09

Mãos na massa: Concluindo o CRUD

Chegou a hora de você executar o que foi visto na aula! Para isso, execute os passos listados abaixo.

> Se você quiser o projeto completo feito no treinamento anterior, pode baixá-lo aqui (https://caelum-online-public.s3.amazonaws.com/739-python-flask2/01/aula1.zip).

Página de edição de jogo

1) Para editar, é necessário uma nova tela. Então, em **jogoteca/templates**, crie um novo HTML, o **editar.html**, com o mesmo conteúdo do arquivo **novo.html**, mudando somente para onde vai o seu formulário:

```
{% 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 %}
```

2) Agora, em **jogoteca.py**, crie a rota `editar` com o mesmo conteúdo da rota `novo`:

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

3) Modifique a nova rota para a edição de jogos, ou seja, caso o usuário não esteja logado, ele deverá ser redirecionado para a URL `editar`, e o template será o **editar.html** com título **Editando Jogo**:

```
@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')
```

4) Para identificar qual jogo você irá editar, receba o seu `id` como parâmetro na URL e na função:

```
@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')
```

5) Com o `id` em mãos, busque-o no banco e passe-o para o *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)
```

6) Não faz sentido você acessar diretamente a página de edição, a ideia é que você a acesse através da página de lista de jogos, que é a página **lista.html**. Então, nessa página, adicione um novo elemento no *header* da tabela e uma nova `td` com um link para a página **editar** e um parâmetro nomeado que faz referência ao `id` do jogo:

```
{% 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', id=jogo.id) }}">Editar</a>
                    </td>
                </tr>
            {% endfor %}
            </tbody>
        </table>
{% endblock %}
```

7) Falta ainda exibir as informações do jogo no formulário **editar.html**:

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

Atualizando o jogo no banco de dados

8) Falta criar a rota `atualizar`. Em **jogoteca.py**, crie-a baseada na rota `criar`:

```
@app.route('/atualizar', methods=['POST',])
def atualizar():
    nome = request.form['nome']
    categoria = request.form['categoria']
    console = request.form['console']
    jogo = Jogo(nome, categoria, console)
    jogo_dao.salvar(jogo)
    return redirect(url_for('index'))
```

9) Para atualizar o jogo, é preciso do seu `id`, então o ideal é que ele seja passado pelo formulário também. Para o usuário não visualizá-lo, crie um *input* do tipo `hidden`, com o valor do `id` do jogo, na página **editar.html**, antes do `fieldset`:

```
<input type="hidden" name="id" value="{{ jogo.id }}">
```

10) Por fim, voltando ao **jogoteca.py**, na função `atualizar`, capture e passe esse `id` na hora de instanciar o jogo, depois disso teste a aplicação:

```
@app.route('/atualizar', methods=['POST',])
def atualizar():
    nome = request.form['nome']
    categoria = request.form['categoria']
    console = request.form['console']
    jogo = Jogo(nome, categoria, console, id=request.form['id'])
```

```
    jogo_dao.salvar(jogo)
    return redirect(url_for('index'))
```

Deletando um jogo

11) Para deletar um jogo, adicione um novo link na página **listar.html**, semelhante ao link de edição, mas redirecionando para a rota `deletar`:

```
{% 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', id=jogo.id) }}">Editar</a>
                        <a href="{{ url_for('deletar', id=jogo.id) }}">Deletar</a>
                    </td>
                </tr>
            {% endfor %}
            </tbody>
        </table>
{% endblock %}
```

12) E em **jogoteca.py**, crie a rota `deletar`, deletando o jogo através do método `deletar` do `dao`:

```
@app.route('/deletar/<int:id>')
def deletar(id):
    jogo_dao.deletar(id)
    flash('O jogo foi removido com sucesso!')
    return redirect(url_for('index'))
```

Melhorando a usabilidade

13) Para melhorar a criação de jogos, ao invés de acessar diretamente a sua URL, na página **listar.html**, adicione um novo link, antes da criação da tabela:

```
<a class="btn btn-primary" href="{{ url_for('novo') }}">Novo Jogo</a>
```

14) E nas páginas **novo.html** e **editar.html**, adicione um link para voltar à listagem, como último elemento do `fieldset`:

```
<a class="btn btn-danger" href="{{ url_for('index') }}">Voltar</a>
```