**Parte 1 – Setup Inicial**

1. Crie um novo projeto angular:

**ng new crud-angular**

1. Abra o projeto no VSC
2. Instale o BootStrap e o JQuery no projeto”:

**npm install bootstrap --save**

**npm install jquery --save**

1. Configure o BootStrap no projeto, alterando o arquivo angular.json:

"styles": [

"src/styles.css","node\_modules/bootstrap/dist/css/bootstrap.min.css"

],

"scripts": [

"node\_modules/jquery/dist/jquery.min.js",

"node\_modules/bootstrap/dist/js/bootstrap.min.js"

]

1. Crie um novo componente para listar todos os empregados

**ng g c empregados**

1. Altere o template do componente principal para:

<app-empregados></app-empregados>

1. Altere o template do componente empregados para:

<h2>Lista de Empregados</h2>>

1. Execute a aplicação:

**ng serve --open**

**Parte 2: Serviço**

1. Crie a classe empregado dentro do arquivo *empregado.ts*

export class Empregado {

id: number;

nome: string;

funcao: string;

salario: number;

dataAdmissao: string;

}

1. Configure o HttpClientModule no projeto alterando o arquivo app.module.ts:

import { HttpClientModule} from '@angular/common/http';

imports: [

BrowserModule,

**HttpClientModule**

],

1. Crie um serviço para empregado:

**ng g s empregado**

1. Na classe de serviço de empregado faça o importes abaixo:

import { Observable } from 'rxjs';

import { Empregado } from './empregado';

import { HttpClient } from '@angular/common/http';

1. Crie um atributo para retornar para apontar para a url do webservice:

private url = 'http://localhost:3000/funcionarios';

1. No construtor da classe de serviço faça a injeção do HttpClient:

constructor(private http: HttpClient) { }

1. Crie um método que retorne todos os empregados:

getEmpregados(): Observable <Empregado[]> {

return this.http.get<Empregado[]>(this.url);

}

**Parte 3: Listando todos os empregados**

1. Na classe do componente empregado faça os imports abaixo:

import { Empregado } from '../empregado';

import { EmpregadoService } from '../empregado.service';

1. Crie um atributo empregados do tipo array:

empregados: Empregado[];

1. Faça a injeção do serviço pelo construtor:

constructor(private empregadoService: EmpregadoService) { }

1. Crie o método getEmpregados():

getEmpregados(): void {

this.empregadoService.getEmpregados().subscribe(emp => this.empregados = emp);

}

1. Chame o método getEmpregados no ngOnInit:

ngOnInit() {

this.getEmpregados();

}

1. No template do componente de empregados acrescente o código abaixo para mostrar todos os empregados em uma tabela:

<div class="col-md-12">

<div class="table-responsive table-container">

<table class="table">

<thead>

<tr>

<th>Id</th>

<th>Nome</th>

<th>Função</th>

<th>Salário</th>

<th>Data Admissão</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let emp of empregados">

<td class="hidden">{{emp.id}}</td>

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

<td>{{emp.funcao}}</td>

<td>{{emp.salario}}</td>

<td>{{emp.dataAdmissao | date : 'dd/MM/yyyy'}}</td>

</tr>

</tbody>

</table>

</div>

</div>

**Parte 4: Mock para Empregados**

1. Instale o Json Server para criar um mock para os empregados

**npm install -g json-server**

1. Crie um arquivo chamado de **funcionarios.json** com a seguinte estrutura com pelo menos dois funcionários. Crie esse arquivo dentro da pasta src:

{

"funcionarios": [

{

"id": 1,

"nome": "A",

"salario": 1,

"funcao": "A",

"dataAdmissao": "2018-01-01"

},

{

"id": 2,

"nome": "B",

"salario": 2,

"funcao": "B",

"dataAdmissao": "2018-02-02"

}

]

}

1. Execute o servidor para carregar o mock de funcionários:

**json-server ./src/funcionarios.json**

1. Salve todos os arquivos e reinicie o servidor para apresentar o resultado abaixo:



**Parte 5: Botão Remover**

1. Acrescente os seguintes imports na classe de serviço:

import { HttpClient, HttpHeaders } from '@angular/common/http';

import { tap } from 'rxjs/operators';

1. Crie uma constante para indicar o tipo de conteúdo do método de remoção:

const httpOptions = {

headers: new HttpHeaders({ 'Content-Type': 'application/json' })

};

1. Acrescente o método deleteEmpregado na classe de serviço:

deleteEmpregado(id: number): Observable<Empregado> {

const url = `${this.url}/${id}`;

return this.http.delete<Empregado>(url, httpOptions).pipe(

tap(\_ => console.log(`deleted empregado id=${id}`))

);

}

1. Na classe do componente de empregados acrescente o método de remoção:

deleteEmpregado(empregado: Empregado): void {

this.empregados = this.empregados.filter( emp => emp !== empregado );

this.empregadoService.deleteEmpregado(empregado.id).subscribe();

}

1. No template do componente de empregados acrescente mais uma coluna a tabela:

<td>

<button type="button" class="btn btn-success" \*ngIf="btnVisibility" (click)="onSubmit()">Save</button>

</td>

1. Como resultado teremos:



**Parte 6: NavBar**

1. Crie um componente para o Home do projeto:

**ng g c home**

1. Crie um componente para o Sobre do projeto

**ng g c sobre**

1. Crie um componente para o NavBar:

**ng g c navbar**

1. Altere o template do navbar para:

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">

<a class="navbar-brand" href="#">Home</a>

<button class="navbar-toggler"

type="button"

data-toggle="collapse"

data-target="#navbarSupportedContent"

aria-controls="navbarSupportedContent"

aria-expanded="false"

aria-label="Toggle navigation"

>

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav mr-auto">

<li class="nav-item active">

<a class="nav-link" href="/empregados">Empregados</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/sobre">Sobre</a>

</li>

</ul>

</div>

</nav>

1. Crie um módulo para gerenciar as rotas do projeto:

**ng g m app-routing --flat --module=app**

1. Altere o arquivo **app.routing.modulo.ts** para:

import { NgModule } from '@angular/core';

import { RouterModule, Routes } from '@angular/router';

import { EmpregadosComponent} from './empregados/empregados.component';

import { HomeComponent} from './home/home.component';

import { SobreComponent} from './sobre/sobre.component';

const routes: Routes = [

{ path: '', redirectTo: '/home', pathMatch: 'full' },

{ path: 'home', component: HomeComponent },

{ path: 'empregados', component: EmpregadosComponent },

{ path: 'sobre', component: SobreComponent },

];

@NgModule({

imports: [ RouterModule.forRoot(routes) ],

exports: [ RouterModule ]

})

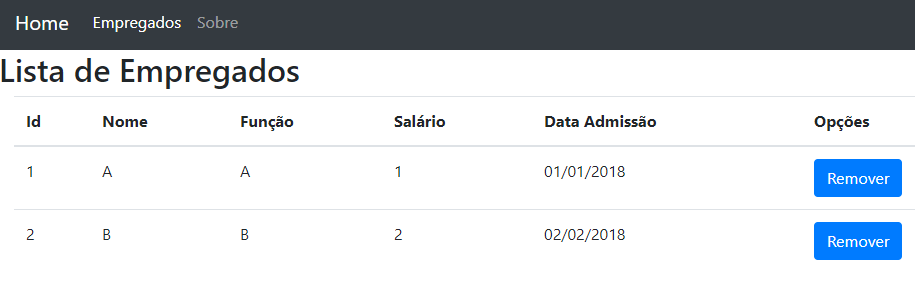
export class AppRoutingModule { }

1. Altere o template do componente principal para:

<app-navbar></app-navbar>

<router-outlet></router-outlet>

1. Resultado final



**Parte 7: Adicionando Novo Empregado Parte I**

1. Adicione um método na classe de serviço para salvar um novo empregado:

createEmpregado(empregado: Empregado): Observable <Empregado> {

return this.http.post<Empregado>(this.url, empregado);

}

1. Crie um componente filho para o componente de empregados para poder adicionar um novo empregado:

**ng g c empregados/empregado-detail**

1. Faça os seguintes imports na classe do componente empregado-detail:

import { Component, OnInit, Output, EventEmitter } from '@angular/core';

import { EmpregadoService } from '../../empregado.service';

import { Empregado } from '../../empregado';

1. Acrescente os atributos para a classe do componente:

empregado: Empregado;

@Output()

emissor: EventEmitter<Empregado> = new EventEmitter<Empregado>();

1. Altere o construtor para injetar as dependências do componente:

constructor(private empService: EmpregadoService) {

}

1. Altere o método ngOnInit para definir novo objeto empregado:

this.empregado = new Empregado();

1. Crie o método na classe do componente para salvar um novo empregado:

onSubmit() {

this.empService.createEmpregado(this.empregado).subscribe(

emp => this.emissor.emit(this.empregado)

);

}

1. No template do componente copie o seguinte código:

<h2 class="text-center">Novo Empregado</h2>

<div class="col-md-6">

<form>

<div class="form-group">

<label for="empId">Empregado Id:</label>

<input type="number" [(ngModel)]="empregado.id" placeholder="Id" name="empId" class="form-control" id="empId">

</div>

<div class="form-group">

<label for="empName">Nome do Empregado:</label>

<input [(ngModel)]="empregado.nome" placeholder="Nome do Empregado" name="empNome" class="form-control" id="empNome">

</div>

<div class="form-group">

<label for="empFuncao">Função do Empregado:</label>

<input [(ngModel)]="empregado.funcao" placeholder="Funcao do Empregado" name="empFuncao" class="form-control" id="empFuncao">

</div>

<div class="form-group">

<label for="empSalario">Salário do Empregado:</label>

<input type="number" [(ngModel)]="empregado.salario" placeholder="0.00" name="empSalario" class="form-control" id="empSalario" >

</div>

<div class="form-group">

<label for="empDataAdm">Data de Admissão:</label>

<input type="date" [(ngModel)]="empregado.dataAdmissao" placeholder="dd/mm/aaaa" name="empDataAdm" class="form-control" id="empDataAdm" >

</div>

<button type="button" class="btn btn-success" (click)="onSubmit()">Save</button>

</form>

</div>

**Parte 8: Adicionando Novo Empregado Parte II**

1. Defina um novo atributo para mostrar ou não o componente filho (empregado-detail). A linha em destaque é a única que foi inserida na classe do componente.

export class EmpregadosComponent implements OnInit {

empregados: Empregado[];

showForm = false;

1. Adicione dois novos métodos para o componente pai (Empregados)

// Mostra o formulario de cadastro ou alteracao de empregado

novoEmpregado() {

this.showForm = true;

}

// Recebe um evento de confirmacao do filho de um novo empregado

novoEmpregadoEvent(novoEmpregado) {

this.showForm = false;

this.empregados.push(novoEmpregado);

}

1. Altere o template de empregados para chamar o componente filho. As linhas em destaque é que foram alteradas ou inseridas. Observe a última desse arquivo!

<app-empregado-detail (emissor) = "empregadoEvent($event)" \*ngIf="showForm == true">

</app-empregado-detail>

<div \*ngIf="showForm == false">

<h2 class="text-center">Lista de empregados</h2>

<button (click)="novoEmpregado()" class="btn btn-primary">Novo Empregado </button>

<div class="col-md-12">

<div class="table-responsive table-container">

<table class="table">

<thead>

<tr>

<th>Id</th>

<th>Nome</th>

<th>Função</th>

<th>Salário</th>

<th>Data Admissão</th>

<th>Opções</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let emp of empregados">

<td class="hidden">{{emp.id}}</td>

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

<td>{{emp.funcao}}</td>

<td>{{emp.salario}}</td>

<td>{{emp.dataAdmissao | date : 'dd/MM/yyyy'}}</td>

<td>

<button (click)="deleteEmpregado(emp)" class="btn btn-primary">Remover</button>

</td>

</tr>

</tbody>

</table>

</div>

</div>

</div>

1. Final:





**Parte 9: Editando um Empregado:**

1. Altere a classe de serviço para poder atualizar e pesquisar um empregado:

updateEmpregado(empregado: Empregado): Observable <Empregado> {

return this.http.put<Empregado>(this.url + '/' + empregado.id, empregado);

}

1. Acrescente os seguintes atributos no componente filho:

empFormLabel = 'Novo Empregado';

btnVisibility = true;

1. Acrescente o decorator @Input para o atributo empregado

@Input()

empregado: Empregado;

1. Na classe do componente filho (empregado-detail), no método ngOnInit, altere o código para ajustar a interface (edição ou novo):

ngOnInit() {

if (this.empregado === null) {

console.log('Novo Empregado');

this.empregado = new Empregado();

this.empFormLabel = 'Novo Empregado';

this.btnVisibility = true;

} else {

this.empFormLabel = 'Alterar Empregado';

this.btnVisibility = false;

}

}

1. Acrescente o método no componente filho para alterar um empregado:

onUpdate() {

this.empService.updateEmpregado(this.empregado).subscribe(

emp => this.emissor.emit()

);

}

1. Altere o template do filho para ajustar os textos relacionados ao novo empregado ou edição de empregado.

<h2 class="text-center">{{empFormLabel}}</h2>

<div class="col-md-6">

<form>

<div class="form-group">

<label for="empId">Empregado Id:</label>

<input type="number" [(ngModel)]="empregado.id" placeholder="Id" name="empId" class="form-control" id="empId">

</div>

<div class="form-group">

<label for="empName">Nome do Empregado:</label>

<input [(ngModel)]="empregado.nome" placeholder="Nome do Empregado" name="empNome" class="form-control" id="empNome">

</div>

<div class="form-group">

<label for="empFuncao">Função do Empregado:</label>

<input [(ngModel)]="empregado.funcao" placeholder="Funcao do Empregado" name="empFuncao" class="form-control" id="empFuncao">

</div>

<div class="form-group">

<label for="empSalario">Salário do Empregado:</label>

<input type="number" [(ngModel)]="empregado.salario" placeholder="0.00" name="empSalario" class="form-control" id="empSalario" >

</div>

<div class="form-group">

<label for="empDataAdm">Data de Admissão:</label>

<input type="date" [(ngModel)]="empregado.dataAdmissao" placeholder="dd/mm/aaaa" name="empDataAdm" class="form-control" id="empDataAdm" >

</div>

<button type="button" class="btn btn-success" \*ngIf="btnVisibility" (click)="onSubmit()">Save</button>

<button type="button" class="btn btn-success" \*ngIf="!btnVisibility" (click)="onUpdate()">Update</button>

</form>

</div>

1. Altere o template do componente pai para fazer o Input do empregado atual no componente filho:

<app-empregado-detail [empregado]="empregadoAtual" (emissor) = "empregadoEvent($event)" \*ngIf="showForm == true">

</app-empregado-detail>

1. Altere o template do componente pai e acrescente o código para editar um empregado:

<button type="button" class="btn btn-success" \*ngIf="btnVisibility" (click)="onSubmit()">Save</button>

<button type="button" class="btn btn-success" \*ngIf="!btnVisibility" (click)="onUpdate()">Update</button>

1. Crie um novo atributo para guardar o empregado que será editado no componente pai:

empregadoAtual: Empregado;

1. Crie o método editEmpregado na classe do componente empregados:

editEmpregado(empregado: Empregado) {

this.empregadoAtual = empregado;

this.showForm = true;

}

1. Altere o método de evento no componente pai:

// Recebe um evento de confirmacao do filho de um novo ou alteracao de empregado

empregadoEvent(empregado) {

this.showForm = false;

if (empregado != null) {

this.empregados.push(empregado);

}

}