Proyecto nuevo SOCKET

Chat

* Creamos nueva carpeta dentro de 03\_Server

express --view=pug --git Chat

* Accedemos dentro de Chat

cd Chat

Instalamos todo lo de package.json

npm install

* Vamos a instalar una librería para poder capturar eventos.

npm install socket.io

* Accedemos a nuestra aplicación.

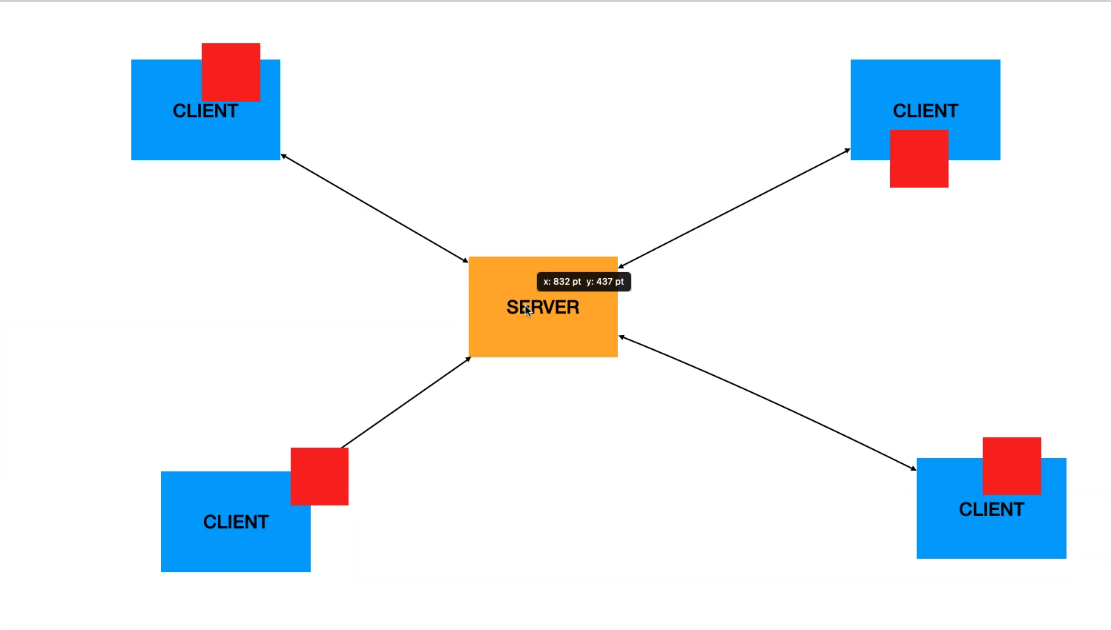
code .

* Vamos a package.json y configuramos el scripts

  "scripts": {

    "start": "node ./bin/www",

    "dev": "nodemon ./bin/www"

ESTRUCTURA DE CHAT

* **Dentro de www cambiamos el tipo de archivo a extensión js.**

**Vamos hacer una interacción entre cliente y servidor.**

* **Primera parte vamos a empezar por la parte del SERVIDOR**

const soketIo = *require*('socket.io');

* **Configuración servidor WS**

**Importamos una función a partir del servidor.**

**Justo debajo de la línea 23**

* ***(var server = http.createServer(app))***

const io = *socketIO*(server)

**Escuchamos el evento ‘conecction’, es decir cuando app escuche ese evento ejecuta esta función.**

io.*on*('conecction', (socket) => {

    console.*log*('Se ha conectado un nuevo cliente');

});

* **Dentro de routes/index.js**

**Especificamos una serie de código pug, es decir mandar una vista, recordamos que para mandar una vista al navegador utilizábamos como respuesta res.render**

router.*get*('/', (req, res) => {

  res.*render*('chat');

});

* **Dentro de /view generamos la vista, creamos un fichero llamado chat.pug**

**El código JavaScript en pug se aplica con la palabra scripts.**

extends layout.pug

block content

    h1 Bienvenido al chat

    script.

* **En la cabecera de nuestro HTML es decir el layout.pug tenemos que decirle de donde va a sacar a la librería.**

    script(src="/socket.io/socket.io.js")

* **Continuamos dentro de chat.pug**

**Vamos hacer una función**

**Cuando yo imprima este archivo en el navegador se va a lanzar al servidor por ejemplo como el console.log que creamos en www,.js**

extends layout.pug

block content

    h1 Bienvenido al chat

    script.

        console.*log*('Me conecto al chat');

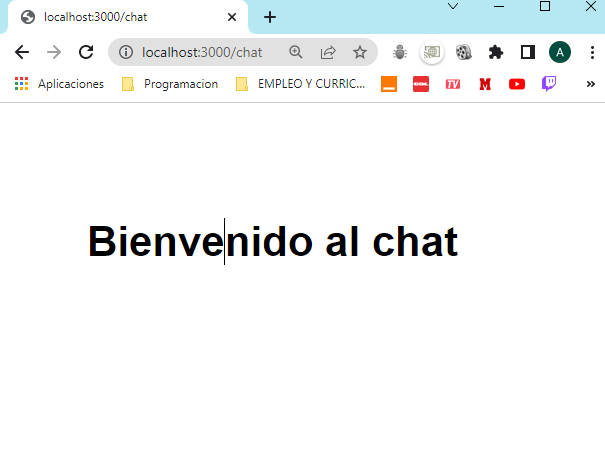
        const socket = *io*();

* **Desde la Terminal arrancamos npm run dev**

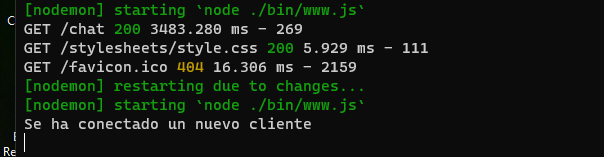
**npm run dev**

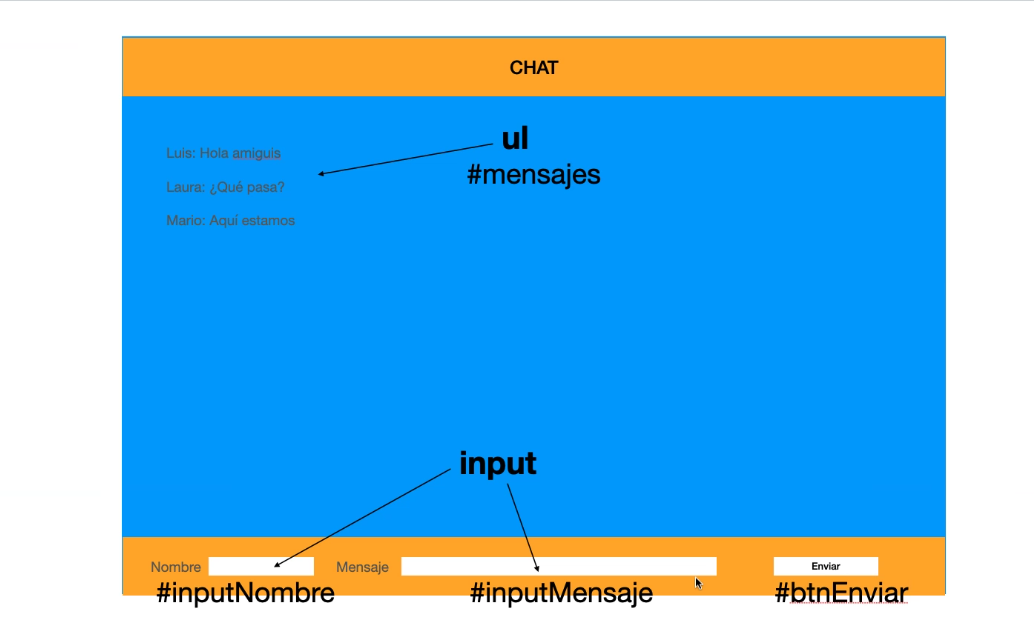
* **Probamos en el Navegador**

**localhost:3000/chat**



* **En la consola no debe aparecer el console.log que hemos realizado en el** [**www.js**](http://www.js)



**EJEMPLO DE LA INTERFAZ**

* **Dentro de la carpeta /JavaScript creamos el fichero client-chat.js**
* const socket = *io*();
* **Dentro de chat.pug**

**Editamos el script con la ruta del nuevo fichero js para no trabajar todo dentro del chat.pug**

    script(src="/javascripts/client-chat.js")

* **Dentro de client-chat.js**

**Nos traemos los botones y los inputs**

const btnEnviar = document.*getElementById*('btnEnviar');

const inputNombre = document.*getElementById*('inputNombre');

const inputMensaje = document.*getElementById*('inputMensaje');

**Escuchamos el click de esos botones, si hace click quiero que hagas esta función.**

**NOTA: El (event) se utiliza para saber la información en este caso donde hacemos (click) en este caso pues no necesitamos eso.**

btnEnviar.*addEventListener*('click', () => {

});

**Vamos el recuperar los valores de los campos**

const socket = *io*();

const btnEnviar = document.*getElementById*('btnEnviar');

const inputNombre = document.*getElementById*('inputNombre');

const inputMensaje = document.*getElementById*('inputMensaje');

btnEnviar.*addEventListener*('click', (e) => {

    const body = {

        nombre: inputNombre.value,

        mensaje: inputMensaje.value

    }

});

**Emitimos un evento tipo mensaje\_chat es decir el click, podemos cambiarle el nombre para poder identificarlo nosotros de mejor manera.**

const socket = *io*();

const btnEnviar = document.*getElementById*('btnEnviar');

const inputNombre = document.*getElementById*('inputNombre');

const inputMensaje = document.*getElementById*('inputMensaje');

btnEnviar.*addEventListener*('click', (e) => {

    const body = {

        nombre: inputNombre.value,

        mensaje: inputMensaje.value

    }

    socket.*emit*('mensaje\_chat', body);

});

**EMITIMOS**

**NOTA: CON ESTO HEMOS CONSEGUIDO EMITAR HACIA EL SERVIDOR.**

**EMIT: ESTA EN EL CLIENTE QUE ES EL QUE MANDA HACIA EL SERVIDOR**

**Dentro de** [**www.js**](http://www.js)

**Tenemos que poner a escuchar sobre elemento socket (mensaje\_chat)**

*//Configuración servidor WS*

const io = *socketIO*(server);

io.*on*('connection', (socket) => {

  console.*log*('Se ha conectado un nuevo cliente');

  socket.*on*('mensaje\_chat', (body) => {

    console.*log*(body);

  })

});

**NOTA: Si escribimos en el navegador en el campo mensaje lo que se escriba ahí enviamos y nos reflejara el mismo mensaje que ponemos en el campo nos aparecerá en la consola.**

**YA TENEMOS EL BODY**

* **Dentro de** [**www.js**](http://www.js)

**Tenemos que enviar este console.log a todos los clientes conectados.**

*//Configuración servidor WS*

const io = *socketIO*(server);

io.*on*('connection', (socket) => {

  console.*log*('Se ha conectado un nuevo cliente');

  socket.*on*('mensaje\_chat', (body) => {

    io.*emit*('mensaje\_chat', body);

  })

});

* **Dentro de client-chat.js**

**Ahora tenemos que hacer lo mismo que en** [**www.js**](http://www.js) **en la configuración del servidor para que esté al loro de ON suscribirnos a ese evento, es decir que esté pendiente del ON**

const socket = *io*();

const btnEnviar = document.*getElementById*('btnEnviar');

const inputNombre = document.*getElementById*('inputNombre');

const inputMensaje = document.*getElementById*('inputMensaje');

btnEnviar.*addEventListener*('click', (e) => {

    const body = {

        nombre: inputNombre.value,

        mensaje: inputMensaje.value

    }

    socket.*emit*('mensaje\_chat', body);

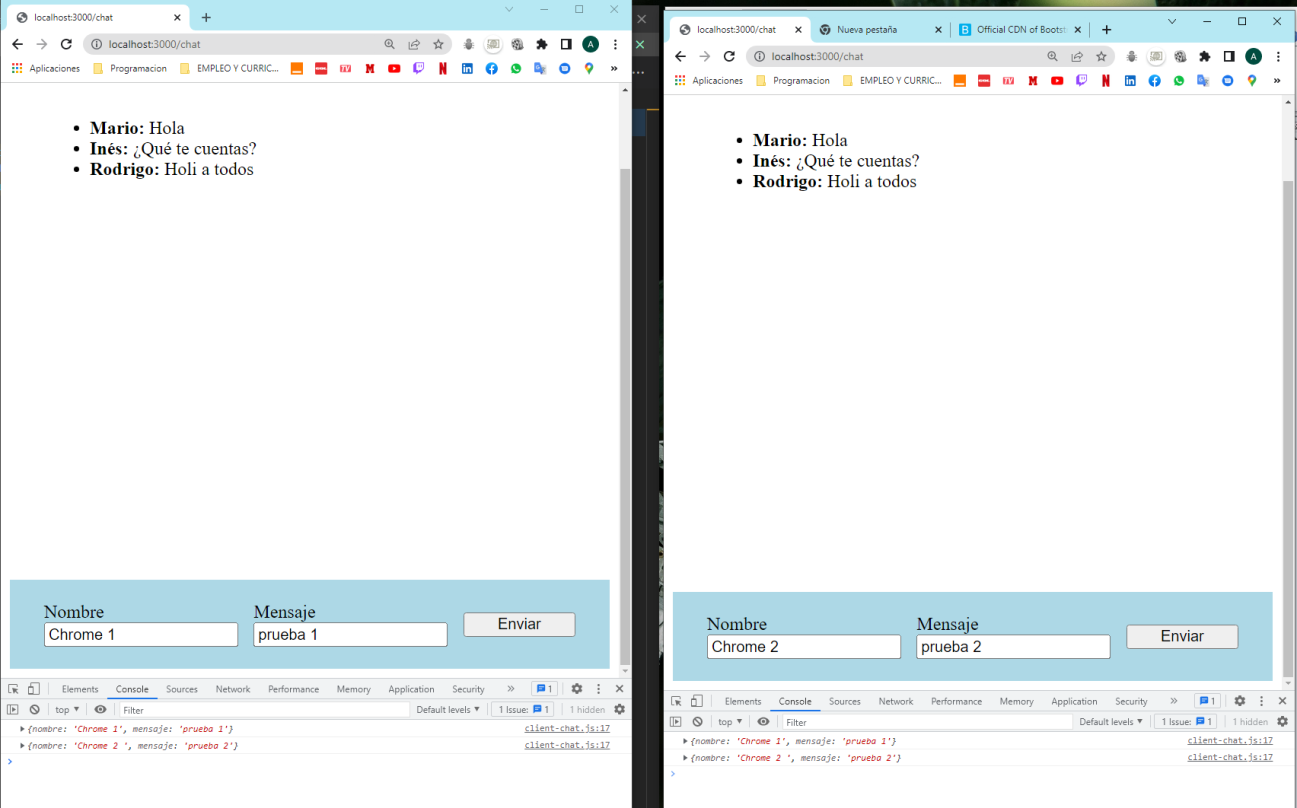
});

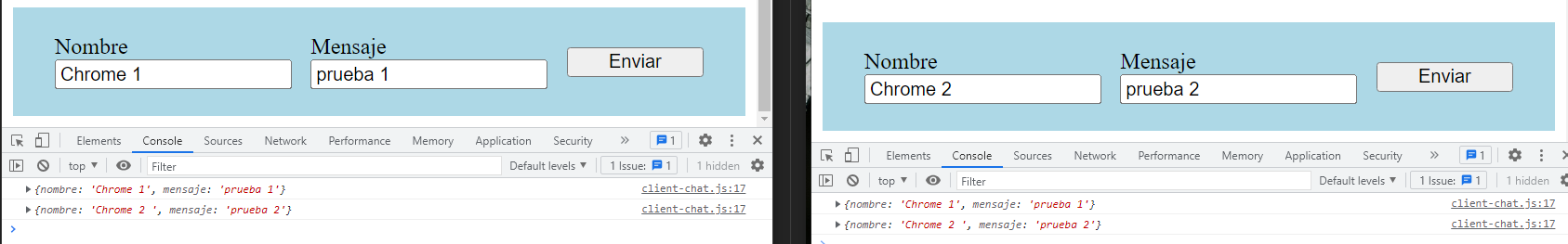
socket.*on*('mensaje\_chat', (body) => {

    console.*log*(body);

})

* **Comprobamos en el navegador, abrimos varios navegadores para simular varios clientes a la vez.**





**PRACTICA PROFESOR**

**- Probamos a crear un nuevo elemento LI donde colocamos nombre y texto. ( CreateElement, innerHTML )**

**- Agregar ese elemento LI al UL mensajes (appendChild)**

* **Dentro de client-chat.js**

const socket = *io*();

const btnEnviar = document.*getElementById*('btnEnviar');

const inputNombre = document.*getElementById*('inputNombre');

const inputMensaje = document.*getElementById*('inputMensaje');

const mensajes = document.*getElementById*('mensajes');

btnEnviar.*addEventListener*('click', (e) => {

    const body = {

        nombre: inputNombre.value,

        mensaje: inputMensaje.value

    }

    socket.*emit*('mensaje\_chat', body);

});

socket.*on*('mensaje\_chat', (body) => {

    console.*log*(body);

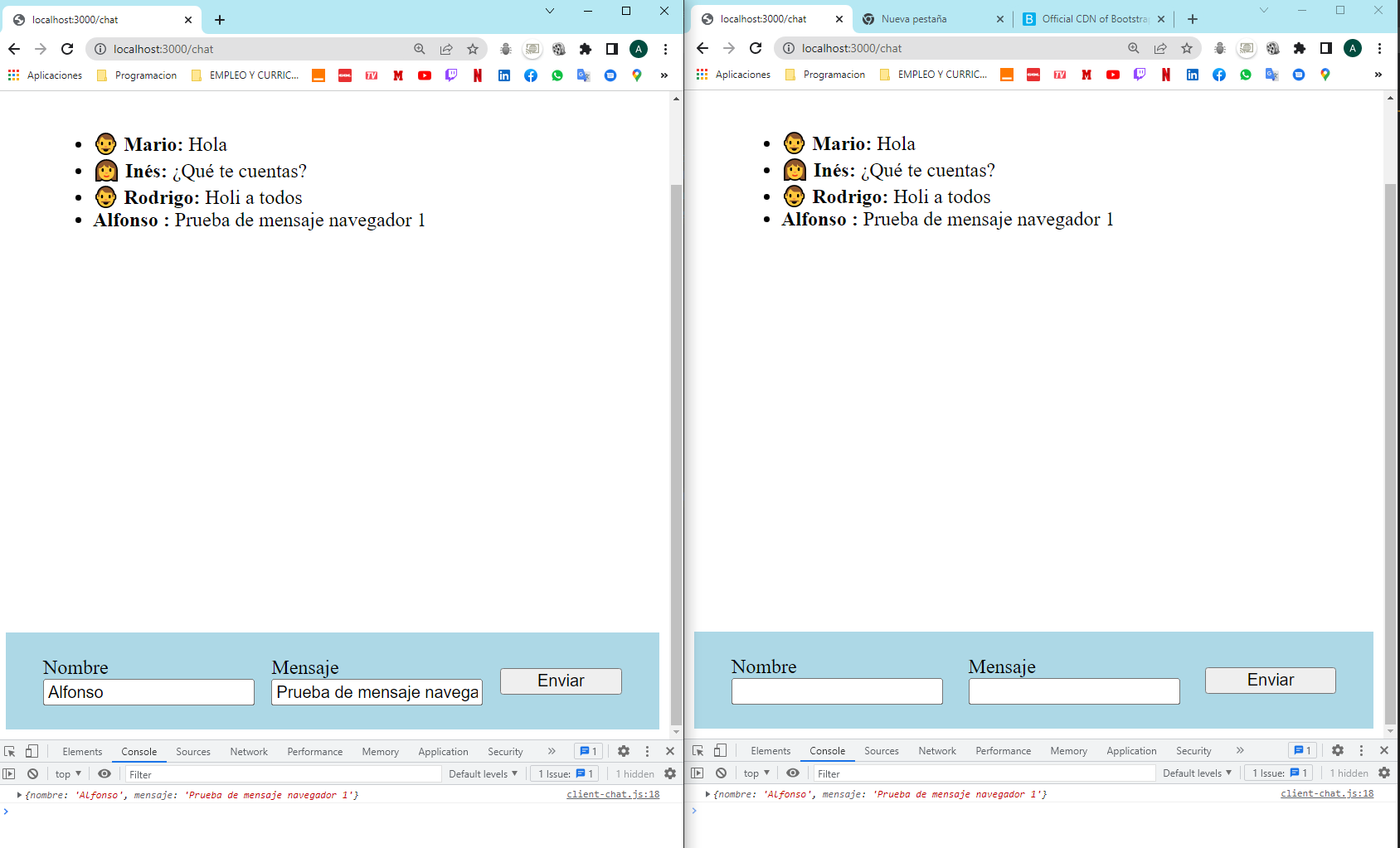
    const li = document.*createElement*('li');

    li.innerHTML = `<strong> ${body.nombre} : </strong> ${body.mensaje}`;

    mensajes.*appendChild*(li);

})

* **Lanzamos de nuevo los navegadores y probamos que se visualizan los mensajes al momento.**



* **Como saber las persona que están conectados dentro de** [**www.js**](http://www.js)
* **Recibir todos menos al que se conecta.**

*//Configuración servidor WS*

const io = *socketIO*(server);

io.*on*('connection', (socket) => {

  console.*log*('Se ha conectado un nuevo cliente');

*//desde el server para el que este solo conectado*

  socket.broadcast.*emit*('mensaje\_chat', {

    nombre: 'INFO',

    mensaje: 'Se ha conectado un nuevo usuario'

  });

*//emitir desde el server para todo el mundo*

  socket.*on*('mensaje\_chat', (body) => {

    io.*emit*('mensaje\_chat', body);

  })

});

* **Probamos en el navegador y cada vez que se abra un nuevo navegador nos aparecerá en el chat que se ha conectado un nuevo usuario.**
* **Número de clientes conectados**

**Número de clientes conectados incluso al que no esté conectado.**

*//Configuración servidor WS*

const io = *socketIO*(server);

io.*on*('connection', (socket) => {

  console.*log*('Se ha conectado un nuevo cliente');

*//desde el server para el que este solo conectado*

  socket.broadcast.*emit*('mensaje\_chat', {

    nombre: 'INFO',

    mensaje: 'Se ha conectado un nuevo usuario'

  });

*//desde el server para todos hasta los desconectados, número de usuarios conectados.*

  io.*emit*('usuarios\_chat', io.engine.clientsCount);

*//emitir desde el server para todo el mundo*

  socket.*on*('mensaje\_chat', (body) => {

    io.*emit*('mensaje\_chat', body);

  })

});

* **Dentro de client-chat.js**
* **Hacer algo cuando me desconecte** [**www.js**](http://www.js)

**Cuando cierre una de esas pestañas con las que estoy trabajando y emitir a todos, y se actualiza de nuevo la lista de usuarios actualizada.**

*//Configuración servidor WS*

const io = *socketIO*(server);

io.*on*('connection', (socket) => {

  console.*log*('Se ha conectado un nuevo cliente');

*//desde el server para el que este solo conectado*

  socket.broadcast.*emit*('mensaje\_chat', {

    nombre: 'INFO',

    mensaje: 'Se ha conectado un nuevo Usuario'

  });

*//desde el server para todos hasta los desconectados numero de usuarios conectados.*

  io.*emit*('usuarios\_chat', io.engine.clientsCount);

*//emitir desde el server para todo el mundo*

  socket.*on*('mensaje\_chat', (body) => {

    io.*emit*('mensaje\_chat', body);

  });

  socket.*on*('disconnect', () => {

    io.*emit*('mensaje\_chat', {

      nombre: 'INFO',

      mensaje: 'Se ha desconectado un Usuario'

    });

  io.*emit*('usuarios\_chat', io.engine.clientsCount);

  });

});

* **Mensaje que yo escrito, ver en mi chat que mi mensaje aparezca de otro color.**

**Dentro de client-chat.js**

**Lo que intentamos en primero que cuando pongamos un mensaje que nos salga la id nuestra.**

const socket = *io*();

const btnEnviar = document.*getElementById*('btnEnviar');

const inputNombre = document.*getElementById*('inputNombre');

const inputMensaje = document.*getElementById*('inputMensaje');

const mensajes = document.*getElementById*('mensajes');

const numUsuarios = document.*getElementById*('numUsuarios')

btnEnviar.*addEventListener*('click', (e) => {

    const body = {

        nombre: inputNombre.value,

        mensaje: inputMensaje.value,

        socketId: socket.id

    }

    socket.*emit*('mensaje\_chat', body);

});

socket.*on*('mensaje\_chat', (body) => {

    const li = document.*createElement*('li');

    li.innerHTML = `<strong> ${body.nombre} : </strong> ${body.mensaje}`;

*if* (body.socketId === socket.id) {

        li.classList.*add*('propietario');

    }

    mensajes.*appendChild*(li);

})

socket.*on*('usuarios\_chat', (clientsCount) => {

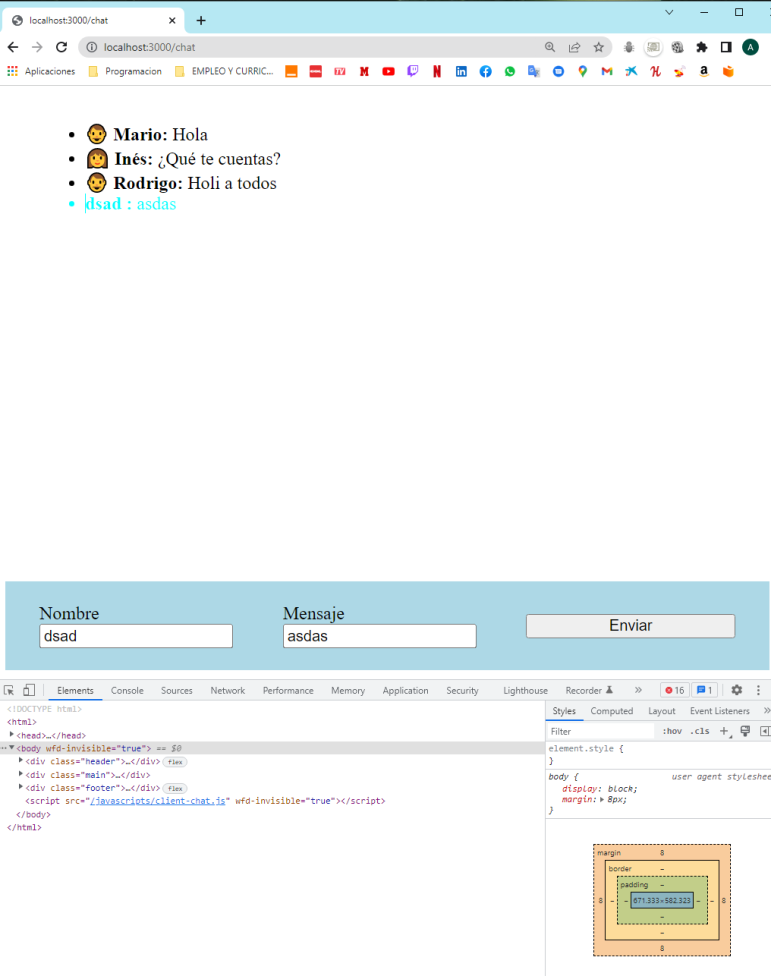
    numUsuarios.innerText = clientsCount

})

**Dento de chat.pug**

**Probar en el navegador**

**Listo**



* **Instalamos librería , dotenv y mongoose**

**Terminal**

**npm install mongoose dotenv**

* **Raiz del proyecto creamos el fichero .env**
* **Dentro de .env**

MONGO\_URL="mongodb://127.0.0.1:27017/chatDB"

* **Raíz del proyecto creamos la carpeta /config**
* **Dentro de la carpeta /config creamos el fichero db.js**
* **Dentro de db.js**

const mongoose = *require*('mongoose');

mongoose.*connect*(process.env.MONGO\_URL);

* **Dentro de** [**www.js**](http://www.js)

*//Config .en*

*require*('dotenv').*config*();

*//Config db*

*require*('../config/db');

* **Creamos carpeta /models**
* **Creamos dentro de /models creamos mensaje.model.js**
* **Dentro de mensaje.model.js**

**Tenemos que generar el esquema.**

**Tener un modelo que me permita guardar en base de datos e interactuar con ella mensajes.**

const mongoose = *require*('mongoose');

const Schema = mongoose.Schema;

const mensajeShema = new *Schema*({

    nombre: String,

    mensaje: String,

    socketId: String

}, { timestamps: *true* });

module.exports = mongoose.*model*('mensaje', mensajeShema);

* **Dentro de** [**www.js**](http://www.js)

**Importamos el modelo de mensajes es decir mensaje.model.js que es donde tenemos el modelo.**

const Mensaje = *require*('../models/mensaje.model');

Que guarde en la base de datos.

*//Configuración servidor WS*

const io = *socketIO*(server);

io.*on*('connection', (socket) => {

  console.*log*('Se ha conectado un nuevo cliente');

*//desde el server para el que este solo conectado*

  socket.broadcast.*emit*('mensaje\_chat', {

    nombre: 'INFO',

    mensaje: 'Se ha conectado un nuevo Usuario'

  });

*//desde el server para todos hasta los desconectados numero de usuarios conectados.*

  io.*emit*('usuarios\_chat', io.engine.clientsCount);

*//emitir desde el server para todo el mundo*

  socket.*on*('mensaje\_chat', *async* (body) => {

*//TODO guardar ese mensaje en la BD*

*await* Mensaje.*create*(body);

    io.*emit*('mensaje\_chat', body);

  });

  socket.*on*('disconnect', () => {

    io.*emit*('mensaje\_chat', {

      nombre: 'INFO',

      mensaje: 'Se ha desconectado un Usuario'

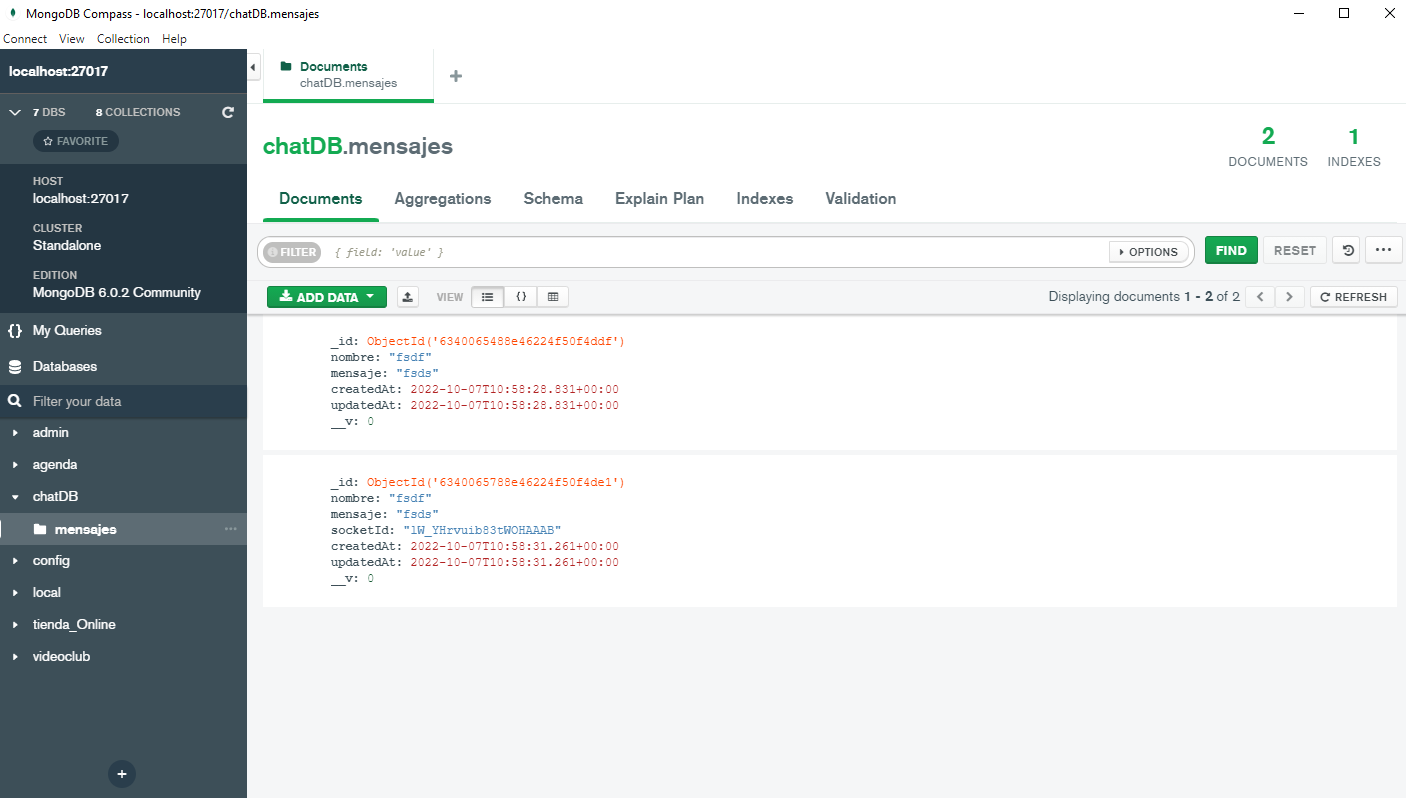
    });

    io.*emit*('usuarios\_chat', io.engine.clientsCount);

  });

});

* **Probamos agregar los mensajes en el navegador, nos debería insertar el mensaje en la BD, dentro de MongoDB Compass lo verificamos.**



* **Cuando cargue la vista quiero que me pase los cinco últimos mensajes.**
* **Dentro de index.js**

**Porque es el único acceso donde puede acceder a la BD**

**-Requerimos el modelo de Mensaje para usarlo.**

const Mensaje = *require*('../models/mensaje.model');

**-Recuperar de la BD los 5 últimos mensajes ( aggregate -> $sort, $limit )**

**Dentro de index.js**

var express = *require*('express');

var router = express.*Router*();

const Mensaje = *require*('../models/mensaje.model');

*/\* GET home page. \*/*

router.*get*('/', function (req, res, next) {

  res.*render*('index', { title: 'Express' });

});

router.*get*('/chat', *async* (req, res) => {

  const mensajes = *await* Mensaje.*aggregate*([

    { $sort: { createdAt: -1 } },

    { $limit: 5 }

  ]);

  console.*log*(mensajes);

  res.*render*('chat', { mensajes: mensajes.*reverse*() });

});

module.exports = router;

**-Le paso el array de mensajes a la vista a través del método render**

**-Dentro de la vista (chat.pug) recorrer el array y mostrar los mensajes dentro del L**

**Dentro de chat.pug**

extends layout.pug

block content

    .header

        h1 Bienvenido al chat

        h2 Usuarios conectados: <span id ="numUsuarios"></span>

    .main

        ul#mensajes

            each mensaje in mensajes

                li <strong>#{mensaje.nombre}</strong> #{mensaje.mensaje}

    .footer

        .control

            label.form-label Nombre

            input#inputNombre.form-control(type="text")

        .control

            label.form-label Mensaje

            input#inputMensaje.form-control(type="text")

        .control

            button#btnEnviar.btn.btn-success Enviar

    script(src="/javascripts/client-chat.js")

**Probar en el navegador la lista de mensajes que nos salen.**