



TP2

Teleinformática

Chat en Nodejs

Gustavo Marin
Alejandro Coello

12 de agosto de 2019

1. Objetivo

Definir e implementar un simple protocolo de comunicación para una aplicación CHAT basada en MQTT.

2. Protocolo del chat

El protocolo está definido para utilizar un tópico MQTT ('chat') para poder interactuar entre diferentes implementaciones.

La aplicación de chat debería permitir pasar como argumento, el nombre ('Nick Name') de la persona que ingresa al chat (ver como procesar process.argv en Node.js)

Cuando alguien ingresa al chat, anuncia su ingreso, enviando un mensaje con su Nick Name a todos.

2.1. Mensajes para todos

El formato es el siguiente:

[nickname] mensaje

El carácter "[" seguido del nick_name (sin espacios), seguido del carácter "]", seguido de un espacio, y luego el mensaje en texto hasta el fin de línea

2.2. Mensajes privados

El formato es el siguiente:

[nick_name] @nick_name_destino mensaje

El carácter "[" seguido del nick_name (sin espacios), seguido del carácter "]", seguido de un espacio, seguido del carácter "@", inmediatamente seguido por el nick name de la persona con la que se quiere chatear en privado **@nick_name_destino**, seguido de un espacio, y luego el mensaje en texto hasta el fin de línea.

Para diferenciar el mensaje privado, se deberá utilizar la impresión con algún color diferente.

2.3. Mensajes enviados y recibidos

Los mensajes enviados por el usuario deberán ser imprimidos, con el carácter 'mayor que' seguido de espacio ">" y los mensajes recibidos, con el carácter 'menor que' seguido de un espacio "<".

3. Implementación

Se inicio el proyecto utilizando **npm init** en el terminal, se llenaron los datos correspondientes. Se utilizaron los siguientes packages:

- **mqtt**
Es utilizado para poder utilizar el protocolo mqtt.
- **readline**
Es utilizado para poder leer los mensajes escritos en el terminal.
- **beepbeep**
Es utilizado para notificar con un *beep* cuando un mensaje es recibido.
- **replace-string**
Es utilizado para reemplazar caracteres dentro de una variable tipo *string*

Estos fueron instaladas utilizando **npm install package_name --save**

Posteriormente para poder utilizar estos packages en el script se tuvo que añadir el siguiente código:

```
// mqtt package
var mqtt = require('mqtt')
var client = mqtt.connect('mqtt://research.upb.edu')
// beepbeep package
var beep = require('beepbeep')
// replace-string package
const replaceString = require('replace-string');
// readline package
var readline = require('readline');
var rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
  terminal: false
});
// Variable de tipo string que contiene el nombre de usuario
var username = process.argv[2].toString()
```

Posteriormente se realizar una suscripción al tópico chat y si se logra conectar de forma correcta se mandara un mensaje indicando que el usuario se ha conectado al chat

```
client.on('connect', function () {
  client.subscribe('chat', function (err) {
    if (!err) {
      client.publish('chat', username + ' has joined the chat')
    }
  })
})
```

Posteriormente se realiza una impresión de los mensajes recibidos en el tópico chat, es aquí además donde se realizará la verificación si es que el mensaje es enviado o recibido, para añadir ¿y ¡, y para realizar el envío de mensajes privados.

```
client.on('message', function (topic, message) {
  var br_1 = message.indexOf('[')+1;
  var br_2 = message.indexOf(']');
  var user = message.slice(br_1, br_2);

  if (user == username){
    message = '> ' + message;
  }
  else {
    replaceString(message.toString(), '>', '<');
    message = '< ' + message;
  }

  var at_pos = message.indexOf("@");
  var sp_pos = message.indexOf(" ", at_pos+1);
  var pr_msg_1 = message.slice(0, at_pos-1);
  var pr_msg_2 = message.slice(sp_pos, message.length);
  var cl_un = message.slice(at_pos+1, sp_pos);
  var pr_msg = pr_msg_1 + pr_msg_2;

  if (at_pos < 0) {
    console.log(message.toString())
    beep()
  }
  else {
    if (cl_un == username){
      console.log('\x1b[36m%s\x1b[0m', pr_msg.toString());
      beep()
    }
  }
})
```

Posteriormente se realiza la lectura de lo que se escribe en el terminal utilizando `readline`, añadiendo el formato que se solicitó, además de añadir la condición **quit** para abandonar el chat.

```
rl.on('line', function(line){
  if (line == 'quit'){
    client.publish('chat', username + ' has left the chat')
    client.end()
    setTimeout(function(){process.exit()},500)
  }
  else {
    client.publish('chat', '[' + username + ']' + line)
  }
})
```

Finalmente se añadió una condición para indicar que el usuario abandonó el chat cuando se utiliza **Ctrl + C** para terminar la aplicación.

```
process.on('SIGINT', function(){
  client.publish('chat', username + ' has left the chat')
  setTimeout(function(){process.exit()},500)
})
```

4. Resultados

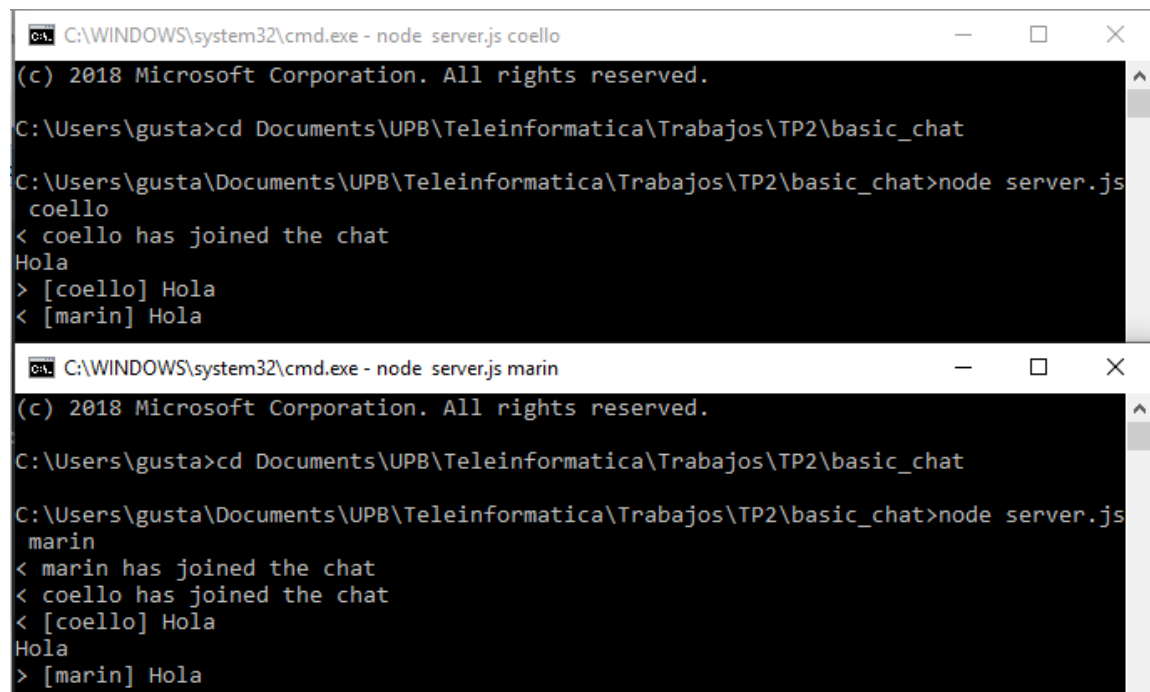
A continuación se adjuntaran imágenes de la aplicación funcionando.

En la siguiente imagen se muestra el mensaje en el cual se anuncia que el usuario se ha conectado al chat.

```
C:\Users\gusta\Documents\UPB\Teleinformatica\Trabajos\TP2\basic_chat>node server.js marin
< marin has joined the chat
```

Figura 1: Anuncio de conexión al chat

En la siguiente imagen se puede ver que el formato es el correcto, el nombre de usuario dentro de los brackets, seguido de un espacio y seguido del mensaje, finalmente se puede observar que los mensajes enviados tienen el símbolo > y los mensajes recibidos el símbolo <.

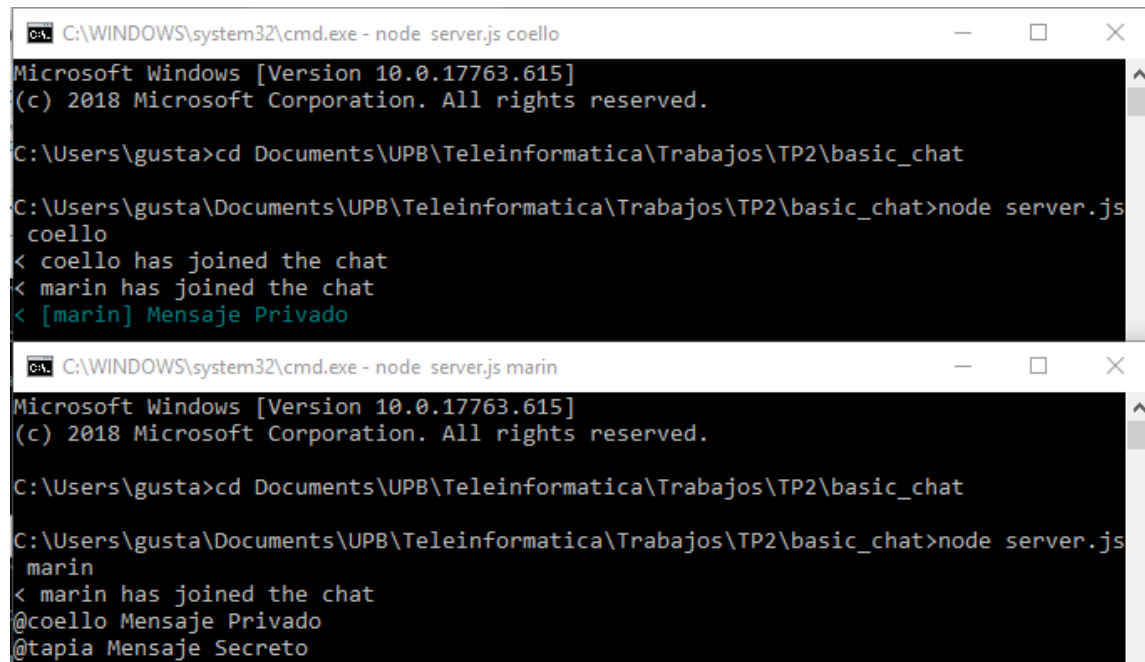


```
C:\WINDOWS\system32\cmd.exe - node server.js coello
(c) 2018 Microsoft Corporation. All rights reserved.
C:\Users\gusta>cd Documents\UPB\Teleinformatica\Trabajos\TP2\basic_chat
C:\Users\gusta\Documents\UPB\Teleinformatica\Trabajos\TP2\basic_chat>node server.js
coello
< coello has joined the chat
Hola
> [coello] Hola
< [marin] Hola

C:\WINDOWS\system32\cmd.exe - node server.js marin
(c) 2018 Microsoft Corporation. All rights reserved.
C:\Users\gusta>cd Documents\UPB\Teleinformatica\Trabajos\TP2\basic_chat
C:\Users\gusta\Documents\UPB\Teleinformatica\Trabajos\TP2\basic_chat>node server.js
marin
< marin has joined the chat
< coello has joined the chat
< [coello] Hola
Hola
> [marin] Hola
```

Figura 2: Aplicación Chat funcionando

En la siguiente imagen se puede ver el funcionamiento de los mensajes privados. Estos son imprimidos de un color diferente, además que solo pueden ser vistos por el destinatario correcto.



```
C:\WINDOWS\system32\cmd.exe - node server.js coello
Microsoft Windows [Version 10.0.17763.615]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\gusta>cd Documents\UPB\Teleinformatica\Trabajos\TP2\basic_chat

C:\Users\gusta\Documents\UPB\Teleinformatica\Trabajos\TP2\basic_chat>node server.js
coello
< coello has joined the chat
< marin has joined the chat
< [marin] Mensaje Privado

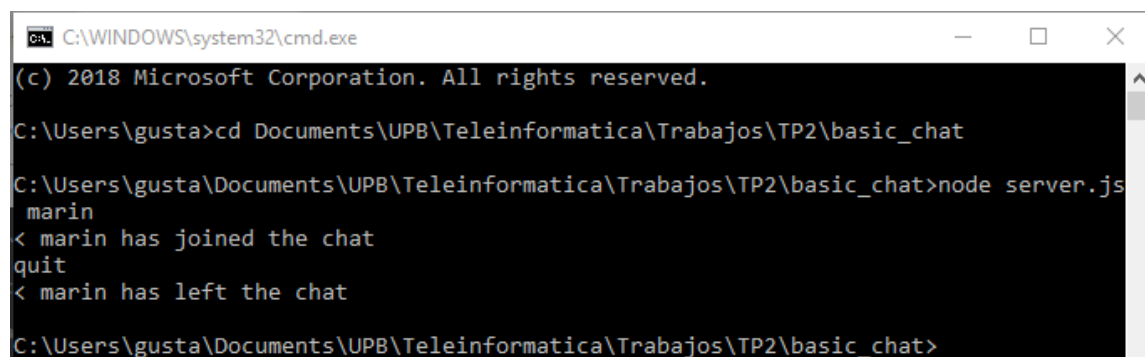
C:\WINDOWS\system32\cmd.exe - node server.js marin
Microsoft Windows [Version 10.0.17763.615]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\gusta>cd Documents\UPB\Teleinformatica\Trabajos\TP2\basic_chat

C:\Users\gusta\Documents\UPB\Teleinformatica\Trabajos\TP2\basic_chat>node server.js
marin
< marin has joined the chat
@coello Mensaje Privado
@tapia Mensaje Secreto
```

Figura 3: Mensajes Privados

Finalmente se implemento el abandono del chat mediante el comando **quit**, además de mandar un mensaje de que se esta abandonando el chat.



```
C:\WINDOWS\system32\cmd.exe
(c) 2018 Microsoft Corporation. All rights reserved.

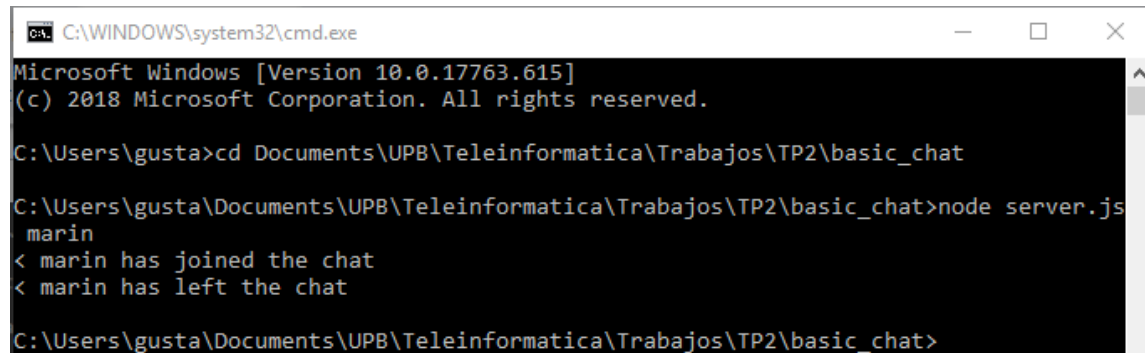
C:\Users\gusta>cd Documents\UPB\Teleinformatica\Trabajos\TP2\basic_chat

C:\Users\gusta\Documents\UPB\Teleinformatica\Trabajos\TP2\basic_chat>node server.js
marin
< marin has joined the chat
quit
< marin has left the chat

C:\Users\gusta\Documents\UPB\Teleinformatica\Trabajos\TP2\basic_chat>
```

Figura 4: Finalización del chat con quit

El mensaje de abandono del chat es también publicado cuando se cierra la aplicación utilizando **Ctrl + C**.



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.17763.615]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\gusta>cd Documents\UPB\Teleinformatica\Trabajos\TP2\basic_chat

C:\Users\gusta\Documents\UPB\Teleinformatica\Trabajos\TP2\basic_chat>node server.js
marin
< marin has joined the chat
< marin has left the chat

C:\Users\gusta\Documents\UPB\Teleinformatica\Trabajos\TP2\basic_chat>
```

Figura 5: Finalización del chat con Ctrl + C

5. Conclusiones

- La implementación del chat con el formato indicado fue exitosa.
- Nodejs es una herramienta muy útil para el desarrollo de múltiples aplicaciones.
- Los paquetes de javascript son muy útiles para el uso rápido de funciones específicas.