02

Editando jogos

Transcrição

Nosso objetivo agora é criarmos uma página para editar os jogos da lista. Primeiramente iremos trabalhar com a própria tela de edição, que será bem parecida com a tela `/novo`.

Clicando com o botão direito na pasta "templates" e em seguida em "New > HTML File", criaremos um novo arquivo HTML, chamado `editar`. Em seguida, copiaremos o código de `novo.html` e colaremos no novo arquivo.

Inicialmente, precisamos alterar a rota para qual o usuário é mandado quando envia uma requisição - nesse caso, chamaremos a rota de `atualizar`.

```
{% extends "template.html" %}
{% block conteudo %}
      <form action="{{ url_for('atualizar') }}" method="post">
        <fieldset>
          <div class="form-group">
            <label for="nome">Nome</label>
            <input type="text" id="nome" name="nome" class="form-control">
          </div>
          <div class="form-group">
            <label for="categoria">Categoria</label>
            <input type="text" id="categoria" name="categoria" class="form-control">
          </div>
          <div class="form-group">
            <label for="console">Console</label>
            <input type="text" id="console" name="console" class="form-control">
          </div>
          <button type="submit" class="btn btn-primary btn-salvar">Salvar</button>
        </fieldset>
      </form>
{% endblock %}
```

No projeto **jogoteca**, para criarmos um novo jogo, utilizamos as rotas `/novo` e `/criar` - a primeira delas renderiza a tela do formulário, e a outra processa a requisição depois que o formulário é enviado. Vamos copiar o código de `/novo`, já que os processos são bem parecidos.

Definiremos essa nova rota como `/editar`. Como o usuário precisará estar logado para fazer edições, manteremos a parte do código referente à verificação de login, mudando somente a rota para qual ele é enviado - se ele estiver logado, deve ser encaminhado para `editar`.

Além disso, o `render_template()` deve receber o HTML que acabamos de criar, e seu título será "Editando jogo". Assim, teremos:

```
@app.route('/editar')
def editar():
    if 'usuario_logado' not in session or session['usuario_logado'] == None:
        return redirect(url_for('login', proxima=url_for('editar')))
    return render_template('editar.html', titulo='Editando jogo')
```

Como a nossa rota `/atualizar` processará a requisição do formulário, ela deve ser bem parecida com `/criar`. Portanto, copiaremos o código de `/criar` e o editaremos para que se adeque à nossa nova rota. Por enquanto, para facilitar nosso raciocínio, vamos colocar somente um `pass`, e mais tarde trabalharemos com mais detalhes nessa função.

```
@app.route('/atualizar', methods=['POST',])
def atualizar():
    pass
```

Acessando a URL http://127.0.0.1:5000/editar (http://127.0.0.1:5000/editar), será exibida a tela "Editando jogo", com os campos "Nome", "Categoria" e "Console". Porém, todos eles estarão vazios, e na verdade queremos selecionar um jogo para editar - ou seja, não queremos entrar diretamente na página de edição, mas sim a partir da lista de jogos (`/` , ou `index()`).

Seria interessante se, ao lados dos jogos, tivéssemos um botão que nos levasse à página `/editar`.

Em `lista.html`, temos uma tabela com um campo em cada coluna. No *header*, adicionaremos mais uma coluna com `<th></th>`, de modo a manter um visual homogêneo.

Na parte dos dados, adicionaremos um `<td></td>` que consistirá em um link para `editar` (`<a hred ="{{ url_for('editar")>Editar</a>`).

```
{% extends "template.html" %}
{% block conteudo %}
        <table class="table table-striped table-responsive table-bordered">
            <thead class="thead-default">
                <tr>
                    <th>Nome</th>
                    <th>Categoria</th>
                    <th>Console</th>
                    <th></th>
                </tr>
            </thead>
            <tbody>
        {% for jogo in jogos %}
                <tr>
                    <td>{{ jogo.nome }}</td>
                    <td>{{ jogo.categoria }}</td>
                    <td>{{ jogo.console }}</td>
                    <td><a href = "{{ url_for('editar') }}">Editar</a></td>
```

Ainda precisamos informar, de alguma forma, qual jogo estamos acessando. O ideal seria que, na função `editar()`, recebêssemos alguma informação que identificasse o jogo que queremos editar e então passar essa informação para o *template* que renderizá o formulário para fazermos essa edição.

Para isso, com o auxílio dos símbolos `<>`, podemos definir que a rota `/editar` receberá um parâmetro. Entre eles, passaremos a informação - nesse caso, um `int` que representa o `id` do jogo no banco de dados. Assim, a função `editar()` também deverá receber `id` como parâmetro.

```
@app.route('/editar/<int:id>')
def editar(id):
    if 'usuario_logado' not in session or session['usuario_logado'] == None:
        return redirect(url_for('login', proxima=url_for('editar')))
    return render_template('editar.html', titulo='Editando jogo')
```

Agora, quando chamarmos a rota `/editar`, na verdade teremos `/editar/id` - ou seja, o número de identificação daquele jogo. Como esse parâmetro será passado para a função `editar()`, poderemos acessá-lo da forma que quisermos.

Para recuperarmos o jogo do banco a partir do `id`, usaremos o método `busca_por_id()` do objeto `jogo_dao`, passando `id` como parâmetro. Então, criaremos uma variável `jogo` que receberá o retorno desse método.

Em seguida, passaremos essa informação para o *template* simplesmente listando-a nos argumentos de `render_template()`:

```
@app.route('/editar/<int:id>')
def editar(id):
    if 'usuario_logado' not in session or session['usuario_logado'] == None:
        return redirect(url_for('login', proxima=url_for('editar')))
    jogo = jogo_dao.busca_por_id(id)
    return render_template('editar.html', titulo='Editando jogo', jogo=jogo)
```

Agora precisamos consertar nosso código em `lista.html`. Em `{{ url_for('editar') }}`, passaremos um parâmetro nomeado - nesse caso, `id`, que usará o `jogo` que estamos iterando como argumento (`jogo.id`).

```
        {% for jogo in jogos %}
                <tr>
                    <td>{{ jogo.nome }}</td>
                    <td>{{ jogo.categoria }}</td>
                    <td>{{ jogo.console }}</td>
                    <td><a href = "{{ url_for('editar', id=jogo.id) }}"></a></td>
```

Com esse código implementado, teremos um botão funcional de **Editar** ao lado dos nossos jogos. Se clicarmos no primeiro ("God of War 4"), por exemplo, seremos redirecionados para http://127.0.0.1:5000/editar/1 (http://127.0.0.1:5000/editar/1). No entanto, os dados "Nome", "Categoria" e "Console" continuarão sem preenchimento - ou seja, ainda temos que implementar no formulário a forma de mostrar os dados na página.

Para isso, em cada `input`, colocaremos um `value` referente a cada característica do jogo que está sendo renderizado (`nome`, `categoria` e `console`):

```
{% extends "template.html" %}
{% block conteudo %}
      <form action="{{ url_for('atualizar') }}" method="post">
        <fieldset>
          <div class="form-group">
            <label for="nome">Nome</label>
            <input type="text" id="nome" name="nome" class="form-control" value="{{ jogo.nome }}
          </div>
          <div class="form-group">
```

```
            <label for="categoria">Categoria</label>
            <input type="text" id="categoria" name="categoria" class="form-control" value="{{ j
          </div>
          <div class="form-group">
            <label for="console">Console</label>
            <input type="text" id="console" name="console" class="form-control" value="{{ jogo.
          </div>
          <button type="submit" class="btn btn-primary btn-salvar">Salvar</button>
        </fieldset>
      </form>
{% endblock %}
```

Com isso, as informações do jogo aparecerão na tela `/editar`. Porém, a edição ainda não está funcionando - se salvarmos qualquer alteração, receberemos um **ValueError: View function did not return a response**. Para resolvermos isso, teremos que implementar uma lógica de atualização no nosso banco de dados, e é isso que faremos no próximo vídeo. Até lá!