

ANGULAR 12.

GESTIÓN DE RUTAS Y LAZY LOAD



INTRODUCCIÓN

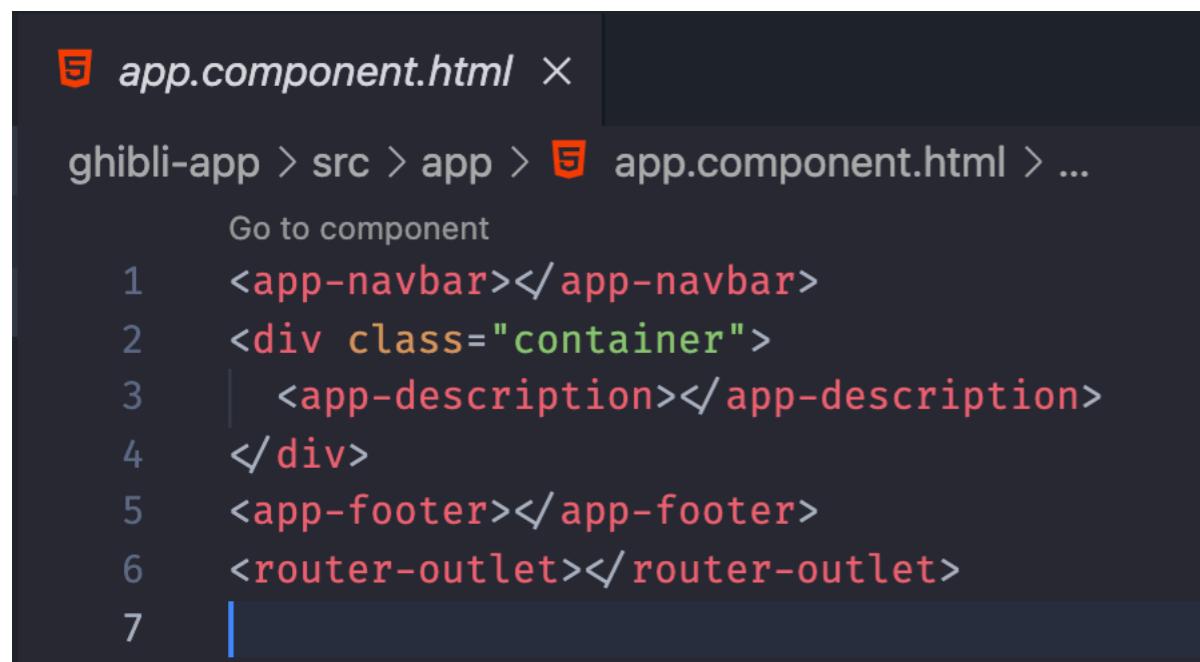
Vamos a ver el sistema de rutas de Angular y las opciones que tenemos para que el navbar y el footer se mantengan fijos y cambie el contenido interno de nuestra aplicación en función de la ruta que se establezca.

Vamos a crear un sistema de rutas y cargaremos los módulos en función de la ruta utilizando **lazy load**, es decir, sólo cargaremos los módulos cuando se navegue hacia ellos.

En el momento que se cargue un módulo, el fichero encargado de las rutas de ese módulo será el que se encargue de su gestión.

En nuestra aplicación, actualmente tenemos 3 componentes, **navbar**, **footer** y **description**. Teníamos un módulo **core** con nuestros componentes y módulos **ghibli** y **shared** inicialmente vacíos.

Lo que queremos es que **app.component** cargue el navbar y el footer de forma fija, pero que el sistema de rutas cargue el contenido en función de la ruta, la descripción u otras cosas, como un listado de películas, personajes...



The screenshot shows a code editor with the file `app.component.html` open. The file contains the following HTML code:

```
1 <app-navbar></app-navbar>
2 <div class="container">
3   <app-description></app-description>
4 </div>
5 <app-footer></app-footer>
6 <router-outlet></router-outlet>
7 |
```

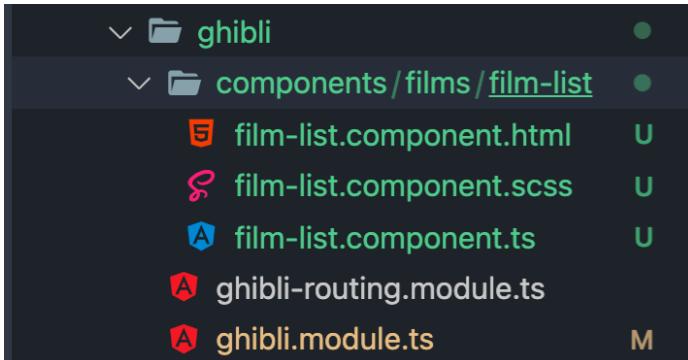
The code editor interface includes a breadcrumb navigation bar at the top showing the file path: `ghibli-app > src > app > app.component.html > ...`. There is also a "Go to component" link and a status bar at the bottom.

CREACIÓN DE FILMS

Vamos a crear el componente que permite listar las películas y le añadiremos la funcionalidad mas adelante obteniendo los datos del API Rest.

Vamos a crear el componente en el módulo de **ghibli** porque en un futuro solicitará ese recurso al API Rest.

```
inma@MacBook-Pro-de-Inmaculada ghibli-app % ng g c ghibli/components/films/film-list --skip-tests
CREATE src/app/ghibli/components/films/film-list/film-list.component.scss (0 bytes)
CREATE src/app/ghibli/components/films/film-list/film-list.component.html (24 bytes)
CREATE src/app/ghibli/components/films/film-list/film-list.component.ts (287 bytes)
UPDATE src/app/ghibli/ghibli.module.ts (391 bytes)
inma@MacBook-Pro-de-Inmaculada ghibli-app %
```

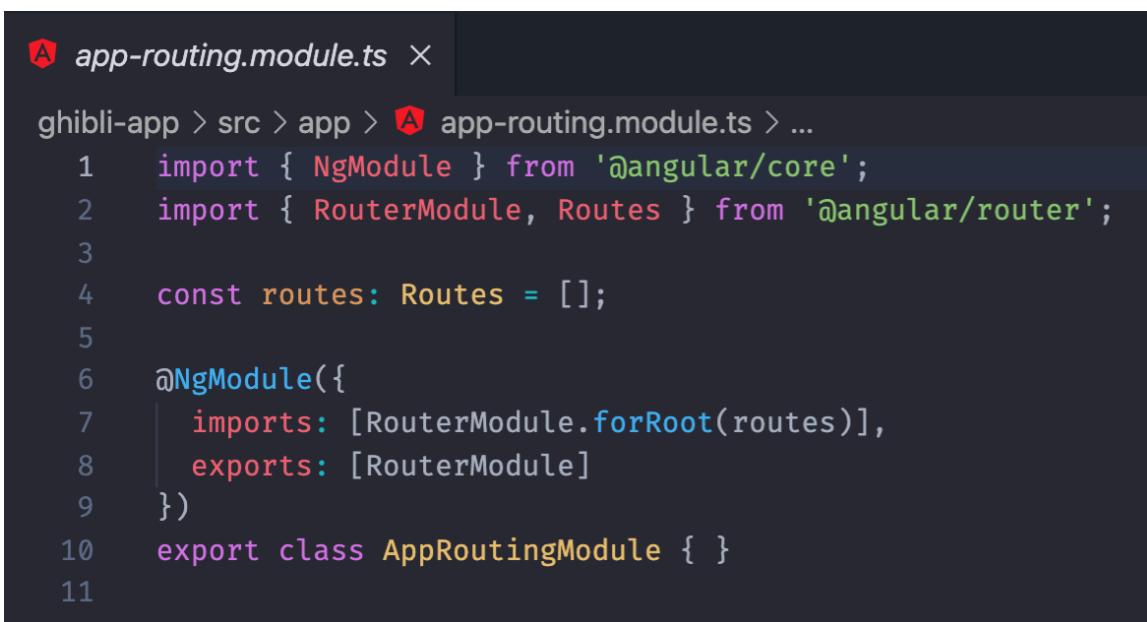


SISTEMA DE RUTAS

Las SPA liberan al servidor de una parte del trabajo al reducir llamadas. Son más óptimas. Antes se desplegaban múltiples páginas en distintas direcciones, el servidor procesaba la ruta y remitía el contenido al navegador. Esto ha cambiado en los últimos años y ahora delega más responsabilidad en los navegadores. Para aplicaciones empresariales, se distribuye actualmente esa carga y es el navegador el que prepara la vista, ejecutando instrucciones y solicitando datos.

Una de las responsabilidades de las que el navegador se hace cargo es determinar qué vista debe mostrar en cada dirección. Vamos a ver cómo lo resuelve Angular.

Recordemos que teníamos un fichero que se creaba cuando generábamos el proyecto que es **app-routing.module.ts**.



```
A app-routing.module.ts ×
ghibli-app > src > app > A app-routing.module.ts > ...
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3
4 const routes: Routes = [];
5
6 @NgModule({
7   imports: [RouterModule.forRoot(routes)],
8   exports: [RouterModule]
9 })
10 export class AppRoutingModule { }
```

El array de rutas está vacío aún. El **app-routing** va a importar, configurar y exportar el módulo **RouterModule** y a su vez el **app.module** va a importar este **AppRoutingModule**.

Vamos a asignar que cuando se establezca una ruta se cargue un componente concreto y sólo se cargue cuando se establezca esa ruta.

Vamos a crear una ruta vacía, la ruta raíz y de este modo a esa ruta estará asociado el componente descripción y que dependa de esa ruta. Después modificaremos el navbar para que acceda a esas rutas.

El primer fichero que se carga de rutas es **app-routing.module**, ahí crearemos la ruta raíz.

app.component.html M X

ghibli-app > src > app > app.component.html > 6

Go to component

```
1  <app-navbar></app-navbar>
2  <div class="container">
3  |  <router-outlet></router-outlet>
4  </div>
5  <app-footer></app-footer>
6
7
```

app-routing.module.ts M X

ghibli-app > src > app > app-routing.module.ts > ...

```
1  import { NgModule } from '@angular/core';
2  import { RouterModule, Routes } from '@angular/router';
3  import { DescriptionComponent } from './core/components/description/description.component';
4
5  const routes: Routes = [
6  {
7    path: '',
8    component: DescriptionComponent,
9  },
10 ];
11
12 @NgModule({
13   imports: [RouterModule.forRoot(routes)],
14   exports: [RouterModule],
15 })
16 export class AppRoutingModule {}
```

DescriptionComponent lo exportábamos, ahora no lo vamos a exportar, porque lo que queremos es que ese componente se cargue únicamente cuando se navegue hacia esa ruta. **Lazy Load**.

core.module.ts M X

```
ghibli-app > src > app > core > A core.module.ts > CoreModule
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3
4 import { CoreRoutingModule } from './core-routing.module';
5 import { NavbarComponent } from './components/navbar/navbar.component';
6 import { FooterComponent } from './components/footer/footer.component';
7 import { DescriptionComponent } from './components/description/description.component';
8
9
10 @NgModule({
11   declarations: [
12     NavbarComponent,
13     FooterComponent,
14     DescriptionComponent
15   ],
16   imports: [
17     CommonModule,
18     CoreRoutingModule
19   ],
20   exports: [
21     NavbarComponent,
22     FooterComponent
23   ]
24 })
25 export class CoreModule { }
```

app-routing.module.ts M ×

```
ghibli-app > src > app > A app-routing.module.ts > ...
1 √ import { NgModule } from '@angular/core';
2   import { RouterModule, Routes } from '@angular/router';
3   import { DescriptionComponent } from './core/components/description/description.component';
4
5 √ const routes: Routes = [
6   {
7     path: '',
8     loadChildren:() => import('./core/core.module').then((m) => m.CoreModule),
9   },
10 ];
11
12 √ @NgModule({
13   imports: [RouterModule.forRoot(routes)],
14   exports: [RouterModule],
15 })
16 export class AppRoutingModule {}
```

core-routing.module.ts M ×

```
ghibli-app > src > app > core > A core-routing.module.ts > [o] routes > ↴ component
● 1   import { NgModule } from '@angular/core';
2   import { RouterModule, Routes } from '@angular/router';
3   import { DescriptionComponent } from './components/description/description.component';
4
5   const routes: Routes = [
6     {
7       path: '',
8       component: DescriptionComponent
9     }
10 ];
11
12 @NgModule({
13   imports: [RouterModule.forChild(routes)],
14   exports: [RouterModule]
15 })
16 export class CoreRoutingModule { }
```

De este modo se carga ante esa ruta el **core.module** y cuando se carga ese módulo se carga también su sistema de rutas y se carga la ruta asociada, que en este caso es **DescriptionComponent**.

De este modo se carga ante esa ruta el **core.module** y cuando se carga ese módulo se carga también su sistema de rutas y se carga la ruta asociada, que en este caso es **DescriptionComponent**.

Ahora vamos a modificar el **navbar** añadiendo las rutas mediante las propiedades **routerLink** de Angular. También utilizaremos la clase **active** y una opción del sistema de enrutado para determinar si debe estar la opción activa o no. Le indicamos que la ruta tiene que ser exacta porque no queremos que se active cuando sea la concatenación de varias cosas por ejemplo /film... o cualquier otra cosa. Por eso ponemos exact.

```
(navbar.component.html) M ×  
app > src > app > core > components > navbar > navbar.component.html > nav.navbar.navbar-expand-lg.navbar-dark.bg-dark > div.container-fluid > div#navbarGhibli  
Go to component  
1 <nav class="navbar navbar-expand-lg navbar-dark bg-dark">  
2   <div class="container-fluid">  
3     <a class="navbar-brand" routerLink="/"></a>  
4     <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarGhibli" aria-controls="navbarGhibli"  
5       <span class="navbar-toggler-icon"></span>  
6     </button>  
7     <div class="collapse navbar-collapse" id="navbarGhibli">  
8       <ul class="navbar-nav me-auto mb-2 mb-lg-0">  
9         <li class="nav-item">  
10          <a class="nav-link active" routerLinkActive="active" [routerLinkActiveOptions]={exact: true}" routerLink="/Home"></a>  
11        </li>  
12      </ul>  
13      <ul class="navbar-nav ml-auto">  
14        <li class="nav-item">  
15          <a class="nav-link" href="https://github.com/igijon/ghibli_angular_app_2021" target="_blank" rel="noopener">  
16            <svg xmlns="http://www.w3.org/2000/svg" width="25" height="25" class="navbar-nav-svg d-inline-block align-text-top" viewBox="0 0 25 25">  
17              </svg>  
18            </a>  
19        </li>  
20        <li class="nav-item">  
21          <a class="nav-link" href="https://twitter.com/InmaculadaGijn1" target="_blank" rel="noopener">  
22            <svg xmlns="http://www.w3.org/2000/svg" width="25" height="25" class="navbar-nav-svg d-inline-block align-text-top" viewBox="0 0 25 25">  
23              </svg>  
24        </li>  
25      </ul>  
26    </div>  
27  </nav>
```

Ahora vamos a añadir una ruta. En **app-routing** vamos a crear una nueva ruta:

Estamos indicando que cuando se navegue a esa ruta, se cargue el módulo ghibli.

app-routing.module.ts M X

```
ghibli-app > src > app > A app-routing.module.ts > ...
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3 |
4 const routes: Routes = [
5   {
6     path: '',
7     loadChildren:() => import('./core/core.module').then((m) => m.CoreModule),
8   },
9   {
10     path: 'ghibli',
11     loadChildren:() => import('./ghibli/ghibli.module').then((m) => m.GhibliModule),
12   }
13 ];
14
15 @NgModule({
16   imports: [RouterModule.forRoot(routes)],
17   exports: [RouterModule],
18 })
19 export class AppRoutingModule {}
```

ghibli-routing.module.ts M X

```
ghibli-app > src > app > ghibli > A ghibli-routing.module.ts > [o] routes > ⚡ path
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3 import { FilmListComponent } from './components/films/film-list/film-list.component';
4
5 const routes: Routes = [
6   {
7     path: 'films',
8     component: FilmListComponent,
9   },
10];
11
12 @NgModule({
13   imports: [RouterModule.forChild(routes)],
14   exports: [RouterModule],
15 })
16 export class GhibliRoutingModule {}
```

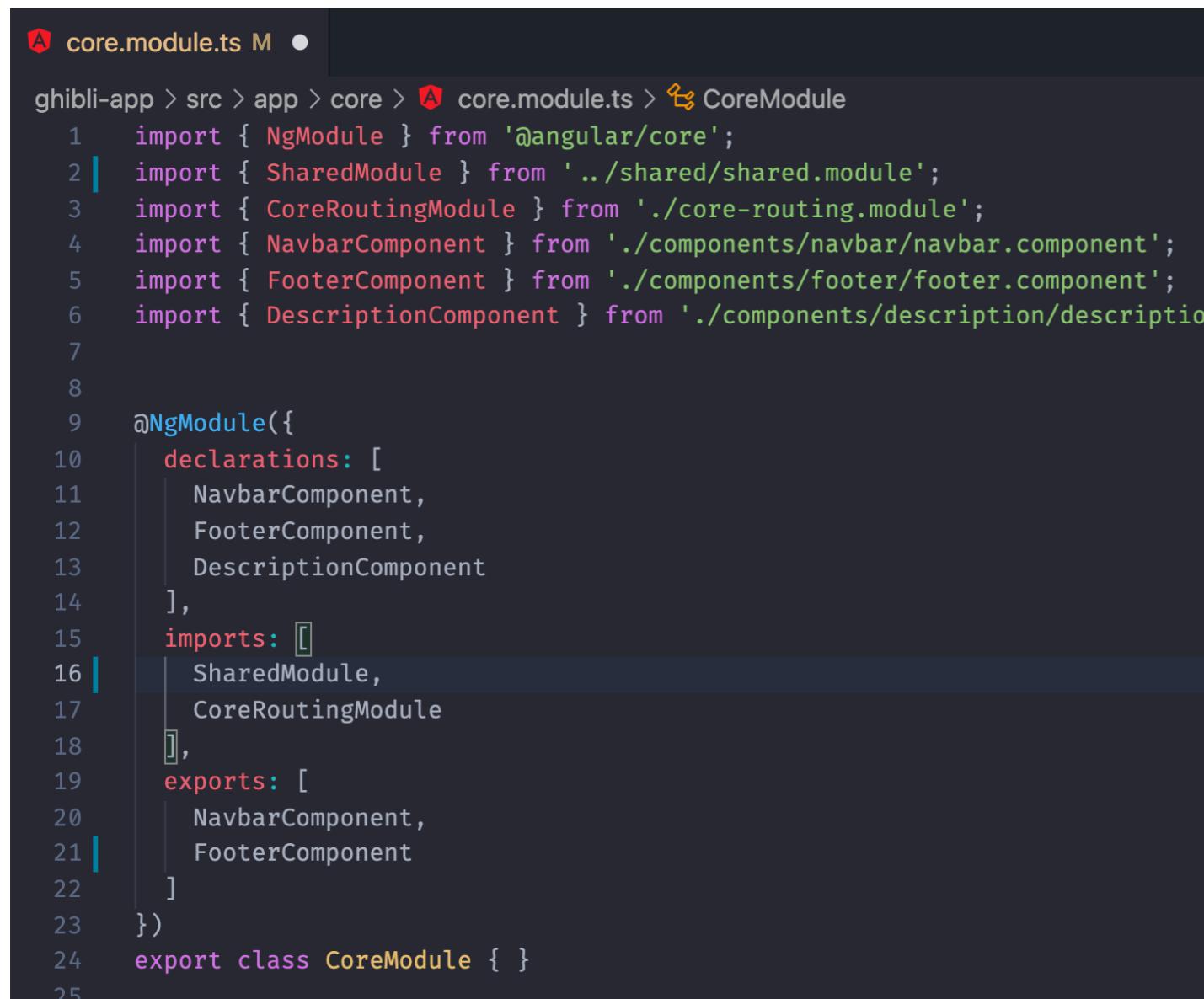
En el fichero **app-routing** vamos a añadir una última ruta para que en el caso de que la ruta introducida sea errónea redireccione a la ruta principal.

```
A app-routing.module.ts M ×
ghibli-app > src > app > A app-routing.module.ts > [o] routes
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3
4 const routes: Routes = [
5   {
6     path: '',
7     loadChildren: () => import('../core/core.module').then((m) => m
8   },
9   {
10    module: "/Users/inma/Curso 2021-2022/CIFP Virgen De
11    Gracia/DWEC/ghibli_angular_app_2021/ghibli-
12    path: 'gl' app/src/app/ghibli/ghibli.module"
13    loadChildren: () => import('../ghibli/ghibli.module').then((m) =>
14    },
15    {
16      path: '**',
17      redirectTo: ''
18    }
19 ];
20
21 @NgModule({
22   imports: [RouterModule.forRoot(routes)],
23   exports: [RouterModule],
24 })
25 export class AppRoutingModule {}
```

MÓDULO SHARED

Vamos a añadir algunas cosas al módulo shared.

En todos los módulos hacemos un import común que es el **CommonModule**. Todos los import comunes podemos sacarlos de los módulos y añadirlos al **SharedModule** y éste añadirlo al módulo **app.module**. De momento vamos a comenzar con el **CommonModule**, sacándolo de todos los módulos: **core**, **ghibli** y añadiéndolo en el módulo **shared**. En los módulos **core** y **ghibli** deberemos importar ahora el módulo shared.



A core.module.ts M ●

```
ghibli-app > src > app > core > A core.module.ts > CoreModule
1 import { NgModule } from '@angular/core';
2 import { SharedModule } from '../shared/shared.module';
3 import { CoreRoutingModule } from './core-routing.module';
4 import { NavbarComponent } from './components/navbar/navbar.component';
5 import { FooterComponent } from './components/footer/footer.component';
6 import { DescriptionComponent } from './components/description/description.component';

7

8
9 @NgModule({
10   declarations: [
11     NavbarComponent,
12     FooterComponent,
13     DescriptionComponent
14   ],
15   imports: [
16     SharedModule,
17     CoreRoutingModule
18   ],
19   exports: [
20     NavbarComponent,
21     FooterComponent
22   ]
23 })
24 export class CoreModule { }
```

ghibli.module.ts X

```
ghibli-app > src > app > ghibli > A ghibli.module.ts > GhibliModule
1 import { NgModule } from '@angular/core';
2 import { GhibliRoutingModule } from './ghibli-routing.module';
3 import { FilmListComponent } from './components/films/film-list/film-list.component';
4 import { SharedModule } from '../shared/shared.module';

5
6
7 @NgModule({
8   declarations: [
9     FilmListComponent
10    ],
11   imports: [
12     SharedModule,
13     GhibliRoutingModule
14   ]
15 })
16 export class GhibliModule { }
17
```

shared.module.ts X

```
ghibli-app > src > app > shared > A shared.module.ts > SharedModule
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3
4 import { SharedRoutingModule } from './shared-routing.module';
5
6
7 @NgModule({
8   declarations: [],
9   imports: [
10     CommonModule,
11     SharedRoutingModule
12   ]
13 })
14 export class SharedModule { }
15
```

AÑADIR ITEM NAVBAR

Añadimos el item en el navbar para films.

```
(navbar.component.html) M X

and-lg.navbar-dark.bg-dark > div.container-fluid > div#navbarGhibli.collapse.navbar-collapse > ul.navbar-nav.me-auto.mb-2.mb-lg-0
Go to component
1 <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
2   <div class="container-fluid">
3     <a class="navbar-brand" routerLink="/"></a>
4     <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarGhibli">
5       <span class="navbar-toggler-icon"></span>
6     </button>
7     <div class="collapse navbar-collapse" id="navbarGhibli">
8       <ul class="navbar-nav me-auto mb-2 mb-lg-0">
9         <li class="nav-item">
10           <a class="nav-link active" routerLinkActive="active" [routerLinkActiveOptions]={{exact: true}}>
11             </a>
12           <li class="nav-item">
13             <a class="nav-link" routerLinkActive="active" routerLink="ghibli/films">Films</a>
14           </li>
15         </ul>
16         <ul class="navbar-nav ml-auto">
17           <li class="nav-item">
18             <a class="nav-link" href="https://github.com/igijon/ghibli_angular_app_2021" target="_blank" rel="noopener">
19               <svg xmlns="http://www.w3.org/2000/svg" width="25" height="25" class="navbar-nav-svg d-inline">
20                 <a>
21               </li>
```

