

Componente Editar Cliente



Editar un Cliente

En esta lección, vamos a editar un cliente ya existente en la base de datos de **Cloud Firestore**. Para ello vamos a modificar el componente de editar-cliente y también el servicio de clientes.

Código del archivo `clientes.service.ts`

```
import { Injectable } from '@angular/core';
import { Cliente } from '../modelo/cliente.modelo';
import {
  Firestore,
  collection,
  collectionData,
  addDoc,
  docData,
} from '@angular/fire/firestore';
import { Observable } from 'rxjs';
import { query, orderBy, doc } from 'firebase/firestore';

@Injectable({
  providedIn: 'root',
})
```

```
export class ClienteServicio {  
  private clientesRef: any; // Definir la referencia aquí  
  clientes: Observable<Cliente[]>;  
  
  constructor(private firestore: Firestore) {  
    // Inicializar la referencia a la colección 'clientes' dentro del constructor  
    this.clientesRef = collection(this.firebaseio, 'clientes');  
  
    // Crear una consulta para ordenar los clientes por nombre ascendente  
    const consulta = query(this.clientesRef, orderBy('nombre', 'asc'));  
  
    // Obtener los datos de la consulta como un Observable  
    this.clientes = collectionData(consulta, { idField: 'id' }) as Observable<  
      Cliente[]  
    >;  
  }  
  
  // Método para obtener los clientes  
  getClientes(): Observable<Cliente[]> {  
    return this.clientes;  
  }  
  
  // Método para agregar un cliente  
  agregarCliente(cliente: Cliente) {  
    // Reutilizamos la referencia 'clientesRef'  
    return addDoc(this.clientesRef, cliente);  
  }  
  
  // Método para obtener un cliente por ID  
  getCliente(id: string): Observable<Cliente | null> {  
    // Usamos 'doc' para obtener la referencia al documento y 'docData' para obtener los datos  
    const clienteDocRef = doc(this.firebaseio, `clientes/${id}`);  
  
    // Retornamos los datos como un Observable  
    return docData(clienteDocRef, { idField: 'id' }) as Observable<Cliente>;  
  }  
}
```

Código del archivo `editar-cliente.component.ts`

```
import { Component } from '@angular/core';  
import { Cliente } from '../../../../../modelo/cliente.modelo';  
import { ClienteServicio } from '../../../../../servicios/cliente.service';
```

```
import { ActivatedRoute, Router, RouterModule } from '@angular/router';
import { CommonModule } from '@angular/common';
import { FormsModule, NgForm } from '@angular/forms';

@Component({
  selector: 'app-editar-cliente',
  standalone: true,
  imports: [CommonModule, FormsModule, RouterModule], // Importamos módulos necesarios
  templateUrl: './editar-cliente.component.html',
  styleUrls: ['./editar-cliente.component.css']
})
export class EditarClienteComponent {

  cliente: Cliente = {
    nombre: '',
    apellido: '',
    email: '',
    saldo: undefined,
  };

  id: string | null = null;

  constructor(
    private clientesServicio: ClienteServicio,
    private router: Router,
    private route: ActivatedRoute
  ) {}

  ngOnInit() {
    // Obtener el 'id' del parámetro de ruta
    this.id = this.route.snapshot.paramMap.get('id');

    // Verificar si el 'id' no es null antes de buscar el cliente
    if (this.id) {
      this.clientesServicio.getCliente(this.id).subscribe((cliente: Cliente | null) => {
        if (cliente) {
          // Asignar el cliente si existe
          this.cliente = cliente;
        } else {
          console.log('Cliente no encontrado: ' + this.id);
          // Redirigir al inicio si no se encuentra el cliente
          this.router.navigate(['/']);
        }
      });
    } else {
  
```

```

        console.log('ID no proporcionado');
        // Redirigir al inicio si no hay ID
        this.router.navigate(['/']);
    }

}

eliminar() {
    throw new Error('Method not implemented.');
}

guardar(ngForm: NgForm) {
    throw new Error('Method not implemented.');
}
}
}

```

Código del archivo editar-cliente.component.html

```

<header id="main-header" class="py-2 bg-info text-white">
    <div class="container">
        <h1><i class="bi bi-person"></i> Editar Cliente</h1>
    </div>
</header>

<form #clienteForm="ngForm" (ngSubmit)="guardar(clienteForm)">
    <!-- Botones -->
    <section id="actions" class="py-4 mb-4">
        <div class="container d-flex justify-content-between">
            <a [routerLink]="/" class="btn btn-primary">
                <i class="bi bi-arrow-left"></i> Regresar al Inicio
            </a>
            <button type="submit" class="btn btn-success">
                <i class="bi bi-check"></i> Guardar Cliente
            </button>
            <a href="#" (click)="eliminar()" class="btn btn-danger">
                <i class="bi bi-trash"></i> Eliminar Cliente
            </a>
        </div>
    </section>

    <section id="details">
        <div class="container">
            <div class="card">
                <div class="card-header">
                    <h4>Editar Cliente</h4>
                </div>

```

```
<div class="card-body">
    <!-- Campo de Nombre -->
    <div class="mb-3 d-flex align-items-center">
        <label for="nombre" class="form-label me-2"
            style="min-width: 100px;">Nombre</label>
        <input type="text" id="nombre" name="nombre"
            class="form-control" required minlength="2"
            [(ngModel)]="cliente.nombre" #nombre="ngModel" />
        @if (nombre.invalid && nombre.touched) {
            <div class="text-danger">
                Nombre es requerido y debe tener al menos 2 caracteres.
            </div>
        }
    </div>

    <!-- Campo de Apellido -->
    <div class="mb-3 d-flex align-items-center">
        <label for="apellido" class="form-label me-2"
            style="min-width: 100px;">Apellido</label>
        <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>

    <!-- Campo de Email -->
    <div class="mb-3 d-flex align-items-center">
        <label for="email" class="form-label me-2"
            style="min-width: 100px;">Email</label>
        <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>

    <!-- Campo de Saldo -->
    <div class="mb-3 d-flex align-items-center">
```

```

<label for="saldo" class="form-label me-2"
      style="min-width: 100px;">Saldo</label>
<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>
</div>
</section>
</form>

```

Resultado Final hasta el momento

Control Clientes Inicio Bienvenido

Editar Cliente

← Regresar al Inicio Guardar Cliente Eliminar Cliente

Editar Cliente	
Nombre	Carlos
Apellido	Lara
Email	clara@mail.com
Saldo	300

© GlobalMentoring.com.mx

Conclusión

Con estos cambios ya hemos cargado el cliente seleccionado en la plantilla de editar cliente. Ya solo falta terminar el código de los botones de Guardar Cliente y Eliminar Cliente.

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