

Componente de Cliente – parte 2



Listado de Clientes y Resumen del Total de Clientes y Saldo

En esta lección, completaremos el listado de personas y las tarjetas que mostrarán el **resumen del total de clientes** y el **total del saldo de todos los clientes**. También mejoraremos el código para hacerlo más eficiente y moderno, utilizando las últimas características de TypeScript y Angular.

Código del archivo `clientes.component.html`:

```


<section id="actions" class="py-4 mb-4">
  <div class="container">
    <div class="row">
      <div class="col-md-3">
        <a href="#" class="btn btn-primary w-100"
           data-bs-toggle="modal" data-bs-target="#agregarClienteModal">
          <i class="bi bi-plus-circle"></i> Agregar Cliente
        </a>
      </div>
    </div>
  </div>
</section>



```

```
<section id="clientes">
  <div class="container">
    <div class="row">
      <div class="col-md-9">
        <div class="card">
          <div class="card-header">
            <h4>Listado de Clientes</h4>
          </div>
          <table class="table table-striped">
            <thead class="table-dark">
              <tr>
                <th>Nombre</th>
                <th>Saldo</th>
                <th></th>
              </tr>
            </thead>
            <tbody>
              <!-- Iteración sobre la lista de clientes -->
              @for (cliente of clientes; track cliente) {
                <tr>
                  <td>{{cliente.nombre}} {{cliente.apellido}}</td>
                  <td>{{cliente.saldo | currency: 'MXN':'$'}}</td>
                  <td>
                    <a routerLink="/cliente/editar/{{cliente.id}}" class="btn btn-secondary">
                      <i class="bi bi-pencil-square"></i> Editar
                    </a>
                  </td>
                </tr>
              }
            </tbody>
          </table>
        </div>
      </div>
    </div>
    <!-- Tarjetas laterales -->
    <div class="col-md-3">
      <div class="card text-center bg-danger text-white mb-3">
        <div class="card-body">
          <h3>Saldo Total</h3>
          <h4 class="display-4">
            {{getSaldoTotal() | currency:'MXN':'$'}}
          </h4>
        </div>
      </div>
    </div>
  </div>
</section>
```

```

<div class="card text-center bg-success text-white mb-3">
  <div class="card-body">
    <h3>Total Clientes</h3>
    <h4 class="display-4">
      <i class="bi bi-people-fill"></i> {{clientes?.length}}
    </h4>
  </div>
</div>
</div>
</div>
</section>

```

Comentarios importantes:

- Botón Agregar Cliente:** Utiliza un modal Bootstrap para abrir una ventana de diálogo donde se podrá agregar un cliente nuevo que agregaremos más adelante.
- Tabla de Clientes:** Se recorre la lista de clientes mediante un bucle `@for`. Se muestran el nombre y saldo de cada cliente, y un botón de edición con el icono de lápiz de Bootstrap (`bi-pencil-square`).
- Tarjeta de Saldo Total:** Muestra el saldo total calculado con el método `getSaldoTotal()` en formato de moneda mexicana (MXN).
- Tarjeta de Total de Clientes:** Muestra el número total de clientes, con el icono de personas de Bootstrap (`bi-people-fill`).

Ahora en la clase de componente de clientes agregamos el método `getSaldo`, hemos agregado dos versiones de este método, pero dejamos activada la versión más moderna con su explicación respectiva.

Archivo `clientes.component.ts`:

```

import { Component } from '@angular/core';
import { Cliente } from '../../../../../modelo/cliente.modelo';
import { ClienteServicio } from '../../../../../servicios/cliente.service';
import { CommonModule } from '@angular/common';
import { RouterModule } from '@angular/router';

@Component({
  selector: 'app-clientes',
  standalone: true,
  imports: [CommonModule, RouterModule],
  templateUrl: './clientes.component.html',
  styleUrls: ['./clientes.component.css']
})
export class ClientesComponent {

```

```

    clientes: Cliente[] | null = null;

constructor(private clientesServicio: ClienteServicio) {}

ngOnInit() {
  // Al inicializar el componente, se suscribe al servicio para obtener la lista de clientes
  this.clientesServicio.getClientes().subscribe(clientes => {
    this.clientes = clientes; // Almacena el listado en la propiedad 'clientes'
  });
}

getSaldoTotal(): number {
  return this.clientes?.reduce((total, cliente) => total + (cliente.saldo ?? 0), 0) ?? 0;
}

// getSaldoTotal() {
//   let saldoTotal: number = 0;
//   if(this.clientes){
//     this.clientes.forEach( cliente => {
//       if (cliente.saldo !== undefined) {
//         saldoTotal += cliente.saldo;
//       }
//     });
//   }
//   return saldoTotal;
// }
}

```

Explicación del método `getSaldoTotal`:

- Método `reduce`:** Utilizamos el método `reduce` de los arrays para sumar el saldo de todos los clientes. Este método toma un acumulador (`total`) y va sumando el saldo de cada cliente. Si el saldo de un cliente es `undefined`, usamos el operador **nullish coalescing** (`??`) para tratarlo como `0`.
- Optional chaining (`?.`):** Esta característica verifica de manera concisa si la propiedad `clientes` es `null` o `undefined` antes de intentar aplicar `reduce`. Si no hay clientes, la operación no se ejecutará y se devolverá `0`.
- Nullish coalescing (`??`):** Si el array de clientes es `null` o `undefined`, devolvemos `0` como valor predeterminado para el saldo total.

Paso a Paso de la Expresión

- this.clientes?.** (Encadenamiento Opcional - **Optional Chaining**)
 - this.clientes hace referencia a una propiedad del objeto actual que se espera sea un **arreglo** de clientes.

- El **operador ?.** (encadenamiento opcional) se utiliza para verificar si clientes no es null ni undefined antes de intentar llamar al método reduce.
 - Si **clientes** es null o undefined, el resultado de la expresión hasta este punto será undefined y la operación no seguirá con el reduce.
2. **reduce()** (Método de Reducción)
- **reduce()** es un método de los arreglos en JavaScript que se utiliza para **reducir** todos los elementos de un arreglo a un solo valor acumulado.
 - La **función de reduce()** se define como $(total, cliente) \Rightarrow total + (cliente.saldo ?? 0)$.
 - **Argumentos de reduce():**
 - **total:** Es el acumulador que se actualiza en cada iteración del arreglo.
 - **cliente:** Es el elemento actual del arreglo sobre el que se está iterando.
 - **0:** Es el **valor inicial** del acumulador total. Este valor se pasa como segundo argumento del método `reduce()`, indicando que `total` comienza en 0.
3. **$(total, cliente) \Rightarrow total + (cliente.saldo ?? 0)$** (Función de Callback de `reduce()`)
- Esta es una función **arrow function** que se ejecuta para cada elemento del arreglo `clientes`.
 - **total + (cliente.saldo ?? 0):**
 - **cliente.saldo** hace referencia al saldo de cada cliente.
 - **??** (Operador de Coalescencia Nula - **Nullish Coalescing**): Este operador se utiliza para proporcionar un valor por defecto si `cliente.saldo` es null o undefined.
 - Si `cliente.saldo` es null o undefined, entonces se tomará el valor 0 como saldo.
 - **total + (cliente.saldo ?? 0):** Se suma el saldo del cliente (o 0 si está indefinido) al total acumulado.
4. **$?? 0$** (Operador de Coalescencia Nula - **Nullish Coalescing**)
- **??** se usa nuevamente al final para manejar el caso en el que `this.clientes` es null o undefined.
 - Si `this.clientes` no existe (es null o undefined), entonces toda la expresión anterior será undefined, por lo que **$?? 0$** asegura que el valor de retorno sea 0 en ese caso.

Beneficios del Código Modernizado:

- **Mayor legibilidad:** El código es más fácil de leer y entender.
- **Concisión:** Se eliminan las estructuras condicionales adicionales que eran necesarias en versiones anteriores de TypeScript/Angular.
- **Eficiencia:** Se aprovechan las mejoras recientes del lenguaje para hacer el código más eficiente.

Con estos cambios, el código es más conciso y moderno, utilizando las características más recientes de TypeScript y Angular.

Resultado Final hasta el momento

The screenshot shows a web application titled "Control de Clientes". At the top, there's a navigation bar with "Control Clientes" on the left and "Inicio Bienvenido" on the right. Below the title, there's a button labeled "+ Agregar Cliente". On the left, there's a table titled "Listado de Clientes" showing two rows of data:

Nombre	Saldo	
Juan Perez	\$100.00	<input checked="" type="button"/> Editar
Karla Gomez	\$200.00	<input checked="" type="button"/> Editar

On the right side, there are two summary boxes: "Saldo Total" showing "\$300.00" and "Total Clientes" showing "2". At the bottom of the page, there's a yellow footer bar with the text "© GlobalMentoring.com.mx".

Con estos cambios, el código es más conciso y moderno, utilizando las características más recientes de TypeScript y Angular.

¡Sigue adelante con tu aprendizaje , el esfuerzo vale la pena!

Saludos 

Ing. Ubaldo Acosta

Fundador de [GlobalMentoring.com.mx](http://www.globalmentoring.com.mx)