DESAFIO 5.2

Prólogo:
No desafio 5.2 seu objetivo é entender o conceito por trás dos componentes e aprender a reutilizá-los através de **herança** ou através do uso do **CSS selector**. Serão apresentados dois métodos de troca de informações entre componentes, um utilizando **property binding** que foi aprendido no desafio 5.1 e outro através das rotas, tanto por **state** como por **queryParams**. Abordaremos também maneiras de chamar e configurar uma rota, além dos tipos de carregamento de conteúdo que podem ser definidos através das rotas: **Eager Load** e **Lazy Load**. Por fim, aprender a transformar dados através dos pipes padrões do Angular e a criação de pipes customizados. Ao final desse desafio, deverá ser entregue ao tutor as respostas no readme e o projeto criado para solucionar o desafio, ambos versionador no Git.

ITEM 1: **https://bityli.com/XdyMGG**
Pesquise os palavras em **vermelho**

Item 1: **Reutilização de componentes**

a) A respeito de componente, responda:

I. O que é um componente em aplicações Angular?
II. Do que um componente é composto?
III. Qual é o comando utilizado para criar um componente utilizando Angular CLI?
IV. Qual a importância da reutilização de componentes?
V. Qual a funcionalidade do selector no exemplo abaixo?

Exemplo:
```
@Component({
  selector: 'app-component-overview',
})
```

VI. Explique a funcionalidade de **templateUrl** e template nos exemplos abaixo e quando devemos utilizar cada uma delas.

Exemplo 1:
```
@Component({
  selector: 'app-component',
  templateUrl: './component.component.html'
})
```

Exemplo 2:
```
@Component({
  selector: 'app-component-overview',
  template: '<h1>Hello World!</h1>'
})
```

VII. Explique a funcionalidade de **styleUrls** e **styles** nos exemplos abaixo e quando devemos utilizar cada um deles.

Exemplo 1:
```
@Component({
  selector: 'app-component',
  templateUrl: './component.component.html',
  styleUrls: ['./component.component.css']
})
```

Exemplo 2:
```
@Component({
  selector: 'app-component-overview',
  template: '<h1>Hello World!</h1>',
  styles: ['h1 { font-weight: normal; }']
})
```

Hands-on

I. Abra o repositório da trilha.
II. Crie uma pasta para o **Desafio Component**, se já criou siga para o próximo passo.
III. Abra o cmd e vá até a pasta do Desafio Component.

- Para chegar a pasta utilize o comando abaixo, substituindo "caminho-da-pasta" pelo caminho da pasta criada para o Desafio Component:
cd caminho-da-pasta

IV. Digite o comando para criação de um projeto angular:
ng new desafio-component

V. Serão feitas algumas perguntas no terminal durante o processo, responda como abaixo, digite ou selecione a resposta e clique na tecla Enter:

- Would you like to add Angular routing? **Resposta: y**
- Which stylesheet format would you like to use? **Resposta: SCSS**

VI. Quando o processo terminar, abra o VSCode, clique em arquivo e depois em abrir pasta.
VII. Encontre e selecione a pasta desafio-component, que está dentro da pasta criada para o Desafio Component .
VIII. Depois de abrir o projeto, abra o arquivo **src**/**app**/**app.component.html** e substitua todo o conteúdo dele por:

```
<header>
<h1>Desafio Component</h1>
</header>
```

IX. Rode a aplicação no CMD, Git Bash, ou terminal integrado do VSCode, com o comando: **npm start**
X. Abra o navegador no link: **http://localhost:4200/** e veja o resultado.

Existem 2 formas de reutilizar componentes no Angular que são muito utilizadas:

- Utilizando Herança, através do uso de extends
- Chamando o componente através do seu seletor

Reutilizando códigos através de herança

Para aplicar o passo a passo a seguir, utilizaremos a aplicação desafio-component que criamos.

I. No VSCode, clique com o botão direito do mouse na pasta **src**/**app** e clique em nova pasta e dê o nome de componentes
II. Clique com o botão direito do mouse na pasta **src**/**app**/**components** e clique em Abrir terminal integrado
III. No terminal aberto utilize o comando abaixo para criar um novo componente chamado componente-pai. **ng generate component componente-pai**
IV. Exclua os arquivos:

- src/**app**/**components**/**componente-pai**/**componente-pai.component.html**
- src/**app**/**components**/**componente-pai**/**componente-pai.component.scss**

V. Abra o arquivo:

- src/**app**/**components**/**componente-pai**/**componente-pai.component.ts**
e o deixe como no exemplo abaixo:

Exemplo:
```
import { Component, Input } from '@angular/core';

@Component({
selector: 'app-componente-pai',
template: ""
})

export class ComponentePaiComponent {

@Input() numeroComponent: number = 0;

isPar(valor: number): string {
return valor % 2 === 0 ? "par" : "impar";
}
}
```

VI. No terminal aberto no item 2 utilize o comando abaixo para criar um novo componente chamado componente-filho: **ng generate component componente-filho**
VII. Abra o arquivo:

- src/**app**/**components**/**componente-filho**/**componente-filho.component.ts**
e o deixe como no exemplo abaixo:

Exemplo:
```
import { ComponentePaiComponent } from
  '../componente-pai/componente-pai.component';
import { Component } from '@angular/core';

@Component({
selector: 'app-componente-filho',
templateUrl: './componente-filho.component.html',
styleUrls: ['./componente-filho.component.scss']
})

export class ComponenteFilhoComponent extends
  ComponentePaiComponent {
}
```

OBS: Através do extends**, marcado acima, o componente filho recebe todos os métodos e todas as propriedades do componente pai, essa é uma excelente maneira de reaproveitar códigos.**

VIII. Abra o arquivo:

- src/**app**/**components**/**componente-filho**/**componente-filho.component.html**
e substitua todo o seu conteúdo por:

```
<hr aria-hidden="true">
<p>O número {{ numeroComponent }} é {{ isPar(numeroComponent) }}</p>
```

OBS: Perceba que a propriedade numeroComponent é uma propriedade do componente pai que está sendo utilizada no componente filho, assim como o método isPar(). Isso funcionará para todos os métodos e propriedades públicas e protegidas do componente pai.

Reutilizando componentes através do seletor

Para aplicar o passo a passo a seguir, utilizaremos a aplicação desafio-component que criamos.

I. Abra o arquivo **src**/**app**/**app.component.html** e adicione o seguinte código uma linha depois da última linha do arquivo:

```
  <main>
    <app-componente-filho [numeroComponent]="1"></app-componente-filho>
  </main>
```

OBS 1: Veja o resultado no seu navegador, deve ser algo parecido com:

Imagem 1 - Imagem do estado atual da aplicação, para o leitor de telas será um título de nível 1, Desafio Component. Seguido pelo texto, o número 1 é impar.

OBS 2: Perceba que o componente filho foi mostrado na tela através da utilização de uma tag HTML customizada, composta pelo seu seletor <app-componente-filho> e que utilizamos o Property Binding aprendido no Desafio 5.1 para passar um valor para o input numeroComponent do nosso componente.

Exercício Prático

Utilizando o que foi aprendido até agora, através da reutilização de códigos e componentes:

I. Adicione uma verificação se o número passado é primo e imprima na tela da seguinte maneira:

Imagem 2 - Imagem do estado em que a aplicação deve ficar, para o leitor de telas será um título de nível 1, Desafio Component. Seguido pelo texto, o número 1 é impar e é primo.

II. Imprima na tela essa verificação até o número 5, o resultado deve ser:

Imagem 3 - Imagem do estado em que a aplicação deve ficar, para o leitor de telas será um título de nível 1, Desafio Component. Seguido por uma sequência do texto, o número X é impar ou par e é ou não é primo, onde X vai de 1 até 5.

II : **https://bityli.com/qNWd4C**

Item 2: **Rotas**

Sobre o funcionamento das rotas do Angular, responda:

I. Qual é o comando no Angular CLI para criar uma nova aplicação com rotas?
II. Qual é o comando no Angular CLI para criar um novo módulo com rotas?
III. Para que serve o método navigate() da classe Router?
IV. Para que serve o método isactive() da classe Router?
V. Para que serve o arquivo **src/app/app-routing.module.ts**?
VI. Para que serve a tag <router-outlet></router-outlet>?
VII. Dê um exemplo de um botão chamando uma rota através de um método em angular.
VIII. Dê um exemplo de um botão chamando uma rota sem chamar um método em angular.
IX. O que é Lazy Load? Dê um exemplo da definição de uma rota em angular utilizando Lazy Load.
X. O que é Eager Load? Dê um exemplo da definição de uma rota em angular utilizando Eager Load.

Hands-on

Agora vamos criar um componente que irá nos mostrar o resultado do número que foi passado para ele através da rota ao quadrado. Para isso, primeiro vamos fazer com que a tela de lista de números seja chamada por um botão na tela inicial.

Chamada de rotas através de um botão

I. Clique com o botão direito do mouse na pasta **src/app/components** e clique em Abrir terminal integrado
II. No terminal aberto digite o seguinte código para criar um componente nova chamada lista-numeros: **ng generate component lista-numeros**
III. Abra o arquivo **src/app/components/lista-numeros/lista-numeros.component.html** e substitua o conteúdo dele por:

```
<app-component-filho [numeroComponent]="" ngFor="let i of [1,2,3,4,5]">
</app-component-filho>
```

IV. Agora vamos criar uma rota para chamar o nosso novo componente. Abra o arquivo **src/app/app-routing.module.ts** e adicione dentro do array routes o seguinte objeto:

```
{ path: 'lista', component: ListaNumerosComponent},
```

V. Para mostrar o componente da rota na tela é necessário utilizar a tag <router-outlet></router-outlet>, para isso, abra o arquivo **src/app/app.component.html** e adicione a tag <main>+/main> por:

```
<main>
  <button [routerLink]="['lista']">Abrir lista de números</button>
  <router-outlet></router-outlet>
</main>
```

VI. O resultado deve ser o seguinte:

Imagem 4 – Imagem do estado em que a aplicação deve ficar, para o leitor de telas será um título de nível 1, Desafio Component. Seguido por um botão chamado Abrir lista de números

Ao clicar no botão Abrir lista de números a tela com a lista que já haviamos visto deve abrir.

Reforço à chamada de rota através de um botão

I. Clique com o botão direito do mouse na pasta **src/app/components** e clique em Abrir terminal integrado
II. No terminal aberto digite o seguinte código para criar um componente nova chamado quadrado: **ng g c quadrado**

OBS: Note que dessa vez abreviamos o "generate component" com "g c".

III. Abra o arquivo **src/app/components/quadrado/quadrado.component.ts** e crie uma variável **público** chamada **valor** do tipo **number** e inicie ela com **0**
IV. Abra o arquivo **src/app/components/quadrado/quadrado.component.html** e substitua todo o conteúdo dele por:

```
<hr aria-hidden="true">
<p>{{ valor }}² = {{ valor * valor}}</p>
```

V. Agora vamos criar uma rota para chamar o nosso novo componente. Abra o arquivo **src/app/app-routing.module.ts** e adicione dentro do array routes o seguinte objeto:

```
{ path: 'quadrado', component: QuadradoComponent }
```

VI. Com a rota criada, podemos agora criar um botão para chamar esse nosso componente. Abra o arquivo **src/app/components/component-filho/componente-filho.component.html** e adicione o seguinte botão no final do arquivo:

```
<button [routerLink]="['/quadrado']">Quanto é {{ numeroComponent }} ao quadrado?
</button>
```

OBS: Rodando a aplicação você já consegurá visualizar a alteração da rota na barra de endereço do seu navegador ao clicar no botão.

VII. Após clicar no botão "Abrir lista de números" o resultado da tela deve ser parecido com:

Imagem 5 – Imagem do estado em que a aplicação deve ficar, para o leitor de telas será um título de nível 1, Desafio Component. Seguido por um botão "Abrir lista de números" e uma sequência do texto "o numero X é impar ou par e é ou não é primo", juntamente com um botão "Quanto é X ao quadrado?", onde X vai de 1 até 5.

VIII. Clicando em um dos botões de números o resultado deve ser parecido com isso:

Imagem 6 – Imagem do estado em que a aplicação deve ficar, para o leitor de telas será um título de nível 1, Desafio Component. Seguido por um botão "Abrir lista de números" e um texto 0² = 0

Comunicação entre componentes através da rota

I. Abra o arquivo **src/app/components/component-filho/componente-filho.component.html** e adicione o atributo [state]="{ valorRota: numeroComponent }" após a routerLink do button, como no exemplo abaixo:

```
<button [routerLink]="['/quadrado']" [state]="{ valorRota: numeroComponent }">
Quanto é {{ numeroComponent }} ao quadrado?
</button>
```

OBS: Desta forma nós estamos mandando o nosso numeroComponent para a variável valorRota do rota.

II. Abra o arquivo **src/app/components/quadrado/quadrado.component.ts** e substitua o construtor por:

```
constructor( private router: Router,)
  {
    this.valor = this.router.getCurrentNavigation()?.extras.state?.valorRota;
  }
```

OBS: Esse parte do código serve para pegar o valor que colocamos na variável valorRota e passar para dentro da variável valor do componente quadrado.

III. Com isso você já consegurá ver a aplicação funcionando e mostrando os dados corretos, ao clicar em "Abrir lista de números" e depois em "Quanto é 4 ao quadrado?", você deve visualizar a seguinte tela:

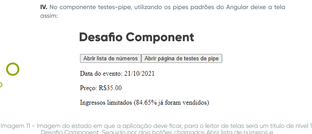

Imagem 7 – Imagem do estado em que a aplicação deve ficar, para o leitor de telas será um título de nível 1, Desafio Component. Seguido por um botão "Abrir lista de números" e um texto 4² = 16

Exercício Prático

a) Criar um novo componente que mostre o resultado de 'X', onde o X deve ser passado através de rotas.

I. Nesse exercício a chamada da rota, deverá ser feita por um botão ao lado do botão "Quanto é X ao quadrado?" com a descrição "Quanto é X ao cubo?"
II. Não é permitido utilizar routerLink e [state], nesse exercício a chamada deve ser através de um método que você deve criar, chamado pelo event binding (click) do button, deve utilizar o método navigate da classe Router e passar os dados pelo queryParams.
III. O resultado deverá ser parecido com:

1) Pagina inicial

Imagem 8 – Imagem do estado em que a aplicação deve ficar, para o leitor de telas será um título de nível 1, Desafio Component. Seguido por um botão "Abrir lista de números"

2) Após clicar em "Abrir lista de números"

3)Após clicar no botão "Quanto é 4 ao cubo?"

Imagem 9 – Imagem do estado em que a aplicação deve ficar, para o leitor de telas será um título de nível 1, Desafio Component. Seguido por um botão "Abrir lista de números" e um texto 4³ = 64

Item 3: **Pipes**

Por fim, respondas as seguintes perguntas sobre pipes

I. Qual é a utilidade dos pipes em aplicações Angular?
II. Cite os 6 pipes que fazem parte do pacote inicial do Angular.
III. Descreva a funcionalidade de cada um dos pipes citados acima e dê um exemplo da sua utilização, juntamente com o resultado em tela.
IV. O que são custom pipes?

Hands-on

Para treinar a utilização de pipes, iremos criar uma espécie de convite para um evento, neste convite teremos informações como data, preço, porcentagem de ingressos vendidos e um código do evento. Para cada um desses campos utilizaremos um pipe, existente ou customizado.

Utilização de pipes padrão do Angular

I. Crie um novo componente chamado testes-pipe, dentro da pasta **src/app/components.**
II. Crie uma rota que chame esse nosso novo componente.
III. Crie um botão que abra esse novo componente na página inicial, a tela inicial deve ficar parecida com:

Imagem 10– Imagem do estado em que a aplicação deve ficar, para o leitor de telas será um título de nível 1, Desafio Component. Seguido por dois botões chamados Abrir lista de números e Abrir página de testes de pipe

IV. No componente testes-pipe, utilizando os pipes padrão do Angular deixe a tela assim:

Imagem 11 – Imagem do estado em que a aplicação deve ficar, para o leitor de telas será um título de nível 1, Desafio Component. Seguido por dois botões chamados Abrir lista de números e Abrir página de testes de pipe e uma sequência dos textos, Data do evento: 21/10/2021, Preço: R$35,00, Ingressos limitados(84.65% já foram vendidos)

Criação de um pipe customizado

I. Crie uma pasta chamada pipe em src/app, a estrutura deve ficar assim:

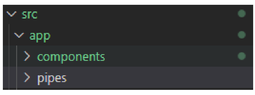

Imagem 12 - Imagem da estrutura de pastas que a aplicação deve ter, deve ser src > app > components e pipes

II. Abra o terminal integrado na pasta pipes e digite o seguinte comando para criar um pipe customizado chamado teste: **ng generate pipe codigo-convite**
III. No arquivo **src/app/pipes/codigo-convite.ts**, note que existe o seguinte código:

```
@Pipe({
  name: 'codigoConvite'
})
```

OBS: O "name" é como se fosse o seletor de um component. é ele que será utilizado nos templates para que a informação passe pelo pipe e retorne formatada ou modificada.

IV. Todos os names de pipes devem ser dados utilizando a formatação lowerCamelCase.
V. Abra o arquivo **src/app/components/testes-pipe/teste-pipe.component.html** e adicione a seguinte linha no final do arquivo:

```
<p>Código: {{ '12345678901234567890' | codigoConvite }}</p>
```

OBS: Assim você estará informando a aplicação para transformar a string '12345678901234567890' de acordo com o método transform do pipe codigoConvite que criamos.

VI. Abra o arquivo **src/app/pipes/codigo-convite.pipe.ts**, e perceba que atualmente o método transform está retornando apenas null, se você verificar a tela verá que nenhum dado estará aparecendo no campo código.
VII. Para fazermos um teste troque o retorno do método transform para 'testeando testes' e verifique o resultado na aplicação.

OBS: Faça alguns testes brincando com o retorno para ver diferentes resultados.

VIII. Agora para formatar nosso código de convite altere a função transform para:

```
transform(value: string): string {
        return value.substring(0, 4) + ' - ' + value.substring(4, 12) + ' / ' + value.substring(12, 20);
    }
```

IX. O resultado no seu navegador deve ser mais ou menos assim:

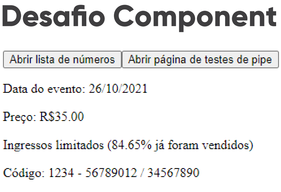

Imagem 13 - Imagem do estado em que a aplicação deve ficar, para o leitor de telas será um título de nível 1, Desafio Component. Seguido por dois botões chamados Abrir lista de números e Abrir página de testes de pipe e uma sequência dos textos, Data do evento: 21/10/2021, Preço: R$35.00, Ingressos limitados(84.65% já foram vendidos), Código: 1234 - 56789012 / 3456789

OBS: Utilizando pipes é possível formatar valores que foram passados, retornar valores diferentes de acordo com os valores passados, fazer operações matemáticas com os valores, retornar mensagens caso os valores não estejam de acordo com o esperado e muito mais. Pipes são extremamente versáteis e úteis para diversas ocasiões. Além disso, é possível esperar argumentos nos pipes e retornar valores de acordo com esses argumentos, assim como é feito no pipe currency no exemplo a seguir: {{ 12 | currency:'BRL' }}.

Exercício prático

I. Crie um pipe que receberá uma string e um argumento do tipo cpf | cnpj e retornará uma string com a formatação de acordo com o argumento recebido:

- CPF = 000.000.000-00
- CNPJ = 00.000.000/0000-00

O tela final deve ser parecida com:

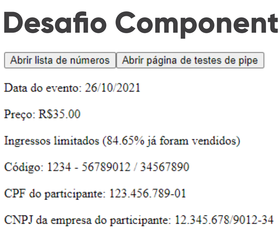

Imagem 14 - Imagem do estado em que a aplicação deve ficar, para o leitor de telas será um título de nível 1, Desafio Component. Seguido por dois botões chamados Abrir lista de números e Abrir página de testes de pipe e uma sequência dos textos, Data do evento: 21/10/2021, Preço: R$35.00, Ingressos limitados(84.65% já foram vendidos), Código: 1234 - 56789012 / 3456789, CPF do participante: 123.456.789-01, CNPJ da empresa do participante: 12.345.678/9012-34