

# Rutas en Angular



## Uso de rutas en Angular

En esta guía, aprenderás cómo configurar y utilizar rutas en una aplicación Angular. Explicaremos el propósito de cada archivo y cambio realizado, incluyendo el uso de `RouterModule`, `routerLink`, y `router-outlet`.

---

### Paso 1: Definir las rutas en `app.routes.ts`

El primer paso en Angular para habilitar la navegación es definir las rutas dentro de un archivo de configuración.

#### Código de `app.routes.ts`

```
import { Routes } from '@angular/router';
import { ProductoListaComponent } from './producto-lista/producto-lista.component';

export const routes: Routes = [
  { path: 'productos', component: ProductoListaComponent },
  { path: '', redirectTo: 'productos', pathMatch: 'full' }
];
```

## Explicación del código:

1. Importamos **Routes** desde `@angular/router` porque es la estructura que Angular utiliza para definir rutas.
  2. Definimos un array de rutas (**routes**):
    - o `{ path: 'productos', component: ProductoListaComponent }:`  
Define que cuando el usuario navegue a `productos`, se renderizará `ProductoListaComponent`.
    - o `{ path: '', redirectTo: 'productos', pathMatch: 'full' }:`  
Redirige automáticamente la URL raíz (/) a `productos`, asegurando que la aplicación tenga una página de inicio.
- 

## Paso 2: Configurar el enrutamiento en `app.component.ts`

Para que Angular pueda reconocer y manejar las rutas definidas en `app.routes.ts`, necesitamos importar `RouterModule`.

### Código de `app.component.ts`

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

@Component({
  selector: 'app-root',
  imports: [RouterModule],
  templateUrl: './app.component.html'
})
export class AppComponent {
```

## Explicación del código:

1. Importamos `RouterModule`:
    - o Este módulo es necesario para que Angular reconozca y maneje las rutas.
    - o Sin esta importación, Angular no sabría cómo interpretar `<router-outlet>` y los enlaces con `routerLink`.
  2. Añadimos `RouterModule` en `imports` dentro del decorador `@Component`:
    - o Esto es obligatorio en aplicaciones **standalone** para que las rutas funcionen correctamente.
    - o Si no se importara, Angular mostraría errores relacionados con `router-outlet` y `routerLink`.
-

## Paso 3: Modificación de `app.component.html`

### Código de `app.component.html`

```
<div class="container">
  <nav class="navbar navbar-expand-lg navbar-dark"
  style="background-color: #8c0aeef;">
    <div class="container-fluid">
      <a class="navbar-brand"
      routerLink="productos">Sistema de Inventarios</a>
      <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>
      <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav">
          <li class="nav-item">
            <a class="nav-link active" aria-current="page"
            routerLink="productos"
            routerLinkActive="active">Inicio</a>
          </li>
          <li class="nav-item">
            <a class="nav-link"
            href="/agregar">Agregar Producto</a>
          </li>
        </ul>
      </div>
    </div>
  </nav>
</div>

<div class="container">
  <router-outlet></router-outlet>
</div>
```

### Explicación de los cambios y conceptos clave:

#### 1. Uso de `routerLink` en lugar de `href`

```
<a class="nav-link" routerLink="productos"
routerLinkActive="active">Inicio</a>
```

- ¿Por qué usar `routerLink` en lugar de `href`?

- `href` provoca que el navegador recargue completamente la página, lo que interrumpe la experiencia de usuario y pierde el estado de la aplicación.
  - `routerLink` permite la navegación **sin recargar** y mantiene el estado de la aplicación.
  - `routerLinkActive="active"` resalta el enlace cuando la ruta está activa.
- 

## 2. Uso de `<router-outlet>`

```
<div class="container">
  <router-outlet></router-outlet>
</div>
```

- **¿Qué hace `<router-outlet>`?**
    - Es un marcador de posición donde Angular renderiza dinámicamente los componentes basados en la ruta actual.
    - Cuando el usuario navega a `productos`, el componente `ProductoListaComponent` se renderiza automáticamente en este espacio.
    - `router-outlet` **se relaciona directamente con `app.routes.ts`**, ya que usa las rutas definidas en ese archivo para determinar qué componente debe mostrarse.
- 

## Paso 4: Integrar las rutas en `app.config.ts`

Para que las rutas funcionen, es necesario proporcionar la configuración de rutas en el archivo `app.config.ts`. Esta configuración ya viene por default, esta debe ser la configuración:

### Código de `app.config.ts`

```
import { ApplicationConfig, provideZoneChangeDetection } from '@angular/core';
import { provideRouter } from '@angular/router';

import { routes } from './app.routes';
import { provideHttpClient } from '@angular/common/http';

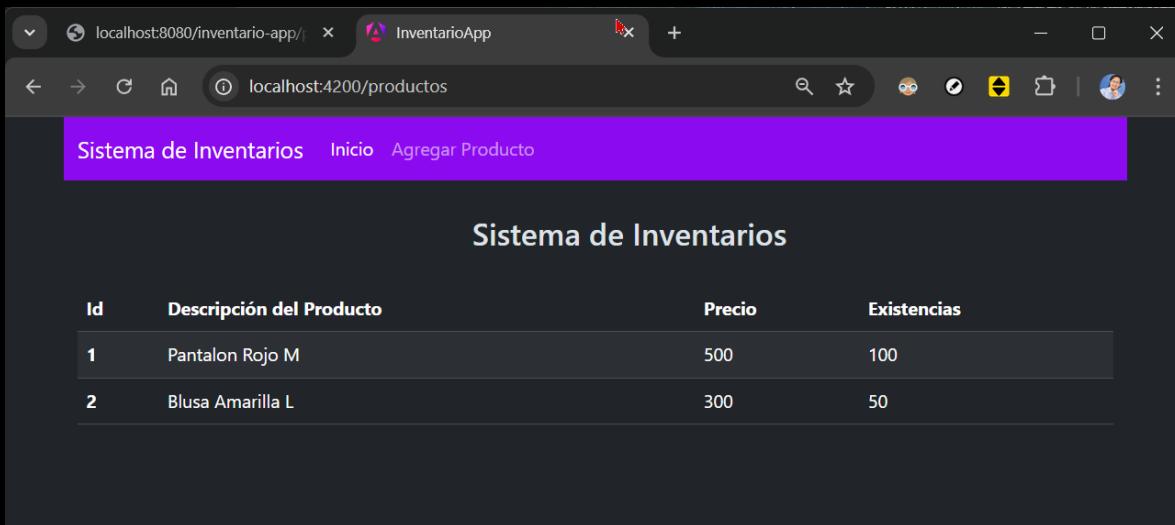
export const appConfig: ApplicationConfig = {
  providers: [
    provideZoneChangeDetection({ eventCoalescing: true }),
    provideRouter(routes),
    provideHttpClient()
  ],
};
```

## Explicación del código:

1. Importamos `provideRouter` de `@angular/router` para habilitar el enrutamiento.
  2. Pasamos `routes` a `provideRouter(routes)`, conectando la configuración de rutas a la aplicación, realizando esta configuración en los proveedores de servicio de nuestra aplicación.
- 

## Ejecutamos la aplicación

- Ejecutamos la aplicación con el comando: `ng serve -o`
- Entramos a la URL: <http://localhost:4200/>



## Conclusión

Con estos cambios, ahora la aplicación Angular tiene un sistema de enrutamiento funcional y optimizado:

- `app.routes.ts` define las rutas y redirecciones.
- `RouterModule` se importa en `app.component.ts` para habilitar las rutas en la aplicación standalone.
- `routerLink` se usa en lugar de `href` para evitar recargas innecesarias.
- `router-outlet` se usa para renderizar dinámicamente los componentes según la ruta actual.
- Las rutas se configuran en `app.config.ts` usando `provideRouter(routes)`.

Este enfoque mejora la eficiencia y la experiencia del usuario en la navegación dentro de la aplicación. 

Saludos!

**Ing. Ubaldo Acosta**

Fundador de [GlobalMentoring.com.mx](#)