

Componente Agregar Cliente – parte 2



Agregar un Cliente – Parte 2

En esta lección, vamos a agregar validaciones a cada uno de los campos del formulario para asegurar que los datos ingresados sean correctos antes de guardar el cliente en la base de datos de **Cloud Firestore**.

Código del archivo `clientes.component.html`

```
<!-- Botones -->
<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>

<!-- Listado Clientes -->
```

```
<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>

      <!-- 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>
  </div>
</section>
```



```
<input type="text" id="apellido" name="apellido"
       class="form-control" required minlength="2"
       [(ngModel)]="cliente.apellido" #apellido="ngModel" />
@if (apellido.invalid && apellido.touched) {
<div class="text-danger">
    Apellido es requerido y debe tener al menos 2 caracteres.
</div>
}
</div>
</div>

<!-- Campo de Email -->
<div class="row mb-3">
    <label for="email" class="col-form-label col-sm-3">Email</label>
    <div class="col-sm-9">
        <input type="email" id="email" name="email"
               class="form-control" required email
               [(ngModel)]="cliente.email" #email="ngModel" />
@if (email.invalid && email.touched) {
<div class="text-danger">
    Por favor ingresa un email válido.
</div>
}
</div>
</div>

<!-- Campo de Saldo -->
<div class="row mb-3">
    <label for="saldo" class="col-form-label col-sm-3">Saldo</label>
    <div class="col-sm-9">
        <input type="number" id="saldo" name="saldo"
               class="form-control" required min="0.01"
               step="0.01" [(ngModel)]="cliente.saldo" #saldo="ngModel" />
@if (saldo.invalid && saldo.touched) {
<div class="text-danger">
    El saldo es requerido y debe ser un número positivo.
</div>
}
</div>
</div>
</div>

<!-- Footer del modal con el botón de guardar -->
<div class="modal-footer">
    <button type="submit" class="btn btn-primary" [disabled]="clienteForm.invalid">
```

```
        Guardar
    </button>
</div>
</form>
</div>
</div>
</div>
```

Comentarios adicionales sobre las validaciones:

- **Nombre y Apellido:** Ambos campos son obligatorios y deben tener al menos 2 caracteres.
- **Email:** Se asegura que el email sea válido.
- **Saldo:** El saldo es un campo obligatorio y debe ser un número mayor a 0.

Código del 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';
import { FormsModule, NgForm } from '@angular/forms';

@Component({
  selector: 'app-clientes',
  standalone: true,
  imports: [CommonModule, RouterModule, FormsModule],
  templateUrl: './clientes.component.html',
  styleUrls: ['./clientes.component.css']
})
export class ClientesComponent {
  clientes: Cliente[] | null = null;
  cliente: Cliente = {
    nombre: '',
    apellido: '',
    email: '',
    saldo: undefined ,
  };
}

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;
}

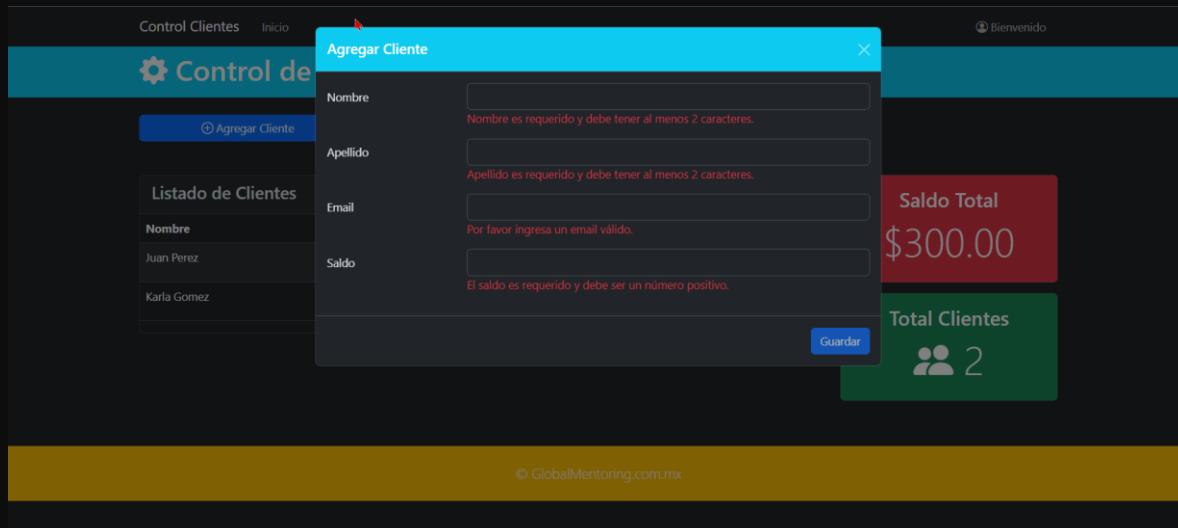
agregar(clienteForm: NgForm) {
  const { value, valid } = clienteForm;

  if (valid) {
    // Aquí puedes agregar la lógica para guardar el cliente
    // Limpiamos el formulario para agregar un nuevo cliente
    clienteForm.resetForm();
  }
}
}
}

```

Hemos agregado el módulo de FormsModule para poder usar el concepto de two way binding en nuestro formulario.

Resultado Final hasta el momento



Conclusión

Con estos cambios ya está listo el formulario para poder agregar un nuevo cliente ya validado antes de agregar este nuevo cliente a la base de datos de Cloud Firebase.

Recuerda que tienes disponible el proyecto con todo el código completamente funcional de esta lección en caso de que lo necesites.

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

Saludos 

Ing. Ubaldo Acosta

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