***- Fazendo a rota autenticar, criando a sessão do usuário através do express-session***

Instalando o express-session = npm install express-session - -save

Abrindo o arquivo server.js e adicionando o mesmo como middleware

Config > server.js

/\* importar modulo do express-validator \*/

var expressSession = require('express-session');

/\* configurar o middleware express-session \*/

app.use(expressSession({

secret: 'cmievoeveiokfvoisjuio', // chave usado para assinar o cookie de sessão

resave: false, // true: sessão é regravada no servidor, mesmo não havendo modificação durante o request

saveUninitialized: false // true: criar um sessão nova sempre que a mesma for modificada

}));

- Configurar um nova rota lá no index.js

Routes > index.js

module.exports = function (application) {

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

application.app.controllers.index.home(application, req, res);

});

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

application.app.controllers.index.autenticar(application, req, res);

});

}

- Implementando o controller > index.js

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

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

}

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

var dadosForm = req.body;

req.assert('email', 'Campo e-mail não deve ser vazio').notEmpty();

req.assert('senha', 'Campo senha não deve ser vazio').notEmpty();

var erros = req.validationErrors();

if (erros) {

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

return;

}

}

/\* Testar a validação se esta tudo certo \*/

Obs: esta tendo uma validação no controller > chat.js. Retirar essa validação de lá.

Alterando a view > index.ejs

A action agora vai para o “/autenticar” e method=”POST”

<div class="panel-body">

<form accept-charset="UTF-8" role="form" class="form-signin" action="/autenticar" method="POST">

<fieldset>

<label class="panel-login">

<div class="login\_result"></div>

</label>

<input class="form-control" placeholder="exemplo@dominio.com.br" id="email" name="email" type="text">

<input class="form-control" placeholder="\*\*\*\*\*\*" id="senha" name="senha" type="text">

<br></br>

<input class="btn btn-lg btn-success btn-block" type="submit" id="entrar" value="Entrar »">

</fieldset>

</form>

</div>

- Criando uma nova função para autenticar no UsuarioDAO.js

function UsuarioDAO(connection) {

this.\_connection = connection;

}

UsuarioDAO.prototype.inserirUsuario = function (usuario) {

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

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

collection.insert(usuario);

mongoClient.close();

})

})

}

UsuarioDAO.prototype.autenticar = function (usuario, callback) {

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

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

// collection.find({ email: { $eq: usuario.email }, senha: { $eq: usuario.senha } })

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

// passando o result para o meu callback, que sera tratado no controller

callback(result);

})

mongoClient.close();

})

})

}

module.exports = function () {

return UsuarioDAO;

}

- Voltando para o controller > index.js

Depois de tudo validado devemos verificar se ouve retorno de algum usuário no banco. Se sim, vamos criar algumas variáveis de sessão.

if (erros) {

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

return;

}

var connection = new application.config.dbConnection();

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

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('chat');

} 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 });

}

});

}

/\* Verificar se esta acessando e validando ao logar. Lembrando que nada impede de eu ir para a rota direto... Apesar da sessão criada eu ainda não estou barrando usuários que não estão logados \*/

**- Autenticando de fato o usuário na sala do chat ”/chat”**

Verificar se o usuario esta autenticado para dar permissão no chat, ou mandar de volta para a raiz (“/”) que é o form de login

Controller > chat.js

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

/\* vai ter um json com os campos e valores enviados do formulario atraves do 'name' \*/

var dadosForm = req.body;

/\* funcao assert vem do expressValidator (esta no app -> application) \*/

/\*req.assert('apelido', 'Nome ou apelido é obrigatório').notEmpty();

req.assert('apelido', 'Nome ou apelido deve ter entre 3 até 15 caracteres').len(3, 15);

var erros = req.validationErrors();

if (erros) {

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

return;

}\*/

/\* verifica se o usuario já esta autenticado \*/

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

/\* emit msgParaCliente (websocket)

utilizando obj do socket.io dentro do controller \*/

application.get('io').emit(

'msgParaCliente',

{ apelido: dadosForm.apelido, mensagem: ' acabou de entrar no chat' }

);

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

return;

}

res.redirect('/');

}

/\* Realizar os testes se esta logando ou não\*/

**- Colocando no chat um botão de sair (“logout”)**

Views > chat.ejs colocar um botão que dará a opção de sair, efetuar logout na aplicação

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

    <ul class="nav navbar-nav">

        <li>

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

        </li>

        <li>

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

        </li>

    </ul>

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

No router > chat.js vamos criar a rota para “/sair”

module.exports = function (application) {

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

application.app.controllers.chat.iniciaChat(application, req, res);

});

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

application.app.controllers.chat.sair(application, req, res);

});

}

No controller > chat.js

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

req.session.destroy(function (err) {

res.redirect("/");

});

}

- No controller > chat.js

Mudar a passagem do parâmetro do atributo

O valor da chat apelido agora vem da sessão

O parâmetro de dadosForm também muda para o que vem da sessão

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

/\* emit msgParaCliente (websocket)

utilizando obj do socket.io dentro do controller \*/

application.get('io').emit(

'msgParaCliente',

{ apelido: req.session.usuario.apelido, mensagem: ' acabou de entrar no chat' }

);

res.render('chat', { usuario: req.session.usuario });

return;

}

E na view > chat.ejs

Vamos mudar o input hidden, para pegar o apelido novamente de quem enviou a mensagem

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

Um detalhe de front-end final é deixar a tela sempre na parte de baixo da conversa, para mostrar o conteúdo mais atualizado possível

            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);

            });