CURSO DE PROGRAMACIÓN JAVA FULLSTACK

Angular

5. Ejercicio

Arturo Bernal Mayordomo

Edición: Noviembre 2019

©: Quedan reservados todos los derechos. (Ley de Propiedad Intelectual del 17 de noviembre de 1987 y Reales Decretos).
Documentación elaborada por la EOI.
Prohibida la reproducción total o parcial sin autorización escrita de la EOI.

Index

1. - Ejercicio

Esta semana vamos a dividir la aplicación en páginas. Para ello tendremos que hacer algunas pequeñas modificaciones a la aplicación existente.

1.1 *Rutas de la aplicación*

Añadiremos las siguientes rutas a la aplicación:

- **'eventos'** → Listado de eventos (EventsShowComponent)
- **'eventos/add'** → Añadir evento (EventAddComponent)
- **'eventos/:id'** → Detalle de un evento (EventDetailComponent). Este componente es nuevo y lo explicaremos más adelante.
- Por defecto, se redirigirá al listado de eventos.

Además, pondremos los enlaces para navegar al listado de eventos y al formulario de añadir evento en la barra superior de navegación:

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
 <a class="navbar-brand" href="#">Angular Events</a>
  <ul class="navbar-nav mr-auto">
  <li class="nav-item">
   <a class="nav-link" ... >Eventos</a>
  </li>
  <li class="nav-item">
   <a class="nav-link" ... >Añadir evento</a>
  </li>
 </ul>
</nav>
```

En el enlace del listado de eventos, se recomienda incluir también el atributo **[routerLinkActiveOptions]="{exact: true}"**, para que no aparezca resaltado cuando navegamos a “Añadir evento”, ya que si no, comprueba que la ruta “eventos/add” contiene “eventos” y la resalta también.

1.2 *Página: Mostrar eventos*

A la página de mostrar eventos le podemos quitar el componente del formulario de añadir evento (ahora estará en una página aparte), así como el método de añadir un evento al array (siempre los cargará del servidor directamente).

Consejo: Los enlaces de ordenar por precio y fecha hacen scroll en la página (culpa de href=”#)”. Lo ideal sería que no fueran enlaces, pero entonces tendríamos que poner CSS para que lo parecieran (tal vez la mejor opción). Otra manera de no recargar la página es cancelando el evento por defecto:

```
<li class="nav-item">
    <a class="nav-link" href="#" (click)="orderDate($event)">Ordenar por fecha</a>
 </li>
```

```
 orderDate(event: Event) { // Recibe evento click de HTML
  event.preventDefault(); // Cancelamos comportamiento por defecto
  ... // Ordenamos array
 }
```

1.3 *Página: Detalle de un evento*

Navegaremos a esta página cuando hagamos clic en la foto o el título de un evento. Esta página recibirá el evento a mostrar obtenido gracias a un Resolve que se explica más adelante. Para mostrar el evento, reutilizaremos el componente **event-item** (sólo hay que ponerle un poco de margen arriba). Cuando el evento se borre desde aquí nos limitaremos a volver al listado de eventos y veremos como ya no aparece:

```
<div class="mt-4">
 <event-item class="card" [event]="event" (deleted)="goBack()"></event-item>
</div>
```

Necesitaremos implementar el método en **EventosService** que nos devuelva la información de un evento a partir de su id. La ruta del servidor será '/eventos/:id'. Es decir, si la id es 6, llamaríamos a '/eventos/6'. El servidor nos devolverá una respuesta con el evento **{evento: Evento}**.

```
getEvento(id: number): Observable<Evento> {
  ...
}
```

1.4 *Página: Añadir evento*

Esta página contendrá el formulario de añadir evento. Este formulario ya no necesita emitir ningún evento, simplemente redirigirá a la página de listado de eventos.

1.5 *Guards y resolvers*

Vamos a añadir los siguientes *guards* a la aplicación:

- **SaveChangesGuard** → Es de tipo CanDeactivate y preguntará si queremos abandonar la página (con un confirm). Se lo aplicaremos a la página del formulario de añadir evento (como en el ejemplo de los apuntes).
- **EventoDetailResolve** → Es de tipo Resolve. Obtendrá el evento cuya id haya sido pasada por parámetro (como en el ejemplo de los apuntes). Hay que aplicárselo a la ruta de detalle de un evento.