Atividade prática - MQTT

1. Crie uma pasta chamada mqtt para este exemplo.

2. Vá até a linha de comando e entre na pasta criada acima. Efetue o download do pacote da biblioteca node.js que encapsula o acesso a mqtt e será utiliza em nosso exemplo, digitando: `npm install mqtt`

O código acima irá baixar e instalar a biblioteca localmente.

3. Crie o arquivo publisher.js que será utilizado como aplicação que publica dados, colocando o seguinte conteúdo:

```
const mqtt = require('mqtt')
const client = mqtt.connect('mqtt://broker.hivemq.com')

var interval = setInterval( function() {
  sendData()
},2000)

client.on('message', () => {
  console.log('message')
})

function sendData()
{
  console.log('publishing')

// COMPLETE COM O CÓDIGO NECESSÁRIO PARA PUBLICAR O DADO
ALEATORIO UTILIZANDO O TOPICO sensores/voltagem
 console.log('published')
}

function randomInt (low, high) {
  return Math.floor(Math.random() * (high - low) + low);
}
```

O código acima publicará uma mensagem em um broker remoto, através da porta padrão 1883 usada pelo MQTT. Para este exemplo funcionar, o seu Firewall não deverá bloquear esta porta.

4. Execute o código anterior através da linha de comando: node publisher.js

5. Crie o arquivo subscriber.js que será utilizado como aplicação que assina o tópico “sensores/voltagem” e recebe dados, colocando o seguinte conteúdo:

```
const mqtt = require('mqtt')
```

```
const client = mqtt.connect('mqtt://broker.hivemq.com')


// COMPLETE COM O CÓDIGO NECESSÁRIO PARA RECEBER
NOTIFICAÇAO DO DADO ATRAVÉS DO TOPICO sensores/voltagem

client.on('connect', () => {
    console.log('connected')
})

client.on('message', (topic, message) => {
  console.log('received message %s %s', topic, message)
})
```

6. Execute o código através da linha de comando:

```
node subscriber.js
```

7. Instale localmente o broker MQTT Mosca (http://www.mosca.io/) como aplicação standalone:

```
npm install mosca pino-pretty -g
```

8. Execute o broker

```
mosca -v | pino-pretty
```

9. É possível instalar o módulo MQTT como uma ferramenta de linha de comando, digitando:

```
npm install mqtt -g
```

Ainda através da linha de comando, tente utilizar a ferramenta mqtt para enviar mensagens através do broker Mosca local que acabou de instalar. Ex:

Em outro console, digite o comando abaixo:

```
mqtt subscribe /topico
```

Em outro console, digite o comando abaixo:

```
mqtt publish /topico "ola mundo”
```

Maiores detalhes aqui: https://www.npmjs.com/package/mqtt

10. Altere o código de publisher.js e subscriber.js para que utilizem o broker de algum dos colegas no laboratório.

11. Assinando diferentes tópicos:

Se quiser assinar todos os tópicos de um broker, use o tópico #
Ex: `client.subscribe('#')`
Tente explorar diferentes hierarquias de tópicos e identificar de onde veio:

```
client.subscribe('meutopico/subtopico1')
client.subscribe('meutopico/subtopico2')
```

Assinando vários tópicos:

```
var topicos=['topico1','topico2','topico3'];
client.subscribe(topicos);
```

Uso de wildcards:
- vários níveis
+ - apenas um nível

Ex: o tópico `predio/#` seria válido para

```
predio/andar1/arcondicionado
predio/sala2/sensorpresenca
predio/camera
```

o tópico `predio/+/sensortemperatura` funciona para

```
predio/andar1/sensortemperatura
predio/sala2/sensortemperatura
predio/recepcao/sensortemperatura
```

12. Desafio: Tente se comunicar com um broker MQTT através do browser (alguns brokers suportam WebSockets, que é um dos protocolos que podem ser usados por baixo do MQTT)

13. Aplicação:
Crie uma interface Web para se conectar ao seu broker MQTT rodando localmente. Uma página HTML deve se conectar através de WebSockets com uma aplicação Node.JS que estará conectada ao broker. Através da interface Web você enviará os canais que quer fazer subscribe ou unsubscribe. Cada mensagem MQTT que chegar nesta aplicação deve ser disponibilizada para os clientes através de EventSource.