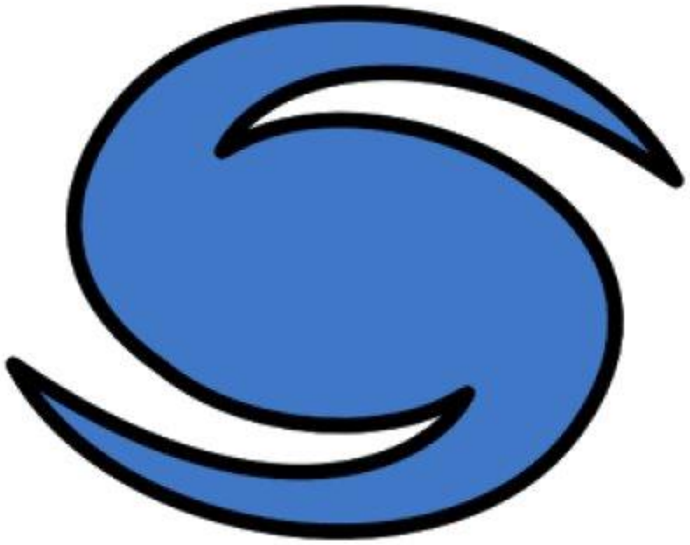
DOCUMENTACIÓN PROYECTO ANYPOINT



Forecast Chiclana

Francisco Jesús Díaz Zájara

Antonio Jesús Morales Rodríguez

1) Introducción

Nuestra aplicación utiliza la página de Winguru para extraer la información del pronóstico del tiempo en una zona determinada (en este caso de la zona de Sancti Petri en la localidad de Chiclana de la Frontera) y realiza una publicación de dicha información en la cuenta de Twitter '*Forecast Chiclana*' cada 3 horas. La aplicación nos permitirá conocer datos como la zona del pronóstico del tiempo, la dirección y la velocidad del viento, la dirección, el período y el tamaño de las olas, así como la temperatura media de la zona.

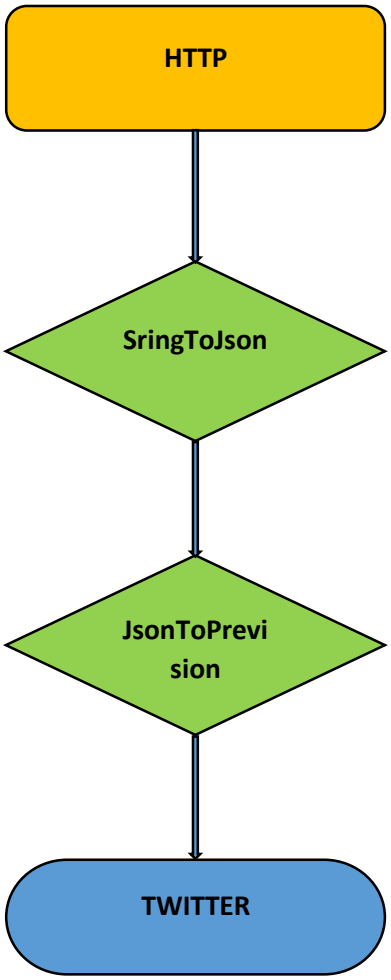
2) Descripción de la fuente de información utilizada (flujos de una plataforma de Internet de las Cosas):

La fuente de información que hemos utilizado para extraer los datos es la página de *Windguru*, en concreto de la siguiente URL: www.windguru.cz/38931

- *Formato de los datos proporcionados por el flujo. Por ejemplo: dato (localización), tipo (string), descripción (nombre del país en el que se encuentra el sensor):*

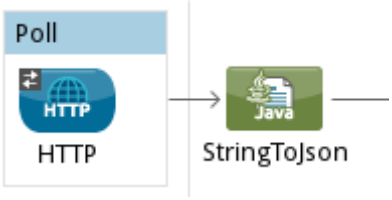
Dato	Tipo	Descripción
Nombre del Spot	String	Lugar del cual se realiza el pronóstico
Día	Int	Día del pronóstico
Hora	Int	Hora del pronóstico
Velocidad del viento	Double	Velocidad media del viento medida en nudos
Dirección del viento	Int	Dirección del viento medida en grados
Dirección de la ola	Int	Dirección de las olas medida en grados
Tamaño de la ola	Double	Tamaño de la ola en metros
Periodo de la ola	Double	Tiempo transcurrido entre cada ola
Temperatura	Int	Temperatura media de la zona en grados centígrados

3) ***Esquema del flujo AnyPoint implementado: se incluirá un gráfico del flujo y se describirá el esquema seguido (detallando aquellos elementos que se hayan implementado "a mano" como, por ejemplo, el transformador de mensajes).***



En primer lugar, nuestra aplicación se conecta a la página de Windguru a través del conector *http* de forma iterativa cada 3 horas gracias a la herramienta *Poll*.

A continuación, con nuestra clase mapeadora (*StringToJson*), extraemos el código html de la página que contiene los datos que queremos utilizar (nombre del spot, velocidad del viento, dirección del viento, tamaño de la ola, día…) y lo guardamos en un string:




```
public class StringToJson implements Callable{
@Override
    public Object onCall(MuleEventContext eventContext) throws Exception {
        Document doc = Jsoup.parse(eventContext.getMessage().getPayloadAsString());
        Element div = doc.getElementById("div_wgfcst1");
        String data = div.html();
        int inicio = data.indexOf("wg_fcst_tab_data_1 = ") + 21;
        int fin = data.indexOf(";");
        data = data.substring(inicio, fin);
```

Para luego poder pasarlo a formato JSON y devolver el contenido a través del Payload:

```
        JSONObject rawJson = new JSONObject(data);

        JSONObject finalJson = new JSONObject();
        finalJson.put("waveSize", rawJson.getJSONObject("fcst").getJSONObject("3").get("HTSGW"));
        finalJson.put("waveDir", rawJson.getJSONObject("fcst").getJSONObject("3").get("DIRPW"));
        finalJson.put("wavePeriod", rawJson.getJSONObject("fcst").getJSONObject("3").get("PERPW"));
        finalJson.put("windSpeed", rawJson.getJSONObject("fcst").getJSONObject("3").get("WINDSPD"));
        finalJson.put("windDir", rawJson.getJSONObject("fcst").getJSONObject("3").get("WINDDIR"));
        finalJson.put("spotName", rawJson.get("spot"));
        finalJson.put("dayArray", rawJson.getJSONObject("fcst").getJSONObject("3").get("hr_d"));
        finalJson.put("hourArray", rawJson.getJSONObject("fcst").getJSONObject("3").get("hr_h"));
        finalJson.put("temp", rawJson.getJSONObject("fcst").getJSONObject("3").get("TMP"));
        finalJson.put("rain", rawJson.getJSONObject("fcst").getJSONObject("3").get("APCP"));
        finalJson.put("lowCloud", rawJson.getJSONObject("fcst").getJSONObject("3").get("LCDC"));
        finalJson.put("midCloud", rawJson.getJSONObject("fcst").getJSONObject("3").get("MCDC"));
        finalJson.put("highCloud", rawJson.getJSONObject("fcst").getJSONObject("3").get("HCDC"));

        eventContext.getMessage().setPayload(finalJson);
        return eventContext.getMessage().getPayload();
    }
}
```

Tras ésto, creamos nuestra clase *Prevision,* en la cual definimos un constructor donde inicializamos todos los atributos privados de la clase con los parámetros almacenados en el objeto JSON que recibirá por parámetro el objeto Prevision cuando sea creado, como veremos más adelante:

```
public class Prevision {

    private String _spotName;
    private double _waveSize;
    private double _waveDir;
    private double _wavePeriod;
    private double _windSpeed;
    private int _windDir;
    private int _temp;
    private int _hour;
    private int _day;

    public Prevision(JSONObject inJson){

        _spotName = inJson.getString("spotName");
        JSONArray aWaveSize = (JSONArray) inJson.get("waveSize");
        JSONArray awaveDir = (JSONArray) inJson.get("waveDir");
        JSONArray aWavePeriod = (JSONArray) inJson.get("wavePeriod");
        JSONArray aWindSpeed = (JSONArray) inJson.get("windSpeed");
        JSONArray aWindDir = (JSONArray) inJson.get("windDir");
        JSONArray aTemp = (JSONArray) inJson.get("temp");
        JSONArray aHour = (JSONArray) inJson.get("hourArray");
        JSONArray aDay = (JSONArray) inJson.get("dayArray");
```

Como cada parámetro del objeto JSON (spotName, waveSize, windSpeed, windDir…) está formado por un conjunto de valores (tantos como indica la página de Windguru según las horas), utilizamos un bucle for para obtener los valores concretos de cada parámetro según la hora del sistema:

```
    //HORA DEL SISTEMA CON FORMATO
    DateFormat hourFormat = new SimpleDateFormat("HH");
    int systemHour = Integer.parseInt(hourFormat.format(new Date()));

    for(int i = 0; i < aWaveSize.length(); i++)
        if(systemHour >= aHour.getInt(i) && systemHour < aHour.getInt(i+1))
        {
            _waveSize = aWaveSize.getDouble(i+1);
            _waveDir = awaveDir.getDouble(i+1);
            _wavePeriod = aWavePeriod.getDouble(i+1);
            _windSpeed = aWindSpeed.getDouble(i+1);
            _windDir = aWindDir.getInt(i+1);
            _temp = aTemp.getInt(i+1);
            _hour = aHour.getInt(i+1);
            _day = aDay.getInt(i+1);
            break;
        }
}
```

También hemos definido el método *toString()* en el cual daremos formato a la información resultante que será publicada en Twitter:

```
public String toString()
{
    String ret;

    ret = "Nombre del Spot: " + _spotName + "\n";
    ret += "Día: " + _day + "    ";
    ret += "Hora: " + _hour + "\n";
    ret += "Tamaño de la ola: " + _waveSize + "metros.\n";
    ret += "Dirección de la ola: " + _waveDir + "segundos.\n";
    ret += "Periodo de la ola: " + _wavePeriod + "segundos.\n";
    ret += "Velocidad del viento: " + _windSpeed + "nudos\n";
    ret += "Dirección del viento: " + _windDir + "º\n";
    ret += "Temperatura media: " + _temp + "º\n";

    return ret;
}
```

Por último, utilizamos nuestra clase *JsonToPrevision* para guardar en un objeto JSON el Payload que recibimos de la clase *StringToJson* mencionada anteriormente:

y crear un objeto Prevision al que le pasamos por parámetro este objeto JSON (*inJson* en este caso) y que a su vez será enviado al Payload en curso:

```
public class JsonToPrevision implements Callable{
@Override
    public Object onCall(MuleEventContext eventContext) throws Exception {

        JSONObject inJson = (JSONObject) eventContext.getMessage().getPayload();

        Prevision prevision = new Prevision(inJson);

        eventContext.getMessage().setPayload(prevision);
        return eventContext.getMessage().getPayload();
    }
}
```

Enviándolo definitivamente a Twitter: