

Excluir Cliente - *delete*

Em nosso projeto da API com o Framework Angular, atuaremos no **D** do **CRUD** (*Delete* - excluir).

1. Acesse o arquivo **cliente.service.ts**. Adicione o comando a seguir, logo abaixo do método **atualizarClientes()**.

```
excluirCliente(id:any):Observable<Cliente[]>{  
  const urlExcluir = `${this.url}/${id}`;  
  return this._httpClient.delete<Cliente[]>(urlExcluir);  
}
```

Explicando o código

```
excluirCliente(id:any):Observable<Cliente[]>{
```

Esse é o método que acionará a exclusão do registro, ele possui um id de registro como parâmetro. O **Observable** funciona igual aos outros métodos, convertendo os valores da API em um vetor de Clientes.

```
  const urlExcluir = `${this.url}/${id}`;
```

Para atualizar a URL, adicionaremos um **id** igual ao atualizar. O resultado será **http://localhost:3000/clientes/1**.

```
return this._httpClient.delete<Cliente[]>(urlExcluir);
```

Iremos usar o método **delete()** da dependência **_httpClient**, convertendo o **delete()** em **<Cliente[]>**, igual ao **Observable**.

Usamos o **excluirUrl** como parâmetro para saber qual registro deve ser excluído de acordo com o **id**.

A inserção de código em **cliente.service.ts** será realizada conforme abaixo.

```
TS cliente.service.ts U
src > app > TS cliente.service.ts > ...
5
6 @Injectable({
7   providedIn: 'root'
8 })
9 export class ClienteService {
10
11   url: string = "http://localhost:3000/clientes";
12
13   constructor(private _httpClient:HttpClient) { }
14
15   getClientes(): Observable<Cliente[]>{
16     return this._httpClient.get<Cliente[]>(this.url);
17   }
18
19   cadastrarCliente(cliente: Cliente):Observable<Cliente[]>{
20     return this._httpClient.post<Cliente[]>(this.url, cliente);
21   }
22
23   getCliente(id:any):Observable<Cliente[]>{
24     const urlListarUm = `${this.url}?id=${id}`;
25     return this._httpClient.get<Cliente[]>(urlListarUm);
26   }
27
28   atualizarCliente(id: any, cliente: Cliente):Observable<Cliente[]>{
29     const urlAtualizar = `${this.url}/${id}`;
30     return this._httpClient.put<Cliente[]>(urlAtualizar, cliente);
31   }
32
33   excluirCliente(id:any):Observable<Cliente[]>{
34     const urlExcluir = `${this.url}/${id}`;
35     return this._httpClient.delete<Cliente[]>(urlExcluir);
36   }
37
38 }
```

2. Acesse o arquivo **listar-cliente.component.ts** e adicione o código a seguir, logo abaixo do método **listarClientes()**.

```
excluir(id: number){  
  this._clienteService.excluirCliente(id).subscribe(  
    cliente => {  
      this.listarClientes();  
    },  
    err => {alert("Erro ao Excluir")}  
  );  
  
  this._router.navigate(["/lista"]);  
}
```

Explicando o código

```
excluir(id: number){
```

Método que acionará a exclusão do registro. Ele possui um **id** de registro como parâmetro.

```
this._clienteService.excluirCliente(id)
```

Aqui, temos a dependência do **ClienteService**, também chamando o **excluirCliente()** do Serviço e utilizando um **id** do cliente como parâmetro para saber quem deve ser excluído.

```
cliente => {
  this.listarClientes();
```

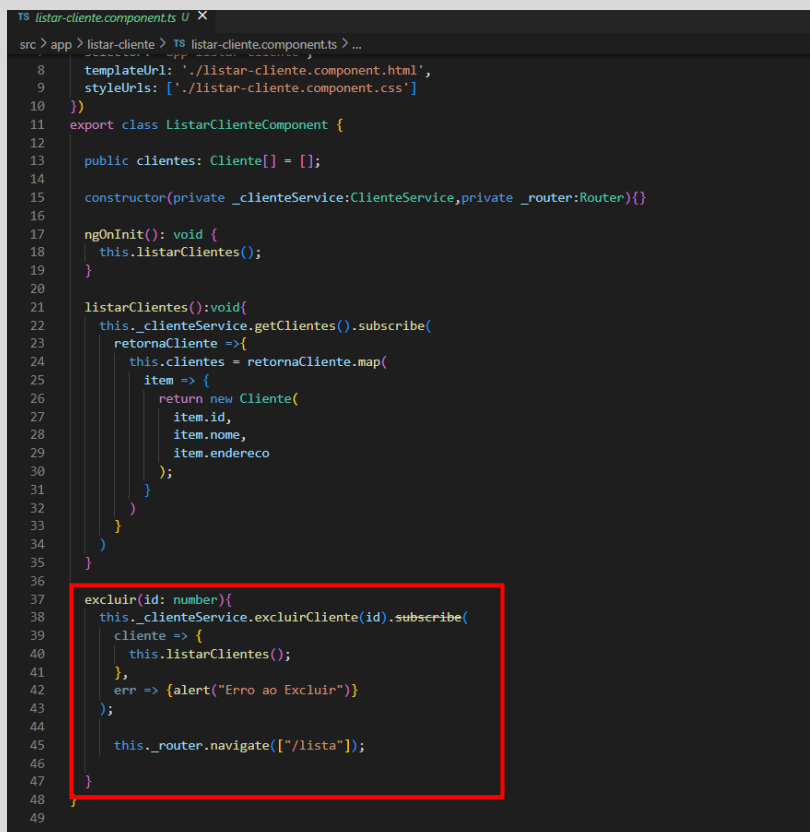
Código que cria uma função dentro da variável cliente para chamar listar clientes novamente, somente para atualizar nossa página.

Caso de problema ele aciona o **err**, acionando um alerta com mensagem de erro.

```
this._router.navigate(["/lista"]);
```

Essa linha está chamando a dependência de rota para utilizar o método **navigate()**, para então redirecionar para a lista.

O código inserido no **listar-cliente.component.ts** ficará conforme destacado abaixo.



```
src > app > listar-cliente > TS listar-cliente.component.ts > ...
8   templateUrl: './listar-cliente.component.html',
9   styleUrls: ['./listar-cliente.component.css']
10 })
11 export class ListarClienteComponent {
12
13   public clientes: Cliente[] = [];
14
15   constructor(private _clienteService: ClienteService, private _router: Router) {}
16
17   ngOnInit(): void {
18     this.listarClientes();
19   }
20
21   listarClientes(): void {
22     this._clienteService.getClientes().subscribe(
23       retornaCliente => {
24         this.clientes = retornaCliente.map(
25           item => {
26             return new Cliente(
27               item.id,
28               item.nome,
29               item.endereco
30             );
31           }
32         );
33       }
34     );
35   }
36
37   excluir(id: number) {
38     this._clienteService.excluirCliente(id).subscribe(
39       cliente => {
40         this.listarClientes();
41       },
42       err => { alert("Erro ao Excluir") }
43     );
44     this._router.navigate(["/lista"]);
45   }
46
47 }
48
49
```

3. Agora iremos adicionar o botão **Excluir** um evento. Acesse o arquivo **listar-cliente.component.html** e substitua pelo código pelo que está abaixo.

```
<table class="table">
  <thead>
    <tr>
      <th scope="col">Id</th>
      <th scope="col">Nome</th>
      <th scope="col">Endereço</th>
      <th scope="col">Editar</th>
      <th scope="col">Excluir</th>
    </tr>
  </thead>
  <tbody>
    <tr *ngFor="let cliente of clientes">
      <th scope="row">{{cliente.id}}</th>
      <td>{{cliente.nome}}</td>
      <td>{{cliente.endereco}}</td>
      <td><a mat-raised-button color="primary"
routerLink="../editar/{{cliente.id}}">Editar</a></td>
      <td><a mat-raised-button color="warn"
(click)="excluir(cliente.id)">Excluir</a></td>
    </tr>
  </tbody>
</table>
```

Explicando o código

```
(click)="excluir(cliente.id)"
```

Esta linha chama o método **excluir()** que criamos no arquivo **lista-cliente.component.ts**

Também foi adicionado **cliente.id**, já listado na API.

Testando a funcionalidade

1. Na aplicação, clique em algum dos botões **Excluir**. O registro selecionado irá desaparecer da lista, conforme indicado nas imagens abaixo.

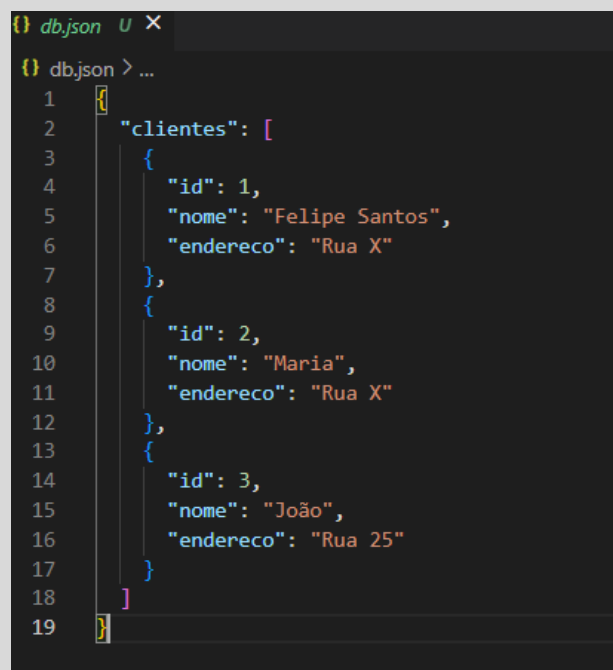
Clientes Cadastro				
Id	Nome	Endereço	Editar	Excluir
1	Felipe Santos	Rua X	<button>Editar</button>	<button>Excluir</button>
2	Maria	Rua X	<button>Editar</button>	<button>Excluir</button>
3	João	Rua 25	<button>Editar</button>	<button>Excluir</button>
4	Maria	Rua x	<button>Editar</button>	<button>Excluir</button>

Clientes Cadastro				
Id	Nome	Endereço	Editar	Excluir
1	Felipe Santos	Rua X	<button>Editar</button>	<button>Excluir</button>
2	Maria	Rua X	<button>Editar</button>	<button>Excluir</button>
3	João	Rua 25	<button>Editar</button>	<button>Excluir</button>

2. Ao atualizar a página, verifique se o registro encontra-se no endereço **localhost:3000/clientes**.

```
[
  {
    "id": 1,
    "nome": "Felipe Santos",
    "endereco": "Rua X"
  },
  {
    "id": 2,
    "nome": "Maria",
    "endereco": "Rua X"
  },
  {
    "id": 3,
    "nome": "João",
    "endereco": "Rua 25"
  }
]
```

3. O registro também é eliminado do arquivo **db.json**.



The screenshot shows a code editor with a file named `db.json` open. The content of the file is a JSON array of three client objects. The first object has `id: 1`, `nome: "Felipe Santos"`, and `endereco: "Rua X"`. The second object has `id: 2`, `nome: "Maria"`, and `endereco: "Rua X"`. The third object has `id: 3`, `nome: "João"`, and `endereco: "Rua 25"`. The code is syntax-highlighted, and line numbers 1 through 19 are visible on the left side of the editor.

```
1 {
2   "clientes": [
3     {
4       "id": 1,
5       "nome": "Felipe Santos",
6       "endereco": "Rua X"
7     },
8     {
9       "id": 2,
10      "nome": "Maria",
11      "endereco": "Rua X"
12    },
13    {
14      "id": 3,
15      "nome": "João",
16      "endereco": "Rua 25"
17    }
18  ]
19 }
```