Angular S6: Routes

Después de esta lección podrás:

1. Entender el concepto SPA
2. Construir las rutas de tu aplicación
3. Comprender cómo manejar parámetros en las rutas

Llega la hora de manejar diferentes rutas en nuestra aplicación, ya que hasta ahora, no hemos navegado y siempre hemos trabajado en la pantalla principal. En esta lección explicaremos en qué consiste una SPA y aprenderemos a enrutar nuestros módulos y componentes.

Angular SPA

¿Qué es SPA? *Single Page Application*, consiste en una aplicación web de página única. Esto quiere decir que toda nuestra aplicación crecerá bajo la misma página, navegando por diferentes rutas. La principal ventaja es que las transiciones serán

más fluidas, no existe recarga completa de la vista y por tanto tendremos el control de la navegación en nuestra aplicación Angular.

Angular Routing

Vamos a aprender a configurar varias rutas en nuestra aplicación, para ello, generemos un proyecto de cero con el CLI, asegurándonos de fijar enrutado.

```
ng new router-app
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? SCSS
```

Si nos fijamos en el **app.module.ts**, vemos que existe ya un módulo autogenerado de enrutado:

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule // Este sería el módulo de enrutamiento
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Vamos a enriquecerlo un poco, creando 3 rutas en nuestra aplicación:

- la ruta base, que redireccionará al listado de usuarios

- la ruta `user-list`, que mostrará usuarios
- la ruta `user-list/:userId` que mostrará el detalle de un usuario

Pero antes de seguir, necesitamos los componentes que se verán en cada ruta:

```
ng generate component user-list
ng generate component user-detail
```

Y en este punto, siempre nos vamos al **app.component.html** a limpiar el contenido y renderizar los nuevos componentes generados:

```
<!-- <app-user-list></app-user-list> -->
<!-- <app-user-detail></app-user-detail> -->

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

Pero esta vez, nos vamos a fijar en la última línea, dejaremos el componente `router-outlet`.

Este elemento será el que nos permitirá enlazar la vista de nuestra SPA, con las rutas definidas en **app-routing.module.ts**:

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { UserListComponent } from './user-list/user-list.component';
import { UserDetailComponent } from './user-detail/user-detail.component';

// Aqui definimos las rutas de nuestra SPA
const routes: Routes = [
  { path: '', redirectTo: 'user-list', pathMatch: 'full' },
  { path: 'user-list', component: UserListComponent },
  { path: 'user-list/:userId', component: UserDetailComponent },
];

@NgModule({
```

```
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Ahora, nuestra URL por defecto de nuestra aplicación debería ser http://localhost:4200/user-list

Solo nos queda probar la navegación, para ello vamos a crear 3 usuarios en el listado de **user-list.component.html**:

```
<p>user-list works!</p>

<div>
  <a [routerLink]="['/user-list/1']">Usuario 1</a>
</div>
<div>
  <a [routerLink]="['/user-list/2']">Usuario 2</a>
</div>
<div>
  <a [routerLink]="['/user-list/3']">Usuario 3</a>
</div>
```

Y aquí es donde está la magia, mediante el atributo `routerLink` establecemos la URL a la que queremos navegar. Por ahora hemos fijado directamente las rutas en el HTML, pero a continuación aprenderemos a hacerlas dinámicas, leyendo parámetros en rutas.

Parámetros en la ruta

Ahora que ya hemos visto como navegar por nuestra aplicación, vamos a aprender a pasar parámetros en la URL. Esto nos va a permitir crear URLs específicas para acceder mediante la SPA a la pantalla que queremos cargar.

Vamos a complicar el anterior ejemplo y crearnos un listado de usuarios a nivel del **user-list.component.ts**:

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

interface User {
  id: number;
  name: string;
}

@Component({
  selector: 'app-user-list',
  templateUrl: './user-list.component.html',
  styleUrls: ['./user-list.component.scss']
})
export class UserListComponent implements OnInit {
  userList: User[];

  constructor() { }

  ngOnInit() {
    this.userList = [
      {
        id: 1,
        name: 'Jose',
      },
      {
        id: 2,
        name: 'Pedro',
      },
      {
        id: 3,
        name: 'Laura',
      }
    ];
  }

}
```

```
<p>user-list works!</p>

<ul>
  <li *ngFor="let user of userList">
    <span>{{user.name}}</span>
    <span> - </span>
    <a [routerLink]="['/user-list', user.id]">Ver usuario</a>
  </li>
</ul>
```

Cambiamos el template para pintar el listado, y enriquecemos el atributo `routerLink` con otro valor en el array, el **id** del usuario. ¡Ya tenemos rutas específicas por usuario!

Ahora en el detalle de cada usuario, vamos a recoger de la URL el id, para saber qué usuario se debe cargar en el detalle.

Modifiquemos el **user-detail.component.ts** de la siguiente manera:

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

@Component({
  selector: 'app-user-detail',
  templateUrl: './user-detail.component.html',
  styleUrls: ['./user-detail.component.scss']
})
export class UserDetailComponent implements OnInit {
  userId: string;

  constructor(private route: ActivatedRoute) { }

  ngOnInit() {
    this.route.paramMap.subscribe(params => {
      this.userId = params.get('userId');
    });
  }
}
```

Hemos inyectado el `ActivatedRoute` del router de Angular, que nos va a dar información sobre la ruta cargada y la URL en nuestra SPA. Como ya sabemos de observables, nos suscribimos a la información que llegue sobre los parámetros, de cara a recoger el **userId**.

De esta manera en nuestro template, podemos visualizar el **id** del usuario:

```
<p>user-detail works!</p>

<p>Ficha de Usuario con id: {{ userId }}</p>
```

Finalmente tenemos ya todas las herramientas para construir una aplicación con varias rutas, que sea consistente. Es decir, si este ejemplo lo combinamos con lo aprendido en la lección de servicios, podemos construir dos estados de una SPA (listado y detalle) que consulten los mismos datos a través de servicios de datos y peticiones HTTP.