**Instituto Tecnológico de Culiacán**



**Ingeniería en tecnologías de la información y comunicación**

**“Proyecto Final: AppClima”**

**Materia:** Programación Web  
**Nombre del alumno:** Jorge Osuna Quintana  
**Maestro:**  M.C. Francisco González Hernández   
**Fecha de entrega:** 18 de mayo de 2020

**Código del servicio geolocalización-service**

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

import {HttpClient} from '@angular/common/http'

@Injectable({

  providedIn: 'root'

})

export class GeolocalizacionService {

  ipURL='https://api.ipify.org?format=json'

  ipStackBase='http://api.ipstack.com/'

  key='f95ed04b6922fd52b78ce7f73dedbcae';

  constructor(private \_http:HttpClient) { }

  getIP(){

    let observer= this.\_http.get(this.ipURL);

    return observer;

  }

  getIPInfo(ip){

    return this.\_http.get(`${this.ipStackBase}${ip}?access\_key=${this.key}&format=1`)

  }

}

**Código del servicio info-clima-service**

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

import {HttpClient} from '@angular/common/http'

import {GeolocalizacionService} from '../app/geolocalizacion.service'

import { mergeMap, map } from 'rxjs/operators';

import { Observable, observable,forkJoin } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class InfoClimaService {

  constructor( private \_http:HttpClient, private ipStack:GeolocalizacionService) {

   }

  getClima(){

    return this.ipStack.getIP().pipe(

      mergeMap(result=>{

        const observeIP=this.ipStack.getIPInfo(result['ip'])

        return observeIP.pipe(

          mergeMap(infoIP=>{

            const infoGeo=

            {

              latitud:infoIP['latitude'],

              longitud: infoIP['longitude'],

            }

    return forkJoin(this.\_http.get(`http://www.7timer.info/bin/api.pl?lon=${infoGeo.longitud}

            &lat=${infoGeo.latitud}&product=civillight&output=json`),observeIP)

          }))

      })

    )

  }

}

**Código del controlador del componente Info-ip**

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

import {GeolocalizacionService } from '../geolocalizacion.service'

import {HttpClient} from '@angular/common/http'

import { mergeMap } from 'rxjs/operators';

@Component({

  selector: 'app-info-ip',

  templateUrl: './info-ip.component.html',

  styleUrls: ['./info-ip.component.css']

})

export class InfoIpComponent implements OnInit {

  ciudad='';

  pais='';

  hora=null;

  loaded=false;

  constructor(private \_http: HttpClient, private ipStack:GeolocalizacionService) { }

  ngOnInit() {

    this.cargarDatosIP()

  }

  cargarDatosIP(){

    this.ipStack.getIP().pipe(

      mergeMap(ip=>{

        return this.ipStack.getIPInfo(ip['ip'])

      })

    ).subscribe(

        data=>{

          this.ciudad=data['city'];

          this.pais=data['country\_name']

          let fechaActual=new Date();

          this.hora=`${fechaActual.getHours()}:${fechaActual.getMinutes()}`

          this.loaded=true;

        }

      )

  }

}

**Código de la vista del componente Info-ip**

<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto"/>

<div class="text-center pt-5" \*ngIf="loaded">

        <h3>{{ciudad}}, {{pais}} {{hora}}</h3>

</div>

**Código del controlador del componente info-clima**

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

import {InfoClimaService } from '../info-clima.service'

import {HttpClient} from '@angular/common/http'

@Component({

  selector: 'app-info-clima',

  templateUrl: './info-clima.component.html',

  styleUrls: ['./info-clima.component.css'],

  providers: [InfoClimaService]

})

export class InfoClimaComponent implements OnInit {

  loaded=false;

  infoClima=null;

  todayInfo=null;

  constructor(

    private climaAPI:InfoClimaService,

    private \_http: HttpClient) { }

  ngOnInit() {

    this.cargarDatosClima()

  }

  cargarDatosClima(){

    this.climaAPI.getClima().subscribe(result=>

      {

        this.infoClima=result[0]['dataseries'].map(x=>this.obtenerDatosDia(x));

        this.todayInfo=this.infoClima[0]

        this.loaded=true

      });

  }

  obtenerDatosDia(dataSerie){

    let fecha=dataSerie['date'].toString();

    let fechaFormatted= new Date(`${fecha.substring(0,4)}/${fecha.substring(4,6)}/${fecha.substring(6,8)}`);

    let meses = ["Enero","Febrero","Marzo","Abril","Mayo","Junio","Julio","Agosto","Septiembre","Octubre","Noviembre","Diciembre"];

    return {

        "FechaUF":fechaFormatted,

        "FechaF":`${fecha.substring(6,8)} de ${meses[fechaFormatted.getMonth()]} de ${fecha.substring(0,4)}`,

        "Max":dataSerie['temp2m']['max'],

        "Min":dataSerie['temp2m']['min'],

        "Clima":dataSerie['weather']

        }

  }

  obtenerDiaSemana(dia){

    let diaSemana='';

    switch(dia){

      case 0:

        diaSemana='Domingo';

      break

      case 1:

        diaSemana='Lunes'

      break

      case 2:

        diaSemana='Martes'

      break

      case 3:

        diaSemana='Miercoles'

      break

      case 4:

        diaSemana='Jueves'

      break

      case 5:

        diaSemana='Viernes'

      break

      case 6:

        diaSemana='Sabádo'

      break

    }

    return diaSemana

  }

  cambiarImagen(clima){

    let climaUrl='';

    switch(clima){

      case 'clear':

      climaUrl='../../assets/Images/Weather/clear\_weather.svg';

      break

      case 'pcloudy':

      climaUrl='../../assets/Images/Weather/partly\_cloudy.svg';

      break

      case 'cloudy':

      climaUrl='../../assets/Images/Weather/cloudy.svg';

      break

      case 'rain':

      climaUrl='../../assets/Images/Weather/rain.svg';

      break

      case 'snow':

      climaUrl='../../assets/Images/Weather/snow.svg';

      break

      case 'thunderstorm':

      climaUrl='../../assets/Images/Weather/thunderstorm.svg';

      break

      case 'lightrain':

      climaUrl='../../assets/Images/Weather/rain\_thunder.svg';

      break

    }

    return climaUrl

  }

}

**Código de la vista del controlador info-clima**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto"/>

</head>

<body>

    <div class="container align-content-center h-100">

        <div \*ngIf="!loaded" class="row">

            <img src="../../assets/Images/loading\_gif\_7.gif" alt="Cargado..." class="m-auto">

        </div>

        <div \*ngIf="loaded">

            <app-info-ip></app-info-ip>

            <div class="row justify-content-center">

                <div class="col-12 col-md-6 col-lg-5 mb-2 px-0">

                    <div class="card bg-light text-center contenedorClima">

                        <div class="card-header bg-light p-0">

                            <p class="mt-3 mb-0"><b>{{obtenerDiaSemana(todayInfo.FechaUF.getDay())}}</b></p>

                            <p class="">{{todayInfo.FechaF}}</p>

                        </div>

                        <img src="{{cambiarImagen(todayInfo.Clima)}}" alt="No se pudo obtener imagen"

                        class="img-clima card-img">

                        <div class="card-footer bg-light">

                            <p class="mb-0">Máxima: {{todayInfo.Max}}°C <i class="fas fa-thermometer-full"></i></p>

                            <p class="mb-0">Minima: {{todayInfo.Min}}°C <i class="fas fa-thermometer-quarter"></i></p>

                        </div>

                    </div>

                </div>

            </div>

            <div class="row justify-content-center">

                <div class="col-12 col-md-6 col-lg-5 mb-2 px-0" \*ngFor="let dataserie of infoClima.slice(1,5)">

                    <div class="card bg-light text-center contenedorClima">

                        <div class="card-header bg-light p-0">

                            <p class="mt-3 mb-0"><b>{{obtenerDiaSemana(dataserie.FechaUF.getDay())}}</b></p>

                            <p class="pb-0">{{dataserie.FechaF}}</p>

                        </div>

                        <img src="{{cambiarImagen(todayInfo.Clima)}}" alt="No se pudo obtener imagen"

                        class="img-clima card-img">

                        <div class="card-footer bg-light">

                            <p class="mb-0">Máxima: {{dataserie.Max}}°C <i class="fas fa-thermometer-full"></i></p>

                            <p class="mb-0">Minima: {{dataserie.Min}}°C <i class="fas fa-thermometer-quarter"></i></p>

                        </div>

                    </div>

                </div>

            </div>

        </div>

    </div>

</body>

</html>

**Vista de la página desde el navegador**

