

1. Routing: Introducción y configuración básica (pag 174)

- 1.1. Creamos el proyecto
ng new Routing-Child
- 1.2. Instalamos Bootstrap
npm install bootstrap jquery @popperjs/core [--save]
- 1.3. Importamos Bootstrap en el archivo angular.json

```
"styles": [  
  "node_modules/bootstrap/dist/css/bootstrap.min.css",  
  "src/styles.scss"  
],  
"scripts": [  
  "node_modules/jquery/dist/jquery.min.js",  
  "node_modules/@popperjs/core/dist/umd/popper.min.js",  
  "node_modules/bootstrap/dist/js/bootstrap.min.js"  
]
```
- 1.4. Creamos los componentes:
ng g c libro-lista
ng g c autor-lista
ng g c not-found-error404
- 1.5. En este punto, desarrollaremos el componente libroLista. Para ello, previamente crearemos el modelo de datos Libro (libro.model.ts) y un chero mock data (mocks.ts) con un conjunto de libros de prueba.

Creamos la carpeta models

Dentro de ella, creamos el la interface libro-model.ts

```
export interface LibroModel {  
  id: number;  
  titulo: string;  
  autor: string;  
}
```

Creamos la carpeta mocks

Dentro de ella, creamos el archivo libro-mock.ts

```
import { LibroModel } from "../interfaces/libro-model";  
  
export const LIBROS: LibroModel[] = [  
  {  
    "id": 1,  
    "titulo": "El Quijote",
```

```

        "autor": "Cervantes"
    },
    {
        "id": 2,
        "titulo": "Hamlet",
        "autor": "Shakespeare"
    }
];

```

- 1.6. Vamos a libro-lista.component.ts

1.6.1. Creamos una propiedad llamada libros que es un Array del libroModel

1.6.2. En el ngOnInit() llamamos a this.libros = LIBROS; para que al iniciarse la app, siempre se despliegue la lista de libros que hemos definido en el libro-mock.ts

```

import { Component, OnInit } from '@angular/core';
import { LibroModel } from '../interfaces/libro-model';
import { LIBROS } from '../mocks/libro-mock';

@Component({
  selector: 'app-libro-lista',
  templateUrl: './libro-lista.component.html',
  styleUrls: ['./libro-lista.component.css']
})
export class LibroListaComponent implements OnInit {

  libros: LibroModel[] = [];

  constructor() { }

  ngOnInit(): void {
    this.libros = LIBROS;
  }

}

```

1.6.3. Ahora en el libro-lista.component.html vamos a crear un para desplegar con la directiva *ngFor todos los libros:

```

<p>libro-lista works!</p>

<h4>Libros:</h4>

<div class="container">

```

```

<ul class="list-group-horizontal">
  <li class="list-group-item active">
    LIBROS
  </li>
  <li
    class="list-group-item"
    *ngFor="let libro of libros"
  >
    {{ libro.titulo }}
  </li>
</ul>
<div>

```

- 1.7. Una vez creados los componentes, pasaremos a realizar la configuración del servicio Router, y para ello vamos al app-routing.module.ts para crear nuestras primeras rutas a nuestros componentes:

```

import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { AutorListaComponent } from
'./autor-lista/autor-lista.component';
import { LibroListaComponent } from
'./libro-lista/libro-lista.component';
import { NotFoundError404Component } from
'./not-found-error404/not-found-error404.component';

const routes: Routes = [
  {
    path: 'libros',
    component: LibroListaComponent
  },
  {
    path: 'autores',
    component: AutorListaComponent
  },
  {
    path: '',
    redirectTo: '/libros',
    pathMatch: 'full'
  },
  {
    path: '**',
    component: NotFoundError404Component
  }
]

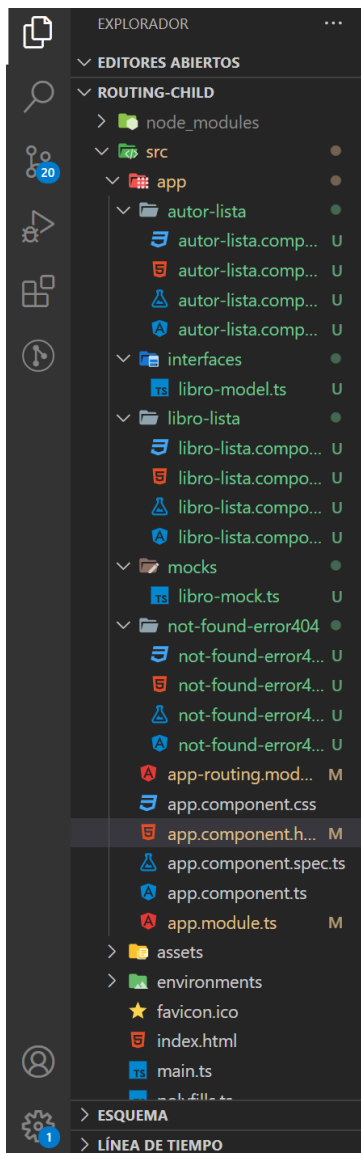
```

```
}  
];
```

- Y ahora en el app.component.html, borramos todo lo que venía de entrada, y ponemos un título y el <router-outlet>

```
<div style="text-align:center">  
  <h3>Servicio Router: ejemplos de uso</h3>  
</div>  
  
<router-outlet></router-outlet>
```

El explorador de archivos del proyecto queda de momento de esta forma:



2. Routing: RouterLinks

- 2.1. Volvemos al app.component.html para crear un navbar:

```
<div style="text-align:center">
  <h3>Servicio Router: ejemplos de uso</h3>
</div>

<ul class="nav">
  <li class="nav-item">
    <a class="nav-link active" routerLink="/libros">Libros</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" routerLink="/autores">Autores</a>
  </li>
</ul>

<router-outlet></router-outlet>
```

- 2.1.1. Vamos al app.component.css y creamos la clase de:

```
.active-link {
  color: orange;
}
```

- 2.1.2. Y volviendo a app.component.html insertamos RouterLinkActive

```
<div style="text-align:center">
  <h3>Servicio Router: ejemplos de uso</h3>
</div>

<ul class="nav">
  <li class="nav-item">
    <a
      class="nav-link active"
      routerLink="/libros"
      routerLinkActive="active-link"
    >
      Libros
    </a>
  </li>
```

```
<li class="nav-item">
  <a
    class="nav-link"
    routerLink="/autores"
    routerLinkActive="active-link"
  >
    Autores
  </a>
</li>
</ul>

<router-outlet></router-outlet>
```

Servicio Router: ejemplos de uso

[Libros](#) [Autores](#)

libro-lista works!

Libros:

LIBROS
El Quijote
Hamlet

3. Routing: Rutas con parámetros y ActivatedRoute

- 3.1. Creamos el componente de libro-detalles
ng g c libro-detalles
- 3.2. Añadimos una ruta para este componente

```
const routes: Routes = [  
  {  
    path: 'libros',  
    component: LibroListaComponent  
  },  
  {  
    path: 'libros/:id',  
    component: LibroDetallesComponent  
  },  
  {  
    path: 'autores',  
    component: AutorListaComponent  
  },  
  {  
    path: '',  
    redirectTo: '/libros',  
    pathMatch: 'full'  
  },  
  {  
    path: '**',  
    component: NotFoundError404Component  
  }  
];
```

En el componente libro-detalles añadiremos el código para primero obtener el identificador del libro (parámetro “:id” del URL) y luego para leer el detalle de ese libro.

- 3.3. Vamos al libro-detalles.component.ts para:
 - 3.3.1. Crear una propiedad llamada libro que siga el LibroModel
 - 3.3.2. Inyectamos la ActivatedRoute en el constructor
 - 3.3.3. Para obtener los parámetros de la ruta, ActivatedRoute nos proporciona el observable paramMap. Para obtenerlos, simplemente nos suscribimos al observable a la espera de recibirlos. De esta manera recibiremos el identificador de libro.

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

```

import { LibroModel } from '../interfaces/libro-model';
import { LIBROS } from '../mocks/libro-mock';

@Component({
  selector: 'app-libro-detalles',
  templateUrl: './libro-detalles.component.html',
  styleUrls: ['./libro-detalles.component.css']
})
export class LibroDetallesComponent implements OnInit {

  libro: LibroModel | undefined;

  constructor(
    private activatedRoute: ActivatedRoute
  ) { }

  ngOnInit(): void {
    this.activatedRoute.paramMap
      .subscribe((paramMaps: ParamMap) => {
        let id = Number(paramMaps.get('id'));
        this.libro = LIBROS[id];
        this.libro = LIBROS.find((item) => item.id === id);
      })
  }
}

```

- 3.4. En el template de libro-detalles.component.html añadiremos el código para visualizar el detalle del libro

```

<p>libro-detalles works!</p>

<div class="container" *ngIf="libro != undefined">

  <ul class="list-group-horizontal">
    <li class="list-group-item active">Detalles de libro</li>
    <li class="list-group-item">Identificador: {{libro.id}}</li>
    <li class="list-group-item">Titulo: {{libro.titulo}}</li>
    <li class="list-group-item">Autor: {{libro.autor}}</li>
  </ul>

  <router-outlet></router-outlet>

```



```
</div>
```

- 3.5. Y, finalmente, añadiremos un enlace o link al componente libro-detalles para cada uno de los libros dentro del libro-lista.component.html:

```
<h4>Libros:</h4>

<div class="container">
  <ul class="list-group-horizontal">
    <li class="list-group-item active">
      LIBROS
    </li>

    <li
      class="list-group-item"
      *ngFor="let libro of libros"
    >
      <a [routerLink]="['/libros', libro.id]">
        {{ libro.titulo }}
      </a>
    </li>
  </ul>
</div>
```

Servicio Router: ejemplos de uso

[Libros](#) [Autores](#)

libro-detalles works!

Detalles de libro

Identificador: 1

Título: El Quijote

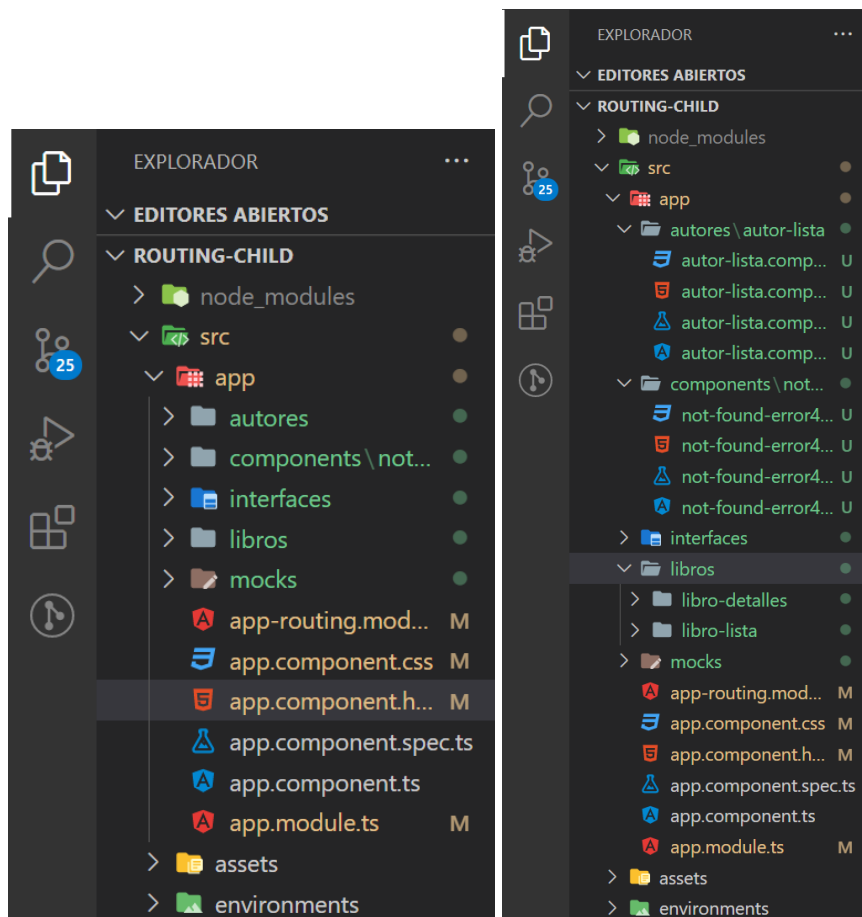
Autor: Cervantes

4. Routing: child routes

Los path de configuraciones hijos son relativos a los de las configuraciones padre. Por ejemplo, `"/actor/:id/biografia"` o `"/actor/:id/filmografia"` en el código anterior. Por otra parte, en el template de todo componente padre siempre habrá que añadir la etiqueta `router-outlet`.

Angular nos proporciona dos maneras de acceder al árbol de rutas activas. La primera consiste en partir de una ruta activa (`ActivatedRoute`) cualquiera, y usar sus propiedades `parent` y `children` para desplazarnos por el árbol. Y la segunda consiste en usar la propiedad `RouterState`. Esta propiedad nos permite obtener el árbol de rutas activas en cualquier momento y lugar de la aplicación.

Nota: Antes de continuar, vamos a refactorizar, para reordenar los componentes de una misma temática en una carpeta correspondiente para cada uno de ellos. Transformamos el directorio de `app` hacia la siguiente estructura:



- 4.1. Vamos a crear un par de componentes de libros:

ng g c libros/libro-opiniones

ng g c libros/libro-imagenes

- 4.2. Seguidamente, en la configuración de rutas, añadiremos las nuevas child routes:

```
{
  path: 'libros/:id',
  component: LibroDetallesComponent,
  children: [
    {
      path: 'imagenes',
      component: LibroImagenesComponent
    },
    {
      path: 'opiniones',
      component: LibroOpinionesComponent
    },
    {
      path: '',
      redirectTo: 'imagenes',
      pathMatch: 'full'
    },
    {
      path: '**',
      component: NotFoundError404Component
    }
  ]
},
```

- 4.3. En el template del componente padre, libro-detalles.component.html, añadiremos dos enlaces a estas nuevas rutas y la etiqueta router-outlet para visualizar sus componentes asociados:

```
<div class="container" *ngIf="libro != undefined">

  <ul class="list-group-horizontal">
    <li class="list-group-item active">Detalles de libro</li>
    <li class="list-group-item">Identificador: {{libro.id}}</li>
    <li class="list-group-item">Titulo: {{libro.titulo}}</li>
    <li class="list-group-item">Autor: {{libro.autor}}</li>
  </ul>
```

```

<ul class="list-group">
  <li class="list-group-item active">
    <h4>Informacion adicional: </h4>
  </li>

  <li class="list-group-item" routerLink='imagenes'>Imagenes</li>
  <li class="list-group-item" routerLink='opiniones'>Opiniones</li>
</ul>

<router-outlet></router-outlet>
</div>

```

y en libro-detalles.component.css añadimos ...

```

.active-link {
  color: orange;
}

```

- 4.4. A continuación, añadiremos el código necesario en libro-imagenes para cargar el identificador de libro que serviría al componente para cargar las imágenes relacionadas.

Para ello, haremos uso de ActivatedRoute primero para acceder al ActivatedRoute parent, y luego para obtener el valor del parámetro.

4.4.1. Creamos la propiedad idLibro de tipo number, y lo ponemos como undefined

4.4.2. Inyectamos el ActivatedRoute en el constructor

4.4.3. Obtenemos el id de cada libro con ActivatedRoute

```

export class LibroImagenesComponent implements OnInit {

  idLibro: number | undefined;

  constructor(
    private activatedRoute: ActivatedRoute
  ) { }

  ngOnInit(): void {
    this.activatedRoute.parent?.paramMap
      .subscribe((paramMaps: ParamMap) => {
        this.idLibro = Number(paramMaps.get('id'));
      })
  }
}

```

- 4.5. Y en libro.imagenes.component.html ponemos...

```
<p>libro-imagenes works!</p>

<p>
  (Imagenes del libro con identificador: {{idLibro}})
</p>

<!-- <div class="container mt-3" *ngIf="libro != undefined">
  
</div> -->
```

Servicio Router: ejemplos de uso

[Libros](#) [Autores](#)

libro-detalles works!

Detalles de libro

Identificador: 1

Título: El Quijote

Autor: Cervantes

Informacion adicional:

[Imagenes](#)

[Opiniones](#)

libro-imagenes works!

(Imagenes del libro con identificador: 1)

Servicio Router: ejemplos de uso

[Libros](#) [Autores](#)

libro-detalles works!

Detalles de libro
Identificador: 2
Titulo: Hamlet
Autor: Shakespeare

Informacion adicional:
Imagenes
Opiniones

libro-imagenes works!

(Imagenes del libro con identificador: 2)

5. Extras

- 5.1. Creamos el componente del home

ng g c components/home

5.1.1. Pasamos todo el contenido del app.component.html (menos el <router-outlet>) hacia el home.component.html

```
<!-- <p>home works!</p> -->

<div style="text-align: center">
  <h3>Servicio Router: ejemplos de uso</h3>
</div>

<ul class="nav">
  <li class="nav-item">
    <a
      class="nav-link active"
      routerLink="/libros"
      routerLinkActive="active-link"
    >
      Libros
    </a>
  </li>

  <li class="nav-item">
    <a
      class="nav-link"
      routerLink="/autores"
      routerLinkActive="active-link"
    >
      Autores
    </a>
  </li>
</ul>
```

5.1.2. Todo el contenido que hemos quitado del app.component.html, es sustituido por el tag de <app-home>

```
<app-home></app-home>

<router-outlet></router-outlet>
```

5.1.3. En el home.component.css insertamos el contenido que había en app.component.css

```
.active-link {  
  color: orange;  
}
```

5.1.3. Podemos observar que todo se queda igual en el navegador, pero hemos modularizado más.

- 5.2. Creamos el componente de Inicio, en el cual se mostrará una presentación a la app

5.2.1. ng g c components/inicio

5.2.2. El app-routing.module.ts añade la ruta hacia el componente de Inicio, y cambia la redirección de la búsqueda “vacía” del componente libro-lista, hacia este nuevo componente de Inicio

```
const routes: Routes = [  
  {  
    path: 'inicio',  
    component: InicioComponent  
  },  
  {  
    path: 'libros',  
    component: LibroListaComponent  
  },  
  {  
    path: 'libros/:id',  
    component: LibroDetallesComponent,  
    children: [  
      {  
        path: 'imagenes',  
        component: LibroImagenesComponent  
      },  
      {  
        path: 'opiniones',  
        component: LibroOpinionesComponent  
      },  
      {  
        path: '',  
        redirectTo: 'imagenes',  
        pathMatch: 'full'  
      },  
    ],  
  },  
]
```



```

    {
      path: '**',
      component: NotFoundError404Component
    }
  ]
},
{
  path: 'autores',
  component: AutorListaComponent
},
// {
//   path: '',
//   redirectTo: '/libros',
//   pathMatch: 'full'
// },
{
  path: '',
  redirectTo: '/inicio',
  pathMatch: 'full'
},
{
  path: '**',
  component: NotFoundError404Component
}
];

```

5.2.3. En el inicio.component.html ponemos algo sencillo de ejemplo como:

```

<p>inicio works!</p>

<h2>Bienvenidos a mi aplicación</h2>

```

5.2.4. Al navbar del home.component.html le añadimos el enlace hacia el nuevo componente

```

<ul class="nav">
  <li class="nav-item">
    <a
      class="nav-link active"
      routerLink="/inicio"
      routerLinkActive="active-link"
    >

```

```

        Inicio
      </a>
    </li>

    <li class="nav-item">
      <a
        class="nav-link active"
        routerLink="/libros"
        routerLinkActive="active-link"
      >
        Libros
      </a>
    </li>

    <li class="nav-item">
      <a
        class="nav-link"
        routerLink="/autores"
        routerLinkActive="active-link"
      >
        Autores
      </a>
    </li>
  </ul>

```

Servicio Router: ejemplos de uso

Inicio Libros Autores

inicio works!

Bienvenidos a mi aplicación

- 5.3. Vamos a conseguir mostrar las imágenes de las portadas de cada libro

5.3.1. Vamos a la interface del libro-model.ts para añadir un campo más...

```

export interface LibroModel {
  id: number;
  titulo: string;

```

```

    autor: string;
    imagen: string;
}

```

5.3.2. Ahora vamos al libro-mock.ts para quitar las dobles comillas, y añadir la ruta de la imagen al campo de la imagen

```

export const LIBROS: LibroModel[] = [
  {
    id: 1,
    titulo: "El Quijote",
    autor: "Cervantes",
    imagen: "../../assets/img/quijote.jpg"
  },
  {
    id: 2,
    titulo: "Hamlet",
    autor: "Shakespeare",
    imagen: "../../assets/img/hamlet.jpg"
  }
];

```

Nota: para que funcione bien, aquí vamos a cambiarle los IDs, para que el primer objeto que tome el futuro Array al que corresponda manejar este objeto, cuente el primer id del primer objeto como 0.

5.3.3. Vamos al libro-imagenes.component.ts para añadir:

5.3.3.1. Una propiedad llamada libro que es un objeto del LibroModel indefinido

5.3.3.2. Obtenemos el id de ese libro

```

export class LibroImagenesComponent implements OnInit {

  idLibro: number | undefined;

  libro: LibroModel | undefined;

  constructor(
    private activatedRoute: ActivatedRoute
  ) { }
}

```

```

ngOnInit(): void {
  this.activatedRoute.parent?.paramMap
    .subscribe((paramMaps: ParamMap) => {
      this.idLibro = Number(paramMaps.get('id'));

      let idLibro = Number(paramMaps.get('id'));
      this.libro = LIBROS[idLibro];
    })
}
}

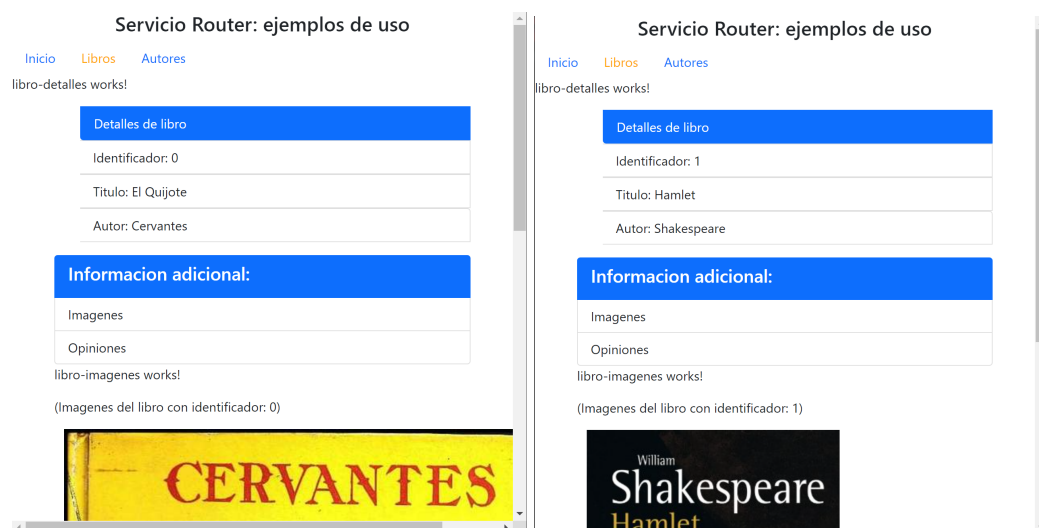
```

5.3.4. En el libro-imagenes.component.html añadimos lo siguiente para mostrar la portada de cada libro

```

<div
  class="container mt-3"
  *ngIf="libro != undefined">
  
</div>

```



- 5.4. Ahora vamos a conseguir mostrar las opiniones de cada libro

5.4.1. ng g interface interfaces/opinion-model

5.4.2. En ese archivo de opinion-model.ts añadimos lo siguiente:

```
export interface OpinionModel {  
  id:number,  
  idLibro:number,  
  titulo:string,  
  descripcion:string  
}
```

5.4.3. Vamos al libro-mock.ts para exportar una nueva constante para las opiniones:

```
export const OPINIONES: OpinionModel[] = [  
  {  
    id: 0,  
    idLibro: 0,  
    titulo: 'Gran libro',  
    descripcion: 'Me ha gustado mucho este libro es uno  
de mis favoritos blablabla'  
  },  
  {  
    id: 1,  
    idLibro: 1,  
    titulo: 'Libro preferido',  
    descripcion: 'Me ha gustado mucho este libro es uno  
de mis favoritos blablabla'  
  },  
  {  
    id: 2,  
    idLibro: 2,  
    titulo: 'Buena Segunda parte',  
    descripcion: 'Me ha gustado mucho este libro es uno  
de mis favoritos blablabla',  
  },  
]
```

5.4.4. Ahora, vamos al libro-opiniones.component.ts para añadir:

- 5.4.4.1. una propiedad del tipo objeto del LibroModel siendo indefinido
- 5.4.4.2. otra propiedad de opiniones del tipo OpinionModel siendo un Array
- 5.4.4.3. Inyectar el ActivatedRoute en el constructor
- 5.4.4.4. Creamos una función en el ngOnInit() para obtener las opiniones a través de los IDs de cada libro
- 5.4.4.5. Creamos alabajo del ngOnInit() un método para obtener las opiniones de cada libro, el cual llamaremos dentro de la función del ngOnInit()

```
export class LibroOpinionesComponent implements OnInit {  
  
  libro : LibroModel | undefined;  
  opiniones: OpinionModel[] = [];  
  
  constructor(  
    private activatedRoute: ActivatedRoute  
  ) { }  
  
  ngOnInit(): void {  
    this.activatedRoute.parent?.paramMap  
      .subscribe((paramMaps: ParamMap) => {  
        let id = Number(paramMaps.get('id'));  
        this.libro = LIBROS[id];  
      })  
    // método para las opiniones  
    this.opinionesDeCadaLibro();  
  }  
  
  opinionesDeCadaLibro(): void {  
    this.opiniones = OPINIONES.filter(  
      (item) => item.idLibro == this.libro?.id);  
  }  
}
```

5.4.5. Luego vamos al libro-opiniones.component.html para insertar esto:

```
<p>libro-opiniones works!</p>
```

```

<div class="container" *ngIf="opiniones != null" >
  <ul class="list-group" *ngFor="let opinion of opiniones">
    <li class="list-group-item">{{opinion.titulo}}</li>
    <li class="list-group-item">{{opinion.descripcion}}</li>
  </ul>
</div>

```

Servicio Router: ejemplos de uso

[Inicio](#)
[Libros](#)
[Autores](#)

libro-detalles works!

Detalles de libro

Identificador: 0

Título: El Quijote

Autor: Cervantes

Información adicional:

Imágenes

Opiniones

libro-opiniones works!

Gran libro

Me ha gustado mucho este libro es uno de mis favoritos blablabla

Servicio Router: ejemplos de uso

[Inicio](#)
[Libros](#)
[Autores](#)

libro-detalles works!

Detalles de libro

Identificador: 1

Título: Hamlet

Autor: Shakespeare

Información adicional:

Imágenes

Opiniones

libro-opiniones works!

Libro preferido

Me ha gustado mucho este libro es uno de mis favoritos blablabla

6. Lazy Loading

- 6.1. Vamos a crear un par de módulos para los componentes de la carpeta libros

ng g m libros/libro --flat

ng g m libros/libro-routing --flat

6.1.1. Ahora vamos al app-routing.module.ts para comentar toda la ruta y los hijos de libros, quedando de esta manera:

```
const routes: Routes = [
  {
    path: 'inicio',
    component: InicioComponent
  },
  {
    path: 'libros',
    // component: LibroListaComponent
    loadChildren: () => import('./libros/libros.module').then((m)
=> m.LibrosModule)
  },
  // {
  //   path: 'libros/:id',
  //   component: LibroDetallesComponent,
  //   children: [
  //     {
  //       path: 'imagenes',
  //       component: LibroImagenesComponent
  //     },
  //     {
  //       path: 'opiniones',
  //       component: LibroOpinionesComponent
  //     },
  //     {
  //       path: '',
  //       redirectTo: 'imagenes',
  //       pathMatch: 'full'
  //     },
  //     {
  //       path: '**',
  //       component: NotFoundError404Component
  //     }
  //   ]
  // },
  // ],
  // ],
```



```

{
  path: 'autores',
  component: AutorListaComponent
},
// {
//   path: '',
//   redirectTo: '/libros',
//   pathMatch: 'full'
// },
{
  path: '',
  redirectTo: '/inicio',
  pathMatch: 'full'
},
{
  path: '**',
  component: NotFoundError404Component
}
];

```

6.1.2. Lo que hemos comentado en app-routing.module.ts lo vamos a copiar y pegar en el nuevo libros-routing.module.ts

```

import { NgModule } from '@angular/core';
// import { CommonModule } from '@angular/common';
import { RouterModule, Routes } from '@angular/router';
import { LibroListaComponent } from
'./libro-lista/libro-lista.component';
import { LibroDetallesComponent } from
'./libro-detalles/libro-detalles.component';
import { LibroImagenesComponent } from
'./libro-imagenes/libro-imagenes.component';
import { LibroOpinionesComponent } from
'./libro-opiniones/libro-opiniones.component';
import { NotFoundError404Component } from
'../components/not-found-error404/not-found-error404.component';

const routes: Routes = [
  {
    path: '',
    component: LibroListaComponent
  },

```

```

    {
      path: 'libros/:id',
      component: LibroDetallesComponent,
      children: [
        {
          path: 'imagenes',
          component: LibroImagenesComponent
        },
        {
          path: 'opiniones',
          component: LibroOpinionesComponent
        },
        {
          path: '',
          redirectTo: 'imagenes',
          pathMatch: 'full'
        },
        {
          path: '**',
          component: NotFoundError404Component
        }
      ]
    },
  ],
]

@NgModule({
  declarations: [],
  imports: [
    // CommonModule
    RouterModule.forChild(routes)
  ],
  exports: [
    RouterModule
  ]
})

export class LibrosRoutingModule { }

```

6.1.3. Ahora, en el libros.module.ts importamos el libros-routing.module.ts

```

@NgModule({
  declarations: [],

```

```

imports: [
  CommonModule,
  LibrosRoutingModule
]
}))

```

6.1.4. Y por último, en libro-lista.component.html, quitamos la barra de la ruta libros

```

<div class="container">
  <ul class="list-group-horizontal">
    <li class="list-group-item active">
      LIBROS
    </li>

    <li
      class="list-group-item"
      *ngFor="let libro of libros"
    >
      <!-- <a [routerLink]="['/libros', libro.id]"> -->
      <a [routerLink]="['libros', libro.id]">
        {{ libro.titulo }}
      </a>
    </li>
  </ul>
</div>

```

6.1.5. Y ya vuelve a funcionar todo correctamente como antes !!

- 6.2. Ahora, vamos hacer lo mismo para los autores ...

```

ng g m autores/autor --flat
ng g m autores/autor-routing --flat

```

6.2.1. Ahora vamos al app-routing.module.ts para comentar la ruta hacia el componente de los autores y después copiarlo y pegarlo en su routing.module correspondiente...

```

const routes: Routes = [
  {
    path: 'inicio',
    component: InicioComponent

```

```

},
{
  path: 'libros',
  // component: LibroListaComponent
  loadChildren: () => import('./libros/libros.module').then((m)
=> m.LibrosModule)
},
{
  path: 'autores',
  // component: AutoresListaComponent
  loadChildren: () => import('./autores/autor.module').then((m)
=> m.AutorModule)
},
// {
//   path: 'libros/:id',
//   component: LibroDetallesComponent,
//   children: [
//     {
//       path: 'imagenes',
//       component: LibroImagenesComponent
//     },
//     {
//       path: 'opiniones',
//       component: LibroOpinionesComponent
//     },
//     {
//       path: '',
//       redirectTo: 'imagenes',
//       pathMatch: 'full'
//     },
//     {
//       path: '**',
//       component: NotFoundError404Component
//     }
//   ]
// },
// {
//   path: 'autores',
//   component: AutorListaComponent
// },
// {
//   path: '',
//   redirectTo: '/libros',

```

```

//   pathMatch: 'full'
// },
{
  path: '',
  redirectTo: '/inicio',
  pathMatch: 'full'
},
{
  path: '**',
  component: NotFoundError404Component
}
];

```

6.2.2. Lo que hemos comentado en app-routing.module.ts lo vamos a copiar y pegar en el nuevo autor-routing.module.ts

```

import { NgModule } from '@angular/core';
// import { CommonModule } from '@angular/common';
import { AutorListaComponent } from
'./autor-lista/autor-lista.component';
import { RouterModule, Routes } from '@angular/router';

const routes: Routes = [
  {
    path: '',
    component: AutorListaComponent
  },
]

@NgModule({
  declarations: [],
  imports: [
    // CommonModule
    RouterModule.forChild(routes)
  ],
  exports: [
    RouterModule
  ]
})
export class AutorRoutingModule { }

```

6.2.3. Ahora, en el `autor.module.ts` importamos el `autor-routing.module.ts`

```
@NgModule({  
  declarations: [],  
  imports: [  
    CommonModule,  
    AutorRoutingModule  
  ]  
})
```

6.2.4. Y con esto, ya vuelve a funcionar todo correctamente !!

Nota: Tener en cuenta que, en el `autor-lista.component.html`, aún no tenemos nada...

7. Crear la vista del componente de autores

- 7.1. Vamos al autor-lista.component.ts

7.1.1. Creamos una propiedad de autoresLibros del tipo del LibroModel siendo un Array

7.1.2. Creamos una propiedad de tipo Set<String>

7.1.3. en el ngOnInit() recorreremos con un For el Array de autoresLibros, y para cada objeto, iremos añadiendo el campo del autor al Array

Nota: Usamos un Set para que no se repitan los autores que aparezcan en más de un libro

```
export class AutorListaComponent implements OnInit {  
  
  autoresLibros: LibroModel[] = [];  
  autores: Set<String> = new Set();  
  
  constructor() { }  
  
  ngOnInit(): void {  
    this.autoresLibros = LIBROS;  
  
    for(let i=0; i<this.autoresLibros.length; i++){  
      this.autores.add(this.autoresLibros[i].autor);  
    }  
  }  
}
```

- 7.2. Ahora vamos al autor-lista.component.html

7.1.4. Añadimos un para mostrar todos los autores a través de un *ngFor como siempre ...

```
<p>autor-lista works!</p>  
  
<ul class="list-group-horizontal">  
  <li class="list-group-item active w-50 m-auto">Autores</li>  
  <li class="list-group-item w-50 m-auto" *ngFor="let autor of  
autores">{{autor}}</li>
```

```
</ul>
```

7.1.4 Comprobamos que los autores se muestran perfectamente !!

Servicio Router: ejemplos de uso

[Inicio](#) [Libros](#) [Autores](#)

autor-lista works!

Autores
Cervantes
Shakespeare

8. Servicios: Definición y uso mediante inyección de dependencias

- 8.1. El componente LibroLista obtiene los libros por sí solo. Esta funcionalidad previsiblemente será usada por otros componentes, por lo que lo mejor será dejarlo como servicio. Vamos a crearlo. También crearemos un segundo servicio para grabar log que será usado por el primero:

```
ng g s services/libro
ng g s services/logger
```

- 8.2. En logger.service creamos una función de log

```
export class LoggerService {

  constructor() { }

  log(message: string) {
    console.log("(" + new Date().toLocaleTimeString() + ") " +
message);
  }
}
```

- 8.3. A continuación, añadiremos el código a libro.service y le inyectaremos el logger.service para que lo use

```
export class LibroService {

  constructor(
    private loggerService: LoggerService
  ) { }

  getLibros() {
    this.loggerService.log("Llamada realizada sobre
LibroService.getLibros");

    return LIBROS;
  }
}
```

- 8.4. Finalmente modificaremos nuestro componente libro-lista para que utilice el servicio libro.service

```
export class LibroListaComponent implements OnInit {  
  
  libros: LibroModel[] = [];  
  
  constructor(  
    private libroService: LibroService  
  ) { }  
  
  ngOnInit(): void {  
    // this.libros = LIBROS;  
    this.libros = this.libroService.getLibros();  
  }  
}
```

- 8.5. Con el inspeccionar de Chrome podremos ver que ha funcionado

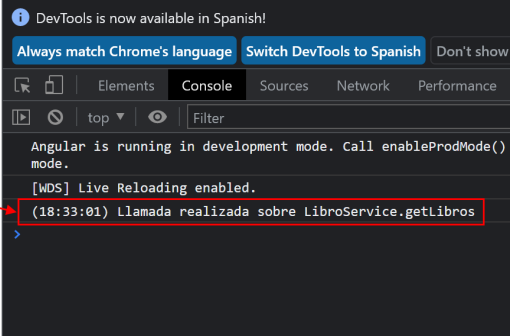
Servicio Router: ejemplos de uso

[Inicio](#) [Libros](#) [Autores](#)

libro-lista works!

Libros:

LIBROS
El Quijote
Hamlet



DevTools is now available in Spanish!

Always match Chrome's language Switch DevTools to Spanish Don't show

Elements Console Sources Network Performance

Angular is running in development mode. Call enableProdMode() mode.

[WDS] Live Reloading enabled.

(18:33:01) Llamada realizada sobre LibroService.getLibros

9. Servicios: Gestión asíncrona con promesas

Muchas veces tendremos servicios que realizarán llamadas a servidores remotos cuyas respuestas no sabemos cuánto tiempo tardarán en llegar.

Esos servicios no pueden gestionarse de la misma forma, ya que produciríamos bloqueos inaceptables en la aplicación. Esos servicios hay que gestionarlos de forma asíncrona. De esta manera, el usuario podrá seguir trabajando mientras se ejecutan los servicios.

Las promesas son una de las herramientas que nos permite la programación asíncrona.

La función ejecutor empieza a ejecutarse de forma asíncrona justo en el momento en el que se crea la promesa.

La función ejecutor, al terminar, deberá llamar a la función resolver, para resolver la promesa con el valor obtenido, o, llamar a la función rechazar para rechazarla con el error producido.

Finalmente, mediante los métodos “then” y “catch” de la promesa, gestionaremos esa resolución y rechazo, respectivamente.

Una llamada al método “then” de una promesa devuelve otra promesa a la que también podemos llamar a su método “then”, y así sucesivamente. Esto nos permitirá encadenar promesas.

- 9.1. Vamos a editar el código del servicio “LibroService” y realizamos las modificaciones comentadas:

```
export class LibroService {  
  
  constructor(  
    private loggerService: LoggerService  
  ) { }  
  
  // getLibros() {  
  //   this.loggerService.log("Llamada realizada sobre  
LibroService.getLibros");  
  
  //   return LIBROS;  
  // }  
  
  getLibros(): Promise<LibroModel[]> {  
  
    return new Promise<LibroModel[]>( (resolve, reject) => {
```

```

        this.loggerService.log("Inicio ejecutor (Promise de
LibroService.getLibros())");

        setTimeout(() => {
            this.loggerService.log("Fin ejecutor (Promise de
LibroService.getLibros())");

            resolve(LIBROS);
        }, 5000);
    });
}
}

```

- 9.2. Finalmente modificaremos el libro-lista.component.ts para que gestione la promesa:

```

export class LibroListaComponent implements OnInit {

    libros: LibroModel[] = [];

    constructor(
        private libroService: LibroService
    ) { }

    ngOnInit(): void {
        // this.libros = LIBROS;
        // this.libros = this.libroService.getLibros();
        this.libroService.getLibros().then((libros) => this.libros =
libros);
    }
}

```

- 9.3. Con el inspeccionar de Chrome podremos ver que ha funcionado

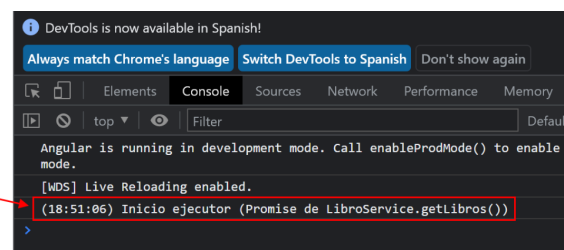
Servicio Router: ejemplos de uso

Inicio **Libros** Autores

libro-lista works!

Libros:

LIBROS

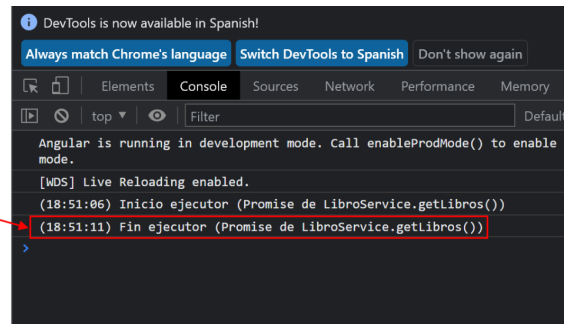


Servicio Router: ejemplos de uso

[Inicio](#) [Libros](#) [Autores](#)
libro-lista works!

Libros:

LIBROS
El Quijote
Hamlet



- 9.4. Y ahora, para probar el encadenamiento de promesas, realizaremos los siguientes cambios:

```
export class LibroListaComponent implements OnInit {

  libros: LibroModel[] = [];

  constructor(
    private libroService: LibroService,
    private loggerService: LoggerService
  ) { }

  ngOnInit(): void {
    // this.libros = LIBROS;
    // this.libros = this.libroService.getLibros();
    // this.libroService.getLibros().then((libros) => this.libros =
libros);
    this.libroService.getLibros()
      .then((libros) => {
        this.loggerService.log("Ejecución del 1º then");

        return libros;
      })
      .then((libros) => {
        this.loggerService.log("Ejecución del 2º then");

        return new Promise<LibroModel[]>((resolve, reject) => {
          this.loggerService.log("Inicio ejecutor (Promise del 2º
then)");

          setTimeout(() => {
            this.loggerService.log("Fin ejecutor (Promise del 2º
then)");
```

```

        resolve(libros);
      }, 5000);
    })
  })
  .then((libros) => {
    this.loggerService.log("Ejecución del 3º then");
    this.libros = LIBROS;
  })
}
}

```

- 9.5. Con el inspeccionar de Chrome podremos ver que ha funcionado

Servicio Router: ejemplos de uso

Inicio **Libros** Autores

libro-lista works!

Libros:

LIBROS

DevTools is now available in Spanish!

Always match Chrome's language Switch DevTools to Spanish Don't show again

Elements Console Sources Network Performance Memory

Angular is running in development mode. Call enableProdMode() to enable mode.

[WDS] Live Reloading enabled.

(19:03:32) Inicio ejecutor (Promise de LibroService.getLibros())

Servicio Router: ejemplos de uso

Inicio **Libros** Autores

libro-lista works!

Libros:

LIBROS

DevTools is now available in Spanish!

Always match Chrome's language Switch DevTools to Spanish Don't show again

Elements Console Sources Network Performance Memory

Angular is running in development mode. Call enableProdMode() to enable mode.

[WDS] Live Reloading enabled.

(19:04:02) Inicio ejecutor (Promise de LibroService.getLibros())

(19:04:07) Fin ejecutor (Promise de LibroService.getLibros())

(19:04:07) Ejecución del 1º then

(19:04:07) Ejecución del 2º then

(19:04:07) Inicio ejecutor (Promise del 2º then)

Servicio Router: ejemplos de uso

Inicio **Libros** Autores

libro-lista works!

Libros:

LIBROS

El Quijote

Hamlet

DevTools is now available in Spanish!

Always match Chrome's language Switch DevTools to Spanish Don't show again

Elements Console Sources Network Performance Memory

Angular is running in development mode. Call enableProdMode() to enable mode.

[WDS] Live Reloading enabled.

(19:06:05) Inicio ejecutor (Promise de LibroService.getLibros())

(19:06:10) Fin ejecutor (Promise de LibroService.getLibros())

(19:06:10) Ejecución del 1º then

(19:06:10) Ejecución del 2º then

(19:06:10) Inicio ejecutor (Promise del 2º then)

(19:06:15) Fin ejecutor (Promise del 2º then)

(19:06:15) Ejecución del 3º then

10. Servicios: Gestión asíncrona con observables (Librería RxJs) (parte I) (pag 207 pdf)

- 10.1. Creamos un nuevo servicio para hacer la prueba con el <Observable>

ng g s servicios/libroObservable

- 10.2. En el nuevo LibroObservableService:

10.2.1. Creamos la propiedad de libros de tipo LibroModel siendo un Array

10.2.2. Creamos el nuevo método de getLibros() con el objeto Observable

```
export class LibroObservableService {

  libros: LibroModel[] = [];

  constructor() { }

  getLibros(): Observable<LibroModel[]> {
    return new Observable<LibroModel[]>((observer) => {

      let libros: LibroModel[] = [];
      observer.next([]);

      LIBROS.forEach((libro, index) => {
        setTimeout(() => {
          libros.push(libro);
          observer.next(libros);
        }, (index + 1) * 1500);
      });

      setTimeout(() => {
        observer.complete();
      }, (LIBROS.length + 1) * 1500);
    });
  }
}
```

- 10.3. En el libro-lista.component.ts ...

10.3.1. Comentamos el trozo de código anterior dentro del ngOnInit() que hicimos para las promesas

10.3.2. Añadimos una nueva propiedad de observableSubs de tipo Subscription siendo undefined

10.3.3. Comentamos la antigua inyección de LibroService, para poner en su lugar el nuevo LibroObservableService

10.3.4. Añadimos el nuevo código para el ngOnInit() para el subscribe()

10.3.5. Añadimos el método de ngOnDestroy() para el unsubscribe()

```
export class LibroListaComponent implements OnInit {

  libros: LibroModel[] = [];

  // observableSubs: Observable<LibroModel[]> | undefined; // esto no
  // sería así
  observableSubs: Subscription | undefined;

  constructor(
    // private libroService: LibroService,
    private loggerService: LoggerService,
    private libroObservableService : LibroObservableService
  ) { }

  ngOnInit(): void {
    // this.libros = LIBROS;
    // this.libros = this.libroService.getLibros();
    // this.libroService.getLibros().then((libros) => this.libros =
    // libros);

    // this.libroService.getLibros()
    //   .then((libros) => {
    //     this.loggerService.log("Ejecución del 1º then");

    //     return libros;
    //   })
    //   .then((libros) => {
    //     this.loggerService.log("Ejecución del 2º then");

    //     return new Promise<LibroModel[]>((resolve, reject) => {
    //       this.loggerService.log("Inicio ejecutor (Promise del 2º
    // then)");
```



```

        //      setTimeout(() => {
        //          this.loggerService.log("Fin ejecutor (Promise del 2°
then)");

        //      resolve(libros);
        //      }, 5000);
        //  })
        //  })
        //  .then((libros) => {
        //      this.loggerService.log("Ejecución del 3° then");
        //      this.libros = LIBROS;
        //  })

        this.observableSubs = this.libroObservableService.getLibros()
            .subscribe(
                (libros) => this.libros = libros,
                (error) => console.log(error),
                () => console.log("this.libroObservableService.getLibros()
FINALIZADO")
            );
    }

    ngOnDestroy(): void {
        if(this.observableSubs){
            this.observableSubs.unsubscribe();
        }
    }
}

```

10.3.6. En el inspeccionar del Chrome, podremos ver que funciona así...

The screenshot shows a web browser at `localhost:4200/libros`. The page title is "Servicio Router: ejemplos de uso". There are three navigation links: "Inicio", "Libros", and "Autores". Below the links, it says "libro-lista works!". Under the heading "Libros:", there is a blue button labeled "LIBROS". The Chrome DevTools console on the right shows two messages: "Angular is running in development mode. Call enableProdMode() to enable production mode." and "[WDS] Live Reloading enabled."

The screenshot shows the same web browser page as before. The "LIBROS" button is highlighted with a red box. Below it, a link "El Quijote" is visible and also highlighted with a red box. The Chrome DevTools console on the right shows the same two messages as in the previous screenshot.

The screenshot shows the same web browser page. The "LIBROS" button is highlighted with a blue box. Below it, the link "El Quijote" is visible. At the bottom, a link "Hamlet" is visible and highlighted with a red box. The Chrome DevTools console on the right shows the same two messages as in the previous screenshots.

Servicio Router: ejemplos de uso

[Inicio](#) [Libros](#) [Autores](#)

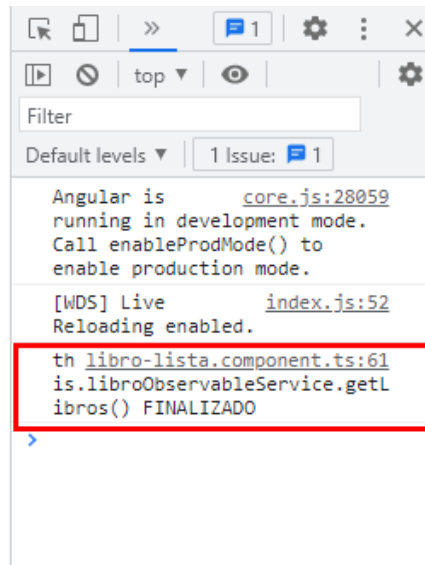
libro-lista works!

Libros:

LIBROS

[El Quijote](#)

[Hamlet](#)



11. Servicios: Gestión asíncrona con observables (Librería RxJs) (parte II) (pag 211 pdf)

ejemplo en internet: <https://coryryan.com/blog/angular-observable-data-services>

Nota: esto NO llegamos a hacerlo...

12. HttpClient: Introducción e instalación

- 12.1. Creamos un nuevo componente:

ng g c components/httpClientTest

- 12.2. Añadimos su ruta al app-routing.module.ts

```
{
  path: 'inicio',
  component: InicioComponent
},
{
  path: 'libros',
  // component: LibroListaComponent
  loadChildren: () => import('./libros/libros.module').then((m) =>
m.LibrosModule)
},
{
  path: 'autores',
  // component: AutoresListaComponent
  loadChildren: () => import('./autores/autor.module').then((m) =>
m.AutorModule)
},
{
  path: 'http',
  component: HttpClientTestComponent
},
}
```

- 12.3. Añadimos su enlace en el navbar del home.component.html

```
<ul class="nav">
  <li class="nav-item">
    <a
      class="nav-link active"
      routerLink="/inicio"
      routerLinkActive="active-link"
    >
      Inicio
    </a>
  </li>
```

```

<li class="nav-item">
  <a
    class="nav-link active"
    routerLink="/libros"
    routerLinkActive="active-link"
  >
    Libros
  </a>
</li>

<li class="nav-item">
  <a
    class="nav-link"
    routerLink="/autores"
    routerLinkActive="active-link"
  >
    Autores
  </a>
</li>

<li class="nav-item">
  <a
    class="nav-link"
    routerLink="/http"
    routerLinkActive="active-link"
  >
    Http
  </a>
</li>
</ul>

```

- 12.4. Ahora, en el app.module.ts

```

...
import { HttpClientModule } from '@angular/common/http';

@NgModule({
  declarations: [
    AppComponent,
    LibroListaComponent,
    AutorListaComponent,
    NotFoundError404Component,

```

```

        LibroDetallesComponent,
        LibroOpinionesComponent,
        LibroImagenesComponent,
        HomeComponent,
        InicioComponent,
        HttpClientTestComponent
    ],
    imports: [
        BrowserModule,
        AppRoutingModule,
        HttpClientModule
    ],
    providers: [],
    bootstrap: [AppComponent]
  })
  ...

```

- 12.5. En el http-client-test.component.ts inyectamos el HttpClient

```

export class HttpClientTestComponent implements OnInit {

  constructor(
    private httpClient: HttpClient
  ) { }

  ngOnInit(): void {
  }

}

```

13. HttpClient: Operaciones Get y Post

- 13.1. A continuación, añadimos las modificaciones en el http-client-test.component.ts

```
export class HttpClientTestComponent implements OnInit {

  resultadoPetición: any;

  constructor(
    private httpClient: HttpClient
  ) { }

  ngOnInit(): void {
    this.get();
  }

  get() {
    this.httpClient.get('https://jsonplaceholder.typicode.com/posts')
      .subscribe((data) => {
        this.resultadoPetición = data;
      });
  }

  post() {
    this.httpClient.post('https://jsonplaceholder.typicode.com/posts',
      {
        title: 'Previsión Viernes.',
        body: 'Parcialmente soleado.',
        userId: 1
      })
      .subscribe((data) => {
        this.resultadoPetición = data;
      });
  }
}
```

- 13.2. Por último, en su html ponemos lo siguiente:

```
<p>http-client-test works!</p>

<h3>Servicio HttpClient: ejemplos de uso</h3>
```

```
<hr>

<h5>Petición:</h5>
<div class="btn-group" role="group" aria-label="Basic example">
  <button
    type="button"
    class="btn btn-secondary"
    (click)="get() "
  >
    Get
  </button>

  <button
    type="button"
    class="btn btn-secondary"
    (click)="post() "
  >
    Post
  </button>
</div>

<hr>

<h5>Resultado:</h5>
<pre>{{resultadoPeticion | json}}</pre>
```


Servicio Router: ejemplos de uso

[Inicio](#) [Libros](#) [Autores](#) [Http](#)

http-client-test works!

Servicio HttpClient: ejemplos de uso

Petición:

Get Post

Resultado:

```
[
  {
    "userId": 1,
    "id": 1,
    "title": "sunt aut facere repellat provident occaecati excepturi opto",
    "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita",
  },
  {
    "userId": 1,
    "id": 2,
    "title": "qui est esse",
    "body": "est rerum tempore vitae\nsequi sint nihil reprehenderit dolor",
  },
  {
    "userId": 1,
    "id": 3,
    "title": "ea molestias quasi exercitationem repellat qui ipsa sit aut",
    "body": "et iusto sed quo iure\nvoluptatem occaecati omnis eligendi ad",
  },
]
```

Servicio Router: ejemplos de uso

[Inicio](#) [Libros](#) [Autores](#) [Http](#)

http-client-test works!

Servicio HttpClient: ejemplos de uso

Petición:

Get Post

Resultado:

```
{
  "title": "Previsión Viernes.",
  "body": "Parcialmente soleado.",
  "userId": 1,
  "id": 101
}
```

- 13.3. Si fuera a trabajar con los datos... en el html...

```
<h5>Resultado:</h5>
<!-- <pre>{{resultadoPeticion | json}}</pre> -->
<ul class="list-group-horizontal">
  <li class="list-group-item" *ngFor="let post of resultadoPeticion">
    <span>{{ post.title }}</span>
    <span>{{ post.body }}</span>
  </li>
</ul>
```

Servicio Router: ejemplos de uso

[Inicio](#) [Libros](#) [Autores](#) [Http](#)

http-client-test works!

Servicio HttpClient: ejemplos de uso

Petición:

Get Post

Resultado:

- sunt aut facere repellat provident occaecati excepturi optio reprehenderitquia et suscipit suscipit recusandae consequuntur expedita et cum reprehenderit molestiae ut ut quas totam nostrum rerum est autem sunt rem eveniet architecto
- qui est esseest rerum tempore vitae sequi sint nihil reprehenderit dolor beatae ea dolores neque fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis qui aperiam non debitis possimus qui neque nisi nulla
- ea molestias quasi exercitationem repellat qui ipsa sit autet iusto sed quo iure voluptatem occaecati omnis eligendi aut ad voluptatem doloribus vel accusantium quis pariatur molestiae porro eius odio et labore et velit aut
- eum et est occaecatiullam et saepe reiciendis voluptatem adipisci sit amet autem assumenda provident rerum culpa quis hic commodi nesciunt rem tenetur doloremque ipsam iure quis sunt voluptatem rerum illo velit
- nesciunt quas odiorepudiandae veniam quaerat sunt sed alias aut fugiat sit autem sed est voluptatem omnis possimus esse voluptatibus quis est aut tenetur dolor neque

... y en el post() del ts...

```
post() {
  this.httpClient.post('https://jsonplaceholder.typicode.com/posts',
    {
      title: 'Previsión Viernes.',
    })
}
```

```
    body: 'Parcialmente soleado.',  
    userId: 1  
  })  
  .subscribe((data) => {  
    // this.resultadoPetición = data;  
  
    this.resultadoPetición = [];  
    this.resultadoPetición.push(data);  
  });  
}
```

Servicio Router: ejemplos de uso

[Inicio](#) [Libros](#) [Autores](#) [Http](#)

http-client-test works!

Servicio HttpClient: ejemplos de uso

Petición:

Get Post

Resultado:

Previsión Viernes.Parcialmente soleado.