- O usuário vai digitar uma informação e quando clicar no botão de enviar essa mensagem vai para o servidor, o vai tratar de distribuir para todos conectados nesse websocket

Esse input para digitar a mensagem não deve estar dentro de um <form>, pois isso tira a opção de enter do teclado.

Colocar os ID’s necessários. Id=”mensagem” | id=”enviar\_mensagem”

View chat.ejs

        <nav class="navbar navbar-default navbar-fixed-bottom" role="navigation">

            <div class="container">

                <div class="panel-body campo-mensagem">

                    <div class="col-md-12">

                        <div class="input-group">

                            <input type="text" class="form-control" placeholder="Mensagem" id="mensagem">

                            <span class="input-group-btn">

                                <button class="btn btn-default" type="button" id="enviar\_mensagem">></button>

                            </span>

                        </div>

                    </div>

                </div>

            </div>

        </nav>

- Emitir a mensagem quando o botão para enviar for clicado

// instancia do obj que se conecta com websocket

            var socket = io('http://localhost:3000');

            // precisamos emitir a para o servidor a mensagem digitada pelo cliente

            $('#enviar\_mensagem').click(function() {

                socket.emit(

                    'msgParaServidor',

                    {apelido: '', mensagem: $('#mensagem').val()}

                );

            });

- No servidor app.js, escutando a mensagem do cliente e emitindo de volta a mensagem para o cliente

/\* criar a conexão por websocket \*/

// funcao ON vai estar escutando req lado do cliente e vice-versa

// estamos escutando eventos de connection, evento padrão do socket.io (propria conexao por parametro [socket])

io.on('connection', function (socket) {

console.log('Usuario conectou');

socket.on('disconnect', function () {

console.log('Usuario desconectou');

});

// escutando a mensagem que veio lá do cliente

socket.on('msgParaServidor', function (data) {

// emitir de volta para o cliente, passando apelido e mensagem

socket.emit('msgParaCliente',

{ apelido: data.apelido, mensagem: data.mensagem }

);

});

});

Fazer os testes e observar que a mensagem só vai para o próprio usuário que digitou.

- Fazendo com que a mensagem seja enviada para todos que estão conectados ao socket do chat (broadcast)

// escutando a mensagem que veio lá do cliente

socket.on('msgParaServidor', function (data) {

// emitir de volta para o cliente, passando apelido e mensagem

socket.emit('msgParaCliente',

{ apelido: data.apelido, mensagem: data.mensagem }

);

// enviar para todos via broadcast

socket.broadcast.emit('msgParaCliente',

{ apelido: data.apelido, mensagem: data.mensagem }

);

});

Fazer o teste das mensagens enviadas para todos

- Recuperando o apelido para ir junto com a mensagem

Pegando apelido lá do app > controller > chat.js e passando para a view chat.ejs. Na view colocaremos esse apelido dentro de um campo oculto para usarmos.

res.render('chat', { dadosForm: dadosForm });

view chat.ejs

<input type="hidden" value="<%= dadosForm.apelido %>" id="apelido">

// precisamos emitir a para o servidor a mensagem digitada pelo cliente

            $('#enviar\_mensagem').click(function() {

                socket.emit(

                    'msgParaServidor',

                    {apelido: $('#apelido').val(), mensagem: $('#mensagem').val()}

                );

            });