01

Autenticação e login

Transcrição

[00:00] Na aula anterior fizemos aquela lógica de confirmação tripla do cadastro de usuário para garantir que o nosso usuário realmente estivesse presente, e que batesse com o token que foi enviado para o email do cliente. Agora temos que permitir que o usuário faça login e autenticar de um modo relativamente básico os dados dele.

[00:24] Para começar vamos fazer uma tela de login, eu vou duplicar aqui o formulário de novo usuário para termos um formulário de login, vamos renomear aqui para formularioDeLogin, o que vamos ter nesse formulário de login? Para começar o formulário não precisa ser um formulário de usuário, ele pode ser um formulário dinâmico porque só vamos usar dois campos do nosso modelo, que são o email e a senha.

[00:55] Aqui o título pode ser login, do mesmo jeito que aqui em baixo, tela de login. Já a rota, vamos precisar fazer uma rota de login, então vou criar aqui um método chamado fazLogin, e o que são os campos que precisamos? Vamos precisar aqui de um email e uma senha, eles já estão aqui, eu só vou adicionar uma coisa aqui que é um pseudo atributo chamado autocomplete que vai impedir que o browser complete o seu campo automaticamente com o que ele tem armazenado.

[01:29] Eu passo aqui false, eu vou fazer a mesma coisa no campo da senha e alterar aqui para o botão escrever entrar, já temos nosso formulário, o próximo passo é criar as rotas e os métodos no controller. Vamos criar as rotas antes, eu vou criar aqui uma rota get, que é /usuario/painel, para quando nosso usuário fizer o login ele é redirecionado para o painel, vai para controllers.UsuarioController.painel.

[02:10] Aqui não precisa de mais nada, eu vou copiar isso aqui porque vamos usar mais uma vez, para o login vamos ter duas rotas, a rota do formulário de login e a rota que realmente faz o login o get e o post, então GET /login e um POST /login. E aqui vamos criar as duas rotas de acordo com o que quisermos.

[02:45] O formulário de login podemos chamar de formulário do login, o post eu vou copiar aqui tudo, vai ser o fazLogin que já especificamos ali no formulário. Já temos as nossas três rotas, então agora falta criar os métodos no controller, eu vou descer aqui e eu vou criar primeiro o formulário de login, public Result formularioDeLogin que vai retornar um ok, com o nosso formulário, formularioDeLogin.render e ele vai receber um formulário dinâmico.

[03:34] Então podemos simplesmente formularios.form, isso gera um formulário dinâmico fica em uma lógica que parece ok, está fácil de entender, agora o que precisamos? Precisamos do método que faz o login, então public Result fazLogin, esse método vai cuidar da nossa lógica de login, return, por enquanto eu não vou fazer nada nele. E o nosso painel, só que o nosso painel é uma página autenticada, então já vamos utilizar o básico do sistema de autenticação do play.

[04:12] Vamos lá, public Result painel que vai por enquanto retornar um ok, com uma frase indicando que esse é o painel do usuário, e como que fazemos essa autenticação básica? É só usar uma anotação a anotação @Autenticated, já vem embutida no play e só precisa fazer isso para que ela funcione. Mas quando você fizer login, você não quer uma tela vazia, então vamos redirecionar direto para o painel?

[04:50] Então redirect para o painel do usuário, é routes.UsuarioController.painel, isso deve bastar, vamos ver se ele compila tudo bonitinho, parece que compilou, então já podemos testar o sistema de login, vamos entrar no nosso formulário e enviar alguma coisa, não precisa nem ser algum dado válido porque ainda não estamos validando isso, então eu vou escrever aqui marco@caelum.com.br e colocar uma senha qualquer.

[05:25] Olha só, caímos no painel e já tomamos um 401 não autorizado, porque já fizemos a autenticação, agora precisamos fazer a nossa lógica de login, então vamos garantir que os dados do formulário estão corretos. Primeiro temos que pegar o email e a senha do nosso usuário, então vamos pegar o formulário e dele a senha, então vamos pegar o formulário, formularios.form.bindFromRequest, tira os argumentos, beleza já temos o formulário.

[06:03] A partir dele podemos pegar a senha, então formulario.get e pegamos o email primeiro, guardamos em uma string chamada e email e agora pegamos a senha, formulario.get senha, só que eu não quero guardar em memória a senha do usuário não encriptada, então eu já vou jogar aqui ela no crypt.sha1 e já guardamos ela em uma variável já autenticada senha.

[06:38] Agora precisamos perguntar para o nosso banco de dados se existe um usuário que tenha esse email e essa senha encriptada, então vamos usar o usuarioDAO, usuarioDAO.comEmailESenha, e passamos o email e a senha do usuário e se ele retornar um objeto, sabemos que o usuário existe. “Optional possivelUsuario =” usuário com email e senha.

[07:14] E se o usuário existir vamos fazer o login dele, se o usuário não existir, vamos redirecionar para o login, então já vou alterar aqui para o formulário de login, if (possivelUsuario .isPresent()), e sabemos que o usuário existe mas ainda não sabemos se ele está verificado, se ele já fez a confirmação por email, então temos que confirmar isso, então vamos lá, pegamos possivelUsuario.get guardamos isso em uma variável e perguntamos if(usuario.isVerificado()), e podemos fazer o login.

[07:56] E como que fazemos o login? Do mesmo modo que o flash, temos a sessão, e podemos inserir dados nessa sessão, então vamos inserir dados na seção, do mesmo modo que no flash passamos uma chave e um valor, no caso a chave aqui vai chamar AUTH e o valor vai ser o nome ou email do usuário, eu prefiro email, usuario.getEmail.

[08:21] Só isso já consegue fazer o login do usuário, com só isso já conseguimos colocar o usuário na sessão, mas ainda não garantimos que esse usuário existe, deixa eu extrair isso para uma constante porque isso vai ser usado lá fora também, eu vou colocar ela como publica exatamente porque ela vai ser usada no sistema de autenticação efetivamente.

[08:50] Mas eu coloquei o usuário na sessão e eu estou redirecionando para o formulário de login aqui? Não faz sentido, vamos mandar uma mensagem de sucesso para o usuário e redirecionar ele para o painel, então vamos lá, flash, vamos colocar aqui uma mensagem de sucesso dizendo que o login foi efetuado com sucesso.

[09:15] E redirecionamos ele para o painel de usuário, routes.UsuarioController.painel, agora a lógica de inserir o usuário na sessão está ok, porém caso o usuário não esteja verificado ainda, temos que mandar uma mensagem de erro para ele, ou um aviso, que seja. Então vamos adicionar um warning, indicando que o usuário não foi confirmado ainda, “Usuário ainda não confirmado, confira seu email”.

[10:00] E caso o usuário nem exista, quer dizer que ele errou o login e a senha dele, então mostramos uma mensagem para ele indicando que as credencias estão erradas, então flash(“danger”, “Credencias inválidas!”. Caso o usuário ainda não tenha saído do confirmado ele vai voltar para tela de login, caso ele tenha errado o usuário ou senha, ele também vai voltar para tela de login, cada um com uma mensagem diferente.

[10:38] Caso ele acerte tudo, o usuário vai cair na sessão e ser redirecionado para o painel, falta agora criar o método do DAO que confere se existe um usuário com email e senha, então vamos criar ele. Ele retorna com um usuário opcional, então vamos vir aqui, procuramos nos usuarios.where.eq, o email tem que ser igual ao argumento email e também .eq a senha tem que ser igual ao argumento senha, lembrando que ela já vem encriptada.

[11:13] E vamos procurar um único objeto com esses parâmetros, guardamos isso em uma única variável e retornamos com um opcional para o nosso controller, então optional.ofNullable(usuário), aqui o DAO está ok, então já temos o

nosso usuário na sessão, já garantimos se ele existe ou não, redirecionar para os lugares corretos e agora precisamos garantir que o usuário seja autenticado.

[11:47] Se eu vier aqui e fazer login ainda não vai acontecer nada, vamos voltar para tela de login e tentar entrar de novo, eu vou tentar entrar com a minha senha correta. Eu ainda estou tomando um unauthorized, por quê? Porque o sistema de autenticação do play ainda não sabe que fomos autenticados, então vamos fazer isso, para fazer uma autenticação customizada, vamos vir aqui e passar uma classe customizada, UsuarioAutenticado.class.

[12:18] Então eu vou criar essa classe "Ctrl + N", eu vou criar no pacote autenticação, autenticadores e vai chamar UsuarioAutenticado, ela vai precisar estender uma classe do play que chama Authenticator do play.mvc.Security, e porque isso? Porque ele tem uma lógica interna através do método getUsername, que se você retorna alguma coisa que não seja nula aqui, ele identifica que existe uma autenticação, se você retorna nulo ele identifica que você não está autenticado.

[12:58] Vou alterar essa variável aqui para chamar context porque é um pouco mais agradável. E como acessamos nosso usuário da sessão a partir daqui? A sessão está dentro do nosso contexto, então context.session e pegamos aqui, get, a nossa chave de autenticação, poderíamos escrever AUTH aqui, mas foi por isso que eu criei uma constante, UsuarioController.AUTH, pegamos aqui se existir algum valor aqui dentro conseguimos fazer a autenticação e ele vai reconhecer isso.

[13:35] Então é isso, agora precisamos ir no controller e importar a classe UsuarioAutenticado e agora conseguimos fazer a nossa autenticação, vamos ver como que fica? Eu vou faço aqui o meu login, vou digitar o meu email correto e a minha senha e agora conseguimos ver o nosso painel de usuário.

[13:58] Então aprendemos por cima a utilizar o sistema de autenticação embutido do play para fazer login e conseguimos acessar uma rota que necessita de um usuário autenticado. No próximo vídeo vamos permitir que o usuário se deslogue e vamos tratar o caso que ele não tem acesso para redirecionar ele para algum lugar e mandar uma mensagem de erro.