Navegación / Routing

Permite navegar dentro de la aplicación a otras partes de la página (SPA) reescribiendo la URL en el navegador. Por ejemplo, si tenemos una barra de navegación y varios componentes, podremos navegar entre ellos. El componente principal es el encargado de realizar toda la gestión de enrutamiento.

Pasos a realizar:

1.- En el componente donde vayamos a usar la navegación, hay que importar las rutas de @angular/router. Ejemplo:

@Componente(….
Imports:[CommonModule**, RouterOutlet, RouterModule],**

import { RouterModule , RouterOutlet } from '@angular/router';

2.- Definir rutas

Dependiendo de la versión de Angular, el archivo de configuración de rutas puede variar:
- Fichero app.routes.ts
- Fichero routing.module.ts

Hay que definir un array con todas las rutas que necesitamos ir definiendo en nuestra aplicación para la navegación. Cada componente será una vista.

const routes: Routes = [];

Ejemplo:
```
const routes: Routes = [
 { path: ‘‘ , component: AppComponent },
 { path: 'planetas', component: PlanetListComponent },
{ path: 'planetas/:id', component: PlanetDetailComponent },
 { path: 'acerca-de', component: AboutComponent },
 { path: '**', redirectTo: '/home' },

];
```

Es necesario: import { RouterModule, Routes } from '@angular/router';

3.- Menú con enlaces para poder navegar.

Usamos [routerLink]=”” o href=”” para navegar a las distintas vistas

** Libros **
o
**<a [routerLink]=”[‘/libros’]”> Libros **

<a routerLink="/">Home

Revisar si está importado RouterLink en el componente donde estén los enlaces:
imports: [RouterLink]

4.- Incluir la etiqueta <router-outlet></router-outlet>

<router-oulet> carga el **componente correspondiente** según la ruta. Se suele incluir en el componente principal.

Ojo! Recordad importar RouterOutlet en el componente donde se vaya a utilizar:
`imports: [RouterOutlet]`

Redirecciones

Si queremos que en algún punto de un componente, por ejemplo tras pulsar un botón, se redirija automáticamente a algún otro componente, tendremos que usar el ***servicio Router***.

Paso1

En el constructor debemos tener un parámetro: **private router: Router**

Paso 2

En la función del componente, llamar al método **navigate** **con el path de la redirección.**

Ejemplo: **this.router.navigate(['/contacto'])**

Donde router es el servicio del Router.

Si queremos que desde un enlace en html, <a>, o desde algún botón, se redirija a otro componente:

**<a [routerLink]="['/contacto']" > Enlace **
**<a [routerLink]="['/libros', book.id]" > Enlace **

¿Cómo rescatamos el parámetro que se pasa en una url? Por ejemplo el id:
En el constructor tenemos que utilizar el servicio **ActivatedRoute** (**private act: ActivatedRoute)**, y utilizarlo de la siguiente forma:

let id = this.act.snapshot.params['id'];

Donde act es el objeto del servicio ActivatedRoute.

ESTILOS CSS

1.- Buscar en google, tailwind css cdn
2.- Copiar la linea de <script> en index.html
3.- Buscar el estilo que queramos en tailblocks.cc y copiar el código!

Ejercicio 1

- Crear un proyecto blog de noticias que tendrá los siguientes componentes: navbar, home, contact y about. La aplicación debe tener un menú con las opciones home, contact y about.
  - Componente principal: incluirá el componente navbar.
  - Componente navbar: tendrá un menú de 3 enlaces: home, contact y about.. Cada uno será un componente. (No olvidar importar RouterLink en el componente navbar)
    Si se carga la raiz de la aplicación o cualquier otra url que no sea la definida en las rutas, debe cargar el componente principal.

Ejercicio 2

- Crear un proyecto biblioteca de la siguiente manera. Tendrá 5 componentes:

  - El **componente principal**:
    - Tiene un título centrado, h1
    - Tiene un menú con las siguientes entradas: Home, Libros y Contacto. Por defecto, la aplicación debe mostrar el HomeComponent.
  - El componente **HomeComponent**:
    - Mensaje central con nuestro texto.
    - Una imagen de nuestra librería.
  - El componente **Libros**
    - Mostrará el listado de los libros de nuestra biblioteca. Los libros los vamos a guardar en el servicio LibrosService (en una carpeta services) en un array de 4 libros. Tendremos una interfaz Libro (en una carpeta model: model/libro.model.ts). Cada libro almacenará el id, un titulo y un autor.
    - El componente Libros mostrará un título: Libros de la biblioteca. Y a continuación, la lista de todos los libros mostrando: id-titulo.
    - Al pulsar el id-titulo (será un enlace), se cargará el componente **LibroDetalle.**
      - El componente LibroDetalle mostrará el título del libro en h2. Y a continuación su id y su descripción en dos div.
      - Tendrá un botón: Volver a los libros, que se encarga de volver al listado de libros.

  - El componente **Contacto.**
    - Mostrará La dirección donde se ubica la librería

Ayuda:

1. Crear proyecto
2. Crear los componentes: ng g c ….
3. ¿Dónde va el menú? Ahí ponemos **<router-outlet></router-outlet>**
4. En app.routes.ts ponemos las rutas → importar componentes
5. En el componente principal, creamos el menú con los enlaces y routerLink. → importar RouterLink
6. Continuar con cada componente.

7. Para llamar a un componente en ruta:EJ: **<a [routerLink]="['/libros', book.id]" > Enlace **

Y en rutas: `path: 'detalle/:id', component: LibroDetalleComponent`

NOTA: En index.html incluir enlaces de bootstrap o tailwindcss

```
<link  rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css"  />
```

Ejercicio 3

- Crear un nuevo componente: **LibroNuevo**.
  En el listado de los libros, en el componente Libros, a continuación del listado, habrá un botón que redirija al componente LibroNuevo.
  Será un formulario con los campos: *Nombre y autor, y c*on un botón *Alta de libro*, que añadirá al listado de libros el nuevo libro. El id se calculará de la siguiente manera: Tomará el último id del array de libros añadiéndole una unidad. **Una vez añadido el libro, automáticamente se redirigirá a la página de listado de libros.**
  También habrá un botón 'Atrás' que vuelve al listado de libros.

- Añadir un nuevo componente: **LibroModificacion**
  En el listado de libros , al lado del nombre del libro, habrá un icono que cuando lo pulsemos cargará el componente libroModificacion. Mostrará un formulario con los datos de dicho libro en campos input para modificarlos si deseamos. El formulario tendrá un botón 'guardar cambios' y automáticamente redirigirá al listado de libros y otro 'volver' que vuelve al listado de libros.

Ejercicio 4

Vamos a crear una aplicación de usuarios accediendo a datos de la api:
https://jsonplaceholder.typicode.com/users

- La aplicación debe tener un menú de navegación con las opciones: Home y Usuarios
- Componente home: Página principal de la aplicación con un texto central en h1.
- Componente lista-usuarios: este componente debe mostrar el listado de usuarios: id, name y username.

  Deben aparecer dentro de una etiqueta <a> para que al pulsar un usuario, se muestre el detalle del usuario. Por lo tanto necesitamos crear otro componente: detalle-usuario. Se mostrará:
  - el nombre, correo y username del usuario
  - de la empresa: los campos name, catchPhrase y bs.
  - de la dirección: street, suite y city.
  - Crear un botón 'Atrás' que redirija al listado de usuarios.