



UNIVERSIDAD DE COLIMA  
Facultad de Telemática

**Ingeniería en Tecnologías de Internet  
Semestre febrero -julio 2021**

Asignatura: Programación distribuida en servicios de internet.

Maestro: Montaña Araujo Sergio Adrian.

“Documentación Chat con NodeJS y Socket.io”

Alumno:

Cesar Adrian Vargas Rangel 4°C

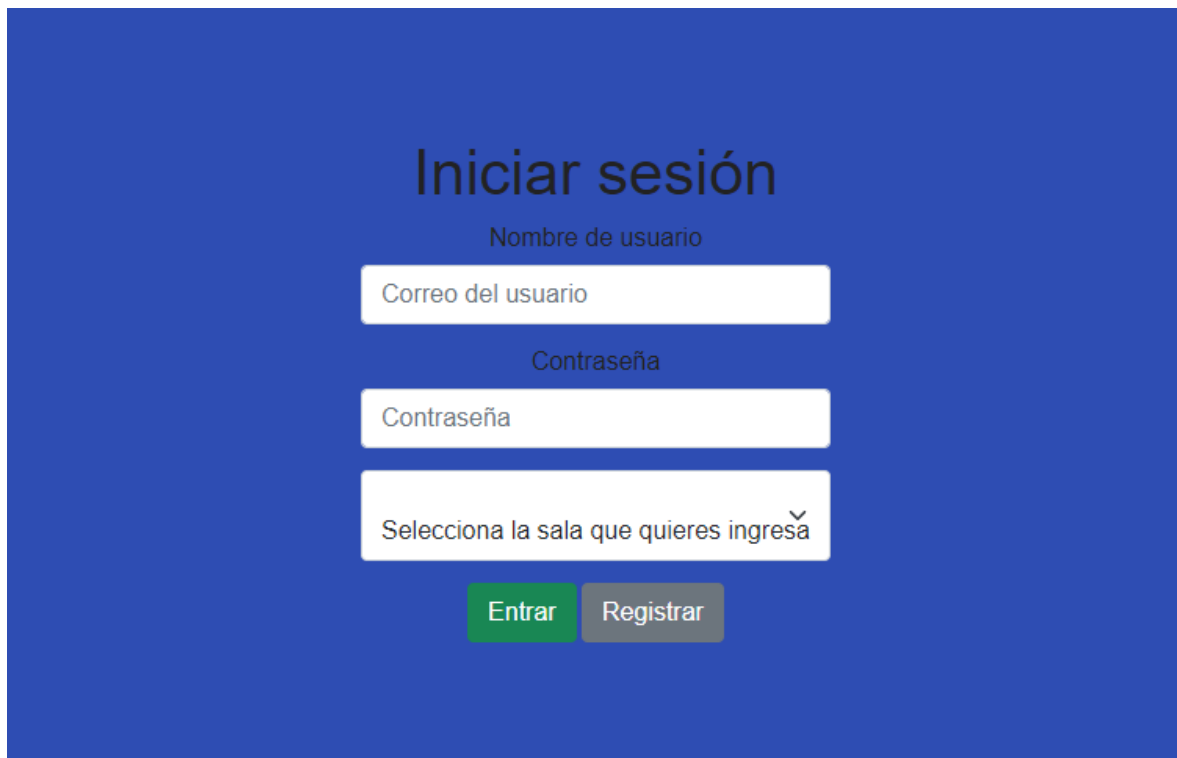
Colima Col., a 25 de junio de 2021

## Índice

Introducción .....	3
Desarrollo .....	4
Index.js.....	5
Index.html .....	9
Conclusión.....	14
Link GitHub.....	14

## Introducción

En este documento se hablará sobre como se llevo a cabo el proyecto de un chat elaborado con HTML, Bootstrap, CSS, NodeJS, Socket.io y la utilización de una base de datos para poder guardar los datos de los usuarios y los mensajes enviados por los mismos, con esto se ha elaborado un chat en el cual se puede entrar y se pueden publicar mensajes de manera instantánea entre dos o más usuarios gracias al Socket.io, en este también se podrán registrar usuarios nuevos para poder acceder al chat, los cuales también pueden cambiar de sala dentro del mismo ya logeados.



## Iniciar sesión

Nombre de usuario

Correo del usuario

Contraseña

Contraseña

Selecciona la sala que quieres ingresar

Entrar Registrar

## Desarrollo

Primero hemos de crear una carpeta donde guardar los archivos, dentro de esta carpeta se guardarán los archivos index.html e index.js, dentro de esta carpeta se instalarán los paquetes necesarios para que funcione el chat los cuales son de NPM.

Los paquetes NPM que se instalaron son los siguientes:

- npm -init: Esto activará la inicialización de tu proyecto. Este comando funciona como una herramienta para crear el archivo package.json de un proyecto.
- npm express: Sirve para la escritura de manejadores de peticiones con diferentes verbos HTTP en diferentes caminos URL (rutas), la integración con motores de renderización de "vistas" para generar respuestas mediante la introducción de datos en plantillas, entre otras cosas.
- npm socket.io: Socket.IO es una biblioteca de JavaScript para aplicaciones web en tiempo real. Permite la comunicación bidireccional en tiempo real entre clientes y servidores web. Tiene dos partes: una biblioteca del lado del cliente que se ejecuta en el navegador y una biblioteca del lado del servidor para Node.js.
- npm install mysql: Este es un controlador node.js para mysql.

- npm install express-session: almacena los datos de sesión en el servidor; sólo guarda el ID de sesión en la propia cookie, no los datos de sesión. De forma predeterminada, utiliza el almacenamiento en memoria y no está diseñado para un entorno de producción.
- npm cookie-parser: nos permite configurar cookies dentro de nuestro servidor.

Todos estos paquetes son necesarios para poder llevar a cabo el chat, ya que son elementos importantes para varias cosas que se realizarán dentro del código.

## Index.js

Por la parte del index.js, se procederá a colocar lo necesario para el funcionamiento:

```
const express = require('express'); //Se llaman los paquetes necesarios
const session = require('express-session');
socket = require('socket.io');
mysql = require('mysql');
cookieParser = require('cookie-parser');

var app = express();
var roomName= ''; //Datos para el cambio de sala
const nameBot= "System"; //Datos para el bot de notificaciones en el chat

var server = app.listen(3030, function () { //Conexion al servidor
  console.log("Servidor en marcha, port 3030.");
});

var io = socket(server);
```

En esta parte se llaman los paquetes de npm que ya se han instalado, se da de alta el servidor y en que puerto se escuchara, en este caso sería en el puerto 3030 y se crea la variable “io” para controlar el socket.io.

```

var sessionMiddleware = session({//Sifrado de contraseñas
  secret: "keyUltraSecret",
  resave: true,
  saveUninitialized: true
});

io.use(function (socket, next) {//Funcion del socket.io
  sessionMiddleware(socket.request, socket.request.res, next);
});

app.use(sessionMiddleware);
app.use(cookieParser());

```

Se utiliza el “sessionMiddleware” para el cifrado de las contraseñas y el “io.use” para obtener los datos del HTML con la función del socket.

```

const config = {//Conexion de la base de datos
  "host": "localhost",
  "user": "root",
  "password": "",
  "base": "chat_progra"
};

var db = mysql.createConnection({
  host      : 'localhost',
  user      : 'root',
  password  : '',
  database  : 'chat_progra'
});

```

También se crea la conexión a la base de datos para poder tener acceso a los datos de los usuarios, las salas creadas previamente en la base de datos y a los mensajes enviados dentro del chat.

```

io.on('connection', function (socket) {
  var req = socket.request;
  console.log(req.session);
  if(req.session.userID != null){//Se comprueba la sesion iniciada
    db.query("SELECT * FROM users WHERE id=?", [req.session.userID], function(err, rows, fields){
      console.log('Sesión iniciada con el UserID: ' + req.session.userID + ' Y nombre de usuario: ' + req.session.Username);
      socket.emit("logged_in", {user: req.session.Username, email: req.session.correo});
    });
  }else{
    console.log('No hay sesión iniciada');
  }
}

```

A partir de aquí se empieza a poner el código necesario para manipular el socket, primero se inicia con una conexión a la base de datos para verificar a los usuarios y ver si están registrados o dar un error de que no se pudo iniciar sesión.

```
socket.on('addUser', function(data){//Se crea la funcion para añadir usuarios
  const user = data.user,
  pass = data.pass,
  email = data.email;

  if(user != "" && pass != "" && email != ""){//Se toman los datos en caso de introducir un nuevo usuario
    console.log("Registrando el usuario: " + user);
    db.query("INSERT INTO users('Username', 'Password', 'email') VALUES(?, ?, ?)", [user, pass, email], function(err, result){
      if(!err)
        throw err;
      console.log(result);
      console.log('Usuario ' + user + " se dio de alta correctamente.");
      socket.emit('UsuarioOK');
    });
  }else{
    socket.emit('vacio');
  }
});
```

Dentro de esta función se coloca lo necesario para añadir un nuevo usuario de ser necesario, se toman los datos como lo son el correo, contraseña y el correo, estos datos se dan de alta en la base datos, para después ser utilizados en el inicio de sesión.

```
socket.on('getSalas', function(data){//Funcion para obtener los datos de las salas de la base de datos
  db.query('SELECT id, nombre_sala FROM salas', function(err, result, fields){
    if(err) throw err;
    socket.emit('Salas', result);
  })
});
```

```
socket.on('cambiodesala', function(data){//Funcion para el cambio de sala dentro del chat
  const idSala = data.idSala,
  nombreSala = data.nombreSala;

  socket.leave(req.session.roomName);

  req.session.roomID = idSala;
  req.session.roomName = nombreSala;

  socket.join(req.session.roomName);
  bottxt('cambioSala');
});
```

En la siguiente parte se crean las funciones para utilizar las salas que se crearon en la base de datos, en la primera imagen se muestra la función para obtener las

salas creadas en la base de datos, y en la siguiente imagen de muestra la función que nos permite el cambio de sala a los usuarios ya dentro del chat.

```
socket.on('salir', function(request, response){
    bottxt('salioUsuario');
    socket.leave(req.session.roomName);
    req.session.destroy();
});
```

A continuación, se crea una función para salir del chat, en la cual se destruye la sesión iniciada ya anteriormente.

```
function bottxt(data){//funcion de un bot para notificar los cambios de sala
    entroSala = 'Bienvenido a la sala ' + req.session.roomName;
    cambioSala = 'Cambiaste a la sala ' + req.session.roomName;
    sefue = 'El usuario ' + req.session.Username + ' ha salido.';

    if(data == "entroSala"){
        socket.emit('mensaje', {
            usuario: nameBot,
            mensaje: entroSala
        });
    }
    if(data == "cambioSala"){
        socket.emit('mensaje', {
            usuario: nameBot,
            mensaje: cambioSala
        });
    }
    if(data == "salioUsuario"){
        socket.emit('mensaje', {
            usuario: nameBot,
            mensaje: sefue
        });
    }
}
});
```

Por último, de ser necesario se puede crear una función la cual tiene el propósito de ser un bot, el cual da una notificación al usuario y a los demás de que ha entrado, salido o cambiado de sala, este bot funciona cuando el usuario ya se a logeado y puede ser útil para saber en qué sala se están mandado los mensajes.



## Index.html

Por la parte del HTML, se crea un menú de inicio de sesión para que el usuario pueda logearse, dentro de este se puede elegir en cual sala se quiere entrar de la misma manera se da la opción con un botón de poder registrar un nuevo usuario en caso de querer dar de alta a uno nuevo.

```
<body class="text-center">
<main class="form-signin">
  <h1>Iniciar sesión</h1>
  <div class="form-group">
    <h6>Nombre de usuario</h6>
    <input type="text" class="form-control" id="userName" placeholder="Correo del usuario" name="username">
  </div><br>
  <div class="form-group">
    <h6>Contraseña</h6>
    <input type="password" class="form-control" id="Password" placeholder="Contraseña" name="password">
  </div>
  <br>
  <div class="form-floating">
    <select class="form-select" name="rooms" id="rooms">
      <option selected>Selecciona la sala que quieres ingresar</option>
    </select>
  </div>
  <br>
  <button class="btn btn-success" type="button" id="Login">Entrar</button>
  <button class="btn btn-secondary" type="button" id="registrar" data-toggle="modal" data-target="#registro">Registrar</button>
</main>
```

# Iniciar sesión

Nombre de usuario

Correo del usuario

Contraseña

Contraseña

Selecciona la sala que quieres ingresar

Entrar Registrar

En esta parte del código se crea lo que es la vista principal donde el usuario puede logearse y agregar uno nuevo.

```
<div id="wrapper" style="display: none;">
  <div id="menu">
    <p>Sala: <b id="SalaNombre"></b></p>
    <p class="bienvenido"> Bienvenido, <b id="usernameTag"></b>, con correo: <b id="emailUser"></b></p>
    <p class="logout"><a id="exit" href="/">Salir del chat</a></p>
  </div>
  <div id="chatbox">
    <!--Caja del chat-->
  </div>
  <div>
    <input name="usermsg" type="text" id="mensaje" size="63" placeholder="Introduzca un mensaje"/>
    <input class="btn btn-secondary btn-sm" type="button" name="submitmsg" id="enviarMensaje" value="Enviar Mensaje"/>
    <select class="form-control" name="roomscambio" id="roomsCambio">
      <option selected>Cambiar de sala</option>
    </select>
  </div>
</div>
```



En esta parte del código se pone lo necesario para poder visualizar el cuadro del chat junto con la sala que se escogió previamente, también agregando lo datos para cuando el usuario a cambiado de sala.

```

<script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
<script src="/socket.io/socket.io.js"></script>
<!--Componentes para funcion del chat-->
<script>
    $(document).ready(function(){
        var socket = io();
        let salas = [];

        socket.emit('getSalas');

        socket.on('Salas', function(data){//Se agrega la funcion de salas
            $.each(data, function(id, val){
                salas.push(data[id].nombre_sala);
            });
            getSalas(salas);
        });

        $('#roomsCambio').change(function(){//Se agrega la funcion para cambiar de salas
            roomID = $(this).val();
            roomName = $(this).find('option:selected').text();
            roomID = roomID + 1;

            $('#SalaNombre').text(roomName);
            $('#chatbox').empty();

            function getSalas(data){
                $.each(data, function(id, val){
                    $('#rooms').append($('', {
                        value: id,
                        text: data[id]
                    }));
                });
                $('#roomsCambio').append($('', {
                    value: id,
                    text: data[id]
                }));
            });
        });
    });
};

```

A partir de aquí se utiliza la función del socket.io que se creó en index.js, en esta parte se obtienen los datos recatados de la base de datos sobre las salas que existen y se mandan al HTML para poder visualizarse por el usuario y poder escoger entre alguna de ellas.

```

$("#Login").click(function(){
  socket.emit("login", {
    user: $("#userName").val(),
    pass: $("#Password").val(),
    roomID: $('#rooms').val(),
    roomName: $('#rooms').find('option:selected').text()
  });
});
$("#sendResgistro").click(function(){
  socket.emit("addUser", {
    user: $("#userNameR").val(),
    pass: $("#PasswordR").val(),
    email: $("#correo").val(),
  });
});

$(".logout").click(function(){
  socket.emit("salir");
});

```

En esta parte del socket.io se obtienen los datos del inicio de sesión que están en el HTML, y se comparan con el que está en la base de datos, también está la función para registrar a los nuevos usuarios, junto con la función de salir del chat.

```

$('#enviarMensaje').click(function(){
  if($("#mensaje").val().length <= 0){
    alert("Escribe el mensaje para poderlo enviar.");
  }else{
    var mensaje = $('#mensaje').val()
    socket.emit('mjsNuevo', mensaje); // Se envia mensaje para la funcion de "mensaje nuevo"
  }
});

```

```

socket.on('mensaje', function(data){ // Función que tiene de respuesta el nuevo mensaje, se conecta y se inserta en la caja del chat.
  if (data.usuario == "System") {
    var nuevoMensaje = '<b>' + data.usuario + ' dice:</b> ' + data.mensaje;
  }else{
    var nuevoMensaje = '<p class="mensajeEnviado"><b>' + data.usuario + ' dice:</b> ' + data.mensaje + '</p>';
  }
  $('#chatbox').append(nuevoMensaje + '<br>');
  $('#mensaje').val("");
});
});

```

Y por último en estas dos imágenes se muestra el cómo trabaja la función para publicar en el chat un mensaje nuevo, pasando desde enviar el mensaje a la base de datos para después publicarlo en el HTML y así que todos los usuarios conectados lo puedan ver.

## Conclusión

En este proyecto e aprendido sobre cómo es que funciona el socket.io y el express para poder crear un chat, de la misma manera como puedo crear una pagina para que un futuro cuando requiera hacer una pagina con un login saber cómo hacerlo, también aprendí como es que funciona el cookie-parser y como crear contraseñas con seguridad, me gusto mucho el poder hacer este proyecto ya que me ayudo a comprender nuevas cosas que me servirán a futuro.

## Link GitHub

[https://github.com/CesarVargas8/Proyecto\\_chat\\_nodejs\\_y\\_socket.io](https://github.com/CesarVargas8/Proyecto_chat_nodejs_y_socket.io)