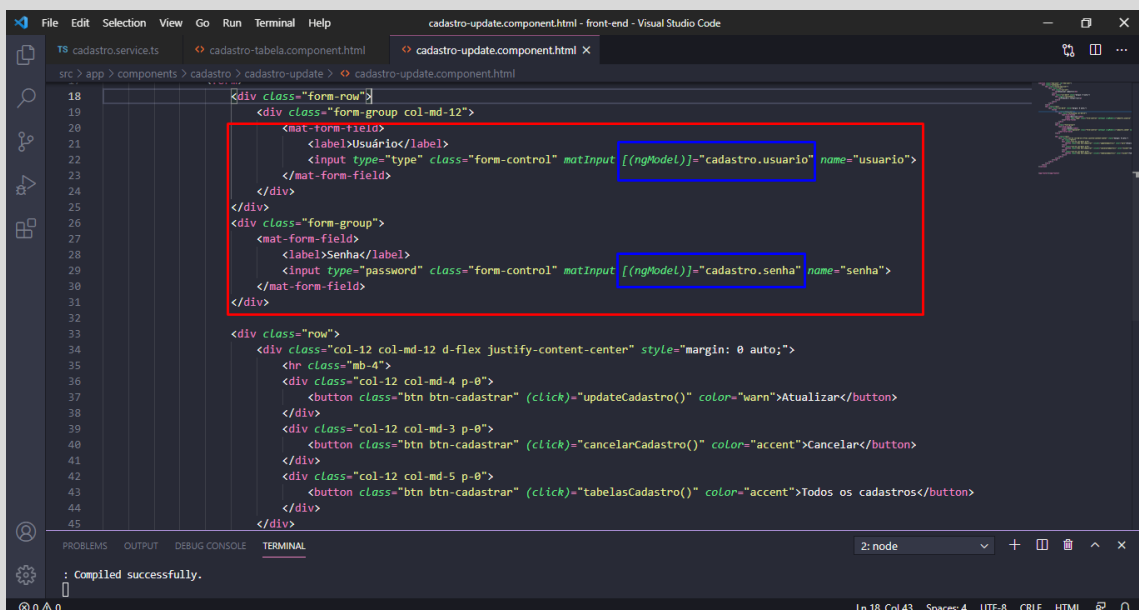


Cadastro *update*

1. Acesse a pasta **cadastro-update**. Existe a opção de editar na tabela em que estão listados todos os dados. O processo de atualização consiste em alterar um dado que já está salvo, sobrescrevendo-o.
2. Acesse o arquivo **cadastro-update.component.html** e observe a sintaxe do código. Na imagem abaixo, analise as linhas 22 e 29, nas quais estamos passando o **[(ngModel)]** para substituir o valor dos atributos.



```
18 <div class="form-row">
19   <div class="form-group col-md-12">
20     <mat-form-field>
21       <label>Usuário</label>
22       <input type="text" class="form-control" matInput [(ngModel)]="cadastro.usuario" name="usuario">
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 [(ngModel)]="cadastro.senha" name="senha">
30   </mat-form-field>
31 </div>
32
33 <div class="row">
34   <div class="col-12 col-md-12 d-flex justify-content-center" style="margin: 0 auto;">
35     <hr class="mb-4">
36     <div class="col-12 col-md-4 p-0">
37       <button class="btn btn-cadastrar" (click)="updateCadastro()" color="warn">Atualizar</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>
```

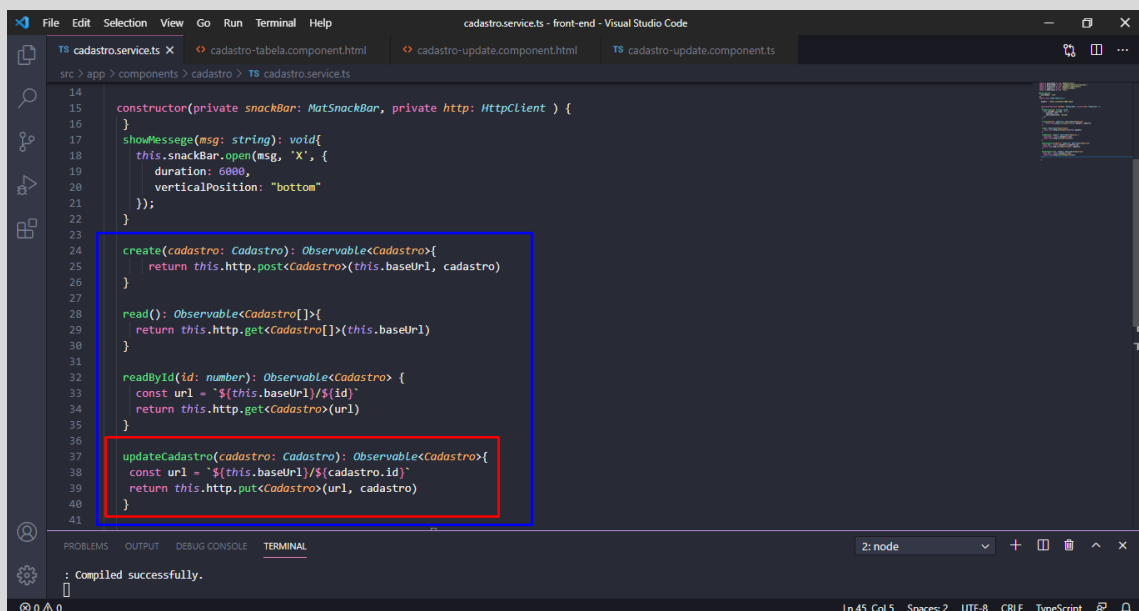
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

2: node

Ln 18, Col 43 Spaces: 4 UTF-8 CRLF HTML

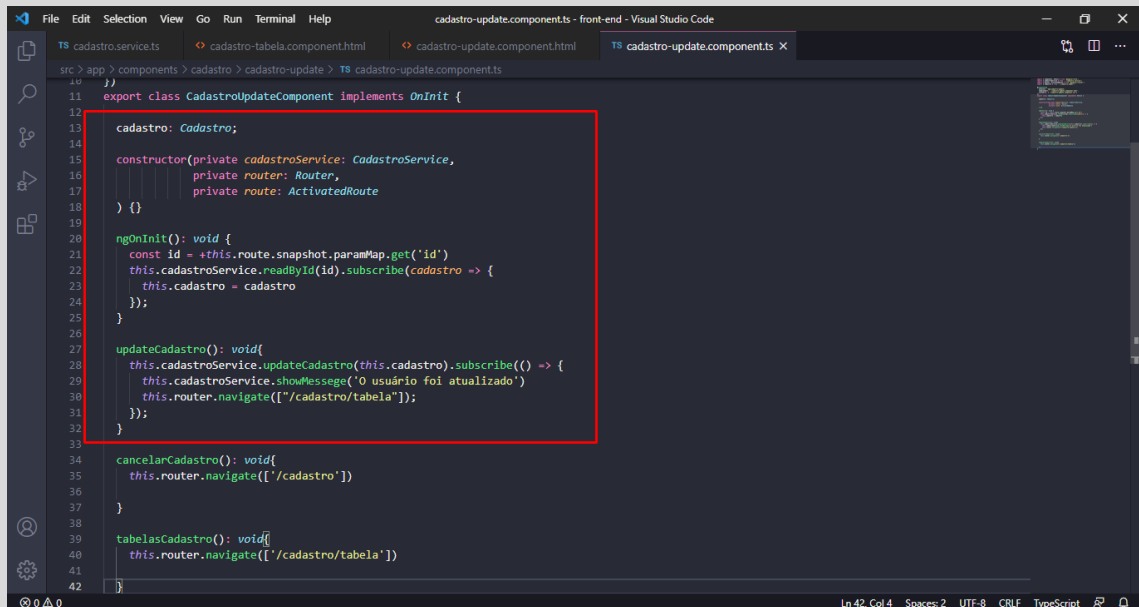
3. No arquivo **cadastro.service.ts** foram adicionadas as linhas de 37 a 40.

A linha 37 funciona de modo semelhante ao método de exclusão, pois ambos afetam apenas um único usuário, uma vez que as alterações estão sendo feitas no id (lembre-se de que o id é único).



```
14
15
16
17 showMessege(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
```

4. Feito isso, no arquivo **cadastro-update.component.ts**, faça conforme destacado na imagem. As linhas de 34 a 40 fazem referência aos botões de voltar e cadastrar.



```
11 export class CadastroUpdateComponent implements OnInit {
12
13   cadastro: Cadastro;
14
15   constructor(private cadastroService: CadastroService,
16               private router: Router,
17               private route: ActivatedRoute) {}
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   updateCadastro(): void {
27     this.cadastroService.updateCadastro(this.cadastro).subscribe(() => {
28       this.cadastroService.showMessage("O usuário foi atualizado")
29       this.router.navigate(["/cadastro/tabela"]);
30     });
31   }
32
33   cancelarCadastro(): void {
34     this.router.navigate(['/cadastro'])
35   }
36
37   tabelasCadastro(): void {
38     this.router.navigate(['/cadastro/tabela'])
39   }
40 }
```