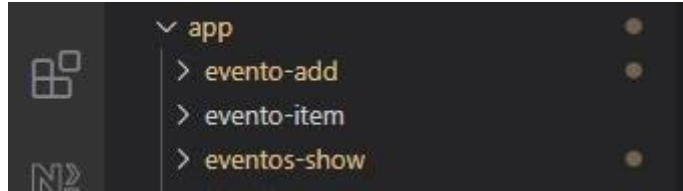


Inma Serrano:

1. Componentes y componentes anidados:

Para mí, los componentes son las diferentes partes/elementos que podemos crear y usar en nuestra aplicación. Ejemplo:

En la práctica de eventos, tenemos el componente add-eventos(sirve para añadir los eventos), evento-item(es el evento en sí) y eventos-shows, que muestra todo el conjunto.



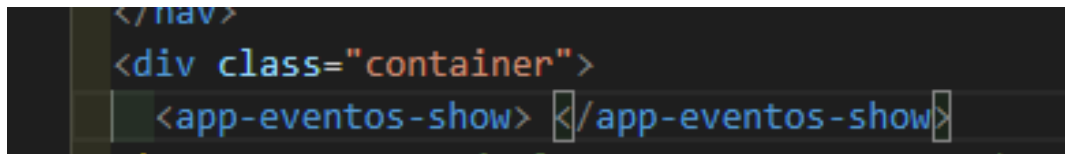
Como podemos ver, estos componentes tienen que relacionarse los unos con los otros. Para ello tenemos los inputs y los outputs. De esta manera el evento add le pasa los datos a los eventos:

```
newEvento: IEvento;  
nombreArchivo: string;  
@Output() creandoEventos = new EventEmitter<IEvento>();
```

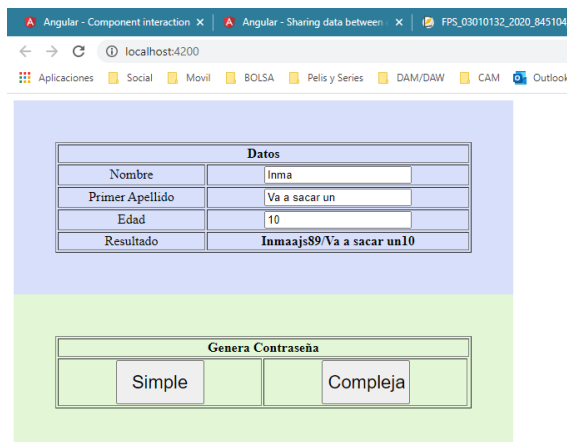
O el input que le pasa que hay eventos al hijo.

2. Plantillas:

Las plantillas son generalmente las vistas. En ellas vamos a ir mostrando los diferentes componentes. En el ejemplo anterior tenemos un componente padre que es app-component y en su vista (html) mostramos al siguiente como ponente que es show-component:



Y así en las siguientes. Para ir mostrando la información que llegará a ver el usuario: como en este examen:

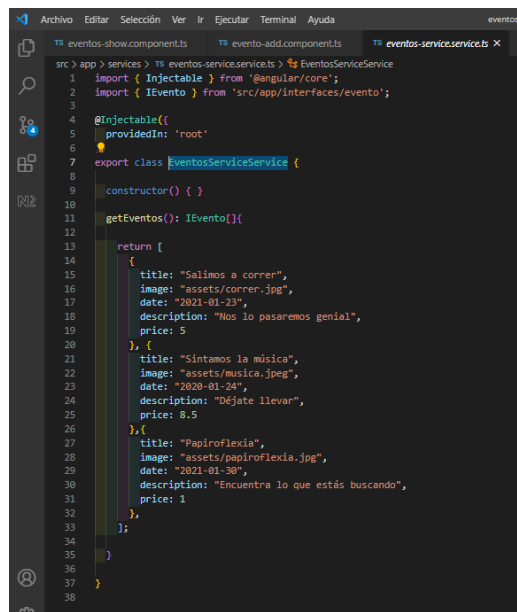
A screenshot of a web browser showing a web application running on localhost:4200. The application has a light blue header with navigation links: 'Aplicaciones', 'Social', 'Movil', 'BOLSA', 'Pelis y Series', 'DAM/DAW', 'CAM', and 'Outlook'. The main content area is divided into two sections. The top section, titled 'Datos', contains a form with four rows: 'Nombre' (value: Inma), 'Primer Apellido' (value: Va a sacar un), 'Edad' (value: 10), and 'Resultado' (value: Inmaajs89/Va a sacar un10). The bottom section, titled 'Genera Contraseña', contains two buttons: 'Simple' and 'Compleja'.

3. Directiva:

Las directivas son comandos que nos ayudan a extender utilidades propias de cada clase. Me encantaría explicártelo mejor, pero la verdad es que no sé.

4. Servicios, servicios web:

Un servicio nos sirve para compartir información entre todos los componentes de la aplicación que nos interese. Por ejemplo, podemos meter dentro un array de eventos y así desde cualquier componente llamarlo. O crear una función para añadir/eliminar eventos y así recurrir a ella cuando se desee. Yo lo resumiría como un directorio.

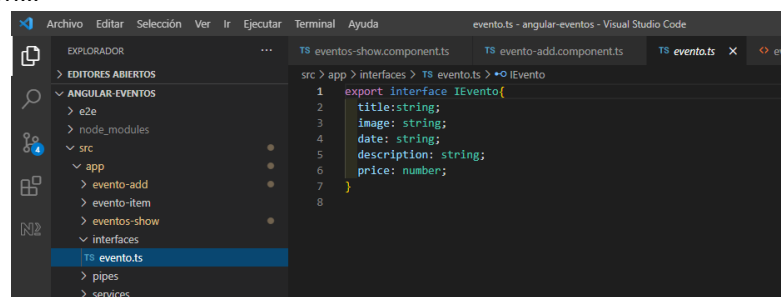


```
1 import { Injectable } from '@angular/core';
2 import { IEvento } from 'src/app/interfaces/evento';
3
4 @Injectable({
5   providedIn: 'root'
6 })
7 export class EventosServiceService {
8
9   constructor() {}
10
11   getEventos(): IEvento[] {
12
13     return [
14       {
15         title: "Salimos a correr",
16         image: "assets/correr.jpg",
17         date: "2021-01-23",
18         description: "Nos lo pasaremos genial",
19         price: 5
20       }, {
21         title: "Sintamos la música",
22         image: "assets/musica.jpeg",
23         date: "2020-01-24",
24         description: "Déjate llevar",
25         price: 8.5
26       }, {
27         title: "Papiroflexia",
28         image: "assets/papiroflexia.jpg",
29         date: "2021-01-30",
30         description: "Encuentra lo que estás buscando",
31         price: 1
32       },
33     ];
34   }
35 }
36
37
38
```

5. Interfaz:

Para mi, una interfaz es como definir una clase. En ella vamos a poner las características que nos interesen de nuestro “producto”, pero no los métodos, tal y como hacíamos en java, ya que estos irán en los servicios.

En nuestro ejemplo, la interfaz era IEventos, donde se establecía que tenía un título, una descripción...



```
1 export interface IEvento {
2   title: string;
3   image: string;
4   date: string;
5   description: string;
6   price: number;
7 }
8
```

6. Angular Routes:

En angular hemos visto como no trabajamos con diferentes paginas web, tal y como hacíamos al inicio de este fp, sino que sobre una vamos añadiendo componentes y esta va cambiando. Para ello usamos las angular routes, las cuales nos van redireccionando de forma interna por nuestra página. Un ejemplo es este método, de la práctica de productos, en la que volvemos hacia atrás o vamos a la página editar:

```
25  
26 goBack() {  
27   this.router.navigate(['/productos']);  
28 }  
29  
30 edit() {  
31   this.router.navigate(['/productos/', this.producto.id, 'edit']);  
32 }  
33
```

7. Observables:

Los observables son unos componentes los cuales, a groso modo y eso entendí, se quedan “mirando” la web para percatarse de los cambios que se produzcan y así actuar en consecuencia. Por ejemplo, en nuestro caso, uno de los observables estaba presente en añadir un evento. Cuando tu le dabas a enviar el observable veía que había pasado y ejecutaba al servicio (mas o menos).

```
getEventos(): Observable<IEvento[]> {  
  return this.http.get<EventosResponse>('/eventos').pipe(  
    map(resp => resp.eventos)  
  );  
}  
  
addEvento(evento: IEvento): Observable<IEvento> {  
  return this.http.post<EventoResponse>('/eventos', evento).pipe(  
    map(resp => resp.evento)  
  );  
}  
  
deleteEvento(id: number): Observable<void> {  
  return this.http.delete<void>('/eventos/${id}');  
}
```