



Desarrollo y Programación de  
Aplicaciones Avanzadas con Angular

Formación online



## Módulo 6. Comunicación entre componentes

# Módulo 6. Comunicación entre componentes

## Input: Padre (A) -> Hijo (B)

Nombre del @input  
declarado en  
component B

<component-b [data]="myData"></component-b>

Selector del  
component

Nombre de la variable  
en el contexto de  
component A

Decorator que lo  
convierte en Input

HTML del Component A

```
import { Component, Input } from '@angular/core';
import { Data } from './data.interface';

@Component({
  selector: 'component-b',
  templateUrl: './component-b.component.html',
  styleUrls: ['./component-b.component.scss']
})
export class ComponentBComponent {
  // puede ser de tipo primitivo/objeto (string, number, Person ... )
  @Input() data: Data;
}
```

Nombre del @input

TS del Component B

# Módulo 6. Comunicación entre componentes

## Output: Hijo (B) -> Padre (A)

```
<component-b [data]="myData" (nameEvt)="imprimirNombre()"></component-b>
```

HTML del Component A

Nombre del output (component B)  
al que vamos a "escuchar" para  
cuando emita

Método del component A  
que se ejecuta cuando el  
output emite un valor

Decorador que lo  
convierte en Output

```
import { Component, Input, Output, EventEmitter } from '@angular/core';
import { Data } from './data.interface';

@Component({
  selector: 'component-b',
  templateUrl: './component-b.component.html',
  styleUrls: ['./component-b.component.scss']
})
export class ComponentBComponent {
  // Ambos Input & Output deben ser importados de @angular/core
  @Input() data: Data;
  @Output() nameEvt = new EventEmitter<string>();

  enviarAlPadre() {
    this.nameEvt.emit('Alejandro');
  }
}
```

Ejecutar este método enviará un valor al  
componente padre, sólo si el padre está escuchando

TS del Component B

# LABORATORIO: Listado de tareas completadas

## EJERCICIO 1

Se pide implementar el listado de tareas mostrado en la tabla siguiente. Debe utilizarse para ello componentes unidades, en concreto deberán utilizarse los siguientes componentes:

- **TableComponent**: dicho componente debe dibujar una tabla a partir de un listado de tareas. La Tarea contiene un nombre y un porcentaje de desarrollo de la misma.
- **TaskComponent**: dicho componente dibuja el nombre de la tarea.
- **ProgressBarComponent**: este componente recibe por parámetro @Input el porcentaje de desarrollo de la tarea y en función de este lo dibuja en un Progress bar con un color diferente en función del grado de desarrollo.
- **LabelComponent**: el comportamiento es similar al ProgressBarComponent pero en lugar de dibujar una barra de progreso, utiliza un badge con el valor que recibe por @Input.

#	Task	Progress	Label
1.	Update software	<div style="width: 55%; background-color: yellow;"></div>	<span>55%</span>
2.	Clean database	<div style="width: 70%; background-color: blue;"></div>	<span>70%</span>
3.	Cron job running	<div style="width: 30%; background-color: red;"></div>	<span>30%</span>
4.	Fix and squish bugs	<div style="width: 90%; background-color: green;"></div>	<span>90%</span>

# LABORATORIO: Listado de tareas completadas II

## EJERCICIO 2

Sobre la tabla anterior, realizar los siguientes cambios:

1. Modificar el componente *LabelComponent* para que en lugar de mostrar un badge, muestre un `<input type="number" />` que permita modificar el valor del porcentaje. Hacer uso de *ngModel* para ello.
2. Cuando ese valor sea modificado, deberá trasladarse al componente *TableComponent*, para que éste se lo comunique al componente *ProgressBarComponent*.
3. Añadir al mismo nivel que el *TableComponent* un contador de tareas finalizadas VS total de tareas, haciendo uso de un nuevo componente *ContadorComponent*.
4. Hacer también un cambio para que si la tarea está completada, el nombre de dicha tarea aparezca tachado.