Semana 5

Curso de Angular

Ejercicio semana 5

Client-side Web Development
2nd course – DAW
IES San Vicente 2017/2018
Author: Arturo Bernal Mayordomo

Index

Ejercicio

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

Rutas de la aplicación

Añadiremos las siguientes rutas a la aplicación:

- **'events'** → Listado de eventos (EventsShowComponent)
- **'events/add'** → Añadir evento (EventAddComponent)
- **'events/: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 “events/add” contiene “events” y la resalta también.

Obtener evento (Servicio web)

Para la página de detalle de un evento, necesitaremos implementar el método en **EventsService** que nos devuelva la información de un evento a partir de su id. La ruta del servidor será ‘/events/{id}’. Es decir, si la id es 6, llamaríamos a ‘/events/6’. El servidor nos devolverá una respuesta con un booleano (**ok**) indicando si todo ha ido bien y el evento (**event**) o un error (**error**). Reutiliza la interfaz **EventResponse** que creamos en la actividad anterior.

```
getEvent(id: number): Observable<IEvent> {
  ...
}
```

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).

Importante: Los enlaces de ordenar por precio y fecha ahora recargarán la página. 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)">Orden por fecha</a>
</li>

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

Recargar los eventos del servidor cada vez que se visite esta página (por ejemplo después de añadir un evento) no es lo más eficiente. Guardar el array de eventos en el servicio EventsService y obtenerlo de ahí las siguientes veces sería más eficiente, aunque más complejo de gestionar ya que habría que comprobar cada cierto tiempo si hay eventos nuevos en el servidor, etc.

Para mantener datos del servidor en un único punto de la aplicación y no tener que recargarlos cada vez que visitamos una ruta, se recomienda usar la librería **@ngrx**. Esta librería son “extensiones reactivas para Angular”, y contiene entre otras cosas la versión de Redux para Angular. Estas extensiones están pensadas para aplicaciones grandes ya que añaden complejidad a la lógica de la aplicación, y se utilizan también en aplicaciones (complejas) hechas con React. Son conceptos avanzados y requieren de cierto dominio trabajando con Observables, por lo que escapan a los contenidos del curso actual.

https://malcoded.com/posts/angular-ngrx-guide

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 (ya no estará):

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

Página: Añadir evento

Esta página contendrá el formulario de añadir evento. Este formulario ya no necesita emitir ningún evento, sino que lo guardará en el servidor y redirigirá a la página de listado de eventos.

Guards

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).

- **EventDetailResolve** → 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.