Cómo usar RouterLink :id con Angular v19

Por: **Ángel Ilai Alcañiz García**

[routerLink]

Índice

Establecer las rutas en app.routes.ts

Una vez creados los componentes que vayamos a utilizar, los configuraremos en app.routes.ts para utilizar el id.

En este proyecto hemos creado un menú de un restaurante con tres categorías: **arroces**, **carnes** y **postres**. Primero, hemos creado un componente para cada una de estas categorías, como siempre.

Por otro lado, hemos creado un componente para cada tipo de plato; en este caso: **arroz**, **carne** y **postre**. Estos últimos deben configurar su ruta agregando "**/:id**" en el nombre de la ruta. De esta forma, establecemos que **la última sección de la URL será dinámica** y le daremos el nombre de "id" (ya que la haremos coincidir con el atributo id de cada plato, como veremos posteriormente), y así es como accederemos a ella.

Debería verse así:

```
import { Routes } from '@angular/router';  207.4k (gzipped: 47.2k)

import { ArrocesComponent } from './components/arroces/arroces.component';
import { ArrozComponent } from './components/arroces/arroz/arroz.component';
import { CarnesComponent } from './components/carnes/carnes.component';
import { CarneComponent } from './components/carnes/carne/carne.component';
import { PostresComponent } from './components/postres/postres.component';
import { PostreComponent } from './components/postres/postre/postre.component';

export const routes: Routes = [
    { path: '', redirectTo: 'arroces', pathMatch: 'full' },
    { path: 'arroces', component: ArrocesComponent },
    { path: 'arroz/:id', component: ArrozComponent},
    { path: 'carnes', component: CarnesComponent},
    { path: 'carne/:id', component: CarneComponent},
    { path: 'postres', component: PostresComponent},
    { path: 'postre/:id', component: PostreComponent}
];
```

Crear un servicio para acceder a los productos

Crearemos un servicio donde tendremos uno o varios **arrays[]** con los platos, así como **métodos** (getters) para **acceder a estos platos**, uno de ellos que **filtrará por id**. Este último lo utilizaremos para acceder a los datos de un plato concreto en las páginas de producto; en nuestro caso, en los componentes arroz, carne y postre.

Añadir los platos

En este caso, hemos creado **tres arrays** con la misma estructura. Cada uno contiene seis platos, y cada plato tiene el atributo "**id**", "nombre", etc., como se ve en la imagen. Los ID se han creado a modo de prueba; los arroces empiezan por ARR01, las carnes por CAR01 y los postres por POS01. En el caso de las carnes, en lugar de "alérgenos" hay un atributo "animal", que indica el tipo de animal de la carne del plato.

Así se ve la primera parte del servicio **platos.service.ts**:

src > app > services > platos.service.ts > PlatosService

```
1   import { Injectable } from '@angular/core';  397.2k (gzipped: 117.3k)
2
3   @Injectable({
4     providedIn: 'root'
5   })
6   export class PlatosService {
7
8     constructor() { }
9
10    // Platos
11  > arroces = […
60    ];
61
62  > carnes = […
111   ];
112
113   postres = [
114     {
115       id: 'POS01',
116       nombre: 'Muerte por chocolate',
117       foto: 'muerte_chocolate',
118       descripcion: 'Tarta 100% sin gluten con 3 capas de bizcocho montadas en crema chocolate y trufa y cubierta con virutas
119       alergenos: ['Huevo', 'Lacteos'],
120       precio: 5.20
121     },
122     {
123       id: 'POS02',
124       nombre: 'Tarta de Oreo',
125       foto: 'tarta_oreo',
126       descripcion: 'Esta elegante tarta sin gluten te sorprenderá por su combinación de chocolates con nata y galletas. Llev
127       alergenos: ['Huevo'],
128       precio: 5.20
```

Añadir los *getters* para los platos y filtrar por id

Por un lado, creamos **un getter para cada array** de platos. Estos getters serán utilizados en los componentes que mostrarán *todos los platos* de una determinada categoría; en este caso, “arroces”, “carnes” y “postres”.

Por otro lado, creamos un **método que filtrará por id**. Este método será utilizado en los componentes específicos de cada producto (plato); en este caso, “arroz”, “carne” y “postre”. Al llamar a este método, deberemos indicarle **el id** del plato, que **obtendremos de la URL** (como veremos posteriormente cuando configuremos estos componentes), así como **la categoría** del plato (arroces, carnes o postres) para saber el array en el que debe buscar el plato.

Así se ven los métodos mencionados en la última parte de **platos.service.ts**:

```
// Getters
getArroces() {
  return this.arroces;
}

getCarnes() {
  return this.carnes;
}

getPostres() {
  return this.postres;
}

getPlatoById(id: string, categoria: string) {
  if(categoria === 'arroces') return this.arroces.find((plato) => plato.id === id);
  else if(categoria === 'carnes') return this.carnes.find((plato) => plato.id === id);
  else if(categoria === 'postres') return this.postres.find((plato) => plato.id === id);
  else return;
}
```

Configurar los componentes que listarán los productos (platos)

En este proyecto tendremos tres componentes que listarán los platos de cada uno de los arrays que tenemos en el servicio, respectivamente. Estos son "**arroces**", "**carnes**" y "**postres**"; al ser muy similares entre ellos, mostraremos cómo configurar uno de ellos, ya que el proceso es el mismo para todos. Por ejemplo, vamos a ver el componente "**arroces**".

Imports

Comenzamos con el **.ts: arroces.component.ts**. Deberemos importar:

- **CommonModule**, para poder utilizar *ngFor.
- **RouterLink,** para redirigir al componente "arroz", que además haremos con el id.
- **PlatosService**, para acceder al array de arroces.

```
1  import { CommonModule } from '@angular/common';  94.3k (gzipped: 26k)
2  import { Component } from '@angular/core';  397.2k (gzipped: 117.3k)
3  import { RouterLink } from '@angular/router';  207.4k (gzipped: 47.3k)
4  import { PlatosService } from '../../services/platos.service';
5
6  @Component({
7    selector: 'app-arroces',
8    imports: [RouterLink, CommonModule],
9    templateUrl: './arroces.component.html',
10   styleUrl: './arroces.component.css'
11 })
```

Acceder a los platos del servicio

Comenzaremos **creando un array vacío** llamado platos, al que le diremos que puede recibir cualquier tipo de dato (any). Seguidamente, estableceremos un **constructor** que recibirá una instancia del servicio PlatosService. Por último, utilizaremos **ngOnInit** para que en el momento en el que se cargue la página, el array "platos" que hemos creado **reciba los datos del método getArroces()** que hemos creado en el servicio.

```
export class ArrocesComponent {

  platos: any[] = [];

  constructor(private platosService: PlatosService) {}

  ngOnInit() {
    this.platos = this.platosService.getArroces();
  }
}
```

Dar formato al HTML

Ahora, vamos a establecer cómo queremos que se muestre ese componente. Lo importante en este paso es:

- Utilizar ***ngFor para repetir un bloque por cada plato**.
- Acceder al atributo "**id**" de cada plato y **añadirlo al final de la ruta**, después del nombre del componente (arroz), en el **routerLink**.
- Acceder a los atributos que queramos mostrar de cada plato utilizando las llaves de la siguiente forma: **{{ plato.atributo }}**.

Podría verse de una forma similar a la siguiente:

```
<div class="productos">
    <h1>🍚 Arroces</h1>

    <section class="products-grid">

        <article *ngFor="let plato of platos">
            <a class="plato-container" [routerLink]="['../arroz', plato.id]">
                <div class="img-container">
                    <img src="platos/{{plato.foto}}.jpg" alt="{{ plato.nombre }}">
                </div>

                <div class="info-container">
                    <h2 class="nombre">{{ plato.nombre }}</h2>
                    <div class="alergenos">
                        <div class="alergeno-container" *ngFor="let alergeno of plato.alergenos">
                            <img src="alergenos/{{alergeno}}.png" >
                        </div>
                    </div>
                    <p class="precio">{{ plato.precio }}€</p>
                </div>
            </a>
        </article>

    </section>
</div>
```

Nota: en el [routerLink], podemos ver que no redirigimos directamente a 'arroz', sino que retrocedemos un directorio en la ruta utilizando '../arroz'. Esto es porque, como la página arroces es en sí un componente al que estamos accediendo mediante un routerLink desde el header, nos encontramos en la ruta '/arroces'. Al usar el routerLink desde esta ubicación, trata de desplazarse a '/arroces/arroz/id'', dirección que no existe. Debemos retroceder a '/' antes para poder acceder a '/arroz/id', y es por esto que usamos los '..'.

El componente que acabamos de crear, una vez añadidos los estilos en nuestro css, se ve de la siguiente manera:

Tenemos un ‘listado’ de los seis arroces que hemos creado en el array en el servicio. De cada uno de ellos mostramos su nombre, el listado de alérgenos, el precio y una imagen.

Al hacer hover sobre uno de ellos, vemos que **nos está dirigiendo a ‘/arroz/ARR03’**, siendo ‘ARR03’ el id que le hemos establecido a este plato. De momento, esta página no existe, por lo que no funcionará. La crearemos en el siguiente paso.

En el caso del componente ‘carnes’ hay una sutil diferencia, pues en lugar de mostrar los alérgenos, muestra el animal del que es la carne de cada plato. El resultado visual es el siguiente, si bien **en cuanto a la funcionalidad funciona exactamente igual**:

Configurar las páginas de producto (plato)

Ahora, vamos a configurar los **componentes para los productos**, es decir: “arroz”, “carne” y “postre”. Igual que anteriormente, voy a mostrar el proceso solo de una de ellas, ya que todas ellas están configuradas de la misma forma. Por ejemplo, seguiremos con “arroz”.

Imports

De nuevo, comenzamos con el **.ts: arroz.component.ts**. Deberemos importar:

- **CommonModule**, para poder utilizar *ngFor.
- **ActivatedRoute**, para acceder al id del plato desde la URL.
- **PlatosService**, para acceder a los datos del plato mediante el método getPlatoById().

```
import { Component } from '@angular/core';  397.2k (gzipped: 117.3k)
import { ActivatedRoute } from '@angular/router';  207.4k (gzipped: 47.3k)
import { PlatosService } from '../../../services/platos.service';
import { CommonModule } from '@angular/common';  94.3k (gzipped: 26k)

@Component({
  selector: 'app-arroz',
  imports: [CommonModule],
  templateUrl: './arroz.component.html',
  styleUrl: './arroz.component.css'
})
```

Acceder a los datos del plato

Comenzaremos **creando un objeto vacío** llamado plato. Seguidamente, estableceremos un **constructor** que recibirá una instancia de ActivatedRoute y una del servicio PlatosService. Por último, utilizaremos **ngOnInit** para que cuando se cargue la página se obtenga **el id del plato desde la URL** mediante un método de ActivatedRoute, y el objeto “plato” **reciba los datos del método getPlatoById()** pasándole el id que hemos obtenido y la categoría del plato; en este caso, “arroces”.

```
export class ArrozComponent {
  plato: any;

  constructor(private route: ActivatedRoute, private platosService: PlatosService) {}

  ngOnInit() {
    const id = this.route.snapshot.paramMap.get('id');
    this.plato = this.platosService.getPlatoById(id!, 'arroces');
  }
}
```

Dar formato al HTML

Por último, estableceremos cómo se va a mostrar este componente. Como siempre, accederemos a los atributos del plato utilizando las llaves **{{ plato.atributo }}**.

Podría verse de esta forma:

```html
<div class="producto">
    <div class="producto-container">

        <div class="img-container">
            <img src="platos/{{plato.foto}}.jpg" alt="{{ plato.nombre }}">
        </div>

        <div class="info-container">

            <h1 class="nombre">{{ plato.nombre }}</h1>

            <div class="alergenos">
                <div class="alergeno-container" *ngFor="let alergeno of plato.alergenos">
                    <img src="alergenos/{{alergeno}}.png" >
                </div>
            </div>

            <div class="descripcion-container">
                <h2 class="descripcion-titulo">Descripción: </h2>
                <p class="descripcion">{{ plato.descripcion }}</p>
            </div>

            <p class="precio">{{ plato.precio }}€</p>

        </div>

    </div>
</div>
```

Una vez añadidos los estilos en nuestro css, se ve de la siguiente manera:

Links relevantes

Web desplegada

https://la-mesa-limpia.vercel.app/

Repositorio de GitHub

https://github.com/softdevilan/la-mesa-limpia

Angular

https://v17.angular.io/tutorial/tour-of-heroes/toh-pt5
https://angular.dev/api/router/ActivatedRoute