



FORMACIÓN EN NUEVAS TECNOLOGÍAS

Angular – TypeScript Avanzado

TypeScript

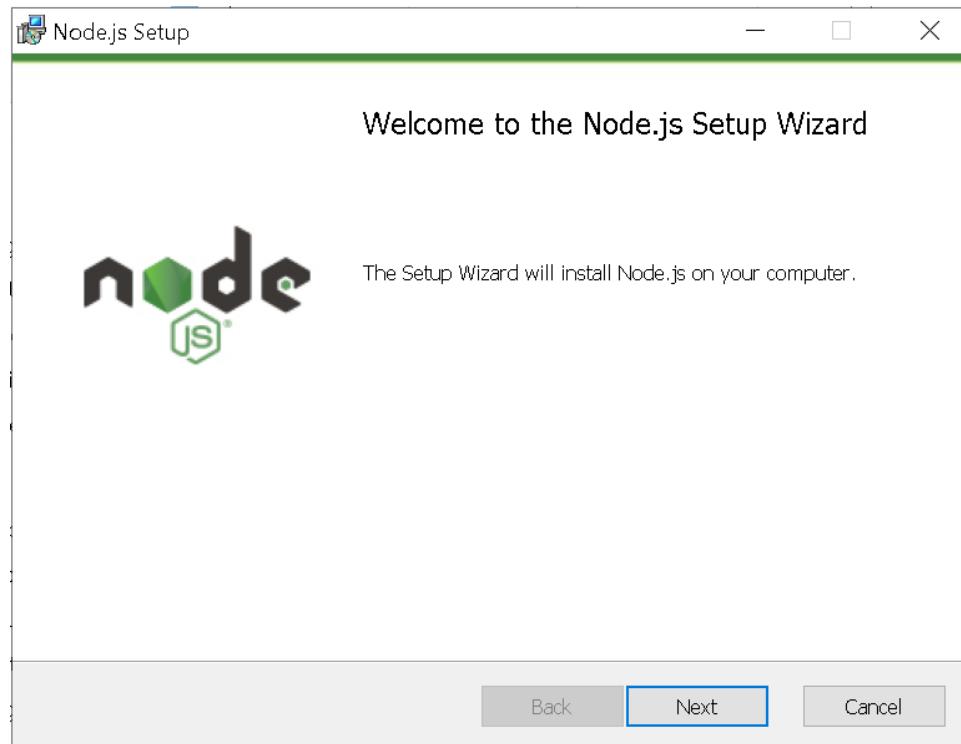
- ▶ TypeScript es un lenguaje de programación fuertemente tipado que se basa en JavaScript de tipado estático y basado en clases, lo que le brinda mejores herramientas a cualquier escala. Lanzado en el 2012 por Microsoft.
- ▶ <https://www.typescriptlang.org/>

- ▶ **TypeScript (TS) es un lenguaje de programación construido a un nivel superior de JavaScript (JS).** Esto quiere decir que TypeScript dota al lenguaje de varias características adicionales que hacen que podamos escribir código con menos errores, más sencillo, coherente y fácil de probar, en definitiva, más limpio y sólido.
- ▶ Fue creado por **Microsoft** en 2012 y, desde entonces, su adopción no ha hecho más que crecer. Especialmente, desde que **Google** decidió adoptarlo como lenguaje por defecto para desarrollar con **Angular**. Aunque, hoy en día, podemos desarrollar con TypeScript en cualquiera de los **frameworks o librerías** más punteras, como son **React** para el frontend o **Node** para el backend.

Como lo obtenemos?

- ▶ Instalamos el nodeJS para obtener el gestor de paquetes de Javascript
 - ▶ Y luego instalamos typescript con npm





Comprobamos por consola que tenemos instalado el nodeJS correctamente

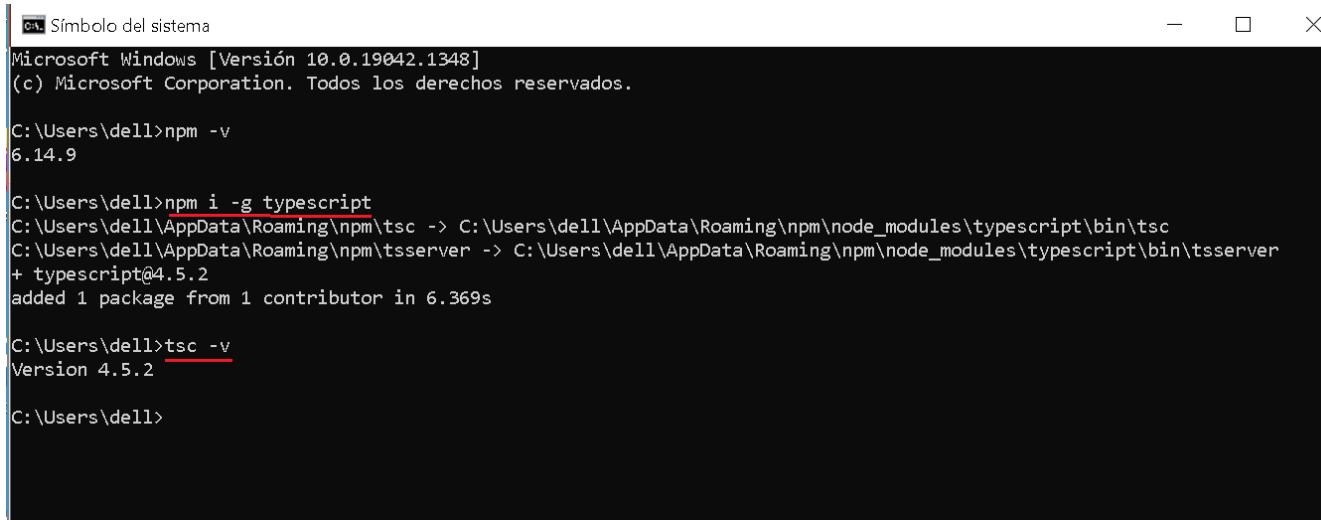
 Símbolo del sistema

```
Microsoft Windows [Versión 10.0.19042.1348]
(c) Microsoft Corporation. Todos los derechos reservados.
```

```
C:\Users\dell>npm -v
6.14.9
```

```
C:\Users\dell>
```

Ahora instalamos typeScript a nivel global



```
C:\ Símbolo del sistema
Microsoft Windows [Versión 10.0.19042.1348]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\dell>npm -v
6.14.9

C:\Users\dell>npm i -g typescript
C:\Users\dell\AppData\Roaming\npm\tsc -> C:\Users\dell\AppData\Roaming\npm\node_modules\typescript\bin\tsc
C:\Users\dell\AppData\Roaming\npm\tsserver -> C:\Users\dell\AppData\Roaming\npm\node_modules\typescript\bin\tsserver
+ typescript@4.5.2
added 1 package from 1 contributor in 6.369s

C:\Users\dell>tsc -v
Version 4.5.2

C:\Users\dell>
```

Si quisiéramos compilarlos nos generaría un archivo js

The screenshot shows a Visual Studio Code interface. On the left is the Explorer sidebar with 'EXPLORADOR' expanded, showing 'EDITORES ABIERTOS' containing 'index.ts' (marked with a TS icon) and 'TYPESCRIPT' containing 'index.js' (marked with a JS icon) and 'index.ts'. The main editor area has a tab for 'index.ts - typescript - Visual Studio Code [Administrador]'. The code in 'index.ts' is:

```
1  console.log("hola como vamos");
```

To the right of the editor is a terminal window titled 'Símbolo del sistema' (Symbol of the system). It shows the command line history:

```
C:\Users\dell>cd Desktop
C:\Users\dell\Desktop>cd typescript
C:\Users\dell\Desktop\typescript>tsc index.ts
C:\Users\dell\Desktop\typescript>
```

Finalmente compilamos el js para obtener el resultado

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 "index.ts - typescript - Visual Studio Code [Administrador]". The left sidebar has icons for Explorador, Editores Abiertos, and TypeScript. The "EDITORES ABIERTOS" section shows "index.ts" (TS icon) with the code: "1 console.log("hola como vamos");". The "TYPESCRIPT" section shows "index.js" (JS icon) and "index.ts" (TS icon). The main editor area shows the same code. To the right is a terminal window titled "Símbolo del sistema" with the command "C:\Users\dell\Desktop\typescript>node index" and the output "hola como vamos".

Instalamos este componente para poder ejecutar los ts con un solo comando

```
C:\ Símbolo del sistema
Microsoft Windows [Versión 10.0.19042.1348]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\dell>npm install -g ts-node
C:\Users\dell\AppData\Roaming\npm\ts-node -> C:\Users\dell\AppData\Roaming\npm\node_modules\ts-node\dist\bin.js
C:\Users\dell\AppData\Roaming\npm\ts-script -> C:\Users\dell\AppData\Roaming\npm\node_modules\ts-node\dist\bin-script-deprecated.js
C:\Users\dell\AppData\Roaming\npm\ts-node-script -> C:\Users\dell\AppData\Roaming\npm\node_modules\ts-node\dist\bin-script.js
C:\Users\dell\AppData\Roaming\npm\ts-node-transpile-only -> C:\Users\dell\AppData\Roaming\npm\node_modules\ts-node\dist\bin-transpile.js
C:\Users\dell\AppData\Roaming\npm\ts-node-cwd -> C:\Users\dell\AppData\Roaming\npm\node_modules\ts-node\dist\bin-cwd.js
npm WARN ts-node@10.4.0 requires a peer of @types/node@* but none is installed. You must install peer dependencies yourself.
npm WARN ts-node@10.4.0 requires a peer of typescript@>=2.7 but none is installed. You must install peer dependencies yourself.

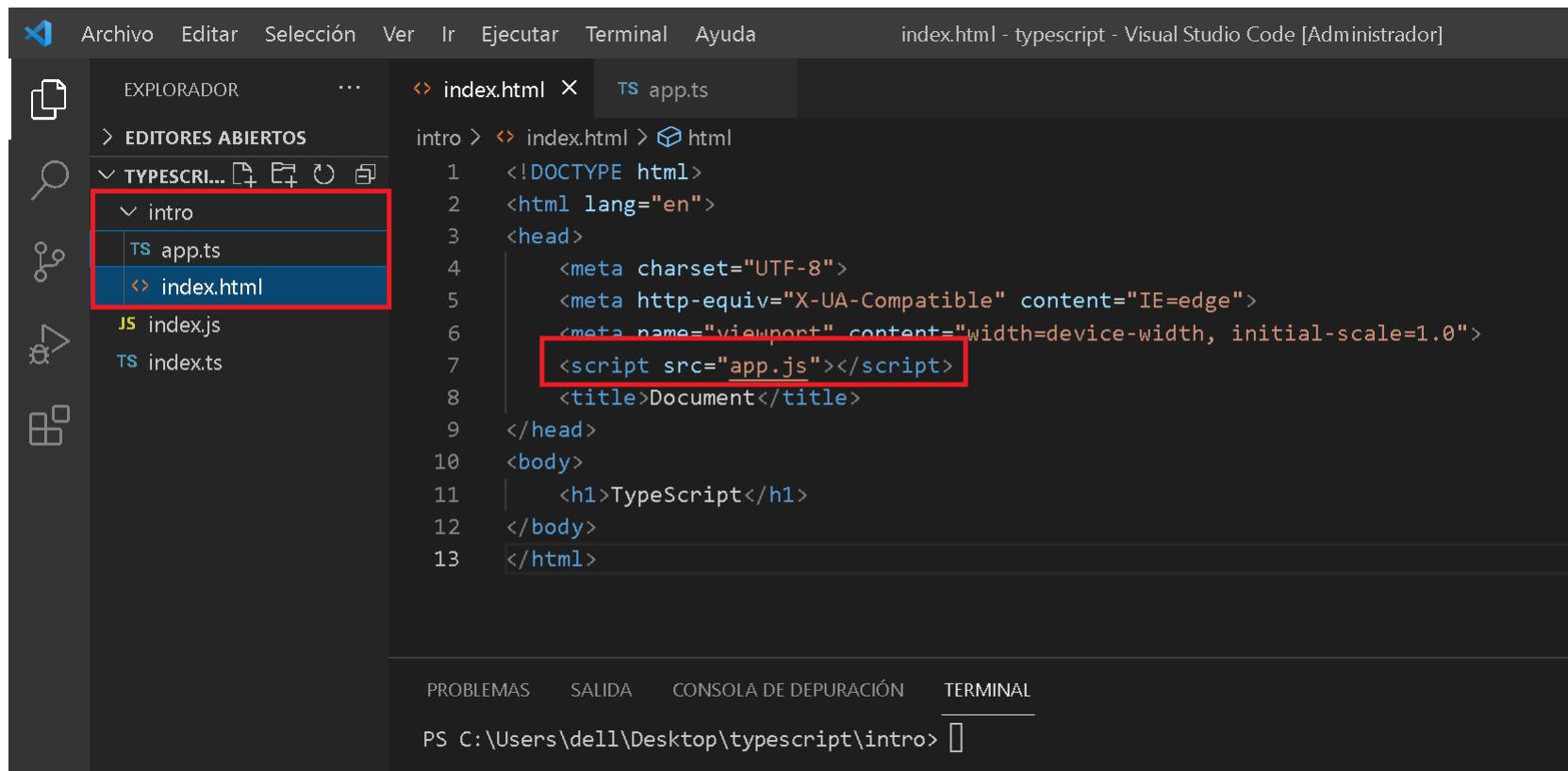
+ ts-node@10.4.0
added 14 packages from 45 contributors in 6s

C:\Users\dell>
```

A tener en cuenta

- ▶ Dentro de visual studio utilizar npx en frente de los comandos

Preparamos nuestro entorno de prueba



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

- File Explorer (Left):** Shows a tree view of files. A red box highlights the "intro" folder, which contains "app.ts" and "index.html".
- Code Editor (Center):** Displays the content of "index.html". The "app.ts" file is currently open in the editor.
- Content of index.html:**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="app.js"></script>
    <title>Document</title>
</head>
<body>
    <h1>TypeScript</h1>
</body>
</html>
```
- Terminal (Bottom):** Shows the command prompt output: "PS C:\Users\dell\Desktop\typescript\intro>"

Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda app.ts - typescript - Visu

EXPLORADOR ...

> EDITORES ABIERTOS

TYPESCRIPT

intro > TS app.ts

1 console.log("Hola iniciamos con esto")

TS app.ts

index.html

JS index.js

TS index.ts

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

PS C:\Users\dell\Desktop\typescript\intro>

Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda app.ts - typescript - Visual Studio Code [Ad]

EXPLORADOR ...

> EDITORES ABIERTOS

▼ TYPESCRIPT

 ▼ intro

 JS app.js

 TS app.ts

 ↳ index.html

 JS index.js

 TS index.ts

introduction > TS app.ts

```
1  console.log("Hola iniciamos con esto")
```

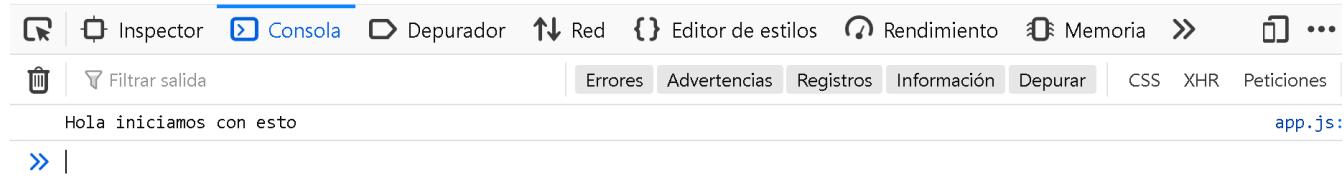
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

```
PS C:\Users\dell\Desktop\typescript\introduction> npx tsc app.ts
PS C:\Users\dell\Desktop\typescript\introduction> node app.js
Hola iniciamos con esto
PS C:\Users\dell\Desktop\typescript\introduction>
```



← → ⌛ ⌄ file:///C:/Users/dell/Desktop/typescript/intro/index.html ☆ ↻ ⌂

TypeScript



The screenshot shows the Chrome DevTools interface with the "Console" tab selected. The console output area displays the message "Hola iniciamos con esto" and the file name "app.js:". The toolbar above the console includes icons for Inspector, Consola (selected), Depurador, Red, Editor de estilos, Rendimiento, Memoria, and more.

Consola

Hola iniciamos con esto

app.js:

Para no escribir doble comando al compilar con el watch actualizamos al instante los archivos ts y js

Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda app.ts - typescript - Visual Studio Code [Administrador]

EXPLORADOR ...

> EDITORES ABIERTOS

✓ TYPESCRIPT

intro

JS app.js

TS app.ts

index.html

JS index.js

TS index.ts

TS app.ts X JS app.js

intro > TS app.ts

1 console.log("Vamos a ello")

JS app.js X

intro > JS app.js

1 console.log("Vamos a ello");

2

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

PS C:\Users\dell\Desktop\typescript\intro> npx tsc app -w

+ ▾ ^

[>] powershell

[>] powershell

[>] powershell

The screenshot shows a Visual Studio Code interface with two terminal panes open. The left terminal is for TypeScript (app.ts) and the right is for JavaScript (app.js). Both terminals contain the same code:

```
1 console.log("TypeScript Mola bastante");
```

The status bar at the bottom indicates:

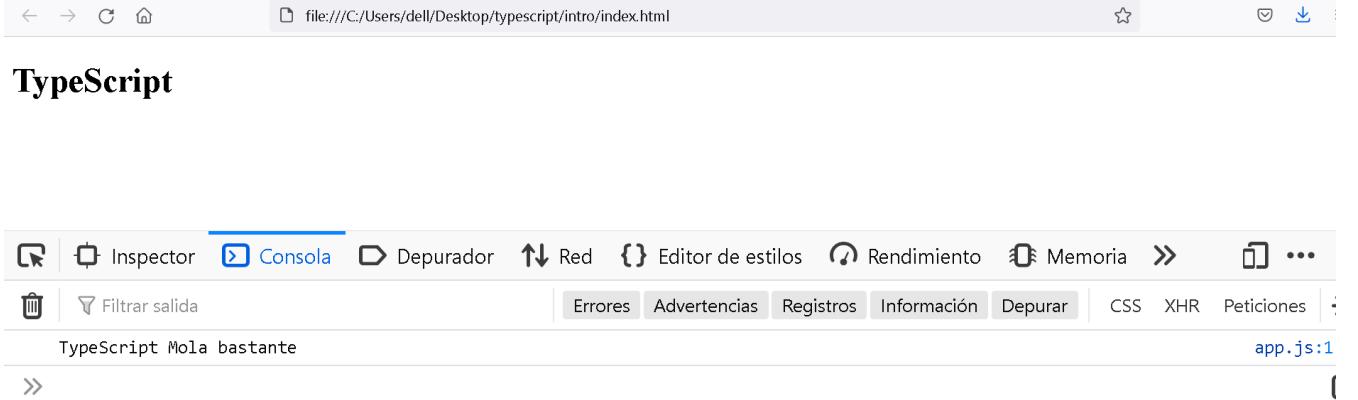
- PROBLEMAS
- SALIDA
- CONSOLA DE DEPURACIÓN
- TERMINAL

The terminal output shows:

```
[5:51:17] File change detected. Starting incremental compilation...
[5:51:17] Found 0 errors. Watching for file changes.
```

On the far right, there is a sidebar with three entries:

- powershell
- powershell
- node



TypeScript

Filtrar salida

Erros Advertencias Registros Información Depurar CSS XHR Peticiones

TypeScript Mola bastante app.js:1

Tipos de valores - Boolean

The screenshot shows the Visual Studio Code interface with a TypeScript file (app.ts) open. The code defines a boolean variable 'estadoCivil' and logs 'casado' to the console if it's true, or 'Soltero' if it's false.

```
TS app.ts 1 X JS app.js
intro > TS app.ts > ...
4  let estadoCivil:boolean = false;
5  estadoCivil = true;
6
7  if(estadoCivil){
8      console.log("casado");
9  }else {
10      console.log("Soltero");
11 }
12
13
```

The browser window shows the output of the code execution:

TypeScript

Console tab (selected):

- Inspector
- Consola (highlighted)
- Depurador

Filtrar salida: casado

Errors, Warnings, Logs, Information, Breakpoints, CSS, XHR, Requests

casado app.js:6:1

PROBLEMAS 1 SALIDA CONSOLA DE DEPURACIÓN TERMINAL

[6:03:28] Found 0 errors. Watching for file changes.

node

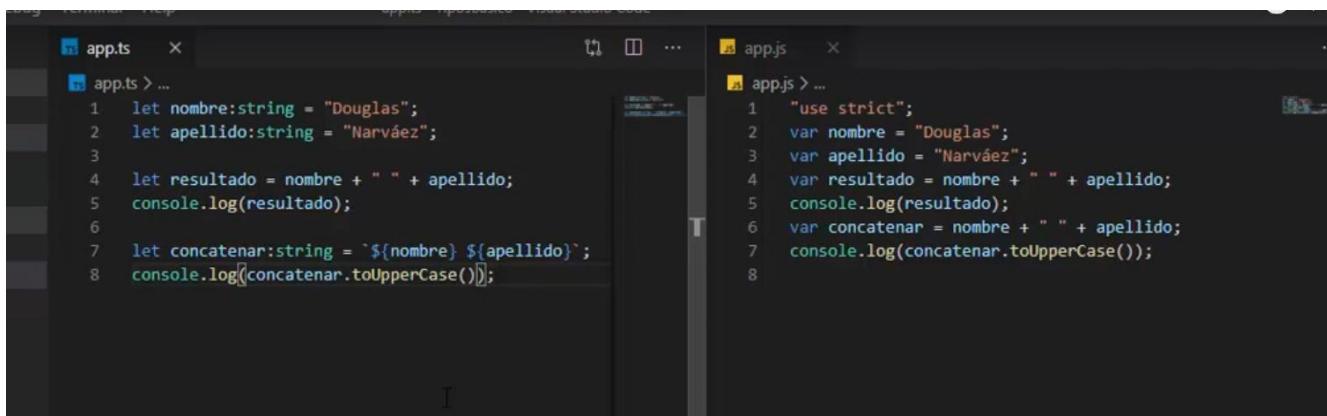
Tipos de valores - number

```
app.ts
1 let numero:number = 12.9;
2
3 let edad:number = 18;
4 if (edad >= 18) {
5     console.log("Es mayor de edad");
6 } else {
7     console.log("Es menor de edad");
8 }
9
10 edad = ObtenerEdad();
11 function ObtenerEdad(){
12     return 30;
13 }
```

```
app.js
1 "use strict";
2 var numero = 12.9;
3 var edad = 18;
4 if (edad >= 18) {
5     console.log("Es mayor de edad");
6 }
7 else {
8     console.log("Es menor de edad");
9 }
10 function ObtenerEdad() {
11     return 30;
12 }
```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS 1: node

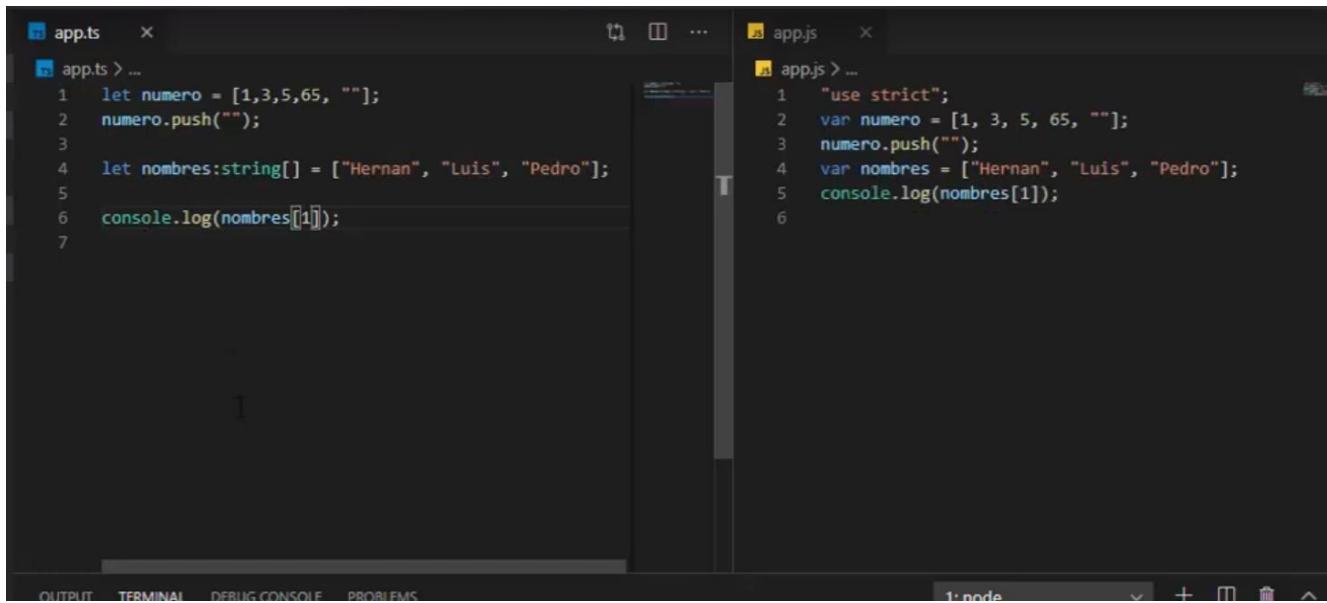
Tipos de valores - String



```
app.ts > ...
1 let nombre:string = "Douglas";
2 let apellido:string = "Narváez";
3
4 let resultado = nombre + " " + apellido;
5 console.log(resultado);
6
7 let concatenar:string = `${nombre} ${apellido}`;
8 console.log(concatenar.toUpperCase());
```

```
app.js > ...
1 "use strict";
2 var nombre = "Douglas";
3 var apellido = "Narváez";
4 var resultado = nombre + " " + apellido;
5 console.log(resultado);
6 var concatenar = nombre + " " + apellido;
7 console.log(concatenar.toUpperCase());
8
```

Tipos de valores - Array



The image shows a screenshot of the Visual Studio Code (VS Code) interface. It features two side-by-side code editors. The left editor is titled "app.ts" and contains the following TypeScript code:

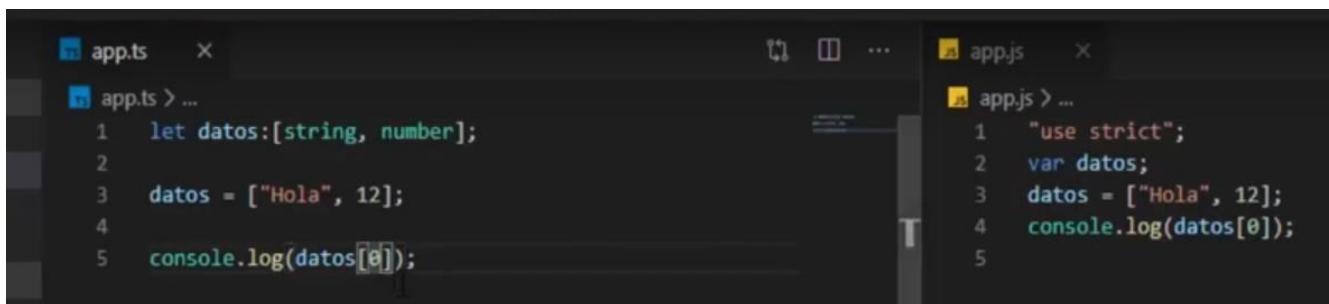
```
app.ts
1 let numero = [1,3,5,65, ""];
2 numero.push("");
3
4 let nombres:string[] = ["Hernan", "Luis", "Pedro"];
5
6 console.log(nombres[1]);
7
```

The right editor is titled "app.js" and contains the following JavaScript code:

```
app.js
1 "use strict";
2 var numero = [1, 3, 5, 65, ""];
3 numero.push("");
4 var nombres = ["Hernan", "Luis", "Pedro"];
5 console.log(nombres[1]);
6
```

At the bottom of the interface, there is a navigation bar with tabs for "OUTPUT", "TERMINAL", "DEBUG CONSOLE", and "PROBLEMS". To the right of the tabs, there is a terminal window labeled "1: node" with some additional icons.

Tipos de valores - Array



```
app.ts
1 let datos:[string, number];
2
3 datos = ["Hola", 12];
4
5 console.log(datos[0]);
```

```
app.js
1 "use strict";
2 var datos;
3 datos = ["Hola", 12];
4 console.log(datos[0]);
5
```

Enums

```
app.ts
enum Estado {
    activo = 2,
    inactivo = 5
}
console.log(Estado.activo);

app.js
use strict";
var Estado;
(function (Estado) {
    Estado[Estado["activo"] = 2] = "activo";
    Estado[Estado["inactivo"] = 5] = "inactivo";
})(Estado || (Estado = {}));
console.log(Estado.activo);
```

Any

The image shows a code editor interface with two files side-by-side:

- app.ts**:
A TypeScript file containing the following code:

```
1 let cualquierValor;
2
3 cualquierValor = "Hola";
4 console.log(cualquierValor.length);
5
6 cualquierValor = false;
7 console.log(cualquierValor);
8
9 cualquierValor = 12;
```
- app.js**:
A JavaScript file containing the following code:

```
1 "use strict";
2 var cualquierValor;
3 cualquierValor = "Hola";
4 console.log(cualquierValor.length);
5 cualquierValor = false;
6 console.log(cualquierValor);
7
```

The code demonstrates how the `Any` type is used in both languages, allowing assignment of different types to the same variable.

Void

```
Terminal Help app.ts - TiposBásico - Visual Studio Code
app.ts > ...
1  function Saludar():void{
2  |  console.log("Hola");
3  }
4
5  let valor = Saludar();
6  console.log(valor);

app.js > ...
1  "use strict";
2  function Saludar() {
3  |  console.log("Hola");
4  }
5
```

Null y Undefined

The screenshot shows a Visual Studio Code interface with two open files: `app.ts` and `app.js`. Both files contain the following code:

```
1 let monto:number = null;
2 let miVariable;
3 console.log(monto);
4 console.log(miVariable);
5
6 console.log(typeof monto);
7 console.log(typeof miVariable);
```

The `app.ts` file is associated with a `tsconfig.json` file, as indicated by the icon in the title bar. The `app.js` file is also shown in the title bar.

Type assertions

The image shows a code editor interface with two tabs: `app.ts` and `app.js`.

`app.ts` content:

```
1 let valor:any = "Hola soy una cadena";
2
3 let resultado:number = (<string> valor).length;
4 let resultado2:number = (valor as string).length;
```

`app.js` content:

```
1 "use strict";
2 var valor = "Hola soy una cadena";
3 var resultado = valor.length;
4 var resultado2 = valor.length;
```

Const

```
app > app.ts > ...
1  const ESTADO:Boolean = false;
2
3  if (true) {
4    const ESTADO = true;
5  } else {
6
7  }
8
9  for (const iterator of [1,2,3,4,5,6,7]) {
10   console.log(iterator);
11 }
12
```

```
build > app.js > ...
1  "use strict";
2  const ESTADO = false;
3  if (true) {
4    const ESTADO = true;
5  } else {
6  }
7
8  for (const iterator of [1, 2, 3, 4, 5, 6, 7]) {
9    console.log(iterator);
10 }
11
```

Array destructuring



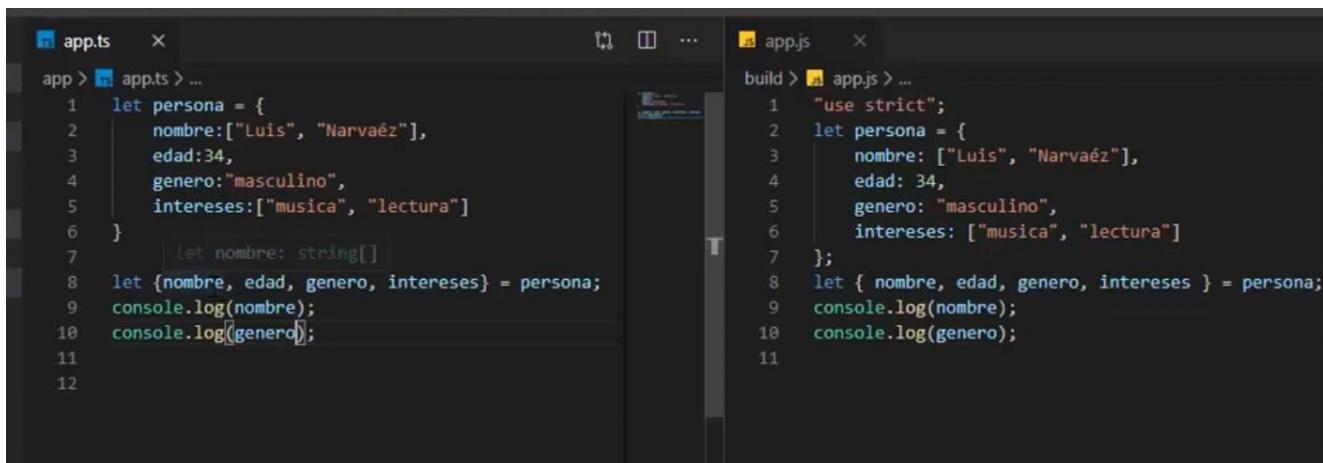
The image shows a screenshot of a code editor with a dark theme. The file is named 'app.ts'. The code demonstrates array destructuring:

```
app > app.ts > ...
1  let frutas:string[] = ["Manzana", "Uva", "Piña"];
2
3  let [item1, item2, item3] = frutas;
4  console.log(item1);
5  console.log(item2);
6  console.log(item3);
7
```

Tuple destructuring

```
app.ts  ✘  
app > app.ts > ...  
1  
2 let persona: [string[], number, string, string[]];  
3 persona = [["Luis", "Narvaéz"], 24, "masculino", ["musica", "lectura"]];  
4  
5 let [nombre, edad, genero, intereses] = persona;  
6 console.log(nombre);  
7 console.log(edad);  
8
```

Object destructuring



The image shows a code editor with two files side-by-side:

app.ts

```
1 let persona = {  
2     nombre:["Luis", "Narvaéz"],  
3     edad:34,  
4     genero:"masculino",  
5     intereses:["musica", "lectura"]  
6 }  
7     let nombre: string[]  
8 let {nombre, edad, genero, intereses} = persona;  
9 console.log(nombre);  
10 console.log(genero);  
11  
12
```

app.js

```
1 "use strict";  
2 let persona = {  
3     nombre: ["Luis", "Narvaéz"],  
4     edad: 34,  
5     genero: "masculino",  
6     intereses: ["musica", "lectura"]  
7 };  
8 let { nombre, edad, genero, intereses } = persona;  
9 console.log(nombre);  
10 console.log(genero);  
11
```

Spread Operador de propagación

```
app > app.ts > ...
1
2
3 let libro ={
4     autor:"Oscar",
5     titulo:"La perla perdida",
6     fecha: new Date(2020, 1, 4)
7 }
8
9 let vehiculo ={
10     color:"Negro",
11     puerta:"A3",
12     marca: "Toyota"
13 }
14
15 let agrupar ={prueba:"Hola", ...libro, ...vehiculo};
16 console.log(agrupar);
17

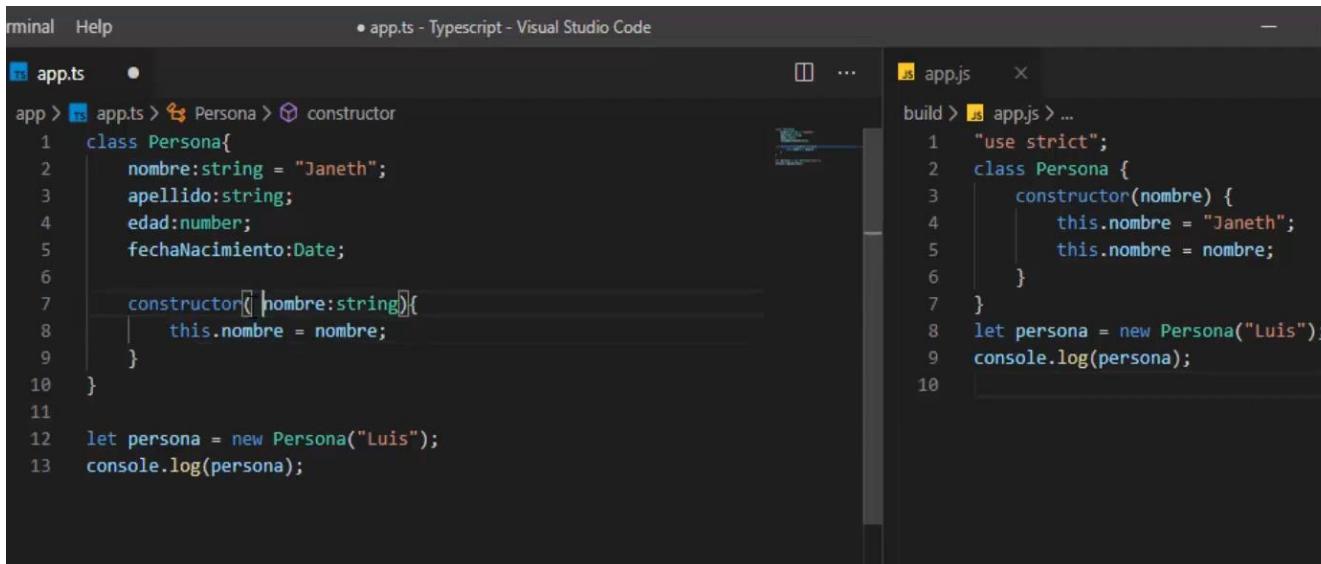
build > app.js > ...
1   "use strict";
2   let libro = {
3       autor: "Oscar",
4       titulo: "La perla perdida",
5       fecha: new Date(2020, 1, 4)
6   };
7   let vehiculo = {
8       color: "Negro",
9       puerta: "A3",
10      marca: "Toyota"
11  };
12  let agrupar = Object.assign(Object.assign({ pru
```

Default Values

```
app > app.ts > ...
1
2  function ObtenerValores(objeto: { valor1: string,
3  |   valor2?: number }) {
4  |   let { valor1, valor2 } = objeto;
5  |   console.log(valor1);
6  |   console.log(valor2);
7  }
8
9  ObtenerValores({valor1:"Camisa volcom"});
10
11 function Saludar(texto:string = "Hola2"){
12   console.log(texto);
13 }
14
15 Saludar("Hola");
16

build > app.js > ...
1  function ObtenerValores(objeto) {
2  |   let { valor1, valor2 } = objeto;
3  |   console.log(valor1);
4  |   console.log(valor2);
5  }
6  ObtenerValores({ valor1: "Camisa volcom" });
7  function Saludar(texto = "Hola2") {
8  |   console.log(texto);
9  }
10 Saludar("Hola");
```

Clases



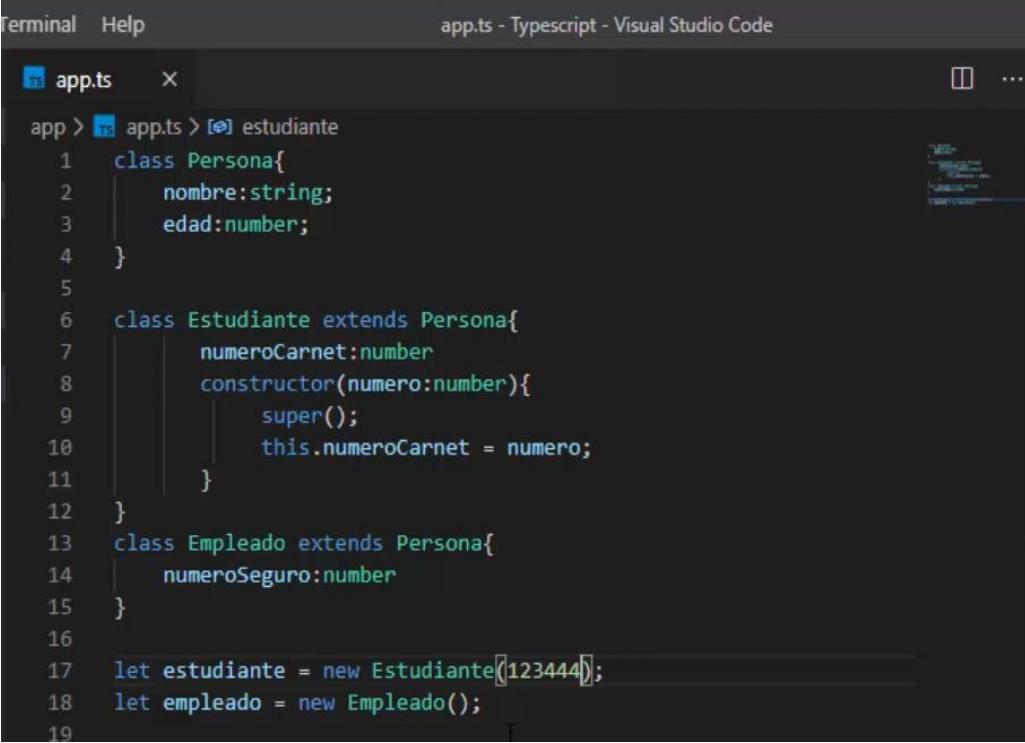
The screenshot shows a Visual Studio Code interface with two files open:

- app.ts**:
A TypeScript file containing a class definition for `Persona`. The class has properties `nombre`, `apellido`, `edad`, and `fechaNacimiento`. It also has a constructor that takes a parameter `nombre` and sets it on the instance.

```
1  class Persona{
2    nombre:string = "Janeth";
3    apellido:string;
4    edad:number;
5    fechaNacimiento:Date;
6
7    constructor(nombre:string){
8      this.nombre = nombre;
9    }
10}
11
12 let persona = new Persona("Luis");
13 console.log(persona);
```
- app.js**:
The corresponding JavaScript output generated by the TypeScript compiler. It includes a strict mode declaration, the class definition with its constructor, and an instantiation of the `Persona` class with the value "Luis".

```
1  "use strict";
2  class Persona {
3    constructor(nombre) {
4      this.nombre = "Janeth";
5      this.nombre = nombre;
6    }
7  }
8 let persona = new Persona("Luis");
9 console.log(persona);
10
```

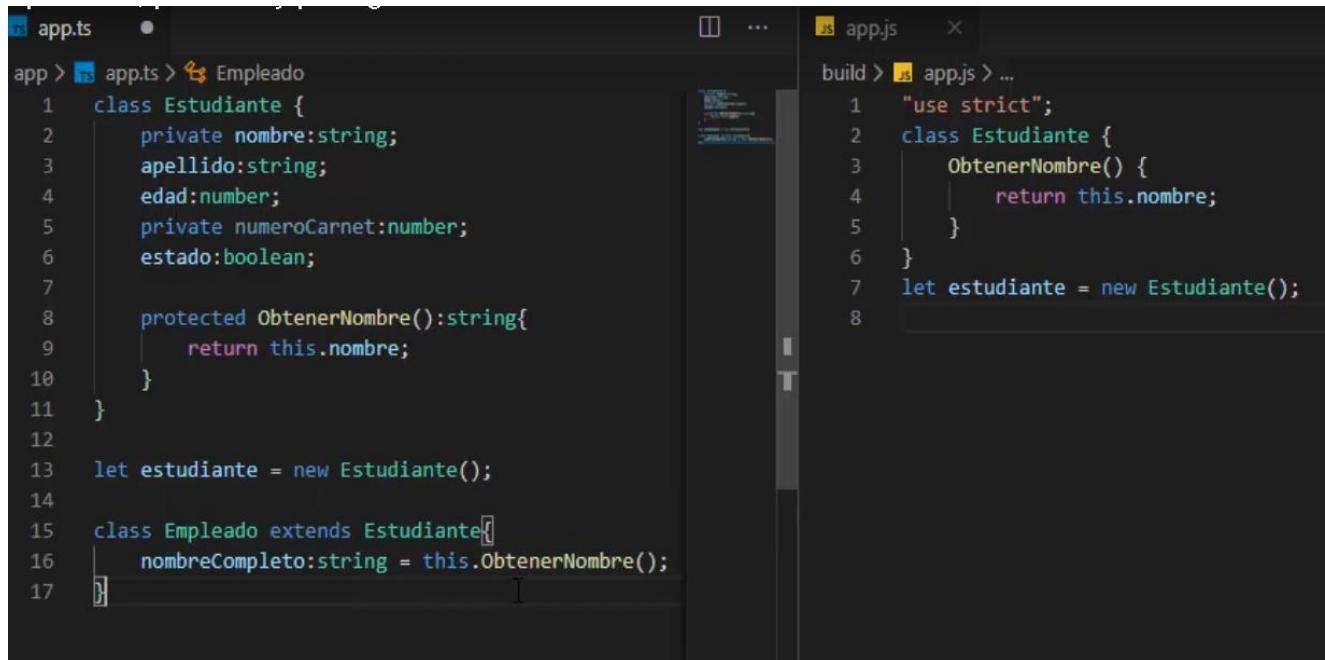
herencia



The screenshot shows a Visual Studio Code interface with a dark theme. The title bar reads "Terminal Help app.ts - Typescript - Visual Studio Code". The main area displays the following TypeScript code:

```
app > app.ts > estudiante
1  class Persona{
2    nombre:string;
3    edad:number;
4  }
5
6  class Estudiante extends Persona{
7    numeroCarnet:number
8    constructor(numero:number){
9      super();
10     this.numeroCarnet = numero;
11   }
12 }
13 class Empleado extends Persona{
14   numeroSeguro:number
15 }
16
17 let estudiante = new Estudiante(123444);
18 let empleado = new Empleado();
```

Modificadores de Acceso



The image shows a code editor interface with two files: `app.ts` and `app.js`.

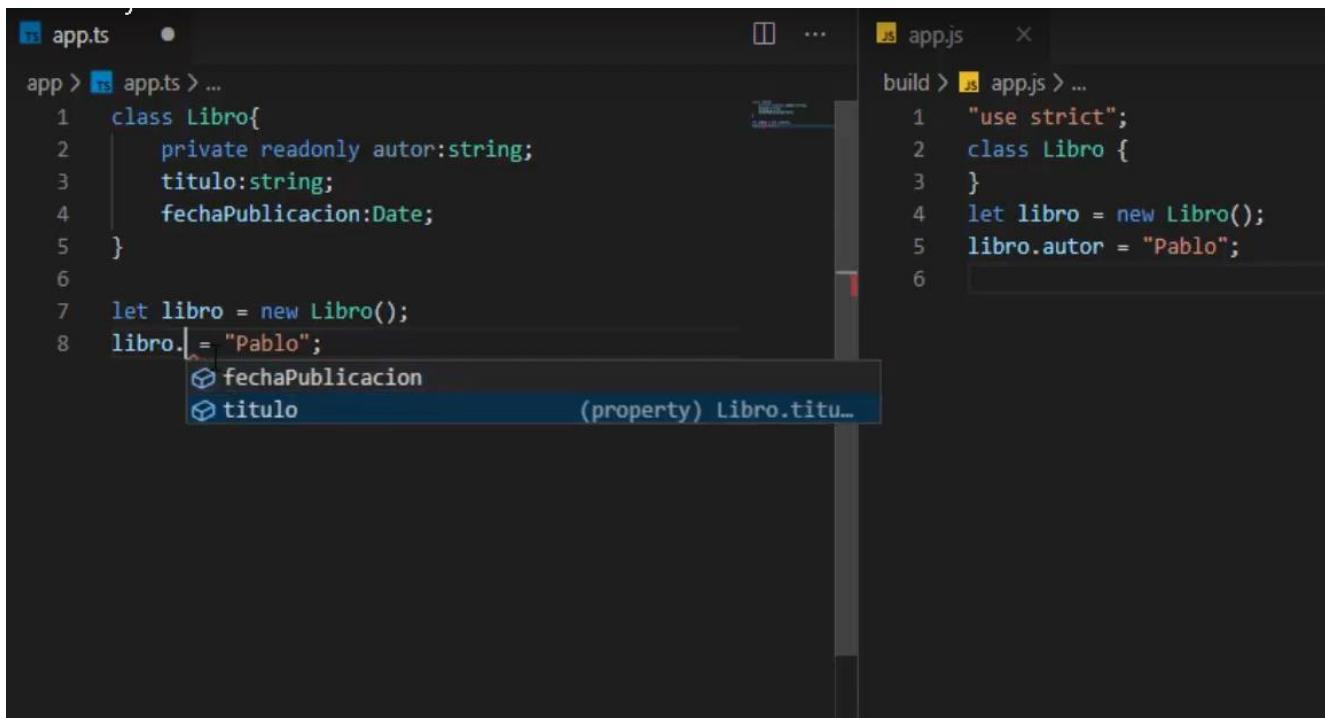
app.ts:

```
app > app.ts > Empleado
1 class Estudiante {
2     private nombre:string;
3     apellido:string;
4     edad:number;
5     private numeroCarnet:number;
6     estado:boolean;
7
8     protected ObtenerNombre():string{
9         return this.nombre;
10    }
11 }
12
13 let estudiante = new Estudiante();
14
15 class Empleado extends Estudiante{
16     nombreCompleto:string = this.ObtenerNombre();
17 }
```

app.js:

```
build > app.js > ...
1 "use strict";
2 class Estudiante {
3     ObtenerNombre() {
4         return this.nombre;
5     }
6 }
7 let estudiante = new Estudiante();
```

Modificador Readonly



The image shows a code editor interface with two files: `app.ts` and `app.js`.

app.ts:

```
app > ts app.ts > ...
1  class Libro{
2    private readonly autor:string;
3    titulo:string;
4    fechaPublicacion:Date;
5  }
6
7  let libro = new Libro();
8  libro.autor = "Pablo";
```

A tooltip is displayed over the line `libro.autor = "Pablo";`, showing the properties of the `Libro` object: `fechaPublicacion` and `titulo`. The `titulo` property is highlighted.

app.js:

```
build > js app.js > ...
1  "use strict";
2  class Libro {
3  }
4  let libro = new Libro();
5  libro.autor = "Pablo";
```

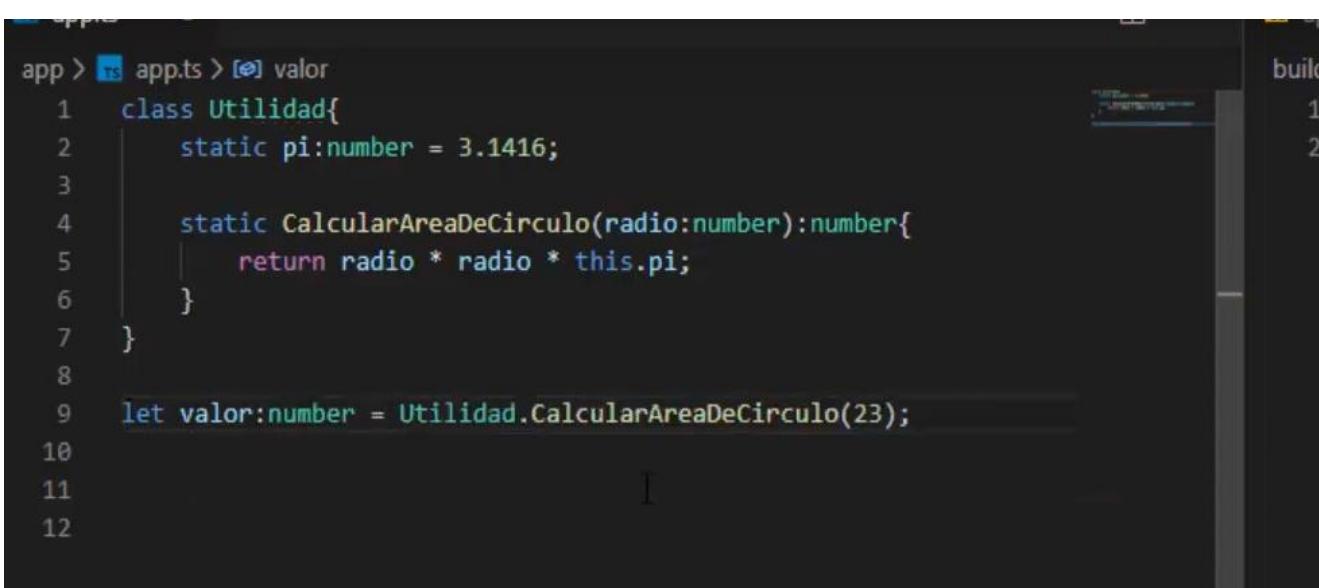
GET y SET

```
app > app.ts > Cliente > nombre
1  class Cliente {
2
3      private _nombre : string;
4      public get nombre(): string {
5          if (this._nombre) {
6              return this._nombre;
7          } else {
8              return this._nombre;
9          }
10     }
11     public set nombre(v: string) {
12         this._nombre = v;
13     }
14
15     constructor(nombre?:string) {
16         this._nombre = nombre;
17     }
18 }
19 }

let cliente = new Cliente();
console.log(cliente.nombre);
```

```
build > app.js > ...
1  class Cliente {
2      constructor(nombre) {
3          this._nombre = nombre;
4      }
5      get nombre() {
6          return this._nombre;
7      }
8      set nombre(v) {
9          this._nombre = v;
10     }
11 }
12 let cliente = new Cliente();
13 console.log(cliente);
14
```

Static



```
app > app.ts > valor
1  class Utilidad{
2      static pi:number = 3.1416;
3
4      static CalcularAreaDeCirculo(radio:number):number{
5          return radio * radio * this.pi;
6      }
7  }
8
9  let valor:number = Utilidad.CalcularAreaDeCirculo(23);
10
11
12
```

Clase Abstracta

```
app > app.ts > Perro > Ruido
1 abstract class Animal{
2     abstract Ruido():void;
3 }
4
5
6 class Gato extends Animal {
7     Ruido(){
8         console.log("miau");
9     }
10 }
11 class Perro extends Animal {
12     Ruido(){
13         console.log("wau");
14     }
15 }
```

```
build > app.js > ...
1 "use strict";
2 class Animal {
3 }
4 class Gato extends Animal {
5     Ruido() {
6         console.log("miau");
7     }
8 }
9 class Perro extends Animal {
10     Ruido() {
11         console.log("wau");
12     }
13 }
14 
```

Interfaces

app.ts

```
app > app.ts > ...
1  interface IMatematica{
2    total?:number,
3    Sumar(a:number, b:number):number,
4    Restar(a:number, b:number):number
5  }
6
7  class Utilidad implements IMatematica {
8    Sumar(a:number, b:number):number{
9      return a + b;
10   }
11   Restar(a:number, b:number):number{
12     return a - b;
13   }
14 }
15
16
17
18
19
20
21
22
23
24
25
26
27
```

app.ts

```
app > app.ts > ...
1  interface IMatematica{
2    total?:number,
3    Sumar(a:number, b:number):number{
4      return a + b;
5    }
6
7  class Utilidad implements IMatematica {
8    Sumar(a:number, b:number):number{
9      return a + b;
10   }
11   Restar(a:number, b:number):number{
12     return a - b;
13   }
14 }
15
16  interface IFigura1{
17    color:string
18 }
19  interface IFigura2 extends IFigura1{
20    alto:number
21 }
22
23  let figura2 = {} as IFigura2;
24  figura2.color = "BLANCO";
25
26
27
```

Decoradores de clases

The screenshot shows the Visual Studio Code interface with two open files: `app.ts` and `app.js`.

app.ts:

```
function DecoradorClase(target:Function){
  target.prototype.Saludar = function(){
    console.log("Hola");
  }
}

@DecoradorClase
class Persona {
  constructor() {
  }
}

let persona = new Persona();
persona.Saludar();
```

app.js:

```
var __decorate = (this && this._decorate) || function(target, name, descriptor, context) {
  var c = arguments.length, r = c < 3 ? target : (c === 3 ? descriptor : undefined);
  if (typeof Reflect === "object" && typeof Reflect.decorate === "function") {
    return Reflect.decorate(descriptor, target, name, context);
  } else for (var i = decorators.length - 1; i >= 0; i--) {
    if (decorators[i].call) {
      r = decorators[i].call(null, descriptor, target, name);
    } else r = decorators[i](target, name);
  }
  return r;
};

function DecoradorClase(target) {
  target.prototype.Saludar = function () {
    console.log("Hola");
  };
}

let Persona = class Persona {
  constructor() {
  }
};

Persona = __decorate([
  DecoradorClase
], Persona);
```

The `app.ts` file contains a class decorator `@DecoradorClase` and a class `Persona`. The `app.js` file shows the generated JavaScript code where the class decorator is converted into a function that uses the `__decorate` helper to modify the prototype of the `Persona` class.

Output Terminal:

```
[22:44:41] Found 1 error. Watching for file changes.
[22:45:08] File change detected. Starting incremental compilation...
[22:45:01] Found 0 errors. Watching for file changes.
```

Angular

- ▶ Angular es un framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página.
- ▶ <https://angular.io/>

Características

- ▶ Aplicaciones web responsivas
- ▶ SPA(Single Page Application)
- ▶ Reactivo RxJs
- ▶ Asíncrono

Angular se compone de

- ▶ Componentes
- ▶ Rutas
- ▶ Directivas
- ▶ Servicios
- ▶ Módulos

Componentes

- ▶ Básicamente se compone de un archivo HTML y una clase Typescript

Servicios

- ▶ Son lugares centralizados de información

Directivas

- ▶ Directivas de componentes, tiene un pedazo de código html reutilizable
- ▶ Directivas Estructurales, modifica el DOM o html ya sea añadiendo/removiendo elementos
- ▶ Directivas de atributos, cambia la apariencia o el comportamiento de un elemento o otra directiva

Rutas

- ▶ Nos ayuda a mostrar diferentes componentes basados en el URL del navegador web

Módulos

- ▶ Nos permite agrupar componentes, rutas , directivas y servicios

Angular contiene

- ▶ **Componentes**
- ▶ **Plantillas**
- ▶ **Directivas**
- ▶ **Eventos**
- ▶ **Pipe**
- ▶ **Rutas de navegación**
- ▶ **Decoradores**
- ▶ **Servicios y HttpClient**
- ▶ **Clases TypeScript**
- ▶ **Formulario**
- ▶ **Data Binding ngModel**
- ▶ **Validaciones**
- ▶ **Módulos**
- ▶ **Angular Material**

Versiones

The screenshot shows a browser window with the Angular website open at angular.io/guide/releases. The page features a sidebar on the left with navigation links like Introduction, Getting Started, Understanding Angular, Developer guides, Best Practices, Angular Tools, Tutorials, and Release Information (with sub-links for Release Practices, Roadmap, and Browser Support). The main content area displays a table of Angular versions under support, followed by a note about unsupported versions. A sidebar on the right provides links to various Angular documentation sections. The browser taskbar at the bottom shows several pinned icons and the system tray.

VERSION	STATUS	RELEASED	ACTIVE ENDS	LTS ENDS
^13.0.0	Active	Nov 04, 2021	May 04, 2022	May 04, 2023
^12.0.0	LTS	May 12, 2021	Nov 12, 2021	Nov 12, 2022
^11.0.0	LTS	Nov 11, 2020	May 11, 2021	May 11, 2022
^10.0.0	LTS	Jun 24, 2020	Dec 24, 2020	Dec 24, 2021

Angular versions v4, v5, v6, v7, v8, and v9 are no longer under support.

Angular versioning and releases

- Angular versioning
- Supported update paths
- Preview releases
- Release frequency
- Support policy and schedule
- LTS fixes
- Deprecation practices
- Public API surface
- Angular Labs

Lenguaje utilizado TypeScript

- ▶ Tipos basados en clases e interfaces
- ▶ Tipos de datos mas fuerte
- ▶ Atributos métodos y constructores
- ▶ Decoradores y mas

Herramientas necesarias

- ▶ Instalar Nodejs
- ▶ Instalar TypeScript
- ▶ **npm i -g typescript**
- ▶ **tsc -v**
- ▶ Instalar angular CLI



A screenshot of a web browser displaying the Angular v11 guide for setting up the local environment. The URL in the address bar is `v11.angular.io/guide/setup-local`. The page has a dark 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 sidebar with navigation links like Introduction, Getting Started (with sub-links for What is Angular?, Try it, and Setup), Understanding Angular, Developer Guides, Best Practices, Angular Tools, Tutorials, Release Information, and Reference. The main content section is titled "Install the Angular CLI". It explains the purpose of the CLI and provides a command-line instruction: `npm install -g @angular/cli@11`. Below this, there's a section titled "Create a workspace and initial application" with a link to "workspace". The right side of the page contains a sidebar with links for "Setting up the local environment and workflow", "Prerequisites", "Install the Angular CLI" (marked with a bullet point), "Create a workspace and application", "Run the application", and "Next steps".

← → ⌂ v11.angular.io/guide/setup-local

≡  ANGULAR FEATURES DOCS RESOURCES EVENTS BLOG Search [Twitter](#) [GitHub](#)

Introduction

Getting Started

What is Angular?

Try it

Setup

Understanding Angular

Developer Guides

Best Practices

Angular Tools

Tutorials

Release Information

Reference

Install the Angular CLI

You use the Angular CLI to create projects, generate application and library code, and perform a variety of ongoing development tasks such as testing, bundling, and deployment.

To install the Angular CLI, open a terminal window and run the following command:

```
npm install -g @angular/cli@11
```

Create a workspace and initial application [🔗](#)

You develop apps in the context of an Angular [workspace](#).

To create a new workspace and initial starter app:

Setting up the local environment and workflow

Prerequisites

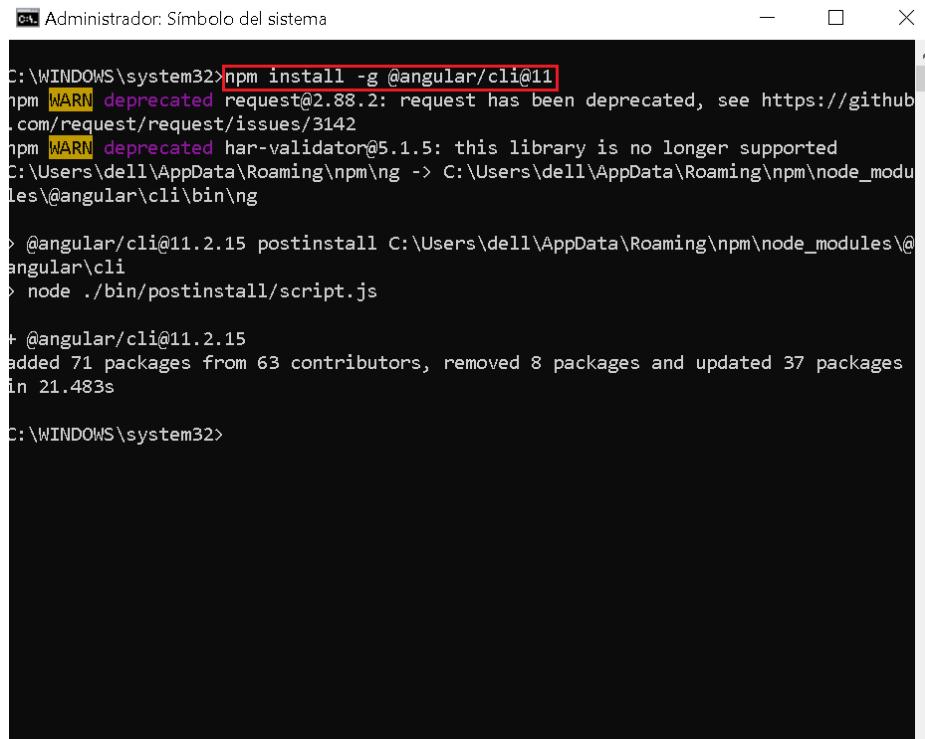
- **Install the Angular CLI**

Create a workspace and application

Run the application

Next steps

En la consola cmd como administrador



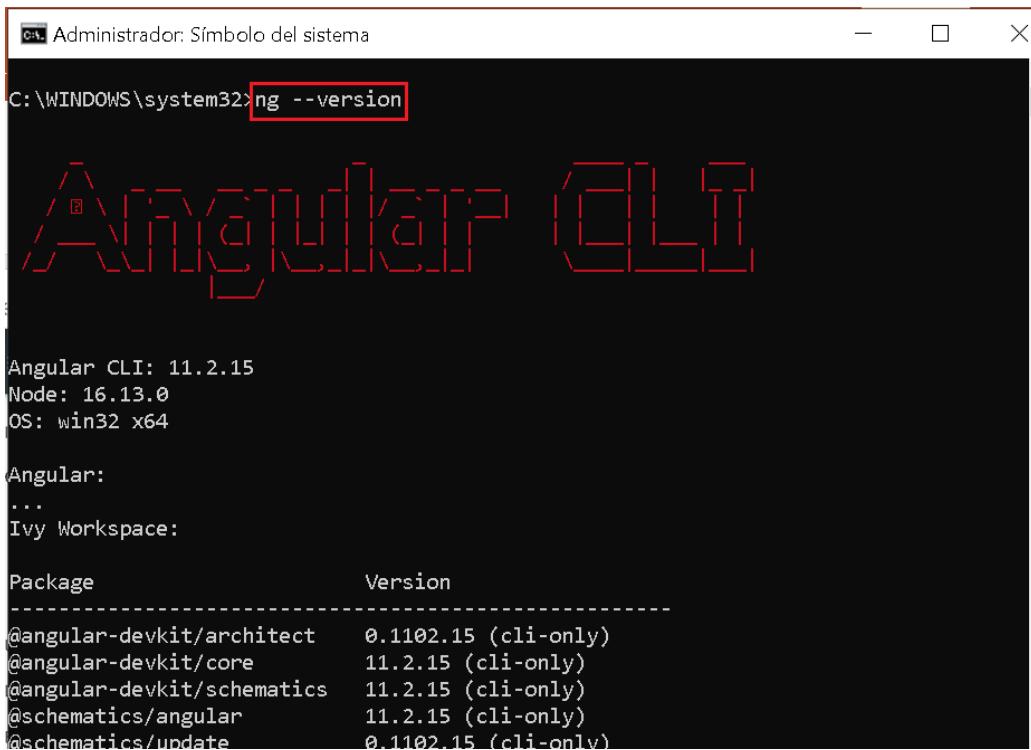
```
C:\WINDOWS\system32>npm install -g @angular/cli@11
npm [WARN] deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
npm [WARN] deprecated har-validator@5.1.5: this library is no longer supported
C:\Users\dell\AppData\Roaming\npm\ng -> C:\Users\dell\AppData\Roaming\npm\node_modules\@angular\cli\bin\ng

> @angular/cli@11.2.15 postinstall C:\Users\dell\AppData\Roaming\npm\node_modules\@angular\cli
> node ./bin/postinstall/script.js

+ @angular/cli@11.2.15
added 71 packages from 63 contributors, removed 8 packages and updated 37 packages in 21.483s

C:\WINDOWS\system32>
```

Comprobamos que fue exitosa la instalación



```
C:\WINDOWS\system32:ng --version

Angular CLI: 11.2.15
Node: 16.13.0
OS: win32 x64

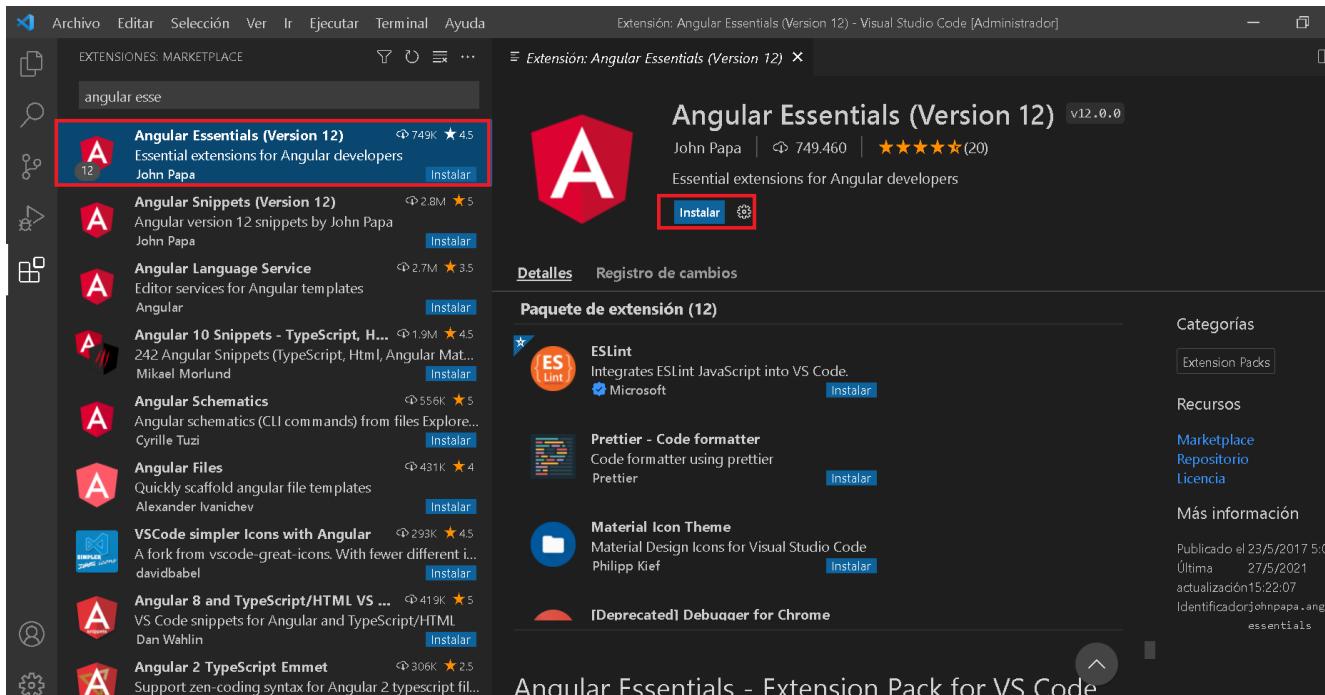
Angular:
...
Ivy Workspace:

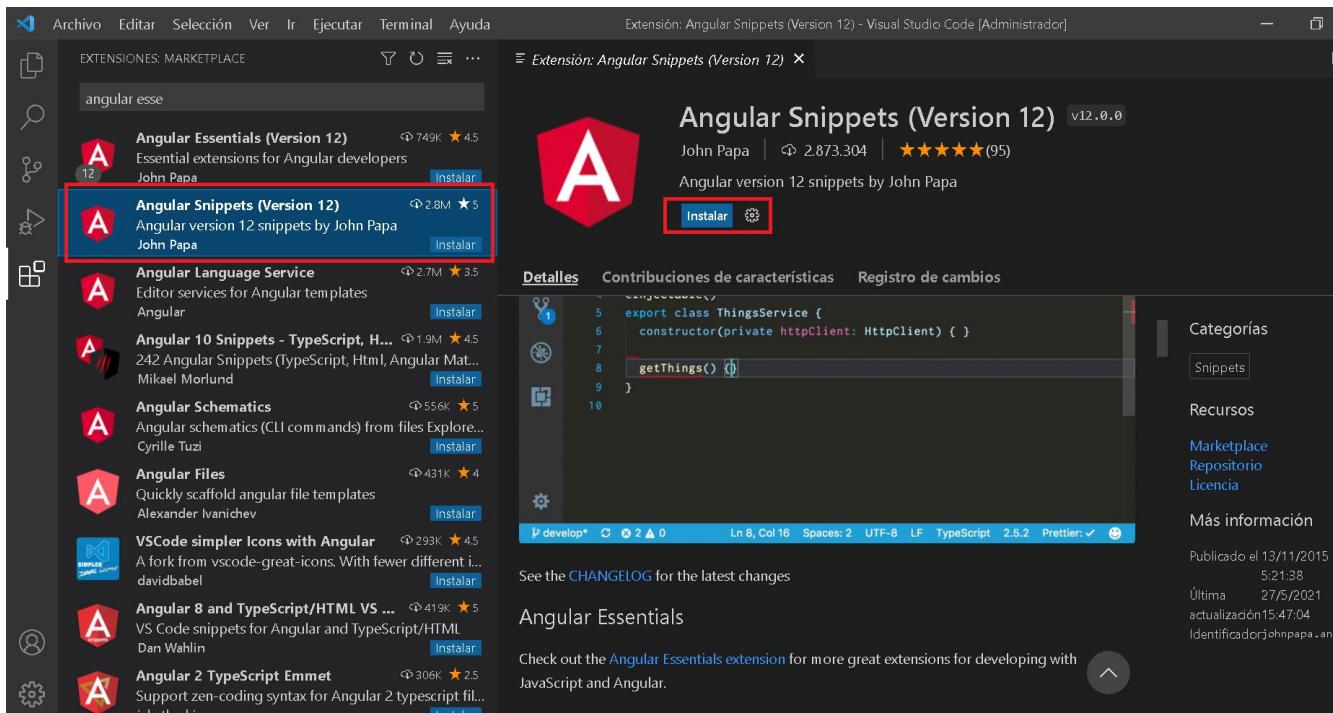
      Package          Version
      -----



@angular-devkit/architect    0.1102.15 (cli-only)
@angular-devkit/core         11.2.15 (cli-only)
@angular-devkit/schematics   11.2.15 (cli-only)
@schematics/angular          11.2.15 (cli-only)
@schematics/update            0.1102.15 (cli-only)
```

Instalamos extensiones necesarias para Visual Studio Code





Extensión: Angular Language Service - Visual Studio Code [Administrador]

EXTENSIONES: MARKETPLACE

angular esse

- Angular Essentials (Version 12)** ⚡ 749K ★ 4.5
Essential extensions for Angular developers
John Papa [Instalar](#)
- Angular Snippets (Version 12)** ⚡ 2.8M ★ 5
Angular version 12 snippets by John Papa
John Papa [Instalar](#)
- Angular Language Service** ⚡ 2.7M ★ 3.5
Editor services for Angular templates
Angular [Instalar](#) **Instalar**
- Angular 10 Snippets - TypeScript, H...** ⚡ 1.9M ★ 4.5
242 Angular Snippets (TypeScript, HTML, Angular Mat..
Mikael Mørlund [Instalar](#)
- Angular Schematics** ⚡ 556K ★ 5
Angular schematics (CLI commands) from files Explore..
Cyrille Tuzi [Instalar](#)
- Angular Files** ⚡ 431K ★ 4
Quickly scaffold angular file templates
Alexander Ivanichev [Instalar](#)
- VSCODE simpler Icons with Angular** ⚡ 293K ★ 4.5
A fork from vscode-great-icons. With fewer different ...
davidabel [Instalar](#)
- Angular 8 and TypeScript/HTML VS ...** ⚡ 419K ★ 5
VS Code snippets for Angular and TypeScript/HTML
Dan Wahlin [Instalar](#)
- Angular 2 TypeScript Emmet** ⚡ 306K ★ 2.5
Support zen-coding syntax for Angular 2 typescript fil...

Angular Language Service v13.0.0

Angular | ⚡ 2.786.784 | ★★★★★(197)

Editor services for Angular templates

[Instalar](#)

Detalles Contribuciones de características Registro de cambios

Angular Language Service

app.component.html

Angular Language Service

Categorías Programming Languages

Recursos

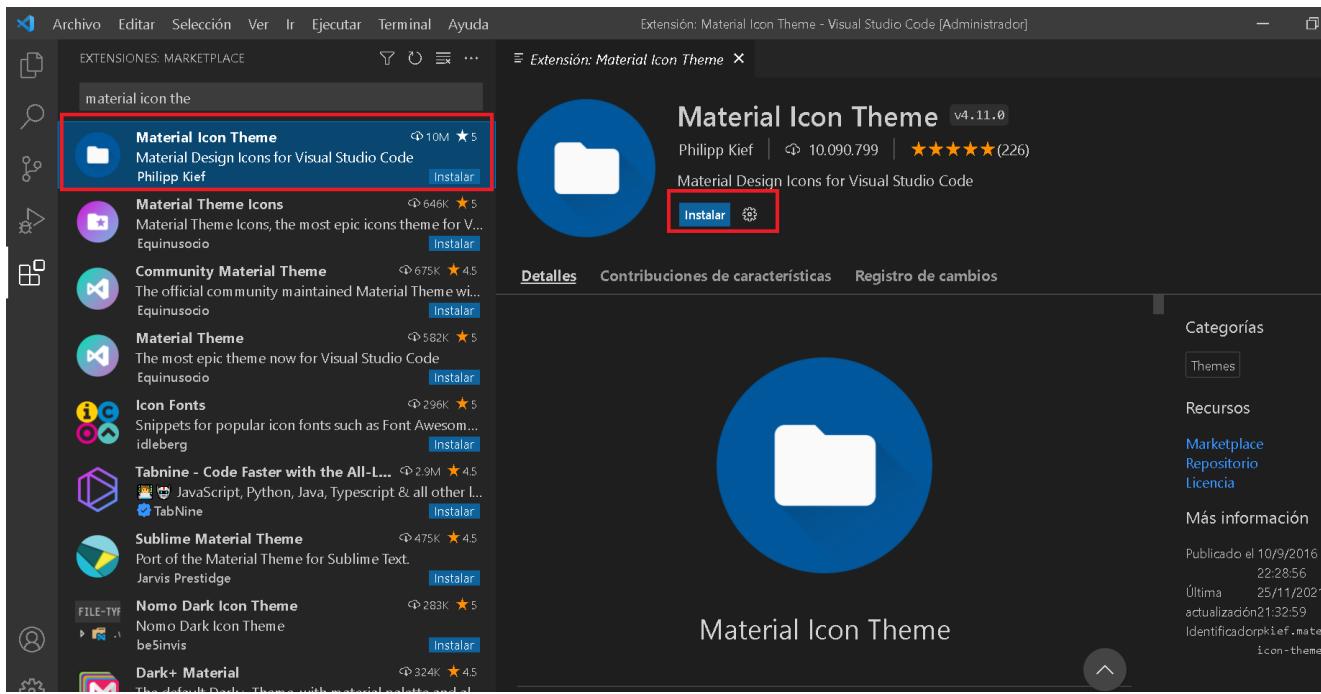
Marketplace Repository

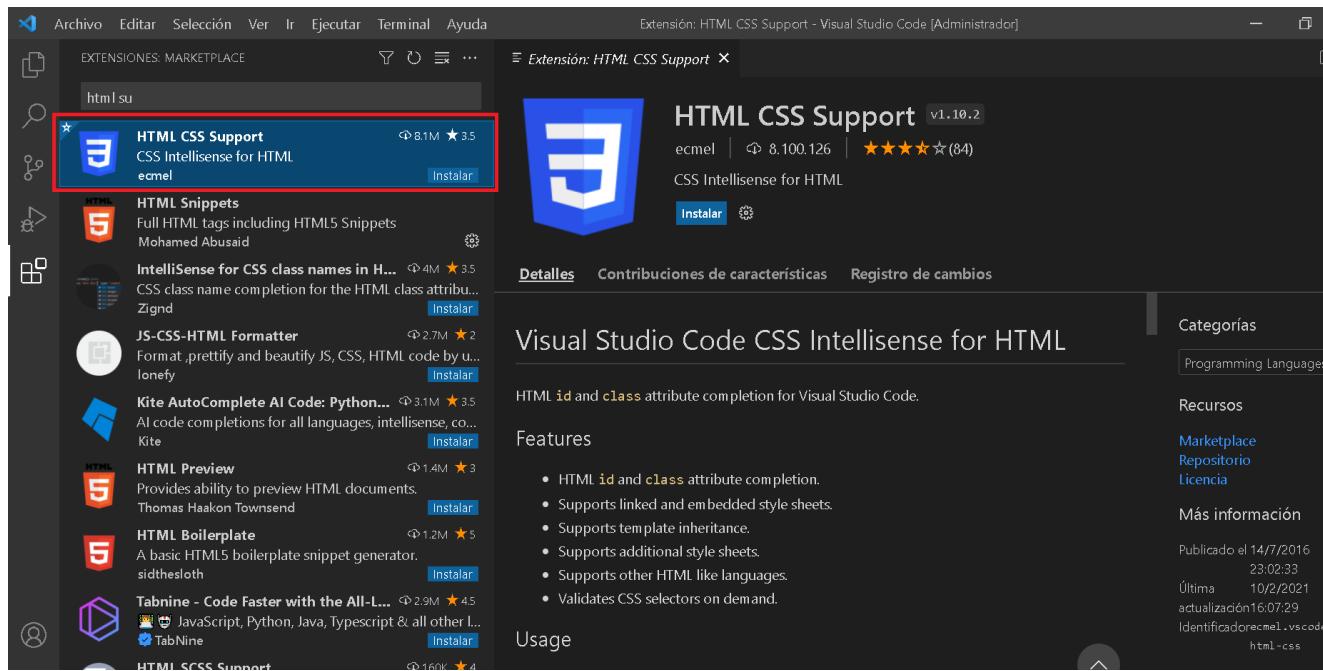
Más información

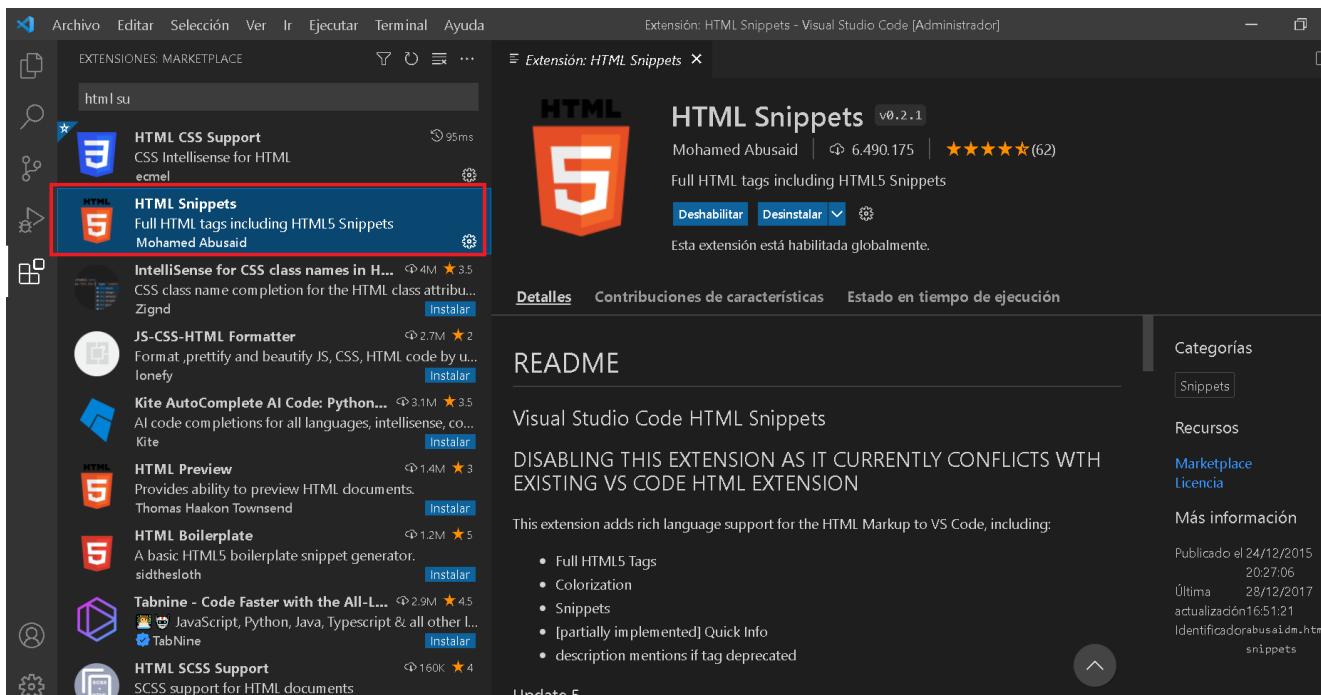
Publicado el 4/4/2017 0:31:22

Última actualización 3/11/2021 21:57:12

Identificador angular.ng-template







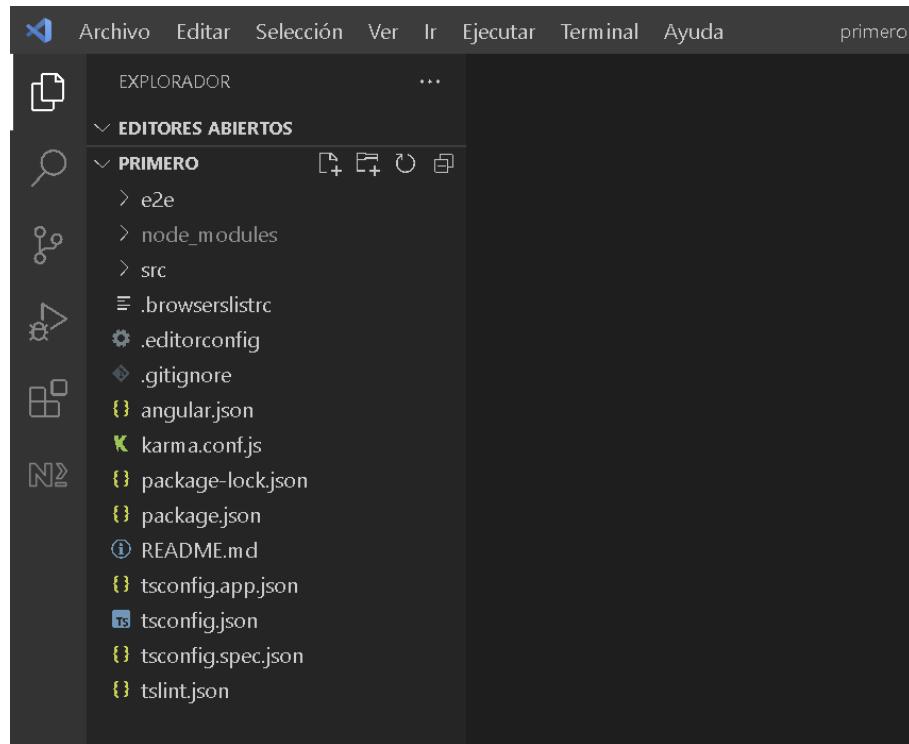
Creamos nuestro primer proyecto

- ▶ Nos dirigimos al directorio deseado desde consola y ejecutamos
- ▶ **ng new nombredeproyecto**

C:\ Administrador: Símbolo del sistema

```
C:\Users\dell\Desktop\formacion_angular>ng new primero
? Do you want to enforce stricter type checking and stricter bundle budgets in the workspace?
  This setting helps improve maintainability and catch bugs ahead of time.
  For more information, see https://angular.io/strict Yes
? Would you like to add Angular routing? No
? Which stylesheet format would you like to use? CSS
CREATE primero/angular.json (3631 bytes)
CREATE primero/package.json (1208 bytes)
CREATE primero/README.md (1017 bytes)
CREATE primero/tsconfig.json (783 bytes)
CREATE primero/tslint.json (3185 bytes)
CREATE primero/.editorconfig (274 bytes)
CREATE primero/.gitignore (631 bytes)
CREATE primero/.browserslistrc (703 bytes)
CREATE primero/karma.conf.js (1424 bytes)
```

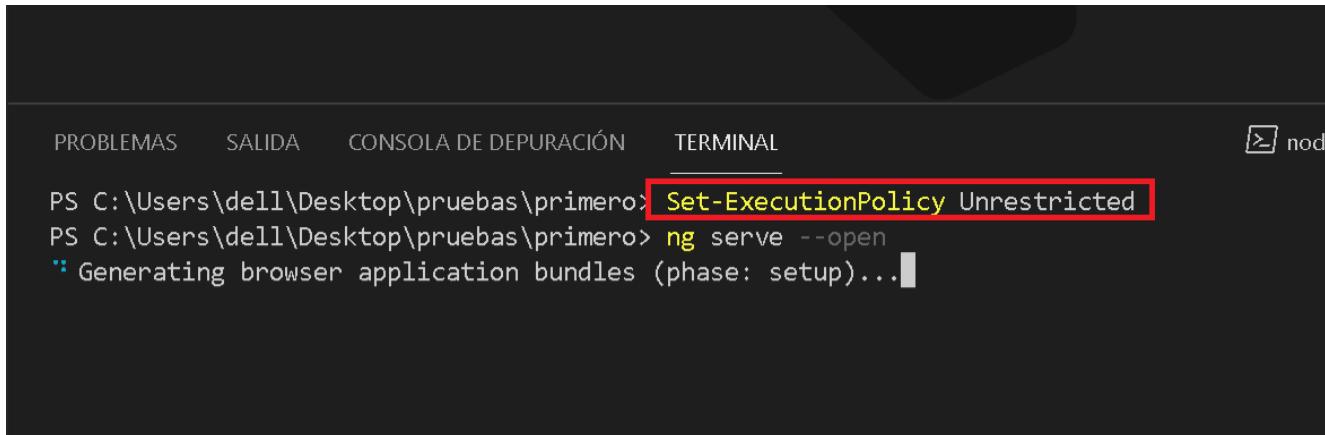
Una vez creado el proyecto podemos abrirlo el proyecto con nuestro editor



Lanzamos nuestro proyecto
angular

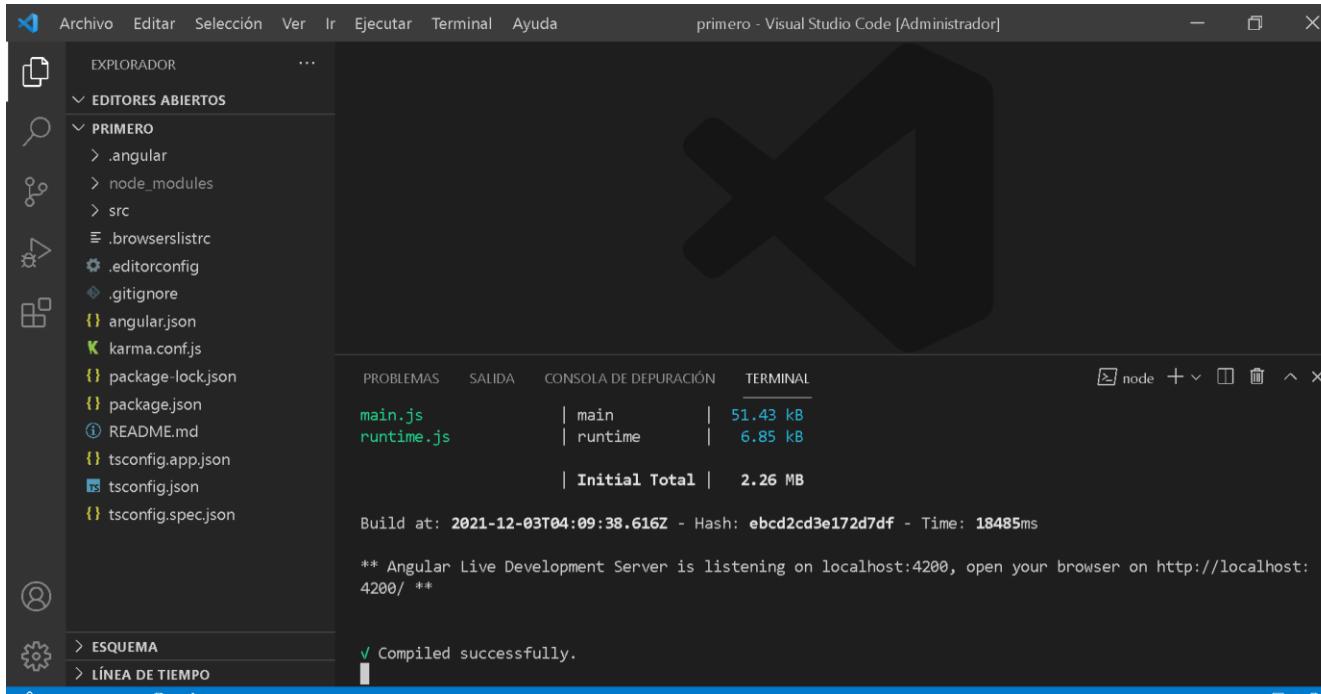
► **ng serve --open**

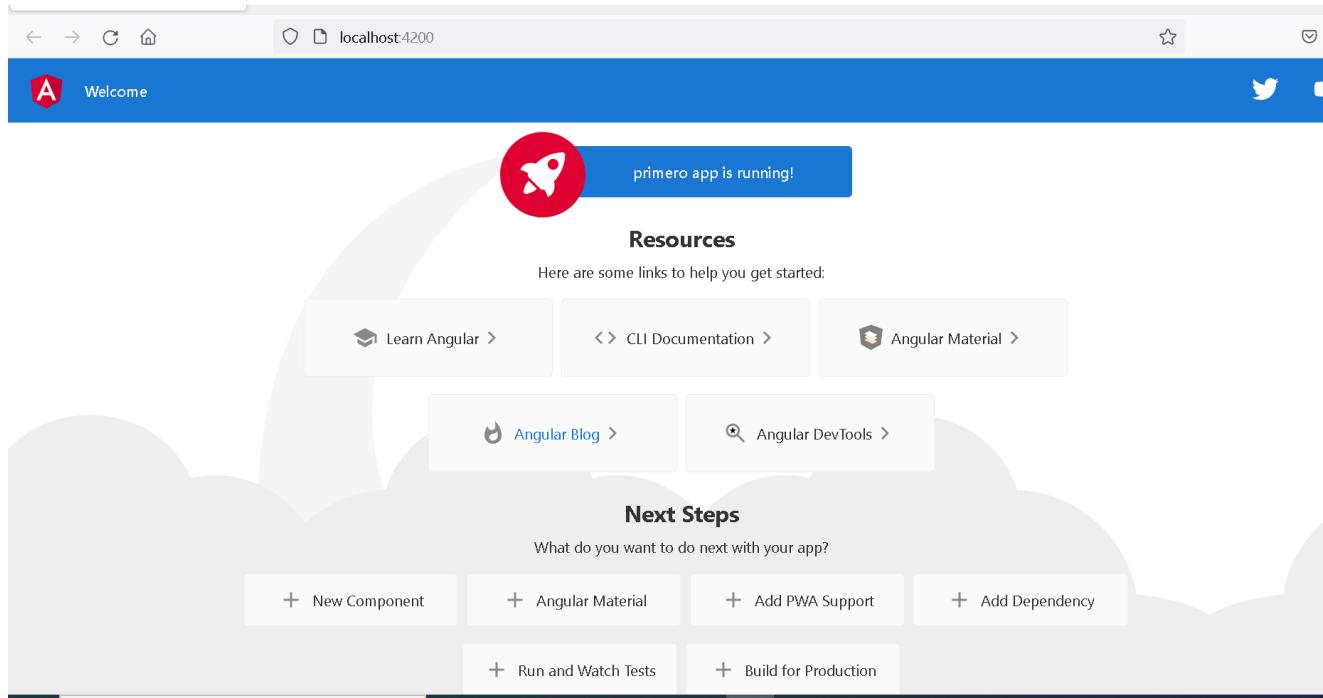
Para el que tiene inconvenientes con su consola desde Visual Studio Code



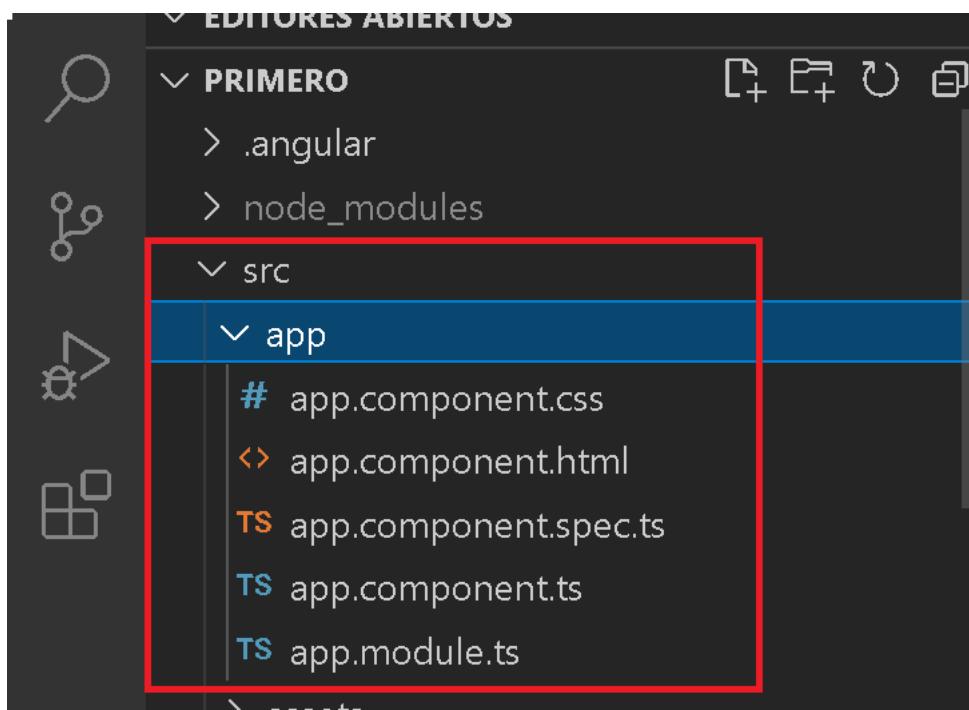
The screenshot shows the Visual Studio Code interface with a dark theme. At the top, there are tabs for PROBLEMAS, SALIDA, CONSOLA DE DEPURACIÓN, and TERMINAL. The TERMINAL tab is underlined, indicating it is active. To the right of the tabs is a search bar containing the text "node". Below the tabs, the terminal window displays the following command-line session:

```
PS C:\Users\dell\Desktop\pruebas\primero> Set-ExecutionPolicy Unrestricted
PS C:\Users\dell\Desktop\pruebas\primero> ng serve --open
  Generating browser application bundles (phase: setup)...
```





Nos dirigimos a este directorio



Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda

app.component.ts - primero - Visual Studio Code

EXPLORADOR

EDITORES ABIERTOS

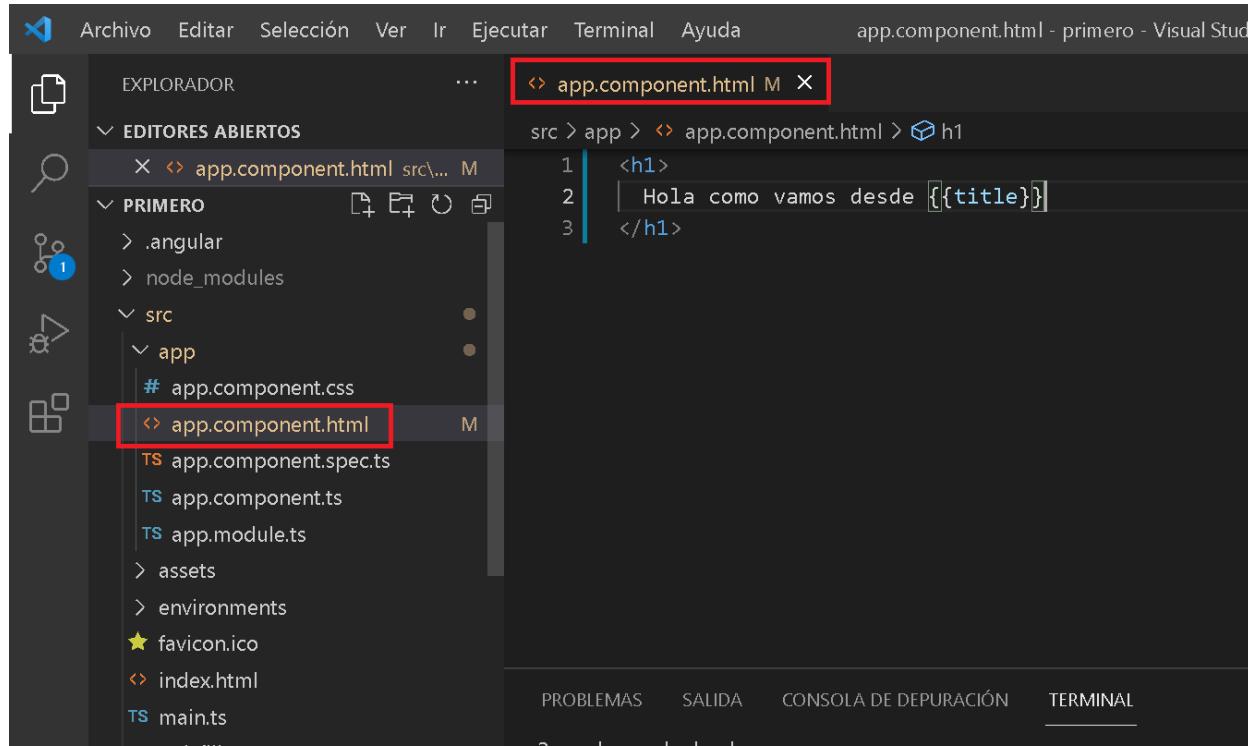
PRIMERO

- > .angular
- > node_modules
- > src
 - > app
 - # app.component.css
 - <> app.component.html M
 - TS app.component.spec.ts
 - TS app.component.ts
 - TS app.module.ts
 - > assets
 - > environments

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

```
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   title = 'primero';
10 }
11
```

Agregamos esto en app.component.html



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

- File Explorer (Left):** Shows the project structure under "PRIMERO". Key files listed include: .angular, node_modules, src (with app, app.component.css, app.component.html, app.component.spec.ts, app.component.ts, app.module.ts), assets, environments, favicon.ico, index.html, and main.ts.
- Editor (Center):** The file "app.component.html" is open. The code content is:

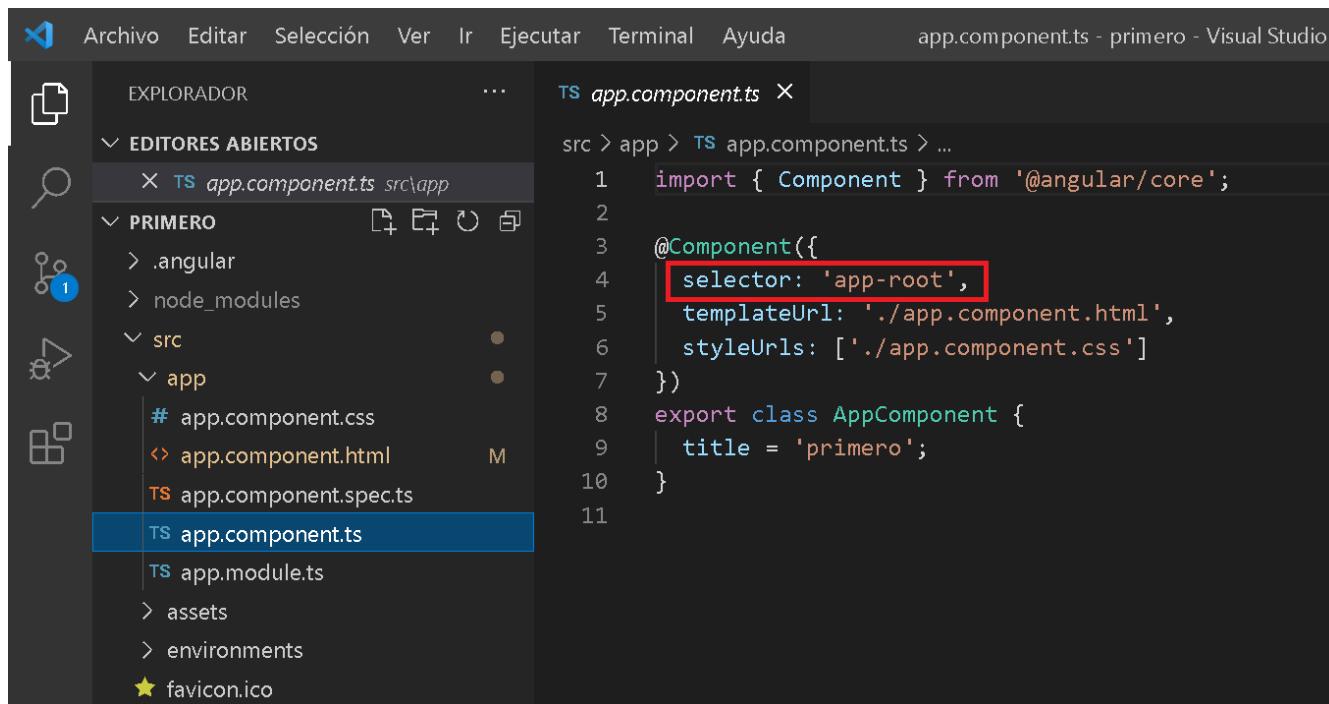
```
<h1>
  Hola como vamos desde {{title}}
</h1>
```
- Status Bar (Bottom):** Shows tabs for PROBLEMAS, SALIDA, CONSOLA DE DEPURACIÓN, and TERMINAL.



Hola como vamos desde primero



El selector app-root es incluida en el index principal para agregar todo el componente a la vista html



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

- File Explorer (Left):** Shows the project structure:
 - PRIMERO** folder contains: .angular, node_modules, src (marked with a blue circle), app (marked with a blue circle), app.component.css, app.component.html, app.component.spec.ts, and app.component.ts (highlighted with a blue selection bar).
 - assets, environments, and favicon.ico are also listed.
- Code Editor (Right):** Displays the content of **app.component.ts**.

```
src > app > app.component.ts > ...
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   title = 'primero';
10 }
```
- Status Bar:** Shows "app.component.ts - primero - Visual Studio Code".

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

- File Explorer (Left):** Shows the project structure for an Angular application named "PRIMERO". The "src" folder contains "app", "assets", "environments", and "environments.ts". The "app" folder contains "app.component.css", "app.component.html", "app.component.spec.ts", "app.component.ts", and "app.module.ts".
- Editor (Center):** Displays the content of the "index.html" file. The code is as follows:

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Primero</title>
<base href="/">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
<app-root></app-root>
</body>
</html>
```

The line containing the placeholder `<app-root></app-root>` is highlighted with a red rectangle.

- Bottom Status Bar:** Shows the build information: "Build at: 2021-12-03T04:17:33.210Z - Hash: 683009ac500d4dbe - Time: 420ms".

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

- File Explorer (Left):** Shows the project structure with files like index.html, app.component.ts, app.module.ts, assets, environments, favicon.ico, and tsconfig.json.
- Editor (Center):** Displays the contents of app.component.ts. The code defines an AppComponent class with properties nombre and formacion.
- Code Block (Highlighted):** The code block from line 11 to line 13 is highlighted with a red rectangle:

```
nombre:string = "Rolando";  
formacion:string = "Angular";
```
- Code Block (Highlighted):** The entire class definition from line 8 to line 15 is highlighted with a red rectangle:

```
export class AppComponent {  
  title = 'primero';  
  
  nombre:string = "Rolando";  
  formacion:string = "Angular";  
}
```
- Bottom Navigation:** Includes tabs for PROBLEMAS, SALIDA, CONSOLA DE DEPURACIÓN, and TERMINAL.

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

- File Explorer (Left):** Shows the project structure with files like `app.component.ts`, `app.component.html`, `app.module.ts`, etc.
- Editor (Center):** Displays the content of `app.component.html`. The code is:

```
src > app > app.component.html > ul
1 <h1>
2 Bienvenidos al la formacion de {{formacion}}
3
4 </h1>
5
6 <ul>
7   <li>El formador {{nombre}}</li>
8   <li>El proyecto es {{title}}</li>
9 </ul>
```
- Terminal (Bottom):** Shows the message "3 unchanged chunks".

Estructura de los componentes

- ▶ Son pequeñas partes o bloques que forman parte de nuestra aplicación, y cumplen un rol específico dentro, en términos de programación son clases de typescript
- ▶ Menú de navegación
- ▶ Barra lateral
- ▶ Un pie de pagina
- ▶ Contenido dinámico o principal

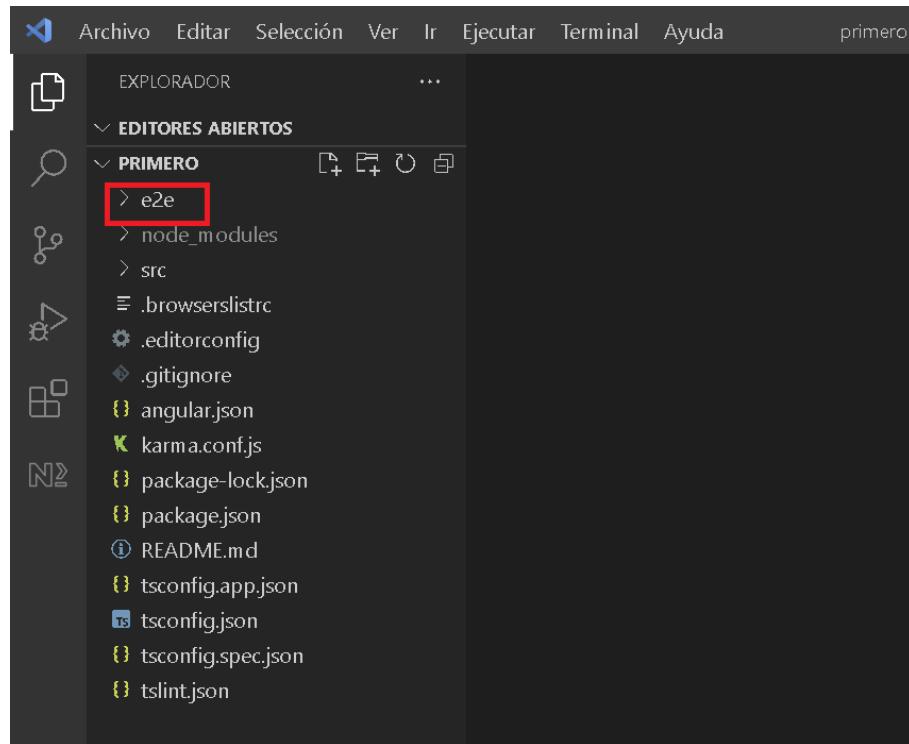
Características de los componentes

- ▶ Clase TS con una función
- ▶ Patrón composite
- ▶ Se puede anidar
- ▶ Se puede enrutar
- ▶ Ciclo de vida
- ▶ MVC
- ▶ Asíncronos
- ▶ Inyección de dependencia
- ▶ Tiene sus propios estilos

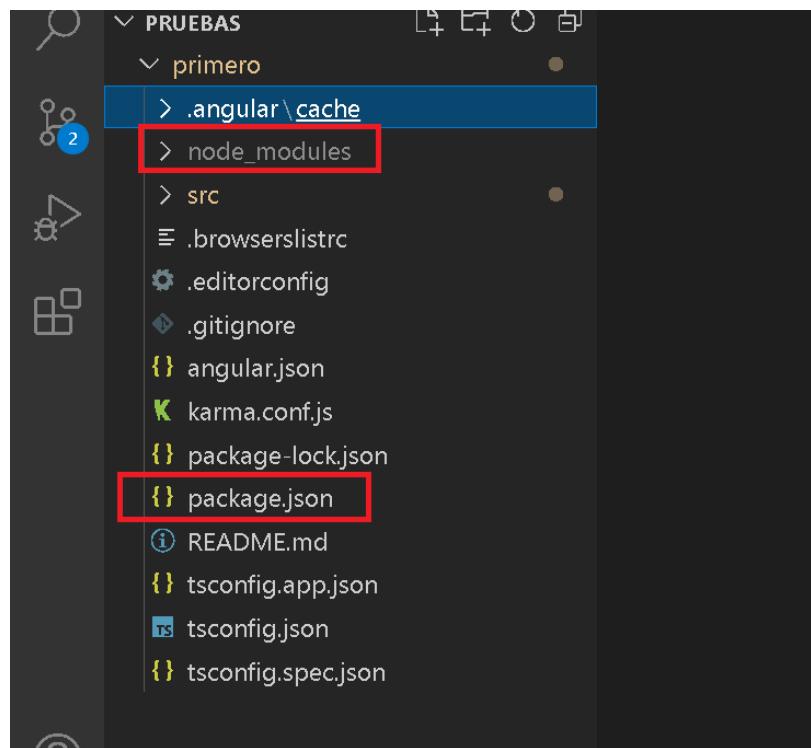
Archivos del proyecto Angular

- ▶ Estaremos viendo los archivo dentro del proyecto angular

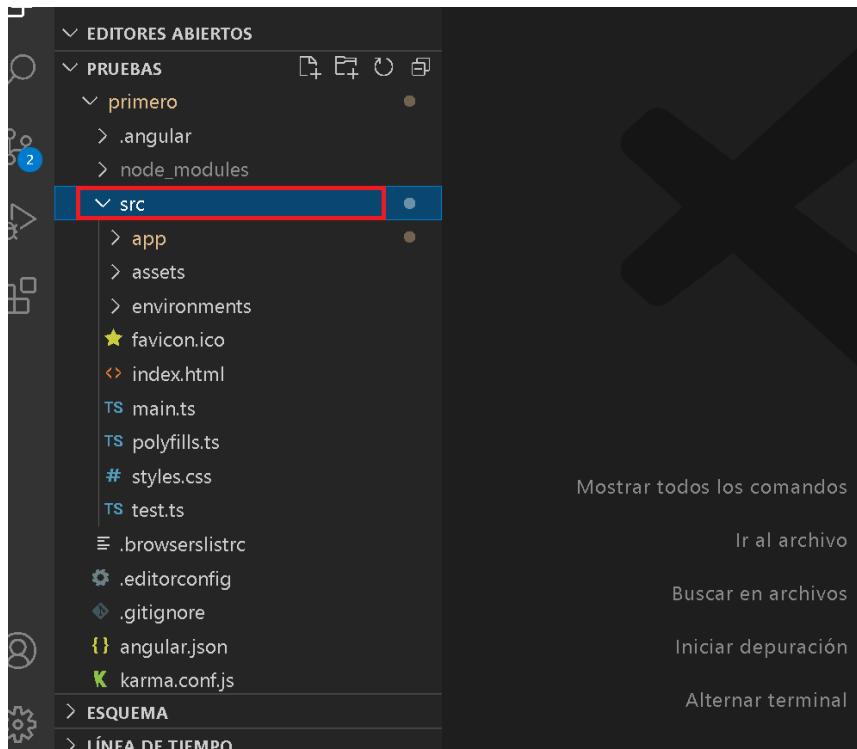
El directorio e2e para pruebas unitarias



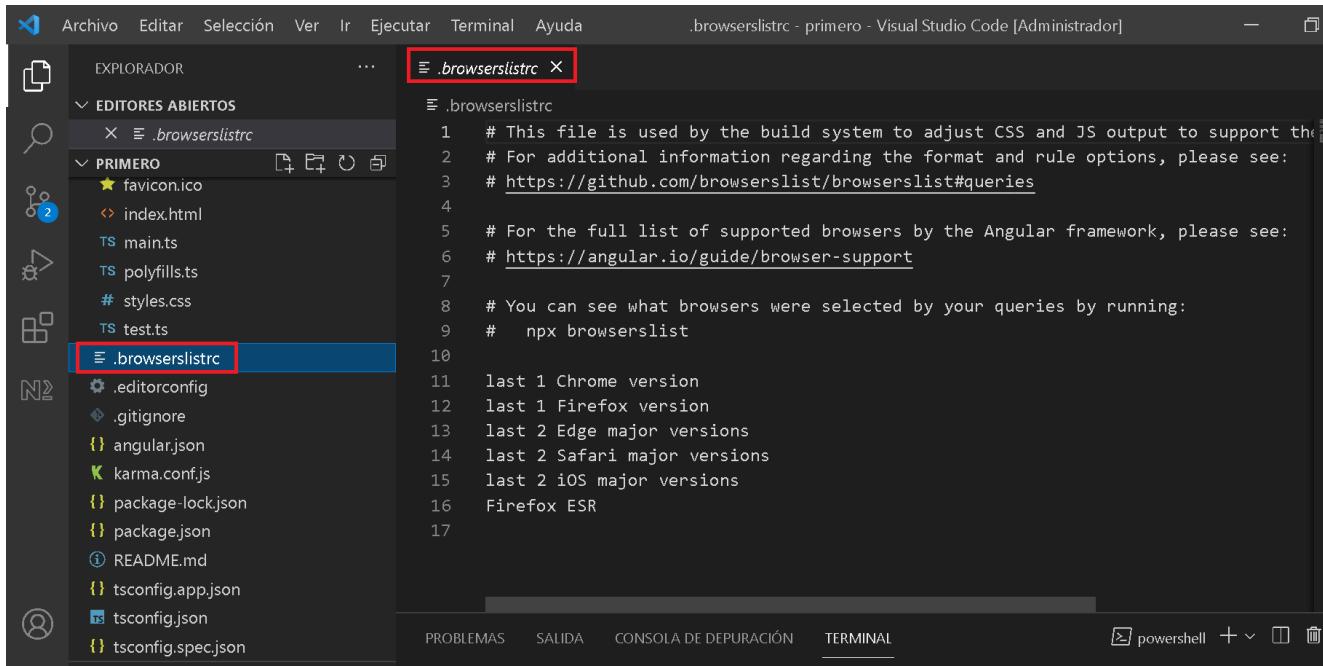
En `node_modules` encuentra todas las librerías y dependencias del proyecto, también maneja dependencias todo lo relacionado con dependencias de angular en general, y es manejado por el `package.json` de forma automática no deberíamos de tocarlo



La carpeta src es super importante contiene todo el código de nuestra aplicación



El archivo browserlistrc para incrementar/modificar compatibilidad con navegadores de internet existentes



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

- File Explorer (Left):** Shows a project structure with files like favicon.ico, index.html, mains.ts, polyfills.ts, styles.css, test.ts, and .browserslistrc.
- Editor (Center):** Displays the content of the .browserslistrc file. The file is a configuration file used by build systems to adjust CSS and JS output for specific browsers. It contains comments and a list of browser queries.
- Bottom Navigation Bar:** Includes tabs for PROBLEMAS, SALIDA, CONSOLA DE DEPURACIÓN, and TERMINAL. The TERMINAL tab is selected.
- Bottom Right:** Shows a powershell terminal icon and other interface elements.

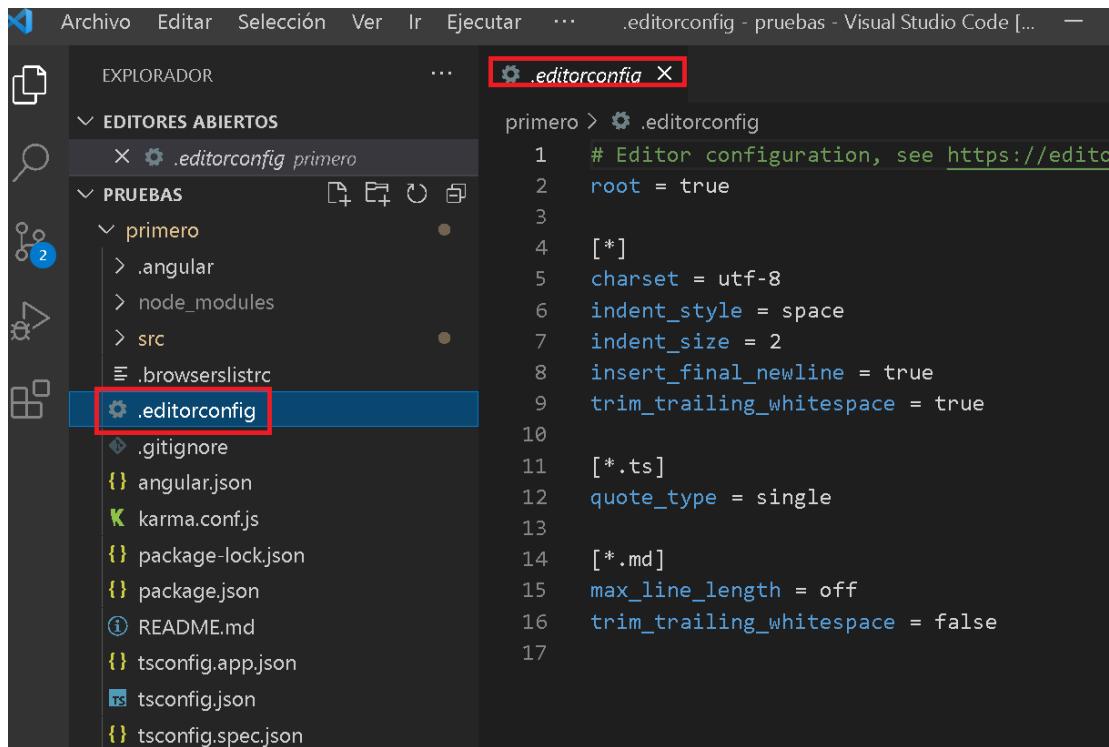
```
# This file is used by the build system to adjust CSS and JS output to support the following browsers.
# For additional information regarding the format and rule options, please see:
# https://github.com/browserslist/browserlist#queries

# For the full list of supported browsers by the Angular framework, please see:
# https://angular.io/guide/browser-support

# You can see what browsers were selected by your queries by running:
# npx browserlist

last 1 Chrome version
last 1 Firefox version
last 2 Edge major versions
last 2 Safari major versions
last 2 iOS major versions
Firefox ESR
```

El archivo editorconfig no se ocupa generalmente contiene toda la configuración del editor que podríamos modificar si lo necesitamos



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays a tree view of files and folders. A file named ".editorconfig" is selected and highlighted with a red border. The main workspace shows the content of the ".editorconfig" file in a code editor window. The file contains configuration settings for various file types and root level.

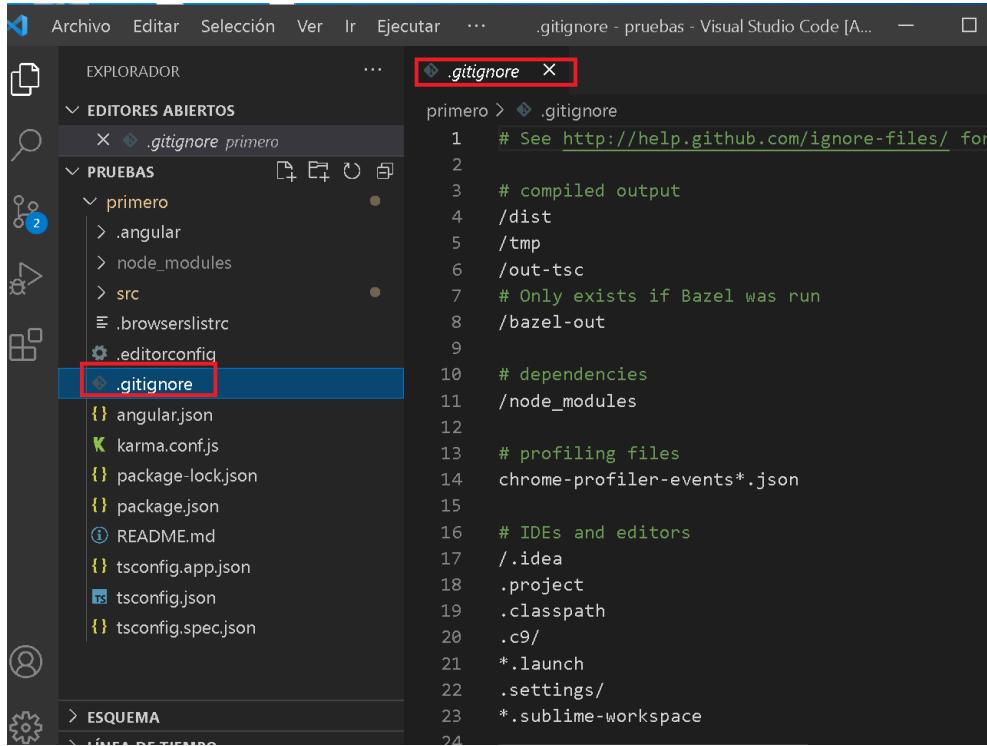
```
# Editor configuration, see https://editorconfig.org
root = true

[*]
charset = utf-8
indent_style = space
indent_size = 2
insert_final_newline = true
trim_trailing whitespace = true

[*.ts]
quote_type = single

[*.md]
max_line_length = off
trim_trailing whitespace = false
```

El archivo .gitignore es un archivo de github que nos permite ocultar ignorar algún archivo o directorio que no quisiéramos compartir en un repositorio git



The screenshot shows the Visual Studio Code interface. The title bar indicates the current file is ".gitignore - pruebas - Visual Studio Code [A...]".

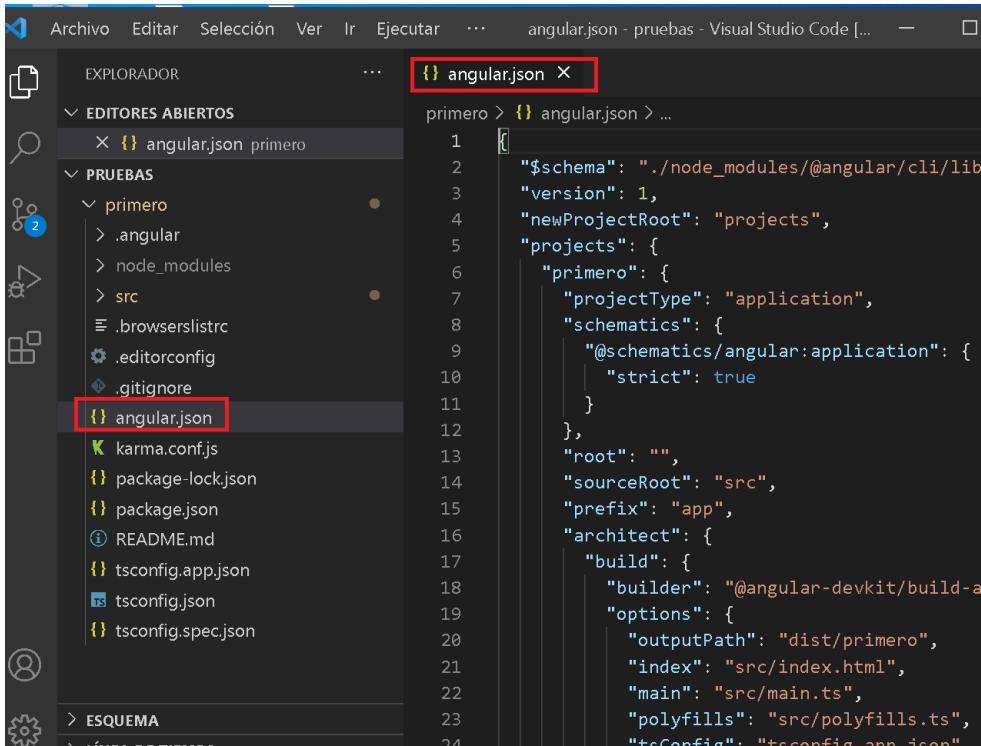
The Explorer sidebar on the left lists several files and folders:

- EDITORES ABIERTOS:
 - .gitignore primero
- PRUEBAS:
 - primero
 - .angular
 - node_modules
 - src
 - browserslistrc
 - .editorconfig
 - .gitignore
 - angular.json
 - karma.conf.js
 - package-lock.json
 - package.json
 - README.md
 - tsconfig.app.json
 - tsconfig.json
 - tsconfig.spec.json

The ".gitignore" file in the Explorer sidebar is highlighted with a red box. The main editor area displays the content of the ".gitignore" file:

```
1 # See http://help.github.com/ignore-files/ for more information
2
3 # compiled output
4 /dist
5 /tmp
6 /out-tsc
7 # Only exists if Bazel was run
8 /bazel-out
9
10 # dependencies
11 /node_modules
12
13 # profiling files
14 chrome-profiler-events*.json
15
16 # IDEs and editors
17 /.idea
18 .project
19 .classpath
20 .c9/
21 *.launch
22 .settings/
23 *.sublime-workspace
24
```

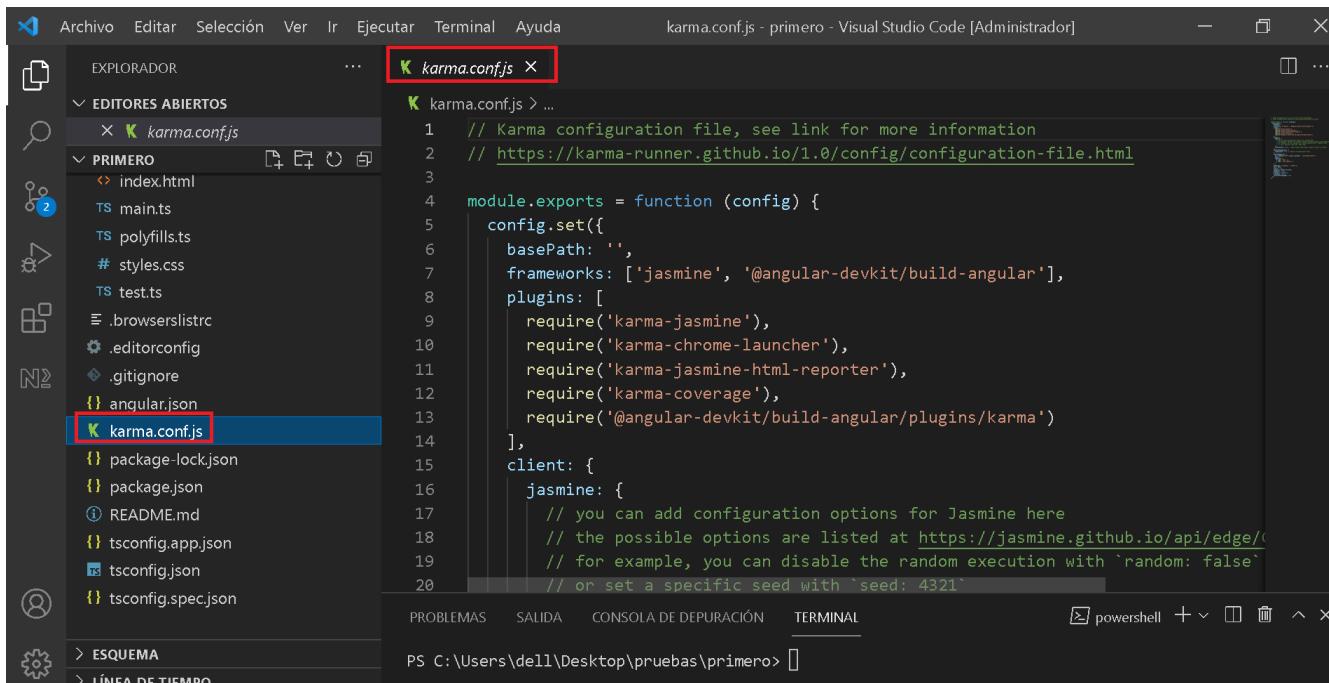
Angular.json es el archivo principal de configuración del proyecto



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

- Editor Area:** Displays the content of the `angular.json` file, which is the main configuration file for an Angular project.
- Explorer Bar:** Shows the project structure under the `PRUEBAS` folder:
 - `primero` (selected)
 - `.angular`
 - `node_modules`
 - `src`
 - `.browserslistrc`
 - `.editorconfig`
 - `.gitignore`
 - `angular.json` (highlighted with a red box)
 - `karma.conf.js`
 - `package-lock.json`
 - `package.json`
 - `README.md`
 - `tsconfig.app.json`
 - `tsconfig.json`
 - `tsconfig.spec.json`
- Status Bar:** Shows the path `primero > {} angular.json > ...` and the file name `angular.json - pruebas - Visual Studio Code [...]`.

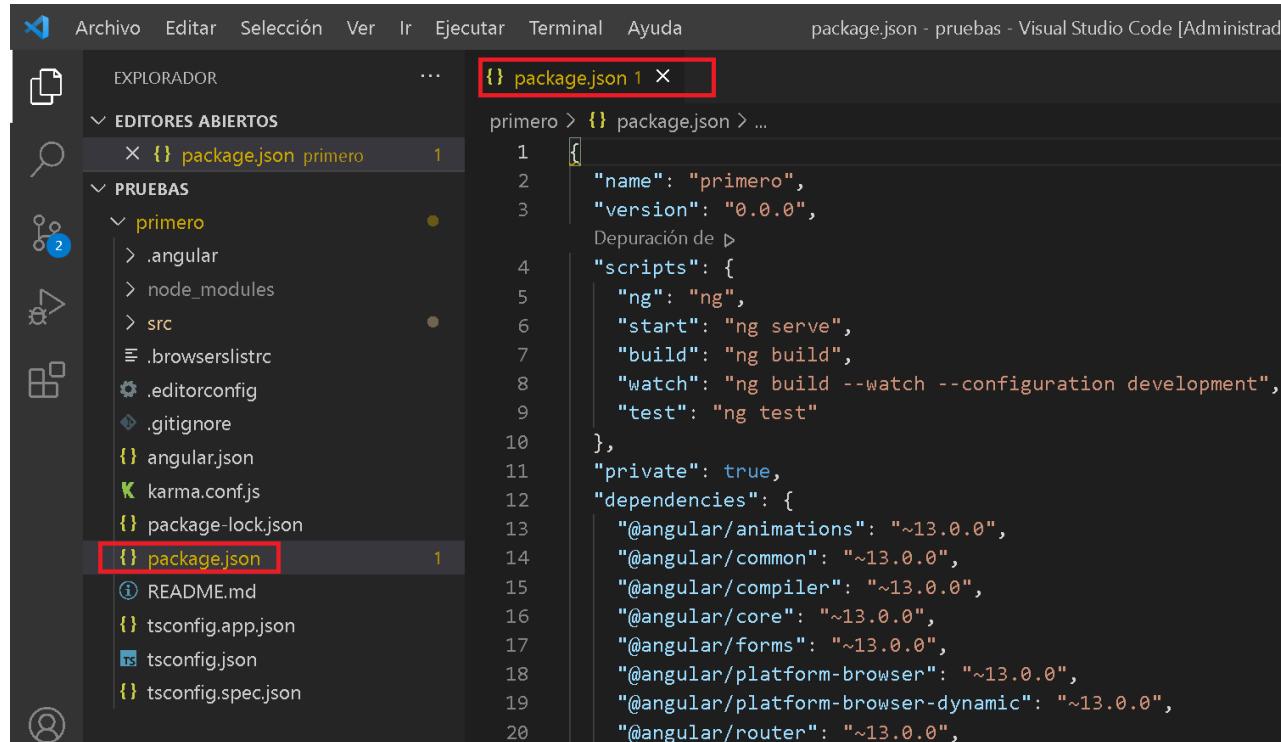
El archivo karma.conf.js son las configuraciones para pruebas unitarias basadas en karma



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

- File Explorer (Left):** Shows a project structure for a folder named "PRIMERO". The "karma.conf.js" file is highlighted with a red box.
- Editor (Center):** Displays the contents of the "karma.conf.js" file. The code is a Karma configuration file, starting with comments about the file being a configuration file and linking to the official documentation. It defines the module exports function, sets up frameworks (jasmine and angular-devkit/build-angular), and includes various plugins like karma-jasmine, karma-chrome-launcher, karma-jasmine-html-reporter, karma-coverage, and karma-angular-devkit. It also defines a client section for Jasmine with options for random execution and a specific seed.
- Terminal (Bottom):** Shows the command "PS C:\Users\dell\Desktop\pruebas\primero>" indicating the current working directory.

El package.json contiene todas las dependencias del proyecto, versiones

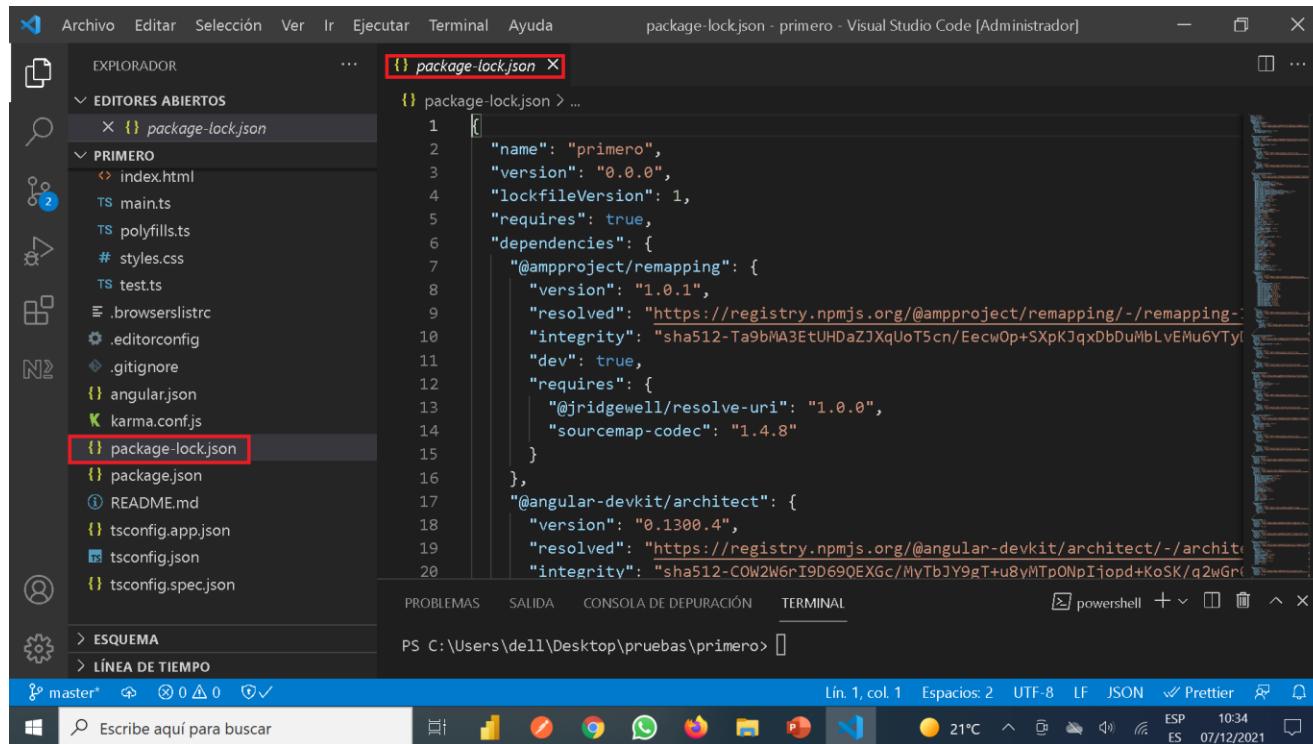


The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows the project structure with files like .angular, node_modules, src, .browserslistrc, .editorconfig, .gitignore, angular.json, karma.conf.js, package-lock.json, README.md, tsconfig.app.json, tsconfig.json, and tsconfig.spec.json.
- Editor (Right):** The file "package.json" is open in the editor. It contains the following JSON code:

```
1  {
2   "name": "primero",
3   "version": "0.0.0",
4   "scripts": {
5     "ng": "ng",
6     "start": "ng serve",
7     "build": "ng build",
8     "watch": "ng build --watch --configuration development",
9     "test": "ng test"
10 },
11 "private": true,
12 "dependencies": {
13   "@angular/animations": "~13.0.0",
14   "@angular/common": "~13.0.0",
15   "@angular/compiler": "~13.0.0",
16   "@angular/core": "~13.0.0",
17   "@angular/forms": "~13.0.0",
18   "@angular/platform-browser": "~13.0.0",
19   "@angular/platform-browser-dynamic": "~13.0.0",
20   "@angular/router": "~13.0.0",
21 }
```

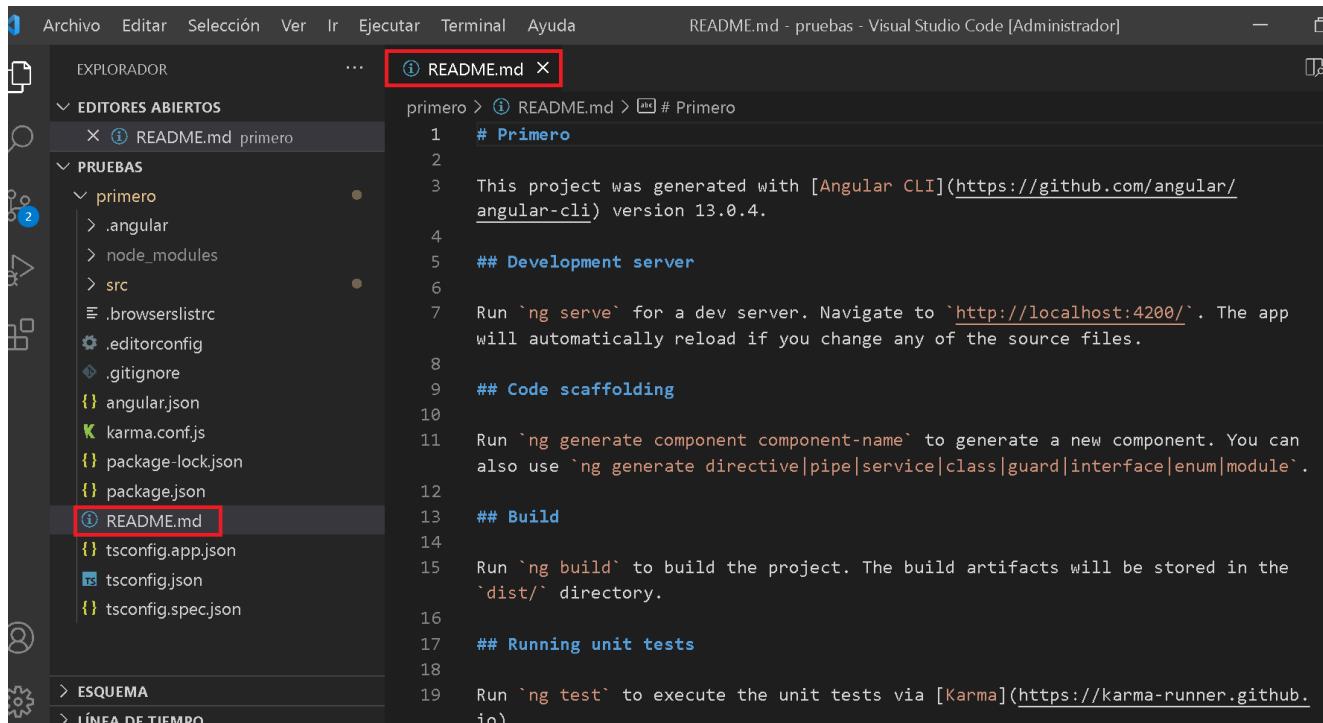
Es un archivo de registro automático de como se construye los módulos de node



The screenshot shows the Visual Studio Code interface with the title bar "package-lock.json - primero - Visual Studio Code [Administrador]". The left sidebar shows a project structure with files like index.html, main.ts, polyfills.ts, styles.css, test.ts, .browserslistrc, .editorconfig, .gitignore, angular.json, karma.conf.js, package.json, README.md, tsconfig.app.json, tsconfig.json, and tsconfig.spec.json. The "package-lock.json" file is selected and highlighted with a red box. The main editor area displays the JSON content of the package-lock.json file, which includes the project name, version, lockfile version, requires, dependencies, and resolved packages. The bottom status bar shows the path "PS C:\Users\dell\Desktop\pruebas\primero>" and the system status "Lín. 1, col. 1 Espacios: 2 UTF-8 LF JSON Prettier 21°C 10:34 ESP ES 07/12/2021".

```
{} package-lock.json > ...
{
  "name": "primero",
  "version": "0.0.0",
  "lockfileVersion": 1,
  "requires": true,
  "dependencies": {
    "@ampproject/remapping": {
      "version": "1.0.1",
      "resolved": "https://registry.npmjs.org/@ampproject/remapping/-/remapping-1.0.1.tgz?integrity=sha512-Ta9bMA3EtUHDaZJXqUoT5cn/EecwOp+SXpKJqxDbDuMbLvEMu6YTyI&cacheTime=1626405400000&lastModified=1626405400000",
      "integrity": "sha512-Ta9bMA3EtUHDaZJXqUoT5cn/EecwOp+SXpKJqxDbDuMbLvEMu6YTyI&cacheTime=1626405400000&lastModified=1626405400000",
      "dev": true,
      "requires": {
        "@jridgewell/resolve-uri": "1.0.0",
        "sourcemap-codec": "1.4.8"
      }
    },
    "@angular-devkit/architect": {
      "version": "0.1300.4",
      "resolved": "https://registry.npmjs.org/@angular-devkit/architect/-/architect-0.1300.4.tgz?integrity=sha512-COW2W6rI9D690EXGc/MyTbJY9gT+u8yMTpONpIiopd+KoSK/a2wGr&cacheTime=1626405400000&lastModified=1626405400000",
      "integrity": "sha512-COW2W6rI9D690EXGc/MyTbJY9gT+u8yMTpONpIiopd+KoSK/a2wGr&cacheTime=1626405400000&lastModified=1626405400000"
    }
  }
}
```

El archivo README.md es básicamente una documentación o guía para el proyecto



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

- Menu Bar:** Archivo, Editar, Selección, Ver, Ir, Ejecutar, Terminal, Ayuda.
- Title Bar:** README.md - pruebas - Visual Studio Code [Administrador].
- Left Sidebar (Explorador):**
 - EDITORES ABIERTOS:** README.md (primero)
 - PRUEBAS:** primero (contiene .angular, node_modules, src)
 - Archivos y carpetas: .browserslistrc, .editorconfig, .gitignore, angular.json, karma.confjs, package-lock.json, package.json, README.md (selecciónada y resaltada), tsconfig.app.json, tsconfig.json, tsconfig.spec.json.
- Right Panel (Editor):** Contenido del archivo README.md:

```
primero > README.md > # Primero
1 # Primero
2
3 This project was generated with [Angular CLI](https://github.com/angular/angular-cli) version 13.0.4.
4
5 ## Development server
6
7 Run `ng serve` for a dev server. Navigate to `http://localhost:4200/`. The app
will automatically reload if you change any of the source files.
8
9 ## Code scaffolding
10
11 Run `ng generate component component-name` to generate a new component. You can
also use `ng generate directive|pipe|service|class|guard|interface|enum|module`.
12
13 ## Build
14
15 Run `ng build` to build the project. The build artifacts will be stored in the
'dist/' directory.
16
17 ## Running unit tests
18
19 Run `ng test` to execute the unit tests via [Karma](https://karma-runner.github.io).
```

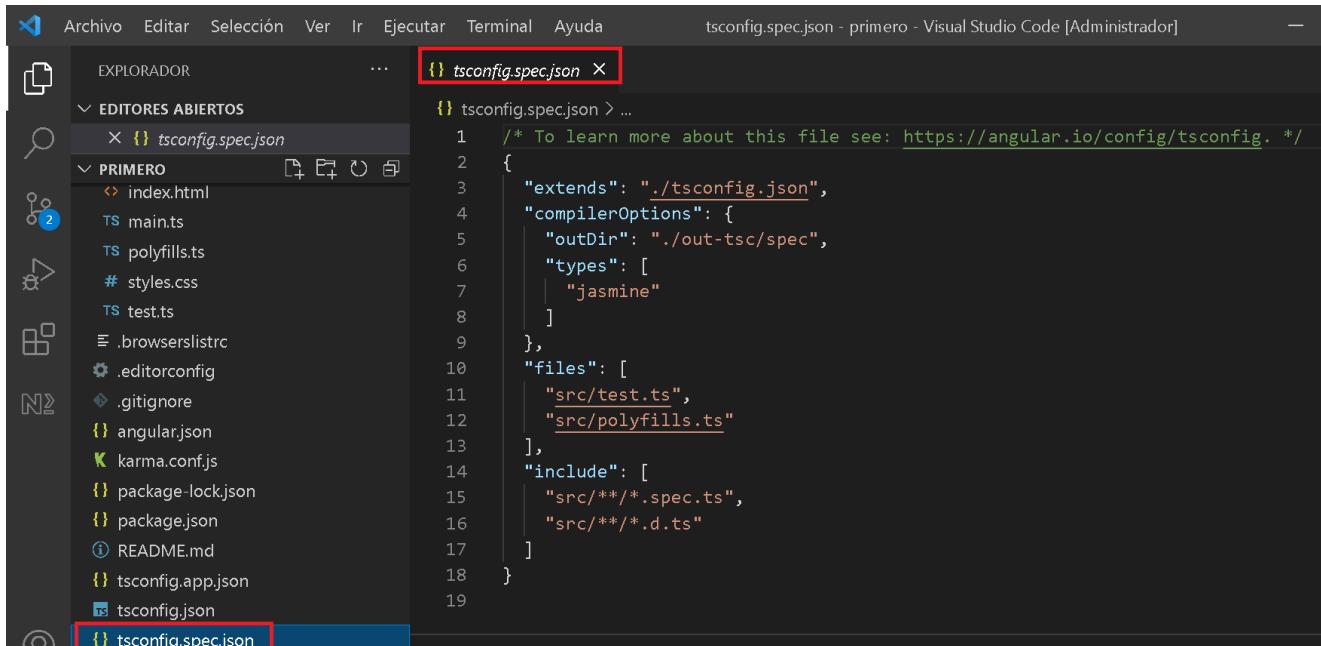
El archivo tsconfig nos ayuda con la alertas, variables, clases ,decoradores todo lo relacionado a como queremos que funcione typescript en nuestro proyecto

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

- File Explorer (Left):** Shows the project structure under "PRUEBAS". The "tsconfig.json" file is highlighted with a red box.
- Search Bar (Top):** Displays "tsconfig.json 1".
- Code Editor (Right):** Shows the content of the "tsconfig.json" file. The file starts with a comment about learning more from the Angular documentation and then defines a configuration object with various compiler options. Lines 1 through 23 are visible.

```
/* To learn more about this file see: https://angular.io/config/tsconfig. */
{
  "compileOnSave": false,
  "compilerOptions": {
    "baseUrl": "./",
    "outDir": "./dist/out-tsc",
    "forceConsistentCasingInFileNames": true,
    "strict": true,
    "noImplicitOverride": true,
    "noPropertyAccessFromIndexSignature": true,
    "noImplicitReturns": true,
    "noFallthroughCasesInSwitch": true,
    "sourceMap": true,
    "declaration": false,
    "downlevelIteration": true,
    "experimentalDecorators": true,
    "moduleResolution": "node",
    "importHelpers": true,
    "target": "es2017",
    "module": "es2020",
    "lib": [
      "es2020",
      "dom"
    ]
}
```

El archivo `tsconfig.spec.json`, llama al archivo `tsconfig.json` y añade mas configuraciones relacionadas a pruebas unitarias

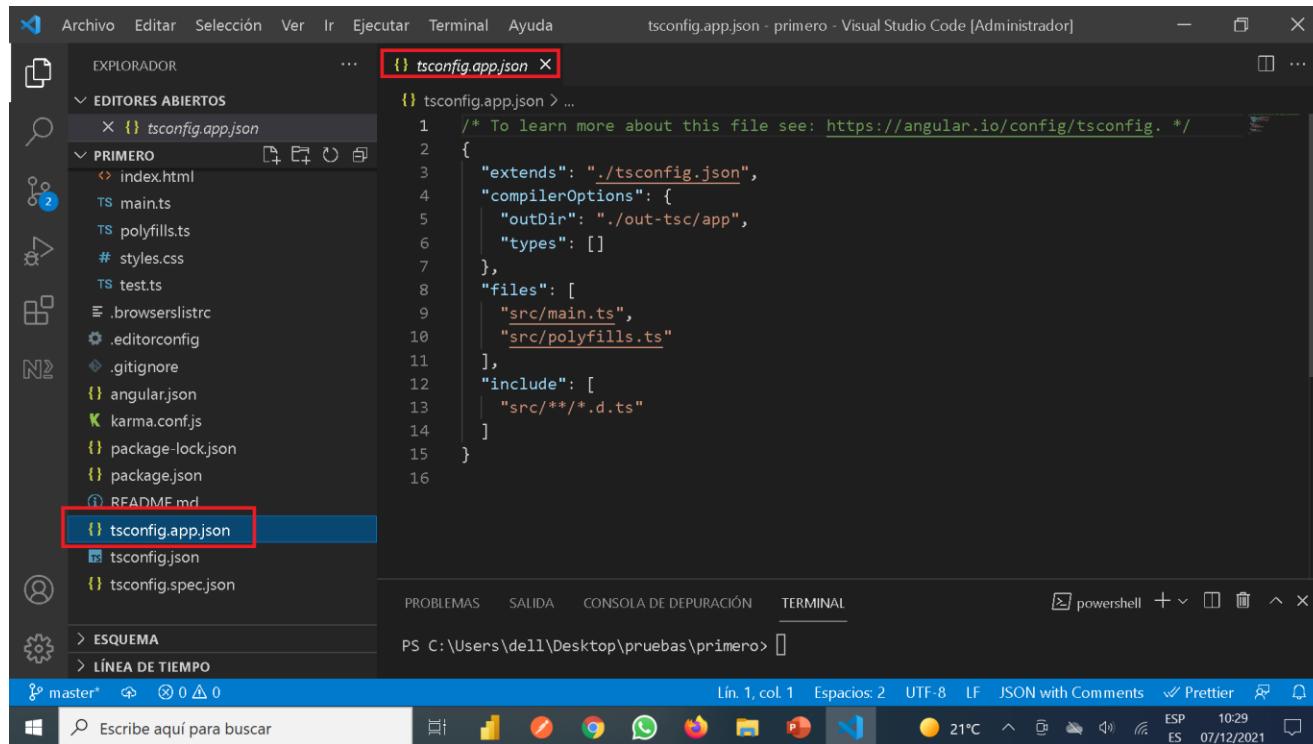


The screenshot shows the Visual Studio Code interface with the following details:

- Menu Bar:** Archivo, Editar, Selección, Ver, Ir, Ejecutar, Terminal, Ayuda.
- Title Bar:** tsconfig.spec.json - primero - Visual Studio Code [Administrador].
- Explorer View:** Shows a tree structure of files and folders. The "PRIMERO" folder is expanded, showing index.html, main.ts, polyfills.ts, styles.css, test.ts, .browserslistrc, .editorconfig, .gitignore, angular.json, karma.conf.js, package-lock.json, package.json, README.md, tsconfig.app.json, and tsconfig.json. The tsconfig.spec.json file is also listed in the sidebar.
- Editor View:** The tsconfig.spec.json file is open in the main editor area. The code content is as follows:

```
/* To learn more about this file see: https://angular.io/config/tsconfig. */
{
  "extends": "./tsconfig.json",
  "compilerOptions": {
    "outDir": "./out-tsc/spec",
    "types": [
      "jasmine"
    ],
    "files": [
      "src/test.ts",
      "src/polyfills.ts"
    ],
    "include": [
      "src/**/*.spec.ts",
      "src/**/*.d.ts"
    ]
  }
}
```

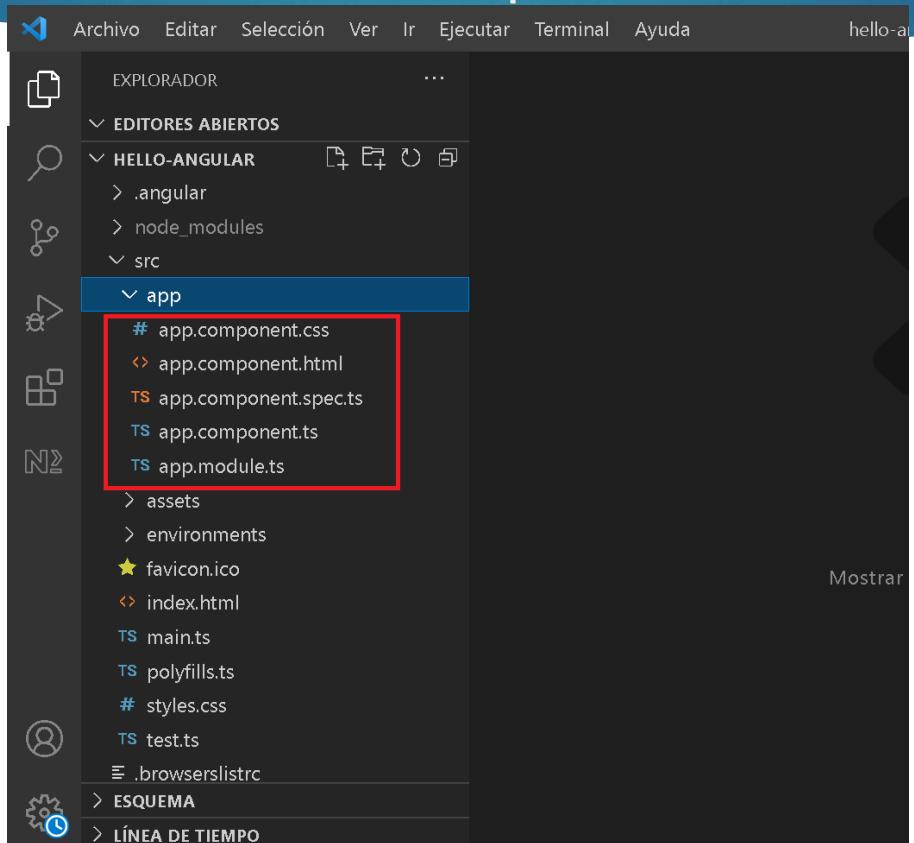
El archivo tsconfig.app.json llama al archivo tsconfig.json y agrega mas configuraciones respecto a la aplicación



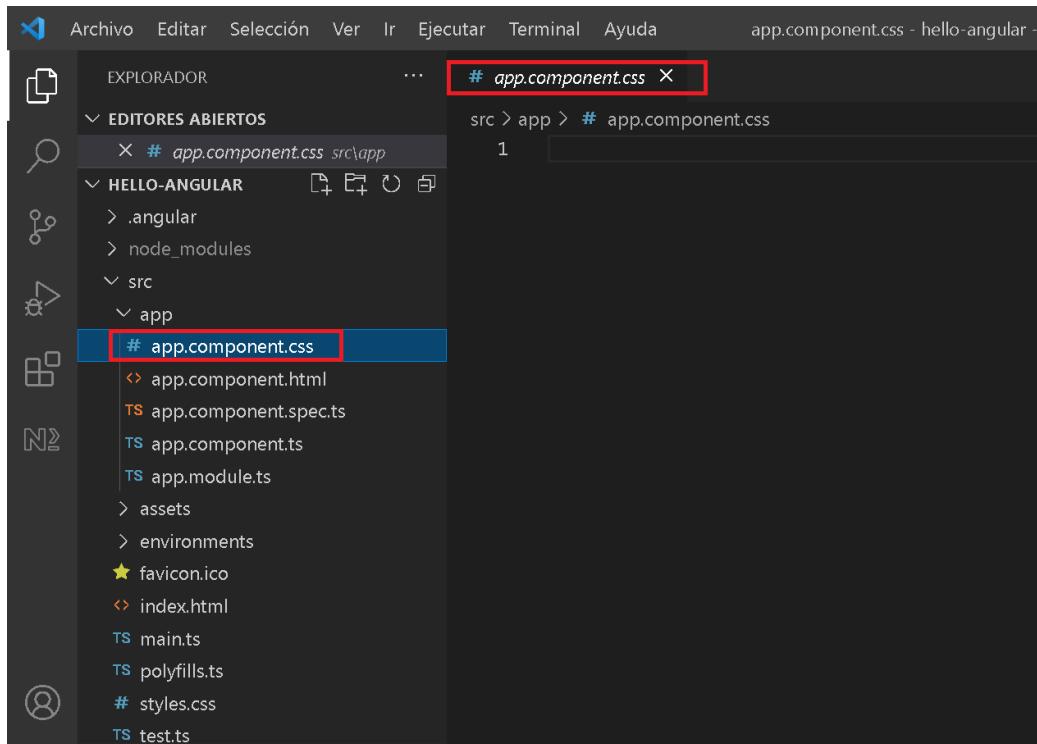
```
/* To learn more about this file see: https://angular.io/config/tsconfig. */
{
  "extends": "./tsconfig.json",
  "compilerOptions": {
    "outDir": "./out-tsc/app",
    "types": []
  },
  "files": [
    "src/main.ts",
    "src/polyfills.ts"
  ],
  "include": [
    "src/**/*.d.ts"
  ]
}
```

Explorando los archivos internos de src

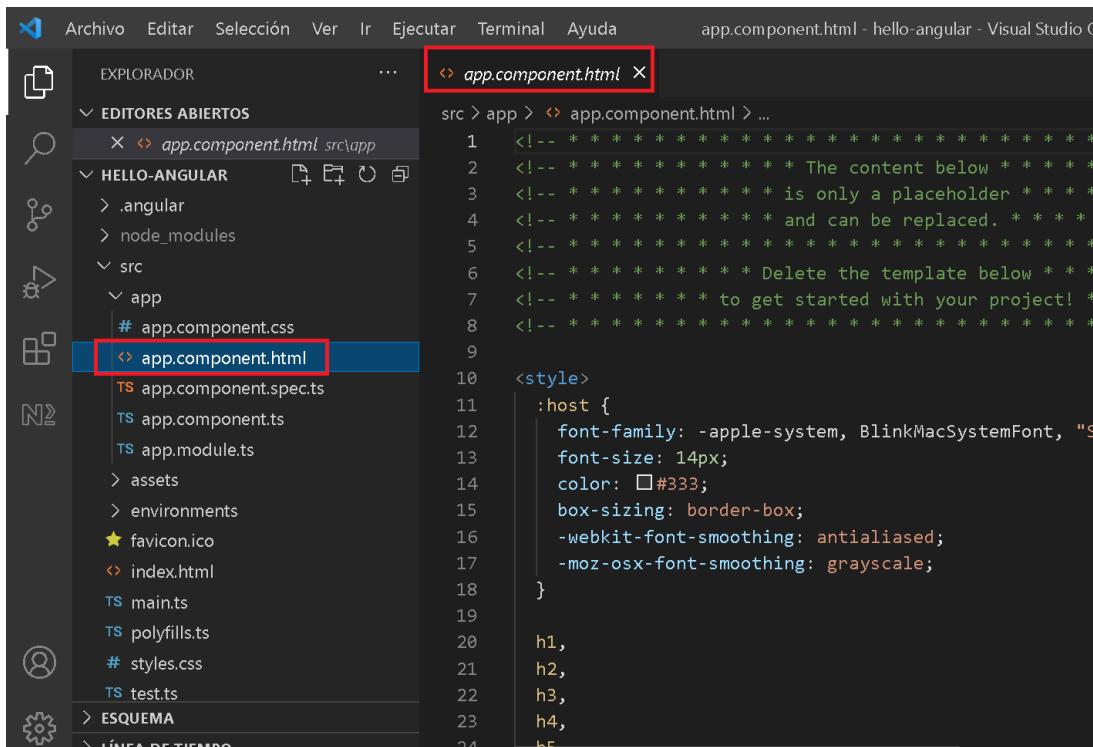
Directorio app, van los componentes de nuestra aplicación



App.component.css se encarga de todo el contenido css del componente



App.component.html contiene todo el html o vista del componente



App.component.spec.ts para realizar pruebas unitarias

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

- File Explorer (Left):** Shows the project structure under "HELLO-ANGULAR". The file "app.component.spec.ts" is highlighted with a red border.
- Editor (Center):** Displays the content of "app.component.spec.ts". The code is written in TypeScript and uses Jest for testing an Angular component. It includes imports for TestBed and AppComponent, a describe block for AppComponent, and two it blocks for testing the component's creation and title.
- Terminal (Top Right):** Shows the path "src > app > app.component.spec.ts".
- Status Bar (Bottom):** Shows "app.component.spec.ts - hello-angular - Visual Studio Code [Administrador]".

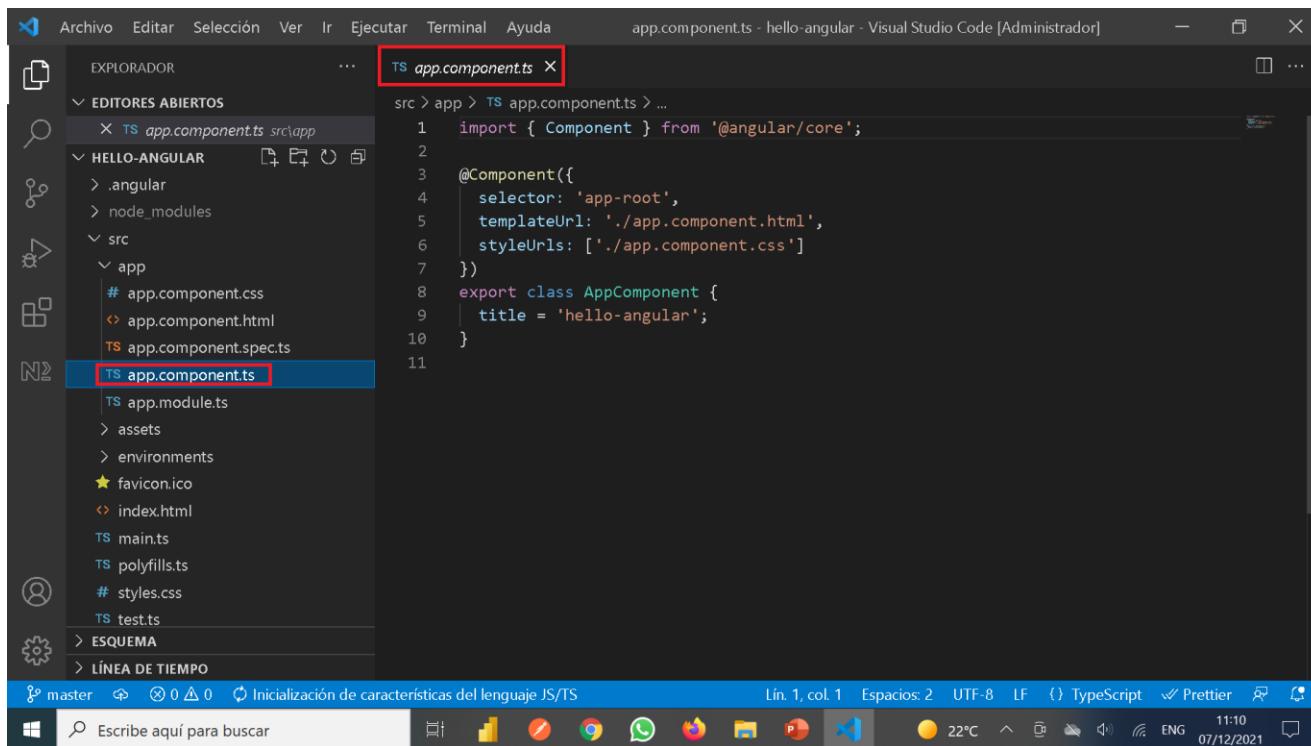
```
import { TestBed } from '@angular/core/testing';
import { AppComponent } from './app.component';

describe('AppComponent', () => {
  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [
        AppComponent
      ],
    }).compileComponents();
  });

  it('should create the app', () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app).toBeTruthy();
  });

  it(`should have as title 'hello-angular'`, () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app.title).toEqual('hello-angular');
  });
});
```

Es una clase que representa a una parte de nuestra aplicación web

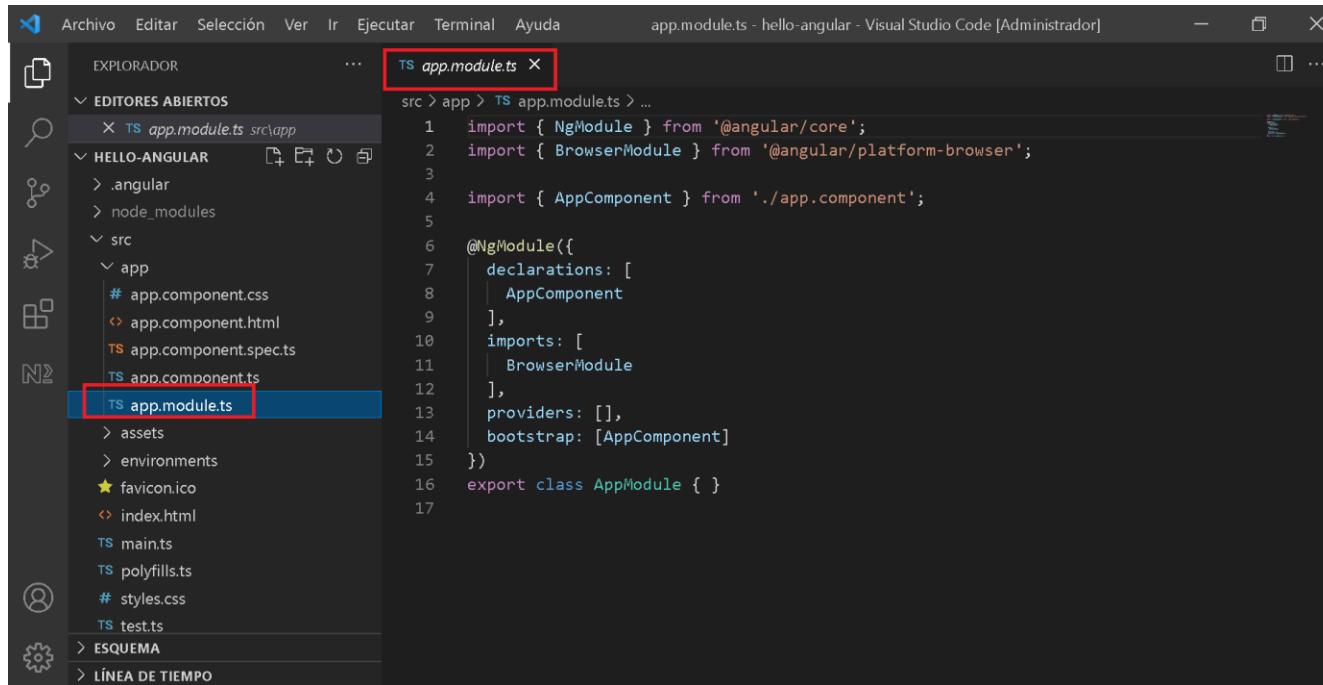


```
src > app > TS app.component.ts > ...
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   title = 'hello-angular';
10 }
11
```

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

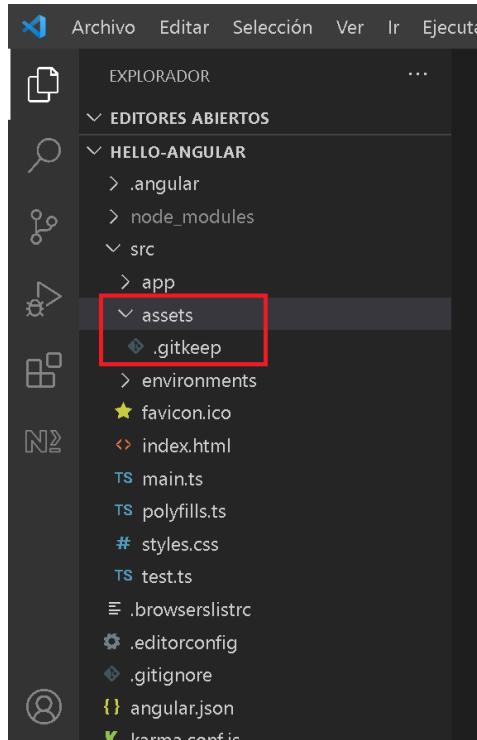
- File Explorer (Left):** Shows the project structure under "HELLO-ANGULAR".
 - src folder contains app, assets, environments, favicon.ico, index.html, main.ts, polyfills.ts, styles.css, and test.ts.
 - app folder contains app.component.css, app.component.html, and app.component.spec.ts.
 - app.component.ts is currently selected and highlighted in blue.
- Editor (Center):** Displays the code for app.component.ts, which defines an Angular component named AppComponent.
- Bottom Status Bar:** Shows the file path "app.component.ts - hello-angular - Visual Studio Code [Administrador]", line 1, column 1, and other status information like "Líneas: 11" and "Espacios: 2".
- Taskbar (Bottom):** Shows various application icons and system status indicators.

app.modules.ts es como un repositorio donde se registra nuestro componente y todo lo que podríamos utilizar en el

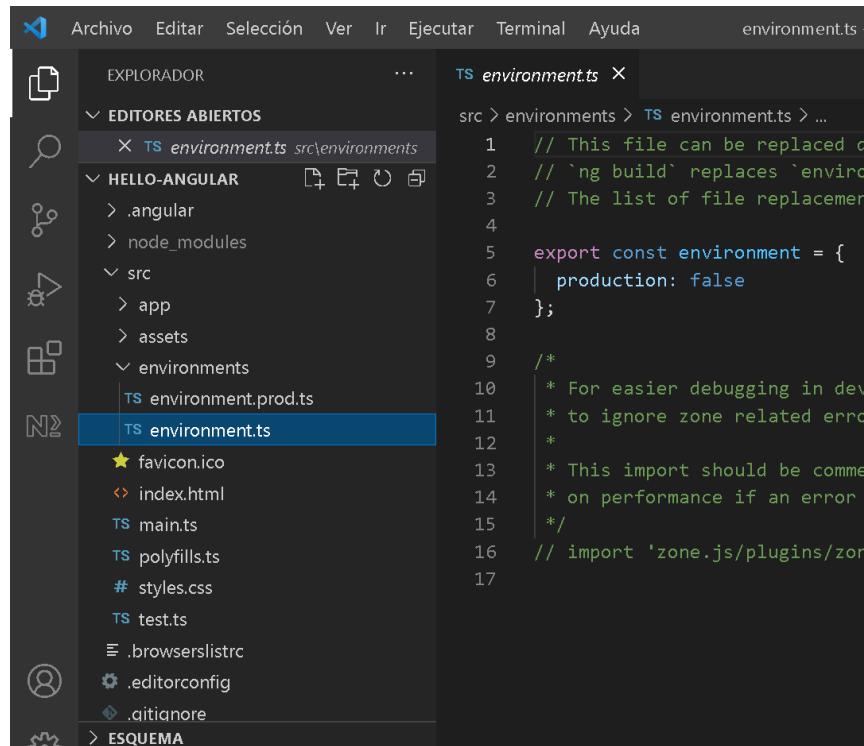


```
src > app > TS app.module.ts > ...
1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3
4 import { AppComponent } from './app.component';
5
6 @NgModule({
7   declarations: [
8     AppComponent
9   ],
10   imports: [
11     BrowserModule
12   ],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

En el directorio assets se guarda todo el contenido estático de nuestra aplicación



El directorio environments contiene los archivos de configuración de ambientes de desarrollo y producción de la aplicación



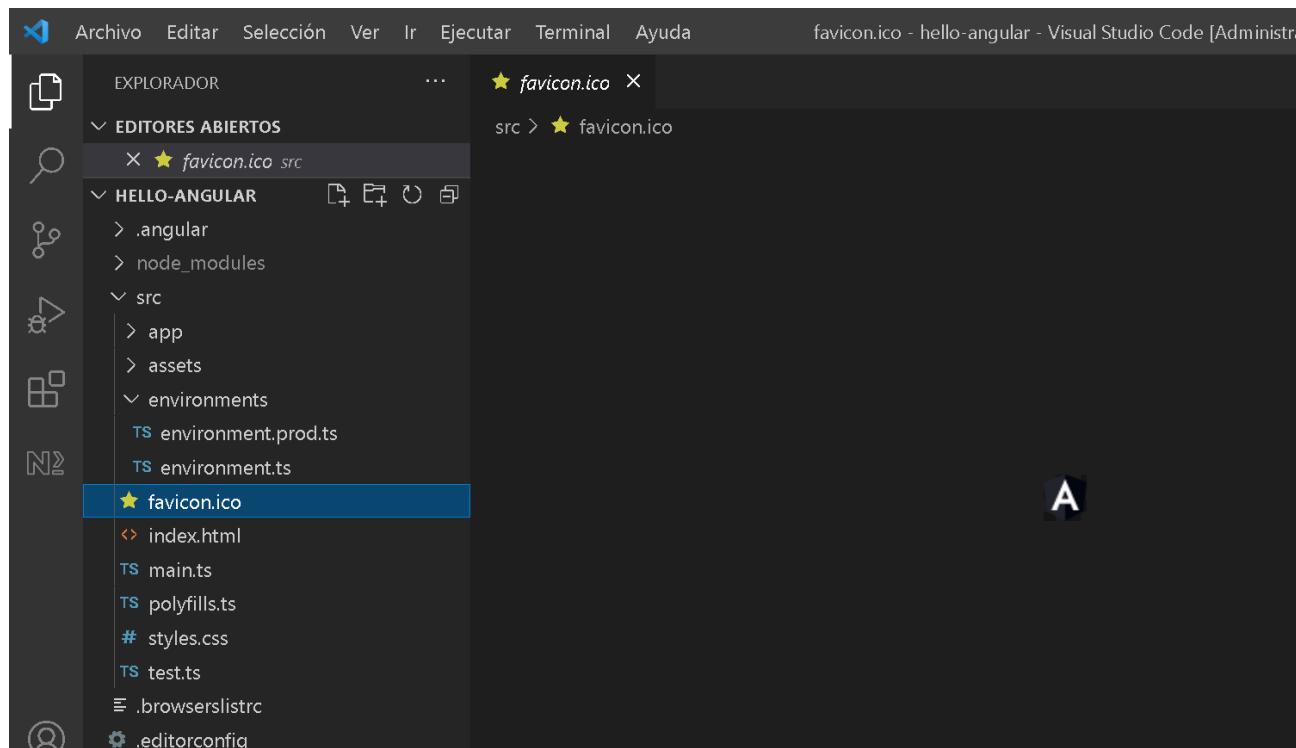
```
// This file can be replaced during the build process.
// `ng build` replaces `environment.ts` with `environment.prod.ts`.
// The list of file replacements is defined in `environments/environment.prod.ts`.

export const environment = {
  production: false
};

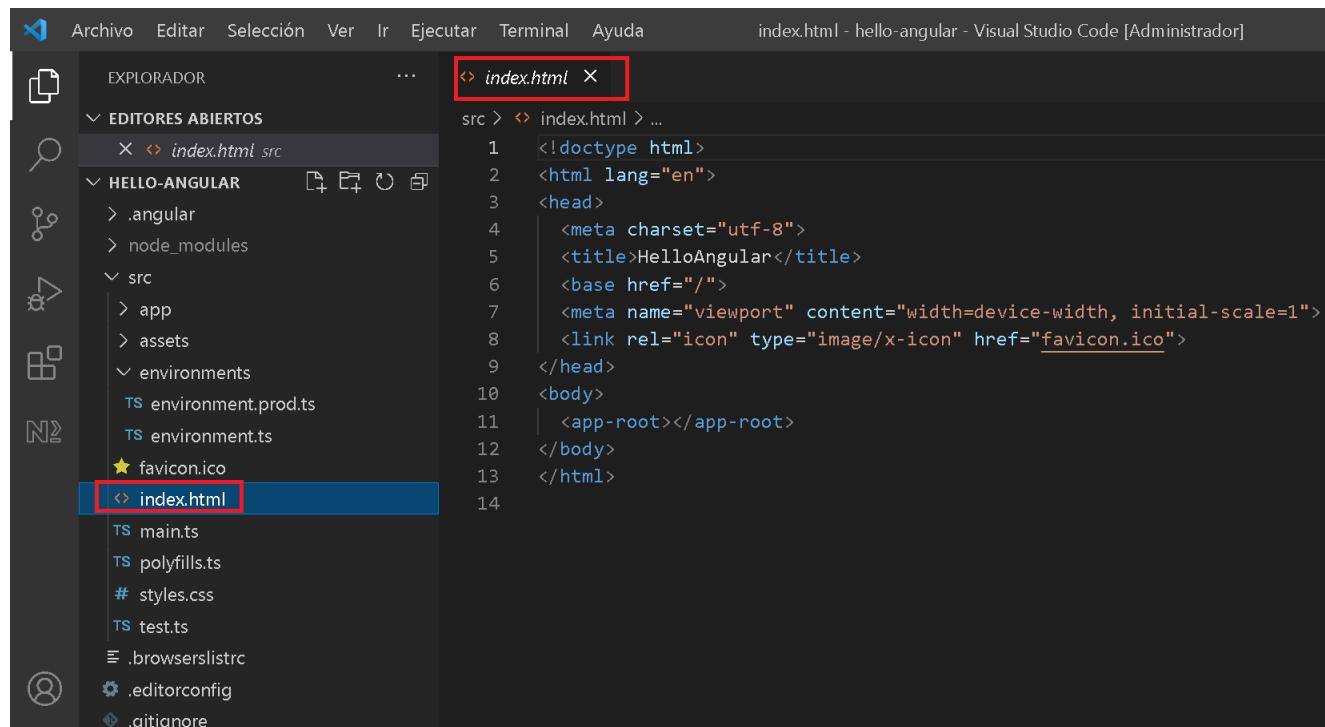
/*
 * For easier debugging in development mode, ignore errors in some specific dev-only packages.
 * See https://ionicframework.com/docs/environment-variables#development-only-error-handling
 */
/* This import should be commented out for performance.
 * See https://zone.js.io/plugins.html#zone-node
 */
// import 'zone.js/plugins/zone-node';


```

El archivo favicon.ico es la imagen que nos muestra por default en el tab de los navegadores



El archivo index.html es la página principal de la aplicación

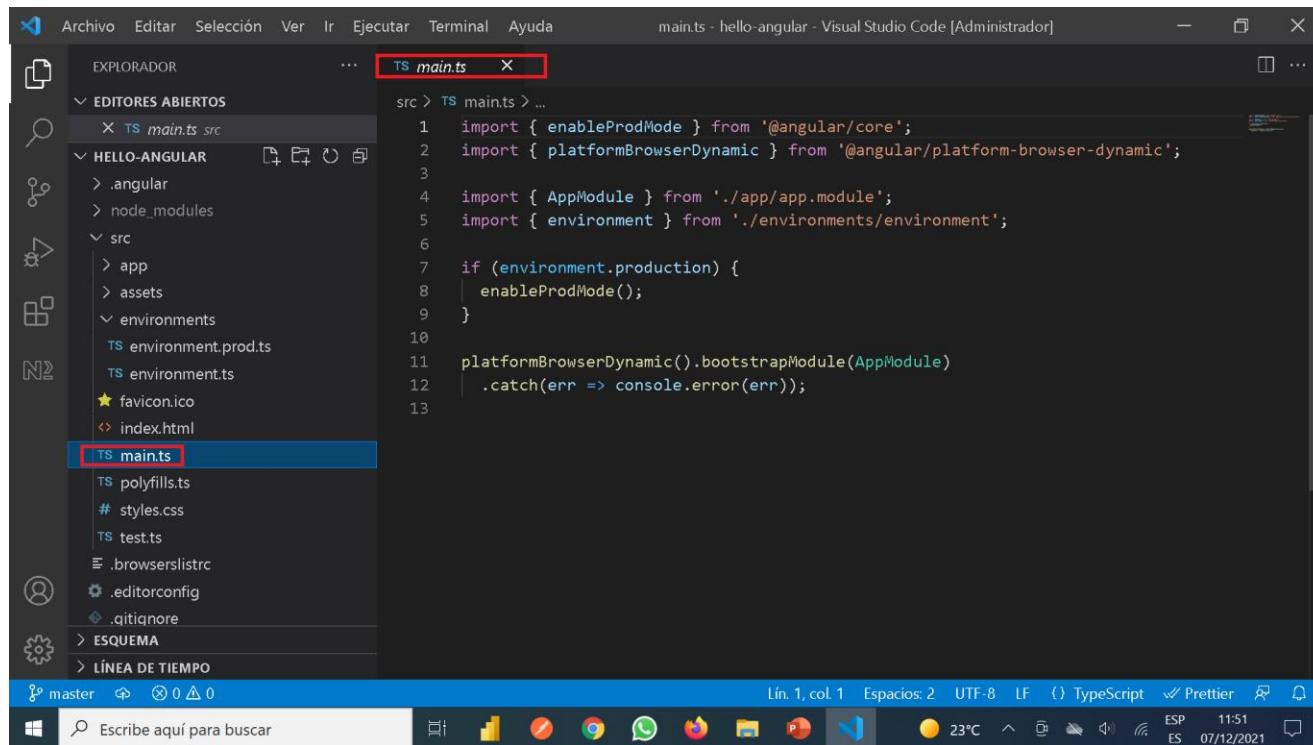


The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows the project structure for "HELLO-ANGULAR". The "src" folder contains "app", "assets", "environments", "environment.prod.ts", "environment.ts", and "favicon.ico".
- Editor Area (Right):** The file "index.html" is open and displayed. It contains the following code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>HelloAngular</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```
- Status Bar:** Shows "index.html - hello-angular - Visual Studio Code [Administrador]".

El archivo main.ts es la clase principal que levanta y arranca el app.module



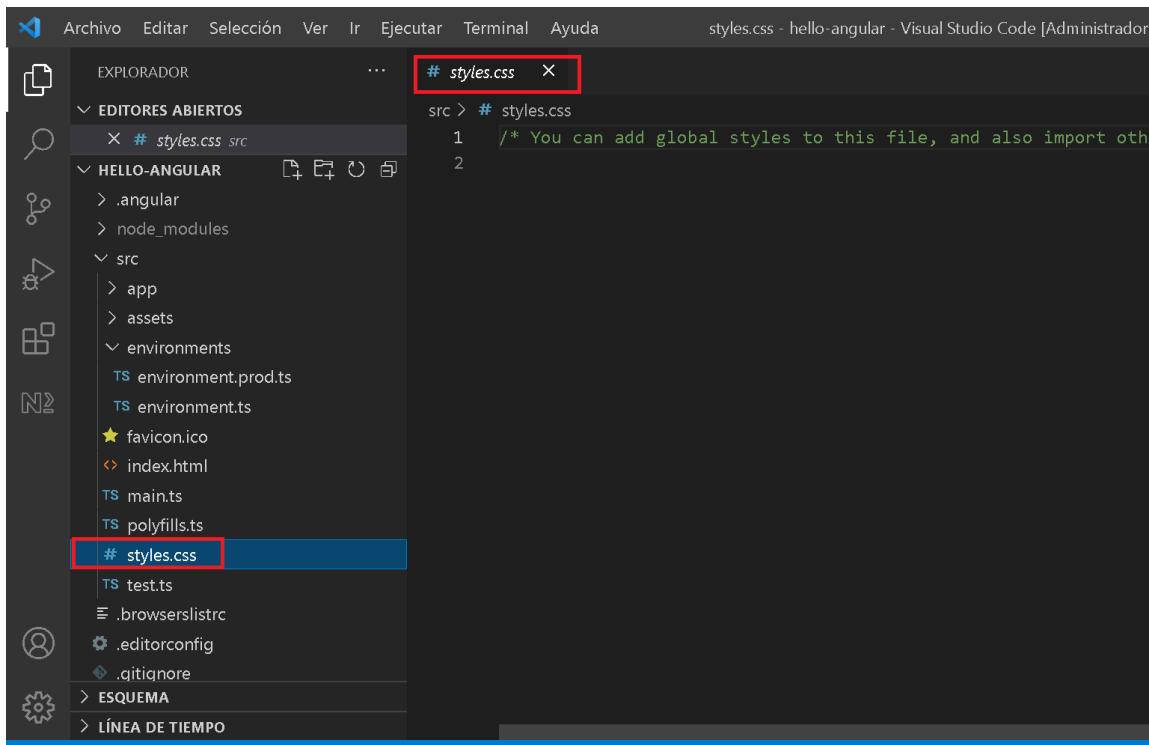
The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows the project structure for "HELLO-ANGULAR". The "src" folder contains "app", "assets", and "environments". Inside "environments", there are "environment.prod.ts" and "environment.ts". Other files shown include "main.ts", "polyfills.ts", "# styles.css", "test.ts", ".browserslistrc", ".editorconfig", ".gitignore", "ESQUEMA", and "LÍNEA DE TIEMPO".
- Editor (Center):** The file "main.ts" is open. The code is as follows:

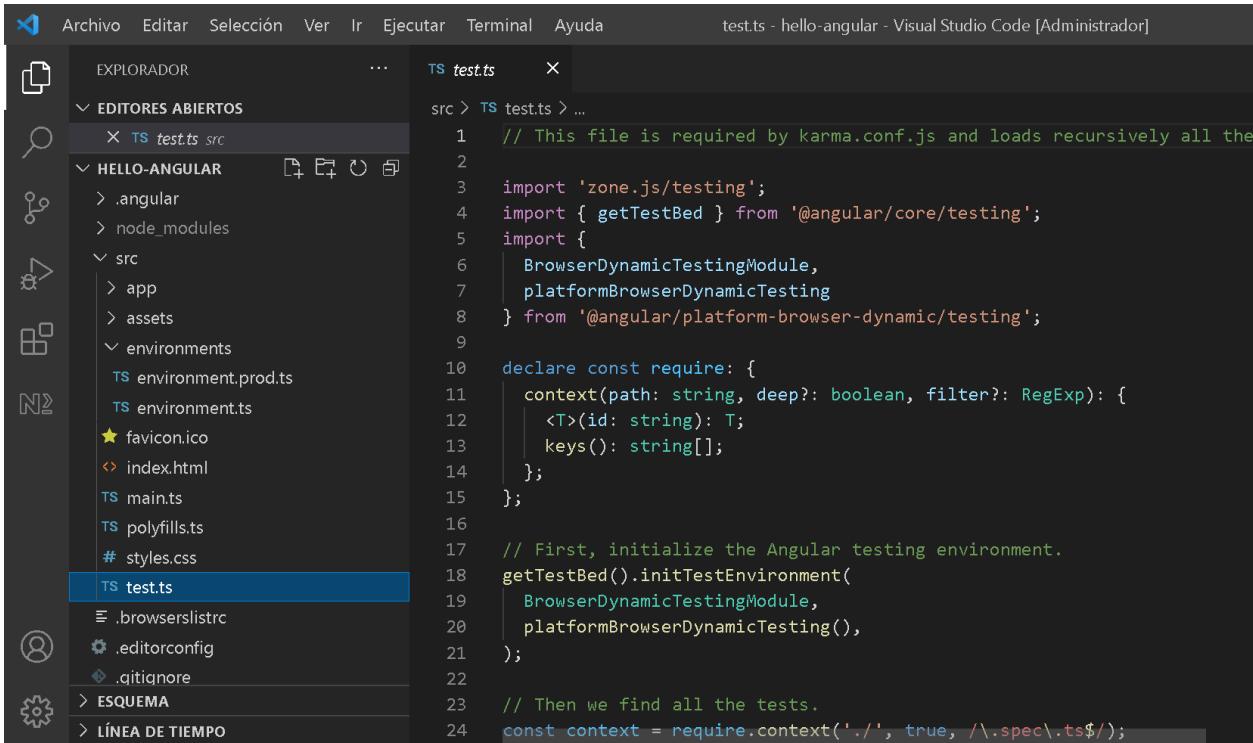
```
1 import { enableProdMode } from '@angular/core';
2 import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
3
4 import { AppModule } from './app/app.module';
5 import { environment } from './environments/environment';
6
7 if (environment.production) {
8   enableProdMode();
9 }
10
11 platformBrowserDynamic().bootstrapModule(AppModule)
12   .catch(err => console.error(err));
```

- Bottom Status Bar:** Shows "Lín. 1, col. 1 Espacios: 2 UTF-8 LF () TypeScript ✓ Prettier ⚡ 23°C 11:51 ESP ES 07/12/2021".

El archivo style.css para aplicar estilos globales en la aplicación



El archivo test.ts para configuración de pruebas unitarias



```
src > TS test.ts > ...
1 // This file is required by karma.conf.js and loads recursively all the
2
3 import 'zone.js/testing';
4 import { getTestBed } from '@angular/core/testing';
5 import {
6   BrowserDynamicTestingModule,
7   platformBrowserDynamicTesting
8 } from '@angular/platform-browser-dynamic/testing';
9
10 declare const require: {
11   context(path: string, deep?: boolean, filter?: RegExp): {
12     <T>(id: string): T;
13     keys(): string[];
14   };
15 };
16
17 // First, initialize the Angular testing environment.
18 getTestBed().initTestEnvironment(
19   BrowserDynamicTestingModule,
20   platformBrowserDynamicTesting(),
21 );
22
23 // Then we find all the tests.
24 const context = require.context('./', true, /\.spec\.ts$/);
```

**Gracias por
vuestra
participación**



¡Seguimos en contacto!

www.iconotc.com

