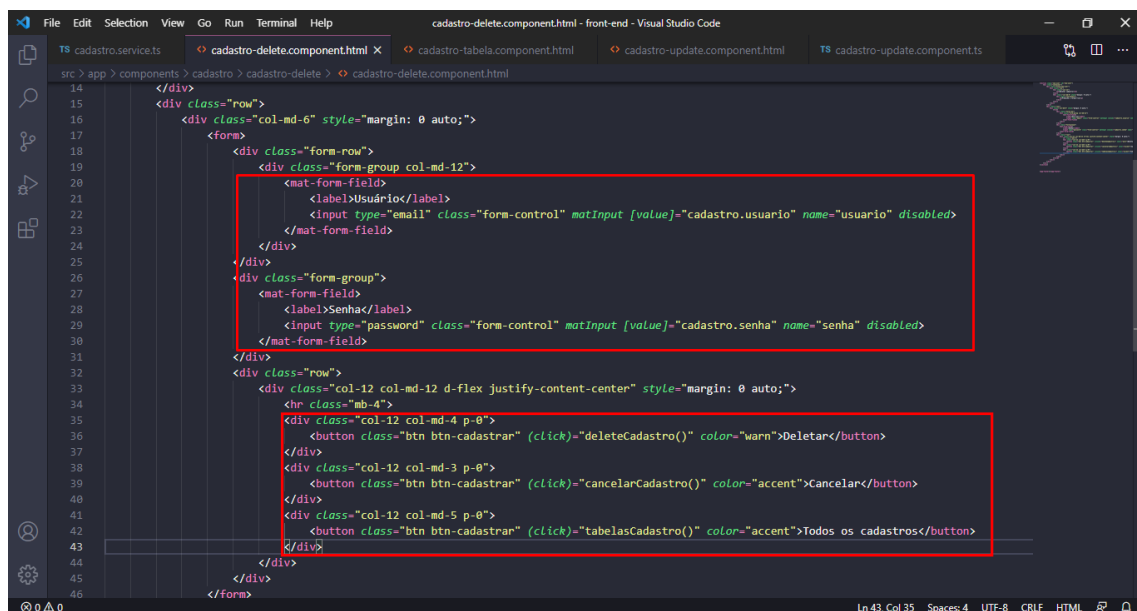


Cadastro *delete*

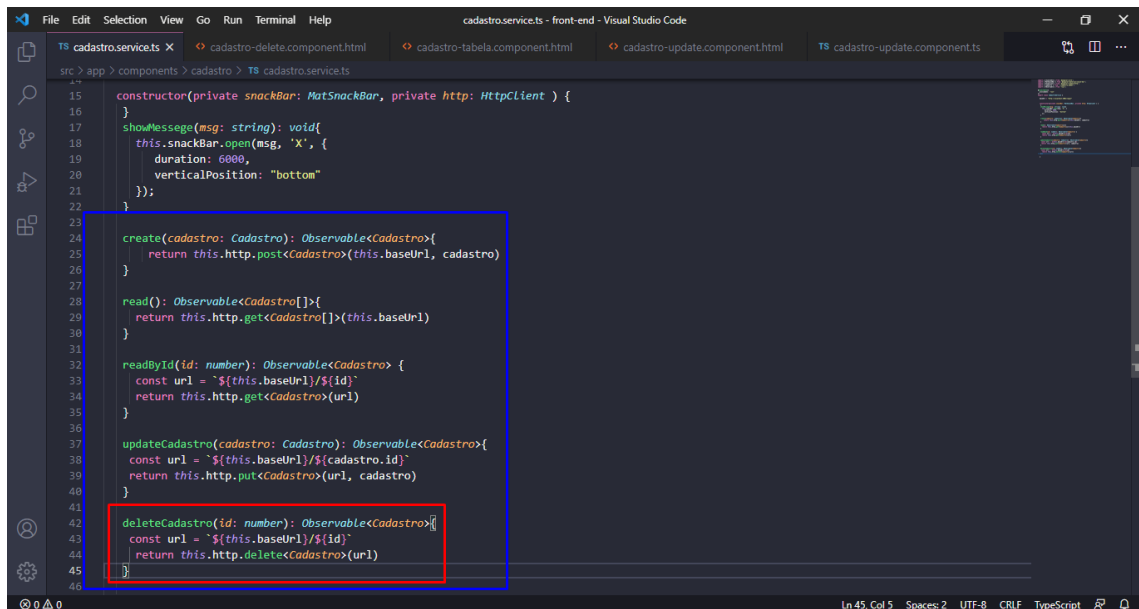
1. Acesse o arquivo **cadastro-delete.component.html**. Você terá acesso à seguinte estrutura pronta:



```
14 </div>
15 <div class="row">
16 <div class="col-md-6" style="margin: 0 auto;">
17 <form>
18 <div class="form-row">
19 <div class="form-group col-md-12">
20 <mat-form-field>
21 <label>Usuário</label>
22 <input type="email" class="form-control" matInput [value]="cadastro.usuario" name="usuario" disabled>
23 </mat-form-field>
24 </div>
25 </div>
26 <div class="form-group">
27 <mat-form-field>
28 <label>Senha</label>
29 <input type="password" class="form-control" matInput [value]="cadastro.senha" name="senha" disabled>
30 </mat-form-field>
31 </div>
32 </div>
33 <div class="row">
34 <div class="col-12 col-md-12 d-flex justify-content-center" style="margin: 0 auto;">
35 <div class="mb-4">
36 <div class="col-12 col-md-4 p-0">
37 <button class="btn btn-cadastrar" (click)="deleteCadastro()" color="warn">Deletar</button>
38 </div>
39 <div class="col-12 col-md-3 p-0">
40 <button class="btn btn-cadastrar" (click)="cancelarCadastro()" color="accent">Cancelar</button>
41 </div>
42 <div class="col-12 col-md-5 p-0">
43 <button class="btn btn-cadastrar" (click)="tabelasCadastro()" color="accent">Todos os cadastros</button>
44 </div>
45 </div>
46 </div>
</form>
</div>
```

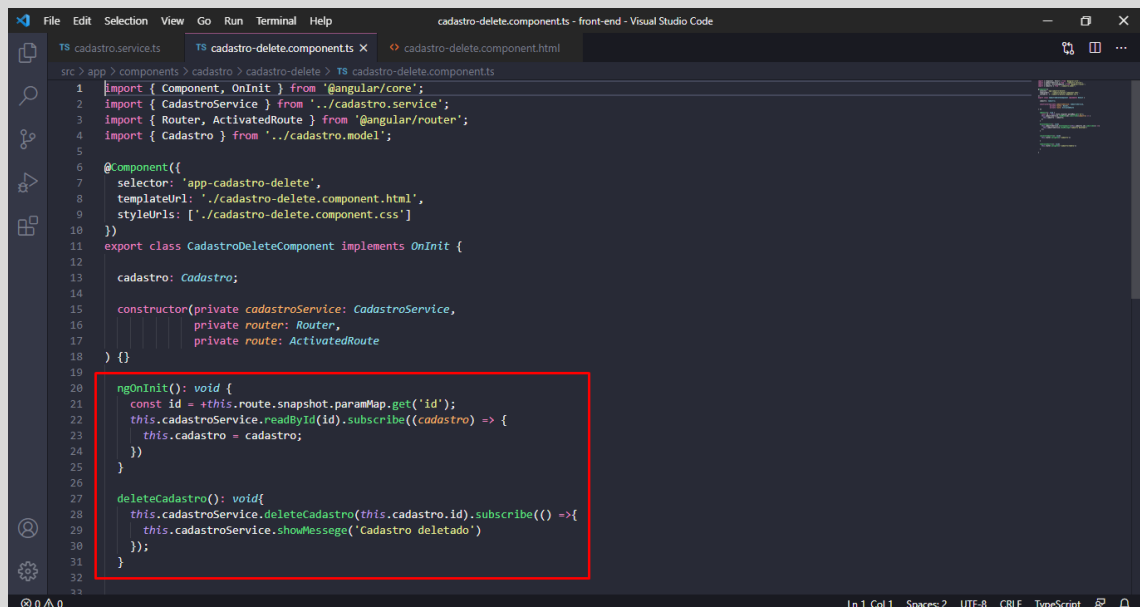
Perceba que os *inputs* estão desabilitados nessa página, o que significa que o usuário não pode inserir dados. Conforme podemos verificar na linha 36, está previsto um clique para o botão deletar, o que confirma a exclusão (*delete*). O método de funcionamento desse botão está sendo configurado no **cadastro.service.ts**. Observe:

Interação com APIs



```
15 constructor(private snackBar: MatSnackBar, private http: HttpClient) {
16 }
17 showMessage(msg: string): void {
18   this.snackBar.open(msg, 'X', {
19     duration: 6000,
20     verticalPosition: 'bottom'
21   });
22 }
23
24 create(cadastro: Cadastro): Observable<Cadastro> {
25   return this.http.post<Cadastro>(this.baseUrl, cadastro)
26 }
27
28 read(): Observable<Cadastro[]> {
29   return this.http.get<Cadastro[]>(this.baseUrl)
30 }
31
32 readById(id: number): Observable<Cadastro> {
33   const url = `${this.baseUrl}/${id}`
34   return this.http.get<Cadastro>(url)
35 }
36
37 updateCadastro(cadastro: Cadastro): Observable<Cadastro> {
38   const url = `${this.baseUrl}/${cadastro.id}`
39   return this.http.put<Cadastro>(url, cadastro)
40 }
41
42 deleteCadastro(id: number): Observable<Cadastro> {
43   const url = `${this.baseUrl}/${id}`
44   return this.http.delete<Cadastro>(url)
45 }
```

2. Quando vamos excluir um usuário, devemos passar o id. Logo após configurar o *service*, adicione o código ao seu arquivo **cadastro-delete.component.ts**, conforme mostra a figura a seguir:



```
1 import { Component, OnInit } from '@angular/core';
2 import { CadastroService } from '../cadastro.service';
3 import { Router, ActivatedRoute } from '@angular/router';
4 import { Cadastro } from '../cadastro.model';
5
6 @Component({
7   selector: 'app-cadastro-delete',
8   templateUrl: './cadastro-delete.component.html',
9   styleUrls: ['./cadastro-delete.component.css']
10 })
11 export class CadastroDeleteComponent implements OnInit {
12   cadastro: Cadastro;
13
14   constructor(private cadastroService: CadastroService,
15     private router: Router,
16     private route: ActivatedRoute
17   ) {}
18
19   ngOnInit(): void {
20     const id = +this.route.snapshot.paramMap.get('id');
21     this.cadastroService.readById(id).subscribe((cadastro) => {
22       this.cadastro = cadastro;
23     });
24   }
25
26   deleteCadastro(): void {
27     this.cadastroService.deleteCadastro(this.cadastro.id).subscribe(() => {
28       this.cadastroService.showMessage('Cadastro deletado')
29     });
30   }
31 }
```

3. Salve todos os arquivos e faça teste em sua máquina.