



www.singulartech.ai



010010011011
011101010010

Angular

Angular es una plataforma muy utilizada de desarrollo de aplicaciones Single Page eficientes y sofisticadas que está soportada por la comunidad de desarrolladores de Google.



Contenido del Curso

1. Introducción

1. Algo de Teoría
2. Algo de Historia
3. Angular Essentials

2. Escenario

1. Herramientas de Desarrollo
2. Preparación del Entorno
3. Creación del Proyecto

3. Modules

1. Modules & Components
2. Generación de Modules

4. Components

1. Templates
2. Syntax

3. Data Binding

4. Pipes
5. Directives

5. Routing

1. RouterModule

6. Services

1. Dependence Injection
2. Generación de Services
3. HttpModule

7. Forms

8. Seguridad

1. Auth0
2. Guards
3. Interceptors

1. Herramientas de Desarrollo



Visual Studio

Link de Descarga:

<https://visualstudio.microsoft.com/downloads/>



Visual Studio Code

Link de Descarga:

<https://code.visualstudio.com/download/>



NodeJS

Link de Descarga:

<https://nodejs.org/en/download/>



Angular Cli

Link de Descarga:

<https://angular.io/cli>

The background is a dark blue gradient with a pattern of small, light blue dots. Overlaid on this are several glowing, magenta-pink lines that form a complex, swirling pattern. A bright yellow and orange light source is visible at the bottom right, casting a glow. The text "1. INTRODUCCIÓN" is centered in a bold, white, sans-serif font.

1. INTRODUCCIÓN

1. Algo de Teoría

Angular es una plataforma de desarrollo de Google, construida en TypeScript.

Un framework basado en componentes para crear aplicaciones web escalables.

Una colección de librerías bien integradas que cubren una amplia variedad de funciones, incluido el enrutamiento, la gestión de formularios, la comunicación cliente-servidor y más.

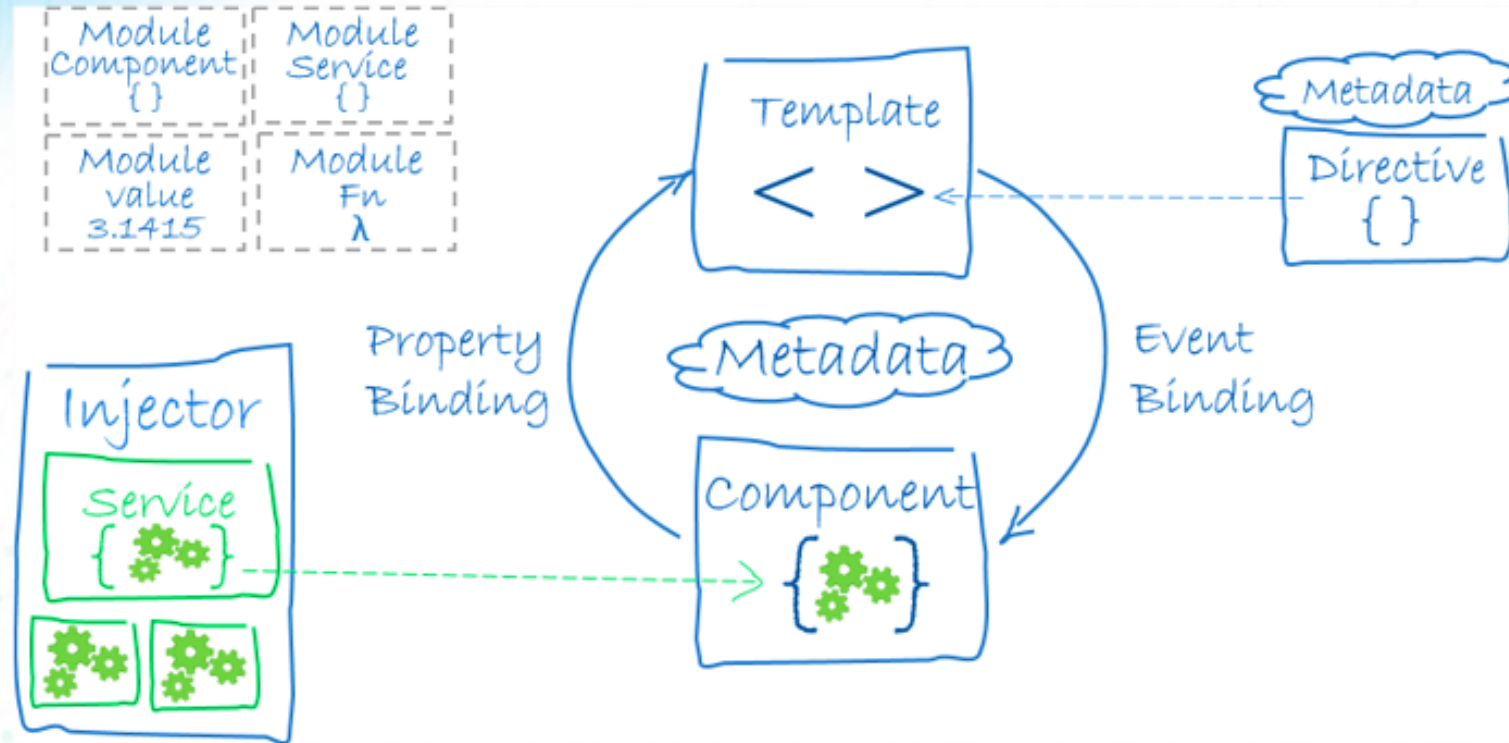
Un conjunto de herramientas de desarrollo para ayudar a desarrollar, compilar, probar y actualizar el código.

2. Algo de Historia

Angular es una plataforma de desarrollo de Google, que fue lanzada en 2010 de la mano de uno de sus colaboradores llamado **Miško Hevery**, quien trabajaba en un proyecto alternativo para construir aplicaciones web de manera más fácil:



3. Angular Essentials



Juntos, un *Component* y un *Template* definen un Angular View.

- Un *Decorator* declarado en un *Component* agrega la metadata, incluyendo una referencia asociada al *Template*.
- Directives y Binding Markup en la plantilla del *Component* modifica las vistas basadas en data y lógica.

La inyección de dependencias provee servicios a un *Component*. Como por ejemplo el Servicio de Routing que permite definir la navegación entre vistas.

3. Angular Essentials

Component

```
1. import { Component } from '@angular/core';
2.
3. @Component ({
4.   selector: 'hello-world-bindings',
5.   templateUrl: './hello-world-bindings.component.html'
6. })
7. export class HelloWorldBindingsComponent {
8.   fontColor = 'blue';
9.   sayHelloId = 1;
10.  canClick = false;
11.  message = 'Hello, World';
12.
13.  sayMessage() {
14.    alert(this.message);
15.  }
16.
17. }
```

Template

```
1. <button
2.   type="button"
3.   [disabled]="canClick"
4.   (click)="sayMessage()">
5.   Trigger alert message
6. </button>
7. <p
8.   [id]="sayHelloId"
9.   [style.color]="fontColor">
10.   You can set my color in the component!
11. </p>
12. <p>My color is {{ fontColor }}</p>
```



2. ESCENARIO

Escenario

Eres un desarrollador web y estás a cargo del mantenimiento de las aplicaciones web de tu compañía

Para este ejercicio, necesitas crear una aplicación web en Angular para consumir apis de teléfonos.

Necesitas aplicar los conceptos básicos de Angular y construir la aplicación de manera que sea extensible e intuitiva.

Objetivo

Construir una aplicación de Angular con las características básicas cómo Routing, Modularización, Formularios, Consumo de Apis y Seguridad.

2. Preparación del Entorno

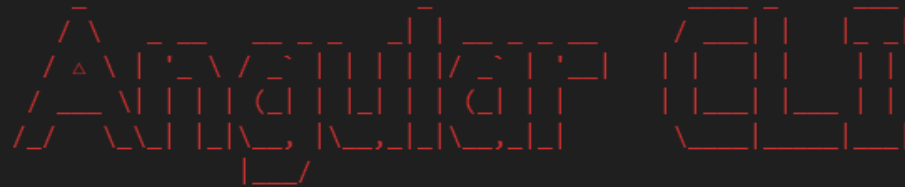
<https://nodejs.org/en/download/>

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

```
ng --version
```

```
Your global Angular CLI version (13.3.3) is greater than your local version (13.1.2).
```

```
To disable this warning use "ng config -g cli.warnings.versionMismatch false".
```



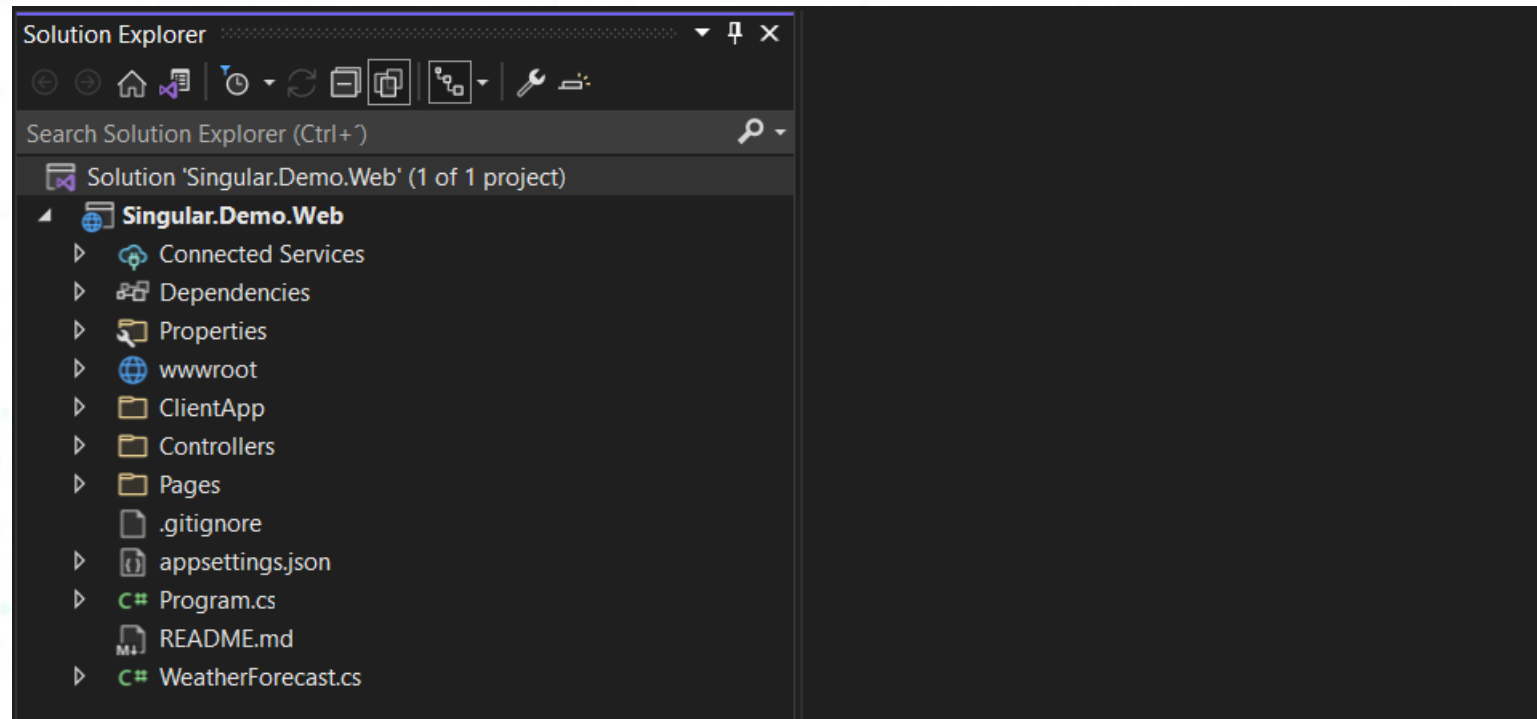
```
Angular CLI: 13.1.2
```

```
Node: 16.13.2
```

```
Package Manager: npm 8.3.2
```


3. Creación del Proyecto

```
dotnet new angular -o Singular.Demo.Web -f net6.0
```



Package.json

```
package.json  X
Schema: https://json.schemastore.org/package.json
1  {
2    "name": "singular.demo.web",
3    "version": "0.0.0",
4    "scripts": {
5      "ng": "ng",
6      "prestart": "node aspnetcore-https",
7      "start": "run-script-os",
8      "start:windows": "ng serve --port 44411 --ssl --ssl-cert %APPDATA%\\ASP
9      "start:default": "ng serve --port 44411 --ssl --ssl-cert $HOME/.aspnet/I
10     "build": "ng build",
11     "build:ssr": "ng run Singular.Demo.Web:server:dev",
12     "watch": "ng build --watch --configuration development",
13     "test": "ng test"
14   },
15   "private": true,
16   "dependencies": {
35  "devDependencies": {
50  "optionalDependencies": {}
51  }
52 }
```

angular.json

```
angular.json  ✎ ✕  
Schema: ./node_modules/@angular/cli/lib/config/schema.json  
1  {  
2    "$schema": "./node_modules/@angular/cli/lib/config/schema.json",  
3    "cli": {  
4      "analytics": "204604bb-81d5-408f-a390-b0ffa1a5a16a"  
5    },  
6    "version": 1,  
7    "newProjectRoot": "projects",  
8    "projects": {  
9      "Singular.Demo.Web": {  
10       "projectType": "application",  
11       "schematics": {  
12         "@schematics/angular:application": {  
13           "strict": true  
14         }  
15       },  
16       "root": "",  
17       "sourceRoot": "src",  
18       "prefix": "app",  
19       "architect": {  
20         "build": {},  
21         "serve": {},  
22         "extract-ill8n": {},  
23         "test": {},  
24         "server": {}  
25       }  
26     },  
27     "defaultProject": "Singular.Demo.Web"  
28   }  
29 }
```

An abstract graphic featuring a dark blue background with a pattern of small, light blue dots. Overlaid on this are several glowing, curved lines in shades of pink and magenta. These lines form a complex, swirling pattern that frames the central text. There are also several small, bright blue and yellow dots scattered throughout the lower half of the image, adding to the cosmic or digital aesthetic.

3. MODULES

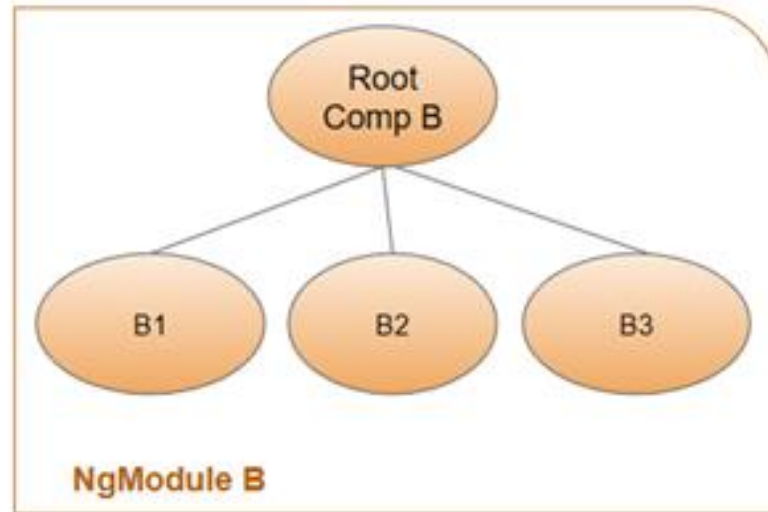
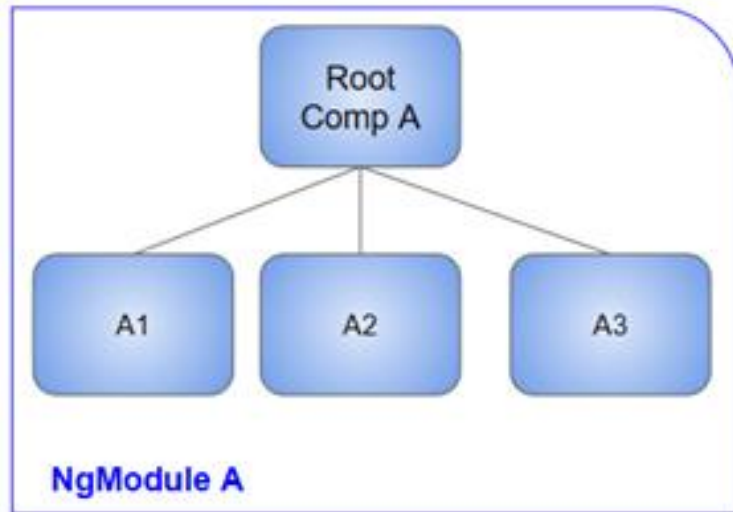
Modules

Las aplicaciones de Angular son modulares, los módulos de Angular se administran bajo el sistema modular de Angular llamado *NgModule*, para declarar un módulo de Angular se agrega un “*Decorator*” llamado **@NgModule()**, el cual es una función que inicializa las siguientes propiedades del módulo.

PROPIEDAD	DETALLES
declarations	Los Components, Directives y Pipes que pertenecen al módulo.
exports	El sub set de declaraciones que deben ser visibles o pueden ser utilizadas en las plantillas de otros módulos..
imports	Otros módulos cuyas clases exportadas son necesarias para ser utilizadas en los Components o plantillas de este módulo.
providers	Lista de servicios del modulo que contribuyen a la lista global de servicios. Los servicios declarados en esta sección puede ser utilizados desde cualquier parte de la aplicación. (También se pueden especificar a nivel de componente)
bootstrap	La vista principal de la aplicación, llamado root component, que contiene todas las demás vistas de la aplicación. Sólo el modulo root debe contener esta declaración.

1. Modules & Components

Los módulos de Angular proveen un contexto de compilación para sus componentes. El modulo *root* siempre tiene un componente root que es creado al momento de iniciar la aplicación pero cualquier módulo puede incluir componentes adicionales que pueden ser cargados a través del Router o creados a través de declaraciones en *Templates*. Los componentes que pertenecen a un módulo comparten un contexto de compilación.



2. Generación de Modules

Se puede agregar un módulo de Angular al módulo principal usando el cliente de Angular

```
ng generate module phones --flat --module=app
```

PARÁMETRO	DETALLES
--flat	Crea el archivo en src/app en lugar de su propio folder.
--module=app	Se usa para registrar el import del módulo en el archivo AppModule que contiene el módulo principal

An abstract graphic featuring a dark blue background with a fine grid of small dots. Two glowing, overlapping loops of pink and magenta light dominate the center. Several small, bright blue and white dots are scattered along the lower part of the loops, resembling a comet's trail or a celestial path.

4. COMPONENTS

Components

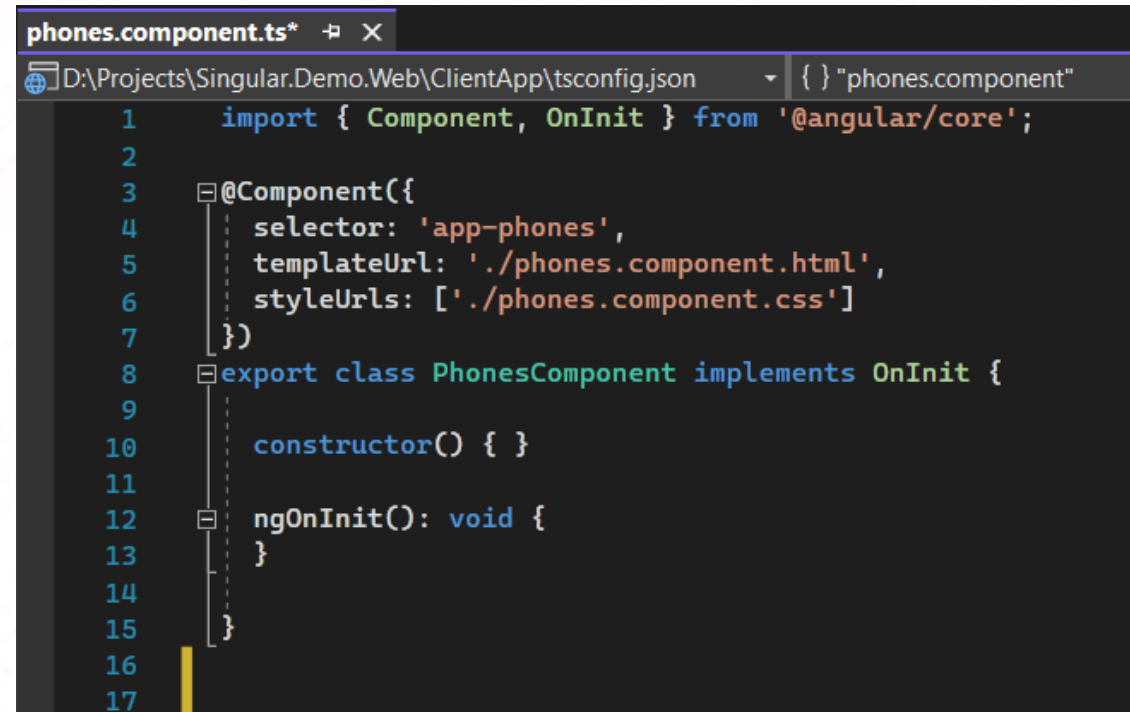
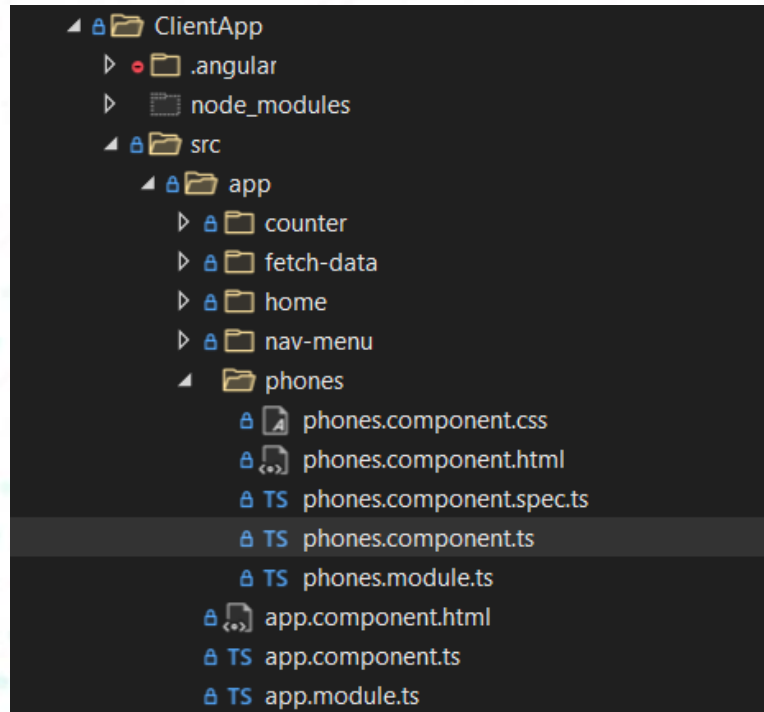
Un componente controla una sección de la pantalla denominada View. Una vista está compuesta por un componente y una plantilla (*Template*). Un componente se define agregando a una clase el *decorator* **@Component()** que es una función que inicializa las propiedades o metadatos, entre las principales propiedad están la plantilla y el selector. Se define la lógica del componente en la clase y se interactúa con la plantilla mediante funciones y propiedades.

OPCIÓN DE CONFIGURACIÓN	DETALLES
selector	Un selector CSS que especifica el tag con el que se puede crear e insertar una instancia del componente en la plantilla de cualquier otro componente en el que se quiera reutilizar.
templateUrl	La ruta relativa del archivo que contiene la plantilla HTML del componente. Alternativamente, se pueden especificar plantillas HTML directamente sin necesidad de un archivo externo mediante la propiedad <i>template</i> . Esta plantilla define la vista del componente.
providers	Un arreglo de servicios que el componente requiere.

Components

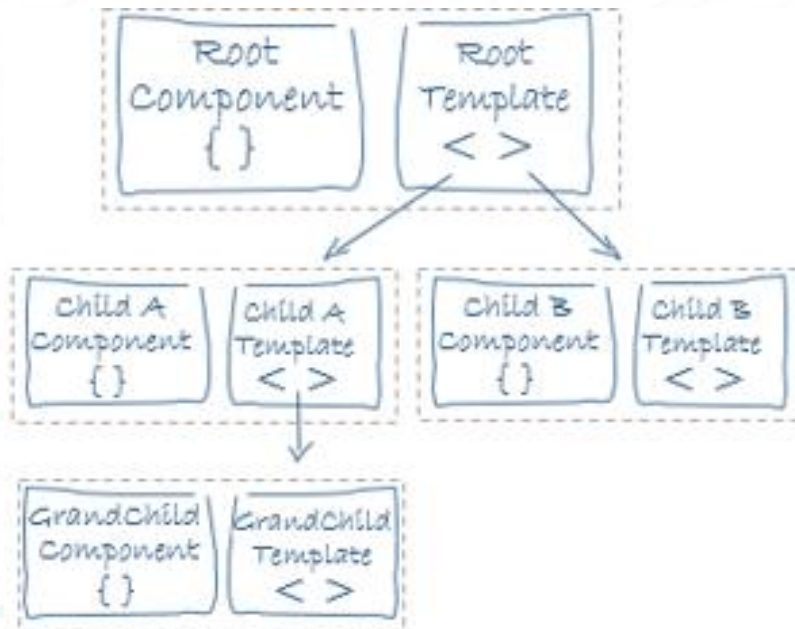
Se puede agregar rutas navegación al módulo principal usando el cliente de Angular

```
ng generate component phones-list --module=phones
```



1. Templates

La vista de un *Component* se define mediante un *Template*.



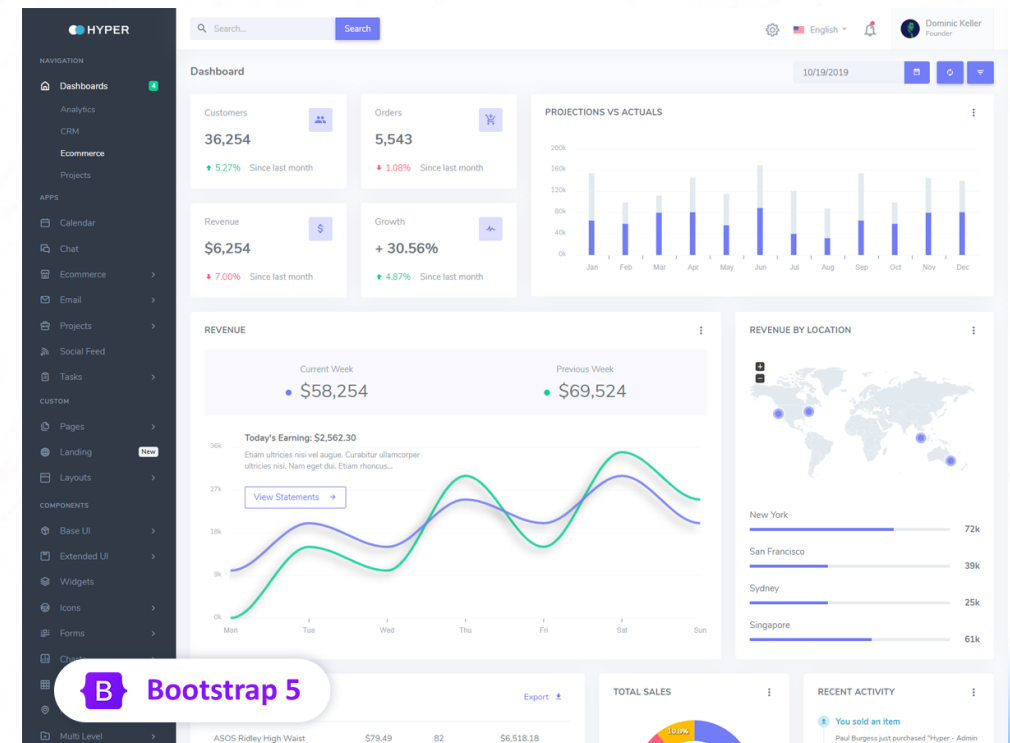
Un *template* es código HTML que le especifica al motor de Angular como se debe renderizar el componente.

Las vistas, que están definidas por un *template* y un *component*, se ordenan jerárquicamente, lo que permite modificar, mostrar u ocultar secciones enteras en una página. El componente también puede definir una jerarquía de vistas que puede contener muchos más componentes embebidos.

2. Templates Design

Cómo referencia de diseño, la plantilla de proyecto de Visual Studio se utiliza Bootstrap como framework por defecto para el diseño de Vistas. Bootstrap es el framework más popular en cuanto a diseño ya sea individual o como base para otros frameworks y soporta también diseño responsivo.

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
Max container width	None (auto)	540px	720px	960px	1140px
Class prefix	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-
# of columns	12				
Gutter width	30px (15px on each side of a column)				
Nestable	Yes				
Column ordering	Yes				



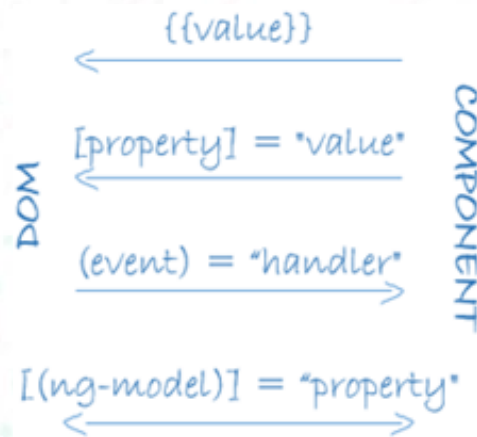
2. Syntax

El código de un *template* es similar al código HTML, excepto porque tiene incluida sintaxis Angular.

ELEMENTO	DETALLES
Elementos HTML	Elementos comunes HTML como <code><div></code> , <code><input></code> , <code></code> , etc.
Built-in Directives	Directivas de Angular de control de flujo por ejemplo *ngIf , *ngFor , *ngSwitch . Las directivas permiten mostrar, ocultar, o iterar elementos para construir colecciones de elementos html u otras vistas.
Attribute Directives	Directivas angular diseñadas para hacer referencia a las propiedades por ejemplo: NgClass: Para agregar o remover clases CSS al elemento. NgStyle: Para agregar o remover estilos HTML al elemento. NgModel: Para especificar la propiedad del componente que se enlaza al valor de un input.
Structural Directives	Directivas personalizadas para control de flujo similares a las directivas Built-in
Data Binding	Similar a la directiva NgModel pero de una forma más declarativa para propiedades, por ejemplo <code>{{title}}</code> , también puede contener declaraciones de eventos por ejemplo (click) , (blur) , etc.
Components	Referencias a otros componentes, por ejemplo <code><app-phones-list></code> .

3. Data Binding

Angular soporta Data Binding (Enlace de datos) bidireccional. Sin un framework los desarrolladores tendrían que actualizar manualmente los datos en HTML, gracias a Angular se puede enlazar los datos mostrados en la plantilla usando propiedades del componente y se pueden actualizar propiedades desde el template hacia las propiedades.



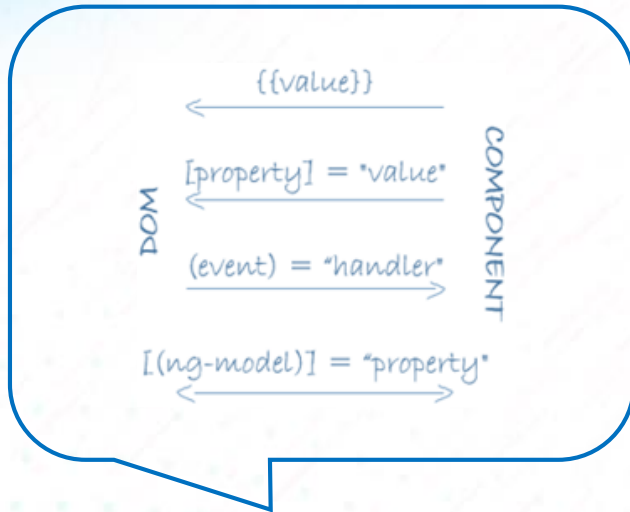
TIPO DE DATA BINDINGS	DETALLES
[brand] property binding	Pasa el valor de brand como propiedad a una instancia del componente phone <phone-creation>.
(click) event binding	Ejecuta el evento click de un elemento al que se le agrega el evento: <button>, <select>, <phone-creation>, etc.
{{phone.number}} interpolation	Muestra el número de teléfono del objeto phone dentro del element al que se agrega.

3. Data Binding

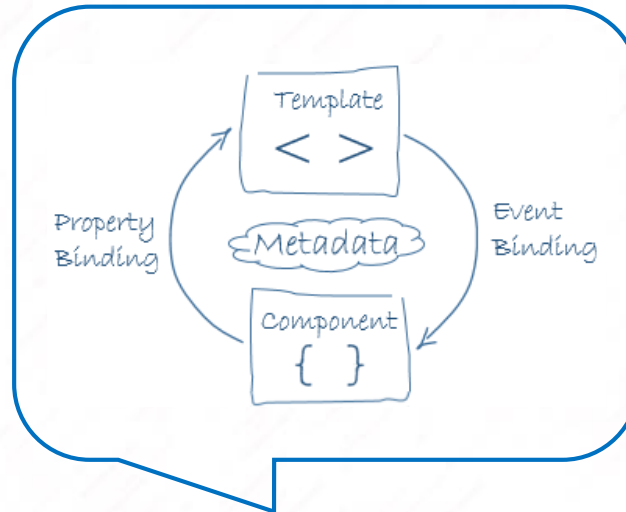
Angular soporta Data Binding (Enlace de datos) bidireccional. Sin un framework los desarrolladores tendrían que actualizar manualmente los datos en HTML, gracias a Angular se puede enlazar los datos mostrados en la plantilla usando propiedades del componente y se pueden actualizar propiedades desde el template hacia las propiedades.

TIPO DE DATA BINDINGS	DETALLES
[brand] property binding	Pasa el valor de brand como propiedad a una instancia del componente phone <phone-creation>,.
(click) event binding	Ejecuta el evento click de un elemento al que se le agrega el evento: <button>, <select>, <phone-creation>, etc.
{{phone.number}} interpolation	Muestra el número de teléfono del objeto phone dentro del element al que se agrega.

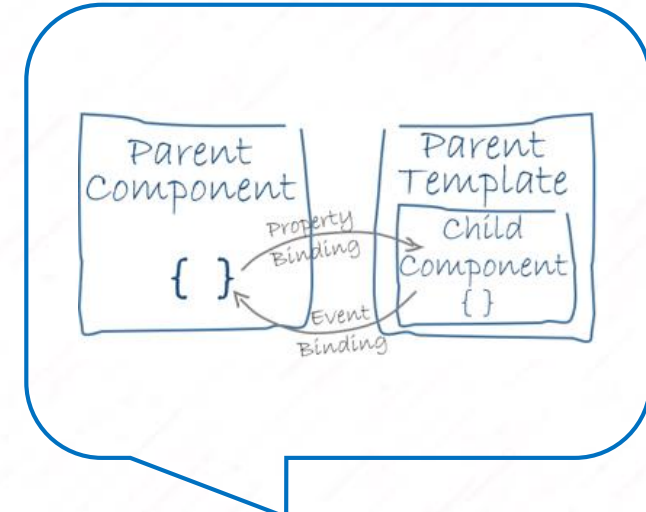
3. Data Binding



Property Binding



Two-Way Binding



Parent-Child Binding

3. Data Binding

```
src > app > phones > TS phones.component.ts > ...
1  ∨ import { Component, OnInit } from '@angular/core';
2  | import { Phone } from './models/phone.model';
3
4  ∨ @Component({
5    selector: 'app-phones',
6    templateUrl: './phones.component.html',
7    styleUrls: ['./phones.component.css']
8  })
9  ∨ export class PhonesComponent implements OnInit {
10 |   phones!: Phone[];
11
12 |   constructor() {
13 |     setTimeout(() => {
14 |       this.phones = [
15 |         { id: 1, brand: 'Samsung', model: 'S9', number: '+51968133858' },
16 |         { id: 2, brand: 'Apple', model: 'iPhone 11', number: '+51994665973' },
17 |         { id: 3, brand: 'Xiaomi', model: 'Mi 11', number: '+51992778439' }
18 |       ]
19 |     }, 5000);
20 |   }
21
22 |   ngOnInit(): void {
23
24 |   }
25 }
```

3. Data Binding

```
Go to component
1 <h1 id="tableLabel">Phones List</h1>
2
3 <p>This component shows the phones list.</p>
4
5 <p *ngIf="!phones"><em>Loading...</em></p>
6
7 <table class='table table-striped' aria-labelledby="tableLabel" *ngIf="phones">
8   <thead>
9     <tr>
10      <th>Id</th>
11      <th>Brand</th>
12      <th>Model</th>
13      <th>Number</th>
14    </tr>
15  </thead>
16  <tbody>
17    <tr *ngFor="let phone of phones">
18      <td>{{ phone.id }}</td>
19      <td>{{ phone.brand }}</td>
20      <td>{{ phone.model }}</td>
21      <td>{{ phone.number }}</td>
22    </tr>
23  </tbody>
24 </table>
```

4. Pipes

Los *Pipes* de Angular te permiten realizar transformaciones a valores para ser mostrados en la vista. Angular posee *Pipes* nativos pero también se pueden crear *Pipes* personalizados utilizando la anotación **@Pipe()**, que es una función que marca una clase para identificarla como *Pipe* e inicializar sus propiedades.

```
{{interpolated_value | pipe_name}}
```

```
<!-- Default format: output 'Jun 15, 2015'-->
```

```
<p>Today is {{today | date}}</p>
```

```
<!-- fullDate format: output 'Monday, June 15, 2015'-->
```

```
<p>The date is {{today | date:'fullDate'}}</p>
```

```
<!-- shortTime format: output '9:43 AM'-->
```

```
<p>The time is {{today | date:'shortTime'}}</p>
```

5. Directives

Las plantillas de Angular son dinámicas, cuando Angular renderiza la plantilla se actualiza una parte del documento HTML (DOM). Las directivas son clases marcadas con el decorator **@Directive()**.

Directivas estructurales

DIRECTIVES	DETAILS
<u>*ngFor</u>	Directiva iterativa que permite replicar secciones html utilizando una lista, Ejm. , <div>.
<u>*ngIf</u>	Directiva condicional que renderiza una sección dependiendo de la condición especificada.

Directivas de Atributos

DIRECTIVES	DETAILS
NgClass	Para agregar o remover clases CSS al elemento.
NgStyle	Para agregar o remover estilos HTML al elemento.
NgModel	Para especificar la propiedad del componente que se enlaza al valor de un input.

The background is a dark blue gradient. It features several glowing, overlapping pinkish-purple elliptical orbits. In the lower-left corner, there is a dense field of small, light blue dots. Along one of the pink orbits, there are several small, bright blue dots. At the bottom right, there is a bright, glowing yellow and orange point, possibly representing a star or a source of light, with a soft blue glow around it.

5. ROUTING

1. RouterModule

Para agregar rutas a nuestra aplicación se utiliza el módulo de Angular llamado RouterModule, cuyos métodos `forRoot()` y `forChild()` se encargan de registrar las rutas ya sea referenciando directamente a un componente o haciendo uso de módulos mediante el uso de la función `loadChildren()`.

```
13  @NgModule({
14    declarations: [
15      AppComponent,
16      NavMenuComponent,
17      HomeComponent,
18      CounterComponent,
19      FetchDataComponent
20    ],
21    imports: [
22      BrowserModule.withServerTransition({ appId: 'ng-cli-universal' }),
23      HttpClientModule,
24      FormsModule,
25      RouterModule.forRoot([
26        { path: '', component: HomeComponent, pathMatch: 'full' },
27        { path: 'counter', component: CounterComponent },
28        { path: 'fetch-data', component: FetchDataComponent },
29        { path: 'phones', loadChildren: () => import('./phones/phones.module').then(m => m.PhonesModule) }
30      ])
31    ],
32    providers: [],
33    bootstrap: [AppComponent]
34  })
35  export class AppModule { }
```

An abstract graphic featuring a dark blue background with a pattern of small, light blue dots. Overlaid on this are several glowing, neon-like lines in shades of pink and magenta. These lines form a complex, swirling pattern that resembles a stylized 'S' or a series of overlapping loops. A bright, glowing pink comet-like shape with a yellow tip is positioned at the bottom right, appearing to move along one of the loops. Several small, bright blue and white dots are scattered along the paths of the glowing lines.

6. SERVICES

Services

Un servicio es un concepto amplio que puede abarcar un valor, una función o un feature que la aplicación necesite. Un servicio es una clase con un propósito bien definido, que debe hacer algo específico y de manera correcta.

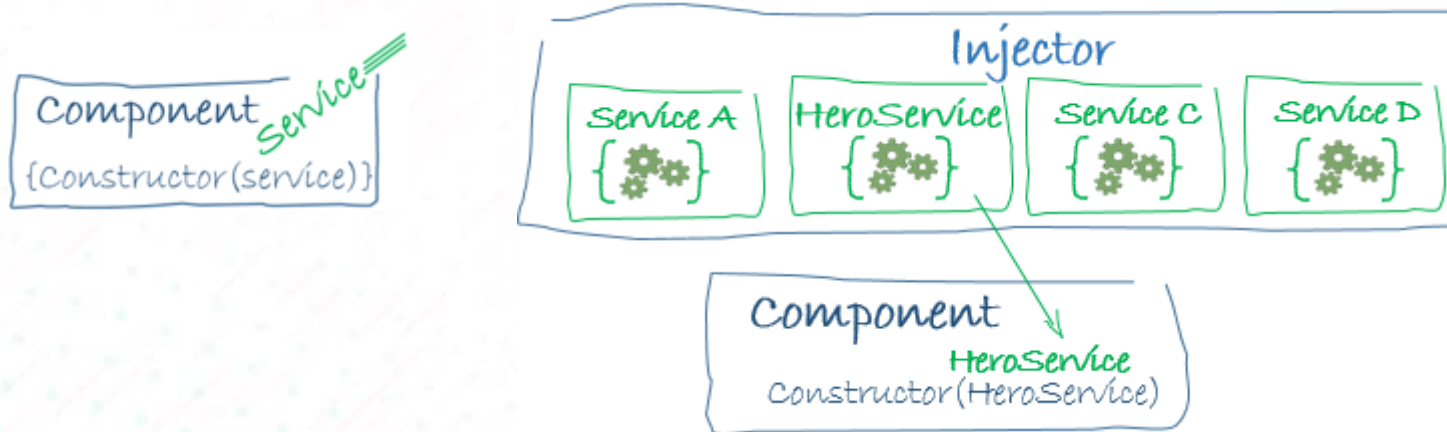
src/app/logger.service.ts (class)

```
export class Logger {  
  log(msg: any) { console.log(msg); }  
  error(msg: any) { console.error(msg); }  
  warn(msg: any) { console.warn(msg); }  
}
```

Angular distingue un Componente de un servicio con el fin de incrementar la modularidad y reusabilidad. A través de la separación de la funcionalidad relacionada con la vista que realiza el componente de otro tipo de proceso, se pueden diseñar los components de una forma más eficiente.

1. Dependence Injection

Angular posee un motor de inyección de dependencias que se utiliza a través de todo el framework para proveer a los components los servicios que necesitan, los servicios se registran en el apartado de *providers* de cada módulo y se pueden utilizar a través de toda la aplicación.



2. Generación de Services

Se puede agregar rutas navegación al módulo principal usando el cliente de Angular

```
ng generate service phones
```

```
phones.service.ts  ➤ ✕
D:\Projects\Singular.Demo.Web\ClientApp\tconfig.json  { } "phones.service"

1  import { Injectable } from '@angular/core';
2
3  @Injectable({
4    providedIn: 'root'
5  })
6  export class PhonesService {
7
8    constructor() { }
9
10 }
```

```
phones.service.spec.ts  ➤ ✕
D:\Projects\Singular.Demo.Web\ClientApp\tconfig.json  { } "phones.service.spec"

1  import { TestBed } from '@angular/core/testing';
2
3  import { PhonesService } from './phones.service';
4
5  describe('PhonesService', () => {
6    let service: PhonesService;
7
8    beforeEach(() => {
9      TestBed.configureTestingModule({});
10     service = TestBed.inject(PhonesService);
11   });
12
13   it('should be created', () => {
14     expect(service).toBeTruthy();
15   });
16 });
17
```

2. HttpModule

Para consumir servicios Rest Api se puede hacer uso del NgModule llamado HttpModule que posee todas las implementaciones necesarias para hacer llamadas http a servicios backend

ng generate service phones

```
phones.service.ts  ➤ ✕
D:\Projects\Singular.Demo.Web\ClientApp\tconfig.json  { } "phones.service"

1  import { Injectable } from '@angular/core';
2
3  @Injectable({
4    providedIn: 'root'
5  })
6  export class PhonesService {
7
8    constructor() { }
9
10 }
```

```
phones.service.spec.ts  ➤ ✕
D:\Projects\Singular.Demo.Web\ClientApp\tconfig.json  { } "phones.service.spec"

1  import { TestBed } from '@angular/core/testing';
2
3  import { PhonesService } from '../phones.service';
4
5  describe('PhonesService', () => {
6    let service: PhonesService;
7
8    beforeEach(() => {
9      TestBed.configureTestingModule({});
10     service = TestBed.inject(PhonesService);
11   });
12
13   it('should be created', () => {
14     expect(service).toBeTruthy();
15   });
16 });
17
```

The background is a dark blue gradient. In the bottom-left corner, there is a grid of small, light blue dots. Two glowing, magenta-pink lines swirl around the center of the image. One line forms a large, open loop, while the other is more complex, crossing itself and ending in a bright, glowing tip. Several small, bright blue and white dots are scattered along the paths of the magenta lines.

7. FORMS

Forms

El módulo de Forms de Angular se integran perfectamente con los formularios de HTML, haciendo uso del mecanismo de data binding y el módulo FormsModule y ReactiveFormsModule, se pueden diseñar formularios y realizar validación de inputs

```
ng generate component phones-form --module=phones
```

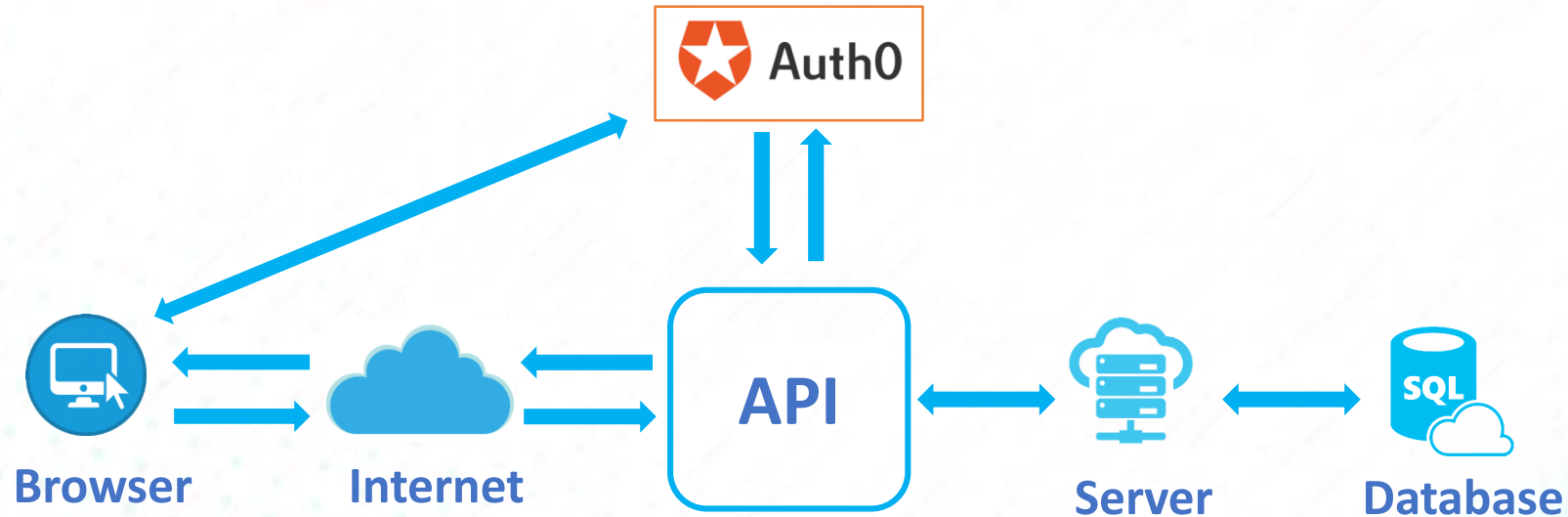
```
20  ✓  ngOnInit(): void {  
21      |  this.createForm();  
22      |  }  
23  
24  ✓  private createForm() {  
25  ✓      this.form = this.formBuilder.group({  
26          |  id: [null],  
27          |  brand: [null, [Validators.required]],  
28          |  model: [null, [Validators.required, Validators.minLength(2)]],  
29          |  number: [null, [Validators.required, Validators.minLength(2)]],  
30          |  });  
31      |  }
```

An abstract graphic featuring a dark blue background with a pattern of small, light blue dots. Overlaid on this are several glowing, neon-like lines in shades of pink and magenta. These lines form a complex, swirling pattern that resembles a stylized 'S' or a series of overlapping loops. A bright, glowing pink comet-like shape with a yellow tip is positioned at the bottom right, appearing to move along one of the loops. Several small, bright blue and white dots are scattered throughout the lower half of the image, some appearing to be part of the glowing lines.

8. SEGURIDAD

Seguridad

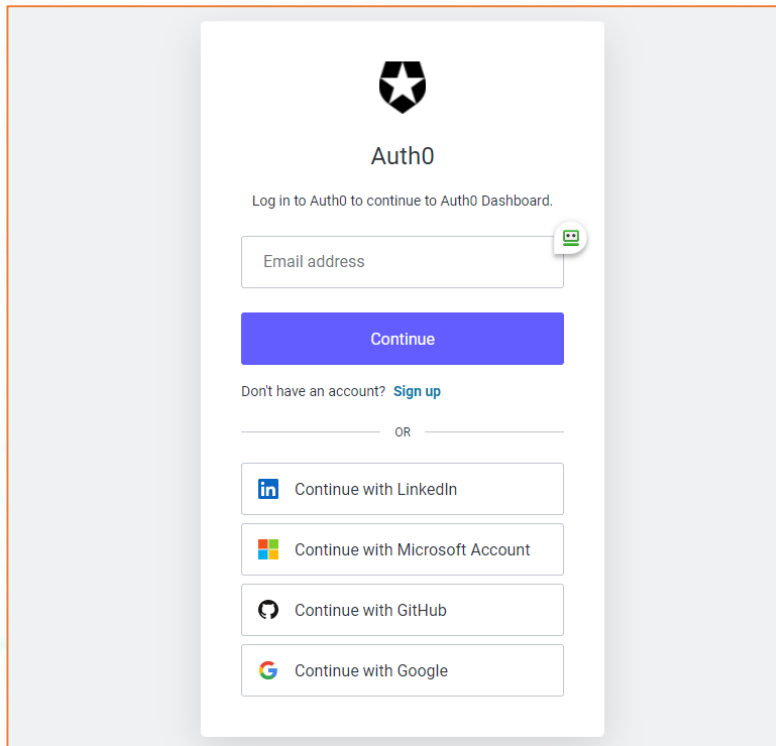
Para asegurar nuestras Apis también vamos a usar el servicio de autorización y autenticación de terceros. Esta vez vamos a utilizar una aplicación de tipo SPA en nuestro servicio Auth0 para poder autenticarnos en nuestra aplicación Angular



1. Auth0

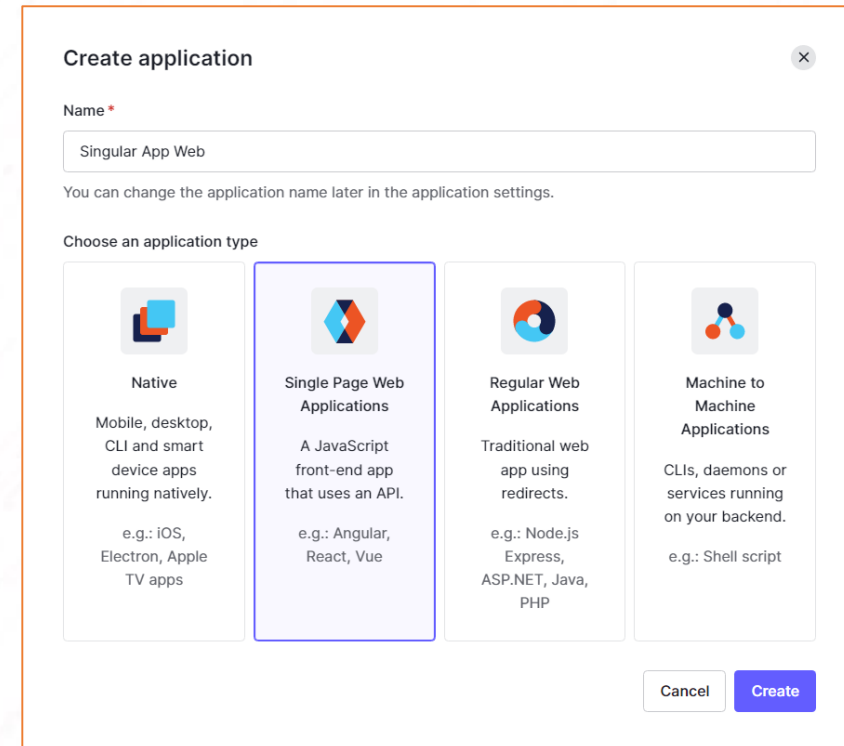
Ingresar  <https://manage.auth0.com/>

Crear Cuenta



The login screen features the Auth0 logo at the top. Below it, a message states 'Log in to Auth0 to continue to Auth0 Dashboard.' There is an input field for 'Email address' with a green eye icon for toggling visibility. A blue 'Continue' button is positioned below the input field. Underneath, a link for 'Sign up' is provided for users without an account. An 'OR' separator is followed by five social login options: 'Continue with LinkedIn', 'Continue with Microsoft Account', 'Continue with GitHub', and 'Continue with Google', each with its respective logo.

Crear Aplicación



The 'Create application' screen has a title bar with a close button. It starts with a 'Name' field containing 'Singular App Web'. A note below states 'You can change the application name later in the application settings.' The 'Choose an application type' section displays four options: 'Native' (Mobile, desktop, CLI and smart device apps running natively, e.g.: iOS, Electron, Apple TV apps), 'Single Page Web Applications' (A JavaScript front-end app that uses an API, e.g.: Angular, React, Vue), 'Regular Web Applications' (Traditional web app using redirects, e.g.: Node.js Express, ASP.NET, Java, PHP), and 'Machine to Machine Applications' (CLIs, daemons or services running on your backend, e.g.: Shell script). The 'Single Page Web Applications' option is highlighted with a blue border. At the bottom right, there are 'Cancel' and 'Create' buttons.

1. Auth0: AuthModule

```
npm install @auth0/auth0-angular
```

```
21 imports: [  
22   BrowserModule.withServerTransition({ appId: 'ng-cli-universal' }),  
23   HttpClientModule,  
24   FormsModule,  
25   RouterModule.forRoot([  
26     { path: '', canActivate: [AuthGuard], component: HomeComponent, pathMatch: 'full' },  
27     { path: 'counter', canActivate: [AuthGuard], component: CounterComponent },  
28     { path: 'fetch-data', canActivate: [AuthGuard], component: FetchDataComponent },  
29     { path: 'phones', canActivate: [AuthGuard], loadChildren: () => import('./phones/phones.module').then(m => m.PhonesModule) }  
30   ]),  
31   // Import the module into the application, with configuration  
32   AuthModule.forRoot({  
33     domain: 'dev-***-****.auth0.com',  
34     clientId: '*****'  
35   })  
36 ],
```

1. Auth0: Users

Ingresar  <https://manage.auth0.com/>

Crear Usuario

Create user

Email *

nelsson.aguilar@singulartech.ai

Password *

.....

Repeat Password *

.....

Connection *

Username-Password-Authentication

Cancel

Create

Add Roles

Select roles to assign to this user. You may assign up to 50 roles per user.

Singular Admin X

Select Roles

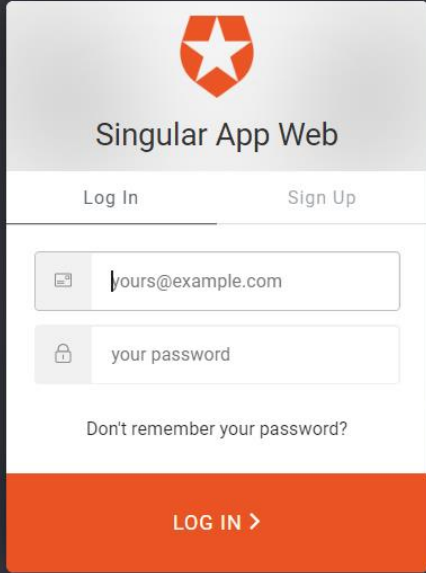
X

▼

Assign

1. Auth0: Login

Auth0 Login



The image shows a login interface for 'Singular App Web'. At the top is a logo consisting of a red shield with a white star. Below the logo, the text 'Singular App Web' is displayed. There are two tabs: 'Log In' (selected) and 'Sign Up'. Below the tabs are two input fields: the first is for an email address, containing 'yours@example.com', and the second is for a password, containing 'your password'. Below the password field is a link that says 'Don't remember your password?'. At the bottom is a large orange button with the text 'LOG IN >'.

1. Auth0: User Info

Singular.Demo.Web

Phones Home Counter Fetch data **nelsson.aguilar@singulartech.ai** Log out

Hello, world!

Welcome to your new single-page application, built with:

- [ASP.NET Core](#) and [C#](#) for cross-platform server-side code
- [Angular](#) and [TypeScript](#) for client-side code
- [Bootstrap](#) for layout and styling

To help you get started, we've also set up:

- **Client-side navigation.** For example, click *Counter* then *Back* to return here.
- **Angular CLI integration.** In development mode, there's no need to run `ng serve`. It runs in the background automatically, so your client-side resources are dynamically built on demand and the page refreshes when you modify any file.
- **Efficient production builds.** In production mode, development-time features are disabled, and your `dotnet publish` configuration automatically invokes `ng build` to produce minified, ahead-of-time compiled JavaScript files.

The `ClientApp` subdirectory is a standard Angular CLI application. If you open a command prompt in that directory, you can run any `ng` command (e.g., `ng test`), or use `npm` to install extra packages into it.

2. Guards

Los *Guards* son mecanismos de seguridad de Angular para asegurar las rutas. A través de la implementación *CanActivate* podemos crear una clase que actúe como Middleware y agregar lógica para validar si una ruta tiene los permisos necesarios para que pueda ser accedida.

```
TS auth.guard.d.ts X
node_modules > @auth0 > auth0-angular > lib > TS auth.guard.d.ts > AuthGuard
1 import { ActivatedRouteSnapshot, RouterStateSnapshot, CanActivate, CanLoad, Route, UrlSegment, CanActivateChild
2 import { Observable } from 'rxjs';
3 import { AuthService } from './auth.service';
4 import * as ɵngcc0 from '@angular/core';
5 export declare class AuthGuard implements CanActivate, CanLoad, CanActivateChild {
6     private auth;
7     constructor(auth: AuthService);
8     canLoad(route: Route, segments: UrlSegment[]): Observable<boolean>;
9     canActivate(next: ActivatedRouteSnapshot, state: RouterStateSnapshot): Observable<boolean>;
10    canActivateChild(childRoute: ActivatedRouteSnapshot, state: RouterStateSnapshot): Observable<boolean>;
11    private redirectIfUnauthenticated;
12    static ɵfac: ɵngcc0.ɵɵFactoryDeclaration<AuthGuard, never>;
13 }
14
15 //# sourceMappingURL=auth.guard.d.ts.map
```

3. Interceptors

Los interceptores de Angular son clases de tipo Middleware que permiten realizar modificaciones a los Requests Http, cómo por ejemplo agregar o leer *headers*, modificar el *body*, entre otros. Los interceptores se agregan a la constante de Angular de tipo Array llamada HTTP_INTERCEPTORS que es utilizada por el HttpClientModule para realizar cambios en el *request* antes de llamar a las apis.

```
1 import { HttpInterceptor, HttpRequest, HttpHandler, HttpEvent } from '@angular/common/http';
2 import { Observable } from 'rxjs';
3 import { AuthClientConfig } from './auth.config';
4 import { Auth0Client } from '@auth0/auth0-spa-js';
5 import { AuthState } from './auth.state';
6 import * as NgModule from '@angular/core';
7 export declare class AuthHttpInterceptor implements HttpInterceptor {
8     private configFactory;
9     private auth0Client;
10    private authState;
11    constructor(configFactory: AuthClientConfig, auth0Client: Auth0Client, authState: AuthState);
12    intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>>;
```

A decorative graphic consisting of two intersecting pink elliptical orbits with a small bright blue and white star at one of the intersection points.

GRACIAS

LIMA - PERÚ

Calle Monte Rosa 270 Of. 502, Santiago de Surco

contacto@singulartech.ai / +51 992778439 / +51 999015208

www.singulartech.ai

A decorative graphic at the bottom of the slide consisting of several small blue and white starburst or light flare effects.