# UNIVERSIDAD DE COLIMA

**Facultad de telemática**

**Ingeniería en tecnologías de la Internet**

**Materia. –** Programación Distribuida

**Maestro. –**Montaño Araujo Fermín Adrián

**Titulo. –**Chat Online con Node js y Sockey.io

**Alumnos.-** Omar Velasco Guzmán

**Grado.-**4

**Grupo.-**C

**FECHA DE ENTREGA.-** 25 de junio del 2021

# Índice

# Introducción

En el presente documento se hablará sobre cómo elaborar un chat online. Se detallará lo que fue necesario hacer para llegar a nuestro objetivo final que es crear una comunicación entre usuarios en salas de chat, permitiéndoles incluso el observar los mensajes mandados con anterioridad, gracias a sus funciones. Esto con la finalidad de implementar lo visto durante el semestre en la materia de Programación Distribuida en un solo proyecto final, donde se pudiera aplicar lenguajes como JavaScript, entornos de JavaScript como Node.Js o bibliotecas de JavaScript como Socket.io. Así como frameworks como Express Js, o simplemente Express, es un framework para Node. js que sirve para ayudarnos a crear aplicaciones web en menos tiempo ya que nos proporciona funcionalidades como el enrutamiento, opciones para gestionar sesiones y cookies, y un largo etc…

Pero ¿qué es un chat online? Es un anglicismo que significa charla. Es uno de los métodos de comunicación digital surgida con las nuevas tecnologías. Consiste en la conversación simultánea entre dos o más personas conectadas a la red. Los mensajes escritos se publican instantáneamente en la pantalla del ordenador. El receptor tiene acceso a ellos sin ningún tipo de retardo y puede contestarlos de igual manera.

Los chats pueden ser públicos o privados. En los chats públicos todos los usuarios conectados a él pueden participar en la conversación. En un chat privado sólo los participantes invitados expresamente pueden hablar.

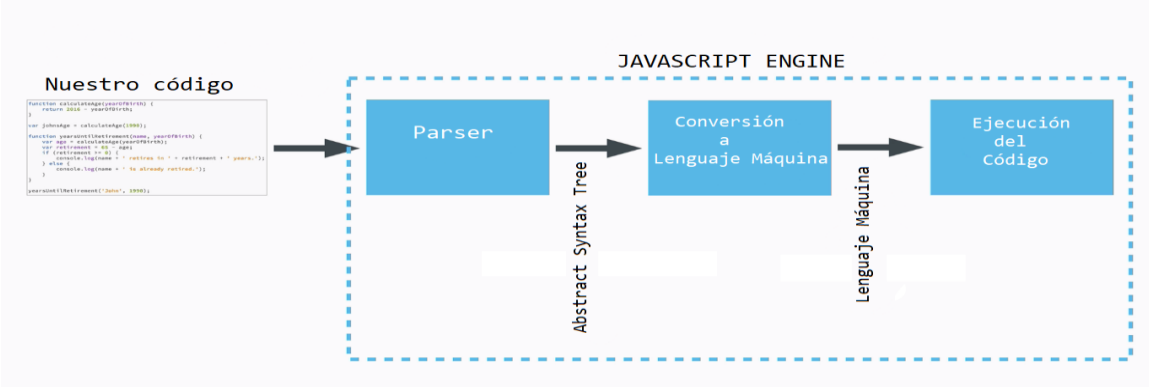
Basándonos en la definición anterior, podemos adelantar el uso de salas de chat, en las cuales solo las personas que se unan a una sala en específico podrán enviar y ver los mensajes de esa sala, con la opción de cambiar de sala y al momento de cambiar ver el historial de esa sala, gracias a la función de “historial” y “armadoHistorial”.

También se buscara la implementación de un ChatBot. Los bot de charla o bot conversacional,  son aplicaciones software que surgen en los años 60, y que simulan mantener una conversación con una persona al proveer respuestas automáticas, las cuales son previamente establecidas por un conjunto de expertos a entradas realizadas por el usuario. Nosotros buscaremos que se haga presente cuando el usuario entre a una sala, o haga un cambio de sala, así como para decirle cuales han sido los mensajes mandado anteriormente.

# Desarrollo

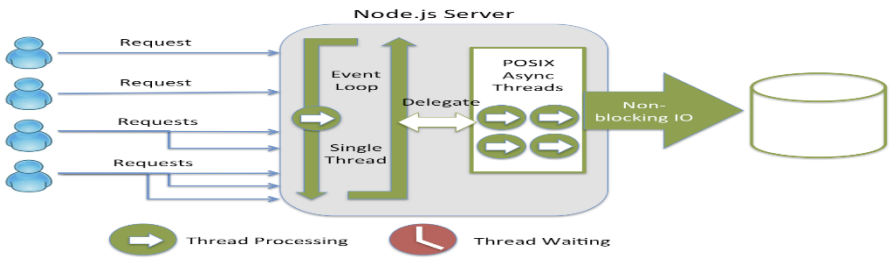
Antes de comenzar, me parece importante el definir ciertos elementos que componen este proyecto. Puesto que algunos son nuevos para nosotros.

Comenzaremos con **JavaScript** el cual es un lenguaje de programación ligero, interpretado, o compilado justo-a-tiempo (just-in-time) con funciones de primera clase. Si bien es más conocido como un lenguaje de scripting (secuencias de comandos) para páginas web, y es usado en muchos entornos fuera del navegador, tal como Node.js, Apache CouchDB y Adobe Acrobat. JavaScript es un lenguaje de programación basada en prototipos, multiparadigma, de un solo hilo, dinámico, con soporte para programación orientada a objetos, imperativa y declarativa (por ejemplo programación funcional).




#1 Funcionamiento de JavaScript

Como vimos en la descripción de JavaScript, es muy usado en diferentes entornos, como lo es **Node.js** que 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.4 Fue creado por Ryan Dahl en 2009 y su evolución está apadrinada por la empresa Joyent, que además tiene contratado a Dahl en plantilla.

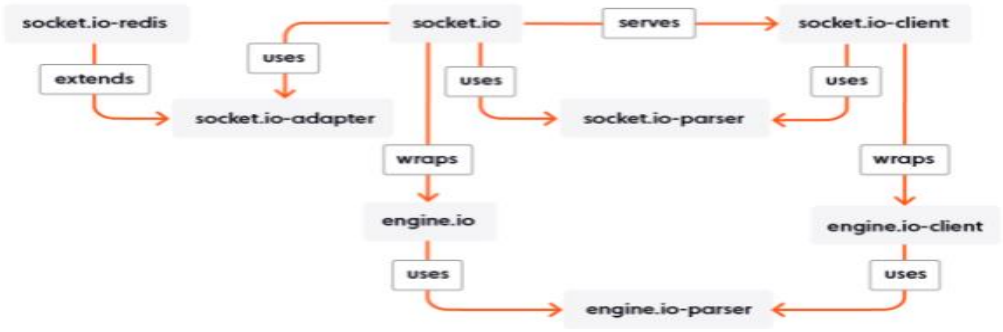



#2 Funcionamiento de Node.js

Por otro lado, me gustaría hablar un poco más de **Socket.io**, ya que es la primera vez que se trabaja con él, así que lo definiremos y hablaremos sobre algunas de las funciones usadas para realizar el código que se verá a continuación.

Socket.io se creó en 2010. Se desarrolló para utilizar conexiones abiertas para facilitar la comunicación en tiempo real, todavía un fenómeno relativamente nuevo en ese momento. Es una biblioteca de JavaScript para aplicaciones web en tiempo real. Permite la comunicación en tiempo real, bidireccional y basada en eventos. Funciona en todas las plataformas, navegadores o dispositivos, centrándose por igual en la fiabilidad y la velocidad. 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.

Para establecer la conexión e intercambiar datos entre el cliente y el servidor, Socket.IO usa Engine.IO . Esta es una implementación de nivel inferior que se usa bajo el capó. Engine.IO se utiliza para la implementación del servidor y Engine.IO-client se utiliza para el cliente.



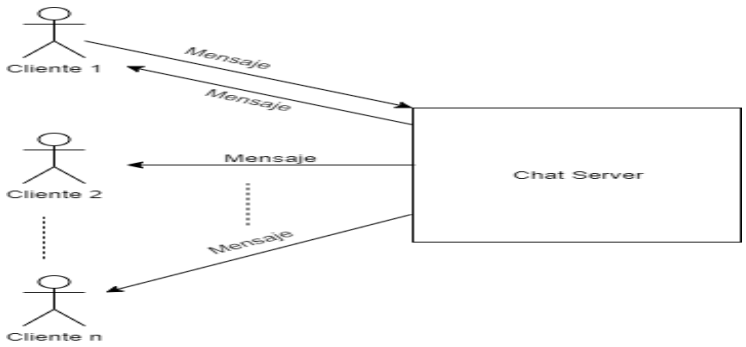

#3 Funcionamiento de Socket.io

Socket.io trae a la mente **WebSockets**. Los WebSockets también son una implementación de navegador que permite la comunicación bidireccional, sin embargo, Socket.IO no lo usa como estándar. Primero, Socket.IO crea una conexión de sondeo largo usando xhr-polling. Luego, una vez establecido, se actualiza al mejor método de conexión disponible. En la mayoría de los casos, esto resultará en una conexión WebSocket.


#4 Imagen representativa

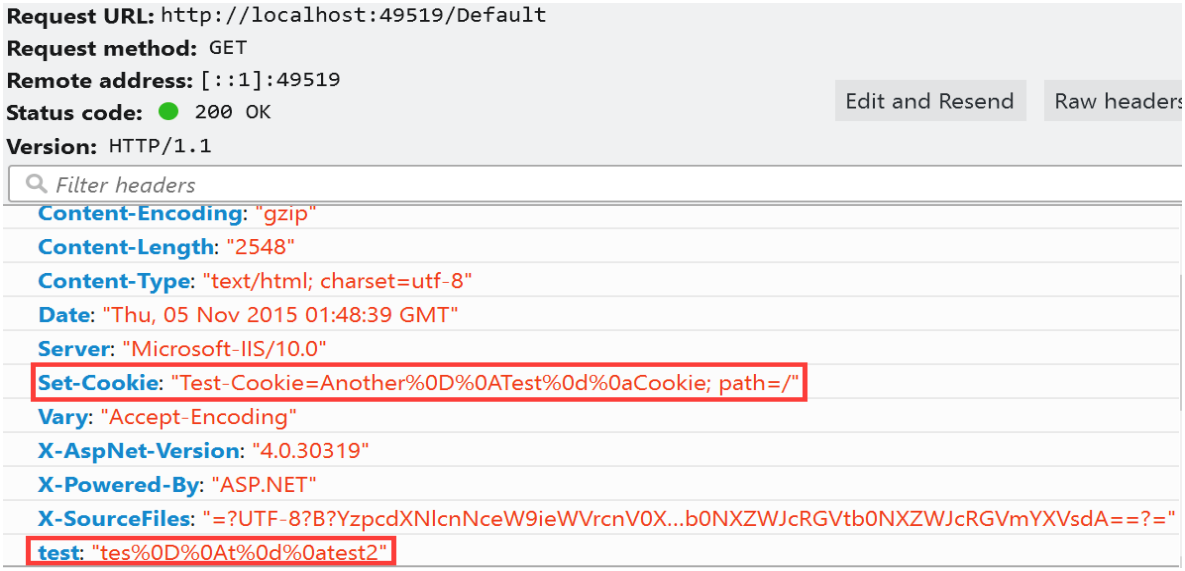
En cierta manera es un buen sustituto de AJAX como tecnología para obtener datos del servidor, ya que no tenemos que pedirlos, el servidor nos los enviará cuando haya nuevos. Uno de los ejemplos más comunes para aprender a utilizar websockets, es desarrollando chat. Que es lo que se verá en unos momentos.

Algunas de las funciones de Socket que utilizamos fueron .on que se usa para agregar eventos a los diferentes elementos del DOM, o el .emit que notificará un mensaje a todos los sockets conectados, y también el .join que pintará los elementos del arreglo separados por un espacio, entre otras funcionalidades.




#5 Funcionamiento de un chat

Otro elemento importante fue **Cookie-Parser** el cuál es un módulo que podemos instalar vía npm y que nos permite configurar cookies dentro de nuestro servidor. Pero ¿Qué es una cookie? Las cookies son pequeñas porciones de datos enviadas por un sitio web y que son almacenadas en el navegador del usuario mientras éste navega por dicho sitio web. Cada vez que el usuario vuelve a ese sitio web, el navegador envía esas porciones de datos a la página web o servidor para conocer la actividad previa del usuario en ese sitio web.



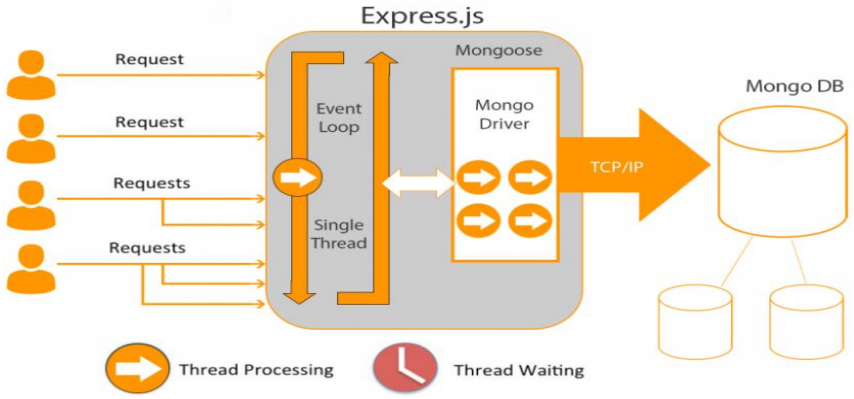

#6 Imagen representativa de una Cookie

Por último **Express.js**, o simplemente Express, es un marco de aplicación web de back-end para Node.js, lanzado como software gratuito y de código abierto bajo la licencia MIT. Está diseñado para crear aplicaciones web y API. Se le ha llamado el marco de servidor estándar de facto para Node.js. Es el framework web más popular de Node, y es la librería subyacente para un gran número de otros frameworks web de Node populares. Proporciona mecanismos para:

- Escritura de manejadores de peticiones con diferentes verbos HTTP en diferentes caminos URL (rutas).
- Integración con motores de renderización de "vistas" para generar respuestas mediante la introducción de datos en plantillas.
- Establecer ajustes de aplicaciones web como qué puerto usar para conectar, y la localización de las plantillas que se utilizan para renderizar la respuesta.
- Añadir procesamiento de peticiones "middleware" adicional en cualquier punto dentro de la tubería de manejo de la petición.

A pesar de que Express es en sí mismo bastante minimalista, los desarrolladores han creado paquetes de middleware compatibles para abordar casi cualquier problema de desarrollo web. Hay librerías para trabajar con cookies, sesiones, inicios de sesión de usuario, parámetros URL, datos POST, cabeceras de seguridad y muchos más. Puedes encontrar una lista de paquetes middleware mantenida por el equipo de Express en Express Middleware (junto con una lista de algunos de los paquetes más populares de terceros).

**Nota:** esta flexibilidad es una espada de doble filo. Hay paquetes de middleware para abordar casi cualquier problema o requerimiento, pero deducir cuáles son los paquetes adecuados a usar algunas veces puede ser un auténtico reto. Tampoco hay una "forma correcta" de estructurar una aplicación, y muchos ejemplos que puedes encontrar en la Internet no son óptimos, o sólo muestran una pequeña parte de lo que necesitas hacer para desarrollar una aplicación web.




#5 Funcionamiento de Express.js

Ahora que hemos comentado sobre los elementos más importantes requeridos para el desarrollo de este proyecto, proseguiremos a mostrar el código utilizado y finalizado con el que se logró nuestro chat online.

## Parte de código de HTML:

index.js | index.html ●

index.html > html > head

```
<!doctype html>
<html lang="es">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="Mark Otto, Jacob Thornton, and Bootstrap contributors">
    <meta name="generator" content="Hugo 0.83.1">
    <title>NodeJs & Socket.IO - Creating a Chat online</title>

    <link rel="canonical" href="https://getbootstrap.com/docs/5.0/examples/sign-in/">




      <!-- Bootstrap core CSS Requerimos a Bootstrap-->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-+0n0xVW2eSR5OomG
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-gtEjrD/SeCtmISkJkNUaaKMoLD0
    <meta name="theme-color" content="#7952b3">

    <!--hoja de estilos para nuestro sitio-->
    <style>
        html,
        body {
        height: 100%;
        }

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

index.js | index.html ×

index.html > html > head

```
        .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; }

            a:hover { text-decoration:underline; }

        #wrapper, #loginform {
            margin:0 auto;
            padding-bottom:25px;
            background:#EBF4FB;
            width:504px;
            border:1px solid #ACD8F0; }
```

index.js index.html

index.html > html > head

```
#loginform { padding-top:18px; }

    #loginform p { margin: 5px; }

#chatbox {
    text-align:left;
    margin:0 auto;
    margin-bottom:25px;
    padding:10px;
    background:#fff;
    height:270px;
    width:430px;
    border:1px solid #ACD8F0;
    overflow:auto; }

#usermsg {
    width:395px;
    border:1px solid #ACD8F0; }

#submit { width: 60px; }

.error { color: #ff0000; }

#menu { padding:12.5px 25px 12.5px 25px; }

.welcome { float:left; }

.logout { float:right; }

.msgln { margin:0 0 2px 0; }
```

index.js index.html

index.html > html > head

```
.form-signin .checkbox {
font-weight: 400;
}

.form-signin .form-floating:focus-within {
z-index: 2;
}

.form-signin input[type="email"] {
margin-bottom: -1px;
border-bottom-right-radius: 0;
border-bottom-left-radius: 0;
}

.form-signin input[type="password"] {
margin-bottom: 10px;
border-top-left-radius: 0;
border-top-right-radius: 0;
}
    .bd-placeholder-img {
        font-size: 1.125rem;
        text-anchor: middle;
        -webkit-user-select: none;
        -moz-user-select: none;
        user-select: none;
    }

    @media (min-width: 768px) {
        .bd-placeholder-img-lg {
        font-size: 3.5rem;
        }
    }
```

index.js index.html

index.html > html > head

```
<!-- Plantilla del Login, obtenida de Bootstrap -->
<link href="signin.css" rel="stylesheet">
</head>
<body class="bg-info 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-success" type="button" id="Login">Entrar</button>
        <button class="w-100 btn btn-lg btn-warning" type="button" id="registrar" data-toggle="modal" data-target="#registro">Registrar</b
        <p class="mt-5 mb-3 text-muted">&copy; Chat Online</p>
    </main>

        <div id="wrapper" style="display: none;">
            <div id="menu">
                <p class="bienvenido"> Bienvenido, <b id="usernameTag"></b>, con correo: <b id="emailUser">,</b>, a la sala: <b id="SalaNo
                <p class="logout "><a id="exit" href="/">Salir del chat</a></p>
            </div>
```

index.html > html > head

```
            <div id="chatbox">
                <!-- Caja del chat que contendrá todos los mensajes. -->
            </div>

                <input name="usermsg" type="text" id="mensaje" size="63"/>
                <input class="btn-success" type="button" name="submitmsg" id="enviarMensaje" value="Enviar Mensaje" />
                <select class="form-group bg-warning"  name="roomsCambio" id="roomsCambio">
                    <option selected>Cambiar de sala</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">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 type="text" class="form-control" id="userNameR" placeholder="name@example.com" name="username" required>
            <label for="floatingInput">Nombre de usuario</label>
            </div>
            <div class="form-floating">
            <input type="password" class="form-control" id="PasswordR" placeholder="Password" name="password" required>
            <label for="floatingPassword">Contraseña</label>
            </div>
```

index.js | index.html

index.html > html > body.bg-info.text-center

```
                </div>
                <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>


        <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/Gp
        <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxh
        <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRR
        <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
        <script src="/socket.io/socket.io.js"></script>
        <!--Iniciamos a usar las funciones del JS-->
        <script>
            $(document).ready(function(){

                var socket = io();  /*declaramos el socket*/
                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,
```

index.js | index.html

index.html > html > body.bg-info.text-center

```
        <script src="/socket.io/socket.io.js"></script>
        <!--Iniciamos a usar las funciones del JS-->
        <script>
            $(document).ready(function(){

                var socket = io();  /*declaramos el socket*/
                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
```

index.js index.html

index.html > html > body.bg-info.text-center > script > ready() callback > click() callback

```
248             socket.emit('cambioSala', {
249                 idSala: roomID,
250                 nombreSala: roomName
251             });
252             socket.emit('historial');
253             console.log('cambio select a ID: ' + roomID + ' con nombre: ' + roomName);
254             });
255
256             /*comienza a funcionar al hacer click en entrar,mandando los datos a su respectiva funcion en forma de arreglo*/
257             $("#Login").click(function(){
258                 socket.emit("login", {
259                     user: $("#userName").val(),
260                     pass: $("#Password").val(),
261                     roomID: $('#rooms').find('option:selected').attr('id'), //Obtenemos el atributo
262                     roomName: $('#rooms').find('option:selected').text()
263                 });
264                 console.log(roomID);
265             });
266
267             $("#sendResgistro").click(function(){
268                     socket.emit("addUser", { /*se mandan los valores del registro*/
269                     user: $("#userNameR").val(),
270                     pass: $("#PasswordR").val(),
271                     email: $("#correo").val(),
272                 });
273             });
274
275             $(".logout").click(function(){ /*salimos*/
276                 socket.emit("salir");
277             });
278
279             $('#registrar').click(function(){ /*se obtienen los valores del registro*/
```

index.js index.html

index.html > html > body.bg-info.text-center > script > ready() callback > socket.on("logged_in") callback

```
278
279             $('#registrar').click(function(){ /*se obtienen los valores del registro*/
280                 $("#userNameR").val("");
281                 $("#PasswordR").val("");
282                 $("#correo").val("");
283             });
284
285             $('#enviarMensaje').click(function(){
286                     if($("#mensaje").val().length <= 0){ /*evalua que se este enviandoa algo*/
287                         alert("Escribe el mensaje para poderlo enviar.");
288                     }else{
289                         var mensaje = $('#mensaje').val()
290                         socket.emit('mjsNuevo', mensaje); // Enviamos el nuevo mensaje a la función de mensaje nuevo.
291
292                     }
293                 });
294
295                 socket.on("logged_in", function(data){
296                     console.log(data);
297                     $(".form-signin").hide(); /*se esconde el formulario de inicio de sesión*/
298                     $("#wrapper").show(); /*se muestra la ventana del chat*/
299                     $('#usernameTag').text(data.user);
300                     $('#emailUser').text(data.email);
301                     $('SalaNombre').text(data.roomName);
302                     socket.emit('historial');
303                 });
304                 /*sirven para verificar campos y mostrar que hace falta*/
305                 socket.on("invalido", function(){
306                     alert("Usuario y/o contraseña incorrectos.");
307                 });
308
309                 socket.on("error", function(){
310                     alert("Error: Intenta de nuevo!");
```

index.js | index.html ●

index.html > html > body.bg-info.text-center > script > ready() callback > socket.on("logged_in") callback

```
309  socket.on("error", function(){
310      alert("Error: Intenta de nuevo!");
311  });
312
313  socket.on("vacio", function(){
314      alert("Error: Llena todos las campos.!");
315  });
316  /*****************************************/
317  socket.on("UsuarioOK", function(){ /*se muestra al aceptarse el registro del usuario*/
318      $('#registro').modal('hide');
319      alert("Dado de alta correctamente.");
320  });
321
322  socket.on('mensaje', function(data){ // Función que tiene de respuesta el nuevo mensaje, concatenamos e insertamos en la caja de chat.
323          if(data.usuario == "BotChat"){
324           var nuevoMensaje = '<small class="bot"><b>' + data.usuario + ' -</b> ' + data.mensaje + '</small>';
325           $('#chatbox').append(nuevoMensaje + '</br>');
326           $('#mensaje').val("");
327          }else{
328           var nuevoMensaje = '<p class="mensajeEnviado"><b>' + data.usuario + ' dice:</b> ' + data.mensaje + '</p>';
329          }
330          $('#chatbox').append(nuevoMensaje + '</br>');
331          $('#mensaje').val("");
332      });
333
334  socket.on('armadoHistorial', function(data){ /*requerimos la funcion armadoHistorial para mostrar el historial de la sala*/
335      var historial = "";
336      $.each(data, function(id, val){
337          historial += '<p class= "mensajeEnviado"><b>' + data[id]['Username'] + 'dijo:</b> ' + data[id]['mensaje'] + '</p>';
338      });
339
340      historial += '<small class="bot"><b>BotChat -</b> Últimos mensajes del historial de la sala</samll>';
```

Go Run Terminal Help ● index.html - TareaT - Visual Studio Code

index.js | index.html ●

index.html > html > body.bg-info.text-center > script

```
324                   var nuevoMensaje = '<small class="bot"><b>' + data.usuario + ' -</b> ' + data.mensaje + '</small>';
325                   $('#chatbox').append(nuevoMensaje + '</br>');
326                   $('#mensaje').val("");
327                  }else{
328                   var nuevoMensaje = '<p class="mensajeEnviado"><b>' + data.usuario + ' dice:</b> ' + data.mensaje + '</p>';
329                  }
330                  $('#chatbox').append(nuevoMensaje + '</br>');
331                  $('#mensaje').val("");
332              });
333
334          socket.on('armadoHistorial', function(data){ /*requerimos la funcion armadoHistorial para mostrar el historial de la sala*/
335              var historial = "";
336              $.each(data, function(id, val){
337                  historial += '<p class= "mensajeEnviado"><b>' + data[id]['Username'] + 'dijo:</b> ' + data[id]['mensaje'] + '</p>';
338              });
339
340              historial += '<small class="bot"><b>BotChat -</b> Últimos mensajes del historial de la sala</samll>';
341
342              $('#chatbox').append(historial + '</br>');
343          });
344
345  });
346
347
348
349
350
351
352  script>
353  >
354
```

Partes del código del JS:

index.js

```
/* Después de descargar nuestros paquetes es necesario requerir los modulos, para ello declaramos las variables,es recomendaod hacerlo con e
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(); /*requerimos las funciones de express*/
var roomName = '';
const nameBot = "BotChat";

/*colgamos nuestro servidor y el puerto en el que trabajará*/
var server = app.listen(3030, function () {
  console.log("Servidor en marcha, port 3030.");
});

/*requerimos las funciones de socket por medio de nuestro servidor*/
var io = socket(server);

/*Configuramos las sesiones*/
var sessionMiddleware = session({              //Definimos middleware para el chat
  secret: "keyUltraSecret",
  resave: true,
  saveUninitialized: true
});
/************************************/

/*Pasamos las sesiones con el socket*/
io.use(function (socket, next) {
  sessionMiddleware(socket.request, socket.request.res, next);
});
/*****************************************************************/
```

index.js

```

 /*Requerimos las funciones de la sesión en app,junto con los sockets ya asignados,así como las funciones de cookieParser*/
 app.use(sessionMiddleware);
 app.use(cookieParser());
 /************************************/

 /*Creamos nuestra conección a nuetsra base de datos*/
 const config = {
   "host": "localhost",
   "user": "root",
   "password": "",
   "base": "chat"
 };

 var db = mysql.createConnection({              //Se configura la conexion a la base de datos de acuerdo a los datos que
                                                // creamos en la base con SQL
   host     : 'localhost',
   user     : 'root',
   password : '',
   database : 'chat'
 });

 db.connect(function (err) {
   if (!!err)
   throw err;

   console.log('MySQL conectado: ' + config.host + ", usuario: " + config.user + ", Base de datos: " + config.base);
 });

 app.use(express.static('./'));

```

index.js index.html

index.js > ...

```
/*si se logra concectar el socket,comienza a realizar todas las funciones/consultas que vienen debajo*/
io.on('connection', function (socket) {
  var req = socket.request;

  console.log(req.session);

    if(req.session.userID != null){
        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);  //Mandam
            socket.emit("logged_in", {user: req.session.Username, email: req.session.correo});  //Mandamos el nombre de usuario, email y cor
        });
    }else{
        console.log('No hay sesión iniciada'); //En el caso contrario, no iniciamos sesion y mandamos mensaje a pantalla
    }

    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;
```

index.js index.html

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

```
                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'); //Mandamos la función de armadoHistorial
                    console.log(req.session);
                    bottxt('entroSala');
                }else{
                    socket.emit("invalido");
                }
            }
        });
    });
    /*************Armamos nuestro historial de cada sala, gracias a esta función y la consulta */
    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
            socket.emit('armadoHistorial', rows);
            console.log(rows);
        });
    });
```

index.js ● index.html ●

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

```
127
128     socket.on('addUser', function(data){
129         const user = data.user,
130         pass = data.pass,
131         email = data.email;
132
133         if(user != "" && pass != "" && email != ""){
134             console.log("Registrando el usuario: " + user);
135             db.query("INSERT INTO users(`Username`, `Password`, `email`) VALUES(?, ?, ?)", [user, pass, email], function(err, result){  //Añ
136               if(!!err)
137               throw err;
138
139               console.log(result);
140
141               console.log('Usuario ' + user + " se dio de alta correctamente!."); //Avisamos al usuario que su registro fue un éxito
142               socket.emit('UsuarioOK');
143             });
144         }else{
145             socket.emit('vacio');
146         }
147     });
148     /*Obtenemos id y nombre dependiendo de la sala a la que cmabiamos*/
149     socket.on('cambioSala', function(data){
150         const idSala = data.idSala,
151         nombreSala = data.nombreSala;
152
153         socket.leave(req.session.roomName);
154
155         req.session.roomID = idSala;
156         req.session.roomName = nombreSala;
157
158         socket.join(req.session.roomName);
159         bottxt('cambioSala');
```

index.js ● index.html ●

index.js > io.on('connection') callback > socket.on('mjsNuevo') callback

```
153         socket.leave(req.session.roomName);
154
155         req.session.roomID = idSala;
156         req.session.roomName = nombreSala;
157
158         socket.join(req.session.roomName);
159         bottxt('cambioSala');
160     });
161
162     socket.on('mjsNuevo', function(data){ // Funcion para el mensaje
163         //La consulta es muy importante, ya que de no estar bien, no guardara mensajes
164             db.query("INSERT INTO mensajes(`mensaje`, `user_id`, `sala_id`, `fecha`) VALUES(?, ?, ?, CURDATE())", [data, req.session.userID,
165               if(!!err)
166               throw err;
167
168               console.log(result);
169
170               console.log('Mensaje dado de alta correctamente!.');
171
172                     socket.broadcast.emit('mensaje', {
173                         usuario: req.session.Username,
174                         mensaje: data
175                     });
176
177                     socket.emit('mensaje', {
178                         usuario: req.session.Username,
179                         mensaje: data
180                     });
181             });
182
183     });
184
```

index.js ● index.html ●

index.js > io.on('connection') callback > socket.on('getSalas') callback

```
184
185     socket.on('getSalas', function (data) { //obtenemos la sala
186         db.query('SELECT id, nombre_sala FROM salas', function(err, result, fields){
187             if(err) throw err;
188             socket.emit('Salas', result);
189         });
190     });
191
192     socket.on('salir', function(request, response){
193         req.session.destroy(); //destruimos la sesión
194     });
195
196     /*Creamos una función de un bot que nos da la bienvenida o nos dice a que sala cambiamos*/
197     function bottxt(data){
198         entroSala = 'Bienvenido a la sala ' + req.session.roomName;
199         cambioSala = 'Cambiaste de sala a ' + req.session.roomName;
200         sefue = 'El usuario ' + req.session.Username + 'ha salido de sala.'
201
202         if(data == "entroSala"){
203             socket.emit('mensaje',{
204                 usuario: nameBot,
205                 mensaje: entroSala
206             });
207         }
208         if(data == "cambioSala"){
209             socket.emit('mensaje',{
210                 usuario: nameBot,
211                 mensaje: cambioSala
212             });
213         }
214         if(data == "salioUsuario"){
215             socket.emit('mensaje',{
216                 usuario: nameBot,
```

index.js ● index.html ●

index.js > io.on('connection') callback > socket.on('getSalas') callback

```
213         }
214         if(data == "salioUsuario"){
215             socket.emit('mensaje',{
216                 usuario: nameBot,
217                 mensaje: sefue
218             });
219         }
220     }
221
222     app.post('/auth', function(request, response) {//Recibimos la autenticacion
223         var username = request.body.username;   //se obtienen datos del usuario
224         var password = request.body.password;
225
226         if(username && password){ //Se revisa si los datos estan vacíos o no
227
228             //Se realiza una consulta para checar usuario y passwd
229             connnection.query('SELECT * FROM users WHERE username = ? AND password = ?' [username, password], function (err, results, fields
230                 if(results.length > 0){
231                     request.session.loggedin = true; //asignamos true al loggedin
232                     request.session.username = username;
233                     response.redirect('/home'); //Redireccionamos a la ruta de home
234                 } else{ //Si el result da 0, el usuario no se encontro y mandamos mensaje a pantalla
235                     response.send('Usuarios y/o contraseña incorrectos');
236                 }
237                 response.end();
238             });
239         } else {
240             response.send('Ingresa usuario y contraseña');
241             response.end(); //Terminamos el proceso
242         }
243     });
244 });
```

# Conclusión

En base a los resultados obtenidos en la realización de este proyecto, podemos concluir que la elaboración de este tipo de retos satisfacen el aprendizaje del alumno, puesto que pone en marcha el ejercitamiento de lo visto en semestres pasados, así como el aprendizaje de nuevas tecnologías, permitiendo en un mismo escenario, resolver problemáticas que ayudan a adquirir una noción más clara del objetivo final: una comunicación al instante, mediante un chat online.

La importancia de saber codificar un chat online se basa en que el comercio electrónico crece diariamente; para que tu negocio capitalice esta oportunidad es muy importante proveer el mejor servicio posible a tus clientes potenciales en el momento en que lo necesitan. Una de las maneras más eficientes es ofrecer atención en línea y personalizada, en tiempo real, a los visitantes de tu sitio web a través de un chat. De esta forma, el chat permite interactuar con nosotros y ponerse en contacto justo cuando les esté surgiendo una duda sobre los servicios que ofreces, los métodos de pago, las características del producto, etcétera.

También en las relaciones humanas toma gran relevancia, ya que nos permite comunicarnos con nuestros seres queridos de manera casi instantánea, haciéndonos sentir más cerca de ellos. En la educación su importancia radica en el intercambio de opiniones de un tema en específico entre el alumno y el profesor, por ejemplo, aclarando dudas.

En cuanto a las tecnologías utilizadas nos quedamos con lo aprendido de las nuevas que utilizamos este semestre, las cuales son Socket.io y Cookie-Parser. Por ejemplo Socket.io es una librería open source con una amplia comunidad que nos ayudará a contruir aplicaciones con conexión persitente entre cliente y servidor. Por lo que contaremos con librerías para cada lado. Y muchos más como Android, iOS, etc. Y ¿Por qué usar Sockets? Los sockets no necesitan que se envíe una petición para poder responder. Ellos permiten un flujo de datos bidireccional por lo tanto solo es necesario escuchar el servidor y éste enviará un mensaje cuando esté disponible. Por su parte Cookie-Parser ayudan a que, por ejemplo Cada vez que visitamos una página, un servidor es el encargado de responder nuestra petición. Cada solicitud que realizamos y la respuesta que recibimos, son totalmente independientes por naturaleza. Sin embargo, el uso de Cookies hace posible la existencia de un estado entre las distintas peticiones que vamos realizando. Es decir, las peticiones HTTP carecen de estado (stateless), pero si se apoyan en el uso de Cookies pueden tener uno (statefull) y con ello modificar lo que los visitantes ven, en función a sus cookies guardadas.

Por último el seguir utilizando Javascript me pareció idóneo, dado que el futuro de este lenguaje no tiene tope, y el uso de Node.js fue gratificante, puesto que seguir aprendiendo acerca de algo que ya te había gustado, siempre viene bien.

# Glosario

- **Anglicismo:** Palabra, expresión o giro procedentes de la lengua inglesa que se usan en otro idioma.
- **Bot:** Es un programa informático que efectúa automáticamente tareas reiterativas mediante Internet a través de una cadena de comandos o funciones autónomas previas para asignar un rol establecido
- **Chat:** Designa una conversación escrita realizada de manera instantánea mediante el uso de un software entre dos o más usuarios conectados a la red, generalmente Internet, ya sea a través de los llamados chats públicos (si cualquier usuario puede entrar) o privados (cuando la entrada está sujeta a autorización).
- **Codificar:** Es la conversión de un algoritmo en programa, utilizando un lenguaje de programación.
- **Función:** Es una sección de un programa que calcula un valor de manera independiente al resto del programa.
- **HTTP:** El Protocolo de transferencia de hipertexto es el protocolo de comunicación que permite las transferencias de información a través de archivos en la World Wide Web.
- **HTML:** Siglas en inglés de HyperText Markup Language, hace referencia al lenguaje de marcado para la elaboración de páginas web.
- **Lenguaje:** Es un lenguaje formal (o artificial, es decir, un lenguaje con reglas gramaticales bien definidas) que le proporciona a una persona, en este caso el programador, la capacidad de escribir (o programar) una serie de instrucciones o secuencias de órdenes en forma de algoritmos con el fin de controlar el comportamiento físico o lógico de un sistema informático,
- **Librería:** Es un conjunto de implementaciones funcionales, codificadas en un lenguaje de programación, que ofrece una interfaz bien definida para la funcionalidad que se invoca.
- **Servidor:** Es un conjunto de computadoras capaz de atender las peticiones de un cliente y devolverle una respuesta en concordancia. Los servidores se pueden ejecutar en cualquier tipo de computadora, incluso en computadoras dedicadas a las cuales se les conoce individualmente como «el servidor».

# Bibliografía


Juarez, P. (12 de Octubre de 2020). *PYM*. Obtenido de https://programacionymas.com/blog/cookies-y-sesiones

NodeJs. (18 de Enero de 2021). *NodeJs*. Obtenido de https://nodejs.org/es/

Pérez, C. C. (24 de Febrero de 2020). *ACADEMIA pagma*. Obtenido de https://www.pragma.com.co/academia/lecciones/implementacion-de-una-aplicacion-de-mensajeria-utilizando-socket

Socket.io. (18 de Marzo de 2020). *Socket.io*. Obtenido de https://socket.io/

Urquiaga, J. C. (2 de Junio de 2016). *DevCode*. Obtenido de https://devcode.la/tutoriales/como-configurar-sesiones-en-expressjs/#:~:text=Cookie%20Parser%20es%20un%20m%C3%B3dulo,Ahora%20tenemos%20que%20instalarlo.&text=Si%20desean%20que%20la%20dependencia,json%20utilicen%20%2D%2Dsave.