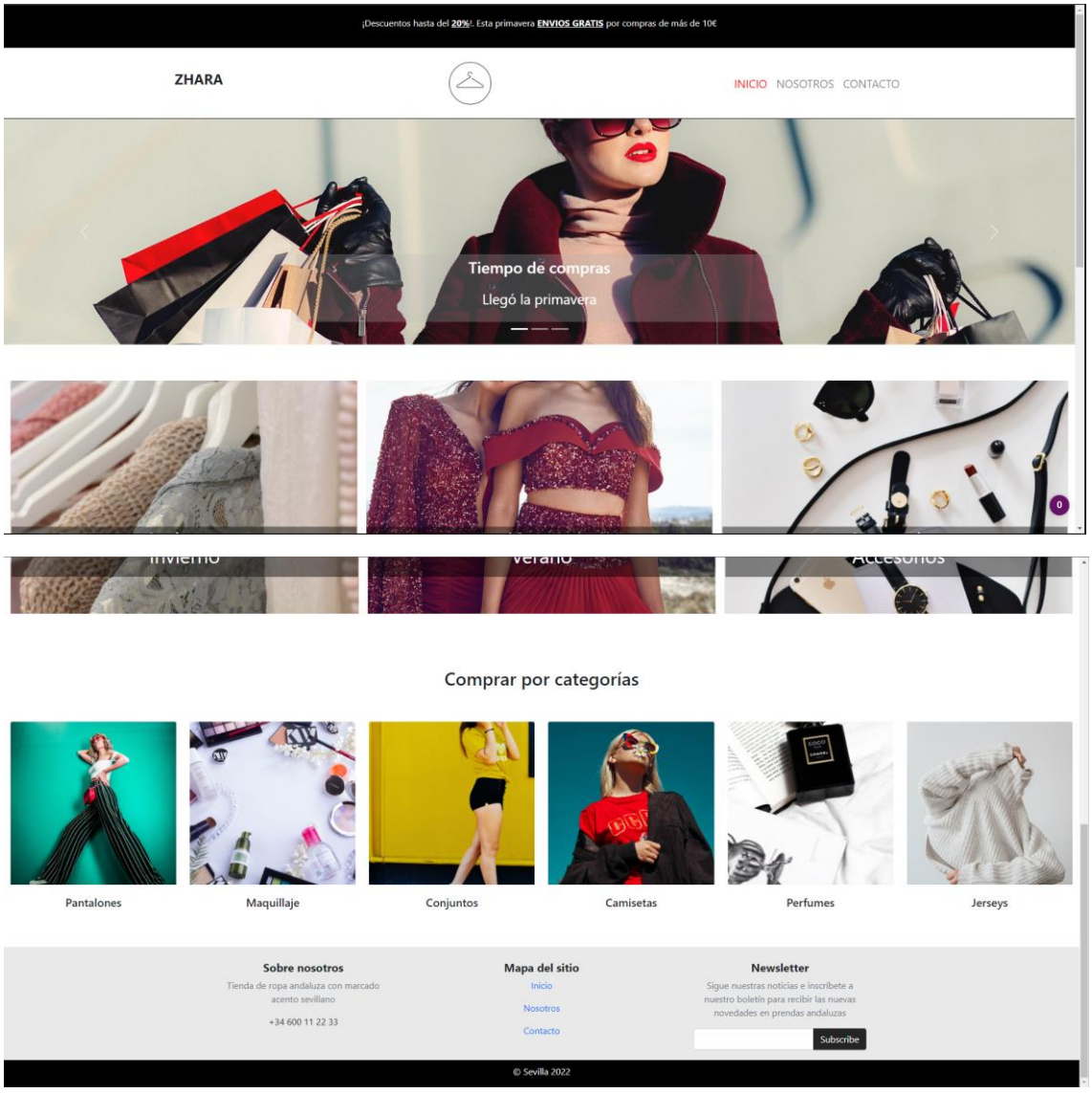
INICIO

Esta es la página principal de Zhara, la tienda de moda, que está compuesta por una cabecera, contenido y footer.

```
1  <app-navbar></app-navbar>
2
3  <router-outlet></router-outlet>
4
5  <app-footer></app-footer>
```



La cabecera está compuesta por un anuncio sobre descuentos y envíos. Tanto el porcentaje como las estaciones del año son de tipo ENUM


```enum
export enum ESTACIONES {
    VERANO = "verano",
    PRIMAVERA = "primavera",
    OTONO = "otoño",
    INVIERNO = "invierno"
}
```

```enum
export enum TIPODESCUENTO {

    BAJO = 10,
    MEDIO = 20,
    ALTO = 30,
    NAVIDAD = 40,
    REBAJAS = 50

}
```

La idea es realizar una función que según la estación del año cambie automáticamente, lo mismo con los descuentos. No dio tiempo a implementarla.

El resto de la cabecera es un navbar de Bootstrap con sus links a las diferentes secciones de la página:

```html
<div class="navbar-nav">
    <a [routerLink]="['']" [routerLinkActive]="'active'" [routerLinkActiveOptions]="{exact:true}"
       class="nav-link">Inicio</a>
    <a [routerLink]="['nosotros']" [routerLinkActive]="'active'"
       [routerLinkActiveOptions]="{exact:true}" class="nav-link">Nosotros</a>
    <a [routerLink]="['contacto']" [routerLinkActive]="'active'"
       [routerLinkActiveOptions]="{exact:true}" class="nav-link">Contacto</a>
</div>
```

CARRUSEL

El carrusel de la pantalla de inicio está compuesto de la estructura base y un submódulo que contiene las imágenes del carrusel.

El módulo base tiene un servicio que conecta con un EndPoint ofrecido por Mockoon, también tiene una clase y una interfaz:

```typescript
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { ICarrusel } from './carrusel.interface';

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

  constructor(private http: HttpClient) { }

  public obtenerImagenes(): Observable<ICarrusel[]> {
    const urlEndPoint = "http://localhost:3002/carrusel";
    return this.http.get<ICarrusel[]>(urlEndPoint);
  }
}
```

```typescript
export class Carrusel {
    private id: number;
    private titulo: string;
    private subtitulo: string;
    private img: string;

```

```typescript
export interface ICarrusel {
    id: number;
    titulo: string;
    subtitulo: string;
    img: string;
}
```

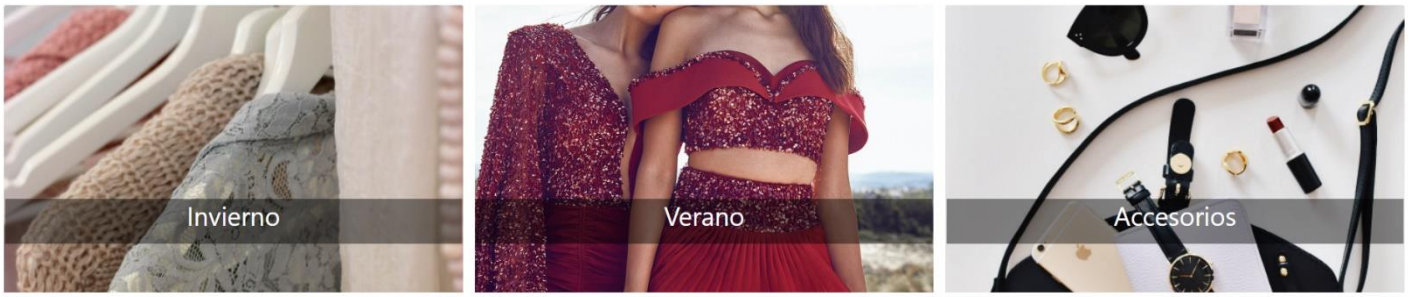
El submódulo ítems consta de la siguiente función que sirve para recuperar los ítems en Mockoon:

```typescript
private obtenerItems() {
  this.itemService.obtenerImagenes().subscribe(
    (data) => {
      data.forEach((imagen) => {
        const imagenAIncluir: Carrusel = new Carrusel(imagen.id, imagen.titulo,
        imagen.subtitulo, imagen.img)
        this.imagenes.push(imagenAIncluir);
      })
    },
  )
}
```

El resultado es el siguiente:

BANNER ARTICULOS

Esta módulo es el que le sigue al carrusel y tiene el siguiente aspecto:



Funciona exactamente igual que el módulo del carrusel, recupera la información de un EndPoint, el módulo principal es el contenedor, tiene la clase, interfaz y endpoint y el submodulo tiene un método subscribe y las recupera.

CATEGORÍAS

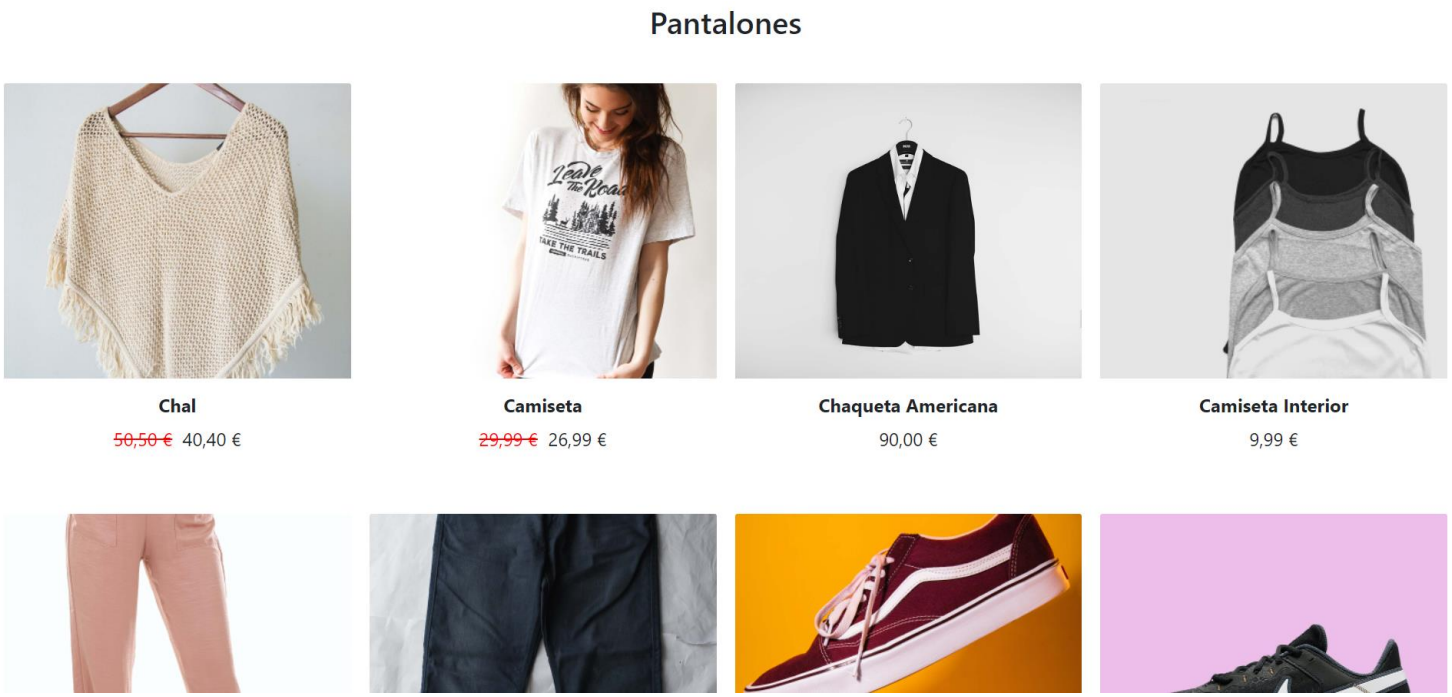
El método de obtención de imágenes es el mismo que el anterior, sólo que esta vez cada imagen llevará a una categoría en concreto

```
const routes: Routes = [
  { path: '', component: InicioComponent },
  { path: 'nosotros', component: NosotrosComponent },
  { path: 'contacto', component: ContactoComponent },
  { path: 'productos/lista/:categoria', component: ListaProductosComponent },
  { path: 'productos/producto/:producto', component: ProductoComponent }
];
```

```
public irACategoria(categoria: String): void {
  this.router.navigate(['productos','lista', categoria])
}
```

La categoría viene dada por el objeto Categorias y su propiedad titulo.

PRODUCTOS



En este caso la obtención de las diferentes categorías es fingida, se obtienen los mismos. Se recuperan las imágenes, precio original y descuento, para el descuento se ha creado una función que lo calcula:

```
public calcularDescuento(precio: number, descuento: number): number {

  const precioFinal = precio - (precio * descuento) / 100;

  return precioFinal;
}
```

Hay un subcomponente para cada ficha de producto, el contenedor obtiene los productos y se los pasa al otro subcomponente:

```
<h1 class="text-center mt-3">{{categoria}}</h1>
<div class="container-fluid">
    <div class="row">
        <app-tarjetas-productos *ngFor="let producto of productos" [producto]="producto" (click)="irAProducto($event)"
            class="col-lg-3 col-md-12 text-center mt-5"></app-tarjetas-productos>
    </div>
</div>
```

También tiene otra función que te lleva al producto en concreto mediante el ID del mismo:

```
public irAProducto(id: number): void {
  this.router.navigate(['productos','producto', id])
}
```

El proceso es muy similar al anterior, se le pasa un objeto productos, pero esta vez sólo debe obtener uno, así que del ID obtenido, se le resta uno al ser un array y obtiene el producto en concreto:

```
<div class="container-fluid mt-3">
    <app-tarjeta-producto *ngIf="id" [productos]="productos[id-1]"></app-tarjeta-producto>
</div>
```

El resultado es el siguiente:

Chal

~~50,50 €~~ 40,40 €

Color

Tallas disponibles

S M L XL

XXL X2L

Se obtiene toda la información y mediante un booleano (siendo TRUE que hay tallas y FALSE que no hay) se genera toda la información. Un ejemplo sin tallas sería el siguiente:

Chaqueta Americana

90,00 €

Color

Tallas disponibles

~~S~~ ~~M~~ ~~L~~ ~~XL~~

~~XXL~~ ~~X2L~~