Trabajo Practico N4

Alumno: Carlos Emanuel Balcazar
Fecha de entrega: Martes 19 de Junio
Repositorio del proyecto: https://github.com/emanuelbalcazar/TNT-node-red

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 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.
Las librerias utilizadas se encuentran en *node-red/src/public/libs* utilizando la librería de Paho para la conexión desde la vista Express a Mqtt. Ademas se tuvo que modificar el proxy reverso agregando la regla de websocket con:

```
 location /mqtt {
   proxy_pass http://mqtt:9883;
   proxy_set_header X-Real-IP $remote_addr;
   proxy_set_header Host $http_host;
   proxy_set_header Upgrade $http_upgrade;
   proxy_set_header Connection "Upgrade";
 }
```

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.
Para esto se utilizaron 3 topicos especificos con el formato */PM/tnt/p1*, */PM/tnt/p2 y /PM/tnt/p3* que modifican los valores de los 3 dibujos pintados en la pantalla de *http://localhost/nodered/index*
Desde el dashboard http://localhost/nodered/ui se accede a los controles que modifican los valores de cada proceso dibujado en svg.

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).
Para modificar los atributos de los dibujos, se utilizo una librería adicional llamada d3-selection.js que permite aplicar un estilo o atributos a multiples elementos obtenidos previamente con d3.selectAll. Para ello se arma un mensaje en una funcion de nodered intermediaria entre un nodo componente del node-dashboard y el nodo mqtt out que provee la interfaz visual de node-red. El codigo que se ejecuta al llegar un nuevo mensaje se encuentra en *node-red/src/public/mqtt.js* y el SVG que se utiliza se encuentra en *node-red/src/public/mqtt.html*

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?
Para hacer que los mensajes sean persistentes, basta con darle el check al nodo mqtt que dice “retained”, esto permite que cuando se conecta un nuevo cliente a un topico, los mensajes que se colocaron en el broker llegan al nuevo cliente por lo cual recibiria las ultimas actualizaciones del dibujo en su pantalla.
Esto difiere de una API Rest porque no tenes formas de subscribirte y recibir los ultimos cambios a no ser que realices una peticion a una url especifica. Ademas en el broker esta permitido

subscribirse a multiples topicos usando los llamados “filtros” lo cual un cliente recibiria actualizaciones de diversos topicos, esto en una Api Rest no es posible por la limitacion del mismo protocolo http. Si uno quisiera obtener multiples actualizaciones en una Api Rest el cliente deberia solicitar la actualizacion a cada url de forma manual (ademas un servidor no puede enviar automaticamente datos a un cliente, el cliente es quien debe solicitar los datos).

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 llegue a hacerlo :v

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?
Para la autenticacion se utilizo una imagen ya preparada obtenida de https://github.com/jllopis/docker-mosquitto que ya incluia el plugin de autenticacion para mqtt. Para la autenticacion se decidio utilizar una api http a la cual mosquitto solicita la autenticacion. Segun el plugin utilizado las rutas que se deben implementar son: /auth, /superuser y /acl la cuales se agregaron al index.js de la aplicación Express que contiene a nodered. Por el momento la api retorna una respuesta con codigo 200 permitiendo que mqtt siga funcionando normalmente.