Ejemplo 2 guiado

Instalar componente navbar en shared (usar navbar-brand)

Instalar componente home como un Jumbotron.

Instalar componente footer.

Instalar componente about.

# Rutas:

Permite navegar entre componentes.

En archivo app.routes.ts, poner completar el array de rutas con {path: ‘’, component: \_} e importar los componentes.

Confirmar que app.config.ts, coincide el nombre de la constante exportada con la usada en app.routes.ts.

En app.component.ts, importar RouterOutlet.

En app.component.html, incluir <router-outlet>.

En navbar.component.ts, importar RouterLink y RouterLinkActive.

En navbar.component.html, incluir en los enlaces routerLink=”<URL\_COMPONENTE>” y routerLinkActive=”active”.

# Interpolación en atributos:

La interpolación en los atributos se puede hacer de dos formas:

<img class="card-img-top" src="{{ rutaLogo(item.imagen) }}"

La forma más aceptada:

<img class="card-img-top" [src]="rutaLogo(item.imagen)"

# Servicios:

Nueva carpeta services dentro de app. Crear archivo de servicio con el snippet de servicio “ng-service”, cambiando el nombre de la clase. Importar y meter en providers de app.config.ts

Incluir como argumento-atributo del constructor. (constructor (private \_miServicio:NombreServicio))

Meter el acceso a la información en el servicio, con los métodos de accesos (exportando).

Utilizar los métodos de acceso del servicio en el componente (this.\_miServicio.getData())

# Rutas con parámetros

Crear componente para elemento único, con el id en la URL.

Usar ActivatedRoute para leer params mediante subscribe:

*constructor*(private *actRoute*:ActivatedRoute,

private *\_unService*: UnService){

actRoute.params.subscribe( *p* *=>* {

this.att = \_unService.getAtt(Number(p['id']));

# Variable para elemento del DOM

# “Listeners”

**En Navbar**

**<input> =>** (keyup.enter) => “funcion(argumento.value)” #argumento

**<button> =>** (click) => “funcion(argumento.value)”

Para evitar que el botón envíe el formulario, cambiar el type=”submit” a type=”button”

Para redirigir a otra URL:

* añadir a rutas de app.router.ts
* En el componente, importar Router, y usar [Router].navigate([‘/ruta’, variable])

# Argumentos padre->hijo (@input)

En el archivo TS del padre:

importar el componente de la tarjeta e incluirlo en imports.

En el archivo HTML del padre:

quitar código repetido de tarjeta, para ponerlo en el HTML del hijo.

Añadir etiqueta para app-componentehijo, poniendo las clases del primer div de la tarjeta “card”

<app-componentehijo class="card col-lg-2 col-sm-6 col-md-4"

[atributo-hijo]=”valor/variable local”>

incluir el ngFor para que se repita:</app-componentehijo>

<app-componentehijo \*ngFor="let item of items; let i = index" [item]="item" [i]="i" ></app-componentehijo>

En el archivo de componente del hijo de la tarjeta:

declarar variable para el item (lenguaje) y para la i.

En el archivo HTML del hijo:

poner el código que se quitó del padre, eliminando el primer div, que ahora se sustituye por la etiqueta app-componentehijo colocada en el padre.

# Argumentos hijo->padre (@output)

TS de componente padre

* importar el componente hijo para usar su etiqueta (selector).

TS de componente hijo

* importar Output y EventEmitter
* añadir un atributo con el decorador @output de tipo EventEmitter y tipo de respuesta (number, string, …)  
  @Output() variable\_emisora:EventEmitter<number>;
* En el constructor, inicializar la variable:  
  this.variable\_emisora = new EventEmitter();
* Método para emitir el evento:  
  botonPulsado(i:number) { this.variable\_emisora.emit(i);

HTML de componente padre

* En la etiqueta del hijo, añadir un listener con el evento declarado en Output.  
  <app-hijo **(variable\_emisora)=”metodo\_padre($event)”** …. let i = index…>

HTML de componente hijo

* En el botón que emitirá el evento:
* <button class="..." (click)="botonPulsado(i)">Ver más...</button>