Lista de Exercícios

# Exercício 01

Antes de dar qualquer passo, devemos garantir que o FormsModule esteja importado no AppModule.

Dentro do app.component implementar uma funcionalidade para inserir em uma lista nome e e-mail de clientes e depois exibir na tela uma tabela com os clientes da lista. Para finalizar, a partir da tabela, deve ser possível remover um registro.

No app.component.html, deve ser disponibilizado dois inputs com [(ngModel)] associando a um objeto no app.component.ts e um botão chamando nosso método que e fará um .push deste objeto na lista que deve ser apresentada em uma tabela logo abaixo dos inputs.

Objetos declarados

cliente: any = {};

clientes: any[] = [];

Forma de realizar um push em uma lista

this.clientes.push({nome: cliente.nome, email: cliente.email});

Utilizar a diretiva \*ngFor para listarmos os clientes em uma lista da seguinte forma:

<table>

<tr>

<th>Nome</th>

</tr>

<tr \*ngFor="let obj of objetos">

<td>{{obj.nome}}</td>

</tr>

</table>

Forma de remover um cliente pelo nome

this.clientes = this.clientes.filter(c => c.nome !== cliente.nome);

# Exercício 02

1. Limpar o conteúdo do app.componente.html com exceção do router-outlet
2. Estilizar o home.component.html como quiser como uma página principal
3. Criar Componente de Pagina não encontrada.
4. Criar rotas para os componentes criados. Para página não encontrada, utilizar **“\*\*”** como path da rota.

# Exercício 03

Com base no exercício 1 e 2, realizar CRUD de clientes utilizando o banco de dados mocado em lista.

* Realizar download do código do repositório abaixo:

<https://github.com/roney-amorim/angular-primeng/tree/develop>

## Campos

Nome, Documento, E-mail e Data de Nascimento.

## Dicas

### Componente de edição:

A rota do componente de editar deve ser feita da seguinte forma para que possamos resgatar o cliente pelo id:

{path: 'item/:id', component: ClienteCadastroComponent}

Necessário importar o ActivatedRoute e injeta-lo no construtor do componente:

constructor(private route: ActivatedRoute) { }

Dentro ngOnInit no mesmo componente, pode acessar os dados nos parâmetros observáveis da rota, assinando-os desta forma:

ngOnInit() {

this.route.params.subscribe(params => {

console.log(params) //log the entire params object

console.log(params['id']) //log the value of id

});

}

# **Exercício 4**

Utilizando as bibliotecas do primeng, vamos criar um componente de autocomplete para listar clientes da nossa aplicação.

Primeiro vamos criar um Módulo e um Componente para este módulo que representará nosso Autocomplete de clientes.

ng g m clientes/auto-complete-cliente

ng g c clientes/auto-complete-cliente

Depois devemos importar o AutoCompleteModule do primeng no nosso AutoCompleteClientesModule

import { AutoCompleteModule } from "primeng/primeng";

Todos os elementos customizados no Angular precisam ser marcados como elementos customizados da seguinte forma:

schemas:[

CUSTOM\_ELEMENTS\_SCHEMA

]

Criando o componente HTML seguido de seu typescript

O componente AutoComplete usa **ngModel** para ligação bidirecional, onde requer uma lista de **sugestões**.

O **completeMethod** obtém o texto da consulta como **event.query** e deve atualizar as **sugestões** com os resultados da pesquisa.

<p-autoComplete [(ngModel)]="cliente" [suggestions]="clientes" (completeMethod)="pesquisar($event)" [size]="30"

placeholder="Countries" [minLength]="1"></p-autoComplete>

Por fim no TypeScript, basta criar um objeto e uma lista de clientes, um Data com os valores a serem consultados e um método de pesquisa (Banco de Dados Mocado):

cliente: any = {};

clientes: any[] = [];

data: string[] = ['JOSE DA SILVA', 'MARIA DA SILVA', 'JOÃO MAIA', 'FRANCISCO S. ALVES', 'LUCAS SOUZA', 'LUCIA NAVES', 'LUIZA MAIA'];

constructor() {}

ngOnInit() {

}

pesquisar(event) {

this.clientes = [];

for (let item of this.data) {

if (item.toLowerCase().indexOf(event.query.toLowerCase()) === 0) {

this.clientes.push(item);

}

}

}

## Componente criado, vamos à utilização do componente em nossa aplicação:

Primeiro passo é importar o módulo do Custom Component no módulo no qual o componente vai ser utilizado

imports: [

CommonModule,

ClientesRoutingModule,

AutoCompleteClientesModule

],

Segundo passo, o mais simples é adicionar a tag do Custom Component no html onde queremos utilizá-lo

Em listar-component.html basta declarar a tag:

<app-auto-complete-clientes></app-auto-complete-clientes>

# Exercício Final

Em cima da aplicação base já criada com o sistema de login realizado.

## 1. Realizar download do código do repositório abaixo

<https://github.com/roney-amorim/angular-jwt-authentication/tree/develop>

## 2. Base das rotas das APIS a serem utilizadas

<https://github.com/roney-amorim/workshop-angular/blob/master/rotas.txt>

## 3. Criar um Crud de clientes

### Validar Campos Obrigatórios (nome, documento e data nascimento)

## 4. Criar um Crud de veículos

### Com base no CRUD de clientes, implementar métodos (buscar, buscarPorId, cadastrar, alterar e remover ) nos componentes e na service de veículos utilizando HttpClient.

### Validar Campos Obrigatórios (Marca, modelo, placa, preço, desconto, ano e motor)

## 5. Criar funcionalidade de vendas

### Tela de vendas de veículos para clientes cadastrados

* Para cada venda deve ser possível incluir descontos direto no valor da venda **(precoVenda)**, desde que não ultrapassem o valor de desconto pré-cadastrado para o veículo a ser vendido.

### Tela de Consulta

* Listar todas as vendas realizadas permitindo realizar o cancelamento de venda por registro.

### Funcionalidade de Cancelamento

* Funcionalidade deve ser possível ser realizada apenas para compras feitas com menos de 7 dias, caso contrário exibir a mensagem de erro “Cadastro de venda vigente a mais de 7 dias ”