

**MANUAL CHATSOCKETIO
COMUNICACIONES II**

**CARLOS BEDOYA
STEPHANY BERRIO**

**UNIVERSIDAD DE ANTIOQUÍA
INGENIERÍA DE SISTEMAS
MEDELLIN
2014**

MANUAL

1. Introducción

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, ya que con pocos comandos permite tener una completa arquitectura para una aplicación web, además facilita la creación del servidor de manera automática.

Para la implementación del chat se utilizó Socket.io, que es una librería en JavaScript para Node.js que permite una comunicación bidireccional en tiempo real entre cliente y servidor, la conexión con el servidor se tiene por el puerto 7001.

2. CONFIGURACIÓN DE AMBIENTE

Para el correcto funcionamiento de la aplicación ChatSocketIO, el equipo debe tener instalado previamente cierto software como se muestra a continuación en los numerales 2.1 y 2.2, en los cuales se explica la forma de instalación de Node.js y npm(Manejador de paquetes de Node).

En caso de tener instalado previamente el software lo único que se debe hacer para desplegar correctamente el proyecto es escribir por consola el comando **nodejs index.js**, en algunos casos en vez de nodejs se debe ingresar node, esto es según la instalación previa del mismo.

2.1. Instalación Sistema Operativo Linux

En el sistema Linux se deben ejecutar varios comandos desde la consola para la correcta instalación de Node.js

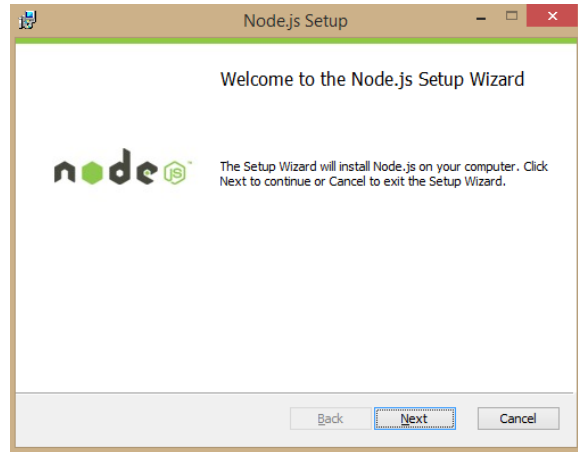
1. `sudo apt-get install nodejs`
2. `sudo apt-get install npm`

Para el despliegue del proyecto no es necesario la instalación de Express, pues esta se debe hacer solo en el momento de la creación del proyecto.

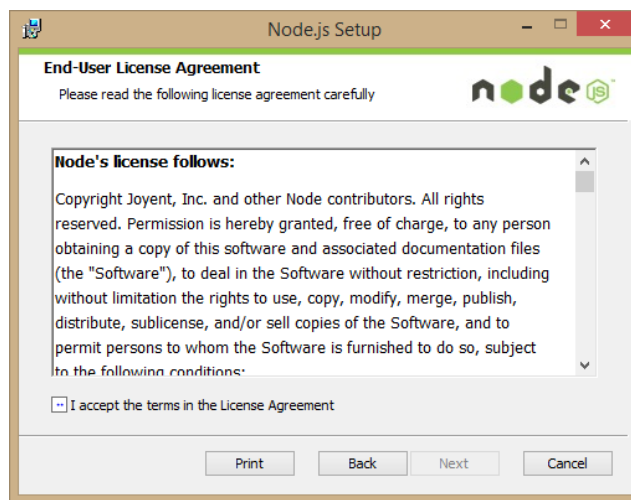
Una vez instalado el software se ejecuta con el comando descrito anteriormente.

2.2. Instalación Sistema Operativo Windows

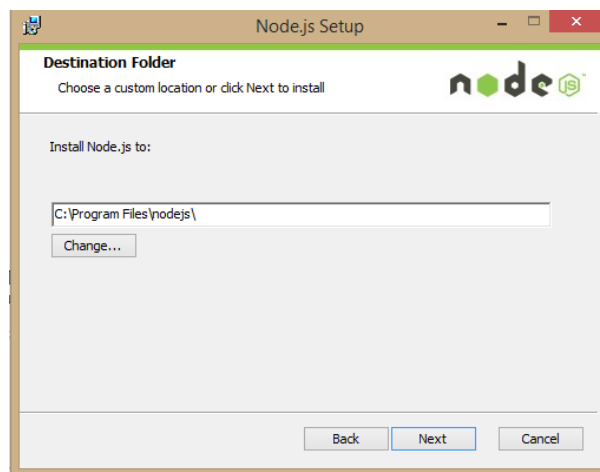
En el sistema operativo Windows la instalación de Node.js es un tanto diferente ya que se realiza a través de un ejecutable como mostramos a continuación y solo es seguir los pasos en las pantallas.



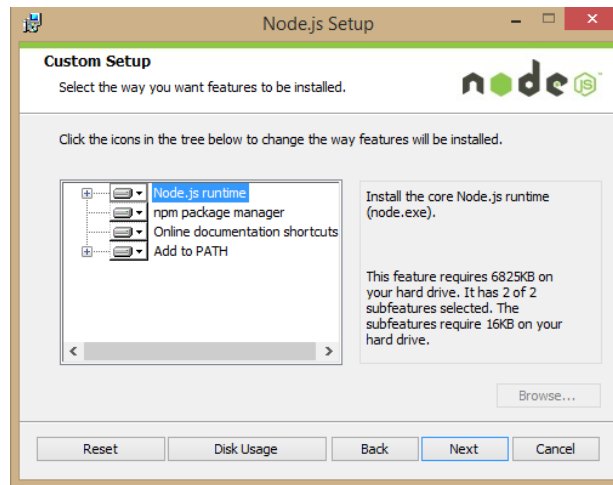
Se aceptan los términos y condiciones de la licencia



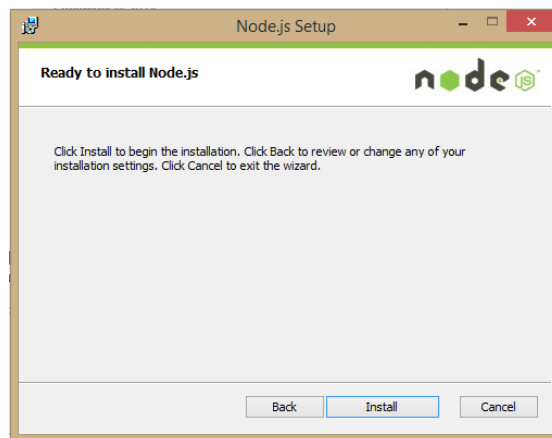
Se elige la carpeta de ubicación para la instalación.



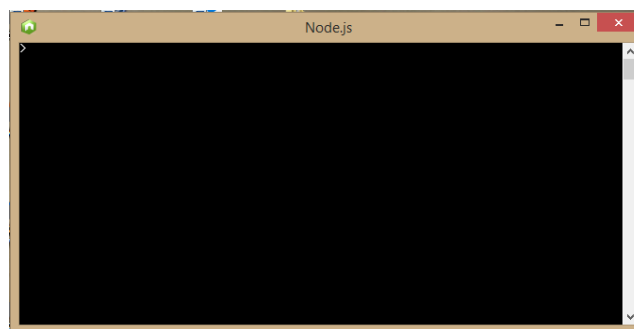
Se selecciona los módulos a instalar, entre los cuales podemos observar que ya viene con la opción de agregar npm



Se selecciona instalar y se procede con la instalación.



Al finalizar se ejecutará una consola como la que se muestra a continuación en la cual se deben ingresar todos los comandos relacionados con Node.js



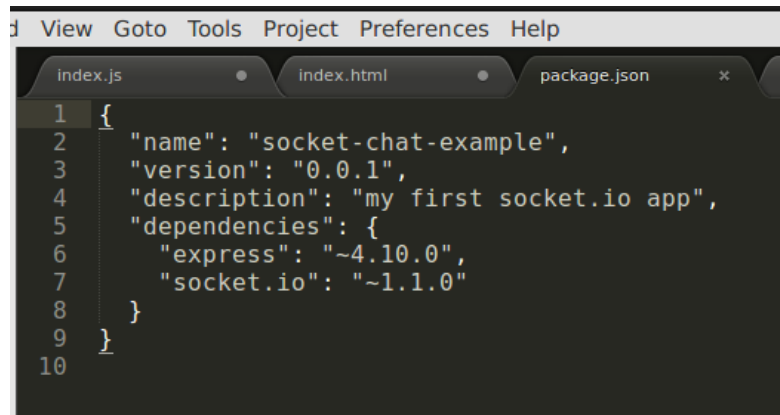
3. Proyecto

Al crear el proyecto con Express se crean por defecto todos los componentes que se muestran a continuación.



Como se puede observar están los módulos, librerías y demás componentes de configuración, junto con la librería Socket.IO la cual empleamos para el desarrollo del chat.

Es de resaltar que el proyecto tiene un archivo de tipo JSON con todas las configuraciones de dependencias o librerías que se usan, algo análogo al Manifest de Android, como se muestra en la siguiente imagen.

A screenshot of a code editor window showing the 'package.json' file. The editor has a menu bar with 'View', 'Goto', 'Tools', 'Project', 'Preferences', and 'Help'. Below the menu bar are tabs for 'index.js', 'index.html', and 'package.json'. The 'package.json' tab is active, displaying the following JSON code:

```
1 {  
2   "name": "socket-chat-example",  
3   "version": "0.0.1",  
4   "description": "my first socket.io app",  
5   "dependencies": {  
6     "express": "~4.10.0",  
7     "socket.io": "~1.1.0"  
8   }  
9 }  
10
```

En este archivo, cada vez que instalemos un nuevo componente quedará registrado en él.

4. Desarrollo

Para la creación del chat básicamente creamos dos archivos index.js e index.html. En el primero encontramos todo el código relacionado con el servidor, todos los Request y los Response necesarios como veremos a continuación.

```
index.js  index.html  package.json  x  top.png

1
2 //importa el modulo express
3 var app= require('express')();
4 //importa modulo http e inicializa el servidor
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
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 });
```

Ahora del lado del cliente tenemos un HTML básico, para la implementación de la vista del usuario como se muestra a continuación.

```

<script src="/socket.io/socket.io.js"></script>
<script src="http://code.jquery.com/jquery-2.1.0.min.js"></script>
<!-- Inicializamos el socket del cliente-->
<script>
$(document).ready(function() {

    var socket =io();

    nombre =prompt('Introduce tu nombre','Nombre de Usuario');
    socket.emit('usuario',nombre);//se envia el usuario ingresado por el usuario al servidor
    mediante emit con la clave usuario y el valor nombre
    // body...
        $('#chat_form').submit(function(e){
            //se envia el mensaje ingresado por el usuario al servidor mediante emit con la clave
            message y el valor $('#m').val() que es enviado como valor
            socket.emit('message',$('#m').val());
            //se limpia lo que esta almacenado en $('#m').val()
            $('#m').val('');
            return false;
        });

        //Socket.on lo que hace es escuchar al servidor que le esta enviando con la clave
        response y el valor data el mensaje enviado por los usuarios
        socket.on('response',function(data) {
            $('#messages').append('<li><h1>').text(data));
        });
    });
});

```

Como podemos observar tenemos un script que ejecuta un código de JQUERY, mediante el cual le enviamos y recibimos información del servidor.

```

</head>
<body>

    <div>
        <div id="Layer1" overflow:"scroll">

            <ul id="messages" ></ul>
        </div>

        <form id="chat_form" >
            <input id="m" autocomplete="off" /><button type="submit">Enviar</button>
        </form>
    </div>
</body>

</html>

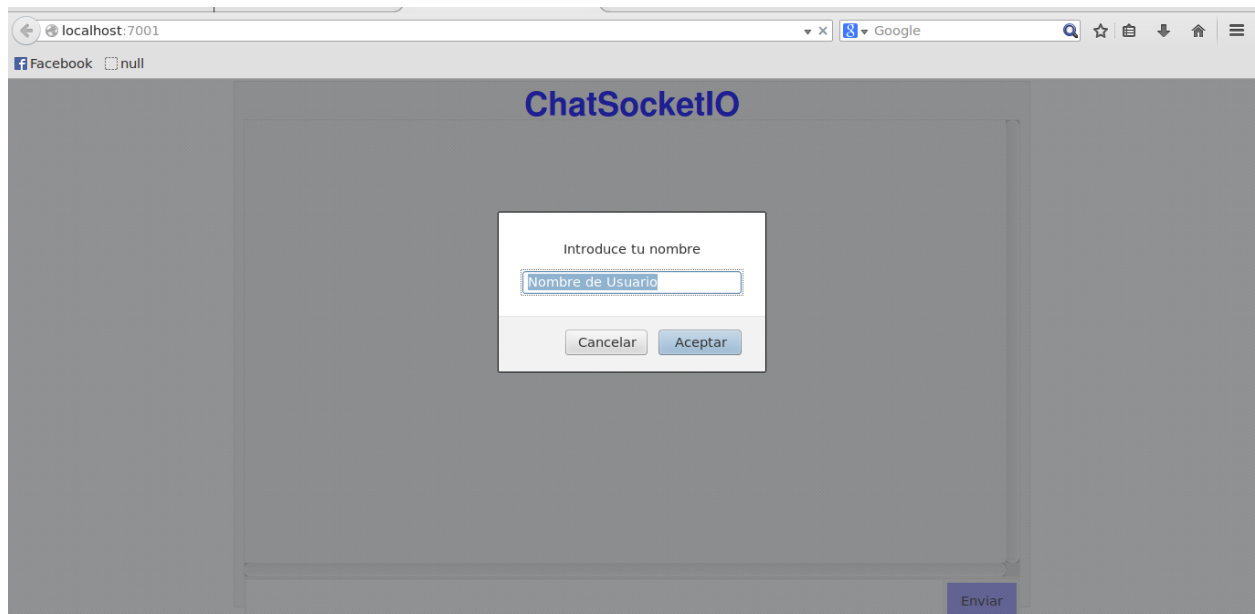
```

En la imagen anterior podemos observar el HTML que da forma al chat.

5. Interfaz ChatSocketIO

Para el uso del chat solo se debe acceder a la IP determinada del servidor en el puerto definido por el mismo, que en nuestro caso es el puerto 7001, al ingresar se le pedirá al

usuario que ingrese un nombre de usuario o su nombre, lo que el usuario desee, como se muestra a continuación.



Una vez ingrese este usuario podrá acceder al chat e iniciar una conversación con los demás integrantes como se muestra a continuación.

