02

Controlando os usuários

Transcrição

[00:00] Já temos a tela onde o usuário logado consegue visualizar os seus pedidos, só que ele ainda não consegue visualizar por status. Tem outra coisa que é importante: estamos chamando essa tela de "/home".

[00:15] Na verdade, a "/home" vai ser uma tela em que o usuário não vai precisar estar logado para ele visualizar e não vai ter esse "Meus Pedidos", vai ser uma tela diferente. E vão aparecer os pedidos que já foram entregues, os principais pedidos. Podemos pensar em uma consulta ao banco de dados mais elaborada, mas acho que vai a cargo de você mesmo decidir como você quer construir. Mas isso nós vamos fazer depois.

[00:44] O importante agora é aqui no próprio "HomeController.java", nós entendermos que essas buscas por `findAllByUsuario` não são legais.

[01:00] Vamos buscar aqui por status os pedidos, vamos pegar todos os pedidos que já foram entregues por status e mostrar algumas informações para os usuários entrarem e já verem alguma coisa - verem quais os pedidos estão finalizados e quando foram finalizados. É interessante eles saberem que o site está funcionando, que é interessante.

[01:24] Tem que ter uma página pública para o usuário e tem que ter a página também privada, a página do próprio usuário, para ele buscar os seus próprios pedidos.

[01:35] Eu vou criar aqui uma nova classe chamada "UsuarioController.java", vou dizer que isso aqui é um `@Controller` e vou dizer que para poder chegar

aqui, tem um `@RequestMapping("usuário")` .

[02:03] Eu vou pegar essa consulta que fizemos aqui no "/home" indo por status - aliás, vou copiar isso aqui também e vou colar no "UsuarioController.java". Também vou precisar do `@AutoWired` .

[02:18] E ele, especificamente, vai buscar os dados do usuário. Aqui no status também vai ser `findByStatusEUsuario` . Eu não tenho problema nenhum em misturar inglês e português, por isso que vocês vão ver que eu não tenho padrão em relação a isso.

[02:36] Eu vou criar aqui um método, vou copiar isso daqui, `@Query("select p from Pedido p join p.user u where u.username = :username")` , nós já vimos como faz antes. Aqui não temos que fazer muita coisa, é só colocarmos o "status" e o "username". Já fizemos isso antes, vamos conseguir fazer rapidamente.

[02:53] Eu só preciso passar aqui o `@Param` , colocar `("username")` e nós vamos usar esse mesmo `Param` para nomear o parâmetro, para podermos usá-lo na query.

[03:09] Então já fizemos o *select* pedido pelo *username*, colocamos um `and` aqui e digitamos `p.status :status` . Esse `@Query("select p from Pedido p join p.user u where u.username = :username and p.status :status)` vai estar mapeado para esse atributo, `status` , e o primeiro vai estar mapeado para esse daqui, `username` . Beleza, vai funcionar!

[03:33] `finByStatusEUsuario` , lembrando que vamos precisar do `Principal principal` também. Eu vou jogar aqui no final, tem o status e no final temos que colocar. Está ficando meio grande essa linha, mas tudo bem. A mesma alteração que eu fiz para buscar pelo usuário eu fiz aqui para buscar pelo usuário e pelo status, o resto permanece igual.

[04:00] Para acessar ele tem que fazer pelo "/usuario", mas não vai ser assim: "Me dê os dados". Digite um *get* para usuário e veja os pedidos do usuário. Então é legal fazermos aqui pedido ou pedidos, `pedido`. Eu estou colocando tudo no singular, mas o ideal é colocar no plural. Vamos fazer isso quando fizermos API REST.

[04:28] Já fiz a alteração, ele está redirecionando aqui para `/home`, eu vou só mudar aqui para `usuario/home`; aqui também, `usuario/home` e aqui também, `/usuario/home`. Se der alguma exceção, ele vai para a *home* do usuário. Isso nós estávamos vendo que o status poderia estar inválido e poderia dar esse `IllegalArgumentException`.

[04:54] Só que não tem nem o *template* do usuário ainda, porque esse *redirect* aqui, é um mapeamento da pasta. Enquanto esse aqui, `pedido/{status}`, é mapeamento da url. Esse aqui, `usuário/home` é mapeamento das pastas dentro de "template".

[05:07] Só relembrando: vou criar aqui uma pasta chamada "usuario" e vou copiar a "home" para cá. Aqui eu vou só alterar umas coisinhas, aqui está `/home/aguardando`, `/home`. Eu vou colocar `/usuario/home`.

[05:25] Lembrando que podemos usar o `th:href` e colocar o `@` aqui, para poder a aplicação mudar de contexto sem quebrar a nossa aplicação. Vou colocar aqui para todos eles, podia ter feito logo tudo, tipo `/usuario/home`, aqui fica `/usuario/home/aguardando`. Vamos fazer tudo certinho, por favor. Aqui porque eu copiei tudo, mas tudo bem. E aqui também. E não esqueça de fechar as chaves.

[06:08] Vamos ver se funciona. O novo, que vai para `/pedido/formulario`, tudo bem, você pode colocar o `th` aqui também.

[06:20] O issitema indicou um problema na sintaxe daquela query que eu fiz. Ele está dizendo que não estava esperando *token*. Faltou o sinal de igual (`=`). Vamos vê-lo carregando de novo. Parece que foi!

[06:42] Vamos entrar na *home* de novo. Deixe-me dar um "logout" aqui, login "joao", "joao". Faço o login, ele vai para a *home*. Vamos alterar logo isso também. Eu vou no WebSecurityConfig, quem que faz o *redirect* no *success* aqui, `.defaultSuccessUrl` . Nele eu vou colocar `("/usuario/home", true)` .

[07:02] A princípio ele vai resetar também por causa disso, eu vou deslogar e vou logar de novo, vou ver se ele muda. Beleza, ele mudou, ele está dando *not found* aqui para "/usuario/home".

[07:16] Isso faz todo sentido, porque eu coloquei `/pedido` , então não foi muito inteligente eu colocar a url `usuario/home` ; mas tudo bem, é porque o nome da página pode ficar assim. Eu vou colocar `pedido` . Vamos voltar para a tela de login. Eu vou dar um logout e entrar com "joao" de novo. Agora foi.

[07:44] O que ficou `/home` foi dentro de usuário, aqui ficou a página principal dele, ficou `/home` , mas as urls ficaram ok. Tipo: `/usuario` , `/pedido` , `pedido/status` . Podemos ver que o `pedido/status` está dando 404 aqui, porque `/usuario/home` . Eu errei.

[08:15] Então: `/usuario/pedido` , `pedido` aqui também. Então `/usuario/pedido` aguardando, aprovado e entregue, e só `/pedido` mesmo. Esse "home" daqui. Pronto. "Aprovado", "Entregue", novo, "Aguardando" também, a mesma coisa. E esse "Novo" vai lá para `pedido/formulario` .

[08:43] Voltando aqui vai para o usuário, "Meus Pedidos", agora faz todo sentido e continuamos tendo a *home*, que ainda está fazendo `findAll` ? Não, a *home* tem que ser ajustada! Vamos melhorá-la, mas ela vai se ajustar.

[08:59] Vamos deixar só o `findAll` mesmo na *home*, ele vai buscar tudo, vai ficar assim mesmo. Depois vamos buscar só o que já foi entregue. Tiramos até esse menu aqui, deixamos tudo mesmo, mas tudo o que foi entregue.

[09:17] Podemos fazer a *home* assim, com todas as entregas que aconteceram por data, mostrando as últimas entregas e tudo mais. Mas a página específica

do usuário, que é `/usuario/pedido`, fica mostrando só os pedidos do usuário.

[09:36] Eu vou criar novos pedidos depois, fora da aula, para nós podermos ver bem essa diferença de um para o outro, e eu vou colocar um status ali de entregue - que é uma coisa que vamos fazer só no final do treinamento.

[09:49] Até o próximo vídeo!