

# PROGRAMAÇÃO WEBII

1

# Deletar Produtos

DELETE

# O que iremos aprender?

- Vamos aprender como Excluir os registros junto a nossa API com o nosso aplicativo Angular.
- Para isso, iremos precisar instalar a biblioteca SweetAlert. Ela irá gerar mensagens para o usuário com um visual bem agradável, algo que deixará nossa aplicação mais interessante.
- Para acessar a biblioteca, acesse o endereço abaixo:  
<https://sweetalert2.github.io/>
- Clique no Show success message para verificar um exemplo









# Veja como ficará nossa aplicação

Veja como ficará nosso código com a instalação dessa biblioteca:

## Listar Produtos

Cadastrar

#	Nome	Validade	Valor	Ações
1	Curso de Angular avançado			 
2	Curso de Ionic			 
3	aaaa			 



**ATENÇÃO!!!**

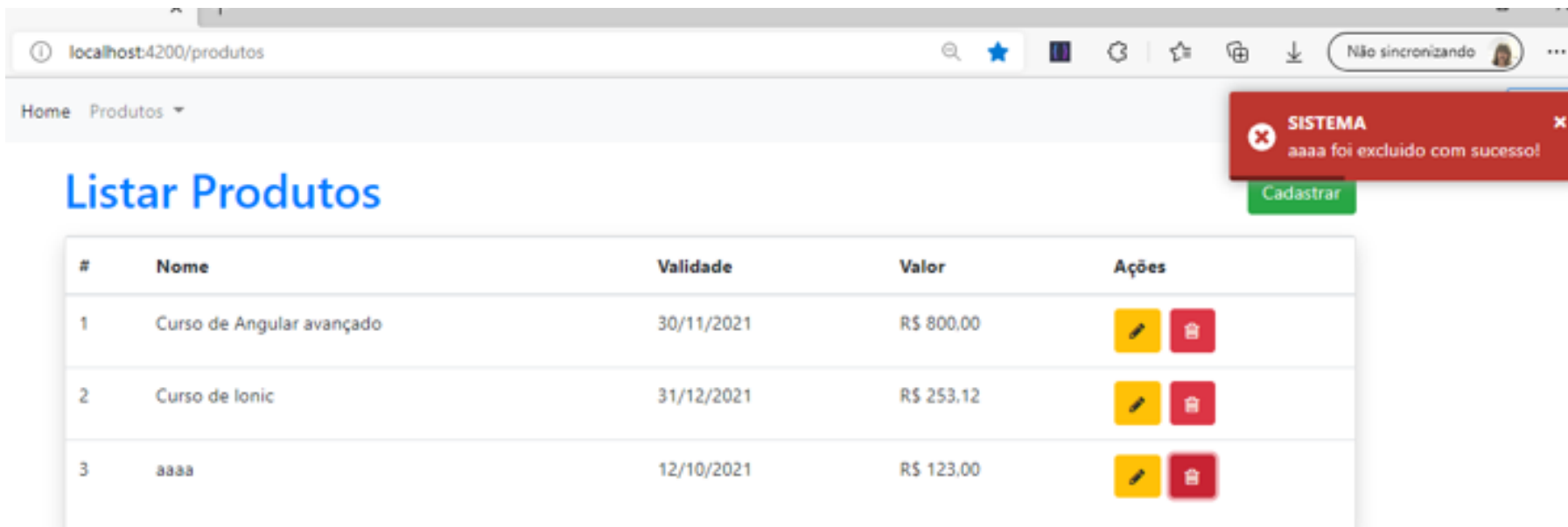
Deseja realmente excluir o produto: aaaa?

Sim, excluir







Não

# Veja como ficará nossa aplicação

Se selecionar Sim, excluir, será exibida a mensagem:



The screenshot shows a web browser at the URL `localhost:4200/produtos`. The page has a navigation bar with "Home" and "Produtos" (with a dropdown arrow). Below the navigation bar is the heading "Listar Produtos". A table displays a list of products. The table has five columns: "#", "Nome", "Validade", "Valor", and "Ações". There are three rows of product data. The third row, with the name "aaaa", has a red success message overlaying it. The message box is red with a white 'x' icon and contains the text "SISTEMA" and "aaaa foi excluido com sucesso!". Below the message box is a green button labeled "Cadastrar".

#	Nome	Validade	Valor	Ações
1	Curso de Angular avançado	30/11/2021	R\$ 800,00	 
2	Curso de Ionic	31/12/2021	R\$ 253,12	 
3	aaaa	12/10/2021	R\$ 123,00	 

# Instalando a biblioteca SweetAlert

- Para fazer a instalação dessa biblioteca, vamos acessar o gitHub
- <https://github.com/sweetalert2/ngx-sweetalert2>
- Vamos ter uma parte dela que é exclusiva para o Angular e é bem simples de instalar. Descendo um pouco no site já será exibido a imagem abaixo:



@sweetalert2/ngx-sweetalert2

Official SweetAlert2 integration for Angular

npm v10.0.0 build passing license MIT

# Instalando a biblioteca SweetAlert

Vamos copiar a linha de instalação:

```
npm install sweetalert2 @sweetalert2/ngx-sweetalert2
```

## Installation & Usage

1. Install *ngx-sweetalert2* and *sweetalert2* via the npm registry:

```
npm install sweetalert2 @sweetalert2/ngx-sweetalert2
```

# Instalando a biblioteca SweetAlert

- Vamos ir no prompt de comando do frontend, vamos parar ele e vamos colar a linha
- Aguardar a instalação

```
npm install sweetalert2 @sweetalert2/ngx-sweetalert2
```

```
C:\angular\cursopwebII\frontend>npm install sweetalert2 @sweetalert2/ngx-sweetalert2
npm WARN @angular/animations@12.1.5 requires a peer of @angular/core@12.1.5 but none is installed. You must install peer dependencies yourself.
npm WARN ajv-keywords@3.5.2 requires a peer of ajv@^6.9.1 but none is installed. You must install peer dependencies yourself.
npm WARN bootstrap@4.6.0 requires a peer of popper.js@^1.16.1 but none is installed. You must install peer dependencies yourself.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.13 (node_modules\webpack-dev-server\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.13: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

+ sweetalert2@11.1.7
+ @sweetalert2/ngx-sweetalert2@10.0.0
added 2 packages from 7 contributors, removed 1 package and audited 1331 packages in 7.149s

90 packages are looking for funding
  run `npm fund` for details

found 30 vulnerabilities (11 moderate, 19 high)
  run `npm audit fix` to fix them, or `npm audit` for details

C:\angular\cursopwebII\frontend>
```




# Configurando a biblioteca SweetAlert

Após a instalação, precisaremos fazer algumas configurações no nosso projeto.

Continuando no GitHub, descendo um pouco mais, precisaremos fazer a importação do nosso módulo.

Vamos copiar a linha ao lado:

```
import { SweetAlert2Module } from  
'@sweetalert2/ngx-sweetalert2';
```

 Always upgrade SweetAlert2 when you upgrade ngx-sweetalert2. The latter is statically linked with SweetAlert2's type definitions.

► [Angular and SweetAlert2 versions compatibility table \(click to show\)](#)

2. Import the module:

```
import { SweetAlert2Module } from '@sweetalert2/ngx-sweetalert2';
```

```
@NgModule({  
  //=> Basic usage (forRoot can also take options, see the wiki)  
  imports: [SweetAlert2Module.forRoot()],  
  
  //=> In submodules only:  
  imports: [SweetAlert2Module],  
  
  //=> In submodules only, overriding options from your root module:  
  imports: [SweetAlert2Module.forChild({ /* options */ })]  
})  
export class AppModule {  
}
```

# Configurando a biblioteca SweetAlert

Vamos abrir o arquivo `app.module.ts`

Vamos colar o módulo no final dos imports.

```
1 import { SharedModule } from './shared/shared.module';
2 import { NgModule } from '@angular/core';
3 import { BrowserModule } from '@angular/platform-browser';
4
5 import { AppRoutingModule } from './app-routing.module';
6 import { AppComponent } from './app.component';
7 import { HeaderComponent } from './template/header/header.component';
8 import { FooterComponent } from './template/footer/footer.component';
9 import { HomeComponent } from './components/home/home.component';
10 import { ListarProdutosComponent } from './components/produtos/listar-produtos/listar-produtos.co
11 import { CadastrarProdutoComponent } from './components/produtos/cadastrar-produto/cadastrar-prod
12
13 import { LOCALE_ID } from '@angular/core';
14 import localePt from '@angular/common/locales/pt';
15 import { registerLocaleData } from '@angular/common';
16 import { FormsModule } from '@angular/forms';
17
18 import { HttpClientModule } from '@angular/common/http';
19
20 import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
21 import { ToastrModule } from 'ngx-toastr';
22 import { AtualizarProdutoComponent } from './components/produtos/atualizar-produto/atualizar-prod
23
24 import { SweetAlert2Module } from '@sweetalert2/ngx-sweetalert2';
25
26 registerLocaleData(localePt);
```

# Configurando a biblioteca SweetAlert

Para deixar nosso projeto mais organizado, vamos levar o import

AtualizarProdutoComponent  
para após o import do  
CadastrarProdutoComponent

```
1 import { SharedModule } from './shared/shared.module';
2 import { NgModule } from '@angular/core';
3 import { BrowserModule } from '@angular/platform-browser';
4
5 import { AppRoutingModule } from './app-routing.module';
6 import { AppComponent } from './app.component';
7 import { HeaderComponent } from './template/header/header.component';
8 import { FooterComponent } from './template/footer/footer.component';
9 import { HomeComponent } from './components/home/home.component';
10 import { ListarProdutosComponent } from './components/produtos/listar-produtos/listar-produtos.co
11 import { CadastrarProdutoComponent } from './components/produtos/cadastrar-produto/cadastrar-prod
12 import { AtualizarProdutoComponent } from './components/produtos/atualizar-produto/atualizar-prod
13
14 import { LOCALE_ID } from '@angular/core';
15 import localePt from '@angular/common/locales/pt';
16 import { registerLocaleData } from '@angular/common';
17 import { FormsModule } from '@angular/forms';
18
19 import { HttpClientModule } from '@angular/common/http';
20
21 import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
22 import { ToastrModule } from 'ngx-toastr';
23
24 import { SweetAlert2Module } from '@sweetalert2/ngx-sweetalert2';
25
```

# Configurando a biblioteca SweetAlert

- O próximo passo é declarar a biblioteca dentro do imports.
- Vamos na documentação e copiamos somente o que está sublinhado:

SweetAlert2Module.forRoot()

```
import { SweetAlert2Module } from '@sweetalert2/ngx-sweetalert2';

@NgModule({
  //=> Basic usage (forRoot can also take options, see the wiki)
  imports: [SweetAlert2Module.forRoot()],

  //=> In submodules only:
  imports: [SweetAlert2Module],

  //=> In submodules only, overriding options from your root module:
  imports: [SweetAlert2Module.forChild({ /* options */ })]
})
export class AppModule {
}
```

# Configurando a biblioteca SweetAlert

Vamos colar, conforme abaixo:

SweetAlert2Module.forRoot()

```
imports: [  
  BrowserModule,  
  AppRoutingModule,  
  FormsModule,  
  SharedModule,  
  HttpClientModule,  
  BrowserAnimationsModule, // required animations module  
  ToastrModule.forRoot(), // ToastrModule added  
  SweetAlert2Module.forRoot()  
],
```

# Configurando a biblioteca SweetAlert

■ Como não iremos mais atualizar o `app.module.ts` já podemos dar start no nosso servidor, pois todas as mudanças estruturais já foram realizadas.

▶ `ng serve -o <ENTER>`

# Configurando o serviço

- Agora vamos abrir o arquivo produtos.service.ts
- Vamos criar o método de excluir
- Vamos copiar o método de atualizar
- Vamos colar logo abaixo:

```
atualizar(prodoto: IProduto): Observable<IProduto> {  
    return this.http.put<IProduto>(`${this.URL}/${produto.id}`, produto).pipe(  
        map(retorno => retorno),  
        catchError(erro => this.exibirErro(erro))  
    );  
}  
  
atualizar(prodoto: IProduto): Observable<IProduto> {  
    return this.http.put<IProduto>(`${this.URL}/${produto.id}`, produto).pipe(  
        map(retorno => retorno),  
        catchError(erro => this.exibirErro(erro))  
    );  
}
```

# Configurando o serviço

- Vamos alterar o nome do método para excluir
- Irá receber como parâmetro um id: number
- Ele irá retornar um Observable, mas dessa vez como já excluímos o produto não tem como retornar um produto, então vamos retornar alguma coisa do tipo any, nos não sabemos o que será retornado por isso iremos colocar aqui o any.
- No método put, ao invés de put iremos chamar delete
- Após o delete, iremos receber um tipo <any> ao invés de <IProduto>
- Na URL ao invés de produto.id, iremos passar somente o id
- Após o id, não precisaremos passar um produto
- Podemos salvar e testar



# Configurando o serviço

- Se der tudo certo, o comando delete para a API ele irá fazer um retorno, vai retornar esse retorno para nós.
- Se der algum erro, o Observable irá interceptar o erro e tratar isso para nós.
- Nosso serviço está pronto.
- Nosso código ficará assim:

```
excluir(id: number): Observable<any> {  
  return this.http.delete<any>(`${this.URL}/${id}`).pipe(  
    map(retorno => retorno),  
    catchError(erro => this.exibirErro(erro))  
  );  
}
```

# Configurando o serviço

Propriedades no botão excluir, utilizando a biblioteca sweetalert:



# Configurando o serviço

- Agora vamos abrir o arquivo `listar-produtos.component.html`

- Vamos procurar o botão excluir

- Após o `<btn btn-danger ml-2>` vamos dar um espaço e digitar o `swal` que é a nossa biblioteca `sweetAlert`

`[swal] = "{}"` Após o igual, temos que digitar aspas e criar um objeto, sem espaço em branco entre as palavras.

- ▶ Dentro do objeto, teremos que passar os parâmetros que queremos que apareça no nosso alerta.
- ▶ `title: 'ATENÇÃO!!!',`
- ▶ `text: 'Deseja realmente excluir o produto: ' + prod.nome + '?',`
- ▶ `showDenyButton: true,`
- ▶ `denyButtonText: 'Não', // texto para o botão de cancelar`
- ▶ `confirmButtonText: 'Sim, excluir', icon: 'warning' // texto para o botão de confirmar com o ícone de warning`

- Quando clicarmos no confirm, irá chamar o método `deletar()`, passando o nosso produto. Após o parênteses, vamos acrescentar:

`(confirm) = "deletar(prod)"`

# Configurando o serviço

Ficará assim:

```
<button
  class="btn btn-danger ml-2" [swal]="{
    title: 'ATENÇÃO!!!',
    text: 'Deseja realmente excluir o produto: ' + prod.nome + '?',
    showDenyButton: true,
    denyButtonText: 'Não',
    confirmButtonText: 'Sim, excluir',
    icon: 'warning'
  }"
  (confirm)="deletar(prod)"
>
  <i class="fa fa-trash-o" aria-hidden="true"></i>
</button>
```

Não pode ter espaço em branco

Esse método ainda não existe, precisaremos criar ele.

# Configurando o serviço

- Para criar o método `deletar()`, vamos abrir o arquivo `listar-produtos.component.ts`
- Após o método `carregarProdutos()`, vamos criar o `deletar()`:
- O método `deletar` irá retornar um **void**
- Irá receber um produto, do tipo `IProduto` (**produto: IProduto**)

```
deletar(produto: IProduto): void {
```

`this.produtosService.excluir(produto.id!).subscribe(() =>{` // **subscribe é para enviar o nosso pedido através dos métodos http. Como não teremos nenhum retorno, só iremos tratar a resposta.**

```
    this.produtosService.exibirMensagem( // vamos tratar a mensagem de erro
```

```
    'SISTEMA',
```

```
    `${produto.nome} foi excluído com sucesso!`,
```

```
    'toast-error' // aqui vamos passar o tipo de erro
```

```
);
```

```
}
```

# Configurando o serviço

Por que símbolo de exclamação?

```
this.produtosService.excluir(produto.id!)
```

No arquivo IProduto.model.ts, o id está como opcional e, estando como opcional nas definições de tipo, por isso precisamos usar uma declaração não nula.

```
src > app > model > TS IProduto.model.ts > IProduto > id
1  export interface IProduto{
2      id?: number;
3      nome: string;
4      validade: Date;
5      precoProduto: number;
6  }
7
```

strictNullChecks alterna para um novo modo de verificação estrita de nulos. No modo de verificação nula estrita, os valores nulos e indefinidos não estão no domínio de todos os tipos e só podem ser atribuídos a eles próprios e a qualquer um (a única exceção sendo que indefinido também pode ser atribuído a void).

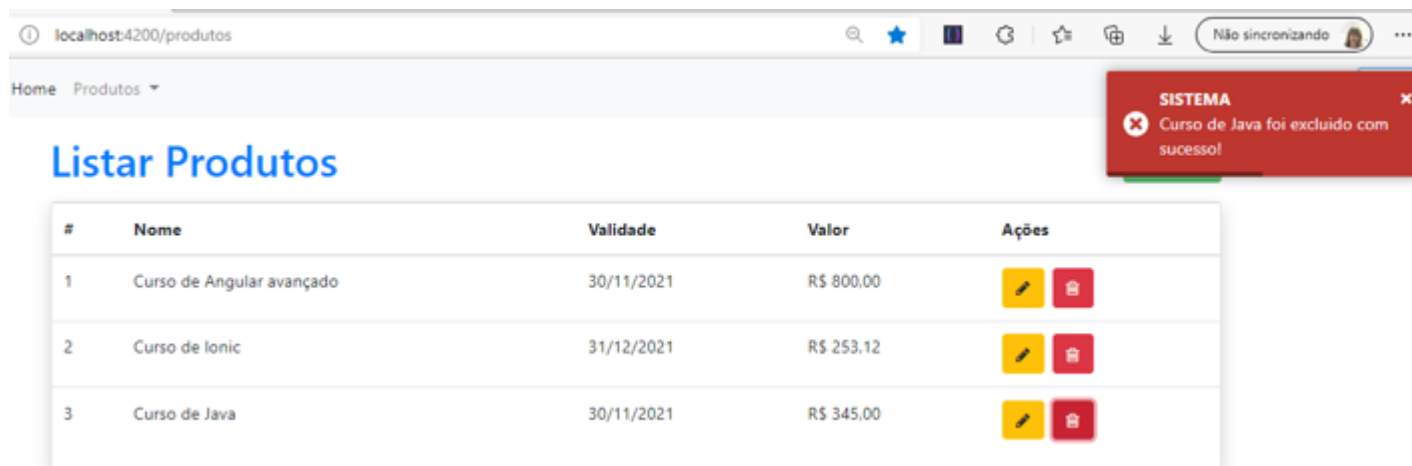
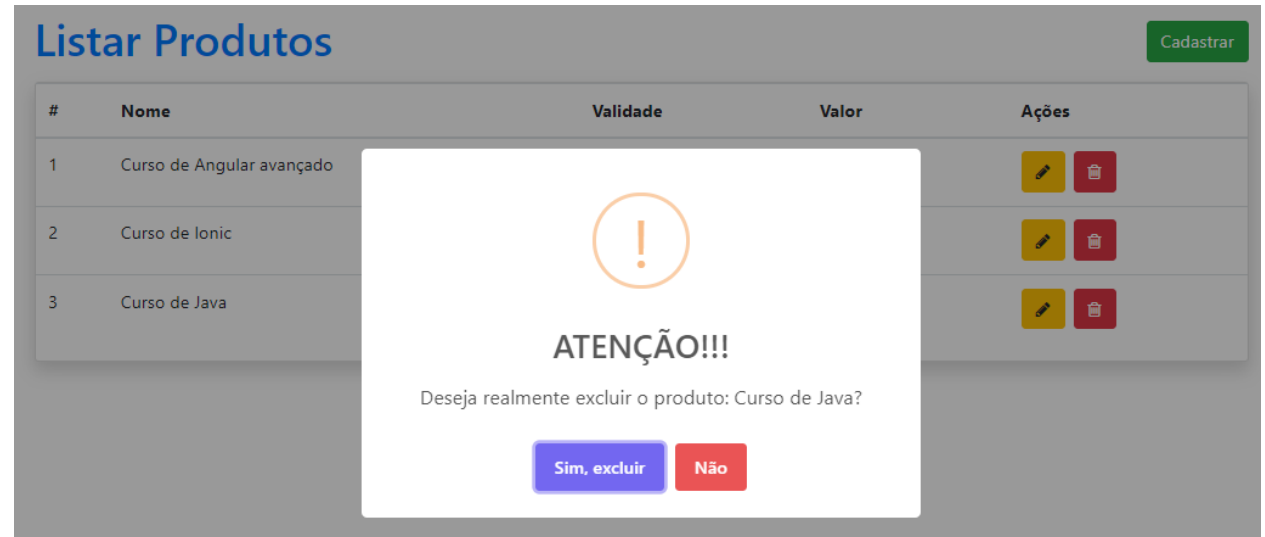
# Configurando o serviço

Ficará assim:

```
deletar(prodoto: IProduto): void{  
  this.produtosService.excluir(prodoto.id!).subscribe(() =>{  
    this.produtosService.exibirMensagem(  
      'SISTEMA',  
      `${prodoto.nome} foi excluido com sucesso!`,  
      'toast-error'  
    );  
  });  
}
```

# Testando...

Foi exibida a mensagem correta, porém na tela continua o curso, que foi excluído, Precisamos fazer esse pequeno ajuste.





# Ajustando...

- Abra o arquivo listar-produtos.component.ts
- Após o método deletar(), vamos carregar o método que exibe a lista de produtos novamente:
  - ▶ this.carregarProdutos();

```
deletar(produto: IProduto): void{  
  this.produtosService.excluir(produto.id!).subscribe(() =>{  
    this.produtosService.exibirMensagem(  
      'SISTEMA',  
      `${produto.nome} foi excluido com sucesso!`,  
      'toast-error'  
    );  
    this.carregarProdutos();  
  });  
}
```

# Testando...



Vamos cadastrar um novo curso e excluir para verificar se está correto.

CRUD Angular Home Produtos ▾

SISTEMA  
Teste foi excluído com sucesso!

Cadastrar

## Listar Produtos

#	Nome	Validade	Valor	Ações
1	Curso de Angular avançado	30/11/2021	R\$ 800,00	 
2	Curso de Ionic	31/12/2021	R\$ 253,12	