Índice

[**Qué es Angular: ventajas y cómo aprenderlo** 1](#_Toc146013328)

[**Para qué sirve Angular** 1](#_Toc146013329)

[**Cuáles son las versiones de Angular** 1](#_Toc146013330)

[**Características de Angular** 2](#_Toc146013331)

[**Ventajas de Angular** 3](#_Toc146013332)

[**Desventajas de Angular** 4](#_Toc146013333)

[**¿Cómo funciona Angular?** 4](#_Toc146013334)

[**Qué hacen Angular y TypeScript** 4](#_Toc146013335)

[**Incremental DOM y Angular Ivy vs Virtual DOM** 4](#_Toc146013336)

[**¿Qué conocimientos previos tener para aprender Angular?** 5](#_Toc146013337)

[**¿Qué ofrece Angular para el desarrollo de una aplicación web?** 5](#_Toc146013338)

[**Qué es un SPA** 5](#_Toc146013339)

[**Instalación de Angular CLI** 6](#_Toc146013340)

[**Pero… ¿Qué es un CLI?** 6](#_Toc146013341)

[**Comandos de Angular para correr tu proyecto** 7](#_Toc146013342)

[**Estructura de un proyecto en Angular** 8](#_Toc146013343)

[**Conceptos básicos de TypeScript para usar Angular** 11](#_Toc146013344)

[**Características de TypeScript** 11](#_Toc146013345)

[**String interpolation** 12](#_Toc146013346)

[**División de responsabilidad** 13](#_Toc146013347)

[**Property Binding** 14](#_Toc146013348)

[**Utilidades** 14](#_Toc146013349)

[**Introducción al Event Binding** 15](#_Toc146013350)

[**Otros eventos que puedes escuchar** 16](#_Toc146013351)

[**Data binding con ngModel** 18](#_Toc146013352)

[**Uso de \*ngIf** 19](#_Toc146013353)

[**If … else** 20](#_Toc146013354)

[**Uso de \*ngFor** 20](#_Toc146013355)

[**Índice de iteración** 21](#_Toc146013356)

[**\*ngFor para arrays** 22](#_Toc146013357)

[**Tipado de objetos con interfaces** 24](#_Toc146013358)

[**Uso de \*ngSwitch** 25](#_Toc146013359)

[**Angular Devtools** 26](#_Toc146013360)

[**Otras extensiones para Angular** 27](#_Toc146013361)

[**Estilos a la lista de productos** 27](#_Toc146013362)

[**Preprocesadores de CSS** 28](#_Toc146013363)

[**Class and style** 29](#_Toc146013364)

[**Style binding** 30](#_Toc146013365)

[**NgClass y NgStyle** 30](#_Toc146013366)

[**Crear un formulario** 31](#_Toc146013367)

[**Despliegue con Firebase Hosting** 34](#_Toc146013368)

# **Qué es Angular: ventajas y cómo aprenderlo**

**Angular** es un framework de JavaScript que posee todo un ecosistema de tecnologías con módulos y paquetes listos para el desarrollo web del lado del cliente. Esta herramienta por defecto no posee estilos, pero se pueden agregar de forma muy sencilla. La parte visual de la aplicación es muy importante porque es lo que más le interesa al usuario final. En ese sentido, **Angular Material** es un framework de estilos para tus aplicaciones.

## **Para qué sirve Angular**

Desarrollado por Google, **Angular es más que un framework**, es una plataforma que nos da la posibilidad de desarrollar aplicaciones web como aplicaciones móviles. Además, es un framework de estructura que nos va a brindar funcionalidades para extender el template de nuestra aplicación.

Angular se emplea para crear aplicaciones de una sola página y aplicaciones web progresivas, así como para el desarrollo multiplataforma. Su objetivo es simplificar tanto el desarrollo como la prueba de dichas aplicaciones al proporcionar un marco para las arquitecturas del lado del cliente, **modelo-vista-controlador** (MVC) y **modelo-vista-modelo de vista** (MVVM).

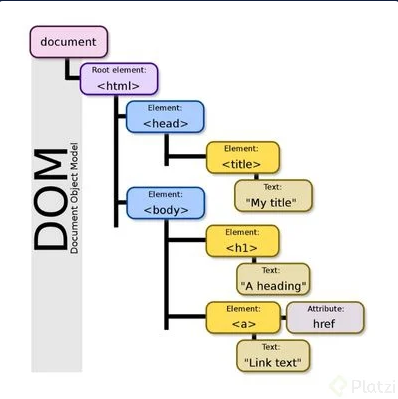
## **Cuáles son las versiones de Angular**

El framework de Angular se va actualizando constantemente, así que antes de decidir qué versión te conviene más, te dejamos el listado con el histórico de versiones que ha tenido angular desde su primera aparición. La versión actual de Angular es 8.2.14. Las versiones anteriores incluyen:

* Angular 7.2.15
* Angular 6.1.10
* Angular 5.2.11
* Angular 4.4.7
* Angular 2.4.10

**AngularJS** fue la primera versión del framework, sucesivamente fue avanzando a nuevas versiones, por ejemplo, Angular 6 marcó un hito en la evolución de esta tecnología en el año 2018. Se trata de una versión importante centrada menos en el framework subyacente y más en la cadena de herramientas y en facilitar el movimiento rápido con Angular en el futuro, como: **ng update**, **ng add**, **Angular Elements, Angular Material** **+** **CDK Components**, **Angular Material Starter Components**, **CLI Workspaces**, **Library Support**, **Tree Shakable Providers**, **Animations Performance Improvements** y **RxJS v6**.

## **Características de Angular**



Estas son las principales características de este framework:

* Está respaldado y desarrollado por Google.
* Utiliza un DOM (Document Object Model) para representar el proyecto desde nodos.
* Es multi-plataforma para desarrollo de aplicaciones web, móviles o de escritorio.
* Tiene excelente experiencia de desarrollo gracias a TypeScript y a herramientas de detección de errores y autocompletado de código propias de Angular.
* Posee una gran comunidad mundial donde sea que te encuentre en todo el mundo.

## **Ventajas de Angular**

Algunas ventajas que trae Angular son:

1. **Fácil de usar**: Angular es fácil de usar y comprender. Tiene una sintaxis simple y es fácil de aprender.
2. **Modular**: Angular es un marco modular, lo que significa que se puede dividir en componentes más pequeños que se pueden reutilizar y combinar para crear aplicaciones más grandes.
3. **Cuenta con una estructura y control de los proyectos**: a esto se suma una gran comunidad para solucionar cualquier problema.
4. **Arquitectura MVC**:Angular utiliza la arquitectura Model-View-Controller (MVC), que facilita el desarrollo y mantenimiento de aplicaciones grandes.



## **Desventajas de Angular**

Sin embargo, Angular también tiene ciertas desventajas.

* **Complejidad**: Angular es un marco complejo y puede ser difícil de aprender para los desarrolladores que son nuevos en los marcos de JavaScript.
* **Rendimiento**: Las aplicaciones angulares pueden ser lentas, especialmente en comparación con otros marcos como React o Vue.
* **Documentación deficiente**: La documentación de Angular no es tan completa.
* **Curva de aprendizaje pronunciada**: Angular tiene una curva de aprendizaje pronunciada y requiere que los desarrolladores tengan una buena comprensión de TypeScript, HTML y CSS.

## **¿Cómo funciona Angular?**

Angular tiene un sistema para crear componentes llamado **Engine Module** o **módulos de Angular**, son contenedores que agrupan componentes y servicios que responden a un mismo dominio (a una misma parte de nuestra aplicación) pero no necesariamente partes físicas, si no a las funcionalidades o procesos que hace nuestra aplicación.

## **Qué hacen Angular y TypeScript**

**Los componentes son la lógica y la interfaz de usuario para cada parte de nuestra aplicación**. Para crear un componente tenemos que crear una clase en Type Script con los métodos que necesitemos para la lógica de nuestro componente, además, en un archivo aparte que parece HTML, pero que no es HTML vamos a definir exactamente los componentes y etiquetas que van a ir en nuestro componente.

Los servicios son agrupaciones de código, de lógica que no necesitamos solo en un componente, sino que los podemos usar en varios componentes por toda la aplicación, estos servicios se los inyectamos a nuestros componentes usando inyección de dependencias.

## **Incremental DOM y Angular Ivy vs Virtual DOM**

**Angular Ivy** se encarga de renderizar nuestros componentes en Angular utilizando una estrategia llamada **Incremental DOM**. Así como React usa Jsx, Angular también tiene su propia sintaxis de HTML para escribir la UI de nuestras aplicaciones, no es HTML puro, tiene variaciones para que podamos iterar en una lista o usar condicionales, lo que hace Angular Ivy es transformar este “HTML” en instrucciones de JS para renderizar nuestros componentes en el DOM.

Con el **Incremental DOM** lo que hacemos es que cada componente se convierte en ciertas instrucciones en código JS y estas instrucciones son las que se ejecutan para renderizar el componente y actualizarlo cuando hay una interacción de los usuarios, en ningún momento se hace una copia del DOM y esto nos ahorra mucha memoria.

## **¿Qué conocimientos previos tener para aprender Angular?**

Se recomienda tener conocimiento básico de Javascript, además de HTML y CSS, y tal vez conocimiento de lo que es TypeScript y sus diferencias con Javascript, ya que todo Angular usa esta tecnología.

## **¿Qué ofrece Angular para el desarrollo de una aplicación web?**

* **Angular Forms**: Poderosa librería lista para la creación de todo tipo de formularios.
* **PWA**: Fácil y rápida integración con PWA para la creación de aplicaciones multi plataforma.
* **Universal**: También denominado como SSR (Server Side Render) para correr en el servidor, aplicaciones de navegador y optimizar el SEO y la velocidad de carga.
* **i18n**: Internacionalización o multi lenguaje.
* **Http**: Cliente Http propio de Angular basado en observables para sacarle el máximo potencial.
* **Material**: Librería para la creación de interfaces de usuario basada en Material Design.
* ¡Y mucho, pero mucho, más!

## **Qué es un SPA**

Las **aplicaciones de página única** (SPA) es una página única (de ahí su nombre) en la que casi toda la información permanece igual y únicamente hay que actualizar ciertos elementos a la vez. Por ejemplo, cuando navegas por tu correo electrónico te darás cuenta de que no hay muchos cambios durante la navegación: La barra lateral y la cabecera permanecen intactas mientras recorres tu bandeja de entrada.

# **Instalación de Angular CLI**

Para comenzar con Angular, en primer lugar, se requiere tener instalado en tu computador **NodeJS** y NPM**.** Puedes verificar dicha instalación con el comando **node -v** y **npm -v**.

**TIP**: Se recomienda utilizar siempre versiones pares de NodeJS (10, 12, 14) ya que a estas se les darán mantenimiento a lo largo de muchos años.

Comienza instalando el CLI de Angular con el comando **npm install -g @angular/cli**. Nótese el **-g** que indica que instalaremos Angular de manera global en tu computador para la posterior creación de tu primer proyecto.

## **Pero… ¿Qué es un CLI?**

CLI son las siglas de Common Line Interface o Interfaz de Línea de Comandos. Son programas para darle ordenes al computador y ejecutar tareas y procesos a través de comandos de texto.

Luego de la instalación del Angular CLI, puedes utilizar **ng version** para corroborar la versión instalada de Angular (actualmente v13). Angular lanza una nueva versión cada 6 meses introduciendo mejoras. No debes preocuparte por utilizar una o varias versiones detrás de la actual. Las versiones de Angular son totalmente compatibles entre sí y no tendrá problemas tu aplicación.

Tu primer proyecto

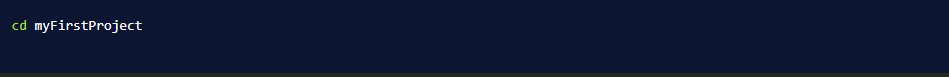
Para crear tu primer proyecto, basta con el comando **ng new myFirstProject**. El CLI de Angular te realizará las siguientes preguntas:

***Would you like to add Angular routing? Which stylesheet format would you like to use?***

Primero pregunta si deseas emplear **Angular Routing** para la creación de rutas en tu aplicación. Se recomienda contestar siempre **Yes**, ya que, rara vez, una app no tendrá rutas. La segunda pregunta que realiza es si se desea utilizar un preprocesador de CSS pudiendo escoger entre SCSS, SASS, LESS o no utilizar un preprocesador. Se recomienda utilizar SCSS ya que es el más utilizado en la industria.

Luego de la creación de tu primer proyecto (puede demorar unos minutos en crearse) es hora de correr el servidor de desarrollo que Angular trae para hacer pruebas.

Ingresamos al directorio de nuestro proyecto:



Y lanzamos el servidor de desarrollo con:



Este comando preparará la aplicación para un entorno de desarrollo, no debe usarse en entornos productivos. Si ingresamos a **localhost:4200/** podrás observar tu primera aplicación de Angular.

# **Comandos de Angular para correr tu proyecto**

El comando **ng serve** se usa para levantar un servidor de desarrollo en nuestro entorno local y probar nuestra aplicación. Puedes complementar este comando con algunas opciones y variantes:

Lanzar servidor y abrir el navegador por defecto automáticamente:



Angular utiliza por defecto el puerto 4200. Si quieres utilizar otro, podemos hacer:



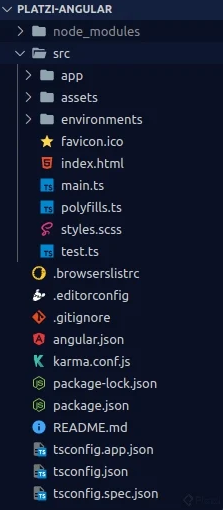
***TIP****: Puedes detener el servidor de prueba con ‘****CTRL + C****’*

Finalmente, si quieres analizar la versión de Angular y la versión de todas las dependencias instaladas, puedes hacer:



# **Estructura de un proyecto en Angular**

Angular posee desde el inicio de un proyecto muchos directorios y archivos que se generan por defecto al crearlo. Mira qué es cada cosa:



De arriba hacia abajo:

* **node\_modules**: Todo proyecto de Javascript posee este directorio donde se almacenan las librerías y dependencias que se descarguen con NPM.
* **src**: Directorio principal del proyecto donde encontramos:
  + **app**: Directorio donde guardaremos todo el código fuente de Angular.
  + **assets**: Directorio para imágenes y otros recursos que la app necesita.
  + **environments**: Directorio de ambientes, por defecto viene con desarrollo y producción.
  + **favicon.ico**: Ícono por defecto que tendrá la pestaña del navegador.
  + **index.html**: Archivo HTML principal desde donde se construye toda la aplicación.
  + **main.ts**: Archivo principal para la configuración de Angular.
  + **polyfills.ts**: Librería para que Angular funcione en navegadores viejos y que la aplicación sea retro compatible.
  + **styles.scss**: Archivo principal de estilos.
  + **test.ts**: Archivo principal para lanzar el ambiente de pruebas de Angular.
* **.**[**browserslistrc**](https://browserslist.dev/?q=bGFzdCAyIHZlcnNpb25z): Lista de navegadores y sus versiones que permite configurar la compatibilidad de la aplicación con cada uno.
* **.**[**editorconfig**](https://editorconfig.org/): Permite autoformatear los archivos, espacios, indentación, etc. Hay que tener instalado la extensión en el editor.
* **.gitignore**: Indicarle a GIT qué archivos/directorios ignorar.
* **angular.json**: Archivo principal con toda la configuración del proyecto Angular.
* **karma.conf.js**: Archivo de configuración de Karma. Karma es un task runner para correr pruebas unitarias.
* **package-lock.json**: Describe el las dependencias exactas que se generaron en la instalación del proyecto.
* **package.json**: Archivo para el manejo de dependencias, scripts y metadatos relevantes para el proyecto.
* [**README.md**](https://tiloid.com/): Archivo markdown para la documentación del proyecto.
* **tsconfig.app.json**: Archivo principal para la configuración de TypeScript.
* **tsconfig.json**: Extensión con más configuraciones de TypeScript.
* **tsconfig.spec.json**: Configuración de TypeScript pero para el ambiente de pruebas.

# **Conceptos básicos de TypeScript para usar Angular**

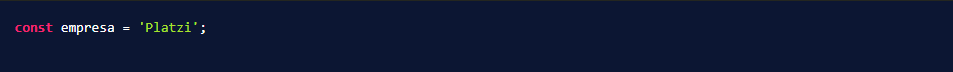
TypeScript es un súper conjunto de JavaScript. Permite escribir código JS utilizando tipado de datos estáticos y clases. Convierte a JavaScript en un lenguaje más firme y seguro, reduciendo la tasa de errores gracias a la detección temprana de bugs.

## **Características de TypeScript**

**Tipado de datos**: Indicar tipo de dato de una variable.



**Inferencia de tipos**: Declaración de variables sin especificar el tipo.



TS automáticamente detectará que la variable es un string y evitará asignar otro tipo de dato.

**Doble tipado**: Asignación de dos tipos de datos a una misma variable.



La variable puede ser tanto del tipo **string** como **number**.

Tipado de parámetros y retornos de una función:



La función **myFunction** espera recibir una variable del tipo **string** y retornará un **number**.

**Clases y POO**: TypeScript le agrega a Javascript la posibilidad de programar Orientado a Objetos.



Para la posterior creación de objetos a partir de esa clase:



# **String interpolation**

**String interpolation** es la manera de enviar datos desde nuestro **componente** hacia **la vista**. Utilizando el doble símbolo de llaves **{{ }}**, o también conocidos como brackets, puedes imprimir el valor de una variable, realizar operaciones matemáticas o hacer el llamado a una función dentro del código HTML.



## **División de responsabilidad**

Un componente de Angular se divide en tres archivos:

* Uno para el código **TypeScript**.
* Otro para el código **HTML**.
* Uno más para el código **CSS**.

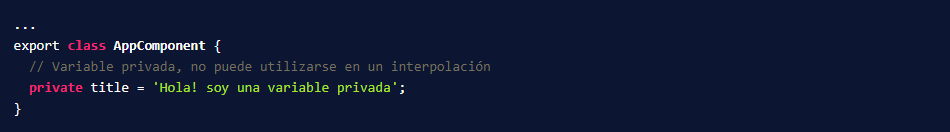


Angular usa el concepto de decoradores para modificar el comportamiento de las clases. La clase **AppComponent** implementa el decorador **@Component()** para indicarle a Angular que esta clase será un componente. Dentro de este decorador, puedes observar el selector del componente (un nombre para el mismo), el template HTML y la hoja de estilos que usará.

Finalmente, dentro de la clase puedes declarar tus **propiedades** y **métodos** como en cualquier clase de la programación orientada a objetos. Tenemos una propiedad llamada **title** que es del tipo **string**. Podemos mostrar el valor de esta variable en el HTML con una interpolación:



Es importante que tengas en cuenta la visibilidad de los atributos y métodos de una clase. Si estos llegaran a ser **private**, no podrás usarlo en el HTML. Las variables deben ser públicas para poder ser compartidas al template.



# **Property Binding**

**Property Binding** es la manera que dispone Angular para **controlar y modificar las propiedades** de los distintos elementos de HTML. Para esto, simplemente utiliza los corchetes **[ ]** para poder modificar dinámicamente ese atributo desde el controlador.

## **Utilidades**

* El atributo **src** de la etiqueta **<img>** para modificar dinámicamente una imagen.
* El atributo **href** de un **<a>** para modificar un enlace.
* El atributo **value** de un **<input>** para autocompletar un valor de un formulario.
* El atributo **disable** de un **<input>** para habilitar/deshabilitar un campo de un formulario.

Si tienes en tu componente:



Puedes modificar el **value** de un campo de un formulario de la siguiente manera:



Se imprime el valor de la propiedad **empresa** como valor de un **<input>** y gracias a la variable habilitado controlas la edición del campo.

# **Introducción al Event Binding**

A lo igual que el **Property Binding** nos permite modificar el valor de los atributos de los elementos HTML desde el controlador, el **Event Binding** permite **controlar los eventos** que suceden en estos elementos. El clic de un botón, detectar cambios en un campo, el envío de un formulario, entre otros eventos. Para esto utiliza los paréntesis **()** para el bindeo de la propiedad del elemento.

Si tienes en tu componente:



Puedes ejecutar el método **enviarFormulario()** cuando se realiza un clic en un botón de la siguiente manera:





# **Otros eventos que puedes escuchar**

Además del evento **clic**, seguramente el más utilizado, hay otros eventos como el **change** para detectar cambios en un campo de formulario, el evento **scroll** para detectar el desplazamiento horizontal/vertical del usuario en el navegador, **onKeyUp** o **onKeyDown** para detectar cuando el usuario aprieta o deja de apretar un botón de su teclado.

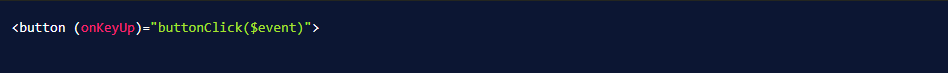
La importancia del **Event Binding** en Angular está dada por la posibilidad de comunicar el componente y la vista, el código TS con el código HTML, intercambiando datos entre ellos.

Puedes enviarle al controlador el evento completo que se produce en la vista. Para esto, solo declara un parámetro **$event** en el método que se encuentra escuchando el evento.

Tienes en el controlador:



Y en el HTML:

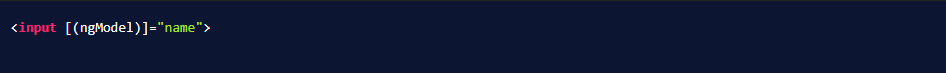


El método **buttonClick()** que recibe como parámetro **$event** del tipo **Event**, en el HTML bindea el evento **onKeyUp** y el método recibe argumento **$event** con el símbolo de pesos delante para que Angular entienda que se trata de un evento.

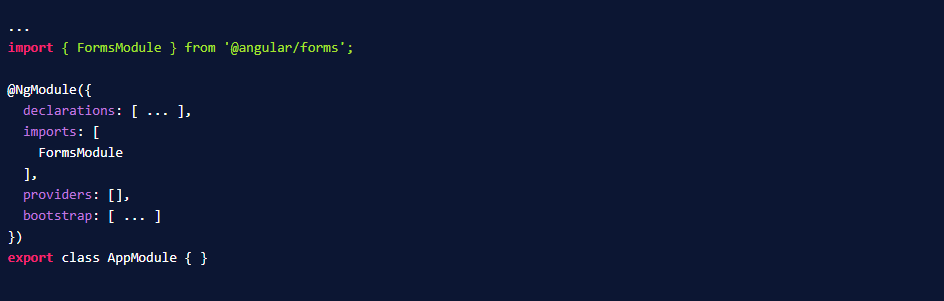
De esta manera, puedes registrar cada pulsación del teclado imprimiendo por consola el evento producido por el usuario.

# **Data binding con ngModel**

El atributo **ngModel** permite el **intercambio de datos de forma bidireccional** entre el componente y la vista. Lo que suceda en el componente, se verá reflejado en la vista. Lo que se suceda en la vista, inmediatamente impactará en el componente.



**ngModel** usar tanto los corchetes **[]** como los paréntesis **()**. De esta manera, se vuelve bidireccional el intercambio de datos. Si no quieres la bidirección, solo colocamos los corchetes [ngModel] para que la comunicación sea unidireccional. Para utilizar **ngModel**, es necesario hacer uso e importar **Angular Forms**. Para esto, dirígete al archivo **app.module.ts** que es el módulo principal de toda aplicación Angular y agrega lo siguiente:



De esta manera puedes importar el módulo **FormsModule** desde **@angular/forms** y agregarlo a **imports** para emplear la propiedad **[(ngModel)]**.

# **Uso de \*ngIf**

El condicional “**If**” es un “**If**” en Javascript, en Java, en PHP, en Python o en cualquier lenguaje. Angular posibilita utilizar este condicionante embebido en el HTML para mostrar o no un elemento. Su sintaxis es algo particular, está compuesta por un asterisco seguido de las iniciales características de Angular “**ng**” y la palabra “**If**”.

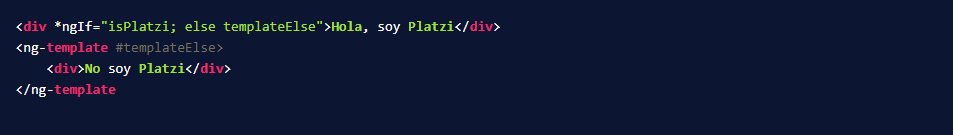


Si la condición dentro del “**If**” se cumple, se mostrará el **<div>** con el respectivo contenido dentro. De lo contrario, el usuario no verá dicho elemento en el navegador. En la condición del **If** puedes colocar cualquier operador lógico:



## **If … else**

Para usar un **else** en Angular, la sintaxis es algo especial. Debes crear un template en tu código HTML usando la etiqueta que provee Angular llamada **<ng-template>** con una Variable de Template, comenzando con **#**, para hacer referencia a este elemento desde tu **If**.



Si la condición del **If** no se cumple, seguido de punto y coma, se coloca la sentencia **else** haciendo referencia a **templateElse**, que es el nombre de la variable del template a mostrar en su lugar.

# **Uso de \*ngFor**

Al igual que con un **If**, Angular permite iterar un array de números, de cadenas de caracteres o de objetos usando **\*ngFor**. Si tienes en tu componente un array de datos:



Puedes mostrar cada elemento iterando el array en un elemento HTML:



El **\*ngFor** crea una variable temporal llamada **str** (o el nombre que más te guste) que contiene cada valor de **myArray**. Finalmente, utilizando una interpolación, muestras el valor de **str**. Quedando tu HTML de la siguiente manera:



## **Índice de iteración**

* **ngFor** también cuenta con un índice con el número de iteraciones. Puedes acceder a este número agregando al **ngFor** **index as i** de la siguiente manera:



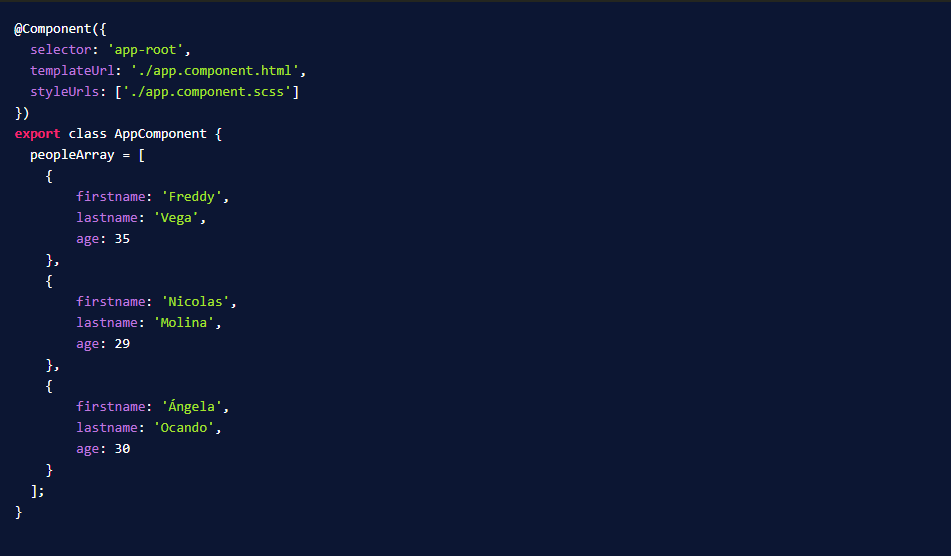
Cada iteración contiene una variable **i** con el índice que le corresponde. Iniciando desde cero, da como resultado:



Más detalles en la documentación oficial [aquí](https://angular.io/api/common/NgForOf).

# **\*ngFor para arrays**

Puedes utilizar **\*ngFor** para iterar y mostrar cada propiedad de un objeto. Considera que en el componente tienes un array de objetos que representan a una persona:



Itera este array en el HTML e imprimimos el valor de cada propiedad de la siguiente manera:



La variable **person** guarda temporalmente el objeto en cada iteración, pudiendo acceder a cada valor usando un punto seguido del nombre de la propiedad.

## **Tipado de objetos con interfaces**

El array **peopleArray** puede contener cualquier cosa, y puede ocasionar comportamientos indeseados en tu aplicación. Puedes crear una interfaz de **Personas** para tipar los objetos del array y asegurar que todos tengan las mismas propiedades.



Tipando el array de la siguiente manera para indicar que el array es de objetos del tipo **Persona**:



# **Uso de \*ngSwitch**

Angular también ofrece la sentencia **\*ngSwitch** y **\*ngSwitchCase** para determinar el flujo de control de tu aplicación y qué elemento mostrar entre múltiples elementos HTML. Además de utilizar un elemento default con **\*ngSwitchDefault** en caso de que ninguna condición se cumpla.



Ejemplo de \*ngSwitchCase

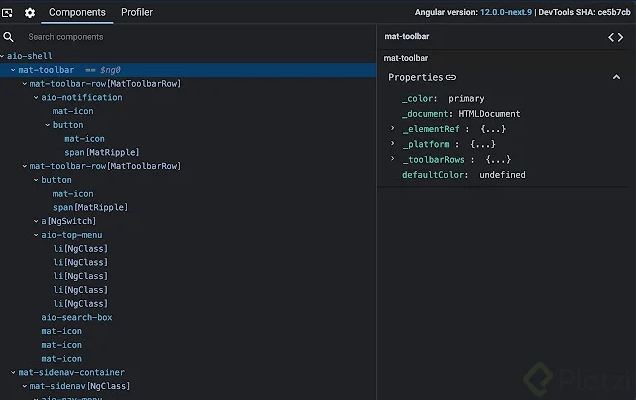


# **Angular Devtools**

**Angular DevTools** es una herramienta complementaria para enriquecer tu experiencia de desarrollo en Angular permitiendo la depuración y el debugging de tu aplicación.

Solo tienes que instalar la extensión del navegador para Google Chrome.

Gracias a estas herramientas, podrás desarrollar y resolver bug más rápido, inspeccionar el valor de las propiedades de tu componente y detectar cuando el usuario realiza alguna acción y se lanza un evento.



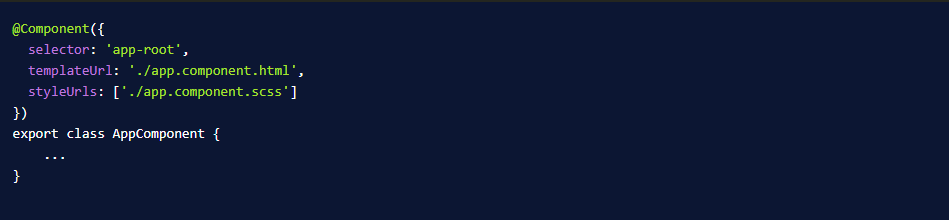
## **Otras extensiones para Angular**

También puedes explorar la extensión **Augury**, una alternativa tanto para Google Chrome como para Mozilla Firefox.

* **TIP**: *Solo puedes utilizar estas extensiones en un ambiente de desarrollo, no funcionan en producción.*

# **Estilos a la lista de productos**

En el decorador de tu componente, verás la propiedad **styleUrls** donde se hace referencia a la hoja de estilos que el componente utiliza.



En este archivo puedes escribir todo el código CSS que tu componente necesita. Por ejemplo, si escogiste SCSS:



## **Preprocesadores de CSS**

Dependiendo el preprocesador que hayas elegido, tus hojas de estilos tendrán una extensión u otra. A excepción de **.css** si no escogiste utilizar un preprocesador.



**TIP**: *Para escoger otro preprocesador, puedes realizar este cambio manualmente en el archivo ‘****angular.json****’ en la raíz de tu proyecto.*

# **Class and style**

La versatilidad de Angular te permite **agregar o quitar clases y estilos** a tus elementos HTML a partir de condicionantes. Así como anteriormente utilizaste los corchetes **[]** para bindear atributos como el **[src]** de una imagen o el **[href]** de un enlace, puedes bindear clases para que Angular las agregue o quite dinámicamente si se cumple una condición de la siguiente manera:



Imagina que tienes en tu componente una propiedad llamada **isActive** que agregará la clase **active-color** si esta es verdadera o quitará la clase si es falsa. Luego ya puedes darle los estilos que más te gusten al elemento HTML en tu hoja de estilos utilizando la clase **active-color**.

## **Style binding**

También puedes añadir estilos **inline** a los elementos HTML bindeando la propiedad **[style]** seguido de la propiedad CSS que quieres modificar dinámicamente:



A partir del valor de **isActive**, dependiendo si este es verdadero o falso, puedes emplear un operador ternario para cambiar el color del párrafo.

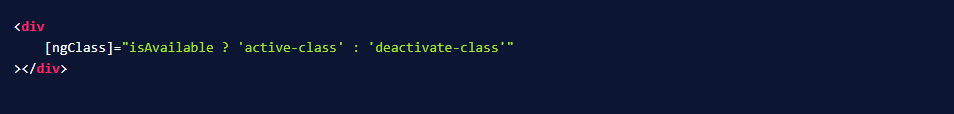
# **NgClass y NgStyle**

Con el binding de **[class]** y **[style]** puedes agregar clases y estilos fácilmente. Pero se vuelve algo complicado en el caso de que necesites agregar varias clases o modificar muchos estilos. Es por esto que Angular ofrece las directivas **ngClass** y **ngStyle** para este propósito.

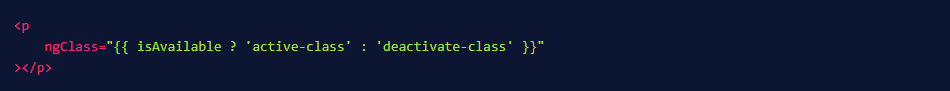
Puedes bindear la directiva **[ngStyle]** o **[ngClass]** y pasarle un objeto con cada propiedad o clase que deseas agregar:



El operador ternario será tu mejor aliado para agregar o quitar clases y estilos:



O puedes usar las Interpolaciones en lugar del binding:



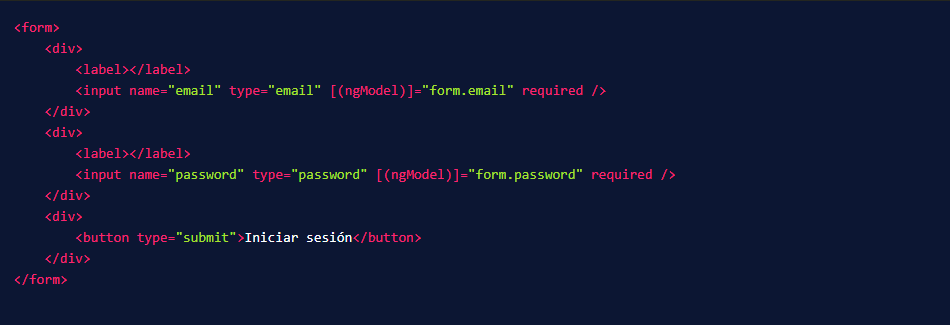
# **Crear un formulario**

Conociendo la directiva **[(ngModel)]** que nos facilita el intercambio de datos de forma bidireccional entre la vista y el componente, puedes crear tu primer formulario apoyándote de esta directiva y de otras características propias de Angular para el manejo y validación de formularios.

**Paso 1**

Crea un simple formulario de Login en el HTML y las variables en el componente para capturar los valores de los campos con **ngModel**:

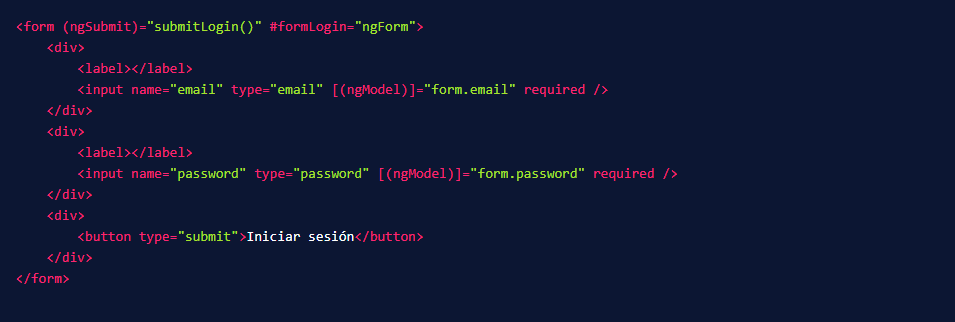




Paso 2

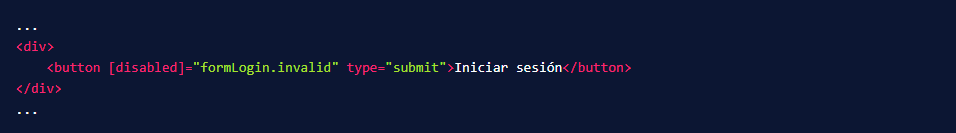
Agrégale al componente un método que responda al evento del envío del formulario llamado **submitLogin()**. Puedes enlazar este método al formulario con la directiva **(ngSubmit)** que va colocada en la etiqueta **<form>** junto con una variable de template para ponerle un nombre al formulario como por ejemplo **#formLogin="ngForm"**. Tienes que igualar el nombre de tu variable a **ngForm** para que Angular reconozca que se trata de un formulario.





Paso 3

Finalmente, gracias a esta conexión de tu formulario con el componente, se activará el método **submitLogin()** al hacer clic en el botón. Para validar tu formulario, utiliza la variable de template para imprimir un mensaje de error en pantalla o deshabilitar el botón de envío de la siguiente manera:



# **Despliegue con Firebase Hosting**

Puedes realizar tu primer deploy de una aplicación Angular utilizando Firebase.

¿Qué es un deploy?

Un deploy, o despliegue, es el proceso de llevar tu software de un entorno local, a un entorno el cual se pueda acceder a través de internet desde cualquier parte del mundo para estar disponible para su uso.

¿Qué es Firebase?

A lo igual que Angular, Firebase es una plataforma de herramientas para desarrolladores de software desarrollado y mantenido por Google. Puedes utilizarlo para, en unos pocos clics, tener tu primera aplicación en un entorno productivo. Además de otros usos que ofrece el servicio

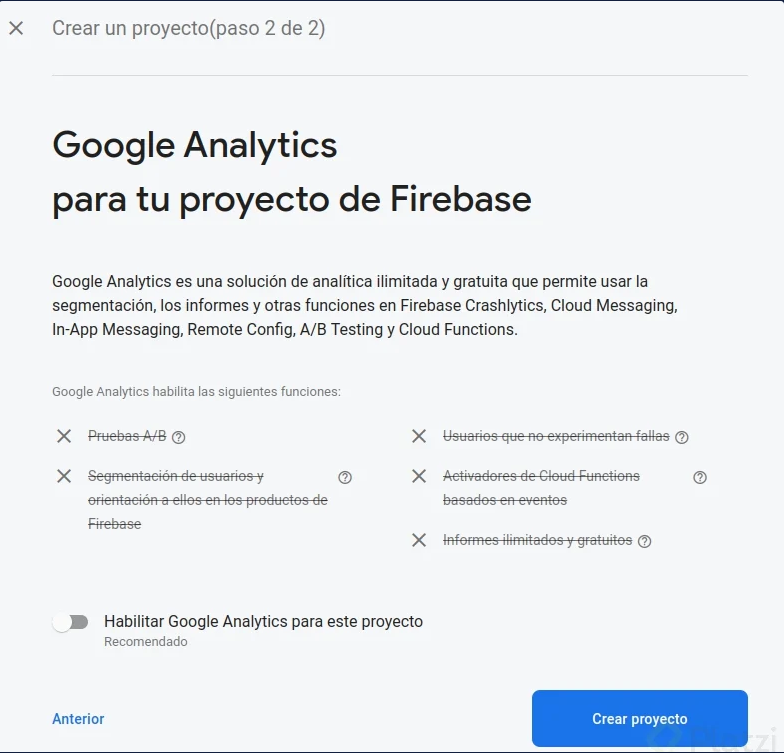
¿Cómo realizo mi primer deploy?

Para esto, [ingresa a la página oficial de Firebase](https://console.firebase.google.com/u/0/?pli=1) con tu cuenta de **@gmail** y comienza creando tu primer proyecto:

Elige un nombre para el mismo:

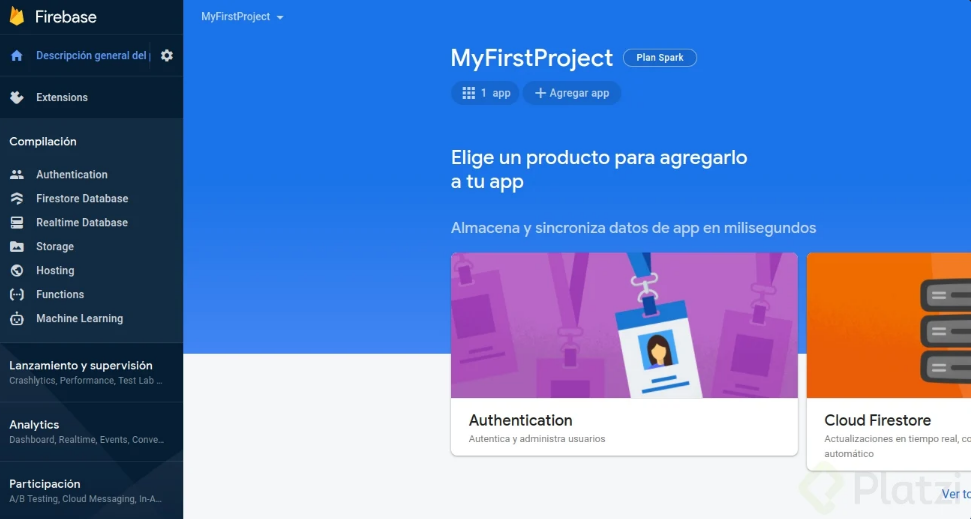


Y elige si utilizar o no Google Analytics en tu aplicación:



**TIP**: *Google Analytics es una herramienta para visualización del tráfico de los usuarios en tu sitio web, comportamiento según audiencia y mucho más. Es una herramienta principalmente utilizada en Marketing Digital.*

Luego de algunos segundos, tu proyecto estará listo.



Firebase ofrece, entre otros servicios:

* Autenticación.
* Hosting.
* Bases de Datos.
* Storage.
* Mensajería.

Para tu primer deploy, usarás los servicios de Hosting que provee un servidor gratuito para tu aplicación. Desde el menú lateral selecciona la opción Hosting.

Firebase solicita que instales algunas dependencias de forma global en nuestro computador. Para eso, desde una terminar utilizarás el comando:



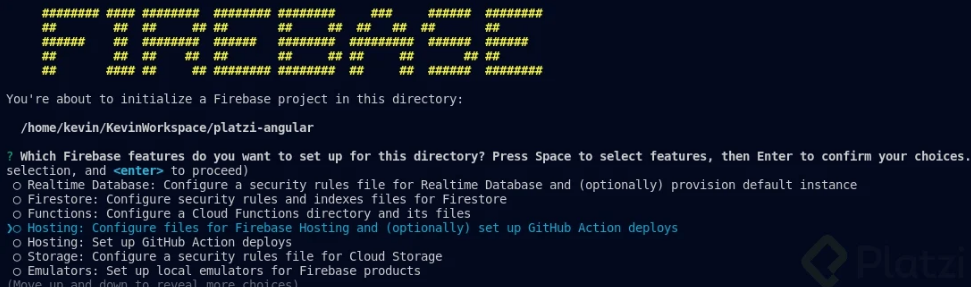
Teniendo el CLI de Firebase instalado, podrás autenticarte para iniciar un proyecto nuevo:



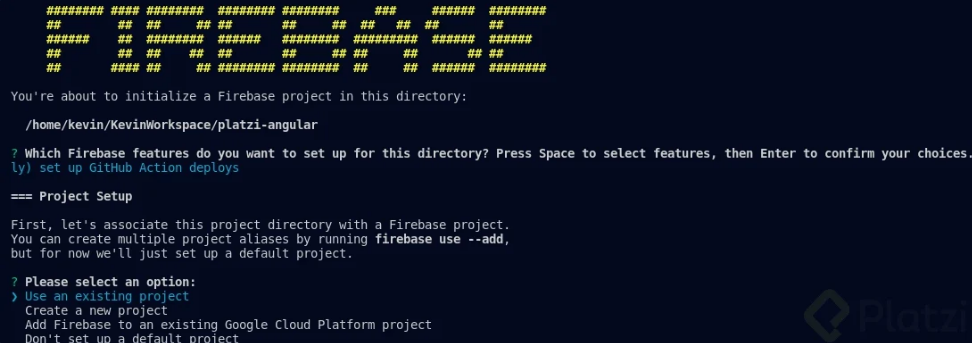
Posteriormente, estando situado desde la terminal en la raíz de tu proyecto, inicializa los servicios de Firebase:



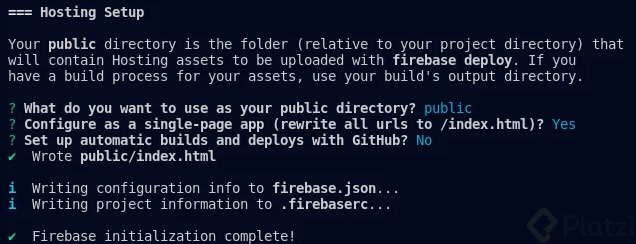
Selecciona la opción **Hosting** para iniciar la configuración de Firebase en tu proyecto:



Elige **Use an existing project** y selecciona el nombre de tu proyecto creado anteriormente desde la web de Firebase:



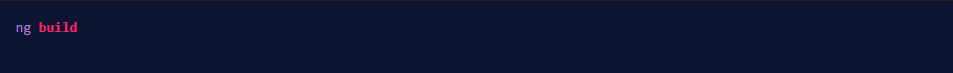
Te recomiendo responder a las preguntas que te realizará el CLI de la siguiente manera para finalizar la inicialización del proyecto:



Se creará en tu proyecto dos archivos, uno llamado **.firebaserc** y otro **firebase.json** que contienen la configuración con el servidor de Firebase con tu proyecto. También creará de una carpeta public con un index.html, que no utilizarás, ya que Angular provee su propio index.html al compilar la aplicación. Puedes borrar este directorio.

Angular build y despliegue final

Con el comando:



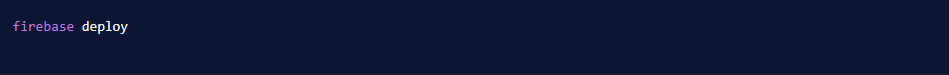
Puedes preparar tu aplicación para un entorno productivo. Creará un directorio llamado **dist** y dentro otro más con el nombre de tu proyecto. Esta es el directorio que Angular provee para la distribución de la aplicación.

Dirígete ahora al archivo **firebase.json** creado al momento de ejecutar la inicialización de Firebase y edita la propiedad public de la siguiente manera:



De esta manera, Firebase desplegará tu aplicación que ya está compilada dentro de ese directorio.

Ahora si es momento del deploy. Para eso, utiliza el comando:



Luego de algunos segundos, Firebase te entregará un URL pública en internet desde la cual podrás acceder a tu proyecto desde cualquier parte.

¡Felicidades, realizaste tu primer despliegue a producción con Firebase!