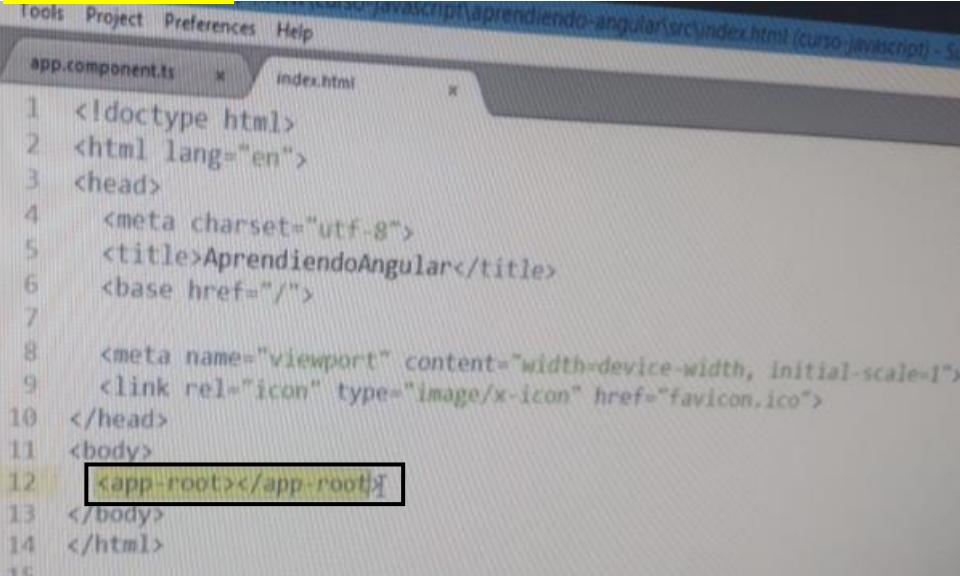
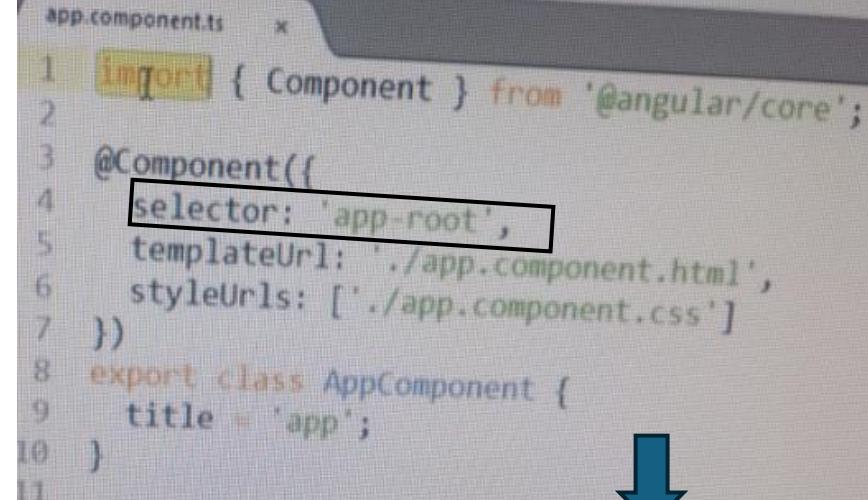


index.html: etiqueta/selector app-root: llama a la app



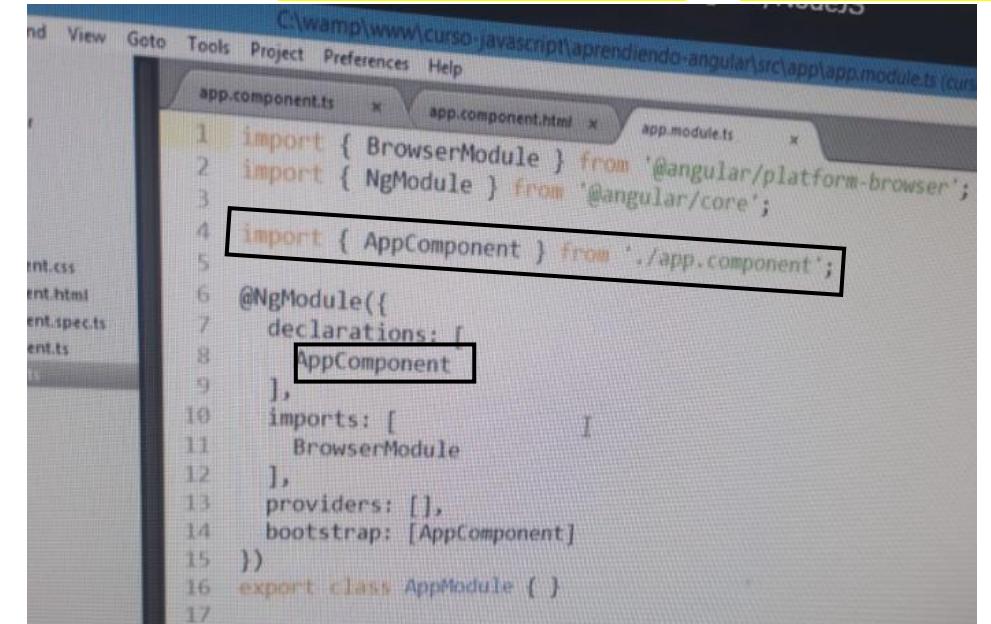
```
1 <!doctype html>
2 <html lang="en">
3 <head>
4   <meta charset="utf-8">
5   <title>AprendiendoAngular</title>
6   <base href="/">
7
8   <meta name="viewport" content="width=device-width, initial-scale=1">
9   <link rel="icon" type="image/x-icon" href="favicon.ico">
10 </head>
11 <body>
12   <app-root></app-root>
13 </body>
14 </html>
```

app.component.ts llama a su vista html y al css



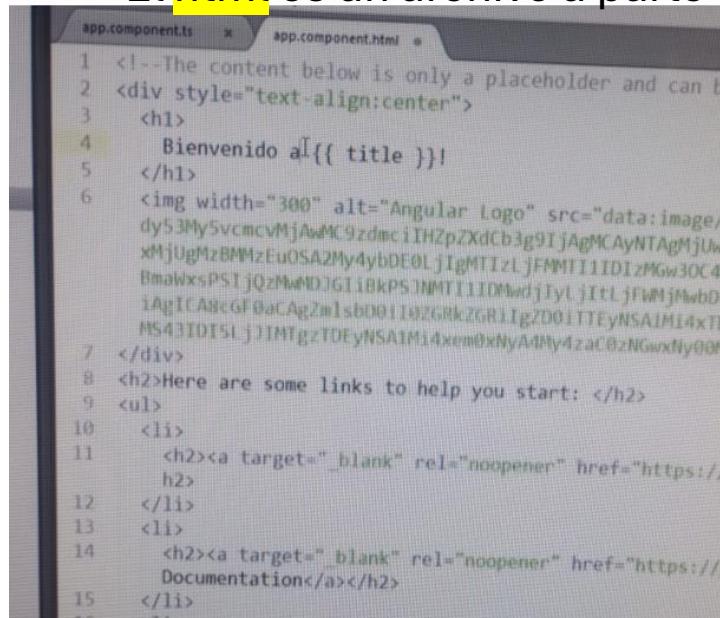
```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   title = 'app';
10 }
```

Se importa la clase AppComponent en app.module.ts



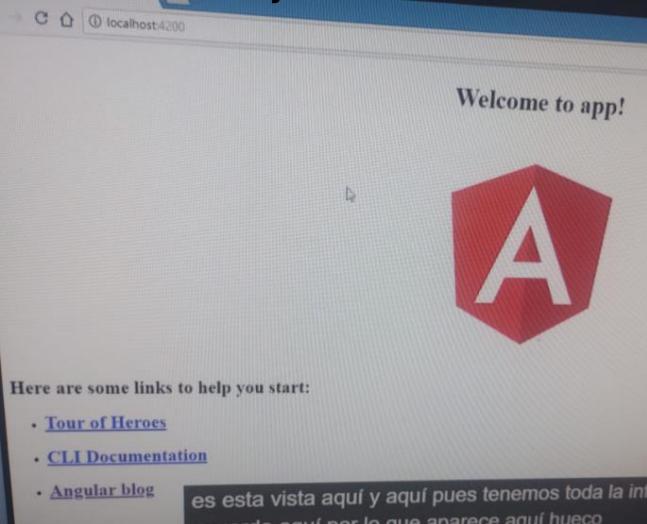
```
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3
4 import { AppComponent } from './app.component';
5
6 @NgModule({
7   declarations: [
8     AppComponent
9   ],
10   imports: [
11     BrowserModule
12   ],
13   providers: [],
14   bootstrap: [AppComponent]
15 })
16 export class AppModule {}
```

El html es un archivo a parte

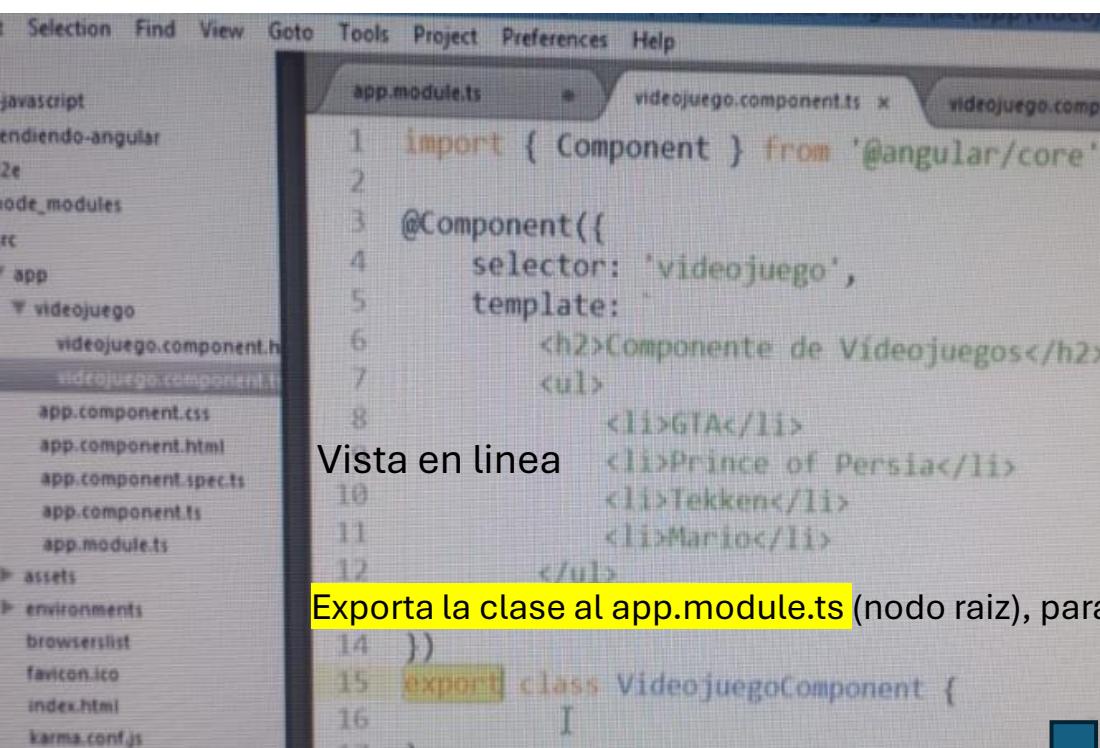


```
1 <!--The content below is only a placeholder and can be removed or modified-->
2 <div style="text-align:center">
3   <h1>
4     Bienvenido a {{ title }}!
5   </h1>
6   
7 </div>
8 <h2>Here are some links to help you start: </h2>
9 <ul>
10   <li>
11     <h2><a target="_blank" rel="noopener" href="https://angular.io">Tour of Heroes</a></h2>
12   </li>
13   <li>
14     <h2><a target="_blank" rel="noopener" href="https://angular.io/docs/ts/latest/guide/CLI-Documentation.html">CLI Documentation</a></h2>
15   </li>
16 </ul>
```

La vista se refleja en localhost:4200



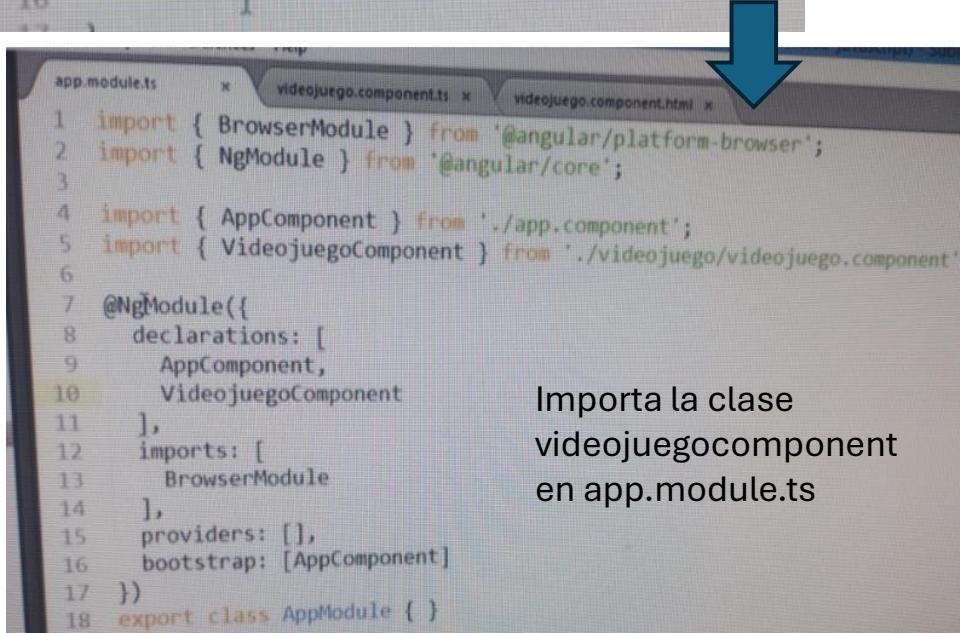
Crea una nueva clase



```
Selection Find View Goto Tools Project Preferences Help
app.module.ts * videojuego.component.ts * videojuego.compon
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'videojuego',
5   template: `
6     <h2>Componente de Videojuegos</h2>
7     <ul>
8       <li>GTA</li>
9       <li>Prince of Persia</li>
10      <li>Tekken</li>
11      <li>Mario</li>
12    </ul>
13  `})
14
15 export class VideojuegoComponent {
```

Vista en linea

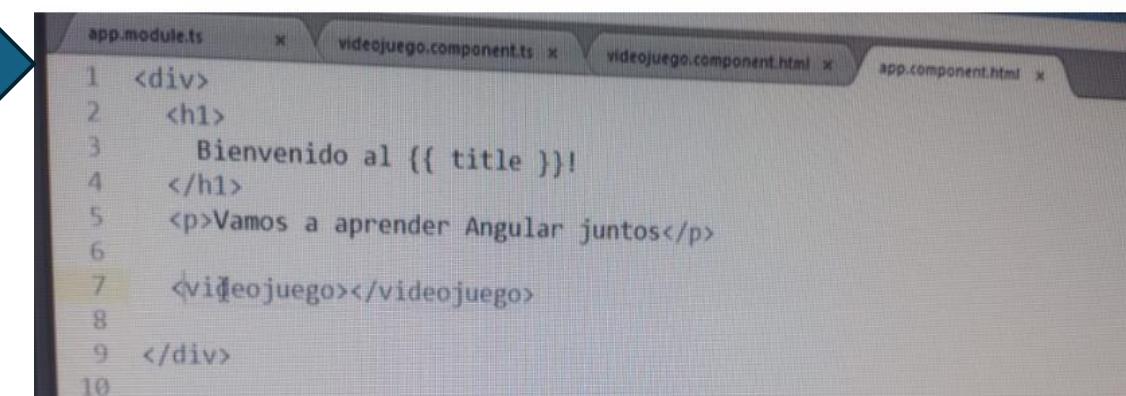
Exporta la clase al app.module.ts (nodo raiz), para poder usarla en otro archivo



```
app.module.ts * videojuego.component.ts * videojuego.component.html *
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3
4 import { AppComponent } from './app.component';
5 import { VideojuegoComponent } from './videojuego/videojuego.component';
6
7 @NgModule({
8   declarations: [
9     AppComponent,
10     VideojuegoComponent
11   ],
12   imports: [
13     BrowserModule
14   ],
15   providers: [],
16   bootstrap: [AppComponent]
17 })
18 export class AppModule { }
```

Importa la clase
videojuegocomponent
en app.module.ts

La incorpora en la vista de la app principal, la vista de la app principal la llama por la etiqueta/selector de app.component.html



```
app.module.ts * videojuego.component.ts * videojuego.component.html * app.component.html *
1 <div>
2   <h1>
3     Bienvenido al {{ title }}!
4   </h1>
5   <p>Vamos a aprender Angular juntos</p>
6
7   <videojuego></videojuego>
8
9 </div>
10
```

Se refleja en la vista de app.component.html



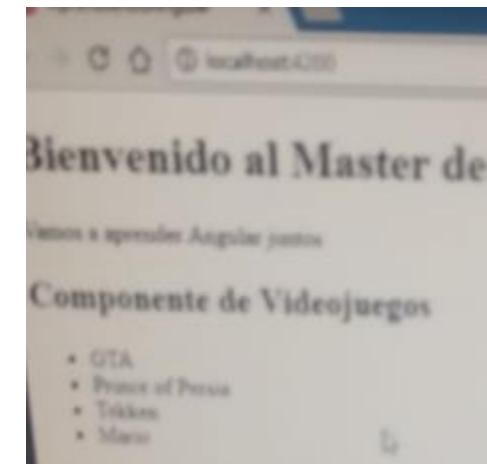
```
<h2>Componente de Vídeojuegos</h2>
<ul>
  <li>GTA</li>
  <li>Prince of Persia</li>
  <li>Tekken</li>
  <li>Mario</li>
</ul>
```

Vista en html

Bindear datos por interpolación

```
import { Component } from '@angular/core';
@Component({
  selector: 'videojuego',
  templateUrl: './videojuego.component.html'
})
export class VideojuegoComponent {
  constructor(){}
  console.log("Se ha cargado el componente: videojuego");
}
```

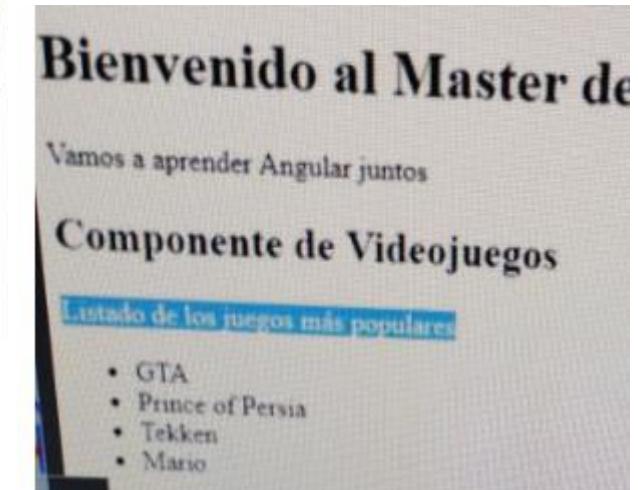
En template indico html



```
import { Component } from '@angular/core';
@Component({
  selector: 'videojuego',
  templateUrl: './videojuego.component.html'
})
export class VideojuegoComponent {
  public titulo: string;
  public listado: string; Propiedades clase
  constructor(){
    this.titulo = "Componente de Videojuegos";
    this.listado = "Listado de los juegos más populares";
  }
  console.log("Se ha cargado el componente: videojuego.component.ts");
}
```

Dar valor a la Propiedad en el constructor

```
<h2>{{titulo}}</h2>
<p>{{listado}}</p>
<ul>
  <li>GTA</li>
  <li>Prince of Persia</li>
  <li>Tekken</li>
  <li>Mario</li>
</ul>
```



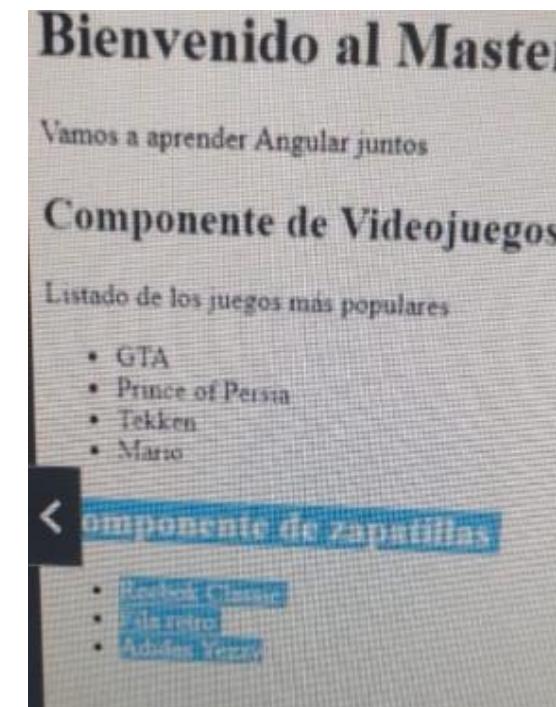
```
app.module.ts
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3
4 import { AppComponent } from './app.component';
5 import { VideojuegoComponent } from './videojuego/videojuego.component';
6 import { ZapatillasComponent } from './zapatillas/zapatillas.component';
7
8 @NgModule({
9   declarations: [
10     AppComponent,
11     VideojuegoComponent,
12     ZapatillasComponent
13   ],
14   imports: [
15     BrowserModule
16   ],
17   providers: [],
18   bootstrap: [AppComponent]
19 })
20 export class AppModule {}
```

Añadimos en app.module.ts los componentes para poder llamarlos en app.component.html, template de app.component.ts, selector <app-root> que es llamado desde index.html.

Desde app.component.html llamamos al componente videojuegos y zapatillas, cada uno con su .ts y su .html

Para generar componentes de forma automática ver comandos angular cli

```
app.component.html
1 <div>
2   <h1>
3     Bienvenido al {{ title }}!
4   </h1>
5   <p>Vamos a aprender Angular juntos</p>
6
7   <videojuego></videojuego> I
8   <zapatillas></zapatillas>
9
10 </div>
```



zapatillas.component.ts

```
zapatillas.component.ts
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'zapatillas',
5   templateUrl: './zapatillas.component.html'
6 })
7 export class ZapatillasComponent {
8   public titulo: string = "Componente de zapatillas";
9 }
```

zapatillas.component.html

```
zapatillas.component.html
1 <h2>{{titulo}}</h2>
2 <ul>
3   <li>Reebok Classic</li>
4   <li>Fila retro</li>
5   <li>Adidas Yezzy</li>
6 </ul>
```

```
1 import { Component, OnInit, DoCheck, OnDestroy } from '@angular/core';
2 @Component({
3   selector: 'videojuego',
4   templateUrl: './videojuego.component.html'
5 })
6 export class VideojuegoComponent implements OnInit, DoCheck, OnDestroy{
7   public titulo: string;
8   public listado: string;
9
10  constructor(){
11    this.titulo = "Componente de Videojuegos";
12    this.listado = "Listado de los juegos más populares";
13
14    console.log("Se ha cargado el componente: videojuego.component.ts");
15  }
16
17  ngOnInit(){
18    console.log("OnInit ejecutado");
19  }
20
21  ngDoCheck(){
22    console.log("DoCheck ejecutado");
23  }
24
25  cambiarTitulo(){
26    this.titulo = "Nuevo título del componente";
27  }
28}
```

Hooks
Ciclos de vida



```
1 export class VideojuegoComponent implements OnInit{
2   public titulo: string;
3   public listado: string;
4
5   constructor(){
6     this.titulo = "Componente de Videojuegos";
7     this.listado = "Listado de los juegos más populares";
8
9     console.log("Se ha cargado el componente: videojuego.component.ts");
10  }
11
12  ngOnInit(){
13    console.log("OnInit ejecutado!!");
14  }
15
16}
```

constructor
Inicio componente



Bienvenido al Master de Java

Vamos a aprender Angular juntos

Componente de Videojuegos

Listado de los juegos más populares

- GTA
- Prince of Persia
- Tekken
- Mario

Elements Console Sources Network

top Filter

Se ha cargado el componente: videojuego.component.ts

Angular is running in the development mode

```
1 this.listado = "Listado de los juegos más populares";
2
3 console.log("Se ha cargado el componente:");
4
5 ngOnInit(){
6   console.log("OnInit ejecutado");
7 }
8
9 ngDoCheck(){
10  console.log("DoCheck ejecutado");
11 }
12
13 cambiarTitulo(){
14  this.titulo = "Nuevo título del componente";
15 }
16
```

Cambios componente

```
1 <h2>{{titulo}}</h2>
2 <p>{{listado}}</p>
3
4 <button (click)="cambiarTitulo()">Cambiar título</button>
5
6 <ul>
7   <li>GTA</li>
8   <li>Prince of Persia</li>
9   <li>Tekken</li>
10  <li>Mario</li>
11 </ul>
12
13 <zapatillas></zapatillas>
```



Bienvenido al Master de Java

Vamos a aprender Angular juntos

Nuevo título del componente

Listado de los juegos más populares

Cambiar título

- GTA
- Prince of Persia

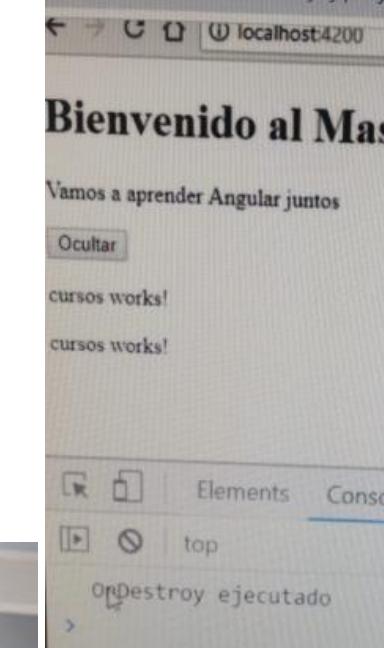
Elements Console Sources Network

top Filter

DoCheck ejecutado

```
app.module.ts * app.component.ts * app.component.html *
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   public title = 'Master de JavaScript y Angular';
10  public mostrar_videojuegos: boolean = true;
11
12  ocultarVideojuegos(){
13    this.mostrar_videojuegos = false;
14  }
15 }
```

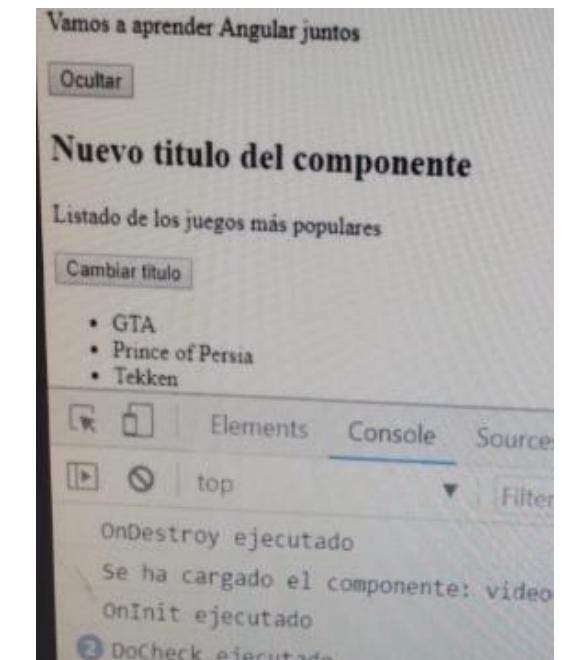
```
app.module.ts * app.component.ts * app.component.html * videojuego.component.ts *
1 <div>
2   <h1>
3     Bienvenido al {{ title }}!
4   </h1>
5   <p>Vamos a aprender Angular juntos</p>
6
7   <button (click)="ocultarVideojuegos()">Ocultar</button>
8
9   <videojuego *ngIf="mostrar_videojuegos"></videojuego>
10  <cursos></cursos>
11  <cursos></cursos>
12
13 </div>
14
```



```
app.module.ts * app.component.ts * app.component.html *
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   public title = 'Master de JavaScript y Angular';
10  public mostrar_videojuegos: boolean = true;
11
12  ocultarVideojuegos(value){
13    this.mostrar_videojuegos = value;
14  }
15 }
```

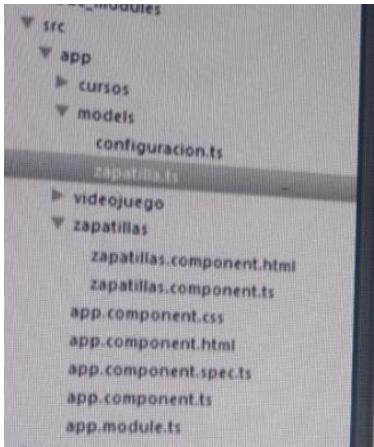
```
app.module.ts * app.component.ts * app.component.html * videojuego.component.ts *
1 <div>
2   <h1>
3     Bienvenido al {{ title }}!
4   </h1>
5   <p>Vamos a aprender Angular juntos</p>
6
7   <button (click)="ocultarVideojuegos(false)" *ngIf="mostrar_videojuegos">Ocultar</button>
8   <button (click)="ocultarVideojuegos(true)" *ngIf="!mostrar_videojuegos">Mostrar</button>
9
10  <videojuego *ngIf="mostrar_videojuegos"></videojuego>
11  <cursos></cursos>
12  <cursos></cursos>
13
14 </div>
```

```
videojuego.component.ts *
1
2   ngOnDestroy(){
3     console.log(" OnDestroy ejecutado");
4   }
5
```



Se ejecuta antes de eliminar instancia/ componente

- <button (click)=="ocultarVideojuegos(false)" *ngIf="mostrar_videojuegos" > Ocultar </button>
- cuando el usuario hace (clic) en el <botón ocultar>, se llama a la función ocultarVideojuegos() pasando el valor (false) como argumento.
- El botón ocultar solo se muestra si mostrar_videojuegos es true
- Cuando la lista de videojuegos este desplegada, habrá un botón de ocultar
- Cuando mostrar_videojuegos es true:El botón "Ocultar" se muestra, y al hacer clic en él, se ejecuta ocultarVideojuegos(false), lo que cambia mostrar_videojuegos a false y esconde los videojuegos
- <button (click)=="ocultarVideojuegos(true)" *ngIf="!mostrar_videojuegos">Mostrar </button>
- cuando el usuario hace (clic) en el <botón mostrar>, se llama a la función ocultarVideojuegos() pasando el valor (true) como argumento.
- El botón mostrar solo se muestra si mostrar_videojuegos es False
- Cuando la lista de videojuegos este escondida, habrá un botón de mostrar
- Cuando mostrar_videojuegos es false:El botón "Mostrar" se muestra, y al hacer clic en él, se ejecuta ocultarVideojuegos(true), lo que cambia mostrar_videojuegos a true, y muestra los videojuegos
- ocultarVideojuegos(value){ this.mostrar_videojuegos = value; }
- función que toma un valor (true o false) como argumento y lo asigna a la variable mostrar_videojuegos
- Leer de izquierda a derecha: muéstrame botón ocultar, cuando mostrar_videojuegos sea true (mostrando contenido) y cuando cliques el boton pasaremos false a la función ocultarvideojuegos para q se oculten contenido
- muéstrame botón mostrar, cuando mostrar_videojuegos sea false (ocultando contenido) y cuando cliques el boton pasaremos true a la función ocultarvideojuegos para q se muestren contenido



```
1 export class Zapatilla{
2   /* ...
3   public nombre: string;
4   public marca: string;
5   public color: string;
6   public precio: number;
7   public stock: boolean;
8
9   constructor(nombre, marca, color, precio, stock){
10    this.nombre = nombre;
11    this.marca = marca;
12    this.color = color;
13    this.precio = precio;
14    this.stock = stock;
15  }
16  */
17  constructor(
18    public nombre: string,
19    public marca: string,
20    public color: string,
21    public precio: number,
22    public stock: boolean
23 ){}
24
25  2 formas iniciar 1 clase
26 }
```

Iniciar clase por modelo en la carpeta models en zapatilla.ts e importar el modelo en zapatillas.component.ts

```
Project Preferences Help
app.module.ts x zapatilla.ts x zapatillas.component.ts x zapatillas.component.html x
1 import { Component, OnInit } from '@angular/core';
2 import { Zapatilla } from '../models/zapatilla';
3
4 @Component({
5   selector: 'zapatillas',
6   templateUrl: './zapatillas.component.html'
7 })
8 export class ZapatillasComponent implements OnInit{
9   public titulo: string = "Componente de zapatillas";
10  public zapatillas: Array<Zapatilla>;
11
12  constructor(){
13    this.zapatillas = [
14      new Zapatilla('Reebok Classic', 'Reebok', 'Blanco', 80, true),
15      new Zapatilla('Nike Runner MD', 'Nike', 'Negras', 60, true),
16      new Zapatilla('Adidas Yezzy', 'Adidas', 'Gris', 180, false)
17    ];
18  }
19
20  ngOnInit(){
21    console.log(this.zapatillas);
22  }
23 }
```

AprendiendoAngular

localhost:4200

- GTA
- Prince of Persia
- Tekken
- Mario

cursos works!

cursos works!

Componente de zapatillas

- Reebok Classic - 80
- Nike Runner MD - 60
- Adidas Yezzy - 180

Elements Console Sources Network

top

(3) [Zapatilla, Zapatilla, Zapatilla]

Angular is running in the development mode.

```
Project Preferences Help
app.module.ts x zapatilla.ts x zapatillas.component.ts x zapatillas.component.html x
1 <h2>{{titulo}}</h2>
2 <ul>
3   <li *ngFor="let deportiva of zapatillas">
4     {{deportiva.nombre}} - <strong>{{deportiva.precio}}</strong>
5   </li>
6 </ul>
```

C:\wamp\www\curso-javascript\aprendiendo-angular\src\app\zapatillas\zapatillas.component.html (curso-javascript) - Sublime Text 2

File Edit Selection Find View Goto Tools Project Preferences Help

FOLDERS

curso-javascript
 aprendiendo-angular
 e2e
 node_modules
 src
 app
 cursos
 models
 configuracion.ts
 zapatilla.ts
 videojuego
 zapatillas
 zapatillas.component.html
 zapatillas.component.ts
 app.component.css
 app.component.html
 app.component.spec.ts
 app.component.ts
 app.module.ts

```
1 <h2>{{titulo}}</h2>
2 <ul>
3   <li *ngFor="let deportiva of zapatillas">
4     {{deportiva.nombre}} - <strong>{{deportiva.precio}}€</strong>
5
6     <span *ngIf="deportiva.precio <= 80"> ¡EN OFERTA!</span>
7
8   </li>
9 </ul>
```



```
1 <h2>{{titulo}}</h2>
2 <ul>
3   <li *ngFor="let deportiva of zapatillas">
4     {{deportiva.nombre}} - <strong>{{deportiva.precio}}€</strong>
5
6     <span *ngIf="deportiva.precio < 80"
7       [style.background]="deportiva.precio < 80 ? 'green' : 'transparent'"
8       [style.color]="deportiva.precio < 80 ? 'white' : 'black'">
9       ¡EN OFERTA!
10      </span>
11    </li>
12 </ul>
```

Bienvenido al Master en JavaScript y Angular!

Aprendiendo Angular con Victor Robles - [victorroblesweb](#)



Componente de zapatillas

- Nike Airmax - 40€ ¡EN OFERTA!
- Reebok Classic - 80€
- Nike Runner MD - 60€ ¡EN OFERTA!
- Adidas Yezzy - 180€

foreach

```
7  })
8 export class ZapatillasComponent implements OnInit{
9   public titulo: string = "Componente de zapatillas";
10  public zapatillas: Array<Zapatilla>;    I
11  public marcas: Array<string>
12

constructor() {
  this.marcas = new Array();
  this.zapatillas = [
    new Zapatilla('Nike Airmax', 'Nike', 'Rojas', 40, true),
    new Zapatilla('Reebok Classic', 'Reebok', 'Blanco', 80, true),
    new Zapatilla('Nike Runner MD', 'Nike', 'Negras', 60, true),
    new Zapatilla('Adidas Yezzy', 'Adidas', 'Gris', 180, false)
  ];
}

ngOnInit(){
  console.log(this.zapatillas);

  this.getMarcas();
}

getMarcas(){
  this.zapatillas.forEach((zapatilla, index) =>{
    this.marcas.push(zapatilla.marca);
    console.log(index);
  });

  console.log(this.marcas);
}
}
```

This.marcas
Acceder propiedad
componente

*2 formas alternativas
public marcas: String[];
public marcas: Array<string>



Bienvenido al Master en JavaScript y Angular!

Aprendiendo Angular con Victor Robles - victorroblesweb

Componente de zapatillas

Elements Console Sources Network Performance Memory Application Security Audits

top Filter Default levels Group similar

```
▶ (4) [Zapatilla, Zapatilla, Zapatilla, Zapatilla]
0
1
2
3
▶ (4) ["Nike", "Reebok", "Nike", "Adidas"]
Angular is running in the development mode. Call enableProdMode() to enable the production mode.
```

▼ (4) [Zapatilla, Zapatilla, Zapatilla, Zapatilla] ⓘ
 ▶ 0: Zapatilla {nombre: "Nike Airmax", marca: "Nike", color: "Rojas", precio: 40, stock: true}
 ▶ 1: Zapatilla {nombre: "Reebok Classic", marca: "Reebok", color: "Blanco", precio: 80, stock: true}
 ▶ 2: Zapatilla {nombre: "Nike Runner MD", marca: "Nike", color: "Negras", precio: 60, stock: true}
 ▶ 3: Zapatilla {nombre: "Adidas Yezzy", marca: "Adidas", color: "Gris", precio: 180, stock: false}
 length: 4
 __proto__: Array(0)

▼ (4) ["Nike", "Reebok", "Nike", "Adidas"] ⓘ
 0: "Nike"
 1: "Reebok"
 2: "Nike"
 3: "Adidas"

```
getMarcas(){
    this.zapatillas.forEach((zapatilla, index) =>{
        if(this.marcas.indexOf(zapatilla.marca) < 0){
            this.marcas.push(zapatilla.marca);
        }
    });
    console.log(this.marcas);
}
```



En el caso q el index sea menor a 0, es decir, q no existe el elemento/marca, o no esté, añadir al array, si está repetido no añadir

Bienvenido al Master en JavaScript y Angular!

Aprendiendo Angular con Víctor Robles - victorroblesweb

Componente de zapatillas

- Nike Airmax - 40€ **[EN OFERTA]**
- Reebok Classic - 80€
- Nike Runner MD - 60€ **[EN OFERTA]**
- Adidas Yezzy - 180€

The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. The output window displays the following results:

```
> (4) [Zapatilla, Zapatilla, Zapatilla, Zapatilla]
< (3) ["Nike", "Reebok", "Adidas"]
  0: "Nike"
  1: "Reebok"
  2: "Adidas"
  length: 3
  > __proto__: Array(0)
```

Angular is running in the development mode. Call enableProdMode() to enable the production mode.

```
app.module.ts x zapatilla.ts x zapatillas.component.ts x zapatillas.component.html * app.component.html x
1 <h2>{{titulo}}</h2>
2 <p>Compra las zapatillas de las mejores marcas</p>
3 <ul>
4   <li *ngFor="let marca of marcas">{{marca}}</li>
5 </ul>
6
7 <p>Las deportivas que tenemos disponibles son:</p>
8 <ul>
9   <li *ngFor="let deportiva of zapatillas">
10    <div *ngIf="deportiva.stock">
11      {{deportiva.nombre}} - <strong>{{deportiva.precio}}€ </strong>
12
13      <span *ngIf="deportiva.precio < 80"
14        [style.background]="deportiva.precio < 80 ? 'green' : 'transparent'"
15        [style.color]="deportiva.precio < 80 ? 'white' : 'black'"
16        >¡EN OFERTA!</span>
17    </div>
18
19   </li>
20 </ul>
```

Ngstyle regla estilo



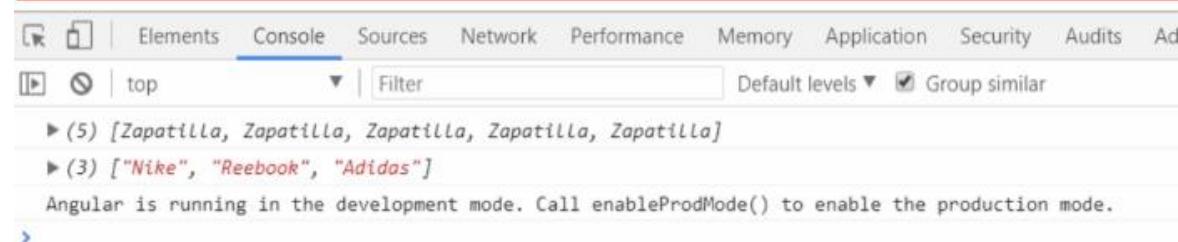
Componente de zapatillas

Compra las zapatillas de las mejores marcas

- Nike
- Reebok
- Adidas

Las deportivas que tenemos disponibles son:

- Nike Airmax - 40€ **[EN OFERTA]**
- Reebok Classic - 80€
- Reebok Spartan - 180€
- Nike Runner MD - 60€ **[EN OFERTA]**
-



```
app.module.ts x zapatilla.ts x zapatillas.component.ts x zapatillas.component.html * app.component.html x
1 <h2>{{titulo}}</h2>
2 <p>Compra las zapatillas de las mejores marcas</p>
3 <ul>
4   <li *ngFor="let marca of marcas">{{marca}}</li>
5 </ul>
6
7 <p>Las deportivas que tenemos disponibles son:</p>
8 <ul>
9   <li *ngFor="let deportiva of zapatillas">
10    <span [ngStyle]="${
11      'text-decoration': !deportiva.stock ? 'line-through' : 'none'
12    }">{{deportiva.nombre}} - <strong>{{deportiva.precio}}€ </strong></span>
13
14    <span *ngIf="deportiva.precio < 80"
15      [style.background]="deportiva.precio < 80 ? 'green' : 'transparent'"
16      [style.color]="deportiva.precio < 80 ? 'white' : 'black'"
17      >¡EN OFERTA!</span>
18
19   </li>
20 </ul>
```

ngStyle: si deportiva no tiene !stock ?, le ponemos el tachado, sino : nada



Componente de zapatillas

Compra las zapatillas de las mejores marcas

- Nike
- Reebok
- Adidas

Las deportivas que tenemos disponibles son:

- Nike Airmax - 40€ **[EN OFERTA]**
- Reebok Classic - 80€
- Reebok Spartan - 180€
- Nike Runner MD - 60€ **[EN OFERTA]**
- Adidas Yeezy - 180€

app.module.ts zapatilla.ts zapatillas.component.ts zapatillas.component.html app.component.html

```

1 <h2>{{titulo}}</h2>
2 <p>Compra las zapatillas de las mejores marcas</p>
3 <ul>
4   <li *ngFor="let marca of marcas; let indice = index">{{indice + ' ' + marca}}</li>
5 </ul>

```

app.module.ts zapatilla.ts zapatillas.component.ts zapatillas.component.html app.component.html

```

4 @Component({
5   selector: 'zapatillas',
6   templateUrl: './zapatillas.component.html'
7 })
8 export class ZapatillasComponent implements OnInit{
9   public titulo: string = "Componente de zapatillas";
10  public zapatillas: Array<Zapatilla>;
11  public marcas: String[];
12  public color: string;
13
14  constructor(){
15    this.color = 'yellow';
16    this.marcas = new Array();
17    this.zapatillas = [
18      new Zapatilla('Nike Airmax', 'Nike', 'Rojas', 40, true),
19      new Zapatilla('Reebok Classic', 'Reebok', 'Blanco', 80, true),
20      new Zapatilla('Reebok Spartan', 'Reebok', 'Negra', 180, true),
21      new Zapatilla('Nike Runner MD', 'Nike', 'Negras', 60, true),
22      new Zapatilla('Adidas Yezzy', 'Adidas', 'Gris', 180, false)
23    ];
24 }

```

<p>El color de la mayoria de nuestras zapatillas es:</p>

<ul [ngSwitch]="color">

<li *ngSwitchCase="'yellow'">

 El color predominantes es el amarillo.

<p>El color de la mayoria de nuestras zapatillas es:</p>

<ul [ngSwitch]="color">

<li *ngSwitchCase="'yellow'">

 El color predominantes es el amarillo.

Componente de zapatillas

Compra las zapatillas de las mejores marcas

- 0 Nike
- 1 Reebok
- 2 Adidas

Project Preferences Help

app.module.ts zapatilla.ts zapatillas.component.ts zapatillas.component.html app.component.html

```

19   [style.color]="deportiva.precio < 80 ? 'white' : 'black'"
20   >jEN OFERTA!</span>
21
22   </li>
23 </ul>
24
25 <p>El color de la mayoria de nuestras zapatillas es:</p>
26 <ul [ngSwitch]="color">
27   <li *ngSwitchCase="'yellow'">
28     El color predominantes es el <span [ngStyle]={"background": color}>amarillo</span>
29     .
30   </li>
31   <li *ngSwitchCase="'red'">
32     El color predominantes es el <span [ngStyle]={"background": color}>rojo</span>.
33   </li>
34   <li *ngSwitchCase="'blue'">
35     El color predominantes es el <span [ngStyle]={"background": color}>azul</span>.
36   </li>
37   <li *ngSwitchCase="'orange'">
38     El color predominantes es el <span [ngStyle]={"background": color}>naranja</span>.
39 </ul>

```

El color de la mayoria de nuestras zapatillas es:

- El color predominantes es el amarillo.

Diferencia aplicar estilos desde el [ngStyle], el *ngIf y desde el

Directiva ng model para modificar una propiedad de un modelo

```
app.module.ts      x  zapatilla.ts      x  zapatillas.component.ts  ●  zapatillas.compon
1 import { Component, OnInit } from '@angular/core';
2 import { Zapatilla } from '../models/zapatilla';
3
4 @Component({
5   selector: 'zapatillas',
6   templateUrl: './zapatillas.component.html'
7 })
8 export class ZapatillasComponent implements OnInit{
9   public titulo: string = "Componente de zapatillas";
10  public zapatillas: Array<Zapatilla>;
11  public marcas: String[];
12  public color: string;
13  public mi_marca: string;
```

```
app.module.ts      x  zapatilla.ts      x  zapatillas.component.ts  x  zapatillas.component.html  x
1 <h2>{{titulo}}</h2>
2
3 <p>Añadir marca</p>
4 <p>
5 <input type="text" [(ngModel)]="mi_marca" />
6 </p>
7 <p>{{mi_marca}}</p>
```



```
app.module.ts      x  zapatilla.ts      x  zapatillas.component.ts  x  zapatillas.component.html  x  app.component.html
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3 import { FormsModule } from '@angular/forms';
4
5 import { AppComponent } from './app.component';
6 import { VideojuegoComponent } from './videojuego/videojuego.component';
7 import { ZapatillasComponent } from './zapatillas/zapatillas.component';
8 import { CursosComponent } from './cursos/cursos.component';
9
10 @NgModule({
11   declarations: [
12     AppComponent,
13     VideojuegoComponent,
14     ZapatillasComponent,
15     CursosComponent
16   ],
17   imports: [
18     BrowserModule,
19     FormsModule
20   ],
21   providers: [],
22   bootstrap: [AppComponent]
23 })
24 export class AppModule { }
```

La directiva **ngModel** en Angular es una herramienta clave para vincular datos entre el modelo de la aplicación y la vista, especialmente en formularios. Es parte del módulo **FormsModule** y se utiliza para implementar la vinculación bidireccional de datos, permitiendo que los cambios en la vista se reflejen en el modelo y viceversa.

Usos de ngModel:

- **Vinculación bidireccional de datos:** Permite sincronizar el estado de un campo de entrada (como un `<input>`, `<select>`, `<textarea>`) con una variable del componente. Esto significa que cualquier cambio en la vista se refleja automáticamente en la variable del componente, y cualquier cambio en la variable se refleja en la vista.
- **Validación de formularios:** `ngModel` puede usarse en conjunto con directivas de validación de Angular para manejar la validación de formularios y mostrar mensajes de error basados en las reglas establecidas.
- **Uso en formularios reactivos y plantillas:** Aunque `ngModel` es más común en formularios basados en plantillas, también puede ser combinado con la arquitectura de formularios reactivos, aunque no es lo más recomendable.

Bienvenido al Master en JavaScript y Angular!

Aprendiendo Angular con Víctor Robles - victorroblesweb

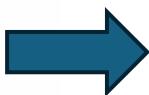
Componente de zapatillas

Añadir marca

```

app.module.ts      x  zapatilla.ts      x  zapatillas.component.ts      x  zapatillas.component.html      x  app.component.html
1 import { Component, OnInit } from '@angular/core';
2 import { Zapatilla } from '../models/zapatilla';
3
4 @Component({
5   selector: 'zapatillas',
6   templateUrl: './zapatillas.component.html'
7 })
8 export class ZapatillasComponent implements OnInit{
9   public titulo: string = "Componente de zapatillas";
10  public zapatillas: Array<Zapatilla>;
11  public marcas: String[];
12  public color: string;
13  public mi_marca: string;
14
15  constructor(){
16    this.mi_marca = "Fila";
17    this.color = 'blue';
18    this.marcas = new Array();
19    this.zapatillas = [
20      new Zapatilla('Nike Airmax', 'Nike', 'Rojas', 40, true),
21      new Zapatilla('Reebok Classic', 'Reebok', 'Blanco', 80, true),
22      new Zapatilla('Reebok Spartan', 'Reebok', 'Negra', 180, true),
23      new Zapatilla('Nike Runner MD', 'Nike', 'Negras', 60, true),
24      new Zapatilla('Adidas Yezzy', 'Adidas', 'Gris', 180, false)
25    ];
26  }
27
28  getMarca(){
29    alert(this.mi_marca);
30  }
31
32
33
34
35
36
37
38
39
39

```



Bienvenido al Master en Java

Aprendiendo Angular con Victor Robles - victorroblesweb

Componente de zapatillas

Añadir marca

Fila

```

app.module.ts      x  zapatilla.ts      x  zapatillas.component.ts      x  zapatillas.component.html
1 <h2>{{titulo}}</h2>
2
3 <p>Añadir marca</p>
4 <p>
5   <input type="text" [(ngModel)]="mi_marca" />
6   <button (click)="getMarca()">Mostrar marca</button>
7 </p>
8 <p>{{mi_marca}}</p>

```



Bienvenido al Master en JavaS

Aprendiendo Angular con Victor Robles - victorroblesweb

Componente de zapatillas

Añadir marca

Fila



```

1 <h2>{{titulo}}</h2>
2
3 <p>Añadir marca</p>
4 <p>
5 <input type="text" [(ngModel)]="mi_marca" />
6 <button (click)="getMarca()">Mostrar marca</button>
7 <button (click)="addMarca()">Añadir marca</button>
8 </p>
9 <strong>{{mi_marca}}</strong>

```

zapatillas.component.ts •

```

34     getMarcas(){
35         this.zapatillas.forEach((zapatilla, index) =>{
36
37             if(this.marcas.indexOf(zapatilla.marca) < 0){
38                 this.marcas.push(zapatilla.marca);
39             }
40
41         });
42
43         console.log(this.marcas);
44     }
45
46     getMarca(){
47         alert(this.mi_marca);
48     }
49
50     addMarca(){
51         this.marcas.push(this.mi_marca);
52     }
53 }

```



Bienvenido al Master en JavaScript y Angular!

Aprendiendo Angular con Víctor Robles - victorroblesweb

Componente de zapatillas

Añadir marca

Fila	Mostrar marca	Añadir marca
------	---------------	--------------

Fila

Compra las zapatillas de las mejores marcas

- 0 - Nike
- 1 - Reebok
- 2 - Adidas
- 3 - Fila

zapatillas.component.html •

```
<p>El color de la mayoria de nuestras zapatillas es:</p>
<input type="text" [(ngModel)]="color" />
<ul [ngSwitch]="'color">
  <li *ngSwitchCase="'yellow'">
    El color predominantes es el <span [ngStyle]="{'background': color}">amarillo</span>
  .
  </li>
  <li *ngSwitchCase="'red'">
    El color predominantes es el <span [ngStyle]="{'background': color}">rojo</span>.
  </li>
  <li *ngSwitchCase="'blue'">
    El color predominantes es el <span [ngStyle]="{'background': color}">azul</span>.
  </li>
  <li *ngSwitchCase="'orange'">
    El color predominantes es el <span [ngStyle]="{'background': color}">naranja</span>.
  </li>
</ul>
```



El color de la mayoria de nuestras zapatillas es:

- El color predominantes es el rojo.

zapatillas.component.html •

```
1 <p>Compra las zapatillas de las mejores marcas</p>
2 <ul>
3   <li *ngFor="let marca of marcas; let indice = index">
4     {{indice + ' - ' + marca}}
5     <button (click)="borrarMarca(indice)">Borrar</button>
6   </li>
7 </ul>
```

zapatillas.component.ts •

```
borrarMarca(index){
  delete this.marcas[index];
}
```



```
reccional Compra las zapatillas de las mejores marcas
  • 0 - Reebok Borrar
  • 1 - Adidas Borrar
```

```
borrarMarca(index){
  // delete this.marcas[index];
  this.marcas.splice(index, 1);
}
```

```
<p>Añadir marca</p>
<p>
<input type="text" [(ngModel)]="mi_marca" (blur)="onBlur()"/>
<button (click)="getMarca()">Mostrar marca</button>
<button (click)="addMarca()">Añadir marca</button>
</p>
<strong>{{mi_marca}}</strong>
```

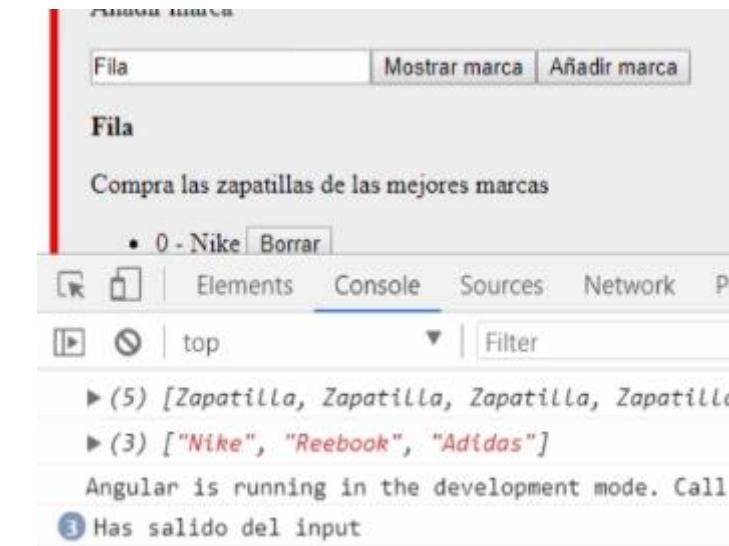
```
onBlur(){
  console.log("Has salido del input");
}
```

evento del DOM blur, que ocurre cuando un elemento (input, textarea) pierde el foco del mouse

```
<p>Añadir marca</p>
<p>
<input type="text" [(ngModel)]="mi_marca" (keyup.enter)="mostrarPalabra()"/>
<button (click)="getMarca()">Mostrar marca</button>
<button (click)="addMarca()">Añadir marca</button>
</p>
<strong>{{mi_marca}}</strong>
```

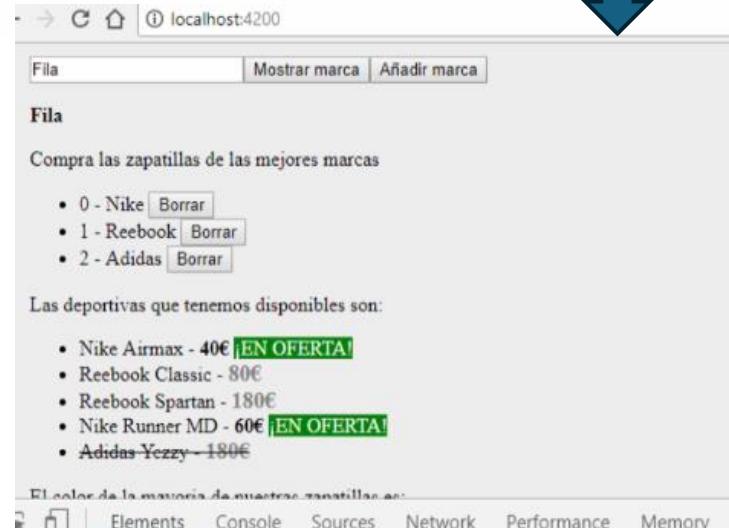
```
mostrarPalabra(){
  alert(this.mi_marca);
}
```

Evento click tecla enter



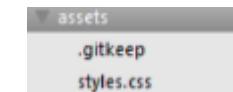
zapatillas.component.html

```
<p>Las deportivas que tenemos disponibles son:</p>
<ul>
  <li *ngFor="let deportiva of zapatillas">
    <span [ngStyle]="{
      'text-decoration': !deportiva.stock ? 'line-through' : 'none'
    }">
      {{deportiva.nombre}} -
      <strong [class.altoPrecio]="deportiva.precio >= 80">
        {{deportiva.precio}}€ </strong>
    </span>
    <span *ngIf="deportiva.precio < 80"
      [style.background]="deportiva.precio < 80 ? 'green' : 'transparent'"
      [style.color]="deportiva.precio < 80 ? 'white' : 'black'">EN OFERTA!</span>
  </li>
</ul>
```



```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>AprendiendoAngular</title>
    <base href="/">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="icon" type="image/x-icon" href="favicon.ico">
    <link type="text/css" href="assets/styles.css" /> == $0
  </head>
  <body>
    <app-root></app-root>
  </body>
</html>
```

ngClass es una directiva que permite agregar o quitar clases CSS a un elemento HTML de manera dinámica. La directiva ngClass se usa para aplicar estilos a los elementos basándose en ciertas condiciones, como propiedades del componente o variables del contexto.



Creamos hoja estilo global

```
1 .altoPrecio{ I
2   color:gray; I
3   font-size: 18px; I
4 }
```



```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>AprendiendoAngular</title>
    <base href="/">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="icon" type="image/x-icon" href="favicon.ico">
    <link type="text/css" href="assets/styles.css" />
  </head>
  <body>
    <app-root></app-root>
  </body>
</html>
```



```
"styles": [
  "src/styles.css",
  "src/assets/styles.css"
],
```



```
USUARIO@NETSW /cygwindo... /cygwindo-angular
$ ng serve
```



```
zapatillas.component.html :  
  
  [class.altoPrecio]="deportiva.precio >= 80"  
  [ngClass]="'fondoRojo', 'subrayado'"  
>  
    {{deportiva.precio}}€  
</strong>  
  
class="fondoRojo subrayado"
```

styles.css

```
1 .altoPrecio{  
2   color:gray;  
3   font-size: 18px;  
4 }  
5  
6 .fondoRojo{  
7   background: red;  
8 }  
9  
10 .subrayado{  
11   text-decoration: underline;  
12 }
```

```
zapatillas.component.html :  
  
  [class.altoPrecio]="deportiva.precio >= 80"  
  [ngClass]={"  
    'fondoRojo': deportiva.precio > 100,  
    'subrayado': deportiva.marca == 'Nike'  
  }"  
>  
    {{deportiva.precio}}€  
</strong>
```



Las deportivas que tenemos disponibles son:

- Nike Airmax - 40€ **[EN OFERTA!]**
- Reebok Classic - 80€
- Reebok Spartan - 180€
- Nike Runner MD - 60€ **[EN OFERTA!]**
- Adidas Yezzy - 180€

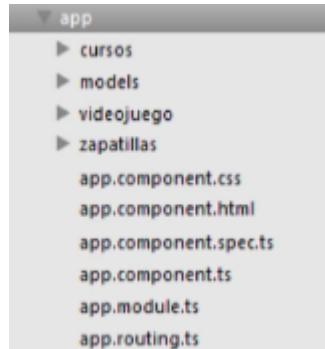
Las deportivas que tenemos disponibles son:

- Nike Airmax - 100€ **[EN OFERTA!]**
- Reebok Classic - 80€
- Reebok Spartan - 180€
- Nike Runner MD - 60€ **[EN OFERTA!]**
- Adidas Yezzy - 180€



Routing angular, diferentes paginas web: **base href="/" es lo necesario para q el router funcione**

```
index.html
1 <!doctype html>
2 <html lang="en">
3 <head>
4   <meta charset="utf-8">
5   <title>AprendiendoAngular</title>
6   <base href="/"> 
7
8   <meta name="viewport" content="width=device-width, initial-scale=1">
9   <link rel="icon" type="image/x-icon" href="favicon.ico">
10 </head>
11 <body>
12   <app-root></app-root>
13 </body>
14 </html>
```



Configurar todo el archivo de rutas angular

```
app.routing.ts
1 // Importar modulos del router de angular
2 import { ModuleWithProviders } from '@angular/core';
3 import { Routes, RouterModule } from '@angular/router';
4
5 // Importar componentes
6 import { HomeComponent } from './home/home.component';
7 import { ZapatillasComponent } from './zapatillas/zapatillas.component';
8 import { VideojuegoComponent } from './videojuego/videojuego.component';
9 import { CursosComponent } from './cursos/cursos.component';
10
11 // Array de rutas
12 const appRoutes: Routes = [
13   {path: '', component: HomeComponent},
14   {path: 'zapatillas', component: ZapatillasComponent},
15   {path: 'videojuego', component: VideojuegoComponent},
16   {path: 'cursos', component: CursosComponent},
17   {path: '**', component: HomeComponent}          ** cuando no encuentra la ruta es
18 ];                                              homecomponent
19
20 // Exportar el modulo del router
21 export const appRoutingProviders: any[] = [];
22 export const routing: ModuleWithProviders = RouterModule.forRoot(appRoutes);
```

- Definición de **'appRoutingProviders'** para almacenar un arreglo de proveedores relacionados con el módulo de enrutamiento de la aplicación.
 - En Angular, los proveedores son responsables de crear y administrar dependencias o servicios que pueden ser inyectados en diferentes partes de la aplicación.
 - Cuando definimos proveedores en un módulo, estos servicios pueden ser utilizados en componentes, directivas, o en otros servicios mediante la Inyección de Dependencias (Dependency Injection, DI).
- La declaración `export const routing: ModuleWithProviders = RouterModule.forRoot(appRoutes);` es una forma de configurar el enrutamiento en una aplicación Angular.
- Esta es una declaración de una constante (const) exportada, llamada **routing**. Al usar **export**, esta constante se puede importar en otros módulos o archivos de la aplicación.
- **ModuleWithProviders** es un tipo genérico que se utiliza para definir un módulo que proporciona servicios o configuraciones específicas en Angular.
 - En el contexto del enrutamiento, **ModuleWithProviders** es una interfaz que se utiliza para indicar que un módulo (como **RouterModule**) puede tener proveedores asociados con él.
 - Esto es importante cuando se configuran módulos que podrían tener dependencias específicas (por ejemplo, servicios o guardias de rutas).
- **RouterModule** es el módulo de enrutamiento principal de Angular. Proporciona las funcionalidades necesarias para gestionar las rutas en una aplicación Angular.
- **forRoot(appRoutes)** es un método estático del **RouterModule** que se utiliza para configurar el enrutamiento de la aplicación con las rutas principales. `forRoot` configura el enrutador a nivel de la aplicación raíz, lo que significa que solo debe llamarse una vez en toda la aplicación (generalmente en el módulo raíz **AppModule**).
- **appRoutes:** Es una constante que contiene un arreglo de rutas (`Routes`) definidas en tu aplicación.
- **router outlet** es una herramienta esencial en Angular para manejar el enrutamiento y la navegación de una aplicación de una manera eficiente y organizada, **router outlet** es el lugar donde Angular inyectará los componentes correspondientes a la ruta activa, para que el componente se muestre.

```

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

import { AppComponent } from './app.component';
import { VideojuegoComponent } from './videojuego/videojuego.component';
import { ZapatillasComponent } from './zapatillas/zapatillas.component';
import { CursosComponent } from './cursos/cursos.component';
import { HomeComponent } from './home/home.component';

@NgModule({
  declarations: [
    AppComponent,
    VideojuegoComponent,
    ZapatillasComponent,
    CursosComponent,
    HomeComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    routing
  ],
  providers: [
    appRoutingProviders
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

Routing Configuracion
Servicio rutas

Componentes: declarations
Modulos: imports
Providers: servicios



localhost:4200/contacto

Bienvenido al Master en Java

Aprendiendo Angular con Victor Robles - victorrobleswell

Componente de zapatillas

Añadir marca

Fila	Mostrar marca	Añadir marca
Fila		

Compra las zapatillas de las mejores marcas

```

app.component.html :
<div [ngStyle]="{
  'background': config.fondo,
  'padding': '20px',
  'border': '5px solid black',
  'border-color': config.color
}">
  <h1>
    Bienvenido al {{ title }}!
  </h1>
  <p>{{descripcion}}</p>
<router-outlet></router-outlet>

```

localhost:4200/cursos

Bienvenido al Master en Java

Aprendiendo Angular con Victor Robles - victorrobleswell

cursos works!

Router-outlet para que se cargue en la url

localhost:4200/zapatillas

Bienvenido al Master en Java

Aprendiendo Angular con Victor Robles - victorrobleswell

Componente de zapatillas

Añadir marca

Fila	Mostrar marca	Añadir marca
Fila		

Compra las zapatillas de las mejores marcas

- 0 - Nike Borrar
- 1 - Reebok Borrar
- 2 - Adidas Borrar

Las deportivas que tenemos disponibles son:

- Nike Airmax - 40€ EN OFERTA!
- Reebok Classic - 80€
- Reebok Spartan - 180€

Menú navegación: para crear un menú de navegación, vamos a crear enlaces para cada una de las páginas de la web con la directiva RouterLink y con para separar los enlaces del menú

```
app.component.html >  
7   <h1>  
8     Bienvenido al {{ title }}!  
9   </h1>  
10  <p>{{descripcion}}</p>  
11  
12  <nav>  
13    <a [routerLink]="/">Home</a>  
14    &nbsp;  
15    <a [routerLink]="/zapatillas">Zapatillas</a>  
16    &nbsp;  
17    <a [routerLink]="/cursos">Cursos</a>  
18    &nbsp;  
19    <a [routerLink]="/videojuego">Videojuego</a>  
20  </nav>
```



Bienvenido al Master en Java

Aprendiendo Angular con Victor Robles - victorroblesweb

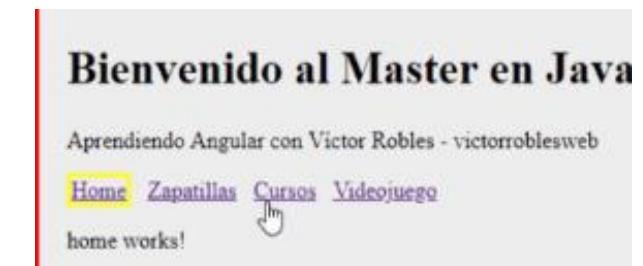
[Home](#) [Zapatillas](#) [Cursos](#) [Videojuego](#)

Para que se marque/resalte la sección del menú que estamos accediendo se puede usar la directiva routerLinkActive

```
app.component.html >  
<nav>  
  <a [routerLink]="/home" [routerLinkActive]="'active'">Home</a>  
  &nbsp;  
  <a [routerLink]="/zapatillas" [routerLinkActive]="'active'">Zapatillas</a>  
  &nbsp;  
  <a [routerLink]="/cursos" [routerLinkActive]="'active'">Cursos</a>  
  &nbsp;  
  <a [routerLink]="/videojuego" [routerLinkActive]="'active'">Videojuego</a>  
</nav>
```



```
styles.css  
.active{  
  border: 3px solid yellow;  
}  
  
app.routing.ts  
// Array de rutas  
const appRoutes: Routes = [  
  {path: '', component: HomeComponent},  
  {path: 'home', component: HomeComponent},  
  {path: 'zapatillas', component: ZapatillasComponent},  
  {path: 'videojuego', component: VideojuegoComponent},  
  {path: 'cursos', component: CursosComponent},  
  {path: '**', component: HomeComponent}  
];
```



Bienvenido al Master en Java

Aprendiendo Angular con Victor Robles - victorroblesweb

[Home](#) [Zapatillas](#) [Cursos](#) [Videojuego](#)

home works!

Parametros por Url

app.routing.ts

```
const appRoutes: Routes = [
  {path: '', component: HomeComponent},
  {path: 'home', component: HomeComponent},
  {path: 'zapatillas', component: ZapatillasComponent},
  {path: 'videojuego', component: VideojuegoComponent},
  {path: 'cursos', component: CursosComponent},
  {path: 'cursos/:nombre', component: CursosComponent},
  {path: '**', component: HomeComponent}
];
{path: 'cursos/:nombre/:followers', component: CursosComponent},
```

- `+param.followers`: parámetro de string a numero
- `ActivatedRoute` representa la ruta actualmente activa/cargada en la aplicación. Proporciona acceso a la información sobre la ruta actual, como los parámetros de ruta, los datos de la ruta, los parámetros de consulta, obtener el ID de un elemento de la URL de la ruta, etc
- `Router` permite la navegación programática en la aplicación. Con él, puedes redirigir a los usuarios a diferentes rutas dentro de la aplicación de Angular
- `_route` es una instancia de `ActivatedRoute`, que es inyectada a través del constructor del componente.
- `params` es un observable proporcionado por `ActivatedRoute` que emite un objeto de parámetros (tipo `Params`) cada vez que la ruta cambia, contiene los parámetros de la ruta, clave-valor de la URL.
- `subscribe` es un método utilizado para suscribirse a los cambios de este observable. Cuando los parámetros de la ruta cambian, la función de devolución de llamada proporcionada a `subscribe` se ejecuta.
- `=>`Es una función de flecha que se ejecuta cada vez que hay un cambio en los parámetros de la ruta.

Recoger parámetros por url

cursos.component.ts

```
1 import { Component, OnInit } from '@angular/core';
2 import { Router, ActivatedRoute, Params } from '@angular/router';
3
4 @Component({
5   selector: 'cursos',
6   templateUrl: './cursos.component.html',
7   styleUrls: ['./cursos.component.css']
8 })
9 export class CursosComponent implements OnInit {
10   public nombre: string; public followers: number;
11   constructor(
12     private _route: ActivatedRoute,
13     private router: Router
14   ) {}
15   ngOnInit() {
16     this._route.params.subscribe((params: Params) => {
17       this.nombre = params.nombre;
18       this.followers = +params.followers;
19       //this.nombre = params['nombre'];
20       console.log(this.nombre);
21     });
22   }
23 }
```

cursos.component.html

```
1 <h1>
2 | Componente de cursos
3 </h1>
4 <h3 *ngIf="nombre">Bienvenido {{nombre}}</h3>
<h3 *ngIf="followers">Tus seguidores son: {{followers}}</h3>
```

Bienvenido al M

Aprendiendo Angular con Victor R

Home Zapatillas Cursos Videos

cursos works!

Elements Console

top

Angular is running in the de

▼ Object
 nombre: "Victor Robles"
 ▶ __proto__: Object

Elements Console Source

top

Angular is running in the develop

Victor Robles

Bienvenido Victor Robles

Tus seguidores son: 30

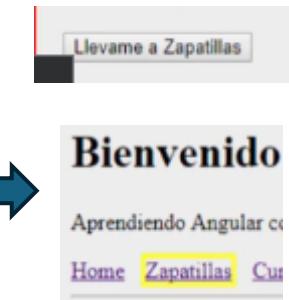
redirigir

cursos.component.html *

```
1 <h1>
2   Componente de cursos
3 </h1>
4
5 <button (click)="redirigir()">Llevame a Zapatillas</button>
6
7 <h3 *ngIf="nombre">Bienvenido {{nombre}}</h3>
8 <h3 *ngIf="followers">Tus seguidores son: {{followers}}</h3>
```

cursos.component.ts *

```
redirigir(){
  this._router.navigate(['/zapatillas']);
}
```



cursos.component.ts *

```
ngOnInit() {
  this._route.params.subscribe((params: Params) => {
    this.nombre = params.nombre;
    this.followers = +params.followers;

    if(this.nombre == 'ninguno'){
      this._router.navigate(['/home']);
    }
  });
}
```

Otro enrutamiento+ validación user

src > app > app-routing.module.ts > ...

```
1 import { registerLocaleData } from '@angular/common';
2 import { NgModule } from '@angular/core';
3 import { RouterModule, Routes } from '@angular/router';
4 import { RegistroApiComponent } from './registro-api/registro-api.component';
5 import { RoversMarteComponent } from './rovers-marte/rovers-marte.component';
6 import { autGuard } from './shared/guards/aut.guard';
7
8 const routes: Routes = [
9   {path: "registro", component:RegistroApiComponent},
10  {path:"rovers", component:RoversMarteComponent, canActivate:[autGuard]},
11  {path:"", component:RegistroApiComponent},
12  {path:"**", component:RegistroApiComponent}
13 ];
14
15 @NgModule({
16   imports: [RouterModule.forRoot(routes)],
17   exports: [RouterModule]
18 })
19 export class AppRoutingModule {}
```

src > app > app.component.html > app-navbar

Go to component

```
1 <app-navbar></app-navbar>
2 <router-outlet></router-outlet>
```

src > app > app.module.ts > ...

```
1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3
4 import { AppRoutingModule } from './app-routing.module';
5 import { AppComponent } from './app.component';
6 import { RegistroApiComponent } from './registro-api/registro-api.component';
7 import { RoversMarteComponent } from './rovers-marte/rovers-marte.component';
8 import { NavbarComponent } from './navbar/navbar.component';
9 import { FormsModule, ReactiveFormsModule } from '@angular/forms';
10 import { HttpClientModule } from '@angular/common/http';
11
12 @NgModule({
13   declarations: [
14     AppComponent,
15     RegistroApiComponent,
16     RoversMarteComponent,
17     NavbarComponent
18   ],
19   imports: [
20     BrowserModule,
21     AppRoutingModule,
22     FormsModule,
23     ReactiveFormsModule,
24     HttpClientModule
25   ],
26   providers: [],
27   bootstrap: [AppComponent]
28 })
29 export class AppModule {}
```

src > app > shared > guards > aut.guard.ts > ...

```
1 import { CanActivateFn } from '@angular/router';
2
3 export const autGuard: CanActivateFn = (route, state) => {
4   //comprobar si en el SS existe el objeto o no y esta validado: puede pasar
5   let escorrecto: boolean= false;
6   const persona = sessionStorage.getItem('registro');
7   if (persona) {
8     try {
9       const personaParsed = JSON.parse(persona);
10      if (personaParsed.user && personaParsed.nombre && personaParsed.mail) {
11        escorrecto=true;
12      }
13    } catch (error) {
14      console.log(error);
15    }
16  }
17  return escorrecto;
18};
```

Ngtemplate: directiva , #crear variables dentro de una vista / plantilla

```
home.component.html *  
  
<h2>Pagina principal</h2>  
  
<div *ngIf="identificado; else noIdentificado">  
  <h3>Estas identificado en la aplicación</h3>  
  <button (click)="unsetIdentificado()">Borrar identificacion</button>  
</div>  
  
<ng-template #noIdentificado>  
  <p>No estas identificado, pulsa este botón para identificarte</p>  
  <button (click)="setIdentificado()">Identificarse</button>  
</ng-template>  
  
home.component.ts  
1 import { Component, OnInit } from '@angular/core';  
2  
3 @Component({  
4   selector: 'app-home',  
5   templateUrl: './home.component.html',  
6   styleUrls: ['./home.component.css']  
7 })  
8 export class HomeComponent implements OnInit {  
9   public identificado: boolean;  
10  
11   constructor(){  
12     this.identificado = false;  
13   }  
14  
15   ngOnInit() {  
16   }  
17  
18   setIdificado(){  
19     this.identificado = true;  
20   }  
21   unsetIdentificado(){  
22     this.identificado = false;  
23   }  
24 }
```

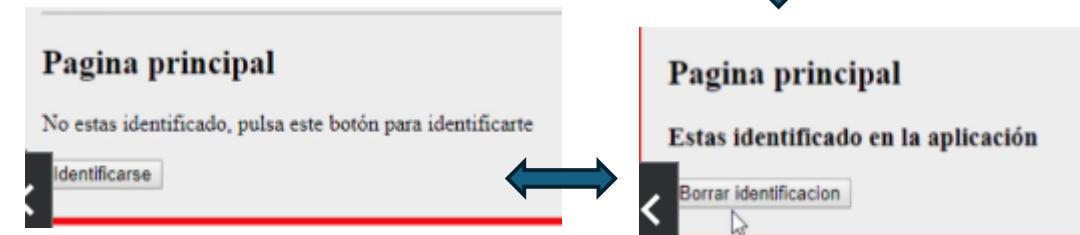


Pagina principal

Estas identificado en la aplicación

Then: tener el código en ng-template separadas, cuando se cumpla el if, es decir identificado es true, then (entonces) carga sIdentificado, sino (else) carga noIdentificado

```
home.component.html *  
1 <h2>Pagina principal</h2>  
2  
3 <div *ngIf="identificado; then siIdentificado else noIdentificado"></div>  
4  
5 <ng-template #siIdentificado>  
6   <h3>Estas identificado en la aplicación</h3>  
7   <button (click)="unsetIdentificado()">Borrar identificacion</button>  
8 </ng-template>  
9  
10 <ng-template #noIdentificado>  
11   <p>No estas identificado, pulsa este botón para identificarte</p>  
12   <button (click)="setIdentificado()">Identificarse</button>  
13 </ng-template>
```



zapatillas.component.ts • Inyectar servicios= inyectar objetos

```

1 import { Component, OnInit } from '@angular/core';
2 import { Zapatilla } from '../models/zapatilla';
3 import { ZapatillaService } from '../services/zapatilla.service';
4
5 @Component({
6   selector: 'zapatillas',
7   templateUrl: './zapatillas.component.html',
8   providers: [ZapatillaService]
9 })
10 export class ZapatillasComponent implements OnInit{
11   public titulo: string = "Componente de zapatillas";
12   public zapatillas: Array<Zapatilla>; Providers/ import
13   public marcas: String[]; Pasar contenido del
14   public color: string;
15   public mi_marca: string;
16
17   constructor(
18     private _zapatillaService: ZapatillaService
19   ){
20     this.mi_marca = "Fila";
21     this.color = 'blue';
22     this.marcas = new Array();
23   }
24
25   ngOnInit(){
26     console.log(this.zapatillas); Entrada** el objeto
27
28     this.getMarcas();
29   }
}

```

zapatilla.service.ts • zapatillas.component.ts

```

1 import { Injectable } from '@angular/core';
2 import { Zapatilla } from '../models/zapatilla';
3
4 @Injectable()
5 export class ZapatillaService{
6   public zapatillas: Array<Zapatilla>;
7
8   constructor(){
9     this.zapatillas = [
10       new Zapatilla('Nike Airmax', 'Nike', 'Rojas', 40, true),
11       new Zapatilla('Reebok Classic', 'Reebok', 'Blanco', 80, true),
12       new Zapatilla('Reebok Spartan', 'Reebok', 'Negra', 180, true),
13       new Zapatilla('Nike Runner MD', 'Nike', 'Negras', 60, true),
14       new Zapatilla('Adidas Yezzy', 'Adidas', 'Gris', 180, false)
15     ];
16   }
17
18   getTexto(){
19     return "Hola Mundo desde un servicio";
20   }
21
22   getZapatillas(): Array<Zapatilla>{
23     return this.zapatillas;
24   }
25 }

```

Una propiedad de una clase es una variable que pertenece a una clase y define un estado o característica de los objetos creados a partir de esa clase, describe o almacena información sobre un objeto de la clase. Por ejemplo, en una clase Persona, propiedades como nombre, edad, y altura describen diferentes aspectos de una persona.

**"inyectar un objeto dentro de la propiedad" significa asignar un objeto a una propiedad de una clase, componente, o función, para que dicha propiedad pueda acceder a las funcionalidades o datos del objeto inyectado. Este proceso se utiliza para hacer que los componentes sean más modulares, reutilizables y fáciles de probar.

- Un **servicio** es una clase que proporciona funcionalidades específicas y compartidas en la aplicación, como lógica de negocio, manejo de datos, o comunicación con APIs externas. Los servicios están diseñados para ser reutilizados en diferentes componentes y otras partes de la aplicación, permitiendo la separación de responsabilidades y el mantenimiento de un código más limpio y organizado.
- Reutilización y Compartición: Los servicios permiten que la lógica común se reutilice en varios componentes, lo cual reduce la duplicación de código y facilita el mantenimiento.
- Inyección de Dependencias (DI): Angular usa un sistema de inyección de dependencias que permite crear y compartir instancias de servicios a lo largo de la aplicación. Cuando un componente necesita un servicio, Angular se encarga de proporcionarlo automáticamente.
- Separación de Lógica de Negocio: Los servicios ayudan a mantener la lógica de negocio separada del código de los componentes, lo que permite que los componentes se enfoquen principalmente en la presentación y la interacción con el usuario.
- Manejo de Datos y Comunicación con APIs: Servicios suelen manejar la lógica para obtener y manipular datos, ya sea interactuando con APIs, gestionando datos locales, o manteniendo el estado compartido de la aplicación.

```
zapatillas.component.ts
```

```
export class ZapatillasComponent implements OnInit{
  public titulo: string = "Componente de zapatillas";
  public zapatillas: Array<Zapatilla>;
  public marcas: String[];
  public color: string;
  public mi_marca: string;

  constructor(
    private _zapatillaService: ZapatillaService
  ){
    this.mi_marca = "Fila";
    this.color = 'blue';
    this.marcas = new Array();
  }
}
```

```
ngOnInit(){
  Llenar variable componente= Llamar al servicio del objeto
  this.zapatillas = this._zapatillaService.getZapatillas();

  this.getMarcas();
}
```



```
ngOnInit(){
  this.zapatillas = this._zapatillaService.getZapatillas();
  alert(this._zapatillaService.getTexto());
  this.getMarcas();
}
```



Un servicio debe proveer de datos y funcionalidad a un componente; para reducir la lógica, refactorizar y hacerlo en archivos separados en lugar de hacer una funcionalidad demasiado grande

Aprendiendo Angular con Víctor Robles - victorroblesweb

Home Zapatillas Cursos Videojuego

Componete de zapatillas

Añadir marca

Fila Mostrar marca Añadir marca

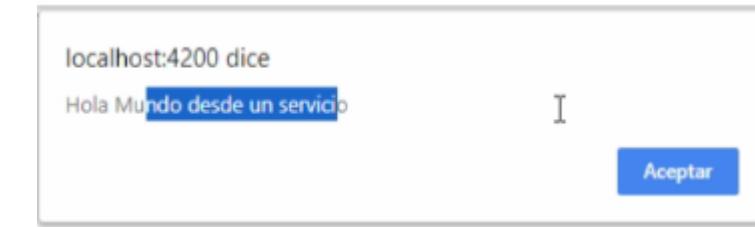
Fila

Compra las zapatillas de las mejores marcas

- 0 - Nike
- 1 - Reebok
- 2 - Adidas

Las deportivas que tenemos disponibles son:

- Nike Airmax - 40€ **EN OFERTA!**
- Reebok Classic - 80€
- Reebok Spartan - 180€
- Nike Runner MD - 60€ **EN OFERTA!**



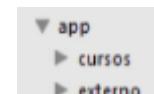
Refactorizar en programación es el proceso de modificar el código de un programa para mejorar su estructura, legibilidad, eficiencia o mantenibilidad, sin alterar su comportamiento o funcionalidad externa. Es una práctica clave en el desarrollo de software que ayuda a mantener el código limpio, comprensible y fácil de modificar o ampliar en el futuro.

Para realizar Peticiones Ajax , httpclient a un servicios externos/api externa

Modulo permite realizar peticiones http , llamado: http client module

Otro que se llama httpmodule: en desuso/obsoleto

Crear 1 nuevo componente/página “externo” para hacer peticiones Ajax y sacar resultados de las peticiones para hacer unas cuantas pruebas-> añadir al app.module y al



```
zapatilla.service.ts x app.module.ts x zapatillas.component.ts x
7 import { VideojuegoComponent } from './videojuego/videojuego.component';
8 import { ZapatillasComponent } from './zapatillas/zapatillas.component';
9 import { CursosComponent } from './cursos/cursos.component';
10 import { HomeComponent } from './home/home.component';
11 import { ExternoComponent } from './externo/externo.component';
12
13 @NgModule({
14   declarations: [
15     AppComponent,
16     VideojuegoComponent,
17     ZapatillasComponent,
18     CursosComponent,
19     HomeComponent,
20     ExternoComponent
21   ],
22   imports: [
23     BrowserModule,
24     FormsModule,
25     routing
26   ],
27   providers: [
28     appRoutingProviders
29   ],
30   bootstrap: [AppComponent]
31 })
32 export class AppModule { }
```

```
zapatilla.service.ts x app.module.ts x zapatillas.component.ts x app.routing.ts x
1 // Importar modulos del router de angular
2 import { ModuleWithProviders } from '@angular/core';
3 import { Routes, RouterModule } from '@angular/router';
4
5 // Importar componentes
6 import { HomeComponent } from './home/home.component';
7 import { ZapatillasComponent } from './zapatillas/zapatillas.component';
8 import { VideojuegoComponent } from './videojuego/videojuego.component';
9 import { CursosComponent } from './cursos/cursos.component';
10 import { ExternoComponent } from './externo/externo.component';
11
12 // Array de rutas
13 const appRoutes: Routes = [
14   {path: '', component: HomeComponent},
15   {path: 'home', component: HomeComponent},
16   {path: 'zapatillas', component: ZapatillasComponent},
17   {path: 'videojuego', component: VideojuegoComponent},
18   {path: 'cursos', component: CursosComponent},
19   {path: 'cursos/:nombre/:followers', component: CursosComponent},
20   {path: 'externo', component: ExternoComponent},
21   {path: '**', component: HomeComponent}
22 ];
23
24 // Exportar el modulo del router
25 export const appRoutingProviders: any[] = [];
26 export const routing: ModuleWithProviders = RouterModule.forRoot(appRoutes);
```

```
app.component.html x
12 <nav>
13   <a [routerLink]="['/home']" [routerLinkActive]="['active']">Home</a>
14   &nbsp;
15   <a [routerLink]="['/zapatillas']" [routerLinkActive]="['active']">Zapatillas</a>
16   &nbsp;
17   <a [routerLink]="['/cursos']" [routerLinkActive]="['active']">Cursos</a>
18   &nbsp;
19   <a [routerLink]="['/videojuego']" [routerLinkActive]="['active']">Videojuego</a>
20   &nbsp;
21   <a [routerLink]="['/externo']" [routerLinkActive]="['active']">Externo</a>
22 </nav>
23
24 <hr>
```

Añadir en la app principal el routerlink: app.component



Ahora hay que crear nuevo servicio para trabajar con httpClient y realizar peticiones Ajax
En app.module hay que importar el modulo para trabajar con httpClient para poder trabajar con las peticiones ajax



```
zapatilla.service.ts x peticiones.service.ts x app.module.ts x
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3 import { FormsModule } from '@angular/forms';
4 import { routing, appRoutingProviders } from './app.routing';
5 import { HttpClientModule } from '@angular/common/http';
6
7 import { AppComponent } from './app.component';
8 import { VideojuegoComponent } from './videojuego/videojuego.component';
9 import { ZapatillasComponent } from './zapatillas/zapatillas.component';
10 import { CursosComponent } from './cursos/cursos.component';
11 import { HomeComponent } from './home/home.component';
12 import { ExternoComponent } from './externo/externo.component';
13
14 @NgModule({
15   declarations: [
16     AppComponent,
17     VideojuegoComponent,
18     ZapatillasComponent,
19     CursosComponent,
20     HomeComponent,
21     ExternoComponent
22   ],
23
24   imports: [
25     BrowserModule,
26     FormsModule,
27     routing,
28     HttpClientModule
29   ],
30   providers: [
31     appRoutingProviders
32   ],
33   bootstrap: [AppComponent]
34 })
35 export class AppModule { }
```

Injectable, HttpClient, y HttpService son elementos clave para la creación de servicios, manejo de inyección de dependencias, y la realización de peticiones HTTP dentro de la aplicación.

- **@Injectable** es un decorador en Angular que se utiliza para marcar una clase como un servicio que puede ser injectado en otros servicios o componentes. Define la clase como una que Angular puede gestionar mediante su sistema de inyección de dependencias, permitiendo que otras clases (como componentes y servicios) reciban instancias de este servicio automáticamente. Se utiliza en la definición de servicios, donde la lógica de negocio y las interacciones con datos se encapsulan.
- **HttpClient** es un servicio proporcionado por Angular que permite realizar peticiones HTTP desde tu aplicación, como GET, POST, PUT, DELETE, etc. Es parte del módulo HttpClientModule. Facilita la comunicación con servidores web y APIs RESTful, manejando la transferencia de datos y el manejo de respuestas y errores. HttpClient se utiliza para interactuar con servicios web y manejar datos en formato JSON, lo cual es esencial para aplicaciones basadas en datos.
- **HttpService** no es una clase nativa de Angular. Son servicios personalizados que utilizan HttpClient para gestionar peticiones HTTP. HttpService en un proyecto Angular, son servicios personalizados creados por un desarrollador para encapsular las interacciones HTTP usando HttpClient.
 - En NestJS, HttpService es un servicio especializado que facilita peticiones HTTP y se utiliza de manera similar a HttpClient en Angular.
- **Encapsular** en programación es un principio fundamental de la programación orientada a objetos (POO) que consiste en restringir el acceso directo a ciertos componentes de un objeto, permitiendo que solo los métodos de ese objeto puedan interactuar con sus datos internos. Este concepto ayuda a proteger los datos, evitar la manipulación indebida y mantener la integridad del estado interno de los objetos. Con diferentes clases, servicios, librerías, private, protected, public.

En `peticiones.service` hay que importar el `HttpClient` y el `HttpHeaders` para poder realizar peticiones Ajax a un servicio /api/url externo y también modificar las cabeceras de esas peticiones

```
app.module.ts x zapatilla.service.ts x peticiones.service.ts *
1 import { Injectable } from '@angular/core';
2 import { HttpClient, HttpHeaders } from '@angular/common/http';
```

Tenemos que importar el observable porque lo que vamos a hacer es recoger la información que nos devuelve el api rest cuando hagamos una petición
`import { Observable } from 'rxjs/Observable';` rxjs es una librería que hemos instalado en el proyecto, si se requiere, también se podría importar modelos u archivos.

De momento usamos el decorador injectable sobre la clase que vamos a exportar `PeticionesService`

```
@Injectable()
export class PeticionesService{
```

Después se crea el constructor de la clase e inyectaremos nuestro servicio/objeto/instancia http por los parámetros de entrada del constructor, para utilizar el servicio de http para realizar peticiones Ajax de manera síncrona con cualquier servidor.

```
constructor(
  public _http: HttpClient
){}
```

- **Síncrono:** En un proceso síncrono, las tareas se ejecutan de manera secuencial, es decir, una tarea debe completarse antes de que comience la siguiente.
- **Asíncrono:** En un proceso asíncrono, las tareas pueden ejecutarse simultáneamente o independientemente unas de otras.

Para realizar peticiones Ajax a servidores backend en la nube: Ejemplo Api de pruebas que hay en internet llamada RedQ en `reqres.in`



The screenshot shows a 'Give it a try' interface. On the left, a sidebar lists various API endpoints with their methods: GET, LIST USERS; GET, SINGLE USER; GET, SINGLE USER NOT FOUND; GET, LIST <RESOURCE>; GET, SINGLE <RESOURCE>; GET, SINGLE <RESOURCE> NOT FOUND; POST, CREATE; PUT, UPDATE; PATCH, UPDATE; DELETE, DELETE. On the right, a 'Request' panel shows a GET request to '/api/users?page=2'. The 'Response' panel shows a successful 200 status code with a JSON response body containing user data across multiple pages.

```
Request
/api/users?page=2
Response
200
{
  "page": 2,
  "per_page": 3,
  "total": 12,
  "total_pages": 4,
  "data": [
    {
      "id": 4,
      "first_name": "Eve",
      "last_name": "Holt",
      "avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/marco_oliveira/100x100.jpg"
    },
    {
      "id": 5,
      "first_name": "Charles",
      "last_name": "Morris",
      "avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/charlesbmorris/100x100.jpg"
    },
    {
      "id": 6,
      "first_name": "Tracey",
      "last_name": "Ramos",
      "avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/traceyramos/100x100.jpg"
    }
  ]
}
```

Peticion a la url q nos devuelva un usuario y nosotros poder mostrar ese usuario

```
peticiones.service.ts * externo.component.ts *
1 import { Injectable } from '@angular/core';
2 import { HttpClient, HttpHeaders } from '@angular/common/http';
3 import { Observable } from 'rxjs/Observable';
4
5 @Injectable()
6 export class PeticionesService{
7   public url: string;
8
9   constructor(
10     public _http: HttpClient
11   ){
12     this.url = "https://reqres.in/";
13   }
14
15   getUser(): Observable<any>{
16     return this._http.get(this.url+'api/users/2');
17   }
18
19 }
```

```
peticiones.service.ts * externo.component.ts *
1 import { Component, OnInit } from '@angular/core';
2 import { PeticionesService } from '../services/peticiones.service';
3
4 @Component({
5   selector: 'app-externo',
6   templateUrl: './externo.component.html',
7   styleUrls: ['./externo.component.css'],
8   providers: [PeticionesService]
9 })
10 export class ExternoComponent implements OnInit {
11
12   constructor(
13     private _peticionesService: PeticionesService
14   ){}
15
16   ngOnInit() {
17     this._peticionesService.getUser().subscribe()
18   }
19
20 }
```

Valor constructor url: la url de la api, valor constante

Para devolver un valor , (ej usuario) usamos un return , con un servicio httpclient para hacer la petición, con su método get a la url constante concatenando con + el 'endpoint' del backend/ a esta api rest . Para obtener ese usuario. Y esto ya es una petición Ajax por http. Esta petición sirve para devolver la información y el método se usa en el componente para conseguir los datos de la api rest . La función getUser(); devuelve un observable De cualquier tipo

Hay que cargar el servicio/proveedor creado (peticiones.service)dentro del componente externo (externo.component.ts) que acabamos de crear e injectarlo como una propiedad dentro del constructor , para poder iniciar un objeto de peticiones.service, de la clase ExternoComponente para poder usar el método de peticiones.service con un objeto de peticiones.service

El método **subscribe** , del modulo Observable, de la librería rxjs, que se utiliza para suscribirse a un dato observable. Los observables son objetos que emiten una secuencia de valores a lo largo del tiempo, y subscribe permite al código recibir esos valores y manejarlos, ya sea en tiempo real o cuando ocurren ciertos eventos. La suscripción al observable permite ejecutar funciones específicas para manejar los datos emitidos, los errores y la finalización del flujo. En frameworks como Angular, RxJS se integra profundamente, y subscribe se utiliza para trabajar con servicios asíncronos, eventos del usuario, y flujos de datos.

Y en el hook/directiva de nGOnInit devuelve un observable que siempre tendrá un método subscribe, para suscribirme y recoger el resultado q me devuelva la petición Ajax.

El método subscribe tiene dos funciones de **callback**: Una función de callback es una función que se pasa como argumento a otra función y que se ejecuta después de que se completa una acción específica o un evento, como la finalización de una operación asíncrona, un clic del usuario, o un cambio de estado.

Una que recoge el resultado y otra que recoge el error en la petición. En caso de error se recoge en un console.log cualquier error que haga.

Y en caso que llegue el resultado , muestro la variable resultado por consola.

```
1 import { Component, OnInit } from '@angular/core';
2 import { PeticionesService } from '../services/peticiones.service'
3
4 @Component({
5   selector: 'app-externo',
6   templateUrl: './externo.component.html',
7   styleUrls: ['./externo.component.css'],
8   providers: [PeticionesService]
9 })
10 export class ExternoComponent implements OnInit {
11
12   constructor(
13     private _peticionesService: PeticionesService
14   ) {}
15
16   ngOnInit() {
17     this._peticionesService.getUser().subscribe(
18       result => {
19         console.log(result);
20       },
21       error => {
22         console.log(<any>error);
23       }
24     );
25   }
26 }
27 }
```



Con la petición Ajax hemos obtenido un objeto json que había guardado en el servidor

The screenshot shows the Network tab in the Chrome DevTools. A request to '/api/users/2' has been selected, highlighted by a blue bar. The Response tab is active, displaying the JSON response: {"data": {"id": 2, "first_name": "Janet", "last_name": "Weaver", "avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/janetweaver/70x100.jpg"}}. The timing panel at the bottom indicates a total duration of 111ms.

Crear una propiedad public user (variable de la clase) y voy a darle el valor que tenga result, con this.user= result.data; porque dentro de data es donde esta el objeto

Al trabajar con observables, para manejar solicitudes HTTP, como en Angular con HttpClient, los datos de la respuesta suelen estar encapsulados dentro de un objeto, y result.data es una convención común para acceder a los datos reales de la respuesta de la información que proviene de apis, backends o servicios que devuelven datos estructurados en json con el valor de result.data que me llega desde el servicio api que he conseguido por Ajax, puedo mostrar esa información en mi vista de mi componente, en su html

```
peticiones.service.ts x externo.component.ts x
4 @Component({
5   selector: 'app-externo',
6   templateUrl: './externo.component.html',
7   styleUrls: ['./externo.component.css'],
8   providers: [PeticionesService]
9 })
10 export class ExteroComponent implements OnInit {
11
12   public user: any;
13
14   constructor(
15     private _peticionesService: PeticionesService
16   ) {}
17
18   ngOnInit() {
19     this._peticionesService.getUser().subscribe(
20       result => {
21         this.user = result.data;
22       },
23       error => {
24         console.log(<any>error);
25       }
26     );
27   }
28
29 }
```

Este div solo mostrara información (*ngIf) si hay un user, si lo hay printara en un h2 el nombre del user y el apellido.

Para printar el avatar (imagen) 

Voy a usar ngModel para cargar el ID del usuario desde la api, mediante una petición a la misma, creando la propiedad /variable userId, y se modifica con ngOnInit, creando un evento al teclear la tecla enter, con keyup.enter



que va a llamar al método cargaUsuario() del externo.componente.ts

```
peticiones.service.ts x externo.component.ts x externo.component.html x
1 <input type="text" [(ngModel)]="userId" (keyup.enter)="cargaUsuario()">
```

- El método cargausuario, tendrá la petición Ajax, con la petición http, + el método getUser, y su parámetro userid, para recoger el userId en el textarea
- Doy valor al userid por defecto en el constructor de la clase externo, q podre cambiar la propiedad con ngmodel
- concateno el userid al final del endpoint de la api, como parámetro para seleccionar los datos del usuario
- Se selecciona y se muestra el user en el html

```
peticiones.service.ts x externo.component.ts x externo.component.html x
10 export class ExteroComponent implements OnInit {
11
12   public user: any;
13   public userId: any;
14
15   constructor(
16     private _peticionesService: PeticionesService
17   ) {
18     this.userId = 1;
19   }
20
21   ngOnInit() {
22     this.cargaUsuario();
23   }
24
25   cargaUsuario(){
26     this._peticionesService.getUser(this.userId).subscribe(
27       result => {
28         this.user = result.data;
29       },
30       error => {
31         console.log(<any>error);
32       }
33     );
34   }
35 }
```

```
peticiones.service.ts x externo.component.ts x externo.component.html x
1 import { Injectable } from '@angular/core';
2 import { HttpClient, HttpHeaders } from '@angular/common/http';
3 import { Observable } from 'rxjs/Observable';
4
5 @Injectable()
6 export class PeticionesService{
7   public url: string;
8
9   constructor(
10     public _http: HttpClient
11   ){
12     this.url = "https://reqres.in/";
13   }
14
15   getUser(userId): Observable<any>{
16     return this._http.get(this.url+'api/users/'+userId);
17   }
18
19 }
```



Aprendiendo Angular con Victor Robles - vitorroblesweb

Home Zapatillas Cursos Videojuego Extero

Network tab showing a timeline from 100 ms to 600 ms. The 'View' dropdown is set to 'Group by frame'. Below the timeline, there's a table with columns 'Name', 'Headers', 'Preview', 'Response', 'Cookies', and 'Timing'. A row for a request labeled '5' is selected, showing details like 'method: GET', 'url: /user/5', 'status: 200 OK', 'content-type: application/json', and 'body: {"data": {"id": 5, "first_name": "Charles", "last_name": "Mone"}, "user": {"id": 5, "first_name": "Charles", "last_name": "Mone", "email": "charles.mone@example.com", "avatar": "https://robohash.org/charles.mone@example.com?size=200x200"}'.

Efecto de carga: que aparezca un cargando mientras carga, si user es false, siempre aparecerá el cargando

```
peticiones.service.ts x externo.component.ts x externo.component.html x
1 <input type="text" [(ngModel)]="userId" (keyup.enter)="cargaUsuario()"/>
2
3 <div *ngIf="!user">
4   Cargando...
5 </div>
6
7 <div *ngIf="user">
8   
9   <h2>{{user.first_name + ' ' + user.last_name}}</h2>
10 </div>

externo.component.ts •

cargaUsuario(){ this.user = false;
```

Pipes / filtros en angular: (tuberías) son una característica que se utiliza para transformar datos directamente en las plantillas de la aplicación. Los pipes permiten modificar la presentación de los datos sin alterar la lógica del componente. Son especialmente útiles para tareas comunes de transformación de datos, como formatear fechas, números, mayúsculas/minúsculas de texto, entre otros. Son una pequeña función que permite procesar información o hacer alguna funcionalidad en la vista.

```
externo.component.ts •

export class ExternoComponent implements OnInit {

  public user: any;
  public userId: any;
  public fecha: any;

  ngOnInit() {
    this.fecha = new Date();
    this.fecha = new Date(2019, 5, 20);
  }
}
```

Aprendiendo Angular con Victor Robles - vitorroblesweb

Home Zapatillas Cursos Videojuego Extero

Thursday, June 20, 2019

externo.component.html •

{{fecha}}

{{fecha | date:'fullDate' }}

{{fecha | date:'dd/MM/yy' }}

Bienvenido al Master en

Aprendiendo Angular con Victor Robles - vitorroblesweb

Home Zapatillas Cursos Videojuego Extero

Sat May 05 2018 18:35:47 GMT+0200 (Hora de verano)

Thu Jun 20 2019 00:00

Moment.js es una biblioteca de JavaScript que se utiliza para manejar, manipular, analizar y mostrar fechas y tiempos de manera sencilla y efectiva.

Filtros transformación de textos

externo.component.html

```
{ { fecha | date:'fullDate' | uppercase } }
```

20/06/19
THURSDAY, JUNE 20, 2019

localhost:4200/externo

externo.component.html

```
{ { 'HOLA MUNDO USANDO PIPES' | lowercase } }
```

localhost:4200/externo

20/06/19
THURSDAY, JUNE 20, 2019
hola mundo usando pipes

Necesita modulo nuevo, añadir en module

app.module.ts

```
import { CalculadoraPipe } from './pipes/calculadora.pipe';
```

```
@NgModule({
```

```
 declarations: [  
  AppComponent,  
  VideojuegoComponent,  
  ZapatillasComponent,  
  CursosComponent,  
  HomeComponent,  
  ExternoComponent,  
  CalculadoraPipe  
,  
,
```

Pipes personalizadas

calculadora.pipe.ts

▼ app
 ▼ pipes

Importar módulos, componentes y @decorador de las pipes, para crear pipes y su interfaz pipetransform

```
import { Pipe, PipeTransform } from '@angular/core';
```

```
@Pipe({  
  name: 'calculadora'  
})
```

PipeTransform es una interfaz de Angular que define la estructura básica que debe seguir cualquier clase que actúe como un pipe. Esta interfaz tiene un método obligatorio llamado transform(), que es donde se implementa la lógica para transformar los datos.

```
export class CalculadoraPipe implements PipeTransform{
```

```
  // dato | calculadora: otroDato
```

```
  // param1                          param2
```

```
  transform(value: any, value_two: any){
```

```
    let operaciones =
```

```
      Suma: ${value+value_two}
```

```
      Resta: ${value-value_two}
```

```
      Multiplicación: ${value*value_two}
```

```
      División: ${value/value_two}
```

```
;
```

```
  return result;
```

```
}
```

externo.component.html

```
{ { 4 | calculadora: 2}}
```

/externo

Suma: 6 - Resta: 2 - Multiplicación: 8 - División: 2

src
 app
 cursos
 externo
 home
 models
 pipes
 services
 videojuego
 zapatillas
 app.component.css
 app.component.html
 app.component.spec.ts
 app.component.ts
 app.module.ts
 app.routing.ts

Instalar rxjs

```
USUARIO@NETSW /cygdrive/c/wamp/www  
$ npm install --save rxjs-compat
```

Formulario de contacto: pagina nueva
Añadir en routing y en app component
Para que este en el navegador,
En base a modelo de usuario

app.routing.ts

```
1 // Importar modulos del router de angular
2 import { NgModuleWithProviders } from '@angular/core';
3 import { Routes, RouterModule } from '@angular/router';
4
5 // Importar componentes
6 import { HomeComponent } from './home/home.component';
7 import { ZapatillasComponent } from './zapatillas/zapatillas.component';
8 import { VideojuegoComponent } from './videojuego/videojuego.component';
9 import { CursosComponent } from './cursos/cursos.component';
10 import { ExternoComponent } from './externo/externo.component';
11 import { ContactoComponent } from './contacto/contacto.component';

// Array de rutas
const appRoutes: Routes = [
  {path: '', component: HomeComponent},
  {path: 'home', component: HomeComponent},
  {path: 'zapatillas', component: ZapatillasComponent},
  {path: 'videojuego', component: VideojuegoComponent},
  {path: 'cursos', component: CursosComponent},
  {path: 'cursos/:nombre/:followers', component: CursosComponent},
  {path: 'externo', component: ExternoComponent},
  {path: 'contacto', component: ContactoComponent},
  {path: '**', component: HomeComponent}
];
// Exportar el modulo del router
export const appRoutingProviders: any[] = [];
export const routing: NgModuleWithProviders = RouterModule.forRoot(appRoutes);
```

app.component.html

```
&nbsp;
<a [routerLink]="/contacto" [routerLinkActive]="'active'>Contacto</a>
```

localhost:4200/contact

```
models
  configuracion.ts
contacto.usuario.ts
zapatilla.ts

contacto.usuario.ts
export class ContactoUsuario{
  constructor(
    public nombre: string,
    public apellidos: string,
    public email: string,
    public mensaje: string
  ){}
}

contacto.component.ts
import { ContactoUsuario } from './models/contacto.usuario';
1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-contacto',
5   templateUrl: './contacto.component.html',
6   styleUrls: ['./contacto.component.css']
7 })
8 export class ContactoComponent implements OnInit {
9
  public usuario: ContactoUsuario;
10
  constructor() {
11   this.usuario = new ContactoUsuario('','','','')
12 }
13
14 ngOnInit() {
15 }
16
17 onSubmit(){
18   console.log("Evento submit lanzado!!");
19 }
20 }
```

Para convertir
Un nombre con
ngsubmit, que

```
<form #formCo
```

Constructor para crear objetos de usuario y modificarlos

```
  contacto.component.html x

<h2>Contacto</h2>

<form>
  <label for="nombre">Nombre</label> <br/>
  <input type="text" name="nombre" /> <br/>

  <label for="apellidos">Apellidos</label> <br/>
  <input type="text" name="apellidos" /> <br/>

  <label for="email">Email</label> <br/>
  <input type="text" name="email" /> <br/>

  <label for="mensaje">Mensaje</label> <br/>
  <textarea name="mensaje"></textarea> <br/>

  <input type="submit" value="Enviar" />

</form>
```

Para convertir este formulario de html-> angular, ponerle
Un nombre con #, ngForm, mas (evento) de submit con
ngsubmit, que al enviar ocurre un método

```
<form #formContacto="ngForm" (ngSubmit)="onSubmit()">
```

Para transformar un campo de formulario de html -> angular, poner nombre a los inputs, y directiva ngmodel: que servirá para la validación de datos y el two data binding de ng model para modificar el objeto de datos que esta en la clase del componente

Two-Way Data Binding (Binding bidireccional): Los datos fluyen en ambas direcciones, lo que permite la sincronización automática entre el modelo y la vista.
ngModel es un ejemplo de binding bidireccional.

```
contacto.component.html x
<h2>Contacto</h2>
<form #formContacto="ngForm" (ngSubmit)="onSubmit()">
  <label for="nombre">Nombre</label> <br/>
  <input type="text" name="nombre" #nombre="ngModel" [(ngModel)]="usuario.nombre"/> <br/>
  <label for="apellidos">Apellidos</label> <br/>
  <input type="text" name="apellidos" #apellidos="ngModel" [(ngModel)]="usuario.apellidos" /> <br/>
  <label for="email">Email</label> <br/>
  <input type="text" name="email" #email="ngModel" [(ngModel)]="usuario.email" /> <br/>
  <label for="mensaje">Mensaje</label> <br/>
  <textarea name="mensaje" #mensaje="ngModel" [(ngModel)]="usuario.mensaje"></textarea> <br/>
  <input type="submit" value="Enviar" />
</form>
```



Aprendiendo Angular con Victor Robles - victorrobles.com

Home Zapatillas Cursos Videojuego Experiencia Contacto

Nombre
Apellido
Email
Mensaje

Enviar

Angular is running in the development mode
Evento submit lanzado!!
ContactoUsuario {nombre: "Antonio", apellidos: "Lopez", email: "antonio@antonio.com", mensaje: "fasdfas", nomrre: "Antonio fasdfas"}
proto : Object

```

1 <h2>Contacto</h2>
2
3 <form #formContacto="ngForm" (ngSubmit)="onSubmit()">
4   <label for="nombre">Nombre</label> <br/>
5   <input type="text" name="nombre" #nombre="ngModel" [(ngModel)]="usuario.nombre"
6     required pattern="[a-zA-Z]+"/>
7   <span *ngIf="nombre.touched && !nombre.valid">
8     El nombre es obligatorio
9   </span>
10  <br/>
11
12  <label for="apellidos">Apellidos</label> <br/>
13  <input type="text" name="apellidos" #apellidos="ngModel" [(ngModel)]="usuario.apellidos"
14    required pattern="[a-zA-Z]+"/>
15  <span *ngIf="apellidos.touched && !apellidos.valid">
16    Los apellidos son obligatorios
17  </span>
18  <br/>
19
20  <label for="email">Email</label> <br/>
21  <input type="email" name="email" #email="ngModel" [(ngModel)]="usuario.email" required
22  />
23  <span *ngIf="email.touched && !email.valid">
24    El email no es válido
25  </span>
26  <br/>
27
28  <label for="mensaje">Mensaje</label> <br/>
29  <textarea name="mensaje" #mensaje="ngModel" [(ngModel)]="usuario.mensaje" required></
30  textarea>
31  <span *ngIf="mensaje.touched && !mensaje.valid">
32    El mensaje es obligatorio
33  </span>
34  <br/>
35
36  <input type="submit" value="Enviar" [disabled]="!formContacto.form.valid"/>
37
38 </form>

```

Validar formularios en angular, poner en cada campo **required**

Validacion con ***ngIf**.

Si quieres que cumpla las reglas de html5 que se quieran añadir hay que añadir un **pattern**

!nombre.valid= no rellenemos datos

nombre.touched= nombre tocado, si yo entro en el textarea, la dejo vacia y salgo, pondrá el nombre es obligatorio

pattern="[^a-zA-Z]" patron de la a – z, un num indefinido de veces

Disabled en el submit, cuando se cumpla una condición ->cuando no sea valido el form contacto

Para vaciar el formulario después de llenarlo

```
<form #formContacto="ngForm" (ngSubmit)="onSubmit(formContacto)">
```

contacto.component.ts

```
onSubmit(form){
  form.reset();
  console.log(this.usuario);
}
```



```
onSubmit(form){
  this.show_data = this.usuario;
  console.log(this.show_data);
  //form.reset();
}
```

Elements Console Sources Network Pe

top Filter

Angular is running in the development mode. Call
>ContactoUsuario {nombre: null, apellidos: null, email: null, mensaje: null, nombre: null}
 ► proto : Object