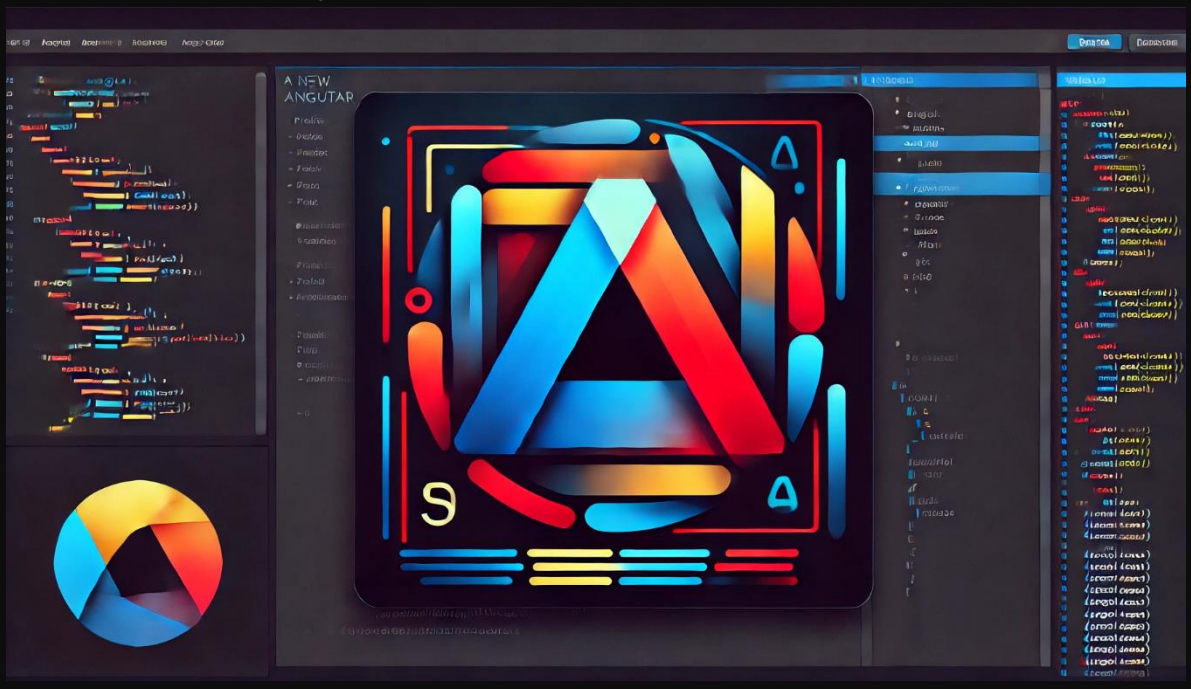
Modificar y Eliminar un Cliente



Modificar y Eliminar un Cliente

En esta lección, vamos a terminar de guardar los cambios para poder modificar un cliente ya existente, y también vamos a agregar el caso de eliminar 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,
  updateDoc,
  deleteDoc,
  doc,
} from '@angular/fire/firestore';
import { Observable } from 'rxjs';
import { query, orderBy } from 'firebase/firestore';
```

```typescript
@Injectable({
  providedIn: 'root',
})
export class ClienteServicio {
  private clientesRef: any;
  clientes: Observable<Cliente[]>;

  constructor(private firestore: Firestore) {
    // Inicializar la referencia a la colección 'clientes' dentro del constructor
    this.clientesRef = collection(this.firestore, '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) {
    return addDoc(this.clientesRef, cliente);
  }

  // Método para obtener un cliente por ID
  getCliente(id: string): Observable<Cliente | null> {
    const clienteDocRef = doc(this.firestore, `clientes/${id}`);
    return docData(clienteDocRef, { idField: 'id' }) as Observable<Cliente>;
  }

  // Método para modificar un cliente
  modificarCliente(cliente: Cliente) {
    const clienteDoc = doc(this.firestore, `clientes/${cliente.id}`);
    return updateDoc(clienteDoc, { ...cliente }); // Actualizamos el documento con los nuevos datos
  }

  // Método para eliminar un cliente
  eliminarCliente(cliente: Cliente) {
    const clienteDoc = doc(this.firestore, `clientes/${cliente.id}`);
    return deleteDoc(clienteDoc); // Eliminamos el documento por su ID
  }
```

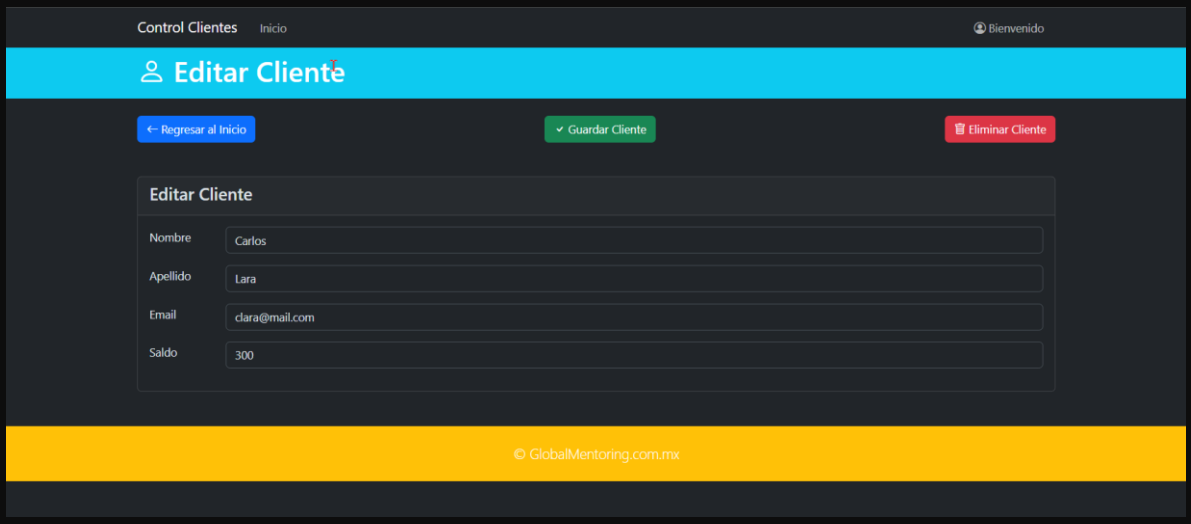
```
}
```

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',
  styleUrl: './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(['/']);
    }
  }

  guardar(clienteForm: NgForm) {
    const { value, valid } = clienteForm;
    if (valid) {
      value.id = this.id;
      //modificar  cliente
      this.clientesServicio.modificarCliente(value);
      this.router.navigate(['/']);
    }
  }

  eliminar() {
    if(confirm('¿Seguro que desea eliminar el cliente?')){
      this.clientesServicio.eliminarCliente(this.cliente);
      this.router.navigate(['/']);
    }
  }
}
```

Resultado Final hasta el momento



Conclusión

Con estos cambios ya hemos terminado las acciones de nuestra aplicación. Ya podemos listar, agregar, editar y eliminar los clientes de nuestra aplicación de control de clientes.

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

¡Muchas felicidades por haber llegado hasta este punto del curso, sigue con tu aprendizaje 🚀, el esfuerzo vale la pena!

Saludos 👋

Ing. Ubaldo Acosta

Fundador de GlobalMentoring.com.mx