

Tema 4 - Angular



Ejercicio 3 – Angular Eventos

Desarrollo web en entorno cliente
IES Pere Maria Orts I Bosch



Ejercicio

Importante:

Antes de realizar esta práctica debes tener hecha la práctica guiada de angular-products de los apuntes “05_Componentes anidados. Servicios”, para ello puedes basarte en el código “04_angular-products” que se encuentra en “00_ejemplos” que es la base para realizar dicha práctica guiada. La práctica angular-products no es necesario entregarla.

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 el interior el elemento <div class="card"> (no incluido). La clase card poncela directamente al elemento <evento-item>.

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

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

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.

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 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) (**es la mejor solución y es la que debéis implementar**) y reasignar el array, y otra es aún usando splice, crear después una copia del array y reasignarla (peor solución que la anterior, no se recomienda). `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, estableciendo 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>

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.

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.