

## Objetivos:

En esta aplicación trabajaremos con una API muy popular llamada TheMovieDB API, para hacer una aplicación de búsqueda de películas, puntualmente tocaremos temas como:

- Http Get
- Operadores de RXJS como el Tap y Map
- Combinar observables
- Módulos
- Estructura de un proyecto real
- Servicios
- Slideshow
- Swiper

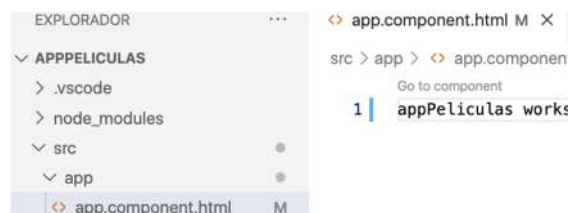
## Paso 1. Creación del proyecto.

```

Last login: Fri Jan 21 16:26:42 on console
jsersan@iMac-de-Jose ~ % cd /Applications/MAMP/htdocs/angular
jsersan@iMac-de-Jose angular % ng new appPelículas
? Would you like to add Angular routing? No
? Which stylesheet format would you like to use? SCSS [ https://sass-lang.com/documentation/syntax#scss ]
CREATE appPelículas/README.md (1058 bytes)
CREATE appPelículas/.editorconfig (274 bytes)
CREATE appPelículas/.gitignore (620 bytes)
CREATE appPelículas/angular.json (3244 bytes)
CREATE appPelículas/package.json (1077 bytes)
CREATE appPelículas/tsconfig.json (863 bytes)
CREATE appPelículas/.browserslistrc (600 bytes)
CREATE appPelículas/karma.conf.js (1430 bytes)
CREATE appPelículas/tsconfig.app.json (287 bytes)
CREATE appPelículas/tsconfig.spec.json (333 bytes)
CREATE appPelículas/.vscode/extensions.json (130 bytes)
CREATE appPelículas/.vscode/launch.json (474 bytes)
CREATE appPelículas/.vscode/tasks.json (938 bytes)
CREATE appPelículas/src/favicon.ico (948 bytes)
CREATE appPelículas/src/index.html (298 bytes)
CREATE appPelículas/src/main.ts (372 bytes)
CREATE appPelículas/src/polyfills.ts (2338 bytes)
CREATE appPelículas/src/styles.scss (80 bytes)
CREATE appPelículas/src/test.ts (745 bytes)
CREATE appPelículas/src/assets/.gitkeep (0 bytes)
CREATE appPelículas/src/environments/environment.prod.ts (51 bytes)
CREATE appPelículas/src/environments/environment.ts (658 bytes)
CREATE appPelículas/src/app/app.module.ts (314 bytes)
CREATE appPelículas/src/app/app.component.scss (0 bytes)
CREATE appPelículas/src/app/app.component.html (23332 bytes)
CREATE appPelículas/src/app/app.component.spec.ts (974 bytes)
CREATE appPelículas/src/app/app.component.ts (217 bytes)

```

## Paso 2.- Abrimos el proyecto creado con Visual Studio:



## Paso 3. Configuración de Bootstrap. De la documentación de bootstrap.

### JS

Many of our components require the use of JavaScript to function. Specifically, they require our own JavaScript plugins and [Popper](#). Place **one of the following <script>s** near the end of your pages, right before the closing </body> tag, to enable them.

### Bundle

Include every Bootstrap JavaScript plugin and dependency with one of our two bundles. Both [bootstrap.bundle.js](#) and [bootstrap.bundle.min.js](#) include [Popper](#) for our tooltips and popovers. For more information about what's included in Bootstrap, please see our [contents](#) section.

```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js">
```

Copy

### Separate #

If you decide to go with the separate scripts solution, Popper must come first (if you're using tooltips or popovers), and then our JavaScript plugins.

```
<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.10.2/dist/umd/popper.min.js">
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.min.js" integr
```

Copy

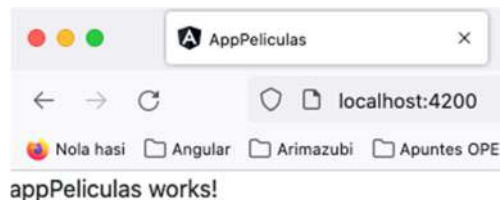
Copiamos las librerías de bootstrap en index.html:

```

<> index.html M X <> app.component.html M
src > <> index.html > html > head
1  <!doctype html>
2  <html lang="en">
3  <head>
4      <meta charset="utf-8">
5      <title>AppPeliculas</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      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
10
11 </head>
12 <body>
13     <app-root></app-root>
14 </body>
15 </html>

```

Levantamos el proyecto.



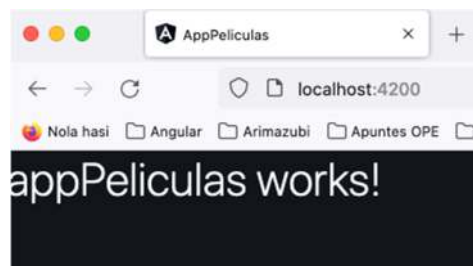
Paso 4. Copiamos el archivo de estilos del material adjunto y lo pegamos en styles.scss

```

<> index.html M <> styles.scss M X <> app.component.html M
src > <> styles.scss > .block
1  /* You can add global styles to this file, and also import other style files */
2
3  body {
4      background-color: #14161E;
5  }
6
7  h1, h2, h3, h4, h5 {
8      font-weight: 200;
9      color: white;
10

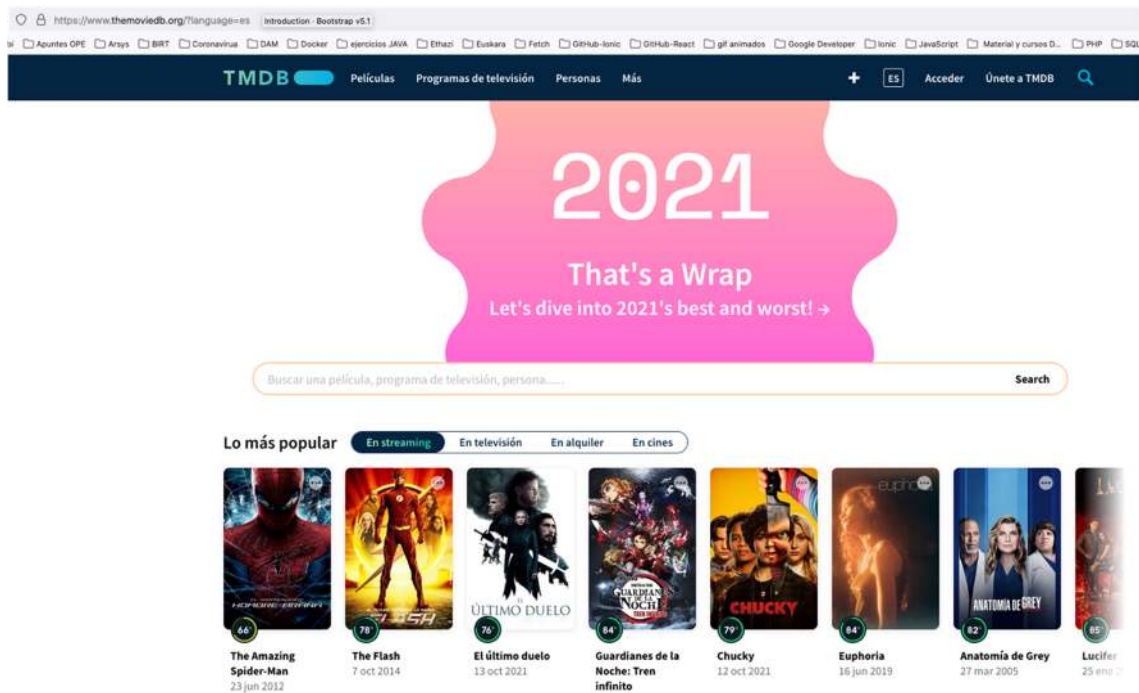
```

Resultado:

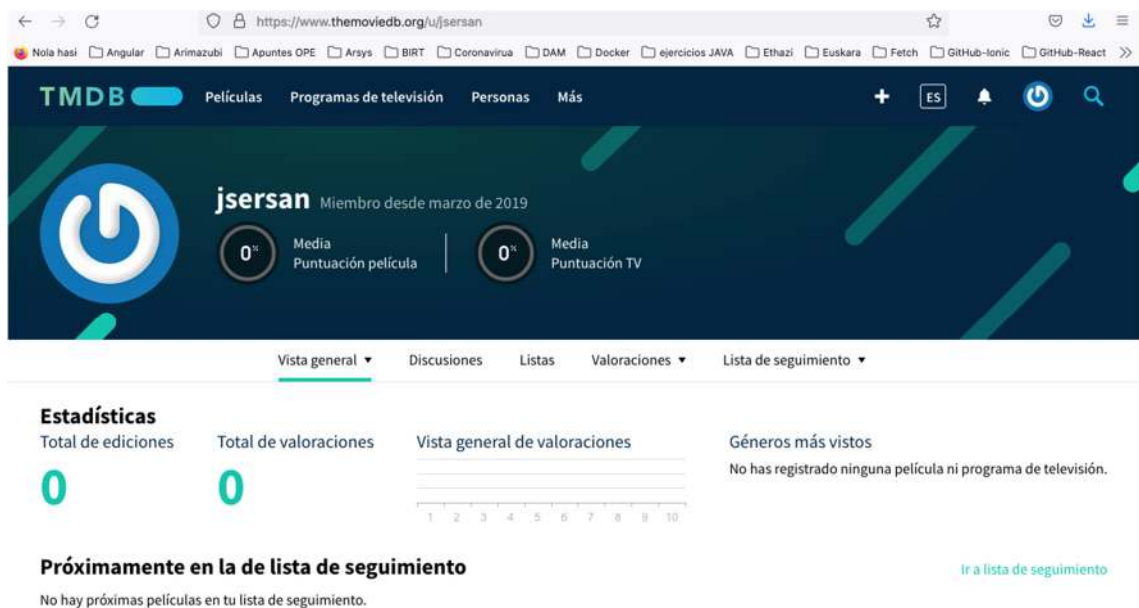


La imagen la colocamos dentro de los Assets.

Paso 5. API movieDB.



Generamos una apiKey en la página oficial. Es necesario previamente crear una cuenta.



En esta página creamos una APIKEY. Si no la tenemos, creamos la cuenta y accedemos:

**Correo electrónico verificado**  
Su correo electrónico ha sido verificado con éxito. Ya puedes iniciar sesión.

**Accede a tu cuenta**  
Para poder editar y valorar en TMDb, así como para obtener recomendaciones personales, deberás acceder con tu cuenta. Si no tienes una, registrarse para obtenerla es gratis y simple. [Pulsa aquí](#) para empezar.  
Si ya te has registrado pero aún no has recibido el correo de confirmación, [pulsa aquí](#) para enviárselo de nuevo.

Usuario

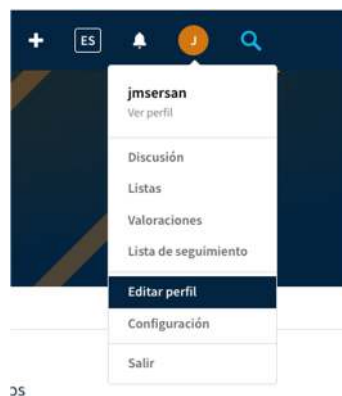
Contraseña

[Acceder](#) [Restablecer contraseña](#)

Tras acceder:

The image shows the user profile page for 'jmsersan' on TMDb. The header includes the user's name, join date (January 2022), and average ratings for movies (0) and TV shows (0). Below the header is a navigation bar with tabs: Vista general (selected), Discusiones, Listas, Valoraciones, and Lista de seguimiento. The main content area is divided into several sections: 'Estadísticas' showing 0 editions and 0 ratings; 'Próximamente en la lista de seguimiento' with a message that no movies are currently in the list; 'Discusiones recientes' with a message that no discussions are being followed; and 'Actividad reciente' with a message that no edits have been made recently. A 'Ver más' link is present at the bottom right of the activity section.

Vamos a Editar Perfil:



Le damos a API:

**API** Vista general Crear

TMDb offers a powerful API service that is free to use as long as you properly attribute us as the source of the data and/or images you use. You can find the logos for attribution [here](#).

#### Documentación

Our primary documentation is located at [developers.themoviedb.org](https://developers.themoviedb.org).

#### Soporte

If you have questions or comments about the information covered here, please create a post on our [support forums](#).

#### Solicitar una clave de API

To generate a new API key, [click here](#).

Solicitamos una clave de API:

**API** Vista general Crear

What type of API key do you wish to register?

#### Developer

- You are an individual
- Your project is still in development
- Your project is non profit
- Your project is ad supported

#### Professional

- You represent a company
- Your project is for profit (not ad supported)
- You are an OEM or hardware vendor

En Developer y aceptando los Términos de Uso, rellenamos este formulario:

**API** Vista general Crear

Tipo de uso

Aplicación de escritorio

Nombre de la aplicación

URL de la aplicación

Si todo va bien obtenemos una clave de API para appPelículas:



## Clave de la API (v3 auth)

50dadd4cc349d67261295f7d03041022

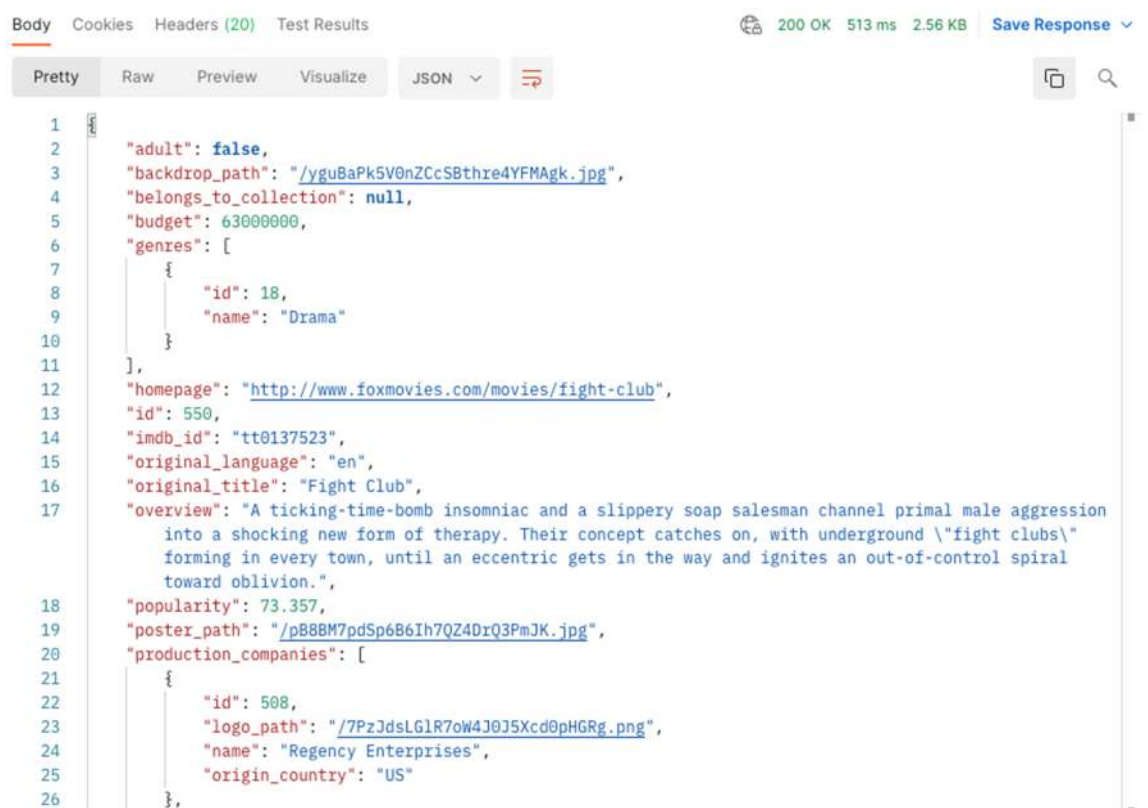
### Ejemplo de solicitud de API

[https://api.themoviedb.org/3/movie/550?api\\_key=50dadd4cc349d67261295f7d03041022](https://api.themoviedb.org/3/movie/550?api_key=50dadd4cc349d67261295f7d03041022)

## Token de acceso de lectura a la API (v4 auth)

yehJbgGciOiiJIUzI1NI9.yeyJhdWQioiI1MGRhZGQ0Y2mNDIkNjcyNjEyOTVmN2QwMA0MTAyMiIsnI1Yil6ijYxZWJKMTQ3M2ZhYmEwMDExMTk2mJFINSIsInjb3Bclwy6WhYcGlfcvmVhZCJdL.CJ2XJzaW9uijoxfQ.952N6k8mXQsvkD9lIdruk36pxgwn79NudLr5eK2kt

Probamos el ejemplo de solicitud de API con PostMan.



A screenshot of the QuickType website. The header is dark blue with the QuickType logo (a green circle with 'qt' in white) and the text 'quicktype' in white. To the right are links for 'GITHUB' and 'BLOG' in white, and a pink button with the text 'OPEN QUICKTYPE' in white. Below the header, on the left, is a small grey box with the text 'aplicaciones'. The main headline is in large, white, sans-serif font, reading 'Convert JSON into gorgeous, typesafe code in any language.'

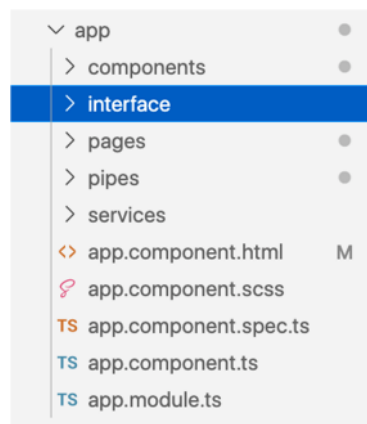
### Paso 6. Módulos en Angular.

Crearemos diferentes módulos en función de las secciones donde trabajemos:

1. Components.
2. Pipes: para filtrar datos.
3. Pages: home, búsqueda y detalles de las películas.

```
jsersan@iMac-de-Jose appPelículas % ng g m components
CREATE src/app/components/components.module.ts (196 bytes)
jsersan@iMac-de-Jose appPelículas % ng g m pipes
CREATE src/app/pipes/pipes.module.ts (191 bytes)
jsersan@iMac-de-Jose appPelículas % ng g m pages
CREATE src/app/pages/pages.module.ts (191 bytes)
jsersan@iMac-de-Jose appPelículas %
```

Luego creo a mano la carpeta services e interface:



Recordad que los módulos son agrupadores de contenidos. Creamos los tres componentes de las páginas de nuestro sitio web:

```
jsersan@iMac-de-Jose appPelículas % ng g c pages/home --skip-tests
CREATE src/app/pages/home/home.component.scss (0 bytes)
CREATE src/app/pages/home/home.component.html (19 bytes)
CREATE src/app/pages/home/home.component.ts (268 bytes)
UPDATE src/app/pages/pages.module.ts (271 bytes)
jsersan@iMac-de-Jose appPelículas %
```

Lo mismo para película y buscar.

Como estos archivos se crearon a mano, sería conveniente que bajáramos y subiéramos de nuevo el proyecto.



## Paso 7. Obtener películas en cartelera.

Recordamos el JSON obtenido en PostMan:

https://api.themoviedb.org/3/movie/now\_playing?api\_key=50dadd4cc349d67261295f7d03041022&language=en-US...

GET https://api.themoviedb.org/3/movie/now\_playing?api\_key=50dadd4cc349d67261295f7d03041022&language=en-US&t...

Params Authorization Headers (5) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION
api_key	50dadd4cc349d67261295f7d03041022	
language	en-US	
page	1	

Body Cookies Headers (15) Test Results Status: 200 OK Time: 368 ms Size: 13.91 KB Save Response

Pretty Raw Preview Visualize JSON

```

1  {
2    "dates": {
3      "maximum": "2022-01-27",
4      "minimum": "2021-12-10"
5    },
6    "page": 1,
7    "results": [
8      {

```

Si copiamos todo el JSON dándole al botón marcado y vamos la página [quicktype.io](https://quicktype.io)

quicktype

Name: Welcome Source type: JSON

To parse this data:

```

import { Convert, Welcome } from "./file";
const welcome = Convert.toWelcome(json);

// These functions will throw an error if the JSON doesn't
// match the expected interface, even if the JSON is valid.

export interface Welcome {
  adult: boolean;
  backdrop_path: string;
  belongs_to_collection: null;
  budget: number;
  genres: Genref[];
  homepage: string;
  id: number;
  imdb_id: string;
  original_language: string;
  original_title: string;
  overview: string;
  popularity: number;
  poster_path: string;
  production_companies: ProductionCompany[];
  production_countries: ProductionCountry[];
  release_date: Date;
  revenue: number;
  runtime: number;
  spoken_languages: SpokenLanguage[];
  status: string;
  tagline: string;
  title: string;
  video: boolean;
}

```

Language: TypeScript

Options: Interfaces only, Transform property names to be JavaScript, Explicitly name unions, Verify JSON parse results at runtime, Make all properties optional

Copy Code

Me crea las interfaces.

Name: CarteleraResponse Source type: JSON

To parse this data:

```

// To parse this data:
// import { Convert, CarteleraResponse } from "./file";
// const carteleraResponse = Convert.toCarteleraResponse(json);
// These functions will throw an error if the JSON doesn't
// match the expected interface, even if the JSON is valid.

export interface CarteleraResponse {
  adult: boolean;
  backdrop_path: string;
}

```

En el proyecto, en la carpeta interface creo un fichero `cartelera.response.ts`:

```

9
10 export interface CarteleraResponse {
11     dates:      Dates;
12     page:       number;
13     results:    Movie[];
14     totalPages: number;
15     totalResults: number;
16 }
17
18 export interface Dates {
19     maximum: Date;
20     minimum: Date;
21 }
22
23 export interface Movie {
24     adult:      boolean;
25     backdropPath: null | string;

```

Le cambio el nombre a la interface y results será ahora un arreglo de Movie. Ahora falta hacer la petición http y construir el arreglo de rutas.

Para consumir este servicio, en app.module.ts (recordad importarlo):

```

src > app > TS app.module.ts > AppModule
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3
4  import { HttpClientModule } from '@angular/common/http';
5
6  import { AppComponent } from './app.component';
7
8  @NgModule({
9      declarations: [
10         AppComponent
11     ],
12     imports: [
13         BrowserModule,
14         HttpClientModule
15     ],

```

Creo un servicio para las películas:

```

jsersan@iMac-de-Jose appPelículas % ng g s services/peliculas
CREATE src/app/services/peliculas.service.spec.ts (372 bytes)
CREATE src/app/services/peliculas.service.ts (138 bytes)
jsersan@iMac-de-Jose appPelículas %

```

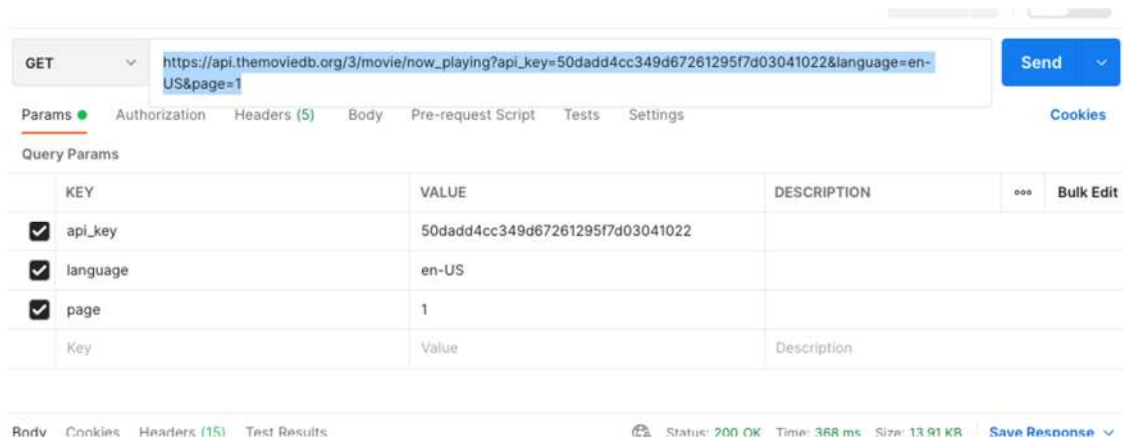
Ahora lo editamos e importamos el módulo httpClient para hacer las peticiones.

```

src > app > services > TS peliculas.service.ts > ...
1  import { HttpClient } from '@angular/common/http';
2  import { Injectable } from '@angular/core';
3
4  @Injectable({
5    providedIn: 'root'
6  })
7  export class PeliculasService {
8
9    constructor( private http: HttpClient) { }
10
11    getCartelera(){
12
13      return this.http.get()
14    }
15  }
16

```

El parámetro del get es la ruta de la petición del postMan:



En el servicio:

```

9    constructor( private http: HttpClient) { }
10
11    getCartelera(){
12
13      return this.http.get('https://api.themoviedb.org/3/movie/now_playing
14    }
15  }
16

```

Guardamos los cambios. Utilizo este servicio en app.component.ts.

```

12  constructor( private peliculasService: PeliculasService){
13    this.peliculasService.getCartelera()
14      .subscribe( resp => {
15        console.log(resp);
16      })
17  }

```

Ejecutamos y en la consola:



Solo me falta indicarle que la respuesta de qué tipo es la respuesta. Eso lo hacemos con observables.

```

3 import { Observable } from 'rxjs';
4 import { CarteleraResponse } from '../interface/cartelera-response';
5
6 @Injectable({
7   providedIn: 'root'
8 })
9 export class PeliculasService {
10
11   constructor( private http: HttpClient) { }
12
13   getCartelera():Observable<CarteleraResponse>{
14
15     return this.http.get('https://api.themoviedb.org/3/movie/now_playing?api_key=8ce924342c266e507090b60881062271');
16
17   }
18 }

```

Me da error porque debemos especificar de qué tipo es el objeto que retorna en getCartelera() del tipo CarteleraResponse.

```

9 export class PeliculasService {
10
11   constructor( private http: HttpClient) { }
12
13   getCartelera():Observable<CarteleraResponse>{
14
15     return this.http.get<CarteleraResponse>('https://api.themoviedb.org/3/movie/now_playing?api_key=8ce924342c266e507090b60881062271');
16
17   }
18 }

```

Ahora en app.component.ts, el tipo de respuesta nos permite elegir:



Por ello puedo acceder a todas la propiedades del objeto:





Paso 8. Implementar rutas en nuestra aplicación.

Aunque sólo tengamos tres páginas, queremos navegar por las tres. Generamos un nuevo módulo `appRouting`:

```
jsersan@iMac-de-Jose appPelículas % ng g m appRouting --flat
CREATE src/app/app-routing.module.ts (196 bytes)
jsersan@iMac-de-Jose appPelículas %
```

Así este fichero `app-routing.module.ts`:

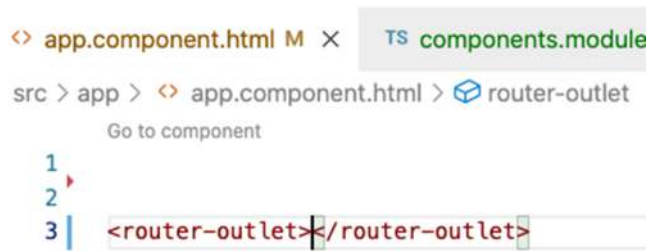
```
8   const routes: Routes = [
9     {
10      path: 'home',
11      component: HomeComponent
12    },
13    {
14      path: 'película/:id',
15      component: PelículaComponent
16    },
17    {
18      path: 'buscar/:texto',
19      component: BuscarComponent
20    },
21    {
22      path: '**',
23      redirectTo: '/home'
24    }
25  ];
26
27  @NgModule({
28    declarations: [],
29    imports: [
30      CommonModule,
31      RouterModule.forRoot(routes)
32    ],
33    exports: [
34      RouterModule
35    ]
36  })
37  export class AppRoutingModule { }
```

Importamos el `app-routing.module` en `app.module.ts` en los `import`:

```
13   imports: [
14     BrowserModule,
15     HttpClientModule,
16     AppRoutingModule
```

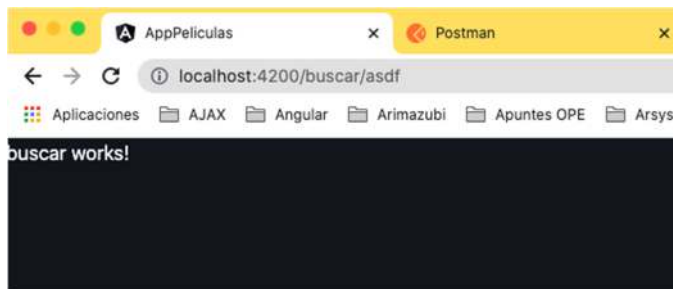


Ahora, en app.component.html:

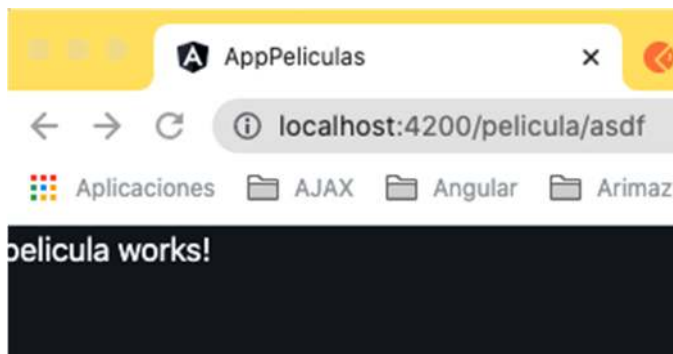


```
<> app.component.html M x TS components.module
src > app > <> app.component.html > router-outlet
Go to component
1
2
3 | <router-outlet>|
```

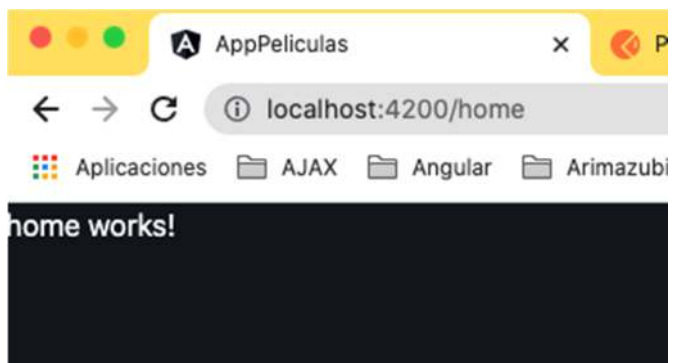
Probamos. En la url ponemos localhost:4200/buscar/



Con película:



Cualquier otra cosa, al home.



Se redirige perfectamente. Las rutas funcionan OK.

Paso 9. Implementar una barra de navegación.

Vamos a incluir un objeto navbar en la app.component.html. Creamos el componente:

```
jsersan@iMac-de-Jose appPelículas % ng g c components/navbar --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/navbar/navbar.component.scss (0 bytes)
CREATE src/app/components/navbar/navbar.component.html (21 bytes)
CREATE src/app/components/navbar/navbar.component.ts (276 bytes)
UPDATE src/app/components/components.module.ts (278 bytes)
jsersan@iMac-de-Jose appPelículas %
```

Vemos el component.module.ts:

```
1  import { NgModule } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3  import { NavbarComponent } from './navbar/navbar.component';
4
5  @NgModule({
6    declarations: [
7      NavbarComponent
8    ],
9    imports: [
10     CommonModule
11   ]
12 })
13 export class ComponentsModule { }
```

Como lo voy a utilizar fuera, en components.module.ts, debemos exportarlo:



```
<> app.component.html M    TS components.module.ts U ×    TS pages.module.ts
src > app > components > TS components.module.ts > ComponentsModule
1  import { NgModule } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3  import { NavbarComponent } from './navbar/navbar.component';
4
5  @NgModule({
6    declarations: [
7      NavbarComponent
8    ],
9    imports: [
10     CommonModule,
11   ],
12    exports: [
13     NavbarComponent
14   ]
15 })
16 export class ComponentsModule { }
17
```

Este components.module.ts lo debemos importar en app.module.ts:

```

8 | import { ComponentsModule } from './components/components.module';
9 |
10 | @NgModule({
11 |   declarations: [
12 |     AppComponent
13 |   ],
14 |   imports: [
15 |     BrowserModule,
16 |     HttpClientModule,
17 |     AppRoutingModule,
18 |     ComponentsModule
19 |   ],
20 |   providers: [],
21 |   bootstrap: [AppComponent]
22 | })
23 | export class AppModule {}
24 |

```

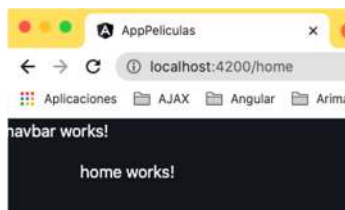
Ahora puedo colocar el navbar en app.component.html:

```

<> app.component.html M x TS components.module.ts U
src > app > <> app.component.html > app-navbar
Go to component
1 | <!-- Navbar -->
2 |
3 | <app-navbar></app-navbar>
4 |
5 | <div class="container mat-5">
6 |   <router-outlet></router-outlet>
7 | </div>

```

Resultado:



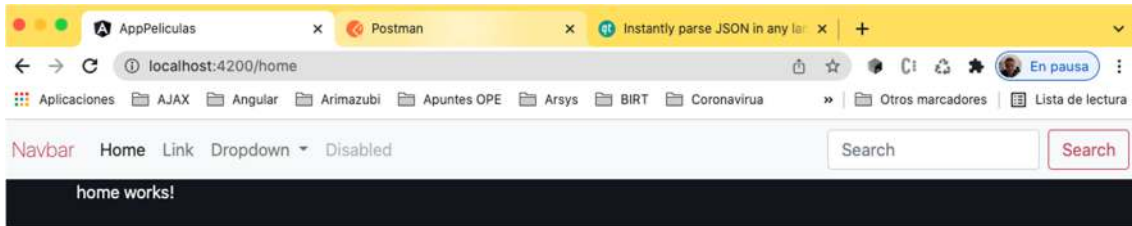
Ahora copiamos de bootstrap el código de un navbar.

```

<> navbar.component.html U x <> app.component.html M TS components.module.ts U TS pag
src > app > components > navbar > <> navbar.component.html > nav.navbar.navbar-expand-lg.navbar-light
Go to component
1 | <nav class="navbar navbar-expand-lg navbar-light bg-light">
2 |   <div class="container-fluid">
3 |     <a class="navbar-brand" href="#">Navbar</a>
4 |     <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
5 |       <span class="navbar-toggler-icon"></span>
6 |     </button>
7 |     <div class="collapse navbar-collapse" id="navbarSupportedContent">
8 |       <ul class="navbar-nav me-auto mb-2 mb-lg-0">
9 |         <li class="nav-item">
10 |           <a class="nav-link active" aria-current="page" href="#">Home</a>
11 |         </li>
12 |         <li class="nav-item">
13 |           <a class="nav-link" href="#">Link</a>
14 |         </li>

```

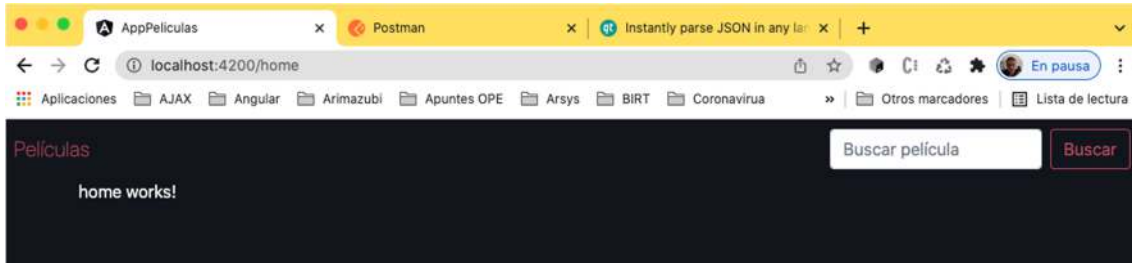
Resultado:



Hacemos unas modificaciones:



Resultado:



Si quiero que cuando haga click en Películas vaya al /home:



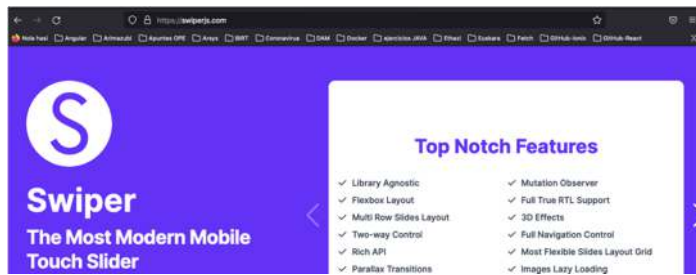
Debo importar el routerLink en components.module.ts:

```
src > app > components > TS components.module.ts > ComponentsModule
1  import { NgModule } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3  import { NavbarComponent } from '../navbar/navbar.component';
4  import { RouterModule } from '@angular/router';
5
6  @NgModule({
7    declarations: [
8      NavbarComponent
9    ],
10   imports: [
11     CommonModule,
12     RouterModule
13   ],
14   exports: [
15     NavbarComponent
16   ]
17 })
18 export class ComponentsModule { }
19
```

Aunque la importación la hemos declarado en más de un módulo, sólo lo hace una vez. La segunda vez cuando la encuentre en la caché va a usar éste y no se duplica. Al probar y poner el ratón sobre el enlace cambia la forma del icono a una mano.

Paso 10. Implementar un Slider para mostrar las últimas películas del cartel.

Lo sacamos de swiperjs.com:



Lo primero es crear un componente slider:

```
jsersan@iMac-de-Jose appPelículas % ng g c components/slideshow --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/slideshow/slideshow.component.scss (0 bytes)
CREATE src/app/components/slideshow/slideshow.component.html (24 bytes)
CREATE src/app/components/slideshow/slideshow.component.ts (288 bytes)
UPDATE src/app/components/components.module.ts (475 bytes)
```

Debemos exportarlo como hicimos con navbar en components.module.ts:

```
4 import { RouterModule } from '@angular/router';
5 import { SlideshowComponent } from './slideshow/slideshow.component';
6
7 @NgModule({
8   declarations: [
9     NavbarComponent,
10    SlideshowComponent
11  ],
12   imports: [
13     CommonModule,
14     RouterModule
15  ],
16   exports: [
17     NavbarComponent,
18     SlideshowComponent
19  ]
20 })
21 export class ComponentsModule { }
```

También debemos importarlo en pages.modules.ts:

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



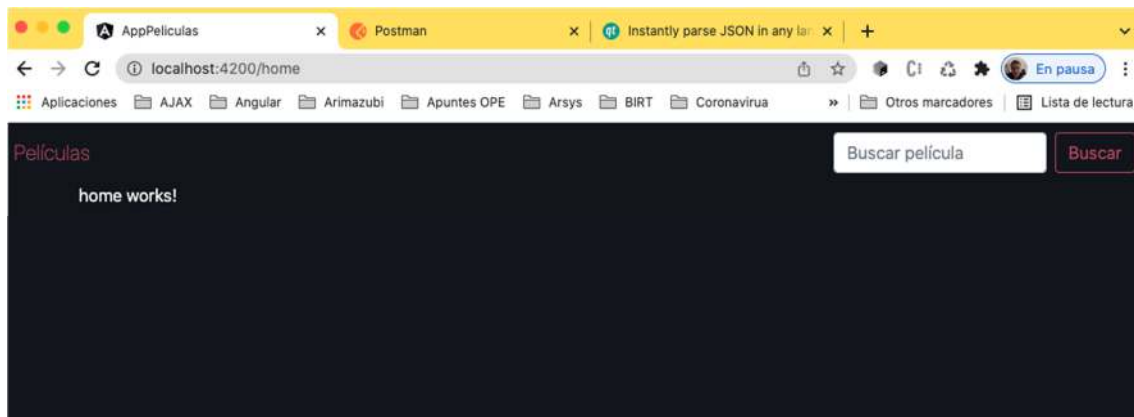
En home incluyo este componente:

```
src > app > pages > home > <> home.component.html >  
Go to component  
1 <p>home works!</p>  
2  
3 <app-slideshow></app-slideshow>
```

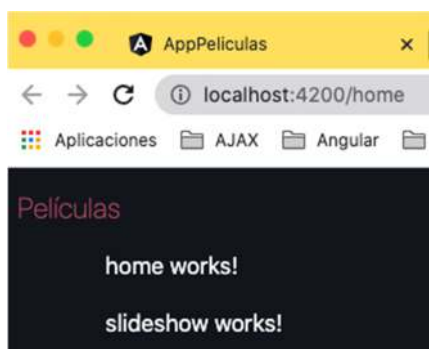
En app.module.ts debemos importar pagesModule:

```
9 import { ComponentsModule } from '../components/components.module';  
10  
11 import { PagesModule } from './pages/pages.module';  
12  
13 @NgModule({  
14   declarations: [  
15     AppComponent  
16   ],  
17   imports: [  
18     BrowserModule,  
19     HttpClientModule,  
20     AppRoutingModule,  
21     ComponentsModule,  
22     PagesModule  
23   ],  
24   providers: [],  
25   bootstrap: [AppComponent]  
26 })  
27 export class AppModule { }
```

Guardamos y en el resultado vemos como no aparece todavía:



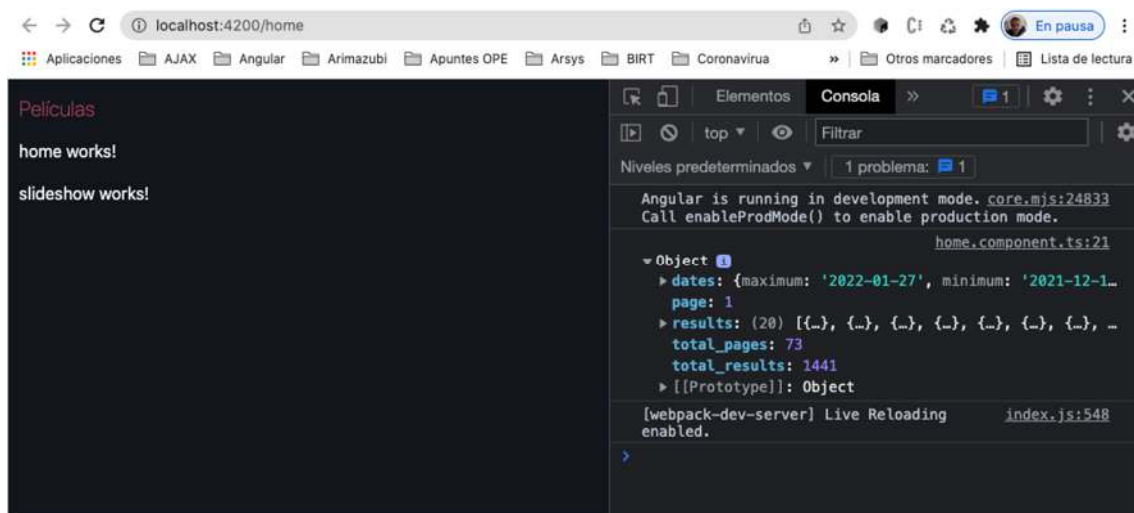
Bajamos y subimos el proyecto:



El homePage va a ser quien lance la petición http para las películas. Hay que moverlo del app.component al homePage. El archivo home.component.ts se queda:

```
src > app > pages > home > TS home.component.ts > HomeComponent > ngOnInit
1  import { Component, OnInit } from '@angular/core';
2  import { PeliculasService } from 'src/app/services/peliculas.service';
3
4  @Component({
5    selector: 'app-home',
6    templateUrl: './home.component.html',
7    styleUrls: ['./home.component.scss']
8  })
9  export class HomeComponent implements OnInit {
10
11    constructor( private peliculasService: PeliculasService){
12
13    }
14
15    ngOnInit(): void {
16
17      // getCartelera
18
19      this.peliculasService.getCartelera()
20        .subscribe( resp => {
21          console.log(resp);
22        })
23
24    }
25
26  }
```

Resultado:



El array de películas una vez cargado debemos enviarlo al slideshow.

Así en [home.component.ts](#) nos creamos un array de esas películas:

```

10 export class HomeComponent implements OnInit {
11
12   public movies: Movie[] = [];
13
14   constructor( private peliculasService: PeliculasService){}
15
16   ngOnInit(): void {
17
18     // getCartelera
19
20     this.peliculasService.getCartelera()
21       .subscribe( resp => {
22         console.log(resp);
23       })
24
25   }
26
27 }

```

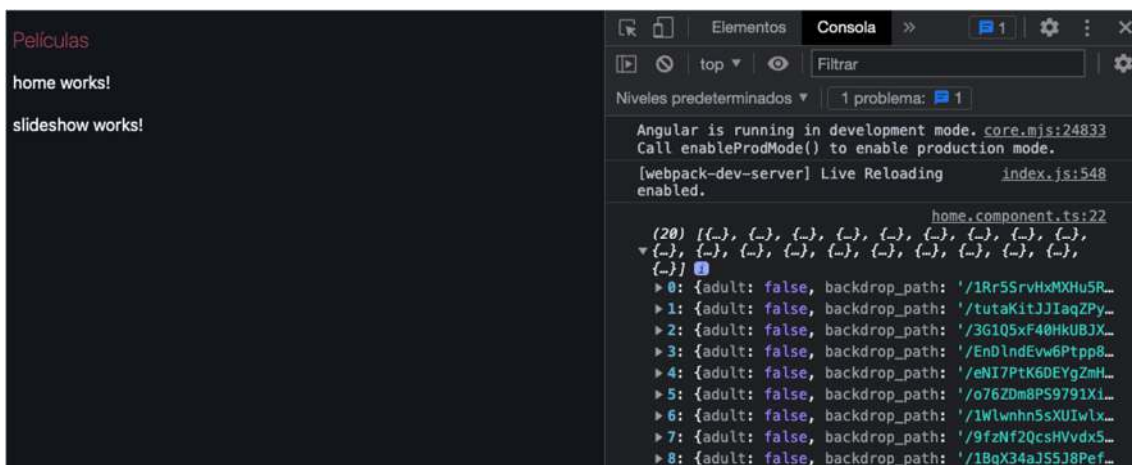
Así la llamada al servicio queda:

```

20     this.peliculasService.getCartelera()
21       .subscribe( resp => {
22         console.log(resp.results);
23         this.movies = resp.results;
24       })

```

Resultado:



Pero esta petición debemos enviarla al componente hijo: el slideshow.

```

<> home.component.html 1, U × TS components.module.ts U TS
src > app > pages > home > <> home.component.html > ...
Go to component
1 <p>home works!</p>
2
3 <app-slideshow [movies]="movies"></app-slideshow>
4

```

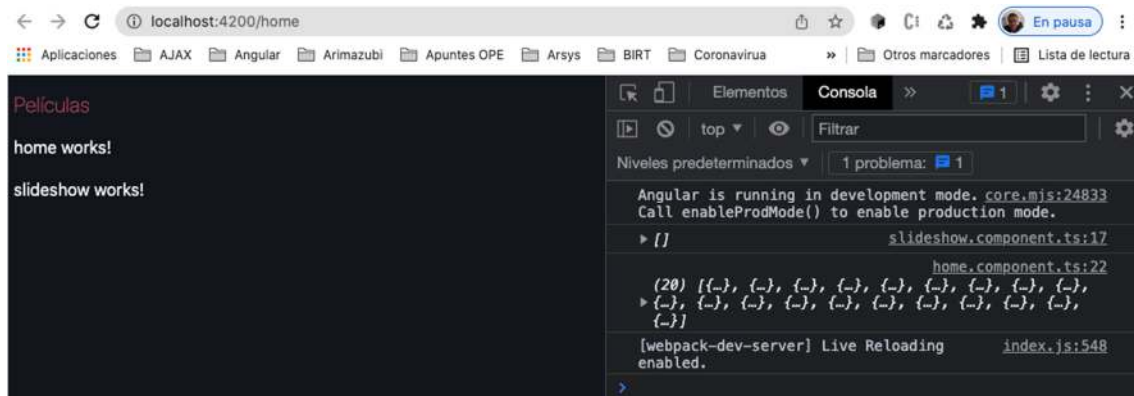
Este objeto movies no está definido y por eso nos marca el error.  
En el componente slideshow.component.ts:

```

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

```

Guardamos los cambios. Vemos como tengo un array vacío:



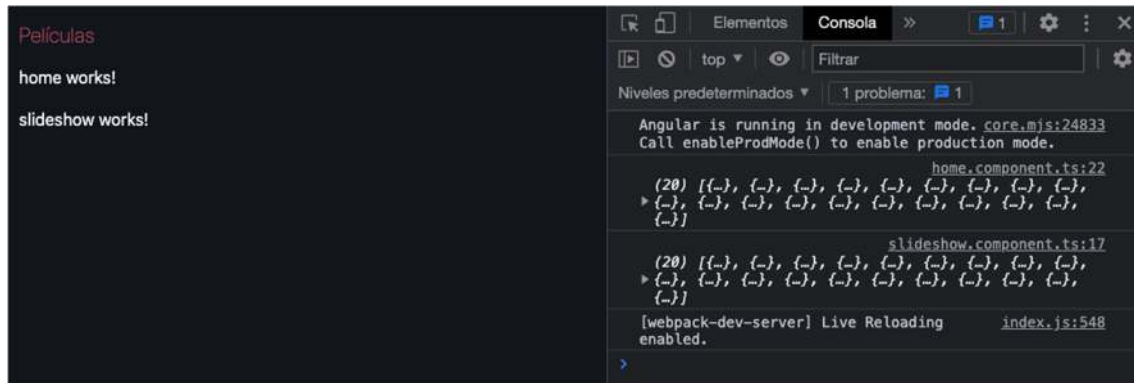
Debiera contener el valor del array que ha importado en home.component.ts. Para ello en home.component.html, importamos al slideshow si la longitud del array  $> 0$  :

```

<> home.component.html U X TS components.module.ts U TS pages.
src > app > pages > home > <> home.component.html > ...
Go to component
1 <p>home works!</p>
2
3 <div class="row" *ngIf="movies.length>0">
4   <div class="col">
5     <app-slideshow [movies]="movies"></app-slideshow>
6   </div>
7 </div>

```

El resultado:



## Paso 11. Implementar un Slider: uso del swiper.

En vez de importarlo en el index, podemos importar los módulos de node que hagan falta:

```
jsersan@iMac-de-José appPelículas % npm install swiper
added 3 packages, removed 1 package, and audited 898 packages in 6s

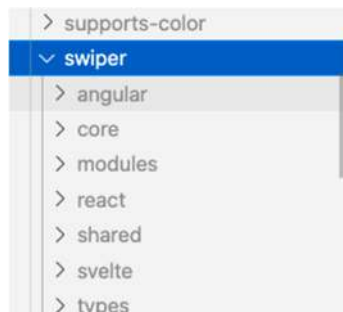
93 packages are looking for funding
  run `npm fund` for details

6 vulnerabilities (3 low, 3 moderate)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

Así, si supervisamos los módulos de node instalados:



En la referencia pone:

If you want to import Swiper with all modules (bundle) then it should be imported from **swiper/bundle**:

```
// import Swiper bundle with all modules installed
import Swiper from 'swiper/bundle';

// import styles bundle
import 'swiper/css/bundle';
```

Para que esto sea parte de mi aplicación, en el fichero angular.json vamos a poner la referencia al fichero de hojas de estilo swiper.css.

```
32 | | | | },
33 | | | | "styles": {
34 | | | |   "src/styles.scss",
35 | | | |   "node_modules/swiper/swiper-bundle.min.css"
36 | | | | },
37 | | | | "scripts": []
38 | | | | }
```

En la carpeta de node\_modules/swiper:

```
JS swiper-bundle.js.map
# swiper-bundle.min.css
JS swiper-bundle.min.js
```



Para que tenga en cuenta esta configuración del angular.json debemos bajar el proyecto y volverlo a subir. Si diera algún error significaría que la ruta del fichero está mal. Utilizaremos los ejemplos de la documentación oficial y copiamos este código:

**Add Swiper HTML Layout**

Now, we need to add basic Swiper layout to our app:

```

<!-- Slider main container -->
<div class="swiper">
  <!-- Additional required wrapper -->
  <div class="swiper-wrapper">
    <!-- Slides -->
    <div class="swiper-slide">Slide 1</div>
    <div class="swiper-slide">Slide 2</div>
    <div class="swiper-slide">Slide 3</div>
    ...
  </div>
  <!-- If we need pagination -->
  <div class="swiper-pagination"></div>

  <!-- If we need navigation buttons -->

```

Lo pegamos en slideshow.component.html:

slideshow.component.html U X home.component.html U

src > app > components > slideshow > slideshow.component.html

Go to component

```

1  <!-- Slider main container -->
2  <div class="swiper">
3    <!-- Additional required wrapper -->
4    <div class="swiper-wrapper">
5      <!-- Slides -->
6      <div class="swiper-slide">Slide 1</div>
7      <div class="swiper-slide">Slide 2</div>
8      <div class="swiper-slide">Slide 3</div>
9      ...
10   </div>
11   <!-- If we need pagination -->
12   <div class="swiper-pagination"></div>
13
14   <!-- If we need navigation buttons -->
15   <div class="swiper-button-prev"></div>
16   <div class="swiper-button-next"></div>
17
18   <!-- If we need scrollbar -->
19   <div class="swiper-scrollbar"></div>
20 </div>
21

```

Resultado:



Nos falta inicializar el swiper:

Finally, we need to initialize Swiper in JS:

```
const swiper = new Swiper('.swiper', {
  // Optional parameters
  direction: 'vertical',
  loop: true,

  // If we need pagination
  pagination: {
    el: '.swiper-pagination',
  },

  // Navigation arrows
  navigation: {
    nextEl: '.swiper-button-next',
    prevEl: '.swiper-button-prev',
  },

  // And if we need scrollbar
  scrollbar: {
    el: '.swiper-scrollbar',
  },
});
```

Este código lo pegamos en el componente `slideshow.component.ts`:

```
19   ngAfterContentInit(): void {
20
21     const mySwiper = new Swiper('.swiper-container', {
22       // Optional parameters
23       direction: 'vertical',
24       loop: true,
25
26       // If we need pagination
27       pagination: {
28         el: '.swiper-pagination',
29       },
30
31       // Navigation arrows
32       navigation: {
33         nextEl: '.swiper-button-next',
34         prevEl: '.swiper-button-prev',
35       },
36
37       // And if we need scrollbar
38       scrollbar: {
39         el: '.swiper-scrollbar',
40       },
41     });
42   }
```

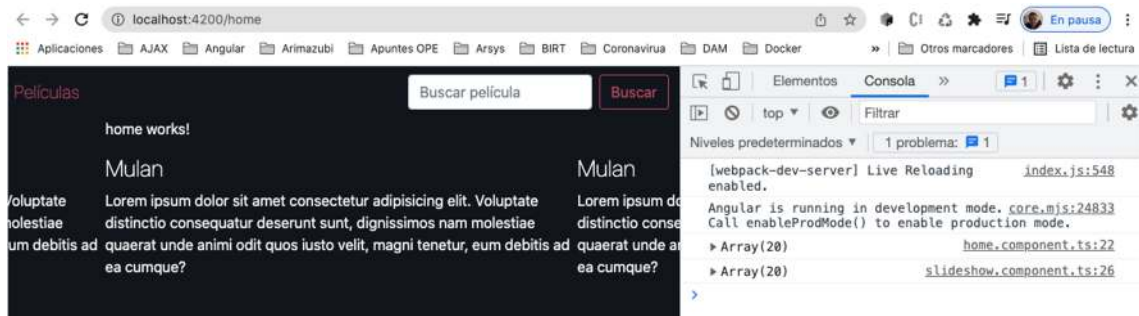
Como tiene que ejecutarse después de cargar la página lo implementamos en un procedimiento `AfterViewInit`. Nos da un error en el swiper porque hay que importarlo:

```
src > app > components > slideshow > TS slideshow.component.ts > ...
1  import { AfterViewInit, Component, Input, OnInit } from '@angular/core';
2  import Swiper from 'swiper';
3  import { Movie } from '../../interface/cartelera-response';
4
5  @Component({
6    selector: 'app-slideshow',
7    templateUrl: './slideshow.component.html',
8    styleUrls: ['./slideshow.component.scss']
9  })
10 export class SlideshowComponent implements OnInit, AfterViewInit {
11
12   @Input() movies: Movie[] = [];
13
14   constructor() { }
15
16   ngAfterViewInit(): void {
17
18     const mySwiper = new Swiper('.swiper-container', {
19
20       loop: true,
```

Limpiamos el código y simplificamos:

```
rc > app > components > slideshow > TS slideshow.component.ts > ...
1  import { AfterViewInit, Component, Input, OnInit } from '@angular/core';
2  import Swiper from 'swiper';
3  import { Movie } from '../../interface/cartelera-response';
4
5  @Component({
6    selector: 'app-slideshow',
7    templateUrl: './slideshow.component.html',
8    styleUrls: ['./slideshow.component.scss']
9  })
10 export class SlideshowComponent implements OnInit, AfterViewInit {
11
12   @Input() movies: Movie[] = [];
13
14   constructor() { }
15
16   ngAfterViewInit(): void {
17
18     const mySwiper = new Swiper('.swiper-container', {
19
20       loop: true,
21
22     })
23   }
24
25   ngOnInit(): void {
26     console.log(this.movies);
27   }
28 }
```

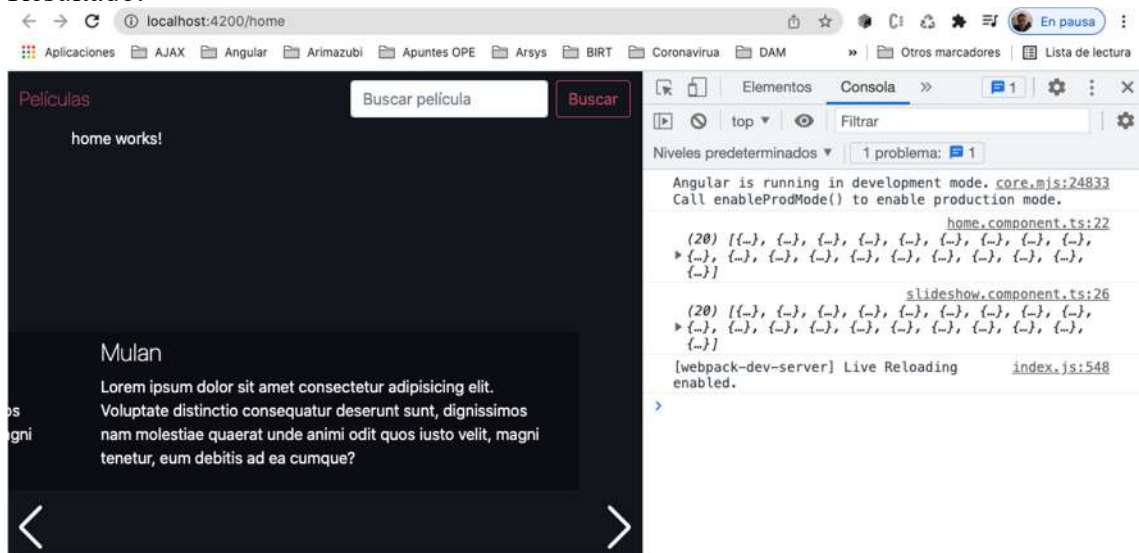
Guardamos la aplicación:



Hay que definir los siguientes estilos en `slideShow.component.css`:

```
src > app > components > slideShow > slideShow.component.scss >
1  .swiper-container {
2    height: 330px;
3    width: 100%;
4  }
5
6  .swiper-button-prev, .swiper-button-next {
7    color: white;
8  }
9
10 .movie-description {
11   background-color: rgba(0,0,0,0.3);
12   bottom: 0;
13   padding: 5px 30px;
14   position: absolute;
15   width: 100%;
16 }
```

Resultado:



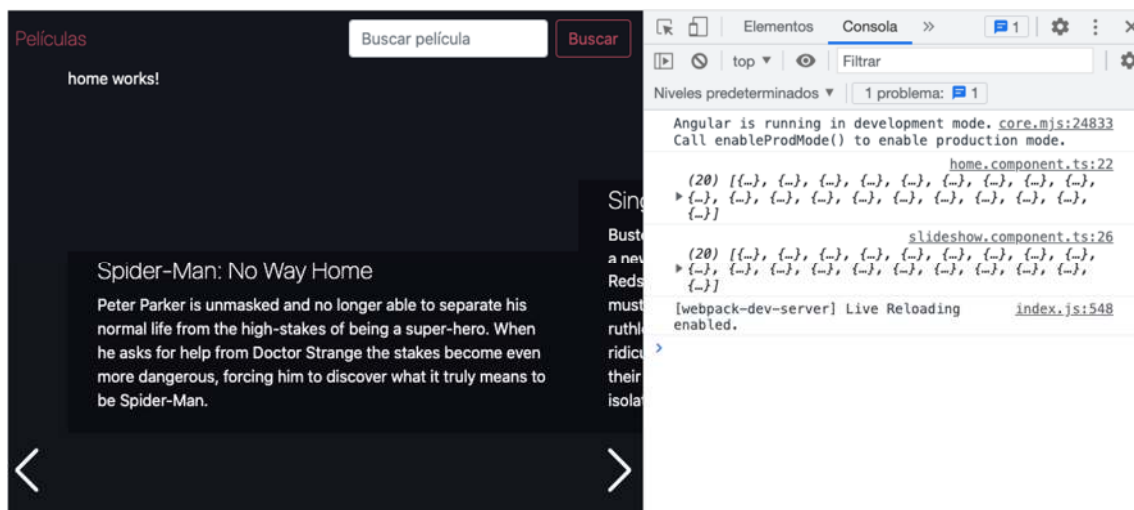
Modifico `slideshow.component.html` para multiplicar el slide por el número de elementos que tenemos en `movie`:

```

1  <!-- Additional required wrapper -->
2  <div class="swiper-wrapper">
3    <!-- Slides -->
4    <div *ngFor="let movie of movies"
5      class="swiper-slide">
6      <div class="movie-description">
7        <h3>{{ movie.title }}</h3>
8        <p>{{ movie.overview }}</p>
9      </div>
10    </div>
11  </div>

```

Vista previa:



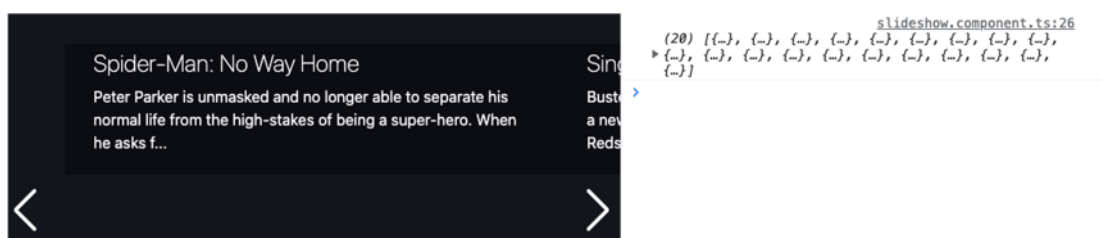
Como el texto es muy largo en algunas películas a través de un pipe lo recortamos.

```

6  <div *ngFor="let movie of movies"
7    class="swiper-slide">
8    <div class="movie-description">
9      <h3>{{ movie.title }}</h3>
10     <p>{{ movie.overview | slice:0:130 }}...</p>
11   </div>
12 </div>
13

```

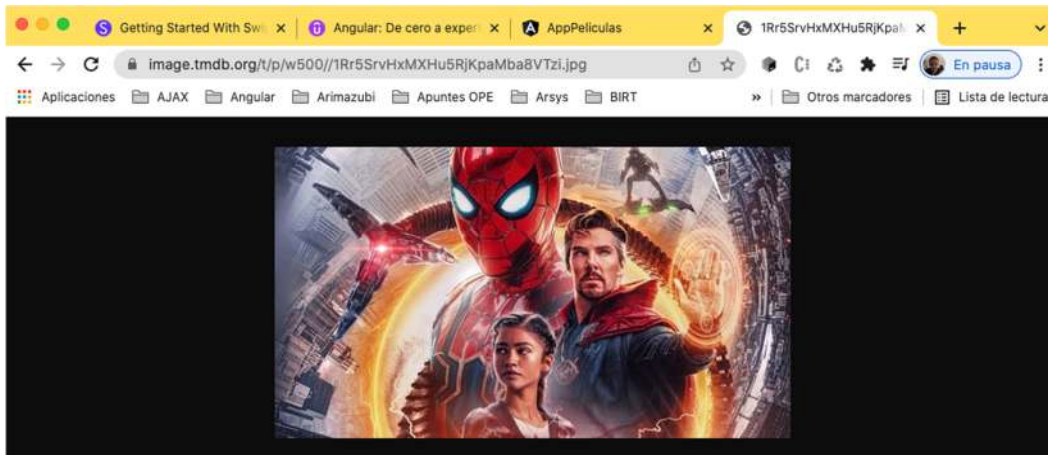
Resultado:





Ahora vamos a trabajar con la imagen de la película. Lo que vamos a visualizar es la imagen de background. Si ponemos en el navegador la siguiente url:

<https://image.tmdb.org/t/p/w500//1Rr5SrvHxMXHu5RjKpaMba8VTzi.jpg>



Modificamos nuestro `slideshow.component.html`.

```

6      <div *ngFor="let movie of movies" class="swiper-slide"
7      [ngStyle]="{
8        'background-size': 'cover',
9        'background-image': 'url()'
10     }">
11
12      <div class="movie-description">
13        <h3>{{ movie.title }}</h3>
14        <p>{{ movie.overview | slice:0:130 }}...</p>
15      </div>
16
17    </div>

```

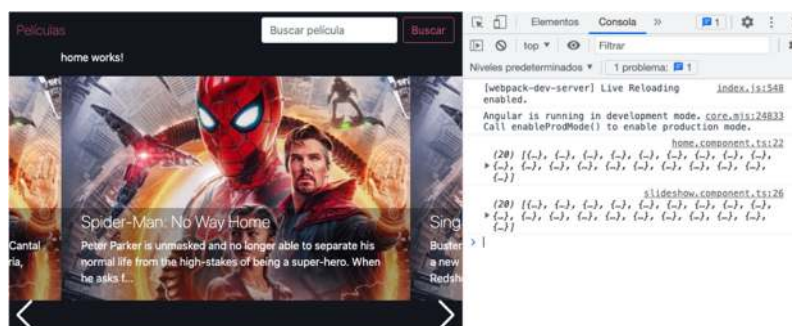
Dentro de url colocamos la que pusimos en la url del navegador:

```

6      <div *ngFor="let movie of movies" class="swiper-slide"
7      [ngStyle]="{
8        'background-size': 'cover',
9        'background-image': 'url(https://image.tmdb.org/t/p/w500//1Rr5SrvHxMXHu5RjKpaMb
10     )'>

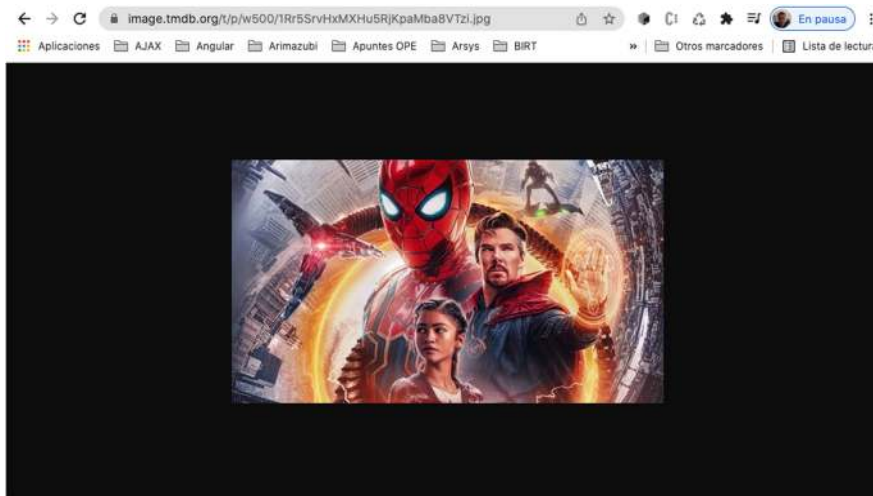
```

Resultado. Ahora se ven bien, aunque sólo sea la misma imagen





Si ponemos en el navegador esta imagen:



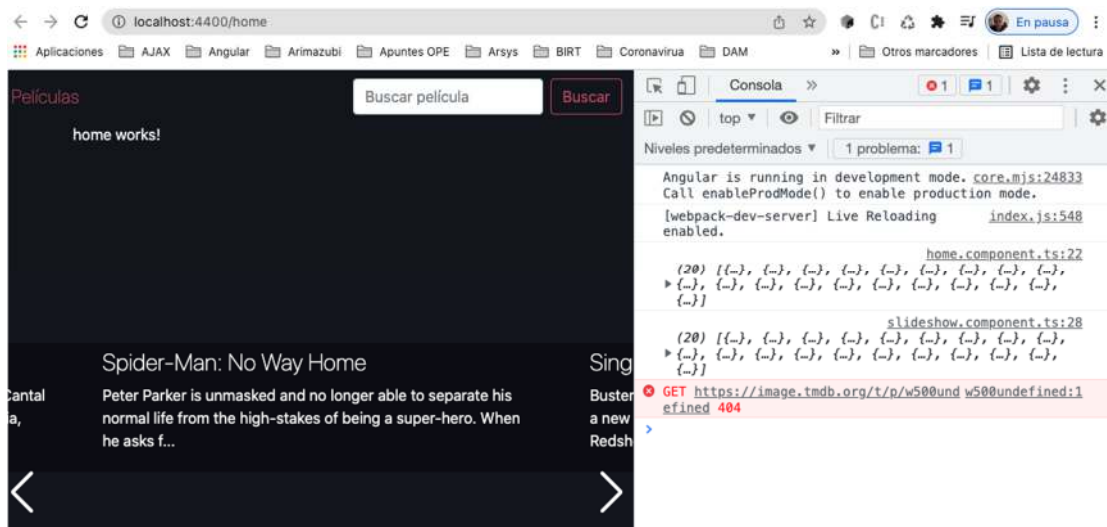
Si queremos que el slide tome todas las imágenes:

```

1  <!-- Slider main container -->
2  <div class="swiper-container">
3    <!-- Additional required wrapper -->
4    <div class="swiper-wrapper">
5      <!-- Slides -->
6      <div *ngFor="let movie of movies" class="swiper-slide"
7        [ngStyle]="{
8          'background-size': 'cover',
9          'background-image': 'url(' + movie.backdrop_path + ')'
10       }">
11
12      <div class="movie-description">
13        <h3>{{ movie.title }}</h3>
14        <p>
15          {{ movie.overview | slice:0:130 }}...
16        </p>
17      </div>
18    </div>
19  </div>

```

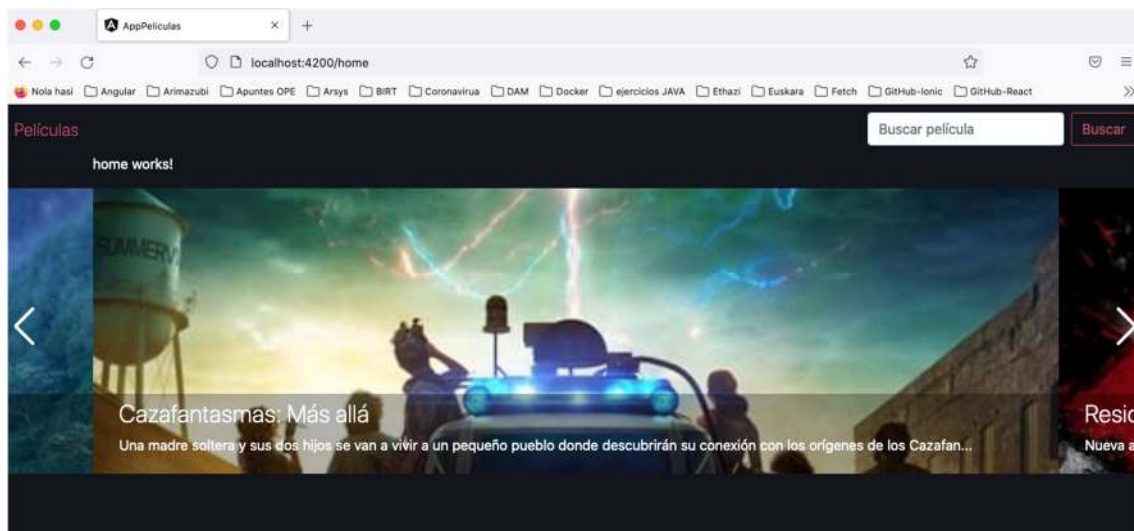
No sale este error. No visualiza la imagen:



El problema es que en el swiper no localiza quién es `backdrop_path` porque en `cartelera-response.ts`. Hemos comprobado que tenemos mal el valor de la propiedad `backdrop_path` del objeto `movie` ( se llamaba `backdropPath`). Vamos al archivo `cartelera-response.ts`:

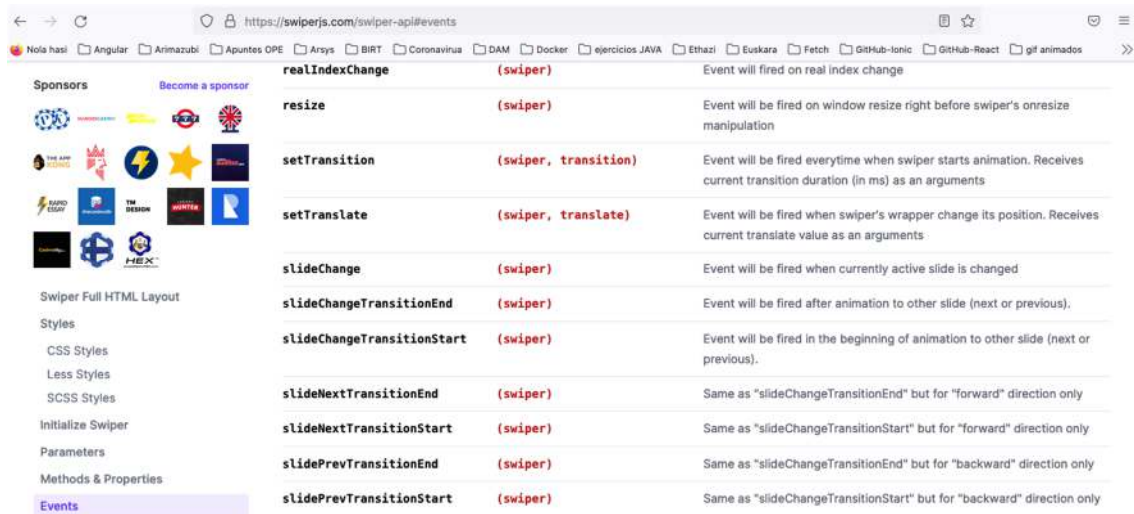
```
--
23 export interface Movie {
24   adult: boolean;
25   backdrop_path: null | string;
26   genreIDs: number[];
27   id: number;
28   originalLanguage: OriginalLanguage;
29   originalTitle: string;
30   overview: string;
31   popularity: number;
32   posterPath: string;
33   releaseDate: Date;
34   title: string;
35   video: boolean;
36   voteAverage: number;
37   voteCount: number;
38 }
--
```

Así, ahora:



## Paso 12. Controles del swiper.

En la documentación de swipe:



Event	Parameters	Description
realIndexChange	(swiper)	Event will be fired on real index change
resize	(swiper)	Event will be fired on window resize right before swiper's onresize manipulation
setTransition	(swiper, transition)	Event will be fired everytime when swiper starts animation. Receives current transition duration (in ms) as an arguments
setTranslate	(swiper, translate)	Event will be fired when swiper's wrapper change its position. Receives current translate value as an arguments
slideChange	(swiper)	Event will be fired when currently active slide is changed
slideChangeTransitionEnd	(swiper)	Event will be fired after animation to other slide (next or previous).
slideChangeTransitionStart	(swiper)	Event will be fired in the beginning of animation to other slide (next or previous).
slideNextTransitionEnd	(swiper)	Same as "slideChangeTransitionEnd" but for "forward" direction only
slideNextTransitionStart	(swiper)	Same as "slideChangeTransitionStart" but for "forward" direction only
slidePrevTransitionEnd	(swiper)	Same as "slideChangeTransitionEnd" but for "backward" direction only
slidePrevTransitionStart	(swiper)	Same as "slideChangeTransitionStart" but for "backward" direction only

Utilizaremos los métodos `swiper.slideNext`, `swiper.slidePrevious`.

Así en `slideshow.component.html`, en los controles de botón-prev, botón-next debemos incluir el siguiente código:

```

24      <!-- If we need navigation buttons -->
25      <div (click)="onSlidePrevious()" class="swiper-button-prev"></div>
26      <div (click)="onSlideNext()" class="swiper-button-next"></div>
27    </div>

```

Marca error porque debemos implementar estos métodos en el componente `slideshow.component.ts`:

```

26      ngOnInit(): void {
27      |
28      |   console.log(this.movies);
29      |   }
30
31      onSlideNext(){
32      |
33      |   }
34
35      onSlidePrevious(){
36      |
37      |   }

```

Como es el objeto `mySwiper` declarado dentro del método `onAfterViewInit()`, declaramos un objeto `public mySwiper` dentro de la clase y fuera de `onAfterViewInit()`.

Así en `ngAfterViewInit`, en el momento que entre se va inicializar:

```

16   constructor() { }
17
18   ngAfterViewInit(): void {
19
20     this.mySwiper = new Swiper('.swiper-container', {
21       loop: true,
22     });
23   }
24
25

```

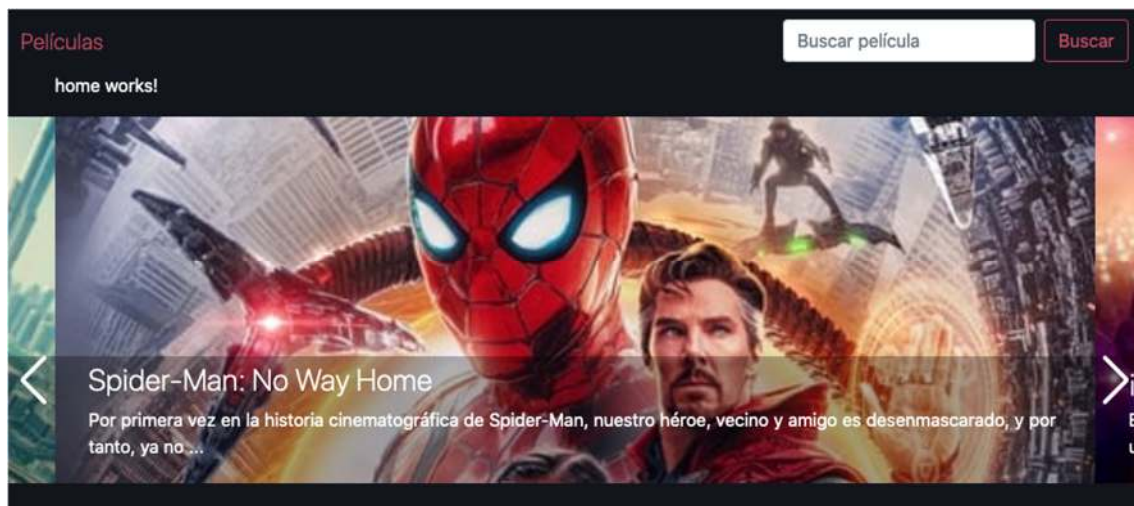
El código de los controles:

```

32   onSlideNext(){
33     this.mySwiper.slideNext();
34   }
35
36   onSlidePrevious(){
37     this.mySwiper.slidePrev();
38   }
39

```

El resultado, donde los controles adelante, atrás funcionan a la perfección.



El componente completo:

```
src > app > components > slideshow > TS slideshow.component.ts > ...
1  import { AfterViewInit, Component, Input, OnInit } from '@angular/core';
2  import { Movie } from '../../interface/cartelera-response';
3  import Swiper from 'swiper';
4
5  @Component({
6    selector: 'app-slideshow',
7    templateUrl: './slideshow.component.html',
8    styleUrls: ['./slideshow.component.scss']
9  })
10 export class SlideshowComponent implements OnInit, AfterViewInit {
11
12   @Input()
13   movies: Movie[] = [];
14
15   public mySwiper!: Swiper;
16
17   constructor() { }
18
19   ngAfterViewInit(): void {
20
21     this.mySwiper = new Swiper('.swiper-container', {
22       |   loop: true,
23     });
24   }
25
26   ngOnInit(): void {
27
28     console.log(this.movies);
29   }
30
31   onSlideNext(){
32     |   this.mySwiper.slideNext();
33   }
34
35   onSlidePrevious(){
36     |   this.mySwiper.slidePrev();
37   }
38
39 }
40 }
```