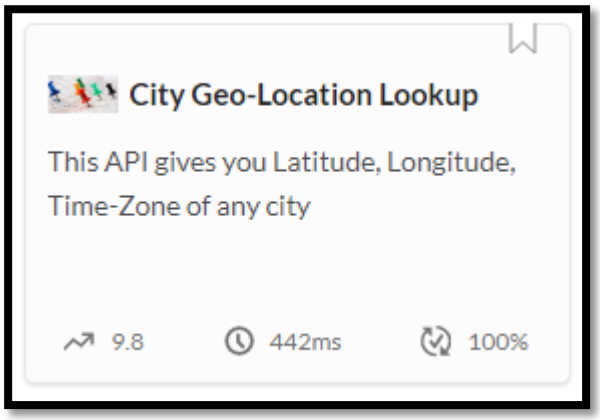
UNIVERSIDAD POLITÉCNICA SALESIANA

CARRERA: COMPUTACIÓN **ASIGNATURA:** PLATAFORMAS WEB

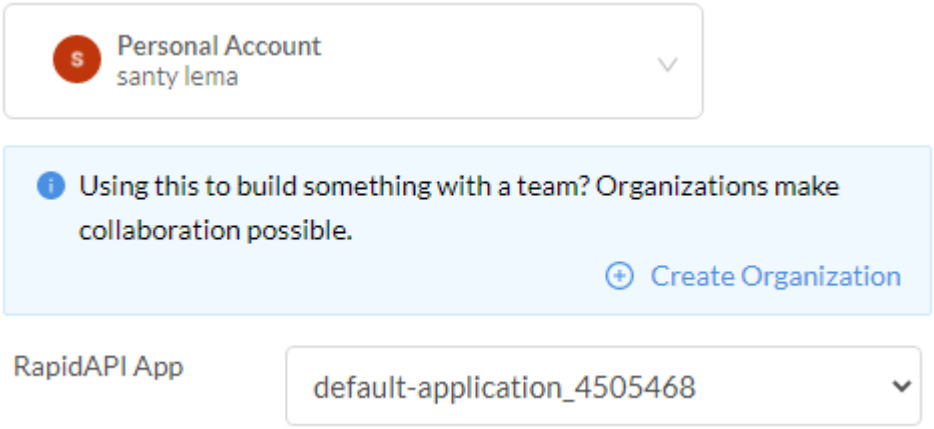
PRACTICA: 04 **TÍTULO:** Aplicación del clima

Nombre: Eduardo Santiago Quisupangui Lema

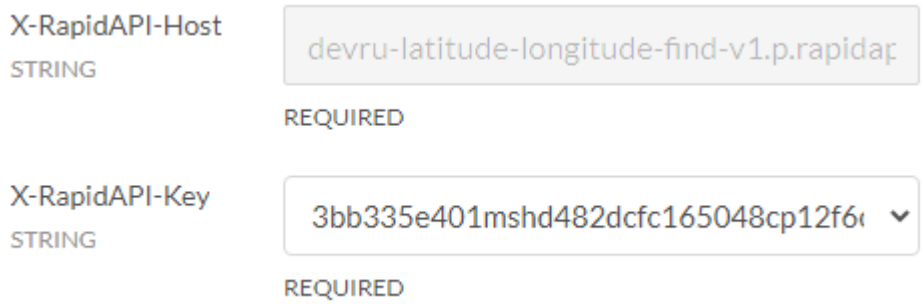
ACTIVIDADES A DESARROLLAR:
1. Creamos un proyecto para que contenga el código del clima
2. ingresamos al siguiente link para poder registrarnos:
https://rapidapi.com/
3.Al registrarse nos dirigimos a buscar el siguiente nombre:
City Geo-Location Lookup API Documentation
Elegimos esta opción:



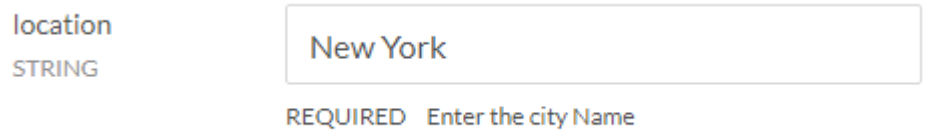
Nos redijera a la siguiente pantalla que podremos ver la altitud y longitud dependiendo el nombre de la ciudad.
Ejemplo:



Lo que es más necesario es el host y el código que nos da la aplicación, con eso podemos ingresar a las diferentes APIS del sistema.



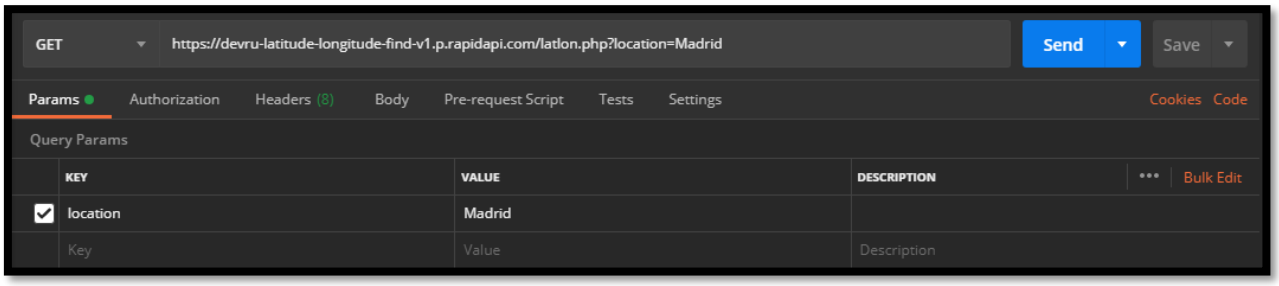
En eso nos da la opción de que país deseamos la información.



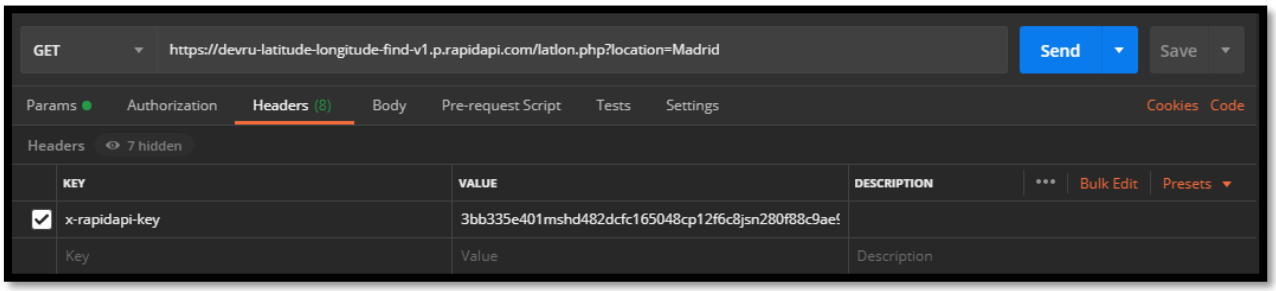
Ejemplo en código que nos proporciona la API.

```
var unirest = require("unirest");

var req = unirest("GET", "https://devru-latitude-longitude-find-
v1.p.rapidapi.com/latlon.php");

req.query({
	"location": "New York"
});

req.headers({
	"x-rapidapi-host": "devru-latitude-longitude-find-v1.p.rapidapi.com",
	"x-rapidapi-key": "3bb335e401mshd482dcfc165048cp12f6c8jsn280f88c9ae93",
	"useQueryString": true
});


req.end(function (res) {
	if (res.error) throw new Error(res.error);

	console.log(res.body);
})
```

4.Utilizamos Postman para probar la página anteriormente mencionada ya que nos votaba resultados nulos.
Para colocar los requerido seguimos los siguientes pasos

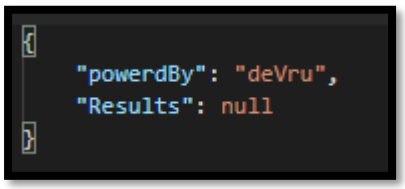


- Colocamos en petición GET
- Ingresamos la dirección url o host
- Nos dirigimos a params
- Colocamos en key
- Ingresamos location y el nombre del país que deseamos consultar



- Nos dirigimos a headers
- Lococamos en key (`x-rapidapi-key`)
- En value colocamos la clave que nos proporciona la aplicación

Resultado nos muestra al dar clic a send nos indica igual que la pagina consultada que el servicio esta caído.



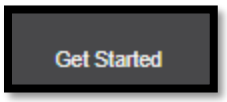
5. Otra opción para poder consultar el clima de diferentes países es el siguiente.

Open weather map
https://openweathermap.org/
Nos registramos en la página mencionada para poder ingresar a la aplicación

Al presionar en la opción de API key nos proporciona una contraseña como esta:

```
56c9e24af341c0a3036daaf10f228fd7
```

Para redirigirse a la aplicación que se va a utilizar seguimos los siguientes pasos:
Hacemos clic en:



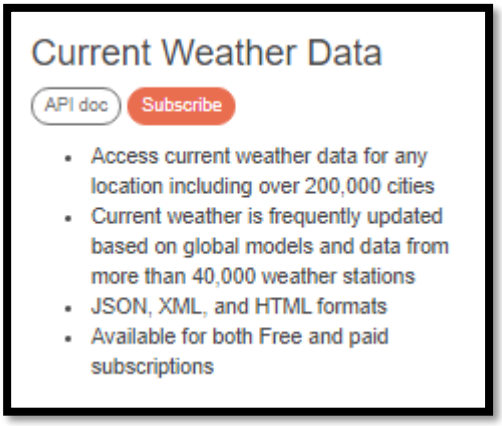
Nos dirigirá a otra pantalla:
Buscamos en la documentación lo siguiente:

II. API Documentation

The OpenWeatherMap API documentation structure is based on our product list. You can choose a product you are interested in and find detailed technical instructions clicking on the button "API doc" opposite the product name.
A brief service description you can find below product name.
On the documentation page, you can find out what types of requests are available, a list of parameters, examples, and other useful information.
Please remember that all Examples of API calls that listed on the documentation page are just samples and do not have any connection to the real API service.

Damos clic en The OpenWeatherMap API documentation

Nos abrirá otra página que podremos elegir la API que vamos a utilizar:



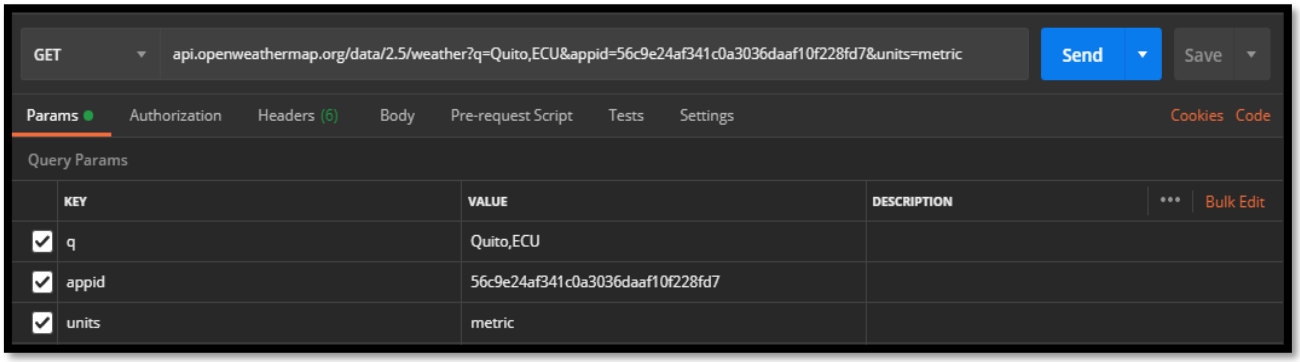
Descripción:

Puede llamar por nombre de ciudad o nombre de ciudad, código de estado y código de país. API responde con una lista de parámetros meteorológicos que coinciden con una solicitud de búsqueda.

Llamada a la API:

`api.openweathermap.org/data/2.5/weather?q={city name}&appid={your api key}`

6.Nos redirigimos a la aplicación del Postman:



- Seguimos los siguientes pasos para probar la API:
- Elegimos la petición GET
- Copiamos y pegamos la url que nos proporcionó la página
- En params en la columna de key
- Ingresamos q para ingresar la ciudad
- Ingresamos appid para el código
- Ingresamos unit para la métrica
- En values
- Ingresamos la ciudad y el código del país
- Pegamos la clave proporcionan para la página
- Y la métrica de cálculo en grados centígrados

Al dar clic a send nos ejecutara e indicara toda la información pedida con los parámetros indicados anteriormente.
Ejemplo:

7.Regresar al proyecto creado anteriormente
Para Instalar la librería necesaria para hacer peticiones http
Ingresamos a visual studio code
En la consola colocamos lo siguiente

Npm i –save axios

8.Codigo para extraer valores del clima con las credenciales generadas por la página de Open weather map.

9.Creamos una carpeta llamada controlador y un script llamado clima.js e ingresamos el siguiente código:

```
//importar la librería instalada de axios
const axios = require('axios');

//función para obtener el clima
//se realiza una funcion async para el retardo que pueda tener al llamar una
 petición
const getClima = async(ciudad) => { //parametro la ciudad que necesite consu
ltar
        const ciudadURL = encodeURI(ciudad); //para codificar la paralbra co
mo Buenos Aires para hacer la peticion get
        //colocamos la url para hacer la petición de la ciudad
        const resp = await axios.get(`https://api.openweathermap.org/data/2.
5/weather?q=${ ciudadURL }&appid=56c9e24af341c0a3036daaf10f228fd7&units=metr
ic`)
            //.data.main.temp.pressure.humidity   precision , humedad
        return resp.data.main.temp; //me debuelve la temperatura de la ciuda
d
    }
    //exportamos la función del clima
module.exports = {
    getClima
}
```

10.Creamos otro script llamado app.js con el siguiente código:

```
//se crear la opcion con yargs que en esta ocacion colocamos c para
//que así colocar -c y el nombre de la cuidad
const clima = require('./controlador/clima'); //para poder llamar al clima
const argv = require('yargs').options({
    ciudad: {
        alias: 'c',
        desc: "nombre de la ciudad",
        demand: true
```

```
    }
}).argv;

console.log(argv.ciudad);

//se crea otra función para poder mostrar la información
const getInformacion = async(ciudad) => {
        try {
            const temp = await clima.getClima(argv.ciudad);
            return `el clima de ${ ciudad } es de ${ temp } grados `; //mues
tra la ciudad y la temperatura
        } catch (e) {
            return `no se pudo obtener el clima de la ${ ciudad }`; //al ing
resar una cidad incorrecta

        }
    }
    //imprimir los resultados
getInformacion(argv.ciudad)
    .then(console.log)
    .catch(console.log)
```

Resultados:

```
PS C:\Users\Santy\OneDrive\Documents\PLATAFORMAS-WEB\05-clima\05-clima> node .\app.js -c "Quito"
Quito
el clima de Quito es de 21 grados
```

```
PS C:\Users\Santy\OneDrive\Documents\PLATAFORMAS-WEB\05-clima\05-clima> node .\app.js -c "Buenos Aires"
el clima de Buenos Aires es de 20.01 grados
```

Deber:

Se realizó el aumento de las opciones –o h para la humedad y –o p para la presión atmosférica
Se llamó a los parámetros de temperatura, de la presión y la humedad.
Como se indica la imagen:

```
return [resp.data.main.temp, resp.data.main.pressure, resp.data.main.humidity];
```

Se aumentó una opción más:

```
    ciudad: {
        alias: 'c',
        desc: "nombre de la ciudad",
        demand: true
    },
    op: {
        alias: 'o',
        desc: "opciones de presion y humedad",

    },
    humedad: {
        alias: 'h',
        desc: "humedad de la ciudad",

    }
}).argv;
```

A continuación, se presentará el código implementado para la presión y la humedad.
Carpeta llamada controlador y un script llamado clima.js

```
//importar la librería instalada de axios
const axios = require('axios');

//función para obtener el clima
//se realiza una funcion async para el retardo que pueda tener al llamar una
 peticion
const getClima = async(ciudad) => { //parametro la ciudad que necesite consu
ltar
    const ciudadURL = encodeURI(ciudad); //para codificar la paralbra como B
uenos Aires para hacer la peticion get
    //colocamos la url para hacer la petición de la ciudad
    const resp = await axios.get(`https://api.openweathermap.org/data/2.5/we
ather?q=${ ciudadURL }&appid=56c9e24af341c0a3036daaf10f228fd7&units=metric`)
        //.data.main.temp.pressure.humidity   presión , humedad
        //let datos =
    return [resp.data.main.temp, resp.data.main.pressure, resp.data.main.hum
idity]; //me debuelve la temperatura de la ciudad
}


//exportamos la función del clima
module.exports = {
    getClima,
}
```

Se creó un script llamado app.js

```
//se crear la opcion con yargs que en esta ocacion colocamos c para
//que así colocar -c y el nombre de la cuidad
const colors = require('colors');
const clima = require('./controlador/clima'); //para poder llamar al clima
const argv = require('yargs').options({
    ciudad: {
        alias: 'c',
        desc: "nombre de la ciudad",
        demand: true
    },
    op: {
        alias: 'o',
        desc: "opciones de presion y humedad",

    },
    humedad: {
        alias: 'h',
        desc: "humedad de la ciudad",

    }
}).argv;

//console.log("la ciudad consultada es: ", argv.ciudad);

//se crea otra función para poder mostrar la información
const getInformacion = async(ciudad) => {
        try {
            const [temp, pressure, humidity] = await clima.getClima(argv.ciu
dad);

            if (argv.op === 'p') {
                console.log(colors.green("Presion Atmosferica"));
                return colors.blue(`El clima de ${ciudad } es de ${ temp } g
rados y una presion ${ pressure }`);
            } else {
                if (argv.op === 'h') {
                    console.log(colors.green("Humedad"));
                    return colors.red(`El clima de ${ ciudad } es de ${ temp
 } grados y una humedad ${ humidity }`);
                } else {
                    console.log(colors.green("Temperatura"));
                    return colors.yellow(`El clima de ${ ciudad } es de ${ t
emp }`);
```

```
                }
            }
            //return `el clima de ${ ciudad } es de ${ temp } grados ${ pres
sure }. ${humidity}`; //muestra la ciudad y la temperatura
        } catch (e) {
            return `no se pudo obtener el clima de la ciudad de ${ ciudad }`
; //al ingresar una cidad incorrecta

        }
    }
    //imprimir los resultados
getInformacion(argv.ciudad)
    .then(console.log)
    .catch(console.log)
```

2. GitHub
https://github.com/Eduardo-Quisupangui/F-Nodejs

RESULTADOS OBTENIDOS:

```
PS C:\Users\Santy\OneDrive\Documents\PLATAFORMAS-WEB\05-clima\05-clima> node .\app.js -c "quito"
Temperatura
El clima de quito es de 20
PS C:\Users\Santy\OneDrive\Documents\PLATAFORMAS-WEB\05-clima\05-clima> node .\app.js -c "quito" -o h
Humedad
El clima de quito es de 20 grados y una humedad 60
PS C:\Users\Santy\OneDrive\Documents\PLATAFORMAS-WEB\05-clima\05-clima> node .\app.js -c "quito" -o p
Presion Atmosferica
El clima de quito es de 20 grados y una presion 1024
```

CONCLUSIONES:

- **Al utilizar APIS gratis o de pago podemos realizar varias cosas como esta que hemos realizado, llamada a varios elementos como el clima con su temperatura, la presión y la humedad.**

- **Al ocupar una API debemos tener en cuenta la documentación que presenta y los parámetros que se debe utilizar para poder ejecutar la API.**