

# PROGRAMAÇÃO WEBII

# 1

## Criação de componentes reutilizáveis

Utilizando Angular

# O que iremos aprender?

- Vamos aprender como criar componentes reutilizáveis utilizando o Angular.
  - Estrutura de Projetos
  - Criação de módulo
  - Criação de componente reutilizável

# Relembrando...

- Considere sua aplicação angular como um edifício. Um edifício pode ter N vários apartamentos nele. Um apartamento é considerado como um **módulo**. Um apartamento pode ter um N número de salas que correspondem aos blocos de construção de um aplicativo angular chamado **componentes**.
- Agora, cada apartamento (**Módulo**) terá salas (**Componentes**), elevadores (**Serviços**) para permitir maior movimentação dentro e fora dos apartamentos, fios (Tubos) para mover informações e torná-las úteis nos apartamentos.
- Você também terá lugares como piscina, quadra de tênis que estão sendo compartilhados por todos os moradores do prédio. Portanto, eles podem ser considerados **componentes no SharedModule**.

# Criação de componentes reutilizáveis

Quando estamos criando aplicações Web, é normal que nossa aplicação tenha vários módulos, várias seções e que essas seções terão funções específicas, especializadas que são daquela parte do nosso sistema. Como por exemplo:

## Módulo de estrutura para projetos maiores

Obs: É apresentando somente elementos que são mostrados neste curso.

```
|-- financeiro
    |-- financeiro.module.ts
    |-- [+] components
    |-- [+] ...
|-- estoque
    |-- estoque.module.ts
    |-- [+] ...
|-- shared
    |-- shared.module.ts
    |-- [+] components
    |-- [+] models
    |-- [+] directives
    |-- [+] pipes
```

# Criação de componentes reutilizáveis

- Vamos ter a área financeira da nossa aplicação. Vamos ter a área de estoque e o Angular fornece uma forma de organizarmos o nosso projeto muito legal que são os módulos, que já vimos no começo do curso.
- Hoje iremos aprender como criar um módulo e como reaproveitar esse módulo para nossos componentes reutilizáveis. Por que isso? É convenção no Angular que nós trabalhemos com uma pasta denominada shared. Essa pasta shared terá o:

|-- shared

|-- shared.module.ts

|-- [+] components

|-- [+] models

|-- [+] directives

|-- [+] pipes

Shared.module -> Esse módulo será responsável por todos os componentes, modelos, diretivas, pipes, elementos do nosso projeto que vão estar compartilhados com todas as áreas do nosso sistema

# Criação de componentes reutilizáveis

- Nós temos aqui, como exemplo, dentro do financeiro que o financeiro vai ter seu módulo, seus componentes. Esses componentes não foram feitos para serem utilizados no resto sistema, Eles foram feitos apenas para serem utilizados dentro da área financeira do sistema. Por isso eles estão agrupados dentro da pasta financeiro.
- A pasta shared vai fornecer elementos para todo o nosso projeto. E por que estamos vendo somente esses elementos quando estou tratando do Angular? Por que o Angular só tem isso? Não, o Angular tem muito mais coisa, nosso projeto pode ter muito mais elementos do Angular, mas aqui está disponibilizado na estrutura somente os elementos que estamos vendo dentro do curso. Aqui não teremos conteúdo que não estaremos vendo dentro do curso.

# Como iremos fazer? (resumo)

- Criaremos um componente reutilizável referente ao card que está na [home.component.html](#)
  - ▶ Criaremos um modulo denominado shared
  - ▶ Dentro do modulo, criaremos uma pasta denominada components e dentro da pasta o componente card-produto
  - ▶ No componente card-produto conterà os comandos do [home.component.html](#)



# Criação de componentes reutilizáveis

- Vamos para o nosso projeto para criarmos ele.
- Vamos acessar a pasta do projeto
  - ▷ Ir para o prompt de comando
  - ▷ Ir para a pasta do projeto
    - ▷ `cd\angular\cursoPWEBII\frontend <ENTER>`

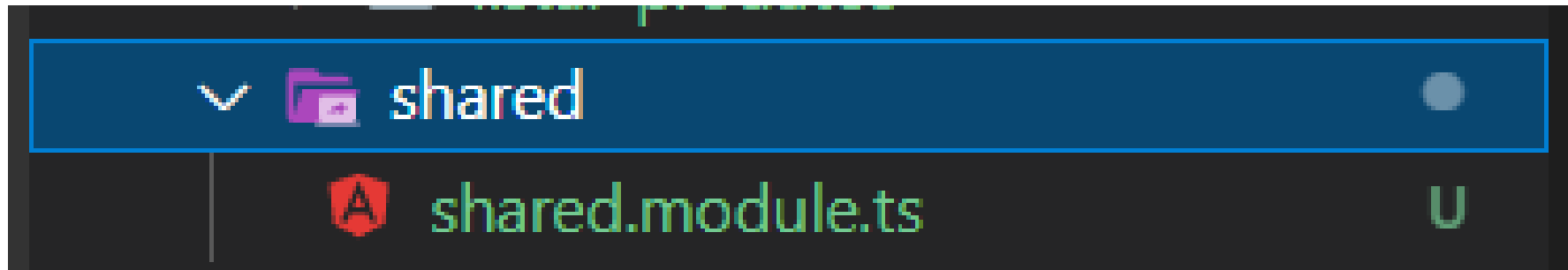
# Criação de componentes reutilizáveis

- Vamos criar o nosso módulo
  - `ng g module shared <ENTER>`
- Por que você não criou uma pastinha para organizar o nosso módulo? Por que o Angular, por padrão, quando ele gera um módulo, ele coloca o módulo dentro de uma pasta
- Vamos acessar o VsCode para verificar
  - `code . <ENTER>`

```
C:\angular\cursopwebII\frontend>ng g module shared
CREATE src/app/shared/shared.module.ts (192 bytes)
```

# Criação de componentes reutilizáveis

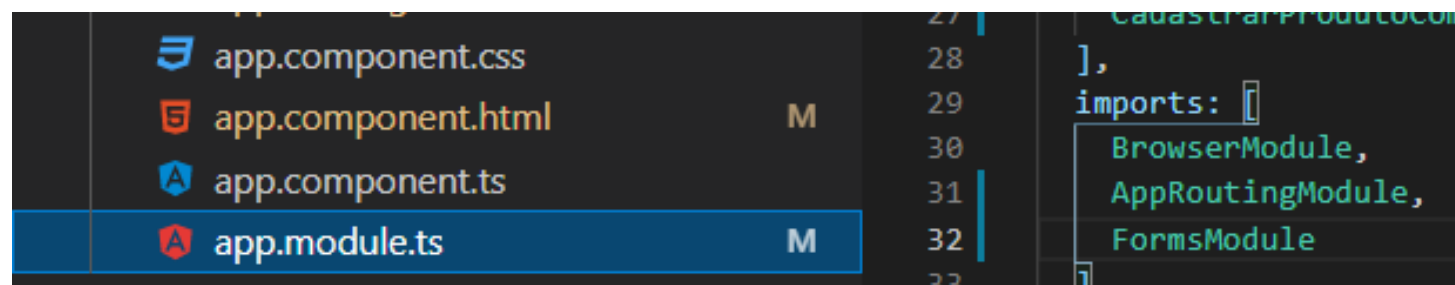
- Acessando o VsCode, verificamos que temos a pastinha shared e o módulo shared.module.ts



# Criação de componentes reutilizáveis

Abrindo o `shared.module.ts`, percebemos que esse módulo está bem vazio. Ele só possui o `CommonModule` aqui dentro dos imports, porque como já vimos na aula de Módulos, vamos ter dentro do `app.module`, por obrigação fazer a importação do `BrowserModule` e nos outros módulos que nossa aplicação for ter nós vamos fazer a importação do `CommonModule` que já irá trazer toda a estrutura do `BrowserModule`, já irá fazer a integração com os outros módulos importantes da nossa aplicação.

```
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3
4
5
6 @NgModule({
7   declarations: [],
8   imports: [
9     CommonModule
10  ]
11 })
12 export class SharedModule { }
```



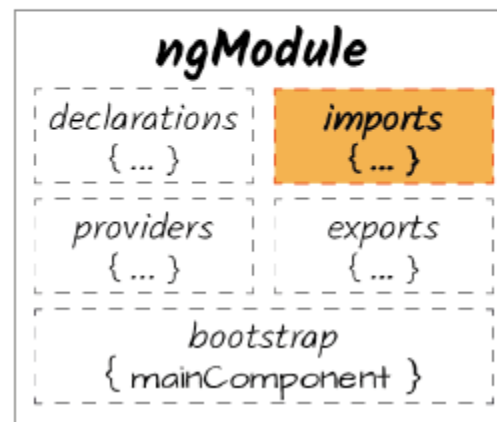
# imports

- Propriedade que define quais módulos a sua aplicação precisa importar para ter acesso aos Componentes, Diretivas e Pipes que foram exportados.
- Geralmente utilizado para importar bibliotecas de terceiros.

```
imports: [  
  HttpClientModule, // Módulo para tratar requisições HTTP.  
  FinanceiroModule, // Adiciona os componentes da área financeira.  
],
```

## MÓDULOS OBRIGATÓRIOS:

- ✓ **BrowserModule:** Deve ser importado no módulo principal da aplicação.
- ✓ **CommonModule:** Deve ser importado nos demais módulos.



# Criação de componentes reutilizáveis

- Agora vamos criar nosso componente reutilizável. Vamos para o prompt do DOS.
- Nós vamos reeditar nossa página home. Vocês podem perceber que estamos trabalhando muito em cima da nossa página home. Nós iremos reeditar a nossa página home, onde agora vamos transformar o card que temos lá em um componente reutilizável.
  - `ng g c shared/components/card-produto - -module=shared` <ENTER>
  - Após o card-produto, temos que colocar um argumento para que esse componente seja criado dentro do shared module e não dentro do app module. Vocês lembram que quando criamos um componente, o Angular já faz uma importação automática do nosso componente dentro do nosso modulo? Aqui, não queremos que o nosso componente card-produto entre dentro do app module, nós queremos que ele entre dentro do modulo shared.

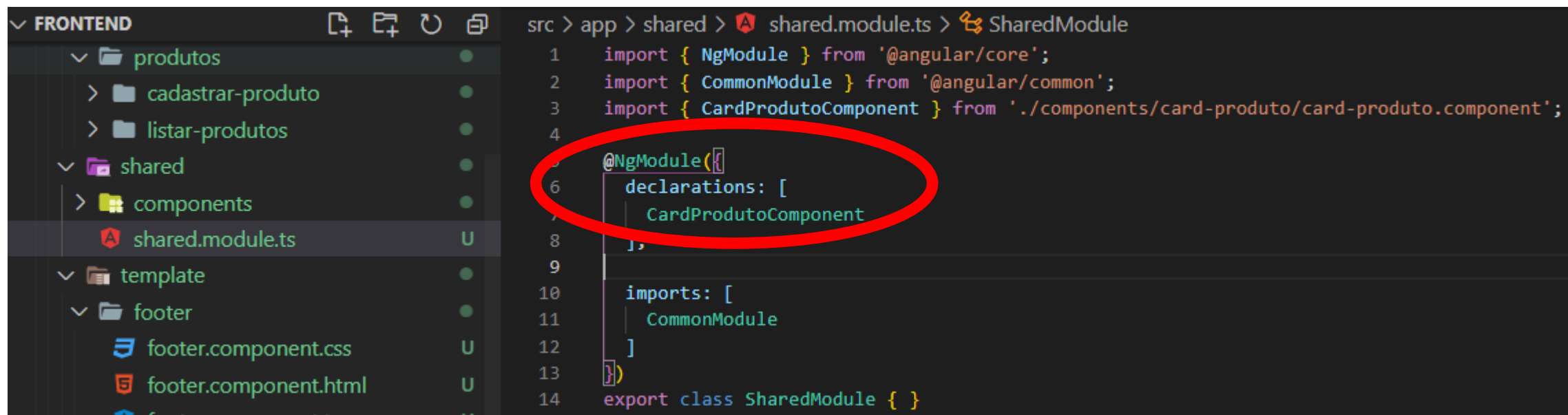
# Criação de componentes reutilizáveis

- Após digitar o comando, será atualizado o nosso shared.module

```
C:\angular\cursopwebII\frontend>ng g c shared/components/card-produto --module=shared
CREATE src/app/shared/components/card-produto/card-produto.component.html (27 bytes)
CREATE src/app/shared/components/card-produto/card-produto.component.ts (298 bytes)
CREATE src/app/shared/components/card-produto/card-produto.component.css (0 bytes)
UPDATE src/app/shared/shared.module.ts (308 bytes)
```

# Criação de componentes reutilizáveis

- Vamos verificar o nosso código. Vamos ter o CardProdutoComponent importado dentro do nosso módulo

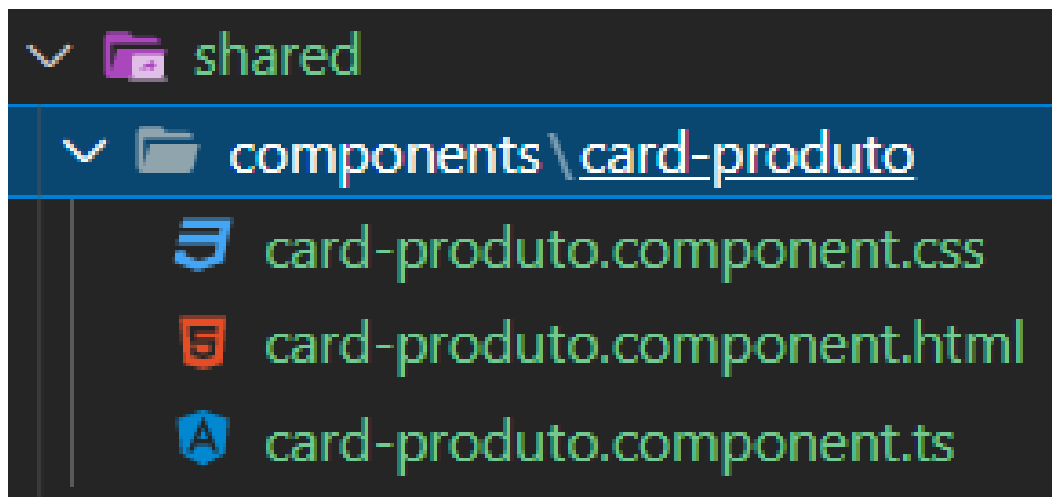


```
src > app > shared > shared.module.ts > SharedModule
1  import { NgModule } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3  import { CardProdutoComponent } from '../components/card-produto/card-produto.component';
4
5  @NgModule({
6    declarations: [
7      CardProdutoComponent
8    ],
9
10   imports: [
11     CommonModule
12   ],
13 })
14 export class SharedModule { }
```



# Criação de componentes reutilizáveis

- Vamos ver o componente. Ele foi criado dentro da pasta shared, dentro de uma pasta componentes. Vamos ter a pasta card-produto
- Dentro de card-produto, vamos ter o css, Html e ts



# Criação de componentes reutilizáveis

- Vamos abrir o nosso arquivo card-produto.html
- Vamos apagar o que conteúdo que está dentro
- Vamos abrir o [home.component.html](#) e recortar todo o card que nós criamos <CTRL><X>
  - ▶ div class="card"...

```
<div class="card" style="width: 18rem">
  <img [src]="foto" class="card-img-top" alt="..." />
  <div class="card-body">
    <h5 class="card-title">
      {{ nomeProduto | uppercase }}
      <span *ngIf="promocao == true; else elseBlock" class="badge badge-success">
        Promoção</span>
      <ng-template #elseBlock>
        <span class="badge badge-primary">Aproveite</span>
      </ng-template>
    </h5>
    <p class="card-text">Identificação: {{ idProduto | number: "2.0-0" }} <br>
      Validade: {{ dataValidade | date: 'shortDate' }}
    </p>
    <a href="#" class="btn btn-primary float-right">
      {{ precoProduto | currency: "BRL" }}</a>
    </div>
  </div>
```

# Criação de componentes reutilizáveis

Vamos colar no card-produto.component.html. Nós tiramos dentro do [home.component.html](#) e trouxemos para o card.component.html

```
src > app > shared > components > card-produto > card-produto.component.html > div.card
1  <div class="card" style="width: 18rem">
2    <img [src]="foto" class="card-img-top" alt="..." />
3    <div class="card-body">
4      <h5 class="card-title">
5        {{ nomeProduto | uppercase }}
6        <span *ngIf="promocao == true; else elseBlock" class="badge badge-success"
7          >Promoção</span>
8      >
9      <ng-template #elseBlock>
10       <span class="badge badge-primary">Aproveite</span>
11     </ng-template>
12   </h5>
13   <p class="card-text">Identificação: {{ idProduto | number: "2.0-0" }} <br>
14     Validade: {{ dataValidade | date : 'shortDate' }}
15   </p>
16   <a href="#" class="btn btn-primary float-right">
17     {{ precoProduto | currency: "BRL" }}</a>
18   >
19 </div>
20 </div>
21
```

# Criação de componentes reutilizáveis

- Agora, toda vez que quisermos um card de um produto, nós iremos chamar esse card que ele já irá trazer toda a estrutura pronta do card para nós.
- Vamos salvar o `card-produto.component.html`
- Agora vamos ter vários erros aqui, já que não temos essas variáveis criadas dentro do nosso `card-produto.component.ts`

# Criação de componentes reutilizáveis

- E como iremos fazer para receber as variáveis que estarão presentes dentro do nosso card?
- Nós iremos utilizar a notação @Input

# Função do decorator @Input

- Enviar informações de um componente Pai para um componente Filho.
- Um componente pode compartilhar dados e informações com outro componente passando dados ou eventos. Um componente pode ser usado dentro de outro componente, criando assim uma hierarquia de componentes.
- O componente que está sendo usado dentro de outro componente é conhecido como componente filho e o componente envolvente é conhecido como o componente pai. Os componentes podem se comunicar de várias maneiras através dos seguintes recursos:

- @Input ()
- @Output ()
- Services

# Função do decorator @Input

```
import { Component } from '@angular/core';

@Component({
  selector: 'artigo',
  template: `
    <h1>{{titulo}}</h1>
    <autoria [autor]="nome"> <autoria>
  `
})
export class AppComponent {
  titulo:string = 'Banco de dados na Web';
  nome:string = 'Macoratti';
}
```

app.component.ts



Componente  
Pai

```
import {Component, Input } from '@angular/core';

@Component({
  selector: 'autoria',
  template: `
    <h3> Escrito por : {{autor}} </h3>
  `
})
export class AutoriaComponent {
  @Input() autor : string;
}
```

autoria.component.ts



Componente  
Filho

# Função do decorator @Input

- No componente filho incluímos o decorator **@Input()**, definindo que a propriedade **autor** poderá ser alterada pelo componente pai. No exemplo ele vai passar a informação de **nome** para a variável **autor**.
- No componente pai incluímos o selector **autoria**, definido para componente filho, de forma a ser renderizado no selector **artigo** na tag **<artigo>** do arquivo **index.html**.
- A partir do componente pai, estamos definindo o valor da propriedade **autor** do componente filho. Para passar um valor para o componente filho, precisamos passar a propriedade do componente filho entre colchetes **[]** e definir seu valor para qualquer propriedade do componente pai.



# Função do decorator @Input

The screenshot shows the Angular website (angular.io) with a dark theme. At the top, there's a navigation bar with the Angular logo and links for FEATURES, DOCS, RESOURCES, EVENTS, and BLOG. A search bar contains the text 'sharing'. Below the navigation bar, there are five columns of links. The 'GUIDES (42)' column is highlighted with a red circle around the link 'Sharing data between child and parent directives and components'. Other links in this column include 'Sharing modules' and 'Guidelines for creating NgModules'. The 'API (13)' column lists 'AbstractControl', 'AbstractFormGroupDirective', 'resetFakeAsyncZone()', and 'HttpContext'. The 'CLI (1)' column lists 'Gathering and Viewing Usage Analytics'. The 'OTHER (1)' column lists 'Angular Contributors'. The 'TUTORIALS (7)' column lists 'Getting started with Angular', 'Tour of Heroes app and tutorial', and 'Deploying an application'.

angular.io

Apps Fatima Oracle MCamicado Outros SAP HTML Cisco Receitas ETEC Facebook Barracão

HELP ANGULAR BY TAKING A 1 MINUTE SURVEY! GO TO SURVEY

ANGULAR FEATURES DOCS RESOURCES EVENTS BLOG

sharing

API (13)

- AbstractControl
- AbstractFormGroupDirective
- resetFakeAsyncZone()
- HttpContext

CLI (1)

- Gathering and Viewing Usage Analytics

GUIDES (42)

- Sharing modules
- Guidelines for creating NgModules
- Sharing data between child and parent directives and components

OTHER (1)

- Angular Contributors

TUTORIALS (7)

- Getting started with Angular
- Tour of Heroes app and tutorial
- Deploying an application

# Função do decorator @Input

```
src > app > components > home > home.component.html >
1 <h1>Seja bem vindo ao curso de Angular</h1>
2 <h4 class="text-danger">{{ anuncio }}</h4>
3 <div class="row">
4   <div class="col">
5     <app-card-produto
6       [foto]="foto"
7       [nomeProduto]="nomeProduto"
8       [promocao]="promocao"
9       [idProduto]="idProduto"
10      [dataValidade]="dataValidade"
11      [precoProduto]="precoProduto"
12    ></app-card-produto>
13  </div>
```

home.component



**Componente  
Pai**

```
1 import { Component, Input, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-card-produto',
5   templateUrl: './card-produto.component.html',
6   styleUrls: ['./card-produto.component.css'],
7 })
8 export class CardProdutoComponent implements OnInit {
9   @Input()
10   foto: string = '';
11   @Input()
12   nomeProduto: string = '';
13   @Input()
14   promocao: boolean = true;
15   @Input()
16   idProduto: number = 0;
17   @Input()
18   dataValidade: string = '';
19   @Input()
20   precoProduto: number = 0;
21
22   constructor() {}
23
24   ngOnInit(): void {}
25 }
```

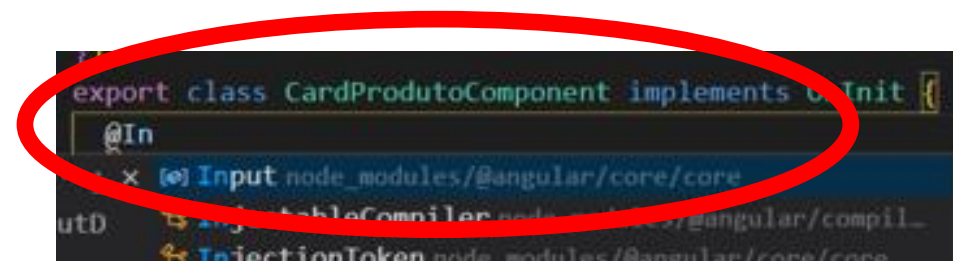
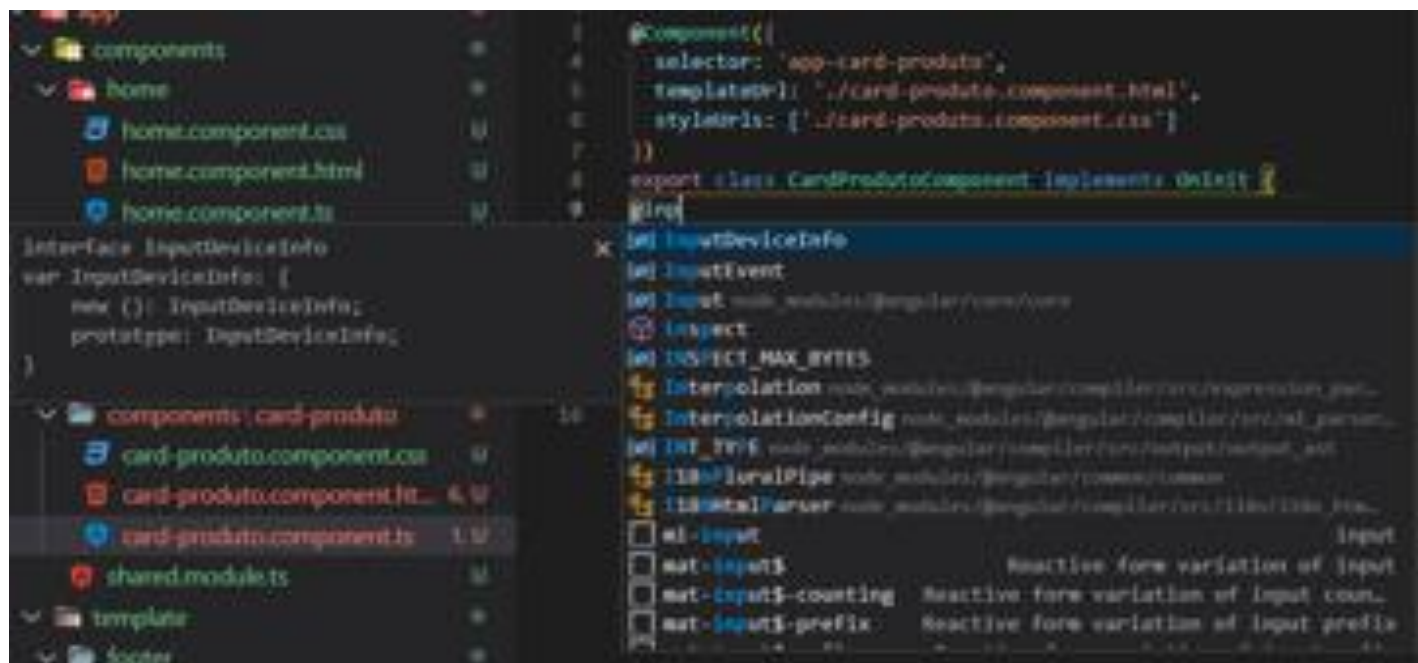
card-produto.component.ts



**Componente  
Filho**

# Criação de componentes reutilizáveis

- E como iremos fazer para receber as variáveis que estarão presentes dentro do nosso card?
- Nós iremos utilizar a notação @In (Ao começar a digitar, já irá aparecer Input (que irá fazer a importação do input que está no core do Angular)



# Criação de componentes reutilizáveis

Ele já irá importar para o Angular, conforme figura abaixo:

```
1  import { Component, Input, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-card-produto',
5    templateUrl: './card-produto.component.html',
6    styleUrls: ['./card-produto.component.css']
7  })
8  export class CardProdutoComponent implements OnInit {
9    @Input()
10    constructor() { }
11
12    ngOnInit(): void {
13    }
14
15  }
```

# Criação de componentes reutilizáveis

- Após o input, iremos abrir e fechar parênteses e colocar o nome da nossa variável que será foto e informar o tipo da variável que será string
  - ▷ @Input() foto:string = '';
- Vamos formatar o código: <ALT><SHIFT><F>
- Vamos para os demais campos:

```
8  export class CardProdutoComponent implements OnInit {  
9      @Input() foto: string = '';  
10  
11      constructor() {}  
12  
13      ngOnInit(): void {}  
14  }
```

# Criação de componentes reutilizáveis

Os outros campos que precisamos receber são: nome do produto, promoção, idProduto, data validade e preco de produto... Vamos criar eles. Vamos copiar a linha e colar 2x, depois só alteramos o conteúdo.

```
8  export class CardProdutoComponent implements OnInit {  
9      @Input() foto: string = '';  
10     @Input() nomeProduto: string = '';  
11     @Input() promocao: boolean = true;  
12     @Input() idProduto: number = 0;  
13     @Input() dataValidade: string = '';  
14     @Input() precoProduto: number = 0;  
15  
16     constructor() {}  
17  
18     ngOnInit(): void {}  
19 }
```

# Criação de componentes reutilizáveis

- Verifica que quando criamos as inputs com o nome de variáveis e seu tipo dentro da nossa classe já sumiu os erros das nossas variáveis que tínhamos aqui dentro do `card-produto.component`.
- Agora o nosso `card-produto` espera receber uma foto, um nome de produto, uma promoção, um `idProduto`, uma data de validade e um preço de um produto.
- Agora vamos utilizar nosso componente dentro do nosso home.
- Porém, antes de ir para lá, precisamos exportar esse componente para que ele fique visível dentro do `app.module`.
- Como iremos fazer isso, vamos abrir o `sharedModule.ts`, inserir uma vírgula após `CommonModule` , digitar a propriedade `exports`, dois pontos : colchetes e dentro vamos digitar `CardProdutoComponent`.

# Criação de componentes reutilizáveis

Como iremos fazer isso, vamos abrir o `sharedModule.ts`, inserir uma vírgula após `CommonModule`, digitar a propriedade `exports`, dois pontos : colchetes e dentro vamos digitar `CardProdutoComponent`.



`shared.module.ts`

```
10     imports: [  
11         |   CommonModule  
12     ],  
13     exports:[CardProdutoComponent]  
14  
15 })  
16 export class SharedModule { }
```



# Criação de componentes reutilizáveis

- Vamos fechar os arquivos abertos
- Para que eu possa utilizar o card-produto eu preciso importar o shared.module dentro do app.module.ts
- Para isso, vamos acessar o app.module.ts e vamos importar o SharedModule. Quando começar a digitar já irá aparecer para importar. É o SharedModule sem a descrição.

```
30 imports: [  
31     BrowserModule,  
32     AppRoutingModule,  
33     FormsModule,  
34     SharedModule  
35 ],
```

[illegible]

# Diferença entre imports e declarations

- O **app.module** é o que reúne tudo, todas as views e partes flexíveis do sistema. Ele precisa definir todas as páginas/views e um array e todas as partes reutilizáveis.
- **declarations** - > Você pode utilizar. Sempre inserimos os componentes aqui.
- **import** -> importa código de outras classes para que você utilize. Sempre inserimos módulos aqui.

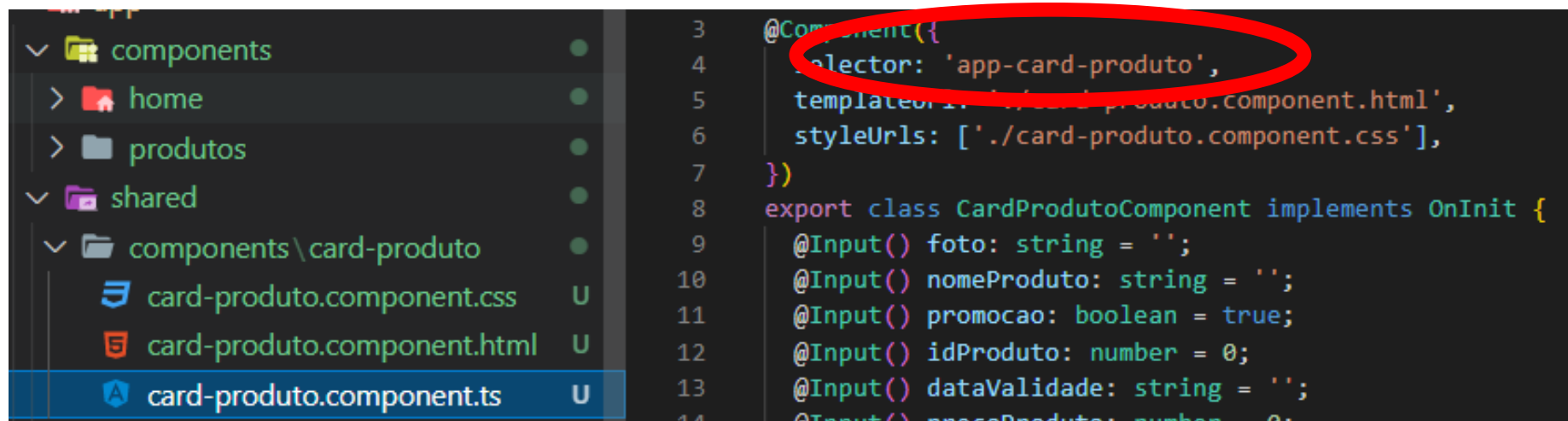
# Criação de componentes reutilizáveis

- E a importação já foi realizada.
- Vamos salvar

```
1 | import { SharedModule } from '../shared/shared.module';
```

# Criação de componentes reutilizáveis

- Agora vamos para o [home.component.html](#)
- O que precisamos inserir nele? Precisamos chamar o card-produto. Para isso, vamos abrir o card-produto.component.ts e verificar o nome que está sendo exibido no selector. Vamos copiar esse nome e colar no [home.component.html](#).
  - ▷ app-card-produto



```
3 @Component({
4   selector: 'app-card-produto',
5   templateUrl: './card-produto.component.html',
6   styleUrls: ['./card-produto.component.css'],
7 })
8 export class CardProdutoComponent implements OnInit {
9   @Input() foto: string = '';
10  @Input() nomeProduto: string = '';
11  @Input() promocao: boolean = true;
12  @Input() idProduto: number = 0;
13  @Input() dataValidade: string = '';
14  @Input() precoProduto: number = 0;
```

# Criação de componentes reutilizáveis

Vamos colar no `home.component.html` , inserindo as tags de html

```
1  <h1>Seja bem vindo ao curso de Angular!</h1>
2
3  <h4 class="text-danger">{{ anuncio }}</h4>
4
5  <app-card-produto></app-card-produto>
6
```

# Criação de componentes reutilizáveis

- O que está faltando agora? Passar os parâmetros para dentro do nosso card.
- Vamos dar um <ENTER> após o <app-card-produto e vamos começar a escrever as propriedades que estamos esperando receber aqui.
- Vamos escrever, entre colchetes foto = "foto" -> a variável também chama foto dentro do meu home.component, vai seguir com o mesmo nome para o outro lado. Fazer isso com as demais variáveis.

```
5  <app-card-produto
6    [foto]="foto"
7    [nomeProduto]="nomeProduto"
8    [promocao]="promocao"
9    [idProduto]="idProduto"
10   [dataValidade]="dataValidade"
11   [precoProduto]="precoProduto"
12 ></app-card-produto>
```

Lembrem-se que isso daqui é o property binding, então estou pegando dados que estão dentro do arquivo ts, por isso estão entre colchetes []

# Property binding

Property binding – utilizamos os colchetes porque precisamos da informação entre aspas.

Type	Syntax	Category
Interpolation Property Attribute Class Style	<pre>{{expression}} [target]="expression"</pre> 	One-way from data source to view target

# Criação de componentes reutilizáveis

- Já passei agora para dentro do meu app.card-produto todas as variáveis que eu tinha dentro do meu home, Estou pegando as variáveis que estão dentro do meu home e passando para card-produto.components através da notação @Input.
- Agora vamos executar nossa aplicação
  - ▷ `ng serve -o` <ENTER>



# Criação de componentes reutilizáveis

Agora nós temos aqui o nosso card. Parece até que não modifiquei ele, mas agora isso daqui é um componente.

CRUD Angular Home Produtos ▾

## Seja bem vindo ao curso de Angular!

O Curso de Angular está em promoção



CREATE READ UPDATE DELETE

C R U D

CURSO DE ANGULAR

Promoção

Identificação: 03

Validade: 31/12/2021

R\$ 2,51

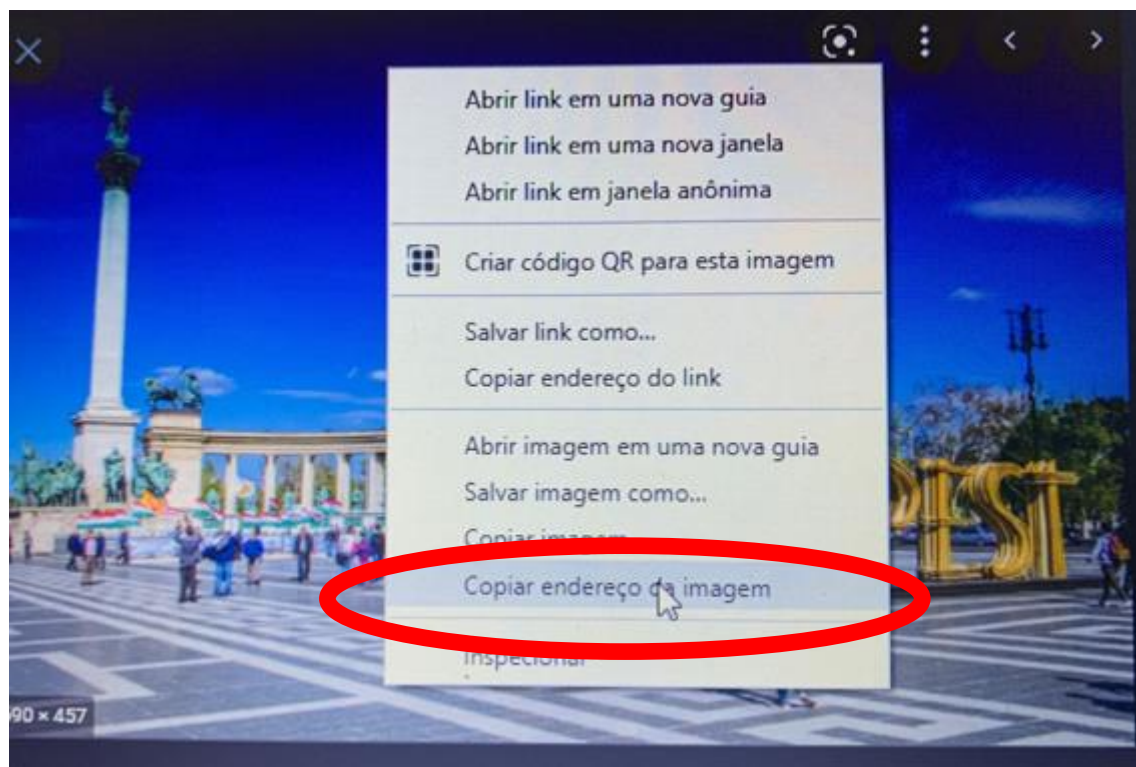
# Criação de componentes reutilizáveis

- Eu posso estar duplicando esse componente na minha tela.
- Acessar o [home.component.html](#)
- Vamos duplicar o conteúdo do que está dentro da tag <app-card-produto

```
1 <h1>Seja bem vindo ao curso de Angular!</h1>
2
3 <h4 class="text-danger">{{ anuncio }}</h4>
4
5 <app-card-produto
6   [foto]="foto"
7   [nomeProduto]="nomeProduto"
8   [promocao]="promocao"
9   [idProduto]="idProduto"
10  [dataValidade]="dataValidade"
11  [precoProduto]="precoProduto"
12 ></app-card-produto>
13
14 <app-card-produto
15   [foto]="foto"
16   [nomeProduto]="nomeProduto"
17   [promocao]="promocao"
18   [idProduto]="idProduto"
19   [dataValidade]="dataValidade"
20   [precoProduto]="precoProduto"
21 ></app-card-produto>
```

# Criação de componentes reutilizáveis

- Vamos alterar o conteúdo da foto, buscando uma imagem na internet.
- Vamos entrar no google e procurar por uma foto angular. Clique em imagem, copie o link da imagem e cole no lugar da foto (Copiar endereço da imagem)



# Criação de componentes reutilizáveis

Vamos colar no lugar da foto (Copiar endereço da imagem)

```
5 <app-card-produto
6   [foto]="foto"
7   [nomeProduto]="nomeProduto"
8   [promocao]="promocao"
9   [idProduto]="idProduto"
10  [dataValidade]="dataValidade"
11  [precoProduto]="precoProduto"
12 ></app-card-produto>
13
14 <app-card-produto
15   [foto]="data:image/jpeg;base64,/9j/4AAQSkZJRgABAQAAQABAAD/2wCEAAoHCBQUExcSFBQXFxcYFxcaGBoX
16   [nomeProduto]="nomeProduto"
17   [promocao]="promocao"
18   [idProduto]="idProduto"
19   [dataValidade]="dataValidade"
20   [precoProduto]="precoProduto"
21 ></app-card-produto>
```

# Criação de componentes reutilizáveis

Vamos alterar os demais conteúdos

```
14 <app-card-produto
15   [foto]="data:image/jpeg;base64,/9j/4AAQSkZJRgABAQAAQABAAAD/2wCEAAoHCBQUExc5FBQXFxcYFxcaGBoXGE
16   [nomeProduto]="Viagem a Budapest"
17   [promocao]="false"
18   [idProduto]="15"
19   [dataValidade]="2021-12-31"
20   [precoProduto]="5434.00"
21 ></app-card-produto>
```

# Criação de componentes reutilizáveis

- Por que está dando erro?
- Quando colocamos colchetes, o que está dentro de aspas, vira código JS, é como se estivéssemos no arquivo .js ou .ts
- Estamos tratando como variáveis e não temos mais variáveis. Vamos apagar os colchetes somente dos campos string.
- Precisamos apenas passar a propriedade, sem passar os colchetes porque os colchetes é o que faz nosso componente ir buscar esses dados dentro do meu `home.component.ts`.

# Criação de componentes reutilizáveis

```
14  <app-card-produto  
15    foto="data:image/jpeg;base64,/9j/4AAQSkZJRgABAQAAQABAAAD/"  
16    nomeProduto="Viagem a Budapest"  
17    [promocao]="false"  
18    [idProduto]="15"  
19    dataValidade = "2021-12-31"  
20    [precoProduto] = "5434.00"  
21  ></app-card-produto>
```

# Criação de componentes reutilizáveis

- Vamos arrumar, colocando colunas
- Após o <h4>, digitar div.row <TAB>
- Dentro, digitar div.col <TAB>

```
2
3   <h4 class="text-danger">{{ anuncio }}</h4>
4
5   <div class="row">
6     <div class="col">
7
8
9     </div>
10  </div>
```



# Criação de componentes reutilizáveis

- Vamos colocar o primeiro card dentro da col, movendo o conteúdo
- <CTRL><X> <CTRL><V>

```
5 <div class="row">
6   <div class="col">
7     <app-card-produto
8       [foto]="foto"
9       [nomeProduto]="nomeProduto"
10      [promocao]="promocao"
11      [idProduto]="idProduto"
12      [dataValidade]="dataValidade"
13      [precoProduto]="precoProduto"
14    ></app-card-produto>
15  </div>
16 </div>
```

# Criação de componentes reutilizáveis

- Vamos criar uma segunda coluna, antes do último div.
- Digitar div.col <TAB>
- Inserir o segundo card lá dentro (mover o card de lugar) <CTRL><X> <CTRL><V>

```
16   <div class="col">
17     <app-card-produto
18       foto="data:image/jpeg;base64,/9j/4AAQSkZJRgABAQAAQABAAD/"
19       nomeProduto="Viagem a Budapest"
20       [promocao]="false"
21       [idProduto]="15"
22       dataValidade="2021-12-31"
23       [precoProduto]="5434.0"
24     ></app-card-produto>
25
26   </div>
27 </div>
```

# Criação de componentes reutilizáveis

- Vejam que estamos trabalhando com componentes reutilizáveis, vejam que só precisei passar o app-card e já consegui passar propriedades para ele para que eu possa reutilizá-lo quantas vezes eu precisar.
- Vamos salvar e verificar como ficou nossa página.

CRUD Angular Home Produtos ▾

## Seja bem vindo ao curso de Angular!

O Curso de Angular está em promoção

  
CREATE READ UPDATE DELETE

# CRUD

### CURSO DE ANGULAR

**Promoção**

Identificação: 03  
Validade: 31/12/2021

R\$ 2,51



### VIAGEM A BUDAPEST

**Aproveite**

Identificação: 15  
Validade: 31/12/2021

R\$ 5.434,00

# Criação de componentes reutilizáveis

- Vejam que estamos trabalhando com componentes reutilizáveis, vejam que só precisei passar o app-card e já consegui passar propriedades para ele para que eu possa reutilizá-lo quantas vezes eu precisar.
- Vamos salvar e verificar como ficou nossa página.

# Resumo

- 1) Criaremos um componente reutilizável referente ao card que está na [home.component.html](#)
  - ▶ Criaremos um modulo denominado shared
  - ▶ Dentro do modulo, criaremos uma pasta denominada components e dentro da pasta o componente card-produto
  - ▶ No componente card-produto conterá os comandos do [home.component.html](#)
- 2) Temos que exportar a classe que temos do componente filho dentro do SharedModule.ts

```
imports: [  
  CommonModule  
]  
exports: [  
  CardProdutoComponent  
]  
})  
export class SharedModule { }
```

# Resumo

- 3) Devemos importar o SharedModule dentro do app.module.ts

```
imports: [  
  BrowserModule,  
  AppRoutingModule,  
  FormsModule,  
  SharedModule  
],
```

# Resumo

- 4) Utilizaremos a notação @Input para receber as variáveis que estão no nosso componente pai
- 5) Vamos utilizar nosso componente dentro da nossa home, passando o selector em forma de tag `<app-card-produto> </app-card-produto>`
  - ▶ Dentro da tag vamos inserir as propriedades que esperamos receber

# Resumo

```
src > app > components > home > home.component.html >
1 <h1>Seja bem vindo ao curso de Angular</h1>
2 <h4 class="text-danger">{{ anuncio }}</h4>
3 <div class="row">
4   <div class="col">
5     <app-card-produto
6       [foto]="foto"
7       [nomeProduto]="nomeProduto"
8       [promocao]="promocao"
9       [idProduto]="idProduto"
10      [dataValidade]="dataValidade"
11      [precoProduto]="precoProduto"
12    ></app-card-produto>
13  </div>
```

home.component



**Componente  
Pai**

```
1 import { Component, Input, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-card-produto',
5   templateUrl: './card-produto.component.html',
6   styleUrls: ['./card-produto.component.css'],
7 })
8 export class CardProdutoComponent implements OnInit {
9   @Input()
10   foto: string = '';
11   @Input()
12   nomeProduto: string = '';
13   @Input()
14   promocao: boolean = true;
15   @Input()
16   idProduto: number = 0;
17   @Input()
18   dataValidade: string = '';
19   @Input()
20   precoProduto: number = 0;
21
22   constructor() {}
23
24   ngOnInit(): void {}
25
26 }
```

card-produto.component.ts



**Componente  
Filho**



