***- Vamos criar estrutura de cadastro e login para um usuário que deseja usar o chat***

- Tela para realização do view > cadastro.ejs

<div class="container">

    <div class="row">

        <h2>Cadastro de novo usuário</h2>

    </div>

    <form class="form-cadastro" action="/cadastrar" method="POST">

        <div class="row">

            <div class="form-group">

                <label for="apelido" class="col-sm-2 col-form-label">Apelido</label>

                <input type="text" class="form-control" id="apelido" name="apelido" placeholder="Apelido para ser exibido no chat">

            </div>

            <div class="form-group">

                <label for="email" class="col-sm-2 col-form-label">E-mail</label>

                <input type="text class="form-control" id="email" name="email" placeholder="E-mail válido para acessar o chat">

            </div>

            <div class="form-group">

                <label for="senha" class="col-sm-2 col-form-label">Senha</label>

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

            </div>

        </div>

        <div class="form-group row">

            <button type="submit" class="btn btn-primary">Confirmar cadastro</button>

            <a href="/" class="btn btn-default">Voltar</a>

        </div>

    </form>

</div>

- Definindo as rotas para o routes > cadastro e cadastrar

module.exports = function (application) {

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

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

});

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

application.app.controllers.cadastro.cadastrar(application, req, res);

});

}

- Criando o arquivo cadastro no controller > cadastro.js

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

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

}

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

var dadosForm = req.body;

req.assert('apelido', 'Apelido não pode ser vazio').notEmpty();

req.assert('email', 'Este e-mail não é válido').isEmail();

req.assert('senha', 'A senha deve conter no minimo 6 caracteres').len(6, 100);

var erros = req.validationErrors();

if (erros) {

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

return;

}

res.redirect('/');

}

Recuperando erros na view > cadastro.ejs

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

                <div class="alert alert-danger">

                    <strong>Atenção</strong>

                    <ul>

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

                            <li><%= validacao[i].msg %></li>

                        <% } %>

                    </ul>

                </div>

            <% } %>

- Ensinando truque para manter os valores nos campos mesmo depois da validação

Controller > cadastro.js

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

if (erros) {

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

return;

}

Na view > cadastro.ejs recuperar esses valores através do “value”

<div class="form-group">

    <label for="apelido" class="col-sm-2 col-form-label">Apelido</label>

    <input type="text" value="<%= dadosForm.apelido %>" class="form-control" id="apelido" name="apelido" placeholder="Apelido para ser exibido no chat">

</div>

<div class="form-group">

    <label for="email" class="col-sm-2 col-form-label">E-mail</label>

    <input type="text" value="<%= dadosForm.email %>" class="form-control" id="email" name="email" placeholder="E-mail válido para acessar o chat">

</div>

<div class="form-group">

    <label for="senha" class="col-sm-2 col-form-label">Senha</label>

    <input type="password" value="<%= dadosForm.senha %>" class="form-control" id="senha" name="senha" placeholder="\*\*\*\*\*\*\*\*\*\*">

</div>

- Fazendo conexão com o mongodb

*Instalando mongo-db = npm install*

Vou criar um novo arquivo “dbConnection.js” na pasta config do projeto.

- config > dbConnection.js

var mongo = require('mongodb');

var connMongoDB = function () {

// console.log("entrou na função de conexão");

var db = new mongo.Db(

'chatrealtime', // bando de dados

new mongo.Server('localhost', 27017, {}), //obj de conexao com mongoDB

{} // config. adicionais

);

return db;

}

module.exports = function () {

return connMongoDB;

}

- No config > server.js lembrar de colocar esse js no autoload

/\* efetua o autoload das rotas, models e controllers para o obj app \*/

consign()

.include('app/routes')

.then('config/dbConnection.js')

.then('app/models')

.then('app/controllers')

.into(app);

Na pasta Model vou criar um js UsuarioDAO.js, seguindo o padrão DAO e utilizando o prototype do javascript para porder instanciar através do “new” esse js, simulando um classe.

- model > 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();

})

})

}

module.exports = function () {

return UsuarioDAO;

}

- Voltando no controller > cadastro.js para inserir o usuário utilizando a função do UsuarioDAO

Após a verificação do erros

if (erros) {

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

return;

}

var connection = application.config.dbConnection();

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

usuarioDAO.inserirUsuario(dadosForm);

res.redirect('/');