CURSO DE PROGRAMACIÓN JAVA FULLSTACK

Angular

3. Ejercicio

Arturo Bernal Mayordomo

Edición: Noviembre 2019

1.1 *Componente evento-item*

Crea un componente llamado **evento-item**. Este componente representará un evento por lo que recibirá los datos del evento como atributo de entrada (**@Input**). La plantilla de este nuevo componente contendrá desde el elemento (incluido). Para no romper la estructura de filas y columnas, las clases "col-6 mt-4" deberías situarlas en el nuevo elemento (evento-item). De esta manera serían elementos hijos de .

Además, vamos a poner un botón para borrar los eventos. Este botón se situará justo después del párrafo de la descripción, así:

```
<button class="btn btn-danger ml-3 float-right" (click)="deleteEvento()">Delete</button>
```

Cuando hagamos clic en el botón, el componente emitirá una señal al padre (puede ser sin valor → EventEmitter<void>), que le indicará que debe eliminar el evento del array.

Sobre borrar un evento cuando se está filtrando por nombre

Por defecto en Angular, por razones de rendimiento, los filtros (pipes) se comportan como filtros puros (pure pipes). Esto quiere decir, que sólo actualizan los datos mostrados (ejecutan el filtro otra vez) si se cambia la referencia al objeto (o array) que se está filtrando, en lugar de si simplemente modificamos dicho objeto internamente.

Esto sucede porque es mucho más simple (en términos de rendimiento) comprobar periódicamente si una referencia a memoria (número) ha cambiado, que analizar en profundidad dicho objeto o array buscando cambios. Por ello, si borramos un elemento del array con la función splice(), que modifica el array original, y estamos filtrando los eventos por nombres, parecerá que no se ha borrado (la vista no se actualiza porque el filtro no se vuelve a ejecutar).

Tenemos varias soluciones pero todas implican generar un array nuevo. Una de ellas es borrar con el método filter (que devuelve un array nuevo) y reasignar el array, y otra es aún usando splice, crear después una copia del array y reasignarla (peor solución que la anterior). **eventos = [...eventos]**. Para ello hemos usado el operador de propagación o spread.

Otra solución (a costa de perder rendimiento), es declarar el filtro como impuro, establececiendo la propiedad **pure** a false en el decorador **@Pipe** como se puede observar en el siguiente enlace:

https://angular.io/guide/pipes#pure-and-impure-pipes

1.2 *Componente evento-add*

Crea un componente llamado **evento-add** y sitúa ahí el formulario y toda la lógica del mismo (transformar la imagen a base64, etc.), además del HTML del formulario, claro está.

Este componente emitirá el objeto del evento (IEvent) cuando enviemos el formulario. El componente padre (**eventos-show**) lo recibirá ($event) y lo añadirá al array de eventos.

1.3 *Crear un servicio*

Crea un servicio llamado **eventos** (EventosService), y dentro del mismo, un método que te devuelva los eventos iniciales, en lugar de tenerlos en el componente **eventos-list** directamente. Desde este componente accede al servicio para obtener dichos eventos.