BITÁCORA

PARTE 1

1. Ajuste inicial de Sheets

Se procede a crear una hoja en Google Sheets y en su Script se crea una mini API que enviará los datos, lo único que hasta ahora se tomará después de haber hecho lo anterior será el enlace de Script.

1. Establecimiento de la lógica de negocio
   1. Se crea la carpeta pages y dentro de ella el componente home
   2. Se usa el componente home dentro del app-component.hmtl, para lo cual se debe importar en el respectivo .ts
   3. Se crea el servicio google-sheets, el cual será el encargado de traer los datos de la mini API

import { Injectable } from '@angular/core'; // Marca la clase como un servicio

import { HttpClient } from '@angular/common/http'; // Permite hacer solicitudes HTTP

import { Observable } from 'rxjs'; // Ayuda a manejar datos que llegan de manera asíncrona

@Injectable({

  providedIn: 'root', // Hace que este servicio esté disponible en toda la aplicación

})

export class GoogleSheetsService {

  private apiUrl = 'https://script.google.com/macros/s/AKfycbxd5qw50sYYJdi\_zdmE7Gm5e6xtarzhBotVDXjvsSXiU1qpauwd7jyoABL-GvxfKcGK/exec'; // Aquí debes poner el enlace de tu mini API

  constructor(private http: HttpClient) {} // HttpClient es necesario para hacer las solicitudes HTTP

  // Este método se encargará de obtener los datos de la mini API

  getData(): Observable<any[]> {

    return this.http.get<any[]>(this.apiUrl); // Hace una solicitud GET y espera recibir datos en formato JSON

  }

}

* 1. Para poder hacer las peticiones http dentro de toda la página, angular debe registrar HttpClientModule como un proveedor global para la aplicación. Esto permite que puedas usar servicios como HttpClient en cualquier componente standalone, sin tener que preocuparte por importarlo en cada uno de ellos, excepto en los servicios. Para ello se agrega lo siguiente en el archivo app.config.ts

import { HttpClientModule } from '@angular/common/http'; // Parte agregada

import { ApplicationConfig, importProvidersFrom, provideZoneChangeDetection } from '@angular/core';

import { provideRouter } from '@angular/router';

import { routes } from './app.routes';

import { provideClientHydration } from '@angular/platform-browser';

export const appConfig: ApplicationConfig = {

  providers: [

    provideZoneChangeDetection({ eventCoalescing: true }), provideRouter(routes), provideClientHydration(),

    // Parte agregada:

    importProvidersFrom(

      HttpClientModule,

    )

  ],

};

* 1. Se consume el servicio google-sheets dentro del componente home.ts, además, para usar las directivas ngIf y ngFor en angular 19, de sebe importar en cada componente el commonModule y declararlo en el decorador principal. Dicho módulo que contiene directivas comunes como \*ngIf, \*ngFor, ngClass, ngStyle, entre otras.

import { CommonModule } from '@angular/common';

import { GoogleSheetsService } from './../../services/google-sheets.service';

import { Component } from '@angular/core';

@Component({

  selector: 'app-home',

  standalone: true,

  imports: [CommonModule], // Necesario para

  templateUrl: './home.component.html',

  styleUrl: './home.component.scss'

})

export class HomeComponent {

  data: any[] = []; // Aquí almacenaremos los datos que lleguen de la API

  constructor(private googleSheetsService: GoogleSheetsService) {}

  ngOnInit(): void {

    // Llamamos al servicio para obtener los datos

    this.googleSheetsService.getData().subscribe({

      next: (response) => {

        this.data = response; // Guardamos los datos en nuestra variable

        console.log('Datos recibidos:', this.data); // Mostramos los datos en la consola

      },

      error: (err) => console.error('Error al obtener los datos:', err), // Mostramos el error si ocurre

    });

  }

}

* 1. Se configura el template de home para visualizar los datos obtenidos por el servicio

<div \*ngIf="data.length > 0; else noData">

  <h2>Datos desde Google Sheets</h2>

  <table>

    <thead>

      <tr>

        <th>Fecha</th>

        <th>Temperatura</th>

        <th>Humedad</th>

      </tr>

    </thead>

    <tbody>

      <tr \*ngFor="let item of data">

        <td>{{ item[0] }}</td> <!-- Suponiendo que la primera columna es la fecha -->

        <td>{{ item[1] }}</td> <!-- Suponiendo que la segunda columna es la temperatura -->

        <td>{{ item[2] }}</td> <!-- Suponiendo que la tercera columna es la humedad -->

      </tr>

    </tbody>

  </table>

</div>

<ng-template #noData>

  <p>No hay datos disponibles.</p>

</ng-template>

* 1. Se ajusta el servicio google-sheets para que hace solicitudes automáticas cada 500 milisegundos y así los cambios que se hagan en la sheet se vean en tiempo real.
     1. Se agrega lo siguiente al servicio google-sheets.ts

import { Observable } from 'rxjs'; // Ayuda a manejar datos que llegan de manera asíncrona

import { interval, switchMap } from 'rxjs'; // Herramientas para manejar flujos de datos reactivos y realizar polling.

// Este método realiza solicitudes a la API cada cierto tiempo (por defecto cada 5000 ms).

  getDataWithPolling(intervalMs: number = 500): Observable<any[]> {

    return interval(intervalMs) // Genera un flujo que emite valores cada `intervalMs` milisegundos.

      .pipe(

        switchMap(() => this.getData()) // Por cada emisión, llama al método `getData` para obtener datos actualizados.

      );

  }

* + 1. Se agrega lo siguiente al archivo home.component.ts

import { Component, OnInit } from '@angular/core'; // Se importa OnInit para poder usarlo

public data: any[] = []; // Se hace pública la variable para almacenar los datos que vienen del servicio

Se reemplaza la función ngOnInit por lo siguiente, para en vez de usar getData, usar getDataWithPolling

  ngOnInit(): void {

    this.googleSheetsService.getDataWithPolling(5000).subscribe({

      next: (data) => {

        this.data = data; // Asigna los datos recibidos a "data"

      },

      error: (err) => console.error('Error al obtener datos:', err),

    });

  }

1. Se realiza los gráficos