**PROJETO FULLSTACK APP DE LOGIN ANGULAR 17 + JAVA SPRING | FRONTEND**

Para ver a versão do Node instalado = **node —version**

Para instalar o angular na versão que te deseja = npm install -g @angular/cli@17

Damos um nome para a aplicação = “minha apliacação”

**Server-Side Rendering (SSR)** = serve para ver quais componentes quer que seja renderizado primeiro

Para entrar na página que fez as instalações = cd minha aplicação, ou seja, cd = para entrar, e o nome da aplicação

Para acessar pelo terminar o vsCode = **code .**

Para iniciar a aplicação =

**Limpar componentes**

**Dentro de src/ app**, temos alguns componentes principais, que foram criados quando criamos a aplicação

No nosso **app.component.html**, vamos tirar tudo que tem dentro dele (style e conteúdo html), deixamos somente a <**router-outlet /**> que exibe de acordo com a rota que o usuário esta navegando

**Para adicionar mais um componente:**

Adicionamos mais um terminal, colocando o seguinte código

**ng g c components/default-login-laout**, **simplificando,**

**ng generate component components/default-login-laou**t, **ou seja,**

rodamos o comandando ng geramos ele na pasta component, dizemos que ele é um componente e damos um nome a ela

**OBS: CASO DE ALGUM PROBLEMA AO EXECUTAR ESTE COMANDO, TROCAR O POWERSHELL PARA O CMD, POIS TEM ALGUMAS RESTRIÇÕES !**

**Para vincular rotas, como por exemplo, a nossa tela de login, com a default-login:**

Passamos então dentro do import da nossa tela de login, a nossa defaultLogin (login.component.ts)

E no login.component.html, passamos o valor dela, no caso (app-default-login-layout)

Vamos então dentro do **app.routes.ts:** onde vamos passar o caminho de como deve seguir a aplicação, **por exemplo:**

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

import { LoginComponent } from './pages/login/login.component';

export const routes: Routes = [

{

path: "login",

component: LoginComponent

}

];

Onde passamos como deve ser chamado no navegador, passando o componente inicial, onde o LoginComponent ja tem a tela inicial/default mais os componentes que foi colocado nela

**Como declarar variáveis no .scss:**

$nome-variavel: propriedade;

**Para importar algo, por exemplo o nosso scss:**

@import “PASSA O CAMINHO”;

E quando for utilizar a variável, colocar $ na frente, **por exemplo,** $gray-bg

**Para chamarmos algum valor dentro do nosso html:**

Entramos dentro do nosso **component.ts**, e importamos ele, **por exemplo:**

No **default-login-layout.component.html:** colocamos no nosso h2:

\*\*<h2>{{ title }}</h2>\*\*

Então para chamarmos ele, dentro do nosso **default-login-layout.component.html,** colocamos o seguinte:

export class DefaultLoginLayoutComponent {

@Input() title: string = "";

}

Colocamos a anotação @input, dando o nome pra ele, dizendo que é uma string e dando o valor dele como “”;

Daí na nossa página de **logn.component.html,** nos passamos os valores:

<app-default-login-layout

title="Login int your account"

primaryBtnText="Login Now"

secondaryBtnText="Signup Now"

>

</app-default-login-layout>

**Funcionalidade <NG-CONTENT>**

<ng-content></ng-content>

<app-default-login-layout

title="Login int your account"

primaryBtnText="Login Now"

secondaryBtnText="Signup Now"

>

<form action="">

<p>Ola mundo</p>

</form>

</app-default-login-layout>

Tudo que for passado ali dentro do **form,** vai entrar para dentro do

**Para colocarmos algo dentro também do nosso formulário:**

Devemos ir no nosso **login.component.ts:**

@Component({

selector: 'app-login',

standalone: true,

imports: [

DefaultLoginLayoutComponent,

ReactiveFormsModule

],

templateUrl: './login.component.html',

styleUrl: './login.component.scss'

})

E dentro dos nossos imports, vamos atribuir a ela o **ReactiveFormsModule**

export class LoginComponent {

loginForm!: FormGroup;

constructor(){

}

Dentro da nossa classe **LoginComponent,** vamos declarar ele como **loginForm,** do tipo **FormGroup,** e o **!** significa que vai ser declarado algum momento, onde ele começa com vazio mas dentro do construtor eu vou declarar ele

constructor(){

this.loginForm = new FormGroup({

email: new FormControl(Validators.required, Validators.email),

password: new FormControl(Validators.required, Validators.minLength(6))

})

}

Dentro do nosso construtor nós vamos chamar este **loginForm**, passando quais atributos ele terá, passando então o email e a senha, dizemos então que ele vai ser um **FromControl,** e vai ter validadores:

**Validator.requerid =** campo que precisa ser preenchido

**Validators . email =** campo que precisa ter as validações de email, @ .com

Validators.minLenght(6) = campo que precisa ter no minimo 6 caracteres

**Criar inputs do que quiser**

Dentro da nossa **primary-input.components.ts =** vamos colocar em cima do @Component:

type InputTypes = "text" | "email" | "password”

Onde falamos que quermos um type com tal nome, que pode ter esses 3 valores

Então dentro da classe **PrimaryInputComponent,** podemos chamar ela, dando diferentes valores

export class PrimaryInputComponent {

@Input() type: InputTypes = "text";

}

Então, lá no nosso html do nosso componente criado, chamamos ele da seguinte forma:

<input [type]="type">

Chamamos ele através da propriedade input, entre colchetes falamos que ele é um tipo, e dentro do =””, chamamos o nome que chamamos ali no nosso **primary-input.components.ts**

**Erro que normalmente da, de colocar uma label, e ela so abre depois de ficar clicando algumas vezes nele:**

Dentro do nosso **primary-input.component.ts** dentro da nossa classe vamos por um:

**implements ControlValueAccessor**

Após isso, em baixo dos imports, vamos criar os **providers**

providers: [

{

provide: NG\_VALUE\_ACCESSOR,

useExisting: forwardRef(() => PrimaryInputComponent),

multi: true

}

],

Para isso, precisamos também passar alguns métodos (implementar)

value: string = ''

onChange: any = () => {}

onTouched: any = () => {}

Salvamos o valor do input em uma variável do tipo input

Criamos o onChange, que recebe nada e retorna nada (sobre mudança)

Criamos o onTouched, que recebe nada e retorna nada (sobre tocado)

**Criar forma para submit, de enviar para o back**

@Output("submit") onSubmit = new EventEmitter();

Criamos então uma função quer quando chamada ira emitir o valor

**Ou seja,** Quando o botão for clicado o onSubmit vai ser escutado

Temos que colocar então esse submit dentro de uma função chamada **(click)= “submit”**

E temos que criar ela dentro do nosso construtor

E nessa função que colocamos no construtor vai ser a seguinte:

submit() {

console.log(this.loginForm.value)

}

**Para criarmos um SERVIÇO / SERVICE**

Entramos no terminal e colocamos:

ng g s service/login

**Ou seja,**

ng generate service nome da pasta/nome-aplicação,

Onde criamos um serviço para pegar os dados do backend e salvar o token ali na sessão do usuário

constructor(private httpClient: HttpClient) { }

Criamos tambem uma função onde o backend vai nos retornar alguns dados

login (name: string, password: string) {

return this.httpClient.post("/login", { name, password })

}

**Ou seja,**

Criamos esta função passando dois parâmetros, onde passamos a url do servidor, (não passamos ali, pois AINDA não temos um back), e colocamos um body passando o o nome e a senha

**Com isso,** vai ser retornado um token, que vai ser para as próximas requisições dentro da aplicação, de forma autenticada