**JavaScript**

**1. Caracteristícas**

1. Lenguaje de programación.
2. -No tiene nada que ver con Java.
3. -Está orientado a desarrollo web.
4. -No se compila

**2. Llamar a java script**

**2. 1. Archivo externo**

<script src="js/JavaScript.js"></script>

**2. 2. Elemento script**

<script>

alert('Hola mundo desde la página')

</script>

**2.3. Evento del elemento HTML**

<button onclick="history.go(-1)">Atrás</button>

**3. Sintaxis y comentarios**

S: Comienzan por letra, guión bajo o dólar.

Saltos de línea: /r

Var exponencial 10e10: Mete el resultado en exponencial y lo devuelve (10 elevado a 10) al poner exponencial.

Se aconseja poner las palabras completas con Camel Case, es decir empezando siempre en minúsculas, y poniendo mayúsculas para indicar una nueva palabra. Ejemplo: mayorDeEdad

Sintaxis

Solo se tiene en cuenta el primer espacio

Distingue mayúsculas de minúsculas

Las líneas de código finalizan opcionalmente con j

Comentarios:

En línea //

En bloque /\* \*/

**4. Variables**

Las variables son espacios de memoria reservados para guardar información durante la ejecución de un programa. Como su propio nombre indica dicha información puede ir variando.

Poseen un identificador (nombre) y un valor (contenido). Podemos declararlas así:

edad =44

O así:

var edad = 20

También puedes inicializar la variable vacía, así:

var nombre;

Otra forma, se usa en casos concretos:

let nombre;

4.1.Tipos de dato

-string

-números

-boleanos

-null (Tiene espacio de memoria, pero el valor es nulo)

-undefined: No está definida y no ocupa espacio en memoria

-array (Conjunto o cadena de datos)

Arrays

var ejemploArray = [“Pera”,”Manzana”,”Naranja”]

ejemploArray[1]

Devolvería manzana porque está indexado al cero.

1. **Constantes**
2. const numero = 8
3. document.write(numero);

**6. Operadores**

**6.1.Aritméticos**

var x =2

var y =3

var z = x+y

Si pones letras para hacer una operación te sale NaN, es decir, not a number

**6.2. Operadores comparativos**

nombre1 == nombre2

nombre1!= nombre2

Exactamente igual:

nombre1 === nombre2

Declara igualdad en todo, tanto el valor como el tipo de datos.

// valor1 = 24

// valor2 ='24'

// valor1==valor2

// true

// valor1===valor2

// false

// valor1!=valor2

// false

// valor1!==valor2

// true

valor2 > valor1 && valor1 == valor2

valor2 > valor1 && valor1 != valor2

valor2 > valor1 || valor1 == valor2

En el último caso con que una de las dos condiciones se cumpla es suficiente, equivale a “o”.

mayorEdad =! mayorEdad

Para cambiar una variable a su contrario, en el caso de datos booleanos.

password == “pepe” ? console.log(‘Bienvenido’) : console.log (‘Contraseña errónea’)

condición a evaluar ? instrucción para true : instrucción para false

**6.3. Operadores de asignación**

valor1 += valor2

Le suma lo que contenga el valor 2

**Paso por valor y paso por referencias**

valor (string números y booleanos)

var usuario= 'Ivan'

var nombre = 'Pedro'

var usuario = nombre

nombre= 'Pedro'

referencia (arrays y objetos)

frutas = ['Pera', 'Manzana', "Caqui"]

postres = frutas

frutas[1] = 'Plátano'

typeof frutas

Te devuelve el tipo que tiene la variable (object, number, string…)

**7. Estructuras condicionales**

Expresiones if (expresión){

}

if (a>b){

Console.log(‘a es mayor que b’)

}

Var nombre = prompt(‘Escriba su nombre’)

if (a>b){

console.log(‘a es mayor que b’);

console.log (‘El valor de a es: ’ + a)

}

for (valor inicial; expresión; actualización) {

instrucción a ejecutar

}

Ejemplo Para incrementar una variable hasta cierto punto:

for ( i = 0; i < 10; i++){

console.log( "i es: " +i)

}

for (i=0;i<5;i++) {

escribe = prompt ('¿Como te llamas?','Escribe tu nombre')

if (escribe == 'basta') break;

}

**7.1. Para arrays u objetos**

Mostrar uno por uno todos los elementos del array

var dias = ['Lunes', 'Martes', 'Miercoles', 'Jueves', 'Viernes', 'Sábado', 'Domingo']

for (dia in dias) { console.log(dias[dia])}

**8.Angular**

Es un framework basado en librerías de desarrollo de código en el cliente y en el servidor.

Servidores de base de datos ------- Servidores de aplicación ----- Cliente

NodeJS

1. Instalar nodejs para construir un servidor local.
2. Instalación de TypeScript (Es un lenguaje diferente a JavaScript)

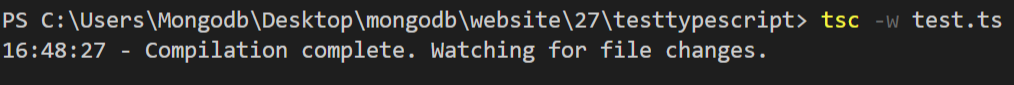
-Sintaxis ---🡪 Nombredelavariable : tipodato = valor;

-Añade tipado estático y objetos basados en clases

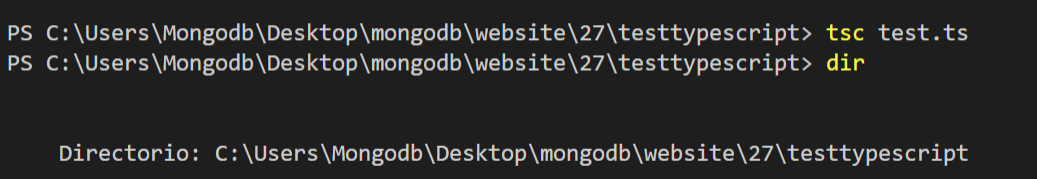
Añadir el tipado estático supone que después no se pueda cambiar el tipo del mismo. Por ejemplo, si ponemos “Juan” en una variable a la que hemos asignado el tipo número, nos dará error.

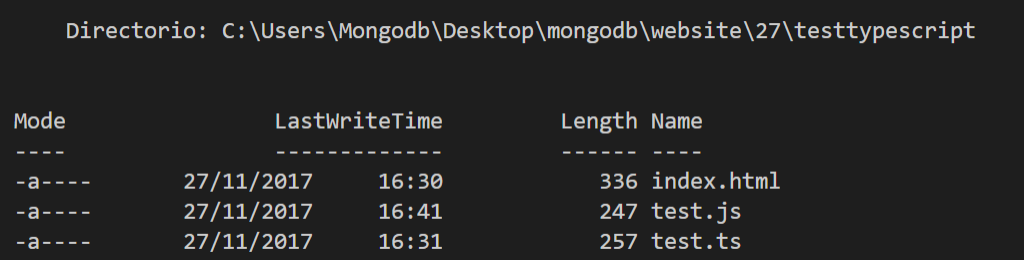
1. Instalar material icon theme en visual basic
2. Para poder ver el .ts tenemos que transpilarlo. Dandole a ver, terminal integrado o control ñ nos sale la consola.
3. Lo transpilamos con el siguiente cógido: tsc test.ts

-Usando tsc –w test.tst puedes compilar los cambios.



1. Con este comando ya tenemos nuestro archivo javascript creado.





El concepto de clase consiste en la creación de un molde para una serie de objetos.

**Angular CLI**

Herramienta de línea de comandos. Antes se empleaban unos archivos denominados package.json.

Instalar angular cli:



Los proyectos en angular se estructuran en modulos y componentes

Angular

--Modulos---------

| componentes |

----------------------

Cada componente tiene:html, css y tipescript

Data binding

Consiste en enlazar el archivo de ts con el html.

Ts 🡨-------------------------------------------🡪 html

Interpolación

Clase componente -----------------🡪 Plantilla {{ }}

-variable (Ejemplo carlos)

-objeto

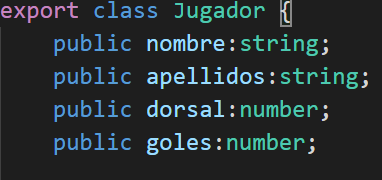
-expresión javascript

Vamos a empezar con el primero, que va en sentido unidireccional.

Se denomina interpolación.

Vamos a crear un modelo con un objeto para que se le aplique la clase.

Podemos crear clases para hacer un objeto a partir de ahí. Para ello la creamos y debemos poner “export” para poder usarla en otro archivo.



Luego, para usarla en el otro archivo ts debemos ponerle delante import



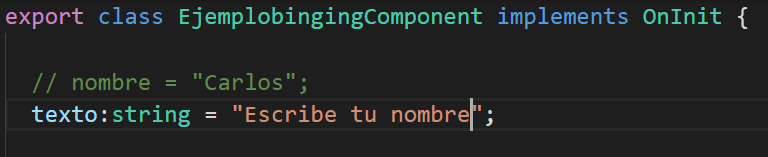
Así pillamos la clase del jugador.

Otra de las opciones para utilizar la interpolación es emplear expresiones JavaScript.

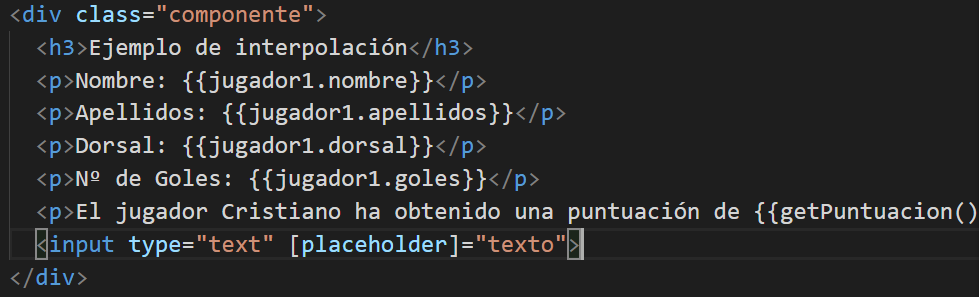
Property Binding

Ejemplo:

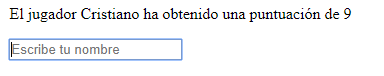
Ejemplobinging.component.ts



Ejemplobinging.component.html <input>



Resultado:

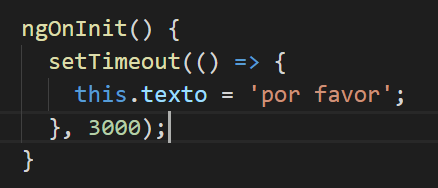


Ejemplo2:

nGoInit

Cuando se inicializa el componente, ejecuta el código que tenga dentro.

En este caso, es una función flecha que se ejecuta a los 3 segundos, sustituyendo en el tiempo indicado el valor de “texto” en el input. Como el objeto texto es el que estamos creando actualmente, usamos .this



Ejemplo3:

Event Binding

Tenemos que coger un evento e igualarlo a una expresión JavaScript

Two-way Binding

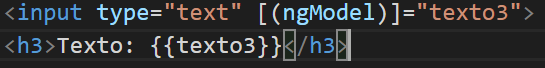
De la clase a la plantilla y de la plantilla a la clase.

Clase 🡨--------------------------------------------🡪 Plantilla

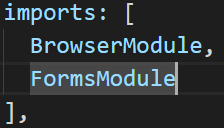
Conecta la clase y el componente en tiempo real.

Ejemplo:

Ejemplobinging.component.html



App.module.ts



import { FormsModule } from '@angular/forms';

Ejemplobinging.component.ts

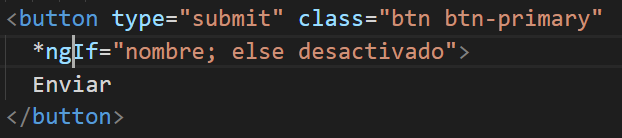


Nglf

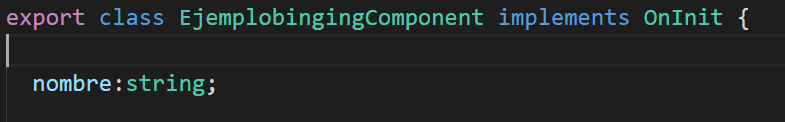
\*ng refiere a la variable nombre. If se refiere a que comienza una especie de minibucle que quiere decir: Si la variable nombre en el input tiene contenido, pues el botón está activado, por el contrario se muestra desactivado.

1.

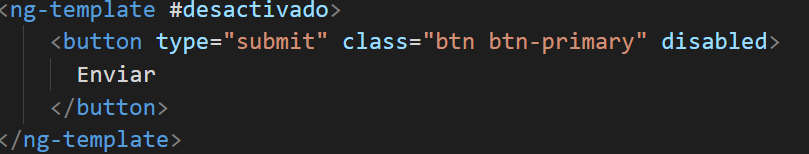
Ejemplobinging.component.html



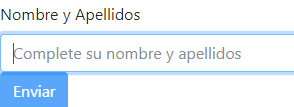
La variable está tipeada en **Ejemplobinging.component.ts**

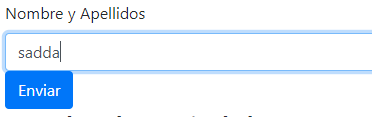


En el caso de estar desactivado, se ejecutaría el siguiente código, es decir, se desactivaría el botón:



Este es el resultado:





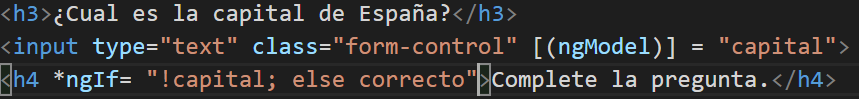
Como hemos podido observar, #desactivado sirve como ancla.

**2.**

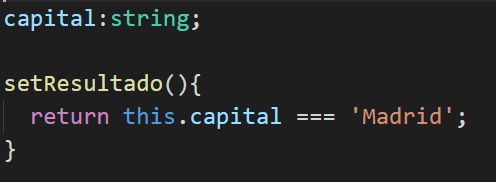
Con el !, que significa distinto, indicamos que si capital no tiene valor, entonces nos indique que completemos la pregunta.

En el input vemos como el ngModel se iguala al contenido de capital, es decir, se recoje la variable en lo que metas por teclado.

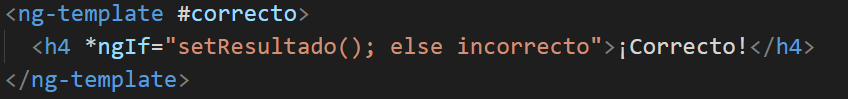
**Ejemplobinging.component.html**



En **Ejemplobinging.component.ts** tipamos la variable de capital, además de declarar la función que nos indicará el resultado correcto de “capital”.

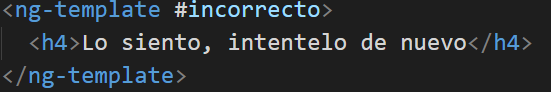


Así, si introducimos “Madrid”, nos saltará al ancla “correcto” **Ejemplobinging.component.html**

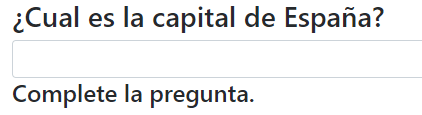


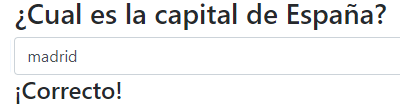
Asimismo, esto permite introducir la función previsamente declarada, mostrar el titulo “Correcto” y como excepción nos manda a #incorrecto.

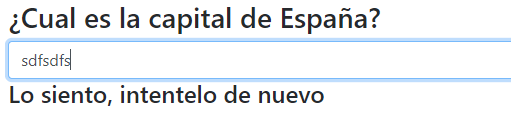
**Ejemplobinging.component.html**



Resultados:

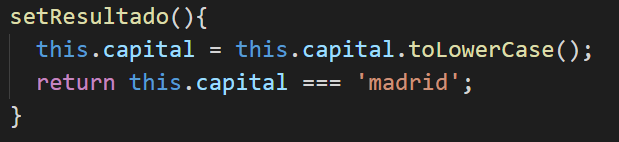






Tenemos también una conversión automática a minúsculas para que no haya error si el usuario lo introduce con minúsculas:

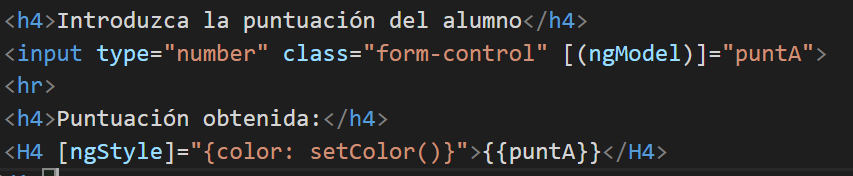
**Ejemplobinging.component.ts**



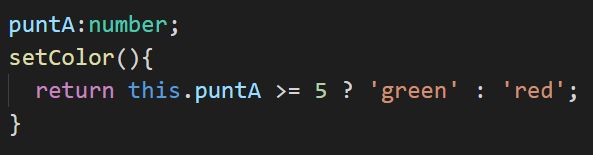
3.

ngStyle nos permite poner propiedades de css y regularlas con expresiones de JavaScript.

**Ejemplobinging.component.html**



**Ejemplobinging.component.ts**



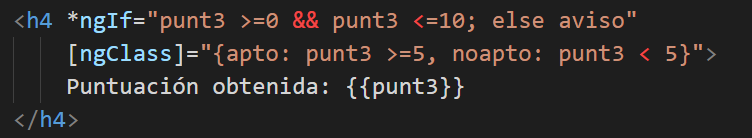
En esta función le indicamos que ponga el contenido de esa variable de un color u otro según sea mayor o menor de cinco.

**ngClass**

Es para meter clases a las que podemos aplicarles propiedades css.

Por ejemplo:

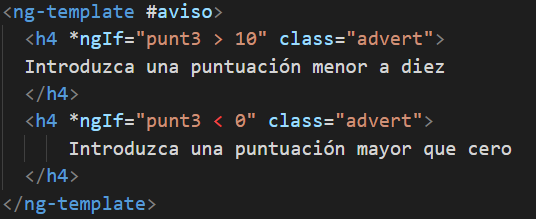
**Ejemplobinging.component.html**

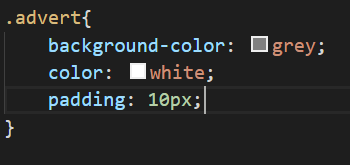


Aquí nos indica también el id aviso al que saltamos luego en caso de cumplir con dichas condiciones.

**Ejemplobinging.component.css**





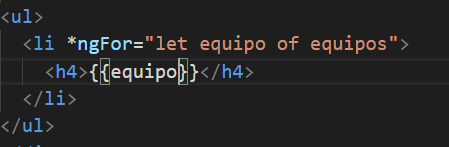


ngFor

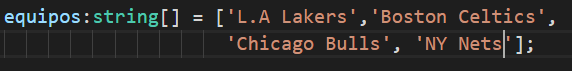
Funciona como un for pero permite ponerlo dentro del elemento.

Muestra uno por uno cada equipo del array equipos declarado en el ts.

**Ejemplobinging.component.html**



**Ejemplobinging.component.ts**



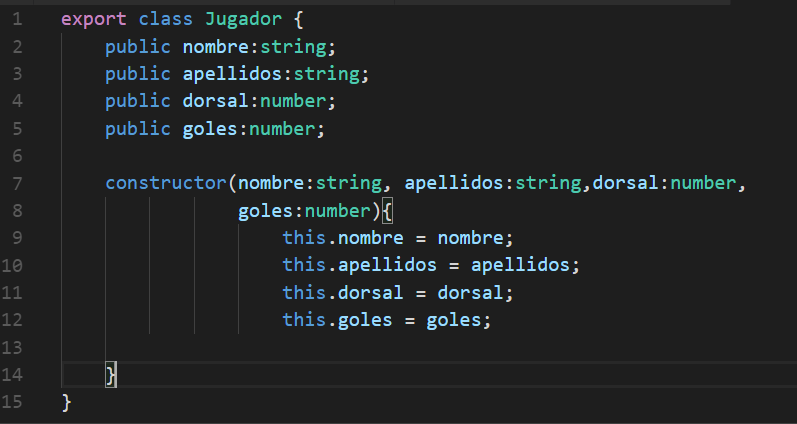
Componentes

Para crear un componente, ng generate component copyright

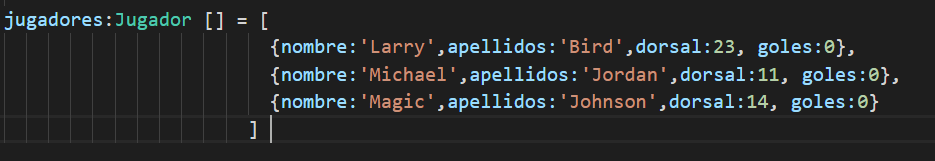
–spec false

Array de objetos

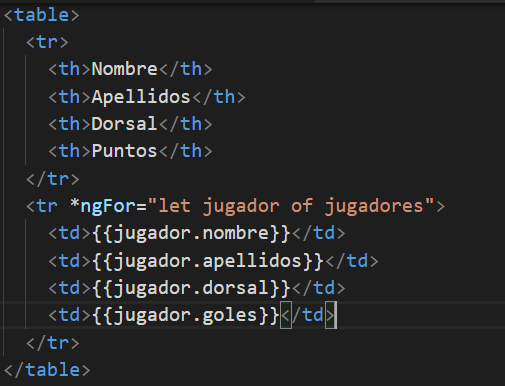
**src\app\modelos\jugador.model.ts**



**Ejemplobinging.component.ts**



**Ejemplobinging.component.html**



Ng Switch

(copiar apuntes)

Paso de valores entre componentes

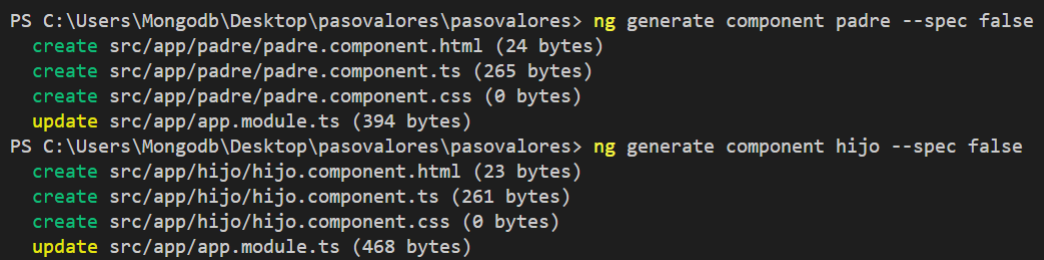
@input

Creamos un nuevo ejercicio en la carpeta del escritorio pasovalores.

ng new pasovalores –Crear el proyecto

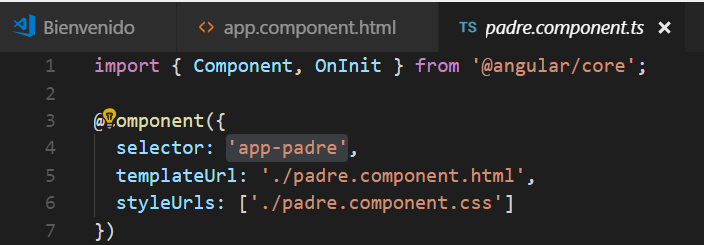
ng server – Arrancar el servidor desde la carpeta en visual studio

Creamos estos dos componentes.



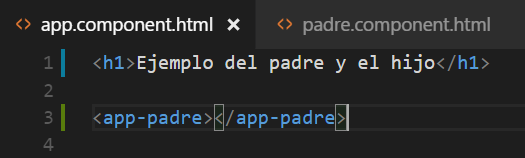
Nos vamos al ts del componente padre, copiando su selector

**padre.component.ts**



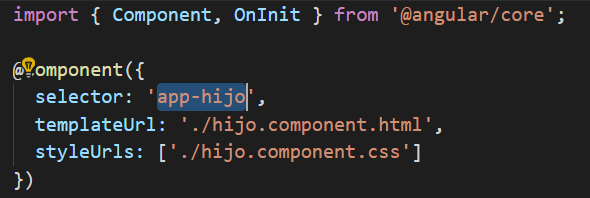
Pegamos este selector como etiqueta en el html en

**app.component.html**



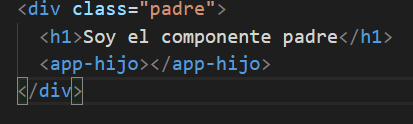
Nos vamos al componente hijo y copiamos su selector

**hijo.component.ts**

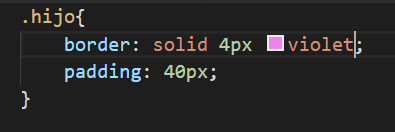


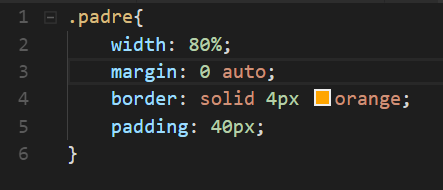
Pegamos su selector en el html del padre

**padre.component.html**



Cada clase tiene sus respectivos css





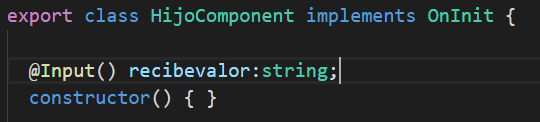
Hemos declarado una variable en el componente padre



Dentro del componente hijo, hemos usado la función input, previamente declarada, para crear una variable

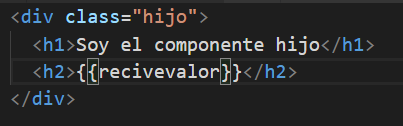
**hijo.component.ts**





Dentro del html del componente hijo, introducimos la variable recibevalor declarada en el ts

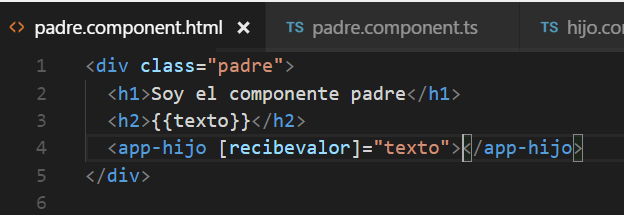
**hijo.component.html**



Aquí recogemos el valor de la variable texto con la variable previa recibevalor.

Como la hemos igualado a texto, esta es la variable del padre que se importará al hijo.

**padre.component.html**



Este es el resultado:

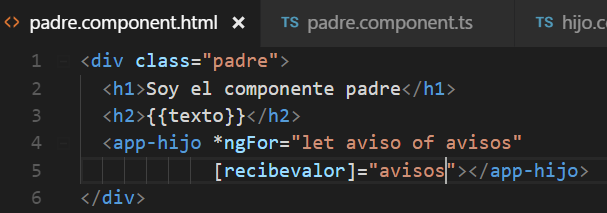


Introducimos el array avisos en

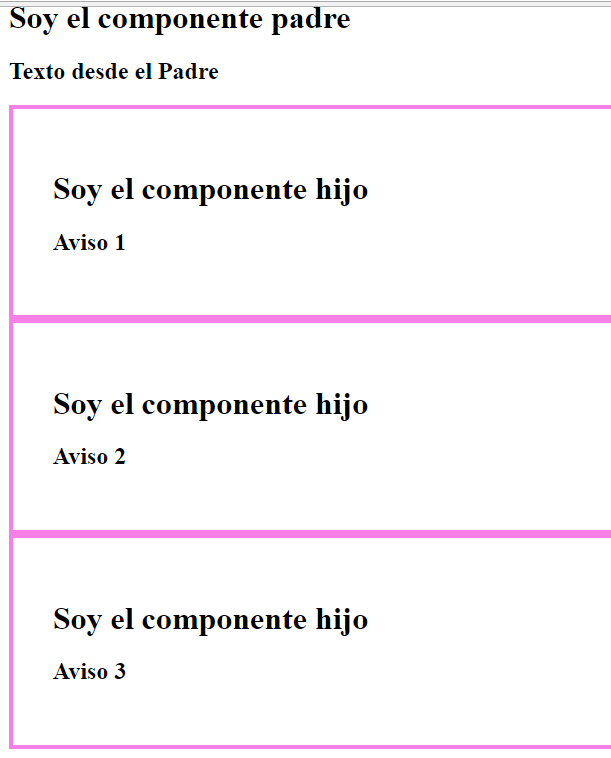
**padre.component.ts**



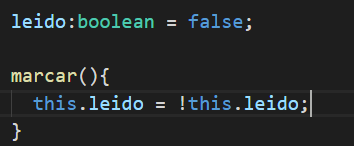
Si introducimos un \*ngFor declarando la variable aviso dentro de avisos, hijo con todas sus propiedades css nos mostrará un elemento con cada elemento del array



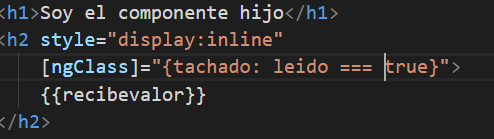
Este es el resultado:



Declaramos la variable leído y una función que pase a lo contrario de la función leído creada ahora mismo, creando así un conmutador que se activará

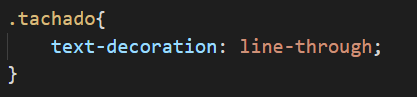


Aplicamos una clase que se ejecuta únicamente cuando leído es true.

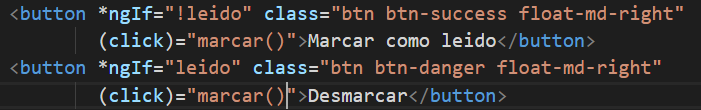


Esta clase previsamente creada le aplicará el estilo tachado al texto.

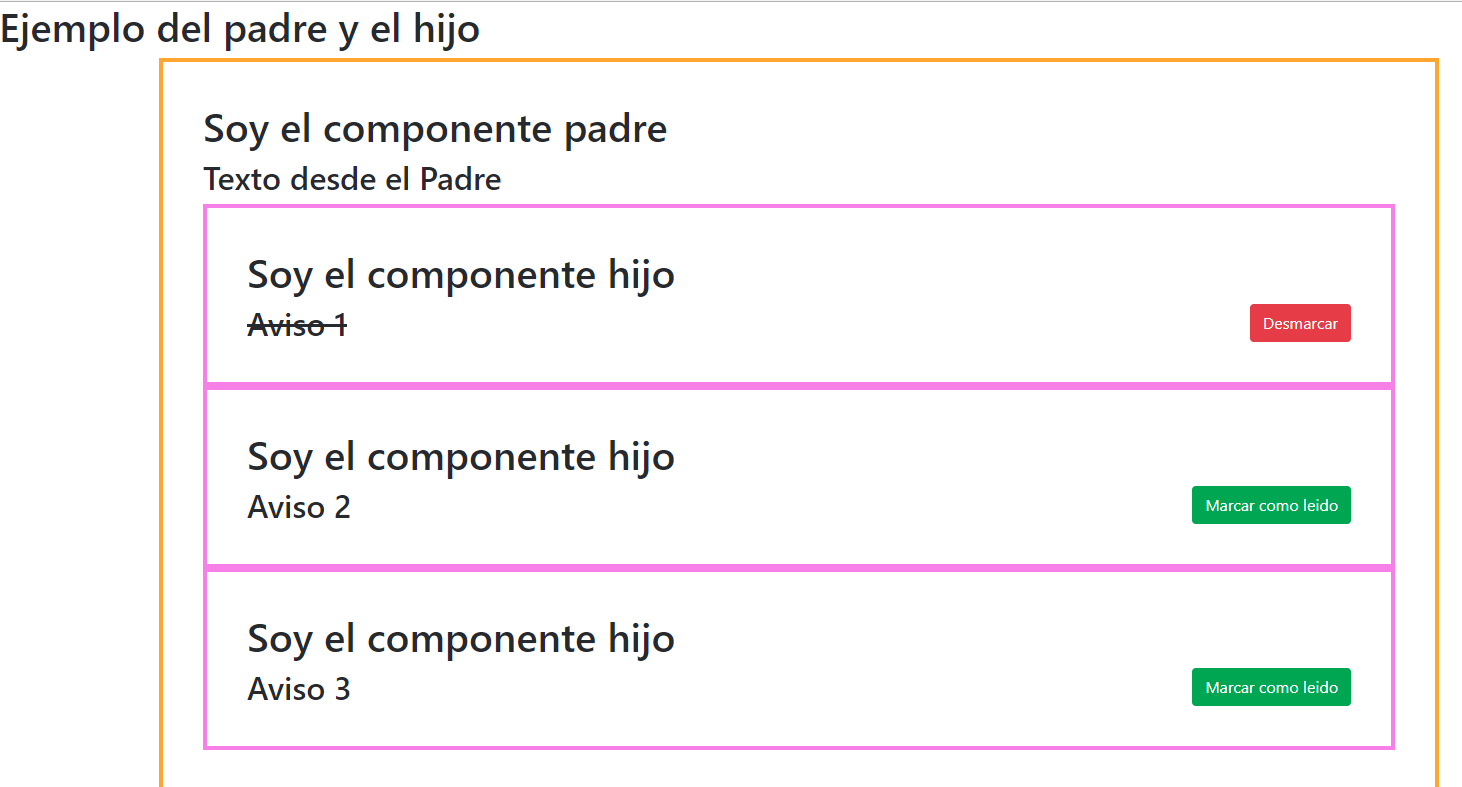
**Hijo.component.css**



Luego creamos el botón que activará la clase tachado y los dos tipos de botones a mostrar según esté marcado o desmarcado (switch boolean)



Este es el resultado:



Ahora vamos a exportar desde el hijo un valor hasta el padre. Se hace con output.

Para ello debemos incluir las funciones que vamos a usar en la importación.

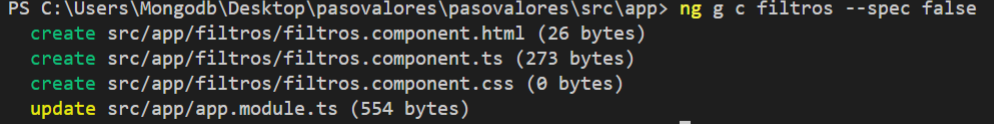
**Hijo.component.ts**



Copiar el resto luego.

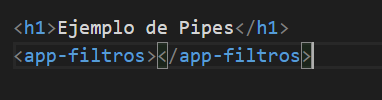
**Pipes**

Instalamos este componente

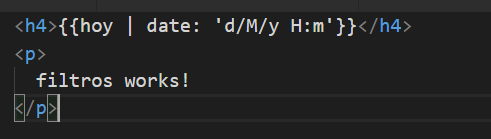


Luego, lo declaramos para que salga en la raíz comentando lo anterior y sustituyéndolo por:

**App.component.html**



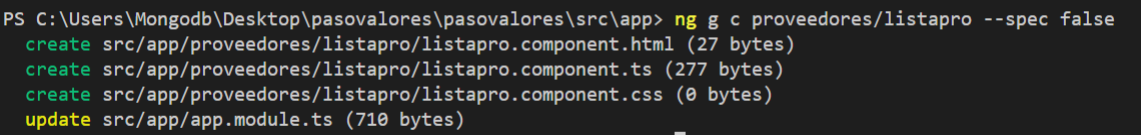
**Filtros.component.html**



1. **Servicios**

**Implementando un servicio:**

Creamos un componente dentro de proveedores

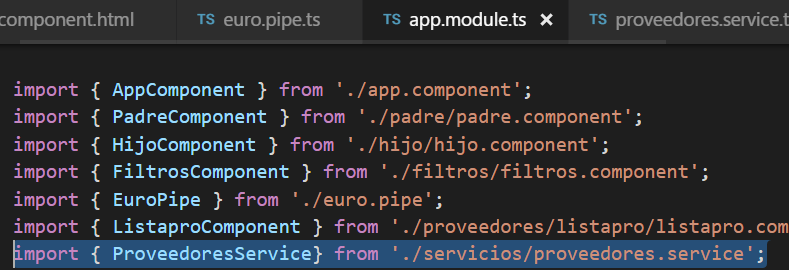


Creamos un servicio llamado proveedotres dentro de la carpeta servicios

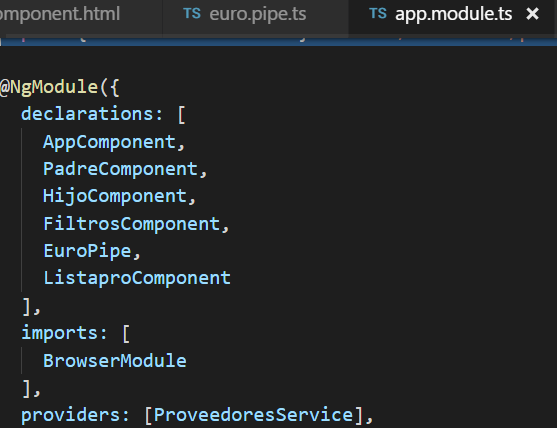


Tenemos que importar nuestro servicio recién creado

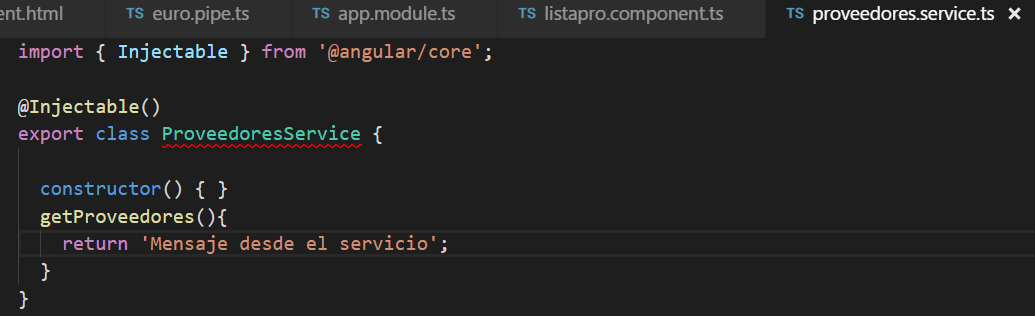
**app.module.ts**



También tenemos que añadirlo a este array

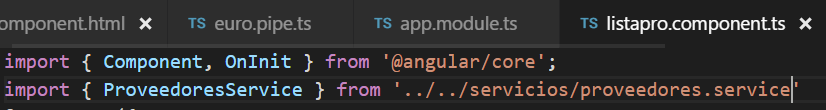


Dentro del servicio, creamos una función para establecer un mensaje



Luego, hay que importar el servicio en la lista de proveedores

**Listaprocomponent.ts**

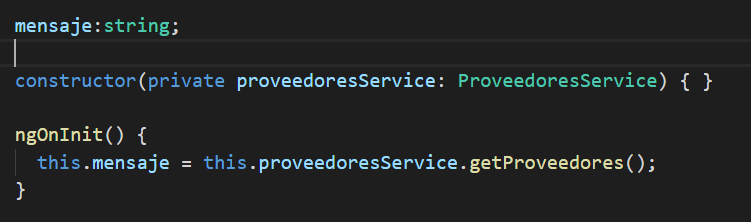


Para enlazar el servicio al componente se hace lo siguiente (Es necesario que la primera sea mayúscula porque es una propiedad)

Listapro.component.ts

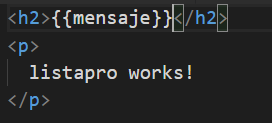


Esta últiman función iguala el mensaje al mensaje del componente. Para ello es necesario tipar la variable mensaje e igualarla a la función para mostrar el mensaje del servicio.

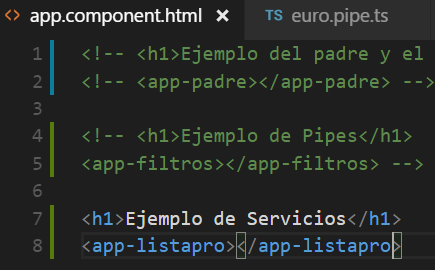


Ahora ponemos en el html el contenido del mensaje para verlo

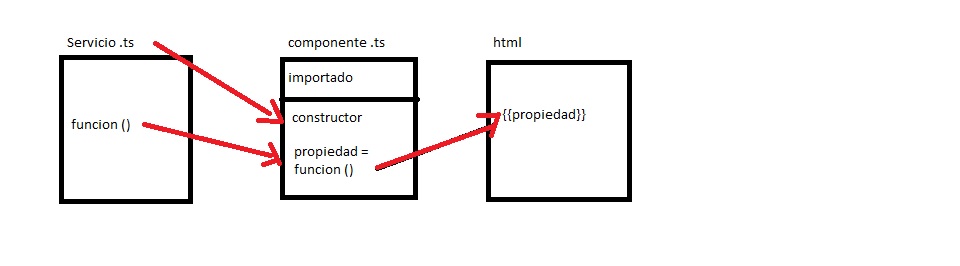
**Listaprocomponent.html**



Añadimos el servicio a **app.component.html**

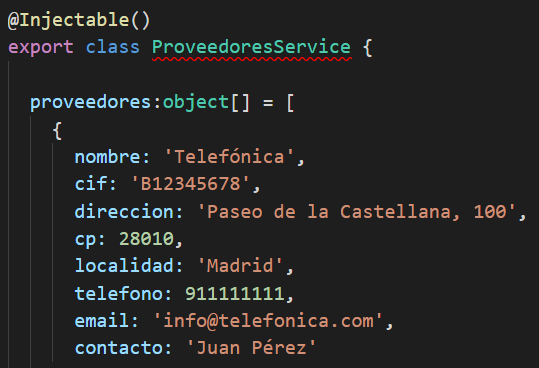


Esquema de funcionamiento:

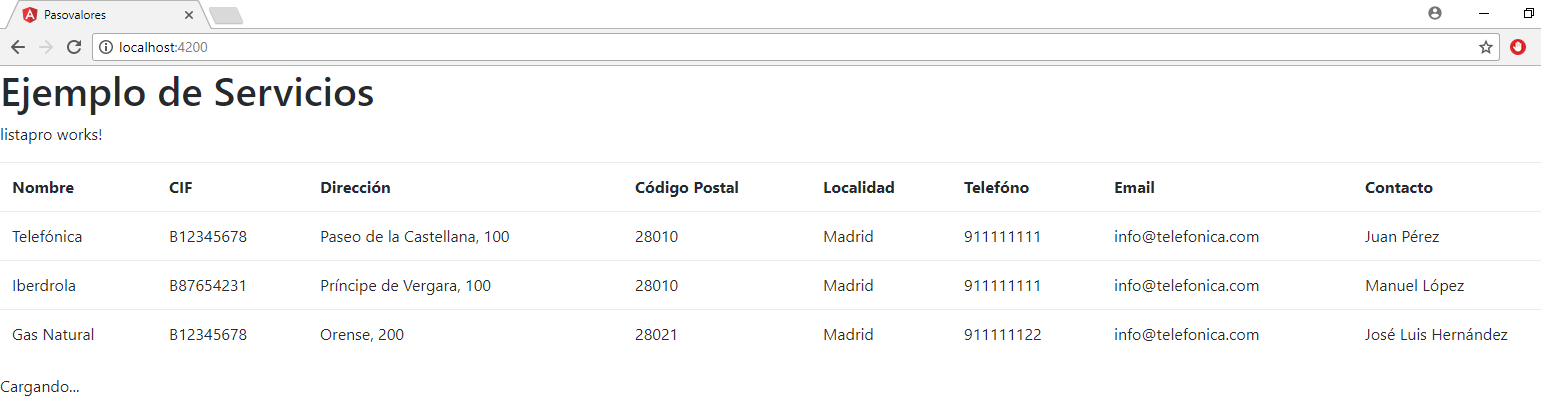


Vamos a crear un array en el servicio

**Proveedores.service.ts**

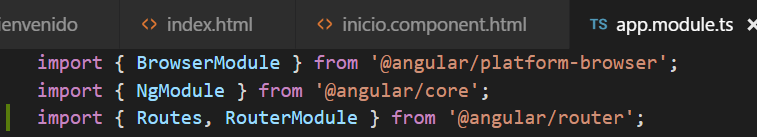


Resultado:

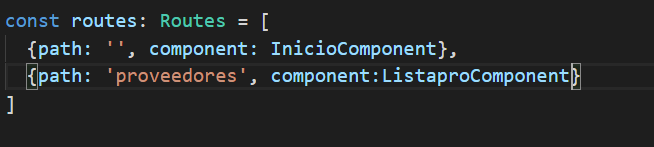


Routing

Importamos los modulos en app.modules, el modulo para la ruta

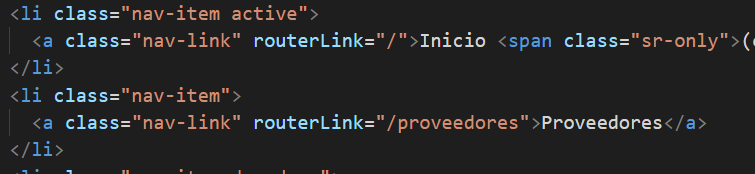


Luego declaramos la constante routas.



Esto implica que / es el componente inicio y /proveedores es el componente proveedores. Las rutas están establecidas de manera que salta el componente que se indique en la url, pero debemos redirigirlas creando una barra de navegación.

Copiamos un menú de la página de bootstrap y dentro del enlace usamos esta propiedad para dirigir cada enlace a cada componente, en este caso inicio y proveedores.



Poniendo esto nos lleva al inicio si el usuario pone un directorio que no exista en la url.

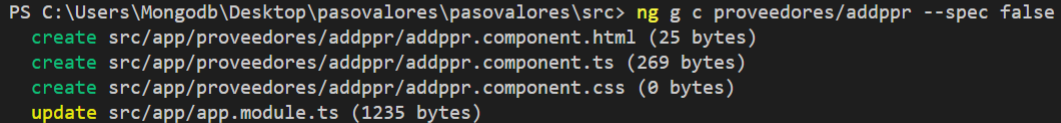
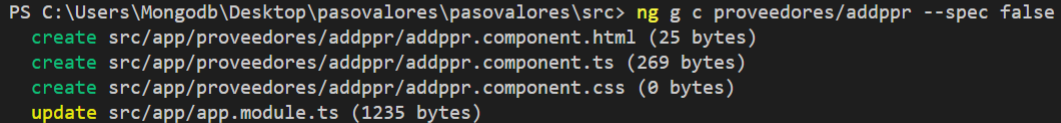


Respecto a los li, podemos el primer atributo para marcarlos como activados, y la segunda para que el li Inicio, que siempre sale marcado por defecto por ser el primero, se marque como activo solo cuando de verdad esté activo.



Formularios

Creamos un componente para crear proveedores

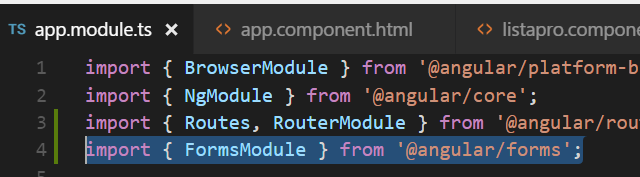


Mas pasos de por medio

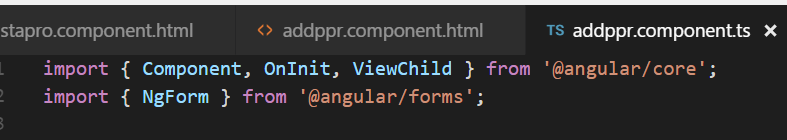
Creamos un formulario que recogerá los datos del nuevo proveedor



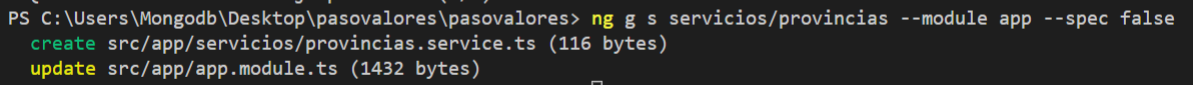
Importamos el siguiente módulo:



Importamos los modulos ViewChild y NgForm. Este último es el que envía los datos a través del formulario.

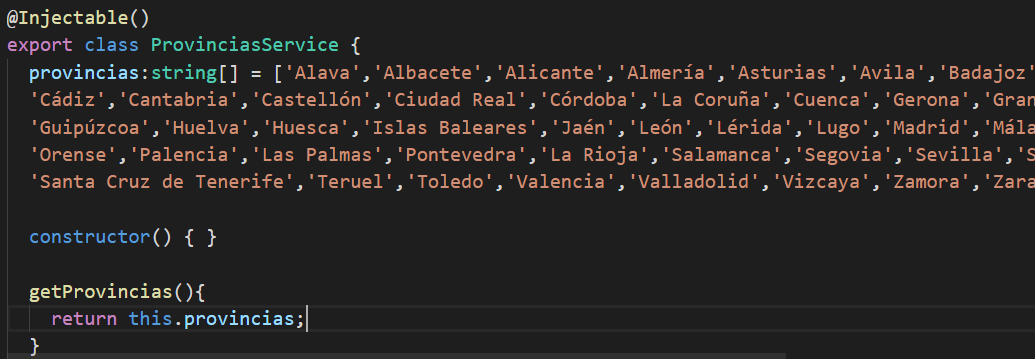


Creamos un componente importado dentro de servicios. –module es para importarlo dentro de servicios



Creamos el array provincias y la función para obtener las provincias

**provincias.service.ts**



Hay que tipar el array en

**Addppr.component.ts**

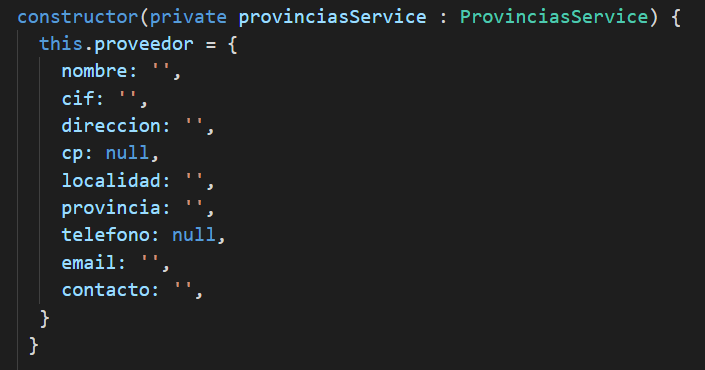


E importar la provincia de dentro de la carpeta de los servicios

**Addppr.component.ts**

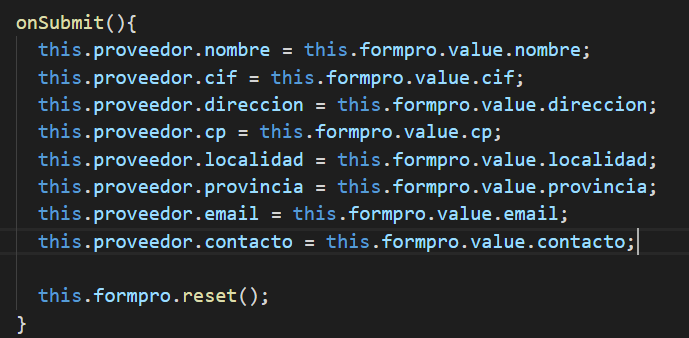


También debemos completar el constructor importando el servicio provincia y rellenando el campo solicitado:



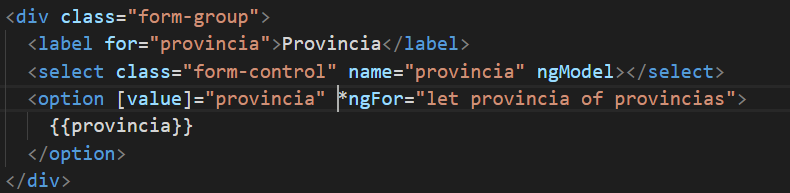
En el onSubmit, que se encarga de recoger los valores del formulario, metemos también la linea que recogerá provincia del campo del formulario.

**addppr.component.ts**

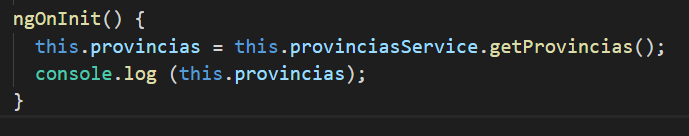


En el html ponemos el campo provincia, como el valor que debe tomar desde el array.

**addppr.component.html**



Para coger el valor del array hace falta rellenar la función que tomará dichos valores.

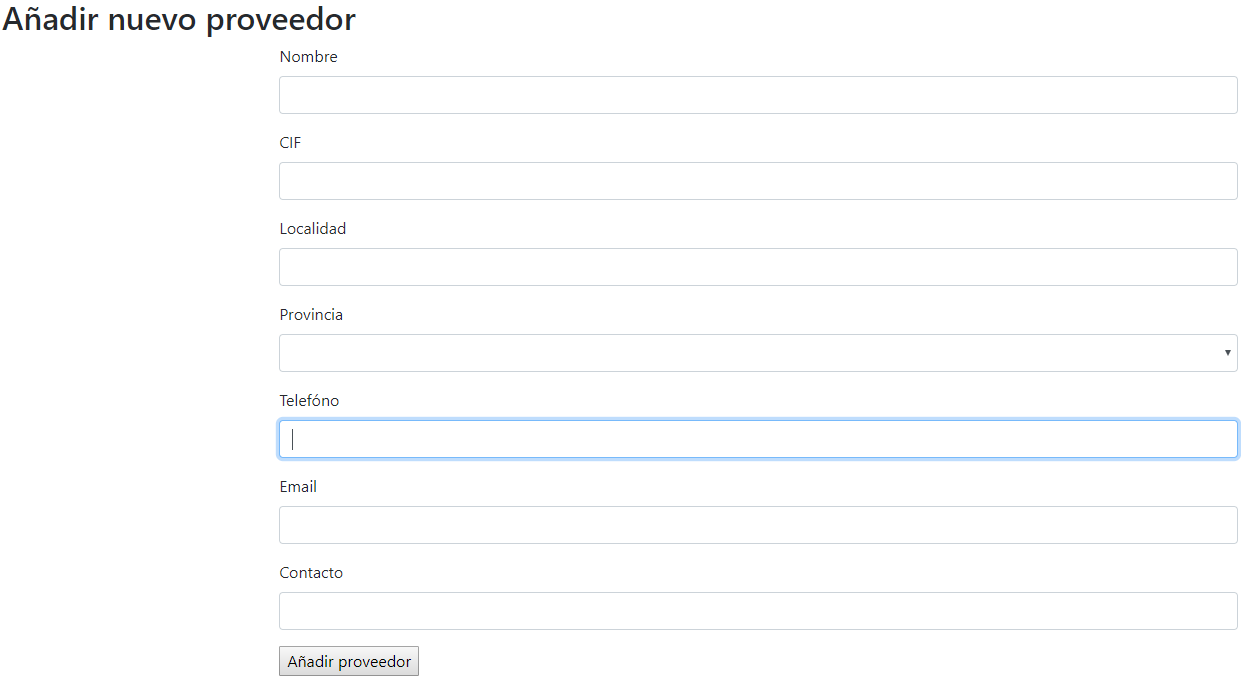


Incluimos el campo requerido y el patrón para evaluar el email.

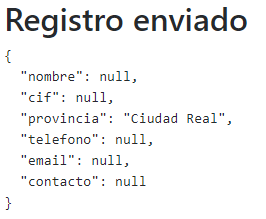
**Addppr.component.html**



El resultado de todo ello es el siguiente:

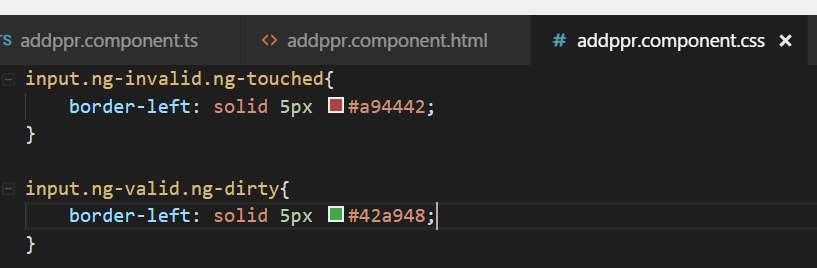


Queda así cuando el usuario ha registrado los campos, ya que hemos puesto que nos lo muestre.



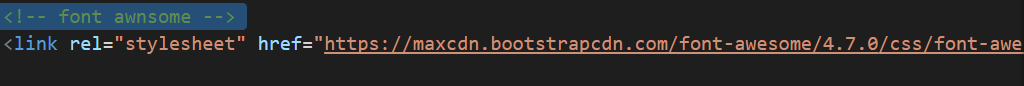
**addpr.component.css**

Aquí especificamos unas clases que se apliquen únicamente cuando se cumplan las siguientes condiciones.

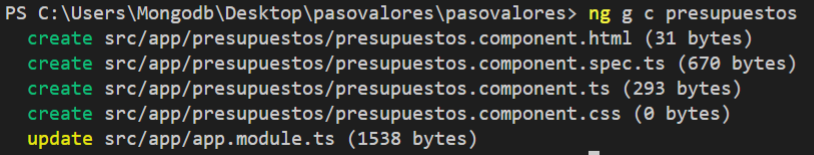


Ponemos el link a Font-awnsome en el

**Index.html**



Creamos el componente presupuestos



Añadimos el siguiente paquete en **app.module.ts**

