**Etapa 5 do trabalho final web 2**

1. Para validar os dados recebidos no formulário foi feito uma alteração do código do arquivo app.component.html:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Document</title>

</head>

<body>

  <!-- COLETANDO DADOS DO BACK -->

  <div \*ngIf="data" id="mostraObjeto" class="mostraObjeto">

    <p>Nome: {{ data.Pessoa1.Nome}}</p>

    <p>Idade: {{ data.Pessoa1.Idade }}</p>

  </div>

  <div>

    <form #form="ngForm" (ngSubmit)="enviarDados()" novalidate>

      <label for="texto">Nome: </label>

      <input type="text" id="texto" [(ngModel)]="nome" name="nome" required minlength="2" maxlength="255" #nomeRef="ngModel">

      <span \*ngIf="nomeRef.invalid && nomeRef.touched" style="color: red;">O nome é obrigatório e deve ter entre 2 e 255 caracteres.</span>

      <br>

      <label for="inteiro">Idade: </label>

      <input type="number" id="inteiro" [(ngModel)]="idade" name="idade" required #idadeRef="ngModel" min="1" max="999">

      <span \*ngIf="idadeRef.invalid && idadeRef.touched" style="color:red;">A idade deve ser um número entre 1 e 999.</span>

      <br>

      <!-- Para o checkbox, você pode querer validar se foi marcado, mas isso depende da sua lógica de negócio -->

      <label for="brasil">Nacionalidade brasileira: </label>

      <input type="checkbox" id="brasil" [(ngModel)]="nacionalidadeBrasileira" name="nacionalidadeBrasileira" required #nacionalidadeRef="ngModel">

      <span \*ngIf="nacionalidadeRef.invalid && nacionalidadeRef.touched" style="color:red;">A nacionalidade é obrigatória.</span>

      <br>

      <label for="sexo">Sexo: </label>

      <input type="radio" id="masculino" name="sexo" value="masculino" [(ngModel)]="sexo" required #sexoRef="ngModel">

      <label for="masculino">Masculino</label>

      <input type="radio" id="feminino" name="sexo" value="feminino" [(ngModel)]="sexo" required #sexoRef="ngModel">

      <label for="feminino">Feminino</label>

      <span \*ngIf="sexoRef.invalid && sexoRef.touched" style="color:red;">A seleção de sexo é obrigatória.</span>

      <select name="cor" [(ngModel)]="cor" required #corRef="ngModel">

        <option value="" disabled>Selecione uma cor</option>

        <option value="azul">Azul</option>

        <option value="vermelho">Vermelho</option>

        <option value="rosa">Rosa</option>

      </select>

      <span \*ngIf="corRef.invalid && corRef.touched" style="color:red;">A seleção de cor é obrigatória.</span>

      <button class="filter-button" type="submit" [disabled]="form.invalid">Enviar</button>

    </form>

  </div>

</body>

</html>

Agora o formulário não envia para o back sem antes ter validado no front, apresentando mensagens pro usuário preencher os campos obrigatórios.

### **Passo 1: Estrutura HTML**

Primeiro, vamos garantir que seu HTML está configurado para mostrar as mensagens de validação, observe que o span está vazio para não mostrar nenhuma mensagem ao início da página além de cada input ter um span:

<form *id*="filterForm" *class*="filter-container" *(submit)*="enviarDados()">

<label *for*="texto">Nome: </label>

<input *class*="filter-input" *type*="text" *id*="texto" *[(ngModel)]*="nome" *name*="nome" *placeholder*="">

<span *id*="txtError" *style*="color: red;"></span>

<br>

<label *for*="inteiro">Idade: </label>

<input *class*="filter-input" *type*="number" *id*="inteiro" *[(ngModel)]*="idade" *name*="idade">

<span *id*="nmbrError" *style*="color:red;"></span>

<br>

<label *for*="brasil">Nacionalidade brasileira: </label>

<input *class*="filter-checkbox" *type*="checkbox" *id*="brasil" *[(ngModel)]*="nacionalidadeBrasileira" *name*="nacionalidadeBrasileira"> Sim

<span *id*="boolError" *style*="color:red;"></span>

<br>

<label *for*="sexo">Sexo: </label>

<input *type*="radio" *id*="masculino" *[(ngModel)]*="sexo" *name*="sexo" *value*="Masculino">Masculino

<input *type*="radio" *id*="feminino" *[(ngModel)]*="sexo" *name*="sexo" *value*="Feminino">Feminino

<!-- <input type="radio" id="Non-Binario" [(ngModel)]="sexo" name="sexo" value="Non-Binario">Non-Binario

<input type="radio" id="Homoafetivo" [(ngModel)]="sexo" name="sexo" value="Homoafetivo">Homoafetivo

<input type="radio" id="Outros" [(ngModel)]="sexo" name="sexo" value="Outros">Outros -->

<span *id*="sxError" *style*="color:red;"></span>

<br>

<select *name*="cor" *[(ngModel)]*="cor">

<option *value*="azul">Azul</option>

<option *value*="vermelho" *selected*>Vermelho</option>

<option *value*="rosa">Rosa</option>

</select>

<span *id*="corError" *style*="color:red;"></span>

<button *class*="filter-button" *type*="submit">Enviar</button>

</form>

### **Passo 2: Criação da Interface no Frontend**

No seu frontend, faça a criação de uma interface sttsFormulario, iremos precisar dela no arquivo data.service.ts e também no app.component.ts.

1. data.service.ts

interface *ErrStts* {

txtErr: *boolean*;

nmbrErr: *boolean*;

boolErr: *boolean*;

sxErr: *boolean*;

corErr: *boolean*;

}

interface *PostResponse*{

sttsForms: *ErrStts*;

}

enviarDado(*dado*: *any*) {

const url = 'http://localhost:8000/formulario';

return *this*.http.post<*PostResponse*>(url, *dado*);

}

Aqui estamos utilizando a propria função de enviar dados do formulário como meio de comunicação para o backend, porém o retorno do back será um objeto json.

### **Passo 3: Função de Validação no Backend**

Agora, vamos criar a função que valida os dados no backend FastAPI, as alterações a seguir foram feitas no main.py:

@app.post("/formulario")

def receberDadosDoForm(*dado*: Formulario):

print(*dado*.\_\_str\_\_)

\_errs = validarFormulario(*dado*)

print(\_errs)

if(\_errs):

# se tiver erro retorno o objeto de erros e paro o processamento por aqui

return \_errs

# tudo ok posso tratar os dados a partir daqui

print("Sucess")

return "Sucess"

def validarFormulario(*dado*: Formulario)-> any:

# posição de erros

\_errs = []

# verificando o texto obrigatorio de min 2 carac e max 255 carac

if len(*dado*.nome) < 2 or len(*dado*.nome) > 255:

# \_errs.append("O texto deve ter no minimo 2 e no máximo 255 caracteres!")

\_errs.append(True)

else:

\_errs.append(False)

if *dado*.idade < 1 or *dado*.idade > 1000:

# \_errs.append("O valor deve ser maior que 0 e menor que 1000!")

\_errs.append(True)

else:

\_errs.append(False)

if *dado*.nacionalidadeBrasileira is None or *dado*.nacionalidadeBrasileira =="":

# \_errs.append("Você deve responder sua nacionalidade!")

\_errs.append(True)

else:

\_errs.append(False)

if *dado*.sexo is None or *dado*.sexo == "":

# \_errs.append("Sexo obrigatório")

\_errs.append(True)

else:

\_errs.append(False)

if *dado*.cor is None or *dado*.cor == "":

# \_errs.append("Selecione uma cor!")

\_errs.append(True)

else:

\_errs.append(False)

return {"sttsForms":{"txtErr":\_errs[0],"nmbrErr":\_errs[1],"boolErr":\_errs[2],"sxErr":\_errs[3],"corErr":\_errs[4]}}

### **Passo 4: Rota de Envio de Dados**

Agora, vamos modificar a rota que recebe os dados do formulário para usar a validação, as alterações foram feitas em app.component.ts, lembre-se de inserir a interface logo após os imports:

interface *ErrStts* {

txtErr: *boolean*;

nmbrErr: *boolean*;

boolErr: *boolean*;

sxErr: *boolean*;

corErr: *boolean*;

}

interface *PostResponse*{

sttsForms: *ErrStts*;

}

enviarDados() {

const dadosFormulario = {

nome: *this*.nome,

idade: *this*.idade,

nacionalidadeBrasileira: *this*.nacionalidadeBrasileira,

sexo: *this*.sexo,

cor: *this*.cor

};

// OBTENDO MENSAGENS DE SPAN ASSOCIADAS NA PAGINA PARA MANIPULAR ELAS

const spanTxt = document.getElementById("txtError") as *HTMLSpanElement*;

const spanNmb = document.getElementById("nmbrError") as *HTMLSpanElement*;

const spanBool = document.getElementById("boolError") as *HTMLSpanElement*;

const spanSx = document.getElementById("sxError") as *HTMLSpanElement*;

const spanCor = document.getElementById("corError") as *HTMLSpanElement*;

// PROCESSANDO DADOS

*this*.dataService.enviarDado(dadosFormulario).subscribe(

(*response*: *PostResponse*) => {

if(*response*.sttsForms.boolErr || *response*.sttsForms.corErr || *response*.sttsForms.nmbrErr || *response*.sttsForms.sxErr || *response*.sttsForms.txtErr){

// recebendo a resposta do metodo post e verificando se ele retornou alguma mensagem de erro

console.log(*response*.sttsForms)

// TRATANDO TEXTO

if(*response*.sttsForms.txtErr)

spanTxt.textContent = "O texto deve ter no minimo 2 e no máximo 255 caracteres!";

else

spanTxt.textContent = "";

// TRATANDO NUMERO

if(*response*.sttsForms.nmbrErr)

spanNmb.textContent = "O valor deve ser maior que 0 e menor que 1000!";

else

spanNmb.textContent = "";

// TRATANDO BOOL

if(*response*.sttsForms.boolErr)

spanBool.textContent = "Você deve responder sua nacionalidade!";

else

spanBool.textContent = "";

// TRATANDO CHECKBOX

if(*response*.sttsForms.sxErr)

spanSx.textContent = "Sexo obrigatório";

else

spanSx.textContent = "";

// TRATANDO RADIOBUTTONS

if(*response*.sttsForms.corErr)

spanCor.textContent = "Selecione uma cor!";

else

spanCor.textContent = "";

}else{

spanTxt.textContent = "";

spanNmb.textContent = "";

spanBool.textContent = "";

spanSx.textContent = "";

spanCor.textContent = "";

alert("Dados enviados com")

}

},

*error* => {

console.error('Erro ao enviar dados:', *error*);

});

}

Esse código assume que você tem um span com id correspondente para cada campo de entrada (por exemplo, erroNome para o campo nome).

Com essas etapas, você tem uma validação de formulário básica no backend que interage com o frontend, exibindo mensagens de erro ao usuário conforme necessário. Lembre-se de adaptar o código conforme o contexto específico da sua aplicação.