Práctica 4: Node-RED

Sistemas Distribuidos

Curso: 2019/2020

Índice

1. Partes de la práctica a entregar **2**

2. Normas para la realización de la práctica **2**

3. Flujo 1: Calculadora de Vectores mediante API REST **2**
- 3.1. Mejoras . . . 3
- 3.2. Enlaces de interés . . . 3

4. Flujo 2: MQTT **3**
- 4.1. Mejoras . . . 4
- 4.2. Enlaces de interés . . . 4

5. Flujo 3: Contributed Nodes **4**
- 5.1. Enlaces de interés . . . 4

6. Evaluación **4**

7. FECHA LIMITE DE ENTREGA **5**

Este documento indica los requisitos para la Práctica 4.

1. Partes de la práctica a entregar

1. Código fuente (ficheros .json de los flujos creados) (.zip)
   - Las prácticas se corregirán en Linux, para aquellos que utilicen un S.O. distinto (MacOS o Windows) se recomienda testear el código en una distribución Linux antes de entregarlo.
2. Memoria descriptiva del trabajo realizado (PDF)
   - Se debe proporcionar una descripción detallada del programa implementado así como de su funcionamiento y ejecución.
   - Cualquier mejora implementada pero no descrita en la memoria no será evaluada.

2. Normas para la realización de la práctica

- Realización por parejas (se admiten trabajos individuales, pero no de tres estudiantes).
- Los trabajos deberán ser entregados obligatoriamente antes de la fecha de entrega fijada en la actividad habilitada en el campus virtual.
- Las dos partes del trabajo son obligatorias. Si una de ellas no se realiza el trabajo se considerará no presentado.

3. Flujo 1: Calculadora de Vectores mediante API REST

Implementar en Node-RED una API REST que ofrezca, al menos, tres `endpoints` para realizar operaciones sobre vectores de enteros. Para ello se será necesario utilzar, entre otros, nodos de tipo ‘‘`http in`’’ y ‘‘`http response`’’.

Los datos se pasarán a los `endpoints` en formato JSON, y las operaciones a implementar serán:

- `/suma`: sumar dos vectores de tres posiciones. Ejemplo de interacción con el `endpoint` mediante `cURL`:

```
$ curl -X GET -d '{"a":[1,2,3],"b":[4,5,6]}'
   -H "Content-type: application/json"
   http://localhost:1880/suma

{"resultado":[5,7,9]}
```

- `/resta`: restar dos vectores de tres posiciones, en el orden vector ‘‘a’’ - vector ‘‘b’’. Ejemplo de interacción con el `endpoint` mediante `cURL`:

```
$ curl -X GET -d '{"a":[1,2,3],"b":[4,5,6]}'
   -H "Content-type: application/json"
   http://localhost:1880/resta

{"resultado":[-3,-3,-3]}
```

- `/multesc`: multiplicar un vector de tres posiciones por un escalar. Ejemplo de interacción con el `endpoint` mediante `cURL`:

```
$ curl -X GET -d '{"a":[1,2,3],"n":5}'
   -H "Content-type: application/json"
   http://localhost:1880/multesc

{"resultado":[5,10,15]}
```

3.1. Mejoras

- Añadir `endpoints` para operaciones adicionales.
- Permitir realizar las operaciones con cualquier longitud de vector.
- Utilizar un nodo de tipo ‘‘`file`’’ donde se mantenga un *log* de todas las peticiones que se reciben. Cada vez que se reciba una petición, se almacenará en el fichero una nueva línea de texto, indicando el tipo de operación solicitada, parámetros y resultado.

3.2. Enlaces de interés

- https://www.w3schools.com/js/js_arrays.asp
- https://www.w3schools.com/js/js_loop_for.asp
- https://www.w3schools.com/js/js_json_arrays.asp

4. Flujo 2: MQTT

Suscribirse a un broker MQTT mediante un nodo de tipo ‘‘`mqtt in`’’.

El broker al que os debéis suscribir está en la dirección https://test.mosquitto.org/, puerto `1883`. Éste es un servidor público y gratuito para realizar pruebas con el protocolo MQTT.

Entre los distintos eventos disponibles, deberéis suscribiros a aquel con topic `/merakimv/Q2GV-Y4R8-5Z3L/light`

El evento nos dará de forma regular la luminosidad en `luxs` que registra una cámara de seguridad. La estructura del evento (payload del mensaje) es de la forma ‘‘`{‘‘lux’’: 80.9}`’’. El payload llegará en formato `string`, deberéis utilizar un nodo de la clase `parser` para transformarlo a un `JavaScript Object` y poder trabajar fácilmente con esta información.

Mediante un nodo `function`, deberéis almacenar el valor máximo registrado de todos los eventos recibidos. Para ello será necesario utilizar el `contexto` de Node-RED.

Cada vez que se reciba un evento, se debe mostrar, por la consola de depuración, tanto el valor de luminosidad recibido como el máximo registrado hasta este momento.

4.1. Mejoras

- Investigar cómo publicar eventos en https://test.mosquitto.org/ y crear un flujo para publicar y recibir dichos eventos. **No publicar información sensible**.

4.2. Enlaces de interés

- https://cookbook.nodered.org/#mqtt
- https://test.mosquitto.org/

5. Flujo 3: Contributed Nodes

Investigar de entre los diferentes nodos contribuidos de la librería de Node-RED, aquel que más os llame la atención.

Describir el porqué en la memoria y mostrar un ejemplo de flujo sencillo.

5.1. Enlaces de interés

- https://flows.nodered.org/search?type=node&sort=downloads

6. Evaluación

La nota de la práctica se evalúa de la siguiente forma:

- Parte 1: 50 %
- Parte 2: 35 %
- Parte 2: 15 %

Cada parte seguirá la siguiente rúbrica:

- El programa tiene errores sintácticos o semánticos: 0 puntos
- El programa tiene errores de ejecución: 3 puntos
- El programa funciona sin errores, pero hace lo mínimo y las **explicaciones** son muy escuetas: 5 puntos

- El programa funciona perfectamente, se ha realizado alguna de las mejoras propuestas: 7,5 puntos
- El programa funciona perfectamente, es avanzado, y se han realizado muchas de las mejoras propuestas: 10 puntos

7. FECHA LIMITE DE ENTREGA

La fecha de entrega y presentación será la especificada en el Campus Virtual.