Introducción a Angular

Que aprenderemos

- ✓Creando una primera APP
- ✓¿Qué son los componentes?
- ✓¿Qué son las directivas estructurales?
- ✓Trabajando de forma local un proyecto en Angular.
- ✓Una breve introducción sobre todos los archivos usados en el QuickStart de Angular.
- ✓Uso de Bootstrap 4 para nuestros estilos.
- ✓Crear archivos .HTML para que se encarguen de la estructura visual de nuestros componentes.
- ✓Crearemos una aplicación con 3 componentes re-utilizables.
- ✓*ngFor y el *ngIf

Angular – Primera app

- ✓Nos ubicamos en la carpeta donde tenemos nuestros proyectos: ProyectosWeb2020
- ✓Desde una ventana de comandos ingresar a dicha carpeta
- ✓Desde una ventana de comandos crear una aplicación angular llamada MyApp01
  - ➢ng new myapp01

  Cuando se pregunte por la generación automática del routing: N

  Seleccionar CSS para nuestro proyecto
- ✓Arrastrar el proyecto creado a nuestro editor de trabajo
- ✓Desde la ventana de comandos ingresar a la carpeta myapp01

  E:\ProyectosWeb2020\myapp01> ng serve –o
- ✓ng serve //levanta un servidor local en el Puerto 4200
- ✓ng serve –-port 4201 // levanta un servidor local en un puerto específico
- ✓ng serve –o //levanta un servidor local y abre automáticamente el web browser por defecto

Angular – Primera app

✓Hagamos algunas modificaciones y veamos los cambios:

app.component.html

```
<h1>
Bienvenidos a {{ title }}!
</h1>
<ul>
<li>Nombre:{{ nombre }}</li>
<li>Apellido:{{apellido}}</li>
</ul>
```

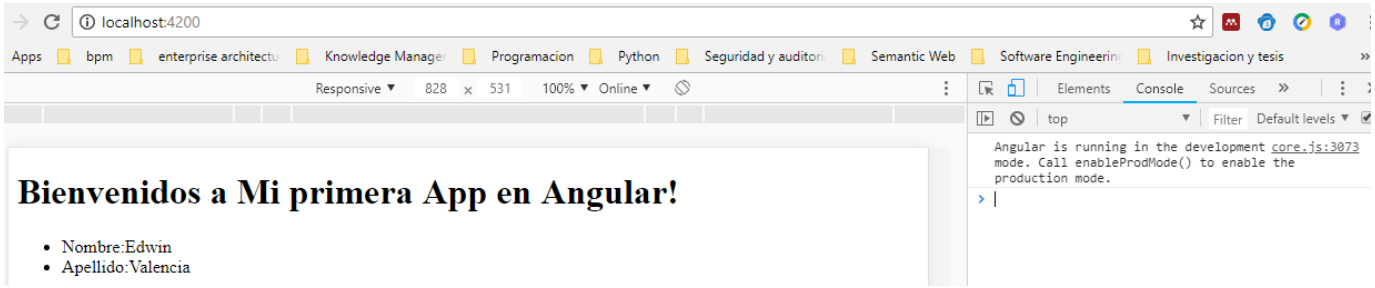
interpolación

Dentro de las llaves se puede colocar cualquier código JavaScript

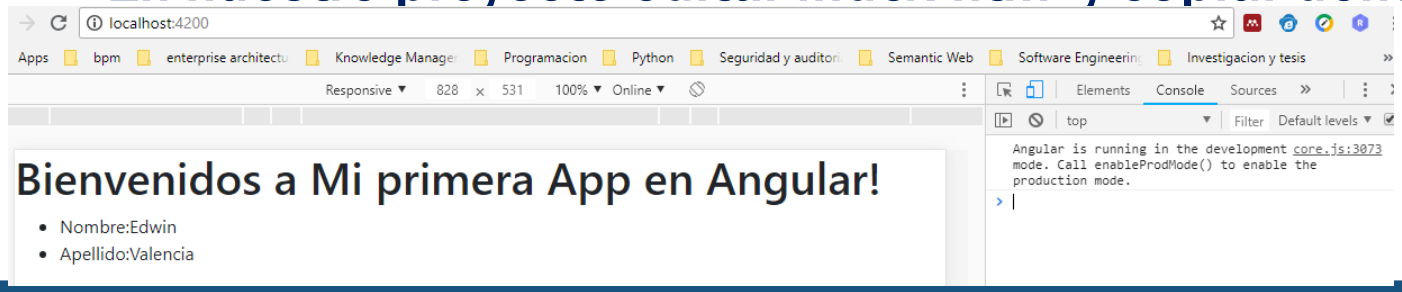
app.component.ts

```
import { Component } from '@angular/core';

@Component({
selector: 'app-root',
templateUrl: './app.component.html',
styleUrls: ['./app.component.css']
})
export class AppComponent {
title = 'Mi primera App en Angular';
nombre = 'Edwin';
apellido = 'Valencia';
}
```

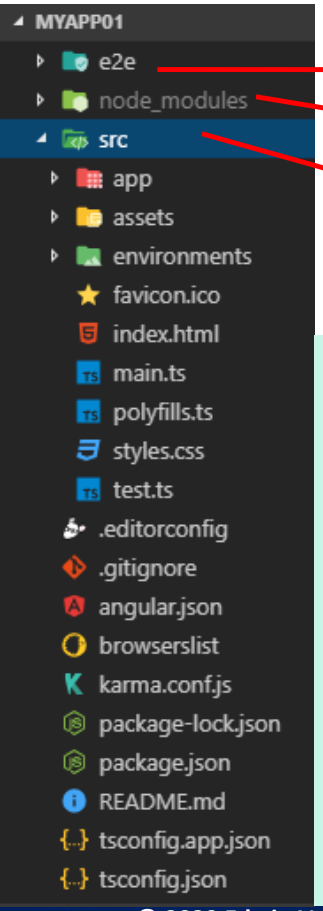
Angular – Primera app



- ✓Mejorando la presentación con bootstrap
- ✓Ir a https://getbootstrap.com/ en download copiar el CDN jsDelivr
- ✓En nuestro proyecto editar index.html y copiar dentro del head



Comprendiendo la estructura de una app Angular



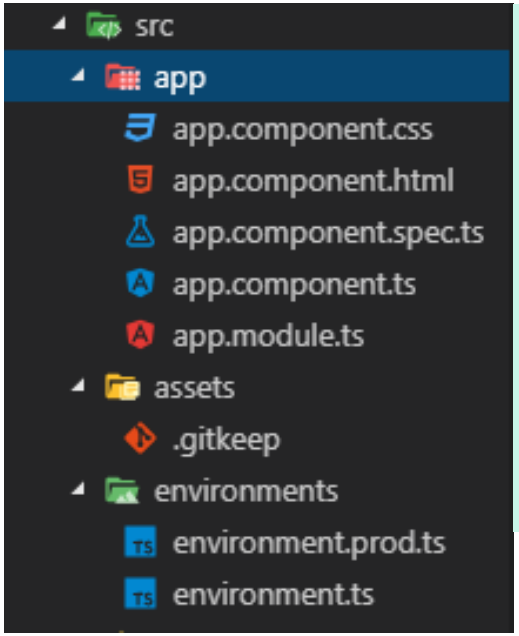
Los archivos dentro de e2e describen un tipo de testing llamado End to End, esto se ejecuta con la ayuda de Jasmine, se utiliza para ejecutar pruebas automaticas

Se encuentran todas nuestras dependencias de Node.js de nuestro proyecto

La carpeta más importante de todas y donde estarás la mayor parte de tu tiempo, ya que en ella se encuentra todo el código de la app

/app: En esta carpeta y sub carpetas, es donde vamos a tener todos nuestros componentes, servicios y aquellos elementos que tengan que ver con las vista de nuestra aplicación.
/assets: En esta carpeta tendremos aquellos assets que no sean propios de un componente en Angular como imágenes, videos o archivos de cualquier tipo. Son los recursos estáticos.
/environments: Nos permite modificar el entorno de trabajo en el que estamos, por lo tanto, cuando lleves algo a producción, modificaremos los respectivos archivos de TypeScript.
favicon.icon: Puedes sustituirlo por el icono de tu preferencia
index.html: Este es el HTML principal y es donde va a existir nuestra Single Web Application.
main.ts: Es el primer archivo en ejecutarse, en él se encuentran todos los parámetros de configuración de la aplicación como el entorno en el que trabajamos, en que archivo tenemos declarados todos nuestros componentes, etc.
polyfills.ts: Este archivo nos asegura que tendremos compatibilidad en todos los navegadores modernos.
styles.css: Son los estilos generales del proyecto.
test.ts: Aquí podemos seguir agregando tests unitarios a nuestros componentes

Comprendiendo la estructura de una app Angular



/app: En esta carpeta y sub carpetas, es donde vamos a tener todos nuestros componentes, servicios y aquellos elementos que tengan que ver con las vista de nuestra aplicación.

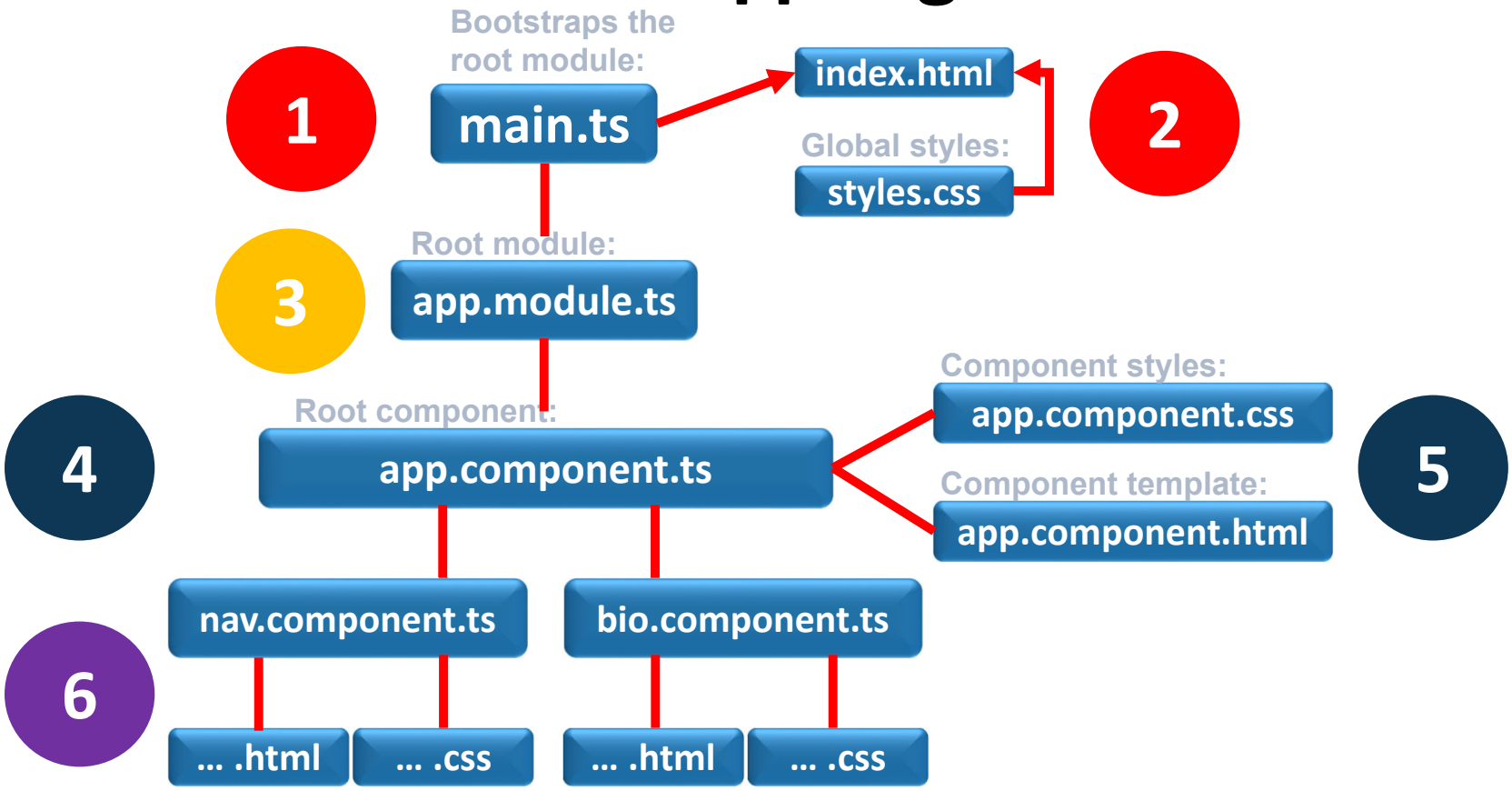
- app.component.css: estilo que se aplica al html
- app.component.html: código html de este componente
- app.component.spect.ts: spect, significa pruebas automaticas
- app.component.ts: código typescript de nuestro componente
- app.module.ts: Es una clase simple con un decorador, define el modulo

/assets: En esta carpeta tendremos aquellos assets que no sean propios de un componente en Angular como imágenes, videos o archivos de cualquier tipo. Son los recursos estáticos.

.gitkeep: mantiene los archivos y carpetas cuando se suba a un repositorio

/environments: Nos permite modificar el entorno de trabajo en el que estamos, por lo tanto, cuando lleves algo a producción, modificaremos los respectivos archivos de TypeScript.

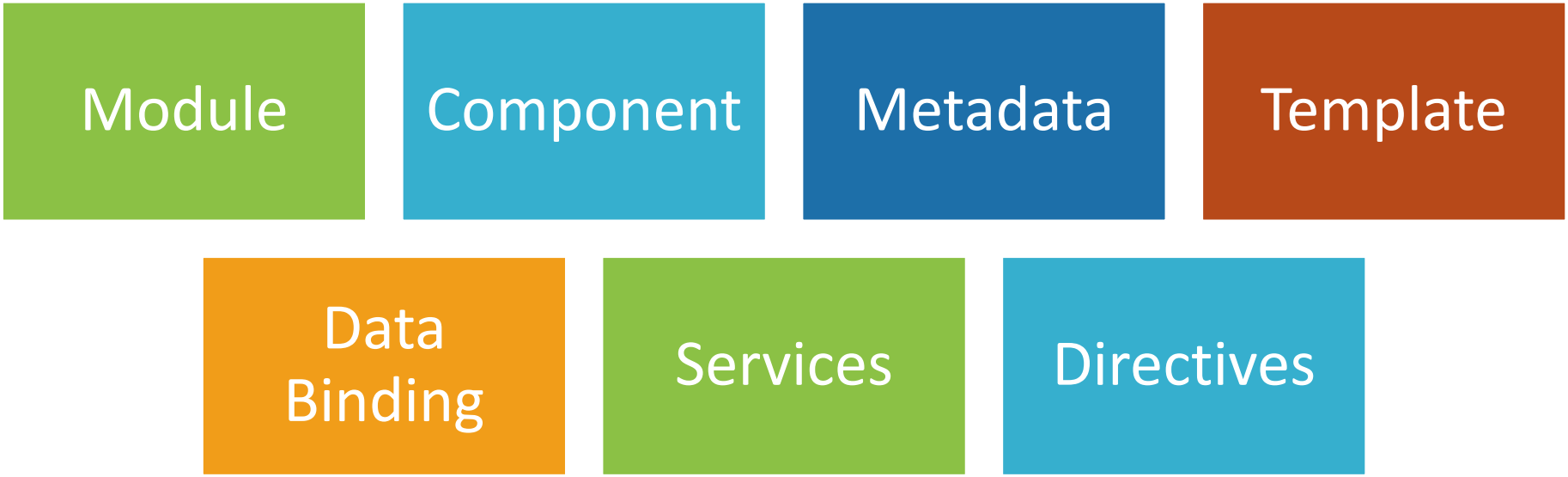
Arquitectura básica de una app Angular



Utilidades de angular CLI

- ✓Component: ng g component my-new-component
- ✓Directive: ng g directive my-new-directive
- ✓Pipe: ng g pipe my-new-pipe
- ✓Service: ng g service my-new-service
- ✓Class: ng g class my-new-class
- ✓Interface: ng g interface my-new-interface
- ✓Enum: ng g enum my-new-enum

Bloques de construcción de angular

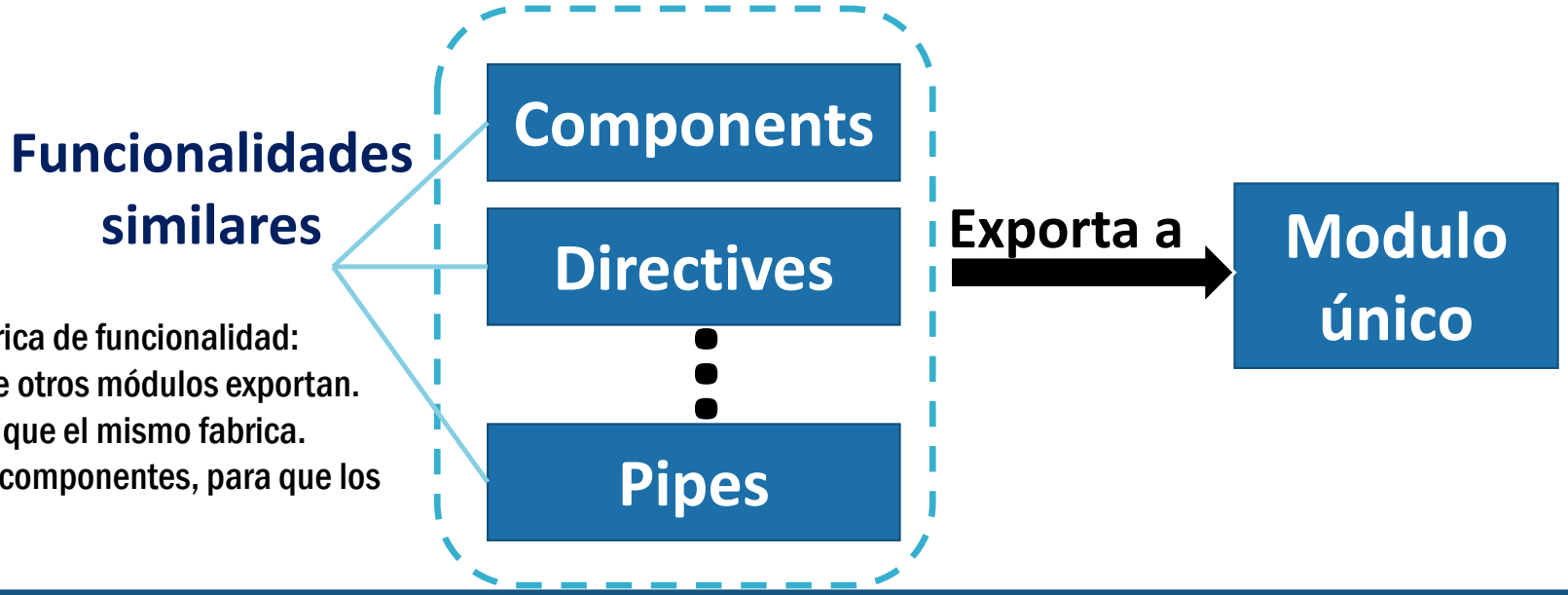


Module

Un modulo es una clase con metadatos @NgModule

Cada App Angular tiene al menos un modulo raíz

Encapsulación de diferentes funcionalidades similares



Un módulo es como una fábrica de funcionalidad:
1. Importa componentes que otros módulos exportan.
2. Declara los componentes que el mismo fabrica.
3. Exporta algunos de estos componentes, para que los consuman otros módulos.

Module

Decorador

Declarando todos los componentes

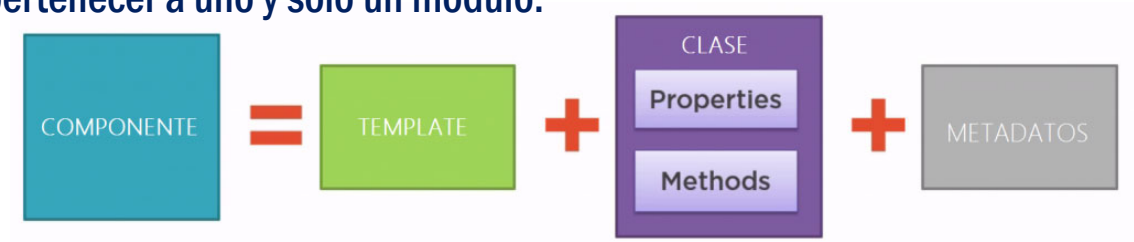
Importando módulos

Proporcionando servicios a todos los componentes del modulo

```
import { BrowserModule } from
'@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';

@NgModule({
declarations: [
AppComponent
],
imports: [
BrowserModule
],
providers: [],
bootstrap: [AppComponent]
})
export class AppModule { }
```

Component

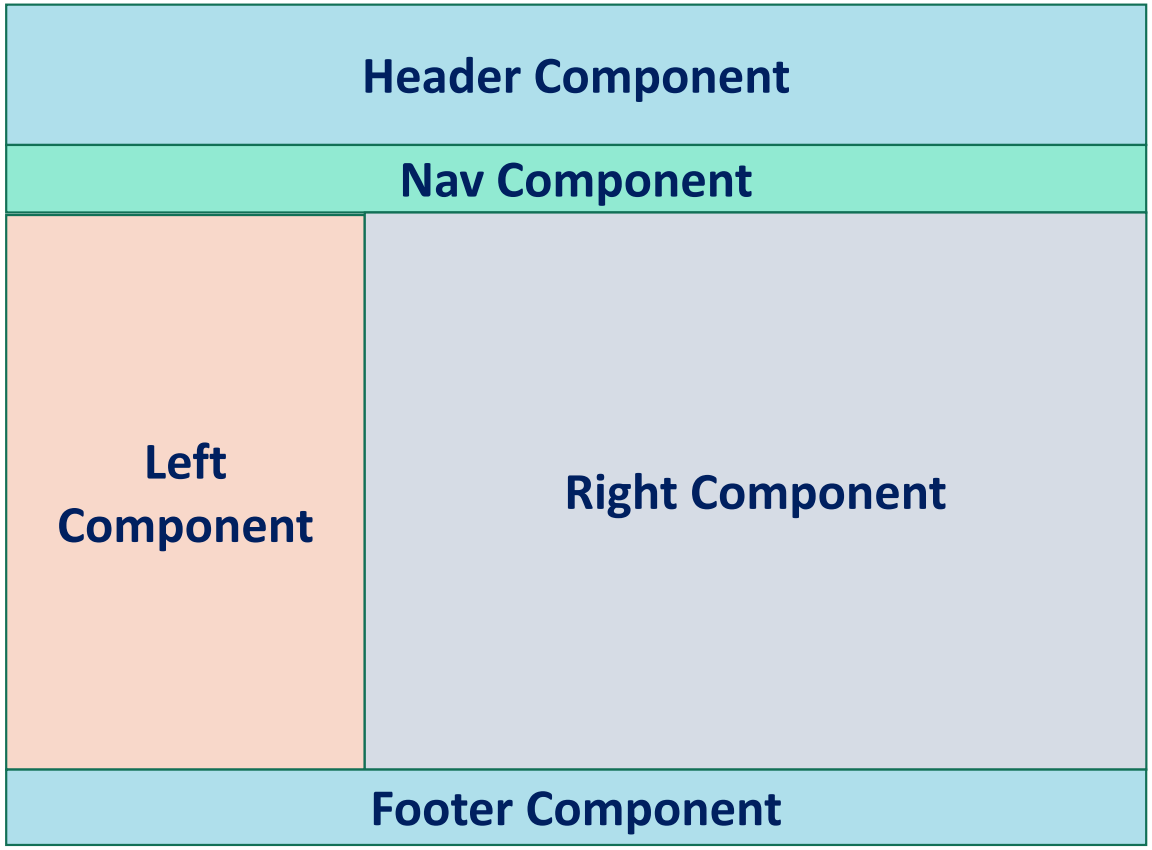
- ✓ El esqueleto de Angular está basado en componentes.
- ✓ Es una vista definida con un template que está asociado a una clase y tiene información adicional mediante metadatos.
- ✓ Un componente controla una parte de la página, llamado vista. Se define la lógica de la aplicación del componente – es decir lo qué hace para admitir la vista dentro de una clase.
- ✓ Un componente puede pertenecer a uno y sólo un módulo.



- ✓ Template: Fragmento de HTML para mostrar la interfaz al usuario definiendo una vista para la aplicación.
- ✓ Clase: Contiene las propiedades y métodos que realiza las acciones para mostrarlas en la vista.
- ✓ Metadatos: Angular necesita los metadatos para componer la vista y para saber cómo el componente va a interactuar con las otras partes de la aplicación.

Component

- ✓Los componentes permiten organizar una app.
- ✓Un componente puede invocar a otros componentes



Component

```
import { Component } from '@angular/core';

@Component({
selector: 'app-root',
templateUrl: './app.component.html',
styleUrls: ['./app.component.css']
})
export class AppComponent {
title = 'Mi primera App en Angular';
nombre = 'Edwin';
apellido = 'Valencia';

}
```

Importar decorador de componente

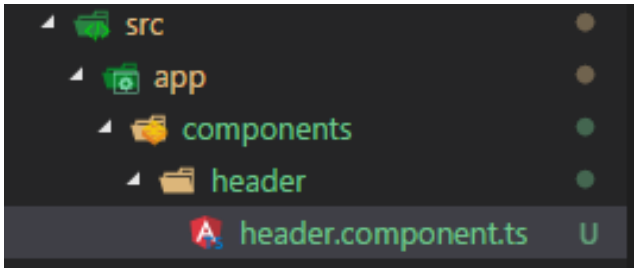
Decorador

Metadatos

Exportar clase componente

Component

- ✓Creando nuestro primer componente de forma manual:
- ✓Dentro de src->app crear una nueva carpeta components y dentro de esta crear la carpeta header
- ✓En la carpeta header crear un archivo header.component.ts



header.component.ts

```
1  import { Component } from '@angular/core';
2
3  @Component({
4      selector: 'app-header',
5      template: `<h1>Header component<h1>`
6  })
7  export class HeaderComponent{
8  }
```

© 2020 Edwin Valencia Castillo

Component

- ✓En el archivo app.module.ts, declaramos nuestro componente para que pueda ser usado
- ✓Insertamos nuestro componente en el app.component.html

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';
import {HeaderComponent} from './components/header/header.component';

@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

app.component.html

```
<app-header></app-header>

<ul>
    <li>Nombre:{{ nombre }}</li>
    <li>Apellido:{{apellido}}</li>
</ul>
```

Component

- ✓Separando el html aparte en un archivo header.component.html
- ✓Crear un nuevo archivo header.component.html dentro de la carpeta header y agregar el código de un navbar copiado de bootstrap
- ✓Modificar el archivo header.component.ts, con el siguiente código:

header.component.ts

```
1  import { Component } from '@angular/core';
2
3  @Component({
4      selector: 'app-header',
5      templateUrl: './header.component.html'
6  })
7  export class HeaderComponent{
8  }
```

- ✓Practica personal: crear un nuevo componente llamado body.component.ts y su template body.component.html
  - ➢ El contenido html debe tener: <h1>Body Component </h1>, el componente debe llamarse app-body

Component

- ✓Creando componentes de manera automática:
- ✓Desde el terminal ejecutar: ng g c components/footer

```
PS E:\AngularPrys\my-app> ng g c components/footer
CREATE src/app/components/footer/footer.component.html (25 bytes)
CREATE src/app/components/footer/footer.component.spec.ts (628 bytes)
CREATE src/app/components/footer/footer.component.ts (269 bytes)
CREATE src/app/components/footer/footer.component.css (0 bytes)
UPDATE src/app/app.module.ts (583 bytes)
```

g->generate, c->component

- ✓Eliminar el archivo footer.component.spec.ts, por el momento no es necesario
- ✓Agregar el componente en app.component.html
- ✓Mejorar la presentación del footer usando estilos

Component

✓Los resultados finales serian:

src > app > components > footer > footer.component.ts > ...

```ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-footer',
  templateUrl: './footer.component.html',
  styleUrls: ['./footer.component.css']
})
export class FooterComponent implements OnInit {
  anio : number;
  fecha : string;
  constructor() {
    this.anio = new Date().getFullYear();
    this.fecha = new Date().toDateString();
   }

}
```

src > app > components > footer > footer.component.html > ...

```html
<footer class="footer bg-dark">
    <div class="container">
        <p>&copy; {{anio}} Escuela Academico de Ingeniería de Sistemas - EAPIS</p>
        <p>última actualización: {{fecha}}</p>
    </div>
</footer>
```

src > app > components > footer > footer.component.css > ...

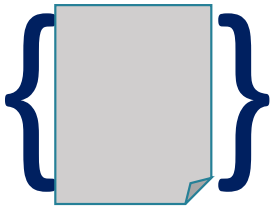
```css
footer {
    color: white;
    position: fixed;
    bottom: 0px;
    width: 100%;
}
```

Metadata

Metadata describe como procesar una clase

Decorator es usado para enlazar metadatos

Ejemplo

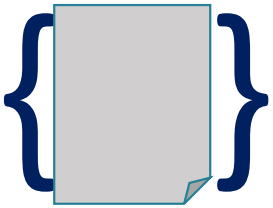


MyClass

\+

```
@Component({
---------------
})
```

Component()



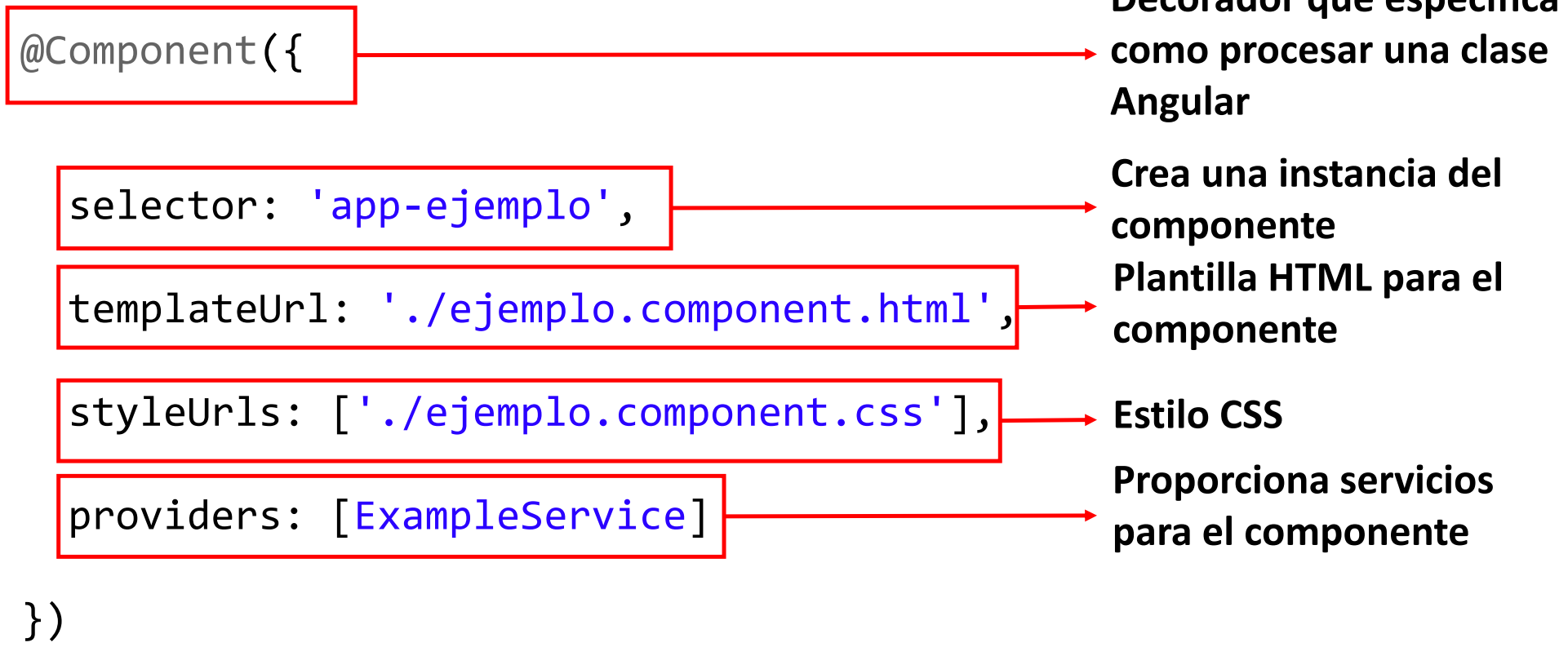
MyClass

\+

```
@NgModule({
---------------
})
```

Module()

Metadata

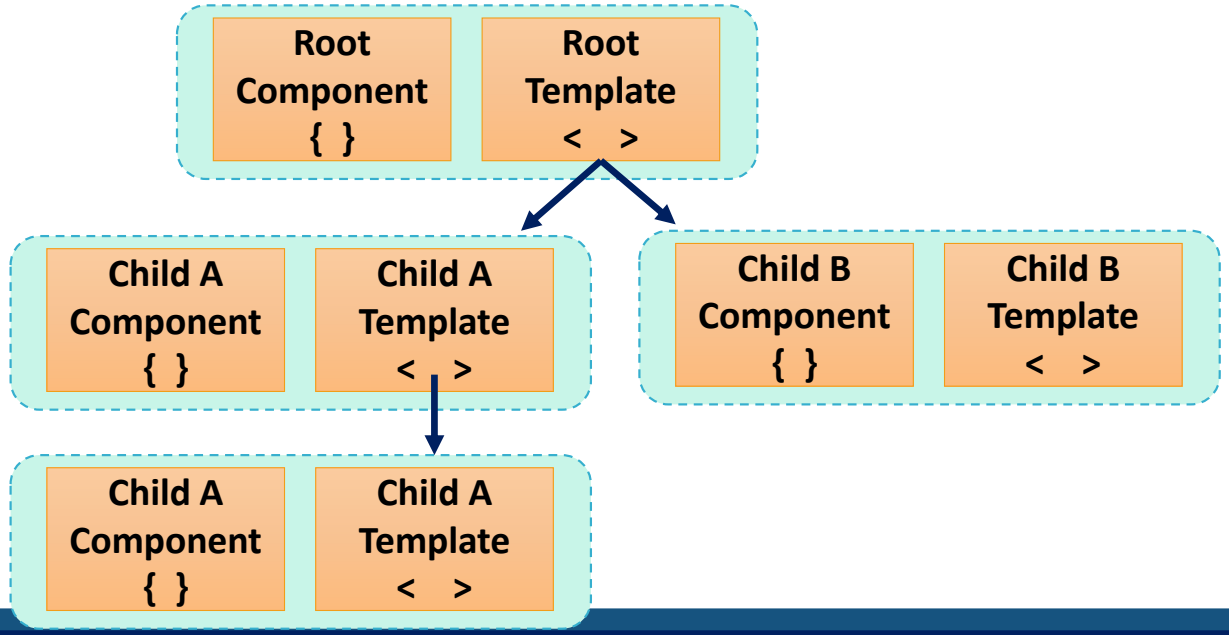

@Component({
selector: 'app-ejemplo',
templateUrl: './ejemplo.component.html',
styleUrls: ['./ejemplo.component.css'],
providers: [ExampleService]
})
Decorador que especifica como procesar una clase Angular
Crea una instancia del componente
Plantilla HTML para el componente
Estilo CSS
Proporciona servicios para el componente

Template

Usado para definir la vista de un componente

Parece HMTL a excepción de algunas diferencias

Describe como el componente es traducido en la página

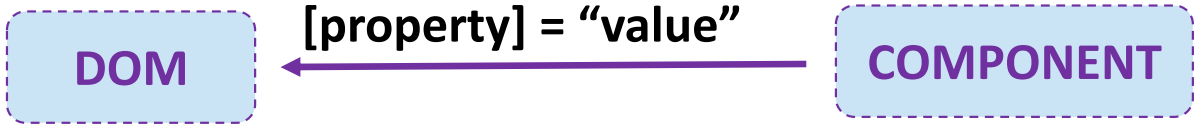


Data Binding

✓El data binding lo podemos definir como la comunicación entre el código Typescript y nuestro HTML, los tipos de Data Binding son:

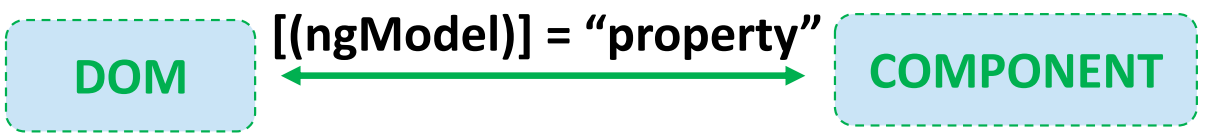
1. INTERPORLATION

2. PROPERTY BINDING



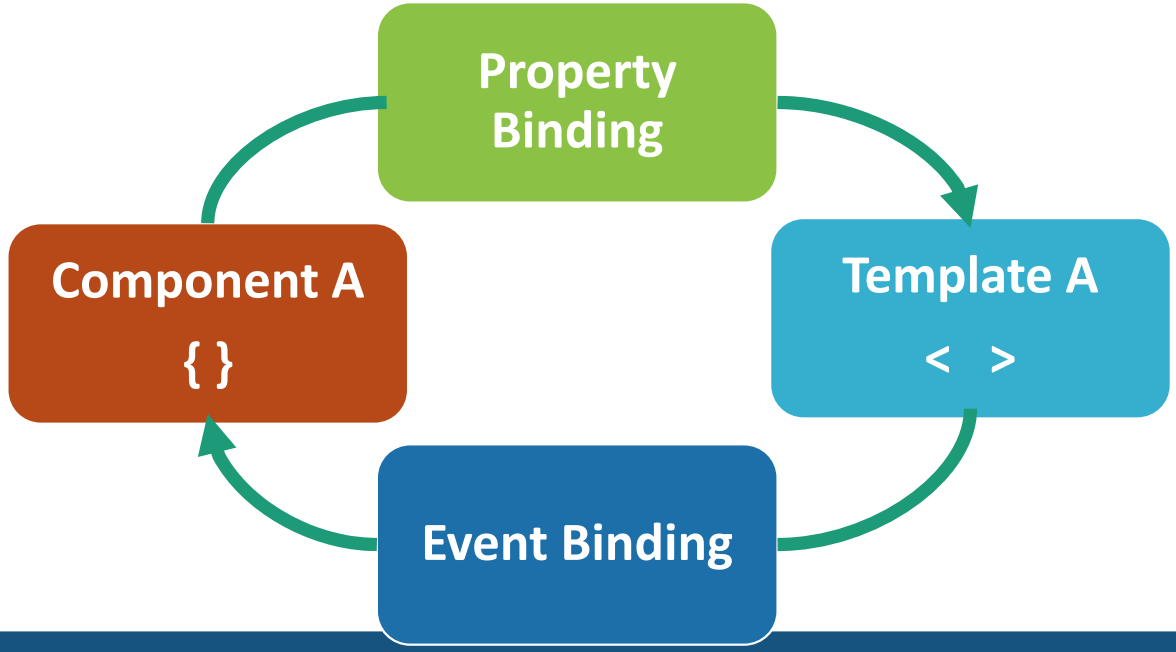
3. EVENT BINDING

4. WAY DATA BINDING



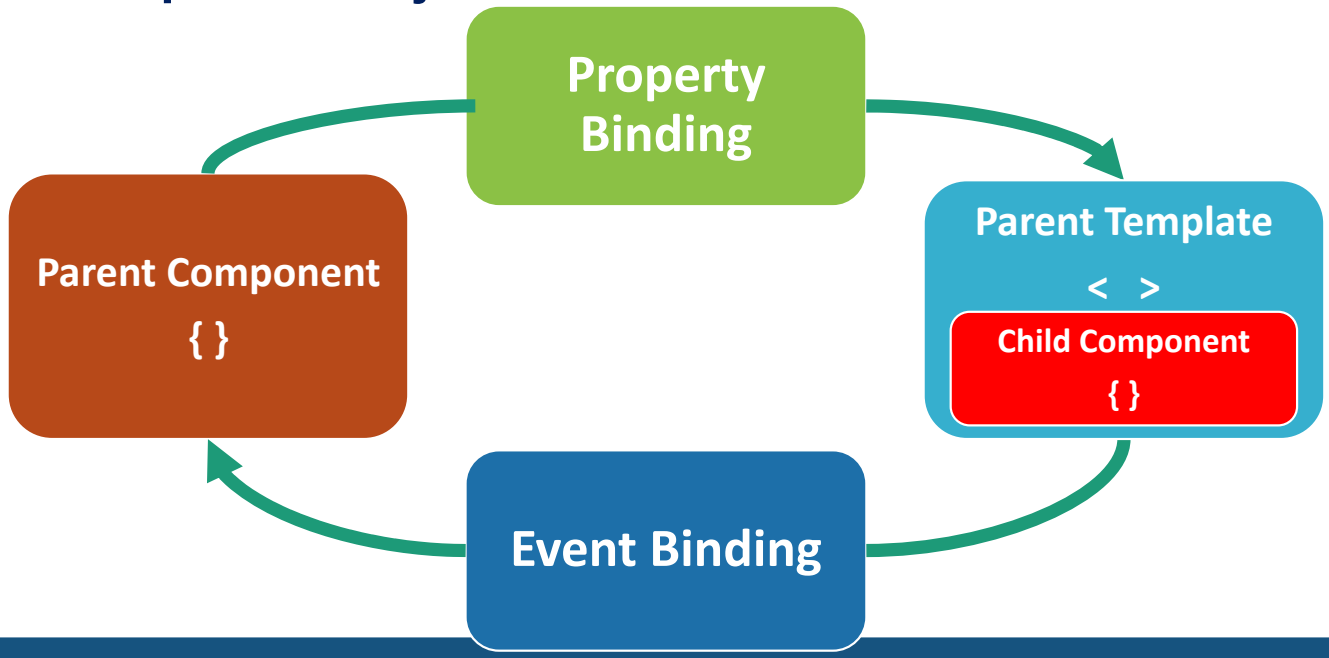
Data Binding

✓Data binding juega un importante rol en la comunicación entre una plantilla y su componente



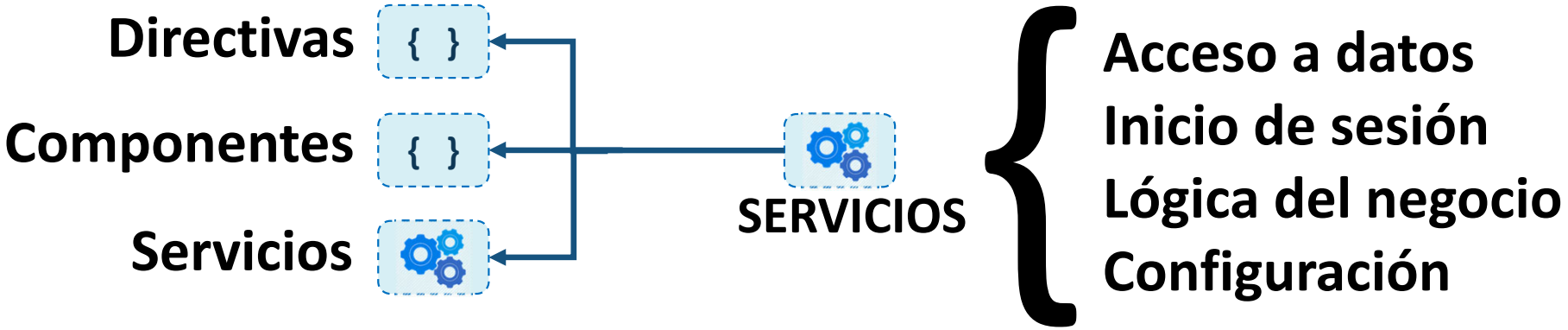
Data Binding

✓Data binding es también importante para la comunicación entre componentes padre e hijo.



Services

- ✓Es una categoría amplia que abarca cualquier valor, función o característica que tu aplicación necesita.
- ✓Ejemplo: logging service | data service | message bus | tax calculator



Services

```
import { Injectable } from '@angular/core';

@Injectable()
export class EjemploService {

  movies: string[] = [“Inception”, “Dark Knight”, “Shutter Island”];

  constructor() { }

  getMovies(): string[]
  {
     return this.movies;
  }

}
```

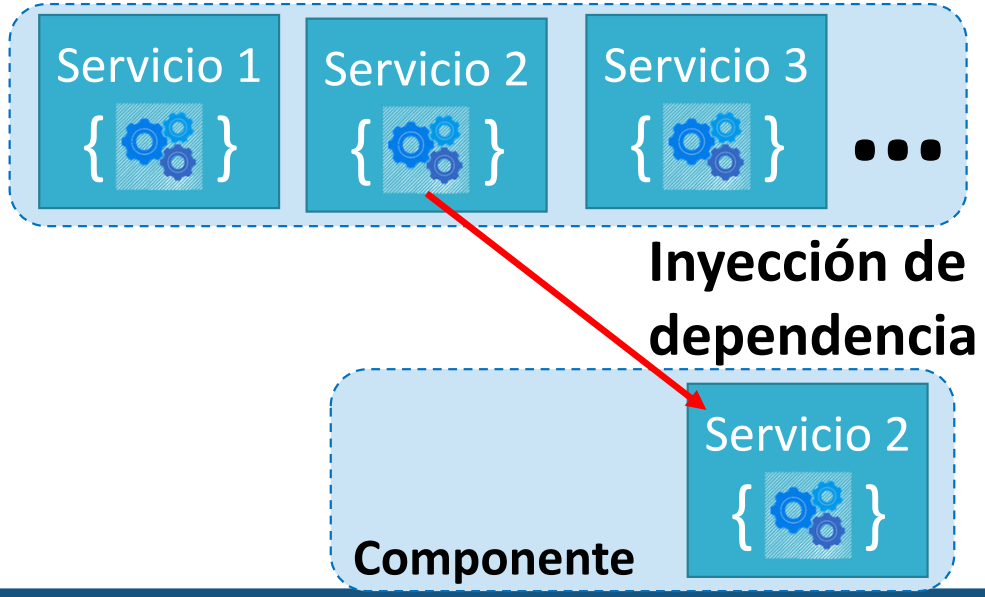
Clase Servicio

Método del servicio para obtener datos

Services

Crea una nueva instancia de una clase junto con sus dependencias requeridas

Usado para proporcionar servicios a un componente



Services

```
import { Component, OnInit } from '@angular/core';
import { EjemploService } from '../ejemplo.service';

@Component({
  selector: 'app-ejemplo',
  templateUrl: './ejemplo.component.html',
  styleUrls: ['./ejemplo.component.css'],
  providers: [EjemploService],
})
export class EjemploComponent implements OnInit {

  movies: string[];
  constructor(private ejempoService: EjemploService) { }

  ngOnInit() {
      this.movies = this.ejempoService.getMovies();
  }
}
```

Importando la clase Servicio

Inyectando el servicio en el componente

Obteniendo datos

Directives

✓Cambian la apariencia o comportamiento de un elemento DOM

COMPONENTES	Directivas con una plantilla
DIRECTIVA ESTRUCTURAL	Agrega o remueve elementos DOM para cambiar su presentación
DIRECTIVA DE ATRIBUTO	Cambia la apariencia o comportamiento de un elemento

Directives

DIRECTIVA ESTRUCTURAL → **Agrega o remueve elementos DOM para cambiar su presentación**

```
<ul>
      <li *ngFor = “let movie of movies”> {{movie}}</li>
</ul>
```

Iteración sobre la lista de películas

Directives

DIRECTIVA DE ATRIBUTO → **Cambia la apariencia o comportamiento de un elemento**

```
import { Directive, ElementRef, HostListener } from '@angular/core';
```
Importando Directive, ElementRef y HostListener

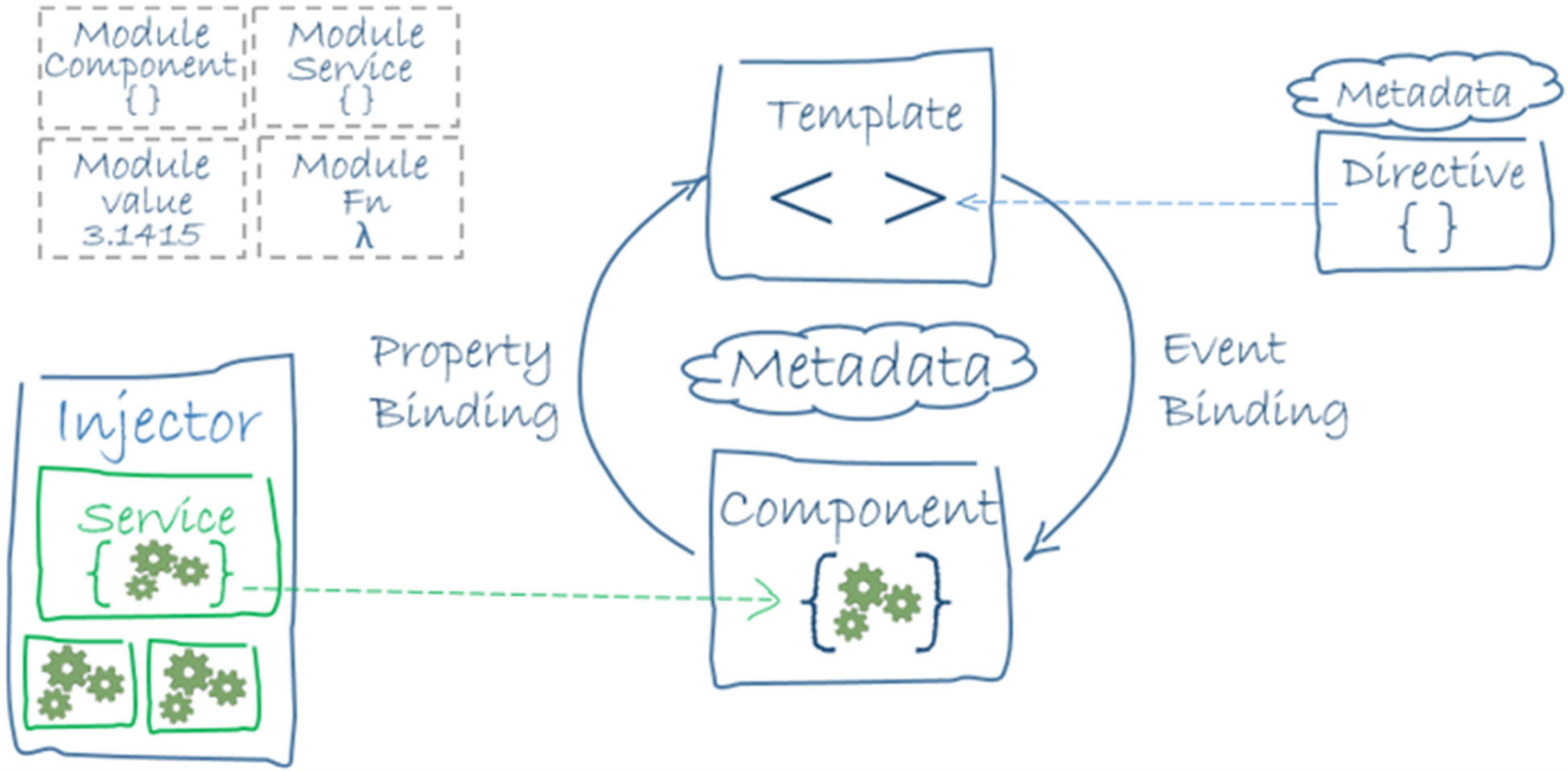
```
@Directive({
  selector: '[appBoldText]',
})
```
Metadata de directiva

```
export class BoldTextDirective {

  constructor(private elementRef: ElementRef) { }
```
Inyección ElementRef Para accesar a los elementos DOM

```
  @HostListener('mouseenter') onMouseEnter(){
     this.elementRef.nativeElement.style.fontWeight = 'bold';
  }
```
Colocar el texto en negrita Cuando el cursor pase

```
  @HostListener('mouseleave') onMouseLeave(){
     this.elementRef.nativeElement.style.fontWeight = null;
  }
}
```
Quitar negrita al texto

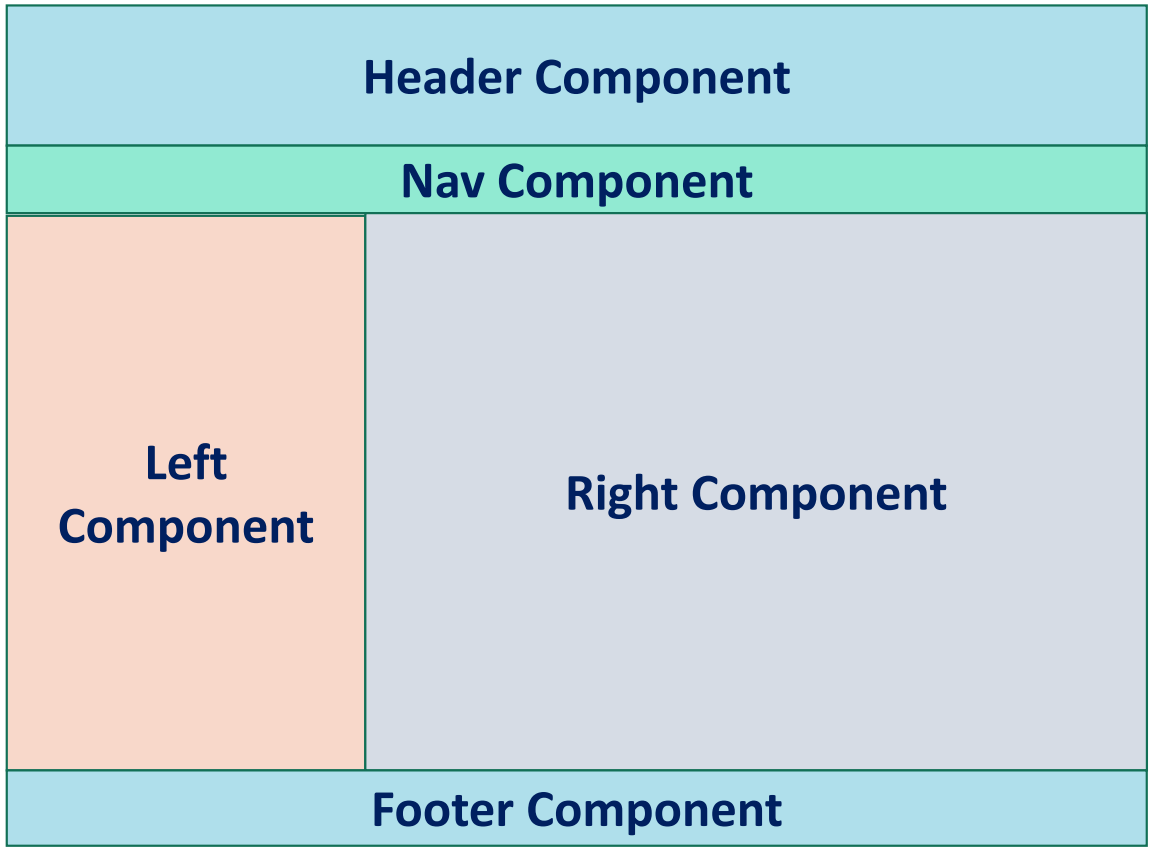
Arquitectura de Angular



Creando un ejemplo con angular - componentes

- ✓Crear un app llamada myapp02 con la siguiente estructura:

- ✓ng new myapp02

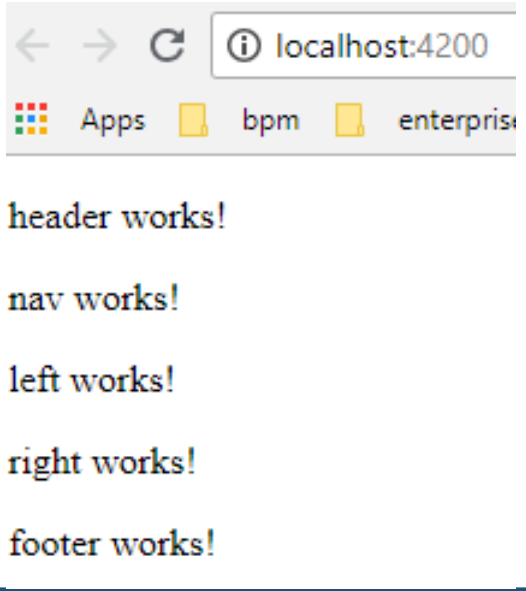


Creando un ejemplo con angular - componentes

1. Dentro de app, crear una carpeta components/common
2. Dentro de la carpeta common crear los componentes:
   header, nav, left, right, footer
3. Editar app.component.html

app.component.html

```
1  <app-header></app-header>
2  <app-nav></app-nav>
3  <app-left></app-left>
4  <app-right></app-right>
5  <app-footer></app-footer>
```



Creando un ejemplo con angular - componentes

- ✓Vamos a mejorar la presentación, usando Bootstrap
- ✓Ir a getbootstrap.com, copiar el CDN de bootstrap la ultima versión

jsDelivr

Skip the download with jsDelivr to deliver cached version of Bootstrap's compiled CSS and JS to your project.

Copy

```
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.mi
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.bundle.min.js" inte
```

Creando un ejemplo con angular - componentes

- ✓En http://getbootstrap.com/components/#navbar-default copiar el componente en el archivo nav.component.html

nav.component.html

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse
    <span class="navbar-toggler-icon"></span>
  </button>

    <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <ul class="navbar-nav mr-auto">
            <li class="nav-item active">
                <a class="nav-link" href="#">Home <span class="sr-only
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#">Link</a>
            </li>
            <li class="nav-item dropdown">
                <a class="nav-link dropdown-toggle" href="#" id="navba
          Dropdown
```

- ✓Probar los cambios en el app

Creando un ejemplo con angular - componentes

- ✓Código para footer.component.html

```
footer.component.html
1  <footer class="footer text-center">
2      <div> &copy; 2018 Copyright: Edwin Valencia
3      </div>
4  </footer>
```

- ✓Código para left.component.html

```
left.component.html
1  <div class="col-xs-12 col-sm-12 col-md-12 col-lg-4">
2      left works!
3  </div>
```

- ✓Código para right.component.html

```
right.component.html
1  <div clas="col-xs-12 col-sm-12 col-md-12 col-lg-8">
2      right works!
3  </div>
```

- ✓Código para app.component.html

```
app.component.html
1  <app-header></app-header>
2  <app-nav></app-nav>
3  <div class="row">
4      <app-left></app-left>
5      <app-right></app-right>
6  </div>
7  <app-footer></app-footer>
```

Creando un ejemplo con angular - componentes

- ✓Agregando estilos

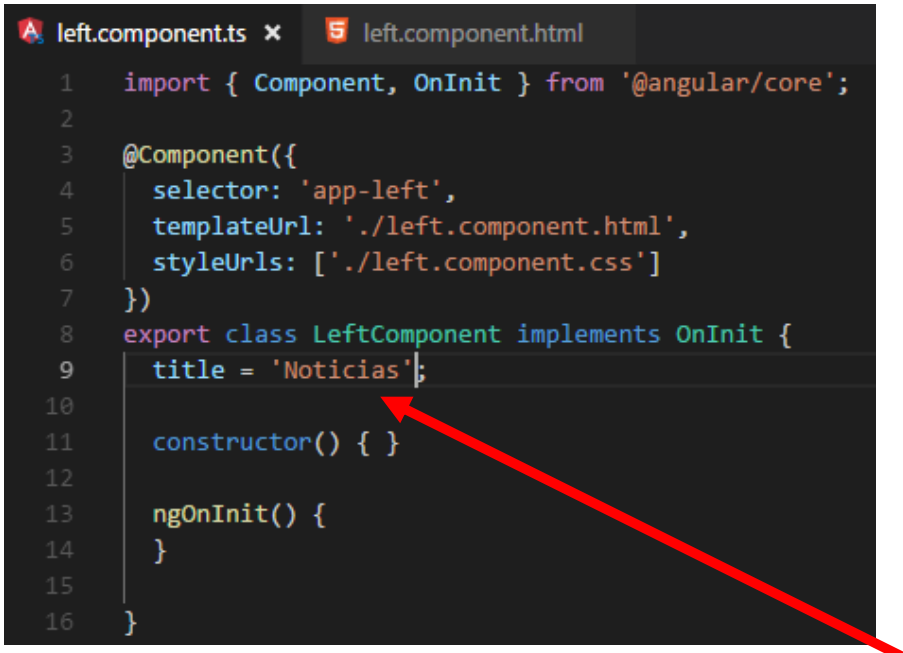
left.component.html

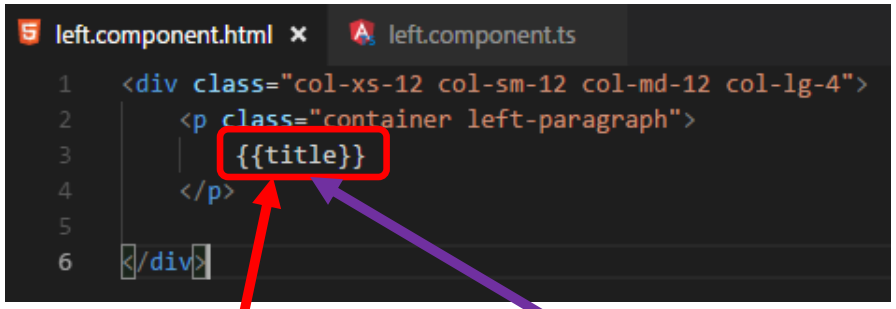
```
<div class="col-xs-12 col-sm-12 col-md-12 col-lg-4">
    <p class="left-paragraph container">
        left works!
    </p>

</div>
```

left.component.css

```
.left-paragraph {
    border: 1px solid red;
    width: 300px;
}
```

Creando un ejemplo con angular - interpolación





SINTAXIS DEL SISTEMA DE PLANTILLAS DE ANGULAR

INTERPOLACION

Creando un ejemplo con angular - interpolación

✓Usando la directiva *ngFor

left.component.ts

```ts
import {Component, OnInit} from '@angular/core';

@Component({
    selector: 'app-left',
    templateUrl: './left.component.html',
    styleUrls: ['./left.component.css']
})
export class LeftComponent implements OnInit{
    titulo = 'Noticias';
    items: Array <string> = ['Noticia 1', 'Noticia 2', 'Noticia 3'];

    constructor() {
    }
    ngOnInit() {    }
}
```

left.component.html

```html
<div class="col-xs-12 col-sm-12 col-md-12 col-lg-4 left-paragraph">
    <h2> {{titulo}}</h2>
    <ul>
        <li *ngFor="let item of items">{{item}}</li>
    </ul>

</div>
```

Creando un ejemplo con angular – más practica

- ✓ En el componente left y right organizar los atributos de las clases y verificar el uso del constructor()
- ✓ En el componente right vamos a crear un botón que permita mostrar u ocultar las noticias:

right.component.html

```
<div class="col-xs-12 col-sm-12 col-md-12 col-lg-8 right-paragraph">
    <button (click)='showNoticias()' class='btn'>
        {{estadoNoticias ? 'Ocultar Noticias': 'Ver Noticias'}}
    </button>
    <div *ngIf='estadoNoticias' class='myComponent'>
        <h2>{{title}}</h2>
        <ul>
            <li *ngFor='let item of noticias'>{{item}}</li>
        </ul>
    </div>

</div>
```

right.component.ts

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-right',
  templateUrl: './right.component.html',
  styleUrls: ['./right.component.css']
})
export class RightComponent implements OnInit {
    title: string;
    noticias: string[];
    estadoNoticias: boolean;

  constructor() {
    this.title = 'Noticias';
    this.noticias = ['Noticia 1', 'Noticia 2', 'Noticia 3'];
    this.estadoNoticias = false;
  }

  ngOnInit() {
  }

  showNoticias(){
    return this.estadoNoticias = !this.estadoNoticias;
  }

}
```

Creando un ejemplo con angular – más practica

- ✓Vamos a crear un formulario para agregar mas noticias

right.component.html

```
<!-- Formulario para crear noticias-->
<form (submit)='nuevaNoticia(noticia)'>
    <input #noticia type="text" class="form-control" name="noticia" placeholder="Nueva Noticia">
</form>

</div>
```

right.component.ts

```
nuevaNoticia(noticia){
  this.noticias.push(noticia.value);
  noticia.value = '';
  return false;
}
}
```

Creando un ejemplo con angular – mas practica

✓Manipulando el contenido

right.component.html ● app.module.ts

```
18      <div>
19          <h3>Modificando del contenido</h3>
20          <label for="title">Nuevo Contenido:</label>
21          <input type="text" name="title" placeholder="Nuevo contenido" [(ngModel)]="title">
22      </div>
```

➢Para que ngModel funcione debemos importar el modulo de manejo de formularios y registrarlo como import en app.module.ts

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

Creando un ejemplo con angular angular – mas practica

- ✓ Manejando servicios:
  - ➢ Crear un servicio post.service.ts
  - ➢ Los datos serán extraídos del servicio REST: https://jsonplaceholder.typicode.com/posts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';
import { HttpClientModule } from '@angular/common/http';
import { MiServicio } from './services/miservicio';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule, HttpClientModule
  ],
  providers: [MiServicio],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Observable, throwError } from 'rxjs';
import { retry, catchError } from 'rxjs/operators';
import { Post } from '../shared/post';


@Injectable()
export class MiServicio{

// Base url
baseurl = 'https://jsonplaceholder.typicode.com/posts';

constructor(private http: HttpClient) { }

// Http Headers
httpOptions = {
  headers: new HttpHeaders({"Content-Type": "application/json"})
}

// GET
GetIssues(): Observable<Post> {
  return this.http.get<Post>(this.baseurl)
  .pipe(
    retry(1),
    catchError(this.errorHandl)
  )
}

// GET
GetIssue(id): Observable<Post> {
  return this.http.get<Post>(this.baseurl +  id)
  .pipe(
    retry(1),
    catchError(this.errorHandl)
  )
}
```

Creando un ejemplo con angular – mas practica

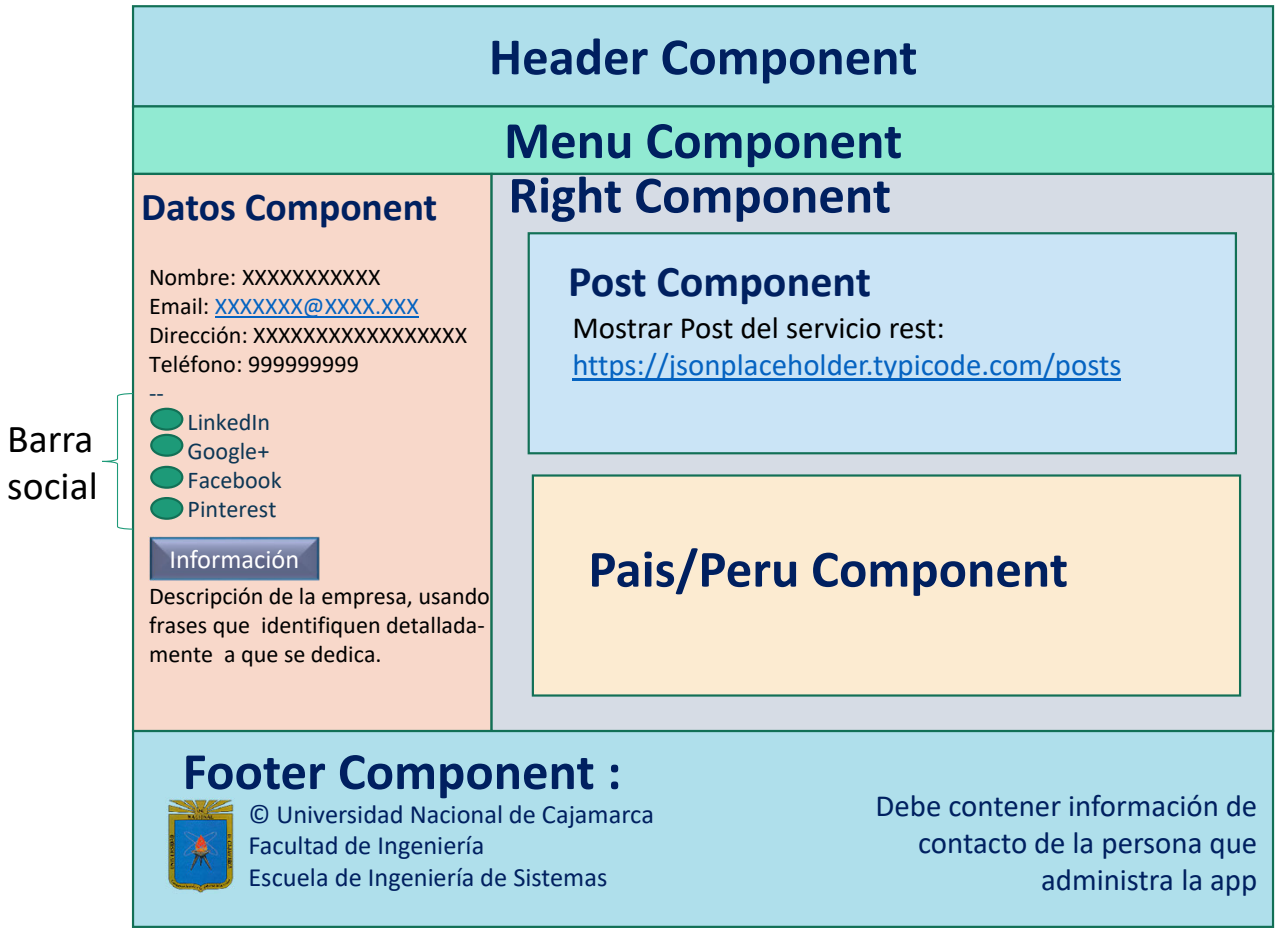
✓Viendo los servicios:

```
export class AppComponent implements OnInit{
  title = 'myApp01';
  IssuesList: any = [];

  ngOnInit() {
    this.loadPosts();
  }

  constructor(public miServicio: MiServicio){ }

   // Issues list
   loadPosts() {
    return this.miServicio.GetIssues().subscribe((data:{})=>{
      this.IssuesList = data;
    })
  }
}
```

```
<!--The content below is only a placeholder and can be replaced.-->
<div style="text-align:center">
    <h1>
        Welcome to {{ title }}!
    </h1>
</div>
<h1>Posts</h1>
<div *ngFor="let post of IssuesList">
    <h3>{{post.id}}: {{post.title}}</h3>
    <p>{{post.body}}</p>

</div>
```

Gracias por la atención

evalencia@unc.edu.pe



Barra social

Organizar un carrousel de imagenes

Incluir dos botones: país, peru, que permita ver el componente país o el componente peru, cuando se haga click.

Pais Component

Paises: En una tabla mostrar código a 3 digitos, nombre, Poblacion, Continente, utilizar el servicio rest:
https://restcountries.eu/rest/v2/all

Peru Component

Peru: Usando etiquetas html mostrar información de Perú, utilizar el servicio rest:
https://restcountries.eu/rest/v2/name/peru?fullText=true
Se debe mostrar la información mas importante