UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

JOSÉ DAVI DE ARAUJO SOUZA

GUIA PARA ESTILIZAÇÃO DE COMPONENTES REUTILIZÁVEIS EM APLICAÇÕES ANGULAR

CAMPINA GRANDE - PB

2023

JOSÉ DAVI DE ARAUJO SOUZA

GUIA PARA ESTILIZAÇÃO DE COMPONENTES REUTILIZÁVEIS EM APLICAÇÕES ANGULAR

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador : Adalberto Cajueiro de Farias

CAMPINA GRANDE - PB

2023

JOSÉ DAVI DE ARAUJO SOUZA

GUIA PARA ESTILIZAÇÃO DE COMPONENTES REUTILIZÁVEIS EM APLICAÇÕES ANGULAR

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Adalberto Cajueiro de Farias

Orientador – UASC/CEEI/UFCG

Dalton Dario Serey Guerrero

Examinador – UASC/CEEI/UFCG

Francisco Vilar Brasileiro

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 28 de junho de 2023.

CAMPINA GRANDE - PB

RESUMO

Dentro da carreira de desenvolvedor web, especializar-se em front-end é comum. Nessa área é frequente encontrar problemas semelhantes que exigem soluções similares, como interfaces de navegação, áreas de busca e formulários. Para lidar com essas situações existem diversos recursos prontos que possibilitam a reutilização de soluções, estabelecendo um padrão. No entanto, ao utilizar componentes reutilizáveis no desenvolvimento front-end com o framework Angular, surgem desafios na personalização dessas soluções para se adequarem a diferentes design. Este trabalho aborda os diversos tipos de componentes reutilizáveis da biblioteca Angular Material, além de explorar diferentes abordagens para estilizá-los no contexto do framework Angular. Foi criado um guia com exemplos práticos e dicas para auxiliar os desenvolvedores front-end na personalização dos componentes já existentes no Angular, superando as dificuldades encontradas. Com essa abordagem aprimorada, os desenvolvedores front-end poderão personalizar os componentes reutilizáveis no Angular de forma mais eficiente, atendendo aos requisitos de design de maneira mais flexível e intuitiva.

Guide to Styling Reusable Components in Angular Applications

ABSTRACT

Within the web developer career, specializing in front-end is common. In this area, it is common to find similar problems that require similar solutions, such as navigation interfaces, search areas and forms. To deal with these situations, there are several ready-made resources that allow the reuse of solutions, establishing a standard. However, when using reusable components in front-end development with the Angular framework, challenges arise in customizing these solutions to suit different designs. This work addresses the different types of reusable components from the Angular Material library, in addition to exploring different approaches to styling them in the context of the Angular framework. A guide was created with practical examples and tips to assist front-end developers in customizing existing components in Angular, overcoming the difficulties encountered. With this improved approach, front-end developers will be able to customize reusable components in Angular more efficiently, meeting design requirements in a more flexible and intuitive way.

Guia para Estilização de Componentes Reutilizáveis em Aplicações Angular

José Davi de Araujo Souza
Universidade Federal de Campina Grande
Campina Grande, Paraíba
jose.davi.souza@ccc.ufcg.edu.br

Adalberto Cajueiro de Farias
Universidade Federal de Campina Grande
Campina Grande, Paraíba
adalberto@computacao.ufcg.edu.br

RESUMO

Dentro da carreira de desenvolvedor web, especializar-se em *front-end* é comum. Nessa área é frequente encontrar problemas semelhantes que exigem soluções similares, como interfaces de navegação, áreas de busca e formulários. Para lidar com essas situações existem diversos recursos prontos que possibilitam a reutilização de soluções, estabelecendo um padrão. No entanto, ao utilizar componentes reutilizáveis no desenvolvimento front-end com o framework Angular, surgem desafios na personalização dessas soluções para se adequarem a diferentes design. Este trabalho aborda os diversos tipos de componentes reutilizáveis da biblioteca *Angular Material*, além de explorar diferentes abordagens para estilizá-los no contexto do framework Angular. Foi criado um guia com exemplos práticos e dicas para auxiliar os desenvolvedores front-end na personalização dos componentes já existentes no Angular, superando as dificuldades encontradas. Com essa abordagem aprimorada, os desenvolvedores front-end poderão personalizar os componentes reutilizáveis no Angular de forma mais eficiente, atendendo aos requisitos de design de maneira mais flexível e intuitiva.

Palavras-chave

Angular, CSS, HTML, estilização, componentes, reutilizáveis, frontend.

Repositório

https://github.com/JoseDavi/guia-tcc

https://guia-tcc.netlify.app/

https://codesandbox.io/p/github/JoseDavi/guia-tcc/

1. INTRODUÇÃO

O desenvolvimento web baseia-se na arquitetura cliente-servidor, na qual o *front-end* é responsável pela parte visual de um site ou aplicativo, incluindo o design, a interface de navegação e as ferramentas de interação com o usuário, como áreas de busca e formulários. Por outro lado, o *back-end* lida com a lógica de negócio e a comunicação com o banco de dados. Assim, os desenvolvedores *front-end* têm a responsabilidade de criar a experiência do usuário em uma aplicação web, desenvolvendo as páginas com as quais o usuário irá interagir [1].

Devido à importância dessa área e à influência da ideologia do *open source*, a existência de componentes e bibliotecas prontas tornou-se uma realidade fornecendo diversas funcionalidades para serem utilizadas em diferentes aplicações. Entre os vários frameworks front-end disponíveis, como Angular, React, Vue, entre outros, o Angular carece de conteúdo auxiliar para estilização visual desses componentes prontos, essa dificuldade foi percebida devido a uma necessidade encontrada em um projeto de pesquisa e desenvolvimento (P&D) do qual participamos, onde nos deparamos com os requisitos de seguir um design e utilizar um conjunto componentes prontos que não tinham a mesma aparência. Dessa forma nos levando a investigar como adaptá-los para o novo estilo e nos deparando com essa falta de material. Devido a essa realidade, optamos por abordar os diversos tipos de componentes reutilizáveis da biblioteca *Angular Material* [5], além de explorar diferentes abordagens para estilizá-los no contexto do framework Angular para criação de um guia prático que auxilie os desenvolvedores front-end que estiverem na mesma situação.

2. PROBLEMA E SOLUÇÃO

Na literatura, existem trabalhos publicados relacionados ao uso de HTML e CSS como o livro "Learn to Code HTML and CSS: Develop and Style Websites" de Shay Howe [3]. No contexto do Angular o livro "Pro Angular 9: Build Powerful and Dynamic Web Apps", de Adam Freeman [4], é um bom exemplo. Quanto à comparação de bibliotecas de componentes do Angular a dissertação de mestrado "Angular Component Library Comparison", de Michael Christopher Cebrian [2], é uma valiosa opção inicial na literatura. No entanto, há escassez de material didático quando se trata da estilização de componentes prontos no Angular, o que dificulta a adaptação do estilo desses componentes para um design específico.

Ao utilizar componentes prontos, diferentemente da criação de um componente próprio, os desenvolvedores desconhecem a estrutura HTML, as classes, os identificadores e as estilizações próprias e herdadas que compõem o componente utilizado. Isso requer a aplicação de um conjunto de medidas para descobrir essas informações, que são fundamentais para estilizar um componente. Portanto, observa-se uma dificuldade na customização de componentes prontos.

Diante da escassez de material didático no contexto do Angular, o objetivo deste trabalho é fornecer uma alternativa para suprir essa lacuna, especialmente para pessoas iniciantes no Framework. Para isso, foi criado um guia que apresenta alternativas para a estilização de componentes prontos, destacando uma sugestão de metodologia a ser seguida em que o foco está em ensinar como aplicar as customizações necessárias, ao invés de prescrever quais estilizações específicas devem ser adicionadas, uma vez que esse conhecimento é fundamentalmente relacionado a HTML e CSS.

O guia oferece uma base sólida com exemplos de personalização com diferentes níveis de complexidade a fim de superar os desafios encontrados na personalização visual desses componentes, capacitando os desenvolvedores *front-end* no processo de aplicação de diferentes designs de forma eficiente, além de melhorar a experiência do usuário nas aplicações desenvolvidas com Angular.

3. METODOLOGIA

Para o desenvolvimento deste trabalho, o processo foi dividido em três etapas: Análise de Componentes Prontos, Proposta de Metodologia para Estilização dos Componentes e a Criação do Guia.

3.1 Análise de Componentes Prontos

Após identificarmos e examinarmos os componentes disponíveis no *Angular Material*, escolhemos os componentes *button*, *datepicker*, *toggle button* e *chips* para ser o nosso material de estudo. Para isso, realizamos uma análise detalhada de cada componente, compreendendo a sua estrutura HTML, classes e estilizações herdadas. O objetivo era obter uma compreensão aprofundada da composição e dos pontos-chave de cada componente para prover exemplos simples com diferentes níveis de modificações para melhor entendimento dos leitores.

3.2 Proposta de Metodologia para Estilização dos Componentes

Com base na análise dos componentes prontos e no estudo das metodologias de estilização, propomos uma metodologia específica para a estilização de componentes reutilizáveis no Angular. Dessa forma, essa metodologia fornecerá diretrizes claras e boas práticas para personalizar componentes, levando em consideração a estrutura dos mesmos, o uso de classes e a aplicação de estilos.

O algoritmo criado consiste na utilização da ferramenta do desenvolvedor (*DevTools*) do navegador para compreensão da estrutura HTML do componente, analisando as propriedades como *className* e *ID*, além dos tipos de elementos que compõem a estrutura do componente.

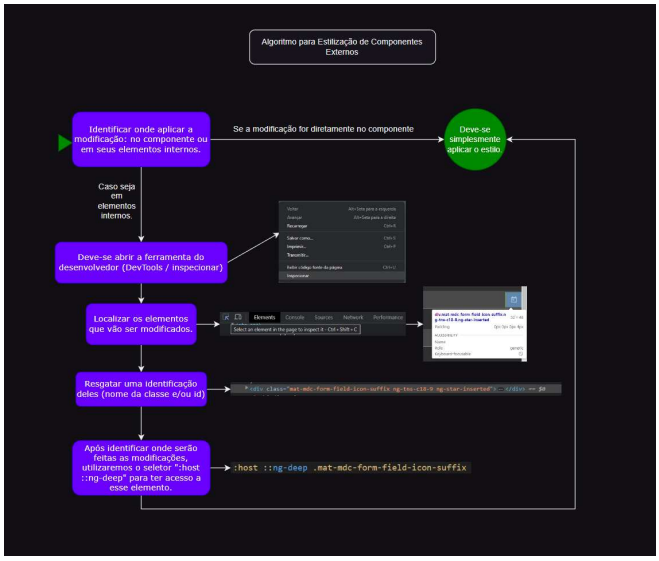


Figura 1: Algoritmo para estilização dos componentes.

3.3 Criação do Guia de Exemplos de Estilização de Componentes

Nesta etapa foi criado o guia de exemplos práticos de estilização de componentes reutilizáveis no Angular, utilizando a metodologia proposta. Esses exemplos abordaram diferentes níveis de complexidade, desde pequenos ajustes nas bordas até ocultação de ícones internos. Assim sendo, fornecemos códigos e explicações para auxiliar os desenvolvedores a implementarem as soluções propostas e serem capazes de aplicar este algoritmo aos seus casos específicos.

4. DESENVOLVIMENTO DO GUIA

O desenvolvimento do guia foi conduzido seguindo uma abordagem sistemática, que envolveu as seguintes etapas descritas abaixo.

4.1 Planejamento

Definirmos o objetivo do guia no qual foram estabelecidos quais ferramentas seriam utilizadas e qual a maneira de disponibilização.

Para definição do escopo, realizamos uma pesquisa exploratória inicial na biblioteca do *Angular Material* a fim de identificar os componentes que seriam utilizados, levando em conta a sua complexidade e possíveis alterações a serem aplicadas.

Como maneira de disponibilização foi escolhida a criação do repositório no *github* e a publicação no *netlify* para haver uma visualização simplificada, sem a necessidade de execução do projeto do github.

4.2 Coleta de informações

Após a escolha dos componentes *button, datepicker, toggle button e chips*, realizamos uma análise detalhada onde exploramos suas estruturas HTML, classes e estilizações herdadas.

Utilizamos o *DevTools* para examinar o código fonte e obter informações relevantes sobre os componentes. Essa metodologia é

a mesma apresentada no algoritmo, em que abrimos o *DevTools* e inspecionamos o componente, dessa forma temos acesso a toda estrutura HTML podendo observar quais elementos o compõem e quais estilos já estão aplicados.

4.3 Ferramentas utilizadas

Durante o desenvolvimento do guia, utilizamos diversas ferramentas para facilitar o processo. As principais incluem:

- *Angular Material*: Utilizamos a biblioteca como base para os componentes abordados no guia. Exploramos sua documentação oficial e recursos para entender a estrutura e funcionalidades dos componentes.
- *CodeSandBox*: Onde hospedamos o guia de forma interativa possibilitando modificações dos usuários em tempo real.
- *Drawio*: Essa ferramenta serviu para geração do diagrama do algoritmo a ser seguido ao estilizar um componente.
- Editor de código: *Visual Studio Code*, para escrever e formatar o conteúdo do guia.
- Navegadores: Foram explorados diferentes navegadores, como Google Chrome, Mozilla Firefox e Microsoft Edge, com suas respectivas *DevTools*, para inspecionar e analisar os componentes prontos. Ao final escolhemos utilizar o Google Chrome para nossos exemplos.
- *Netlify*: Foi utilizado para disponibilizar uma página web com o guia para não haver a necessidade de executar o projeto do *github*.

4.4 Definição das modificações de cada componente escolhido e aplicação do algoritmo

4.4.1 *Button*

O componente escolhido para servir como exemplo mais básico foi o *button*, levando em consideração a estrutura do componente e suas propriedades. Essa escolha foi feita visando exemplificar de forma simples a aplicação de estilizações diretas no componente, sem a necessidade de alterações complexas em seu código-fonte. Optamos por realizar modificações nas bordas do botão, sem a necessidade de modificar sua estrutura interna, pois ao modificar apenas as bordas do botão, exploramos a facilidade de personalização dos componentes reutilizáveis no Angular, destacando a flexibilidade e a capacidade de adaptação desses componentes às necessidades de design específicas.

```
<button mat-raised-button color="primary" class="button">
  Primary
</button>
```

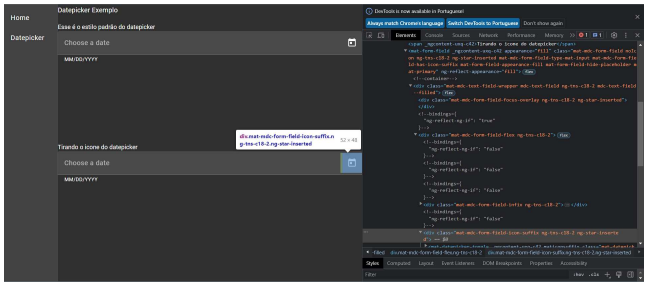
```
button {
  width: 90px;
}

.button {
  border: 2px solid white;
  border-radius: 20px;
}
```

Figura 2: Exemplo de uma mudança nas bordas, antes e depois.

Para utilizar o *button* diferentemente dos outros exemplos os quais são elementos importados e únicos do *Angular Material,* nós adicionamos a propriedade "mat-raised-button" a um elemento *button* nativo do HTML. Com essa estrutura as modificações podem ser feitas diretamente nesse elemento. Então aplicamos o estilo *width* diretamente ao *button*, o que afeta os dois *buttons* presentes na página e adicionamos uma *class* chamada "button" para aplicar as modificações apenas ao nosso exemplo.

4.4.2 *Datepicker*

O *datepicker* foi escolhido como o primeiro exemplo para demonstrar o uso do :host ::ng-deep para acessar e modificar as classes internas do componente. Após a análise de sua estrutura HTML e suas propriedades, decidimos realizar duas modificações específicas: a remoção do ícone e a alteração da cor. Essa escolha foi feita com o objetivo de exemplificar o processo de identificação do local da modificação desejada e a sobreposição dos estilos existentes, destacando a capacidade de personalização dos componentes reutilizáveis no Angular.



```
<mat-form-field appearance="fill" class="noIcon">
  <mat-label>Choose a date</mat-label>
  <input matInput [matDatepicker]="picker1" />
  <mat-hint>MM/DD/YYYY</mat-hint>
  <mat-datepicker-toggle
    matIconSuffix
    [for]="picker1"
  ></mat-datepicker-toggle>
  <mat-datepicker #picker1></mat-datepicker>
</mat-form-field>
```

```
:host ::ng-deep .noIcon .mat-mdc-form-field-icon-suffix {
    display: none;
}
```

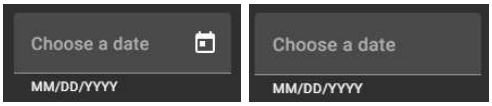


Figura 3: Exemplo removendo o ícone/button, antes e depois.

```
<mat-form-field appearance="fill" class="icon-color">
  <mat-label>Choose a date</mat-label>
  <input matInput [matDatepicker]="picker2" />
  <mat-hint>MM/DD/YYYY</mat-hint>
  <mat-datepicker-toggle
    matIconSuffix
    [for]="picker2"
  ></mat-datepicker-toggle>
  <mat-datepicker #picker2></mat-datepicker>
</mat-form-field>
```

```
:host ::ng-deep .icon-color .mat-mdc-icon-button {
  color: gray;
}
```

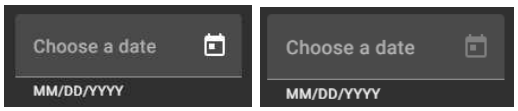


Figura 4: Exemplo de mudando a cor, antes e depois.

No *datepicker* foi aplicado uma *class* para cada exemplo que iríamos modificar, "noIcon" e "icon-color", já que não queríamos modificar todos da mesma forma. Depois abrimos o *DevTools* e inspecionamos o elemento que queremos, após encontrarmos onde devem ser aplicadas as estilizações utilizamos o ":host ::ng-deep + a *class* que criamos + a *class* do elemento que selecionamos" e aplicamos as modificações, isso foi feito para os dois exemplos.

4.4.3 *Toggle button*

Para o componente *toggle button*, foram exploradas diferentes opções de estilização. Além de remover as bordas, também foram realizadas modificações na forma e cor do botão quando selecionado. Essas alterações foram escolhidas com o intuito de exemplificar a flexibilidade dos componentes, permitindo a personalização visual de acordo com as necessidades específicas do design. O *toggle button* serviu como um exemplo adicional ao do *datepicker* para demonstrar a aplicação de estilos distintos, reiterando a capacidade de adaptar o componente a diferentes contextos e interações do usuário.

```
<section class="example">
  <h3>Modificando o toggle button</h3>
  <mat-button-toggle-group name="language" aria-label="Font Style">
    <mat-button-toggle value="pt-br">pt-br</mat-button-toggle>
    <mat-button-toggle value="en">en</mat-button-toggle>
    <mat-button-toggle value="es">es</mat-button-toggle>
  </mat-button-toggle-group>
```

```
.exemple mat-button-toggle-group {
  width: fit-content;
  border: none;
}

:host ::ng-deep .exemple mat-button-toggle-group mat-button-toggle {
  height: 30px;
  width: 55px;
  display: flex;
  align-items: center;
  border-radius: 50% 50% 50% 50%;
  margin: 0 5px;
}

:host ::ng-deep .exemple mat-button-toggle-group .mat-button-toggle-checked {
  background-color: #7b1fa2;
}
```

Figura 5: Exemplo de remoção das bordas, forma e cor. Antes e depois.

Foi utilizado o mesmo algoritmo aplicado no *datepicker* porém dessa vez a *class* "example" foi aplicada no elemento pai do *toggle button,* com essa alternativa nós devemos utilizar a "*class* que criamos + o elemento + a *class*, id ou elemento que queremos modificar", o que pra esse exemplo consiste em utilizar ".example mat-button-toggle-group" tanto isoladamente como na primeira estilização aplicada, quanto junto ao ":host ::ng-deep" para acessar propriedades internas.

4.4.4 *Chips*

No exemplo do componente chips, além de realizar modificações em alguns estilos previamente existentes, foi adicionado um novo elemento dentro do componente, sendo ele um ícone que não estava disponível nativamente. Essa escolha foi feita com o propósito de demonstrar a capacidade de flexibilização do Angular, permitindo a adição de novos elementos além da personalização de elementos previamente disponíveis.

Esse exemplo evidencia a possibilidade de expandir as capacidades dos componentes prontos, enriquecendo, assim, a funcionalidade e oferecendo aos desenvolvedores a liberdade de adicionar novos elementos e recursos de acordo com os requisitos do design e da experiência do usuário na criação de interfaces personalizadas.

```
<mat-chip-listbox aria-label="Fish selection">
  <mat-chip-option>
    <mat-icon>thumb_up</mat-icon>
    Like
  </mat-chip-option>
  <mat-chip-option color="warn">
    <mat-icon>thumb_down</mat-icon>
    Dislike
  </mat-chip-option>
</mat-chip-listbox>
```

```
:host
  ::ng-deep
  .default
  .mdc-evolution-chip__graphic.mat-mdc-chip-graphic.ng-star-inserted {
  display: block;
}

:host
  ::ng-deep
  .default
  .mat-mdc-standard-chip.mat-primary.mat-mdc-chip-selected {
  background-color: #7b1fa2;
}

:host ::ng-deep {
  &.mdc-evolution-chip__graphic.mat-mdc-chip-graphic.ng-star-inserted {
    display: none;
  }

  &.mdc-evolution-chip__text-label.mat-mdc-chip-action-label {
    align-items: center;
    display: flex;
    padding-left: 10px;
  }

  &.mat-mdc-standard-chip.mat-primary.mat-mdc-chip-selected {
    background-color: blue;
  }
}
```

Figura 6: Exemplo de mudando a cor, desabilitando o ícone e inserindo um novo que não estava disponível, antes e depois.

Nesse exemplo foi utilizado mais uma abordagem possível, definimos a *class* "default" que desfaz as modificações aplicadas de forma geral para toda página. Adicionamos um "mat-icon" (ícone) para cada opção do *chips,* após essa modificação criamos uma cadeia de estilização utilizando o ":host ::ng-deep" juntamente com a tag "&", que concatena ":host ::ng-deep" com o que vem depois da "&", após aplicarmos o algoritmo e descobrindo quais as classes que devemos modificar e definimos os novos estilos utilizando o "& + o nome das classes".

4.5 Revisão e refinamento

Após a criação inicial do guia, realizamos revisões e refinamentos para garantir a clareza, precisão e usabilidade do conteúdo. Também solicitamos feedback de outros desenvolvedores e especialistas para melhorar ainda mais a qualidade do guia.

Ao seguir esse processo de desenvolvimento, pudemos criar um guia abrangente e útil, fornecendo aos desenvolvedores front-end recursos e diretrizes claras para a estilização dos componentes reutilizáveis no Angular.

4.5.1 *CodeSandBox*

Ao final foi constatada a necessidade de uma versão do guia interativa pois para utilização e modificação seria necessário o download das ferramentas e execução do projeto no github. Para isso foi utilizado o *CodeSandBox* que possibilita mudanças em tempo real em todo o projeto, dessa forma o leitor pode fazer testes modificando ou adicionando estilizações nos exemplos, melhorando a experiência de aprendizado do usuário.

5. ESTUDO DE CASO DO IMPACTO DO GUIA

A pesquisa foi realizada com alunos e ex-alunos do curso de Ciência da Computação ou afins, com intuito de verificar qual o impacto do guia apresentado, bem como coletar informações de melhorias para o mesmo. Com exceção da primeira pergunta que é relativa ao nível de experiência do participante, as outras perguntas tinham como possibilidade de resposta uma escala de concordância, onde 1 significa discordo totalmente e 5 concordo totalmente.

Obtivemos um total de 21 respostas para o formulário, onde podemos observar na figura 7 que em sua grande maioria de 57,1% dos participantes tem menos de 1 ano de experiência com desenvolvimento *front-end* em Angular, seguido de 14,3% com 1 a 2 anos, 28,6% com 2 a 5 anos e ninguém com mais de 5 anos. Dessa forma as respostas as demais perguntas refletem mais a realidade do propósito do guia, já que o foco são os desenvolvedores iniciantes.

Figura 7: Experiência dos participantes.

Com as respostas da segunda pergunta (ver figura 8) percebemos que as avaliações foram muito positivas variando de 3 a 5, com 42% das avaliações sendo 5, juntamente com a figura 9 que mostra resultados similares corroborando que o algoritmo e demais diretivas apresentadas no guia são consideradas claras e eficazes, proporcionando uma boa metodologia para estilização de componentes prontos no Angular.

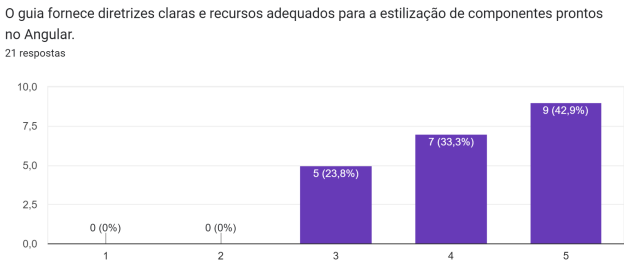


Figura 8: Fornecimento de diretrizes claras e recursos adequados.

Figura 9: Eficácia da metodologia apresentada no guia.

Nesta pergunta (ver figura 10) podemos observar que 1 participante achou que o guia não tem muitos níveis de complexidade, 4 que concordaram completamente, porém a grande maioria acha que têm diferentes níveis de complexidade mas com possibilidade de adição de novos exemplos, o que evidencia uma abertura para melhorias, algo que já é previsto e temos como trabalhos futuros.

Se observarmos a figura 11 podemos presumir que o participante que considerou que o guia não tem exemplos de grande diversidade, também considera que ele não seria tão útil, contudo as demais respostas continuam avaliadas entre 3 e 5, isso demonstra que o guia é considerado útil, mas está não podemos ignorar essa avaliação pois ela demonstra que para atingirmos um maior público devemos melhorar o guia, pressupondo que o participante que divergiu da maioria seja um desenvolvedor mais experiente.

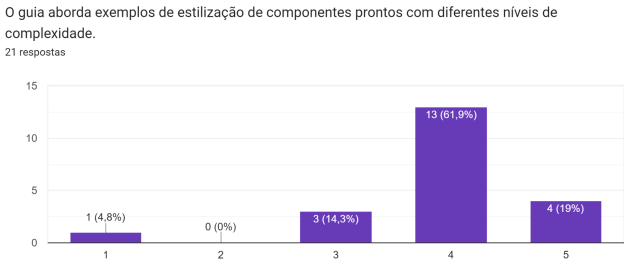


Figura 10: Diferentes níveis de complexidade dos exemplos.

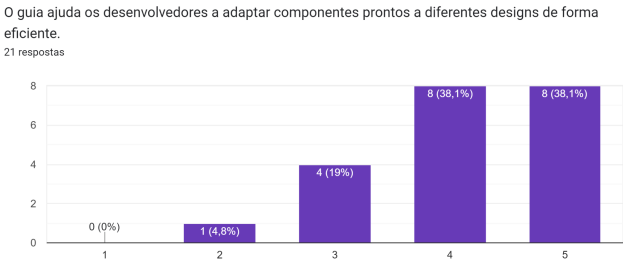


Figura 11: Ajuda do guia na adaptação dos componentes para diferentes designs.

Considerando as figuras 12 e 13, podemos observar que 15 participantes consideram o guia útil para eles (os que votaram entre 4 e 5), 3 foram indiferentes e 3 não acham tão útil, porém 18 consideram ele útil para desenvolvedores iniciantes (os que votaram entre 4 e 5) e 3 foram indiferentes, o que nos permite considerar que mesmo os que não acharam o guia útil para eles não puderam descartar a possibilidade dele ser uma opção viável para desenvolvedores iniciantes, e os que ficaram indiferentes ou acharam úteis para si mesmo recomendam o guia para os iniciantes, mostrando que o mesmo tem um bom impacto para desenvolvedores iniciantes e uma margem para ajudar desenvolvedores já experientes.

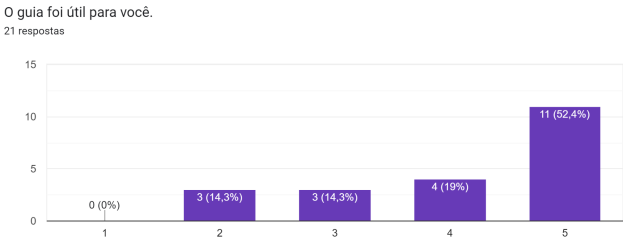


Figura 12: Impacto no guia para o participante.

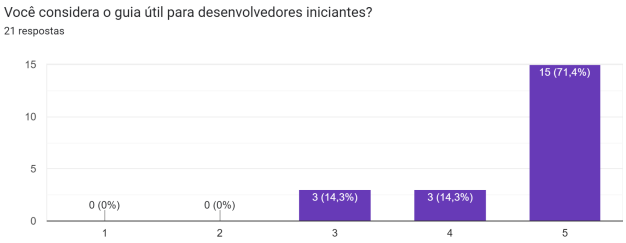


Figura 13: Impacto no guia para o desenvolvedores iniciantes

Para as sugestões de melhoria do guia foram apenas 3 respostas, dentre elas tornar o guia interativo já estava em andamento, mas não ficou pronto quando a pesquisa estava sendo realizada, porém já está disponível uma versão iterativa do guia no CodeSandBox permitindo manipulação dos exemplos.
A criação de subtópicos para compreensão do assunto e requisitos é um ponto que foi adicionado a trabalhos futuros, pois mesmo não sendo o foco do guia, percebemos que é um ponto que seria muito útil e agregaria mais, assim como melhorar a aparência e apresentação dos exemplos, tanto para ajudar no entendimento do guia quanto para atrair mais desenvolvedores.

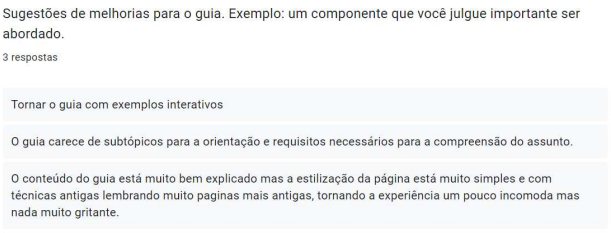


Figura 14: Sugestões de melhorias.

6. CONCLUSÃO E TRABALHOS FUTUROS

Podemos concluir que a estilização de componentes reutilizáveis no Angular é um desafio enfrentado pelos desenvolvedores front-end. Embora existam diversas bibliotecas e recursos prontos disponíveis, é necessário compreender a estrutura interna desses componentes e aplicar modificações de forma eficiente para adaptá-los aos diferentes designs e necessidades de uma aplicação o que, por sua vez, não é complicado, porém a escassez de materiais claros explicando como deve ser feito torna este tópico mais difícil que o necessário.

Durante o desenvolvimento do guia, foram abordados exemplos práticos de estilização de componentes prontos no Angular, utilizando técnicas como modificação direta de propriedades visuais e acesso às classes internas através do " :host ::ng-deep". Esses exemplos proporcionaram uma compreensão mais clara do processo de personalização dos componentes e destacaram a flexibilidade e a adaptabilidade dessas soluções pré-existentes.

Através do estudo de caso realizado podemos observar que o guia precisa de melhorias mas já pode ser considerado útil para desenvolvedores iniciantes e também tem potencial de abranger desenvolvedores mais experientes. Após a realização das melhorias propostas nos trabalhos futuros, acreditamos que esse guia será um importante material acadêmico.

Em suma, a estilização de componentes providos por bibliotecas no Angular requer conhecimento das estruturas internas dos componentes, aplicação de técnicas adequadas e compreensão das propriedades visuais e classes envolvidas. Com o guia desenvolvido, os desenvolvedores têm à disposição um recurso valioso para enfrentar os desafios de personalização e adaptação visual dos componentes, contribuindo para o desenvolvimento de interfaces atraentes e funcionais nas aplicações *front-end.*

Possíveis trabalhos futuros incluem a aplicação do guia em estudos de caso práticos, a pesquisa para identificar quais componentes devem ser adicionados ao guia e a exploração de técnicas avançadas de estilização de componentes. Essas iniciativas visam aperfeiçoar e expandir o guia elaborado, fornecendo diretrizes mais abrangentes, avaliando sua eficácia, identificando componentes relevantes e explorando abordagens avançadas para a personalização visual. Também consideramos adicionar uma introdução ao assunto e formalização do guia para que mesmo desenvolvedores que não estejam familiarizados com esse contexto possam entendê-los e usufruir do guia, assim como melhorar sua aparência.

7. Referências

[1] Site DigitalHouse. Front-end: o que é, para que serve e como aprender? Disponível em: https://www.digitalhouse.com/br/blog/front-end-o-que-e-para-que-serve-e-como-aprender

[2] CEBRIAN, Michael Christopher. Angular Component Library Comparison. Dissertação (Mestrado em Science in Software Engineering) – Departamento de Ciência da Computação, Villanova University. Maio, 2017. Disponível em: https://www.proquest.com/openview/2d417b342f2d2e1dc0ba5b48d0ddd65b/1?pq-origsite=gscholar&cbl=18750

[3] Livro - Learn to Code HTML and CSS: Develop and Style Websites por Shay Howe

[4] Livro - Pro Angular 9: Build Powerful and Dynamic Web Apps por Adam Freeman

[5] Angular material https://material.angular.io/