En esta primera aplicación veremos qué son los componentes, módulos, directivas estructurales en Angular, etc.

1. Componentes y Módulos.

Toda aplicación de angular tiene un módulo raíz en donde defines el componente principal a cargar. En la aplicación Angular por defecto, el módulo raíz es definido dentro de **app.module.ts**. Cuando carga el **AppModule**, éste revisa cuál es el componente incluido y carga ese módulo. El componente AppComponent es definido en el archivo **app.component.ts**.

Un componente es definido usando el decorador **@Component**. Dentro del decorador @Component, puedes definir el componente selector, el componente template, y el style relacionado.

Los **componentes** en Angular son clases normales con un decorador específico.

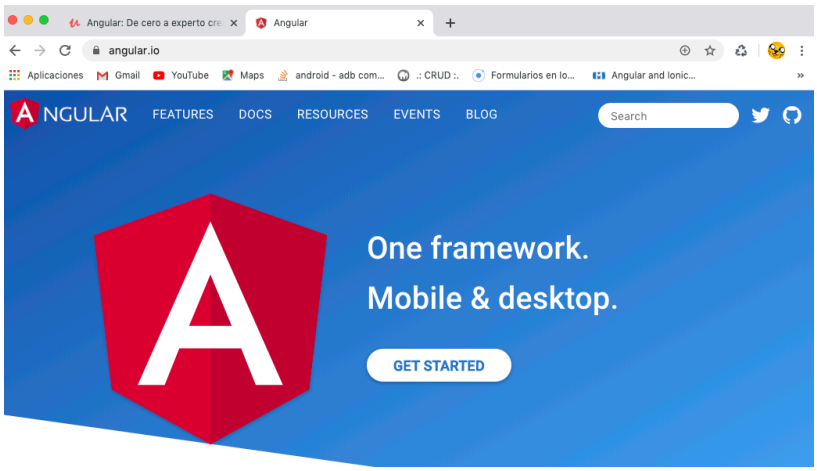
2. Las directivas estructurales.

Las **directivas estructurales** corresponden a elementos en el HTML que permiten añadir, manipular o eliminar elementos del DOM. Estos elementos, en forma de atributos, se aplican a elementos huéspedes. Al hacer esto, la directiva hace lo que debe hacer sobre el elemento huésped y sus elementos hijos. Estas directivas son fácilmente reconocibles debido a que están antecedidas por un asterisco (*) seguido del nombre de la directiva.

```
<div *ngIf="variable">
    <p>Este es un párrafo</p>
</div>
```

Acción de la directiva *ngIf

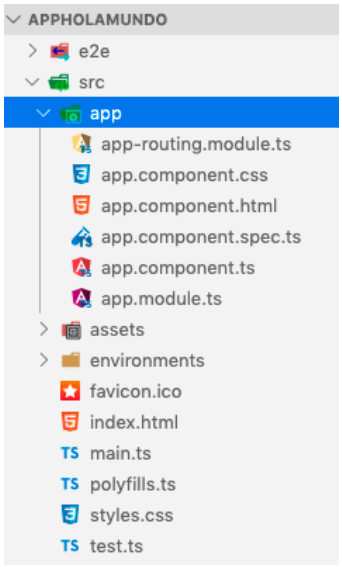
En el sitio angular.io tenemos documentación de cualquier elemento que queramos utilizar en Angular:



1. Inicialización aplicación **appHolaMundo**: ejecutamos el comando de node ng new.

```
iMac-de-Jose:appHeroes jsersan$ cd ..
iMac-de-Jose:htdocs jsersan$ ng new appHolaMundo
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE appHolaMundo/README.md (1030 bytes)
CREATE appHolaMundo/.editorconfig (246 bytes)
CREATE appHolaMundo/.gitignore (631 bytes)
CREATE appHolaMundo/angular.json (3641 bytes)
CREATE appHolaMundo/package.json (1300 bytes)
CREATE appHolaMundo/tsconfig.json (543 bytes)
CREATE appHolaMundo/tslint.json (1953 bytes)
CREATE appHolaMundo/browserslist (429 bytes)
```

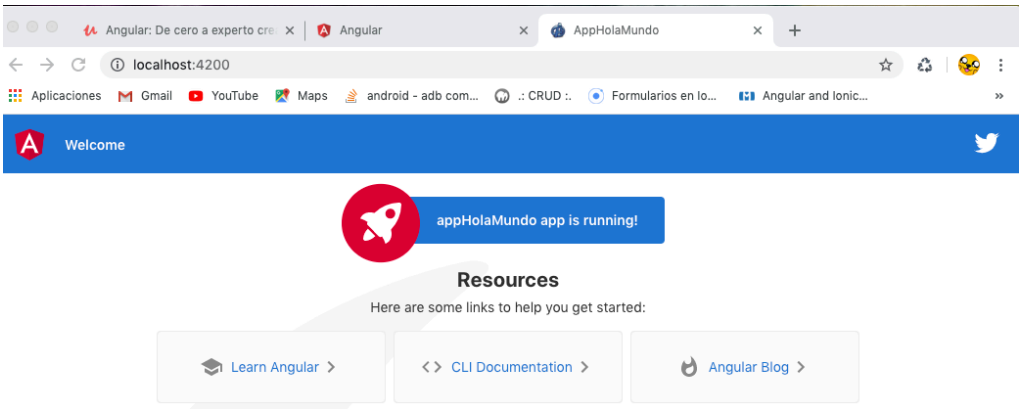
Lo abrimos en Visual Studio Code:



Lanzamos el proyecto:

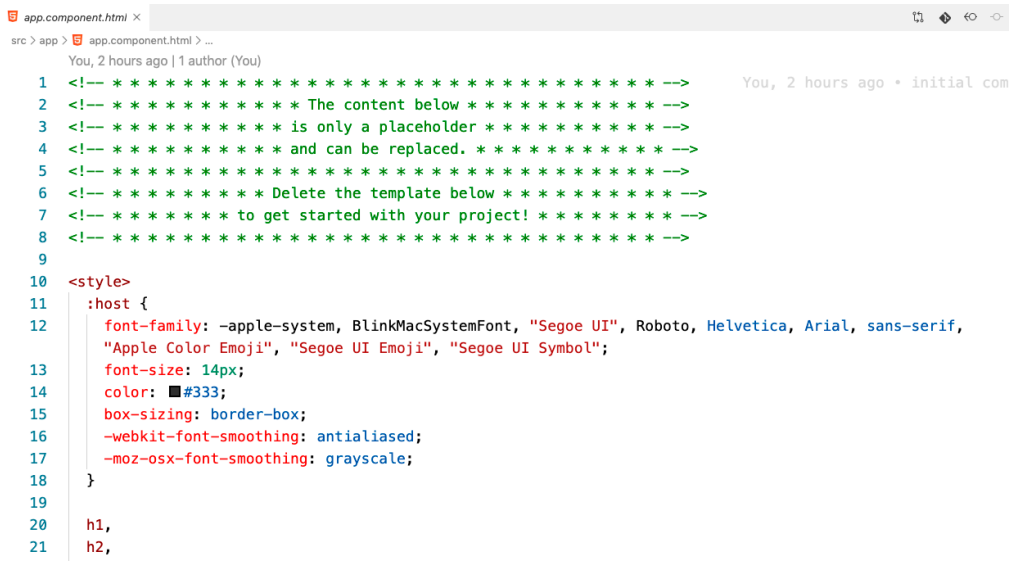
$ ng serve -o

Lanza el proyecto en el navegador por defect.

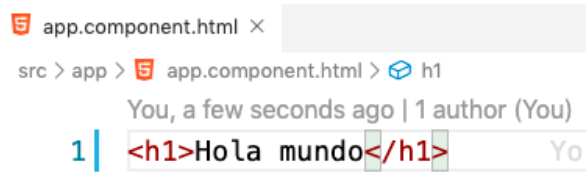


Modificamos app.component.html y app.component.ts:

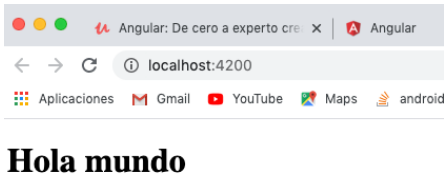
. app.component.html:



Lo borramos todo, y ponemos *"Hola mundo"*



Resultado:



Si modificamos:

El resultado:

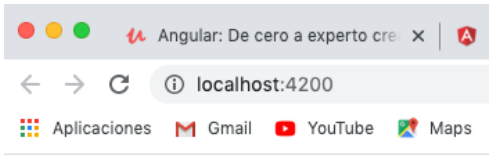
Definimos unas variables en el fichero typescript **app.component.ts**:

```
app.component.html   app.component.ts
src > app > app.component.ts > AppComponent > apellidos
    You, a few seconds ago | 1 author (You)
1   import { Component } from '@angular/core';
2
    You, a few seconds ago | 1 author (You)
3   @Component({
4     selector: 'app-root',
5     templateUrl: './app.component.html',
6     styleUrls: ['./app.component.css']
7   })
8   export class AppComponent {
9
10    nombre = 'Txema';
11    apellidos = 'Serrano Sánchez';      You,
12  }
```

Modificamos el template HTML **app.component.html**:

```
app.component.html ●   app.component.ts
src > app > app.component.html > ul
    You, a few seconds ago | 1 author (You)
1   <h1>Hola mundo</h1>
2
3   <ul>
4       <li>Nombre:{{ nombre }}</li>
5       <li>Apellidos: {{ apellidos }}</li>
6   </ul>      You, a few seconds ago • Uncommit
```

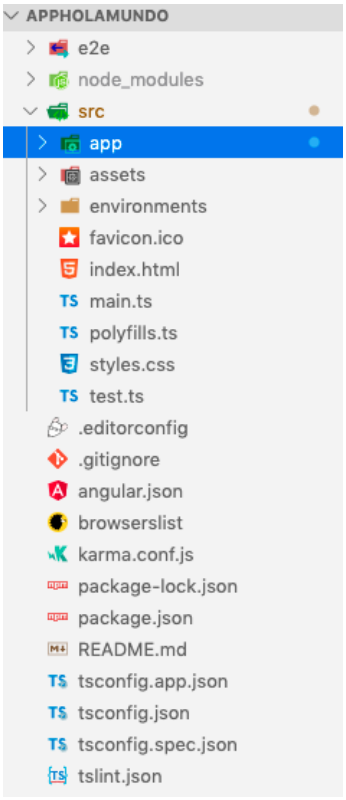
Resultado:



Hola mundo

- Nombre:Txema
- Apellidos: Serrano Sánchez

Estructura del proyecto:



- **e2e**: manejo de las pruebas de extremo a extremo (end to end). Si no se realizan pruebas unitarias no es necesario trabajar con ella.
- **node_modules**: librerías de paquetes de node. SE instalan los nodos en función del archivo package.json.
- **.gitignore**: es para indicar qué archivos y carpetas no quiero subir al repositorio de GitHub.
- **Package-lock.son**: indica cómo fue creado el **package.json**.
- **Package.json**: tiene información de todas las dependencias de **node** del proyecto.
- **README.md**: información genérica del proyecto y será la información que sale en la portada del mismo en GitHub.
- **tsconfig.json:** le indica a typescript como trabajar.
- **tslint.json**: nos obliga a escribir un código más claro de typescript.
- **src**: carpeta source del proyecto:
- **app:** la aplicación de angular.
- **index.html:** página web principal
- **assets:** recursos estáticos (imágenes, archivos JSON, etc.)

El archivo **index.html:**

app.component.html | index.html | app.component.ts

src > index.html > ...

You, 2 hours ago | 1 author (You)

```
1  <!doctype html>      You, 2 hours ago • initial commit
2  <html lang="en">
3  <head>
4    <meta charset="utf-8">
5    <title>AppHolaMundo</title>
6    <base href="/">
7    <meta name="viewport" content="width=device-width, initial-scale=1">
8    <link rel="icon" type="image/x-icon" href="favicon.ico">
9  </head>
10 <body>
11   <app-root></app-root>
12 </body>
13 </html>
```

La información de app-root se encuentra en el archivo de typescript:

app.component.html | app.module.ts | app.component.ts

src > app > app.component.ts > AppComponent > apellidos

You, 22 minutes ago | 1 author (You)

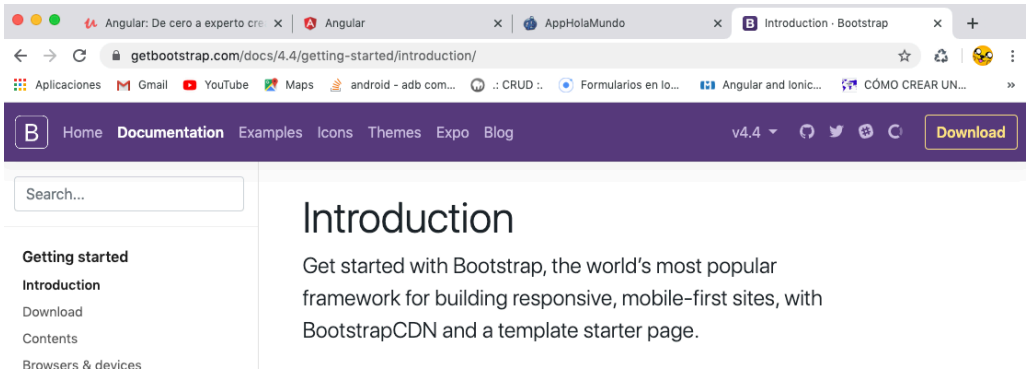
```
1  import { Component } from '@angular/core';
2
```

You, 22 minutes ago | 1 author (You)

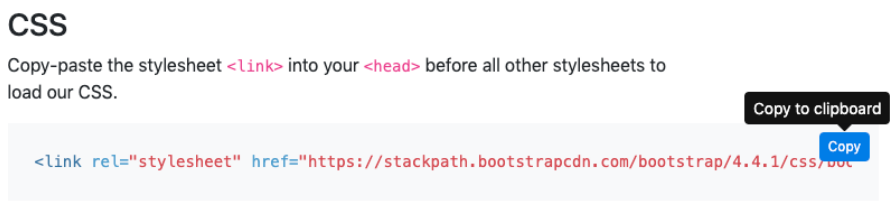
```
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9
10   nombre = 'Txema';
11   apellidos = 'Serrano Sánchez';      You,
12 }
```

2. Utilizando **BootStrap**:

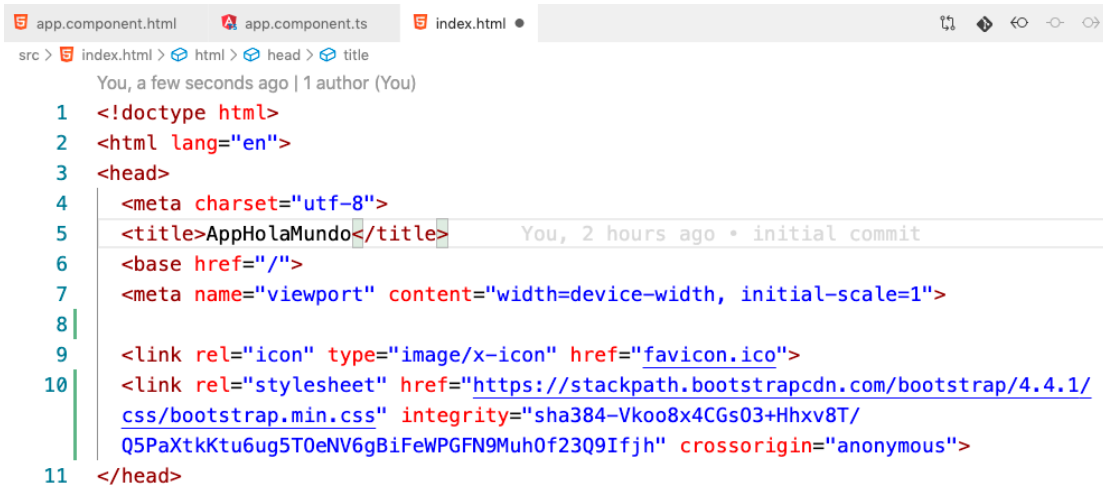
Vamos a la página de Bootstrap:



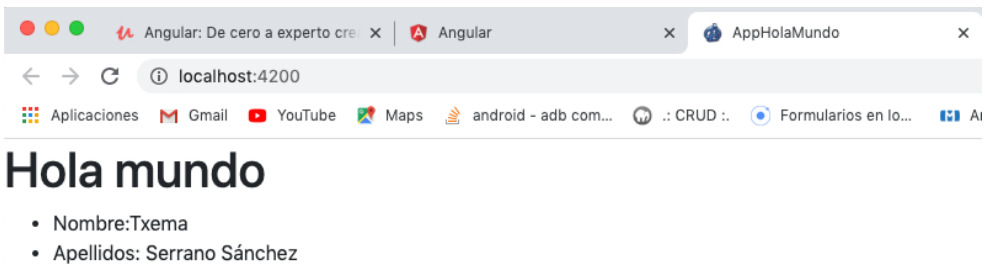
Vamos a esta parte:



Lo copio en **index.html**:



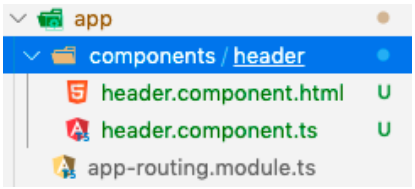
Guardo los cambios y ya se nota el cambio de apariencia (aunque no lo parezca):



Creamos un nuevo componente **header** que estará en la carpeta header:

```
iMac-de-Jose:appHolaMundo jsersan$ ng g c components/header -is --spec=false
Option "spec" is deprecated: Use "skipTests" instead.
CREATE src/app/components/header/header.component.html (21 bytes)
CREATE src/app/components/header/header.component.ts (242 bytes)
UPDATE src/app/app.module.ts (486 bytes)
iMac-de-Jose:appHolaMundo jsersan$
```

Resultado:



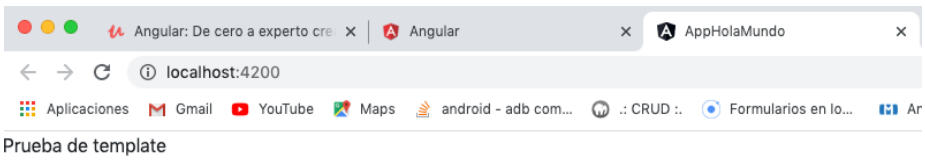
Así en **header.component.ts** cambio el *templateURL* general por uno específico como un template literal:

```
src > app > components > header > header.component.ts > HeaderComponent
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-header',
5    //templateUrl: './header.component.html',
6    template: `Prueba de template`,
7    styles: []
8  })
9  export class HeaderComponent implements OnInit {
10
11   constructor() { }
12
13   ngOnInit() {
14   }
15
16 }
```

Para visualizarlo debemos cambiar **app.component.html**

```
app.component.html ●   app.component.ts   inde
src > app > app.component.html > app-header
      Unsaved changes (cannot determine recent
1   <app-header></app-header>
```

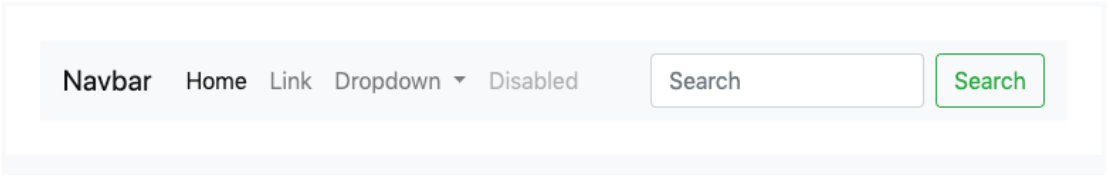
El resultado:



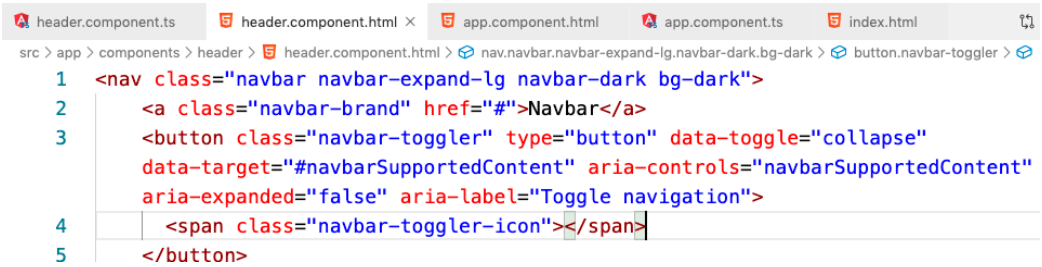
3. **TemplateURL**: separando el HTML del componente.

3.1 Header:

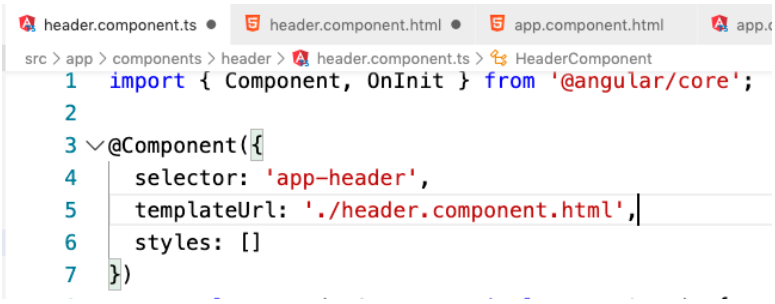
Crearemos por separado el *header, body* y el *footer*. Para el *header*, le pondremos un navbar copiado de **bootstrap**:



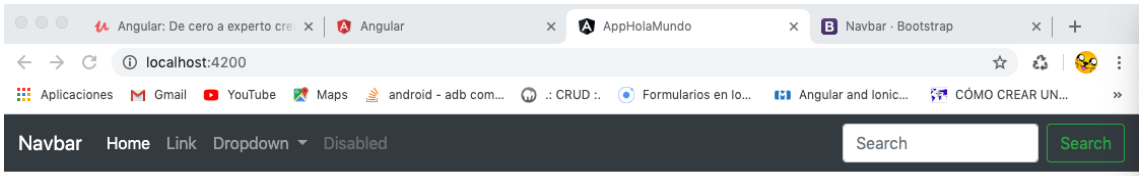
Lo copiamos en el archivo **header.component.html:**



Dejamos **header.component.ts** así:



Resultado:

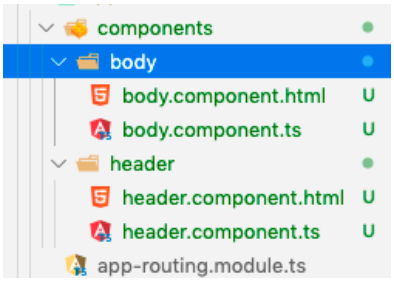


3.2 Body:

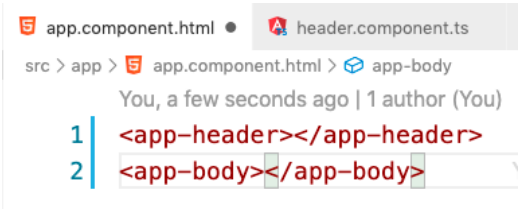
Creamos un componente **body.component**:

```
iMac-de-Jose:appHolaMundo jsersan$ ng g c components/body -is --spec=false
Option "spec" is deprecated: Use "skipTests" instead.
CREATE src/app/components/body/body.component.html (19 bytes)
CREATE src/app/components/body/body.component.ts (236 bytes)
UPDATE src/app/app.module.ts (571 bytes)
iMac-de-Jose:appHolaMundo jsersan$
```

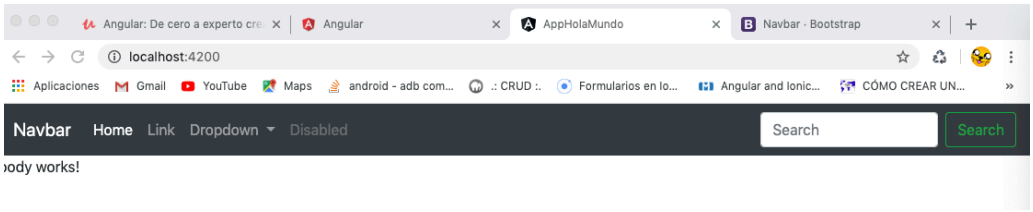
Estructura:



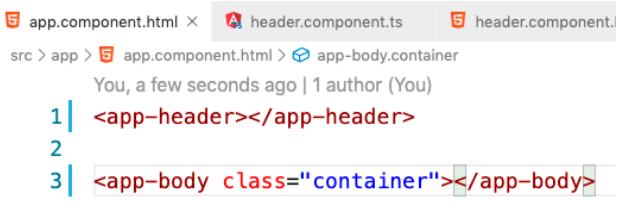
Modifico **app.component.html**:



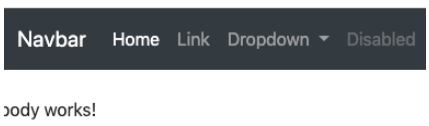
Resultado:



Le puedo poner una clase de bootstrap para darle más espacio al body respecto al header:



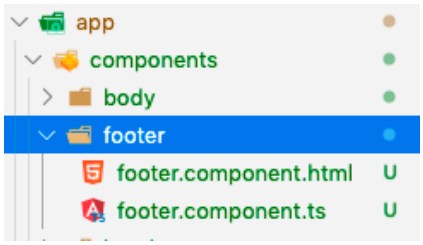
Resultado:



3.3 Footer:

```
iMac-de-Jose:appHolaMundo jsersan$ ng g c components/footer -is --spec=false
Option "spec" is deprecated: Use "skipTests" instead.
CREATE src/app/components/footer/footer.component.html (21 bytes)
CREATE src/app/components/footer/footer.component.ts (242 bytes)
UPDATE src/app/app.module.ts (664 bytes)
iMac-de-Jose:appHolaMundo jsersan$
```

La estructura es:



Esto quedará reflejado en **app.module.ts**:

app.component.html | app.module.ts | header.component.ts | header.component.html | app.co

src > app > app.module.ts > ...

You, a few seconds ago | 1 author (You)

```
1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } from '@angular/core';
3
4  import { AppRoutingModule } from './app-routing.module';
5  import { AppComponent } from './app.component';
6  import { HeaderComponent } from './components/header/header.component';
7  import { BodyComponent } from './components/body/body.component';
8  import { FooterComponent } from './components/footer/footer.component';
9
   You, a few seconds ago | 1 author (You)
10 @NgModule({
11   declarations: [
12     AppComponent,
13     HeaderComponent,
14     BodyComponent,
15     FooterComponent
16   ],
```

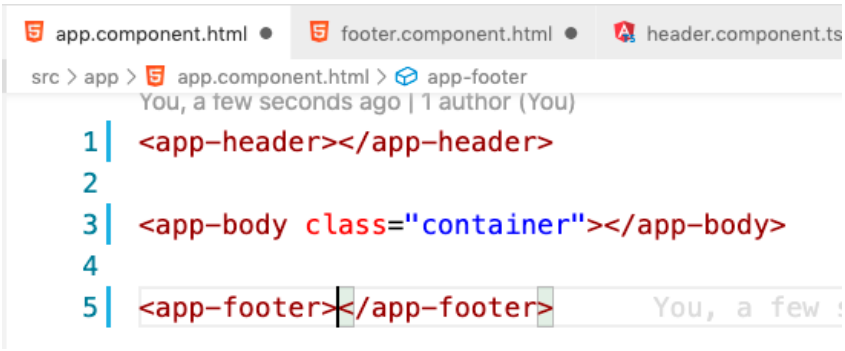
Modificamos el **footer.component.html**:

app.component.html | footer.component.html | header.component.ts

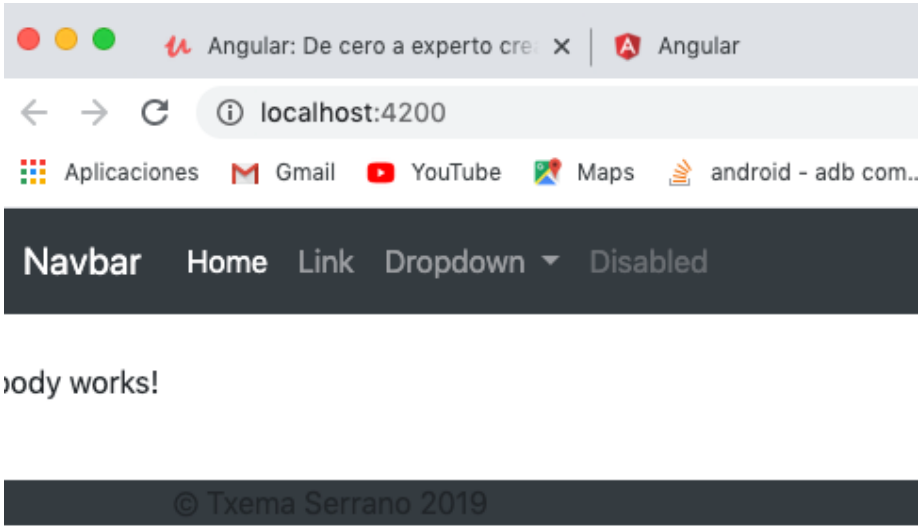
src > app > components > footer > footer.component.html > ...

```
1  <footer class="footer bg-dark">
2
3      <div class="container">
4
5          <p>
6              &copy; Txema Serrano 2019
7          </p>
8      </div>
9
10 </footer>
```

Para que se visualice modifico **app.component.html**:

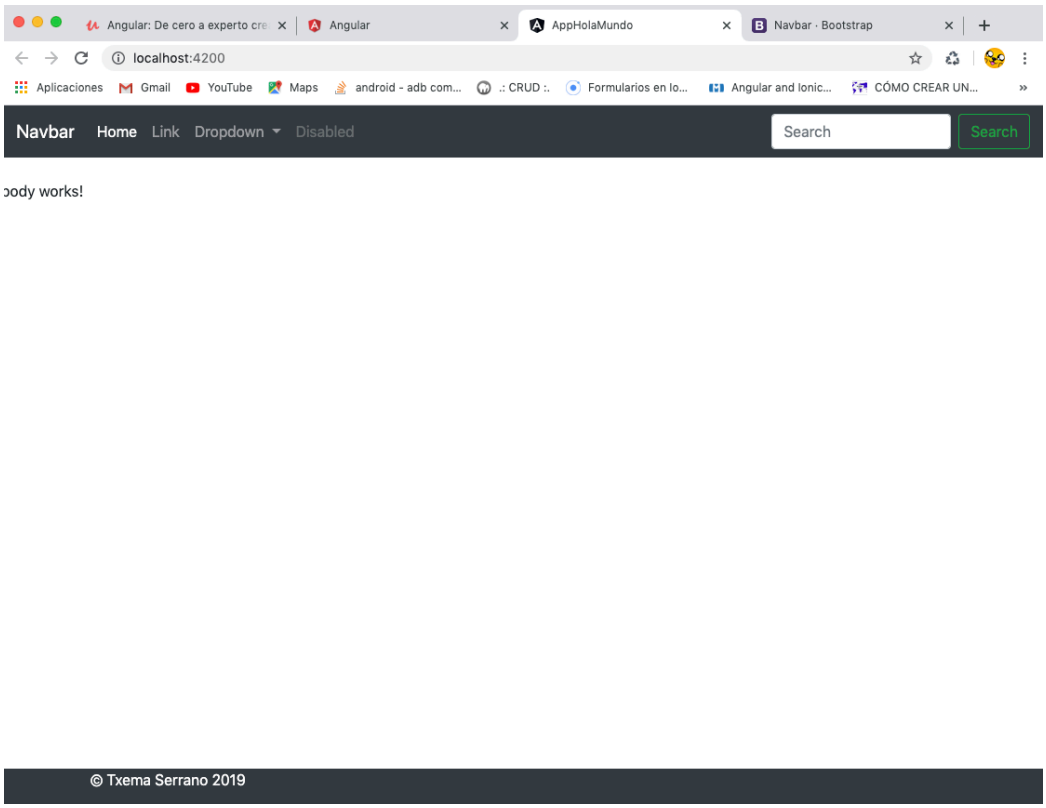


Resultado:



Está muy encima y el texto debe ser tener un color claro. Para hacer eso modificamos el **style.css**:

El resultado:



Podemos centrar el texto del footer, y que el año salga de forma automática:

```
8  export class FooterComponent {
9
10   anio: number;
11
12   constructor() {
13     this.anio = new Date().getFullYear();
14   }
15
16 }
```

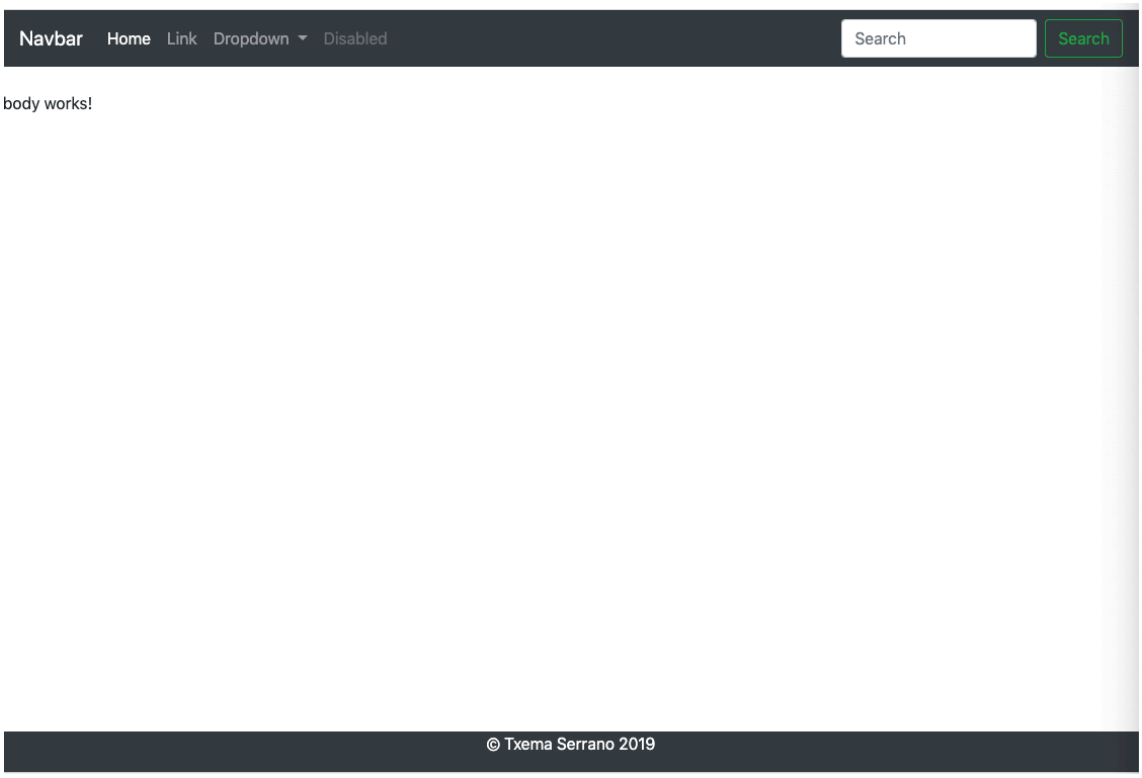
En **footer.component.html**:

app.component.html | styles.css | footer.component.html | footer.

src > app > components > footer > footer.component.html > footer.footer.bg-dark.tex

```
1  <footer class="footer bg-dark text-center">
2
3      <div class="container">
4
5          <p>
6              &copy; Txema Serrano {{ anio }}
7          </p>
8      </div>
9
10 </footer>
```

Ahora:

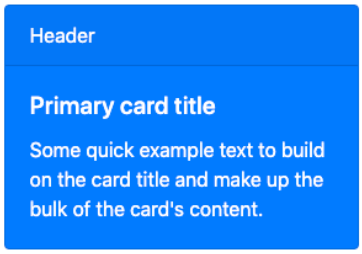

Navbar
Home
Link
Dropdown
Disabled
Search
Search
body works!
© Txema Serrano 2019

4. Estructura del BodyComponent:

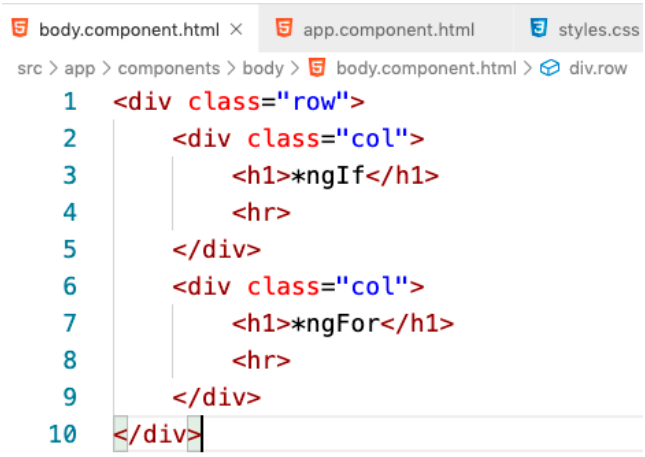
Vamos a getBootstrap y copiamos el código de una tarjeta:

```
<div class="card text-white bg-primary mb-3" style="max-width: 18rem;">
  <div class="card-header">Header</div>
  <div class="card-body">
    <h5 class="card-title">Primary card title</h5>
    <p class="card-text">Some quick example text to build on the card tit
  </div>
</div>
```

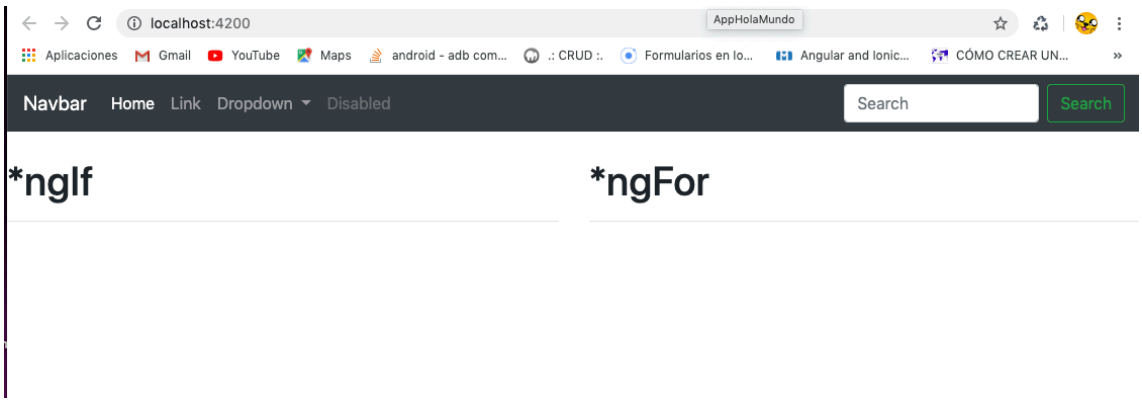
Resultado:



Vamos body.component.html:



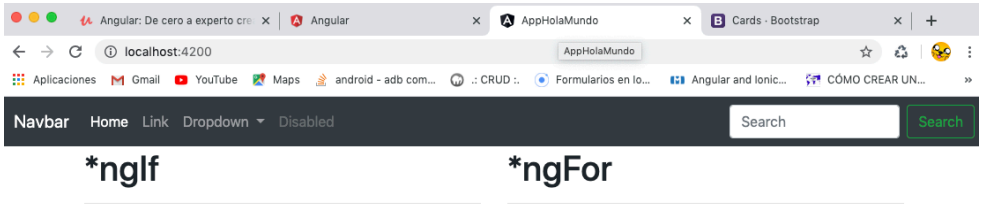
Resultado:



Para que aparezca todo alejado de los bordes, centrado.

```
body.component.html   app.component.html ×
src > app > app.component.html > app-footer
    You, a few seconds ago | 1 author (You)
1   <app-header></app-header>
2
3   <div class="container">
4       <app-body>
5
6       </app-body>
7   </div>
8
9
10  <app-footer></app-footer>
```

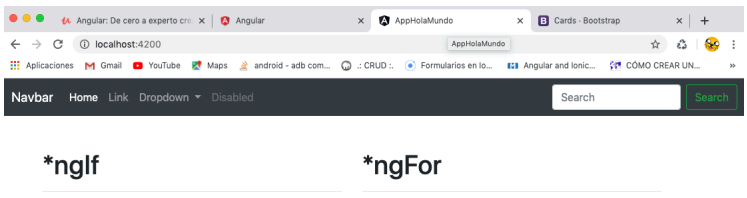
Vemos:



Para separarlo un poco:

```
body.component.html   app.component.html ●
src > app > app.component.html > div.container.m-5
    You, a few seconds ago | 1 author (You)
1   <app-header></app-header>
2
3   <div class="container m-5">
4       <app-body>
5
6       </app-body>
7   </div>
8
9
10  <app-footer></app-footer>
```

Resultado:



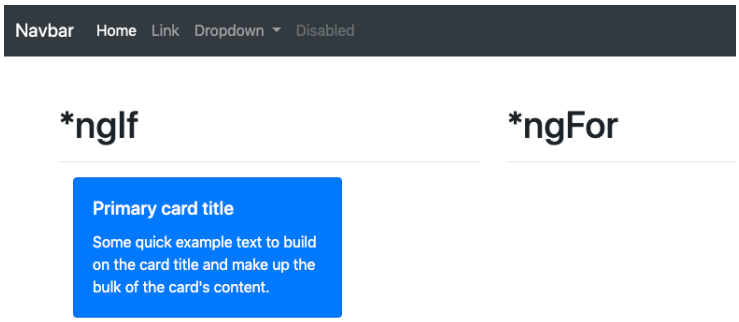
Debajo del ngIf, copiamos el código de la tarjeta:

```
<div class="container">
    <div class="card text-white bg-primary mb-3" style="max-width: 18rem;
    ">
        <div class="card-header">Header</div>
        <div class="card-body">
          <h5 class="card-title">Primary card title</h5>
          <p class="card-text">Some quick example text to build on the
          card title and make up the bulk of the card's content.</p>
        </div>
      </div>
```

Elimino el Header y guardo los cambios:

```
<div class="container">
    <div class="card text-white bg-primary mb-3" style="max-width: 18rem;">
        <div class="card-body">
            <h5 class="card-title">Primary card title</h5>
            <p class="card-text">Some quick example text to build on the
                card title and make up the bulk of the card's content.</p>
        </div>
    </div>
```

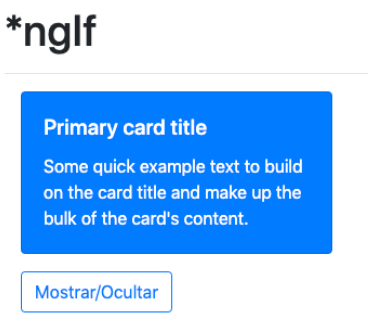
Ahora la apariencia:



Debajo coloco un botón:

```
<button class="btn btn-outline-primary">
    Mostrar/Ocultar
</button>
```

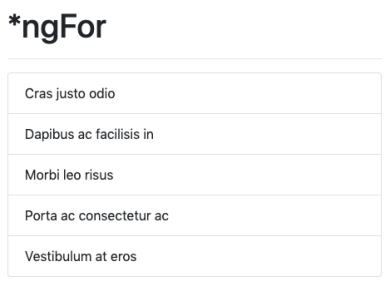
Queda:



Seguimos con la parte del ngFor, donde utilizaremos un list-group:

```
<ul class="list-group">
  <li class="list-group-item">Cras justo odio</li>
  <li class="list-group-item">Dapibus ac facilisis in</li>
  <li class="list-group-item">Morbi leo risus</li>
  <li class="list-group-item">Porta ac consectetur ac</li>
  <li class="list-group-item">Vestibulum at eros</li>
</ul>
```

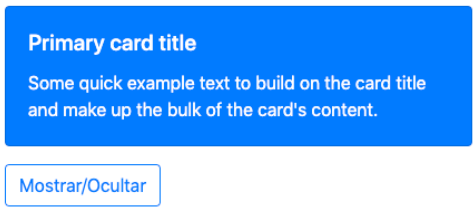
La salida es:



Le decimos al card que tome todo el ancho disponible:

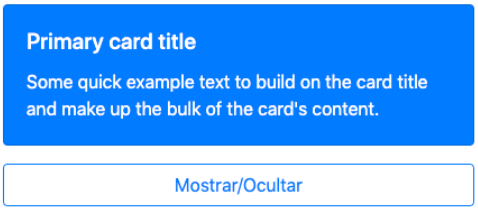
```
<div class="container">
    <div class="card text-white bg-primary mb-3" style="width: 100%;">
        <div class="card-body">
            <h5 class="card-title">Primary card title</h5>
            <p class="card-text">Some quick example text to build on t
            up the bulk of the card's content.</p>
        </div>
    </div>
```

Queda:



Y para que el btn tome todo el ancho, le ponemos la clase block.

```
<button class="btn btn-outline-primary btn-block">
    Mostrar/Ocultar
</button>
```



5. Directivas estructurales *ngIf y *ngFor.

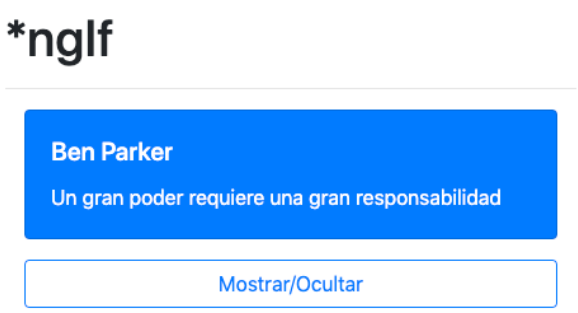
En body.component.ts:

```
8  export class BodyComponent {
9
10   frase: any = {
11     mensaje: 'Un gran poder requiere una gran responsabilidad',
12     autor: 'Ben Parker'
13   };
```

Lo utilizaremos en el HTML:

```
<div class="card text-white bg-primary mb-3" style="width: 100%;">
    <div class="card-body">
        <h5 class="card-title">{{ frase.autor }}</h5>
        <p class="card-text">{{ frase.mensaje }}</p>
    </div>
</div>
```

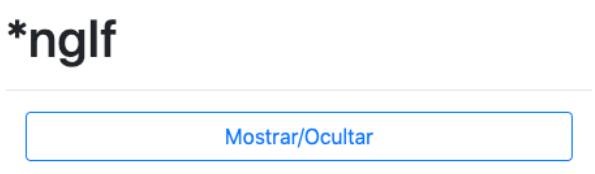
Queda:



Programamos el botón. En el elemento a ocultar coloco *ngif:

```
6          <div class="container">
7              <div *ngIf="false" class="card text-white bg-primary mb-3"
               style="width: 100%;">
8                  <div class="card-body">
9                      <h5 class="card-title">{{ frase.autor }}</h5>
10                     <p class="card-text">{{ frase.mensaje }}</p>
11                 </div>
12             </div>
```

Desaparece:



Si inspecciono el elemento, vemos que el contenido HTML dejó de existir y queda un comentario para que lo trabaje Angular:

```
▼<div class="col">
   <h1>*ngIf</h1>
   <hr>
  ▼<div class="container">
     <!--bindings={
       "ng-reflect-ng-if": "false"
     }-->
     <button class="btn btn-outline-primary btn-block"> Mostrar/Ocultar </button>
   </div>
 </div>
▶<div class="col">…</div>
```

Debo hacer que el botón cambie el valor true/false a medida que lo pulse. En el typeScript, creamos una variable mostrar:

```
8  export class BodyComponent {
9
10   mostrar = true;
11
12   frase: any = {
13     mensaje: 'Un gran poder requiere una gran responsabilidad',
```

Lo utilizaremos en el *ngIf:*

```
<div class="container">
    <div *ngIf="mostrar" class="card text-white bg-primary mb-3" style="width: 100%;">
        <div class="card-body">
            <h5 class="card-title">{{ frase.autor }}</h5>
            <p class="card-text">{{ frase.mensaje }}</p>
        </div>
    </div>

    <button (click)= "mostrar = !mostrar" class="btn btn-outline-primary btn-block">
        Mostrar/Ocultar
    </button>
```

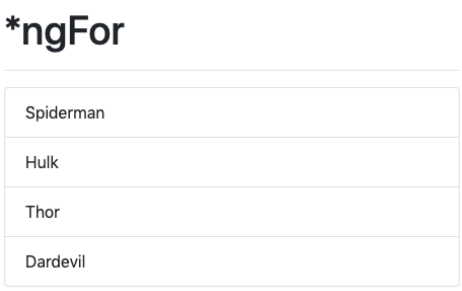
Con el *ngFor*, debemos declarar un array en el typescript:

```
17   personajes: string[] = ['Spiderman','Hulk','Thor','Dardevil'];
```

Para recorrerlo:

```
<ul class="list-group">
    <li *ngFor="let personaje of personajes" class="list-group-item">
        {{ personaje }}
    </li>
</ul>
```

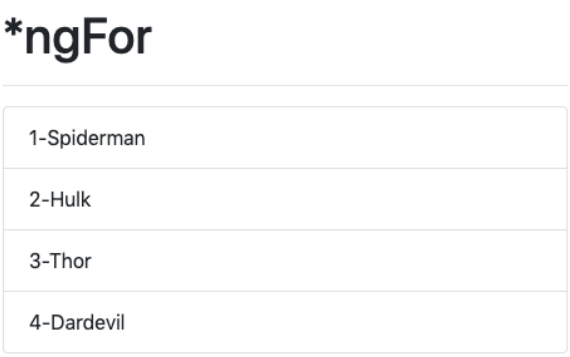
La salida:



Si queremos ponerle un número:

```
<ul class="list-group">
    <li *ngFor="let personaje of personajes; let i = index" class="list-group-item">
        {{ i+1 }}-{{ personaje }}
    </li>
</ul>
```

Tenemos:



Cuestionario:

Pregunta 1:

¿Qué son las directivas estructurales?

- ○ Son instrucciones que pueden modificar la estructura de los archivos de TypeScript.
- ● Son instrucciones que agregan, eliminan o reemplazan elementos HTML en el template.
- ○ Son formas de crear nuevos elementos en los templates.

Pregunta 2:

¿Qué es el siguiente código?

```
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-about',
5    templateUrl: './about.component.html'
6  })
7  export class AboutComponent implements OnInit {
8
9    constructor() { }
10
11   ngOnInit() {
12   }
13
14 }
```

- ○ Es una directiva
- ● Es un componente
- ○ Es una directiva estructural

Pregunta 3:

Dado el siguiente componente, ¿Cómo podemos insertarlo en otro template?

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

@Component({
  selector: 'app-heroes-about',
  templateUrl: './about.component.html'
})
export class AboutComponent {

}
```

- (●) <app-heroes-about></app-heroes-about>
- ( ) <heroes-about></heroes-about>
- ( ) <about-app></about-app>

Pregunta 4:

¿Cuál de las siguientes directivas permiten repetir un bloque de código HTML?

- ( ) *ngIf
- ( ) *ngRepeat
- (●) *ngFor
- ( ) *ngWhile

Pregunta 5:

¿Para qué se utiliza la siguiente instrucción?

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

- ( ) Es para importar los paquetes que existen en angular.
- (●) Sirve para importar el componente que se utiliza como decorador para que las clases de TypeScript se conviertan en componentes.
- ( ) Sirve para tener en memoria todos los componentes que se encuentran disponibles en Angular

Pregunta 6:

En un componente, ¿Qué hace la siguiente instrucción?

```
selector: 'app-about'
```

- ○ Nos permite definir el nombre que tendra el componente.
- ● Nos permite definir el nombre con el cual podremos usar el componente en el HTML.
- ○ Podemos definir el nombre que tendrá la aplicación cuando usamos los componentes.

Pregunta 7:

¿Cuál sería el mejor nombre para el nombre de la clase del siguiente componente ?

"lista-compras"

- ○ export class listaCompras
- ○ export class listaDeComprasComponent
- ● export class ListaComprasComponent

Pregunta 8:

¿Cuál sería el mejor nombre del archivo del siguiente componente?

```
export class SobrePaginaComponent
```

- ○ Sobre-Pagina.component.ts
- ○ SobrePagina.component.ts
- ● sobre-pagina.component.ts
- ○ Sobre-pagina.component.ts

Pregunta 9:

¿Qué error tiene el siguiente código?

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

@Component({
  selector: 'app-about',
  template: './about.component.html'
})
export class AboutComponent implements OnInit {
  constructor() { }
  ngOnInit() {
  }
}
```

- ○ El decorador no ha sido importado correctamente
- ○ El selector esta mal definido
- ○ El nombre del componente esta malo.
- ◉ El template no es válido

Pregunta 10:

¿Qué beneficios tenemos al usar componentes?

- ○ Permiten la re-utilización del código.
- ○ Nos ayuda a segmentar nuestro código en archivos más pequeños que son más fáciles de mantener.
- ○ Nos ayudan a generar rápidamente páginas o secciones de nuestra aplicación.
- ◉ Todas las anteriores.