CHAT CON NODE.JS Y SOCKET.IO

26 JUNIO

.

Creado por: Loredo Ramirez David de Jesus

INTRODUCCION

En este proyecto tuvo como fin poder desarrollar nuevos conocimientos haciendo uso de tecnologías agiles para llevar a cabo la codificación de tal proyecto.

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) 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. Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables, como, por ejemplo, servidores web. Fue creado por Ryan Dahl en 2009 y su evolución está apadrinada por la empresa Joyent, que además tiene contratado a Dahl en plantilla.

De esta forma, usando como backend a Node.js, de igual forma usamos la tecnología Socket.io, Socket.io es una librería que funciona tanto en cliente como servidor precisamente para conseguir la conexión bidireccional que nos permite controlar eventos en tiempo real a través de conexiones TCP y gracias a esto podemos evitar problemas de compatibilidad que hoy en día es un problema que debemos de tener en cuanta a la hora de desarrollar un software.

DESARROLLO (Backend)

Dando inicio al desarrollo, recomiendo usar Visual Studio Code, en nuestra terminal pondremos el siguiente código, “npm install (nombre-dependencia)”, las dependencias son “express,socket.io,mysql,cookie-parser y express-session”, enseguida configuraremos de panera global el puerto, que en mi caso seria el 3000.
Finalmente configuraremos nuestra base de datos.

index.js > io.on('connection') callback > db.query("SELECT * FROM users WHERE id=?") ca

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

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

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

var io = socket(server);

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

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

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

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

Como siguiente paso pasaremos a configurar las conexiones de socket.io para que nuestro chat funcione correctamente.

```
socket.on("login", function (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.join(req.session.roomName);
                socket.emit('armadoHistorial');
                console.log(req.session);
                bottxt('entroSala');
            } else {
                socket.emit("invalido");
            }
        }
    });
});
```

```
socket.on('historial', function () {
    console.log('Buscando 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
        req.session.roomID + ' ORDER BY m.id ASC', function (err, rows, fields) {
            socket.emit('armadoHistorial', rows);
            console.log(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(?, ?, ?, CURDATE())", [data, req.session.userI
        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;

    if (username && password) {


        connnection.query('SELECT * FROM users WHERE username = ? AND password = ?'[username, password], function (err, results,
            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();
    }
});
});
```

DESARROLLO(Frontend)

Una vez que tenemos nuestro Backend terminado, pasaremos a realizar la parte interactiva, donde el usuario tendrá que tener interactividad, por lo tanto debe de ser algo fácil de usar, y de entender claro esta.

Este Frontend fue hecho con la ayuda de HTML para poder maquetar las vistas y JQuery, para poder renderizar en las vistas los datos necesarios para poder dar una buena impresión al usuario.

A continuación se muestra la vista que en primeras instancias, el usuario tendrá una interactividad para poder iniciar sesión y poder registrarse de una manera sencilla.

```
<main class="form-signin">
    <h1 class="h2 mb-3 fw-normal">Iniciar sesión</h1>

    <div class="form-floating">
        <input type="text" class="form-control bg-light" id="userName" placeholder="name@example.com"
            name="username">
        <label for="floatingInput">Nombre de usuario</label>
    </div> <br>
    <div class="form-floating">
        <input type="password" class="form-control bg-light" id="Password" placeholder="Password" name="password">
        <label for="floatingPassword">Contraseña</label>
    </div>
    <div class="form-floating">
        <select class="form-select form-select-sm bg-light" aria-label=".form-select-lg example" name="rooms"
            id="rooms">
            <option selected>Selecciona la sala a ingresar</option>
        </select>
    </div>
    <br>
    <button class="w-100 btn btn-lg btn-primary" type="button" id="Login">Entrar</button> <br> <br>
    <button class="w-100 btn btn-lg btn-warning" type="button" id="registrar" data-toggle="modal"
        data-target="#registro">Registrar</button>
        <p class="mt-5 mb-3 text-muted">&copy; Programación distribuida - 2021</p>
</main>
```

Pero como todo, debe de tener un apartado por si el usuario no esta registrado y desea entrar.

```
<!-- 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">Registro</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-floating">
                    <input required type="text" class="form-control" id="userNameR" placeholder="name@example.com"
                        name="username" required>
                    <label for="floatingInput">Nombre de usuario</label>
                </div> <br>
                <div class="form-floating">
                    <input type="password" class="form-control" id="PasswordR" placeholder="Password"
                        name="password" required>
                    <label for="floatingPassword">Contraseña</label>
                </div> <br>
                <div class="form-floating">
                    <input type="email" class="form-control" id="correo" placeholder="correo" name="correo"
                        required>
                    <label for="floatingPassword">Correo</label>
                </div>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-danger" data-dismiss="modal">Cerrar</button>
                <button type="button" class="btn btn-primary" id="sendResgistro">Registrar</button>
            </div>
        </div>
    </div>
</div>
```

Y como comente, utilizamos JQuery para poder renderizar las vistas, jQuery es una biblioteca multiplataforma de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.
El cual decidimos usar porque es una tecnología que es muy fácil de implementar en proyectos con un grado de complejidad bajo.

```
$(document).ready(function () {

    var socket = io();
    let salas = [];
    socket.emit('getSalas');

    socket.on('Salas', function (data) {
        $.each(data, function (id, val) {
            $('#rooms').append($('<option>', {
                value: data[id].nombre_sala,
                text: data[id].nombre_sala,
                id: data[id].id
            }));
            $('#roomsCambio').append($('<option>', {
                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);
    });
```

```
$("#Login").click(function () {
    socket.emit("login", {
        user: $("#userName").val(),
        pass: $("#Password").val(),
        roomID: $('#rooms').find('option:selected').attr('id'),
        roomName: $('#rooms').find('option:selected').text()
    });
    console.log(roomID);
});

$("#sendResgistro").click(function () {
    socket.emit("addUser", {
        user: $("#userNameR").val(),
        pass: $("#PasswordR").val(),
        email: $("#correo").val(),
    });
});

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

$('#registrar').click(function () {
    $("#userNameR").val("");
    $("#PasswordR").val("");
    $("#correo").val("");
});

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

    }
});
```

```
socket.on("logged_in", function (data) {
    console.log(data);
    $(".form-signin").hide();
    $("#wrapper").show();
    $('#usernameTag').text(data.user);
    $('#emailUser').text(data.email);
    $('#SalaNombre').text(data.roomName);
    socket.emit('historial');
});

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 las campos.!");
});

socket.on("UsuarioOK", function () {
    $('#registro').modal('hide');
    alert("Dado de alta correctamente.");
});
```

```
    socket.on('mensaje', function (data) {
        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) {
        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</samll>';

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

});
```

CONCLUSION

Gracias a este proyecto, se me ocurrieron muchas cosas y muchas maneras para realizar un chat, pudiendo implementarle demás funciones y no solo las básicas.
también tenia mucho que no usaba JQuery, y recordé porque muchas empresas lo siguen usando, ya que es muy simple de implementar y fácil, y lo mejor que hay mucha documentación de ello.

GLOSARIO

NODE.JS: Node.js es un entorno en tiempo de ejecución multiplataforma de código abierto.
SOCKET.IO: Es una biblioteca de JS para aplicaciones web en tiempo real.
HTML: Por sus siglas en inglés de HyperText Markup Language, hace referencia al lenguaje de marcado para la elaboración de páginas web.
JS O JAVASCRIP: Es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.