Servicio Clientes del Proyecto de Control de clientes con Angular

Creación del Servicio para Recuperar Clientes con Cloud Firestore

En esta lección, aprenderemos a crear el servicio que permitirá recuperar los clientes desde la base de datos de **Cloud Firestore**.

1. Creación de la Clase Modelo `Cliente`

Primero, debemos definir la estructura de un cliente. Para esto, creamos una **interface** en lugar de una clase, aunque ambas opciones serían válidas. Primero creamos una carpeta llamada modelo, y posteriormente creamos el archivo cliente.modelo.ts.

Código del archivo `modelo/cliente.modelo.ts`:

```
export interface Cliente{
    id?:string;
    nombre?:string;
    apellido?:string;
    email?:string;
```

```
    saldo?:number;
}
```

Explicación:

- **`id`, `nombre`, `apellido`, `email` y `saldo`**: Estos son los campos básicos que tendrá cada cliente, con la posibilidad de ser opcionales (`?`) en caso de no estar presentes en todos los registros.

2. Creación del Servicio de Clientes

Generamos el servicio utilizando Angular CLI con el siguiente comando:

ng g s servicios/cliente --skip-tests

Este comando crea el servicio **`ClienteServicio`** en la carpeta **servicios**. El flag **`--skip-tests`** evita la creación del archivo de pruebas.

3. Modificación del Servicio para Conectarlo a Firestore

Ahora, modificamos el código del servicio para que pueda conectarse a la base de datos de **Cloud Firestore** y realizar consultas.

Código del archivo `cliente.service.ts`:

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

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

  constructor(private firestore: Firestore) {
    // Realiza la consulta directamente usando la API modular
    const clientesRef = collection(this.firestore, 'clientes');
    const consulta = query(clientesRef, orderBy('nombre', 'asc'));
    this.clientes = collectionData(consulta, { idField: 'id' }) as Observable<Cliente[]>;
  }

  getClientes(): Observable<Cliente[]> {
```

```
    return this.clientes;
  }
}
```

Explicación:

- `@Injectable({ providedIn: 'root' })`: Indica que el servicio será proporcionado a nivel global de la aplicación (singleton), sin necesidad de importarlo en un módulo específico.
- `Firestore`: Utilizamos el servicio **Firestore** de Angular para interactuar con **Cloud Firestore**.
- `collection()`: Esta función obtiene la referencia a la colección `clientes` dentro de Firestore.
- `query()`: Se utiliza para ordenar los resultados de la consulta por el campo `nombre` en orden ascendente.
- `collectionData()`: Convierte los resultados de la consulta en un **Observable** que contiene los datos de los clientes, añadiendo también el **id** del documento con la opción `{ idField: 'id' }`.

Cómo funciona el `idField`:

- Cuando usas `collectionData(consulta, { idField: 'id' })`, le estás diciendo a **Firestore** que tome el **ID** del documento (que es externo a los datos) y lo agregue como un campo con el nombre `id` en el objeto que devuelve.
- Con la opción **idField: 'id'**, lo que obtienes en tu **Observable** será un objeto que incluye el **ID del documento** como parte de los datos:

Conclusión

Con esto hemos creado el servicio de **clientes** que nos permite conectarnos a **Cloud Firestore** y realizar consultas. Ahora podemos obtener y mostrar los datos de los clientes en nuestra aplicación.

¡Sigue adelante con tu aprendizaje!

¡Saludos!

Ing. Ubaldo Acosta

Fundador de GlobalMentoring.com.mx