

Paso 21 Diseño de Pantalla de detalle.

En `pelicula.component.html` hacemos un diseño básico:

```

3   <div class="row">
4     <div class="col-md-6">
5       <!-- Imagen -->
6       <img src="" alt="">
7     </div>
8     <div class="col-md-6">

```

La imagen la vamos a obtener del objeto `movie`. Para ello en `pelicula.component.ts`:

```

3   import { MovieResponse } from '../interface/movie-response';
4   import { PeliculasService } from '../services/peliculas.service';
5
6
7   @Component({
8     selector: 'app-pelicula',
9     templateUrl: './pelicula.component.html',
10    styleUrls: ['./pelicula.component.scss']
11  })
12  export class PeliculaComponent implements OnInit {
13
14    public pelicula: MovieResponse | undefined;
15
16    constructor( private activatedRoute: ActivatedRoute,
17                private peliculasService: PeliculasService) { }
18
19    ngOnInit(): void {
20      const id = this.activatedRoute.snapshot.params["id"];
21      console.log(id);
22
23      this.peliculasService.getPeliculaDetalle(id)
24        .subscribe(movie =>{
25          console.log(movie);
26          this.pelicula = movie;
27        })
28    }

```

Ahora completamos `pelicula.component.html`:

```

3   <div class="row" *ngIf="pelicula">
4     <div class="col-md-6">
5       <!-- Imagen -->
6       <img [src]="pelicula.poster_path | poster" alt="">
7     </div>

```

Nos da error en el pipe poster porque es un pipe personalizado y debemos importarlo. Así en pages.modules.ts:

```

7   import { PipesModule } from '../pipes/pipes.module';
8
9
10  @NgModule({
11    declarations: [
12      HomeComponent,
13      PeliculaComponent,
14      BuscarComponent
15    ],
16    imports: [
17      CommonModule,
18      ComponentsModule,
19      PipesModule
20    ],
21  })
22  export class PagesModule { }

```

Reinicio el proyecto para que recoja los cambios:

Build at: 2022-01-25T17:17:13.782Z - Hash: 54c60aa9e37a22a0 - Time: 6274ms

Error: src/app/pages/pelicula/pelicula.component.html:4:21 - error TS2345: Argument of type 'null' is not assignable to parameter of type 'string'.

```

6   <img [src]="pelicula.poster_path| poster" [alt]="pelicula.title">

```

```

src/app/pages/pelicula/pelicula.component.ts:9:16
9   templateUrl: './pelicula.component.html',
Error occurs in the template of component PeliculaComponent.

```

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

✖ Failed to compile.

Me da el error en la propiedad `poster_path`. Revisamos la interfaz y la documentación. En el objeto `movie` de `película.component.ts`

```
overview: "En un mundo compuesto por dos realidades, lo cotidiano y lo oculto tras ella, Thomas Anderson se ve ob  
popularity: 3117.579  
poster_path: "/dLIwpCTf4Qow01pp85KP9jcfTpu.jpg"
```

En la interfaz `poster_path`: `null`, lo cambiamos a `string`.

```
1  export interface MovieResponse {  
2      adult:                boolean;  
3      backdrop_path:        null;  
4      belongs_to_collection: null;  
5      budget:                number;  
6      genres:                any[];  
7      homepage:             string;  
8      id:                    number;  
9      imdb_id:               string;  
10     original_language:     string;  
11     original_title:         string;  
12     overview:               string;  
13     popularity:             number;  
14     poster_path:            string;
```

Resultado:



Agregaremos unas clases para adaptar el contenido. Para poder poner los datos del rating de las películas en `pelicula.component.html`, debemos importar el módulo `RatingModule` en `pages.module.ts`:

```
TS pages.module.ts U x TS home.component.ts U TS peliculas-poster-grid.comp
src > app > pages > TS pages.module.ts > ...
1  import { NgModule } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3  import { HomeComponent } from './home/home.component';
4  import { PeliculaComponent } from './pelicula/pelicula.component';
5  import { BuscarComponent } from './buscar/buscar.component';
6  import { ComponentsModule } from './components/components.module';
7  import { PipesModule } from './pipes/pipes.module';
8  import { RatingModule } from 'ng-starrating';
9
10 @NgModule({
11   declarations: [
12     HomeComponent,
13     PeliculaComponent,
14     BuscarComponent
15   ],
16   imports: [
17     CommonModule,
18     ComponentsModule,
19     PipesModule,
20     RatingModule
21   ]
22 })
23 export class PagesModule { }
```

En `pelicula.component.html` ya puedo utilizar el rating:

```
3  <div class="row" *ngIf="pelicula">
4    <div class="col-md-6">
5      <!-- Imagen -->
6      <img [src]="pelicula.poster_path| poster"
7          [alt]="pelicula.title"
8          class="img-thumbnail">
9    </div>
10   <div class="col-md-6">
11     <!-- Detalle -->
12     <h3>{{ pelicula.title }}</h3>
13
14     <div class="row">
15       <div class="col">
16         <star-rating [value]="pelicula.vote_average"
17                     [totalstars]="10"
18                     checkedcolor="yellow"
19                     uncheckedcolor="grey"
20                     size="15px"></star-rating>
21       </div>
22       <div class="col">
23         {{ pelicula.vote_average }}
24       </div>
25     </div>
26   </div>
27 </div>
```


Resultado:



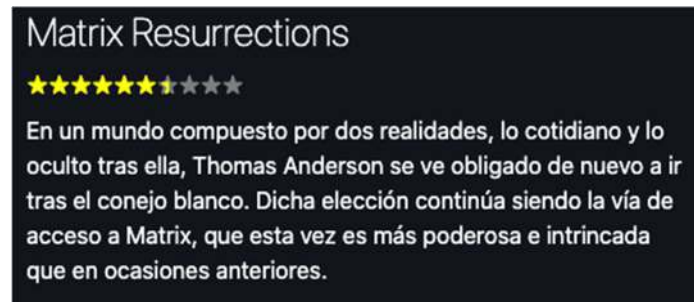
Ahora vamos a colocar un resumen de la película:

```

14     <div class="row">
15         <div class="col">
16             <star-rating [value]="película.vote_average"
17                         [totalstars]="10"
18                         checkedcolor="yellow"
19                         uncheckedcolor="grey"
20                         size="15px"></star-rating>
21         </div>
22         <div class="col text-white">
23             {{ película.vote_average }}
24         </div>
25     </div>
26     <div class="row">
27         <div class="col text-white">
28             {{ película.overview }}
29         </div>
30     </div>

```

Resultado:



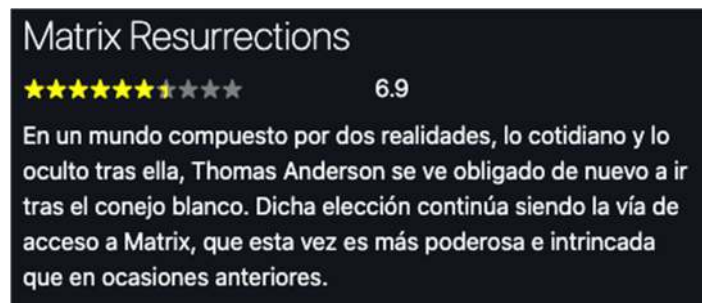
Si queremos que el rating de los votos tenga una posición entera y dos decimales:

```

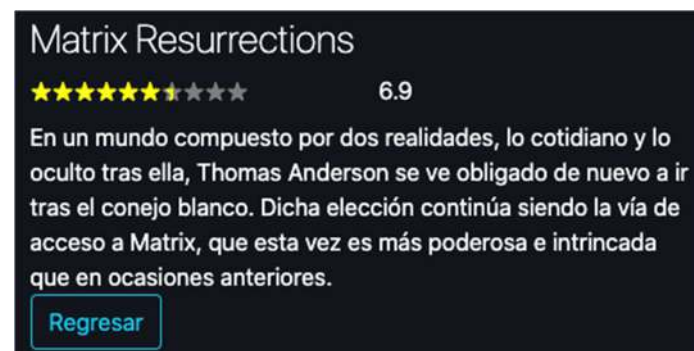
14 <div class="row">
15   <div class="col">
16     <star-rating [value]="pelicula.vote_average"
17                 [totalstars]="10"
18                 checkedcolor="yellow"
19                 uncheckedcolor="grey"
20                 size="15px"></star-rating>
21   </div>
22   <div class="col text-white">
23     {{ pelicula.vote_average | number: '1.1-2' }}
24   </div>
25 </div>

```

Resultado:



Vamos a incluir un botón que nos permita volver a la pantalla anterior. Esto es, volver a la pantalla de búsqueda o a la cartelera:



Demasiado pegado. Modificamos el template en la parte del botón y en del sumario:

```

27 <div class="row mt-2">
28   <div class="col text-white">
29     {{ pelicula.overview }}
30   </div>
31 </div>
32
33 <div class="row mt-3">
34   <div class="col">
35     <button class="btn btn-outline-info"
36       (click)="onRegresar()"
37     >Regresar</button>
38   </div>
39 </div>

```

En el componente debemos implementar la función `onRegresar()` para que haga un retorno condicional. Si estaba en home, retorno al home. Si estaba en buscar, retorno a buscar. Para ello inyectamos en el constructor del componente el `Location`

```

import { Location } from '@angular/common';
...

17 constructor( private activatedRoute: ActivatedRoute,
18               private peliculasService: PeliculasService,
19               private location: Location) { }
--

```

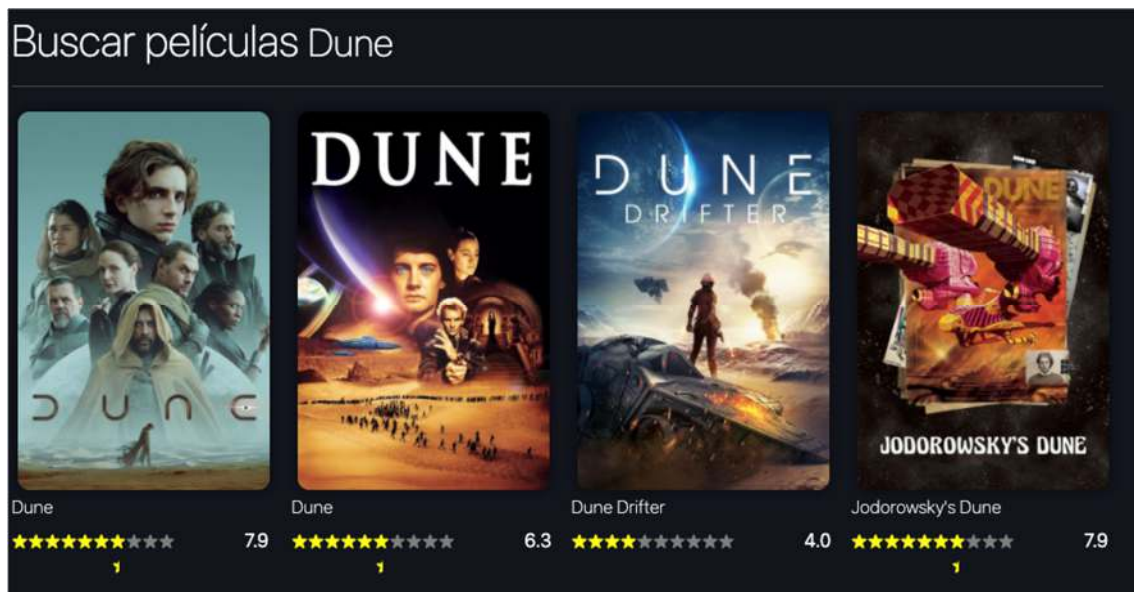
Así:

```

32 onRegresar() {
33   this.location.back();
34 }
--

```

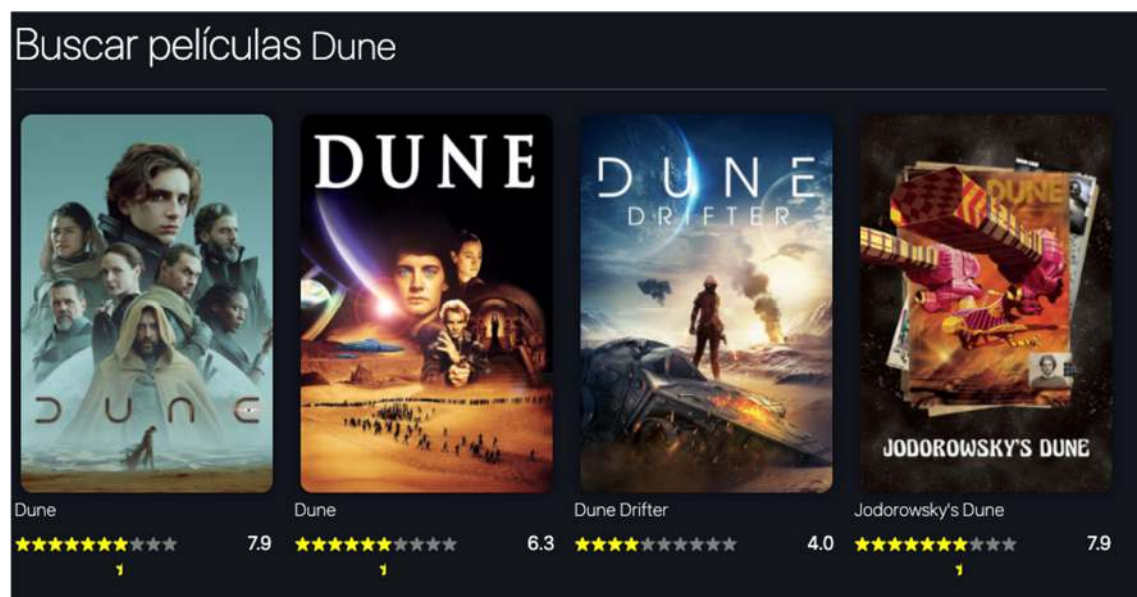
Probamos:



Elegimos la segunda



Si le damos a regresar, regresa a la búsqueda.



Si viniera de la cartelera, retornaría a la cartelera porque el método de retorno es:

```
onRegresar(){
    this.location.back();
}
```

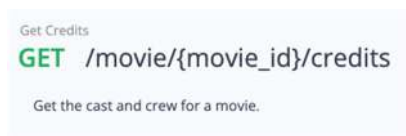
Volver a la anterior localización del historial.

Paso 22 Actores de la película y manejo de errores.

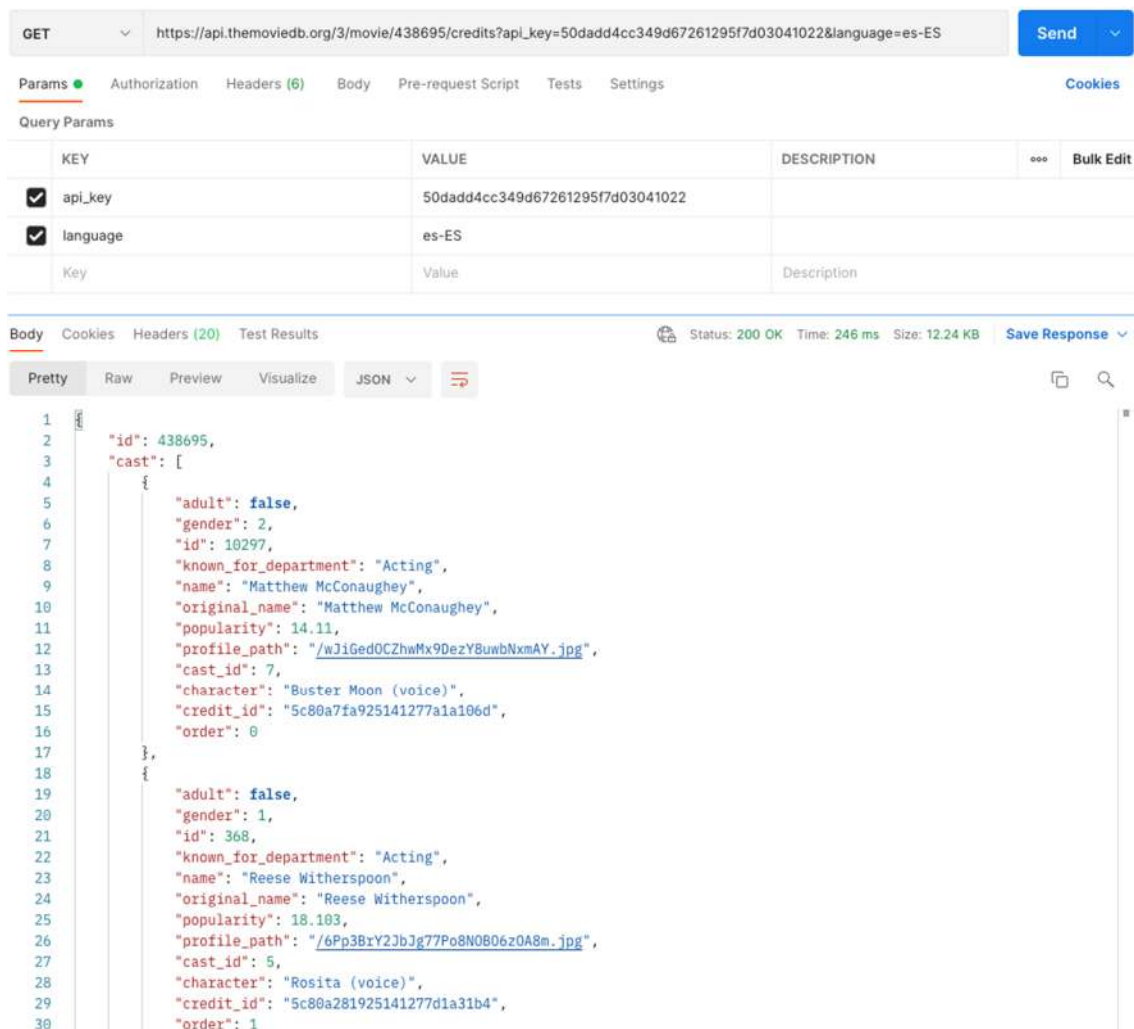
Para el manejo de errores debemos evitar, por ejemplo, si busco una película con un id que no exista, daría este error. Después de los actores resolveremos este problema.



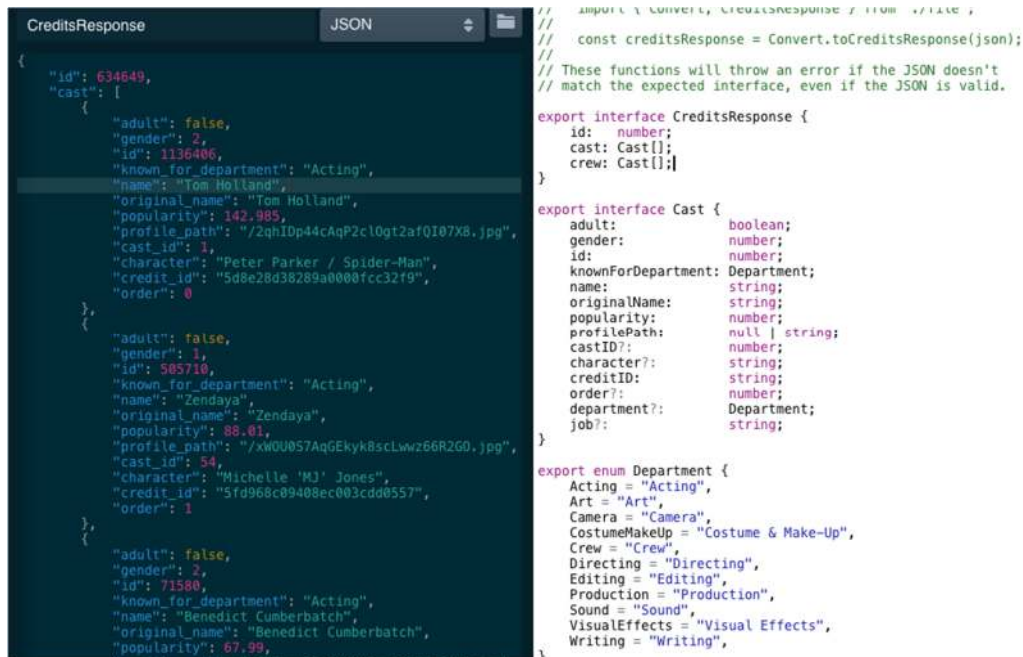
Para conseguir los actores de la película, en la documentación de themoviedb, probamos:



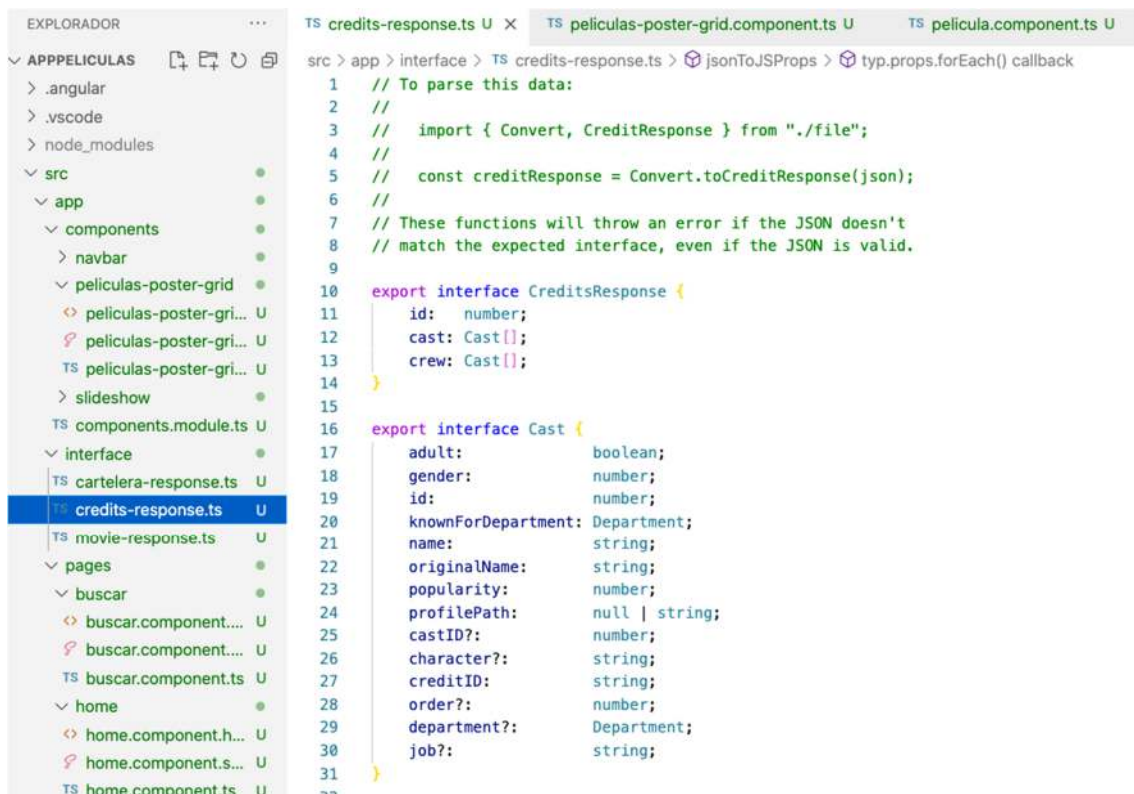
En PostMan:



Copiamos todo este JSON y lo pegamos en quicktype para obtener la interface:



Creamos un fichero en las interfaces credits-reponse.ts



Lo siguiente es crear en el servicio una función que nos recupere los actores.

Tratamiento de errores. ¿qué pasa si pongo un id que no existe? Voy a tener dos errores.

```
Angular is running in development mode. Call enableProdMode() to enable production mode. core.mjs:24833
63464965 pelicula.component.ts:23
[webpack-dev-server] Live Reloading enabled. index.js:548
➤ GET https://api.themoviedb.org/3/movie/63464965/credits?api_key=50dadd4...&language=es-ES&page=1 404 zone.js:2863
➤ ERROR core.mjs:6461
  HttpErrorResponse {headers: HttpHeaders, status: 404, statusText: 'OK', url: 'https://api.themoviedb.org/3/movie/63464965/credits?api_key=50dadd4...&language=es-ES&page=1', ok: false, ...}
➤ GET https://api.themoviedb.org/3/movie/63464965?api_key=50dadd4...&language=es-ES&page=1 404 zone.js:2863
➤ ERROR core.mjs:6461
  HttpErrorResponse {headers: HttpHeaders, status: 404, statusText: 'OK', url: 'https://api.themoviedb.org/3/movie/63464965?api_key=50dadd4...&language=es-ES&page=1', ok: false, ...}
```

Manejemos el error de getPeliculaDetalle() del servicio:

```
66 getPeliculaDetalle(id: string){
67
68     // https://api.themoviedb.org/3/movie/826749?api_key=0976c2b22dfbf0514acc7ac175a
69
70     return this.http.get<MovieResponse>(`${ this.baseUrl}/movie/${id}`, {
71       params: this.params
72     }).pipe(
73       catchError(err=>of(null))
74     )
75
76     // En los params está la apikey y el lenguaje
77
78 }
```

catchError debo importarlo del operador rxjs:

```
4
5 import { catchError, tap } from 'rxjs/operators';
6
```

Guardo los cambios, pero seguimos con problemas:

```
➤ GET https://api.themoviedb.org/3/movie/63464965?api_key=50dadd4...&language=es-ES&page=1 404 zone.js:2863
➤ [webpack-dev-server] Errors while compiling. Reload prevented. index.js:558
➤ [webpack-dev-server] ERROR index.js:558
  src/app/pages/pelicula/pelicula.component.ts:28:11 - error TS2322: Type 'MovieResponse | null' is not assignable to type 'MovieResponse | undefined'.
    Type 'null' is not assignable to type 'MovieResponse | undefined'.
```

Ahora en la función getCast del servicio:

```
80 getCast(id: string){
81
82     // https://api.themoviedb.org/3/movie/634649/credits?api_key=0976c2b22dfbf0514ac
83
84     return this.http.get<CreditsResponse>(`${ this.baseUrl}/movie/${id}/credits`, {
85       params: this.params
86     }).pipe(
87       map( resp=> resp.cast ),
88       catchError(err=>of(null))
89     );
90
91     // No me interesa devolver todo el objeto, sólo el casting: resp.cast
92
93     // En los params está la apikey y el lenguaje
94
95 }
```


Vuelco a cargar la aplicación y recargo.

[illegible]

Vamos a manejar las excepciones en pelicula.component.ts:

```
15 public película: MovieResponse | undefined | null;  
16  
17 constructor( private activatedRoute: ActivatedRoute,  
18             private películasService: PelículasService,  
19             private location: Location) { }  
20  
21 ngOnInit(): void {  
22     const id = this.activatedRoute.snapshot.params["id"];  
23     console.log(id);  
24  
25     this.películasService.getPelículaDetalle(id)  
26     .subscribe(movie =>{  
27         console.log(movie);  
28         this.película = movie;  
29     });  
30 }
```

La salida por consola:

The screenshot shows the Chrome DevTools Console with the following content:

- Angular is running in development mode. Call enableProdMode() to enable production mode.
- 24428
- A log entry: `{adult: false, backdrop_path: '/nNmJRkgBwWmRmzQDe2FwKbPISJV.jpg', belongs_to_collection: {}, budget: 22000000, genres: Array(3), ...}`
- [webpack-dev-server] Live Reloading enabled.

On the right side of the console, there are file names and line numbers corresponding to the log entries:

- core.mjs:24833
- pelicula.component.ts:24
- pelicula.component.ts:30
- index.js:548
- pelicula.component.ts:38

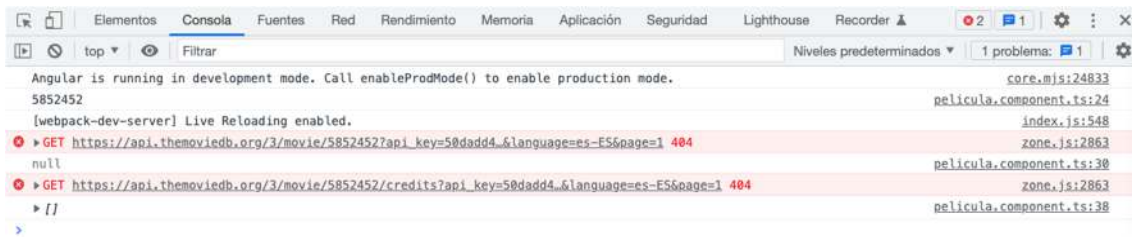
Vamos a manejar los Así en getCast de película.service.ts y que si se produce un error devuelva un array vacío:

```

80  getCast(id: string){
81
82    // https://api.themoviedb.org/3/movie/634649/credits?api_key=0976c2b22dfbf0514acc
83
84    return this.http.get<CreditsResponse>(`${ this.baseUrl}/movie/${id}/credits`, {
85      params: this.params
86    }).pipe(
87      map( resp=> resp.cast ),
88      catchError( err=> of( [] ) )
89    );
90

```

La salida es para es código de película que no existe:



En `getPeliculaDetalle()` en cambio:

```

66 getPeliculaDetalle(id: string) {
67
68   // https://api.themoviedb.org/3/movie/826749?api_key=0976c2b22dfbf0514acc7ac175a
69
70   return this.http.get<MovieResponse>(`${ this.baseUrl}/movie/${id}`, {
71     params: this.params
72   }).pipe(
73     catchError(err => of([null]))
74   )
75
76   // En los params está la apikey y el lenguaje
77
78 }
  
```

Regresamos a `película.component.ts`, para que me lleve a otra página si hay un fallo, inyecto el router:

```

18 constructor( private activatedRoute: ActivatedRoute,
19               private peliculasService: PeliculasService,
20               private location: Location,
21               private router: Router) { }
  
```

Y en la función `getPeliculaDetalle`:

```

29 this.peliculasService.getPeliculaDetalle(id)
30   .subscribe(movie => {
31     console.log(movie);
32     if (!movie) {
33       this.router.navigateByUrl('/home'); // lo envío a otra página
34       return; // volvemos para que no haga nada
35     }
36     this.pelicula = movie;
37   });
  
```

Antes de terminar película.component.ts creo una nueva propiedad pública que se llama cast.

```

4  import { Cast } from 'src/app/interface/credits-response';
5  import { MovieResponse } from 'src/app/interface/movie-response';
6  import { PeliculasService } from 'src/app/services/peliculas.service';
7
8
9
10 @Component({
11   selector: 'app-pelicula',
12   templateUrl: './pelicula.component.html',
13   styleUrls: ['./pelicula.component.scss']
14 })
15 export class PeliculaComponent implements OnInit {
16
17   public pelicula: MovieResponse | undefined | null;
18   public cast: Cast[] | undefined;
19 }

```

Recordemos que Cast[] tiene esta interface:

```

16 export interface Cast {
17   adult: boolean;
18   gender: number;
19   id: number;
20   knownForDepartment: string;
21   name: string;
22   originalName: string;
23   popularity: number;
24   profilePath: null | string;
25   castID?: number;
26   character?: string;
27   creditID: string;
28   order?: number;
29   department?: string;
30   job?: string;
31 }

```

Este objeto Cast[] lo vamos a llenar en el servicio en la función getCast(id) de película.component.ts:

```

44   this.peliculasService.getCast(id)
45     .subscribe( cast =>{
46       console.log(cast);
47       this.cast = cast;
48     } )
49 }

```

Paso 23 SlideShow de actores.

Creamos un nuevo componente slideShow llamado catSlideshow:

```
jsersan@iMac-de-Jose appPelículas % ng g c components/castSlideshow --skipTests
Support for camel case arguments has been deprecated and will be removed in a future major version.
Use '--skip-tests' instead of '--skipTests'.
CREATE src/app/components/cast-slideshow/cast-slideshow.component.scss (0 bytes)
CREATE src/app/components/cast-slideshow/cast-slideshow.component.html (29 bytes)
CREATE src/app/components/cast-slideshow/cast-slideshow.component.ts (307 bytes)
UPDATE src/app/components/components.module.ts (935 bytes)
jsersan@iMac-de-Jose appPelículas %
```

Debemos exportar este componente en components.module.ts:

```
24     exports: [
25       NavbarComponent,
26       SlideshowComponent,
27       PeliculasPosterGridComponent,
28       RatingModule,
29       CastSlideshowComponent
30     ]
31   })
32   export class ComponentsModule {}
33
```

Programemos cast-slideshow.component.ts:

```
TS cast-slideshow.component.ts U TS components.module.ts U TS pag
src > app > components > cast-slideshow > TS cast-slideshow.component.ts >
1  import { Component, Input, OnInit } from '@angular/core';
2  import { Cast } from '../interface/credits-response';
3
4  @Component({
5    selector: 'app-cast-slideshow',
6    templateUrl: './cast-slideshow.component.html',
7    styleUrls: ['./cast-slideshow.component.scss']
8  })
9  export class CastSlideshowComponent implements OnInit {
10
11    @Input() cast: Cast[] = [];
12
13    constructor() {}
14
15    ngOnInit(): void {
16      console.log(this.cast);
17    }
18
19  }
```

Para poder ver esto, vamos a pelicula.component.ts:

```
43 <div class="row">
44   <div class="col">
45     <app-cast-slideshow [cast]="cast"></app-cast-slideshow>
46   </div>
47
48 </div>
```


Esto me provoca el siguiente error:

```
[webpack-dev-server] ERROR
src/app/pages/pelicula/pelicula.component.html:45:30 - error TS2322: Type 'Cast[] | undefined' is not assignable to type 'Cast[]'.
  Type 'undefined' is not assignable to type 'Cast[]'.

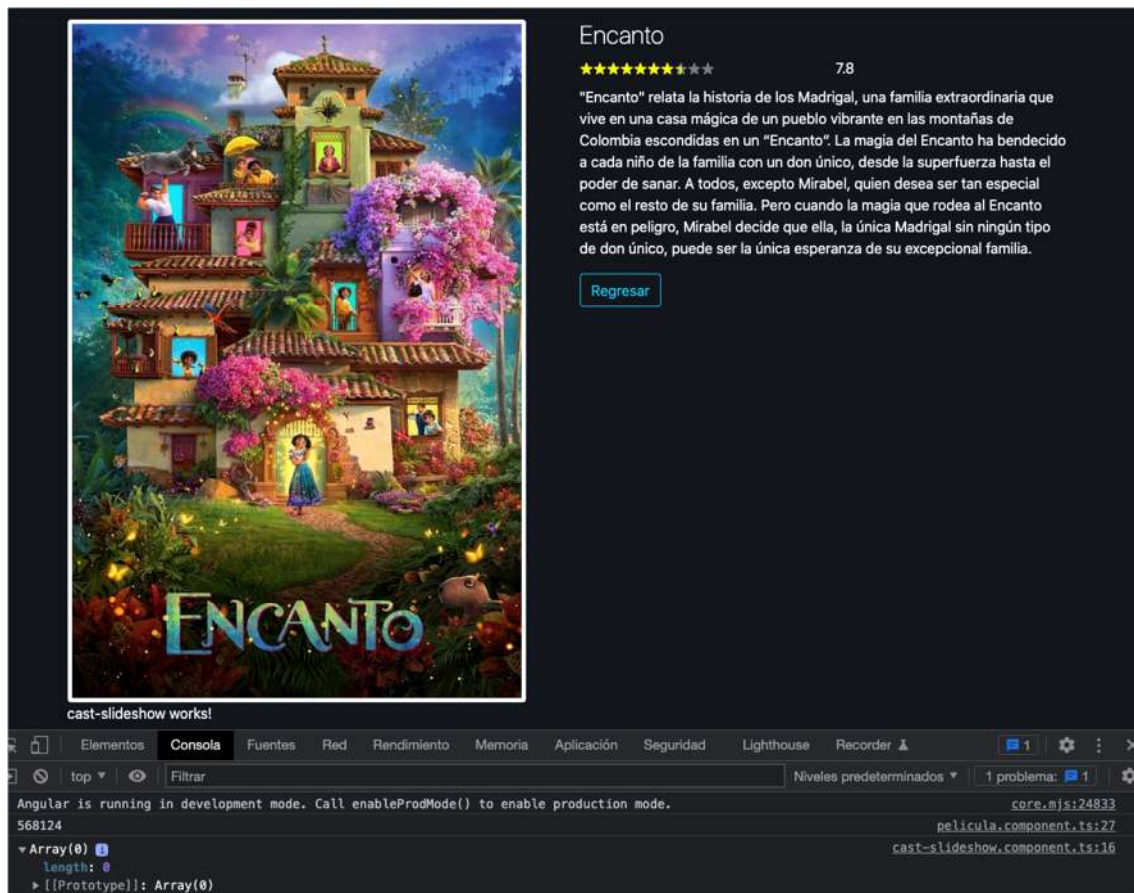
45     <app-cast-slideshow [cast]="cast"></app-cast-slideshow>
                                     ~~~~~

src/app/pages/pelicula/pelicula.component.ts:12:16
12   templateUrl: './pelicula.component.html',
    ~~~~~
Error occurs in the template of component PeliculaComponent.
```

Si lo dejo así, sin la propiedad cast:

```
43 <div class="row">
44   <div class="col">
45     <app-cast-slideshow></app-cast-slideshow>
46   </div>
47 </div>
48 </div>
49
```

El resultado es:



Para evitar que me cante error en `<app-slideshow [cast] = "cast">`, realizo los siguientes cambios:

1. En el archivo `tsconfig.json` ponemos la propiedad `strict` a `false`.

```
tsconfig.json > {} compilerOptions > strict
1  /* To learn more about this file see: https://angular.io/config/tsconfig. */
2  {
3    "compileOnSave": false,
4    "compilerOptions": {
5      "baseUrl": "./",
6      "outDir": "./dist/out-tsc",
7      "forceConsistentCasingInFileNames": true,
8      "strict": false,
```

2. Ahora ya puedo definir en `pelicula.component.ts`:

```
14  })
15  export class PeliculaComponent implements OnInit {
16
17    public pelicula: MovieResponse;
18    public cast: Cast[] = [];
```

3. Y en el template:

```
43  <div class="row">
44    <div class="col">
45      <app-cast-slideshow [cast]="cast"></app-cast-slideshow>
46    </div>
47  </div>
```

La función `getCast` de `pelicula.component.ts`:

```
42  // Obtengo el casting de los actores
43
44  this.peliculasService.getCast(id)
45    .subscribe( cast =>{
46      console.log(cast);
47      this.cast = cast;
48    })
49  }
```

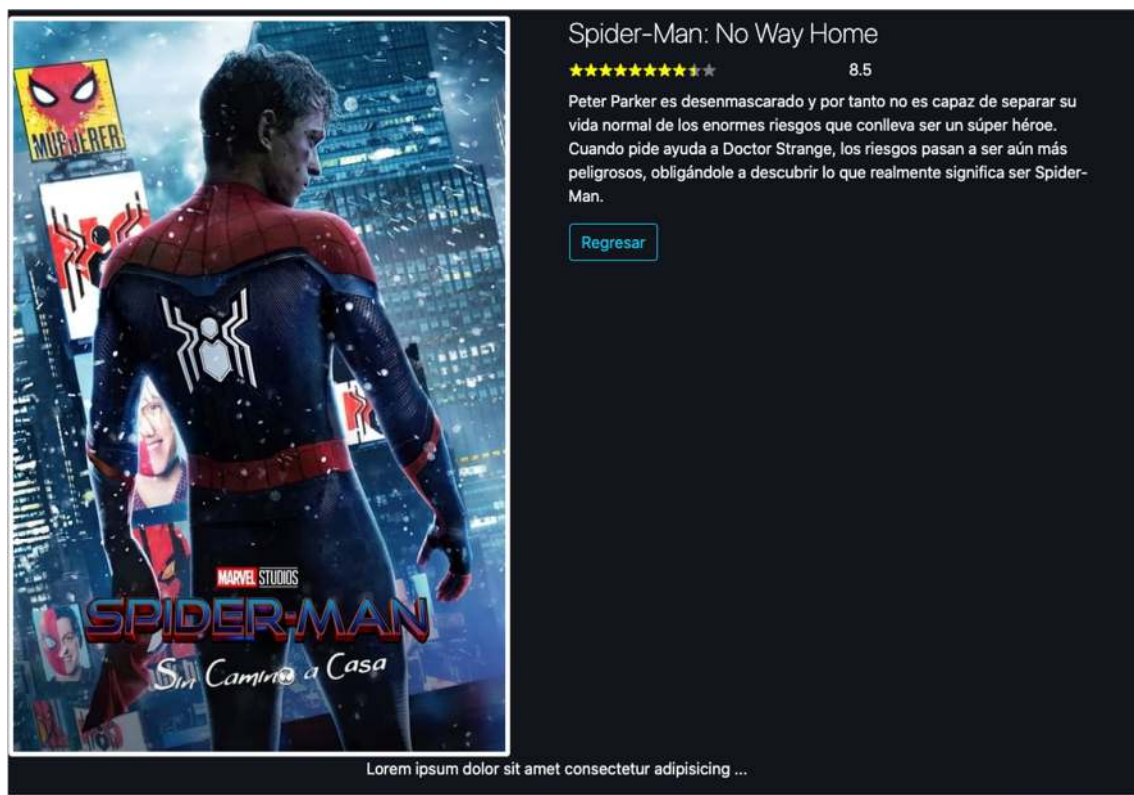
Aquí tenemos el casting proporcionado por el `console.log` de la línea 46 para la película *Spider-Man: No Way Home*:

```
► Object
▼ Array(59)
  ► 0: {adult: false, gender: 2, id: 1136406, known_for_department: 'Acting', name: 'Tom Holland', ...}
  ► 1: {adult: false, gender: 1, id: 505710, known_for_department: 'Acting', name: 'Zendaya', ...}
  ► 2: {adult: false, gender: 2, id: 71580, known_for_department: 'Acting', name: 'Benedict Cumberbatch', ...}
  ► 3: {adult: false, gender: 2, id: 1649152, known_for_department: 'Acting', name: 'Jacob Batalon', ...}
  ► 4: {adult: false, gender: 2, id: 15277, known_for_department: 'Acting', name: 'Jon Favreau', ...}
  ► 5: {adult: false, gender: 2, id: 124, known_for_department: 'Acting', name: 'Jamie Foxx', ...}
```

Hacemos nuestro diseño del slide-show:

```
src > app > components > cast-slideshow > <> cast-slideshow.component.html > ...
Go to component
1
2
3 <div class="swiper-container">
4   <div class="swiper-wrapper">
5     <div class="swiper-slide text-white text-center">
6       Lorem ipsum dolor sit amet consectetur adipisicing ...
7     </div>
8   </div>
9 </div>
10
```

Resultado:



Vamos a separarlo de la imagen. En pelicula.component.html:

```
43 <div class="row mt-5" *ngIf="cast.length > 0">
44   <div class="col">
45     <app-cast-slideshow [cast]="cast"></app-cast-slideshow>
46   </div>
47
48 </div>
49
```

Margin-top = 5 y también que sólo se visualice si el array del casting no está vacío.

Como podemos comprobar ya hay 5 pixeles de separación:



Ahora debemos inicializar el swiper para comprobar que todo funciona correctamente. En cast-slideshow.ts es aconsejable que lo implementemos en AfterViewInit

```
TS cast-slideshow.component.ts 1, U × TS components.module.ts U TS pages.module.ts
src > app > components > cast-slideshow > TS cast-slideshow.component.ts > CastSlideshow
1  import { AfterViewInit, Component, Input, OnInit } from '@angular/core';
2  import { Cast } from '../interface/credits-response';
3
4  @Component({
5    selector: 'app-cast-slideshow',
6    templateUrl: './cast-slideshow.component.html',
7    styleUrls: ['./cast-slideshow.component.scss']
8  })
9  export class CastSlideshowComponent implements OnInit, AfterViewInit {
10
11    @Input() cast: Cast[] = [];
12  }
```

Nos aparece un error en el nombre de la clase porque todavía no hemos implementado este ciclo de vida el AfterViewInit:

```
10  export class CastSlideshowComponent implements OnInit, AfterViewInit {
11
12    @Input() cast: Cast[] = [];
13
14    constructor() { }
15
16  ngOnInit(): void {
17    console.log(this.cast);
18  }
19
20  ngAfterViewInit(): void {
21    const swiper = new Swiper('.swiper-container', {
22      slidesPerView: 5.3,
23      freeMode: true,
24      spaceBetween: 15
25    });
26  }
27
28 }
```

Guardamos los cambios y visualizamos. Todavía no podemos movernos porque sólo tenemos un slide con el contenido de lorem ipsum. Como ya habrán podido adivinar rellenaremos este slide-show con el contenido del array cast a través de un ngFor.

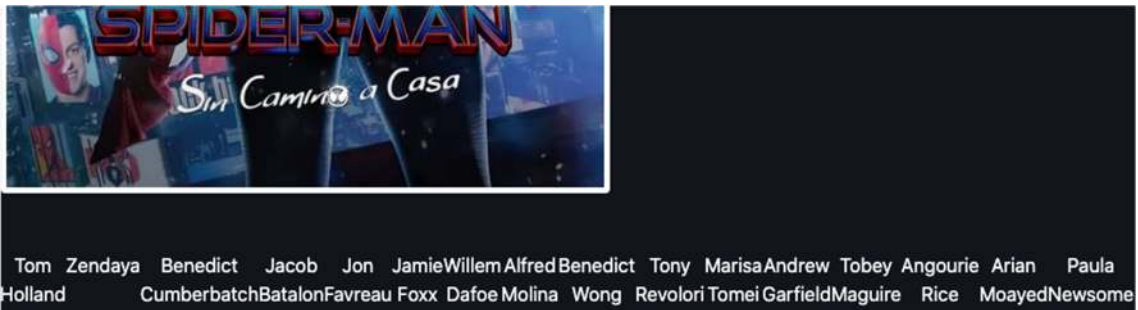
En cast-slideshow.component.html:

```

1
2
3 <div class="swiper-container">
4   <div class="swiper-wrapper">
5     <div *ngFor="let actor of cast">
6       <div class="swiper-slide text-white text-center">
7         {{ actor.name }}
8       </div>
9     </div>
10  </div>
11 </div>

```

Resultado. Hay que no cuadra porque hay muchos más línea que 5.3. Y además no se puede mover:



Para poner las imágenes:

```

src > app > components > cast-slideshow > cast-slideshow.component.html > div.swiper-contai
Go to component
1
2
3 <div class="swiper-container">
4   <div class="swiper-wrapper">
5     <div *ngFor="let actor of cast">
6       <div class="swiper-slide text-white text-center">
7         <img [src]="actor.profile_path" [alt]="actor.name">
8         {{ actor.name }}
9       </div>
10    </div>
11  </div>
12 </div>
13 </div>

```

No está definida esta propiedad así en la interface credits-response.ts:

```

16 export interface Cast {
17   adult: boolean;
18   gender: number;
19   id: number;
20   knownForDepartment: string;
21   name: string;
22   originalName: string;
23   popularity: number;
24   profile_path: null | string;

```

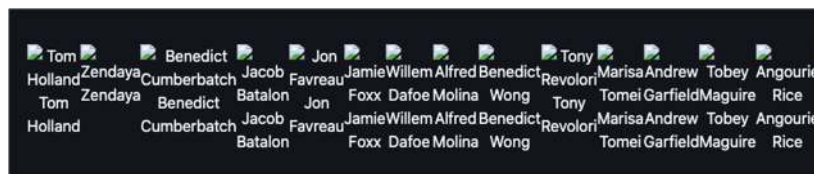
Ahora no hay error:

```

3  <div class="swiper-container">
4    <div class="swiper-wrapper">
5      <div *ngFor="let actor of cast">
6        <div class="swiper-slide text-white text-center">
7          <img [src]="actor.profile_path" [alt]="actor.name">
8            {{ actor.name }}
9          </div>
10        </div>
11      </div>
12    </div>
13  </div>

```

Sale esto:



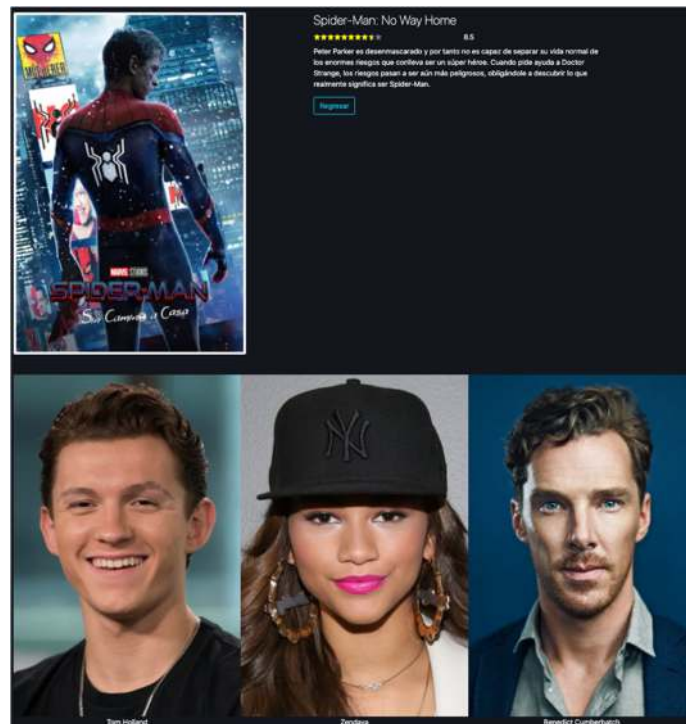
Obviamente no es lo que queremos, lo arreglo con el pipe poster:

```

<div class="swiper-slide text-white text-center">
  <img [src]="actor.profile_path | poster" [alt]="actor.name">
  {{ actor.name }}
</div>

```

Resultado. Todavía las fotos son muy grandes:




Le agregamos un par de clases para ajustar el tamaño de las imágenes, agrego la clase thumbnail que es propia de bootstrap:

```

3 <div class="swiper-container">
4   <div class="swiper-wrapper">
5     <div *ngFor="let actor of cast">
6       <div class="swiper-slide text-white text-center">
7
8         <img [src]="actor.profile_path | poster"
9           [alt]="actor.name"
10          class="img-thumbnail"/>
11         {{ actor.name }}
12       </div>
13     </div>
14   </div>
15 </div>

```

Resultado:






















Spider-Man: No Way Home

★★★★★ 8.5

Peter Parker es desenmascarado y por tanto no es capaz de separar su vida normal de los enormes riesgos que conlleva ser un súper héroe. Cuando pide ayuda a Doctor Strange, los riesgos pasan a ser aún más peligrosos, obligándole a descubrir lo que realmente significa ser Spider-Man.

[Regresar](#)

 Tom Holland
  Zendaya
  Benedict Cumberbatch
  Jacob Batalon
  Jon Favreau
  Jamie Foxx
  Willem Dafoe
  Alfred Molina
  Benedict Wong
  Tony Revolori
  Marisa Tomei
  Andrew Garfield
  Tobey Maguire
  Angourie Rice
  Arian Moayed
  Paula Newsome
  Hannibal Buress
  Martin Starr
  J.B. Smoove

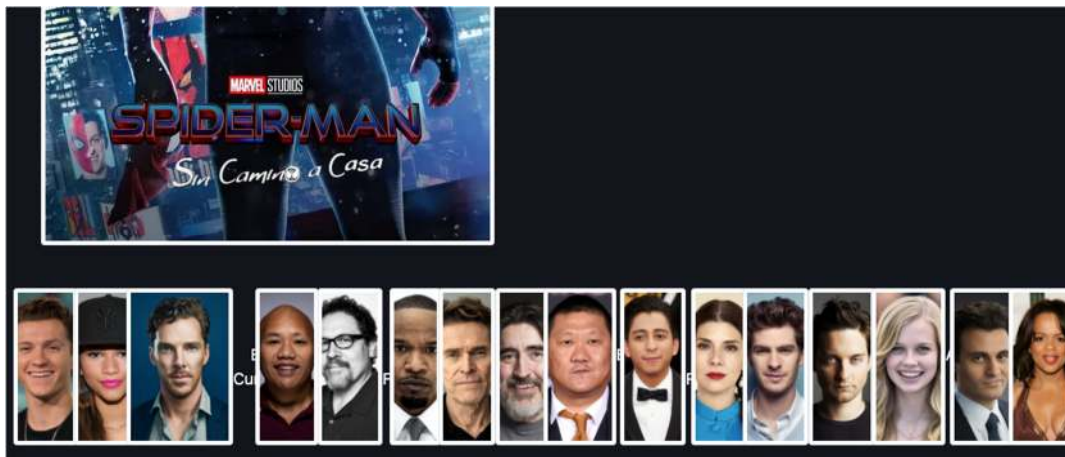
Modifico el css de cat-slideshow para que todas las imágenes sean del mismo tamaño:

```

src > app > components > cat-slideshow > cat-slideshow.component.scss > .swiper-c
1  html, body {
2      position: relative;
3      height: 100%;
4  }
5  body {
6      background: #eee;
7      font-family: Helvetica Neue, Helvetica, Arial, sans-serif;
8      font-size: 14px;
9      margin: 0;
10     padding: 0;
11 }
12 .swiper-container {
13     width: 100%;
14     height: 100%;
15     margin-left: auto;
16     margin-right: auto;
17 }
18
19 .swiper-wrapper {
20     width: 50%;
21 }
22
23 .swiper-slide {
24     text-align: center;
25     font-size: 18px;
26
27     /* Center slide text vertically */
28     display: -webkit-box;
29     display: -ms-flexbox;
30     display: -webkit-flex;
31     display: flex;
32     -webkit-box-pack: center;
33     -ms-flex-pack: center;
34     -webkit-justify-content: center;
35     justify-content: center;
36     -webkit-box-align: center;
37     -ms-flex-align: center;
38     -webkit-align-items: center;
39     align-items: center;
40 }
41
42
43 .swiper-slide img {
44     display: block;
45     width: 100%;
46     height: 100%;
47     object-fit: cover;
48 }

```


El resultado es:



Seguimos con varios problemas:

1. El slide-show.casting no contiene 5,3 imágenes por fila.
2. No se puede mover.
3. El nombre del actor es tapado por la foto.

Tras analizar el código, me he dado cuenta de que me he dejado una clase de bootstrap sin poner en `app.cas-slideshow.component.html` en el `ngFor`. Lo teníamos en un `div` aparte:

```

6   <div class="swiper-container">
7     <div class="swiper-container">
8       <div class="swiper-wrapper">
9
10        <div *ngFor="let actor of cast"
11          class="swiper-slide text-white text-center">
12
13          <img [src]="actor.profile_path | poster"
14            [alt]="actor.name"
15            class="img-thumbnail"/>
16
17          {{ actor.name }}
18        </div>
19      </div>
20    </div>
21  </div>

```

Resultado en el pie de página con todos los problemas solucionados:

