

Vamos a desarrollar una WebApp para Gestión de Productos:

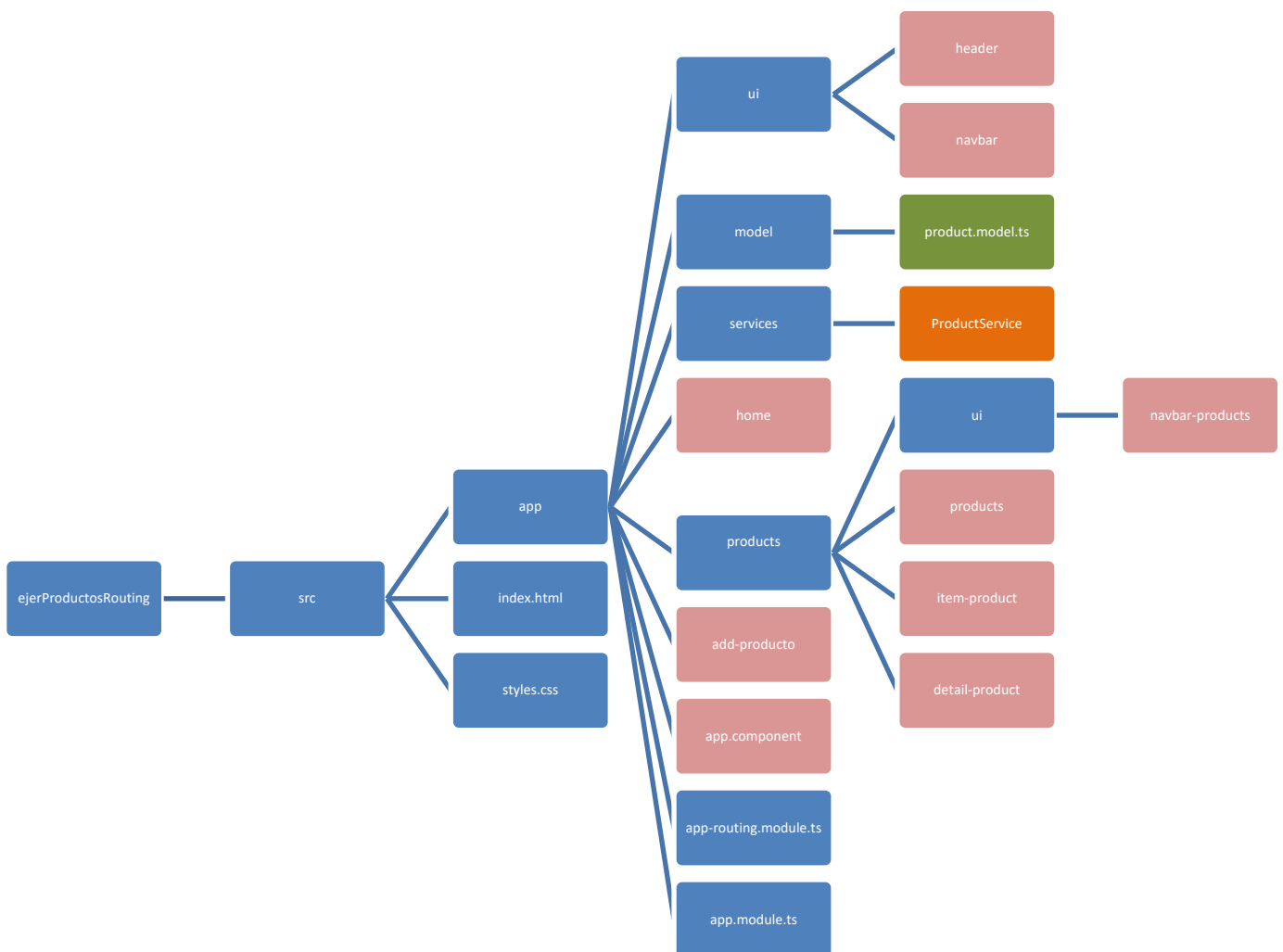
1. Estructura de proyecto

Los colores indican qué son cada elemento:

rojo claro = componentes

verde=modelo de datos

naranja=servicio de acceso a datos.

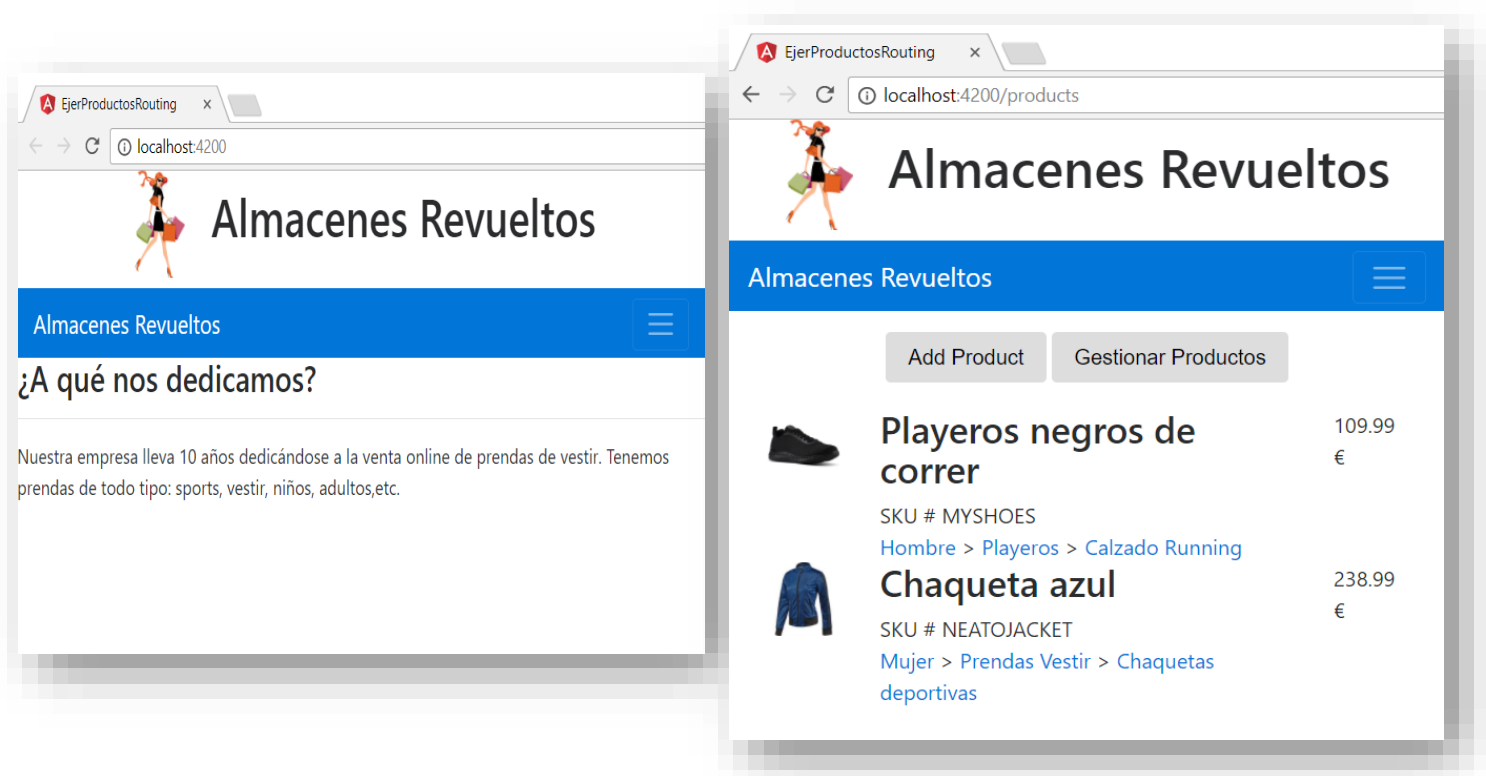


2. Crear la aplicación:

ng new ejerProductosRouting

Y responder **Yes** cuando pregunte por **Routing** y elegir **CSS** como hojas de estilo.

Su aspecto inicial será algo como:



3. Comprobar en index.html que está <base href>

4. Borrar todo el contenido de la plantilla HTML del componente AppComponent, excepto la última línea que dice:

<router-outlet></router-outlet>

Esa línea es una directiva que realmente es el ROUTER de Angular. **Ahí es donde va a ir colocando los diferentes componentes cuando vayamos navegando hacia ellos.** Si se quita, la navegación o Routing no funciona.

5. Copiamos las imágenes que vayamos a usar al proyecto

1º) Creamos dentro de src/assets una carpeta de nombre “images”

2º) En la carpeta src/assets/images copiamos todas las imágenes que vaya a necesitar nuestro proyecto.

6. Añadimos Bootstrap al proyecto

De todas las formas que hay de añadirlo, he elegido usar @import en el fichero styles.css.

Por tanto, dentro de ese fichero escribís:

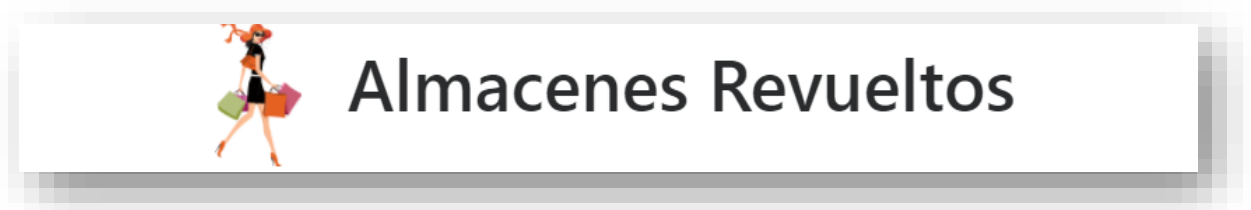
@import 'https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css';

Os debe quedar así:

```
src > styles.css
1  /* You can add global styles to this file, and also import other style files */
2  @import 'https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css';
```

7. Creamos nuestro componente header

Será la cabecera principal de la aplicación.



Contiene el logo y el nombre de la empresa

ng g c ui/header

header.component.html

```
<div class="container">
  <h1 class="col-12">
     {{title}}</h1>
  </div>
```

header.component.ts

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

@Component({
  selector: 'app-header',
  templateUrl: './header.component.html',
  styleUrls: ['./header.component.css']
})

export class HeaderComponent {
  title: string;
  constructor() {
    this.title = 'Almacenes Revueltos';
  }
}
```

Y para que nuestro componente se muestre, añadimos en **app.component.html**:

```
<app-header></app-header>
```

8. Creamos el componente barra de navegación (navBar)

Será la barra de navegación de nuestra app, nuestro menú.

ng g c ui/navbar

Cerrada se muestra así:



Abierta se muestra así:



Como es una barra de navegación, copiamos la de bootstrap y le añadimos el código para que funcione el botón hamburguesa y le adaptamos las opciones del menú al nuestro:

navbar.component.html

```
<nav class="navbar navbar-expand-lg navbar-dark bg-primary">
  <div class="container-fluid">
    <a class="navbar-brand" routerLink="/home">{{title}}</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup" aria-
expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"
(click)="toggleCollapse()"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNavAltMarkup"
[class.show]="show">
      <div class="navbar-nav">
        <a class="nav-link" aria-current="page"
routerLink="/home"
routerLinkActive="active">Home</a>
        <a class="nav-link"
routerLink="/products"
routerLinkActive="active">Productos</a>
        <a class="nav-link" routerLink="/contact"
routerLinkActive="active">Contacto</a>
      </div>
    </div>
  </div>
</nav>
```

navbar.component.ts

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

@Component({
  selector: 'app-navbar',
  templateUrl: './navbar.component.html',
  styleUrls: ['./navbar.component.css']
})
export class NavbarComponent {
```

```

show: boolean = false;
title: string = 'Almacenes Revueltos';

toggleCollapse() {
  this.show = !this.show;
}
}

```

Y para que nuestro componente se muestre, añadimos en **app.component.html**:

```
<app-navbar></app-navbar>
```

9. Creamos nuestro modelo de datos: producto y servicio ProductService

En la práctica real, el modelo de datos suele ser un SERVICIO que incluye todos los métodos para gestionar dicho modelo: extraer datos, añadir, modificar,... y que es quien accede al modelo de datos directamente.

Necesitamos:

- Una clase ProductModel: que represente un producto y lo gestione.
- Un servicio ProductService, que contendrá un array de productos (si fuese real, este array no existiría aquí, sino que habría una BD en la nube a la que se accedería con los métodos que se pondrán en este servicio), así como los métodos para gestionarlos: añadir, obtener uno o varios productos, borrar productos,...

9.1.-product.model

ng g class model/product.model

product.model.ts

```

export class ProductModel {
  static numProducts:number=0;
  idProduct:number;

  constructor(
    public sku:string,
    public name:string,
    public imageUrl:string,
    public department:string[],
    public price:number
  ){
    ProductModel.numProducts++;
  }
}

```

```

        this.idProduct=ProductModel.numProducts;
    }
}

```

9.2.-Servicio ProductService

ng g service services/product

products.service.ts

```

import { Injectable } from '@angular/core';
import { ProductModel } from '../model/product.model';

// Array de productos
const PRODUCTOS: ProductModel[] = [
    new ProductModel('NICEHAT',
        'Un bonito sombrero negro',
        '../assets/images/gorra-negra.jpg',
        ['Hombre', 'Accesorios', 'Gorras'],
        29.99),
    new ProductModel('MYSHOES',
        'Playeros negros de correr',
        '../assets/images/myshoes.jpg',
        ['Hombre', 'Playeros', 'Calzado Running'],
        109.99),
    new ProductModel('NEATOJACKET',
        'Chaqueta azul',
        '../assets/images/chaqueta-azul.jpg',
        ['Mujer', 'Prendas Vestir', 'Chaquetas deportivas'],
        238.99)
]

@Injectable({
    providedIn: 'root'
})
export class ProductService {

    /** Método que devuelve un array con los productos */
    getProducts(): ProductModel[] {
        return PRODUCTOS;
    }

    /** Método que añade un producto que se le pasa */

```

```

addProduct(producto: ProductModel): void {
    PRODUCTOS.push(producto);
}

/** Método que obtiene un producto cuyo id se le pasa */
getProduct(id: number): ProductModel {
    return PRODUCTOS.find(producto => producto.idProduct === id) as
ProductModel;
}

/** Método que borra un producto cuyo id se le pasa */
deleteProduct(id: number): void {
    let indiceProducto = PRODUCTOS.findIndex(producto => producto.idProduct
=== id);
    PRODUCTOS.splice(indiceProducto, 1);
}

createEmptyProduct(): ProductModel {
    return new ProductModel('SKU#', '', '', [''], -1);
}

```

10. Vamos a crear los **COMPONENTES** que aparecen en el menú de navegación:

10.1.-HomeComponent

Simplemente mostrará información de la empresa.

Lo creamos con el comando:

ng g c home

Lo que mostrará será algo como:

¿A qué nos dedicamos?

Nuestra empresa lleva 10 años dedicándose a la venta online de prendas de vestir. Tenemos prendas de todo tipo: sports, vestir, niños, adultos,etc.

[home.component.html](#)

```
<h3>¿A qué nos dedicamos?</h3>
<hr />
<p>
  Nuestra empresa lleva 10 años dedicándose a la venta online de prendas de
  vestir. Tenemos prendas de todo tipo: sports, vestir, niños, adultos,etc.
</p>
```

home.component.ts

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

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent{

  constructor() { }

}
```

10.2.-Añadimos el componente HomeComponent al módulo de Routing

En nuestro fichero de configuración de routing: “**app-routing.module.ts**”, tenemos que:

- 1º) Importar el nuevo componente HomeComponent para poder usarlo
- 2º) Añadir una ruta (que coincida con la que hemos puesto en el componente barra de navegación creado anteriormente para home) en el array de rutas.

app-routing.module.ts

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

// componentes
import { HomeComponent } from './home/home.component';
```

```
// rutas
const routes: Routes = [
  {path: '', component: HomeComponent},
  {path: 'home', component: HomeComponent}
];

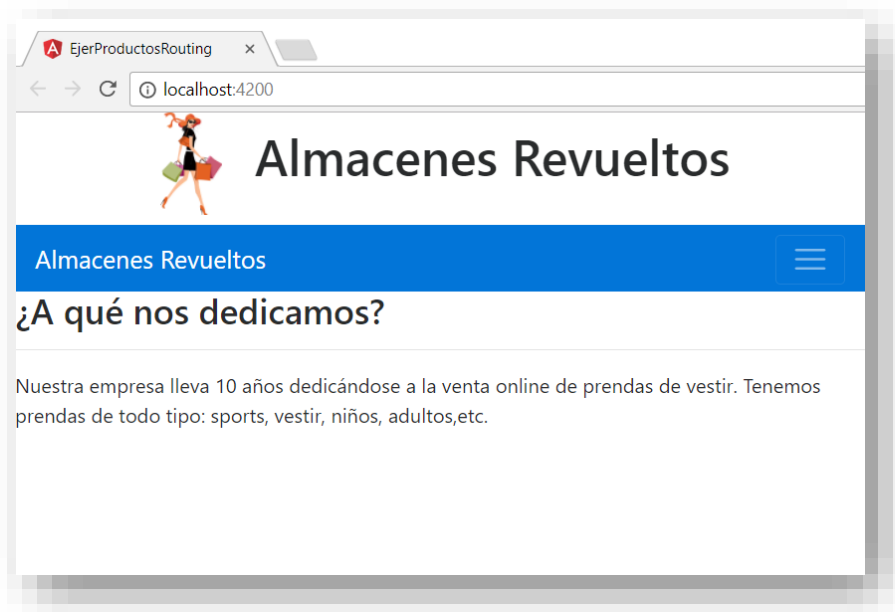
@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

10.3.-Comprobamos que en la plantilla del AppComponent esté la directiva <router-outlet> para que nos cargue los componentes enrutados

En nuestro fichero del componente principal: **app.component.html** debe aparecer:

```
<router-outlet></router-outlet>
```

Si lo probamos, veremos:

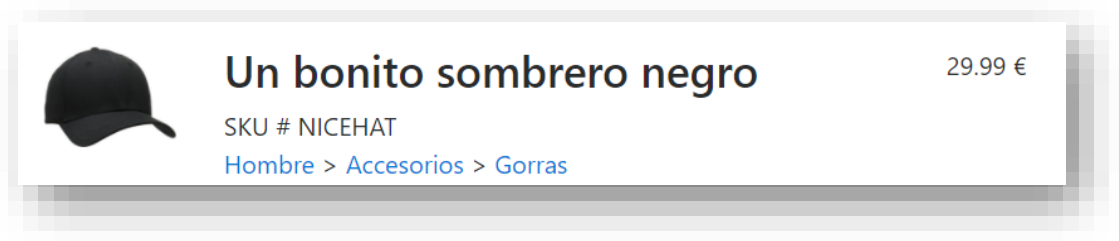


10.4.-Ahora vamos a crear los componentes necesarios para mostrar el listado de productos.

Comenzamos por el componente **ItemProductComponent**

Mostrará los datos abreviados de un producto. **Representa cada una de las cajas o rectángulo del listado de productos.**

Su aspecto visual será:



Lo creamos con el comando:

```
ng g c /products/itemProduct
```

El código del componente es:

ítem-product.component.html

```
<div class="container">
  <div class="row">
    <img class="col-2" [src]="product.imageUrl" [ngStyle]="{ height: '30%' }"
  />
    <div class="col-8">
      <div class="header">
        <h3>{{ product.name }}</h3>
      </div>
      <div>SKU # {{ product.sku }}</div>
      <div>
        <span *ngFor="let name of product.department; let i = index">
          <a href="#">{{ name }} </a>
          <span>{{ i < product.department.length - 1 ? ">" : "" }}</span>
        </span>
      </div>
    </div>
    <div class="col-2">{{ product.price }} €</div>
  </div>
</div>
```

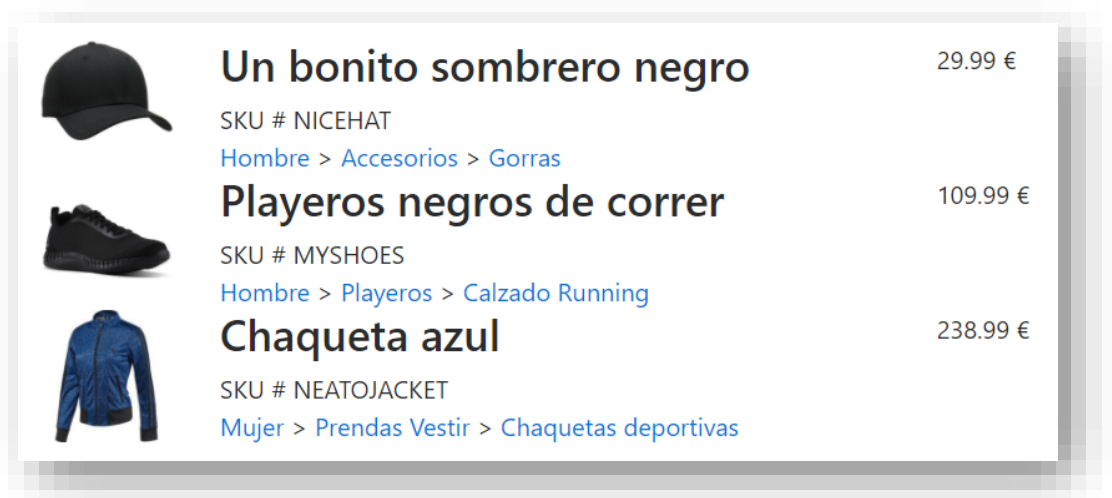
ítem-product.component.ts

```
import { Component, Input } from '@angular/core';
import { ProductModel } from '../../model/product.model'
@Component({
  selector: 'app-item-product',
  templateUrl: './item-product.component.html',
  styleUrls: ['./item-product.component.css']
})
export class ItemProductComponent {
  @Input() product!: ProductModel;
}
```

Ahora creamos el componente que representa el listado de productos y que se llamará: **ProductsComponent**

Mostrará un listado de productos (incorpora el componente **ItemProductComponent** para hacer el listado).

Su aspecto visual será:



Lo creamos con el comando:

```
ng g c /products/products
```

El código del componente es:

products.component.html

```
<div class="container">
  <app-item-product
    *ngFor="let miProducto of productos"
    [product]="miProducto"
    [routerLink]="['product', miProducto.idProduct]"
    routerLinkActivated="active"
  ></app-item-product>
</div>
```

products.component.ts

```
import { Component, OnInit } from '@angular/core';
import { ProductModel } from '../../model/product.model';
import { ProductService } from '../../services/product.service';

@Component({
  selector: 'app-products',
  templateUrl: './products.component.html',
  styleUrls: ['./products.component.css']
})
export class ProductsComponent implements OnInit {
  productos!: ProductModel[];

  constructor(private productService: ProductService) { }

  ngOnInit(): void {
    // obtengo los productos del servicio
    this.productos = this.productService.getProducts();
  }
}
```

Añadimos el componente **ProductsComponent** al módulo de Routing

Para poder navegar hacia este componente nuevo que hemos creado, debemos añadir una nueva ruta en la tabla de rutas del fichero de configuración de routing: **“app-routing.module.ts”**. Tendremos que:

- 1º) Importar el nuevo componente **ProductsComponent** para poder usarlo
- 2º) Añadir una ruta (que coincida con la que hemos puesto en el componente barra de navegación creado al principio, para **Productos**) en el array de rutas.

app-routing.module.ts

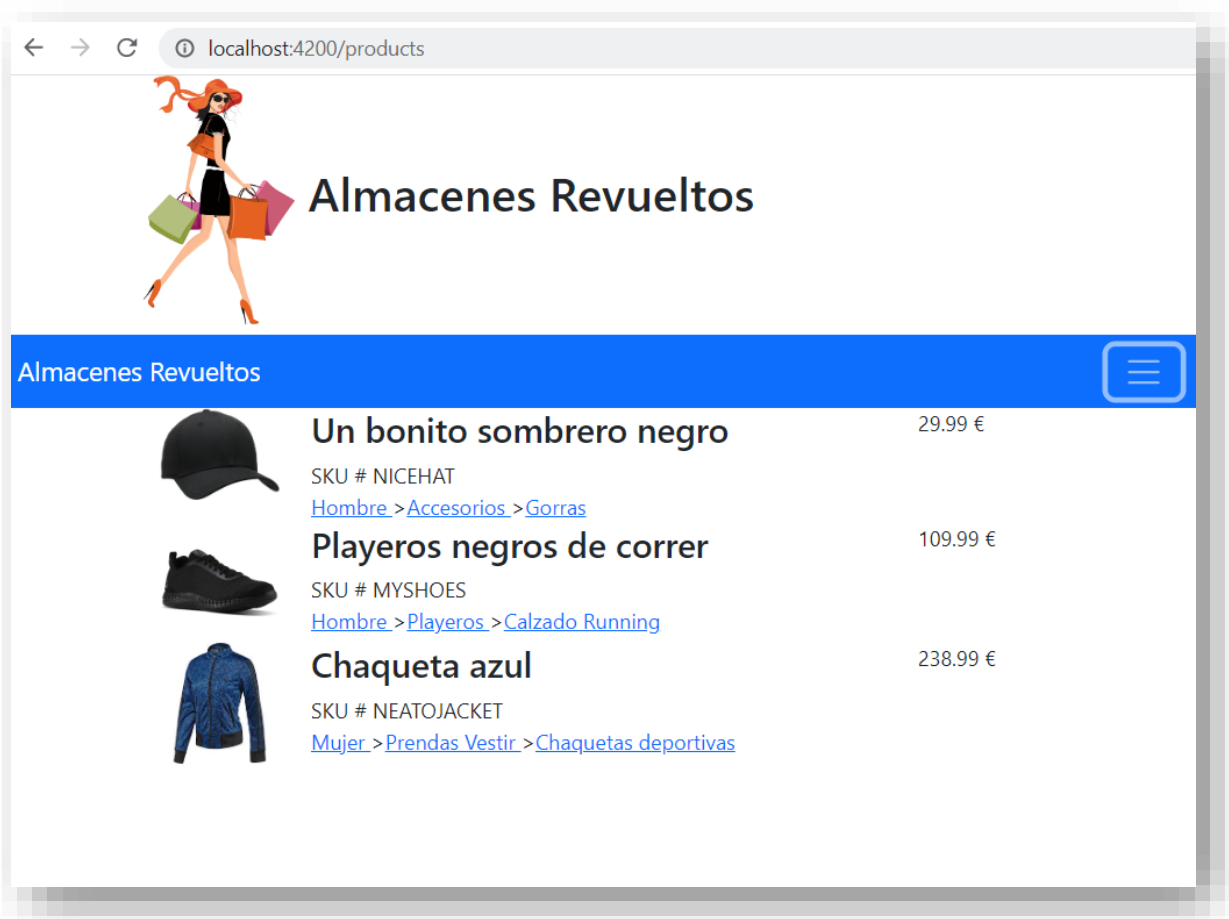
```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';

// componentes
import { HomeComponent } from '../home/home.component';
import { ProductsComponent } from '../products/products/products.component';

const routes: Routes = [
  {path: '', component: HomeComponent},
  {path: 'home', component: HomeComponent},
  {path: 'products', component: ProductsComponent}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Si lo probamos ahora, veremos:



10.5.-Barra de navegación de productos: **NavbarProductsComponent**

Mostrará una barra de navegación con las opciones: añadir, gestionar y borrar, en la cabecera del componente **ProductsComponent** (el listado de productos).

Su aspecto visual será:



Para crearla, tecleamos:

```
ng g c products/ui/navbarProducts
```

Y su código será:

navbar-products.component.html

```
<div class="container">
  <button
    type="button"
    class="btn btn-secondary"
    routerLink="add-product"
    routerLinkActivated="active"
  >
    Add Product
  </button>
  <button
    type="button"
    class="btn btn-secondary"
    routerLink="/home"
    routerLinkActivated="active"
  >
    Gestionar Productos
  </button>
</div>
```

navbar-products.component.ts

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

@Component({
  selector: 'app-navbar-products',
  templateUrl: './navbar-products.component.html',
  styleUrls: ['./navbar-products.component.css']
})
```

```

}))
export class NavbarProductsComponent {

}

```

navbar-products.component.css

```

button{
    margin:10px;
}

```

Y para que nuestra barra se muestre, la añadimos en **products.component.html** de manera que ahora ese fichero, nos queda:

products.component.html

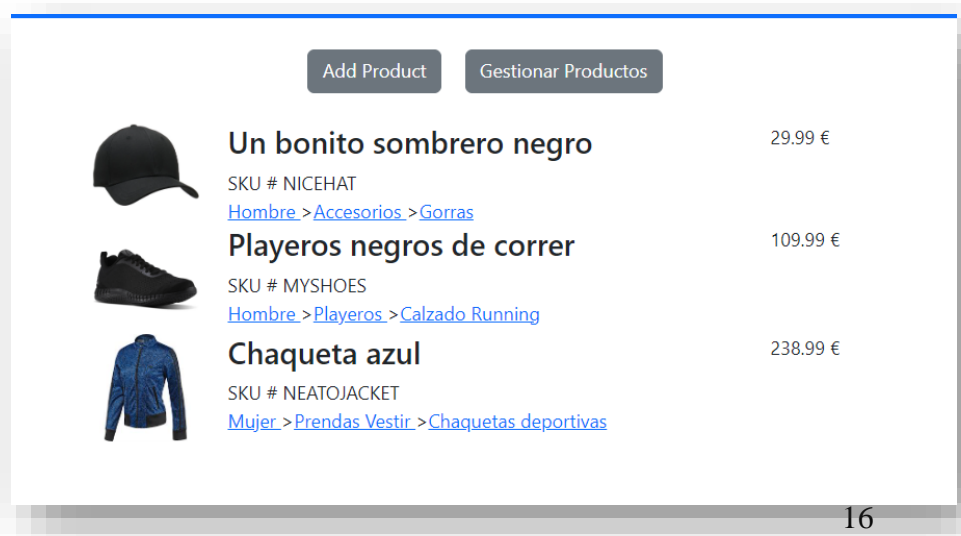
```

<div class="container">
  <div class="row my-3" [ngStyle]="{ 'text-align': 'center' }">
    <app-navbar-products></app-navbar-products>
  </div>
  <app-item-product
    *ngFor="let miProducto of productos"
    [product]="miProducto"
    [routerLink]="['product', miProducto.idProduct]"
    routerLinkActivated="active"
  ></app-item-product>
</div>

```

Con estos cambios, nuestro ProductsComponent, nos ha quedado con una barra superior en la que **aún no funciona nada más que el botón GESTIONAR PRODUCTOS** que nos llevará a Home de nuevo (*por no complicar más el ejercicio*). El botón **ADD PRODUCT** de momento **no funcionará** hasta que creamos el componente añadir

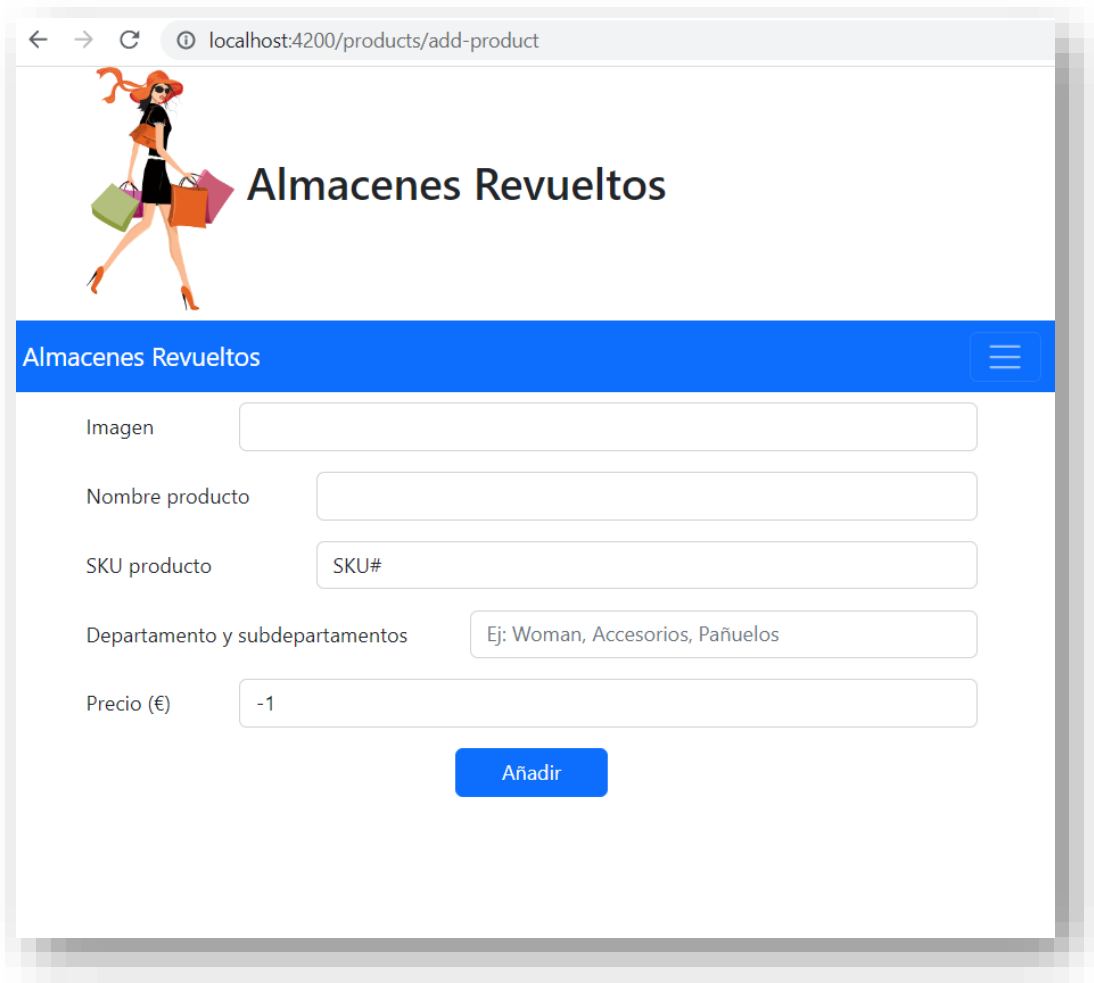
producto que será un formulario para rellenar los datos de un nuevo producto:



10.6.-Vamos a crear ahora el componente **AddProductComponent** para poder darle funcionalidad al botón **ADD PRODUCT** del listado de productos

Corresponde a este componente permitir añadir un nuevo producto a la lista de productos de la tienda. Será llamado desde la barra de navegación que hay dentro del listado de productos a través del botón **ADD PRODUCT**.

Su aspecto visual será:



The screenshot shows a web browser window with the URL `localhost:4200/products/add-product`. The page features a header with the logo of a woman carrying shopping bags and the text 'Almacenes Revueltos'. Below the header is a blue navigation bar with the same text and a hamburger menu icon. The main content area contains a form with the following fields:

- Imagen:
- Nombre producto:
- SKU producto:
- Departamento y subdepartamentos:
- Precio (€):

At the bottom of the form is a blue button labeled 'Añadir'.

Para crearlo dentro de productos, en la terminal tecleamos:

```
ng g c products/addProduct
```

En cuanto a la lógica, lo que tiene que hacer es:

- 1) Mostrar la página para introducir los datos del producto que será un **formulario**.
(la plantilla del componente será)

- 2) Al pulsar un botón grabar o similar en la plantilla del punto 1) me creará un producto con esos datos y lo guardará en el array de productos (realmente sería en una BD si la tuviésemos)
- 3) Nos llevará el router a la página listado de productos para ver el nuevo producto en la lista.

Vamos a implementar esos 3 pasos poco a poco y en orden.

Lo primero creamos la plantilla HTML:

add-product.component.html

```
<div class="container mt-2">
  <form>
    <div class="mb-3 row">
      <label for="imagen" class="col-sm-2 col-form-label">Imagen</label>
      <div class="col">
        <input
          type="text"
          class="form-control"
          [(ngModel)]="producto.imageUrl"
          id="imagen"
          name="imagen"
        />
      </div>
    </div>
    <div class="row mb-3">
      <label for="nombre" class="col-sm-3 col-form-label">Nombre producto</label>
      <div class="col">
        <input
          type="text"
          class="form-control"
          id="nombre"
          [(ngModel)]="producto.name"
          name="nombre"
        />
      </div>
    </div>
    <div class="row mb-3">
      <label for="sku" class="col-sm-3 col-form-label">SKU producto</label>
      <div class="col">
        <input
          type="text"
          class="form-control"
          id="sku"
          [(ngModel)]="producto.sku"
          name="sku"
        />
      </div>
    </div>
  </form>
</div>
```

```

        placeholder="SKU#123"
      />
    </div>
  </div>
  <div class="row mb-3">
    <label for="departamento" class="col-sm-5 col-form-label">
      Departamento y subdepartamentos</label>
    >
    <div class="col">
      <input
        type="text"
        class="form-control"
        id="departamento"
        [(ngModel)]="producto.department"
        placeholder="Ej: Woman, Accesorios, Pañuelos"
        name="departamento"
      />
    </div>
  </div>
  <div class="row mb-3">
    <label for="precio" class="col-sm-2 col-form-label">Precio (€)</label>
    <div class="col">
      <input
        type="number"
        class="form-control"
        id="precio"
        [(ngModel)]="producto.price"
        name="precio"
      />
    </div>
  </div>
</form>
<div class="row">
  <button type="submit" class="col-2 mx-auto btn btn-primary"
    (click)="addProduct()">Añadir
  </button>
</div>
</div>

```

Fijaros que la plantilla usa `[(ngModel)]`, por tanto, si no estuviese ya, habría que importar **FormsModule** en **app.module.ts**

Nos quedará el app.module.ts:

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { AppRoutingModuleModule } from './app-routing.module';

import { AppComponent } from './app.component';
import { HeaderComponent } from './ui/header/header.component';
import { NavbarComponent } from './ui/navbar/navbar.component';
import { HomeComponent } from './home/home.component';
import { ItemProductComponent } from './products/item-product/item-product.component';
import { ProductsComponent } from './products/products/products.component';
import { NavbarProductsComponent } from './products/ui/navbar-products/navbar-products.component';
import { AddProductComponent } from './products/add-product/add-product.component';

@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    NavbarComponent,
    HomeComponent,
    ItemProductComponent,
    ProductsComponent,
    NavbarProductsComponent,
    AddProductComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModuleModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

Fijaros también que en la plantilla enlazamos a “producto.XXX” los input. Por tanto, tenemos que tener en nuestro controlador (.ts) una propiedad de tipo **ProductModel** y llamada “**producto**”.

Una vez que tenemos la plantilla, vamos a nuestro controlador (archivo: **add-product.component.ts**) y nos aseguramos de:

- 1) Declarar una propiedad de tipo ProductModel y llamada producto. Para eso, necesitamos importar la clase ProductModel:

```
import { Component } from '@angular/core';
import { ProductModel } from 'src/app/model/product.model';

@Component({
  selector: 'app-add-product',
  templateUrl: './add-product.component.html',
  styleUrls: ['./add-product.component.css']
})
export class AddProductComponent {
  product!: ProductModel;
}
```

- 2) En el constructor, deberíamos inicializar ese producto a un “new ProductModel()”. Pero tal como tenemos diseñado el ejercicio, tenemos un **PROBLEMA**: **nuestro constructor requiere que le pasemos todos los atributos de un producto y aún no los tenemos, ya que nos los darán los inputs del formulario**; además no podemos tener dos constructores con distintos parámetros. Una **SOLUCIÓN**: es usar el método que crea un producto vacío y que hemos añadido en nuestro servicio de acceso a datos y que es este:

```
/** Método que crea un producto vacío */
createEmptyProduct(): ProductModel {
  return new ProductModel('SKU#', '', '', [''], -1);
}
```

Pero para poder usar ese método y dado que está en un servicio, tenemos que **INYECTAR EL SERVICIO ProductService** en nuestro componente **AddProductComponent**. Como ya lo hicimos en el listado de productos, para inyectarlo consiste en definir una variable de ese tipo como parámetro del constructor del componente. Así, nuestro fichero **add-product.component.ts** nos quedará:

```
import { Component } from '@angular/core';
import { ProductModel } from 'src/app/model/product.model';
import { ProductService } from 'src/app/services/product.service';

@Component({
  selector: 'app-add-product',
  templateUrl: './add-product.component.html',
  styleUrls: ['./add-product.component.css']
})
export class AddProductComponent {
  product!: ProductModel;

  constructor(private _productService: ProductService){}
}
```

Ahora ya podemos inicializar nuestro producto en **add-product.component.ts** llamando al método “**createEmptyProduct()**” del servicio de acceso a datos. Como todos los códigos que accedan a un servicio **no puede hacerse desde el constructor**, sino desde el método **ngOnInit** de la interface **OnInit**. Así que implementamos dicha interface y en el método **ngOnInit()** accederemos al servicio:

```
import { Component, OnInit } from '@angular/core';
import { ProductModel } from 'src/app/model/product.model';
import { ProductService } from 'src/app/services/product.service';

@Component({
  selector: 'app-add-product',
  templateUrl: './add-product.component.html',
  styleUrls: ['./add-product.component.css']
})
export class AddProductComponent implements OnInit {
  producto!: ProductModel;

  constructor(private _productService: ProductService) { }

  ngOnInit(): void {
    this.producto = this._productService.createEmptyProduct();
  }
}
```

- 3) Ahora creamos el método **addProduct()** que es llamado desde la plantilla al pulsar el botón AÑADIR y que añade el producto al array de productos del servicio (en realidad lo añadiría a la BD si la hubiera) y navega al listado de productos. **Para navegar desde código necesito inyectar también el servicio Router en el constructor:**

```
export class AddProductComponent implements OnInit {
  producto!: ProductModel;

  constructor(private _productService: ProductService,
    private _router: Router) { }

  ngOnInit(): void {
    this.producto = this._productService.createEmptyProduct();
  }

  /** Método que se llama desde la plantilla html al pulsar el botón añadir */
  addProduct(): void {
    // antes de añadir el producto, se debe manipular el departamento, pues se ha
    // recogido como un array con un solo string separado por comas y en el modelo debe ser un
    // array de strings. Para convertirlo hacemos:
```

```

let dpto: string = this.producto.department.toString();
this.producto.department = dpto.split(',');
// ahora ya puedo añadir el producto llamando al método addProduct del servicio
this._productService.addProduct(this.producto);
// y navego al listado de productos
this._router.navigateByUrl('products');
}
}

```

Fijaros que en este método extraigo el departamento a un string, lo convierto a Array de strings y luego lo vuelvo a asignar.

Es necesario hacerlo porque producto.departamento es de tipo Array<string>, pero desde la plantilla lo escribimos como “Mujer, Pantalones, Sport” y esto no es un ARRAY, es un STRING. Si lo dejásemos así, se guardaría como si fuese un array de un string.

Fijaros que después de añadir, llamamos al router.navigateByUrl() para que se muestre el listado de productos. Para que eso funcione, previamente, en el constructor tuvimos que inyectarlo. Fijaros en el constructor.

4) Por último, tenemos que añadir esta nueva ruta al módulo de enrutado: `app-routing.module.ts`:

```

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

// componentes
import { HomeComponent } from '../home/home.component';
import { AddProductComponent } from '../products/add-product/add-product.component';
import { ProductsComponent } from
'../products/products/products.component';

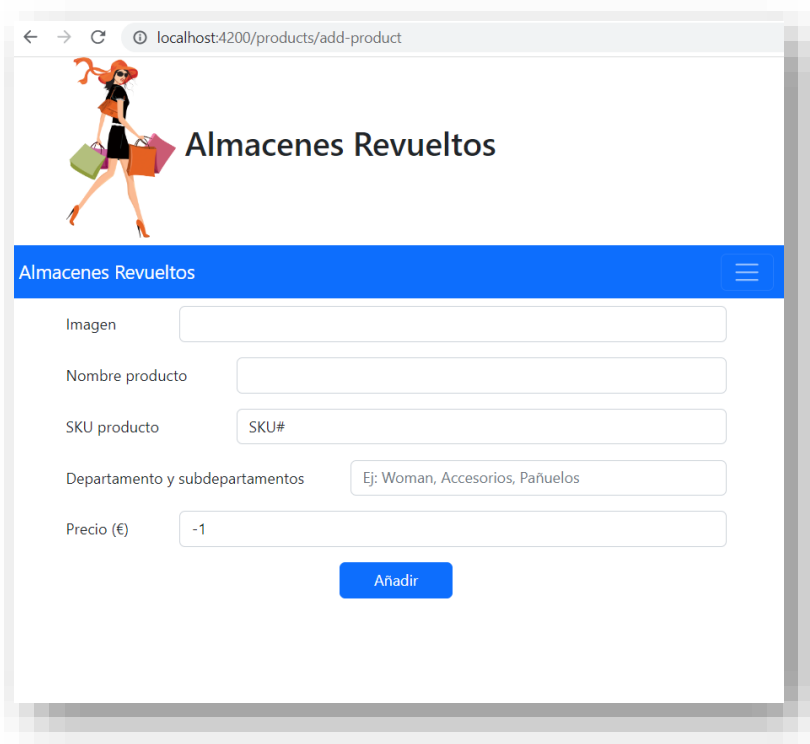
const routes: Routes = [
  {path: '', component: HomeComponent},
  {path: 'home', component: HomeComponent},
  {path: 'products', component: ProductsComponent},
  {path: 'products/add-product', component: AddProductComponent}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }


```

EJEMPLO EN FUNCIONAMIENTO DEL ADD PRODUCTO

Componente:



localhost:4200/products/add-product

 Almacenes Revueltos

Almacenes Revueltos

Imagen

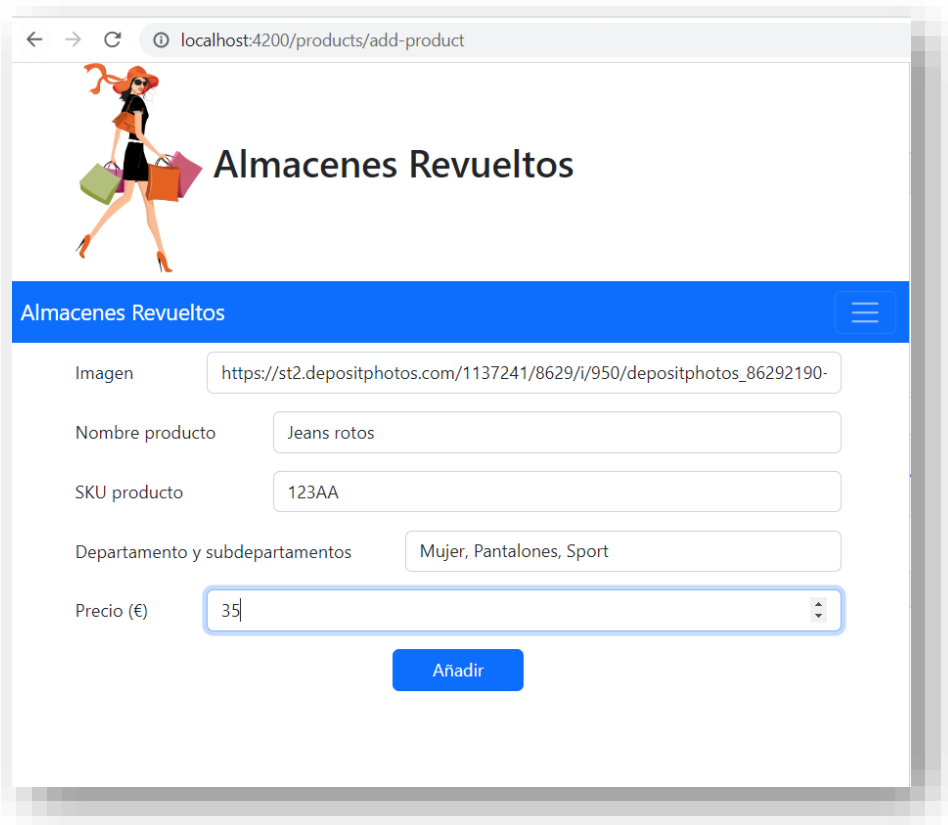
Nombre producto

SKU producto


Departamento y subdepartamentos

Precio (€)

Si rellenamos ahora el formulario con datos de uno vaqueros rotos:



localhost:4200/products/add-product

 Almacenes Revueltos

Almacenes Revueltos

Imagen

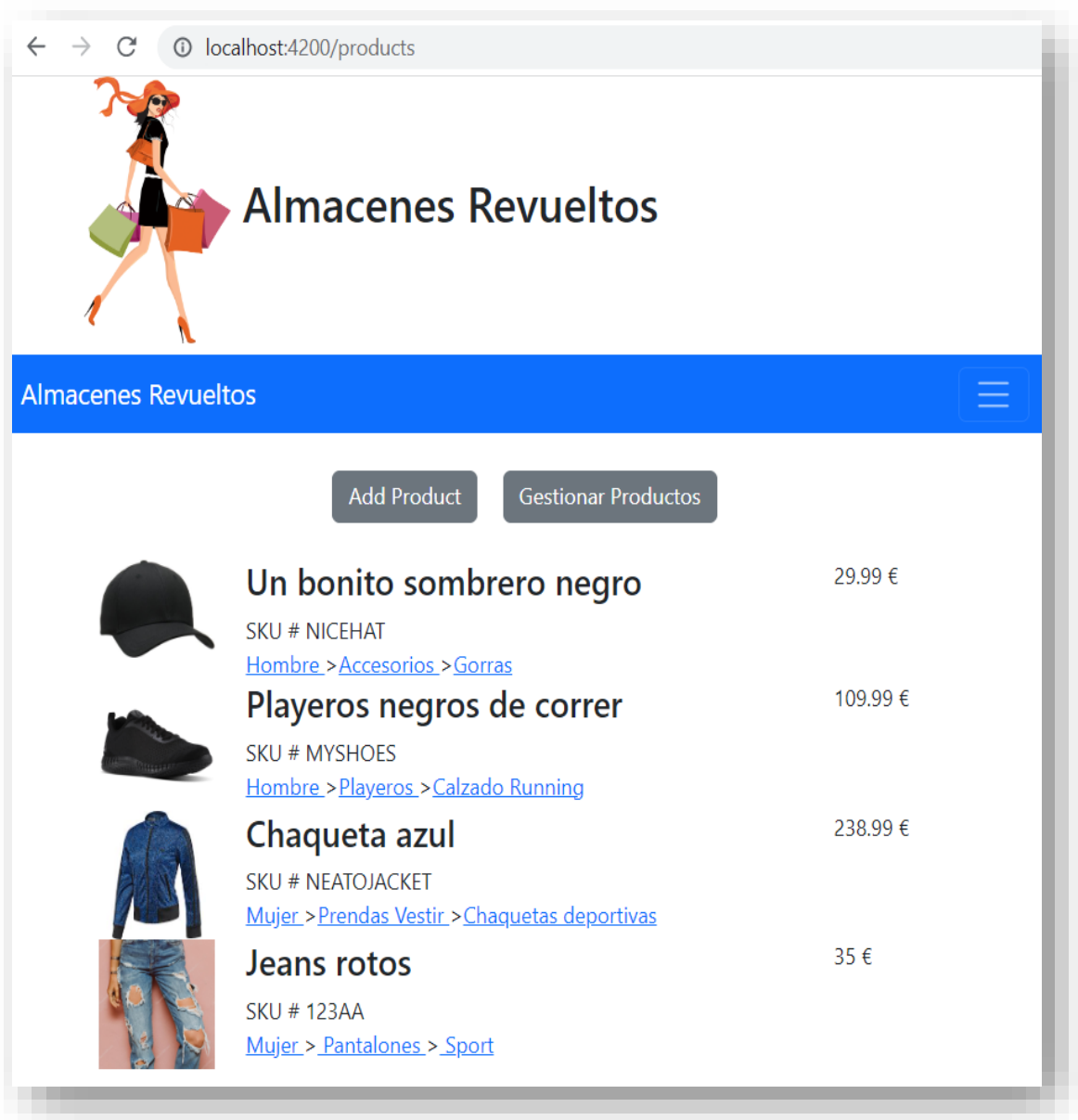
Nombre producto

SKU producto

Departamento y subdepartamentos

Precio (€)

Y al pulsar AÑADIR:



10.7.-Vamos a crear ahora el componente **DetailProductComponent** para poder darle funcionalidad a cuando se hace click sobre un elemento de la lista de productos (da igual en qué lugar del item se haga click)

Cuando estamos en el listado de productos, si pulsamos sobre uno de ellos, navegaremos por su id a la página de detalles del producto, donde habría más información del mismo.

Su aspecto visual será:



Para crearlo en la carpeta products, tecleamos en la terminal:

```
ng g c products/detailProduct
```

PASOS:

- 1) Lo primero que tenemos que hacer, es **recoger el ID que vendrá en la URL y averiguar qué producto es el** que nos viene de la URL, usando **paramMap** de **ActivatedRoute**. En el fichero **detail-product.component.ts** escribimos:

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { ProductModel } from 'src/app/model/product.model';
import { ProductService } from 'src/app/services/product.service';
```

```

@Component({
  selector: 'app-detail-product',
  templateUrl: './detail-product.component.html',
  styleUrls: ['./detail-product.component.css']
})
export class DetailProductComponent implements OnInit {
  idProductoElegido!: number;
  productoElegido!: ProductModel;

  constructor(private _activatedRoute: ActivatedRoute,
    private _productService: ProductService) { }

  ngOnInit(): void {
    // recojo el id que llega en la url a este componente al navegar hacia él
    let idRecogidoDeLaUrl = this._activatedRoute.snapshot.paramMap.get('id');
    // compruebo que no sea nulo
    if (idRecogidoDeLaUrl !== null) {
      this.idProductoElegido = +idRecogidoDeLaUrl;
      this.productoElegido = this._productService.getProduct(this.idProductoElegido);
    }
  }
}

```

- 2) Ya podemos **diseñar la plantilla HTML** de este componente, tomando los datos de la propiedad “*productoElegido*”:

detail-product.component.html

```

<div class="container">
  <img class="col-12" [src]="productoElegido.imageUrl" />
  <div class="col-12">
    <div class="row text-center mt-3">
      <h2>{{ productoElegido.name }}</h2>
    </div>
    <div class="row justify-content-center">
      SKU # {{ productoElegido.sku }}
    </div>
    <div class="text-center">
      <span *ngFor="let name of productoElegido.department; let i = index">
        <a href="#">{{ name }} </a>
        <span>{{ i < productoElegido.department.length - 1 ? ">" : "" }}</span>
      </span>
    </div>
    <div class="row justify-content-center">{{ productoElegido.price }} €</div>
    <div class="text-center">
      <button
        class="btn btn-primary"

```

```

      (click)="deleteProduct(productoElegido.idProduct)"
    >
      Borrar
    </button>
  </div>
</div>
</div>

```

- 3) Como hemos puesto un **botón de borrar**, vamos a programar su evento click, mediante la creación del método **deleteProduct(...)** en el **fichero controlador de este componente (.ts)**:

La forma que he pensado para borrarlo ha sido:

- Lo quito del array de productos que está en nuestro servicio de acceso a datos, llamando a los métodos del servicio.
- Puesto que ya no existirá, **dirijo al ROUTER a la página** del listado de **PRODUCTOS**, usando **router.navigateByUrl()**. Para poder llamar a los métodos del router debo inyectarlo en el constructor del componente **DetailProductComponent**:

```

import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { ProductModel } from 'src/app/model/product.model';
import { ProductService } from 'src/app/services/product.service';

@Component({
  selector: 'app-detail-product',
  templateUrl: './detail-product.component.html',
  styleUrls: ['./detail-product.component.css']
})
export class DetailProductComponent implements OnInit {
  idProductoElegido!: number;
  productoElegido!: ProductModel;

  constructor(private _activatedRoute: ActivatedRoute,
    private _productService: ProductService,
    private _router: Router) { }

  ngOnInit(): void {
    // recojo el id que llega en la url a este componente al navegar hacia él
    let idRecogidoDeLaUrl = this._activatedRoute.snapshot.paramMap.get('id');
    // compruebo que no sea nulo
    if (idRecogidoDeLaUrl !== null) {
      this.idProductoElegido = +idRecogidoDeLaUrl;
      this.productoElegido = this._productService.getProduct(this.idProductoElegido);
    }
  }
}

```

```

}

/** Método que borra el producto del array de productos
 *
 */
deleteProduct(id: number): void {
  this._productService.deleteProduct(id);
  this._router.navigateByUrl('products');
}
}

```

- 4) Debemos asegurarnos también, por si no lo hemos hecho primero, que haya una ruta de navegación **/products/product/:id** hacia este componente en nuestro fichero de **ROUTING. Y de importar el COMPONENTE:**

app-routing.module.ts

```

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

// componentes
import { HomeComponent } from '../home/home.component';
import { AddProductComponent } from '../products/add-product/add-product.component';
import { DetailProductComponent } from '../products/detail-product/detail-product.component';
import { ProductsComponent } from '../products/products/products.component';

const routes: Routes = [
  {path: '', component: HomeComponent},
  {path: 'home', component: HomeComponent},
  {path: 'products', component: ProductsComponent},
  {path: 'products/add-product', component: AddProductComponent},
  {path: 'products/product/:id', component: DetailProductComponent}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }

```

- 5) Debemos asegurarnos también, que el componente **ProductsComponent** hayamos configurado en su HTML con **RouterLink** que *cuando se pulse sobre un elemento nos lleve a esa ruta que acabamos de poner en la tabla de rutas del ROUTING:*

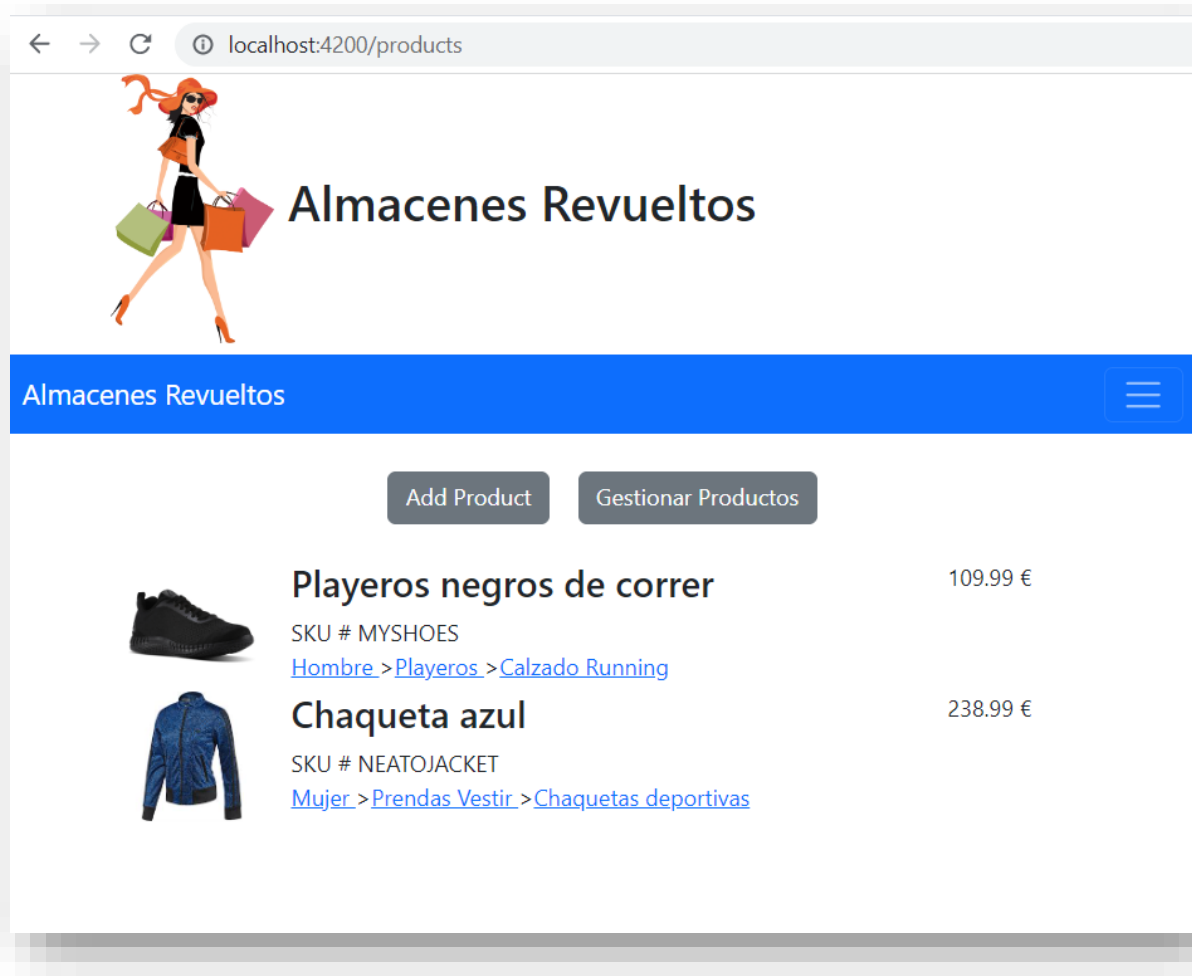
products.component.html.ts

```
<div class="container">
  <div class="row my-3" [ngStyle]="{ 'text-align': 'center' }">
    <app-navbar-products></app-navbar-products>
  </div>
  <app-item-product
    *ngFor="let miProducto of productos"
    [product]="miProducto"
    [routerLink]="['product',miProducto.idProduct]"
    routerLinkActive="active"
  ></app-item-product>
</div>
```

Si probamos la aplicación, veremos:



Si pulsamos BORRAR:



10.8.-Vamos a crear ahora el componente **ContactComponent**

Mostrará un formulario de contacto.

Su aspecto visual será:

Nombre:

Correo electrónico

Mensaje:

Enviar Volver

Para crearlo, tecleamos:

ng g c contact

Y su código será:

contact.component.html

```
<form class="p-3">
  <div class="mb-3">
    <label for="nombre" class="form-label">Nombre:</label>
    <input class="form-control" type="text" id="nombre" [disabled]="enviando"
  />
</div>
<div class="mb-3">
  <label for="email" class="form-label">Correo electrónico</label>
  <input
    class="form-control"
    type="email"
    placeholder="name@example.com"
    id="email"
    [disabled]="enviando"
  />
</div>
<div class="mb-3">
  <label for="mensaje" class="form-label">Mensaje:</label>
  <textarea
    class="form-control"
    rows="3"
    [(ngModel)]="mensaje"
    name="textoMensaje"
    [disabled]="enviando"
  ></textarea>
  <br />
</div>
<div *ngIf="!enviando">
  <div class="text-center">
    <button class="btn btn-primary me-2" (click)="enviarMensaje()">
      Enviar
    </button>
    <button class="btn btn-primary ms-2" (click)="volverAtras()">
      Volver
    </button>
  </div>
</div>
<div *ngIf="detalles">
  {{ detalles }}
</div>
</form>
```


Como estoy usando **[(ngModel)]** para hacer binding en doble sentido en `<textarea>`, necesito importar el módulo **FormsModule** en *app.module.ts* (si no lo habíamos hecho ya con *AddProductComponent*). Por tanto, comprobamos que en el fichero **“app.module.ts”** esté lo remarcado en morado:

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';
import { HeaderComponent } from './ui/header/header.component';
import { NavbarComponent } from './ui/navbar/navbar.component';
import { HomeComponent } from './home/home.component';
import { ItemProductComponent } from './products/item-product/item-product.component';
import { ProductsComponent } from './products/products/products.component';
import { NavbarProductsComponent } from './products/ui/navbar-products/navbar-products.component';
import { AddProductComponent } from './products/add-product/add-product.component';
import { DetailProductComponent } from './products/detail-product/detail-product.component';
import { ContactComponent } from './contact/contact.component';

@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    NavbarComponent,
    HomeComponent,
    ItemProductComponent,
    ProductsComponent,
    NavbarProductsComponent,
    AddProductComponent,
    DetailProductComponent,
    ContactComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Por otro lado, como en el formulario hay un botón VOLVER ATRÁS, hay que importar el **servicio Location** de Angular e inyectarlo en el controlador del componente, para usar su método **back()**.

Así el código del controlador de nuestro componente, nos quedará:

contact.component.ts

```
import { Component } from '@angular/core';
import { Location } from '@angular/common';
@Component({
  selector: 'app-contact',
  templateUrl: './contact.component.html',
  styleUrls: ['./contact.component.css']
})
export class ContactComponent {
  enviando: boolean;
  mensaje: string;
  detalles!: string;

  constructor(private _location: Location) {
    this.enviando = false;
    this.mensaje = '';
  }

  /** Método que envia el mensaje y vuelve al componente anterior */
  enviarMensaje() {
    this.enviando = true;
    this.detalles = 'Enviando mensaje...';

    setTimeout(() => {
      this.enviando = false;
      this.volverAtras();
    }, 2000);
  }

  /** Método que navega hacia atrás a la página anteriormente visitada */
  volverAtras() {
    this._location.back();
  }
}
```

10.9.-Añadimos el componente ContactComponent al módulo de Routing

En nuestro fichero de configuración de routing: “**app-routing.module.ts**”, tenemos que:

- 1º) Importar el nuevo componente ContactComponent para poder usarlo
- 2º) Añadir una ruta (que coincida con la que hemos puesto en el componente barra de navegación creado al principio para Contacto) en el array de rutas.

app-routing.module.ts

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

// componentes
import { HomeComponent } from '../home/home.component';
import { AddProductComponent } from '../products/add-product/add-product.component';
import { DetailProductComponent } from '../products/detail-product/detail-product.component';
import { ProductsComponent } from '../products/products/products.component';
import { ContactComponent } from '../contact/contact.component';

const routes: Routes = [
  { path: '', component: HomeComponent },
  { path: 'home', component: HomeComponent },
  { path: 'products', component: ProductsComponent },
  { path: 'products/add-product', component: AddProductComponent },
  { path: 'products/product/:id', component: DetailProductComponent },
  { path: 'contact', component: ContactComponent }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

11. Añadimos al módulo de Routing la ruta para errores, que vaya a home

El fichero “**app-routing.module.ts**” ya existe y estará casi configurado. Acordaros que además de las rutas, **hay que importar cada componente al que queráis enrutar**.

```
const routes: Routes = [  
  {path: '', component: HomeComponent},  
  {path: 'home', component: HomeComponent},  
  {path: 'products', component: ProductsComponent},  
  {path: 'products/product/:id', component: DetailProductComponent},  
  {path: 'contact', component: ContactComponent},  
  {path: 'products/add-product', component: AddProductComponent},  
  {path: '**', redirectTo: 'home'}  
];
```