07

Recuperando listagem persistida

Transcrição

De volta ao nosso projeto, iremos integrar a parte de listagem ao nosso banco de dados. Em `jogoteca.py`, temos o seguinte código:

```
@app.route('/')
def index():
    return render_template('lista.html', titulo='Jogos',
                           jogos=lista)
```

É nessa função `index()` que fazemos a listagem, fazendo um `render_template()` da `lista.html` em memória, mas queremos pegar essa lista do banco de dados. A primeira coisa que precisamos fazer, então, é nos livrarmos da lista que criamos em `jogoteca.py`:

```
jogo1 = Jogo('Super Mario', 'Acao', 'SNES')
jogo2 = Jogo('Pokemon Gold', 'RPG', 'GBA')
jogo3 = Jogo('Mortal Kombat', 'LutA', 'SNES')
lista = [jogo1, jogo2, jogo3]
```

Isso fará com que nosso `render_template()` quebre. Para consertarmos isso, criaremos uma nova `lista` que conterá os dados recebidos de `dao.py` - ou seja, `jogo_dao`. Esse objeto pode receber o método `listar()`, que serve justamente para listar os nossos jogos:

```
@app.route('/')
def index():
    lista = jogo_dao.listar()
    return render_template('lista.html', titulo='Jogos',
                           jogos=lista)
```

Somente com essa alteração, já conseguiremos listar os jogos corretamente na página http://127.0.0.1:5000/ (http://127.0.0.1:5000/) - inclusive o teste que executamos no vídeo anterior, e que iremos excluir futuramente.

Nome	Categoria	Console
God of War 4	Ação	PS4
NBA 2k18	Esporte	Xbox One
Rayman Legends	Indie	PS4
Super Mario RPG	RPG	SNES
Super Mario Kart	Corrida	SNES
Fire Emblem Echoes	Estratégia	3DS
teste	teste	teste

O próximo passo é passarmos a lógica de `Usuario` também para o banco de dados. Primeiramente, vamos remover a lista de usuários que criamos em `jogoteca.py` :

```
usuario1 = Usuario('luan', 'Luan Marques', '1234')
usuario2 = Usuario('nico', 'Nico Steppat', '7a1')
usuario3 = Usuario('flavio', 'Flávio', 'javascript')

usuarios = { usuario1.id: usuario1,
             usuario2.id: usuario2,
             usuario3.id: usuario3 }
```

Em seguida, vamos importar a clase `UsuarioDao` do módulo `dao` , criar um objeto do tipo `usuario_dao` e colocar uma instância dentro dele fazendo referência a `UsuarioDao()` , passando nosso banco de dados `db` .

O objetivo `usuarios` da nossa função `autenticar()` terá quebrado, e teremos que alterar esse código.

```
@app.route('/autenticar', methods=['POST',])
def autenticar():
    if request.form['usuario'] in usuarios:
        usuario = usuarios[request.form['usuario']]
        if usuario.senha == request.form['senha']:
            session['usuario_logado'] = usuario.id
            flash(usuario.nome + ' logou com sucesso!')
            proxima_pagina = request.form['proxima']
            return redirect(proxima_pagina)
```

Para pegarmos um usuário do banco, usaremos `usario_dao.buscar_por_id()` , passando o nome de usuário que estamos recebendo na requisição (`request.form['usuario']`).

Em seguida, criaremos uma variável `usuario` para receber esse valor. Dessa forma, poderemos reutilizá-la no restante do código.

```
usuario = usuario_dao.buscar_por_id(request.form['usuario'])
```

Se `usuario` existe (ou seja, se a `id` não retornar `null`), queremos comparar a senha desse usuário (`usuario.senha`) com a senha que está sendo recebida na requisição (`request.form(['senha']):

```
@app.route('/autenticar', methods=['POST',])
def autenticar():
    usuario = usuario_dao.buscar_por_id(request.form['usuario'])
    if usuario:
        if usuario.senha == request.form['senha']:
            session['usuario_logado'] = usuario.id
            flash(usuario.nome + ' logou com sucesso!')
            proxima_pagina = request.form['proxima']
            return redirect(proxima_pagina)
    else :
        flash('Não logado, tente de novo!')
        return redirect(url_for('login'))
```

Pronto! Nossa lógica de usuário já estará funcionando corretamente na aplicação, e isso pode ser verificado em http://127.0.0.1:5000/login (http://127.0.0.1:5000/login).

Agora, se reiniciarmos nosso servidor, os dados que inserimos anteriormente (os novos jogos) continuarão persistindo, pois estão sendo guardados no banco de dados.

O próximo passo é pensarmos em uma maneira de deletar ou editar dados na nossa aplicação. Até lá!