Carlos Bedoya
Stephany Berrio
Comunicaciones II

ChatSocketIO

ChatSocketIO

ChatSocketIO es una aplicación web la cuál está desarrollada con **Node.js**, **Socket.IO** y **Express**.

La aplicación es desarrollada en lenguaje JavaScript, debido a las características de Node.js, el cual para nuestra aplicación hace las veces de servidor, además implementamos Express, que es un framework para Node. js, que permite construir aplicaciones robustas de manera rápida y eficiente, y que además facilita la creación del servidor de manera automática.

Socket.IO

Socket.io es una librería que nos permite manejar eventos en tiempo real mediante una conexion TCP y todo ello en Javascript.
Establece un tipo de comunicación bidireccional entre el cliente y el servidor servidor

Socket-io

- Lado del cliente(client-side) librería que corre en el navegador
- Lado del servidor(server-side), librería para Node.js

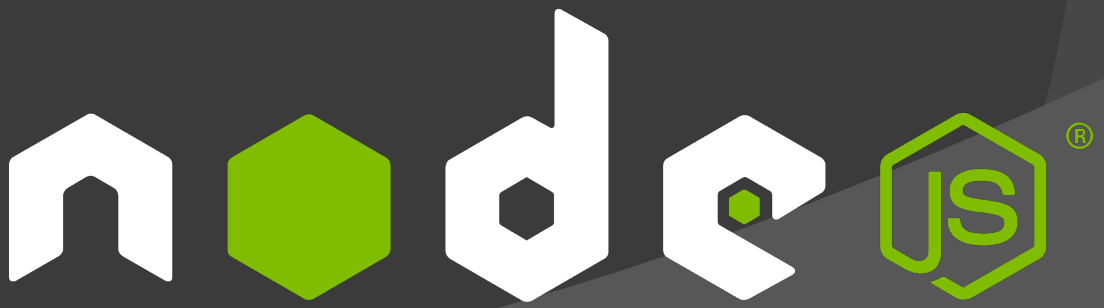
Ambas tienen un API casi idéntica , y al igual que node.js tienen una arquitectura event-driven

Características

- Utiliza primariamente el protocolo webSocket con “polling” como opción de retorno
- Puede ser usado como un simple envoltorio para WebSocket.
- Transmisión por múltiples sockets
- Permite almacenamiento de los datos asociados a cada cliente

Node.js

Node.js es un entorno de programación en la capa del servidor basado en el lenguaje de programación **Javascript**, asíncrono, con I/O de datos en una **arquitectura orientada a eventos** y basado en el motor Javascript V8 de Google.



Node.js

- Al contrario que la mayoría del código JavaScript, este no se ejecuta en un navegador, sino en el servidor.
- Node.js incorpora varios "módulos básicos"
- Los módulos pueden instalarse como archivos simples, normalmente se instalan utilizando el Node Package Manager (npm)

Express

Express es un Framework para aplicaciones web con Node.js, diseñado para la construcción de una , varias páginas o aplicaciones web híbridas , que además proporciona un robusto conjunto de características para aplicaciones web y móviles.

Características

- Ofrece Router de URL (Get, Post, Put …)
- Facilidades para motores de plantillas (Jade, EJS, JinJS, entre otros)
- Middeleware via Connect
- Test coverage

Implementación

index.js | index.html | package.json | top.png

```
1
2  //importa el modulo express
3  var app= require('express')();
4  //importa modulo http e inicializa el srvidor
5  var http= require('http').Server(app);
6  var io= require('socket.io')(http);// importa socket.io
7  var username={};//Arreglo con los usuarios conectados, contiene los id
8
9
10 //recibe las peticiones de la url '/' req es la peticion
11 app.get('/', function(req, res){
12     //res es la respuesta q en este caso es un html
13     res.sendFile(__dirname+'/index.html');
14 });
15
16 app.get('/form', function(req, res){
17     //res es la respuesta que en este caso es un html
18     res.sendFile(__dirname+'/form.html');
19 });
20
21 //Mientras haya conexion llama la funcion
22 io.on('connection',function(socket){
23     console.log('usuario conectado id: '+ socket.id);
24
25     //Cuando un cliente envia un mensaje el servidor lo recibe
26     //y lo retransmite a los demas clientes
27     socket.on('message', function(data) {
28         console.log(data);
29         io.emit('response',username[socket.id]+" : "+data);
30     });
31
```

Implementación

```
31
32      //Guardamos al usuario en un vector username mediante el id
33      socket.on('usuario', function(user) {
34          console.log(user);
35          username[socket.id]=user;
36      });
37
38
39  });
40
41  //se pone al servidor a escuchar en el puerto 7001
42  http.listen(7001, function(){
43
44      console.log('Servidor inicializado en el puerto 7001' );
45  });
```