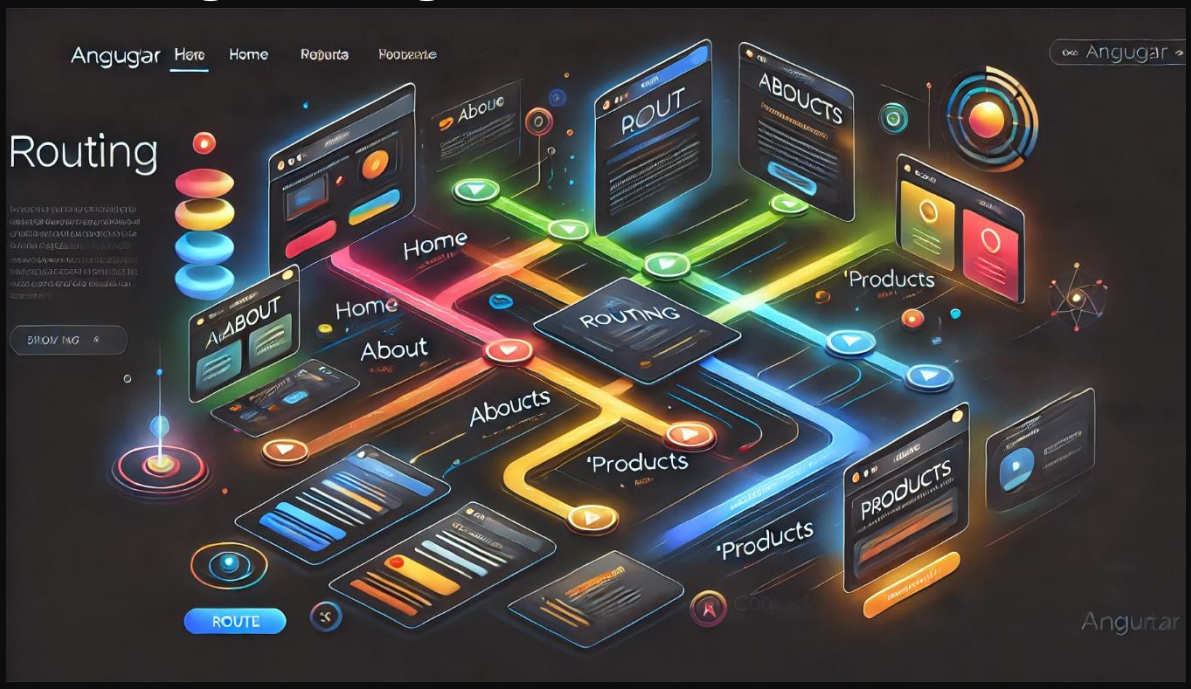
# Routing en Angular

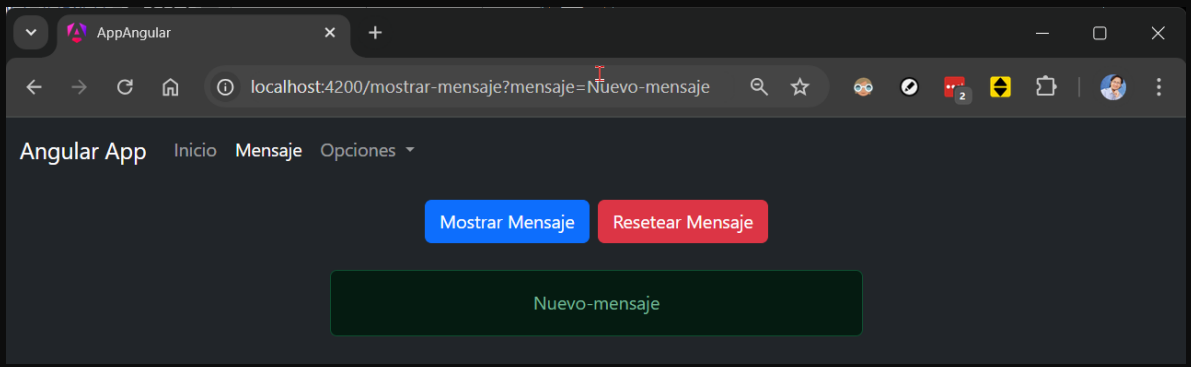



## Routing en Angular

### ¿Qué es el Routing en Angular?

**Routing** en Angular, es el mecanismo que permite navegar entre diferentes vistas o componentes en una aplicación de una sola página (SPA - Single Page Application). A través del routing, podemos crear una experiencia de navegación sin necesidad de recargar toda la página. En lugar de cargar una nueva página desde el servidor, Angular simplemente actualiza el contenido dentro del DOM de la aplicación en función de la ruta solicitada.

En términos más simples, el routing permite definir y gestionar las rutas (URLs) dentro de la aplicación, mapeándolas a diferentes componentes.



**Sintaxis básica del Routing**

Para configurar el routing en Angular, se utiliza el **Angular Router**, que es parte del paquete `@angular/router`. La configuración del router se realiza en el archivo `app.routes.ts`, y aquí es donde se definen las rutas que la aplicación puede manejar.

**Pasos básicos para configurar el Routing en Angular:**

1. **Confirmar el archivo de rutas**: Angular crea de manera automática el archivo app.routes.ts para trabajar con el concepto de routing cuando se crea un nuevo proyecto.
2. **Definir las rutas**: En el archivo `app.routes.ts`, se definen las rutas usando un arreglo de objetos.
3. **Agregar el router outlet**: Este es el lugar donde se renderizarán los componentes que correspondan a la ruta solicitada en la plantilla donde se agregue este componente de <app-outlet/>.

**Ejemplo básico de Routing**

1. **Modificar el archivo de rutas `app.routes.ts` (agregarlo a nivel de la carpeta app en caso de que no exista)**:

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { InicioComponent } from './inicio/inicio.component';
import { AcercaComponent } from './acerca/acerca.component';

// Definimos las rutas
const routes: Routes = [
  { path: '', component: InicioComponent }, // Ruta para el componente Inicio
  { path: 'acerca', component: AcercaComponent }, // Ruta para componente Acerca
];
```

2. **Agregar el `<router-outlet>` en `app.component.html`**:

   El **`<router-outlet>`** es un marcador de posición donde Angular cargará los componentes en función de la ruta actual.

```
<nav>
    <a routerLink="">Inicio</a>
    <a routerLink="acerca">Acerca</a>
</nav>

<!-- Aquí se renderizan los componentes de las rutas -->
<router-outlet></router-outlet>
```

3. **componentes**:

   La directiva `routerLink` se utiliza en los elementos HTML para establecer enlaces de navegación. En el ejemplo anterior, el usuario puede hacer clic en los enlaces para navegar a las rutas **Inicio** y **Acerca**.

**Explicación del código:**

- **`routes`**: Define las rutas de la aplicación. Cada ruta se define como un objeto con propiedades como:
  - **`path`**: Especifica la URL de la ruta.
  - **`component`**: Define el componente que se debe cargar cuando la ruta está activa.
- **`RouterModule.forRoot()`**: Inicializa el router con las rutas principales de la aplicación.
- **`<router-outlet>`**: Es el marcador de posición que Angular utiliza para renderizar el componente correspondiente a la ruta activa.
- **`routerLink`**: Se utiliza para vincular rutas en los elementos de navegación. En este caso, el usuario puede hacer clic en "Inicio" o "Acerca" para navegar entre esos componentes.

**Cargar un componente de forma predeterminada**

En la configuración de las rutas, podemos definir una ruta predeterminada utilizando una cadena vacía (`''`), que se cargará cuando el usuario acceda al dominio raíz de la aplicación:

```
{ path: '', redirectTo: '/inicio', pathMatch: 'full' },
```

Este fragmento de código redirige la ruta raíz (`/`) a la ruta `/inicio`.

**Manejo de rutas no encontradas (404)**

Es importante también manejar las rutas que no existan en la aplicación, mostrando una página de error 404.

```
{ path: '**', component: NoEncontradoComponent }
```

La ruta `**` actúa como un "comodín" y coincide con cualquier URL que no haya sido definida previamente, cargando el componente `NoEncontradoComponent`.

**Rutas anidadas (Rutas hijas)**

En aplicaciones más grandes, puedes definir rutas anidadas para componentes hijos, lo que permite una estructura jerárquica de navegación.

```
const routes: Routes = [
```

```
  { path: 'configuracion', component: ConfiguracionComponent, children: [
      { path: 'perfil', component: PerfilComponent },
      { path: 'ayuda', component: AyudaComponent }
    ]
  }
];
```

Por lo tanto, las rutas hijas se deben acceder anteponiendo la ruta padre como sigue:

```
<a [routerLink]="['/configuracion/perfil']">Perfil</a>
<a [routerLink]="['/configuracion/ayuda']">Ayuda</a>
```

**Detalles adicionales:**

1. **Query Parameters (Parámetros de consulta)**: Podemos pasar parámetros en la URL que pueden ser utilizados por el componente para modificar su comportamiento o mostrar datos específicos.

   ```
   <a [routerLink]="['/perfil']" [queryParams]="{id:1, nombre:'Juan',
   edad:25}">Perfil con múltiples parámetros</a>
   ```

   Esto genera una URL como `/perfil?id=1&nombre=Juan&edad=25`

2. **Recuperar el valor del parámetro suscribiéndose al parámetro:** Podemos recuperar el valor de parámetro suscribiéndonos el cambio de parámetros:

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

@Component({
  selector: 'app-perfil',
  templateUrl: './perfil.component.html',
})
export class PerfilComponent implements OnInit {
  id: number;
  nombre: string;
  edad: number;

  constructor(private route: ActivatedRoute) {}

  ngOnInit(): void {
    // Suscribirse a los parámetros de consulta
    this.route.queryParams.subscribe(params => {
      this.id = params['id'];
      this.nombre = params['nombre'];
      this.edad = params['edad'];
```

```
    });
  }
}
```

3. **Recuperar el valor del parámetro usando snapshot:** Podemos recuperar el valor de parámetro de otra forma, cuando sabemos que el valor de parámetro no va a cambiar una vez que se envía el valor. Tenemos que pasar el objeto ActivatedRoute al constructor de la clase para poderlo utilizar:

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

@Component({
  selector: 'app-perfil',
  templateUrl: './perfil.component.html',
})
export class PerfilComponent implements OnInit {
  id: number;
  nombre: string;
  edad: number;

  constructor(private route: ActivatedRoute) {}

  ngOnInit(): void {
    // Obtener múltiples parámetros con snapshot
    this.id = this.route.snapshot.queryParams['id'];
    this.nombre = this.route.snapshot.queryParams['nombre'];
    this.edad = this.route.snapshot.queryParams['edad'];
  }
}
```

**Ejemplo de Routing**

Vamos a poner en práctica los conceptos de routing en la aplicación de angular que hemos venido trabajando, vamos a reutilizar varios componentes para simplemente direccionarlos con el concepto de rutas.

**Archivo app.routes.ts:**

```
import { Routes } from '@angular/router';
import { ListadoProductosComponent } from './listado-usuarios/listado-
usuarios.component';
import { MostrarMensajeComponent } from './mostrar-mensaje/mostrar-mensaje.component';
import { PadreComponent } from './padre/padre.component';
```

```
import { HijoComponent } from './padre/hijo/hijo.component';
import { SaludarComponent } from './saludar/saludar.component';

export const routes: Routes = [
    { path: '', component: ListadoProductosComponent },
    { path: 'mostrar-mensaje', component: MostrarMensajeComponent },
    {
        path: 'configuracion', children: [
            { path: 'padre', component: PadreComponent },
            { path: 'hijo', component: HijoComponent },
        ],
    },
];
```

**Archivo app.component.html:**

```
<app-navegacion/>
<router-outlet/>
```

**Creación del componente de navegación:**

ng g c navegacion --skip-tests

**Código navegacion.component.html:**

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <div class="container-fluid">
        <!-- Marca del sitio -->
        <a class="navbar-brand" [routerLink]="['/']">Angular App</a>

        <!-- Botón de menú para dispositivos pequeños -->
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
            data-bs-target="#navbarNav"
            aria-controls="navbarNav" aria-expanded="false"
            aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>

        <!-- Opciones de la barra de navegación -->
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav me-auto mb-2 mb-lg-0">

                <!-- Opción Inicio -->
                <li class="nav-item">
```

```
                    <a class="nav-link" [routerLink]="['/']" routerLinkActive="active"
                        [routerLinkActiveOptions]="{ exact: true }"
                         aria-current="page">Inicio</a>
                </li>

                <!-- Opción Mensaje -->
                <li class="nav-item">
                    <a class="nav-link" [routerLink]="['/mostrar-mensaje']"
                    routerLinkActive="active"
                    [queryParams]="{mensaje:'Nuevo-mensaje'}">Mensaje</a>
                </li>

                <!-- Menú desplegable para Configuración -->
                <li class="nav-item dropdown">
                    <a class="nav-link dropdown-toggle" href="#"
                        id="navbarDropdown" role="button"
                        data-bs-toggle="dropdown" aria-expanded="false">
                        Opciones
                    </a>
                    <ul class="dropdown-menu" aria-labelledby="navbarDropdown">
                        <!-- Subopción Padre -->
                        <li><a class="dropdown-item"
                                [routerLink]="['/configuracion/padre']"
                                routerLinkActive="active">Padre</a></li>
                        <!-- Subopción Hijo -->
                        <li><a class="dropdown-item"
                                [routerLink]="['/configuracion/hijo']"
                                routerLinkActive="active">Hijo</a></li>
                    </ul>
                </li>
            </ul>
        </div>
    </div>
</nav>
```

**Código navegacion.component.ts:**

```
import { Component } from '@angular/core';
import { RouterModule } from '@angular/router';

@Component({
  selector: 'app-navegacion',
  standalone: true,
  imports: [RouterModule],
  templateUrl: './navegacion.component.html',
```

```
  styleUrl: './navegacion.component.css'
})
export class NavegacionComponent {

}
```

Es importante que se importe el módulo RouterModule para que funcione correctamente las rutas configuradas con routerLink.

Para que el menú de navegación funcione correctamente, es importante no solamente configurar Bootstrap, sino también agregar PopperJs. Vamos a agregarlo:

npm install @popperjs/core –save

Además se debe configurar el archivo angular.json, debe quedar como sigue:

```
            ...
            "styles": [
              "node_modules/bootstrap/dist/css/bootstrap.min.css",
              "src/styles.css"
            ],
            "scripts": [
              "node_modules/@popperjs/core/dist/umd/popper.min.js",
              "node_modules/bootstrap/dist/js/bootstrap.min.js"
            ]
            ...
```

Una vez que ya tenemos configurado el menú de navegación, ya podremos acceder a los componentes, solo hicimos algunos ajustes para mostrar títulos o centrar los elementos de los componentes que estamos reutilizando como sigue:

**Código listado-usuarios.component.html:**

```
<div class="container text-center">
    <h1 class="text-center text-warning my-4">Listado de Personas</h1>
</div>

<div class="row justify-content-center">
    <div class="col-md-6">
        <ul class="list-group">
            @if (usuarios.length > 0) {
            @for (usuario of usuarios; track usuario.id) {
            <li class="list-group-item">{{ usuario.id }} {{ usuario.name }}</li>
            }
            } @else {
            <li class="list-group-item text-danger">No hay usuarios disponibles.</li>
```

```
            }
        </ul>
    </div>
</div>
```

**Código mostrar-mensaje.component.ts:**

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

@Component({
  selector: 'app-mostrar-mensaje',
  standalone: true,
  imports: [],
  templateUrl: './mostrar-mensaje.component.html',
  styleUrl: './mostrar-mensaje.component.css'
})
export class MostrarMensajeComponent {
  mensaje: string = '';

  constructor(private route: ActivatedRoute) {}

  ngOnInit(): void {
    // Obtener el parámetro con snapshot
    this.mensaje = this.route.snapshot.queryParams['mensaje'];
  }
  mostrarMensaje(): void {
    this.mensaje = '¡Hola, has hecho clic en el botón!';
  }

  resetearMensaje(): void {
    // Reiniciamos el valor de la variable
    this.mensaje = '';
  }
}
```

Podemos observar que en este componente estamos procesando el parámetro enviado desde la ruta del componente de navegación.

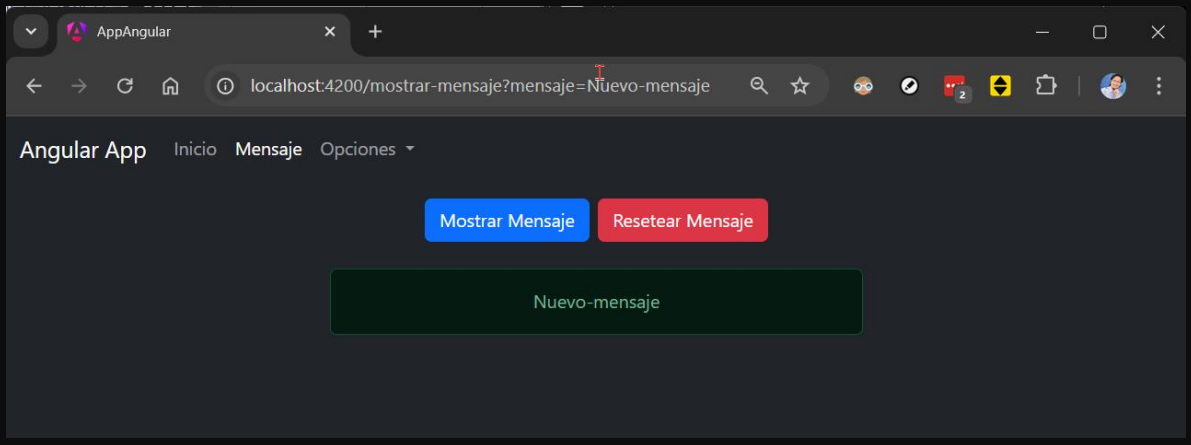
**Código padre.component.html.** Este componente no tiene ningún cambio.

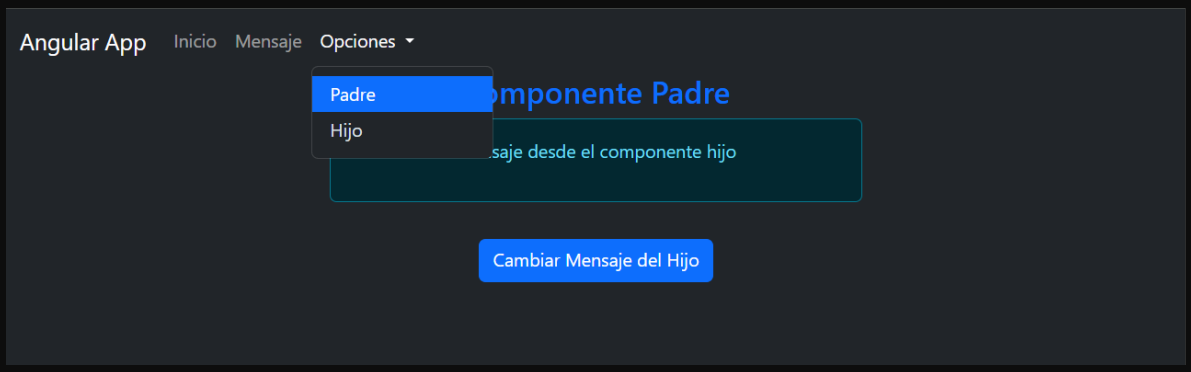
**Código hijo.component.html.**

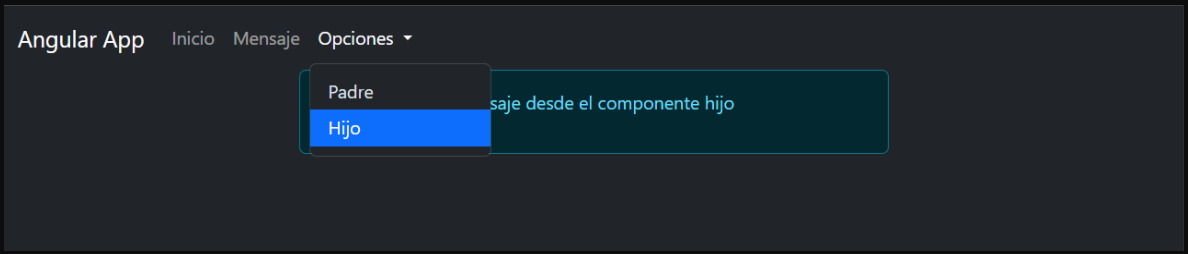
```
<div class="alert alert-info w-50 mx-auto text-center">
    <p>{{ mensaje }}</p>
```

```
</div>
```

**Resultado Final:**




## Conclusión

El **routing (ruteo) en Angular** es un concepto fundamental para desarrollar aplicaciones SPA eficientes, donde la navegación entre diferentes vistas no requiere recargar la página completa. Es posible definir rutas básicas, rutas hijas, rutas predeterminadas, rutas de error y realizar cargas diferidas para mejorar el rendimiento.

Saludos!

Ing. Ubaldo Acosta

Fundador de GlobalMentoring.com.mx