**Laboratorio – Navegación en Angular**

Este lab se basa en el laboratorio anterior, implementación de un proyecto básico en Angular.

En esta etapa de desarrollo, la aplicación de la tienda en línea tiene un catálogo de productos básico.

En las siguientes secciones, agregará las siguientes funciones a la aplicación:

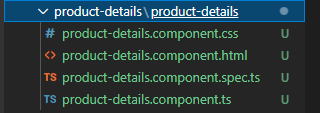
* Escriba una URL en la barra de direcciones para navegar a la página del producto correspondiente.
* Haga clic en los enlaces de la página para navegar dentro de su aplicación de una sola página.
* Haga clic en los botones de avance y retroceso del navegador para navegar por el historial del navegador de forma intuitiva.

**Asociar una ruta de URL con un componente**

La aplicación ya usa el enrutador angular para navegar hasta ProductListComponent. Esta sección le muestra cómo definir una ruta para mostrar detalles de productos individuales.

1. Genere un nuevo componente para los detalles del producto. En la terminal, genere un nuevo componente de detalles del producto ejecutando el siguiente comando:

ng generate component product-details



1. En app.module.ts, agregar un route para *product details*, con una ruta de products/:productId y ProductDetailsComponent para el componente.

src/app/app.module.ts

content\_copy@[NgModule](https://angular.io/api/core/NgModule)({

imports: [

[BrowserModule](https://angular.io/api/platform-browser/BrowserModule),

[ReactiveFormsModule](https://angular.io/api/forms/ReactiveFormsModule),

RouterModule.forRoot([

{ path: '', component: ProductListComponent },

{ path: 'products/:productId', component: ProductDetailsComponent },

])

],

declarations: [

AppComponent,

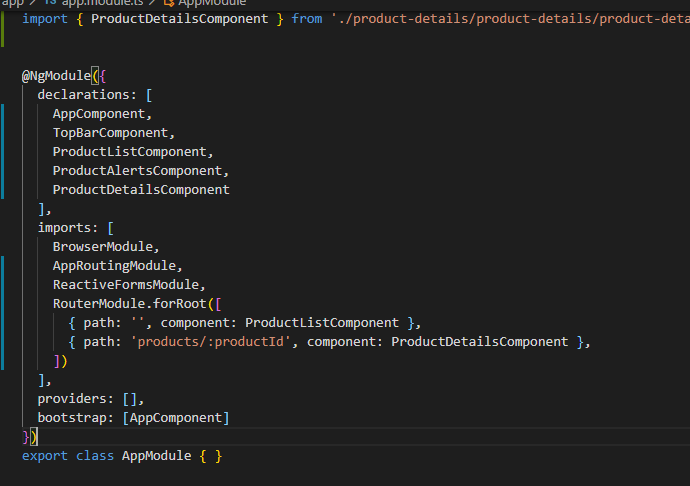
TopBarComponent,

ProductListComponent,

ProductAlertsComponent,

ProductDetailsComponent,

],



1. Abrir product-list.component.html.
2. 4. Modifique el ancla del nombre del producto para incluir un routerLink con product.id como parámetro.

***product-list.component.html***

<div \*[ngFor](https://angular.io/api/common/NgForOf)="let product of products">

<h3>

<a

[title]="product.name + ' details'"

[[routerLink](https://angular.io/api/router/RouterLink)]="['/products', product.id]">

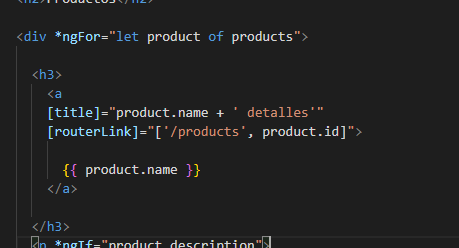
{{product.name}}

</a>

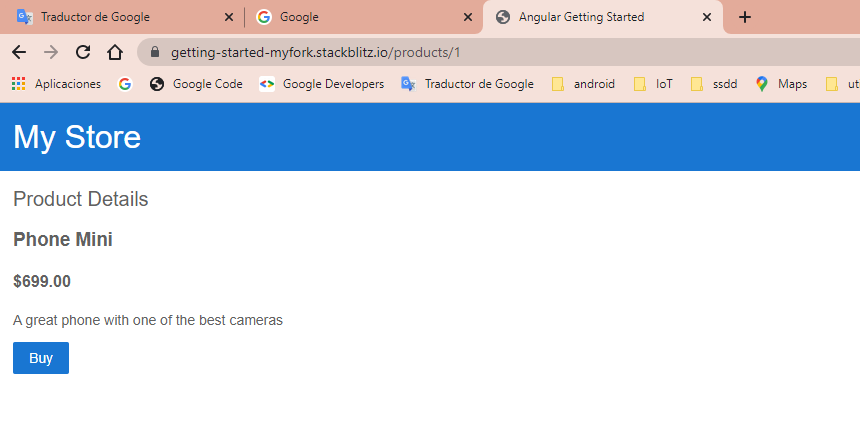
</h3>

<!-- . . . -->

</div>



La directiva RouterLink le ayuda a personalizar el elemento de anclaje. En este caso, la ruta, o URL, contiene un segmento fijo, / productos. El segmento final es variable, insertando la propiedad id del producto actual. Por ejemplo, la URL de un producto con un ID de 1 sería similar a https://getting-started-myfork.stackblitz.io/products/1.



5. Verifique que el enrutador funcione según lo previsto haciendo clic en el nombre del producto. La aplicación debería mostrar ProductDetailsComponent, que actualmente dice "¡los detalles del producto funcionan!"

Observe que cambia la URL en la ventana de vista previa. El segmento final es productos / # donde # es el número de la ruta en la que hizo clic.

|  |  |
| --- | --- |
|  |  |

**Ver detalles del producto**

ProductDetailsComponent maneja la visualización de cada producto. El enrutador angular muestra componentes basados en la URL del navegador y sus rutas definidas.

En esta sección, usará el enrutador angular para combinar los datos de los productos y la información de la ruta para mostrar los detalles específicos de cada producto.

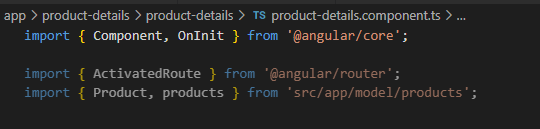
1. En product-details.component.ts, agregar import [ActivatedRoute](https://angular.io/api/router/ActivatedRoute) from @angular/router, y el array de products de../products.

***product-details.component.ts***

import { [Component](https://angular.io/api/core/Component), [OnInit](https://angular.io/api/core/OnInit) } from '@angular/core';

import { [ActivatedRoute](https://angular.io/api/router/ActivatedRoute) } from '@angular/router';

import { Product, products } from '../products';



1. Definir la propiedad producto en producto detail.

***product-details.component.ts***

export class ProductDetailsComponent implements [OnInit](https://angular.io/api/core/OnInit) {

product: Product | undefined;

/\* ... \*/

}

1. Inject [ActivatedRoute](https://angular.io/api/router/ActivatedRoute) en el constructor() by agregando *private route:*[*ActivatedRoute*](https://angular.io/api/router/ActivatedRoute) como un argumento dentro de los paréntesis del constructor.

***product-details.component.ts***

content\_copyexport class ProductDetailsComponent implements [OnInit](https://angular.io/api/core/OnInit) {

product: Product | undefined;

constructor(private route: [ActivatedRoute](https://angular.io/api/router/ActivatedRoute)) { }

}

ActivatedRoute es específico para cada componente que carga el enrutador angular. ActivatedRoute contiene información sobre la ruta y los parámetros de la ruta.

Al inyectar ActivatedRoute, está configurando el componente para usar un servicio.

1. En el metodo ngOnInit(), extraer *productId* de los parámetros de route y encuentra el correspondiente producto en el array products.

***product-details.component.ts***

ngOnInit() {

// First get the product id from the current route.

const routeParams = this.route.snapshot.paramMap;

const productIdFromRoute = Number(routeParams.get('productId'));

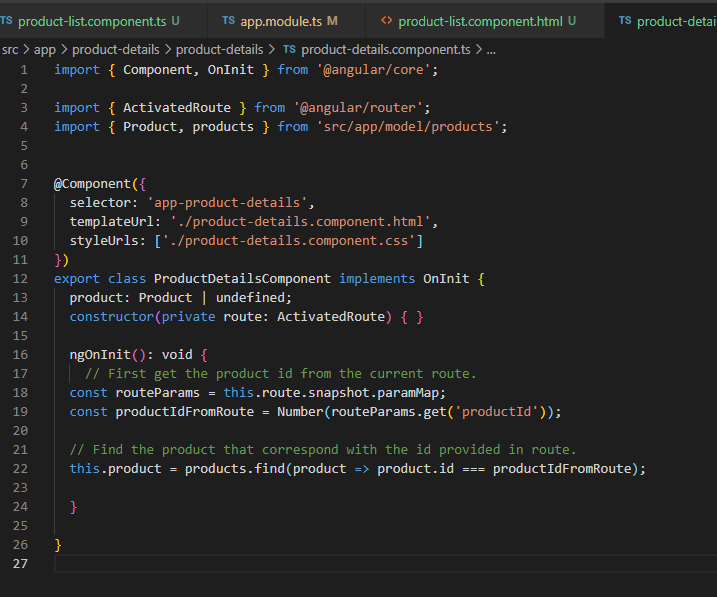
// Find the product that correspond with the id provided in route.

this.product = products.find(product => product.id === productIdFromRoute);

}

Los parámetros de la ruta corresponden a las variables de ruta que define en la ruta. Para acceder a los parámetros de la ruta, usamos route.snapshot, que es la ActivatedRouteSnapshot que contiene información sobre la ruta activa en ese momento en particular. La URL que coincide con la ruta proporciona el productId. Angular usa productId para mostrar los detalles de cada producto único.

Con todo ello, el código quedara asi:



1. Actualice la plantilla ProductDetailsComponent para mostrar los detalles del producto con un \* ngIf. Si existe un producto, el <div> se representa con un nombre, precio y descripción.

***product-details.component.html***

<h2>Product Details</h2>

<div \*[ngIf](https://angular.io/api/common/NgIf)="product">

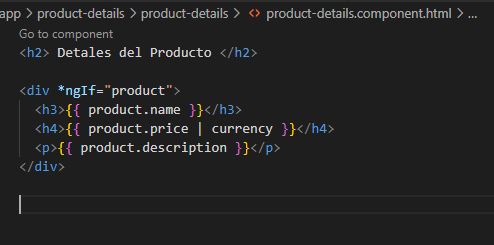
<h3>{{ product.name }}</h3>

<h4>{{ product.price | [currency](https://angular.io/api/common/CurrencyPipe) }}</h4>

<p>{{ product.description }}</p>

</div>

La línea, <h4> {{product.price | currency}} </h4>, utiliza la tubería de moneda para transformar product.price de un número a una cadena de moneda. Una tubería es una forma en que puede transformar datos en su plantilla HTML.



Cuando los usuarios hacen clic en un nombre en la lista de productos, el enrutador los dirige a la URL distinta del producto, muestra ProductDetailsComponent, y muestra los detalles del producto.

|  |  |
| --- | --- |
|  |  |

**Manejo de datos**

En esta etapa de desarrollo, la aplicación de la tienda tiene un catálogo de productos con dos vistas: una lista de productos y detalles del producto. Los usuarios pueden hacer clic en el nombre de un producto de la lista para ver los detalles en una nueva vista, con una URL o ruta distinta.

Este paso del tutorial lo guía a través de la creación de un carrito de compras en las siguientes fases:

* Actualice la vista de detalles del producto para incluir un botón Comprar, que agrega el producto actual a una lista de productos que administra un servicio de carrito.
* Agregue un componente de carrito, que muestra los artículos en el carrito.
* Agregue un componente de envío, que recupera los precios de envío de los artículos en el carrito utilizando HttpClient de Angular para recuperar los datos de envío de un archivo .json.

## Crear el servicio shopping cart

En Angular, un servicio es una instancia de una clase que se puede poner a disposición de cualquier parte de su aplicación utilizando el sistema de inyección de dependencias de Angular.

Actualmente, los usuarios pueden ver la información del producto y la aplicación puede simular el intercambio y las notificaciones sobre los cambios del producto.

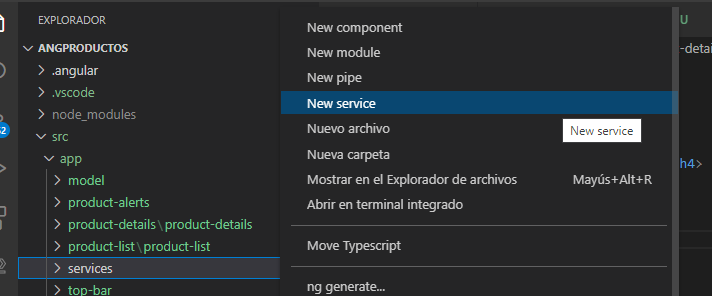
El siguiente paso es crear un formulario para que los usuarios agreguen productos a un carrito. Este procedimiento lo guía para agregar un botón **Buy** y configurar un servicio de carrito para almacenar información sobre los productos en el carrito.

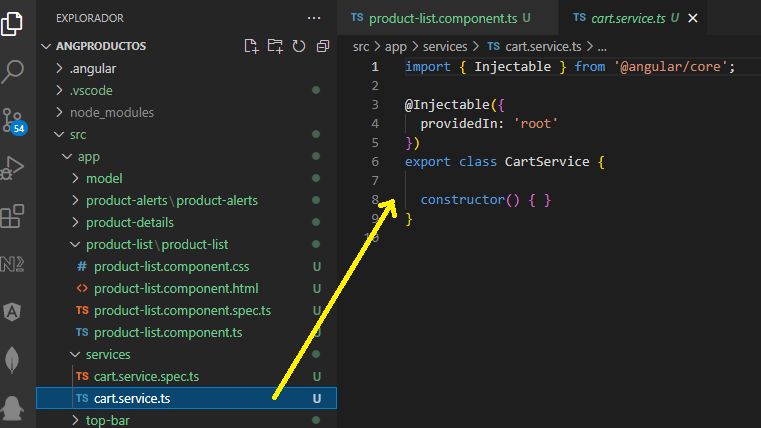
**Definir un servicio de carrito**

Esta sección lo guía a través de la creación del CartService que rastrea los productos agregados al carrito de compras.

1. En la terminal, genere un nuevo servicio de carrito ejecutando el siguiente comando:

ng generate service cart





1. Hacer Import con la interfaz Product de ./products.ts en el archivo *cart.service.ts* y, en la clase CartService, defina una propiedad ítems para almacenar la matriz de los actuales productos en el carrito.

***cart.service.ts***

import { Product } from './products';

/\* . . . \*/ In

export class CartService {

items: Product[] = [];

/\* . . . \*/

}

1. Definir los metodos para añadir items al carrito, devolver ítems del carrito, y limpiar los ítems del carrito.

***cart.service.ts***

export class CartService {

items: Product[] = [];

/\* . . . \*/

addToCart(product: Product) {

this.items.push(product);

}

getItems() {

return this.items;

}

clearCart() {

this.items = [];

return this.items;

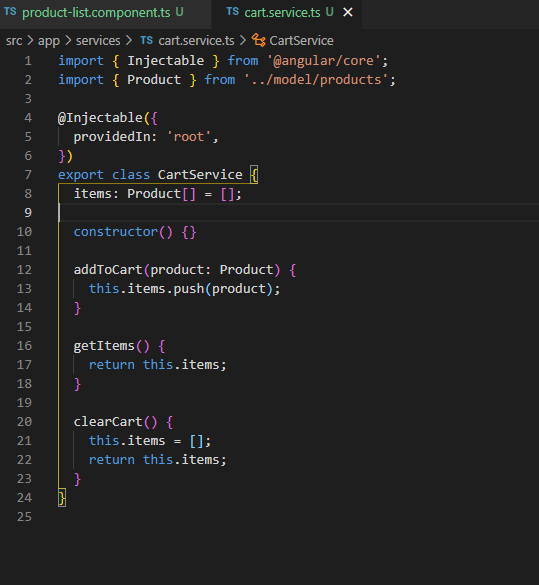
}

/\* . . . \*/

}

* + El método addToCart () agrega un producto a una matriz de elementos.
  + El método getItems () recopila los artículos que los usuarios agregan al carrito y devuelve cada artículo con su cantidad asociada.
  + El método clearCart () devuelve una matriz vacía de elementos, que vacía el carrito.

El código del Service quedara de la siguiente manera:



### **Uso del servicio cart**

A continuación, se muestra el uso de CartService para agregar un producto al carrito.

1. En *product-details.component.ts*, hacer import al servicio cart.

***product-details.component.ts***

import { [Component](https://angular.io/api/core/Component), [OnInit](https://angular.io/api/core/OnInit) } from '@angular/core';

import { [ActivatedRoute](https://angular.io/api/router/ActivatedRoute) } from '@angular/router';

import { Product, products } from '../products';

import { CartService } from '../cart.service';

1. Inyectar el servicio cart agregándolo al constructor().

***product-details.component.ts***

export class ProductDetailsComponent implements [OnInit](https://angular.io/api/core/OnInit) {

constructor(

private route: [ActivatedRoute](https://angular.io/api/router/ActivatedRoute),

private cartService: CartService

) { }

}

1. Definir el metodo addToCart(), que agrega el producto actual al carrito.

***product-details.component.ts***

export class ProductDetailsComponent implements [OnInit](https://angular.io/api/core/OnInit) {

addToCart(product: Product) {

this.cartService.addToCart(product);

window.alert('Your product has been added to the cart!');

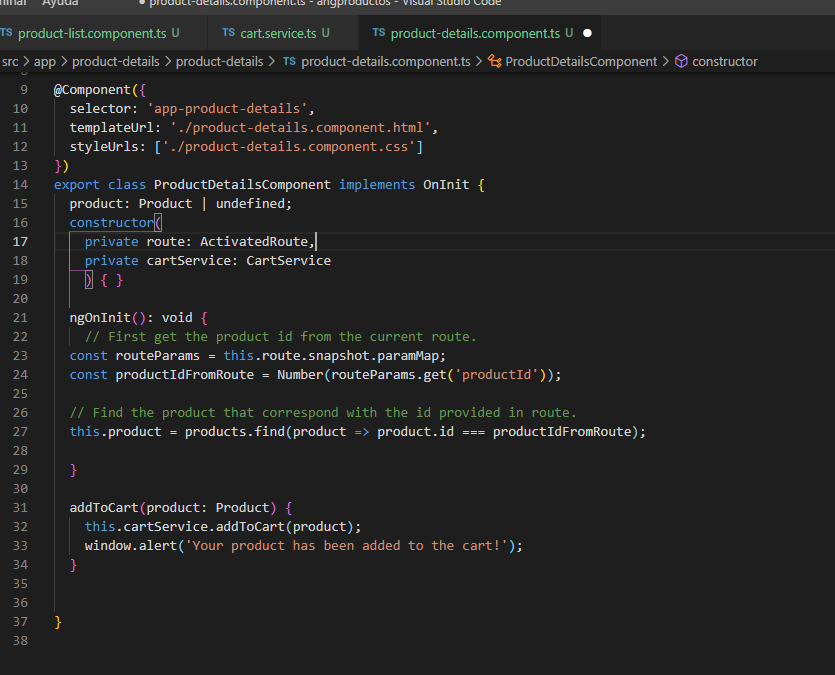
}

}

El método addToCart () hace lo siguiente:

* + Toma el producto actual como argumento.
  + Utiliza el método CartService addToCart () para agregar el producto al carrito.
  + Muestra un mensaje de que ha agregado un producto al carrito.

Completado el código la clase queda de la siguiente manera:



1. En *product-details.component.html*, agregue un botón con la etiqueta Buy y vincule el evento click () al método addToCart (). Este código actualiza la plantilla de detalles del producto con un botón Comprar que agrega el producto actual al carrito.

***product-details.component.html***

<h2>Product Details</h2>

<div \*[ngIf](https://angular.io/api/common/NgIf)="product">

<h3>{{ product.name }}</h3>

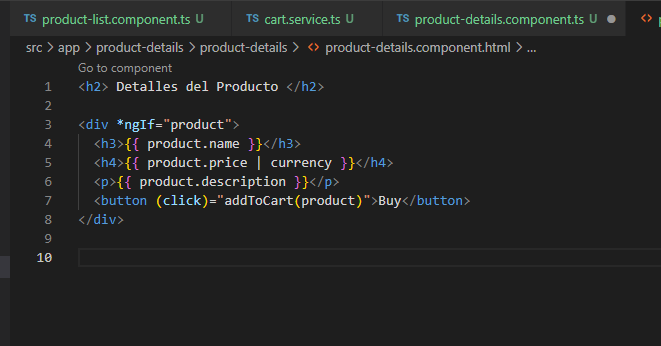
<h4>{{ product.price | [currency](https://angular.io/api/common/CurrencyPipe) }}</h4>

<p>{{ product.description }}</p>

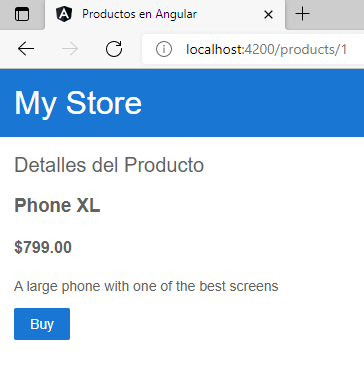
<button (click)="addToCart(product)">Buy</button>

</div>

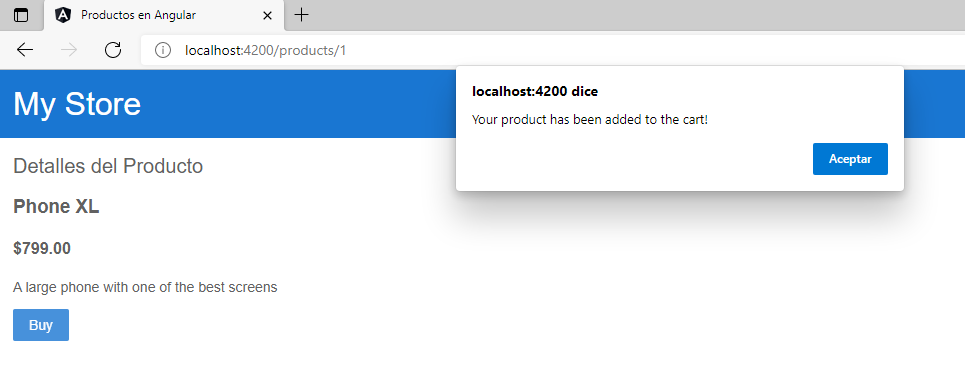
Asi queda el código:



1. Verifique que el nuevo botón Buy aparece como se esperaba actualizando la aplicación y haciendo clic en el nombre de un producto para mostrar sus detalles.



1. Haga clic en el botón Buy para agregar el producto a la lista almacenada de artículos en el carrito y mostrar un mensaje de confirmación.



## Crear la vista del carrito

Para que los clientes vean su carrito, puede crear la vista del carrito en dos pasos:

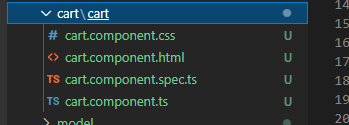
1. Cree un componente de carro y configure el enrutamiento al nuevo componente.
2. Muestre los artículos del carrito.

### **Configurar el componente del carrito**

Para crear la vista de carrito, siga los mismos pasos que hizo para crear ProductDetailsComponent y configurar el enrutamiento para el nuevo componente.

1. Genere un nuevo componente llamado carrito en la terminal ejecutando el siguiente comando:

ng generate component cart



Este comando generará el archivo *cart.component.ts* y los archivos de estilos y plantillas asociados.

***cart.component.ts***

import { [Component](https://angular.io/api/core/Component) } from '@angular/core';

@[Component](https://angular.io/api/core/Component)({

selector: 'app-cart',

templateUrl: './cart.component.html',

styleUrls: ['./cart.component.css']

})

export class CartComponent {

constructor() { }

}

Puede ignorar CartComponent ngOnInit () para esta clase.

1. Tenga en cuenta que el *CartComponent* recién creado se agrega a las declaraciones del módulo en *app.module.ts*.

***app.module.ts***

import { CartComponent } from './cart/cart.component';

@[NgModule](https://angular.io/api/core/NgModule)({

declarations: [

AppComponent,

TopBarComponent,

ProductListComponent,

ProductAlertsComponent,

ProductDetailsComponent,

CartComponent,

],

1. Aun en *app.module.ts*, agregar un route para el componente *CartComponent*, con una ruta de cart.

***app.module.ts***

@[NgModule](https://angular.io/api/core/NgModule)({

imports: [

[BrowserModule](https://angular.io/api/platform-browser/BrowserModule),

[ReactiveFormsModule](https://angular.io/api/forms/ReactiveFormsModule),

RouterModule.forRoot([

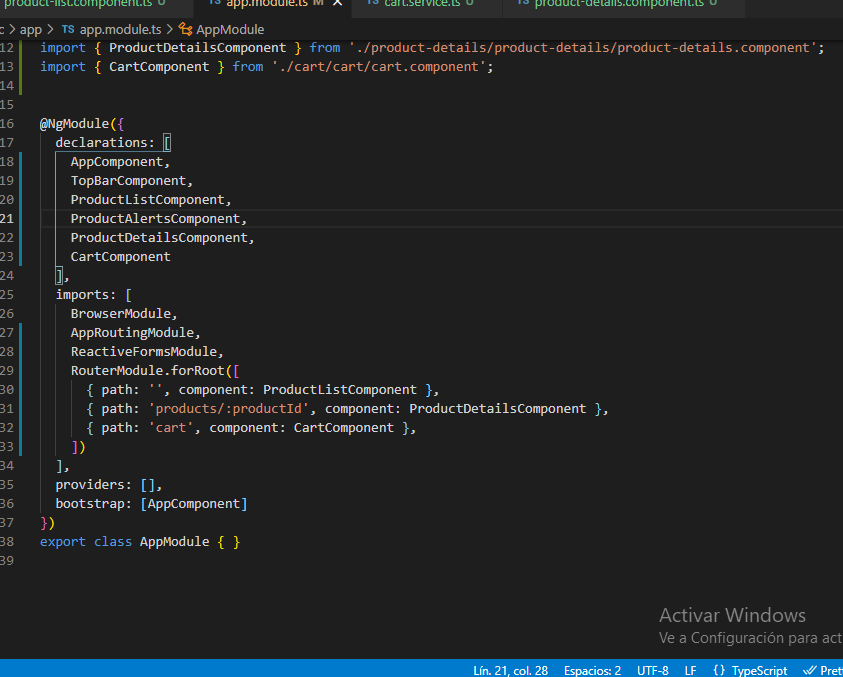
{ path: '', component: ProductListComponent },

{ path: 'products/:productId', component: ProductDetailsComponent },

{ path: 'cart', component: CartComponent },

])

],



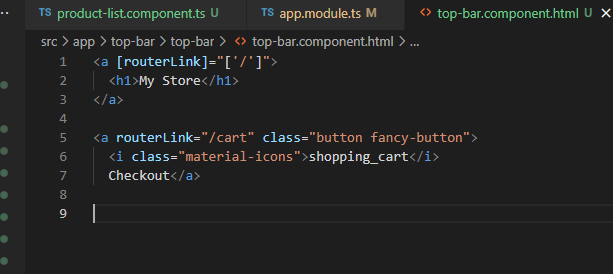
1. Actualice el botón buy para que se dirija a la URL / cart. *En top-bar.component.html*, agregue una directiva *routerLink* que apunte a */ cart.*

***top-bar.component.html***

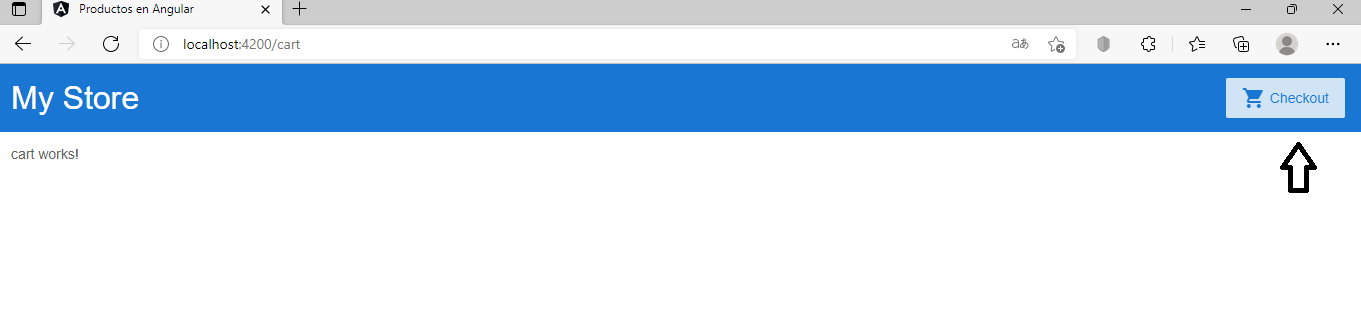
<a [routerLink](https://angular.io/api/router/RouterLink)="/cart" class="button fancy-button">

<i class="material-icons">shopping\_cart</i>Checkout

</a>



1. Verifique que el nuevo CartComponent funcione como se esperaba haciendo clic en el botón Pagar. ¡Puedes ver el "carrito funciona!" texto predeterminado y la URL tiene el patrón /cart.



**Mostrar los artículos del carrito**

Esta sección le muestra cómo utilizar el servicio de carrito para mostrar los productos en el carrito.

1. En *cart.component.ts*, hacer import CartService desde el archivo *cart.service.ts*.

***cart.component.ts***

import { [Component](https://angular.io/api/core/Component) } from '@angular/core';

import { CartService } from '../cart.service';

1. Inyectar *CartService* en *CartComponent* puede usarlo agregándolo al constructor ().

***cart.component.ts***

export class CartComponent {

constructor(

private cartService: CartService

) { }

}

1. Definir la propiedad items para almacenar productos en el cart.

***cart.component.ts***

export class CartComponent {

items = this.cartService.getItems();

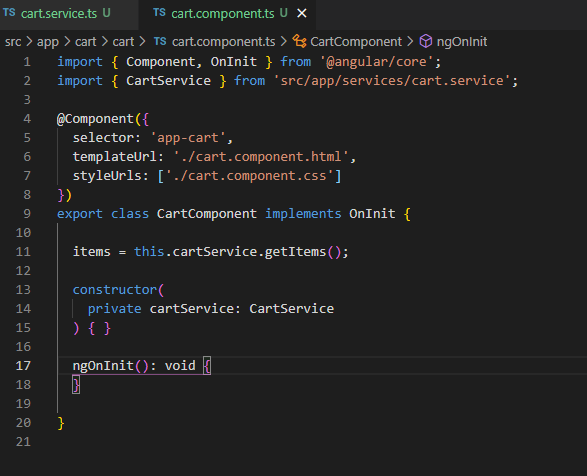
constructor(

private cartService: CartService

) { }

}

Este código establece los elementos mediante el método CartService getItems (). Se definió este método cuando creó cart.service.ts.



1. Actualice la plantilla del carrito con un encabezado y use un <div> con un \* ngFor para mostrar cada uno de los artículos del carrito con su nombre y precio. La plantilla CartComponent resultante es la siguiente.

***cart.component.html***

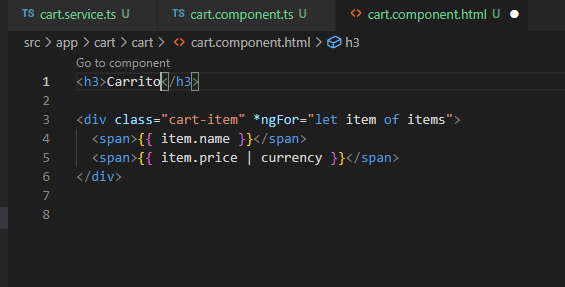
<h3>Cart</h3>

<div class="cart-item" \*[ngFor](https://angular.io/api/common/NgForOf)="let item of items">

<span>{{ item.name }}</span>

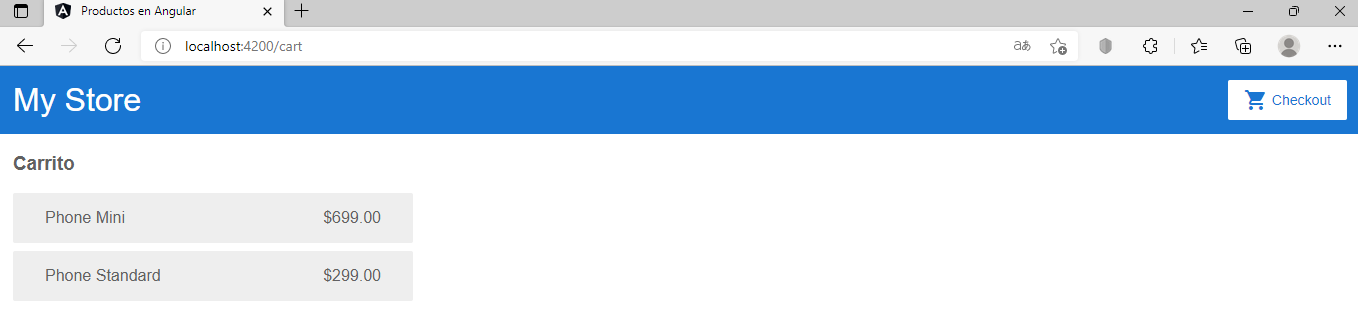
<span>{{ item.price | [currency](https://angular.io/api/common/CurrencyPipe) }}</span>

</div>



4. Verifique que su carrito funcione como se esperaba:

* + Haga clic en Mi tienda
  + Haga clic en el nombre de un producto para mostrar sus detalles.
  + Haga clic en Comprar para agregar el producto al carrito.
  + Haga clic en Pagar para ver el carrito.



## Recuperar precios de envíos

Los servidores a menudo devuelven datos en forma de flujo. Las transmisiones son útiles porque facilitan la transformación de los datos devueltos y la modificación de la forma en que solicita esos datos. Angular HttpClient es una forma integrada de obtener datos de API externas y proporcionarlos a su aplicación como un flujo.

Esta sección le muestra cómo usar HttpClient para recuperar los precios de envío de un archivo externo.

Recuerdese que la aplicación de esta lab viene con datos de envío predefinidos en *assets /shipping.json*. Utilice estos datos para agregar los precios de envío de los artículos en el carrito.

***src/assets/shipping.json***

[

{

"type": "Overnight",

"price": 25.99

},

{

"type": "2-Day",

"price": 9.99

},

{

"type": "Postal",

"price": 2.99

}

]

### **Configurar AppModule para usar HttpClient**

Para usar HttpClient de Angular, debe configurar su aplicación para usar HttpClientModule.

HttpClientModule de Angular registra los proveedores que su aplicación necesita para usar el servicio HttpClient en toda su aplicación.

1. En *app.module.ts*, importe HttpClientModule desde el paquete @ angular/ common/http en la parte superior del archivo con las otras importaciones. Como hay varias otras importaciones, este fragmento de código las omite por brevedad. Asegúrese de dejar las importaciones existentes en su lugar.

***app.module.ts***

import { [HttpClientModule](https://angular.io/api/common/http/HttpClientModule) } from '@angular/common/[http](https://angular.io/api/common/http)';

1. Para registrar los proveedores HttpClient de Angular globalmente, agregue *HttpClientModule* a la matriz de importaciones AppModule @*NgModule().*

***app.module.ts***

@[NgModule](https://angular.io/api/core/NgModule)({

imports: [

[BrowserModule](https://angular.io/api/platform-browser/BrowserModule),

[HttpClientModule](https://angular.io/api/common/http/HttpClientModule),

[ReactiveFormsModule](https://angular.io/api/forms/ReactiveFormsModule),

RouterModule.forRoot([

{ path: '', component: ProductListComponent },

{ path: 'products/:productId', component: ProductDetailsComponent },

{ path: 'cart', component: CartComponent },

])

],

declarations: [

AppComponent,

TopBarComponent,

ProductListComponent,

ProductAlertsComponent,

ProductDetailsComponent,

CartComponent,

],

bootstrap: [

AppComponent

]

})

export class AppModule { }

### 

### **Configurar CartService para usar HttpClient**

El siguiente paso es inyectar el servicio HttpClient en su servicio para que su aplicación pueda obtener datos e interactuar con API y recursos externos.

1. En *cart.service.ts*, import [HttpClient](https://angular.io/api/common/http/HttpClient) del paquete @angular/common/[http](https://angular.io/api/common/http).

***cart.service.ts***

import { [Injectable](https://angular.io/api/core/Injectable) } from '@angular/core';

import { [HttpClient](https://angular.io/api/common/http/HttpClient) } from '@angular/common/[http](https://angular.io/api/common/http)';

import { Product } from './products';

1. Inyectar [HttpClient](https://angular.io/api/common/http/HttpClient) en el constructor de CartService.

***cart.service.ts***

export class CartService {

items: Product[] = [];

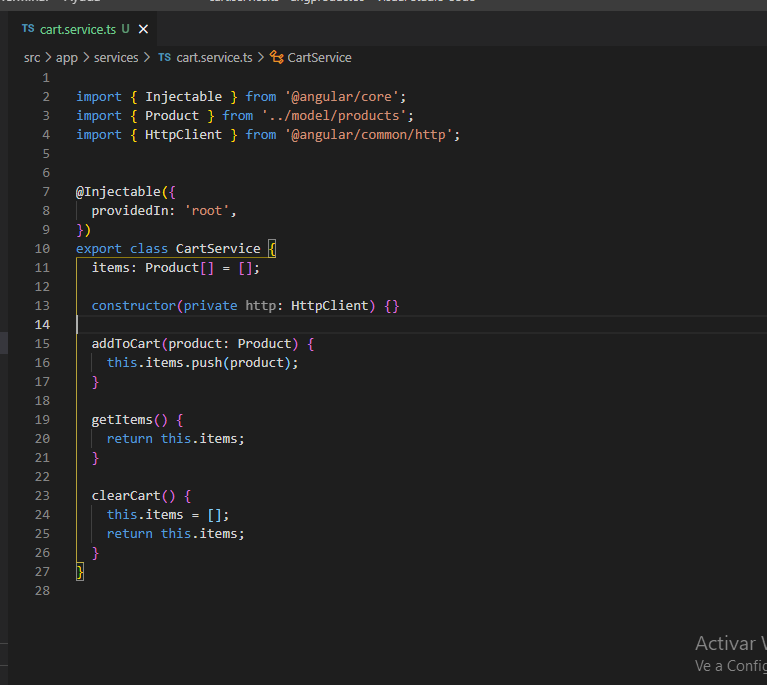
constructor(

private [http](https://angular.io/api/common/http): [HttpClient](https://angular.io/api/common/http/HttpClient)

) {}

/\* . . . \*/

}



### **Configurar CartService para obtener precios de envío**

Para obtener datos de envío, desde shipping.json, se usa el método HttpClient get ().

1. En cart.service.ts, debajo del método clearCart (), defina un nuevo método getShippingPrices () que use el método HttpClient get ().

***cart.service.ts***

export class CartService {

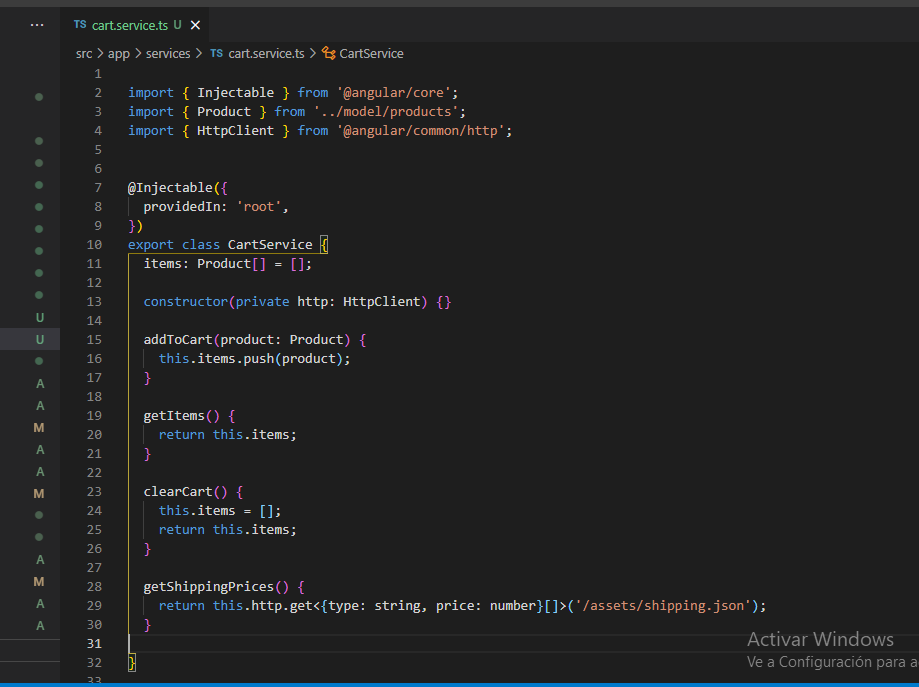
/\* . . . \*/

getShippingPrices() {

return this.http.get<{type: string, price: number}[]>('/assets/shipping.json');

}

}

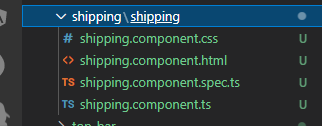


## Crea un componente de envío

Ahora que ha configurado su aplicación para recuperar datos de envío, puede crear un lugar para representar esos datos.

1. Genere un componente de carrito llamado envío en la terminal ejecutando el siguiente comando:

ng generate component shipping



Este comando generará el archivo *shipping.component.ts* y los archivos de estilos y plantillas asociados.

***shipping.component.ts***

import { [Component](https://angular.io/api/core/Component) } from '@angular/core';

@[Component](https://angular.io/api/core/Component)({

selector: 'app-shipping',

templateUrl: './shipping.component.html',

styleUrls: ['./shipping.component.css']

})

export class ShippingComponent {

constructor() { }

}

1. En app.module.ts, agregue una ruta para el envío. Especifique una ruta de envío y un componente de ShippingComponent.

***app.module.ts***

@[NgModule](https://angular.io/api/core/NgModule)({

imports: [

[BrowserModule](https://angular.io/api/platform-browser/BrowserModule),

[HttpClientModule](https://angular.io/api/common/http/HttpClientModule),

[ReactiveFormsModule](https://angular.io/api/forms/ReactiveFormsModule),

RouterModule.forRoot([

{ path: '', component: ProductListComponent },

{ path: 'products/:productId', component: ProductDetailsComponent },

{ path: 'cart', component: CartComponent },

{ path: 'shipping', component: ShippingComponent },

])

],

declarations: [

AppComponent,

TopBarComponent,

ProductListComponent,

ProductAlertsComponent,

ProductDetailsComponent,

CartComponent,

ShippingComponent

],

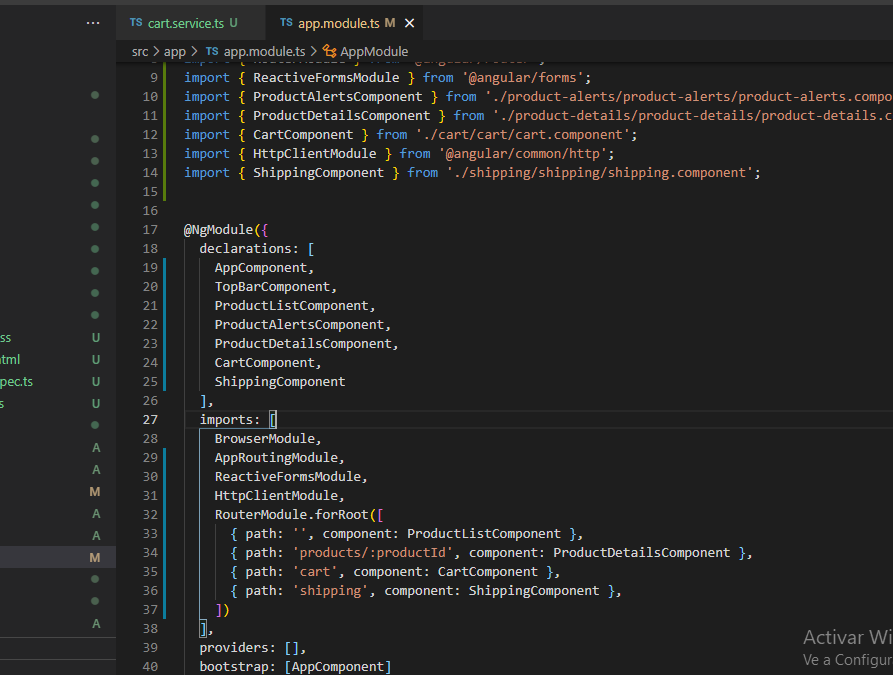
bootstrap: [

AppComponent

]

})

export class AppModule { }



Todavía no hay un enlace al nuevo componente de envío, pero puede ver su plantilla en el panel de vista previa ingresando la URL que especifica su ruta. La URL tiene el patrón: https://angular-ynqttp--4200.local.webcontainer.io/shipping donde la parte angular-ynqttp--4200.local.webcontainer.io.

### **Configuración de ShippingComponent para utilizar CartService**

Esta sección muestra la modificación de ShippingComponent para recuperar datos de envío a través de HTTP del archivo shipping.json.

1. En shipping.component.ts, import CartService.

***shipping.component.ts***

import { [Component](https://angular.io/api/core/Component) } from '@angular/core';

import { CartService } from '../cart.service';

1. Inyectar *cart service* en el ShippingComponent constructor().

***shipping.component.ts***

constructor(private cartService: CartService) { }

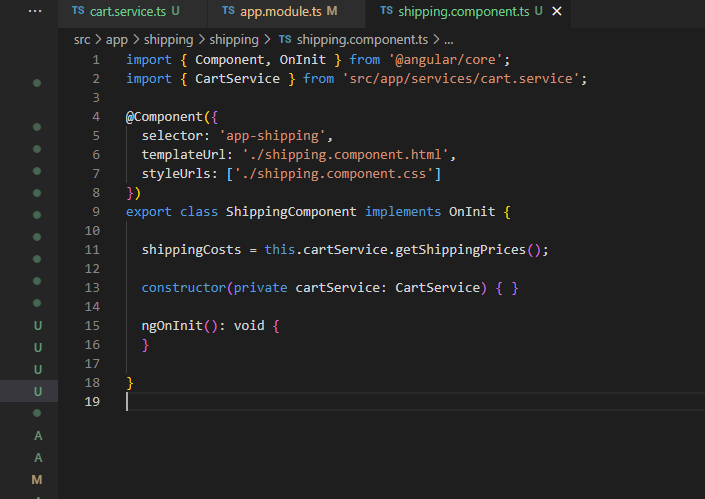
1. Defina una propiedad shippingCosts que establezca la propiedad shippingCosts mediante el método getShippingPrices () de CartService..

***shipping.component.ts***

export class ShippingComponent {

shippingCosts = this.cartService.getShippingPrices();

}



1. Actualice la plantilla ShippingComponent para mostrar los tipos de envío y los precios mediante la canalización asíncrona.

***shipping.component.html***

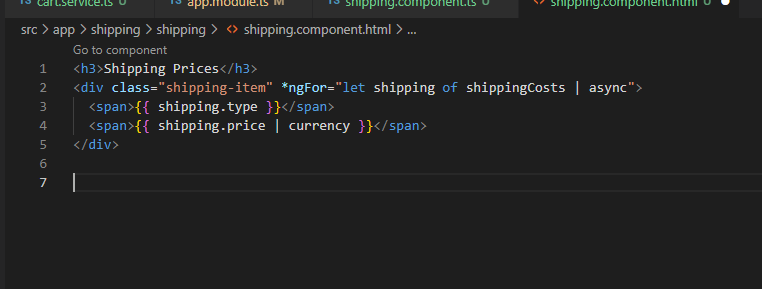
<h3>Shipping Prices</h3>

<div class="shipping-item" \*[ngFor](https://angular.io/api/common/NgForOf)="let shipping of shippingCosts | [async](https://angular.io/api/common/AsyncPipe)">

<span>{{ shipping.type }}</span>

<span>{{ shipping.price | [currency](https://angular.io/api/common/CurrencyPipe) }}</span>

</div>



La tubería asíncrona devuelve el último valor de un flujo de datos y continúa haciéndolo durante la vida útil de un componente determinado. Cuando Angular destruye ese componente, la tubería asíncrona se detiene automáticamente.

1. Agregue un enlace desde la vista Componente del carrito a la vista Componente de envío.

***cart.component.html***

<h3>Cart</h3>

<p>

<a [routerLink](https://angular.io/api/router/RouterLink)="/shipping">Shipping Prices</a>

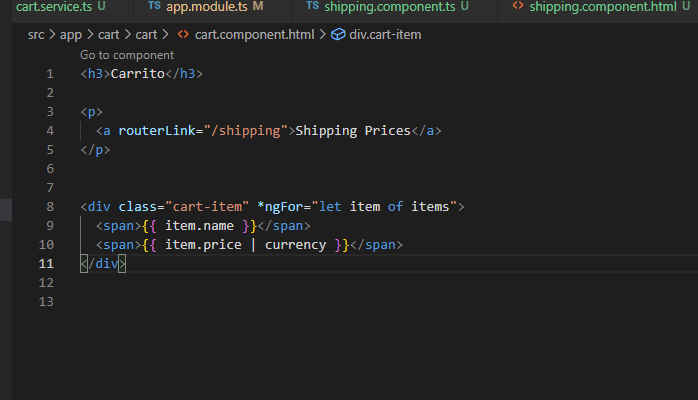
</p>

<div class="cart-item" \*[ngFor](https://angular.io/api/common/NgForOf)="let item of items">

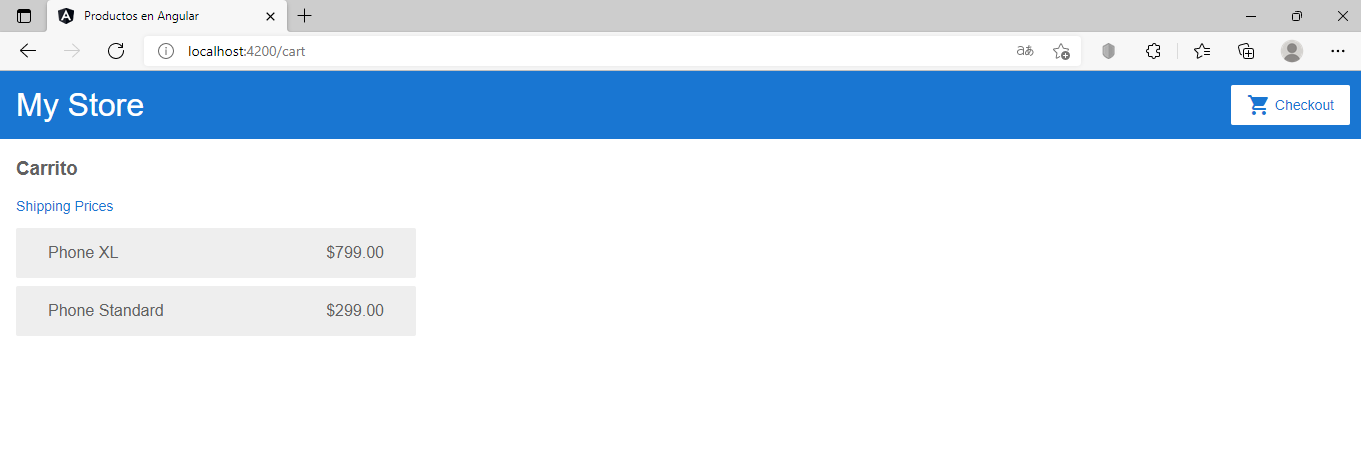
<span>{{ item.name }}</span>

<span>{{ item.price | [currency](https://angular.io/api/common/CurrencyPipe) }}</span>

</div>



1. Haga clic en el botón Checkout para ver el carrito actualizado. Recuerde que cambiar la aplicación hace que se actualice la vista previa, lo que vacía el carrito.



Haga clic en el enlace para navegar a los precios de envío.

