Universidad de Alicante

Ingeniería Robótica

Práctica 1 Comunicaciones

CAPTURAR DATOS EN LA NUBE CON INTERFACES API

Marina Villanueva Pelayo

28 de febrero de 2022

Índice

1. USO GENERAL DE APIs

Las siglas de API significan en inglés *application programming interface*, que se traduce como interfaz de programación de aplicaciones. Esto se refiere a un conjunto de protocolos que permiten que un software se comunique con otro. Son de gran utilidad ya que permiten el acceso a los desarrolladores de otros programas a ciertas partes de su biblioteca para llevar a cabo determinadas acciones.

Las APIs se usan continuamente en el mundo digital. Algunas sirven para consultar el tiempo, pagar en plataformas como Paypal, trabajar con GitHub...

En este proyecto se van a trabajar con dos APIs concretas: una proporcionada por Openweather, relacionada con el tiempo meteorológico; y otra proporcionada por la REE (red eléctrica española).

2. CAPTURAR DATOS DE API USANDO *PYTHON*

La primera API que se va a usar va a ser la de Openweather. Openweather es un servicio online que proporciona información meteorológica a nivel global, incluyendo datos meteorológicos actuales, pronósticos, predicciones e históricos para cualquier ubicación geográfica.

Con la aplicación de Google Colab se ha ejecutado un primer código de prueba que muestra información general del tiempo actual en un determinado lugar, que en este caso se ha escogido Alicante:

```
# Obtener datos
import datetime
import json
import urllib.request

def time_converter(time):
    converted_time = datetime.datetime.fromtimestamp(
        int(time)
    ).strftime('%I:%M %p')
    return converted_time

def url_builder1(city):
    user_api = 'fb03fdadda480e9cf789a32cbf2716d9'  # Obtain yours form: http://openweathermap.org/
    unit = 'metric'  #imperial (Fahrenheit), metric (Celsius), and the default is Kelvin.
    api = 'http://api.openweathermap.org/data/2.5/weather?q='
    full_api_url = api + str(city) + '&mode=json&units=' + unit + '&APPID=' + user_api
    print(full_api_url)
    return full_api_url

def data_fetch(full_api_url):
    url = urllib.request.urlopen(full_api_url)
    output = url.read().decode('utf-8')
    raw_api_dict = json.loads(output)
    url.close()
    #print(raw_api_dict)
    return raw_api_dict

if __name__ == '__main__':
    try:
        now=datetime.datetime.now()
        datos=data_fetch(url_builder1('Alicante,ES'))
        print(datos)
        print(now)
        for key in datos['main']:
          print(key, ":", datos['main'][key])
        for key in datos['wind']:
          print(key, ":", datos['wind'][key])
        for key in datos['clouds']:
          print(key, ":", datos['clouds'][key])
        for key in datos['sys']:
          print(key, ":", datos['sys'][key])
        print(time_converter(1612681284))
        print(time_converter(datos['sys']['sunrise']))
        print(time_converter(datos['sys']['sunset']))
```

```
    except IOError:
        print('error')
```

El resultado de este código es el siguiente: una información general a cerca de la temperatura, viento, humedad, atardecer y amanecer... del momento en el que se ejecuta el código, es decir, funciona en tiempo real.

```
http://api.openweathermap.org/data/2.5/weather?q=Alicante,ES&mode=json&units=metric&APPID=fb03fdadda480e9cf789a32cbf2716d9
{'coord': {'lon': -0.4815, 'lat': 38.3452}, 'weather': [{'id': 801, 'main': 'Clouds', 'description': 'few clouds', 'icon': '02d'}], 'base': 'stations', 'ma
2022-02-10 13:05:38.684860
temp : 14.52
feels_like : 14.01
temp_min : 12.62
temp_max : 16.48
pressure : 1028
humidity : 76
speed : 1.54
deg : 130
all : 20
type : 1
id : 6391
country : ES
sunrise : 1644476311
sunset : 1644514436
07:01 AM
06:58 AM
05:33 PM
```

.

Como ejercicio de ampliación se ha ideado una aplicación de aviso meteorológico en carreteras. El código lee la información de Openweather y mostraría la información en un cartel luminoso en la carretera. En este caso se ha dejado de lado el hecho de sacar la información a otra pantalla, y simplemente se muestra por terminal cuando la situación pueda ser relevante para el viaje, como temperaturas demasiado altas, lluvia muy intensa o gran cantidad de nubes. Además, no tiene sentido que el código se ejecute solo una vez, por lo que se ha añadido que cada 60s se vuelva a ejecutar para volver a comprobar la información y actualizar la salida por pantalla:

```
# Obtener datos
import datetime
import json
import urllib.request

def time_converter(time):
    converted_time = datetime.datetime.fromtimestamp(
        int(time)
    ).strftime('%I:%M %p')
    return converted_time

def url_builder1(city):
    user_api = 'fb03fdadda480e9cf789a32cbf2716d9'  # Obtain yours form: http://openweathermap.org/
    unit = 'metric'  # For Fahrenheit use imperial, for Celsius use metric, and the default is
        Kelvin.
    api = 'http://api.openweathermap.org/data/2.5/weather?q='      # Search for your city ID here:
        http://bulk.openweathermap.org/sample/city.list.json.gz
    full_api_url = api + str(city) + '&mode=json&units=' + unit + '&APPID=' + user_api
    print(full_api_url)
    return full_api_url

def data_fetch(full_api_url):
    url = urllib.request.urlopen(full_api_url)
    output = url.read().decode('utf-8')
    raw_api_dict = json.loads(output)
    url.close()
    #print(raw_api_dict)
    return raw_api_dict

def funcionRepetida():
    if __name__ == '__main__':
        try:
            city='Londres'
            msg1 = 'Informaci n en las carreteras de ' + city
            now=datetime.datetime.now()
            datos=data_fetch(url_builder1(city))
            print(msg1)
            if datos['main']['temp'] > 35:
              print('RIESGO DE ALTAS TEMPERATURAS')
            if datos['wind']['speed'] > 50:
              print('ATENCI N FUERTES RACHAS DE VIENTO')
```

```
40              if datos['clouds']['all'] > 50:
41                print('EXTREME LA PRECAUCION. AVISO POR NUBES')
42          except IOError:
43              print('error')
44    time.sleep(60)
45
46  while True:
47    funcionRepetida()
```

Por ejemplo, en caso de mostrar la información en carreteras londinenses, el resultado actualmente sería un aviso por nubes:

```
http://api.openweathermap.org/data/2.5/weather?q=Londres&mode=json&units=metric&APPID=fb03fdadda480e9cf789a32cbf2716d9
Información de las carreteras de Londres
EXTREME LA PRECAUCIÓN. AVISO POR NUBES
```

.

La información que ofrece Openweather es:

```
▾ clouds:
    all:            75
  dt:               1646217875
```

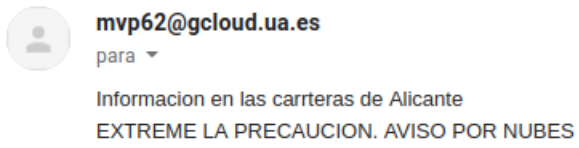
.

.

Otra opción que se puede añadir a este código es la recibir esta información por email de cara a realizar un viaje en las próximas horas y querer tenerla guardada. En este caso se ha añadido este extra:

```
1 # Enviar email
2 server = smtplib.SMTP_SSL("smtp.gmail.com", 465)
3 server.login("mvp62@gcloud.ua.es", "contrasena")
4 msg = msg1 + msg2
5 server.sendmail("mvp62@gcloud.ua.es", "marinavillanuevape@gmail.com", msg)
6
7 server.quit()
```

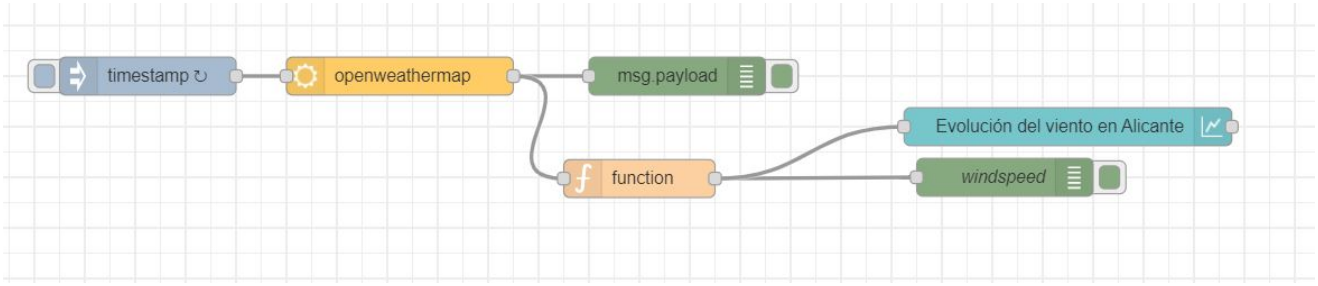
Como se puede comprobar, el email se ha enviado y recibido correctamente:



3. CAPTURAR DATOS DE API USANDO NODERED

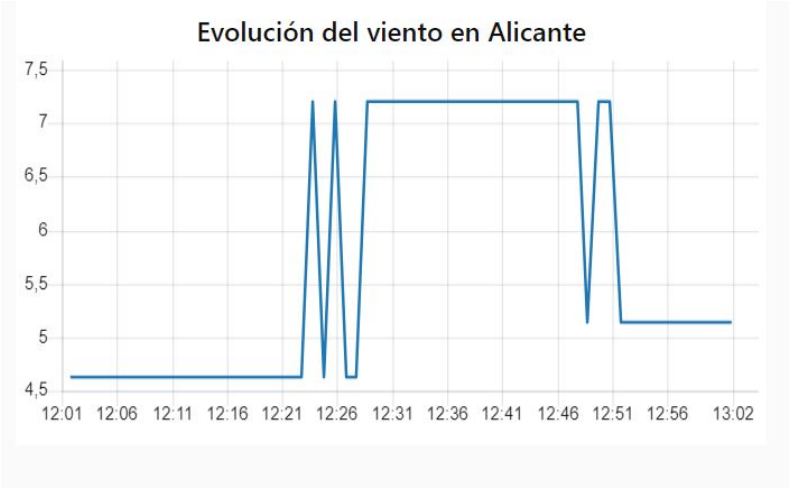
En este caso se va a utilizar Nodered, una herramienta de programación visual que trabaja con nodos que se conectan entre ellos. Además permite usar una versión online con FRED o descargar el programa en nuestro PC.

En primer lugar y como continuación a lo aprendido en clase, se ha diseñado un flujo que muestra la evolución en tiempo real de la velocidad del viento desde el momento en el que se ejecuta el programa. Para ello se accede a la variable de *windspeed* cada 1 minuto, y se pasan los datos a un gráfico de líneas.




```
1 // Codigo de la funcion
2 variable = msg.payload
3 msg.payload = variable.windspeed
4
5 return msg;
```

Para poder obtener un resultado rápido se han obtenido valores durante una hora, sin embargo se puede modificar el rango desde segundos hasta días.



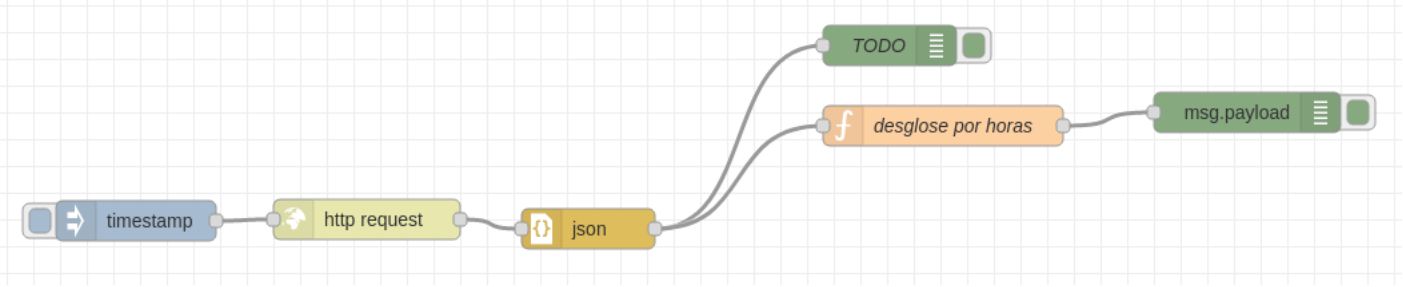
Como se puede observar, ha habido rachas de viento con velocidades desde los 4.6 m/s hasta los 7.4 m/s.

Otra ampliación que se ha llevado a cabo usando la API de Openweather ha sido la de añadir al dashboard una entrada de texto que permite introducir desde ahí la ciudad que se desea:

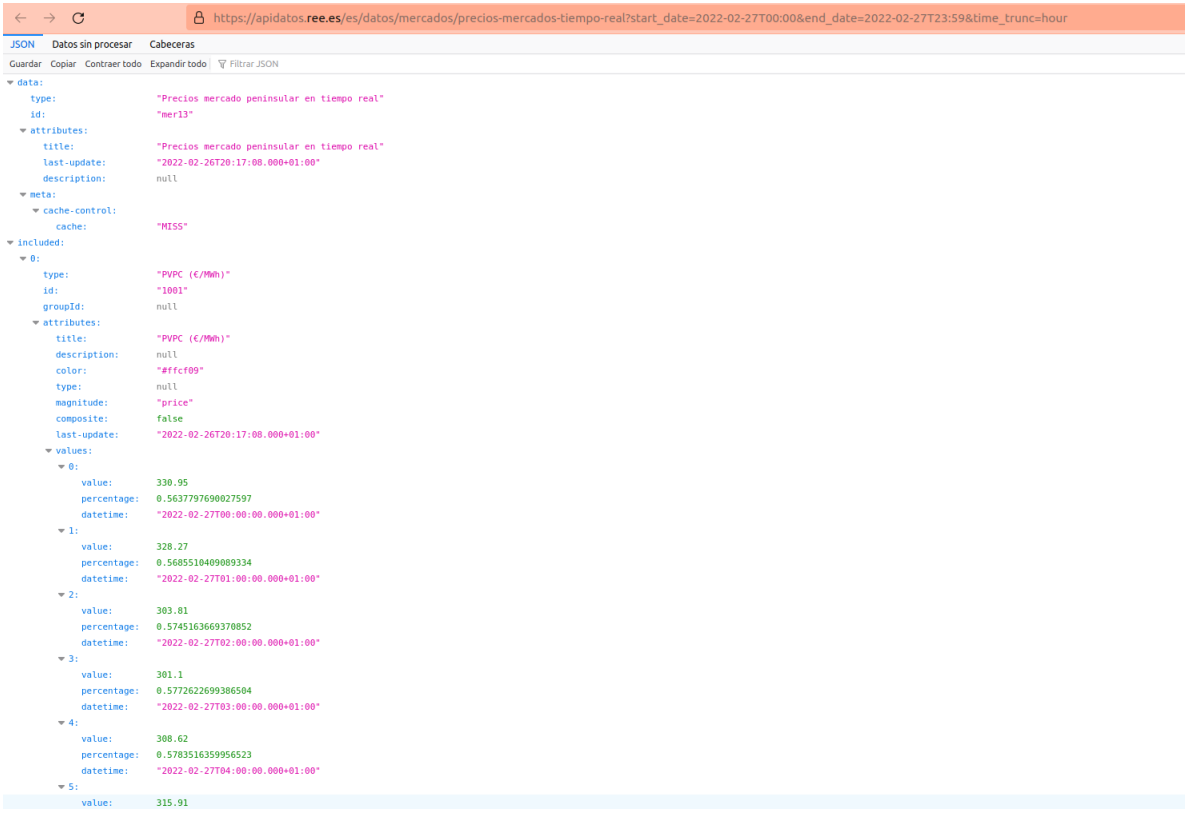
El diagrama de flujo que que hace esto posible es el siguiente:

Para ello basta con guardar la entrada leida en *msg.location.city*, y conectarlo al nodo de Openweather.

Para seguir ampliando conocimientos, a continuación se va a utilizar una API de la de Red Eléctrica Española. Esta ofrece información eléctrica de la demanda, la generación, el intercambios entre países, precios... La escogida muestra en concreto información acerca del precio del mercado peninsular en tiempo real. El flujo de Nodered básico es el siguiente:

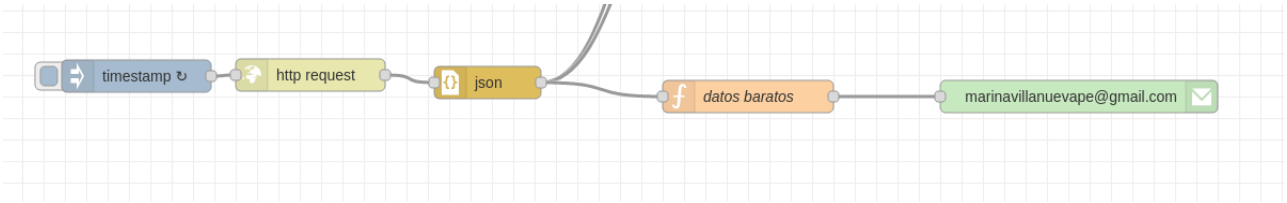


Con el primer debug (nodo verde) se obtiene toda la información del dia:



El segundo debug muestra el resultado de la función anterior, cuyo trabajo es acceder al vector que ofrece la información de cada hora:

Pudiendo acceder a toda esta información, y debido al constante incremento en el precio de la electricidad, se ha ideado una aplicación que notifique a los usuarios de la REE mediante un email de cual va a ser la hora más económica para la electricidad del día siguiente, con el fin de poder ahorrar un poco en sus facturas. El flujo sería el siguiente:

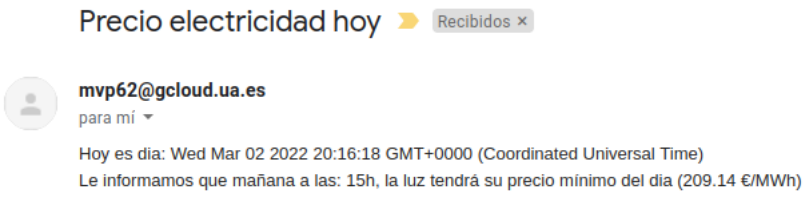


Como la información que ofrece la API es diaria, se ha configurado que el primer nodo repita el flujo cada x tiempo establecido. En este caso es todos los días a las 21:00h.

En el nodo de la función *datos baratos*, se recorre toda la información de la API y se obtiene el precio mínimo y la hora a la que la electricidad está más barata. Luego se guarda en un mensaje que se envía al correo electrónico del nodo final.

```
var variable = msg.payload;

var message = "";
var d = new Date();

var minimo = variable.included[1].attributes.values[0].value; // guarda el precio de la primera hora
// En caso de que haya otro precio menor, se actualizara
var hora = 0;
var precio = 0; // guarda el precio de cada hora

for(var i = 0; i < 24; i++){ //recorre el vector con la informaci n de las 24 horas del dia
    precio = variable.included[1].attributes.values[i].value;

    if(precio < minimo){
        minimo = precio;
        hora = i;
    }
}

msg.topic = "Precio electricidad hoy"; // asunto del email
message = "Hoy es dia: " + d;
message += "\nLe informamos que manana a las: " + hora + "h, la luz tendra su precio minimo del dia (" + minimo + " E/MWh)"

msg.payload = message;

return msg;
```

Como se puede comprobar, el email se ha enviado y recibido correctamente:



Para completar el email, lo ideal sería adjuntar una gráfica con la evolución temporal a lo largo del día ya que aunque se informe a los consumidores de cual es la hora más económica, también se considera útil tener un conocimiento del precio durante el resto del día. Sin embargo esto no es fácil ya que ninguna de las diferentes APIs que proporciona la REE ofrecen la información del precio en el momento de solicitud, si no que devuelve 24 vectores con información por horas. Por lo tanto esta parte del proyecto se queda pendiente para intentarlo cuando se disponga de más conocimiento.

4. CONCLUSIÓN

Esta práctica ha resultado de gran utilidad a la hora de ampliar nuestros conocimientos ya que hasta el momento la única forma de comunicación entre softwares que se había aprendido era la serial. Además con las ideas de ampliación a lo visto en clase se ha investigado como enviar información a distintas redes sociales ya sea email como se ha reflejado en esta práctica, o Twitter, Telegram... como han realizado otros compañeros.

Aunque en esta memoria todo el código se puede copiar y probar (no son imágenes, es texto), se adjunta también el enlace de Github con algunas evidencias en orden temporal: https://github.com/mariinaVillanueva/comunicaciones_p1

Referencias

1. https://openweathermap.org/
2. https://www.ree.es/es/apidatos
3. https://keepcoding.io/blog/que-es-una-api-y-para-que-sirve/
4. https://www.youtube.com/watch?v=Y_tnWTjTfzY
5. https://www.youtube.com/watch?v=rokLM4ZsMJQt=468s
6. https://www.iteramos.com/pregunta/53398/cual-es-la-mejor-manera-de-ejecutar-repetidamente-una-funcion-cada-x-segundos-en-python
7. https://www.youtube.com/watch?v=y5bEJAiV92s