

Consulta de dados – READ

Agora, estudaremos um exemplo de como criar uma requisição para consulta de dados (READ).

1. No arquivo **cadastro-read.component.html**, localizado dentro da pasta **Front-com-angular\src\app\components\cadastro\cadastro-read**, iremos inserir o atributo ***matCellDef="let row":{{row.usuario}}**.

```

1 <div class="container">
2   <div class="row">
3     <div class="col-12">
4       <div class="mat-elevation-z8">
5         <table mat-table class="full-width-table" matSort aria-label="Elements" [dataSource]="cadastros">
6
7           <ng-container matColumnDef="id">
8             <th mat-header-cell *matHeaderCellDef mat-sort-header>Id</th>
9             <td mat-cell *matCellDef="let row">{{row.id}}</td>
10          </ng-container>
11
12          <ng-container matColumnDef="usuario">
13            <th mat-header-cell *matHeaderCellDef mat-sort-header>Usuário</th>
14            <td mat-cell *matCellDef="let row">{{row.usuario}}</td>
15          </ng-container>
16
17          <ng-container matColumnDef="senha">
18            <th mat-header-cell *matHeaderCellDef mat-sort-header>Senha</th>
19            <td mat-cell *matCellDef="let row">{{row.senha}}</td>
20          </ng-container>
21
22          <ng-container matColumnDef="ações">
23            <th mat-header-cell *matHeaderCellDef mat-sort-header>Ações</th>
24            <td mat-cell *matCellDef="let row">
25              <a routerLink="/cadastro/update/{{row.id}}">
26                <i class="material-icons">edit</i>
27              </a>
28              <a routerLink="/cadastro/delete/{{row.id}}">

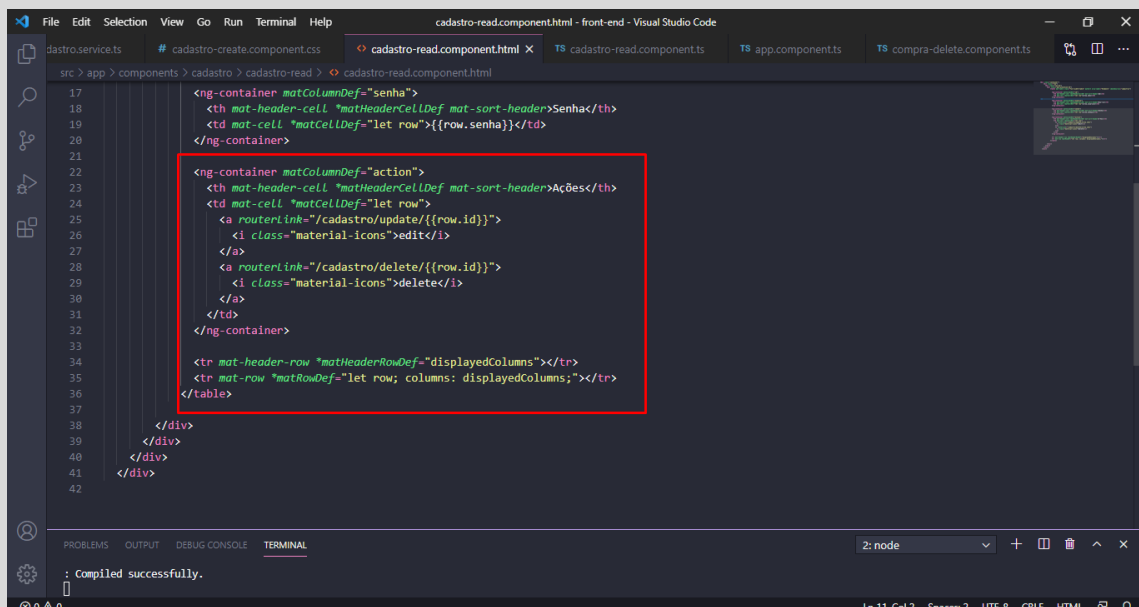
```

No Angular, um valor entre duas chaves representa uma variável, ou seja, um valor do TypeScript apresentado no HTML. O atributo **matCellDef** é utilizado para capturar o valor da tabela (na linha 8, é o nome das colunas e, na linha 9, o valor).

Na linha 22, há uma ação para os botões, ou seja, quando o usuário clicar em um botão para editar ou excluir dados no Front-End, uma outra ação será gerada no Back-End.

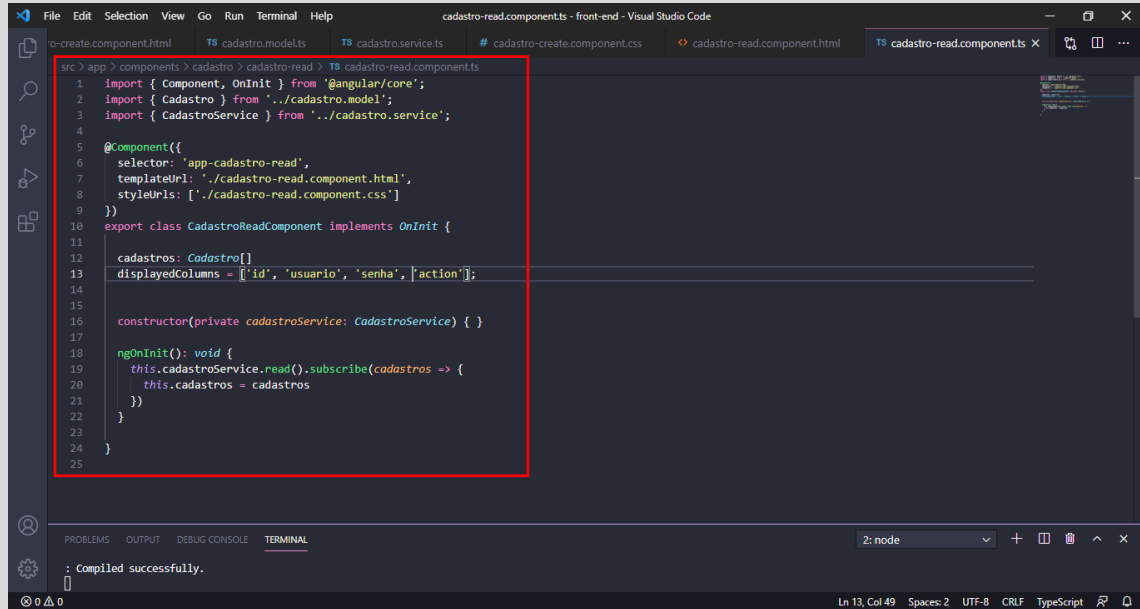
Perceba que, nas linhas 25 e 28, passamos o id específico que queremos editar. Na linha 25, ele levará à página em que poderemos editar e, também, deletar o id.

Testando pelo Postman, ao atualizar (PUT) ou deletar (DELETE) um conjunto de dados específicos, você precisará da URL e do id, por exemplo, `http://localhost:3001/login/3`. No Front-End, por outro lado, o usuário não vê esse processo.



```
17 <ng-container matColumnDef="senha">
18   <th mat-header-cell "matHeaderCellDef mat-sort-header">Senha</th>
19   <td mat-cell "matCellDef" let row">{{row.senha}}</td>
20 </ng-container>
21
22 <ng-container matColumnDef="action">
23   <th mat-header-cell "matHeaderCellDef mat-sort-header">Ações</th>
24   <td mat-cell "matCellDef" let row">
25     <a routerLink="/cadastro/update/{{row.id}}">
26       <i class="material-icons">edit</i>
27     </a>
28     <a routerLink="/cadastro/delete/{{row.id}}">
29       <i class="material-icons">delete</i>
30     </a>
31   </td>
32 </ng-container>
33
34 <tr mat-header-row "matHeaderRowDef"=displayedColumns"></tr>
35 <tr mat-row "matRowDef" let row; columns: displayedColumns;"></tr>
36 </table>
37
38 </div>
39 </div>
40 </div>
41 </div>
42
```

2. No arquivo **cadastro-read.component.ts**, adicione o seguinte código, conforme destacado na imagem:



```

src > app > components > cadastro > cadastro-read > TS cadastro-read.component.ts
1 import { Component, OnInit } from '@angular/core';
2 import { Cadastro } from '../cadastro.model';
3 import { CadastroService } from '../cadastro.service';
4
5 @Component({
6   selector: 'app-cadastro-read',
7   templateUrl: './cadastro-read.component.html',
8   styleUrls: ['./cadastro-read.component.css']
9 })
10 export class CadastroReadComponent implements OnInit {
11
12   cadastros: Cadastro[]
13   displayedColumns = ['id', 'usuario', 'senha', {action: ''}];
14
15   constructor(private cadastroService: CadastroService) { }
16
17   ngOnInit(): void {
18     this.cadastroService.read().subscribe(cadastros => {
19       this.cadastros = cadastros
20     })
21   }
22 }
23
24
25









```

Importante

No Angular, as colunas são passadas pelo arquivo **.ts**, diferentemente do que ocorre no HTML.

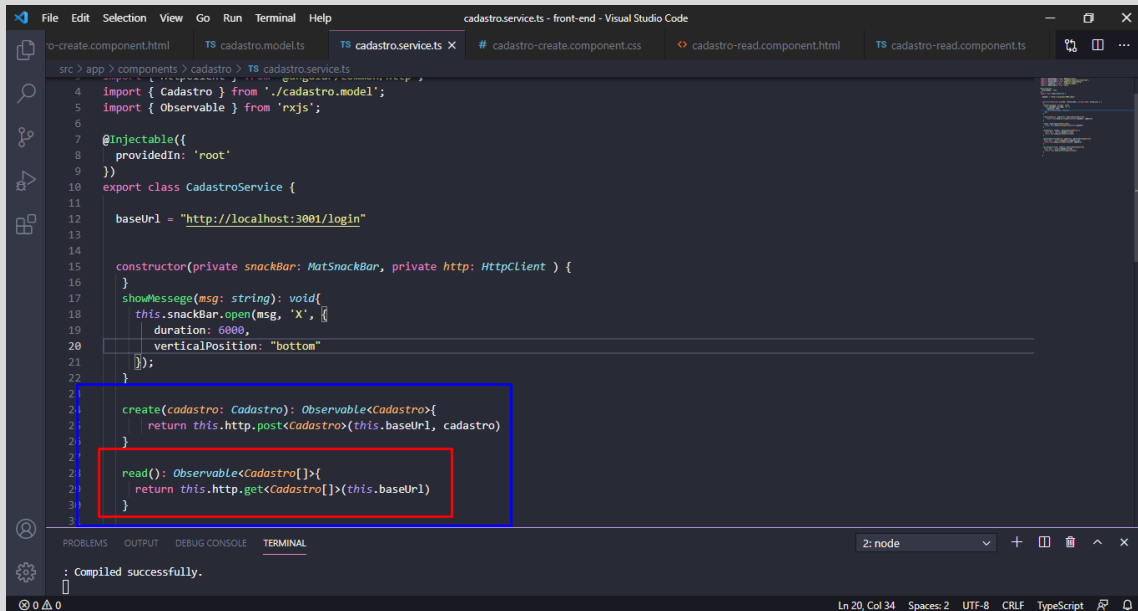


Cada valor (“Id”, “Usuário”, “Senha”, “Ação”) representa uma coluna, como mostrado na imagem:

Id	Usuário	Senha	Ações
1	Vitória Pinho	admin	 
2	Stefany Oliveira	admin123	 
3	Maria da Silva	admin123	 
4	Claudia	admin	 

HOME
CADASTRAR
TODOS OS CADASTROS

3. Em seu arquivo **cadastro.service.ts**, adicione o código a seguir:



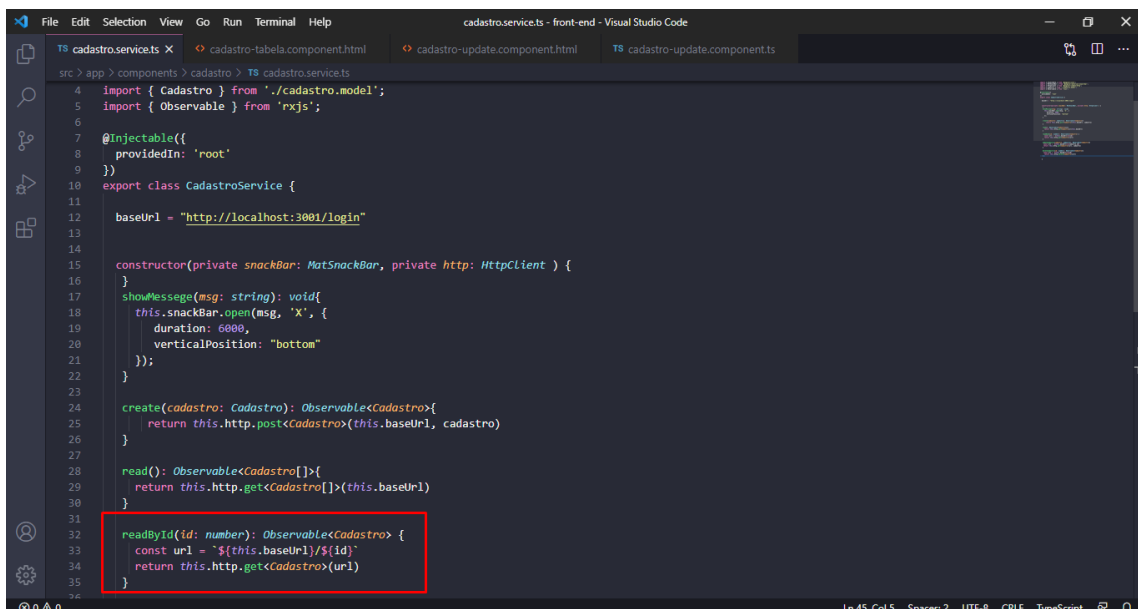
```

src > app > components > cadastro > TS cadastro.service.ts
1 import { Cadastro } from './cadastro.model';
2 import { Observable } from 'rxjs';
3
4 @Injectable({
5   providedIn: 'root'
6 })
7 export class CadastroService {
8
9   baseUrl = "http://localhost:3001/login"
10
11   constructor(private snackBar: MatSnackBar, private http: HttpClient) {
12
13   }
14
15   showMessage(msg: string): void {
16     this.snackBar.open(msg, 'X', {
17       duration: 6000,
18       verticalPosition: "bottom"
19     });
20   }
21
22   create(cadastro: Cadastro): Observable<Cadastro>{
23     return this.http.post<Cadastro>(this.baseUrl, cadastro)
24   }
25
26   read(): Observable<Cadastro[]>{
27     return this.http.get<Cadastro[]>(this.baseUrl)
28   }
29 }

```

Perceba que a diferença entre adicionar (CREATE) e mostrar todos (READ) reside nos métodos utilizados (POST e GET).

4. Caso deseje listar apenas um único usuário, faça conforme mostrado na imagem a seguir e salve seu arquivo.



```

src > app > components > cadastro > TS cadastro.service.ts
1 import { Cadastro } from './cadastro.model';
2 import { Observable } from 'rxjs';
3
4 @Injectable({
5   providedIn: 'root'
6 })
7 export class CadastroService {
8
9   baseUrl = "http://localhost:3001/login"
10
11   constructor(private snackBar: MatSnackBar, private http: HttpClient) {
12
13   }
14
15   showMessage(msg: string): void {
16     this.snackBar.open(msg, 'X', {
17       duration: 6000,
18       verticalPosition: "bottom"
19     });
20   }
21
22   create(cadastro: Cadastro): Observable<Cadastro>{
23     return this.http.post<Cadastro>(this.baseUrl, cadastro)
24   }
25
26   read(): Observable<Cadastro[]>{
27     return this.http.get<Cadastro[]>(this.baseUrl)
28   }
29
30   readById(id: number): Observable<Cadastro> {
31     const url = `${this.baseUrl}/${id}`
32     return this.http.get<Cadastro>(url)
33   }
34 }

```