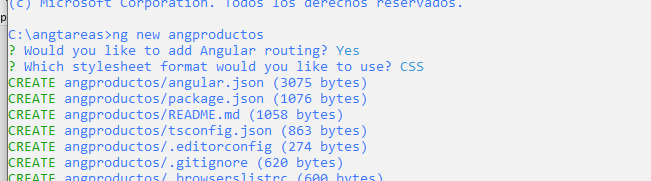
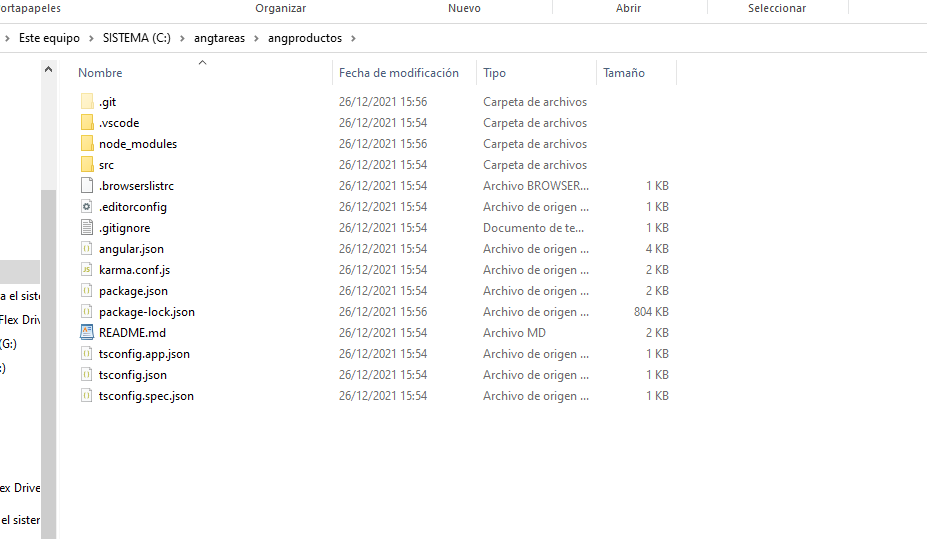
**Laboratorio básico - Angular Framework**

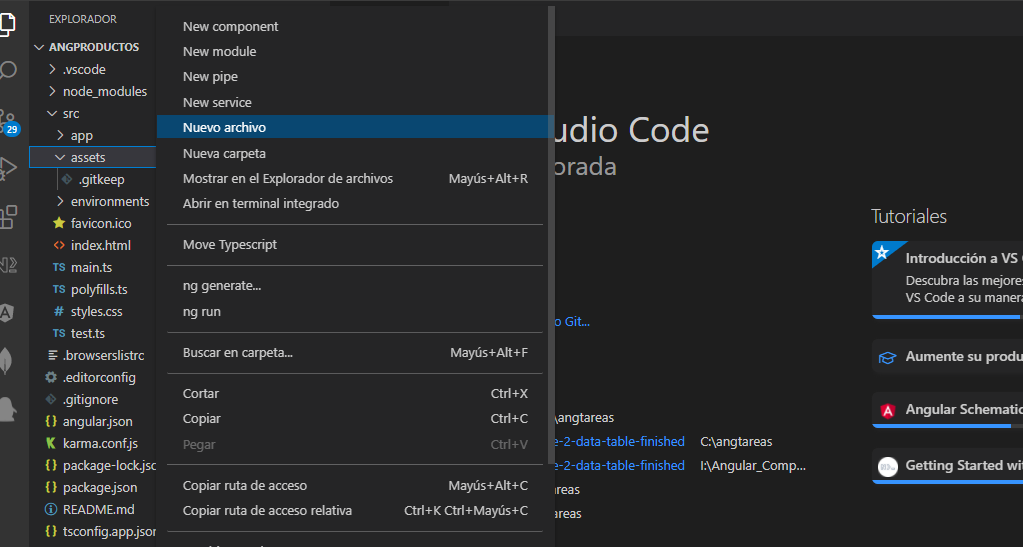
Crear un proyecto Angular angproductos





Una vez generado el proyecto, se generaran los siguientes archivos:

1. **Archivo shipping.json en assets**



En dicho archivo se cargarán los siguientes datos:

[

{

"type": "Overnight",

"price": 25.99

},

{

"type": "2-Day",

"price": 9.99

},

{

"type": "Postal",

"price": 2.99

}

]

1. **styles.css**

/\* Global Styles \*/

\* {

font-family: 'Roboto', Arial, sans-serif;

color: #616161;

box-sizing: border-box;

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

}

body {

margin: 0;

}

.container {

display: flex;

flex-direction: row;

}

router-outlet + \* {

padding: 0 16px;

}

/\* Text \*/

h1 {

font-size: 32px;

}

h2 {

font-size: 20px;

}

h1, h2 {

font-weight: lighter;

}

p {

font-size: 14px;

}

/\* Hyperlink \*/

a {

cursor: pointer;

color: #1976d2;

text-decoration: none;

}

a:hover {

opacity: 0.8;

}

/\* Input \*/

input {

font-size: 14px;

border-radius: 2px;

padding: 8px;

margin-bottom: 16px;

border: 1px solid #BDBDBD;

}

label {

font-size: 12px;

font-weight: bold;

margin-bottom: 4px;

display: block;

text-transform: uppercase;

}

/\* Button \*/

.button, button {

display: inline-flex;

align-items: center;

padding: 8px 16px;

border-radius: 2px;

font-size: 14px;

cursor: pointer;

background-color: #1976d2;

color: white;

border: none;

}

.button:hover, button:hover {

opacity: 0.8;

font-weight: normal;

}

.button:disabled, button:disabled {

opacity: 0.5;

cursor: auto;

}

/\* Fancy Button \*/

.fancy-button {

background-color: white;

color: #1976d2;

}

.fancy-button i.material-icons {

color: #1976d2;

padding-right: 4px;

}

/\* Top Bar \*/

app-top-bar {

width: 100%;

height: 68px;

background-color: #1976d2;

padding: 16px;

display: flex;

flex-direction: row;

justify-content: space-between;

align-items: center;

}

app-top-bar h1 {

color: white;

margin: 0;

}

/\* Checkout Cart, Shipping Prices \*/

.cart-item, .shipping-item {

width: 100%;

min-width: 400px;

max-width: 450px;

display: flex;

flex-direction: row;

justify-content: space-between;

padding: 16px 32px;

margin-bottom: 8px;

border-radius: 2px;

background-color: #EEEEEE;

}

Este estilo se aplicará a todos los componentes del proyecto.

1. Index.html

La pagina html del proyecto, se inicializara con el siguiente código:

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Productos en Angular</title>

<base href="/">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="icon" type="image/x-icon" href="favicon.ico">

<link

href="https://fonts.googleapis.com/icon?family=Material+Icons"

rel="stylesheet"/>

</head>

<body>

<app-root></app-root>

</body>

</html>

A continuación se revisa el código del componente principal..

**app.component.ts (sin cambios)**

**app.component.html**

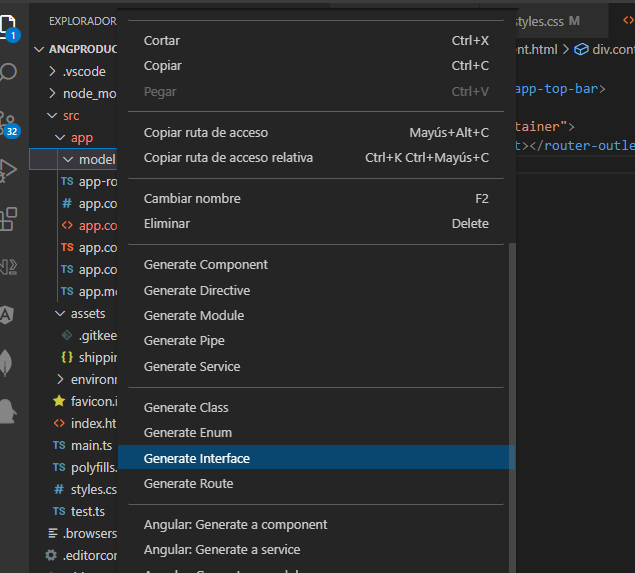
<app-top-bar></app-top-bar>

<div class="container">

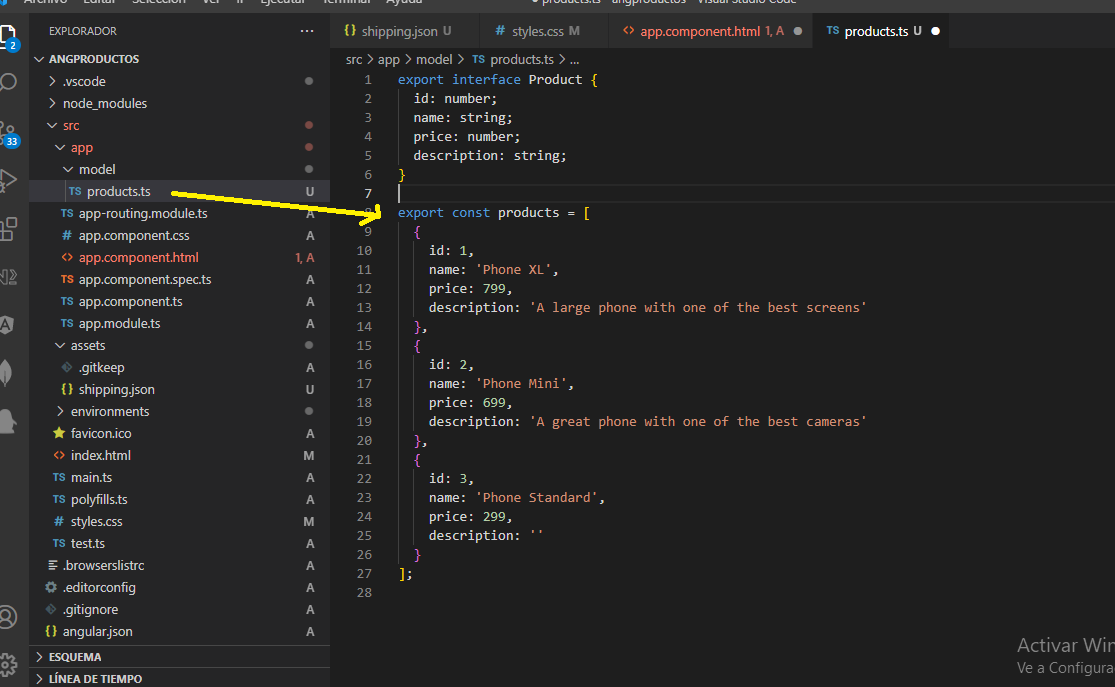
<router-outlet></router-outlet>

</div>

Luego se creará la carpeta modulo y allí crearemos la interface Product y valores de esta clase serán generados.



Una vez creada, se renombra la interface a Product y se creara una constante products para declarar datos usando esta estructura.



El código queda asi:

export interface Product {

id: number;

name: string;

price: number;

description: string;

}

export const products = [

{

id: 1,

name: 'Phone XL',

price: 799,

description: 'A large phone with one of the best screens'

},

{

id: 2,

name: 'Phone Mini',

price: 699,

description: 'A great phone with one of the best cameras'

},

{

id: 3,

name: 'Phone Standard',

price: 299,

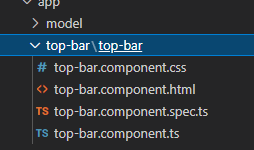
description: ''

}

];

**b. Componente top-bar**

Se crea la carpeta top-bar y luego el componente top-bar



Código inicial de top-bar.component.html:

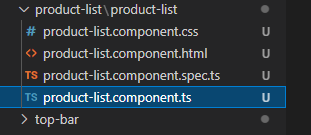
<a [routerLink]="['/']">

<h1>My Store</h1>

</a>

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

1. **Componente product-list**



El código inicial de producto-list.component.ts es:

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

import { products } from 'src/app/model/products';

@Component({

selector: 'app-product-list',

templateUrl: './product-list.component.html',

styleUrls: ['./product-list.component.css']

})

export class ProductListComponent implements OnInit {

products = products;

constructor() { }

ngOnInit(): void {

}

share() {

window.alert('The product has been shared!');

}

}

Con todo lo implementado, de manera inicial, se definira app.module.

**app.module.ts**

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';

import { TopBarComponent } from './top-bar/top-bar/top-bar.component';

import { ProductListComponent } from './product-list/product-list/product-list.component';

import { RouterModule } from '@angular/router';

import { ReactiveFormsModule } from '@angular/forms';

@NgModule({

declarations: [

AppComponent,

TopBarComponent,

ProductListComponent

],

imports: [

BrowserModule,

AppRoutingModule,

ReactiveFormsModule,

RouterModule.forRoot([

{ path: '', component: ProductListComponent },

])

],

providers: [],

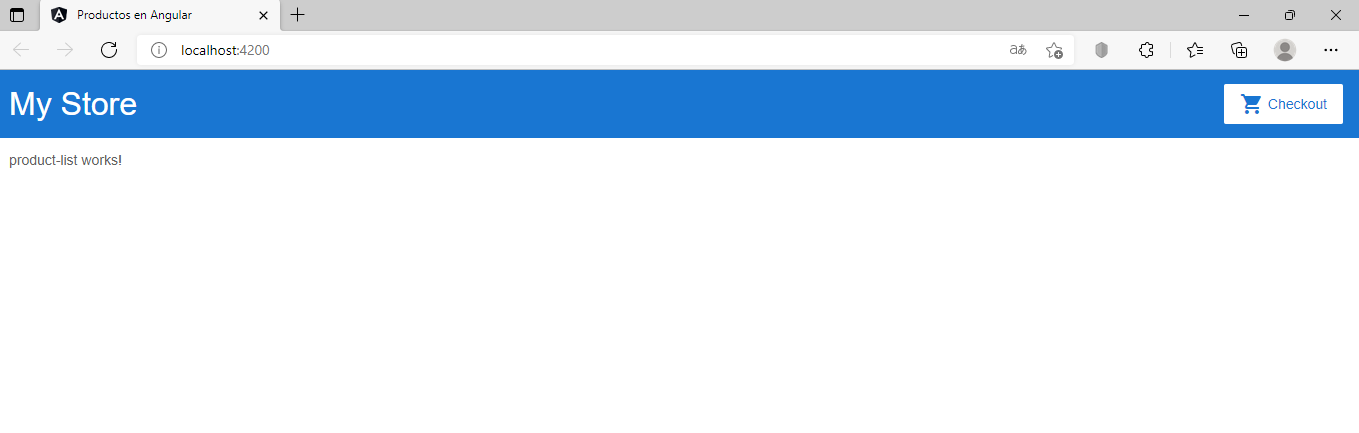
bootstrap: [AppComponent]

})

export class AppModule { }

Luego, se ejecuta la aplicación a través del comando:

*ng serve –o*

**

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

En esta sección, se actualizará la aplicación para mostrar una lista de productos. Se usarán datos predefinidos de productos del archivo products.ts y métodos del archivo product-list.component.ts. Esta sección le guía a través de la edición del HTML, también conocido como la plantilla.

1. En el folder product-list  , abrir product-list.component.html.
2. Añadir la directiva \*[ngFor](https://angular.io/api/common/NgForOf) en un  <div>, como se muestra.

<h2>Products</h2>

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

</div>

Con \*[ngFor](https://angular.io/api/common/NgForOf), la etiqueta <div> se repite por cada producto de la lista

Las directivas estructurales dan forma o remodelan la estructura del DOM, añadiendo, eliminando y manipulando elementos.

1. Dentro de <div>, agregar un <h3> y {{ product.name }}. La sentencia {{ product.name }}  es un ejemplo de la sintaxis de interpolación de Angular. Interpolacion {{ }} le permite representar el valor de la propiedad como texto.

El código completo es el siguiente:

<h2>Productos</h2>

<div \*ngFor="let product of products">

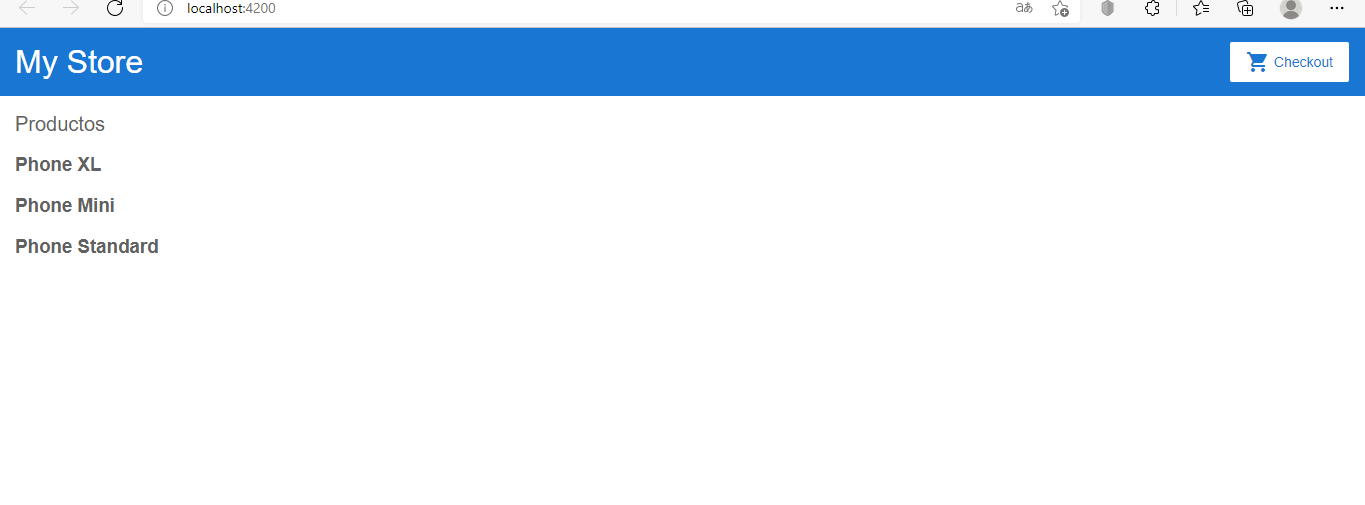
<h3>

{{ product.name }}

</h3>

</div>

Quedando la pagina de la siguiente manera:



1. Para hacer que cada nombre de producto sea un enlace a los detalles del producto, añada el elemento <a> alrededor de {{ product.name }}.
2. Establezca el título para que sea el nombre del producto utilizando la sintaxis de vinculación de propiedades [ ], como sigue:

<h2>Products</h2>

<div \*ngFor="let product of products">

<h3>

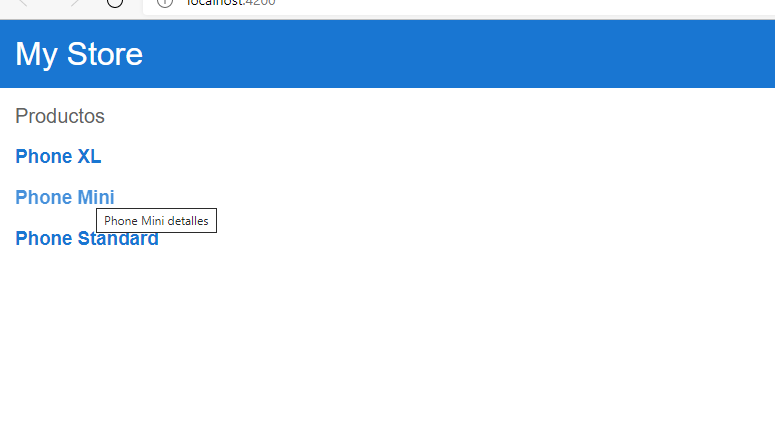
<a [title]="product.name + ' details'">

{{ product.name }}

</a>

</h3>

</div>



Añade las descripciones de los productos. En un elemento <p>, utiliza una directiva \*ngIf para que Angular solo cree el elemento <p> si el producto actual tiene una descripción.

<h2>Products</h2>

<div \*ngFor="let product of products">

<h3>

<a [title]="product.name + ' details'">

{{ product.name }}

</a>

</h3>

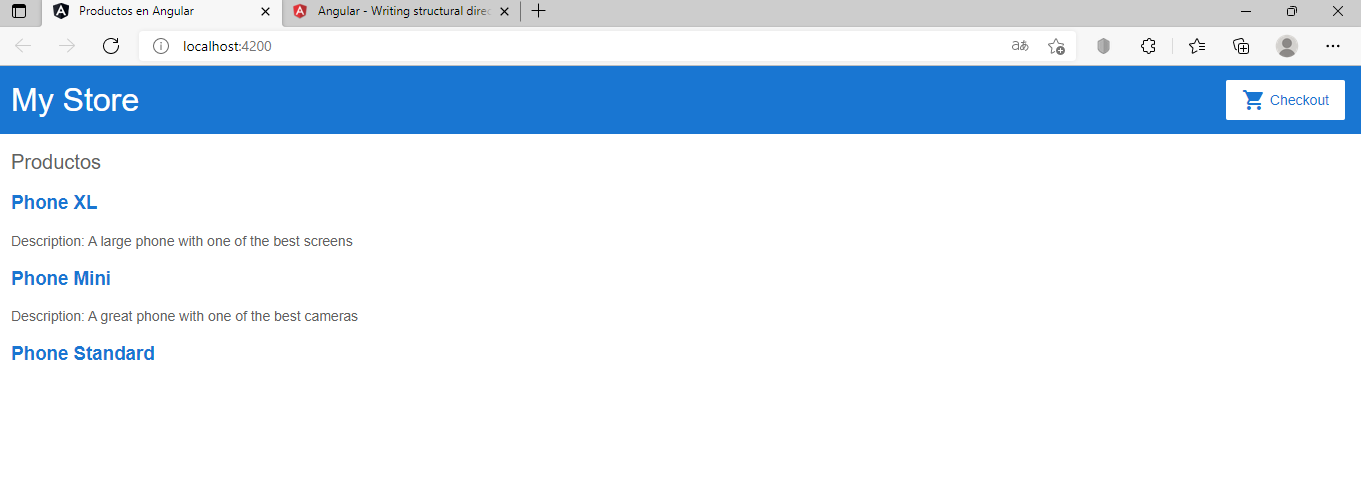
<p \*ngIf="product.description">

Description: {{ product.description }}

</p>

</div>

Se muestra el resultado:



La aplicación muestra ahora el nombre y la descripción de cada producto de la lista. Observa que el producto final no tiene un párrafo de descripción. Angular no crea el elemento <p> porque la propiedad de descripción del producto está vacía.

1. Agregue un botón para que los usuarios puedan compartir un producto. Vincula el evento de clic del botón al método share () en product-list.component.ts. El enlace de eventos usa un conjunto de paréntesis, (), alrededor del evento, como en el evento (clic) en el elemento <button>.

<h2>Products</h2>

<div \*ngFor="let product of products">

<h3>

<a [title]="product.name + ' details'">

{{ product.name }}

</a>

</h3>

<p \*ngIf="product.description">

Description: {{ product.description }}

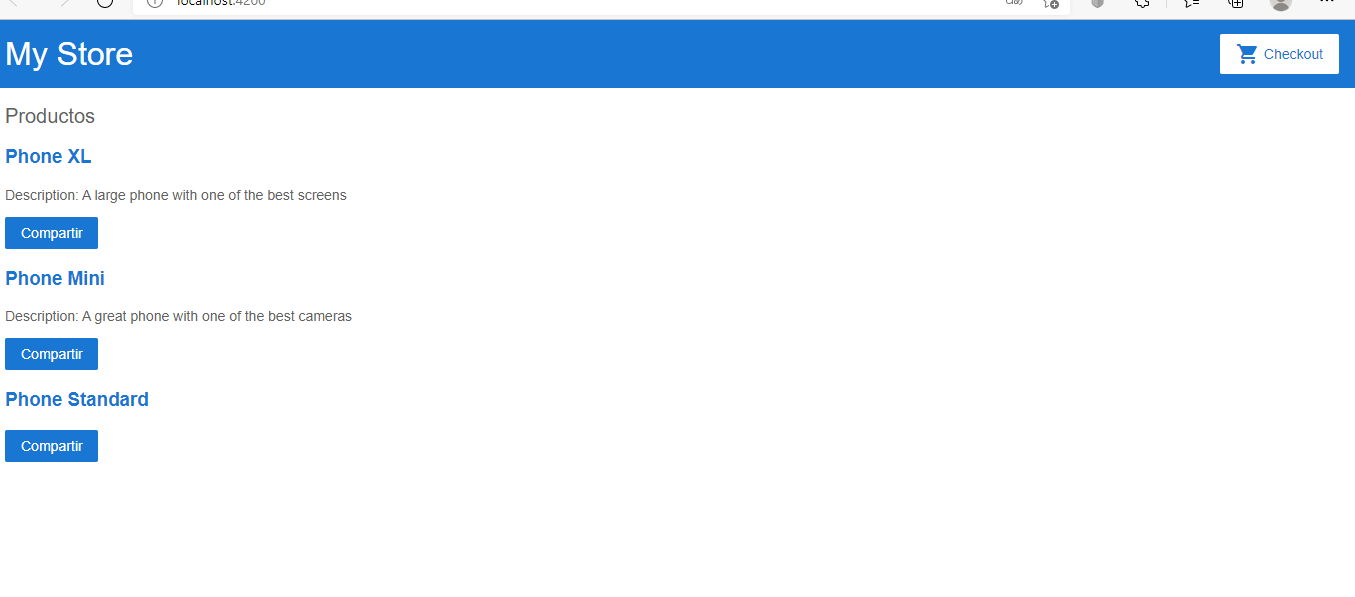
</p>

<button (click)="share()">

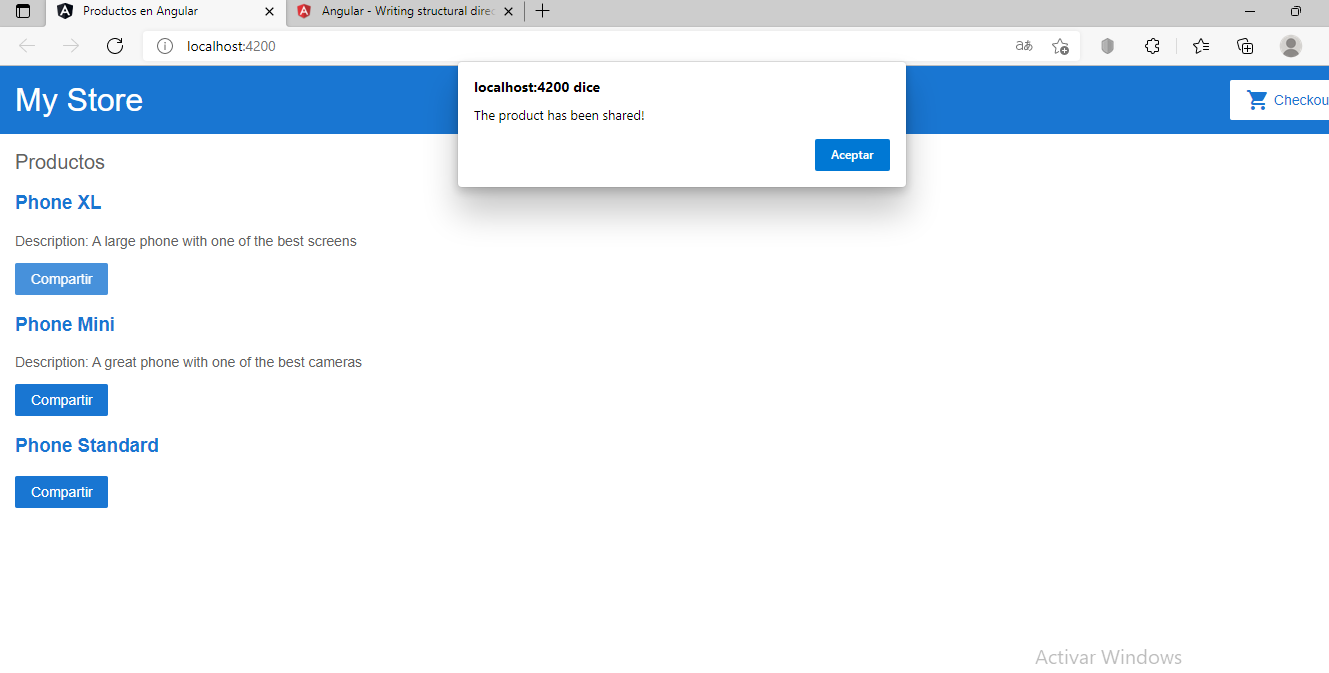
Share

</button>

</div>



al hacer clic en el botón Compartir, se activa una alerta que dice: "¡El producto ha sido compartido!".

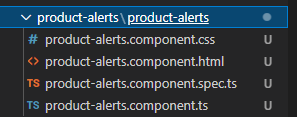


Pasar datos a un componente secundario

Actualmente, la lista de productos muestra el nombre y la descripción de cada producto. ProductListComponent también define una propiedad de productos que contiene datos importados para cada producto de la matriz de productos en products.ts.

El siguiente paso es crear una nueva función de alerta que utilice datos de productos de *ProductListComponent*. La alerta verifica el precio del producto y, si el precio es superior a $ 700, muestra un botón Notificarme que permite a los usuarios registrarse para recibir notificaciones cuando el producto sale a la venta.

Esta sección lo guía a través de la creación de un componente secundario, *ProductAlertsComponent*, que puede recibir datos de su componente principal, *ProductListComponent*. Haga clic derecho en la carpeta de la aplicación y use el generador angular para generar un nuevo componente llamado *products-alerts*.



Para configurar ProductAlertsComponent para recibir datos del producto, primero importe Input desde @ angular / core.

En la definición de la clase ProductAlertsComponent, defina una propiedad denominada product con un decorador @Input (). El decorador @Input () indica que el valor de la propiedad pasa desde el componente principal del componente, ProductListComponent

A partir de allí, se colocará el código de este componente:

***product-alerts.component.ts***

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

import { Input } from '@angular/core';

import { Product } from 'src/app/model/products';

@Component({

selector: 'app-product-alerts',

templateUrl: './product-alerts.component.html',

styleUrls: ['./product-alerts.component.css']

})

export class ProductAlertsComponent implements OnInit {

@Input() product!: Product;

constructor() { }

ngOnInit(): void {

}

}

***product-alerts.component.html***

<p \*ngIf="product && product.price > 700">

<button>Notify Me</button>

</p>

Para que ProductAlertsComponent esté disponible para otros componentes de la aplicación, agréguelo a las declaraciones de AppModule en app.module.ts (en Visual Studio Code esa declaración es automática)

import { ProductAlertsComponent } from './product-alerts/product-alerts.component';

@NgModule({

declarations: [

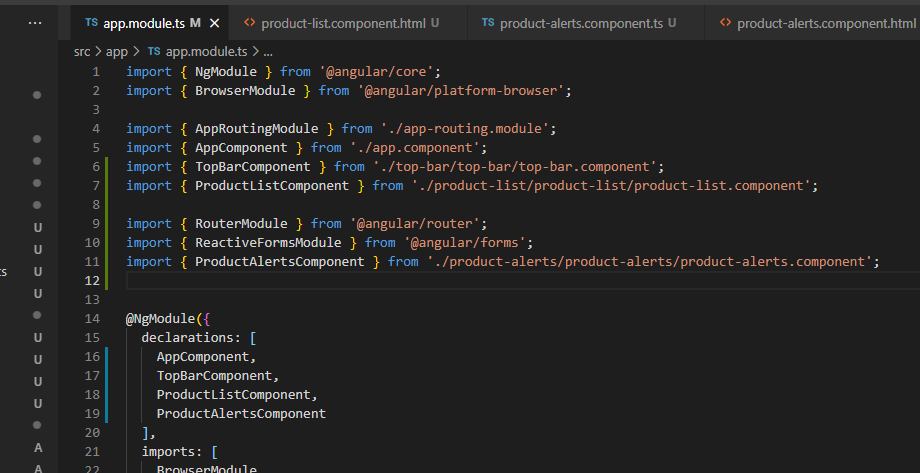
AppComponent,

TopBarComponent,

ProductListComponent,

ProductAlertsComponent,

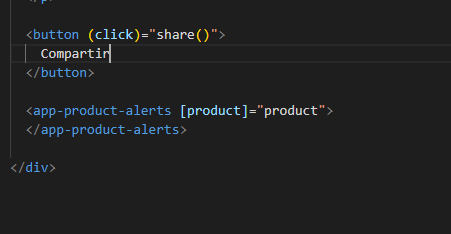
],



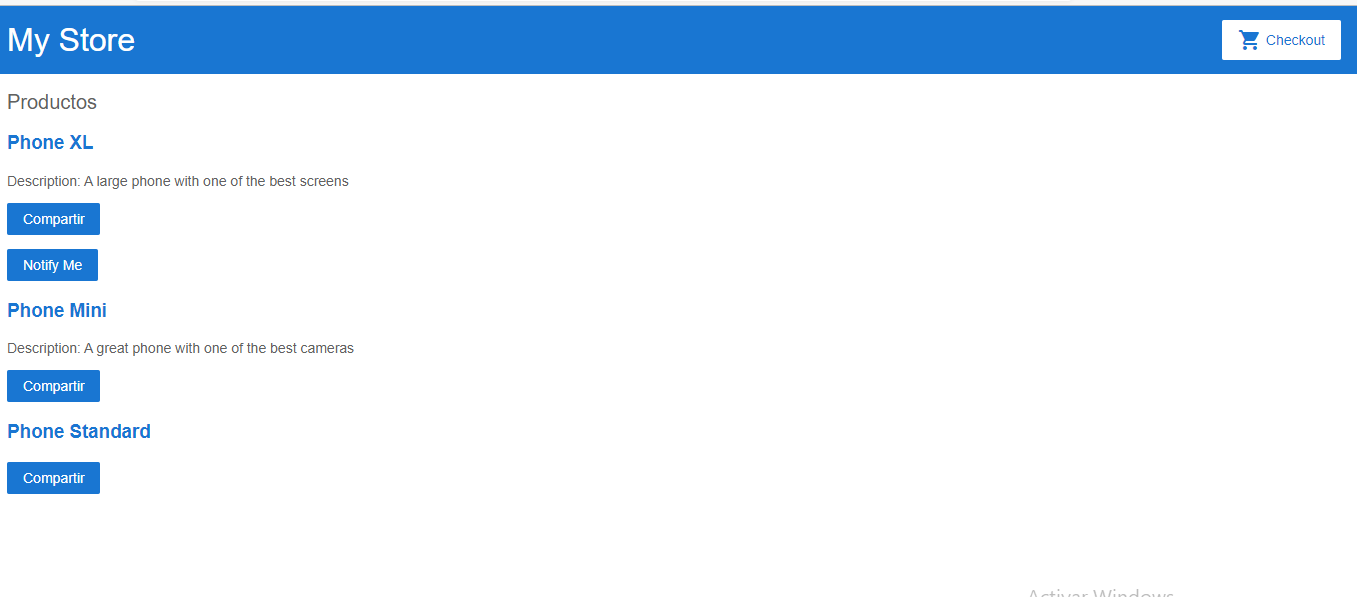
1. Finalmente, para mostrar ProductAlertsComponent como un elemento secundario de ProductListComponent, agregue el selector, <app-product-alerts> a *product-list.component.html*. Pase el producto actual como entrada al componente mediante el enlace de propiedad.

<app-product-alerts [product]="product">

</app-product-alerts>



Lo que producirá la siguiente respuesta:



El componente de alerta de nuevo producto toma un producto como entrada de la lista de productos. Con esa entrada, muestra u oculta el botón Notificarme, según el precio del producto. El precio del Phone XL es de más de $ 700, por lo que el botón Notificarme aparece en ese producto.

**Pasar datos a un componente principal**

Para que el botón Notificarme funcione, el componente secundario debe notificar y pasar los datos al componente principal. ProductAlertsComponent necesita emitir un evento cuando el usuario hace clic en Notificarme y ProductListComponent necesita responder al evento.

*En los componentes nuevos, el generador angular incluye un constructor vacío (), la interfaz OnInit y el método ngOnInit (). Dado que estos pasos no los usan, los siguientes ejemplos de código los omiten por brevedad.*

Se abre el archivo product-alerts.component.ts.

* 1. En product-alerts.component.ts, importar Output y EventEmitter desde @ angular / core.
  2. En la clase de componente, defina una propiedad llamada notify con un decorador @Output () y una instancia de EventEmitter (). Configurar ProductAlertsComponent con un @Output () permite que ProductAlertsComponent emita un evento cuando cambia el valor de la propiedad de notificación.
  3. En *product-alerts.component.html*, actualice el botón **Notificarme** con un enlace de evento para llamar al método notify.emit ().
  4. Defina el comportamiento que ocurre cuando el usuario hace clic en el botón. El elemento principal, ProductListComponent, no ProductAlertsComponent, actúa cuando el elemento secundario genera el evento. En product-list.component.ts, defina un método onNotify (), similar al método share ().

***product-alerts.component.ts***

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

import { Input } from '@angular/core';

import { Product } from 'src/app/model/products';

import { Output, EventEmitter } from '@angular/core';

@Component({

  selector: 'app-product-alerts',

  templateUrl: './product-alerts.component.html',

  styleUrls: ['./product-alerts.component.css']

})

export class ProductAlertsComponent implements OnInit {

  @Input() product!: Product;

  @Output() notify = new EventEmitter();

  constructor() { }

  ngOnInit(): void {

  }

}

***product-alerts.component.html***

<p \*ngIf="product && product.price > 700">

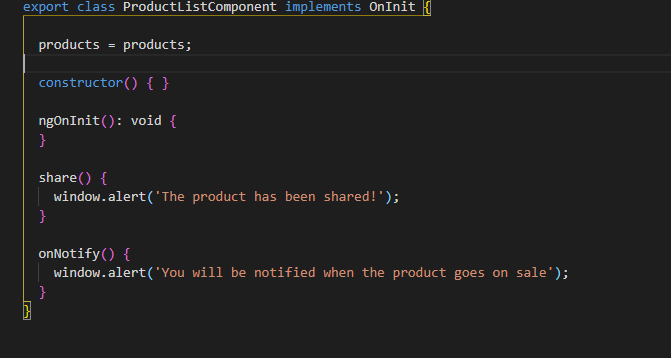
<button (click)="notify.emit()">Notify Me</button>

</p>

1. Defina el comportamiento que ocurre cuando el usuario hace clic en el botón. El elemento principal, ProductListComponent, (no ProductAlertsComponent), actúa cuando el elemento secundario genera el evento. En *product-list.component.ts,* defina un método onNotify (), similar al método share ().

onNotify() {

window.alert('You will be notified when the product goes on sale'); }



1. Actualice ProductListComponent para recibir datos de ProductAlertsComponent.

En *product-list.component.html*, vincule <app-product-alerts> al método onNotify () del componente de lista de productos. <app-product-alerts> es lo que muestra el botón Notificarme.

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

<button (click)="share()">

Share

</button>

<app-product-alerts

[product]="product"

(notify)="onNotify()">

</app-product-alerts>



Ejecutándose el proyecto y presionándose al botón “notify me” se obtiene:

