

# Trabajo Práctico N°4

*Nahuel Defossé*

2018

## Taller de Nuevas Tecnologías

### **Notas**

Los trabajos pueden realizarse en grupos de hasta dos personas. La fecha de entrega será pautada, en dos semanas a partir de la fecha en la cual esté disponible.

El formato de entrega puede ser PDF o HTML (o enlace a repositorio dónde debe haber un documento README.md, README.rtf y/o un Cuaderno Jupyter). Se deberá incluir evidencia de el trabajo realizado. Los trabajos no aprobados tendrán un período de recuperación de una semana posterior a la fecha de recepción de las devoluciones.

### **Broker de mensajes MQTT**

1. Descargue la biblioteca cliente MQTT para JavaScript (cliente web) y cree una subscripción a un tópico en una vista Express. Publique a este tópico desde NodeRED utilizando el Broker de mensajes y reciba la salida en el cliente, utilizando D3 sobre elementos DOM (no jQuery).

Puede utilizar una lista, o un textarea . Para esto utilizará el proxy reverso, accediendo al transporte WebSockets de mosquitto.

Hecho. Los fuentes pueden consultarse en <https://github.com/poximan/TNT-PM/tree/master/Practica/tp4/src/img-node-red/embed/src/public>

2. Genere un árbol de tópicos del estilo: /PM/Brewery/P1/valor1 , dónde el prefijo PM indica la ubicación geográfica, Brewery el proceso de general fabricación de cerveza, P1 un subproceso, y valor1 un valor de medida. Utilizando el node-red-dashboard permita ingresar

los valores de diversos procesos, publicándolos en el broker MQTT.

Hecho en el cliente y en el dashboard.

Para el cliente consultar dirección indicada en punto 1.

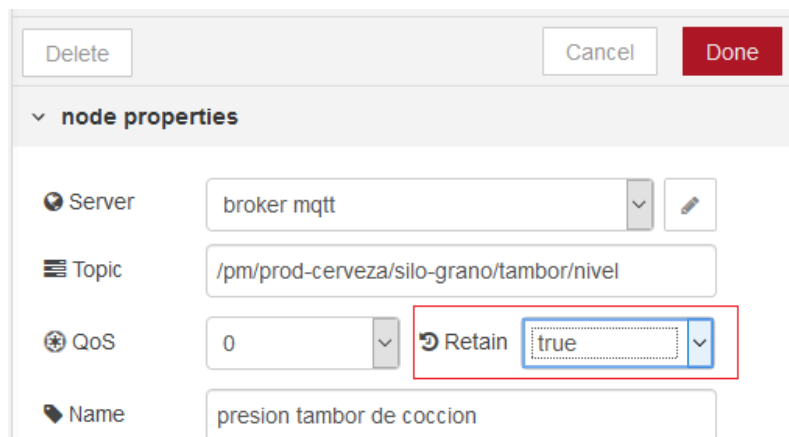
En caso del dashboard, puede probarse importando este ejemplo para nodered <https://github.com/poximan/TNT-PM/blob/master/Practica/tp4/test.txt>

3. Utilizando un gráfico SVG a dónde se utilicen identificadores acordes a los sub-procesos del punto 3, genere el código necesario para que las publicaciones efectúen cambios sobre atributos (ej: text, fill, stroke). Si lo cree conveniente puede utilizar clases CSS (al menos debe funcionar en 2 navegadores).

Hecho. Los fuentes pueden consultarse en <https://github.com/poximan/TNT-PM/tree/master/Practica/tp4/src/img-node-red/embed/src/public>

4. A partir del código del punto 2, haga que los valores sean persistentes (retained). ¿Qué diferencia hay con una API REST? ¿Y con un broker AMQP?

Esto fue embebido en el ejemplo nodered, importar fuente <https://github.com/poximan/TNT-PM/blob/master/Practica/tp4/test.txt>. Una vez importado, observar esta configuración



The screenshot shows the Node-RED MQTT message editor interface. At the top, there are buttons for 'Delete', 'Cancel', and 'Done'. Below this is a section titled 'node properties'. The properties listed are: 'Server' (broker mqtt), 'Topic' (/pm/prod-cerveza/silo-grano/tambor/nivel), 'QoS' (0), 'Retain' (checked, with a dropdown menu showing 'true'), and 'Name' (presion tambor de coccion). The 'Retain' checkbox and its dropdown menu are highlighted with a red rectangle.

5. Modifique el SVGStorage del práctico anterior para publicar, de forma persistente, los archivos que se suben (puede utilizar el nombre como parte del tópico, siempre que no tenga caracteres ilegales).

No realizado.

6. Agregue autenticación a MQTT (investigue si existen imágenes de mosquitto que provean esta funcionalidad). ¿Qué sería necesario para asegurar que las credenciales no viajan en texto plano?

Existen algunas imágenes docker que proveen este servicio:

- <https://hub.docker.com/r/jllopis/mosquitto/>
- <https://hub.docker.com/r/sneck/mosquitto-auth/>
- <https://hub.docker.com/r/simonqbs/mosquitto/>
- <https://hub.docker.com/r/homeit/homeit-mosquitto-auth/>

Se utilizó la primera (jllopis) con autenticación mongo. No funcionó por problemas de integración con las bibliotecas adicionales que requiere mongo-auth para funcionar:

<https://github.com/mongodb/mongo-c-driver>

<https://github.com/mongodb/libbson>

Este punto quedó superado por la autenticación http requerido en el practico siguiente.