Técnicas de implementação de componentes dinâmicos em Angular aplicados para *e-commerce*

Diego Garcia Vieira[1], Lucas Baggio Figueira[2]

[1,2] Faculdade de Tecnologia de Ribeirão Preto (FATEC)
Ribeirão Preto, SP – Brasil

[1]diego.fismat@gmail.com, [2]lucas.figueira@fatec.sp.gov.br

Resumo. *Este artigo descreve técnicas utilizadas para a criação de componentes dinâmicos para e-commerce, utilizando o framework Angular. No decorrer deste artigo iremos apresentar a natureza do problema que atinge alguns softwares que são utilizados publicamente e que necessitam de um ritmo rápido de mudança. Objetivo final deste artigo é informar ao leitor meios de dinamizar os componentes de um website, gerando grande flexibilização e facilidade de manutenção no produto final.*

Abstract. *This article describes techniques used to create dynamic components for e-commerce using the Angular framework. Throughout this article we will present the nature of the problem that affects some software that are used publicly, which demands quick changes. The final objective of this article is to inform the reader of ways to streamline the components of a website, generating great flexibility and ease of maintenance in the final product.*

1 Introdução

Atualmente, no mercado de trabalho, necessitamos de softwares capazes de se adaptarem de maneira muito rápida e contínua (CI&T, 2019). A crescente complexidade criada pelo nosso complexo mercado, fornece uma série de mudanças ao longo do processo de amadurecimento de um software. Visando atender essa demanda, se faz necessárias técnicas inteligentes que permitem acomodar a natureza complexa que o mercado de trabalho impõe no nosso cotidiano. Ao longo deste artigo iremos apresentar uma técnica utilizada para construir componentes dinâmicos, em específico para um software de *e-commerce*.

É possível verificar algumas soluções no mercado que, apesar de atenderem o objetivo final, pecam na questão de facilidade de manutenção e impõem uma complexidade exponencial conforme as mudanças vão sendo requisitadas. Uma das técnicas mais comuns encontradas no mercado de trabalho é a separação do código para cada cliente, ou seja, teremos versões diferentes do mesmo software que vão evoluindo separadamente. Um dos principais problemas que isso traz é o compartilhamento de

código e funções comuns do sistema, pois requer uma constante sincronização entre as diferentes versões do software.

A técnica apresentada aqui visa sanar esse problema, permitindo que partes comuns funcionais do sistema possam caminhar juntas, e regiões do software que precisam ser desmembradas em diferentes componentes, possam assim fazê-lo, de forma dinâmica, sem que haja necessidade de se criar outras versões do código.

Nosso desafio aqui é criar, de maneira uniforme e simples, os componentes dinâmicos capazes de suportar constantes mudanças e customizações que serão requisitadas ao longo do ciclo de vida de um software.

Pode-se dizer que o Angular (AFONSO, 2018) é uma ferramenta que foi criada tendo como um dos seus critérios resolver o problema de coesão e acoplamento de forma simples e dinâmica (CADU, 2010). Por meio de sua estrutura feita com base em componentes, podemos montar um projeto com coesão e baixo acoplamento (CADU, 2010), onde conseguimos, de forma granular, ativar ou desativar partes do nosso software. Portanto, dada essa natureza fragmentada inerente à própria estrutura do Angular, nos possibilitou alcançar a flexibilidade necessária e desejada do *e-commerce* estudado neste artigo.

2 Pesquisa bibliográfica

No começo do desenvolvimento da WEB, os *browsers* competiam entre si com a criação de funcionalidades, visando alcançar uma certa vantagem competitiva. Existia uma grande diversidade entre os *browsers*, e muitas vezes nós desenvolvedores éramos obrigados a manter versões diferentes do nosso site afim de suportar vários *browsers* diferentes. Houve então um momento em que se criaram um conjunto de regras e padrões afim de viabilizar o desenvolvimento da WEB (NOTTINGHAM, 2020).

Um dos efeitos dessa padronização foi a criação dos *Web Components*, que nada mais são que componentes baseados em padrões de desenvolvimento agnósticos a qualquer ferramenta de desenvolvimento (KARÉN, 2019). *Browsers* através de padrões e especificações técnicas foram capazes de suportar uma interface comum universal capaz de prover aos desenvolvedores meios de reutilizarem seus códigos com estes componentes.

Visando obter-se o melhor dos padrões do ecossistema WEB (NOTTINGHAM, 2020), o Angular (AFONSO, 2018) foi escolhido pois faz uso extensivo dos padrões modernos atuais (KARÉN, 2019). Portanto ficamos seguros de que, com nossa escolha, poderemos produzir um *e-commerce* que irá ser funcional em qualquer *browser* escolhido, e poderemos utilizar da sua natureza flexível e dinâmica dos componentes web para dar alta agilidade e baixa complexidade, trazendo um alto reuso de código para o desenvolvimento. De acordo com a Figura 1, podemos verificar que o Angular (AFONSO, 2018), possui 100% de compatibilidade com a especificação de *Web Components* (KARÉN, 2019), garantindo assim que tenhamos um software com coesão e baixo acoplamento (CADU, 2010) e possamos atender a natureza ágil e complexa do desenvolvimento de software (CI&T, 2019).

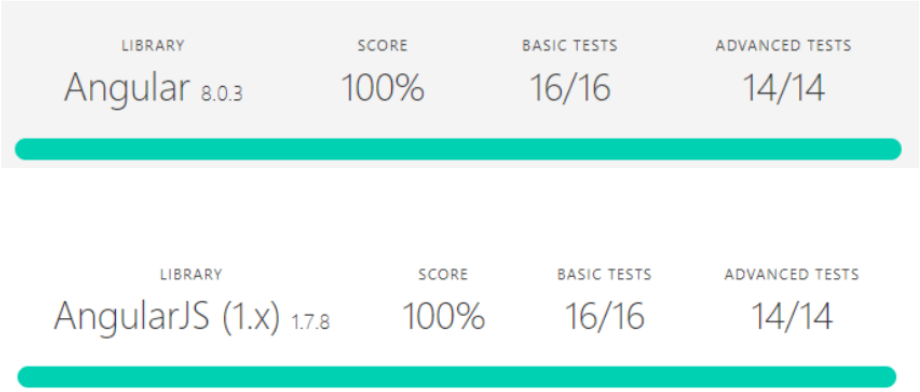

Figura 1. Compatibilidade (fonte: KARÉN 2019)

3 Contextualização e objetivos

O presente artigo tratará da aplicação de componentes dinâmicos para quatro diferentes clientes, que apesar de possuírem peculiaridades, possuem também um fluxo em comum, que é a venda *online*. Iremos mostrar como separar, criar, e relacionar componentes diferentes atrelados a um fluxo em comum a aos quatro clientes:

- Cliente A
- Cliente B
- Cliente C
- Cliente D

O *e-commerce* estudado neste artigo é utilizado para a venda de passagens de

transporte rodoviário para o cidadão. Portanto, o processo de venda em nada difere de um cliente para outro, pois os quatro clientes estudados utilizam o *e-commerce* para esse tipo de venda. Porém, como cada cliente possui uma marca própria assim como uma identidade, se faz necessário que tenhamos mudanças de certas partes do software.

Mostraremos como podemos ter uma página inicial de um *e-commerce* customizada, onde teremos três partes customizadas:

- Header (cabeçalho da página)
- Home (seção entre cabeçalho da página e o rodapé)
- Footer (rodapé da página)

Estas partes, apesar de serem individuais, todas servem como página inicial *do e-commerce*, sendo que, a partir delas, os usuários que acessarem o site poderão realizar uma busca para assim poderem comprar suas passagens. A Figura 2 ilustra o componente que é comum a todas essas partes.

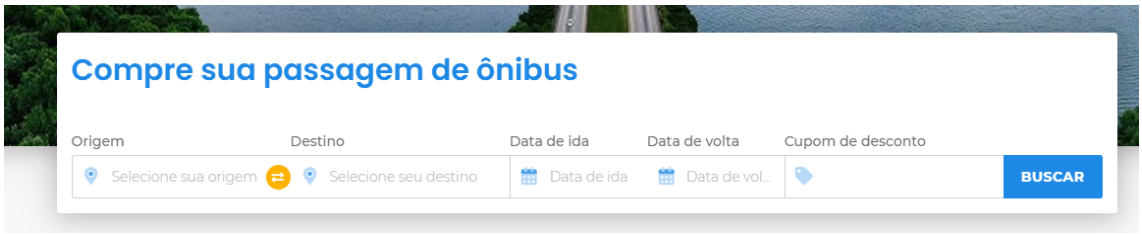

Figura 2. Componente de busca (Fonte: próprio autor)

4 Estruturação básica

Foi escolhida uma estruturação baseada em separação por pastas, já seguindo as melhores práticas advogadas pelo time de desenvolvimento que criou o Angular. Na Figura 3 temos a estrutura de separação dos componentes mencionados acima. Em vermelho temos o *footer* (rodapé da página), *header* (cabeçalho) e *home* (seção entre o cabeçalho e rodapé) respectivamente. Cada pasta com o nome de um cliente possui um componente específico para a pasta acima (em vermelho), ou seja, na hierarquia da pasta *footer* temos quatro pastas, e cada pasta possui um componente específico para se ter um *footer* funcional. O mesmo ocorre para a pasta *header* e *home*, para cada empresa teremos, especificamente, um componente cuja função é determinada pela pasta em vermelho.

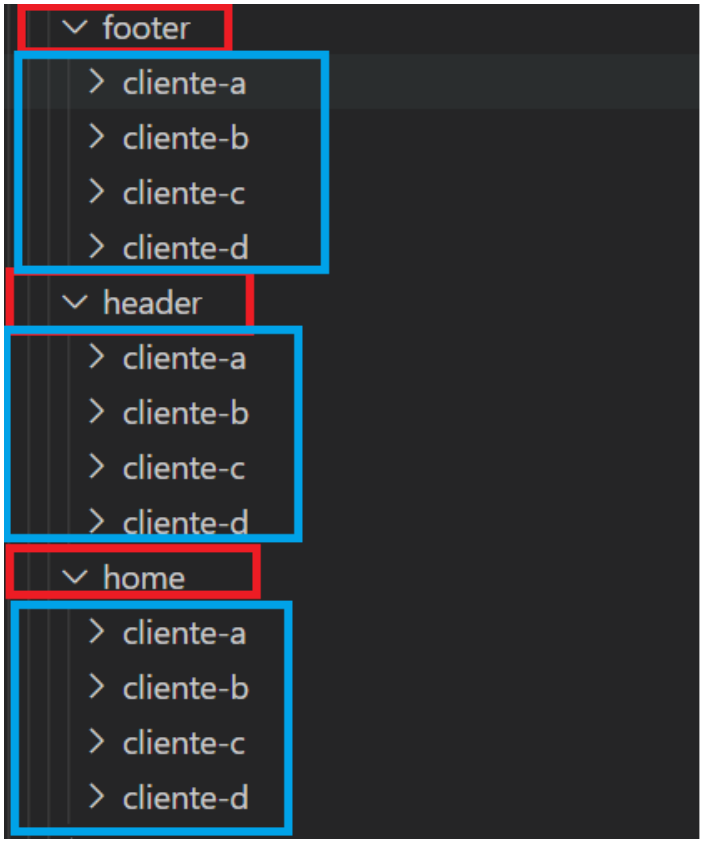

Figura 3. Estrutura de pastas (Fonte: próprio autor)

Seguindo esta estrutura, seremos capazes de termos uma página inicial totalmente separada sem que isso cause uma fragmentação de ordem maior no software em questão. Sendo que todas páginas iniciais possuem o componente ilustrado na Figura 2, que é responsável por iniciar o processo de busca, que é comum a todos os quatro clientes.

4.1 *Component container*

Usaremos o conceito de *component container* (NIELSEN, 2018) para exemplificar como é possível que troquemos os três componentes: *Header, Home* e *Footer* de forma dinâmica. O *container* nada mais é que um componente Angular capaz de abrigar outros componentes Angular. Ele é utilizado como base para que possamos escolher quais componentes serão utilizados, conforme uma determinada configuração. Na Figura 4 mostramos a estrutura básica do componente escolhido (TEKTUTORIALSHUB, 2020). Será o app-component, pois é o componente inicial nos projetos Angular.

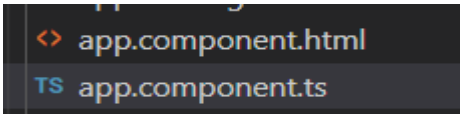

Figura 4. Componente container escolhido (Fonte: próprio autor)

4.2 Estrutura de HTML do componente container

Os componentes dinâmicos precisam serem renderizados no DOM (W3SCHOOLS, 2020), ou seja, precisam de algum local onde possamos indicar qual componente iremos renderizar. Por exemplo, querem decidir se iremos renderizar o *Header* da Guanabara, ou o *Header* do Cliente A. A Figura 5 mostra como podemos criar de forma dinâmica componentes app.component.html. Utilizaremos a tag HTML ng-template (CUSTÓDIA, 2017) que é específica do Angular, cujo objetivo é exatamente podermos representar qualquer estrutura que esteja no padrão HTML.

```
<div class="container-fluid">
    <div class="row">
        <div class="col">
            <ng-template #headerContainer></ng-template>
            <router-outlet></router-outlet>
            <ng-template #footerContainer></ng-template>
        </div>
    </div>
</div>
```

Figura 5. Componente container escolhido (Fonte: próprio autor)

Outra tag utilizada do Angular será a router-outlet (ADRIANO, 2017). Esta tag é utilizada para que determinada URL requisitada ao nosso site possa fornecer um determinado componente. No nosso caso, a URL requisitada será a página inicial e tag irá nos ajudar a renderizar o componente responsável pela *Home*.

Portanto nossa tag em vermelho será responsável por acomodar o *Header*, a tag

em azul será responsável por acomodar a *Home*, e a tag em amarelo será utilizada para acomodar o *Footer*.

4.3 Implementação do componente container

A tag ng-template (CUSTÓDIA, 2017) precisa ser referenciada por um nome prefixado pelo símbolo: "#". Ou seja, para o *header* iremos referenciá-lo por #headerContainer e para o *footer* como #footerContainer. No nosso *component container* iremos, a partir de uma variável de configuração, escolher qual componente será injetada nessas tags ng-template (CUSTÓDIA, 2017). Para isso precisamos ter uma referência das tags através dos seus identificadores, como mostramos na Figura 6:

```
@ViewChild('headerContainer', { static: true, read: ViewContainerRef }) headerContainer: ViewContainerRef;
@ViewChild('footerContainer', { static: true, read: ViewContainerRef }) footerContainer: ViewContainerRef;
```

Figura 6. Componente container escolhido (Fonte: próprio autor)

Agora, a partir de uma variável de configuração (Figura 8), iremos definir qual componente será utilizado a partir da configuração do nosso ambiente (Linha 159). Para isso, iremos utilizar o ComponenteFactoryResolver (Figura 7). A partir deste objeto disponibilizado pelo Angular, podemos de forma dinâmica, conforme mostra a Figura 8, injetar os componentes nas tag ng-template (CUSTÓDIA, 2017). O primeiro passo é obter-se uma instância do componente que queremos aplicar na tag ng-template (CUSTÓDIA, 2017), isso é feito na Linha 157 para o componente *Header* do cliente A. Com a instância do componente em mãos fazer com que o container crie dentro de si a instância (Linha 159).

```
private resolver: ComponentFactoryResolver,
```

Figura 7. Componente container escolhido (Fonte: próprio autor)

```
159         if (environment.Cliente === 'A') {
160             const headerFactory = this.resolver.resolveComponentFactory(ClienteAHeaderComponent);
161             const footerFactory = this.resolver.resolveComponentFactory(ClienteAFooterComponent);
162             const headerComponentRef = this.headerContainer.createComponent(headerFactory);
163             const footerComponentRef = this.footerContainer.createComponent(footerFactory);
164 
165             route = [
166                 { path: '', component: ClienteAHomeComponent }
167             ];
```

Figura 8. Injetando os componentes Header e Footer (Fonte: próprio autor)

4.4 Usando a tag router-outlet

Um outro ponto muito importante é que num projeto Angular temos a renderização de componentes conforme a URL em que se está navegando, por exemplo, na Figura 7, nós precisamos criar um objeto route para passarmos como configuração para nosso router que é responsável por gerenciar essas rotas. Na Linha 162 temos a criação deste objeto passando o componente ClienteAHomeComponent determinado a partir do environment. Na Linha 166 passamos o objeto route para esta configuração.

5 Avaliando os resultados

Para avaliarmos nosso processo, basta inspecionar o HTML gerado na nossa aplicação. Conforme as Figuras 9 e 10 podemos verificar que conseguimos produzir um HTML dinâmico, conforme a configuração do *enviroment* e a partir de componentes dinâmicos.

```
<div class="container-fluid">
  <div class="row">
    <div class="col">
      <!---->
      <app-cliente-a-header>…</app-cliente-a-header>
      <router-outlet></router-outlet>
      <app-cliente-a-home>…</app-cliente-a-home>
      <!---->
      <app-cliente-a-footer>…</app-cliente-a-footer>
    </div>
  </div>
</div>
```

Figura 9. HTML gerado para o Cliente A (Fonte: próprio autor)

```
<div class="container-fluid">
  <div class="row">
    <div class="col"> == $0
      <!---->
      <app-cliente-b-header>…</app-cliente-b-header>
      <router-outlet></router-outlet>
      <app-cliente-b-home>…</app-cliente-b-home>
      <!---->
      <app-cliente-b-footer>…</app-cliente-b-footer>
    </div>
  </div>
</div>
```

Figura 10. HTML gerado para o Cliente B (Fonte: próprio autor)

6 Conclusão

Vimos que, a partir de uma ferramenta moderna e flexível como o Angular, somos capazes de criar componentes dinâmicos que visam atender às demandas de mudanças e complexidades intrínsecas do nosso mercado. De forma bem simplificada, podemos ter n componentes e combinações diferentes para que possamos obter o resultado desejado, que é um software funcional e capaz de atender aos anseios de seu público, assim como também é fácil de manter e com um custo de complexidade baixo. Fica claro que as possibilidades são muitas e as aplicações também.

REFERÊNCIAS

ADRIANO, T. S. Angular 5: Trabalhando com Rotas. Medium, 2017. Disponível em <https://medium.com/angularbr/angular-5-trabalhando-com-rotas-8335617fcdbc>. Acesso em: 19 de junho de 2020.

AFONSO, A. O que é Angular. Blog Alga Works, 2018. Disponível em <https://blog.algaworks.com/o-que-e-angular>. Acesso em: 25 de junho de 2020.

CADU. Entendendo Coesão e Acoplamento. Devmedia, 2010. Disponível em <https://www.devmedia.com.br/entendendo-coesao-e-acoplamento/18538>. Acesso em: 18 de junho de 2020.

CI&T. Como o Agile revoluciona o mercado de trabalho, 2019. Disponível em <https://br.ciandt.com/blog/agile-a-metodologia-que-revolucionou-o-mercado-de-tecnologia>. Acesso em: 18 de junho de 2020.

CUSTÓDIA, H. Angular: NgTemplate. Medium, 2017. Disponível em < https://medium.com/criciumadev/angular-ngtemplate-d59b32d47d01>. Acesso em: 16 de junho de 2020.

KARÉN, M. The State of Web Components. Medium, 2019. Disponível em <https://medium.com/swlh/the-state-of-web-components-e3f746a22d75>. Acesso em: 25 de junho de 2020.

MOZILLA. HTML: Linguagem de Marcação de Hipertexto, 2020. Disponível em <https://developer.mozilla.org/pt-BR/docs/Web/HTML>. Acesso em: 16 de junho de 2020.

NIELSEN, L. G. B. Container components with Angular. In Depth Dev, 06 de novembro 2018. Disponível em <https://indepth.dev/container-components-with-angular>. Acesso em: 19 de junho de 2020.

NOTTINGHAM, M. What is the Web. Mnot's Blog, 2020. Disponível em <https://www.mnot.net/blog/2014/12/04/what_is_the_web>. Acesso em: 25 de junho de 2020.

TEKTUTORIALSHUB Angular Folter Structure Best Pratices, 2020. Disponível em <https://www.tektutorialshub.com/angular/angular-folder-structure-best-practices>. Acesso em: 19 de junho de 2020.

W3SCHOOLS. What is the HTML DOM, 2020. Disponível em <https://www.w3schools.com/whatis/whatis_htmldom.asp>. Acesso em: 16 de junho de 2020.

WEBSHARE O que é *browser* ou navegador, 2020. Disponível em < https://www.webshare.com.br/glossario/o-que-e-browser-ou-navegador>. Acesso em: 25 de junho de 2020.