



CHAT CON NODEJS Y SOCKET.IO

PROGRAMACIÓN DISTRIBUIDA

JOSÉ DE JESUS DÍAZ SALAZAR
FACULTAD DE TELEMÁTICA

INTRODUCCION

En este Proyecto abordaremos la implementación de NODE.JS y su Framework más famoso y utilizado a nivel mundial, Express.js para hacer la parte del back end, junto de la mano de socket.io para realizar en conjunto a una base de datos SQL, en este caso MySql un chat.

¿Qué son estas cosas que te acabo de mencionar?

Node.js es un entorno de tiempo de ejecución de JavaScript (de ahí su terminación en .js haciendo alusión al lenguaje JavaScript). Este entorno de tiempo de ejecución en tiempo real incluye todo lo que se necesita para ejecutar un programa escrito en JavaScript. También Node.js utiliza un modelo de entrada y salida sin bloqueo controlado por eventos que lo hace ligero y eficiente (con entrada nos referimos a solicitudes y con salida a respuestas). Puede referirse a cualquier operación, desde leer o escribir archivos de cualquier tipo hasta hacer una solicitud HTTP.

Socket.io en un principio estaba pensado para trabajar con servidores escritos en NodeJS. Socket.IO es una biblioteca de JavaScript para aplicaciones web en tiempo real, pero gracias a su éxito ha sido portado a muchos otros lenguajes como enseñaré en la siguiente lista:

- GO
- PYTHON
- JAVA

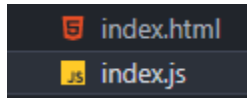
Nosotros vamos a hablar aquí de tecnologías web, por lo tanto vamos a ver como implementar un simple chat usando la librería oficial de JavaScript de socket.io e implementando socket.io en un servidor escrito en NodeJS con el framework express.

MySQL es un sistema de gestión de bases de datos que cuenta con una doble licencia. Por una parte es de código abierto, pero por otra, cuenta con una versión comercial gestionada por la compañía Oracle. Actualmente, es la base de datos de código abierto más famosa y utilizada en el mundo entero.

MySQL sirve para almacenar toda la información que se desee en bases de datos relacionales, como también para administrar todos estos datos sin apenas complicaciones gracias a su interfaz visual y a todas las opciones y herramientas de las que dispone. Es algo esencial, sobre todo en webs que cuentan con la opción de registrar usuarios para que inicien sesión.

DESARROLLO

Empezamos el proyecto abriendo nuestro editor de código, visual studio code y creando 2 documentos, index.js e index.html.



Después instalamos las dependencias que ocuparemos con npm install (dependencias) y las crearemos constantes para después requerirlas.

```
const express = require('express');
const socket = require('socket.io');
const mysql = require('mysql');
const cookieParser = require('cookie-parser');
const session = require('express-session');
```

Posteriormente añadiremos una variable = server, asignando el puerto en el que funcionará nuestra app localmente, en este caso es el 3000.

```
var app = express();
var roomName = '';
const nameBot = "BotChat";
const port = process.env.PORT || 3000;
```

```
var server = app.listen(port, function () {
  console.log("Servidor en marcha, port.", port);
});
```

El paso siguiente es declarar socket, como var io = socket.

Y declarar el middleware que en este caso es sesión middleware

```
var io = socket(server);

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

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

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

Iniciaremos una configuración de la base de datos y la guardaremos en la variable db, llenado los campos de host, user, password y nombre de la base de datos

```
const config = {
  "host": "localhost",
  "user": "root",
  "password": "",
  "base": "chat"
};

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

Como paso siguiente tenemos que configurar las conexiones de socket y como tal, el funcionamiento que tendrá nuestro chat, dentro de los cuales se encuentra el login, aquí entra en función MySQL, así como los requerimientos que se piden de el historial, manejo de salas y el registro de los usuarios.

```
socket.on("login", function (data) {
  console.log(' console log de la data: ${data}');
  console.log(data);
  const user = data.user,
        pass = data.pass;
  roomId = data.roomID;
  roomName = data.roomName;

  db.query("SELECT * FROM users WHERE Username=?", [user], function (err, rows, fields) {
    if (rows.length == 0) {
      console.log("El usuario no existe, favor de registrarse!");
    } else {
      console.log(rows);

      const dataUser = rows[0].Username,
            dataPass = rows[0].Password,
            dataCorreo = rows[0].email;

      if (dataPass == null || dataUser == null) {
        socket.emit("error");
      }
      if (user == dataUser && pass == dataPass) {
        console.log("Usuario correcto!");
        socket.emit("logged_in", { user: user, email: dataCorreo, room: roomName, roomID: roomId });
        req.session.userID = rows[0].id;
        req.session.Username = dataUser;
        req.session.correo = dataCorreo;
        req.session.roomID = roomId;
        req.session.roomName = roomName;
        req.session.save();
        socket.emit('armadoHistorial');
        socket.join(req.session.roomName);

        console.log('aqui va el id de room id: ' + req.session.roomID);
        console.log(req.session);

        bottxt('entroSala');
      } else {
        socket.emit("invalido");
      }
    }
  });
});
```

```

socket.on('historial', function () {
  console.log('Buscamos historial de la sala: ' + req.session.roomName);

  db.query('SELECT s.nombre_sala, u.Username, m.mensaje FROM mensajes m INNER JOIN salas s ON s.id = m.sala_id INNER JOIN users u ON u.id = m.user_id WHERE m.sala_id = ' +
    req.session.roomID + ' ORDER BY m.id ASC', function (err, rows, fields) {
    socket.emit('armadoHistorial', rows);
    console.log('rows: ' + rows);
  });
});

socket.on('addUser', function (data) {
  const user = data.user,
        pass = data.pass,
        email = data.email;

  if (user != "" && pass != "" && email != "") {
    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');
  }
});

```

```

socket.on('cambioSala', function (data) {
  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');
});

socket.on('mjsNuevo', function ({data}) {
  // id de la sala
  db.query("INSERT INTO mensajes('mensaje', 'user_id', 'sala_id', 'fecha') VALUES (5, 5, 5 , CURDATE())", [data, req.session.userID, req.session.roomID],
    (err, result) => {
      if (!!err)
        throw err;

      console.log(result);

      console.log('Mensaje dado de alta correctamente.');
```

```

      socket.broadcast.emit('mensaje', {
        usuario: req.session.Username,
        mensaje: data
      });

      socket.emit('mensaje', {
        usuario: req.session.Username,
        mensaje: data
      });
    });
});

```

```

socket.on('getSalas', function (data) {
    db.query('SELECT id, nombre_sala FROM salas', function (err, result, fields) {
        if (err) throw err;
        socket.emit('Salas', result);
    });
});

socket.on('salir', function (request, response) {
    req.session.destroy();
});

function bottxt(data) {
    entroSala = 'Bienvenido a la sala ' + req.session.roomName;
    cambioSala = 'Cambiaste de sala a ' + req.session.roomName;
    sefue = 'El usuario ' + req.session.Username + 'ha salido de sala.'

    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
        });
    }
}

app.post('/auth', function (request, response) {
    var username = request.body.username;
    var password = request.body.password;

```

```

app.post('/auth', function (request, response) {
    var username = request.body.username;
    var password = request.body.password;

    if (username && password) {

        connection.query('SELECT * FROM users WHERE username = ? AND password = ?'[username, password], function (err, results, fields) {
            if (results.length > 0) {
                request.session.loggedin = true;
                request.session.username = username;
                response.redirect('/home');
            } else {
                response.send('Usuarios y/o contraseña incorrectos');
            }
            response.end();
        });
    } else {
        response.send('Ingresa usuario y contraseña');
        response.end();
    }
});

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

```

Tenemos la otra parte que se ha estado haciendo a la parte, el Front end, en este caso es el index.html, donde se dará la vista y diseño de lo que el usuario final observará.

```
</main>

<div id="wrapper" style="display: none;">
  <div id="menu">
    <p class="bienvenido"> Welcome, <b id="usernameTag"></b>, email : <b id="emailUser"></b>, Room: <b id="SalaNombre"></b></p>
    <button class="logout btn btn-danger "><a id="exit" href="/" >Exit</a></button>
  </div>

  <div id="chatbox" class=" form-control">
    <!-- Caja del chat que contendrá todos Los mensajes. -->
  </div>

  <input name="usermsg" type="text" id="mensaje" size="63"/>
  <input class="btn-primary" type="button" name="submitmsg" id="enviarMensaje" value="Enviar Mensaje" />

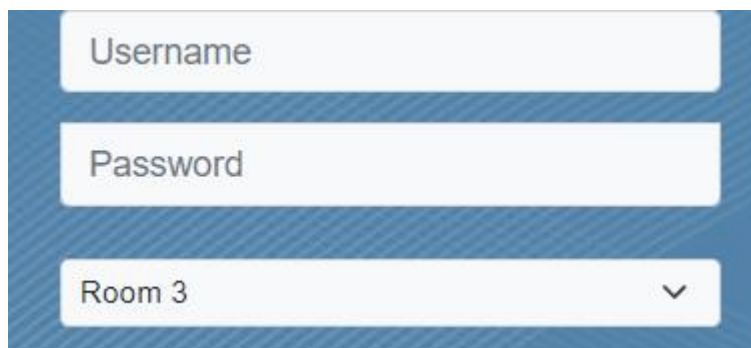
  <select class="form-group bg-secondary" name="roomsCambio" id="roomsCambio">
    <option selected>Room 1</option>
    <option selected>Room 2</option>
    <option selected>Room 3</option>
  </select>
</div>

<!-- aqui va lo de cambio de sala-->

<!-- Modal -->
<div class="modal fade" id="registro" tabindex="-1" role="dialog" aria-labelledby="exampleModallabel" aria-hidden="true">
<div class="modal-dialog" role="document">
  <div class="modal-content">
    <div class="modal-header">
      <h5 class="modal-title" id="exampleModallabel">Register</h5>
      <button type="button" class="close" data-dismiss="modal" aria-label="Close">
        <span aria-hidden="true">&times;</span>
      </button>
    </div>

    <div class="modal-body">
      <div class="form-group">
        <input type="text" class="form-control" id="userNameR" placeholder="Username" name="username" required>
        <br>
      </div>

      <div class="form-group">
        <input type="password" class="form-control" id="PasswordR" placeholder="Password" name="password" required>
        <br>
      </div>
    </div>
  </div>
</div>
```



Username

Password

Room 3

Al igual que se estará utilizando bootstrap4 y css para dar diseño a gusto propio y tener el control completo de los diseños.

```
<style>
  html,
  body {
    height: 100%;
  }

  body {
    display: flex;
    align-items: center;
    padding-top: 40px;
    padding-bottom: 40px;
    background-color: #f5f5f5;
    background-image: url('./img/fondo.jpg');
  }

  .form-signin {
    width: 100%;
    max-width: 330px;
    padding: 15px;
    margin: auto;
  }

  body {
    font: 12px arial;
    color: #222;
    text-align: center;
    padding: 35px;
  }

  form, p, span {
    margin: 0;
    padding: 0;
  }

  input { font: 12px arial; }

  a {
    color: #0000FF;
    text-decoration: none;
  }

```


Un modal para el registrar.

```
<!-- Login -->
<link href="signin.css" rel="stylesheet">
/head>
body class="bg-info text-center" >
  <main class="form-signin ">
    
    <h1 class="h2 mb-3 fw-normal">Sign In</h1>

    <div class="form-group">
      <input type="text" class="form-control bg-light" id="userName" placeholder="Username" name="username">
      <br>
    </div>

    <div class="form-group">
      <input type="password" class="form-control bg-light" id="Password" placeholder="Password" name="password">
      <br>
    </div>

    <div class="form-group">
      <select class="form-select form-select-sm bg-light" aria-label=".form-select-lg example" name="rooms" id="rooms" >
        <option selected>Room 1</option>
        <option selected>Room 2</option>
        <option selected>Room 3</option>
      </select>
    </div>

    <br>
    <button class="w-100 m-100 btn btn btn-primary form-group" type="button" id="Login">Join</button><br>

    <button class="w-100 m-100 btn btn btn-secondary form-group" type="button" id="registrar" data-toggle="modal" data-target="#reg
```

Nos encontraremos por último un script en el HTML en el cual estaremos implementando jQuery, donde tendremos el botchat, que nos recibe cuanto entramos, cuando presionemos los botones mande los valores al index.js, que pinte la pantalla con los mensajes de las distintas salas, mensajes si todo funciona bien o si tienen algún error y este mismo error.

```
<script>
$(document).ready(function(){
  var socket = io(); /*declaramos el socket*/
  let salas=[];
  socket.emit('getSalas');

  socket.on('Salas', function(data){
    console.log(data);
    $.each(data, function(id,val){

      $('#rooms').append($('', {
        value: data[id].nombre_sala,
        text: data[id].nombre_sala,
        id: data[id].id
      }));

      $('#roomsCambio').append($('', {
        value: data[id].nombre_sala,
        text: data[id].nombre_sala,
        id: data[id].id
      }));
    });
  });

  $('#roomsCambio').change(function(){
    roomId = $(this).find('option:selected').attr('id');
    roomName = $(this).find('option:selected').text();
    $('#SalaNombre').text(roomName);
    $('#chatbox').empty();

    socket.emit('cambioSala', {
      idSala: roomId,
      nombreSala: roomName
    });
    socket.emit('historial');
    console.log('cambio select a ID: ' + roomId + ' con nombre: ' + roomName);
  });
});
```

```

socket.on("logged_in", function(data){
    console.log(data);
    $(".form-signin").hide(); /*se esconde el formulario de inicio de sesión*/
    $("#wrapper").show(); /*se muestra la ventana del chat*/
    $('#usernameTag').text(data.user);
    $('#emailUser').text(data.email);
    $('#SalaNombre').text(data.roomName);
    socket.emit('historial');
});
/*sirven para verificar campos y mostrar que hace falta*/
socket.on("invalido", function(){
    alert("Usuario y/o contraseña incorrectos.");
});

socket.on("error", function(){
    alert("Error: Intenta de nuevo!");
});

socket.on("vacio", function(){
    alert("Error: Llena todos los campos!");
});
/*****/
socket.on("UsuarioOK", function(){ /*se muestra al aceptarse el registro del usuario*/
    $('#registro').modal('hide');
    alert("Dado de alta correctamente.");
});

socket.on('mensaje', function(data){ // Función que tiene de respuesta el nuevo mensaje, concatenamos e insertamos en la caja de chat.
    if(data.usuario == "BotChat"){
        var nuevoMensaje = '<small class="bot"><b>' + data.usuario + ' -</b> ' + data.mensaje + '</small>';
        $('#chatbox').append(nuevoMensaje + '</br>');
        $('#mensaje').val("");
    }else{
        var nuevoMensaje = '<p class="mensajeEnviado"><b>' + data.usuario + ' dice:</b> ' + data.mensaje + '</p>';
    }
    $('#chatbox').append(nuevoMensaje + '</br>');
    $('#mensaje').val("");
});

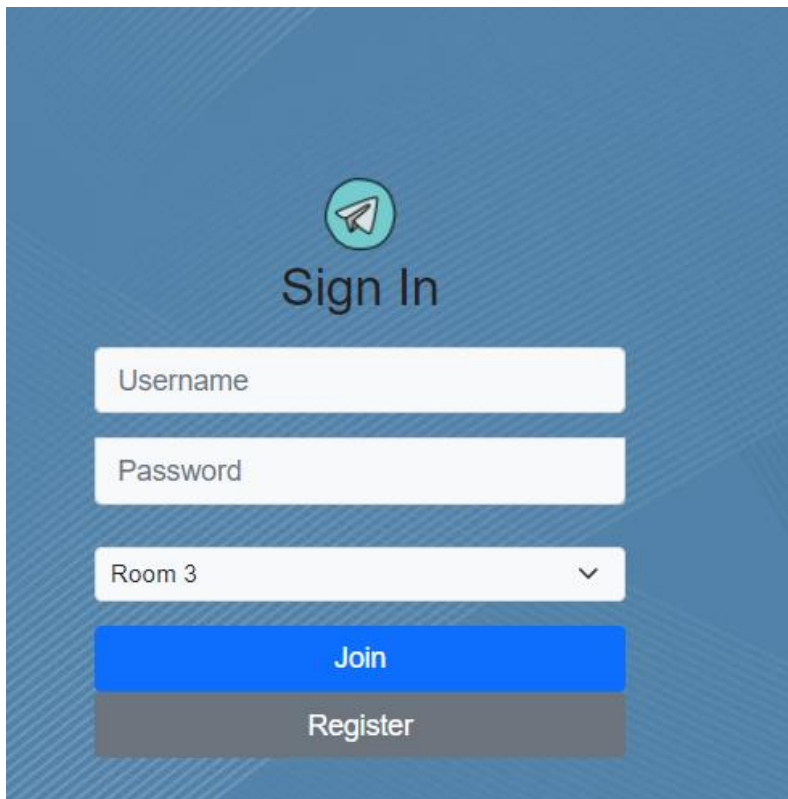
socket.on('armadoHistorial', function(data){ /*requerimos la funcion armadoHistorial para mostrar el historial de la sala*/
    var historial = "";
    $.each(data, function(id, val){
        historial += '<p class="mensajeEnviado"><b>' + data[id]['Username'] + ' dijo:</b> ' + data[id]['mensaje'] + '</p>';
    });

    historial += '<small class="bot"><b>BotChat -</b> Últimos mensajes del historial de la sala</small>';

    $('#chatbox').append(historial + '</br>');
});

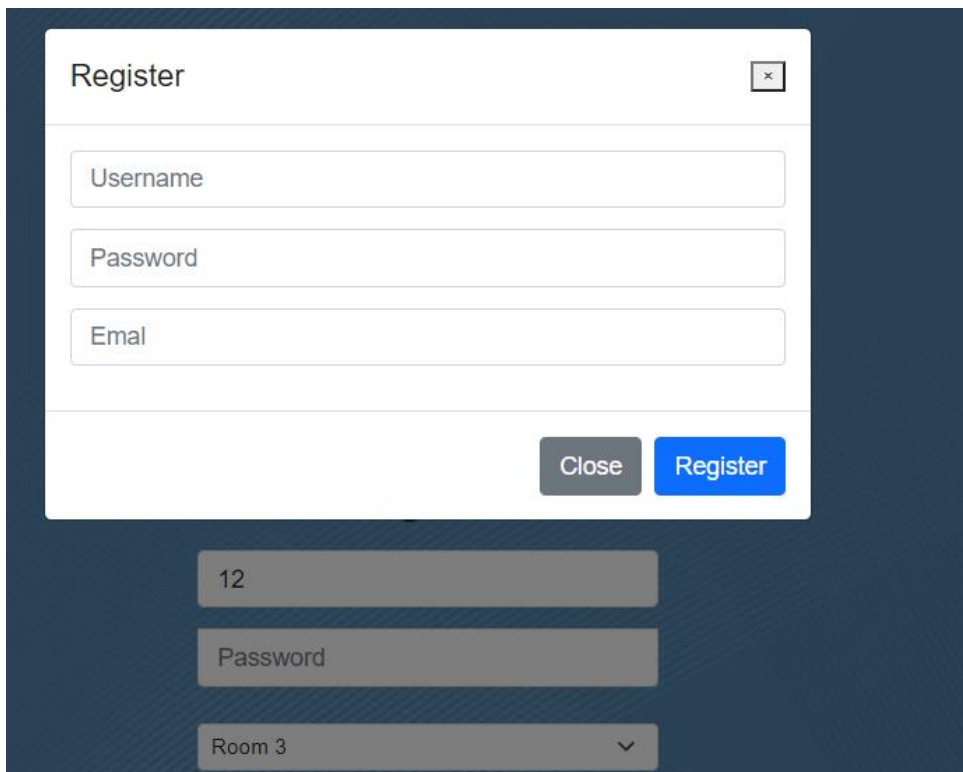
```

Por ultimo tenemos ya el producto final, de como se ve el login.}



A sign-in form with a blue background and a subtle pattern. At the top center is a circular icon containing a white paper airplane. Below the icon is the text "Sign In". The form consists of three input fields: "Username", "Password", and "Room 3" (which is a dropdown menu). Below these fields are two buttons: a blue "Join" button and a grey "Register" button.

De como se ve el modal del register.



A register modal window with a white background and a dark blue border. The modal has a title bar with the text "Register" and a close button (an 'x' in a square). Inside the modal are three input fields: "Username", "Password", and "Email". Below these fields are two buttons: a grey "Close" button and a blue "Register" button. The modal is displayed over a dark blue background that shows a portion of the sign-in form from the previous image, including the "Room 3" dropdown and the "Join" and "Register" buttons.

Y del funcionamiento del chat, de como se ven los chats y los botones de salas, así como el botón de salida.



Conclusión

Gracias a este proyecto que llevamos durante la parcial, aprendí a implementar algo desconocido para mí, que es el socket.io con express y así saber como se puede lograr hacer un chat en real time como las grandes compañías lo hacen así como Facebook, whatsapp, telegrama etc. Me dio a entender que tan importante es Node en la actualidad y la potencia y lo poderoso que pude llegar a ser, manejándose inclusive con stacks basados en JS así como MERN, MEAN. MEVN entre otros.

Glosario

NODE.JS = Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google.

SOCKET.IO = Socket.IO es una biblioteca de JavaScript para aplicaciones web en tiempo real.

BD O DB = Base de datos o data base

JS = JavaScript

HTML = Hyper Text Markup Language

MYSQL = MySQL es un sistema de gestión de bases de datos relacional basado en SQL desarrollado bajo licencia dual.