***- Fazendo exemplo de juntando canais de comunicação***

Para fazer um teste simulando salas diferentes, ou seja canais de socket diferente, vamos fazer uma adaptação na view e no app.js

View > chat.ejs

        <script>

            // instancia do obj que se conecta com websocket

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

            var conversation\_id = 1001;

            if($('#apelido').val() == 'paulo' || $('#apelido').val() == 'outro'){

                var conversation\_id = 1002;

            }

            socket.emit('subscribe', conversation\_id);

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

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

                socket.emit(

                    'msgParaServidor',

                    {room: conversation\_id, apelido: $('#apelido').val(), mensagem: $('#mensagem').val()}

                );

                $('#mensagem').val("");

            });

            // escutar msgParaCliente

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

                var html = '';

                html += '<div class="dialogo">';

                    html += '<h4>' + data.apelido + '</h4>';

                    html += '<p>' + data.mensagem + '</p>';

                html += '</div>';

                $('#dialogos').append(html);

                // deixar a tela para mostrar a mensagem mais atualizada

                window.scrollTo(0, document.body.scrollHeight);

            });

        </script>

Na raiz e arquivo app.js

/\* 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');

});

// vou fazer a junção no canal

socket.on('subscribe', function (room) {

console.log('joining room', room);

socket.join(room);

});

// escutando a mensagem que veio lá do cliente

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

console.log("conversa na sala: " + data.room);

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

// envia para todos da sala, incluindo o remetente tbm

io.sockets.in(data.room).emit('msgParaCliente',

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

);

// enviar para todos da sala via broadcast, deixa de fora o remetente

/\*socket.broadcast.to(data.room).emit('msgParaCliente',

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

);\*/

});

});

/\* fazer o teste de comunicação entre usuário e outro privado, se estiverem no mesmo canal vai haver a conversa \*/

Ao invés de autenticar e ir direto para o chat, vamos mudar isso e colocar para ir à tela dos usuários fazendo a listagem. Como o chat não é mais publico para todos não faz sentido ir direto para a view do chat

Então ao autenticar vamos fazer o redirect para usuários

Controllers > index.js

usuarioDAO.autenticar(dadosForm, function (result) {

// console.log(result);

if (result[0] != undefined) {

// criando minha variavel de sessão, se encontrou o usuario e senha

req.session.autorizado = true;

req.session.usuario = result[0];

}

if (req.session.autorizado === true) {

res.redirect('/usuarios');

} else {

/\* simulando estrutura gerada pelo req.validationErrors(),

\*\* para exibir mensagem de que e-mail não foi encontrado no banco de dados \*/

erros = [{ msg: 'E-mail e/ou senha incorreto(s)' }];

res.render('index', { validacao: erros });

}

});

Pela rota, routes > usuarios.js

module.exports = function (application) {

application.get('/usuarios', function (req, res) {

application.app.controllers.usuarios.usuarios(application, req, res);

});

}

Controller > usuário.js

module.exports.usuarios = function (application, req, res) {

if (!req.session.autorizado) {

res.redirect('/');

return;

}

var connection = application.config.dbConnection();

var usuarioDAO = new application.app.models.UsuarioDAO(connection);

usuarioDAO.getUsuarios(function (err, result) {

res.render('usuarios', { usuarios: result })

})

}

Adicionar novo método no model para buscar todos os usuários do banco

UsuarioDAO.prototype.getUsuarios = function (callback) {

this.\_connection.open(function (err, mongoClient) {

mongoClient.collection("usuarios", function (err, collection) {

collection.find().toArray(function (err, result) {

callback(err, result)

})

mongoClient.close()

})

})

}

/\* Na view > chat.ejs, retirar todo o html e js referente aos participantes \*/

/\* Ajustar menu de navegação \*/

Pegar a view > usuários.ejs

Logo depois fazer a listagem dinâmica dos usuário com <% %>

<!DOCTYPE HTML>

<html lang="pt-br">

    <head>

        <meta charset="UTF-8">

        <title>Multiroom Chat</title>

        <!-- JQUERY -->

        <script src="js/jquery-2.2.1.js"></script>

        <!-- bootstrap - link cdn -->

        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css" integrity="sha384-1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjWPGmkzs7" crossorigin="anonymous">

        <!-- estilo -->

        <link href="css/style.css" rel="stylesheet">

    </head>

    <body>

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

            <div class="container">

                <div class="navbar-header">

                    <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#navbar-collapse-1" id="btn\_navbar\_toggle">

                        <span class="sr-only">Toggle navigation</span>

                        <span class="icon-bar"></span>

                        <span class="icon-bar"></span>

                        <span class="icon-bar"></span>

                    </button>

                    <a class="navbar-brand" href="#">Multiroom Chat</a>

                </div>

                <div class="collapse navbar-collapse" id="navbar-collapse-1">

                    <ul class="nav navbar-nav">

                        <li>

                            <a href="/chat" id="exibe\_chat">Chat</a>

                        </li>

                        <li>

                            <a href="/usuarios" id="exibe\_participantes">Participantes</a>

                        </li>

                    </ul>

                    <a href="/sair" class="btn btn-default navbar-btn navbar-right">Sair</a>

                </div>

            </div>

        </nav>

        <div class="container">

            <div class="row participantes" id="participantes">

                <div class="col-md-1"></div>

                <div class="col-md-10">

                    <% if(usuarios.length > 0) { %>

                        <% for(var i = 0; i < usuarios.length; i++) { %>

                            <span class="participante">

                                <img src="images/ico\_usuario.png" />

                                <%= usuarios[i].email %> | <%= usuarios[i].apelido %>

                            </span>

                        <% } %>

                    <% } %>

                </div>

                <div class="col-md-1"></div>

</div>

</div>

        <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js" integrity="sha384-0mSbJDEHialfmuBBQP6A4Qrprq5OVfW37PRR3j5ELqxss1yVqOtnepnHVP9aJ7xS" crossorigin="anonymous"></script>

        <!-- rota importada automaticamente, tem tudo que precisamos do socket do lado do cliente -->

        <script src="/socket.io/socket.io.js"></script>

        <script>

        </script>

    </body>

</html>

Voltando no chat.ejs

Mudando o emit da inscrição na sala

            var conversation\_id = 1001;

            if($('#apelido').val() == 'paulo' || $('#apelido').val() == 'outro'){

                var conversation\_id = 1002;

            }

            socket.emit('join', {

                room\_name: conversation\_id,

                apelido: $('#apelido').val()

            });

Mudando tbm a parte de enviar\_mensagem

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

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

                socket.emit('msgParaServidor', {

                    room\_name: room\_name,

                    apelido: $('#apelido').val(),

                    mensagem: $('#mensagem').val()

                });

                $('#mensagem').val("");

            });

No app.js, tbm ajustes na inscrição e na escuta -> emit para o cliente

// vou fazer a junção no canal

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

console.log('joining room', data.room\_name + ' apelido: ' + data.apelido);

socket.join(data.room\_name);

});

// escutando a mensagem que veio lá do cliente

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

console.log("conversa na sala: " + data.room\_name);

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

// envia para todos da sala, incluindo o remetente tbm

io.sockets.in(data.room\_name).emit('msgParaCliente', {

apelido: data.apelido,

mensagem: data.mensagem

});

// enviar para todos da sala via broadcast, deixa de fora o remetente

/\*socket.broadcast.to(data.room).emit('msgParaCliente',

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

);\*/

});

Agora é fazer uma implementação em que o usuário logado possa criar uma sala nova.

Começando pela view > usuários.ejs

            <div class="row">

                <div class="col-md-1"></div>

                <div class="col-md-10">

                    <h2>Salas</h2>

                    <form action="/createRoom" method="post">

                        Nome da sala: <input type="text" name="room">

                        <button type="submit">Criar Sala</button>

                    </form>

                    <ul id="list-rooms">

                    </ul>

                </div>

                <div class="col-md-1"></div>

            </div>

No arquivo de routes > usuários.js

application.post('/createRoom', function (req, res) {

application.app.controllers.usuarios.createRoom(application, req, res);

});

Controller > usuários.js

module.exports.createRoom = function (application, req, res) {

var nameRoom = req.body.room;

res.redirect('/chat?idRoom=' + nameRoom);

}

View > chat.ejs

        <script>

            // instancia do obj que se conecta com websocket

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

            var room\_name = window.location.href.split('?')[1];

            socket.emit('join', {

                room\_name: room\_name,

                apelido: $('#apelido').val()

            });

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

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

                socket.emit('msgParaServidor', {

                    room\_name: room\_name,

                    apelido: $('#apelido').val(),

                    mensagem: $('#mensagem').val()

                });

                $('#mensagem').val("");

            });

            // escutar msgParaCliente

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

                var html = '';

                html += '<div class="dialogo">';

                    html += '<h4>' + data.apelido + '</h4>';

                    html += '<p>' + data.mensagem + '</p>';

                html += '</div>';

                $('#dialogos').append(html);

                // deixar a tela para mostrar a mensagem mais atualizada

                window.scrollTo(0, document.body.scrollHeight);

            });

        </script>

App.js

// vou fazer a junção no canal

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

console.log('joining room', data.room\_name + ' apelido: ' + data.apelido);

socket.join(data.room\_name);

socket.broadcast.to(data.room\_name).emit('msgParaCliente', {

apelido: data.apelido,

mensagem: ' acabou de entrar no chat'

});

socket.broadcast.emit('new room', data.room\_name);

});

// escutando a mensagem que veio lá do cliente

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

console.log("conversa na sala: " + data.room\_name);

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

// envia para todos da sala, incluindo o remetente tbm

io.sockets.in(data.room\_name).emit('msgParaCliente', {

apelido: data.apelido,

mensagem: data.mensagem

});

// enviar para todos da sala via broadcast, deixa de fora o remetente

/\*socket.broadcast.to(data.room).emit('msgParaCliente',

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

);\*/

});

});

View > usuário.ejs

        <script>

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

            socket.on('new room', function(room) {

                $('#list-rooms').append(

                    '<li><a href="/chat?'+room+'">'+room.substring(7)+'</a></li>');

            });

        </script>

Fim

Emitindo quais salas já tem no chat atualmente, para um usuário que acabou de entrar na listagem de usuários

App.js

socket.emit('rooms', io.sockets.adapter.rooms);

views > usuarios.ejs

        <script>

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

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

                var ks = Object.keys(data);

                console.log(data);

                // Limpa a lista

                $('#list-rooms').html('');

                for (var i = 1; i < ks.length ; i++) {

                    var roomLink = ks[i];

                    if(roomLink.includes("idRoom")) {

                        $('#list-rooms').append('<li><a href="/chat?'+roomLink+'">'+roomLink.substring(7)+'</a></li>');

                    }

                }

            });

            socket.on('new room', function(room) {

                $('#list-rooms').append(

                    '<li><a href="/chat?'+room+'">'+room.substring(7)+'</a></li>');

            });

        </script>