Proyecto: Chat con NodeJS

PROGRAMACIÓN DISTRIBUIDA

LUIS ALBERTO MONTANO RUVALCABA
VILLA DE ALVAREZ, COLIM 2021 |

Contenido

Introducción

El presente proyecto fue desarrollado con la intención de generar nuevos conocimientos en el área de programación, entre ellos el descubrimiento de una nueva implementación de JavaScript, más conocido como NodeJS. ¿Qué es NodeJS? 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 aporta muchos beneficios y soluciona muchísimos problemas, por lo que sería más que interesante realizar nuestro curso de Node.js para obtener las bases, conceptos y habilidades necesarias que nos motiven a profundizar en sus opciones e iniciar la programación.

Node.js fue creado por los desarrolladores originales de JavaScript. Lo transformaron de algo que solo podía ejecutarse en el navegador en algo que se podría ejecutar en los ordenadores como si de aplicaciones independientes se tratara. Gracias a Node.js se puede ir un paso más allá en la programación con JavaScript no solo creando sitios web interactivos, sino teniendo la capacidad de hacer cosas que otros lenguajes de secuencia de comandos como Python pueden crear.

De esta forma, mediante el uso de esta tecnología se creó el back-end de nuestra aplicación de chat, no sin antes haber implementado otra herramienta importante llamada Socket.io: Socket.io es una librería que nos permite controlar eventos en tiempo real a través de conexiones TCP y nos ayuda a evitar problemas de compatibilidad entre equipos.

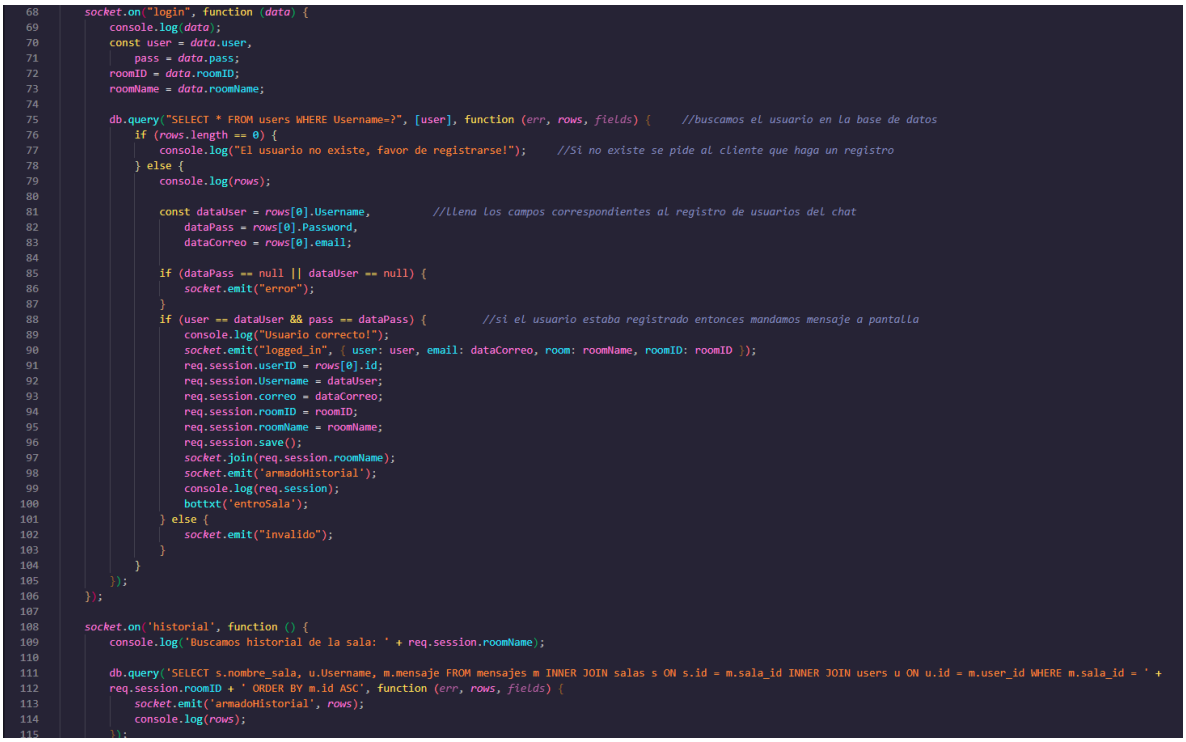
Está desarrollado completamente en JavaScript y, su objetivo es hacer que las aplicaciones en tiempo real tengan posibilidad de ejecutarse en cualquier navegador, incluidos los dispositivos móviles, salvando las diferencias entre los diferentes protocolos.

Desarrollo

Para realizar el chat, comenzamos con declarar las librerías que usaremos a lo largo del proyecto y que a su vez harán que todo el desarrollo sea posible de una manera óptima, seguido de ello añadimos una variable asignando el puerto en que correrá nuestra app localmente, o el puerto donde se visualizará una vez desplegada. Proseguimos declarando middleware hasta llegar a la configuración de la base de datos (previamente realizada con MySQL), tal como se muestra en la siguiente imagen:

```
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 en marcha, port.", port);
});

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());

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

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

Una vez hecho el paso anterior, procedemos a configurar la conexión con socket, escribimos el código que definirá el comportamiento de la app, tales como el login (que en esta parte realizamos consultas para comparar datos ingresados por el usuario con datos en la DB), el historial y el registro de usuario, contenido que encontraremos en la figura siguiente:



```
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) {     //buscamos el usuario en la base de datos
        if (rows.length == 0) {
            console.log("El usuario no existe, favor de registrarse!");     //Si no existe se pide al cliente que haga un registro
        } else {
            console.log(rows);

            const dataUser = rows[0].Username,          //llena los campos correspondientes al registro de usuarios del chat
                dataPass = rows[0].Password,
                dataCorreo = rows[0].email;

            if (dataPass == null || dataUser == null) {
                socket.emit("error");
            }
            if (user == dataUser && pass == dataPass) {         //si el usuario estaba registrado entonces mandamos mensaje a pantalla
                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('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);
    });
```

Una vez se han terminado de escribir los módulos necesarios para la correcta funcionalidad de la app, continuamos con el desarrollo del front-end y ello lo crearemos mediante el uso de Bootstrap y algunas opciones con meramente CSS puro, que pueden ser apreciadas en el ejemplo de abajo:

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

    body {
        display: flex;
        align-items: center;
        padding-top: 40px;
        padding-bottom: 40px;
        background-color: #f5f5f5;
    }

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

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

Iniciamos con el maquetado del login, donde se usan diversas etiquetas HTML para mostrar al usuario dónde ingresar los datos que vaya necesitando, en cuanto a los estilos visuales estos son manejados mediante clases de Bootstrap, los cuales nos facilitan mucho el diseño de nuestro sistema web.

```
<body class="text-center">
    <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>
        <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>
        <button class="w-100 btn btn-lg btn-secondary" type="button" id="registrar" data-toggle="modal"
            data-target="#registro">Registrar</button>
        <p class="mt-5 mb-3 text-muted">&copy; Chat NodeJS and Socket.io</p>
    </main>
```

Llegado a este punto nos restará solamente configurar nuestro código de Socket, para ello lo introduciremos como script debajo del HTML que hemos escrito previamente e iniciaremos declarando una función que se encargará de hacer que todo se ejecute correctamente cuando sea necesario, ya que esta es la que se encargará de la interacción del usuario. Entre lo más destacable encontramos código que se encarga de extraer datos de la BD y presentarlos en la página.

```
271     <script>
272         $(document).ready(function () {
273
274             var socket = io();  /*declaramos el socket*/
275             let salas = [];
276             socket.emit('getSalas');
277
278             socket.on('Salas', function (data) {
279                 $.each(data, function (id, val) {
280                     $('#rooms').append($('<option>', {
281                         value: data[id].nombre_sala,
282                         text: data[id].nombre_sala,
283                         id: data[id].id
284                     }));
285                     $('#roomsCambio').append($('<option>', {
286                         value: data[id].nombre_sala,
287                         text: data[id].nombre_sala,
288                         id: data[id].id
289                     }));
290                 });
291             });
292
293             $('#roomsCambio').change(function () {
294                 roomID = $(this).find('option:selected').attr('id');
295                 roomName = $(this).find('option:selected').text();
296
297                 $('#SalaNombre').text(roomName);
298                 $("#chatbox").empty();
299
```

Conclusión

Finalmente, al término del proyecto conocí dos nuevas implementaciones de JavaScript, NodeJS y Socket.io, con los cuales, en el caso de Socket.io, existen muchas aplicaciones y ejemplos que se pueden hacer. Por ejemplo, controlar un auto vía web con node.js instalado en Raspberry Pi, control de Arduino con node.js, o incluso un chat entre distintos equipos. En cuanto a Node, es una de las tecnologías más usadas hoy en día y se ha convertido en una de las plataformas más populares utilizadas para el desarrollo de aplicaciones web, aplicaciones de escritorio y servicios. Con este proyecto darse cuenta de ello fue fácil, interesante y dinámico.

Glosario

NodeJs: Librería de Javascript

Socket.io: Librería para JavaScript.

BD o DB: Base de datos.

Código: Instrucciones escritas en un lenguaje de programación.

JavaScript: Lenguaje de programación.

HTML: Lenguaje de marcado/etiquetado.