

En src/styles.css aplicamos estilos globales

The screenshot shows the Visual Studio Code interface with the following details:

- Toolbar:** Archivo, Editar, Selección, Ver, Ir, Ejecutar, Terminal, Ayuda.
- Title Bar:** styles.css - primero - Visual Studio Code [Administrador].
- Left Sidebar (Explorador):** Shows the project structure:
 - EDITORES ABIERTOS: # styles.css (src)
 - PRIMERO:
 - .angular
 - node_modules
 - src
 - app
 - assets
 - environments
 - favicon.ico
 - index.html
 - main.ts
 - polyfills.ts
 - # styles.css (highlighted with a red box)
 - test.ts
 - .browserslistrc
 - .editorconfig
 - .gitignore
 - angular.json
 - karma.conf.js
- Terminal:** powershell +
- Editor Area:** The code editor displays the contents of the styles.css file, which includes global styles for the html and body elements and a bold font weight for dd elements.

```
/* You can add global styles to this file, and also import other style files */
/*
 * {
 *   font-family: Helvetica, Arial, sans-serif;
 *   font-weight: 200;
 * }
html, body {
  background: white;
  margin: 20px;
  color: #3e4144;
  -webkit-font-smoothing: antialiased;
}
dd {
  font-weight: bold;
}
button {
```

The screenshot shows the Visual Studio Code interface with a dark theme. The top menu bar includes Archivo, Editar, Selección, Ver, Ir, Ejecutar, Terminal, and Ayuda. The title bar indicates the file is "styles.css - primero - Visual Studio Code [Administrador]".

The left sidebar contains icons for Explorador, Editores Abiertos, PRIMERO, src, .angular, node_modules, app, assets, environments, favicon.ico, index.html, main.ts, polyfills.ts, test.ts, .browserslistrc, .editorconfig, .gitignore, angular.json, karma.conf.js, and Esquema. The file "styles.css" is listed under "EDITORES ABIERTOS" and is highlighted with a red box.

The main editor area displays the following CSS code:

```
src > # styles.css > html  
button {  
    background-color: black;  
    border-radius: 5px;  
    border: 0px;  
    color: white;  
    cursor: pointer;  
    margin-right: 5px;  
    margin-left: 5px;  
    padding: 5px 10px;  
}  
  
button:hover {  
    background-color: #3e4144;  
}  
  
button:focus{  
    outline: none;  
}
```

The bottom navigation bar includes PROBLEMAS, SALIDA, CONSOLA DE DEPURACIÓN, TERMINAL, and a powershell icon. There are also tabs for "+" and "ESQUEMA".



EXPLORADOR

...

styles.css M X



EDITORES ABIERTOS

X # styles.css src M



PRIMERO

> .angular

> node_modules



src

> app

> assets

> environments

★ favicon.ico

↳ index.html

TS main.ts

TS polyfills.ts

styles.css M

TS test.ts

≡ .browserslistrc

⚙ .editorconfig

❖ .gitignore

{ angular.json

K karma.conf.js

> ESQUEMA

src > # styles.css > ↗ html

```
29 }
30
31 button:hover {
32   background-color: #3e4144;
33 }
34
35 button:focus{
36   outline: none;
37 }
38
39 .p-1 {
40   padding: 1px;
41 }
42
```

PROBLEMAS

SALIDA

CONSOLA DE DEPURACIÓN

TERMINAL

powershell + ▾

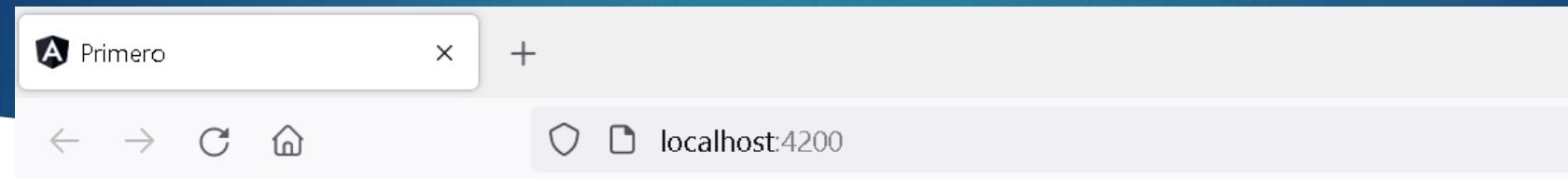
En app.component.ts

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows the project structure with files like `app.module.ts`, `index.html`, `main.ts`, `polyfills.ts`, and `styles.css`. The file `app.component.ts` is selected and highlighted with a red border.
- Editor (Center):** Displays the content of `app.component.ts`. A red box highlights the template URL and the template itself, which contains an `h1` tag and a `ul` list with three `li` items.
- Status Bar (Bottom):** Shows tabs for `PROBLEMAS`, `SALIDA`, `CONSOLA DE DEPURACIÓN`, and `TERMINAL`.

```
src > app > TS app.component.ts > AppComponent
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   template: `
7     <h1>Hola como vamos </h1>
8     <ul>
9       <li>tarea 1</li>
10      <li>tarea 2</li>
11      <li>tarea 3</li>
12    </ul>
13  `,
14   styleUrls: ['./app.component.css']
15 })
16
17
18 export class AppComponent {
19   title = 'primero';
20 }
```

Comprobamos que podemos tener definida el html dentro del ts



Hola como vamos

- tarea 1
- tarea 2
- tarea 3

Creo un ejemplo de contador

Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda app.component.html - primero - Visual Studio Code [Administrador]

TS app.component.ts M X ↗ app.component.html M ... ↗ app.component.html M X

```
src > app > TS app.component.ts > ⚒ AppComponent > 🔑 titulo
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
9
10 export class AppComponent {
11   titulo = 'Contador';
12
13 }
14
15
```

src > app > ↗ app.component.html > ⚒ span

Go to component

```
1 <h1>
2   {{titulo}}
3 </h1>
4
5
6 <span> 0 </span>
7
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

TS app.component.ts M X

<> app.component.html M

...

src > app > TS app.component.ts > AppComponent > numero

```
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
9 export class AppComponent {
10   titulo:string = 'Contador';
11   numero:number = 10;
12
13 }
14
15
```

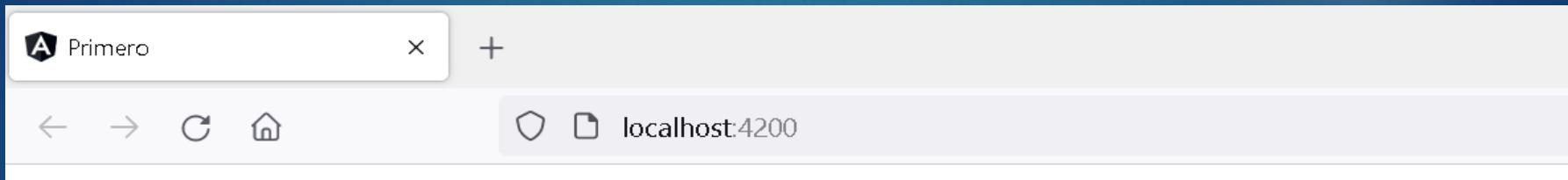
<> app.component.html M X

src > app > <> app.component.html > button

Go to component

```
1 <h1>
2   {{titulo}}
3
4 </h1>
5
6 <button> +1 </button>
7 <span>{{numero}} </span>
8 <button> -1 </button>
```

9



Contador

+1 10 -1

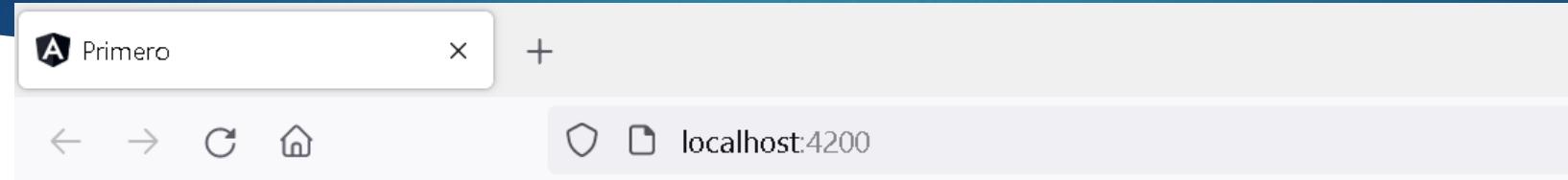
Agregamos un evento al botón

The screenshot shows the Visual Studio Code interface with two tabs open:

- app.component.ts**: The code defines an Angular component named `AppComponent` with a selector of `'app-root'`, a template URL of `'./app.component.html'`, and style URLs of `'./app.component.css'`. It contains properties `titulo` and `numero`, and methods for incrementing and decrementing `numero`.
- app.component.html**: The template contains an `<h1>` element with the binding `{{titulo}}`, a `<button>` element with the attribute `(click)="numero = numero+1"` and the text `+1`, a `` element with the binding `{{numero}}`, and another `<button>` element with the text `-1`.

The `<button>` element in the `app.component.html` file is highlighted with a red box, indicating it is the current selection or being edited.

Nos debería de aumentar en 1 el contador



Contador

+1 11 -1

Así quedaría agregando funcionalidad al botón -1



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** On the left, there are icons for files, folders, and a search bar. A blue circle with the number "3" is visible on the folder icon.
- Editor:** The main area displays the content of `app.component.html`. The file is currently active, indicated by a red border around its tab.
- Code Content:**

```
src > app > app.component.html > ...
    Go to component
1  <h1>
2  |   {{titulo}}
3
4  </h1>
5
6  <button (click)="numero = numero+1"> +1 </button>
7  <span>{{numero}} </span>
8  <button (click)="numero = numero-1"> -1 </button>
9
```
- Top Bar:** The menu bar includes Archivo, Editar, Selección, Ver, Ir, Ejecutar, Terminal, and others. The tab for `app.component.ts` is also visible.

Mejoramos nuestro código

The screenshot shows two open files in Visual Studio Code:

- app.component.ts**:
A TypeScript file containing the definition of the `AppComponent`. It includes the component's selector, template URL, and style URLs. It also defines a `sumar()` method that increments the `numero` property by 1.

```
src > app > app.component.ts > AppComponent > sumar
3   @Component({
4     selector: 'app-root',
5     templateUrl: './app.component.html',
6     styleUrls: ['./app.component.css']
7   })
8
9   export class AppComponent {
10    titulo:string = 'Contador';
11    numero:number = 10;
12
13    sumar(){
14      this.numero +=1;
15    }
16
17  }
18
19
```
- app.component.html**:
An HTML file defining the UI for the `AppComponent`. It contains an `<h1>` element with a binding for the `titulo` property. It also features two buttons: one to increment the `numero` and another to decrement it.

```
src > app > app.component.html > ...
...
1   <h1>
2     {{titulo}}
3
4   </h1>
5
6   <button (click)="sumar()"> +1 </button>
7   <span>{{numero}} </span>
8   <button (click)="numero = numero-1"> -1 </button>
9
```

Annotations:

- A red box highlights the `sumar()` method in `app.component.ts`.
- A red box highlights the increment button in `app.component.html`.

Bottom navigation bar:

- PROBLEMAS
- SALIDA
- CONSOLA DE DEPURACIÓN
- TERMINAL
- node
- +
- ▼
-
-



TS app.component.ts M X

<> app.component.html M

...

```
src > app > TS app.component.ts > AppComponent > restar
  3  @Component({
  4    selector: 'app-root',
  5    templateUrl: './app.component.html',
  6    styleUrls: ['./app.component.css']
  7  })
  8
  9
10 export class AppComponent {
11   titulo:string = 'Contador';
12   numero:number = 10;
13
14   sumar(){
15     this.numero +=1;
16   }
17
18   restar(){
19     this.numero -=1;
20   }
21
22 }
```

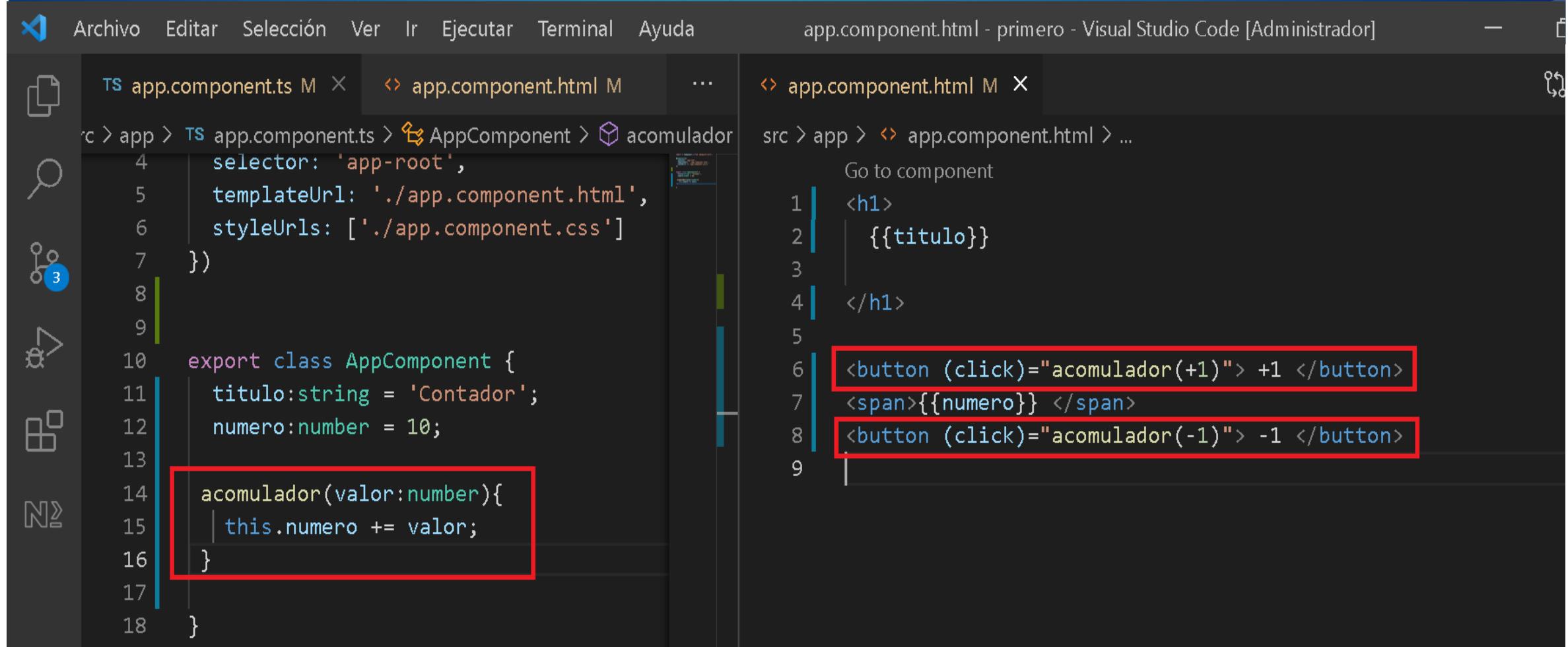
<> app.component.html M X

src > app > <> app.component.html > button

Go to component

```
1 <h1>
2   {{titulo}}
3
4 </h1>
5
6 <button (click)="sumar()"> +1 </button>
7 <span>{{numero}} </span>
8 <button (click)="restar()"> -1 </button>
```

Optimizamos aun mas



The screenshot shows two files open in Visual Studio Code:

- app.component.ts**:
A TypeScript file containing the definition of the `AppComponent`. It includes properties `selector`, `templateUrl`, and `styleUrls`, and a method `acomulador` which increments the `numero` property.

```
src > app > TS app.component.ts > AppComponent > acomulador
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 }
8
9
10 export class AppComponent {
11   titulo:string = 'Contador';
12   numero:number = 10;
13
14   acomulador(valor:number){
15     this.numero += valor;
16   }
17
18 }
```
- app.component.html**:
An HTML template for the `AppComponent`. It contains an `<h1>` element with the `titulo` variable, and two `<button>` elements that call the `acomulador` method with arguments `+1` and `-1` respectively.

```
src > app > app.component.html > ...
1
2   {{titulo}}
3
4 </h1>
5
6 <button (click)="acomulador(+1)"> +1 </button>
7 <span>{{numero}} </span>
8 <button (click)="acomulador(-1)"> -1 </button>
9
```

Both the `acomulador` method in `app.component.ts` and the corresponding button click handlers in `app.component.html` are highlighted with red boxes.

TS app.component.ts M X

<> app.component.html M

...

```
src > app > TS app.component.ts > AppComponent > base
  4   selector: 'app-root',
  5   templateUrl: './app.component.html',
  6   styleUrls: ['./app.component.css']
  7 }
  8
  9
10 export class AppComponent {
11   titulo:string = 'Contador';
12   numero:number = 10;
13   base:number = 5;
14
15   acomulador(valor:number){
16     this.numero += valor;
17   }
18 }
19
20
```

<> app.component.html M X

src > app > <> app.component.html > ...

Go to component

```
1   <h1>
2     {{titulo}}
3     Base de acumulador {{base}}
4   </h1>
5
6   <button (click)="acomulador(base)"> +{{base}} </button>
7   <span>{{numero}} </span>
8   <button (click)="acomulador(-base)"> -{{base}} </button>
9
```

Pasamos el Acumulador a un componente

The screenshot shows the Visual Studio Code interface with the following details:

- File Menu:** Archivo, Editar, Selección, Ver, Ir, Ejecutar, ...
- Title Bar:** acumulador.component.ts - primero - Visual Studio
- Left Sidebar:** EXPLORADOR, EDITORES ABIERTOS, PRIMERO, .angular, node_modules, src, app, PROBLEMAS, SALIDA, TERMINAL.
- Editor Area:** The file "acumulador.component.ts" is open, highlighted with a red border. The code is as follows:

```
1 import { Component } from "@angular/core";
2
3
4 @Component({
5   selector: 'app-acumulador',
6   template: `
7     <div>
8       <h1>Contador</h1>
9       <p>${contador}</p>
10      <button (click)="incrementar()>Incrementar</button>
11    </div>
12  })
13
14  export class ContadorComponent{
15}
```

Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda

acumulador.component.ts - primero - Visual Studio Code [Administrador]

ts acumulador.component.ts U X

↳ app.component.html M

↳ ...

↳ app.component.html M X

src > app > ts acumulador.component.ts > ...

```
1 import { Component } from "@angular/core";
2
3
4 @Component({
5   selector: 'app-acumulador',
6   template: `
7     ^
8   `
9 })
10
11 export class ContadorComponent{
```

src > app > ↳ app.component.html > ...

```
1 Go to component
2 <h1>
3   {{titulo}}
4   Base de acumulador {{base}}
5   </h1>
6   <button (click)="acomulador(base)"> +{{base}} </button>
7   <span>{{numero}} </span>
8   <button (click)="acomulador(-base)"> -{{base}} </button>
```



Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda

acumulac

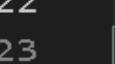


TS acumulador.component.ts 9, U X

TS app.component.ts M



```
src > app > TS acumulador.component.ts > ContadorComponent
  1   import { Component } from "@angular/core";
  2
  3
  4   @Component({
  5     selector: 'app-acumulador',
  6     template: `
  7
  8     <h1>
  9       {{titulo}}
10       Base de acumulador {{base}}
11     </h1>
12
13     <button (click)="acomulador(base)"> +{{base}} </button>
14     <span>{{numero}} </span>
15     <button (click)="acomulador(-base)"> -{{base}} </button>
16
17   `
18 })
19
20 export class ContadorComponent{
```



24

acumulador.component.ts 9, U X

app.component.ts M

src > app > acumulador.component.ts > ContadorComponent

```
1 import { Component } from '@angular/core';
2
3
4 @Component({
5   selector: 'app-acumulador',
6   template: `
7     <h1>
8       {{titulo}}
9       Base de acumulador {{base}}
10    </h1>
11
12    <button (click)="acomulador(base)"> +{{base}} </button>
13    <span>{{numero}} </span>
14    <button (click)="acomulador(-base)"> -{{base}} </button>
15
16  `
17
18})
19
20 export class ContadorComponent{
```



app.component.ts M X

src > app > app.component.ts > AppComponent

```
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 }
8
9
10 export class AppComponent {
11   titulo:string = 'Contador';
12   numero:number = 10;
13   base:number = 5;
14
15   acomulador(valor:number){
16     this.numero += valor;
17   }
18
19
20}
```

ts acumulador.component.ts U X ts app.component.ts M

src > app > ts acumulador.component.ts > ContadorComponent > acomulador

```
5   selector:'app-acumulador',
6   template: `
7
8     <h1>
9       {{titulo}}
10      Base de acumulador {{base}}
11    </h1>
12
13    <button (click)="acomulador(base)"> +{{base}} </button>
14    <span>{{numero}} </span>
15    <button (click)="acomulador(-base)"> -{{base}} </button>
16
17  `
18 }
19
20 export class ContadorComponent{
21   titulo:string = 'Contador';
22   numero:number = 10;
23   base:number = 5;
24
25   acomulador(valor:number){
26     this.numero += valor;
27   }
28
29 }
```

Ahora agregamos este nuevo componente al principal

The screenshot shows the Visual Studio Code interface with two tabs open:

- acumulador.component.ts**: Contains the code for the `ContadorComponent`. It includes an `selector: 'app-acumulador'` and a template with an `h1` element and two buttons for incrementing and decrementing a value.
- app.component.html**: Contains the main application structure. A context menu is open over the `<app-acumulador>` selector, with the "Go to component" option highlighted.

The code in `acumulador.component.ts` is as follows:

```
5 selector: 'app-acumulador',
6 template: `
7
8 <h1>
9   {{titulo}}
10  Base de acumulador {{base}}
11 </h1>
12
13 <button (click)="acomulador(base)"> +{{base}} </button>
14 <span>{{numero}} </span>
15 <button (click)="acomulador(-base)"> -{{base}} </button>
16
17
18 >
19
20 export class ContadorComponent{
21   titulo:string = 'Contador';
22   numero:number = 10;
23   base:number = 5;
24
25   acomulador(valor:number){
26     this.numero += valor;
27   }
28
29 }
```

The code in `app.component.html` is as follows:

```
1 <!-- app component.html -->
2 <app-acumulador></app-acumulador>
3
```

El error nos indica que el componente aun no ha sido agregado el modulo

A screenshot of a terminal window from a code editor. The title bar shows the path: 'src/app/app.component.html' and 'src/app/app.component.ts'. The window has tabs for 'PROBLEMAS' (with 1 error), 'SALIDA', 'CONSOLA DE DEPURACIÓN', and 'TERMINAL'. The 'TERMINAL' tab is active.

```
14 | <span>{{numero}}</span>
    |
PROBLEMAS 1 SALIDA CONSOLA DE DEPURACIÓN TERMINAL node + ⌂ ⌂
```

2 unchanged chunks

Build at: 2021-12-07T12:38:54.071Z - Hash: ec56f3aa7b49f626 - Time: 332ms

Error: src/app/app.component.html:2:1 - error NG8001: 'app-acumulador' is not a known element:

1. If 'app-acumulador' is an Angular component, then verify that it is part of this module.
2. If 'app-acumulador' is a Web Component then add 'CUSTOM_ELEMENTS_SCHEMA' to the '@NgModule.schemas' of this component to suppress this message.

2 <app-acumulador></app-acumulador>
~~~~~

src/app/app.component.ts:5:16

5 templateUrl: './app.component.html',  
~~~~~

Error occurs in the template of component AppComponent.

x Failed to compile.

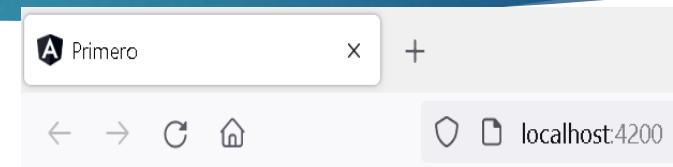
```
src > app > app.module.ts > AppModule
1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3 import { ContadorComponent } from './acumulador.component';
4
5 import { AppComponent } from './app.component';
6
7 @NgModule({
8   declarations: [
9  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

```
s acumulador.component.ts U X
src > app > ts _acumulador.component.ts > ↗ ContadorComponent
  6   template: `
  7
  8   <h1>
  9     {{titulo}}
10    Base de acumulador {{base}}
11   </h1>
12
13   <button (click)="acomulador(base)"> +{{base}} </button>
14   <span>{{numero}} </span>
15   <button (click)="acomulador(-base)"> -{{base}} </button>
16
17   `
18 }
19
20 export class ContadorComponent{
21   titulo:string = 'Contador';
22   numero:number = 10;
23   base:number = 5;
24
25   acomulador(valor:number){
26     this.numero += valor;
27   }
28
29 }
```

Con esto ya nos funciona el llamado al componente

The screenshot shows the Visual Studio Code interface. The top menu bar includes Archivo, Editar, Selección, Ver, Ir, Ejecutar, Terminal, and Ayuda. Below the menu is a toolbar with icons for Save, Undo, Redo, Find, Replace, Copy, Paste, and others. The title bar indicates the file is "app.component.html - primera". The left sidebar has sections for EXPLORADOR, EDITORES ABIERTOS, and PRIMERO. Under PRIMERO, there are files like .angular, node_modules, src, app, acumulador.component.ts, app.component.css, and app.component.html. The "app.component.html" file is highlighted with a red box. The main editor area shows the following code:

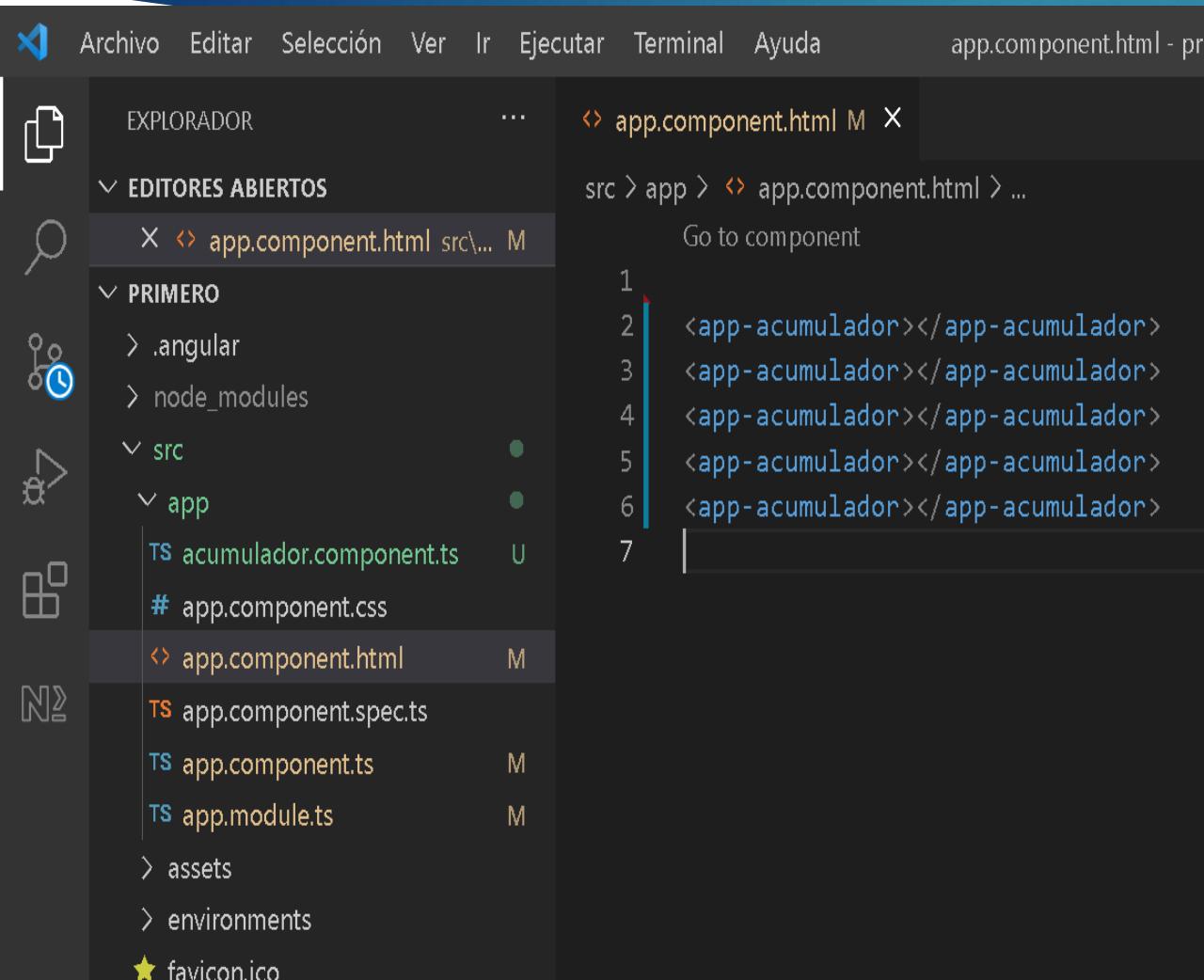
```
src > app > app.component.html > ...
  Go to component
1
2 <app-acumulador></app-acumulador>
3
```



Contador Base de acumulador 5

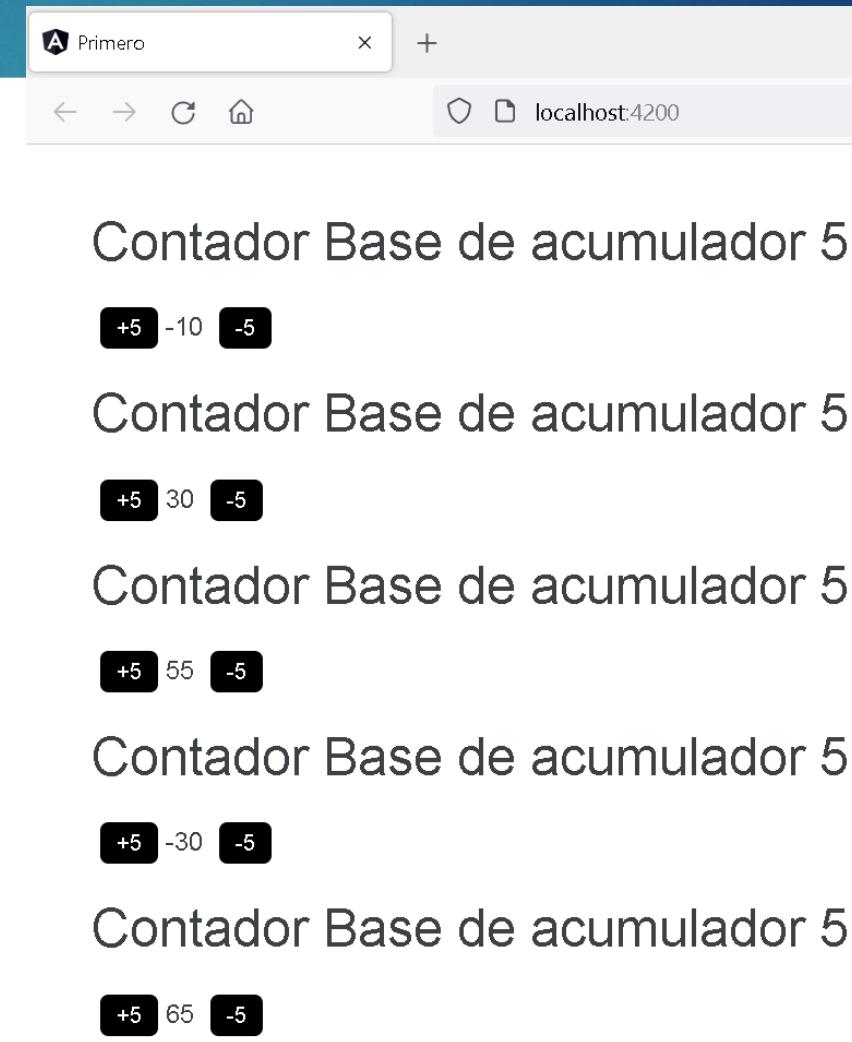
+5 5 -5

Podemos tener varias instancias del componente y todas se manejan de forma independiente



The screenshot shows the Visual Studio Code interface. The top menu bar includes Archivo, Editar, Selección, Ver, Ir, Ejecutar, Terminal, and Ayuda. The title bar indicates the file is app.component.html - pr. The left sidebar has icons for Explorador, Editores Abiertos, PRIMERO, src, app, and various files like acumulador.component.ts, app.component.css, and app.module.ts. The main editor area displays the following code:

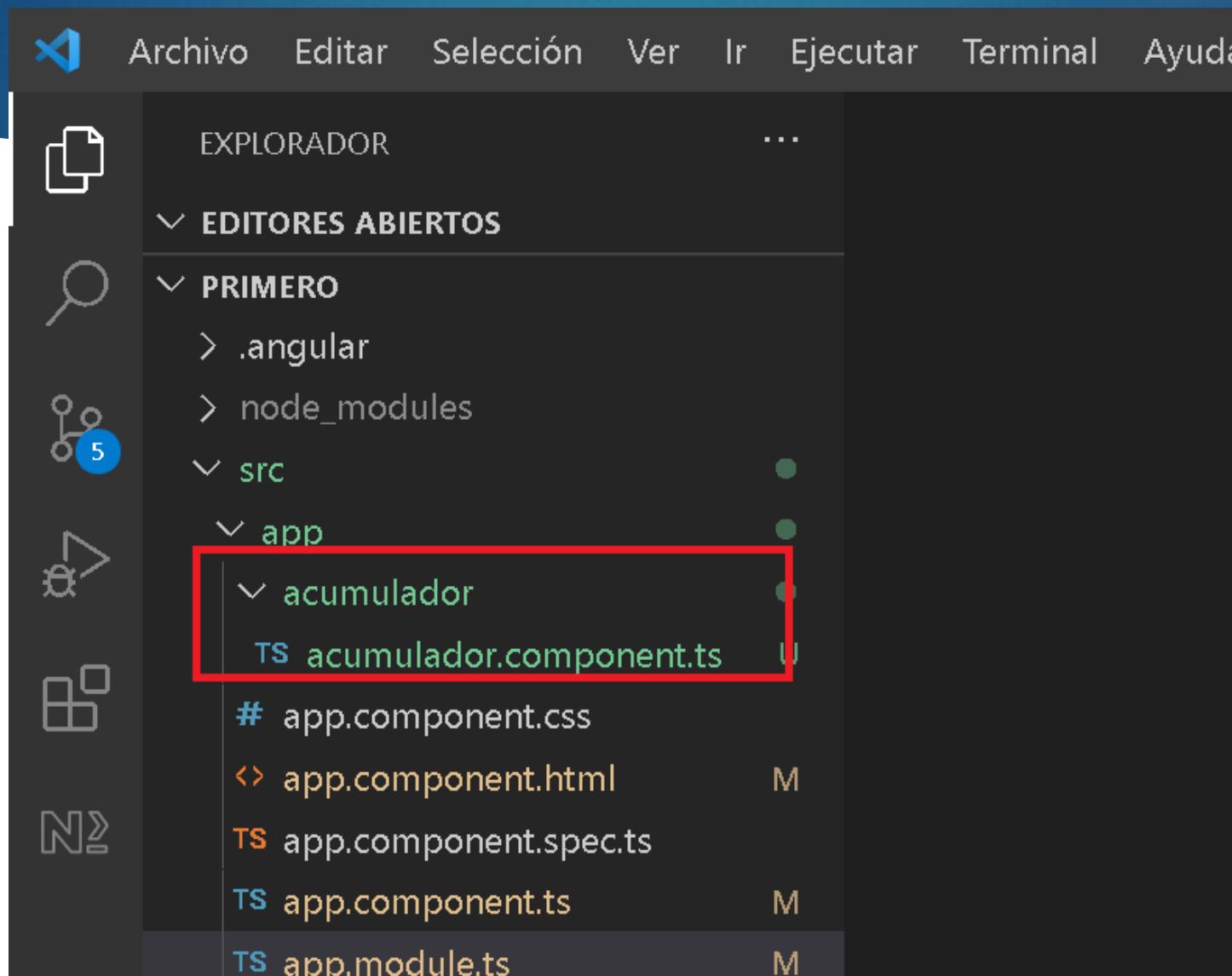
```
src > app > app.component.html > ...
Go to component
1
2 <app-acumulador></app-acumulador>
3 <app-acumulador></app-acumulador>
4 <app-acumulador></app-acumulador>
5 <app-acumulador></app-acumulador>
6 <app-acumulador></app-acumulador>
7
```



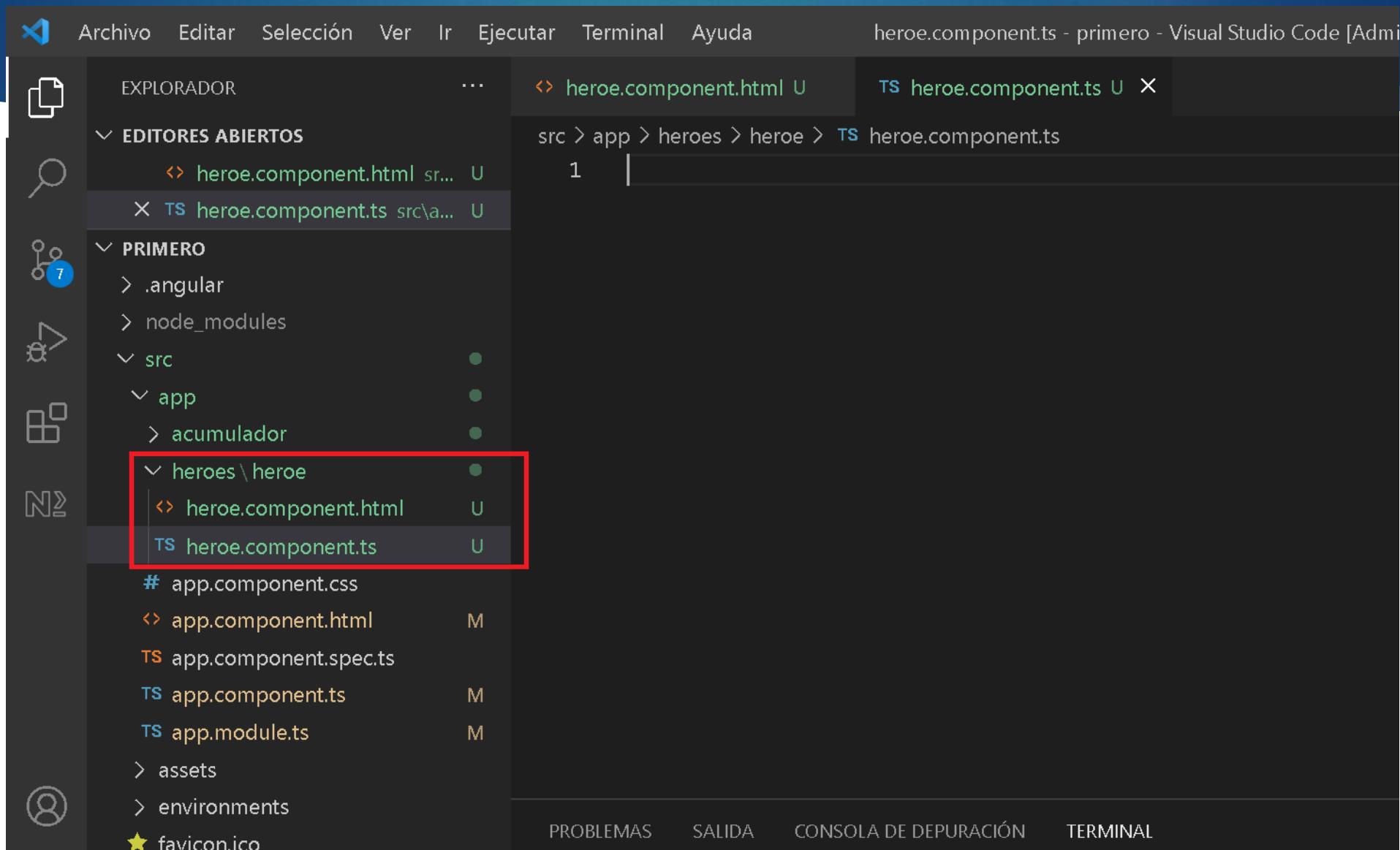
The screenshot shows a browser window with the URL localhost:4200. The page displays four identical components, each with a value of 5. Each component has three buttons: +5, 30, and -5.

- Contador Base de acumulador 5
+5 -10 -5
- Contador Base de acumulador 5
+5 30 -5
- Contador Base de acumulador 5
+5 55 -5
- Contador Base de acumulador 5
+5 -30 -5
- Contador Base de acumulador 5
+5 65 -5

Podemos crear directorios para acomodar mejor los componentes



Ahora creamos un nuevo componente



EXPLORADOR ... hero.component.html U TS hero.component.ts U X

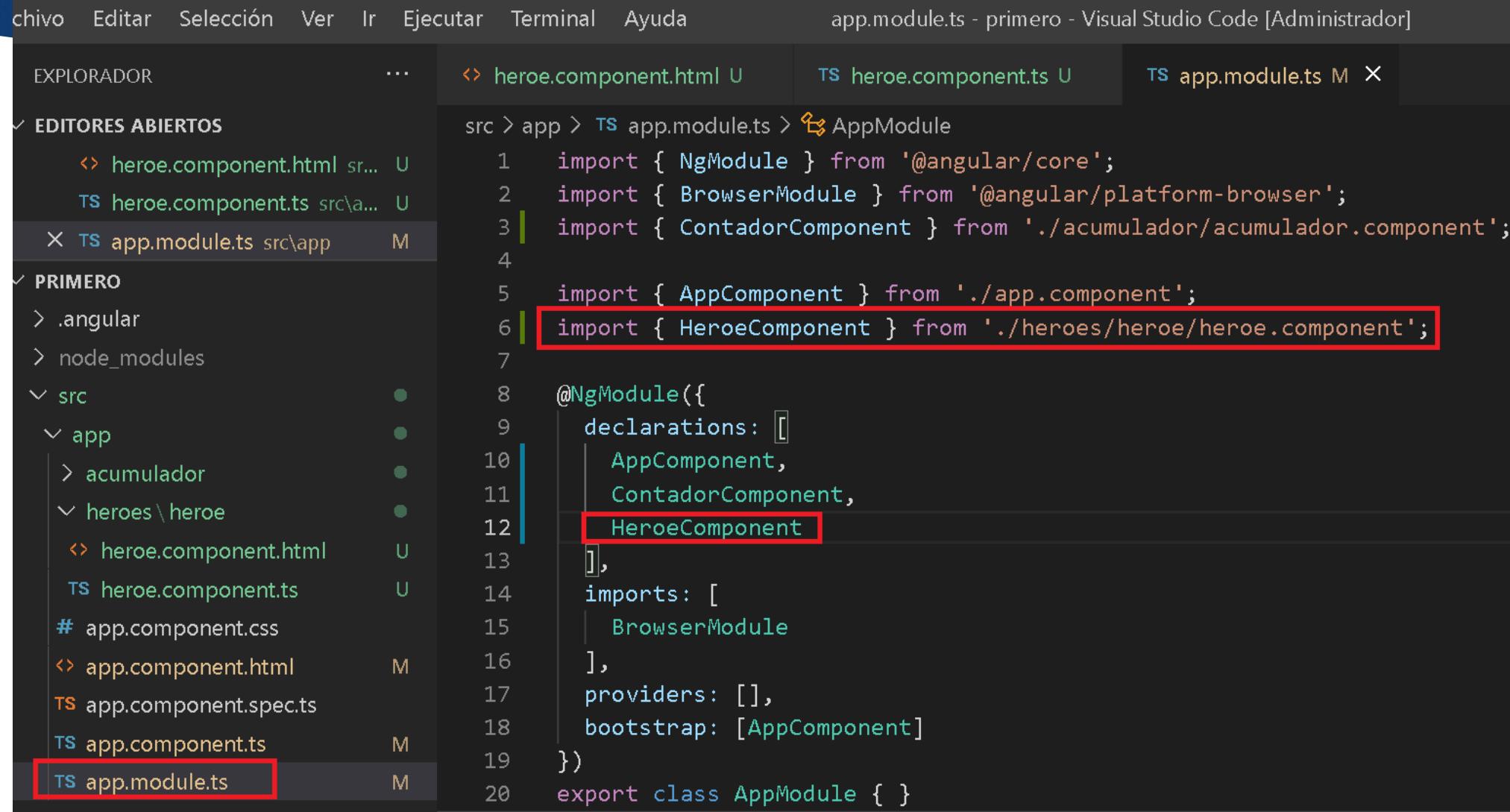
EDITORES ABIERTOS hero.component.html sr... U
X TS hero.component.ts src\... U

PRIMERO .angular node_modules src ●
app acumulador ●
heroes\hero ●
hero.component.html U
TS hero.component.ts U

src > app > heroes > hero > TS hero.component.ts > HeroeComponent

```
1 import { Component } from "@angular/core";
2
3
4 @Component({
5   selector: "app-hero",
6   templateUrl: "hero.component.html"
7 })
8
9 export class HeroeComponent{
10
11 }
12
```

No olvidemos agregar el nuevo componente a app.module.ts



The screenshot shows the Visual Studio Code interface with the following details:

- Menu Bar:** Archivo, Editar, Selección, Ver, Ir, Ejecutar, Terminal, Ayuda.
- Title Bar:** app.module.ts - primero - Visual Studio Code [Administrador]
- Explorer:** Shows the project structure:
 - EDITORES ABIERTOS: hero.component.html, hero.component.ts, app.module.ts
 - PRIMERO: .angular, node_modules, src (with app, acumulador, heroes\hero subfolders)
 - src\app\acumulador\acumulador.component.ts
 - src\app\heroes\hero\hero.component.html, hero.component.ts, app.component.css, app.component.html, app.component.spec.ts, app.component.ts
 - src\app\module.ts (highlighted with a red border)
- Code Editor:** The app.module.ts file is open, showing the following code:

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

The line `import { HeroeComponent } from './heroes/heroe/heroe.component';` is highlighted with a red box.

EXPLORADOR ...

EDITORES ABIERTOS

- X <> heroе.component.html sr... U
- TS heroе.component.ts src\... U
- TS app.module.ts src\app M

PRIMERO

- > .angular
- > node_modules
- src
 - app
 - acumulador
 - heroes\heroе
 - <> heroе.component.html U
 - TS heroе.component.ts U

src > app > heroes > heroе > <> heroе.component.html > h

Go to component

1 <h1>Componente de Heroе</h1>

2

EXPLORADOR

↔ app.component.html M X

EDITORES ABIERTOS

X ↔ app.component.html src\... M

PRIMERO

src

app

> acumulador

heroes\hero

↔ hero.component.html

TS hero.component.ts

app.component.css

↔ app.component.html M

TS app.component.spec.ts

TS app.component.ts M

TS app.module.ts M

> assets

> environments

★ favicon.ico

↔ index.html

src > app > ↔ app.component.html > app-hero

Go to component

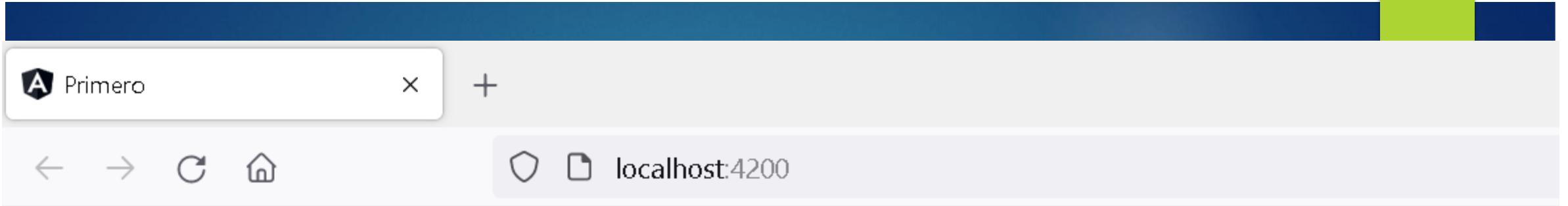
1 | <! --<app-acumulador></app-acumulador>-->

2 |

3 | <app-hero></app-hero>

4 |

5 |



Componente de Heroe

Agregamos lo siguiente

Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda heroе.component.ts - primero - Visual Studio Code [Administrador] —

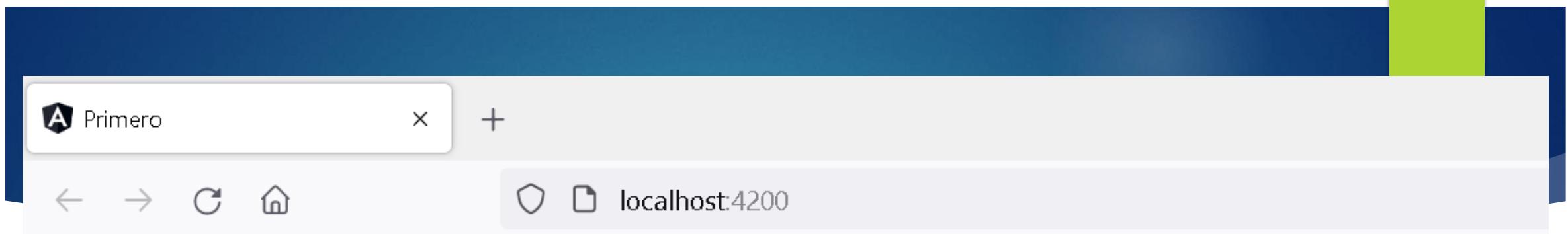
heroе.component.ts U X heroе.component.html U X

```
src > app > heroes > heroе > heroе.component.ts > HeroеComponent
```

```
3
4  @Component({
5    selector: "app-heroе",
6    templateUrl: "heroе.component.html"
7  })
8
9  export class HeroеComponent{
10    nombre: string = "Ironman";
11    edad: number = 45;
12
13    get nombreCapitalizado(){
14      return this.nombre.toUpperCase();
15    }
16    obtenerNombre():string{
17      return `${this.nombre} - ${this.edad}`;
18    }
19}
```

```
src > app > heroes > heroе > heroе.component.html > dl > Go to component
```

```
1  <h1>Componente de Heroе</h1>
2
3  <dl>
4    <td>Nombre:</td>
5    <dd>{{nombre}}</dd>
6
7    <td>Edad:</td>
8    <dd>{{edad}}</dd>
9
10   <td>Funcion:</td>
11   <dd>{{obtenerNombre()}}</dd>
12
13   <td>Capitalizado:</td>
14   <dd>{{nombreCapitalizado}}</dd>
15
16
```



Componente de Heroe

Nombre:

Ironman

Edad:

45

Funcion:

Ironman - 45

Capitalizado:

IRONMAN

Agregamos estas funciones y botones

Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda

heroе.component.html - primero - Visual Studio Code [Administrador]

heroе.component.ts U X

```
o > heroes > heroе > heroе.component.ts > HeroeComponent > cam
10    nombre: string = "Ironman";
11    edad: number = 45;
12
13    get nombreCapitalizado(){
14        return this.nombre.toUpperCase();
15    }
16    obtenerNombre():string{
17        return `${this.nombre} - ${this.edad}`;
18    }
19
20    cambiarNombre():void{
21        this.nombre = "Spiderman";
22    }
23
24    cambiarEdad():void{
25        console.log('hey...');
26        this.edad = 30;
27    }
28
29 }
```

heroе.component.html U X

```
src > app > heroes > heroе > heroе.component.html > dl > button
1     <h1>Componente de Heroe</h1>
2
3     <dl>
4         <td>Nombre:</td>
5         <dd>{{nombre}}</dd>
6
7         <td>Edad:</td>
8         <dd>{{edad}}</dd>
9
10        <td>Funcion:</td>
11        <dd>{{obtenerNombre()}}</dd>
12
13        <td>Capitalizado:</td>
14        <dd>{{nombreCapitalizado}}</dd>
15
16        <button (click)="cambiarNombre()">Cambiar Héroe</button>
17        <button (click)="cambiarEdad()">Cambiar Héroe</button>
18
19
```

Componente de Heroe

Nombre:

Spiderman

Edad:

30

Funcion:

Spiderman - 30

Capitalizado:

SPIDERMAN

Cambiar Héroe

Cambiar Héroe

One Way Data Binding – Enlazado en una sola vía

The screenshot shows a web browser window with the URL `localhost:4200`. The main content area displays a component titled "Componente de Heroe" with the following data:

- Nombre: Spiderman
- Edad: 30
- Función: Spiderman - 30
- Capitalizado: SPIDERMAN

Below the data are two buttons: "Cambiar Héroe" and "Cambiar Héroe".

At the bottom of the screen, the browser's developer tools are open, specifically the "Consola" tab. The console output is as follows:

```
Disconnected! index.js:548
Trying to reconnect... index.js:548
Angular is running in development mode. Call enableProdMode() to enable production mode. core.mjs:24856:16
Live Reloading enabled. index.js:548
hey... heroе.component.ts:25:12
```

The message "hey..." is highlighted with a red rectangle, and the file path "heroе.component.ts:25:12" is also highlighted with a red rectangle.

También podemos generar componentes por comando

PROBLEMAS

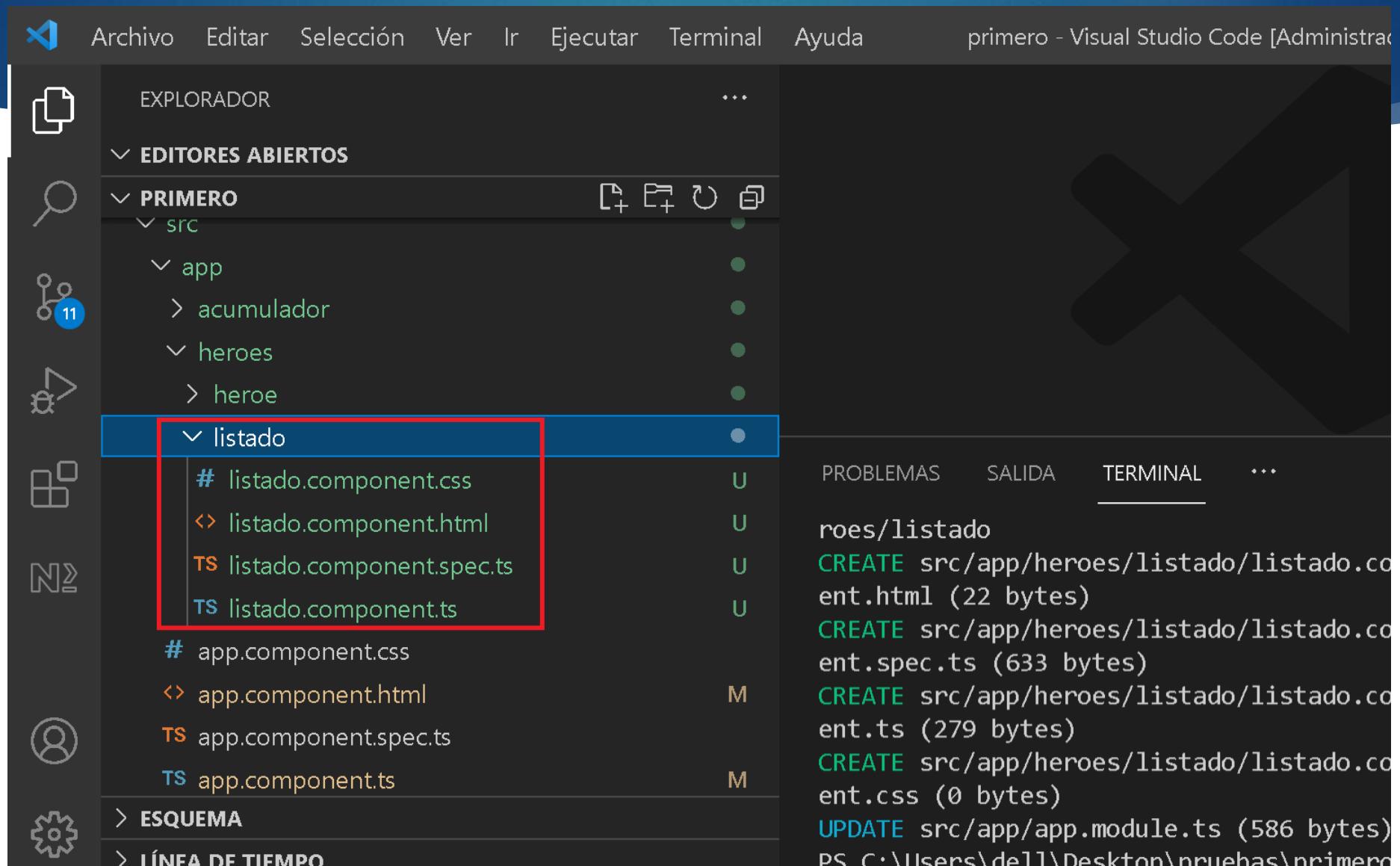
SALIDA

CONSOLA DE DEPURACIÓN

TERMINAL

```
PS C:\Users\dell\Desktop\pruebas\primero> ng g c heroes/listado
CREATE src/app/heroes/listado/listado.component.html (22 bytes)
CREATE src/app/heroes/listado/listado.component.spec.ts (633 bytes)
CREATE src/app/heroes/listado/listado.component.ts (279 bytes)
CREATE src/app/heroes/listado/listado.component.css (0 bytes)
UPDATE src/app/app.module.ts (586 bytes)
PS C:\Users\dell\Desktop\pruebas\primero>
```

Nos genera de manera automática estos archivos



También agrega la función al app.module.ts

The screenshot shows a Visual Studio Code editor window with the title bar "app.module.ts - primero - Visual Studio Code". The menu bar includes "Archivo", "Editar", "Selección", "Ver", "Ir", "Ejecutar", "Terminal", and "Ayuda". The code editor displays the following TypeScript code for the application module:

```
src > app > TS app.module.ts > ...
1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3 import { ContadorComponent } from './acumulador/acumulador.component';
4
5 import { AppComponent } from './app.component';
6 import { HeroeComponent } from './heroes/heroe/heroe.component';
7 import { ListadoComponent } from './heroes/listado/listado.component';
8
9 @NgModule({
10   declarations: [
11     AppComponent,
12     ContadorComponent,
13     HeroeComponent,
14     ListadoComponent
15   ],
16   imports: [
17     BrowserModule
18   ],
19   providers: [],
20   bootstrap: [AppComponent]
```

Several parts of the code are highlighted with red boxes: the file path "TS app.module.ts", the imports for "ContadorComponent", "ListadoComponent", and "ListadoComponent" again, and the declaration for "ListadoComponent".

Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda app.component.html - primero - Visual Studio Code [A]

EXPLORADOR

EDITORES ABIERTOS

- listado.component.html src\app\heroes\listado... U
- app.component.html src\app M

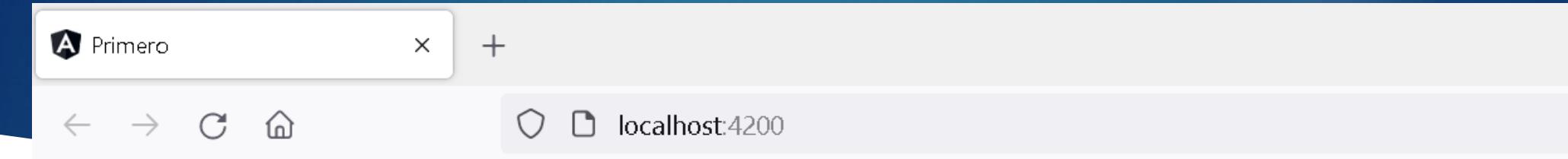
PRIMERO

- heroes
 - heroе
 - listado
 - # listado.component.css
 - listado.component.html
 - listado.component.spec.ts
 - listado.component.ts
- # app.component.css
- app.component.html M
- app.component.spec.ts
- app.component.ts M
- app.module.ts M

src > app > app.component.html > app-listado

Go to component

```
1 <!--<app-acumulador></app-acumulador>-->
2
3 <!--<app-heroe></app-heroe>-->
4
5 <app-listado></app-listado>
6
7
```

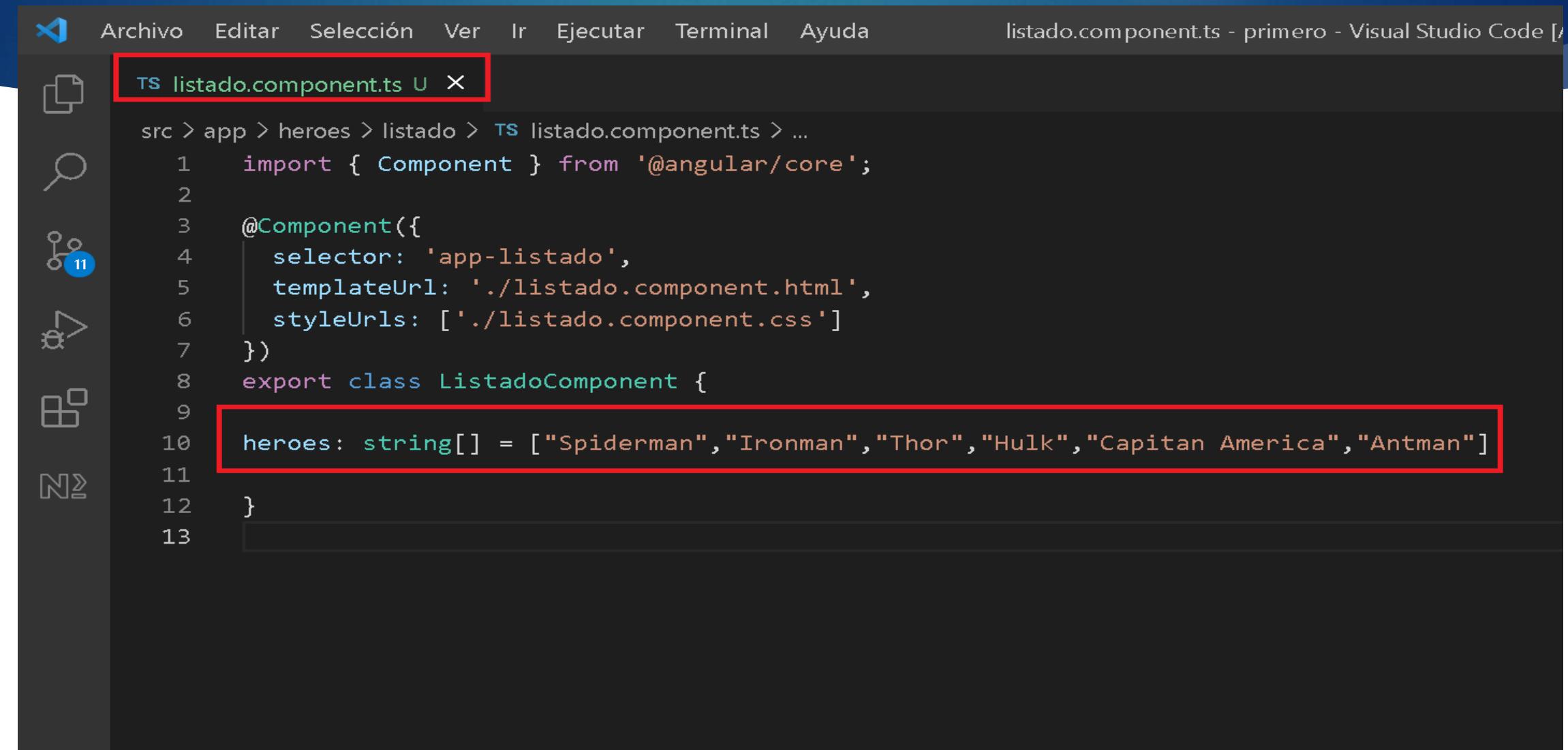


listado works!

The screenshot shows the Visual Studio Code interface with the following details:

- Top Bar:** Archivo, Editar, Selección, Ver, Ir, Ejecutar, Terminal, Ayuda.
- Terminal Tab:** listado.component.html - primero - Visual Studio Code [Ad]
- Terminal Content:** A file browser path: src > app > heroes > listado > listado.component.html > ...
A code editor with two lines:
 - 1 <p>listado works!</p>
 - 2 |
- Explorer Pane:** Shows the project structure.
 - EDITORES ABIERTOS:** listado.component.html (selected), app.component.html
 - PRIMERO:** heroes (with heroe and listado subfolders), listado (with listado.component.css, listado.component.html, listado.component.spec.ts, listado.component.ts), app.component.css, app.component.html, app.component.spec.ts, app.component.ts, app.module.ts
- Sidebar Icons:** File, Search, Share, Run, Preview, Split View, New File.
- Status Bar:** Shows 11 open files.

Directiva *ngFor



The screenshot shows the Visual Studio Code interface with the file `listado.component.ts` open. The file path is `src > app > heroes > listado > listado.component.ts`. The code defines a component named `ListadoComponent` with an array of heroes.

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-listado',
5   templateUrl: './listado.component.html',
6   styleUrls: ['./listado.component.css']
7 })
8 export class ListadoComponent {
9
10   heroes: string[] = ["Spiderman", "Ironman", "Thor", "Hulk", "Capitan America", "Antman"]
11
12 }
13
```



Archivo Editar Selección Ver Ir Ejecutar ... listado.componente



TS listado.component.ts U

↔ listado.component.html U X



src > app > heroes > listado > ↔ listado.component.html > ...

Go to component

```
1 <p>listado de Héroes</p>
2
3 <ul>
4   <li *ngFor="let heroe of heroes; let i = index">
5     {{i+1}} - {{heroe}}
6   </li>
7 </ul>
8
```



11



listado de Héroes

- 1 - Spiderman
- 2 - Ironman
- 3 - Thor
- 4 - Hulk
- 5 - Capitan America
- 6 - Antman

ts listado.component.ts U X

↳ listado.component.html U

```
src > app > heroes > listado > ts listado.component.ts > ...
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-listado',
5   templateUrl: './listado.component.html',
6   styleUrls: ['./listado.component.css']
7 })
8 export class ListadoComponent {
9
10 heroes: string[] = ["Spiderman", "Ironman", "Thor", "Hulk", "Capitan America", "Antman"];
11
12 borrarHeroe(){
13   this.heroes.pop();
14 }
15
16 }
17 |
```



Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda

listado.c...



TS listado.component.ts U

↔ listado.component.html U X



src > app > heroes > listado > ↔ listado.component.html > ...



11

Go to component

```
1  <p>listado de Héroes</p>
2
3  <ul>
4    <li *ngFor="let heroе of heroes; let i = index">
5      {{i+1}} - {{heroе}}
6    </li>
7  </ul>
8
9  <button (click)="borrarHeroe()">Borrar Héroe</button>
```



TS listado.component.ts U X

↳ listado.component.html U

src > app > heroes > listado > TS listado.component.ts > ListadoComponent

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-listado',
5   templateUrl: './listado.component.html',
6   styleUrls: ['./listado.component.css']
7 })
8 export class ListadoComponent {
9
10 heroes: string[] = ["Spiderman", "Ironman", "Thor", "Hulk", "Capitan America", "Antman"];
11 heroeBorrado: string | undefined = "";
12
13   borrarHeroe(){
14     this.heroeBorrado = this.heroes.pop();
15   }
16 }
17
```



Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda

listado.component.html - prime



TS listado.component.ts U

↔ listado.component.html U X

src > app > heroes > listado > ↔ listado.component.html > ...

Go to component

```
1  <p>listado de Héroes</p>
2
3  <ul>
4    <li *ngFor="let hero of heroes; let i = index">
5      | {{i+1}} - {{hero}}
6    </li>
7  </ul>
8
9  <p>Heroe Borrado {{heroeBorrado}}</p>
10
11
12 <button (click)="borrarHeroe()">Borrar Héroe</button>
13
```

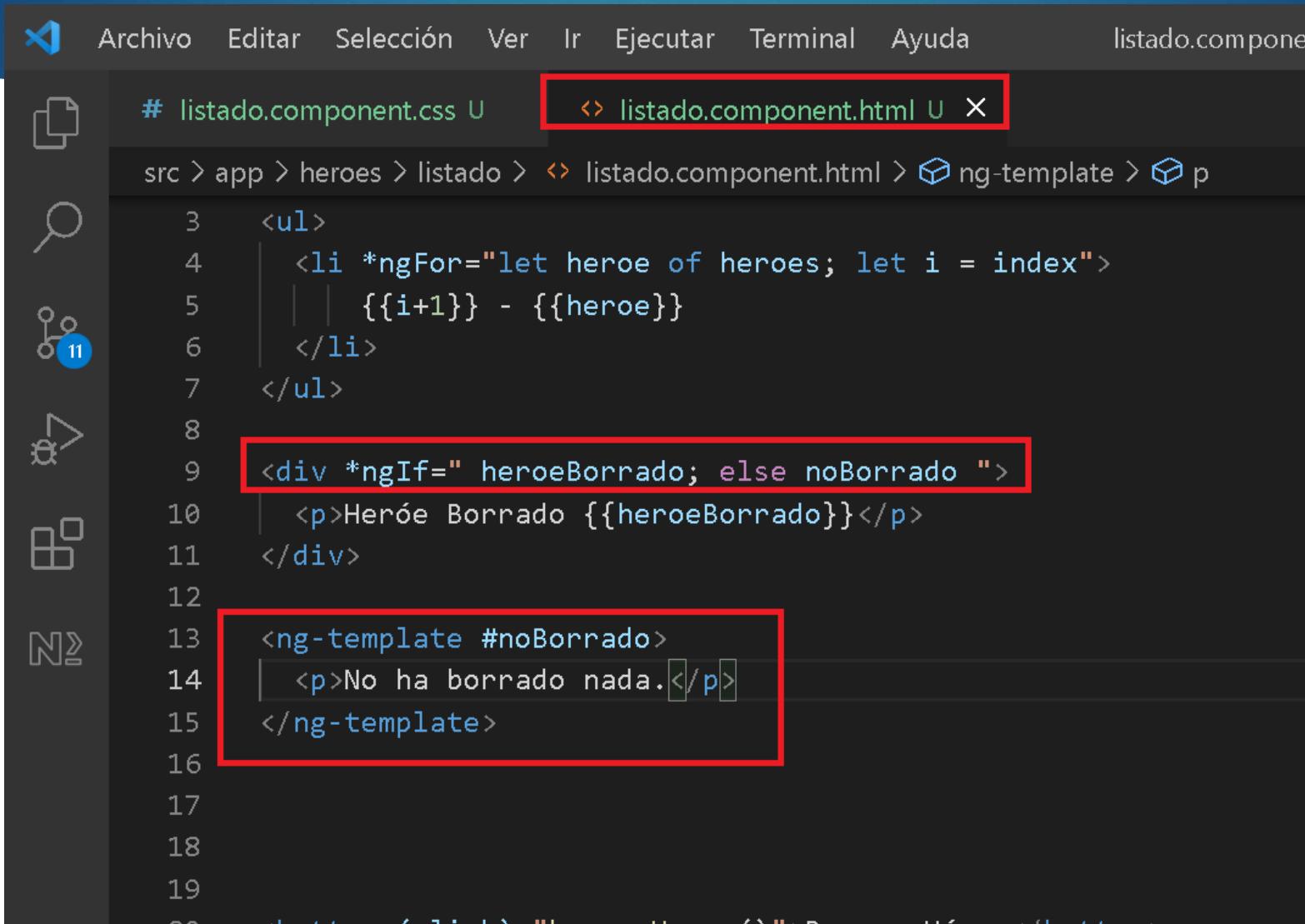
Directiva *ngIf



The screenshot shows a code editor interface with a dark theme. On the left is a vertical toolbar with icons for file operations, search, and other functions. The main area displays a file named "listado.component.html". A red box highlights the file tab. Another red box highlights the code block containing the *ngIf directive.

```
# listado.component.css U ⌂ listado.component.html U X
src > app > heroes > listado > ⌂ listado.component.html > button
    Go to component
1  <p>listado de Héroes</p>
2
3  <ul>
4      <li *ngFor="let heroe of heroes; let i = index">
5          {{i+1}} - {{heroe}}
6      </li>
7  </ul>
8
9  <div *ngIf=" heroeBorrado !== '' ">
10     <p>Heróe Borrado {{heroeBorrado}}</p>
11 </div>
12
13
14
15 <button (click)="borrarHeroe()">Borrar Héroe</button>
16
```

Ng-Template y el ngIf-else, atravez de una referencia local se puede llamar a templates



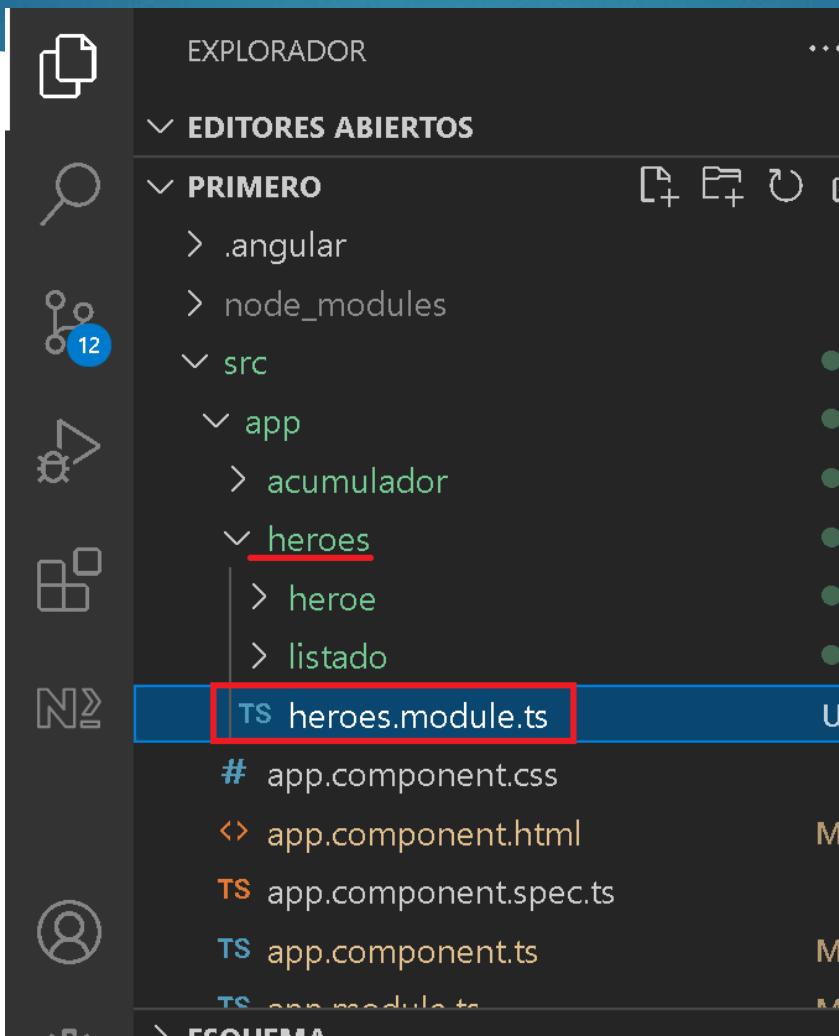
The screenshot shows a code editor in Visual Studio Code with the file `listado.component.html` open. The file path is visible in the title bar: `src > app > heroes > listado > listado.component.html`. The code uses `*ngIf` and `*ngTemplateOutlet` to conditionally render content based on the value of `heroeBorrado`.

```
src > app > heroes > listado > listado.component.html > ng-template > p
3   <ul>
4     <li *ngFor="let heroe of heroes; let i = index">
5       {{i+1}} - {{heroe}}
6     </li>
7   </ul>
8
9   <div *ngIf=" heroeBorrado; else noBorrado ">
10    <p>Heróe Borrado {{heroeBorrado}}</p>
11  </div>
12
13  <ng-template #noBorrado>
14    <p>No ha borrado nada.</p>
15  </ng-template>
16
```

Módulos

- ▶ Los módulos lo podemos tener en varios componentes, esto nos ayuda a encapsular las configuraciones de cada modulo, a no acumular todo el proyecto en un solo modulo y también mejorando el rendimiento no llamando a clases y vistas que no serán utilizadas inicialmente.

En héroes creamos un nuevo modulo



TS heroes.module.ts U X

```
src > app > heroes > TS heroes.module.ts > 📁 HeroesModule
  1 import { CommonModule } from '@angular/common';
  2 import { NgModule } from '@angular/core';
  3 import { HeroeComponent } from './heroe/heroe.component';
  4 import { ListadoComponent } from './listado/listado.component';
  5
  6 @NgModule([
  7   declarations: [
  8     HeroeComponent,
  9     ListadoComponent
 10   ],
 11   exports: [
 12     ListadoComponent
 13   ],
 14   imports: [
 15     CommonModule
 16   ]
 17 ])
 18 })
 19
 20 export class HeroesModule{}
```

Removemos estas llamadas de app.module.ts

The image shows a screenshot of the Visual Studio Code interface. On the left is the Explorer sidebar, which lists files and folders. In the center is the code editor with the file `app.module.ts` open. A red box highlights the title bar of the code editor. Another red box highlights the imports section of the code, specifically the declarations array.

EXPLORADOR

EDITORES ABIERTOS

- heroes.module.ts
- app.module.ts

PRIMERO

- node_modules
- src
 - app
 - acumulador
 - heroes
 - hero
 - listado
 - heroes.module.ts
 - app.component.css
 - app.component.html
 - app.component.spec.ts
 - app.component.ts
 - app.module.ts
- assets
- environments
- favicon.ico

ESQUEMA

TS heroes.module.ts U TS app.module.ts M X

```
src > app > TS app.module.ts > AppModule
1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3 import { ContadorComponent } from './acumulador/acumulador.component';
4
5 import { AppComponent } from './app.component';
6 import { HeroeComponent } from './heroes/heroe/heroe.component';
7 import { ListadoComponent } from './heroes/listado/listado.component';
8
9 @NgModule({
10   declarations: [
11     AppComponent,
12     ContadorComponent,
13     HeroeComponent,
14     ListadoComponent
15   ],
16   imports: [
17     BrowserModule
18   ],
19   providers: [],
20   bootstrap: [AppComponent]
21 })
22 export class AppModule { }
```

EXPLORADOR

EDITORES ABIERTOS

PRIMERO

heroes.module.ts

app.module.ts

app.component.css

app.component.html

app.component.spec.ts

app.component.ts

app.module.ts

assets

environments

favicon.ico

ESQUEMA

heroes.module.ts U

app.module.ts M X

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

Ahora agregamos al modulo creado

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORADOR**: Shows the project structure:
 - EDITORES ABIERTOS**: heroes.module.ts (U), app.module.ts (M)
 - PRIMERO**: node_modules, src (with app, acumulador, heroes),英雄 (hero, listado), heroes.module.ts (U), app.component.css, app.component.html (M), app.component.spec.ts, app.component.ts (M), app.module.ts (M), assets, environments, favicon.ico
 - ESQUEMA**
 - LÍNEA DE TIEMPO**
- EDITOR**: The file **app.module.ts** is open and displayed. The code is:

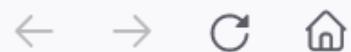
```
src > app > TS app.module.ts > AppModule
1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3 import { ContadorComponent } from './acumulador/acumulador.component';
4
5 import { AppComponent } from './app.component';
6 import { HeroesModule }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

The line `import { HeroesModule } from './heroes/heroes.module';` and the word `HeroesModule` in the imports section are highlighted with a red box.

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORADOR**: Shows the project structure. The file `app.component.html` is selected and highlighted with a red border.
- EDITORES ABIERTOS**: Shows three files open:
 - `heroes.module.ts` (U)
 - `app.module.ts` (M)
 - `app.component.html` (src\app) (M)
- PRIMERO**: Shows the contents of the `src/app` folder:
 - `acumulador`
 - `heroes`: Contains `heroe` and `listado`.
 - `heroes.module.ts` (U)
 - `app.component.css` (#)
 - `app.component.html` (src\app) (M) - This file is also highlighted with a red border.
 - `app.component.spec.ts`
 - `app.component.ts` (M)
 - `app.module.ts` (M)
- Editor**: The `app.component.html` file is open. The code shown is:

```
src > app > app.component.html > ...
    Go to component
1 | <!--<app-acumulador></app-acumulador>-->
2 |
3 | <!--<app-heroe></app-heroe>-->
4 |
5 | <app-listado></app-listado>
6 |
7 | 
```
- Terminal**: Not visible in the screenshot.



localhost:4200

listado de Héroes

- 1 - Spiderman
- 2 - Ironman
- 3 - Thor
- 4 - Hulk
- 5 - Capitan America
- 6 - Antman

No ha borrado nada.

[Borrar Héroe](#)

EXPLORADOR

EDITORES ABIERTOS

PRIMERO

src

app

acumulador

acumulador.component.ts

acumulador.module.ts

heroes

hero

listado

heroes.module.ts

app.component.css

app.component.html

app.component.spec.ts

app.component.ts

app.module.ts

assets

ESQUEMA

TS acumulador.module.ts U X

src > app > acumulador > TS acumulador.module.ts > AcumuladorModule

```
1 import { CommonModule } from "@angular/common";
2 import { NgModule } from "@angular/core";
3 import { ContadorComponent } from "./acumulador.component";
4
5
6 @NgModule({
7   declarations:[
8     ContadorComponent
9   ],
10  exports: [
11    ContadorComponent
12  ],
13  imports: [
14    CommonModule
15  ]
16})
17 export class AcumuladorModule{}
```

PROBLEMAS 1 SALIDA CONSOLA DE DEPURACIÓN TERMINAL

Compiled successfully.

Removemos esto

The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists files and folders. In the center is the code editor with the file `app.module.ts` open. The code is written in TypeScript and defines the `AppModule`.

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

The line `import { ContadorComponent } from './acumulador/acumulador.component';` and the declaration `ContadorComponent` in the declarations array are highlighted with red boxes, indicating they are being removed.

At the bottom of the code editor, a message `✓ Compiled successfully.` is displayed.

EXPLORADOR

EDITORES ABIERTOS

PRIMERO

src

app

acumulador

heroes

app.component.css

app.component.html

TS app.component.spec.ts

TS app.component.ts

TS app.module.ts

acumulador.module.ts

PROBLEMAS 1 SALIDA CONSOLA DE DEPURACIÓN TERMINAL

ESQUEMA

TS app.module.ts M X

```
src > app > TS app.module.ts > AppModule
1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3 import { AcumuladorModule } from './acumulador/acumulador.module';
4 import { AppComponent } from './app.component';
5 import { HeroesModule } from './heroes/heroes.module';
6
7
8 @NgModule({
9   declarations: [
10     AppComponent,
11   ],
12   imports: [
13     BrowserModule,
14     HeroesModule,
15     AcumuladorModule
16   ],
17   providers: [],
18   bootstrap: [AppComponent]
19 })
20 export class AppModule { }
21
22
```

✓ Compiled successfully.

← → C ⌂



localhost:4200

Contador Base de acumulador 5

+5 15 -5

Creamos un nuevo modulo en el mismo proyecto

- ▶ Comando
- ▶ `ng g module nombre_modulo`

Creamos otro componente en el mismo proyecto

- ▶ En este ejemplo vemos como debemos de crear los componentes inicialmente

Generamos el modulo por comando

```
PS C:\Users\dell\Desktop\pruebas\primero> ng g module dbz
CREATE src/app/dbz/dbz.module.ts (189 bytes)
PS C:\Users\dell\Desktop\pruebas\primero>
```

Revisamos lo generado

The image shows a screenshot of the Visual Studio Code (VS Code) interface. On the left, the Explorer sidebar displays a tree view of project files. A red box highlights the file 'dbz.module.ts' located in the 'src/app/dbz' directory. In the center, the Editor tab shows the code for 'dbz.module.ts'. Another red box highlights the title bar of the editor window, which also displays 'dbz.module.ts'. The code itself is an Angular module definition:

```
src > app > dbz > TS dbz.module.ts > ...
1 import { NgModule } from '@angular/core'
2 import { CommonModule } from '@angular/common'
3
4
5
6 @NgModule({
7   declarations: [],
8   imports: [
9     CommonModule
10   ]
11 })
12 export class DbzModule { }
```

Agrego un componente

```
PS C:\Users\dell\Desktop\pruebas\primero> ng g c dbz/mainPage  
  
CREATE src/app/dbz/main-page/main-page.component.html (24 bytes)  
CREATE src/app/dbz/main-page/main-page.component.spec.ts (641 bytes)  
CREATE src/app/dbz/main-page/main-page.component.ts (286 bytes)  
CREATE src/app/dbz/main-page/main-page.component.css (0 bytes)  
UPDATE src/app/dbz/dbz.module.ts (283 bytes)  
PS C:\Users\dell\Desktop\pruebas\primero>
```

The screenshot shows a dark-themed file explorer window. At the top, there's a toolbar with icons for creating files, opening files, saving, and other operations. Below the toolbar, the file tree displays the following structure:

- PRIMERO
 - .angular
 - node_modules
 - src
 - app
 - acumulador
 - dbz
 - main-page
 - # main-page.component.css U
 - <> main-page.component.h... U
 - TS main-page.components... U
 - TS main-page.component.ts U
 - dbz.module.ts U
 - heroes

PROBLEMAS SALIDA
PS C:\Users\dell\De

Realizamos los siguientes cambios

The screenshot shows the VS Code interface with the following details:

- EXPLORADOR**: Shows the project structure:
 - EDITORES ABIERTOS**: main-page.component.html
 - PRIMERO**: .angular, node_modules
 - src**: app, dbz
 - app: acumulador
 - dbz: main-page
 - main-page.component.css
 - main-page.component.html** (highlighted with a red box)
 - main-page.component.spec.ts
 - main-page.component.ts
 - dbz.module.ts
- EDITOR**: main-page.component.html (highlighted with a red box). The code editor shows:

```
src > app > dbz > main-page > main-page.component.html > ...
Go to component
1 <p>Dragon Ball Z</p>
2 
```

EXPLORADOR

EDITORES ABIERTOS

- main-page.component.html U
- app.module.ts src\app M
- dbz.module.ts src\app... U

PRIMERO

- app
 - acumulador
 - dbz
 - main-page
 - # main-page.compon... U
 - main-page.compon... U
 - main-page.compon... U
 - main-page.compon... U
 - dbz.module.ts U
 - heroes
 - # app.component.css
 - app.component.html M
 - app.component.spec.ts
 - app.component.ts M

ESQUEMA

main-page.component.html U

app.module.ts M X

dbz.module.ts U

```
src > app > app.module.ts > AppModule
1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3 import { AcumuladorModule } from './acumulador/acumulador.module';
4 import { AppComponent } from './app.component';
5 import { DbzModule } from './dbz/dbz.module';
6 import { HeroesModule } from './heroes/heroes.module';
7
8 @NgModule({
9   declarations: [
10     AppComponent,
11   ],
12   imports: [
13     BrowserModule,
14     HeroesModule,
15     AcumuladorModule,
16     DbzModule
17   ],
18   providers: [],
19   bootstrap: [AppComponent]
20 })
21 export class AppModule { }
22
23
```

EXPLORADOR

… main-page.component.html U TS app.module.ts M TS dbz.module.ts U X ↗ app.compo

EDITORES ABIERTOS

- ↳ main-page.component.html U
- TS app.module.ts src\app M
- ✗ TS dbz.module.ts src\ap... U
- ↳ app.component.htm... M

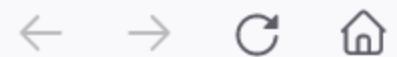
PRIMERO

- > acumulador ●
- ✓ dbz ●
- ✓ main-page ●
 - # main-page.compon... U
 - ↳ main-page.compon... U
 - TS main-page.compon... U
 - TS main-page.compon... U
 - TS dbz.module.ts U
- > heroes ●
- # app.component.css
- ↳ app.component.html M

```
src > app > dbz > TS dbz.module.ts > DbzModule
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { MainPageComponent } from './main-page/main-page.component';
4
5
6
7 @NgModule({
8   declarations: [
9     MainPageComponent
10 ],
11   imports: [
12     CommonModule
13 ],
14   exports: [
15     MainPageComponent
16   ]
17 })
18 export class DbzModule { }
19
```

The screenshot shows a development environment in VS Code with the following interface elements:

- EXPLORADOR**: Shows a file tree with the following structure:
 - PRIMERO**:
 - > acumulador
 - ✓ dbz
 - ✓ main-page
 - # main-page.component.html
 - ↳ main-page.component.ts
 - TS main-page.component.css
 - TS main-page.component.spec.ts
 - > heroes
 - # app.component.css
 - ↳ app.component.html
 - TS app.component.spec.ts
 - TS app.component.ts
 - EDITORES ABIERTOS**: Shows the contents of `app.component.html`. The line `<app-main-page></app-main-page>` is highlighted with a red box.
 - EDITOR**: Shows the code for `app.component.html`. The line `<app-main-page></app-main-page>` is highlighted with a red box.
 - SEARCH**: Shows the search results for `app.component.html`.



localhost:4200

Dragon Ball Z

Modificamos de esta manera

The image shows a screenshot of the Visual Studio Code (VS Code) interface. On the left, the Explorer sidebar displays various files and folders. In the center, the main editor area shows the content of the file `# styles.css`. A red box highlights the title bar of the editor tab, and another red box highlights the entire code block.

```
padding: 1px;
}
.row{
  display: flex;
}
.col{
  flex-grow: 1;
  margin-right: 10px;
}
input{
  display: block;
  margin: 5px;
}
```

main-page.component.html U X

```
c > app > dbz > main-page > main-page.component.html > ...
    Go to component
1   <h1>Dragon Ball Z</h1>
2   <hr>
3
4   <div class="row">
5
6       <div class="col">
7           <h3>Personajes</h3>
8           <hr>
9           <ul>
10          <li>Krilin - 1000</li>
11          <li>Gohan - 1500</li>
12          <li>Goku - 20.000</li>
13      </ul>
14  </div>
15
16  <div class="col">
17      <h3>Agregar</h3>
18      <hr>
19
20      <form>
21          <input type="text" placeholder="Nombre">
22          <input type="text" placeholder="Poder">
23          <button>Agregar</button>
24      </form>
25  </div>
26 </div>
27
```

Dragon Ball Z

Personajes

- Krilin - 1000
- Gohan - 1500
- Goku - 20.000

Agregar

Nombre

Poder

Agregar

EXPLORADOR ... <> main-page.component.html U TS main-page.component.ts U X

> EDITORES ABIERTOS

PRIMERO

> .angular

> node_modules

src

> app

> acumulador

> dbz

> main-page

main-page.component.css U

<> main-page.component.html U

TS main-page.component.spec.ts U

TS main-page.component.ts U

TS dbz.module.ts U

> heroes

app.component.css

<> app.component.html M

TS app.component.spec.ts

TS app.component.ts M

TS dbz.module.spec.ts M

src > app > dbz > main-page > TS main-page.component.ts > MainPa

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-main-page',
5   templateUrl: './main-page.component.html',
6   styleUrls: ['./main-page.component.css']
7 })
8 export class MainPageComponent {
9
10 agregar(){
11   console.log("envio formulario");
12 }
13
14 }
15
```



Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda

main-page.component.html - primero - Visual Studio Code [A]



EXPLORADOR

...

main-page.component.html U X

TS main-page.component.ts U

TS dbz.module.ts U

> EDITORES ABIERTOS



PRIMERO



> .angular

> node_modules



src



> app

> acumulador



> dbz



> main-page



main-page.component.css U

<> main-page.component.html U

TS main-page.component.spec.ts U

TS main-page.component.ts U

TS dbz.module.ts U

> heroes

app.component.css

<> app.component.html M

TS app.component.spec.ts

TS app.component.ts M

TS app.module.ts M

src > app > dbz > main-page > main-page.component.html > div.row > div.col

```
8   <hr>
9   <ul>
10  |   <li>Krillin - 1000</li>
11  |   <li>Gohan - 1500</li>
12  |   <li>Goku - 20.000</li>
13  |</ul>
14  </div>
15
16  <div class="col">
17  |   <h3>Agregar</h3>
18  |   <hr>
19
20  <form (ngSubmit)="agregar()">
21  |   <input type="text" placeholder="Nombre">
22  |   <input type="text" placeholder="Poder">
23  |   <button type="submit">Agregar</button>
24  </form>
25  </div>
26  </div>
27
```

FormsModule

- Es un modulo para manejar formulario en angular, para utilizarlo debemos invocarlo en el modulo del componente

Importamos el FormsModule

```
<> main-page.component.html U      TS dbz.module.ts U X

src > app > dbz > TS dbz.module.ts > DbzModule
1   import { NgModule } from '@angular/core';
2   import { CommonModule } from '@angular/common';
3   import { MainPageComponent } from './main-page/main-page.component';
4   import { FormsModule } from '@angular/forms';
5
6
7
8   @NgModule({
9     declarations: [
10       MainPageComponent
11     ],
12     imports: [
13       CommonModule,
14       FormsModule
15     ],
16     exports: [
17       MainPageComponent
18     ]
19   })
20   export class DbzModule { }
```

Una vez implementado el modulo FormsModule no debe refrescar al llamar al submit

The screenshot shows a web application running at `localhost:4200`. The title bar says "Dragon Ball Z". On the left, there's a sidebar titled "Personajes" with a list:

- Krilin - 1000
- Gohan - 1500
- Goku - 20.000

On the right, there's a form titled "Agregar" with fields for "Nombre" and "Poder", and a "Agregar" button.

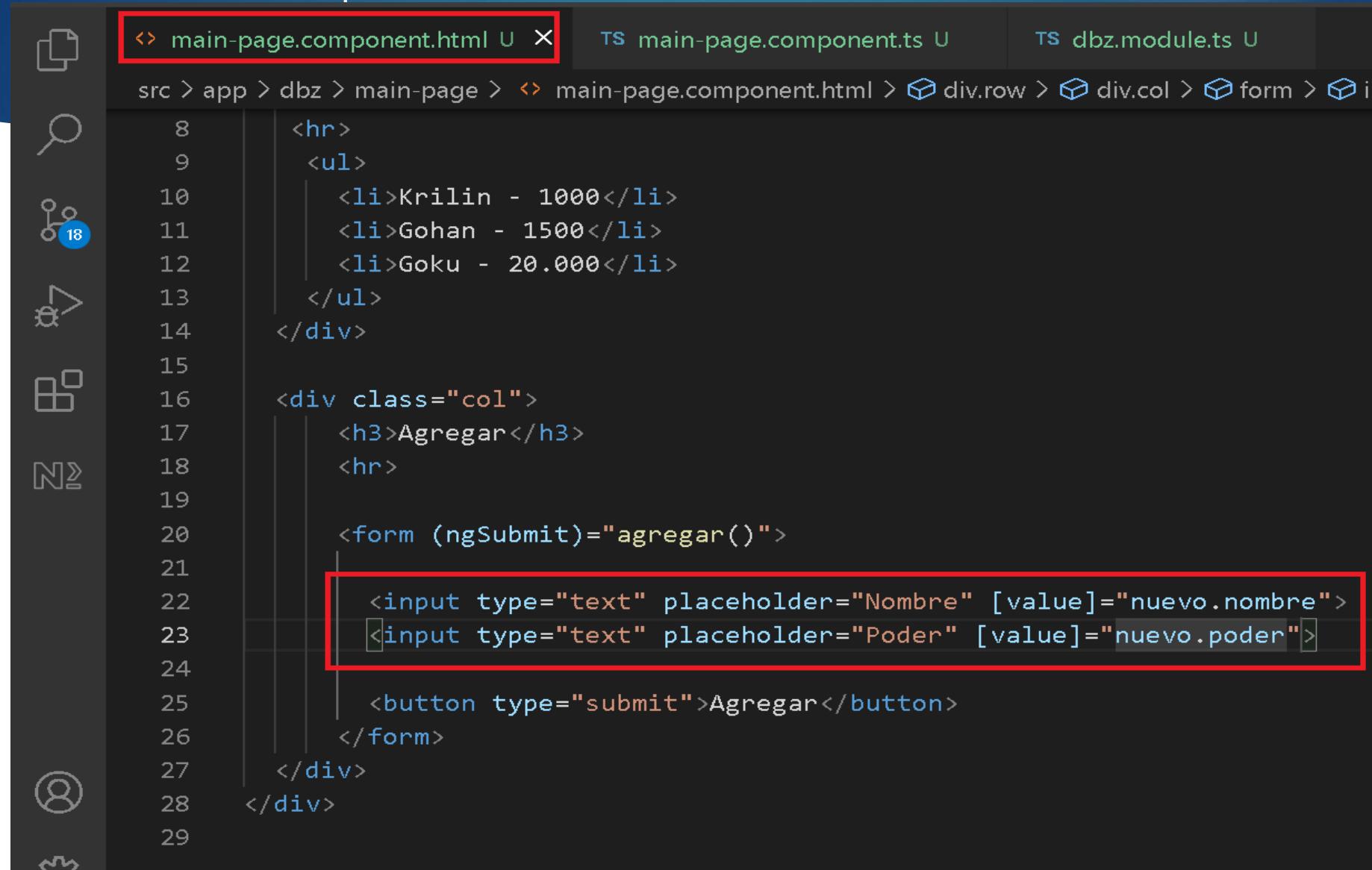
At the bottom, the browser's developer tools "Consola" tab is active, showing the following logs:

- [webpack-dev-server] Disconnected! index.j
- [webpack-dev-server] Trying to reconnect... index.j
- [webpack-dev-server] Live Reloading enabled. index.j
- Angular is running in development mode. Call enableProdMode() to enable production mode. core.mjs:248
- envio formulario main-page.component.ts:6

Continuamos con el formulario

```
↳ main-page.component.html U TS main-page.component.ts U X TS dbz.module.ts U
src > app > dbz > main-page > TS main-page.component.ts > MainPageComponent > agregar
1   import { Component } from '@angular/core';
2
3   interface Personaje{
4     nombre:string;
5     poder:number;
6   }
7
8   @Component({
9     selector: 'app-main-page',
10    templateUrl: './main-page.component.html',
11    styleUrls: ['./main-page.component.css']
12  })
13  export class MainPageComponent {
14
15    nuevo: Personaje ={
16      nombre:"Vegeta",
17      poder: 20000
18    }
19
20    agregar(){
21      console.log("envio formulario");
22    }
23
24  }
```

El [value] nos indica que queremos capturar un valor desde nuestra clase componente ts



The screenshot shows a code editor interface with a dark theme. On the left is a sidebar with various icons: a file, a magnifying glass, a circular icon with '18' (likely notifications), a search icon, a grid icon, a double-page icon, a user icon, and a network icon. The main area has tabs for 'main-page.component.html' (highlighted with a red box), 'main-page.component.ts', and 'dbz.module.ts'. The code in 'main-page.component.html' is:

```
src > app > dbz > main-page > main-page.component.html > div.row > div.col > form > input  
8   <hr>  
9   <ul>  
10  <li>Krilin - 1000</li>  
11  <li>Gohan - 1500</li>  
12  <li>Goku - 20.000</li>  
13  </ul>  
14 </div>  
15  
16 <div class="col">  
17   <h3>Agregar</h3>  
18   <hr>  
19  
20   <form (ngSubmit)="agregar()">  
21  
22     <input type="text" placeholder="Nombre" [value]="nuevo.nombre">  
23     <input type="text" placeholder="Poder" [value]="nuevo.poder">  
24  
25     <button type="submit">Agregar</button>  
26   </form>  
27 </div>  
28 </div>  
29
```

The code block from line 22 to 23 is highlighted with a red box. Both lines contain an 'input' tag with a 'placeholder' attribute and a '[value]' attribute set to 'nuevo.nombre' and 'nuevo.poder' respectively.

Dragon Ball Z

Personajes

- Krilin - 1000
- Gohan - 1500
- Goku - 20.000

Agregar

Agregar

main-page.component.html U TS main-page.component.ts U X dbz.module.ts U

src > app > dbz > main-page > TS main-page.component.ts > MainPageComponent > agregar

```
12      })
13  export class MainPageComponent {
14
15
16    nuevo: Personaje ={
17      nombre:"Vegeta",
18      poder: 20000
19    }
20
21    cambiarNombre( event:any) {
22      console.log(event);
23    }
24
25    agregar(){
26      console.log(this.nuevo);
27    }
28
29  }
30
```

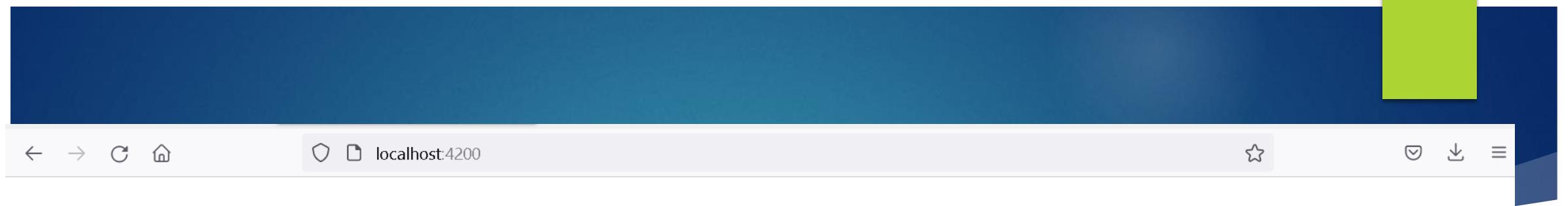
main-page.component.html U X

main-page.component.ts U

dbz.module.ts U

src > app > dbz > main-page > main-page.component.html > div.row > div.col > form >

```
16    <div class="col">
17        <h3>Agregar</h3>
18        <hr>
19
20        <form (ngSubmit)="agregar()">
21
22            <input
23                type="text"
24                placeholder="Nombre"
25                [value]="nuevo.nombre"
26                (input)="cambiarNombre( $event )"
27            >
28
29            <input type="text" placeholder="Poder" [value]="nuevo.poder">
30
31            <button type="submit">Agregar</button>
32        </form>
33    </div>
34 </div>
```



Dragon Ball Z

Personajes

- Krilin - 1000
- Gohan - 1500
- Goku - 20.000

Agregar

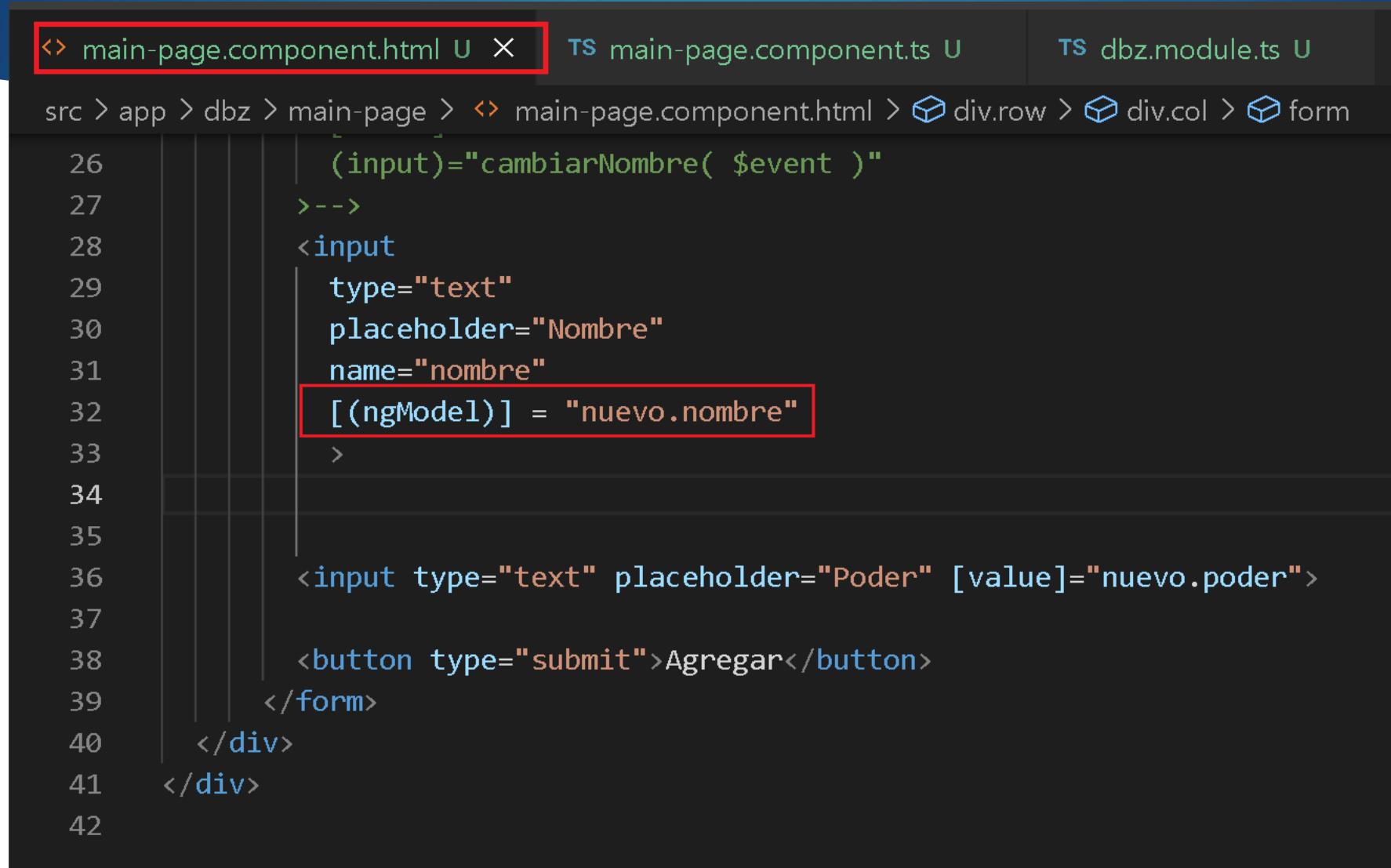
Agregar

Inspector Consola Depurador Red Editor de estilos Rendimiento Memoria Almacenamiento ...

Filtrar salida Errores Advertencias Registros Información Depurar CSS XHR Peticiones

```
▶ Object { nombre: "Vegeta", poder: 20000 } main-page.component.ts:26:10
▶ input { target: input, isTrusted: true, isComposing: false, inputType: "insertText", data: "w", view: Window, detail: 0, layerX: 0, layerY: 0, which: 0, ... }
▶ input { target: input, isTrusted: true, isComposing: false, inputType: "insertText", data: "e", view: Window, detail: 0, layerX: 0, layerY: 0, which: 0, ... }
```

ng Model : se trata de un enlace, entre algo que tienes en la definición del componente con un campo de formulario del template (vista) del componente.



The screenshot shows a code editor with two tabs open: 'main-page.component.html' and 'main-page.component.ts'. The 'main-page.component.html' tab is active and contains the following code:

```
src > app > dbz > main-page > main-page.component.html > div.row > div.col > form
26      (input)="cambiarNombre( $event )"
27      >-->
28      <input
29          type="text"
30          placeholder="Nombre"
31          name="nombre"
32          [(ngModel)] = "nuevo.nombre"
33      >
34
35
36      <input type="text" placeholder="Poder" [value]="nuevo.poder">
37
38      <button type="submit">Agregar</button>
39
40  </div>
41 </div>
42
```

The line '[(ngModel)] = "nuevo.nombre"' is highlighted with a red box. The 'main-page.component.ts' tab is also visible, showing the corresponding TypeScript code for the component.

Ya no necesitamos la función cambiarNombre

```
<> main-page.component.html U    TS main-page.component.ts U X    TS dbz.module.ts U
src > app > dbz > main-page > TS main-page.component.ts > MainPageComponent > agregar

12      })
13  export class MainPageComponent {
14
15
16  nuevo: Personaje ={
17    nombre:"Vegeta",
18    poder: 20000
19  }
20
21  /*cambiarNombre( event:any) {
22  |  console.log(event);
23  }*/
```

main-page.component.html U X

main-page.component.ts U

dbz.module.ts U

src > app > dbz > main-page > main-page.component.html > div.row > div.col > form > input

```
26          (input)="cambiarNombre( $event )"
27          >-->
28          <input
29              type="text"
30              placeholder="Nombre"
31              name="nombre"
32              [(ngModel)] = "nuevo.nombre"
33          >
34          <input
35              type="text"
36              placeholder="Poder"
37              name="poder"
38              [(ngModel)] = "nuevo.poder"
39          >
40
41          <button type="submit">Agregar</button>
42      </form>
43  </div>
44 </div>
```

Dragon Ball Z

Personajes

- Krilin - 1000
- Gohan - 1500
- Goku - 20.000

Agregar

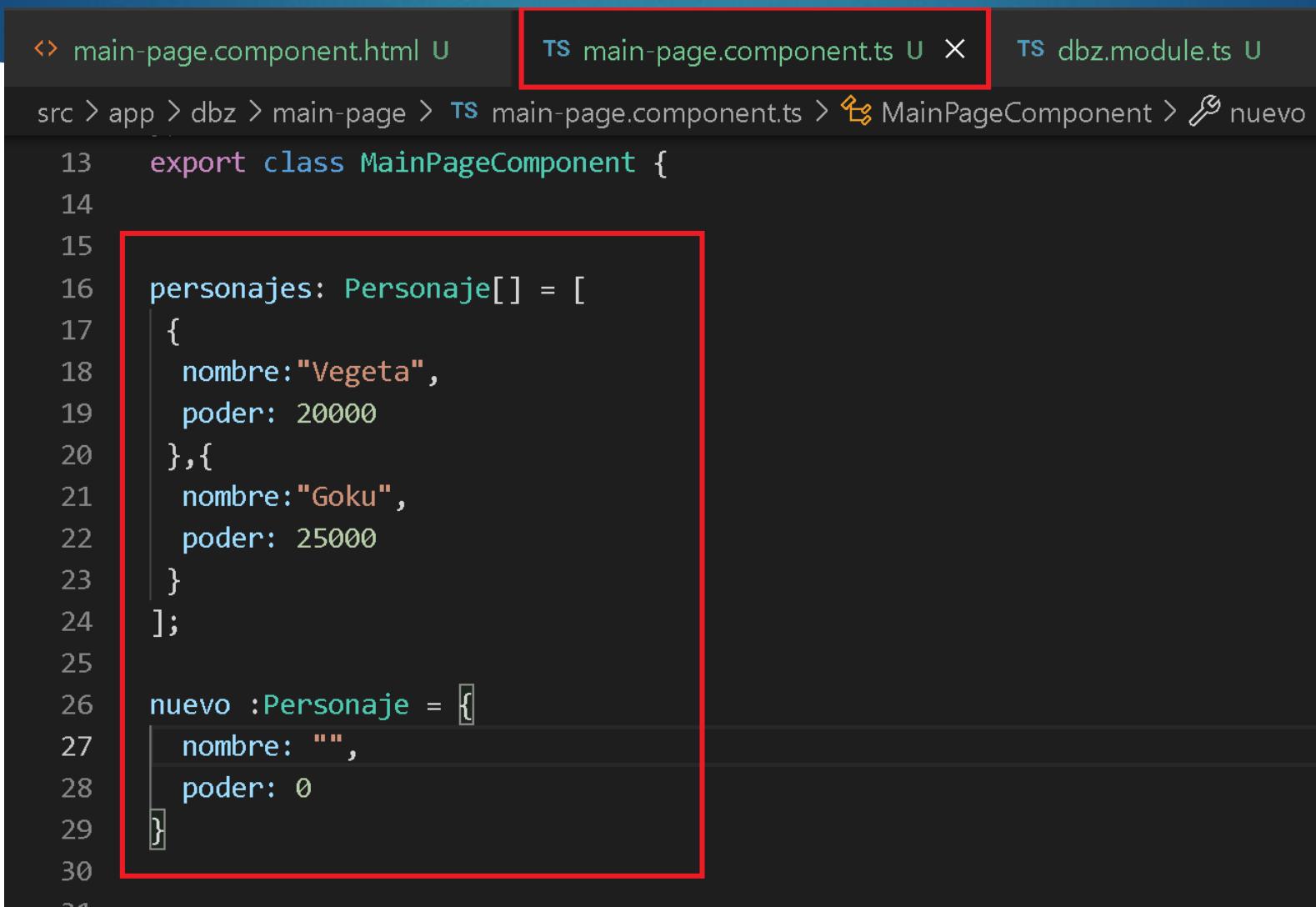
Agregar

Inspector Consola Depurador Red Editor de estilos Rendimiento Memoria Almacenamiento ...

Filtrar salida Errores Advertencias Registros Información Depurar CSS XHR Peticiones

(i) [webpack-dev-server] Trying to reconnect... index.js:548
(i) [webpack-dev-server] Live Reloading enabled. index.js:548
Angular is running in development mode. Call enableProdMode() to enable production mode. core.mjs:24856:16
▶ Object { nombre: "Vegeta", poder: 20000 } main-page.component.ts:26:10
▶ Object { nombre: "Trunks", poder: "32000" } main-page.component.ts:26:10

En main-page.component.ts agregamos las siguientes lineas



The screenshot shows a code editor with three tabs: main-page.component.html, main-page.component.ts (which is the active tab), and dbz.module.ts. The main-page.component.ts tab has a red border around it. The file content is as follows:

```
src > app > dbz > main-page > main-page.component.ts > MainPageComponent > nuevo

13  export class MainPageComponent {
14
15
16    personajes: Personaje[] = [
17      {
18        nombre: "Vegeta",
19        poder: 20000
20      },
21      {
22        nombre: "Goku",
23        poder: 25000
24      }
25    ];
26
27    nuevo :Personaje = {
28      nombre: "",
29      poder: 0
30    };
31
```

A red box highlights the character declarations starting from line 16, specifically the array assignment and the two character objects.

main-page.component.html U

TS main-page.component.ts U X

TS dbz.module.ts U

src > app > dbz > main-page > TS main-page.component.ts > MainPageComponent > nuevo

29 }

30

31

```
32 agregar(){
33     if(this.nuevo.nombre.trim().length === 0){
34         return;
35     }
36     console.log(this.nuevo);
37     this.personajes.push(this.nuevo);
38     this.nuevo = {
39         nombre: '',
40         poder: 0
41     }
42 }
```

43

44 }

45

También agregamos esto en la lista de personajes dentro
de main-page.component.html

```
  <ul>
    <li *ngFor="let item of personajes">
      {{item.nombre}} - {{item.poder}}
    </li>
  </ul>
```

Probamos si podemos agregar personajes

The screenshot shows a web application interface for managing characters from Dragon Ball Z. The title bar indicates the URL is `localhost:4200`.

The main content area has two sections:

- Personajes**: A list of existing characters with their names and power levels:
 - Vegeta - 20000
 - Goku - 25000
 - Maestro Roshi - 1000
 - Piccolo - 12000
- Agregar**: A form for adding new characters. It contains a text input field labeled "Nombre" with the value "0", and a button labeled "Agregar".

At the bottom of the screen is a developer tools console. The tabs at the top of the console are Inspector, Consola (which is selected), Depurador, Red, Editor de estilos, Rendimiento, Memoria, Almacenamiento, and others. The console output shows the following messages:

- [webpack-dev-server] Trying to reconnect... index.js
- [webpack-dev-server] Live Reloading enabled. index.js
- Angular is running in development mode. Call enableProdMode() to enable production mode. core.mjs:248
- ▶ Object { nombre: "Maestro Roshi", poder: "1000" } main-page.component.ts:
- ▶ Object { nombre: "Piccolo", poder: "12000" } main-page.component.ts:

Componentes Hijos

```
PS C:\Users\dell\Desktop\pruebas\primero> ng g c dbz/personajes --skipTests
Support for camel case arguments has been deprecated and will be removed in a future major version.
Use '--skip-tests' instead of '--skipTests'.
CREATE src/app/dbz/personajes/personajes.component.html (25 bytes)
CREATE src/app/dbz/personajes/personajes.component.ts (291 bytes)
CREATE src/app/dbz/personajes/personajes.component.css (0 bytes)
UPDATE src/app/dbz/dbz.module.ts (484 bytes)
PS C:\Users\dell\Desktop\pruebas\primero>
```

EXPLORADOR

> EDITORES ABIERTOS

▼ PRIMERO

< app

> acumulador

< dbz

< main-page

main-page.component.css

<> main-page.component.html

TS main-page.component.spec.ts

TS main-page.component.ts

< personajes

personajes.component.css

<> personajes.component.html

TS personajes.component.ts

TS dbz.module.ts

> heroes

app.component.css

<> app.component.html

TS app.component.spec.ts

...

<> main-page.component.html U X

src > app > dbz > main-page > <> main-page.component.html > div.row > div.col >

2 <hr>

3

4 <div class="row">

5

6 <div class="col">

7 <h3>Personajes</h3>

8 <hr>

9 <app-personajes>[</app-personajes>

10 <!--

11 <li *ngFor="let item of personajes">

12 | {{item.nombre}} - {{item.poder}}

13 |

14 -->

15 </div>

16 <div class="col">

17 <h3>Agregar</h3>

18 <hr>

19

20

21

Dragon Ball Z

Personajes

personajes works!

Agregar

Nombre

0

Agregar

Realizamos el intercambio

```
main-page.component.html
src > app > dbz > main-page > main-page.component.html > div.row
2   <hr>
3
4   <div class="row">
5
6     <div class="col">
7       <h3>Personajes</h3>
8       <hr>
9       <app-personajes></app-personajes>
10      <!--<ul>
11
12        <li *ngFor="let item of personajes">
13          {{item.nombre}} - {{item.poder}}
14        </li>
15      </ul>-->
16    </div>
17
18    <div class="col">
19      <h3>Agregar</h3>
20      <hr>
```

```
personajes.component.html
src > app > dbz > personajes > personajes.component.html > .
Go to component
1   <p>personajes works!</p>
2
```

```
<> main-page.component.html U X
> app > dbz > main-page > <> main-page.component.html > div.row > div.col
  2   <hr>
  3
  4   <div class="row">
  5
  6     <div class="col">
  7       <h3>Personajes</h3>
  8       <hr>
  9       <app-personajes></app-personajes>
 10
 11   </div>
 12
 13   <div class="col">
 14     <h3>Agregar</h3>
 15     <hr>
 16
 17     <form (ngSubmit)="agregar()">
 18
 19       <input
 20         type="text"
 21         placeholder="Nombre"
```



```
<> personajes.component.html 1, U X
```

```
src > app > dbz > personajes > <> personajes.component.html > ul
  Go to component
  1   <ul>
  2
  3     <li *ngFor="let item of personajes">
  4       {{item.nombre}} - {{item.poder}}
  5     </li>
  6   </ul>
  7
```

TS personajes.component.ts U

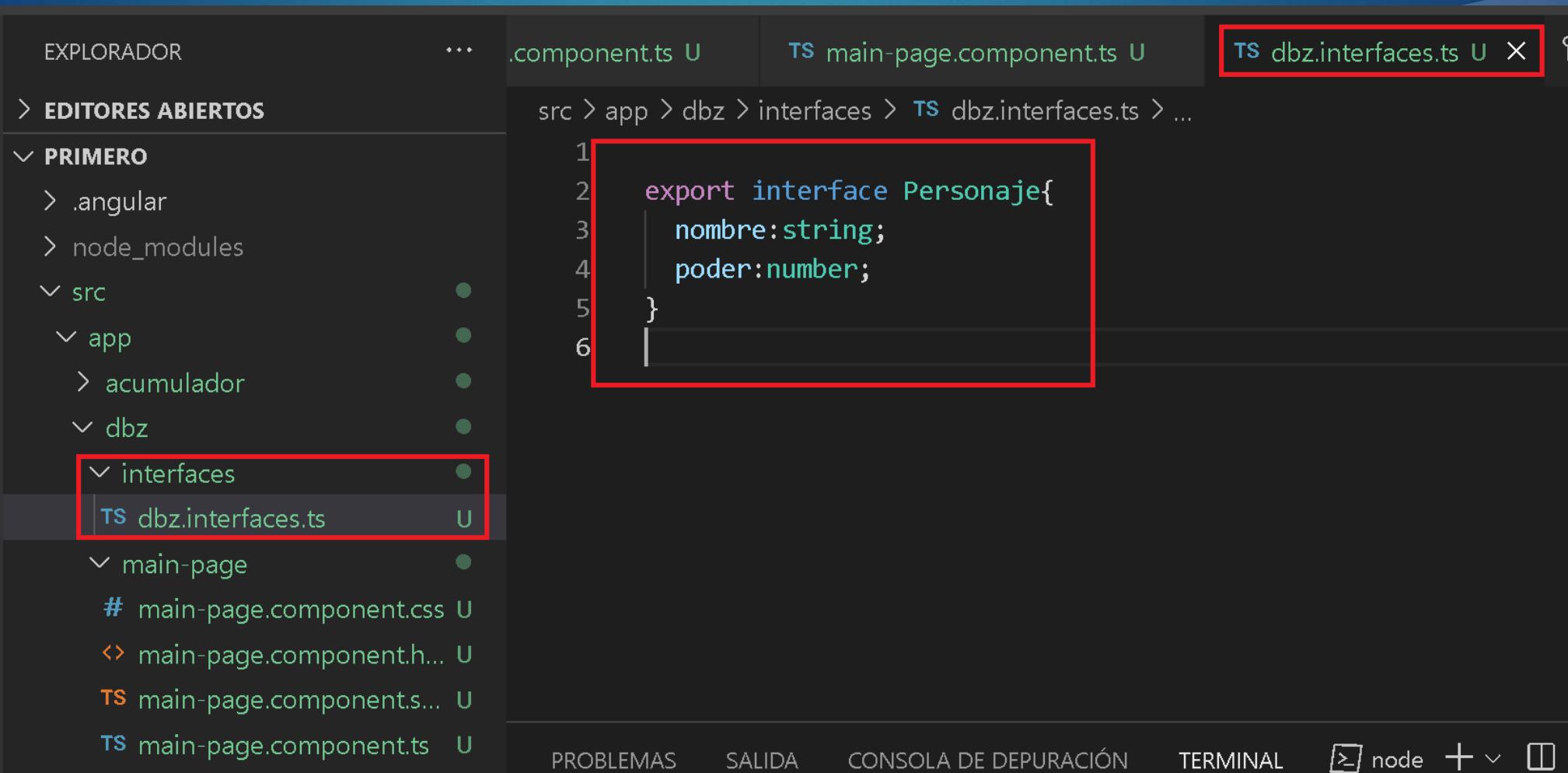
↔ main-page.component.html U X

TS main-page.component.ts U

src > app > dbz > main-page > ↔ main-page.component.html > 📁 div.row

```
4   <div class="row">
5
6     <div class="col">
7       <h3>Personajes</h3>
8       <hr>
9       <app-personajes [personajes]="personajes"></app-personajes>
10
11    </div>
12
13    <div class="col">
14      <h3>Agregar</h3>
15      <hr>
16
17      <form (ngSubmit)="agregar()">
18
```

Cambiamos de lugar nuestra interface

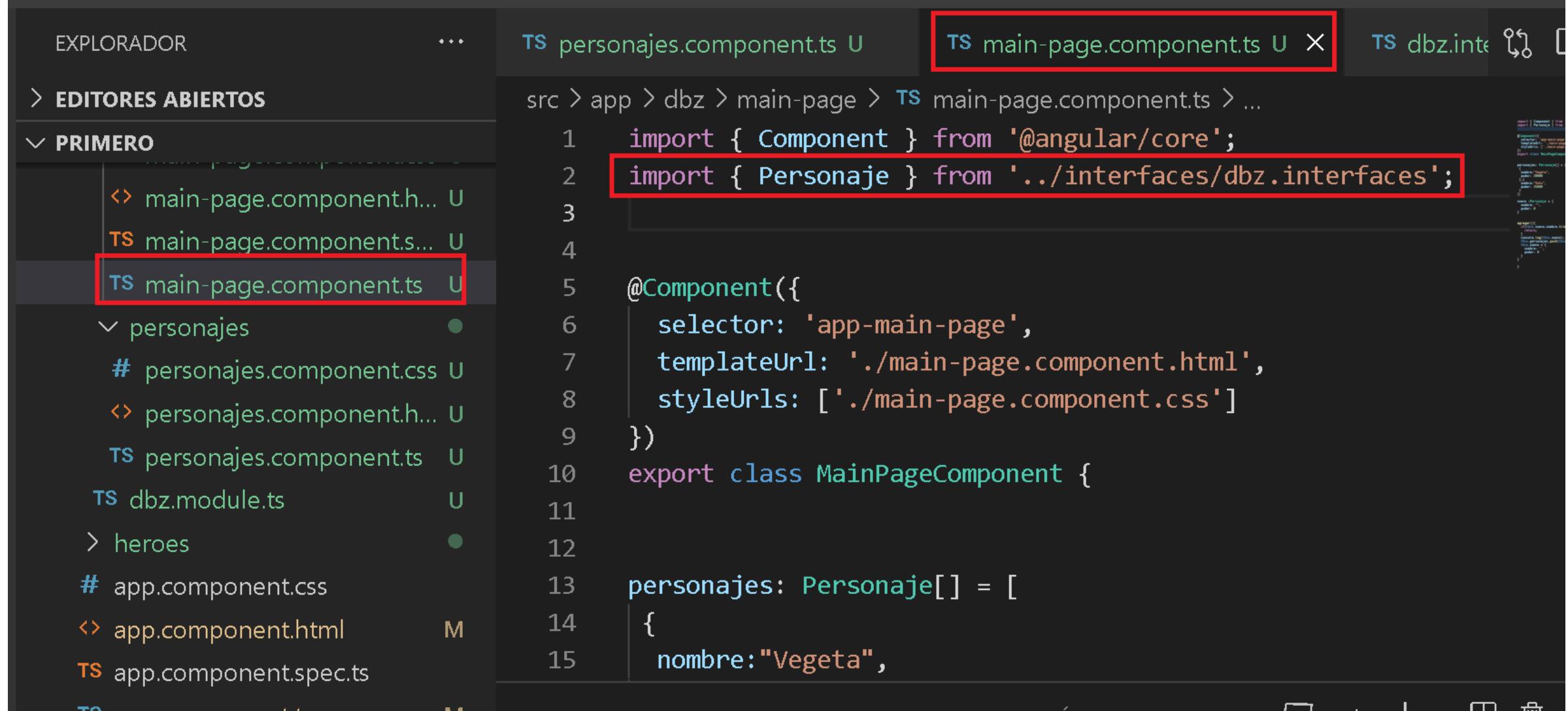


The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORADOR**: Shows the project structure:
 - > EDITORES ABIERTOS
 - ✓ PRIMERO
 - > .angular
 - > node_modules
 - ✓ src
 - ✓ app
 - > acumulador
 - ✓ dbz
 - ✓ interfaces
 - ✓ dbz.interfaces.ts
 - ✓ main-page
 - # main-page.component.css
 - ↳ main-page.component.h...
 - TS main-page.components.s...
 - TS main-page.component.ts
 - EDITOR**: The file `dbz.interfaces.ts` is open, showing the following code:

```
1 export interface Personaje{  
2     nombre:string;  
3     poder:number;  
4 }  
5  
6 
```
 - PESTANAS**: Other tabs visible include `.component.ts`, `main-page.component.ts`, and `dbz.interfaces.ts` (which is active).
 - FOOTER**: Standard VS Code footer with links to PROBLEMAS, SALIDA, CONSOLA DE DEPURACIÓN, TERMINAL, and a node.js icon.

Con este cambio importamos la interfaz en main-page.component.ts



The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORADOR**: Shows the project structure with files like main-page.component.hbs, main-page.component.ts, and dbz.module.ts.
- EDITORES ABIERTOS**: Shows multiple tabs: personajes.component.ts, main-page.component.ts (highlighted with a red box), and dbz.interfaces.ts.
- PRIMERO**: Shows the main-page.component.ts file content:

```
1 import { Component } from '@angular/core';
2 import { Personaje } from '../interfaces/dbz.interfaces';
3
4
5 @Component({
6   selector: 'app-main-page',
7   templateUrl: './main-page.component.html',
8   styleUrls: ['./main-page.component.css']
9 })
10 export class MainPageComponent {
11
12
13   personajes: Personaje[] = [
14     {
15       nombre: "Vegeta",
16       poder: 1000,
17       color: "blue"
18     },
19     {
20       nombre: "Goku",
21       poder: 1500,
22       color: "red"
23     }
24   ]
25 }
```

Con @input hacemos referencia a un valor del componente padre

```
TS personajes.component.ts U X  ↳ main-page.component.html U TS main-page
src > app > dbz > personajes > TS personajes.component.ts > ...
1 import { Component, Input } from '@angular/core';
2
3 @Component({
4   selector: 'app-personajes',
5   templateUrl: './personajes.component.html',
6   styleUrls: ['./personajes.component.css']
7 })
8 export class PersonajesComponent {
9
10 @Input() personajes:any[] = [];
11
12 }
13
```

Con esto probamos que funciona todo como debería teniendo diferentes componentes para ello



Personajes

- Vegeta - 20000
- Goku - 25000
- Yamcha - 12
- Yayirobe - 500

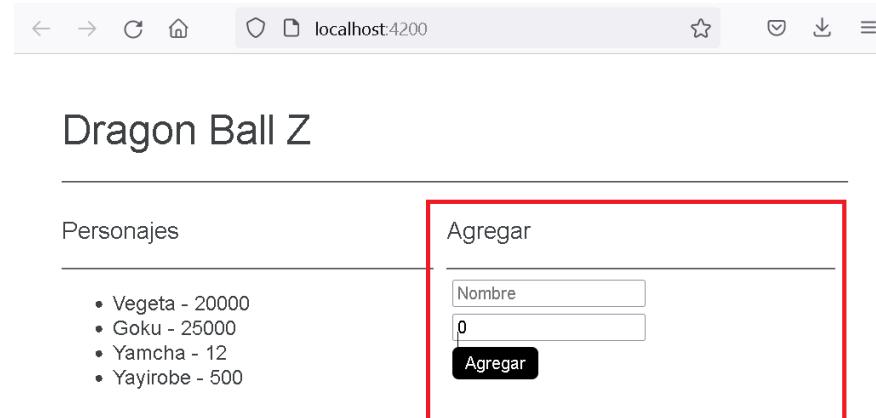
Agregar

Nombre

Agregar

Ejercicio

- ▶ Pasar el formulario a un componente hijo llamado agregar siguiendo los pasos anteriores y probar que todo funcione



```
PS C:\Users\dell\Desktop\pruebas\primero> ng g c dbz/agregar --skipTests
Support for camel case arguments has been deprecated and will be removed in a
Use '--skip-tests' instead of '--skipTests'.
CREATE src/app/dbz/agregar/agregar.component.html (22 bytes)
CREATE src/app/dbz/agregar/agregar.component.ts (279 bytes)
CREATE src/app/dbz/agregar/agregar.component.css (0 bytes)
UPDATE src/app/dbz/dbz.module.ts (570 bytes)
PS C:\Users\dell\Desktop\pruebas\primero> █
```

main-page.component.html U X

```
src > app > dbz > main-page > main-page.component.html > div.row > div.col > app-agregar
      Go to component
1   <h1>Dragon Ball Z</h1>
2   <hr>
3
4   <div class="row">
5
6     <div class="col">
7       <h3>Personajes</h3>
8       <hr>
9       <app-personajes [personajes]="personajes"></app-personajes>
10
11    </div>
12
13    <div class="col">
14      <app-agregar></app-agregar>
15    </div>
16  </div>
17
```

agregar.component.html 5, U X

```
src > app > dbz > agregar > agregar.component.html > form
      Go to component
1   <h3>Agregar</h3>
2   <hr>
3
4   <form (ngSubmit)="agregar()">
5
6     <input
7       type="text"
8       placeholder="Nombre"
9       name="nombre"
10      [(ngModel)] = "nuevo.nombre"
11
12     <input
13       type="text"
14       placeholder="Poder"
15       name="poder"
16      [(ngModel)] = "nuevo.poder"
17
18     <button type="submit">Agregar</button>
19
20 </form>
21
```

```
src > app > dbz > main-page > main-page.component.ts > MainPageComponent
17 },{
18   nombre: "Goku",
19   poder: 25000
20 }
21 ];
22
23 nuevo :Personaje = {
24   nombre: "",
25   poder: 0
26 }
27
28
29 agregar(){
30   if(this.nuevo.nombre.trim().length === 0){
31     return;
32   }
33   console.log(this.nuevo);
34   this.personajes.push(this.nuevo);
35   this.nuevo = {
36     nombre: '',
37     poder: 0
38   }
39 }
40
41 }
42
```

```
src > app > dbz > agregar > agregar.component.ts > AgregarComponent
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-agregar',
5   templateUrl: './agregar.component.html',
6   styleUrls: ['./agregar.component.css']
7 })
8 export class AgregarComponent {
9
10
11
12 }
```

ts agregar.component.ts U X

src > app > dbz > agregar > ts agregar.component.ts > AgregarComponent > personajes

```
1 import { Component, Input } from '@angular/core';
2 import { Personaje } from '../interfaces/dbz.interfaces';
3
4 @Component({
5   selector: 'app-agregar',
6   templateUrl: './agregar.component.html',
7   styleUrls: ['./agregar.component.css']
8 })
9 export class AgregarComponent {
10
11   @Input() personajes:Personaje[] = [];
12
13   @Input() nuevo :Personaje = {
14     nombre: '',
15     poder: 0
16   }
17
18
19   agregar(){
20     if(this.nuevo.nombre.trim().length === 0){
21       return;
22     }
23     console.log(this.nuevo);
24     this.personajes.push(this.nuevo);
25     this.nuevo = {
26       nombre: '',
27       poder: 0
28     }
29   }
30
31
32 }
```

agregar.component.ts U

main-page.component.html U X

: > app > dbz > main-page > main-page.component.html > div.row > div.col

Go to component

```
1  <h1>Dragon Ball Z</h1>
2  <hr>
3
4  <div class="row">
5
6      <div class="col">
7          <h3>Personajes</h3>
8          <hr>
9          <app-personajes [personajes]="personajes"></app-personajes>
10
11     </div>
12
13     <div class="col">
14         <app-agregar [personajes]="personajes"></app-agregar>
15     </div>
16 </div>
17
```

@Output() nos sirve para emitir eventos desde el componente hijo al componente padre

```
TS agregar.component.ts U < main-page.component.html U TS main-page.component.ts U
src > app > dbz > agregar > TS agregar.component.ts > AgregarComponent > personajes
1 import { Component, Input, Output, EventEmitter } from '@angular/core';
2 import { Personaje } from '../interfaces/dbz.interfaces';
3
4 @Component({
5   selector: 'app-agregar',
6   templateUrl: './agregar.component.html',
7   styleUrls: ['./agregar.component.css']
8 })
9 export class AgregarComponent {
10
11   @Input() personajes:Personaje[] = [];
12
13   @Input() nuevo :Personaje = {
14     nombre: '',
15     poder: 0
16   }
17
18   @Output() onNuevoPersonaje: EventEmitter<Personaje> = new EventEmitter();
19
20   agregar(){
21     if(this.nuevo.nombre.trim().length === 0){
22       return;
23     }
24     console.log(this.nuevo);
25     this.personajes.push(this.nuevo);
26     this.onNuevoPersonaje.emit(this.nuevo);
27
28     this.nuevo = {
29       nombre: '',
30       poder: 0
31     }
32   }
33 }
```

agregar.component.ts U main-page.component.html U main-page.component.ts U X

src > app > dbz > main-page > main-page.component.ts > MainPageComponent

```
5  @Component({
6    selector: 'app-main-page',
7    templateUrl: './main-page.component.html',
8    styleUrls: ['./main-page.component.css']
9  })
10 export class MainPageComponent {
11
12
13   personajes: Personaje[] = [
14     {
15       nombre:"Vegeta",
16       poder: 20000
17     },
18     {
19       nombre:"Goku",
20       poder: 25000
21     }
22   ];
23
24   agregarNuevoPersonaje( personaje : Personaje){
25     console.log(personaje);
26   }
27
28 }
29
```

ts agregar.component.ts U

↳ main-page.component.html U X

ts main-page.component.ts U

src > app > dbz > main-page > ↳ main-page.component.html > div.row > div.col > app-agregar

Go to component

```
1  <h1>Dragon Ball Z</h1>
2  <hr>
3
4  <div class="row">
5
6      <div class="col">
7          <h3>Personajes</h3>
8          <hr>
9          <app-personajes [personajes]="personajes"></app-personajes>
10
11     </div>
12
13     <div class="col">
14
15         <app-agregar
16             [personajes]="personajes"
17             (onNuevoPersonaje) = "agregarNuevoPersonaje( $event [])">
18         </app-agregar>
19
20     </div>
21 </div>
22
```

Servicios

- ▶ Los servicios en angular son manejados de manera muy eficiente, es muy parecida a la manera de trabajar con un patrón de diseño singleton. Es como tener una clase que este de manera global en nuestra aplicación.

Creamos un directorio y un archivo para nuestro servicio

The screenshot shows the VS Code interface with the following details:

- EXPLORADOR**: Shows the project structure:
 - > EDITORES ABIERTOS
 - PRIMERO
 - > .angular
 - > node_modules
 - src
 - app
 - > acumulador
 - dbz
 - > agregar
 - > interfaces
 - main-page
 - # main-page.component...
 - <> main-page.component...
 - TS main-page.component...
 - TS main-page.component...
 - > personajes
 - services
 - TS dbz.service.ts
 - > heroes
- EDITOR**: The file `dbz.service.ts` is open and highlighted with a red border. It contains the following code:

```
1 import { Injectable } from '@angular/core';
2
3
4 @Injectable()
5 export class DbzService{
6
7     constructor(){
8         console.log('Servicio Iniciado');
9     }
10}
11
```
- STATUS BAR**: Shows the status bar with the text "src > app > dbz > services > dbz.service.ts > DbzService > constructor()".

Se añade el servicio al modulo

The screenshot shows a code editor interface with a dark theme. On the left is a sidebar titled 'EXPLORADOR' containing a tree view of project files. In the center is the code editor area with three tabs at the top: 'TS dbz.service.ts U', 'TS dbz.module.ts U X', and 'TS main-page.component.ts U'. The 'TS dbz.module.ts U' tab is active and highlighted with a red border. The code in the editor is as follows:

```
src > app > dbz > TS dbz.module.ts > DbzModule
  5   import { PersonajesComponent } from './personajes/personajes.component';
  6   import { AgregarComponent } from './agregar/agregar.component';
  7   import { DbzService } from './services/dbz.service';
  8
  9
 10
 11 @NgModule({
 12   declarations: [
 13     MainPageComponent,
 14     PersonajesComponent,
 15     AgregarComponent
 16   ],
 17   imports: [
 18     CommonModule,
 19     FormsModule
 20   ],
 21   exports: [
 22     MainPageComponent
 23   ],
 24   providers: [
 25     DbzService
 26   ]
 27 })
 28 export class DbzModule { }
```

Two specific sections of the code are highlighted with red boxes: the import statement 'import { DbzService } from './services/dbz.service';' and the providers array 'providers: [DbzService]'. These highlights indicate the steps being demonstrated in the video.

El servicio es ejecutado cuando se es solicitado o creado una instancia, por medio de inyección de dependencia

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORADOR**: Shows the project structure:
 - > EDITORES ABIERTOS
 - < PRIMERO
 - > .angular
 - > node_modules
 - < src
 - < app
 - > acumulador
 - < dbz
 - > agregar
 - > interfaces
 - < main-page
 - # main-page.component... U
 - ↳ main-page.component... U
 - TS main-page.component... U
 - TS main-page.component... U
 - > personajes
 - < services
 - TS dbz.service.ts U
 - TS dbz.module.ts U
 - > heroes
 - > ESQUEMA
 - > LÍNEA DE TIEMPO
 - EDITORES ABIERTOS**: Shows the code editor with the file `main-page.component.ts` open. The file content is:

```
src > app > dbz > main-page > TS main-page.component.ts > MainPageComponent >
1   import { Component } from '@angular/core';
2   import { Personaje } from '../interfaces/dbz.interfaces';
3   import { DbzService } from '../services/dbz.service';
4
5
6   @Component({
7     selector: 'app-main-page',
8     templateUrl: './main-page.component.html',
9     styleUrls: ['./main-page.component.css']
10    })
11   export class MainPageComponent {
12
13     constructor(private dbzService:DbzService){
14
15   }
16
17     personajes: Personaje[] = [
18       {
19         nombre:"Vegeta",
20         poder: 20000
21       },{
22         nombre:"Goku",
23         poder: 25000
24     }
```
 - TERMINAL**: Not visible in the screenshot.

Dragon Ball Z

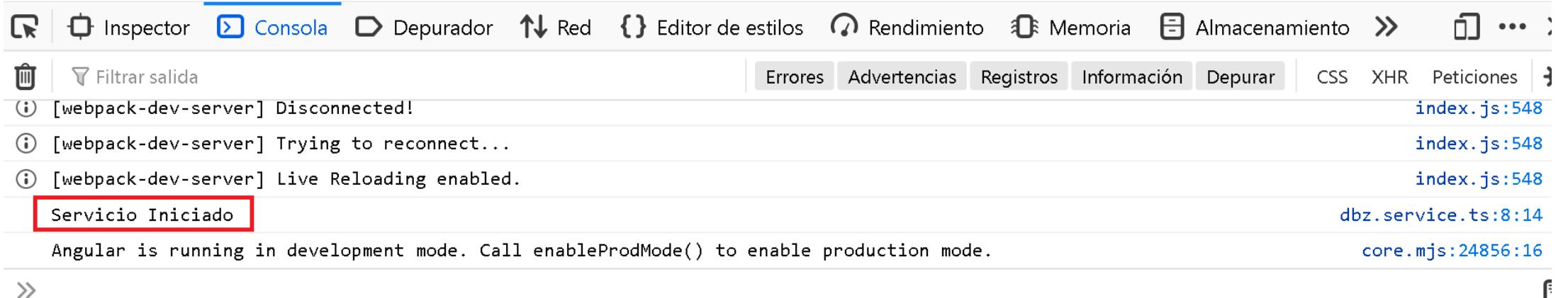
Personajes

- Vegeta - 20000
- Goku - 25000

Agregar

Nombre

Agregar



Inspector Consola Depurador Red Editor de estilos Rendimiento Memoria Almacenamiento ...

Filtrar salida Errores Advertencias Registros Información Depurar CSS XHR Peticiones

Disconnected! index.js:548

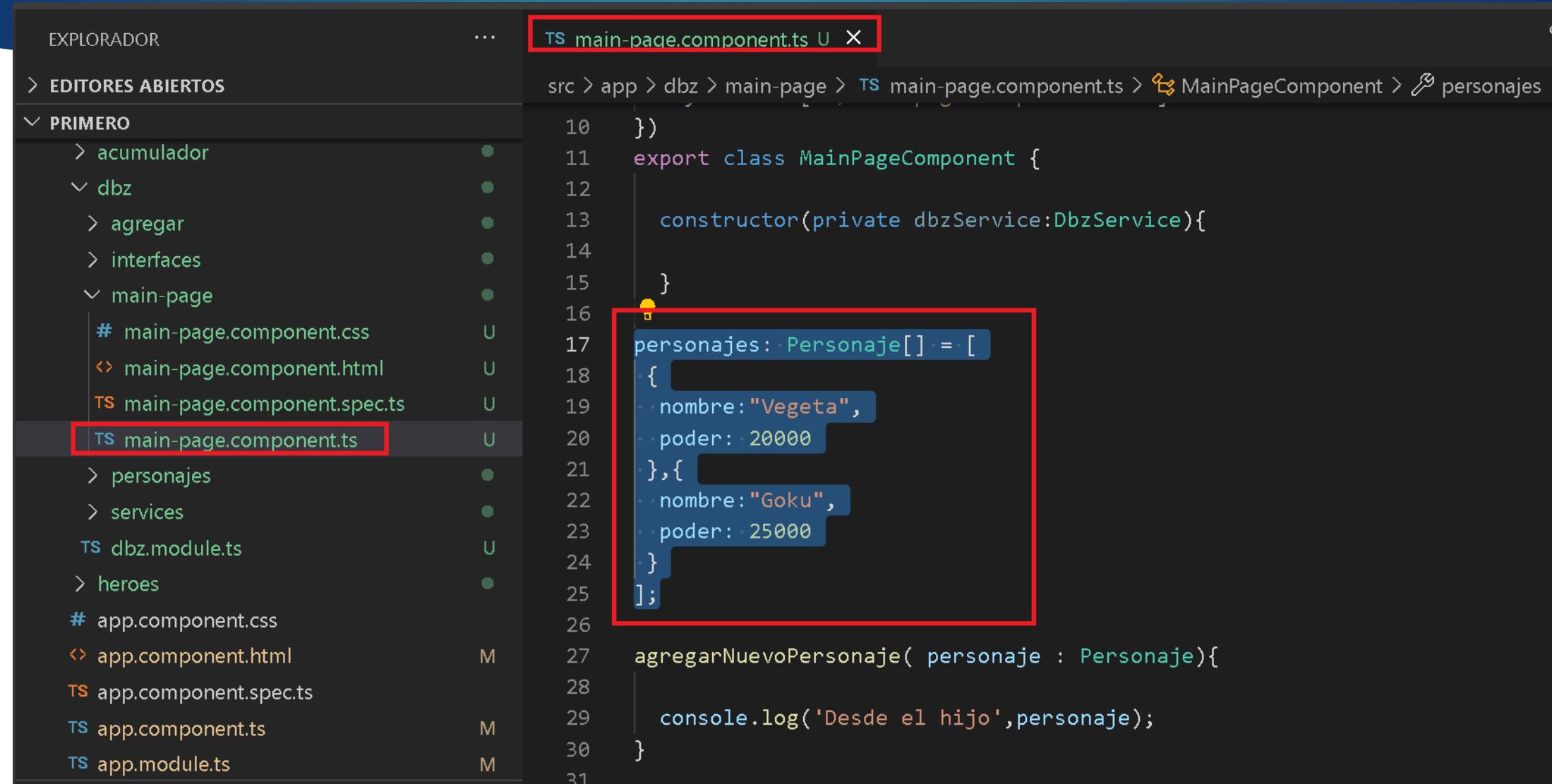
Trying to reconnect... index.js:548

Live Reloading enabled. index.js:548

Servicio Iniciado dbz.service.ts:8:14

Angular is running in development mode. Call enableProdMode() to enable production mode. core.mjs:24856:16

Centralizamos los datos en los servicios, pasamos nuestro array de personajes a nuestro servicio



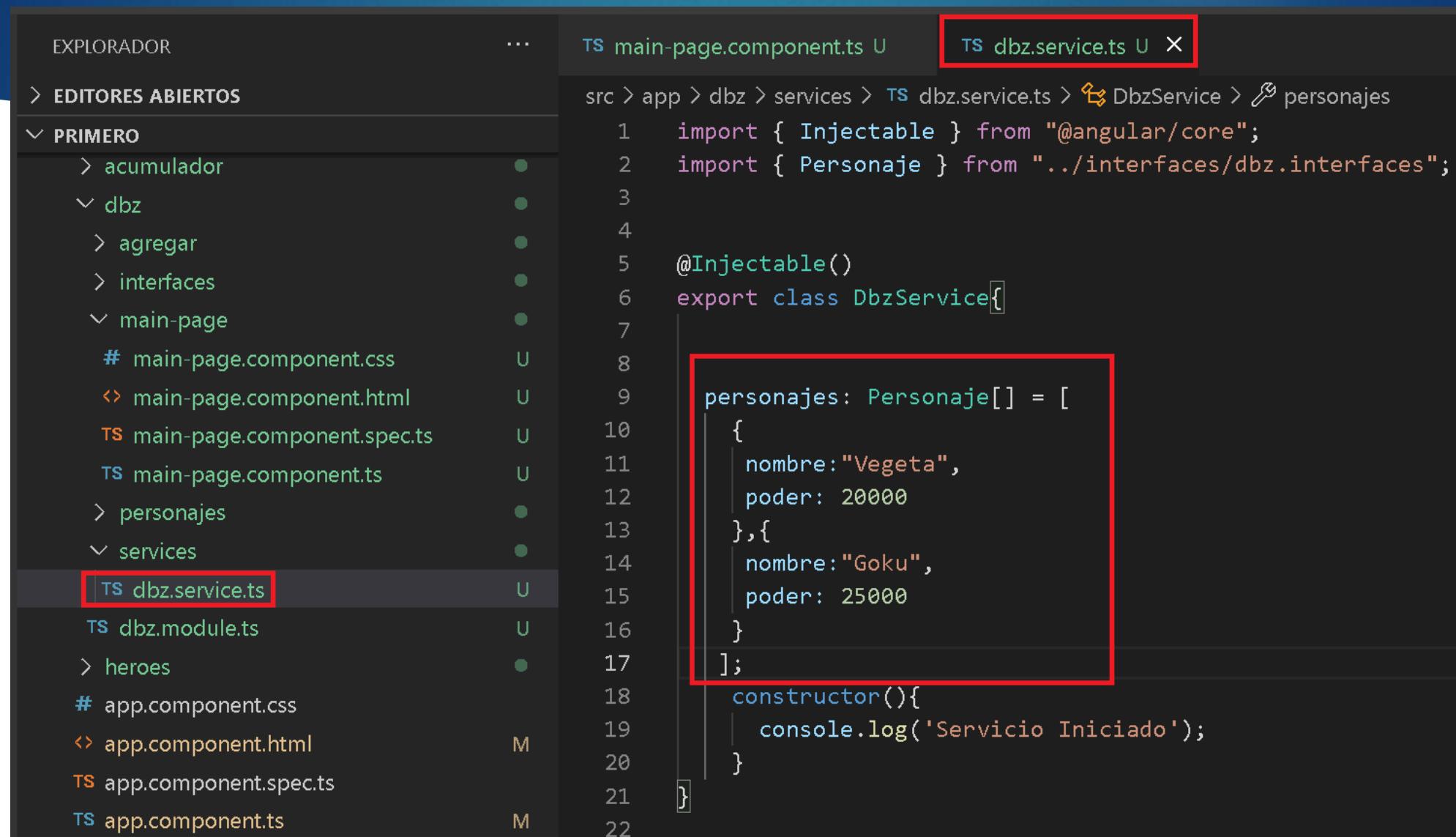
The image shows a screenshot of the Visual Studio Code (VS Code) interface. On the left, the Explorer sidebar displays a project structure under the 'PRIMERO' folder. The 'main-page' folder contains several files: main-page.component.css, main-page.component.html, main-page.component.spec.ts, main-page.component.ts, personajes, services, dbz.module.ts, heroes, app.component.css, app.component.html, app.component.spec.ts, app.component.ts, and app.module.ts. The 'main-page.component.ts' file is currently open in the editor.

The code in the editor is as follows:

```
src > app > dbz > main-page > TS main-page.component.ts > MainPageComponent > personajes
10  })
11  export class MainPageComponent {
12
13    constructor(private dbzService:DbzService){
14
15  }
16
17  personajes: Personaje[] = [
18  {
19    nombre: "Vegeta",
20    poder: 20000
21  },
22  {
23    nombre: "Goku",
24    poder: 25000
25  }
26
27  agregarNuevoPersonaje( personaje : Personaje){
28
29    console.log('Desde el hijo',personaje);
30  }
31
```

A red box highlights the line 'constructor(private dbzService:DbzService){' and the array initialization 'personajes: Personaje[] = ['. Another red box highlights the array elements, specifically the 'nombre' and 'poder' properties of the 'Vegeta' and 'Goku' objects.

Lo pasamos al servicio creado



EXPLORADOR

EDITORES ABIERTOS

PRIMERO

- acumulador
- dbz
- agregar
- interfaces
- main-page
 - main-page.component.css
 - main-page.component.html
 - main-page.component.spec.ts
 - main-page.component.ts
- personajes
- services
 - dbz.service.ts
 - dbz.module.ts
- heroes
- app.component.css
- app.component.html
- app.component.spec.ts
- app.component.ts

TS main-page.component.ts U TS dbz.service.ts U X

```
src > app > dbz > services > TS dbz.service.ts > DbzService > personajes
1 import { Injectable } from "@angular/core";
2 import { Personaje } from "../interfaces/dbz.interfaces";
3
4
5 @Injectable()
6 export class DbzService{
7
8
9   personajes: Personaje[] = [
10     {
11       nombre: "Vegeta",
12       poder: 20000
13     },
14     {
15       nombre: "Goku",
16       poder: 25000
17     }
18   ];
19   constructor(){
20     console.log('Servicio Iniciado');
21   }
22 }
```

Tendremos estos errores en main-page.component.html quitamos o comentamos el código que ya no utilizaremos

The screenshot shows a code editor interface with several tabs open. The current tab is 'main-page.component.html' (highlighted with a red border). The left sidebar shows a file tree with the following structure:

- > EDITORES ABIERTOS
- PRIMERO
 - > .angular
 - > node_modules
 - src
 - app
 - > acumulador
 - > dbz
 - > agregar
 - > interfaces
 - main-page
 - # main-page.... U
 - <> main-pa... 2, U
 - TS main-page.... U
 - TS main-page.... U
 - > personajes
 - services
 - TS dbz.service... U
 - TS dbz.module.ts U
 - > heroes

Quedaría así nuestro main-page.component.html

The screenshot shows the VS Code interface with the following details:

- EXPLORADOR**: Shows the project structure with files like .angular, node_modules, src, app, acumulador, dbz, agregar, interfaces, main-page, and services.
- EDITORES ABIERTOS**: Shows three tabs: main-page.component.ts, dbz.service.ts, and main-page.component.html. The main-page.component.html tab is active and highlighted with a red border.
- Content of main-page.component.html:**

```
src > app > dbz > main-page > main-page.component.html > div.row > div.col > app-agregar
4   <div class="row">
5
6   <div class="col">
7     <h3>Personajes</h3>
8     <hr>
9     <!--<app-personajes [personajes]="personajes"></app-personajes>-->
10    <app-personajes></app-personajes>
11
12   <div class="col">
13
14     <!--<app-agregar
15       [personajes]="personajes"
16       (onNuevoPersonaje) = "agregarNuevoPersonaje( $event )">
17     </app-agregar>-->
18
19     <app-agregar
20       (onNuevoPersonaje) = "agregarNuevoPersonaje( $event )">
21     </app-agregar>
22
23   </div>
24 </div>
```

Two specific sections of the code are highlighted with red boxes:
 - The first section highlights the opening and closing tags of the `<app-personajes>` component.
 - The second section highlights the opening and closing tags of the `<app-agregar>` component.

Modificamos el acceso a nuestros datos personajes a privado, también agregamos un get para poder acceder a ellos, los ... (tres puntos en el return) significa operador spread este separa cada uno de los elementos independiente que tiene este arreglo y crear uno nuevo, esto lo hacemos por default el get se hace por referencia y con esto evitamos la manipulación

The screenshot shows the VS Code interface with the Explorer, Editor, and Terminal panes visible. The Explorer pane shows the project structure with files like main-page.component.ts, dbz.service.ts, and dbz.module.ts. The Editor pane displays the dbz.service.ts file, which contains the following code:

```
src > app > dbz > services > dbz.service.ts > DbzService > (get) personajes
1 import { Injectable } from "@angular/core";
2 import { Personaje } from "../interfaces/dbz.interfaces";
3
4
5 @Injectable()
6 export class DbzService{
7
8     private _personajes: Personaje[] = [
9         {
10            nombre: "Vegeta",
11            poder: 20000
12        },
13        {
14            nombre: "Goku",
15            poder: 25000
16        }
17    ];
18
19    get personajes(): Personaje[] {
20        return [...this._personajes];
21    }
22
23    constructor(){
24        console.log('Servicio Iniciado');
25    }
26}
```

Two sections of the code are highlighted with red boxes: the private array declaration at lines 8-17 and the get personajes() method at lines 18-20.

En personajes.component.ts llamamos al servicio y devolvemos un get personajes

The screenshot shows the VS Code interface with the following details:

- EXPLORADOR**: Shows the project structure with files like main-page.component.ts, personajes.component.ts (selected), dbz.service.ts, and dbz.module.ts.
- EDITORES ABIERTOS**: Shows three open files: main-page.component.ts, personajes.component.ts (highlighted with a red border), and dbz.service.ts.
- Content of personajes.component.ts:**

```
1 import { Component, Input } from '@angular/core';
2 import { DbzService } from '../services/dbz.service';
3
4 @Component({
5   selector: 'app-personajes',
6   templateUrl: './personajes.component.html',
7   styleUrls: ['./personajes.component.css']
8 })
9 export class PersonajesComponent {
10   // @Input() personajes: any[] = [];
11
12   get personajes(){
13     return this.dbzService.personajes;
14   }
15
16   constructor(private dbzService:DbzService){
17
18   }
19 }
20
```

Annotations highlight specific parts of the code:

 - A red box surrounds the line `// @Input() personajes: any[] = [];`.
 - A red box surrounds the `get personajes(){...}` getter definition.
 - A red box surrounds the `constructor(...)` constructor definition.- Bottom Status Bar:** PROBLEMAS, SALIDA, CONSOLA DE DEPURACIÓN, TERMINAL.
- Terminal Output:** Compiled successfully.

Deberíamos de poder ver el listado desde el servicio

A screenshot of a web browser window. The address bar shows 'localhost:4200'. The page title is 'Dragon Ball Z'. On the left, there is a list titled 'Personajes' with two items: 'Vegeta - 20000' and 'Goku - 25000'. On the right, there is a form titled 'Agregar' with fields for 'Nombre' (containing '0') and a 'Agregar' button. A red box highlights the 'Personajes' list.

Personajes

- Vegeta - 20000
- Goku - 25000

Agregar

Nombre

0

Agregar

Ahora agregamos al servicio la funcionalidad de crear personaje

The screenshot shows the Visual Studio Code interface with the Explorer and Editor panes visible. The Explorer pane on the left shows the project structure with files like .angular, node_modules, and various components and services under the src/app/dbz directory. The Editor pane on the right displays the dbz.service.ts file. A red box highlights the file name in the title bar and the file tab in the bottom navigation bar. Another red box highlights the new method definition at the bottom of the code.

```
TS dbz.service.ts U X
src > app > dbz > services > TS dbz.service.ts > ...
9   nombre: "Vegeta",
10  poder: 20000
11 },
12   nombre: "Goku",
13  poder: 25000
14 }
15 ];
16
17 get personajes(): Personaje[]{
18   return [...this._personajes];
19 }
20 constructor(){
21   console.log('Servicio Iniciado');
22 }
23
24 agregarPersonaje( personaje: Personaje){
25   this._personajes.push(personaje);
26 }
```

EXPLORADOR

EDITORES ABIERTOS

PRIMERO

- > .angular
- > node_modules
- src
 - app
 - > acumulador
 - dbz
 - agregar
 - # agregar.component.css
 - agregar.component.html
 - TS agregar.component.ts
 - interfaces
 - main-page
 - personajes
 - # personajes.component.css
 - personajes.component.html
 - TS personajes.component.ts
 - services
 - TS dbz.service.ts
 - TS dbz.module.ts
 - heroes
 - # app.component.css

TS agregar.component.ts U X

```
src > app > dbz > agregar > TS agregar.component.ts > ...
```

```
8     styleUrls: ['./agregar.component.css']
9 })
10 export class AgregarComponent {
11
12     @Input() personajes:Personaje[] = [];
13
14     @Input() nuevo :Personaje = {
15         nombre: '',
16         poder: 0
17     }
18
19     constructor(private dbzService:DbzService){}
20 // @Output() onNuevoPersonaje: EventEmitter<Personaje> = new EventEmitter();
21     agregar(){
22         if(this.nuevo.nombre.trim().length === 0){ return;}
23         console.log(this.nuevo);
24         //this.personajes.push(this.nuevo);
25         //this.onNuevoPersonaje.emit(this.nuevo);
26         this.dbzService.agregarPersonaje(this.nuevo);
27         this.nuevo = {
28             nombre: '',
29             poder: 0
30         }
31     }
32 }
```

El main-page.component.html queda así

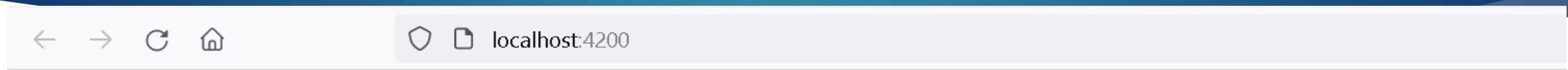
The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar showing project files. The main area is the Editor with the file `main-page.component.html` open. The code is as follows:

```
src > app > dbz > main-page > main-page.component.html > div.row > div.col > app-agregar
4   <div class="row">
5
6     <div class="col">
7       <h3>Personajes</h3>
8       <hr>
9       <!--<app-personajes [personajes]="personajes"></app-personajes>-->
10      <app-personajes></app-personajes>
11    </div>
12
13    <div class="col">
14
15      <!--<app-agregar
16          [personajes]="personajes"
17          (onNuevoPersonaje) = "agregarNuevoPersonaje( $event )">
18      </app-agregar>-->
19      <app-agregar>
20      </app-agregar>
21
22    </div>
23  </div>
24
```

Two specific sections of the code are highlighted with red boxes:

- A red box surrounds the line `<app-personajes></app-personajes>` at line 10.
- A red box surrounds the entire `<app-agregar>` element starting at line 19.

Finalmente debería mostrar y agregar los datos desde servicios



Dragon Ball Z

Personajes

- Vegeta - 20000
- Goku - 25000
- Majin Boo - 150000000

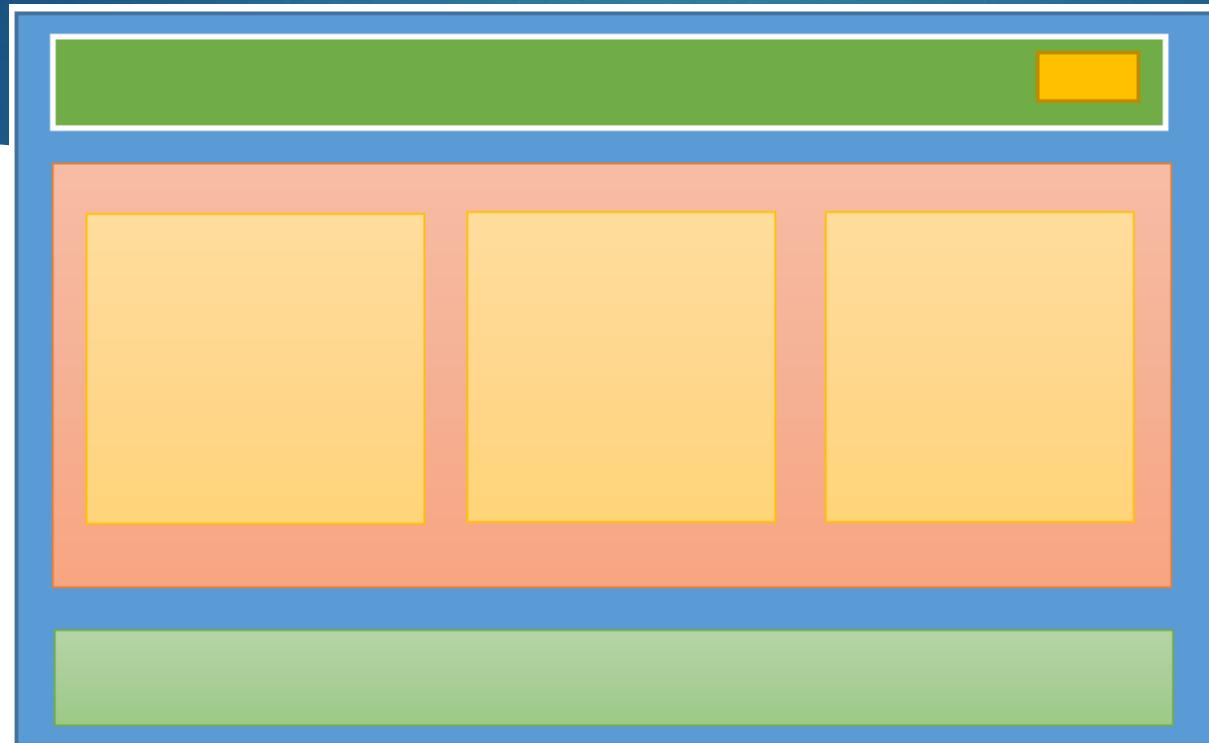
Agregar

Nombre

0

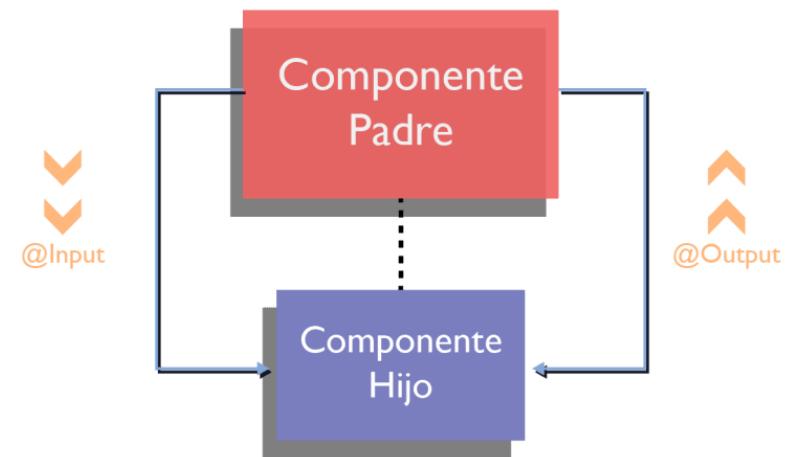
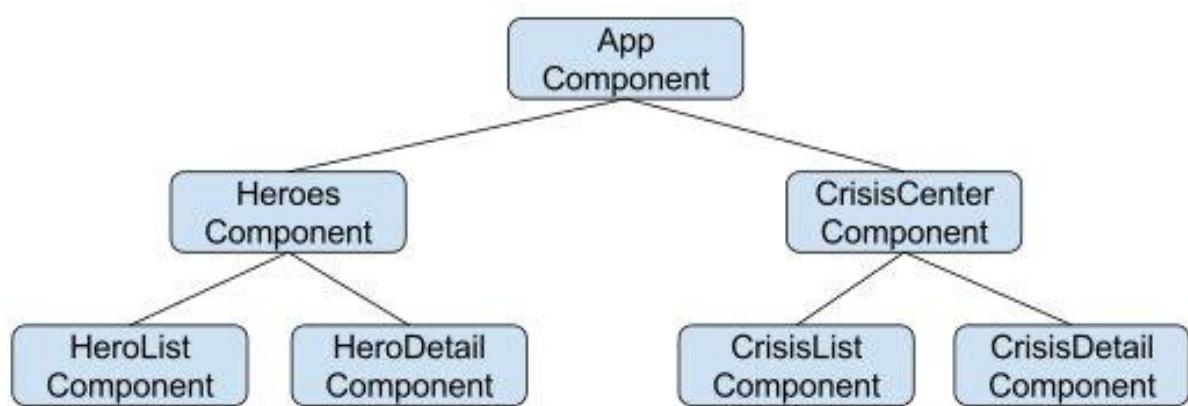
Agregar

Resumen

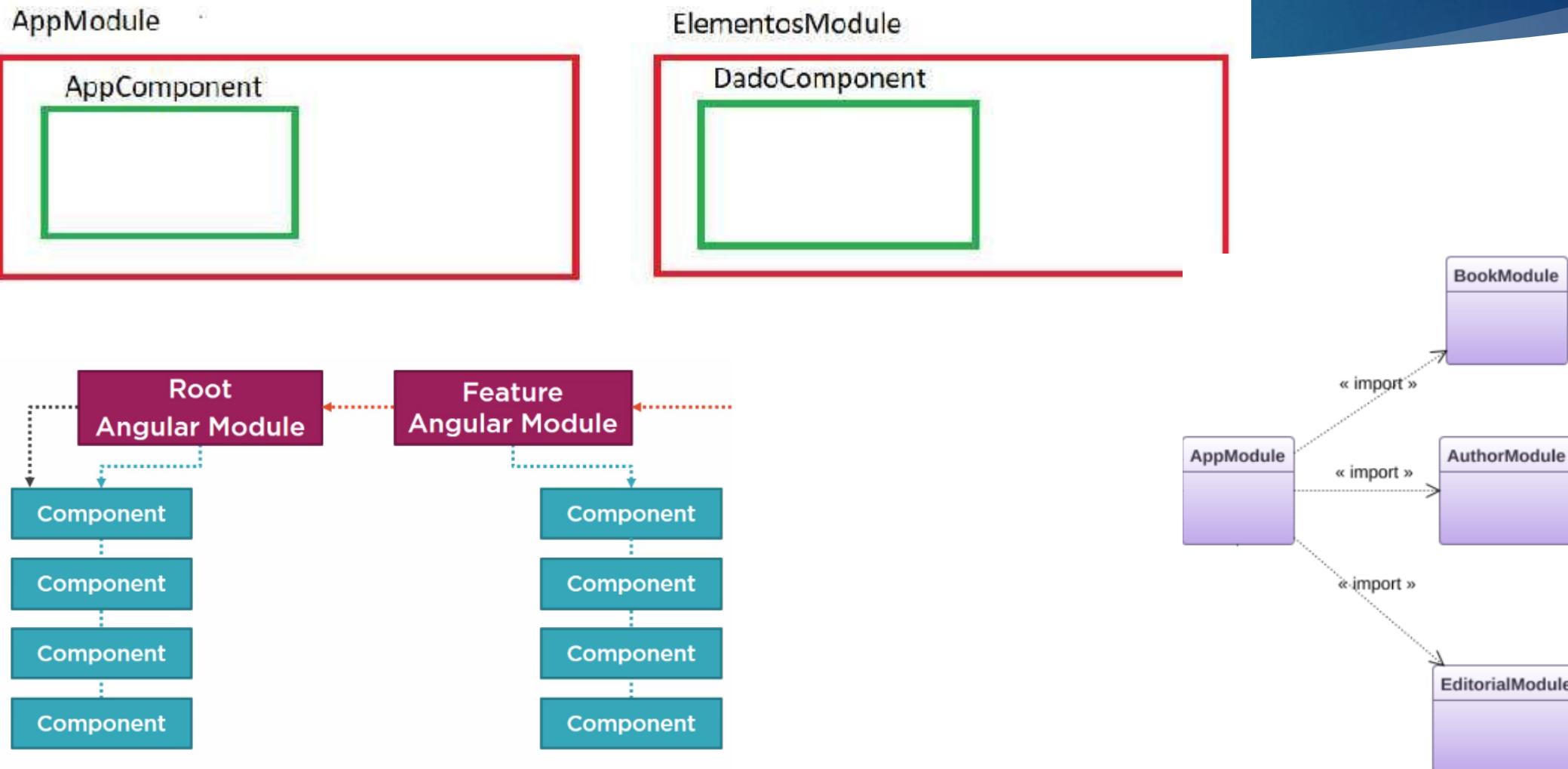


Aplicación = Componente + Componente + Componente

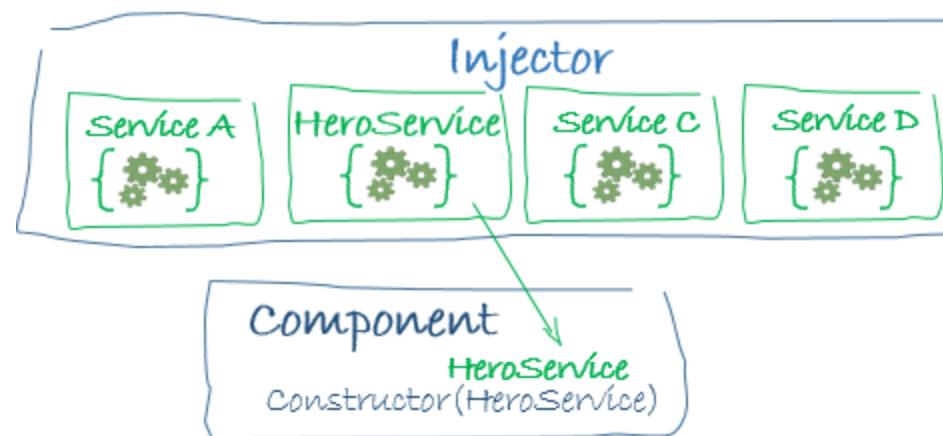
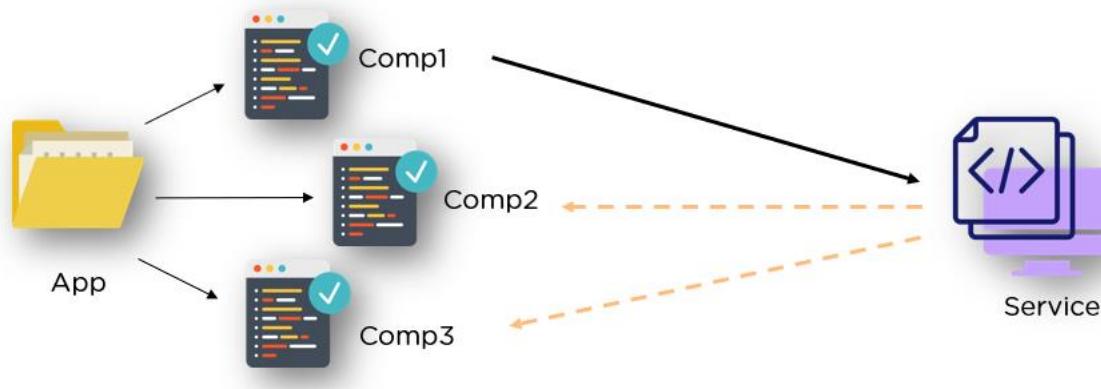
Resumen Componente



Resumen Modulo



Resumen Servicios



Lanzar nuestra app a producción

▶ ng build --prod

The screenshot shows a Visual Studio Code interface with the following details:

- EXPLORADOR (File Explorer):** Shows the project structure under "PRIMERO". The "dist\primero" folder is selected and highlighted with a red box.
- TERMINAL:** Displays the command "ng build --prod" being run in a Windows PowerShell window. The output indicates that the "--prod" option is deprecated and that the build is complete.
- PROBLEMAS (Problems):** Shows no errors or warnings.
- SALIDA (Output):** Shows the build results, including the creation of four files: main.js, polyfills.js, runtime.js, and styles.css. The total size of these files is 183.82 kB.
- CONSOLA DE DEPURACIÓN (Debug Console):** Shows the build timestamp, hash, and time taken.

Initial Chunk Files	Names	Size
main.167deb787e63fe31.js	main	146.13 kB
polyfills.86269f80ff058116.js	polyfills	36.19 kB
runtime.deaf7ab852f1abc5.js	runtime	1.04 kB
styles.02c9bc3fc0d033996.css	styles	465 bytes

Initial Total | 183.82 kB

Build at: 2021-12-12T18:58:49.374Z - Hash: 93be2d83abe4eda5 - Time: 23683ms
PS C:\Users\dell\Desktop\pruebas\primero> []



Platform Pricing Enterprise Jamstack Community Docs

Contact sales Log in [Sign up →](#)

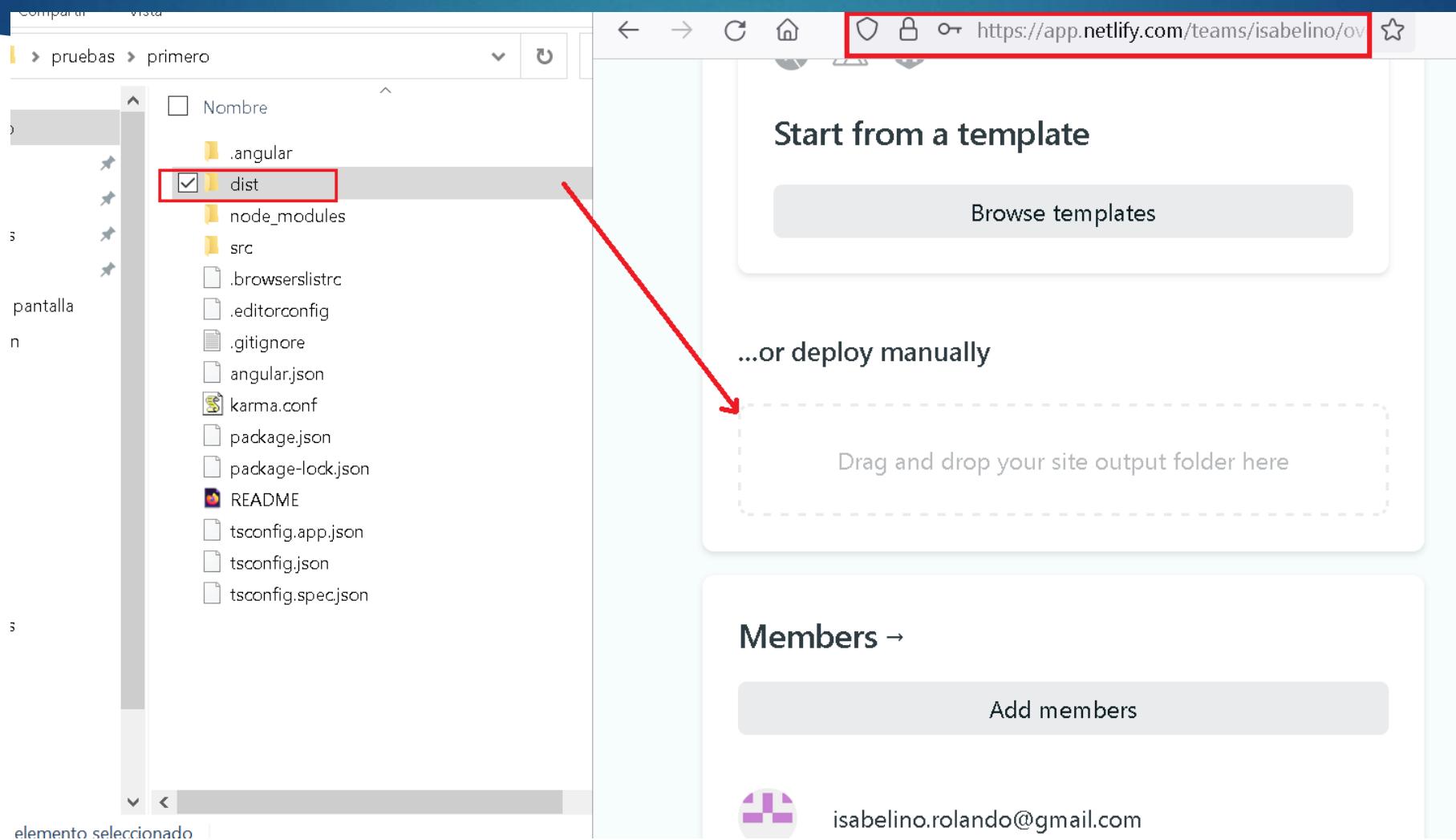
Build, deploy, & scale modern web projects

Millions of developers use Netlify to ship faster,
simplify their workflows, and scale effortlessly.

[Get started for free](#)

[Contact sales →](#)

Arrastramos el archivo dist de esta manera



The screenshot shows the Netlify Deploy section for the site 'unruffled-poitras-80b380'. A red box highlights the URL link in the 'Deploys' section. Another red box highlights the 'Published' status in the 'Deploys' list.

Deploys for unruffled-poitras-80b380

- <https://unruffled-poitras-80b380.netlify.app>

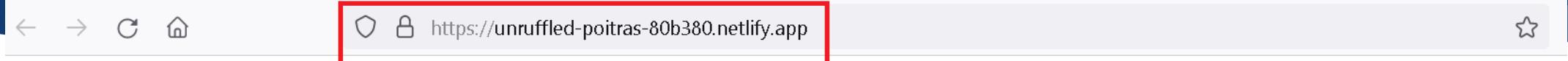
Manual deploys.

Auto publishing is on. Deploys are published automatically.

Deploy settings Notifications Stop auto publishing

Deploys

Production	Status	Date	Message
Production	Published	Today at 8:05 PM	Deployed in 1s
No deploy message			



Dragon Ball Z

Personajes

- Vegeta - 20000
- Goku - 25000
- Mister Satan - 34

Agregar

Nombre

0

Lifecycle hook

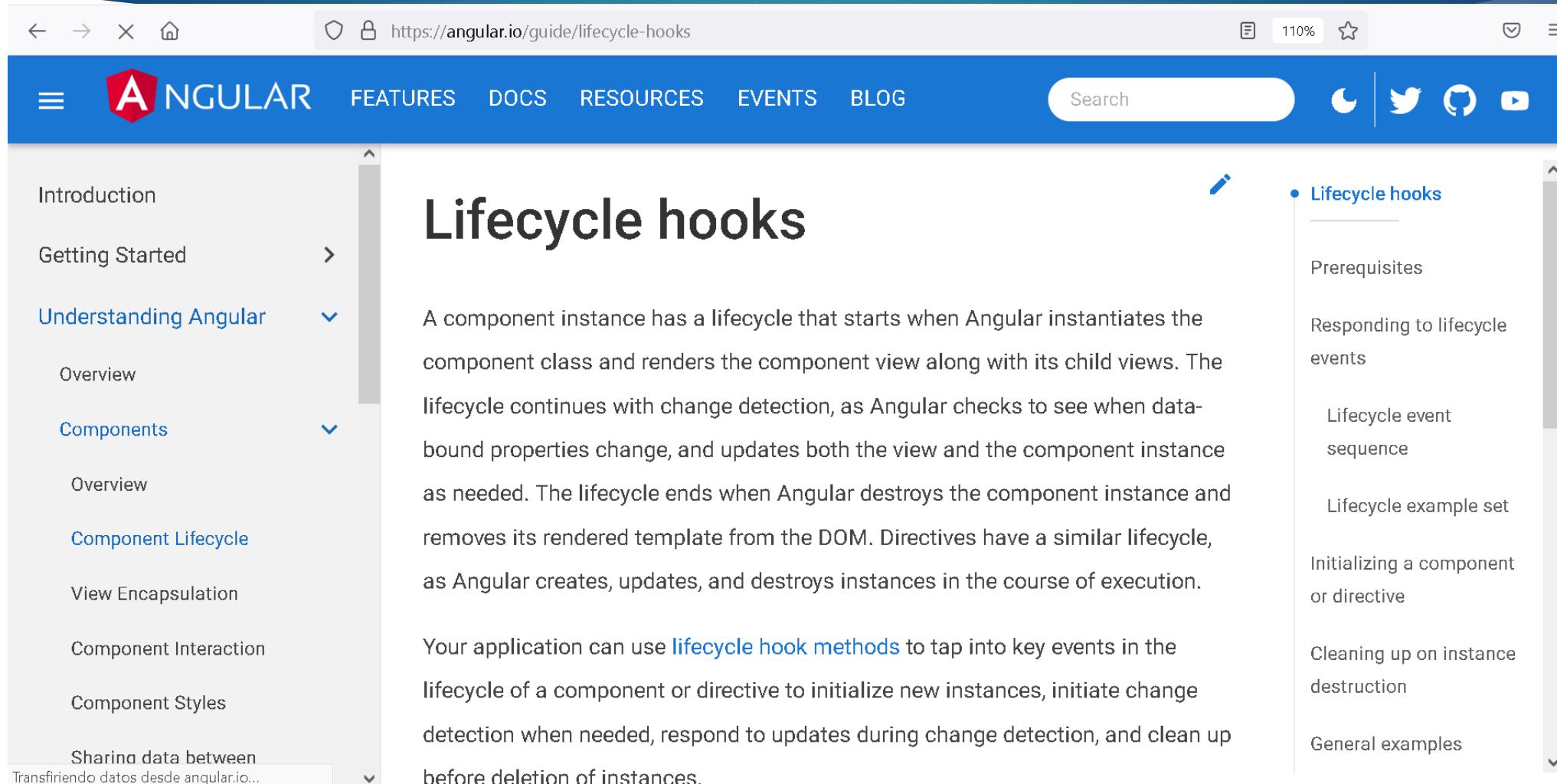
- ▶ Son métodos que podemos llamar cuando angular ejecute ciertos momentos como montaje de componentes, cuando se inicializa, cuando se actualiza el dom entre otros.

Creamos un nuevo proyecto angular

0:~ npm

```
C:\Users\dell\Desktop\pruebas> ng new ciclode-vida
? Would you like to add Angular routing? No
? Which stylesheet format would you like to use? CSS
CREATE ciclode-vida/angular.json (3075 bytes)
CREATE ciclode-vida/package.json (1076 bytes)
CREATE ciclode-vida/README.md (1057 bytes)
CREATE ciclode-vida/tsconfig.json (863 bytes)
CREATE ciclode-vida/.editorconfig (274 bytes)
CREATE ciclode-vida/.gitignore (620 bytes)
CREATE ciclode-vida/.huskyrc (600 bytes)
```

La guía oficial



A screenshot of a web browser displaying the official Angular guide for lifecycle hooks. The page has a blue header with the Angular logo and navigation links for Features, Docs, Resources, Events, and Blog. A search bar and social sharing icons are also present. The main content area features a large title "Lifecycle hooks" with a detailed description of the component lifecycle. To the left is a sidebar with navigation links for Introduction, Getting Started, Understanding Angular (Overview and Components), Component Lifecycle, View Encapsulation, Component Interaction, Component Styles, and Sharing data between components. On the right, there is a sidebar with links for Lifecycle hooks, Prerequisites, Responding to lifecycle events, Lifecycle event sequence, Lifecycle example set, Initializing a component or directive, Cleaning up on instance destruction, and General examples.

Introduction

Getting Started

Understanding Angular

- Overview
- Components

Component Lifecycle

View Encapsulation

Component Interaction

Component Styles

Sharing data between components

Transfiriendo datos desde angular.io...

Lifecycle hooks

A component instance has a lifecycle that starts when Angular instantiates the component class and renders the component view along with its child views. The lifecycle continues with change detection, as Angular checks to see when data-bound properties change, and updates both the view and the component instance as needed. The lifecycle ends when Angular destroys the component instance and removes its rendered template from the DOM. Directives have a similar lifecycle, as Angular creates, updates, and destroys instances in the course of execution.

Your application can use [lifecycle hook methods](#) to tap into key events in the lifecycle of a component or directive to initialize new instances, initiate change detection when needed, respond to updates during change detection, and clean up before deletion of instances.

- [Lifecycle hooks](#)
- Prerequisites
- Responding to lifecycle events
- Lifecycle event sequence
- Lifecycle example set
- Initializing a component or directive
- Cleaning up on instance destruction
- General examples

Creamos los siguientes componentes

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORADOR**: Shows the file structure of the project "CICLODE-VIDA". A folder named "pages\pagina1" is selected and highlighted with a red box.
- EDITORES ABIERTOS**: Shows the files in the selected folder:
 - pagina1.component.html
 - pagina1.component.ts
 - app.component.css
 - app.component.html
 - app.component.spec.ts
 - app.component.ts
 - app.module.ts
- CICLODE-VIDA**: Shows icons for creating, opening, saving, and deleting files.
- PROBLEMAS**: Shows the command run in the terminal: `ng g c pages/pagina1 --skipTests -is`. It also displays a warning message about deprecated camel case arguments.
- SALIDA**: Shows the output of the command: CREATE src/app/pages/pagina1/pagina1.component.html (22 bytes), CREATE src/app/pages/pagina1/pagina1.component.ts (254 bytes), and UPDATE src/app/app.module.ts (406 bytes).
- CONSOLA DE DEPURACIÓN**: Shows the command run in the terminal: `PS C:\Users\dell\Desktop\pruebas\ciclude-vida> ng g c pages/pagina1 --skipTests -is`.
- TERMINAL**: Shows the command run in the terminal: `PS C:\Users\dell\Desktop\pruebas\ciclude-vida>` .
- powerShell**: Shows the current shell type as PowerShell.

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORADOR**: Shows the file structure of the project "CICLODE-VIDA". A red box highlights the folder "components\muestra-nombre".
- EDITORES ABIERTOS**: Shows the files "muestra-nombre.component.html" and "muestra-nombre.component.ts" are open.
- CICLODE-VIDA**: Shows the project structure with "node_modules", "src", "app", "components", "pages", and "assets" folders.
- PROBLEMAS**: Shows no problems.
- SALIDA**: Shows the command "ng g c pages/pagina1 --skipTests -is".
- CONSOLA DE DEPURACIÓN**: Shows the message: "Support for camel case arguments has been deprecated and will be removed in a future major version. Use '--skip-tests' instead of '--skipTests'." followed by "CREATE src/app/pages/pagina1/pagina1.component.html (22 bytes)" and "CREATE src/app/pages/pagina1/pagina1.component.ts (254 bytes)".
- TERMINAL**: Shows the command "ng g c components/muestraNombre --skipTests -is".
- STATUS BAR**: Shows the path "PS C:\Users\dell\Desktop\pruebas\ciclude-vida>" and the status "CREATE src/app/components/muestra-nombre/muestra-nombre.component.html (29 bytes)".

El app.component.html lo dejamos así

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORADOR**: Shows the project structure:
 - EDITORES ABIERTOS: app.component.html (highlighted with a red box)
 - CICLODE-VIDA
 - node_modules
 - src
 - app
 - components\muestra-nombre
 - pages\pagina1
 - # app.component.css
 - app.component.html (highlighted with a red box)
 - TS app.component.spec.ts
 - TS app.component.ts
 - TS app.module.ts
 - assets
 - environments
 - favicon.ico
 - index.html
 - main.ts
 - polyfills.ts
 - # styles.css
 - EDITOR**: app.component.html (highlighted with a red box)

```
<h1>Ciclo de Vida de Angular</h1>
<hr>
```
 - TERMINAL**: PS C:\Users\dell\Desktop\pruebas\ciclode-vida>



localhost:4200

Ciclo de Vida de Angular

Método de gancho	Propósito	Momento
<code>ngOnChanges()</code>	<p>Responder cuando Angular establece o restablece propiedades de entrada vinculadas a datos. El método recibe un <code>SimpleChanges</code> objeto de valores de propiedad actuales y anteriores.</p> <p>Tenga en cuenta que esto sucede con mucha frecuencia, por lo que cualquier operación que realice aquí tendrá un impacto significativo en el rendimiento. Consulte los detalles en Uso de ganchos de detección de cambios en este documento.</p>	<p>Se llama antes <code>ngOnInit()</code> (si el componente tiene entradas vinculadas) y cada vez que cambian una o más propiedades de entrada vinculadas a datos.</p> <p>Tenga en cuenta que si su componente no tiene entradas o lo usa sin proporcionar ninguna entrada, el marco no llamará <code>ngOnChanges()</code>.</p>

Método de gancho	Propósito	Momento
ngOnInit()	Inicialice la directiva o el componente después de que Angular muestre primero las propiedades enlazadas a datos y establezca las propiedades de entrada de la directiva o del componente. Consulte los detalles en Inicialización de un componente o directiva en este documento.	Llamado una vez, después de la primera <code>ngOnChanges()</code> . <code>ngOnInit()</code> todavía se llama incluso cuando <code>ngOnChanges()</code> no lo es (que es el caso cuando no hay entradas vinculadas a la plantilla).

Método de gancho	Propósito	Momento
ngDoCheck()	<p>Detecte y actúe sobre los cambios que Angular no puede o no detectará por sí solo. Consulte los detalles y el ejemplo en Definición de la detección de cambios personalizada en este documento.</p>	<p>Se llama inmediatamente después <code>ngOnChanges()</code> de cada ejecución de detección de cambios e inmediatamente después <code>ngOnInit()</code> de la primera ejecución.</p>

Método de gancho	Propósito	Momento
<code>ngAfterContentInit()</code>	<p>Responda después de que Angular proyecte contenido externo en la vista del componente o en la vista en la que se encuentra una directiva.</p> <p>Consulte los detalles y el ejemplo en Respuesta a cambios en el contenido de este documento.</p>	Llamado <i>una vez</i> después del primero <code>ngDoCheck()</code> .

Método de gancho	Propósito	Momento
<code>ngAfterContentChecked()</code>	<p>Responda después de que Angular verifique el contenido proyectado en la directiva o el componente.</p> <p>Consulte los detalles y el ejemplo en Respuesta a los cambios de contenido proyectados en este documento.</p>	<p>Llamado después <code>ngAfterContentInit()</code> y todos los posteriores <code>ngDoCheck()</code>.</p>

Método de gancho	Propósito	Momento
<code>ngAfterViewInit()</code>	<p>Responda después de que Angular inicialice las vistas del componente y las vistas secundarias, o la vista que contiene la directiva.</p> <p>Consulte los detalles y el ejemplo en Responder para ver los cambios en este documento.</p>	Llamado <i>una vez</i> después del primero <code>ngAfterContentChecked()</code> .

Método de gancho**Propósito****Momento****ngAfterViewChecked()**

Responda después de que Angular verifique las vistas del componente y las vistas secundarias, o la vista que contiene la directiva.

Llamado después del

ngAfterViewInit() y todos

los siguientes

ngAfterContentChecked().

Método de gancho	Propósito	Momento
ngOnDestroy()	<p>Limpieza justo antes de que Angular destruya la directiva o el componente. Cancelé la suscripción a Observables y desconecte los controladores de eventos para evitar pérdidas de memoria.</p> <p>Consulte los detalles en Limpieza de la destrucción de instancias en este documento.</p>	Se llama inmediatamente antes de que Angular destruya la directiva o el componente.

En pagina1.component.ts agregamos esto

TS pagina1.component.ts X

```
src > app > pages > pagina1 > TS pagina1.component.ts > Pagina1Component
1 import { AfterContentChecked, AfterContentInit,
2   AfterViewChecked, AfterViewInit,
3   Component, DoCheck, OnChanges, OnDestroy,
4   OnInit, SimpleChanges } from '@angular/core';
5
6 @Component({
7   selector: 'app-pagina1',
8   templateUrl: './pagina1.component.html',
9   styles: [
10 ]
11 })
12
13 export class Pagina1Component implements OnInit, OnChanges, DoCheck, AfterContentInit,
14 AfterContentChecked, AfterViewInit, AfterViewChecked, OnDestroy {
15
16   constructor() { }
17   ngOnChanges(changes: SimpleChanges): void {
18     console.log('ngOnChanges');
19   }
20   ngDoCheck(): void {
21     console.log('ngDoCheck');
22   }
23   ngAfterContentInit(): void {
24     console.log('ngAfterContentInit');
25   }
26 }
```

TS pagina1.component.ts U X

src > app > pages > pagina1 > **TS** pagina1.component.ts > Pagina1Component

```
15  
16     constructor() { }  
17     ngOnChanges(changes: SimpleChanges): void {  
18         console.log('ngOnChanges');  
19     }  
20     ngDoCheck(): void {  
21         console.log('ngDoCheck');  
22     }  
23     ngAfterContentInit(): void {  
24         console.log('ngAfterContentInit');  
25     }  
26     ngAfterContentChecked(): void {  
27         console.log('ngAfterContentInit');  
28     }  
29     ngAfterViewInit(): void {  
30         console.log('ngAfterContentInit');  
31     }  
32     ngAfterViewChecked(): void {  
33         console.log('ngAfterViewChecked')  
34     }  
35     ngOnDestroy(): void {  
36         console.log('ngOnDestroy');  
37     }  
38
```

TS pagina1.component.ts U X

```
src > app > pages > pagina1 > TS pagina1.component.ts > Pagina1Component  
+--  
20  ngDoCheck(): void {  
21    |   console.log('ngDoCheck');  
22  }  
23  ngAfterContentInit(): void {  
24    |   console.log('ngAfterContentInit');  
25  }  
26  ngAfterContentChecked(): void {  
27    |   console.log('ngAfterContentInit');  
28  }  
29  ngAfterViewInit(): void {  
30    |   console.log('ngAfterContentInit');  
31  }  
32  ngAfterViewChecked(): void {  
33    |   console.log('ngAfterViewChecked')  
34  }  
35  ngOnDestroy(): void {  
36    |   console.log('ngOnDestroy');  
37  }  
38  
39  ngOnInit(): void {  
40    |   console.log('ngOnInit')  
41  }  
42  
43  }
```

EXPLORADOR ... TS pagina1.component.ts U <> app.component.html M X

EDITORES ABIERTOS

- TS pagina1.component.ts src\ap... U
- X <> app.component.html src\app M

CICLODE-VIDA

- src
- app
 - > components
 - pages\pagina1
 - <> pagina1.component.html U
 - TS pagina1.component.ts U
 - # app.component.css
 - <> app.component.html M
 - TS app.component.spec.ts
 - TS app.component.ts
 - TS app.module.ts M
 - > assets

ESQUEMA

LÍNEA DE TIEMPO

src > app > <> app.component.html > app-pagina1

Go to component

```
1 <h1>Ciclo de Vida de Angular</h1>
2 <hr>
3
4 <app-pagina1></app-pagina1>
5
6
```

PROBLEMAS SALIDA TERMINAL ... > r

e: 353ms

✓ Compiled successfully.

A screenshot of a web browser window displaying the Angular lifecycle log in the developer tools' "Consola" tab. The page title is "localhost:4200". The main content area shows the heading "Ciclo de Vida de Angular" and the message "pagina1 works!". The developer tools sidebar includes icons for Inspector, Consola (which is selected), Depurador, and other tabs like Filtrar salida, Errores, Advertencias, Registros, Información, Depurar, CSS, XHR, and Peticiones. The "Consola" tab lists the following log entries:

- Disconnected! (index.js:54)
- Trying to reconnect... (index.js:54)
- ngOnInit (pagina1.component.ts:40:1)
- ngDoCheck (pagina1.component.ts:21:1)
- ngAfterContentInit (pagina1.component.ts:24:1)
- ngAfterContentInit (pagina1.component.ts:27:1)
- ngAfterContentInit (pagina1.component.ts:30:1)
- ngAfterViewChecked (pagina1.component.ts:33:1)
- Angular is running in development mode. Call enableProdMode() to enable production mode. (core.mjs:24856:1)
- ngDoCheck (pagina1.component.ts:21:1)
- ngAfterContentInit (pagina1.component.ts:27:1)
- ngAfterViewChecked (pagina1.component.ts:33:1)
- Live Reloading enabled. (index.js:54)

The "Consola" tab has a "Filtrar salida" (Filter output) dropdown set to "Filtrar salida" (Filter output). There are also buttons for "CSS", "XHR", and "Peticiones" (Requests).

EXPLORADOR ... TS pagina1.component.ts U TS app.component.ts M X ↗ app.compo

> EDITORES ABIERTOS

✓ CICLODE-VIDA

✓ src ●
 ✓ app ●
 > components ●
 ✓ pages \pagina1 ●
 < pagina1.component.html U
 TS pagina1.component.ts U
 # app.component.css
 < app.component.html M
 TS app.component.spec.ts
 TS app.component.ts M
 TS app.module.ts M
 > assets
 > environments
 ★ favicon.ico
> ESQUEMA
> LÍNEA DE TIEMPO

src > app > TS app.component.ts > AppComponent

```
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   mostrar:boolean = true;
10 }
11
```

PROBLEMAS SALIDA TERMINAL ... ➤ node +

e: 353ms

✓ Compiled successfully.

EXPLORADOR

> EDITORES ABIERTOS

✓ CICLODE-VIDA

✓ src

 ✓ app

 > components

 ✓ pages\pagina1

 <> pagina1.component.html

 TS pagina1.component.ts

app.component.css

<> app.component.html M

TS app.component.spec.ts

TS app.component.ts M

TS app.module.ts M

> assets

> environments

★ favicon.ico

> ESQUEMA

> LÍNEA DE TIEMPO

... component.ts U TS app.component.ts M <> app.component.html M X

src > app > <> app.component.html > ...

Go to component

1 <h1>Ciclo de Vida de Angular</h1>

2 <hr>

3

4 <button (click)="mostrar = !mostrar">Cerrar</button>

5

6 <app-pagina1 *ngIf="mostrar"></app-pagina1>

7

PROBLEMAS SALIDA TERMINAL ...

e: 403ms

✓ Compiled successfully.

A screenshot of a web browser window displaying the Angular Lifecycle Hooks console output. The browser address bar shows "localhost:4200". The main content area displays the title "Ciclo de Vida de Angular" and a "Cerrar" button, which is highlighted with a red box. The right side of the screen shows the developer tools' "Consola" tab with the following log entries:

Message	File	Line
[webpack-dev-server] Disconnected!	index.js	548
[webpack-dev-server] Trying to reconnect...	index.js	548
[webpack-dev-server] Live Reloading enabled.	index.js	548
ngOnInit	pagina1.component.ts	40:12
ngDoCheck	pagina1.component.ts	21:14
ngAfterContentInit	pagina1.component.ts	24:14
ngAfterContentInit	pagina1.component.ts	27:14
ngAfterContentInit	pagina1.component.ts	30:14
ngAfterViewChecked	pagina1.component.ts	33:12
Angular is running in development mode. Call enableProdMode() to enable production mode.	core.mjs	24856:16
ngDoCheck	pagina1.component.ts	21:14
ngAfterContentInit	pagina1.component.ts	27:14
ngAfterViewChecked	pagina1.component.ts	33:12
ngOnDestroy	pagina1.component.ts	36:14

The "ngOnDestroy" entry is also highlighted with a red box. The browser interface includes standard navigation buttons (back, forward, search) and a toolbar with tabs like Inspector, Consola, and Depurador.

EXPLORADOR ... TS pagina1.component.ts U X TS app.component.ts M < app.compor

> EDITORES ABIERTOS

✓ CICLODE-VIDA

- ✓ src
- ✓ app
 - > components
 - ✓ pages\pagina1
 - < pagina1.component.html
 - TS pagina1.component.ts
 - # app.component.css
 - < app.component.html M
 - TS app.component.spec.ts
 - TS app.component.ts M
 - TS app.module.ts M
 - > assets
 - > environments
 - ★ favicon.ico
- > ESQUEMA

src > app > pages > pagina1 > TS pagina1.component.ts > Pagina1Component

```
38
39   ngOnInit(): void {
40     console.log('ngOnInit')
41   }
42
43   guardar(){}
44 }
45 }
46 }
47 }
```

PROBLEMAS SALIDA TERMINAL ...

node + □

e: 245ms

✓ Compiled successfully.

EXPLORADOR ... TS pagina1.component.ts U <> pagina1.component.html U X

> EDITORES ABIERTOS

✓ CICLODE-VIDA

- ✓ src ●
- ✓ app ●
- > components ●
- ✓ pages\pagina1 ●
- <> pagina1.component.html U**
- TS pagina1.component.ts U
- # app.component.css
- <> app.component.html M
- TS app.component.spec.ts
- TS app.component.ts M
- TS app.module.ts M
- > assets
- > environments
- ★ favicon.ico

> ESQUEMA

src > app > pages > pagina1 > <> pagina1.component.html > ...

Go to component

1 <p>pagina1 works!</p>

2 <button (click)="guardar()">Click</button>

3

PROBLEMAS SALIDA TERMINAL ... node

e: 184ms

The screenshot shows a browser window with the URL `localhost:4200`. The main content area displays the title "Ciclo de Vida de Angular" and a message "pagina1 works!". Below the message is a button labeled "Click". The browser's developer tools are open, specifically the "Consola" tab. The console output shows three entries, each with a red border:

- ngDoCheck pagina1.component.ts:21:4
- ngAfterContentInit pagina1.component.ts:27:4
- ngAfterViewChecked pagina1.component.ts:33:2

The "Click" button is highlighted with a red box in the screenshot.

EXPLORADOR

EDITORES ABIERTOS

CICLO DE VIDA

src

app

- components
- pages\pagina1
 - pagina1.component.html
 - pagina1.component.ts
- # app.component.css
- app.component.html
- app.component.spec.ts
- app.component.ts
- app.module.ts

assets

environments

favicon.ico

index.html

component.ts M

pagina1.component.html U

pagina1.component.ts U

TS app.module.ts M X

```
src > app > TS app.module.ts > AppModule
  6   import { MuestraNombreComponent } from './components/muestra-nombre/muestra-nombr
  7   import { FormsModule } from '@angular/forms';
  8
  9   @NgModule({
 10     declarations: [
 11       AppComponent,
 12       PaginalComponent,
 13       MuestraNombreComponent
 14     ],
 15     imports: [
 16       BrowserModule,
 17       FormsModule
 18     ],
 19     providers: [],
 20     bootstrap: [AppComponent]
 21   })
 22   export class AppModule { }
```

EXPLORADOR

... component.ts M pagina1.component.html U **TS pagina1.component.ts U X** TS app.module.ts M

EDITORES ABIERTOS

CICLO DE VIDA

- src
- app
 - components
 - pages\pagina1
 - pagina1.component.html U
 - TS pagina1.component.ts** U

app.component.css
↳ app.component.html M
TS app.component.spec.ts
TS app.component.ts M
TS app.module.ts M
assets
environments
★ favicon.ico
↳ index.html
TS main.ts
TS polyfills.ts
styles.css

ESQUEMA

LÍNEA DE TIEMPO

```
src > app > pages > pagina1 > TS pagina1.component.ts > Pagina1Component > ngOnChanges
+ |  ORIGIN, SIMPLECHANGES FROM angular/core ,
5
6  @Component({
7    selector: 'app-pagina1',
8    templateUrl: './pagina1.component.html',
9    styles: [
10   ]
11 })
12
13 export class Pagina1Component implements OnInit, OnChanges, DoCheck, AfterContentIn
14 AfterContentChecked, AfterViewInit, AfterViewChecked, OnDestroy {
15
16   nombre:string = "Rolando";
17
18   constructor() { }
19   ngOnChanges(changes: SimpleChanges): void {
20     console.log('ngOnChanges');
21   }

```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

node + □

Build at: 2021-12-13T04:23:35.856Z - Hash: 521d4fe07e850e38 - Time: 322ms

✓ Compiled successfully.

EXPLORADOR

EDITORES ABIERTOS

CICLO DE VIDA

- src
- app
 - components
 - pages\pagina1
 - pagina1.component.html

app.component.css

<> app.component.html M

TS app.component.spec.ts

TS app.component.ts M

TS app.module.ts M

> assets

> environments

★ favicon.ico

<> index.html

component.ts M

<> pagina1.component.html U X

TS pagina1.component.ts U

TS app

src > app > pages > pagina1 > <> pagina1.component.html > ...

Go to component

```
1 <p>pagina1 works!</p>
2 <input type="text"
3     placeholder="Hola Mundo"
4     [(ngModel)]="nombre">
5 <button (click)="guardar()">Click</button>
6 
```

A screenshot of a web browser window displaying the Angular lifecycle console output. The browser address bar shows "localhost:4200". The main content area displays the title "Ciclo de Vida de Angular" and a message "pagina1 works!". Below this is an input field containing "Rolando!" and a button labeled "Click". A "Cerrar" button is also visible. The right side of the browser shows the developer tools with the "Consola" tab selected. The console output lists three events:

Event	File
ngDoCheck	pagina1.component.ts:23:1
ngAfterContentInit	pagina1.component.ts:29:1
ngAfterViewChecked	pagina1.component.ts:35:1

Ejemplo de onChange

The screenshot shows the VS Code interface with the following details:

- EXPLORADOR**: Shows the project structure with files like `muestra-nombre.component.ts`, `pagina1.component.html`, and `app.module.ts`.
- EDITORES ABIERTOS**: Shows the file `muestra-nombre.component.ts` open.
- CICLO DE VIDA**: Shows the component tree with `muestra-nombre` as the active component.
- Content of `muestra-nombre.component.ts`:**

```
src > app > components > muestra-nombre > muestra-nombre.component.ts > MuestraNombreComponent > ngOnChanges

1 import { Component, Input, OnChanges, OnInit, SimpleChanges } from '@angular/core';
2
3 @Component({
4   selector: 'app-muestra-nombre',
5   templateUrl: './muestra-nombre.component.html',
6   styles: [
7     ]
8 })
9 export class MuestraNombreComponent implements OnInit, OnChanges {
10
11   @Input() nombre!: string;
12
13   constructor() { }
14
15   ngOnChanges(changes: SimpleChanges): void {
16     console.log('Cambios');
17   }
18
19   ngOnInit(): void {
20   }
21
22 }
```

Annotations highlight specific parts of the code:

- A red box surrounds the file name `muestra-nombre.component.ts` in the Explorer and the component declaration in the code.
- A red box surrounds the `OnChanges` interface implementation in the code.
- A red box surrounds the `@Input()` annotation in the code.
- A red box surrounds the `ngOnChanges` method and its body in the code.

EXPLORADOR ... TS muestra-nombre.component.ts U pagina1.component.html U X

> EDITORES ABIERTOS

✓ CICLO DE VIDA

- ✓ src ●
- ✓ app ●
- ✓ components\muestra-n... ●
- ▷ muestra-nombre.comp... U
- TS muestra-nombre.comp... U
- ✓ pages\pagina1 ●
- ▷ pagina1.component.html U
- TS pagina1.component.ts U

app.component.css

▷ app.component.html M

TS app.component.spec.ts

TS app.component.ts M

TS app.module.ts M

> assets

> environments

★ favicon.ico

▷ index.html

TS main.ts

src > app > pages > pagina1 > ▷ pagina1.component.html > ...

Go to component

```
1  <p>pagina1 works!</p>
2  <input type="text"
3  ||| placeholder="Hola Mundo"
4  [(ngModel)]="nombre">
5  <button (click)="guardar()">Click</button>
6
7  <app-muestra-nombre [nombre]="nombre"></app-muestra-nombre>
8
9  |
```

The screenshot shows a browser window with the URL `localhost:4200`. The main content area displays the title "Ciclo de Vida de Angular" and a message "pagina1 works!". Below this, there is an input field containing "Rolando2" and a button labeled "Click". A second message "muestra-nombre works!" is also visible.

The browser's developer tools are open, specifically the "Consola" tab. The log output is as follows:

Event	File	Line Number
ngDoCheck	pagina1.component.ts	23:14
ngAfterContentInit	pagina1.component.ts	29:14
Cambios	...tra-nombre.component.ts	16:12
ngAfterViewChecked	pagina1.component.ts	35:12

The word "Cambios" in the third row is highlighted with a red box. The "Cambios" tab is also highlighted in the console tabs bar.