



CJAVA

siempre para apoyarte



Visión

Poder aportar al desarrollo del País usando tecnología Java.

Quienes Somos

Somos una organización orientada a **desarrollar, capacitar e investigar tecnología JAVA** a través de un prestigioso staff de profesionales a nivel nacional.





CJAVA

siempre para apoyarte

cjavaperu.com
info@cjavaperu.com

COMUNIDAD

ACADEMICA



CJAVA

siempre para apoyarte



CJAVA
siempre para apoyarte

Servicio de Capacitación

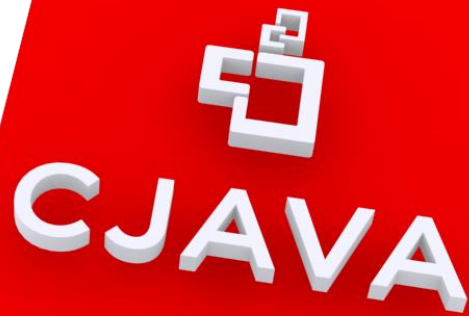
- **Programer** (Antes Java Developer Junior - 80 horas)
[Certificado: Java Programer]
- **Developer** (Antes Java Web Developer - 80 horas)
[Certificado: Java Developer]
- **Expert** (Antes Java Developer Senior - 80 horas)
[Certificado: Java Expert]
- **Arquitect** (Antes ADS-RUP - 80 horas)
[Certificado: Java Arquitect]
- **Carrera** (12 meses de contenido Java)
[Diploma: Carrera Java]

Architect

Expert

Developer

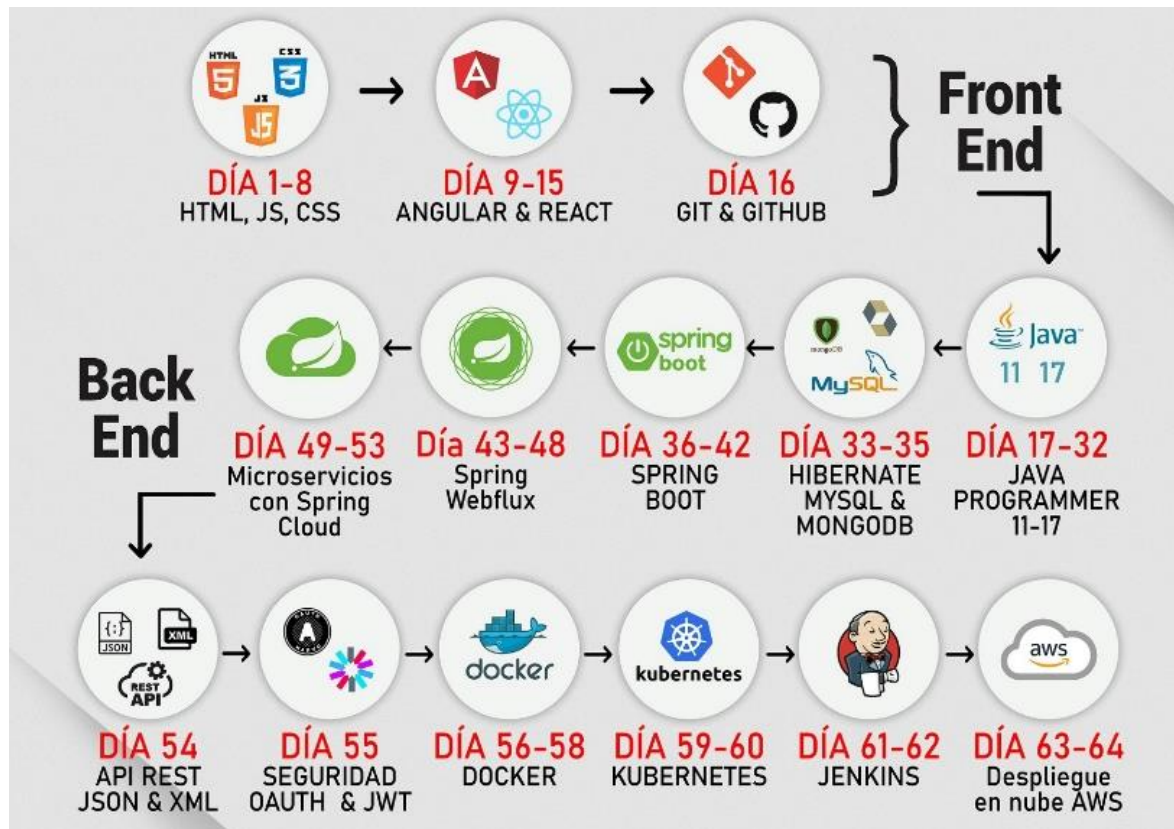
Programmer



Java FullStack Developer

emaravi@cjavaperu.com

Introducción a Full Stack Development



Introducción a Full Stack Development

Front-end Development: Discutir las tecnologías y herramientas utilizadas en el desarrollo front-end, como HTML, CSS, JavaScript y React.



Directivas en Angular

Edwin Maravi

Contenidos o temas

- ¿Qué es una directiva?
- Tipos de directivas

¿Qué es una directiva?

Una directiva es considerada como una parte muy importante del núcleo de Angular, ya que nos permite extender la funcionalidad del HTML usando, para ello, una nueva sintaxis.

Directivas



¿Qué es una directiva?

Las directivas están en todas nuestras aplicaciones Angular y sirven para:

Modificar
nuestro DOM dentro
de nuestras páginas
o templates



Modificar la
estructura del
DOM (Document
Object Model)



Modificar la
apariciencia de
nuestra aplicación
web

Además, permiten:

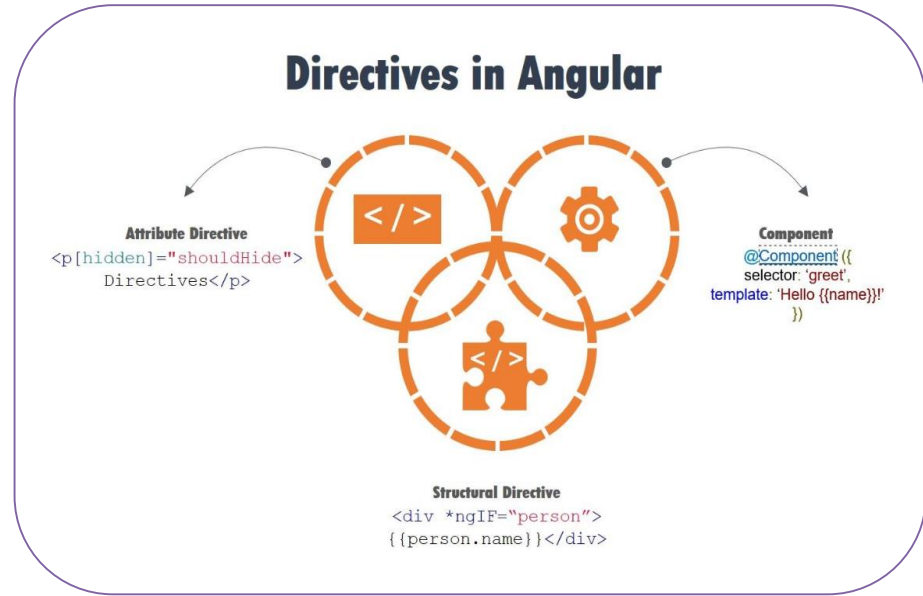
Usar sentencias if y
bucles for



Agregar otro
comportamiento al
código HTML en un
proyecto angular

¿Qué es una directiva?

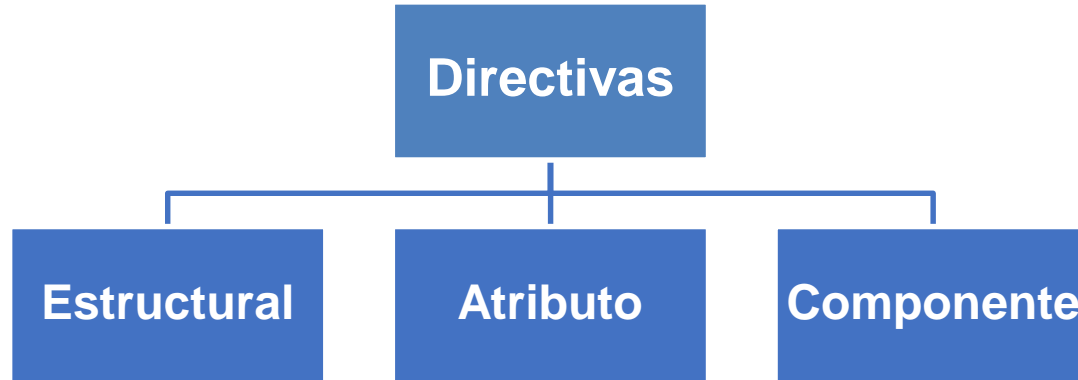
- Cada directiva que usamos tiene un nombre, y determina dónde puede ser usada, sea en un elemento, atributo, componente o clase.



Recuperado de <https://javadesde0.com/introduccion-a-las-directivas-y-tipos-de-directivas-en-angular/>

Tipos de Directivas

Angular ofrece un conjunto de directivas. Actualmente existen tres tipos de directivas:



Tipos de Directivas

Directiva de tipo Estructural

Las directivas estructurales permiten añadir, manipular o eliminar elementos del DOM o elementos HTML.

Directiva de tipo Atributo

Una directiva de atributo cambia la apariencia o comportamiento de un elemento, componente u otra directiva.

Directiva de Componentes

Una directiva de componente es aquella que administra una región de HTML de una manera nativa como un elemento HTML. Técnicamente es una directiva con un template.



CJAVA
siempre para apoyarte

Caso o reto a resolver

Crea directivas en una aplicación angular



Recurso del caso

Para realizar esta actividad, observa los siguientes videos donde se explica cómo crear una directiva en angular.

Videos:

Leifer Mendez. (13 de enero de 2021). *Generando DIRECTIVAS (DIRECTIVE) en ANGULAR 11* [Archivo de video]. Youtube.

<https://www.youtube.com/watch?v=2N-G0ACbAWc>

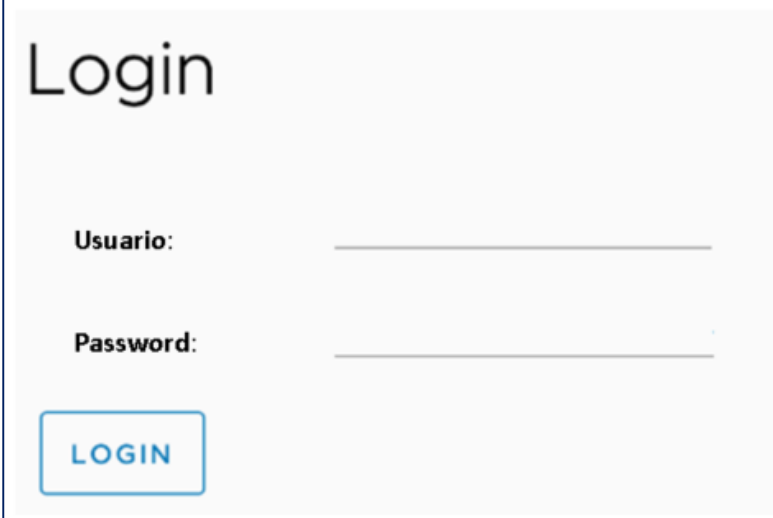
Kevin Davila. (20 de junio de 2022). *¡Crea tu propia Directiva en Angular!* [Archivo de video]. Youtube.

<https://www.youtube.com/watch?v=XQXwDGiDIHc&t=262s>

Indicaciones para realizar la actividad

Para realizar esta actividad, ten en cuenta lo siguiente:

Crea el siguiente layout:



A login form layout within a light gray rectangular container. At the top left of the container is the text "Login". Below this, there are two labels: "Usuario:" and "Password:". Each label is followed by a horizontal input field. At the bottom left of the container is a blue rectangular button with the text "LOGIN" in blue capital letters.

Indicaciones para realizar la actividad

Para realizar esta actividad, ten en cuenta lo siguiente:

- En el archivo del componente principal “app.component.ts” crea una clase con dos propiedades usuario y password. Por ejemplo:

```
export class AppComponent{  
    usuario = 'Angular'  
    password = "123456"  
}
```

- Luego agrega una directiva ngIf para comparar si los datos que el usuario ha ingresado a través del formulario login son iguales a los datos de la clase AppComponent.

Indicaciones para realizar la actividad

Para realizar esta actividad, ten en cuenta lo siguiente:

- Si las credenciales coinciden se debe mostrar un mensaje de bienvenida.
- Si las credenciales no coinciden se debe mostrar un mensaje de credenciales incorrectas.
- Otra funcionalidad que debe tener la aplicación es que, si las credenciales son correctas, se debe mostrar una lista de enlaces debajo del mensaje de bienvenida. Usa la directiva ngFor para generar los enlaces HTML.

Indicaciones para realizar la actividad

Para realizar esta actividad, ten en cuenta lo siguiente:

Login

Usuario:

Password:

ngIf devuelve true ,
si las credenciales
son verdaderas

Bienvenido usuario : Angular

[Home](#)

[Support](#)

[Contact](#)

Utiliza ngFor para
generar los enlaces
HTML

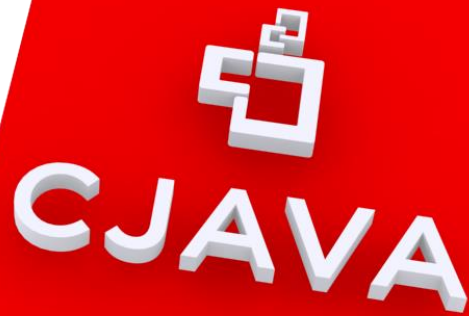
Login

Usuario:

Password:

ngIf devuelve false ,
si las credenciales
son falsas.

Credenciales incorrectas , inténtelo nuevamente.



Formularios en Angular

Edwin Maravi

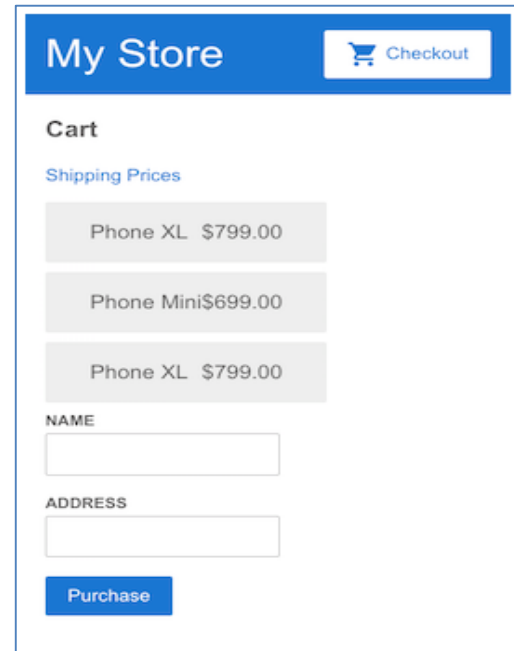
Contenidos o temas

- Introducción a los formularios en Angular
- Template Forms vs Reactive Forms
- Validaciones

Introducción

¿Qué es un formulario?

- Los formularios son una parte integral de una aplicación web.
- Prácticamente todas las aplicaciones vienen con formularios para ser llenados por los usuarios.
- Se utilizan formularios para diferentes actividades, como iniciar sesión o registrarse como usuario, reservar un vuelo, realizar un pedido, etc.

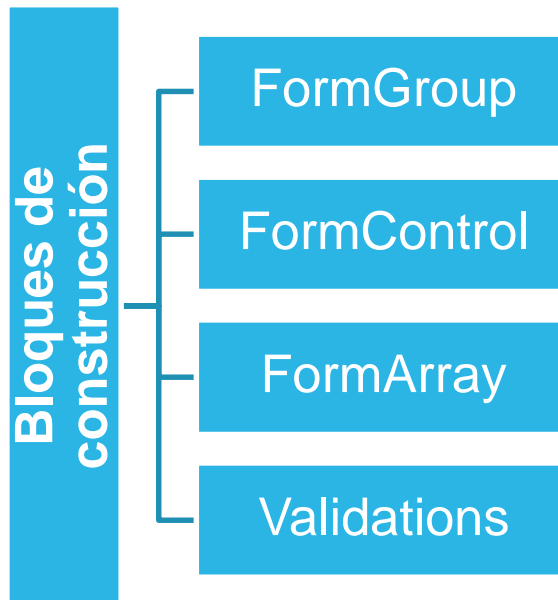


The screenshot shows a web form titled "My Store" with a blue header. In the top right corner of the header is a "Checkout" button with a shopping cart icon. Below the header, the form is titled "Cart". Under "Cart", there is a section for "Shipping Prices" which lists three items: "Phone XL \$799.00", "Phone Mini\$699.00", and "Phone XL \$799.00". Below the shipping prices, there are two input fields: "NAME" and "ADDRESS". At the bottom of the form is a blue "Purchase" button.

Recuperado de
<https://angular.io/start/start-forms>

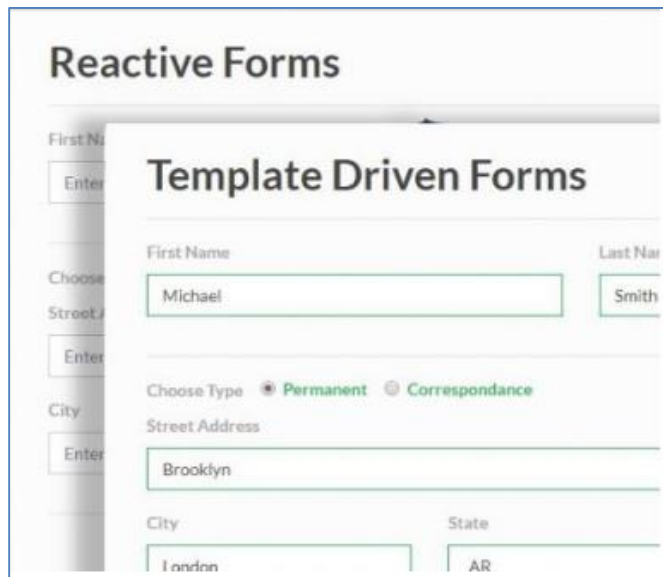
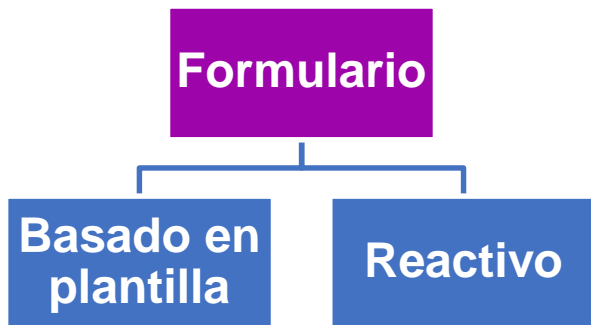
Introducción

Los componentes básicos de un formulario son:



Template Form vs Reactive Form

- Angular proporciona **dos enfoques diferentes** para manejar la entrada del usuario a través de formularios:



The screenshot shows a web form with the following fields and controls:

- First Name:** Text input field containing "Michael".
- Last Name:** Text input field containing "Smith".
- Choose Type:** Radio button group with "Permanent" selected and "Correspondance" unselected.
- Street Address:** Text input field containing "Brooklyn".
- City:** Text input field containing "London".
- State:** Text input field containing "AR".

Recuperado de
<https://cookiecorporation.com/validacion-de-contrasena-angular-reactive-forms/>

Template Form vs Reactive Form

La tabla describe las diferencias claves entre un formulario reactivo y un formulario basado en plantillas:

	Reactivo	Basado en plantillas
Configuración del modelo de formulario	Creado en la clase de componente	Creado por directivas
Modelo de datos	Estructurado	No estructurado
Flujo de datos	Sincrónico	Asincrónica
Validación del formulario	Funciones	Directivas

¿Cómo crear formularios en Angular?

Para crear formularios en Angular, usaremos el siguiente comando:

```
ng new angularforms
```

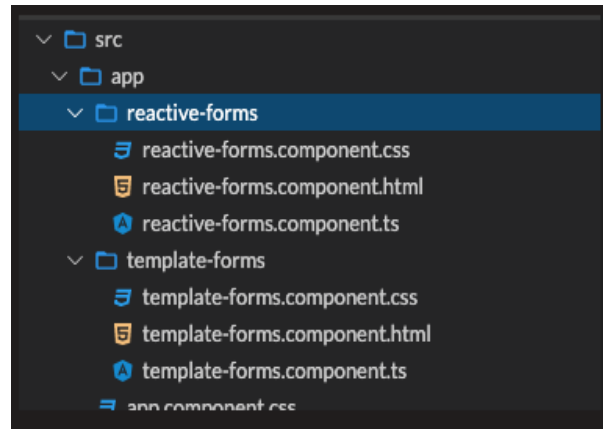
Según el tipo de formulario usaremos:

Formulario reactivo

ng generate component reactiveforms

Formulario de plantilla

ng generate component templateforms



Recuperado de
<https://mugan86.medium.com/formulario-s-en-angular-diferencias-template-y-reactive-forms-e37af5e30b81>

Validaciones en formularios

Las validaciones en Angular son consideradas como la parte integral de la gestión de cualquier conjunto de formularios. Veamos algunas características:

HTML5

Se basan en validaciones de HTML5.

Entrada de usuario

Validan la entrada del usuario en cuanto a precisión e integridad.

Cliente

Todas las validaciones se realizan en el lado del cliente.

Personalización

Angular proporciona un conjunto de validadores incorporados, así como la capacidad de crear validaciones personalizadas.

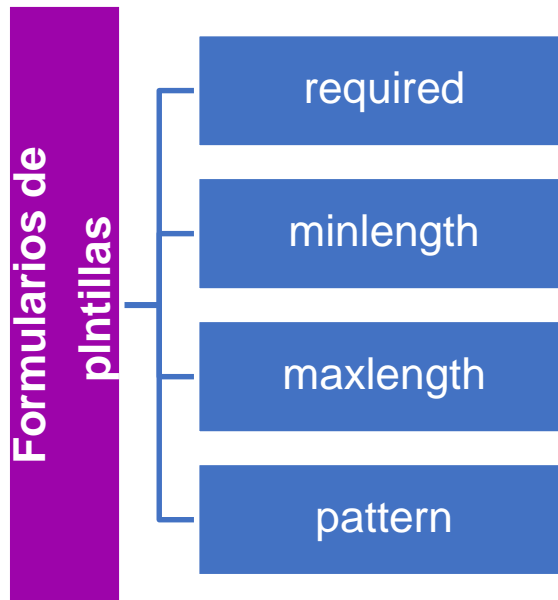
Validaciones en formularios

Según el enfoque, las validaciones de formularios se realizan de la siguiente manera:

Formularios basados en plantillas	Formularios reactivos
Están relacionados a directivas de plantilla que proporcionan directivas de validación personalizadas que abarcan ciertas funciones de validación.	Definen funciones personalizadas que reciben un control de validación.

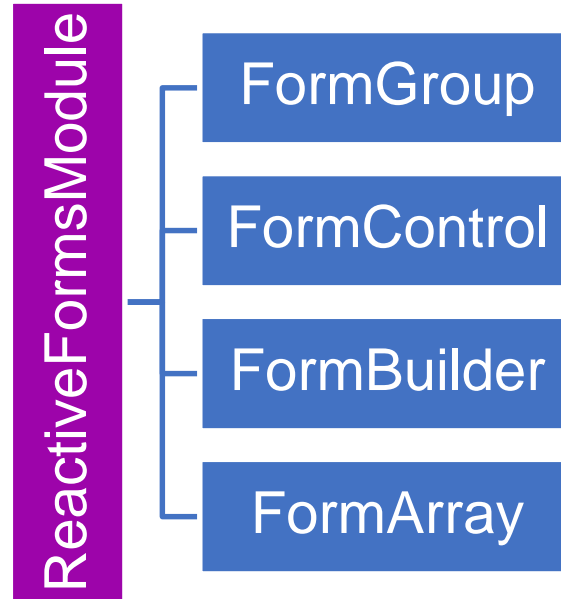
Validaciones en Formularios

- Para agregar validaciones a un **formulario de plantillas**, agregue los mismos atributos que de validación que haría con la validación de un formulario HTML nativo.



Validaciones en Formularios

- Los **formularios reactivos** Angular también se conocen como formularios dirigidos por modelos.
- Emplean una técnica en la que los formularios se diseñan en el componente y luego se realizan los enlaces para el HTML.
- También hacen uso de clases disponibles en el módulo `ReactiveFormsModule`.





Comunicación entre componentes

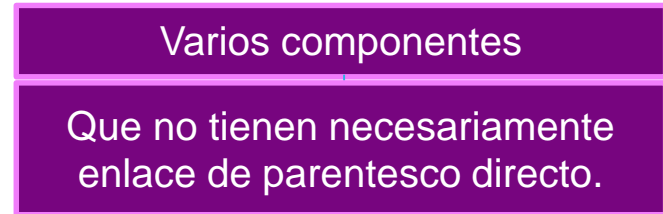
Edwin Maravi

Contenidos o temas

- Introducción
- Comunicación entre componentes
- Comunicación desde el componente padre al hijo mediante @Input
- Comunicación desde el componente hijo al padre mediante @viewChild
- Emisión de eventos personalizados con @output

Introducción

- Las interacciones entre componentes son un aspecto muy importante, porque los componentes están en el corazón de las aplicaciones Angular.
- Sin comunicación entre componentes, los desarrollos serían prácticamente imposibles. Afortunadamente, Angular ofrece muchas posibilidades para que varios componentes se puedan comunicar entre ellos, ya sea:





Introducción

@Input vs. @Output



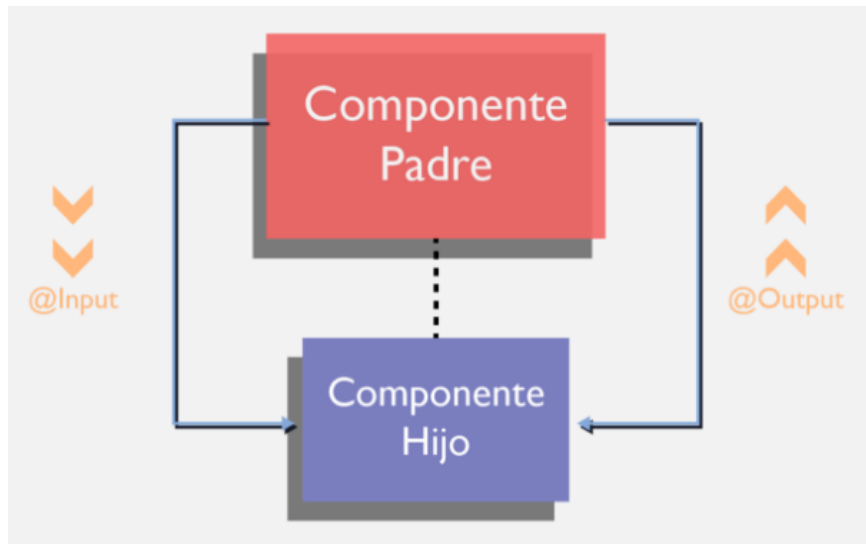
- Son dos de los decoradores más típicos de Angular.
- Permiten comunicar componentes entre ellos.



Recuerda que, habitualmente, una página de Angular está compuesta por varios componentes y en ocasiones estos componentes necesitan comunicarse para pasarse información entre ellos.

Comunicación entre componentes

- Observa la figura, donde se muestra el esquema de comunicación entre el componente padre y un componente hijo.

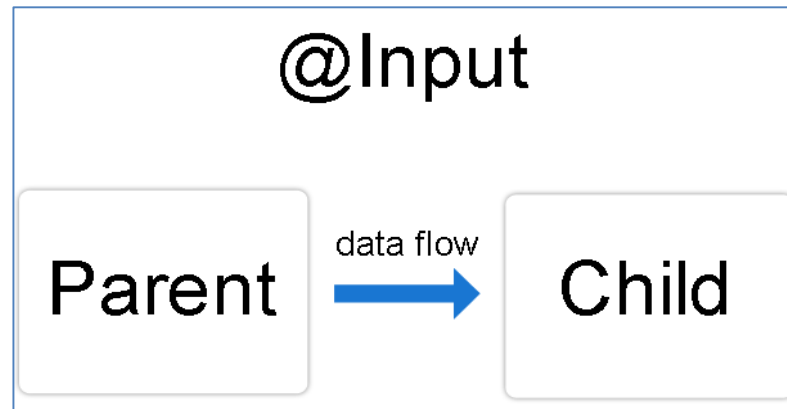


Recuperado de: <https://dev.to/vanessamarely/arquitectura-de-componentes-en-angular-29k8>



Comunicación del componente padre al hijo (@Input)

- Este método es la utilización simple de un input. El componente padre tiene una propiedad que va a transmitir al componente hijo, con ayuda de un input de este componente hijo.
- El paso de información de padres a hijos se realiza por medio de propiedades del componente. La información se brinda desde el template, usando los atributos de la etiqueta del componente.



Recuperado de: <https://angular.io/guide/inputs-outputs>



Comunicación del componente hijo al padre (@ViewChild)

- ViewChild es un decorador que se utiliza sobre una propiedad de la clase, que representa a un componente, permite obtener las instancias de elementos nativos, directivas y componentes que estén en el template del mismo.
- El decorador @ViewChild significa buscar ese elemento dentro de la plantilla del componente. El parámetro que pasamos como primer argumento en ViewChild es el tipo de componente o la referencia que queremos buscar.



Recuperado de: <https://www.c-sharpcorner.com/article/simplifying-viewchild-and-viewchildren-in-angular-5/>



Emisión de eventos personalizados con @output

- Observa la figura donde se muestra la implementación de un evento con el decorador @Output.

Raising an Event (@Output)

product-list.component.ts

```
@Component({
  selector: 'pm-products',
  templateUrl: './product-list.component.html'
})
export class ProductListComponent { }
```

star.component.ts

```
@Component({
  selector: 'pm-star',
  templateUrl: './star.component.html'
})
export class StarComponent {
  @Input() rating: number;
  starWidth: number;
  @Output() notify: EventEmitter<string> =
    new EventEmitter<string>();
}
```

product-list.component.html

```
<td>
  <pm-star [rating]='product.starRating'>
</pm-star>
</td>
```



Emisión de eventos personalizados con @output

- Para lograr la comunicación de hijos a padres, están involucrados dos componentes:

El componente hijo será el encargado de escalar el evento hacia el padre, para avisarle de un suceso. Al avisarle, el hijo podrá comunicar un dato que el padre deba conocer, relacionado lógicamente con ese suceso.



El componente padre será capaz de capturar el evento personalizado emitido por el hijo y recuperar aquel dato que fue enviado.



Emisión de eventos personalizados con @output

- Para implementar eventos personalizados, utilizamos los siguientes objetos:

Clase EventEmitter

En angular la clase que implementa objetos capaces de emitir un evento se llama EventEmitter.



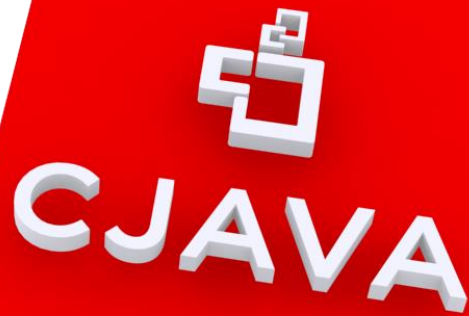
Decorador @Output

La propiedad de tipo EventEmitter, es necesaria para emitir el evento personalizado, debe ser decorada con @Output. Esto le dice al framework que va a existir una vía de comunicación desde el hijo al padre.



Método emit()

Cuando queramos disparar el evento personalizado invocaremos el método emit() del objeto EventEmitter.



Routing

Edwin Maravi

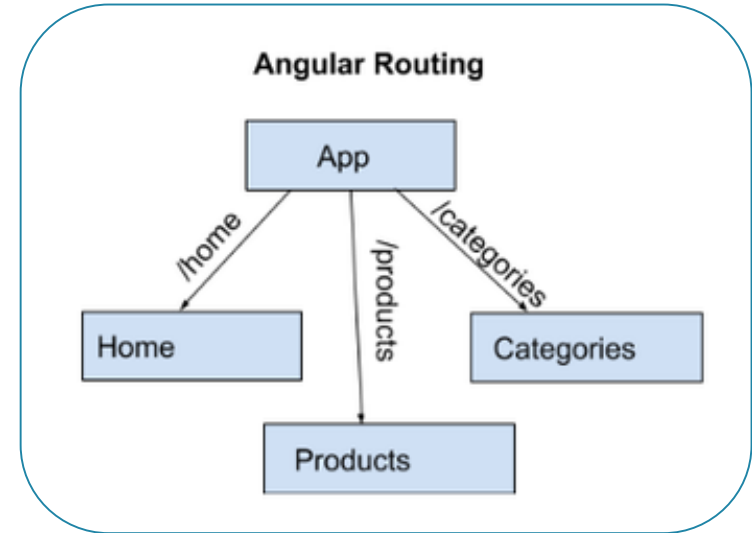
Contenidos o temas

- Sistema de routing
- Configuración de rutas

Sistema de routing

El framework de Angular dispone de un potente sistema de rutas, facilitando la operativa de los sitios de una sola página (SPA). Los elementos básicos son:

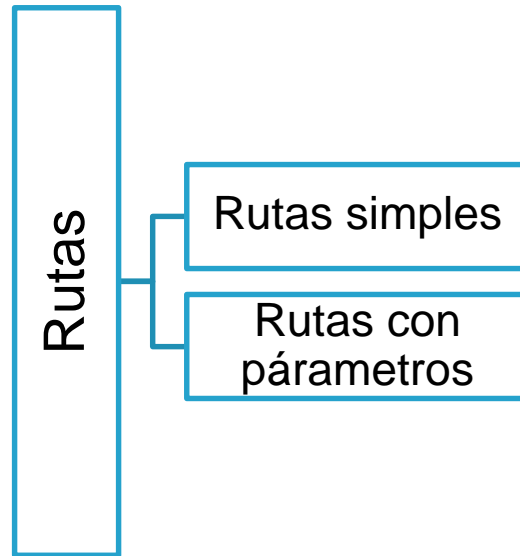
- El módulo del sistema de routing se denomina RouterModule
- Rutas de aplicación
- Enlaces de navegación
- Contenedor



Recuperado de: <https://www.quora.com/What-is-the-purpose-of-Angular-Router>

Sistema de routing

En angular existen dos tipos de routing:



Configuración de rutas – Pasos

Para la configuración de rutas, se debe tener en cuenta los siguientes pasos:

1. Importar el código del sistema de routing: RouterModule y Routes.

2. Agregar la lista de rutas que tendrá nuestra aplicación.

3. Declarar el sistema de routing en el imports del decorador @NgModule

4. Definir la plantilla del componente principal.



CJAVA
siempre para apoyarte

Configuración de rutas

La imagen muestra el mapeo de los componentes del modulo RouterModule en el módulo app.module.ts

"PATH" INDICA EL PATH RELATIVO DE LA URL, O SEA, CUANDO A TU APP LE PEGUEN A "[HTTP://LOCALHOST:4200/TEST](http://localhost:4200/test)" ANGULAR LO VA A MACHEAR CON EL COMPONENTE **TESTCOMPONENT**

NOS IMPORTAMOS EL MÓDULO RouterModule Y LA CLASE ROUTES PARA SER USADO EN ESTA CLASE

"COMPONENT" INDICA EL COMPONENTE QUE SE VA A MACHEAR CON "PATH"

IMPORTAMOS EL MÓDULO RouterModule EN APPMODULE Y AL MISMO TIEMPO LO INICIALIZAMOS CON LA FUNCIÓN "FORROOT()"

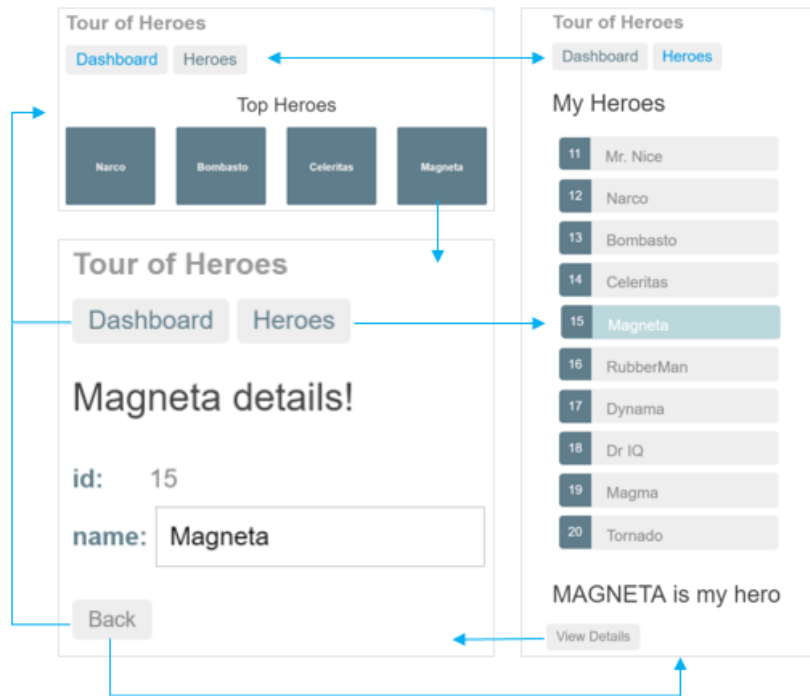
```
app.module.ts | index.html | app.component.html
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3 import { RouterModule, Routes } from '@angular/router';
4
5 import { AppComponent } from './app.component';
6 import { TestComponent } from './test.component';
7
8 const routes: Routes = [
9   {
10     path: 'test',
11     component: TestComponent
12   }
13 ];
14
15 @NgModule({
16   declarations: [
17     AppComponent,
18     TestComponent,
19   ],
20   imports: [
21     BrowserModule,
22     RouterModule.forRoot(routes)
23   ],
24   providers: [],
25   bootstrap: [AppComponent]
26 })
27 export class AppModule {}
```

ESTE ARRAY DE ROUTE SE LO MANDA AL MÓDULO RouterModule PARA QUE SE QUEDE ESCUCHANDO LOS CAMBIOS DE URL EN EL BROWSER

Recuperado de: <https://gustavodohara.com/blogangular/navegar-las-paginas-aplicacion-angular-mucho-mas/>

Configuración de rutas – Ejemplo

La imagen muestra la estructura de una aplicación SPA con enrutamiento:



Recuperado de:
<https://docs.angular.lat/tutorial/toh-pt5>



CJAVA

siempre para apoyarte

Gracias