

PROGRAMAÇÃO WEBII

1

Atualizar Produtos

PUT

O que iremos aprender?

- Vamos aprender como alterar os produtos junto a nossa API com o nosso aplicativo Angular.
- Nós poderíamos reutilizar nossa tela de cadastro, pois facilitaria o nosso desenvolvimento uma vez que já temos ela aqui pronta e nós só teríamos que trazer os dados do nosso produto que está cadastrado aqui para nossa interface
- Mas, como estamos aprendendo Angular pela primeira vez, vamos criar um novo componente exclusivo para alterar o produto, facilitando nossa programação.

Cadastrar Produto

Nome do Produto:

Validade do Produto:

Preço do Produto:

Voltar

Salvar

Alterar produto

- Vamos começar parando o servidor do Angular, caso estiver sendo executado.
- Vamos criar o componente.
- g- generate
- c – componente
- Components – pasta que estamos agrupando os componentes

➤ ng g c components/produtos/atualizar-produto <ENTER>

```
C:\angular\cursopwebII\frontend>ng g c componentes/produtos/atualizar-produto
CREATE src/app/components/produtos/atualizar-produto/atualizar-produto.component.html (32 bytes)
CREATE src/app/components/produtos/atualizar-produto/atualizar-produto.component.ts (318 bytes)
CREATE src/app/components/produtos/atualizar-produto/atualizar-produto.component.css (0 bytes)
UPDATE src/app/app.module.ts (1910 bytes)
```

- Já criou o componente e podemos executar o servidor novamente:
 - ng serve -o <ENTER>

Alterar produto

Vamos para o nosso código, acessar `app.module.ts` e verificaremos que já existe o `AtualizarProdutoComponent` nas `declarations`, isso significa que está pronto.

```
@NgModule({  
  declarations: [  
    AppComponent,  
    HeaderComponent,  
    FooterComponent,  
    HomeComponent,  
    ListarProdutosComponent,  
    CadastrarProdutoComponent,  
    AtualizarProdutoComponent  
  ],  
})
```

Alterar produto

- Agora vamos no nosso serviço, abrir o arquivo: `produtos.service.ts` e vamos criar os métodos:
 - ▶ `buscarDados`, porque precisamos buscar o cadastro do produto para que possamos exibí-lo na tela para depois alterá-lo
 - ▶ Vamos criar o método de `atualizar()`

Alterar produto

- Iniciando pelo método de buscar, vamos copiar o método buscarTodos() e vamos colar logo embaixo:

```
buscarTodos(): Observable<IProduto[]> {  
    return this.http.get<IProduto[]>(this.URL).pipe(  
        map(retorno => retorno),  
        catchError(erro => this.exibirErro(erro))  
    );  
}  
  
buscarTodos(): Observable<IProduto[]> {  
    return this.http.get<IProduto[]>(this.URL).pipe(  
        map(retorno => retorno),  
        catchError(erro => this.exibirErro(erro))  
    );  
}
```

Alterar produto

Vamos alterar o nome do método para:

- ▶ buscarPorId, irá receber o parâmetro do ID do tipo number. Ele irá retornar um Observable de Produto, mas vamos retirar o array.

```
buscarPorId(id: number): Observable<IProduto> {
```

- ▶ Nós iremos novamente fazer um get, mas deverá retornar somente um produto. Para isso, vamos retirar o array da linha do get <iProduto>(. Após o get, vamos transformar em um template string, colocando uma crase, `\${`. Após a URL, fechamos a chaves, inserimos uma barra / e vamos passar o id `\${id}`, ficando assim:

```
buscarPorId(id: number): Observable<IProduto> {  
  return this.http.get<IProduto>(`${this.URL}/${id}`).pipe(  
    map((produto) => produto),  
    catchError((error) => throwError('Erro ao buscar produto'))  
  )  
}
```


Alterar produto

- Vamos pegar o retorno, vamos enviar ele caso esteja tudo certo.
- Se não estiver certo, se der algum erro durante a busca, o próprio Observable vai tratar o nosso erro aqui.
- Dessa maneira, já estou conseguindo buscar um produto para alterar ele.

```
buscarPorId(id: number): Observable<IProduto> {  
  return this.http.get<IProduto>(`${this.URL}${id}`).pipe(  
    map(retorno => retorno),  
    catchError(erro => this.exibirErro(erro))  
  );  
}
```

Alterar produto

- O próximo passo é atualizar o nosso produto.
- Vamos copiar o método cadastrar e copiar logo abaixo:

```
cadastrar(produto: IProduto): Observable<IProduto> {  
    return this.http.post<IProduto>(this.URL, produto).pipe(  
        map(retorno => retorno),  
        catchError(erro => this.exibirErro(erro))  
    );  
}  
  
cadastrar(produto: IProduto): Observable<IProduto> {  
    return this.http.post<IProduto>(this.URL, produto).pipe(  
        map(retorno => retorno),  
        catchError(erro => this.exibirErro(erro))  
    );  
}
```

Alterar produto

- Vamos alterar o nome para atualizar.
- Também vou receber um produto,
- Ao invés de fazer um post, vamos mudar para put
 - Vamos alterar a URL para template string, inserindo uma crase \${
 - Após a URL, fechamos a chaves, inserimos uma barra / e vamos passar o id \${produto.id}, ficando assim:

```
atualizar(produto: IProduto): Observable<IProduto> {  
    return this.http.put<IProduto>(`${this.URL}/${produto.id}`, produto).pipe(  
        ...  
    )  
}
```

Alterar produto

- A nossa URL, o nosso atualizar produto está pronto
- Se der tudo certo na atualização, recebemos os dados e retornamos ele
- Se der erro, o Observable tratará esse erro para nós.
- Isso significa que nosso serviço está pronto.

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

Alterar produto

Agora vamos preparar nossa rota.

- Vamos abrir o arquivo app-routing.module.ts

- Vamos criar uma nova rota, para isso vamos colocar um vírgula no final da rota
CadastroProdutoComponent

- Vamos digitar:

- ▶ `{path: 'produtos/atualizar/:id'}`
- ▶ O próprio Angular já conhece esses `:id`. Toda vez que quisermos passar um valor utilizando a URL, nós vamos utilizar `:e` o nome, como vamos passar o `id`, vamos passar o `id`. Você pode colocar o nome que vc quiser.
- ▶ Continuando, vamos digitar a virgula, `component` e o nome do componente que ele irá chamar:
- ▶ `{path: 'produtos/atualizar/:id', component: AtualizarProdutoComponent}`
- ▶ Ao digitar `AtualizarComponent` (clique para fazer o auto import)

Alterar produto

Ficará assim:

```
const routes: Routes = [  
  {path:'', component: HomeComponent},  
  {path: 'produtos', component: ListarProdutosComponent},  
  {path: 'produtos/cadastrar', component: CadastrarProdutoComponent},  
  {path: 'produtos/atualizar/:id', component: AtualizarProdutoComponent}  
];
```

```
import { AtualizarProdutoComponent } from './components/produtos/atualizar-produto/atualizar-produto.component';  
import { NgModule } from '@angular/core';  
import { RouterModule, Routes } from '@angular/router';
```

Alterar produto

- As rotas estão agora atualizadas, já temos uma rota para atualizar produtos.
- Para deixar mais organizado, vamos retirar o import da linha de cima e inserir após o último import:

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';

import { ListarProdutosComponent } from './components/produtos/listar-produtos/listar-produtos.component';
import { HomeComponent } from './components/home/home.component';
import { CadastrarProdutoComponent } from './components/produtos/cadastrar-produto/cadastrar-produto.component';
import { AtualizarProdutoComponent } from './components/produtos/atualizar-produto/atualizar-produto.component';
```

Alterar produto

- Vamos abrir o arquivo listar-produtos.component.html
- Nós temos um botão que vai nos levar para o alterar e vamos colocar a rota que foi criada:

```
<td>  
  <button class="btn btn-warning">  
    <i class="fa fa-pencil" aria-hidden="true"></i>  
  </button>
```


Alterar produto

Vamos digitar após o btn-warning":

- ▶ routerLink = "/produtos/atualizar/"
- ▶ Aqui passaremos a interpolação e chamamos o nosso produto (prod.id)
- ▶ routerLink = "/produtos/atualizar/{{prod.id}}"
- ▶ Já estamos com a nossa rota pronta para funcionar. Então nosso routerLink vai nos levar para /produtos/atualizar e o id que está vindo do nosso produto





```
<button  
  class="btn btn-warning"  
  routerLink="/produtos/atualizar/{{ prod.id }}"  
>
```

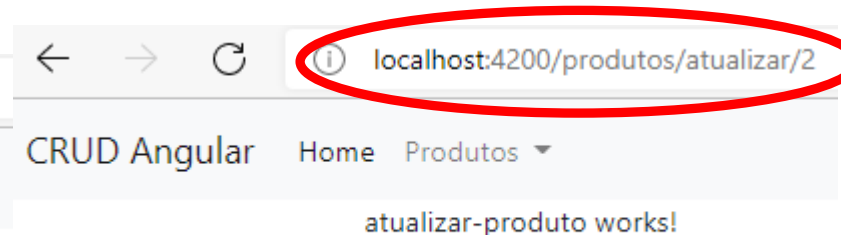
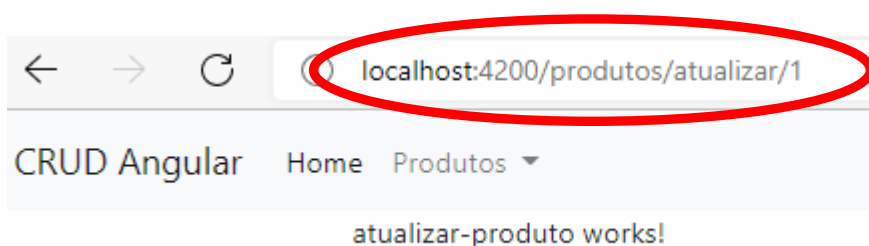
Alterar produto

- Vamos testar?
- Vamos abrir listar produtos, ao clicar no botão editar, irá para a outra página onde exibirá o código do produto e exibirá o conteúdo do componente atualizar.

Listar Produtos

Cadastrar

#	Nome	Validade	Valor	Ações
1	Curso de Angular	31/12/2021	R\$ 550,54	 
2	Curso de Ionic	31/12/2021	R\$ 253,12	 



Alterar produto

- Apague o conteúdo da página atualizar-produto.component.html
- Agora vamos fazer um copiar e colar da nossa página cadastrar-produto.component.html para atualizar-produto.component.html
- Onde está cadastrar produto, vamos alterar para atualizar produto

```
1 <h1 class="my-3 text-primary">Atualizar Produto</h1>  
2
```

Alterar produto

■ Lá embaixo, no botão success, vamos alterar de verde para ficar amarelo.

```
<button class="btn btn-default" routerLink="/produtos">Voltar</button>  
<button class="btn btn-success float-right" (click)="salvarProduto()">  
|   Salvar  
</button>
```

```
<button class="btn btn-default" routerLink="/produtos">Voltar</button>  
<button class="btn btn-warning float-right" (click)="salvarProduto()">  
|   Salvar  
</button>
```

Alterar produto

- Ele está com erro porque ainda não fizemos a parte do TypeScript
- Vamos abrir o arquivo atualizar-produto.component.ts
- Vou abrir o arquivo cadastrar-produto.component.ts
- Vamos copiar da próxima linha após o export class até o final

```
11 export class CadastrarProdutoComponent implements OnInit {
12
13   produto: IProduto = {
14     nome: '',
15     validade: new Date(),
16     precoProduto: 0
17   };
18
19
20
21   constructor(private produtosService: ProdutosService, private router: Router) {}
22
23   ngOnInit(): void {}
24
25   salvarProduto(): void {
26     this.produtosService.cadastrar(this.produto).subscribe(retorno => {
27       this.produto = retorno;
28       this.produtosService.exibirMensagem(
29         'Sistema',
30         `${this.produto.nome} foi cadastrado com sucesso. ID: ${this.produto.id}`,
31         'toast-success'
32       );
33       this.router.navigate(['/produtos']);
34     });
35
36   }
37 }
```

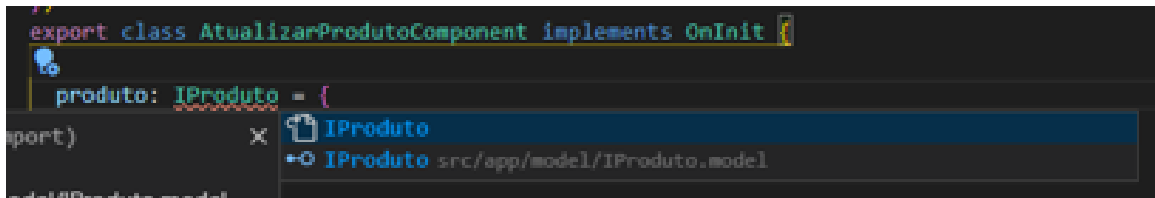
Alterar produto

- Abro o arquivo atualizar-produto.component.ts
- Apago após a linha do export class
- Colo o conteúdo do cadastrar
- Podemos fechar o cadastrar-produto.component.ts

```
8   export class AtualizarProdutoComponent implements OnInit {  
9  
10      constructor() {}  
11  
12      ngOnInit(): void {  
13      }  
14  
15  }
```

Alterar produto

- Vamos fazer as importações agora:
- Levar o mouse até o final da palavra IProduto
- Pressione <CTRL> <barra de espaço>
- Irá exibir o IProduto para fazer a importação e é só clicar com o mouse.
- Fará a importação automaticamente.



```
import { IProduto } from '../../../model/IProduto.model';  
import { Component, OnInit } from '@angular/core';
```

Alterar produto

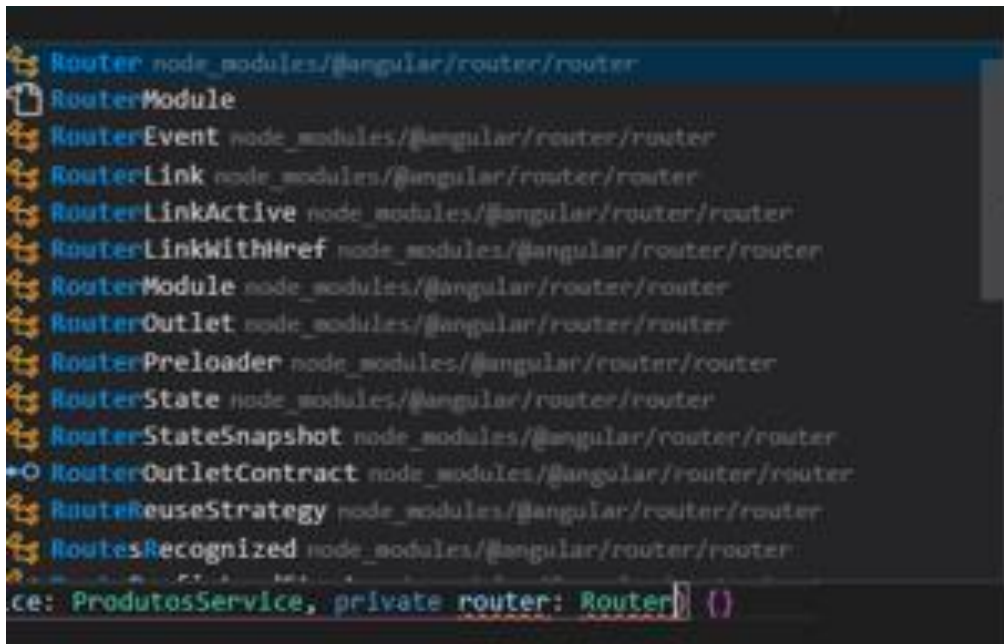
- Próximo será o ProdutosService, mesmo procedimento:
- Após o 2º ProdutosService
- Pressione <CTRL> <barra de espaço>



```
import { ProdutosService } from './../../../../services/produtos.service';  
import { IProduto } from './../../../../model/IProduto.model';  
import { Component, OnInit } from '@angular/core';
```


Alterar produto

- Próximo será o Router, mesmo procedimento:
- Após o 2º Router
- Pressione <CTRL> <barra de espaço>

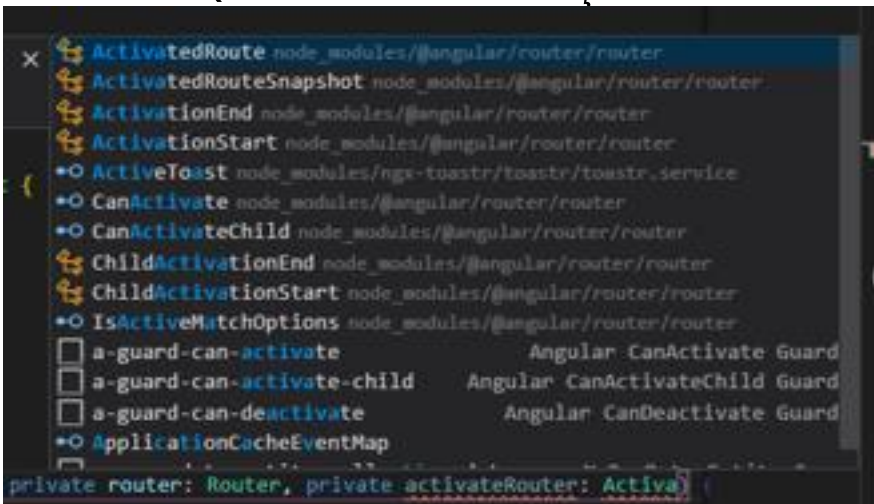


A screenshot of an IDE's autocomplete dropdown for the Router module. The list includes Router, RouterModule, RouterEvent, RouterLink, RouterLinkActive, RouterLinkWithHref, RouterOutlet, RouterPreloader, RouterState, RouterStateSnapshot, RouterOutletContract, RouterReuseStrategy, and RoutesRecognized. The RouterModule entry is highlighted. Below the list, the start of a TypeScript class is visible: `ce: ProdutosService, private router: Router {}`.

```
import { ProdutosService } from '../../../services/produtos.service';
import { IProduto } from '../../../model/IProduto.model';
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
```

Alterar produto

- Já temos aqui tudo o que precisamos para fazer o atualizar produto
- Próximo passo, vamos colocar uma vírgula no construtor, após o Router e vamos chamar mais uma classe do Angular que é responsável por pegar os parâmetros que estão sendo passados de uma tela para outra.
- Para isso, vamos escrever:
 - `private activateRouter:`
 - Quando começar a escrever `Activa` Já será exibido para realizarmos o auto import.



```
import { ProdutosService } from '../../../services/produtos.service';  
import { IProduto } from '../../../model/IProduto.model';  
import { Component, OnInit } from '@angular/core';  
import { ActivatedRoute, Router } from '@angular/router';
```

Alterar produto

- Vamos organizar as importações do construtor, dando <ENTER> em cada uma delas.
- Assim fica mais fácil identificar todo mundo que instanciamos dentro do nosso construtor

```
constructor(  
  private produtosService: ProdutosService,  
  private router: Router,  
  private activateRouter: ActivatedRoute) {}
```

Alterar produto

- Agora precisamos ter uma maneira de obter o parâmetro da rota. Existem 2 maneiras de obter o parâmetro da rota:
 - 1. Instantâneo
`(route.snapshot.paramMap.get)`.
Leia-o durante o init.
 - Use o Snapshot se você precisar apenas do valor inicial do parâmetro uma vez durante a inicialização do componente e não espere que o URL seja alterado enquanto o usuário ainda estiver no mesmo componente.
- I.e. se estiver em uma rota de produto/2, e a única maneira de chegar ao produto/3 é voltar à tela de pesquisa de produto e clicar em um detalhe do produto (deixando o componente de detalhes e reabrindo-o com uma nova rota) param)

Alterar produto

■ 2. Observável

(route.paramMap.subscribe).

Inscreva-se durante o init.

- Use o Observável se for possível que a rota seja alterada enquanto o usuário ainda estiver no mesmo componente e, portanto, a inicialização do Componente não seria chamada novamente, mas o observável chamaria sua lógica inscrita quando o URL fosse alterado.
- I.e. se estiver em uma rota product/2 e você tiver um botão "next" para ir para o próximo produto de registro de identificação/3, portanto, o usuário não deixou/reabriu o componente, mas a URL recebeu um novo parâmetro.
- <https://www.ti-enxame.com/pt/angular/qual-e-diferenca-entre-activatedroute-e-activatedroutesnapshot-no-angular4/834296909/>

Alterar produto

- Nesse momento, vamos fazer uso novamente do ngOnInit.
- Toda vez que nosso componente for ser inicializado, após a chave, vamos criar uma constante, já que esse valor nunca será alterado durante a execução desse componente e vamos chamar de id e será igual a Number(). Nos iremos fazer uma conversão aqui do valor que iremos receber.
- Dentro do get, iremos passar o id
 - `const id = Number(this.activateRouter.snapshot paramMap.get('id'));`
 - Essa linha gigante que escrevemos aqui agora é para buscar os dados do nosso parâmetro id que estamos passando através do navegador, quando clicamos no botão alterar.

```
ngOnInit(): void {  
  const id = Number(this.activateRouter.snapshot paramMap.get('id'));  
}
```

Alterar produto

Continuando...

Agora vamos chamar o serviço para buscarmos os dados do produto que nós queremos atualizar:

```
this.produtosService.buscarPorId(id).subscribe(retorno => {  
  
})
```

- ▶ Nesse momento, vamos pegar o retorno e tratar ele aqui em uma função. Dentro da função:

```
this.produto = retorno;
```

```
ngOnInit(): void {  
  const id = Number(this.activateRouter.snapshot.paramMap.get('id'));  
  this.produtosService.buscarPorId(id).subscribe(retorno => {  
    this.produto = retorno;  
  });  
}
```




Vamos inserir um
ponto e vírgula no final

Alterar produto

- Está feito. Agora, quando o usuário clicar no botão Atualizar Produto vocês irão ver que agora irá trazer os dados do produto para nós.
- Vamos verificar se está funcionando.

Listar Produtos

[Cadastrar](#)

#	Nome	Validade	Valor	Ações
1	Curso de Angular	31/12/2021	R\$ 550,54	 
2	Curso de Ionic	31/12/2021	R\$ 253,12	 


Atualizar Produto

Nome do Produto:

Curso de Angular

Validade do Produto:

31/12/2021



Preço do Produto:

550,54

Voltar

Salvar


Alterar produto

■ Agora precisamos fazer com que o botão Salvar envie uma solicitação para atualização do nosso curso

Atualizar Produto

Nome do Produto:

Validade do Produto:



Preço do Produto:

Voltar

Salvar

Alterar produto

- Agora precisamos fazer com que o botão Salvar envie uma solicitação para atualização do nosso curso
- Vamos abrir o arquivo atualizar-produto.component.ts
- No lugar do método cadastrar, vamos alterar para atualizar

```
salvarProduto(): void {  
  this.produtosService.cadastrar(this.produto).subscribe(retorno => {  
    this.produto = retorno;  
  });  
}
```

```
salvarProduto(): void {  
  this.produtosService.atualizar(this.produto).subscribe(retorno => {  
    this.produto = retorno;  
  });  
}
```

Alterar produto

- Já estamos fazendo subscribe
- Já estamos pegando o retorno do nosso produto
- Vamos escrever aqui o que deu sucesso e agora temos tudo certinho aqui que ele já conseguiu fazer a atualização.
- Vamos só alterar a mensagem para ao invés de escrever foi cadastrado, vamos alterar para foi atualizado com sucesso e não precisamos o id do nosso produto.
- Vamos salvar e testar

ANTES

```
salvarProduto(): void {  
  this.produtosService.cadastrar(this.produto).subscribe(retorno => {  
    this.produto = retorno;  
    this.produtosService.exibirMensagem(  
      'Sistema',  
      `${this.produto.nome} foi cadastrado com sucesso. ID: ${this.produto.id}`,  
      'toast-success'  
    );  
    this.router.navigate(['/produtos']);  
  });  
}
```

DEPOIS





```
salvarProduto(): void {  
  this.produtosService.atualizar(this.produto).subscribe(retorno => {  
    this.produto = retorno;  
    this.produtosService.exibirMensagem(  
      'Sistema',  
      `${this.produto.nome} foi atualizado com sucesso.`,  
      'toast-success'  
    );  
    this.router.navigate(['/produtos']);  
  });  
}
```

Alterar produto

Vamos no listar Produtos e vamos selecionar, por exemplo, o curso de Angular

Listar Produtos

[Cadastrar](#)

#	Nome	Validade	Valor	Ações
1	Curso de Angular	31/12/2021	R\$ 550,54	 
2	Curso de Ionic	31/12/2021	R\$ 253,12	 


Alterar produto

Vamos alterar os dados abaixo e clicar no botão Salvar

Atualizar Produto

Nome do Produto:

Validade do Produto:



Preço do Produto:

[Voltar](#)



[Salvar](#)

Alterar produto

E já temos os novos dados sendo exibidos na tela de Produtos:

Listar Produtos

[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	