LABORATÓRIO 2

Criando uma aplicação de compra.

Objetivo: Avançar na aplicação Angular criando rotas e diferentes visualizações.

Inicialmente, vamos construir um template parecido com a imagem abaixo:

Como visto nos slides, uma visualização do Angular pode ser uma junção de vários componentes. Sendo assim, podemos criar um componente de barra superior com um botão e abaixo dele um componente com a palavra Produtos.

1- Vamos construir a barra superior. Para isso, construiremos um componente com o nome barra-superior, dentro da nossa pasta app:

$ ng generate c barra-superior

Dentro do HTML criado, vamos colar o seguinte código

```
<a [routerLink]="['/']">
  <h1>Minha Loja</h1>
</a>

<a class="button fancy-button"><i class="material-icons">shopping_cart</i>Carrinho</a>
```

A tag <a> dentro do HTML é um link. A tag <h1> é um título.

A barra superior está criada.

2- O próximo passo é chamar, em nosso html original, a nossa nova barra.

Primeiro, vamos checar se o nosso novo component está declarado no módulo. Isso é feita nas declarações:

```
6   import { BarraSuperiorComponent } from './barra-superior/barra-superior.component';
7
8   @NgModule({
9     declarations: [
10      AppComponent,
11      BarraSuperiorComponent
12    ],
```

Isso foi feito automaticamente na criação do componente pelo Angular CLI.

Vamos também solucionar alguns problemas de estilo. Vamos importar os seguintes códigos na nossa folha de estilo principal:

```
/* Global Styles */

* {
  font-family: 'Roboto', Arial, sans-serif;
  color: #616161;
  box-sizing: border-box;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

body {
  margin: 0;
}

.container {
  display: flex;
  flex-direction: row;
}

router-outlet + *  {
  padding: 0 16px;
}

/* Text */

h1 {
  font-size: 32px;
}
```

```
h2 {
  font-size: 20px;
}

h1, h2 {
  font-weight: lighter;
}

p {
  font-size: 14px;
}

/* Hyperlink */

a {
  cursor: pointer;
  color: #1976d2;
  text-decoration: none;
}

a:hover {
  opacity: 0.8;
}

/* Input */

input {
  font-size: 14px;
  border-radius: 2px;
  padding: 8px;
  margin-bottom: 16px;
  border: 1px solid #BDBDBD;
}

label {
  font-size: 12px;
  font-weight: bold;
  margin-bottom: 4px;
  display: block;
  text-transform: uppercase;
}

/* Button */
.button, button {
  display: inline-flex;
  align-items: center;
  padding: 8px 16px;
```

```
  border-radius: 2px;
  font-size: 14px;
  cursor: pointer;
  background-color: #1976d2;
  color: white;
  border: none;
}

.button:hover, button:hover {
  opacity: 0.8;
  font-weight: normal;
}

.button:disabled, button:disabled {
  opacity: 0.5;
  cursor: auto;
}

/* Fancy Button */

.fancy-button {
  background-color: white;
  color: #1976d2;
}

.fancy-button i.material-icons {
  color: #1976d2;
  padding-right: 4px;
}

/* Top Bar */

app-barra-superior {
  width: 100%;
  height: 68px;
  background-color: #1976d2;
  padding: 16px;
  display: flex;
  flex-direction: row;
  justify-content: space-between;
  align-items: center;
}

app-barra-superior h1 {
  color: white;
  margin: 0;
}
```

```
/* Checkout Cart, Shipping Prices */

.cart-item, .shipping-item {
  width: 100%;
  min-width: 400px;
  max-width: 450px;
  display: flex;
  flex-direction: row;
  justify-content: space-between;
  padding: 16px 32px;
  margin-bottom: 8px;
  border-radius: 2px;
  background-color: #EEEEEE;
}
```

Além disso, vamos importar algumas fontes de ícones diretamente no index.html:

```
<link
  href="https://fonts.googleapis.com/icon?family=Material+Icons"
  rel="stylesheet"
/>
```

Agora, vamos importar o component no HTML, utilizando o selector da nossa nova barra:

```
1  <!-- Toolbar -->
2  <app-barra-superior></app-barra-superior>
```

Nossa barra superior está pronta e importada.

3- Agora, construiremos o local onde uma lista de produtos vai aparecer. Para isso, novamente vamos acessar a pasta src/app e criar um novo Component:

```
$  ng generate c lista-produtos
```

No novo HTML, vamos colocar apenas a tag <h2> com o nome Produtos, por enquanto:

```
<h2>Produtos</h2>
```

Precisamos também informar ao sistema que redirecione para o Component criado ao entrar em um caminho específico da url. Para isso, podemos definir as rotas no routing.module:

```
{path: '', component: ListaProdutosComponent}
```

Também devemos importar esse novo componente no HTML. Vamos colocar a seguinte informação:

```
<!-- Toolbar -->
<app-barra-superior></app-barra-superior>

<div class="container">
        <router-outlet></router-outlet>
</div>
```

A tag <router-outlet> informa para o angular o lugar específico que a redirecionamento da rota vai carregar o Component.

Concluímos, portanto, a proposta inicial. Continuando:

Criando a lista de produtos:

1. Vamos usar dados pré-definidos. Para isso, criaremos um arquivo na nossa pasta app, chamada produtos.ts. Vamos carregar esses arquivos como array:

```
export interface Produto {
  id: number;
  nome: string;
  preco: number;
  descricao: string;
}

export const produtos = [
  {
    id: 1,
    nome: 'Telefone Samsung',
    preco: 799,
    descricao: 'Celular com novas funções e ótima câmera'
  },
  {
    id: 2,
    nome: 'Telefone Iphone',
    preco: 699,
    descricao: 'Celular muito tecnológico com tela muito boa'
  },
  {
    id: 3,
    nome: 'Telefone Motorola',
    preco: 299,
```

```
    descricao: ''
  }
];
```

2. Depois, temos que carregar esses dados no Componente Listar Produtos. Para isso, importamos no arquivo typescript e recebemos em uma variável
3. Abra o HTML da pasta lista-produtos
4. Adicionar uma com um *ngFor

```
<h2>Produtos</h2>

<div *ngFor="let produto of produtos">
</div>
```

A diretiva *ngFor informa ao Angular que essa deve ser repetida pelo número de produtos na lista.

Existe também, por exemplo, o *ngIf.

Estruturas diretivas modificam a estruturas da DOM, adicionando, removendo e manipulando elementos.

5. Dentro da , vamos adicionar a tag <h3> e a variável {{produto.nome}}. Esse é um exemplo da sintaxe de interpolação. Interpolação deixa você renderizar o valor da propriedade como um texto.

```
<h2>Produtos</h2>

<div *ngFor="let produto of produtos">
      <h3>
         {{ produto.nome }}
      </h3>
</div>
```

6. Para transformar cada nome em um link de produto, vamos adicionar a tag <a>
7. Vamos setar o título do nome do produto usando a propriedade []:

```
<h2>Produtos</h2>

<div *ngFor="let produto of produtos">
  <h3>
    <a [title]="produto.nome + 'detalhar'">
     {{ produto.nome }}
    </a>
  </h3>
</div>
```

8. Agora, ao visualizar a nossa aplicação, conseguimos verificar a dica ao passar o mouse em cima do link.
9. Vamos adiconar a descrição dos produtos. Dentro da nossa diretiva ngFor*, vamos adicionar uma tag <p> com as descrições:

```
<h2>Produtos</h2>

<div *ngFor="let produto of produto">

  <h3>
    <a [title]=" produto.name + ' detalhar">
      {{ produto.name }}
    </a>
  </h3>

  <p *ngIf=" produto.descricao">
    Descrição: {{ produto.descricao}}
  </p>

</div>
```

A diretiva *ngIf garante que a tag <p> só seja criada para Produtos que tenham descrição.

10. Vamos agora adicionar um botão de Compartilhar os Produtos. Primeiro, vamos criar um método no nosso Componente:

```
compartilhar() {
    window.alert('O produto foi compartilhado!');
  }
```

Depois disso, precisamos criar um botão dentro do nosso HTML:

```
<button type="button" (click)="compartilhar()">
    Compartilhar
  </button>
```

Agora, ao clicar no botão, vamos receber um alerta que o produto foi compartilhado.

Passando dados para componentes filhos:

1. Inicialmente, criaremos um novo Component, alerta-produtos, via ng generate.
2. Importamos os produtos.
3. Dentro do Component, definimos uma propriedade chamada produto com um @Input() decorator. Esse decorator indica que o valor da propriedade vai ser passado pelo componente pai, ListaProdutos.

```
export class AlertaProdutosComponent
{

  @Input() produto!: Produto;

}
```

4. No HTML desse novo Component, vamos adicionar um botão de alerta:

```
<p *ngIf="produto && produto.preco > 700">
  <button type="button">Alerta</button>
</p>
```

Sendo assim, quando o preço do produto for acima de 700, um botão de alerta vai aparecer vai aparecer.

5. Finalmente, para mostrar esse novo Componente como um filho da lista de Produtos, vamos adicionar o selector do novo Component no HTML da lista de Produtos. Vamos passar o produto como um input do Component, utilizando a vinculação de propriedade (property biding).

```
<app-alerta-produtos
  [produto]="produto">
</ app-alerta-produtos>
```

Passando dados para componentes pai:

Para fazer com que o botão de notificação funcione, o componente filho precisa notificar e passar os dados para o componente pai. Sendo assim, o AlertaProdutosComponent precisa emitir um evento quando o usuário clicar Promoção e o ListaProdutosComponent precisa responder esse event.

1. No componente de alerta de produtos, vamos importar o decorator Output.

Na classe, vamos definir esse novo Output:

```
@Output() alerta= new EventEmitter();
```

2. No HTML, vamos atualizar o botão Promoção com um evento para chamar esse output:

```
<p *ngIf="produto && produto.preco> 700">
  <button type="button" (click)="alerta.emit()"> Alerta</button>
</p>
```

3. Agora temos que estabelecer o comportamento do componente pai ao receber a notificação de clique. O componente pai que vai agir quando o filho levantar um evento

No component Listar Produtos, vamos definir um método onNotificacao(), similar ao compartilhar:

```
onAlerta () {
   window.alert('Quando o produto estiver em promoção, você será avisado');
  }
```

4. Precisamos também atualizar o HTML do listar produtos, fazendo a vinculação do Output do componente filho com o método no componente pai

```
(alerta)=" onAlerta ()"
```

Terminamos aqui o Laboratório 2. Conseguimos criar uma aplicação que lista os produtos, além de ter passado por vários conceitos do Angular. Dúvidas?