populares e as menos populares. Dessa forma espera-se que ao final do artigo possamos saber se utilizar de uma biblioteca mais popular traz alguma vantagem para o desenvolvimento da aplicação em relação ao uso de uma menos popular.*

1. Introdução

As interfaces visuais no mundo digital se tornam amplamente necessárias, especialmente para possibilitar que usuários completem das mais diversas tarefas em produtos digitais. Nos últimos anos se tornou fundamental definir a maneira como um usuário interage com uma aplicação. Dessa forma designers e desenvolvedores vem se preocupando cada vez mais com o design de interação e a arquitetura de informações, ou seja, como a aplicação reage a interações e como mostrar as informações relevantes ao usuário [Krug 2014]. 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[1]. 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

[1]Site de armazenamento repositórios de código aberto e privado

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 *react-components*[2], 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*[3], 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?*
2. *Com que frequência a biblioteca recebe atualizações?*
3. *A escolha de uma biblioteca mais popular tem um grande impacto no desempenho da aplicação?*

Para coleta das métricas serão utilizadas três ferramentas. A API do Github será utilizada para recolhermos as bibliotecas e suas estrelas, assim como suas *issues*, *pull requests* e *releases*. A API do StackExchange será utilizada para recolhermos as perguntas relacionadas a cada biblioteca e a ferramenta Lighthouse será utilizada para recolhermos as métricas de performance.

Uma visão inicial dos resultados mostra que de fato existe uma diferença grande entre os repositórios mais e menos populares. Percebe-se que conforme eles se tornam mais populares eles passam a ter uma comunidade mais ativa, possuindo mais perguntas e essas sendo mais respondidas. Além disso é perceptível que a variação em performance entre os Top 10 repositório mais populares é muito menor que os demais, mostrando que eles se tornam mais estáveis conforme se tornam mais populares.

O restante deste artigo está organizado da seguinte forma: A Seção 2 tem como objetivo mostrar outros trabalhos que tenham tratado de assuntos relacionados, comparando-os com nossa solução e mostrando as diferenças entre eles. A Seção 3 mostra qual o processo utilizado para realização dos experimentos. A Seção 4 mostra quais valores obtivemos para as métricas propostas a fim de responder as perguntas já citadas. Na Seção 5 discutimos os resultados obtidos com nossas hipóteses enquanto na Seção 6 discutimos possíveis ameaças à validade de nosso experimento. Por último a Seção 7 traz as conclusões obtidas durante o projeto, relacionando os resultados obtidos com as respostas propostas.

[2]Tópico de componentes react existente no Github

[3]React JS é Uma biblioteca JavaScript para criar interfaces de usuário

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? [López de la Mora and Nadi 2018]. Neste trabalho, 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 [Desolda et al. 2017]. Neste trabalho, 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 [Balland et al. 2013]. Neste trabalho, 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.

How Do Open Source Software Contributors Perceive and Address Usability? Valued Factors, Practices, and Challenges [W. Wang 2020]. Ne trabalho, a questão sobre como a contribuição em repositórios de código aberto é abordada, sobretudo relacionando com o tema de usabilidade. O principal objetivo do estudo é entender como, e se ,a usabilidade é considerada no desenvolvimento de tais projetos. Em geral, a base teórica apresentada serviu de inspiração para a definição de objetivos deste trabalho, sendo assim sua principal contribuição.

Understanding UI Integration [Daniel et al. 2007]. Neste trabalho, 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.

Enfim, neste trabalho tentaremos apresentar métricas que visam facilitar a escolha de desenvolvedores quanto à adoção das bibliotecas de componentes por meio da comparação daquelas mais populares e as menos, além das que compreendem uma popularidade intermediária dentro dos repositórios analisados.

3. Metodologia

A Figura 1 descreve o fluxo de coleta das métricas utilizado. Nela são mostrados todos os passos tomados desde a coleta da biblioteca de interfaces até a escrita deste artigo.

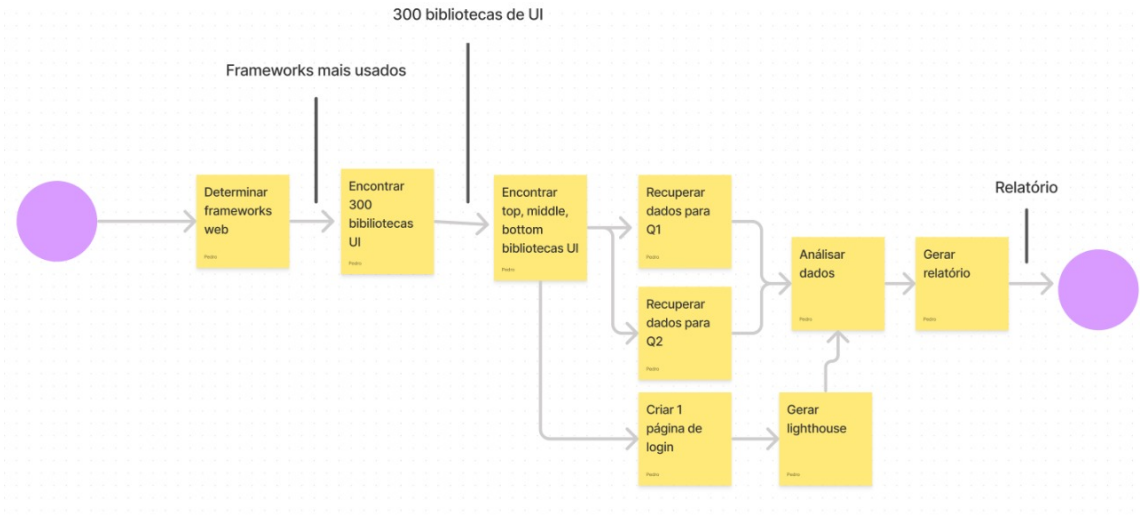

Figure 1. Desenho da metodologia

Tendo o problema e uma possível solução bem definidos se faz necessário a definição de uma metodologia a ser seguida a fim de chegar em resultados relevantes para a pesquisa, essa que pode ser vista em detalhes na Figura 1. Por conta disso a primeira ação a ser tomada foi descobrir o *framework* ou biblioteca, de desenvolvimento de interfaces por meio de componentes, mais popular da linguagem Javascript, essa sendo o ReactJS. Com isso, o próximo passo foi recuperar as 300 bibliotecas mais populares, classificadas com base no número de estrelas de seus repositórios. Por fim essa lista foi quebrada em 3, sendo que do primeiro grupo as 10 bibliotecas mais populares foram selecionadas, do segundo foram pegas as 10 primeiras bibliotecas e do terceiro as 10 menos populares, de forma que os 3 grupos serão referidos como os Top 10, Middle 10 e Bottom 10 bibliotecas, podendo ser vistas em mais detalhes nas Figuras 2, 3 e 4 respectivamente.

A Figura 2 mostra as 10 bibliotecas de componentes que foram classificadas como Top 10, seus nomes com dono e o número de estrelas de seus repositórios.

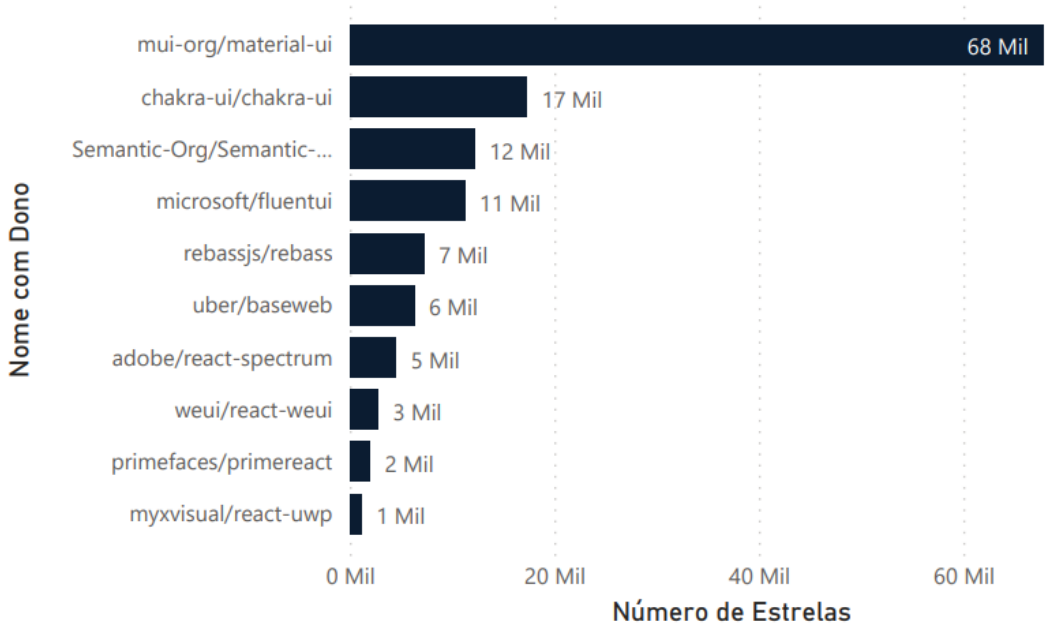

Figure 2. Imagem das bibliotecas classificadas como Top 10 com seus nomes e número de estrelas

A Figura 3 mostra as 10 bibliotecas de componentes que foram classificadas como Middle 10, seus nomes com dono e o número de estrelas de seus repositórios.

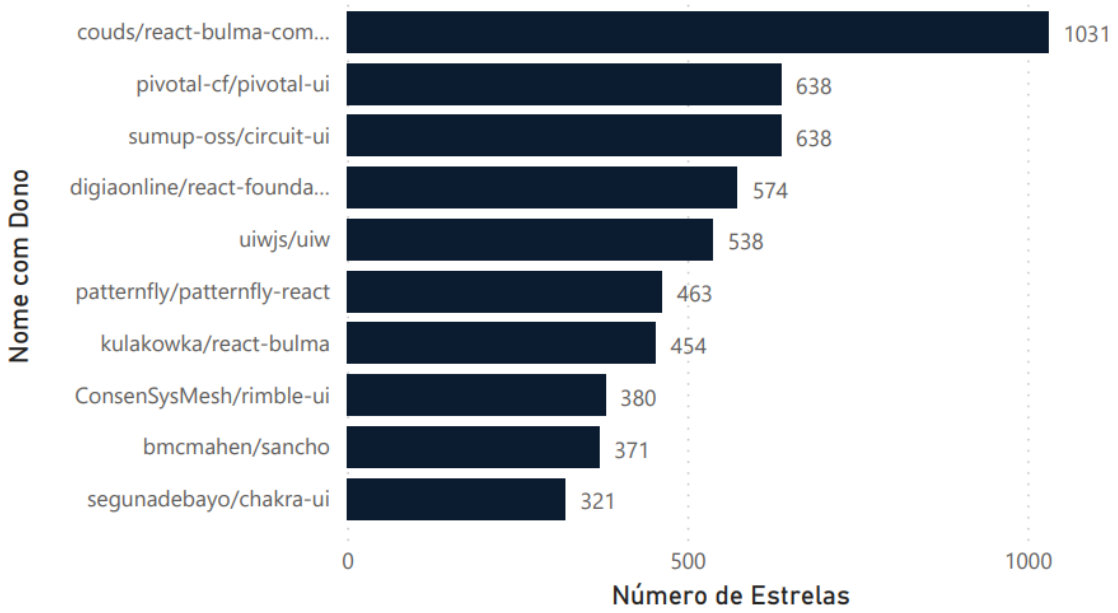

Figure 3. Imagem das bibliotecas classificadas como Middle 10 com seus nomes e número de estrelas

A Figura 4 mostra as 10 bibliotecas de componentes que foram classificadas como Bottom 10, seus nomes com dono e o número de estrelas de seus repositórios.

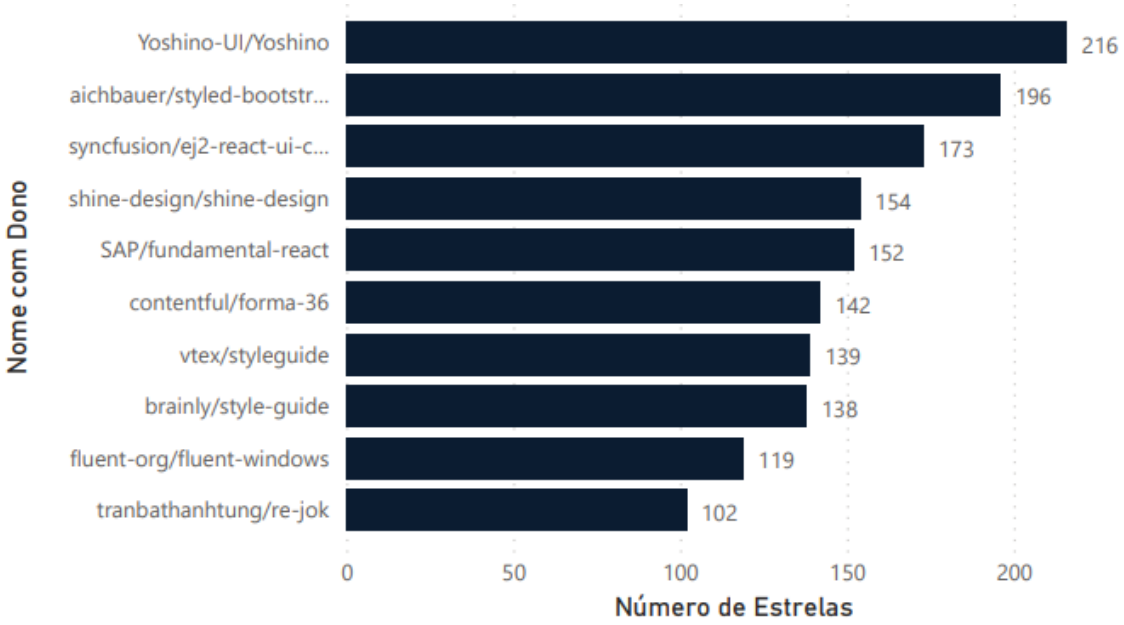

Figure 4. Imagem das bibliotecas classificadas como Bottom 10 com seus nomes e número de estrelas

3.1. Questão 1 - Qual a probabilidade de você ser respondido quando pergunta algo sobre a biblioteca para a comunidade?

Primeiramente, assumimos que perguntas realizadas no site StackOverflow associadas à tais bibliotecas possuem representam uma grande porcentagem do total de perguntas feitas às bibliotecas. 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. Com isso será possível identificar a probabilidade de você ser respondido quando existir alguma duvida sobre a biblioteca

3.1.1. Hipótese Nula, Alternativas, Variáveis dependentes e independentes

Hipóteses:

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.

Alternativa: Acredita-se a popularidade das bibliotecas de componentes não interfere na taxa de respostas dentro do StackOverflow e no número de *closed issues*, com mais de um participante, ao compara-las com as demais biblioteca.

Variáveis:

Dependentes: Percentual de perguntas respondidas no StackOverflow e número de *issues* fechadas com mais de um participante.

Independentes: Grupo de 30 bibliotecas componentes classificadas entre Top 10, Middle 10 e Bottom 10 de acordo com sua popularidade.

3.2. Questão 2 - Com que frequência a biblioteca recebe atualizações?

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 a métrica 1.

A fim de entender e caracterizar a frequência que as bibliotecas de componentes 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, Alternativas, Variáveis dependentes e independentes

Hipóteses:

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

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

Variáveis:

Dependentes: Índice de atualizações feita por cada comunidade de biblioteca UI

Independentes: Grupo de 30 bibliotecas UI separadas entre Top 10, Middle 10 e Bottom 10 de acordo com sua popularidade.

3.3. Questão 3 - A escolha de uma biblioteca mais popular tem um grande impacto no desempenho da aplicação?

Com as bibliotecas a serem analisadas definidas será desenvolvida uma tela de login, essa que pode ser vista na Figura 5, com cada biblioteca selecionada, buscando

entender o nível de performance de cada uma dentro de sua classificação. Utilizando a ferramenta Google Chrome Lighthouse[4], será possível calcular o desempenho de cada tela a partir de métricas definidas por ele como:

1. *First Contentful Paint*, que tem como objetivo calcular o tempo necessário para que uma aplicação web tenha sido completamente pintada na tela do usuário.
2. *Time to Interactive* que mostra o tempo necessário para que uma aplicação Web possa receber interações do usuário.

A Figure 5 mostra a tela uma tela de login simples, para cada biblioteca de componentes uma versão dessa tela foi desenvolvida, a fim de recuperar as métricas de *First Contentful Paint* e *Time to Interactive*.

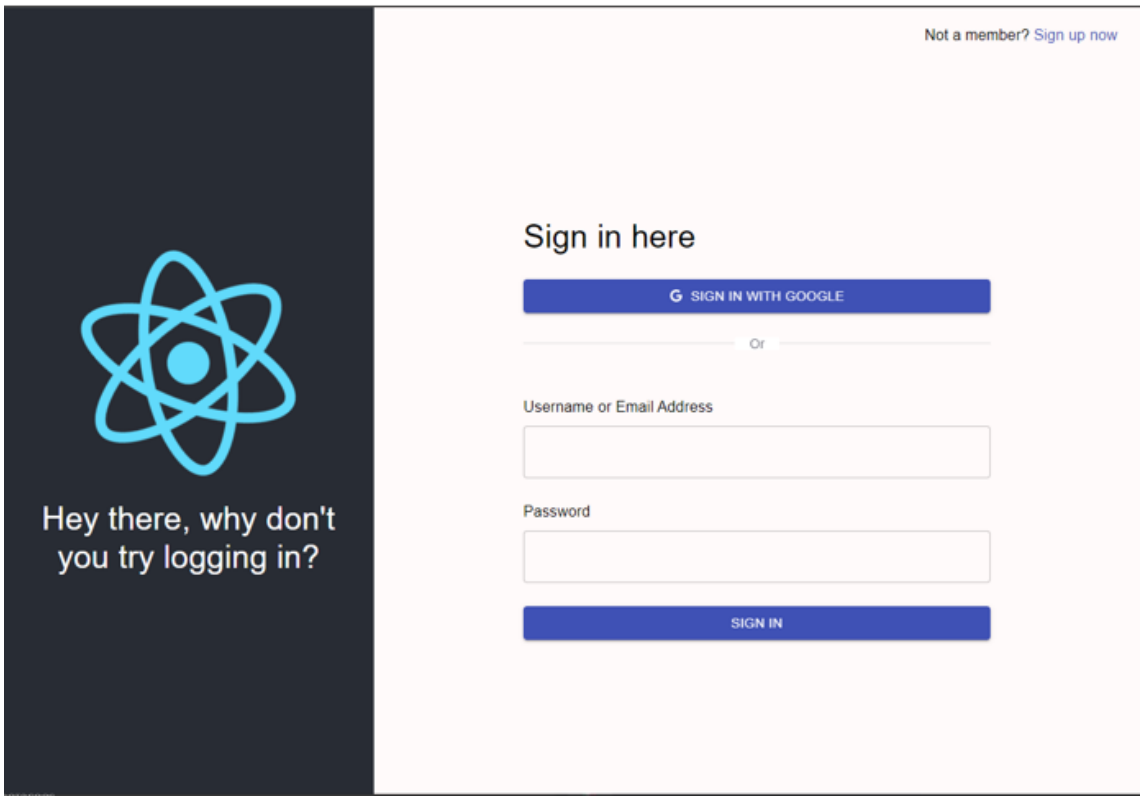

Figure 5. Tela de login desenvolvida utilizando cada biblioteca de componentes.

3.3.1. Hipótese Nula, Alternativas, Variáveis dependentes e independentes

Hipóteses:

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

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

Variáveis:

[4]Projeto *open-source* do Google Chrome que informa a qualidade do site web

Dependentes: Os indices de *First Contentful Paint* e *Time to Interactive* de cada biblioteca

Independentes: Grupo de 30 bibliotecas de componentes 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 detalhada na Seção 3. Os valores foram organizados de forma a ficarem juntos das perguntas que os definiram como métricas.

4.1. Questão 1 - Qual a probabilidade de você ser respondido quando pergunta algo sobre a biblioteca para a comunidade?

Issues e questões respondidos dentro da comunidade são fatores extremamente relevantes no momento da escolha de uma biblioteca de componentes, visto que, em diversas situações o desenvolvedor se depara com problemas que ele não está apto a resolver sozinho, tornando necessário recorrer á 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* na Seção 3 buscamos 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 componentes, gerando os seguintes resultados.

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

A Figura 6 mostra a distribuição mediana da relação entre o número *issues* fechadas com participação de mais de uma pessoa e o número total de *issues*, classificadas pela popularidade da biblioteca pelas classificação de Top, Middle e Bottom 10.

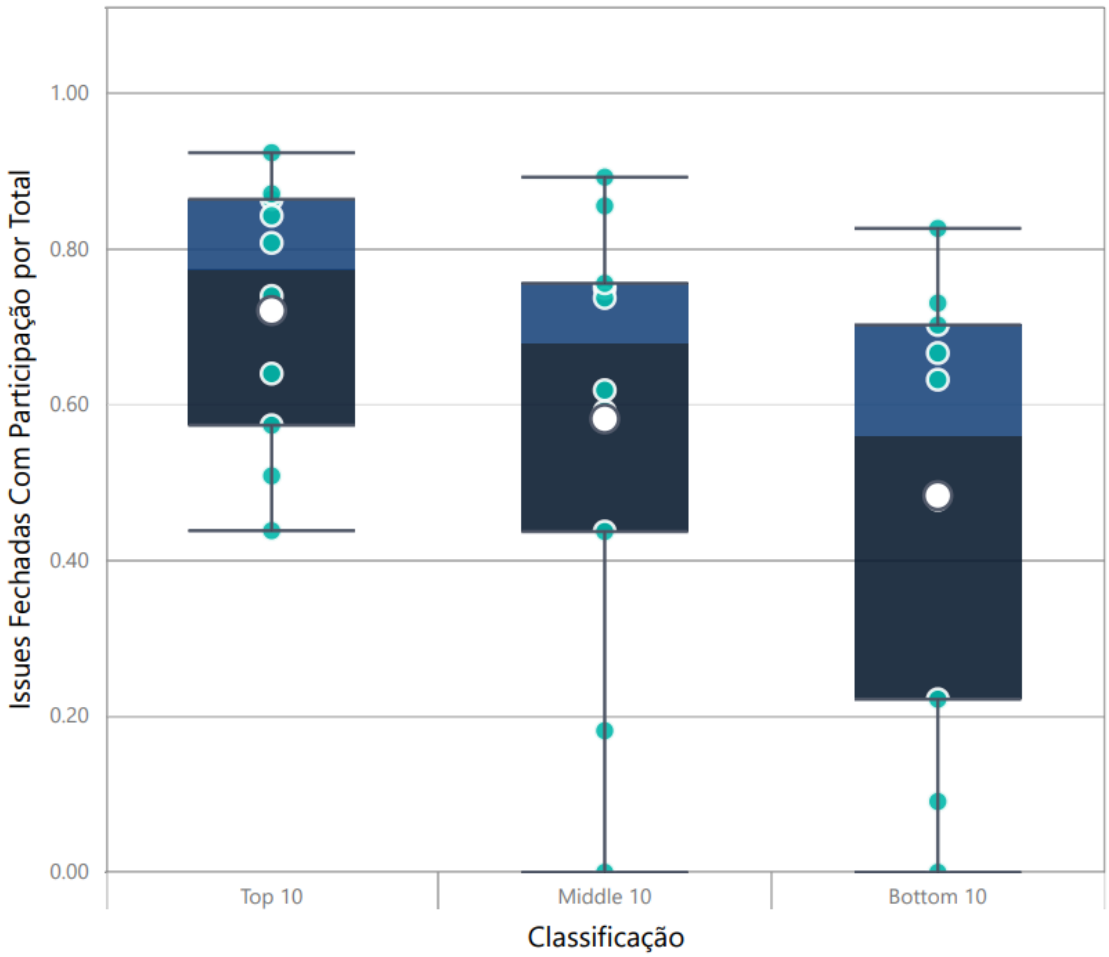

Figure 6. Distribuição Mediana de issues Fechadas Com Participação por Total de issues Classificadas Pela Popularidade

A partir da Figura 6 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. Entretanto podemos perceber que 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 enquanto o Top 10 teve 44%.

Também é possível perceber um crescimento da mediana das taxas de *closed issue por total issue* onde o 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, que tem a menor mediana de *closed issues* com mais de um participante, obtendo apenas 48%.

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

A Figura 7 mostra a distribuição mediana da relação entre o número perguntas respondidas e o número total de perguntas, classificadas pela popularidade da biblioteca pelas classificação de Top, Middle e Bottom 10.

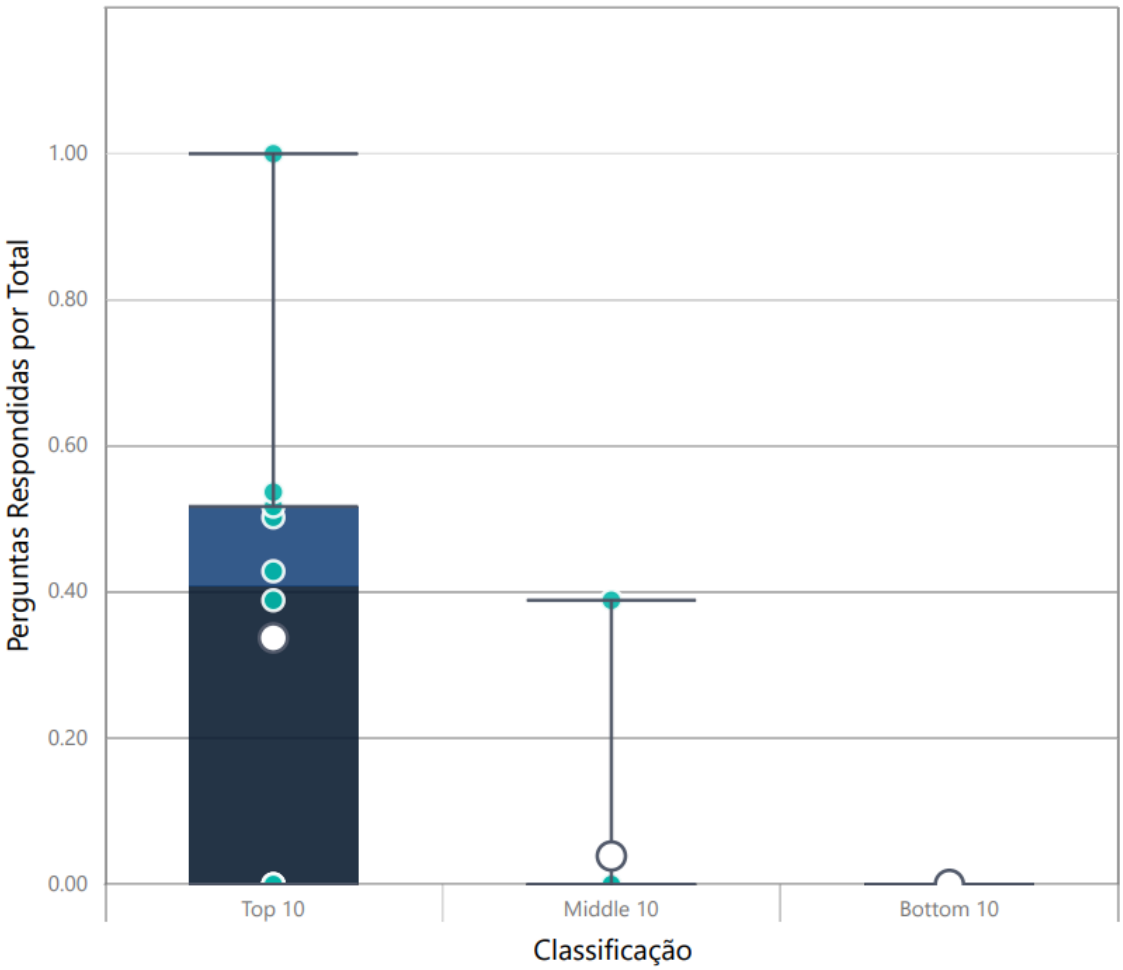

Figure 7. Distribuição Mediana de Perguntas Respondidas pelo Total de Perguntas Classificadas Pela Popularidade

Ao analisar o gráfico referente a Figura 7 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* enquanto o Bottom 10 não teve perguntas.

4.2. Questão 2 - Com que frequência a biblioteca recebe atualizações?

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*.

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

A partir da análise dos dados, do gráfico representado na Figura 8 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 ao analisarmos metade dos repositórios de cada grupo, ela é muito mais significante e demonstra a diferença entre tais grupos.

A Figura 8 mostra a distribuição mediana do número de *releases* por ano de idade completo da biblioteca, classificadas pela popularidade da biblioteca pelas classificação de Top, Middle e Bottom 10.

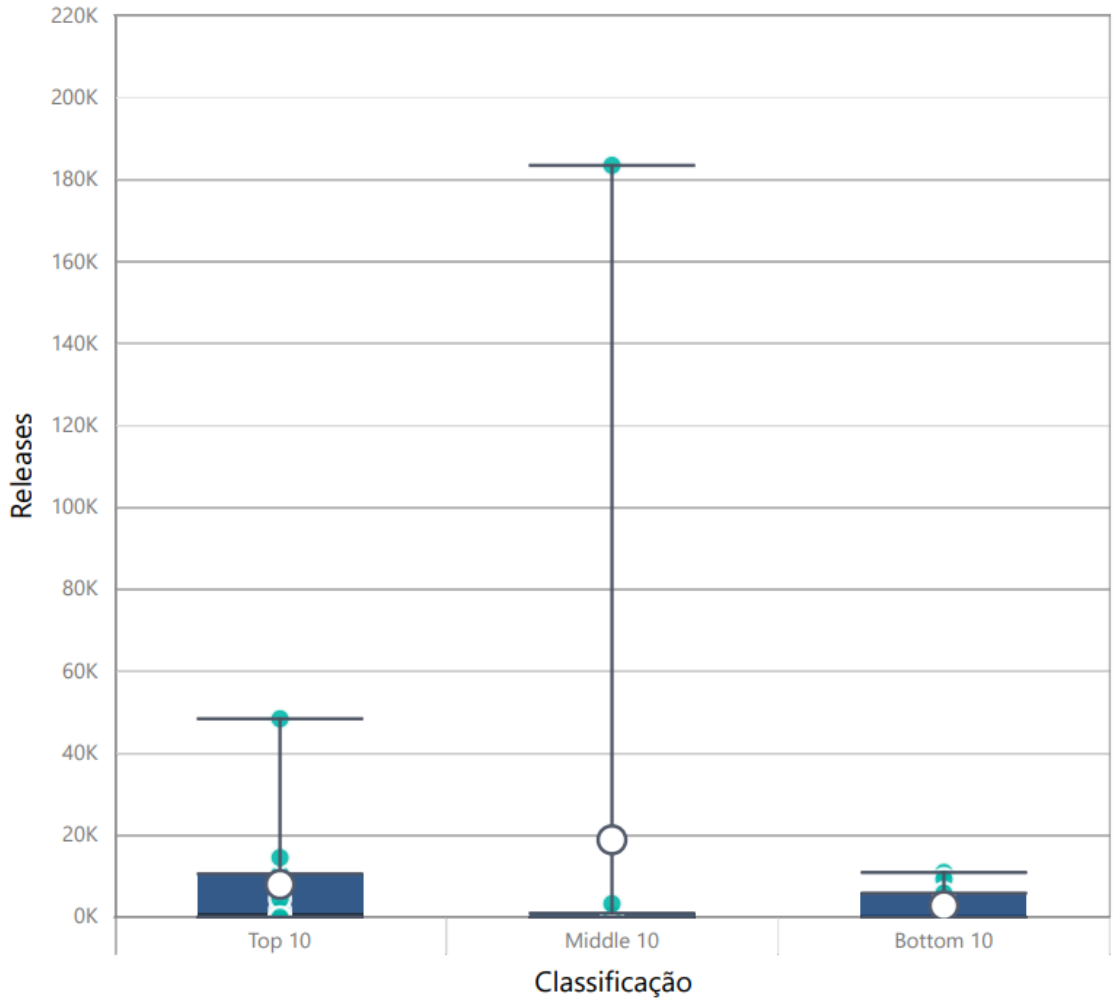

Figure 8. Distribuição mediana de releases separados por popularidade dos repositórios

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

A partir da análise dos dados, do gráfico representado na Figure 9 e com auxílio de informações complementares da ferramenta, podem ser destacados os seguintes resultados:

1. Resultados do tempo mediano de fechamento de um PR de cada grupo

    (a) Os repositórios mais populares apresentaram o tempo mediano de 6 dias completos para o fechamento de um *pull request*, sendo o maior entre os grupos;

    (b) Os repositórios intermediários apresentaram o tempo mediano de 3 dias completos para o fechamento de um *pull request*;

    (c) Os repositórios menos populares apresentaram o tempo mediano de 2 dias completos para o fechamento de um *pull request*.

2. Resultados do interquartil de cada grupo

    (a) O interquartil dos Top 10 repositórios apresentaram o valor de 7 dias completos para o fechamento de um *pull request*, sendo o maior entre os grupos;

    (b) O interquartil dos repositórios intermediários apresentaram o valor de 5 dias completos para o fechamento de um *pull request*;

    (c) O interquartil dos repositórios menos populares apresentaram o valor de 1 dia completo para o fechamento de um *pull request*.

Com base nos resultados apontados acima, percebemos que o tempo mediano de fechamento de um *pull request* segue uma proporção direta com a popularidade do grupo de repositórios. Isso quer dizer que os repositórios mais populares demoram mais para fechar um PR, enquanto os menos populares fecham mais rapidamente entre os grupos. Em relação aos repositórios intermediários, o tempo de fechamento também é intermediário entre os grupos comparados.

Apesar do tempo mediano de fechamento seguir uma proporção direta, os resultados do interquartil de cada grupo apontam que a diferença entre os repositórios menos populares e aqueles intermediários é maior do que aparenta. Isso se dá porque o interquartil representa 50% dos valores analisados, distribuídos ao meio de cada coluna do gráfico. Enquanto o tempo mediano é muito próximo entre tais grupos, vale destacar que os repositórios intermediários apresentaram um tempo maior de fechamento de 5 dias completos, enquanto os repositórios menos populares apresentaram o tempo de 1 dia completo, o que reforça essa diferença.

A Figura 9 mostra a distribuição mediana do tempo médio para fechamento de *pull requests* em dias, classificadas pela popularidade da biblioteca pelas classificação de Top, Middle e Bottom 10.

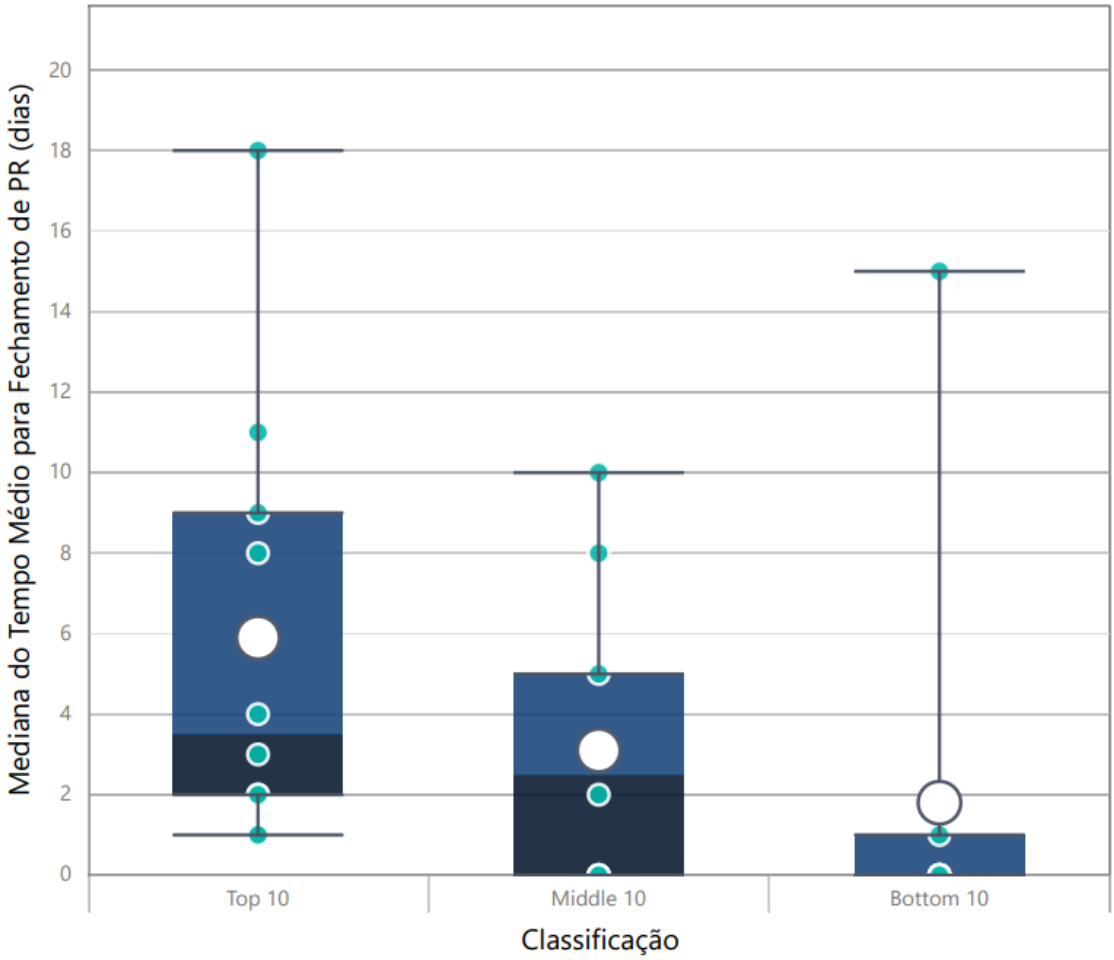

Figure 9. Distribuição do tempo mediano de aprovação de pull requests dos Top 10 repositórios

4.3. Questão 3 - A escolha de uma biblioteca mais popular tem um grande impacto no desempenho da aplicação?

Após o desenvolvimento de tela de login com as bibliotecas selecionadas, utilizamos a ferramente do Google Chrome Lighthouse a fim de obter valores para as métricas de *Time to Interactive* e *First Contentful Paint*, obtivemos os seguintes resultados.

4.3.1. *First Contentful Paint*

Por meio da análise do gráfico representado pela Figura 10 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. O maior valor foi apresentado pelo grupo dos Middle 10, sendo que os 3 conjuntos de bibliotecas apresentam um valor mínimo de 0,4 segundos.

A Figura 10 mostra a distribuição mediana do *First Contentful Paint*, classificadas pela popularidade da biblioteca pelas classificação de Top, Middle e Bottom 10.

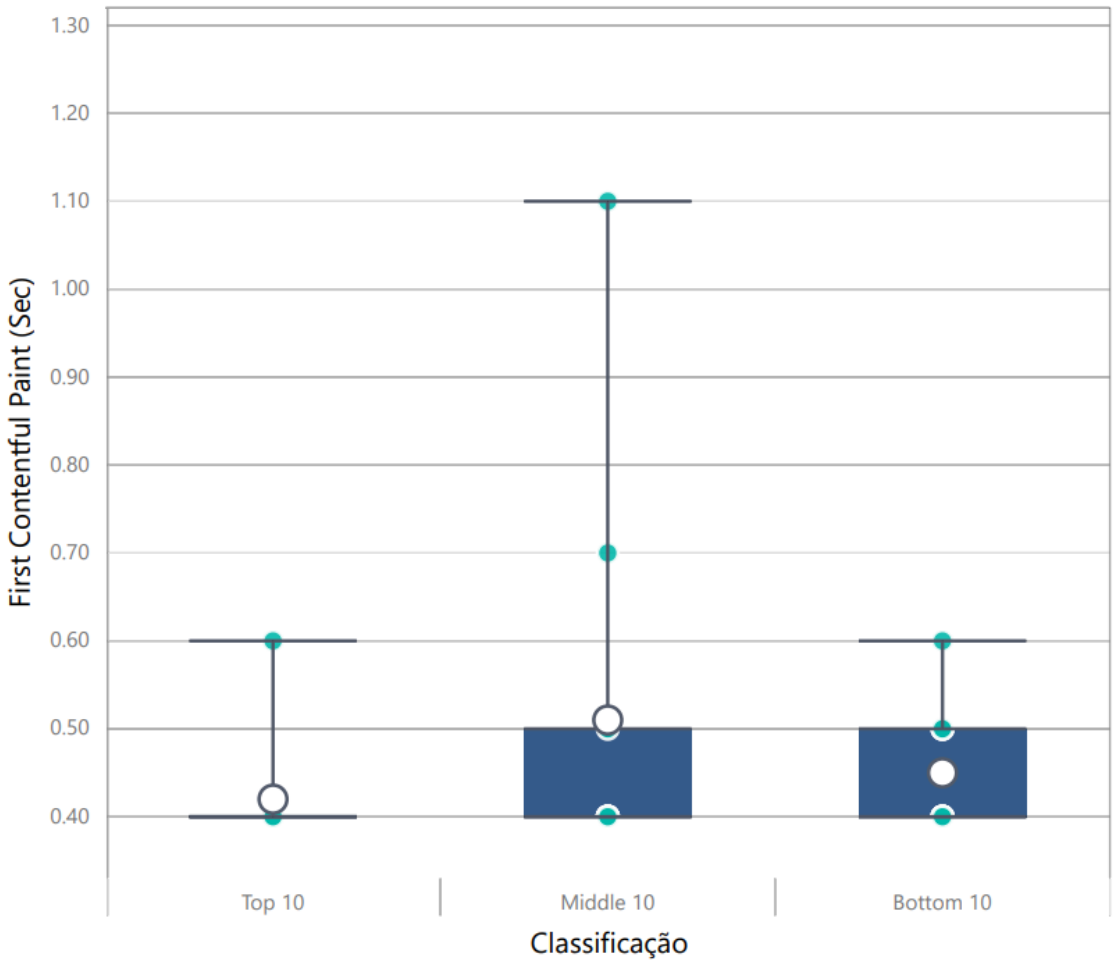

Figure 10. Distribuição mediana da primeira pintura por popularidade dos repositórios

4.3.2. *Time to Interactive*

Por meio da análise do gráfico representado pela Figura 11 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 apenas.

A Figura 11 mostra a distribuição mediana do *Time to interactive*, classificadas pela popularidade da biblioteca pelas classificação de Top, Middle e Bottom 10.

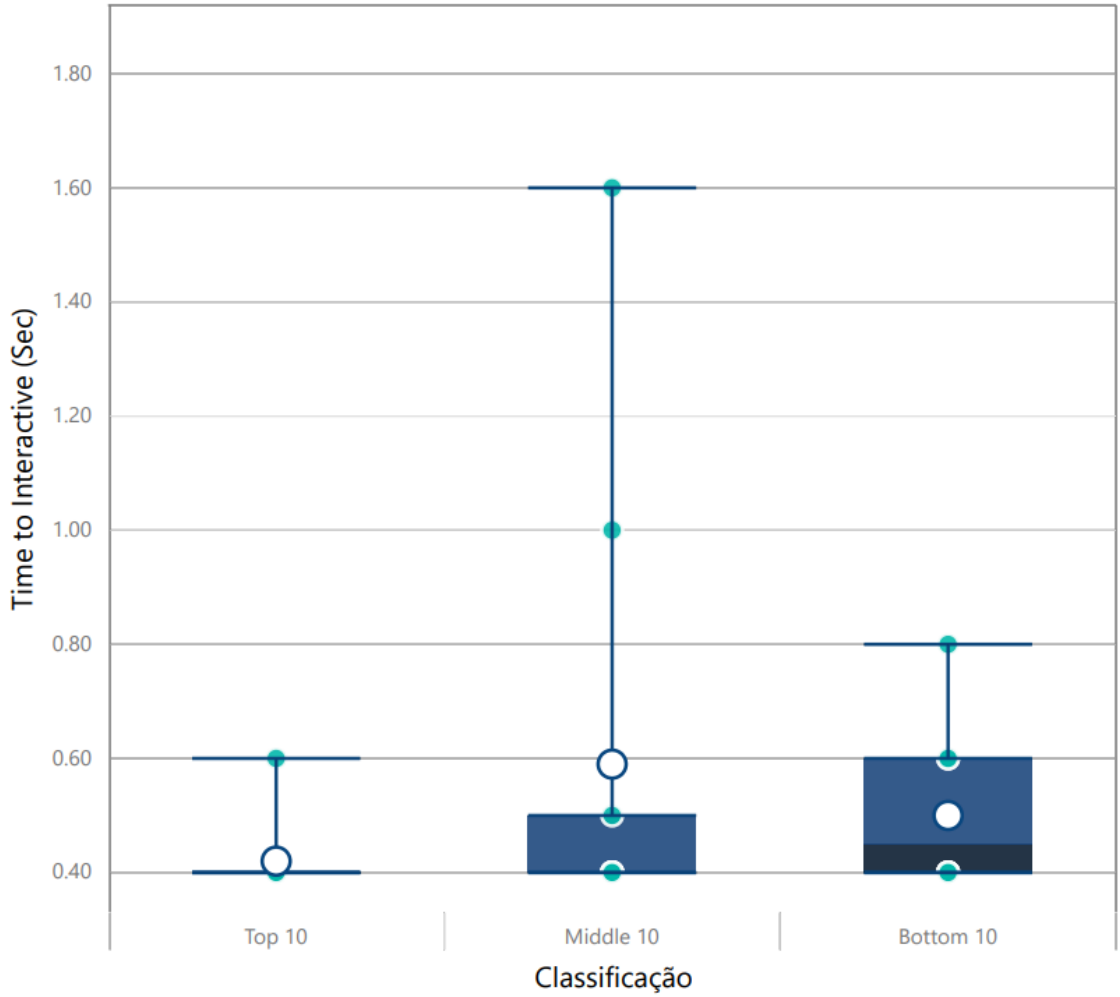

Figure 11. Distribuição mediana do tempo para iteravel por popularidade dos repositórios

5. Discussão

Se analisarmos a frequência com que dúvidas são respondidas, as Top 10 bibliotecas obtiveram resultados muito maiores, não só de respostas como também de perguntas. Isso já era de se esperar, já que devido ao aumento de popularidade essas bibliotecas mais pessoas passam a ter interesse sobre ela, de forma que mais perguntas surgem, e por consequência, mais pessoas buscam responder para ajudar a biblioteca a crescer. Isso mostra que analisando de um ponto de vista de suporte da comunidade, as bibliotecas mais populares tem de fato uma grande vantagem, já que por mais pessoas estarem utilizando-as, mais pessoas podem responder dúvidas sobre ela.

Quando analisamos a frequência de atualização das bibliotecas as Top 10 também tem uma vantagem, recebendo um maior número de versões ao longo de um mesmo período de tempo, de forma que podemos perceber isso tanto quando olhamos para o número de *releases* quanto quando analisamos o tempo médio para fechamento de *pull requests*. Entretanto ao comprarmos os Middle 10 e Bottom 10 quanto ao número de *releases* percebemos que por mais que os Middle possuam o maior número de *releases* a mediana de *releases* dos Bottom 10 é maior. Acredita-se que isso ocorre por os Bottom

10 terem um maior número de mudanças a serem feitas inicialmente, com o objetivo de atrair mais contribuidores.

Por último temos os parâmetros de performance relacionados às bibliotecas. Observando essas métricas fica claro a superioridade das Top 10 sobre as demais nesse quesito. A mediana de *Time to Interactive* e *First Contentful Paint* do Top 10 foram consideravelmente menores que ambas as Bottom e Middle, o que mostra q com o aumento da popularidade as bibliotecas se tornam mais performáticas. Contudo, é perceptível também que a mediana dos Bottom 10 consegue passar a dos Middle na métrica de *Time to Interactive* e empatar com a mediana de *First Contentful Paint*, isso pode ser uma consequência de elas serem bibliotecas ainda pequenas, que não possuem tantas funcionalidades e que por conta disso não tornam a aplicação que utiliza elas tão pesada.

6. Ameaças à Validade

Construção

- As métricas possuem relação temporal?
- As métricas coletadas são suficientes para comparação das bibliotecas?

As métricas buscadas durante esse experimento foram recuperadas para o ano de 2020. Não sendo possível se certificar que os resultados seriam os mesmos para os demais anos. Da mesma forma não é possível afirmar que as métricas escolhidas são suficientes para que a comparação dos repositórios seja de fato útil.

Conclusão

- As métricas coletadas traduzem fielmente o suporte de cada biblioteca?
- As métricas de performance são de fato relevantes na escolha de uma biblioteca?

O suporte à uma biblioteca pode ser determinado também pela qualidade da documentação dela, podendo ser interessante analisar essa métrica em um projeto futuro. Dessa mesma forma existe a dúvida se as métricas de performance são tão interessantes aos desenvolvedores na hora de fazer a escolhe de uma biblioteca de componentes.

Interna

Não foram encontradas ameaças à validade interna visto que os repositórios buscados foram datados e caso existissem problemas de rede ou hardware a pesquisa pode ser feita novamente até o momento em que fossem buscados todos os dados necessários.

Externa

- Resultados obtidos permitem generalização para outros experimentos?

Os resultados deste experimento são referentes a bibliotecas de componentes React que são *open source* e estão dispostas no Github. Dessa forma não é possível dizer que os resultados deste experimento refletem a realidade para bibliotecas de Vue ou Angular[5] por exemplo.

7. Conclusão

O principal objetivo desta pesquisa era buscar parâmetros a fim de definir possíveis benefícios de se utilizar bibliotecas de componentes mais populares, assim como definir

[5] Ambas bibliotecas de desenvolvimento de interface por meio da utilização de componentes

se de fato as bibliotecas mais populares são melhores que as menos conhecidas. Dentro deste contexto foram analisadas *issues*, perguntas, *releases* e a performance de cada uma das bibliotecas de nosso *dataset* que era composto por 30 bibliotecas divididas em Top 10, Middle 10 e Bottom 10.

Após termos obtido todas as métricas definidas inicialmente para o experimento e analisá-las é possível perceber que as bibliotecas mais populares de fato possuem uma vantagem sobre as demais. Quando analisamos a frequência de respostas à perguntas das bibliotecas é possível ver claramente que conforme as bibliotecas se tornam mais populares elas recebem mais perguntas e consequentemente mais respostas, de forma que uma biblioteca classificada como Top 10 tinha um alto percentual de perguntas respondidas. Já ao analisarmos a frequência de atualização os resultados não foram tão diferentes, as bibliotecas mais populares possuem mais *releases* e fecham *pull requests* mais rapidamente, entretanto quando comparamos a mediana de *releases* entre os Middle 10 e Bottom 10 é possível ver que os Bottom lançam mais versões que os Middle. Por fim, quando olhamos para a performance novamente os Top 10 tem vantagem, sendo que os Middle e Bottom possuem valores bem semelhantes.

Por fim é possível imaginar trabalhos futuros que envolvessem o estudo das documentações dessas bibliotecas, essas que são cruciais para utilização das bibliotecas. Também poderia ser interessante desenvolver uma pesquisa com foco na qualidade dos componentes apresentados e na adequação deles para com o *design guideline*[6] proposto pela biblioteca.

Referências

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.

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.

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.

[6]Padrões de design definidos a fim de criar interfaces padronizadas. Exemplos são o Material Design da Google e o Fluent Design da Microsoft.