ANGULAR

# Curso básico programación *single-page application*

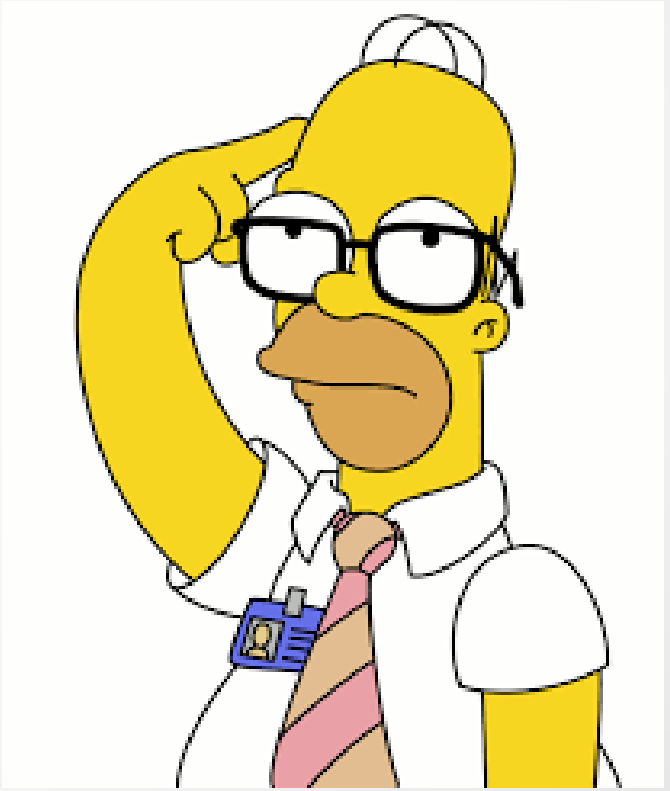
Jonatan Lucas
Jonatan.lucas@centic.es
@Jon_Lucas_

dualiza Bankia

Región de Murcia

# Resumen

- Introducción
- Arquitectura de una app
- Módulos
- Componentes
- Servicios e ID
- Routing
- HTTP request



# Introducción



# Single-page application

Consiste en una página web donde el contenido se carga en la primera petición.

- HTML
- CSS
- JavaScript
-
- Conseguimos más fluidez en nuestra web. Ejemplos de SPA pueden ser:
- Youtube
- Netflix
- Gmail

# ¿Qué es Angular 2?

Angular 2 es un *framework* de JavaScript para construir aplicaciones web y móviles desarrollada en lenguaje TypeScript.

- Pre-requisitos:
  - Conocimientos de desarrollo web con HTML y JavaScript.
  - Programación con TypeScript.
- Pre-instalado:
  - Node.js versiones superiores a 8.x o 10.x junto con npm
  - Editor de texto, se recomienda Visual Studio Code

# TypeScript

- TypeScript nos permite escribir aplicaciones en JavaScript de una forma más sencilla.
- Es un lenguaje orientado a objetos.
- Pequeñas diferencias con JS.

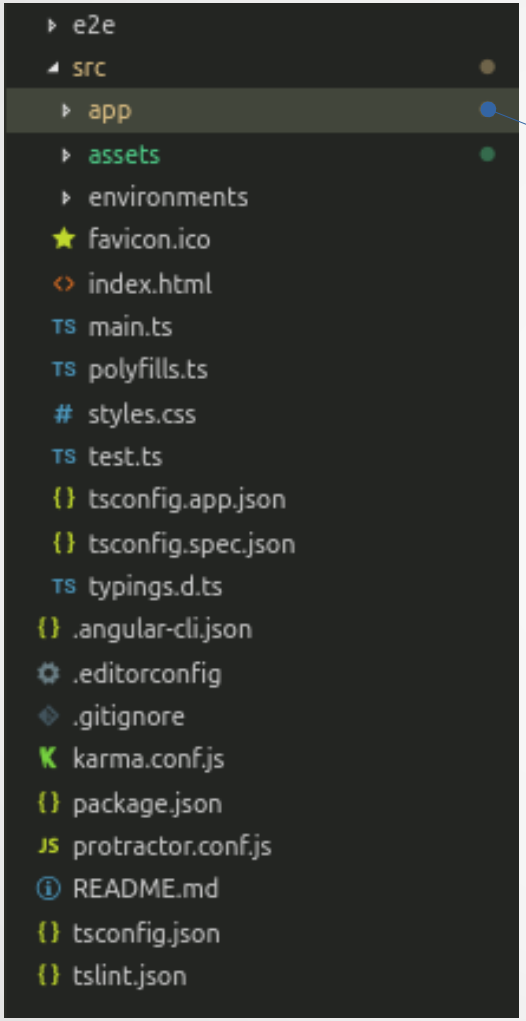
# Install time

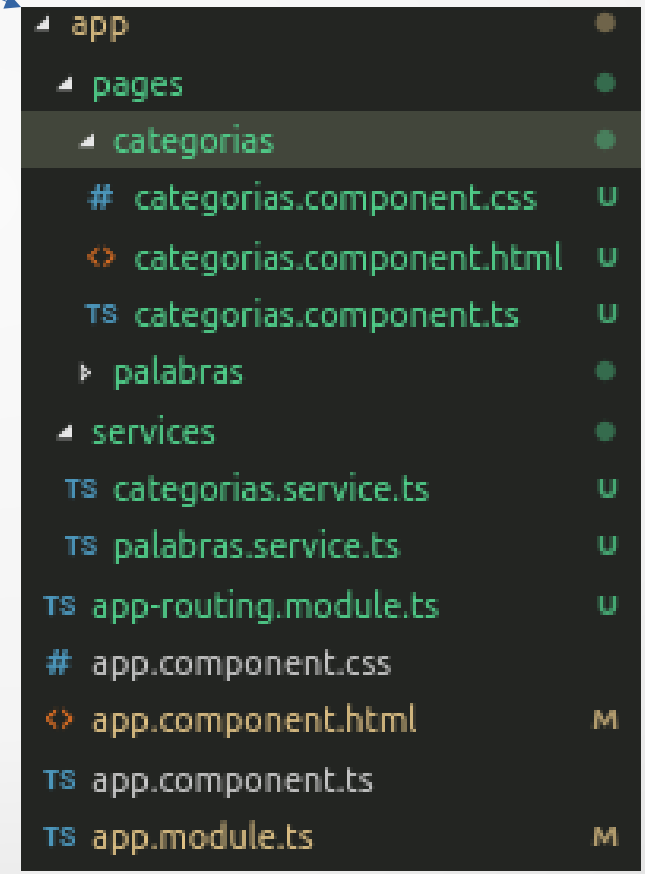


# Arquitectura de una app

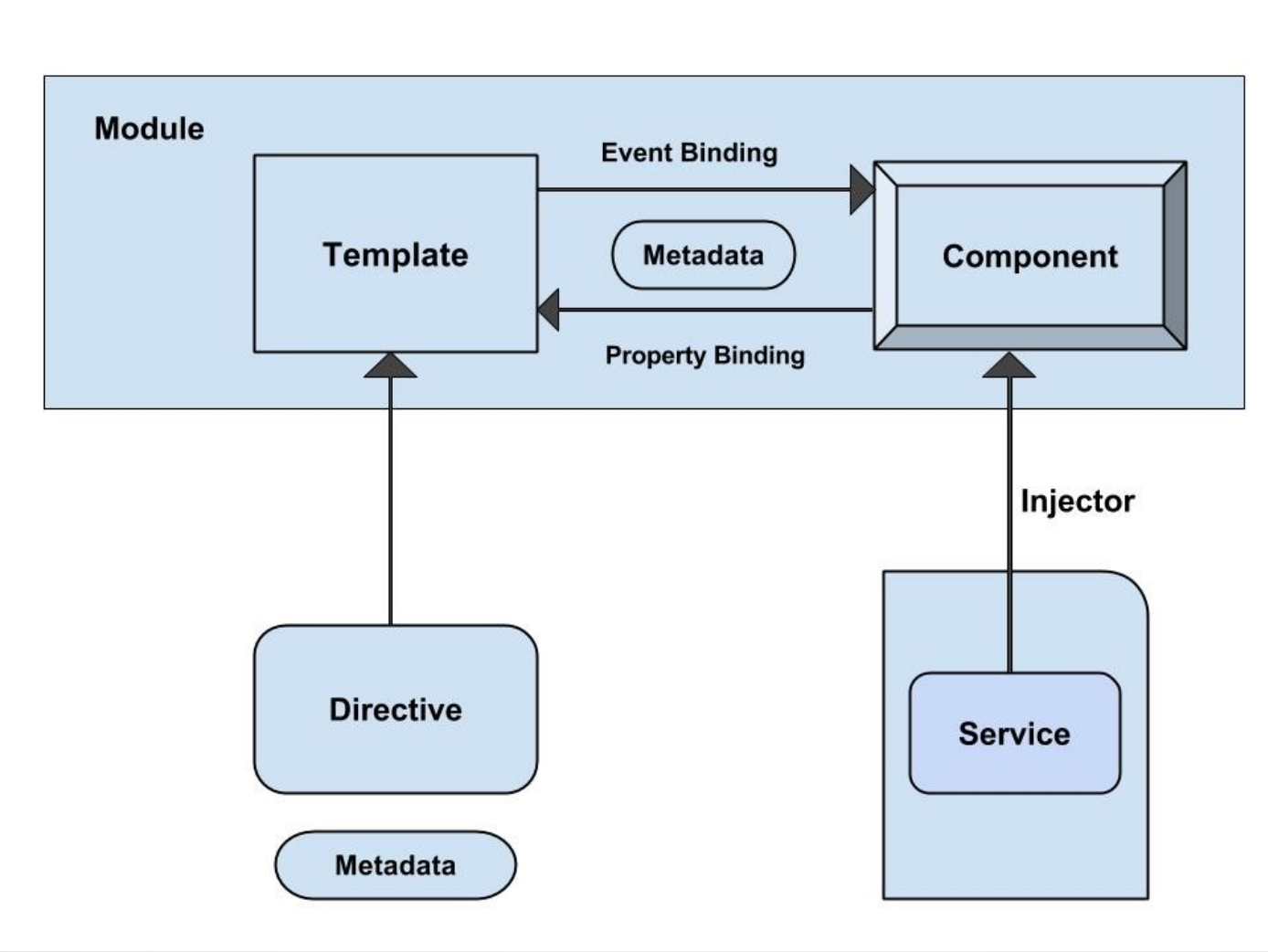


# Arquitectura app



# Arquitectura app



# Módulos



# Módulos

- Las aplicaciones en Angular son modulares.
- Posee su propio sistema de modulación, NgModules
- Cada módulo puede contener componentes, servicios y otros archivos de código necesario.
- Al menos se debe contener un módulo por app.
- Se encargar de lanzar todos los archivos necesarios para nuestra aplicación.

# Metadatos módulos

src/app/app.module.ts

```
import { NgModule }      from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
@NgModule({
  imports:      [ BrowserModule ],
  providers:    [ Logger ],
  declarations: [ AppComponent ],
  exports:      [ AppComponent ],
  bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

# Componentes



# Componentes

- Definen la lógica de la aplicación.
- Angular crea, actualiza y destruye componentes a medida que el usuario se mueve a través de la aplicación.
- Cada vista de Angular esta compuesta por:
  - Un fichero HTML para la plantilla.
  - Un fichero TS para el componente.
  - Un fichero CSS para el diseño.

```
ng generate component name_component
```

# Componentes

src/app/hero-list.component.ts (metadata)

```
@Component({
  selector:    'hero-list',
  templateUrl: './hero-list.component.html',
  providers:  [ HeroService ]
})
export class HeroListComponent implements OnInit {
/* . . . */
}
```

src/app/hero-list.component.ts (class)

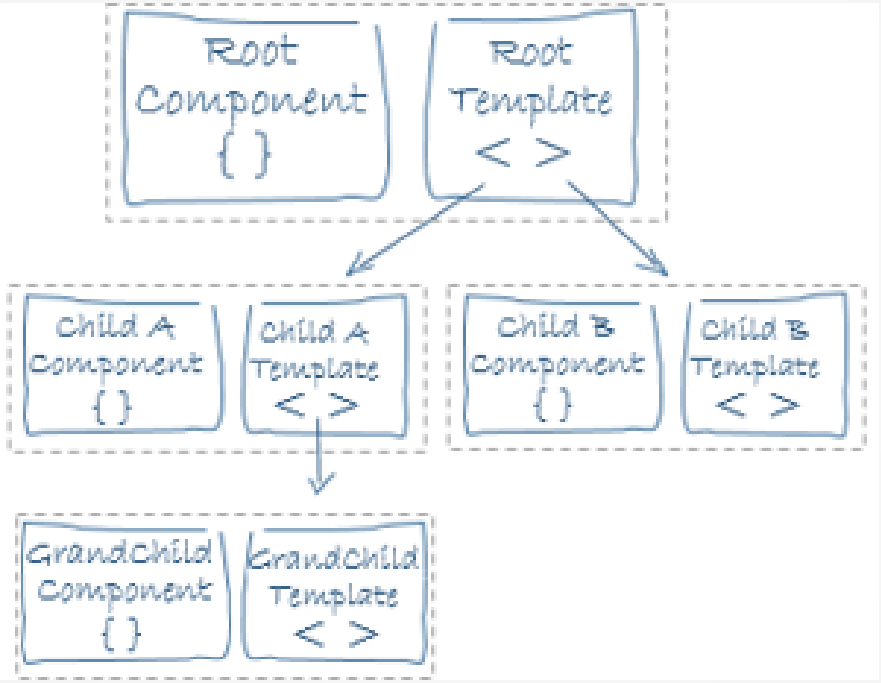
```
export class HeroListComponent implements OnInit {
  heroes: Hero[];
  selectedHero: Hero;

  constructor(private service: HeroService) { }

  ngOnInit() {
    this.heroes = this.service.getHeroes();
  }

  selectHero(hero: Hero) { this.selectedHero = hero; }
}
```

# Plantillas y vistas



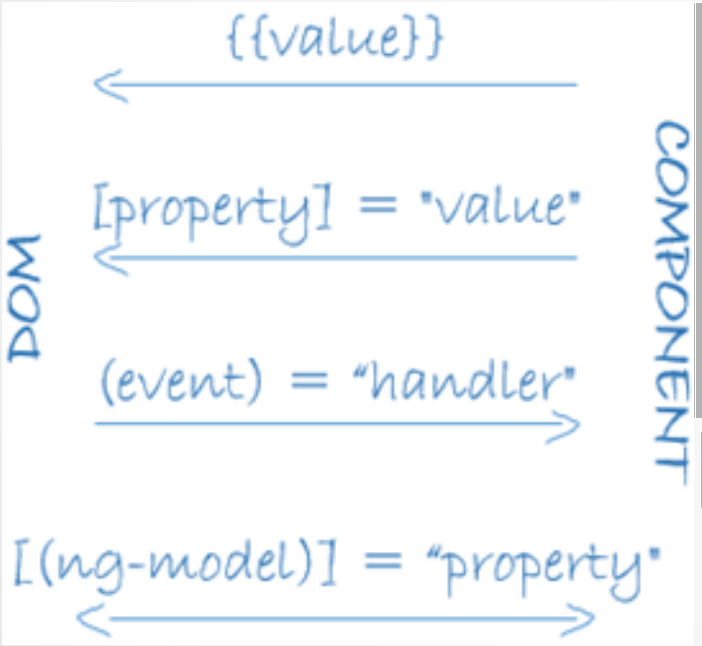
# Sintaxis plantilla

```
<h2>Hero List</h2>

<p><i>Pick a hero from the list</i></p>
<ul>
  <li *ngFor="let hero of heroes" (click)="selectHero(hero)">
    {{hero.name}}
  </li>
</ul>

<app-hero-detail *ngIf="selectedHero" [hero]="selectedHero"></app-hero-detail>
```

# Data binding




```
<li>{{hero.name}}</li>

<app-hero-detail
[hero]="selectedHero"></app-hero-detail>

<li (click)="selectHero(hero)"></li>
```

```
<input [(ngModel)]="hero.name">
```

# Directivas

- Se usan para modificar el DOM y proporcional a la plantilla de HTML un aspecto más dinámico

```
<li *ngFor="let hero of heroes"></li>

<app-hero-detail *ngIf="selectedHero"></app-hero-detail>
```

- *ngFor: muestra una etiqueta <li> por cada objeto heroe del array de heroes.
- *ngIf: incluye el componente solo si existe el objeto *selectedHero*.

# Servicios e DI



# Servicios e ID

- Es una clase amplia que abarca cualquier valor, función o característica que necesita una app.
- Angular distingue los componentes de los servicios para mejorar la modularidad y la reutilización.
- Los services se encargan sobre todo de recuperar datos del servidor, validar la entrada de usuarios o iniciar sesión directamente.
  - Para implementar esta lógica en los componentes Angular utiliza la inyección de dependencias

```
ng generate service name_service
```

# Servicios e ID

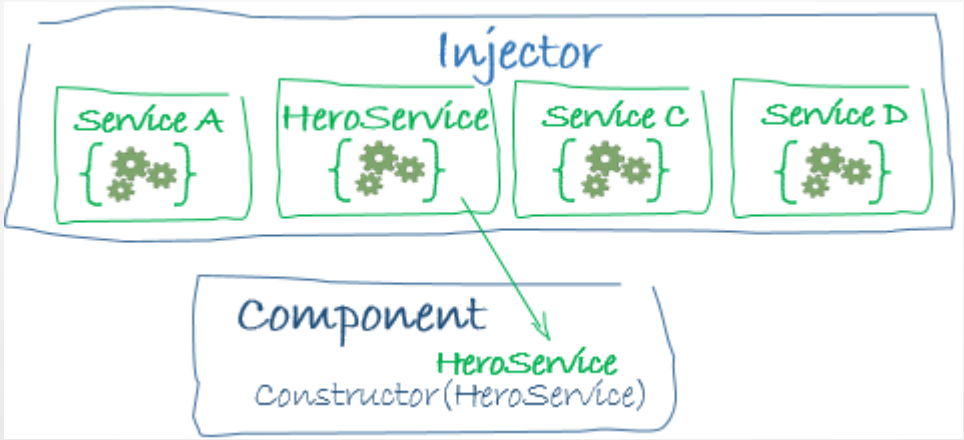
```
@Injectable({})
export class HeroService {
  private heroes: Hero[] = [];

  constructor(
    private backend: BackendService,
    private logger: Logger) { }

  getHeroes() {
    this.backend.getAll(Hero).then( (heroes: Hero[]) => {
      this.logger.log(`Fetched ${heroes.length} heroes.`);
      this.heroes.push(...heroes); // fill cache
    });
    return this.heroes;
  }
}
```

# Servicios e ID

- Enlaza los componentes con los servicios.
- Cuando Angular crea una instancia de una clase componente, determina qué servicios u otras dependencias necesita ese componente al observar los parámetros del constructor.
- Cuando un componente depende de un servicio, el inyector crea una instancia de ese servicio.



# Http Request



# Http Request

- Las aplicaciones de Angular se comunican con los servidores a través de los *services*.
- Para trabajar con las peticiones utilizaremos el objeto de *HttpClient* que se basa en la interfaz *XMLHttpRequest*.
- Un método de *HttpClient* no comienza su solicitud HTTP hasta que llamemos al método *.subcribe()*
- *HttpClient* admite solicitudes de GET, POST, PUT y DELETE.

# Http Request

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

```
export class HeroService {
  private heroes: Hero[] = [];

  constructor(
    private backend: BackendService,
    private logger: Logger) { }

  getHeroes() {
    this.backend.getAll(Hero).then( (heroes: Hero[]) => {
      this.logger.log(`Fetched ${heroes.length} heroes.`);
      this.heroes.push(...heroes); // fill cache
    });
    return this.heroes;
  }
}
```

# Http Request

```
import { HttpHeaders } from '@angular/common/http';

const httpOptions = {
  headers: new HttpHeaders({
    'Content-Type':  'application/json',
    'Authorization': 'my-auth-token'
  })
};
```

# Routing



# Routing

- El archivo de rutas indica a Angular que componentes cargar según la ruta que visitemos.
- Podremos pasar parámetros sin problema.
- El objeto encargado de realizar las rutas para Angular es *Router*.
- Pero para poder trabajar con este objeto y rutas en Angular debemos colocar esta etiqueta en el index.html

```
<base href=”/”>
```

- En el AppComponent debemos sustituir el código HTML por la siguiente etiqueta:

```
<router-outlet></router-outlet>
```

# Routing

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

Angular te genera el siguiente código, pero no es el correcto a usar.

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';

@NgModule({
  imports: [
    CommonModule
  ],
  declarations: []
})

export class AppRoutingModule { }
```

# Routing

```
import { NgModule }             from '@angular/core';
import { RouterModule, Routes } from '@angular/router';

import { HeroesComponent } from ‘./heroes/heroes.component’;
import { HeroDetailComponent  } from
‘./heroDetail/heroDetail.component’;
import { PageNotFoundComponet } from ‘./page/page.component’;

const    routes:   Routes =  [
    {   path: ‘heroes’, component:  HeroesComponent  },
    {   path: ‘heroes/:id’, component:  HeroDetailComponent   },
    {   path: ‘’, redirecTo:  ‘/heroes’, pathMatch:  ‘full’  },
    {   path: ‘**’, component:  PageNotFoundComponent  }
];

@NgModule({
  exports: [ RouterModule ],
  imports: [ RouterModule.forRoot(routes) ]
})
export class AppRoutingModule {}
```

# Routing

- Angular tiene dos maneras diferentes de navegar entre sus vistas:
  - Con el atributo routerLink en el HTML.
  -
  -
  -
  -
  -
  - El objeto *router* desde el componente.

```
<a routerLink="/categorias">Categorias</a>
<a routerLink="/palabras">Palabras</a>
```

# Routing

- Extraer parámetros de la url

```
{ path:  'heroes/:id', component:  HeroDetailComponent  }
```

```
import { ActivatedRoute } from '@angular/router';

…

constructor(…, private activatedRoute: ActivatedRoute ) {
   activatedRoute.params.subscribe( params => {
        this.id = +params['id'];
   });
   …
}
…
```

# Bibliografía


- Documentación oficial de Angular. Recuperado de https://angular.io/docs
- Programacion JJE (14 May 2018). Que es SPA (Single-page Application). Medium. Recuperado de https://medium.com/@programacionjje/que-es-spa-single-page-application-4dbd3694fac9
- Wikipedia. Single-page application. Recuperado de https://es.wikipedia.org/wiki/Single-page_application