05 CLIMA MUNDO

EXPLICACION DE APPS

**Objetivo: crear un programa que reciba desde la consola la direccion de una ciudad y obtener su clima.**

**CLASE 53.- Inicio del proyecto – Aplicación del Clima del Mundo**

Objetivo: Recibir desde la consola el nombre de una ciudad y mostrarlo comprobando que se leyo correctamente de ‘yargs’

En consola:

* ‘cd’ 05-clima-mundo
* ‘npm init’, ‘enter’ a todo // inicializar proyecto de node, crear el ‘package.json’
* ‘npm i –save yargs’ // importar el pck ‘yargs’

En VC code:

* Crear el file: App.js
* Declarar la const ‘argv’; requiere el pck ‘yargs’, variable ‘.argv’; no define el argumento ‘comando’; define el argumento ‘options’; define que recibira de yargs el objeto ‘decripcion’, alias ‘d’, demand=true; declara un ‘console.log’ para mostrar el contenido del ‘argv.descripcion’

En consola:

* ‘node app -d “Madrid España”
* Comprobar que muestre el texto tecleado en la consola

**CLASE 54.- Lectura – City-Geo-Location**

Objetivo: Crear cuenta en la API ‘City Geo-Location lookup’

* Entrar al portal de la Rapid API, City Geo-Location Lookup con el URL:

https://rapidapi.com/dev132/api/city-geo-location-lookup

* Crear cuenta / Registrarse

**CLASE 55.- City-Geo-Location**

Objetivo: Hacer peticion de datos de una ciudad en Postman tomando el ‘URL’, el ‘Key’, y el ‘Value’ de la API ‘City Geo-Location lookup’

* Entrar a la Rapid API, ‘City Geo-Location Lookup’ (con el URL dado y con la cuenta creada previamente)
* Abrir ‘Postman’ (con el icono que se genero al instalarlo anteriormente)
* En la API ‘City Geo-Location Lookup’ seleccionar en la instrucción ‘unirest’ en el argumento ‘.get’ el URL: <https://devru-latitude-longitude-find-v1.p.rapidapi.com/latlon.php?location=New+York> y copiarlo a Postman en la ventana ‘GET’; dar click al boton azul ‘Send’; en la parte de abajo debe aparecer el mensaje de error
  + ‘Missing RapidAPI’ applicacion key’
* Para solucionarlo:
  + En la API ‘City Geo..’ en la instrucción ‘unirest’ en el argumento ‘.header’ seleccionar el texto ‘X-RapidAPI-Key’ y copiarlo a ‘Postman’ en la pestaña ‘Headers’ en la ventana ‘Key’
  + En la API ‘City Geo..’ seleccionar el valor de la ventana ‘X-RapidAPI-Key’ (ojo aparece bloqueado pero si se puede copiar) y copiarlo a ‘Postman’ en la pestaña ‘Headers’ en la ventana ‘Value’
* En ‘Postman’ dar click al boton azul ‘Send’; en la parte de abajo deben aparecer los datos de la ciudad indicada en el URL copiado a la ventana ‘Get’ ; si hay registradas varias ciudades del mundo con el nombre en la ‘URL’, deben aparecer todas abajo en el objeto ‘Results’; si no hay ninguna ciudad con ese nombre aparecera en blanco en el objeto ‘Results’
* Probar con otras ciudades

**CLASE 56.- PETICIONES HTTP – AXIOS**

Objetivo: Recibir el nombre de una ciudad desde la consola y obtener sus datos geograficos invocando a la API ‘City Geo Location lookup’ desde un ‘app.js’

Existen dos pcks para invocar la API ‘City Geo Location lookup’:

* ‘npmjs.com/package/axios’
* ‘npmjs.com/package/request’

Ambos son similares porque trabajan con peticiones a servicios ‘REST’; la diferencia es que ‘axios’ trabaja en base a ‘promesas’ y ‘request’ en base a ‘callbacks’

Usaremos ‘axios’

En consola

* ‘cd’ 05-clima-mundo
* ‘npm i axios // instalar axios - no es necesario el ‘—save’ .. lo hace por default

En VS Code

* abrir el file ‘app.js’;
* declarar la ‘const axios’ con ‘require’ al pck ‘axios’;

En la pagina ‘www.npmjs.com/package/axios’, en la seccion ‘creating an instance’

* seleccionar el codigo ‘var instance = axios.create({…}]’
* copiarlo al file ‘app.js’;
* cambiar el ‘var’ por ‘const’

En ‘postman’,

* seleccionar el URL de la ventana ‘Get’ <https://devru-latitude-longitude-find-v1.p.rapidapi.com/latlon.php?location=Mexico>, (en el URL viene la ciudad de Ottawa)
* copiarlo al file ‘app.js’, a la ‘const instance =’, en el argumento ‘baseURL: ’ (sustituirlo)

En VS Code, en ‘app.js’, en el ‘const instance’

* quitar el argumento ‘timeout:’

En ‘postman’, en la pestaña ‘headers’, en la ventana ‘key’,

* seleccionar el dato en la ventana (‘X-RapidAPI-key’) y copiarlo al ‘app.js’, en el ‘const instance =’, en el argumento ‘headers’, en el primer parametro (sustituirlo)

En ‘postman’, en la pestaña ‘headers’, en la ventana ‘Value’,

* seleccionar el valor en la ventana y copiarlo al ‘app.js’, en el ‘const instance =’, en el argumento ‘headers’, en el segundo parametro (sustituirlo).

En VS code, en ‘app.js’,

* declarar la peticion de la ejecucion de la instancia ‘instance.get()…’,
* poner las respectivas ‘.then’ en caso de exito y ‘.catch’ en caso de no exito

En consola, ejecutar el ‘app.js’,

* teclear ‘node app.js -d ottawa’; deben aparecer todos los datos correspondientes a la ciudad indicada, poner atencion al ‘status: 200’, indica que no hubo error;

En VS Code,

En app.js

* En ‘instance.get()’,
  + en el ‘.then’ poner ‘resp.data.Results[0]’ // ‘resp’: es toda la informacion que se obtuvo con el ‘instance’; ‘data’: es el bloque de donde estan los datos de las ciudades; ‘Results’: es la seccion donde estan los datos especificos de las ciudades; ‘[0]’ para que muestre solo la primera ciudad (el primer dato en ‘Results’), si no se pone el ‘[0]’ se mostraran todas las ciudades que estan registradas con el nombre de la ciudad recibida de la consola en ‘direccion’
  + en el ‘.catch’ poner el mensaje de error
* Declarar la ‘const encodedUlr =encodedURI(argv.direccion)’ antes del const instance; // para tomar la direccion almacenada en ‘argv.’ en ‘yargs’ que viene de la consola y convertirla a codigo URI
* poner la constante ‘encodedUlr’ en el ‘const instance =’ al final del argumento ‘baseURL:’ con ${ encodedUlr }; // para que invoque la ciudad recibida de la consola

En consola,

* ejecutar el ‘app.js’, teclear ‘node app.js -d “${ ciudad }”; checar que aparezcan los dato de la ciudad almacenados en ‘data.Results[0]’
* probar con varias ciudades

**CLASE 57.- OPTIMIZACION DE CODIGO**

Objetivo: Optimizar el codigo para que la peticion de datos con la API ‘City Geo Location lookup’ quede independiente en el file ‘lugar.js’ en una funcion, y pueda ser llamado desde cualquier file ‘\*.js’ el cual solo debe enviarle el nombre de la ciudad cuya informacion se quiere obtener, sabiendo que el nombre de la ciudad fue recibido desde la consola

En VS Code, en ’05-CLIMA-MUNDO’

* Crear un folder ‘lugar’ y en este crear el file ‘lugar.js’
* Mover del file ‘app.js’ al file ‘lugar.js’ :
* la constante ‘const axios = require(..) ‘
* la constante ‘const encodedUlr =’
* la constante ‘const instance = axios.create(…)’
* el llamado a la constante ‘const instance=’ con la instrucción ‘instance.get() ..’

En el file ‘lugar.js’

* Crear la funcion de flecha ‘const getLugarLatLng = ‘, que recibira el argumento ‘dir’ del app.js llamador; este argumento sera recibido como ‘dir’ ya que ‘direccion’ es declarado posteriormente como una constante a devolver al file llamador
  + mover a esta funcion:
    - la constante ‘const encodedUlr =encodeURI(dir)’
    - la constante ‘const instance = axios.create(…)’
    - el llamado a la constante ‘const instance=’ con la instrucción ‘instance.get() ..’
  + Declarar la constante ‘const resp = await instance.get()’ que sera la que invoque a ‘instance’;
  + Poner un ‘if’ para verificar si lo recibido en la constante ‘resp’ no contiene ningun dato, y de ser verdadero mandar con ‘throw’ un mensaje de error, y terminar
  + Declarar la constante ‘const data = resp.data.Results[0]’ // se le asignan los datos recibidos de la ciudad
  + Declarar la constante ‘const direccion = data.name” // argumento ‘name’ = nombre ciudad
  + Declarar la constante ‘const lat = data.lat’ // argumento lat’ - latitud
  + Declarar la constante ‘const lng = data.lon’ // argumento lon’ – longitud
  + poner en esta funcion el ‘return’ {‘direccion, ‘lat’, ‘lng’]
* Poner la instrucción ‘module.exports = { getLugarLatLng}’ // para compartir esta funcion con el ‘app.js’ que lo llame

En el file ‘app.js’

* Declarar la constante ‘const lugar = require ‘./lugar/lugar’) // para poder invocar cualquier funcion en el
* Poner el llamado ‘lugar.getLugarLatLng(argv.direccion); poner el ‘.then’ para en caso de éxito publicar el valor recibido

En consola

* ‘cd’ 05-clima-mundo
* ejecutar el ‘app.js’, teclear ‘node app.js -d “New York”; checar que aparezcan los datos de la ciudad: ‘direccion’, ‘lat’, ‘lng’
* probar con otras ciudades

**CLASE 58.- OPEN WEATHER MAP**

Objetivo: Recibir el nombre de una ciudad desde la consola en una ‘app.js’, obtener sus datos geograficos (latitud y longitud) invocando a la API ‘City Geo Location lookup’ y obtener su clima invocando la API ‘OpenWeatherMap.org’

Entrar a la pagina web ‘OpenWeatherMap.org’ (desde el explorador de internet)

* Crear cuenta con la opcion ‘Sign up’
* Ingresar a la pagina con la opcion ‘Sign in’
* Entrar a la opcion ‘API keys’ dando click a la opcion ‘API keys’ en el segundo menu horizontal; abajo se muestran las ‘API keys’ registradas; en la columna ‘Key’ se muestran los valores de las ‘Key’ y en la columna ‘name’ el nombre de las mismas; la primera es una ‘API key’ por default y ademas se pueden crear mas; esta API key sera necesaria para hacer las peticiones
* Entrar a la opcion ‘API’ dando click a la opcion ‘API’ en el primer menu horizontal; se muestran tres opciones ‘Current weather data’, ‘5 day / 3 hour forecast’, y ’16 day / daily forecast’
* Entrar a la opcion ‘Current weather data’ dando click en el boton ‘API doc’; en esta pantalla, en la parte derecha se muestran todas las opciones para obtener clima
* Localizar e ir la seccion ‘By geographic coordinates’; aquí muestra el URL para el llamado de la API en la opcion ‘API call’ (este URL no tiene datos numericos en los argumentos ‘lat’ y ‘lng’)
* En la opcion ‘Examples of API calls’ seleccionar el URL y copiarlo a Postman en el recuadro ‘GET’ (este URL si contiene datos numericos en los argumentos ‘lat’ y ‘lon’, los cuales son latitud y longitud de alguna ciudad del mundo)

En Postman

* Dar click al boton ‘Send’; en la ventana de abajo aparece el mensaje de error ‘Invalid API key’ porque no se le esta mandando la ‘key’
* En la parte superior, en la columa ‘KEY’, en el recuadro disponible despues de ‘lat’ y ‘lon’ teclar ‘appid’

En ‘OpenWeatherMap’,

* en la opcion ‘API keys’, en la columna ‘Key’ con el name ‘Default’, seleccionar el URL ‘98a0b16a1acf8477412ed269e86e95cf’ y copiarlo a ‘Postman’ en el recuadro ‘VALUE’ en la misma linea donde se tecleo ‘appid’

En Postman

* Dar click al boton ‘Send’
* En la ventana inferior aparecera una respuesta con los datos correspondientes a la ciudad cuyas coordenadas fueron proporcionadas en el URL en ‘GET’; uno de esos objetos es el clima ‘Weather’ y otro es el ‘main’ donde como primer elemento esta la temperatura en grados kelvin;
* para obtener la temperatura en grados celcius, en la parte superior, en la columna ‘KEY’ en el siguiente recuadro disponible debajo de ‘appid’ teclear ‘units’, y en la columna ‘VALUE’ en la misma linea teclear ‘metric’
* Dar click en el boton ‘Send’
* Ahora en la ventana inferior apareceran los datos de la ciudad pero la temperatura se mostrara en grados celcius

NOTA: El URL que esta en el recuadro ‘GET’ es el que debe llamarse desde la ‘app.js’para obtener los datos de temperaturas de una ciudad

Nuevamente usaremos el pck ‘npmjs.com/package/axios’para invocar ahora la API ‘OpenWeatherMap.org’

En VS Code,

en ’05-CLIMA-MUNDO’,

* crear un folder ‘clima’ y en este crear el file ‘clima.js’

En ‘clima.js’

* Declarar ‘const axios = require(‘axios’)’
* Crear funcion de flecha ‘const getClima = async (lat, lng) => {’ – recibe los argumentos ‘lat’ y ‘lng’
  + Declarar ‘const resp = await axios.get(`https://api.openweathermap.org/data/2.5/weather?lat=${ lat }&lon=${ lng }&appid=98a0b16a1acf8477412ed269e86e95cf&units=metric`); este URL es del recuadro ‘GET de Postman; lo señalado en amarillo son los argumentos recibidos en la declaracion de la funcion
  + Declarar el ‘return resp.data.main.temp’ que sera el dato que envie de retorno la funcion
* Declarar ‘module.exports’ para compartir la funcion ‘getClima’;

En ‘app.js’

* Declarar const clima = require(./clima/clima’)
* Crear la instrucción para llamar a la funcion ‘clima.getClima( 40.750000, -74.000000 )’ – (por el momento teclear las coordenadas directamente)
  + Declarar el ‘.then(console.log)’
  + Declarar el ‘.catch(console.log)’

En consola

* ‘cd’ 05-clima-mundo
* ejecutar el ‘app.js’, teclear ‘node app.js -d “New York”; debe publicarse la temperatura de la ciudad
* probar con otras ciudades

**CLASE 59.- CONECTANDO AMBOS SERVICIOS (GEO Y WEATHER)**

Objetivo: Recibir el nombre de una ciudad desde la consola en una ‘app.js’, y en una sola funcion obtener sus datos geograficos (latitud y longitud) invocando a la API ‘City Geo Location lookup’ y obtener su clima invocando la API ‘OpenWeatherMap.org’

En VS Code

En file ‘app.js’

* Declara la funcion de flecha ‘getInfo’; recibe el argumento ‘direccion’; tiene una instsruccion ‘try-catch’;
  + En el ‘try’
    - declara la constante ‘coord’ a la cual le asigna las coordenadas que se reciben al invocar la funcion ‘getLugarLatLng’ ubicada en ‘lugar.js’ (si la obtencion de coordenadas fue exitosa)
    - declara la constante ‘temp’ a la cual le asigna la temperatura que se reciba al invocar la funcion ‘getClima’ ubicada en ‘clima.js’ (si la obtencion de temperatura fue exitosa)
    - declara el ‘return’ para devolver el nombre de la ciudad y su temperatura
  + En el ‘catch’
    - Declara el ‘return’ para devolver un mensaje de no éxito en la busqueda del clima de la ciudad
* Crea el llamado a la funcion ‘getInfo’; le envia el nombre de la ciudad recibido de la consola que esta almacenado en el pck ‘yargs’ en el objeto ‘argv.direccion’;
  + Declara el ‘.then’ para publicar la ciudad y la temperatura en caso de éxito
  + Declara el ‘.catch’ para publicar un msg en caso de no exito

En consola

* ‘cd’ 05-clima-mundo
* ejecutar el ‘app.js’, teclear ‘node app.js -d “New York”; debe publicarse la temperatura de la ciudad
* probar con otras ciudades