



Django Framework do Zero

Como Criar um Aplicativo Web

Canal [youtube.com/@VamosCodar](https://www.youtube.com/@VamosCodar) - Wanderson Reis

19 - Cadastro Completo de Alternativas de Respostas

Requisitos

- 1) Aulas 01 a 18 - Django Framework do Zero - Playlist:

<https://www.youtube.com/playlist?list=PLFOqHo8oljzewcT23HCxJV0xWO451CTJe>

Introdução

Na implementação de sistemas é necessário gerenciar corretamente os relacionamentos entre os models (ou tabelas). A camada do banco de dados em geral (conforme a configuração) não permitirá a criação de um novo registro que exige uma chave estrangeira se esta estiver ausente. Por isso, no caso do nosso sistema de enquetes, ao criar uma nova alternativa (choice) precisamos associar a qual pergunta ela se relaciona. Existe uma dependência obrigatória, alternativa não existe sem a respectiva pergunta. Hoje vamos implementar o CRUD completo para as alternativas, garantindo a criação correta do relacionamento.

Nesta aula:

- 1) Gestão de alternativas durante edição da Pergunta
- 2) Criar a ChoiceCreateView para registrar novas alternativas
- 3) Criar a ChoiceUpdateView para editar alternativas
- 4) Criar a ChoiceDeleteView para excluir alternativas
- 5) Preparar os templates para os formulários
- 6) Criar as rotas para o CRUD de alternativas

Observação: Lembre-se de estar com o ambiente virtual ativado (venv).

Passo 1: Gestão de alternativas durante edição da Pergunta

Em questão de telas, podemos organizar o fluxo de diversas formas. Como exemplo, vamos permitir a criação de novas alternativas para a pergunta diretamente na tela de edição de perguntas. A ideia é mostrar todas as alternativas com opções de criar, editar e excluir.

Editar o arquivo **template/polls/question_form.html** e modificá-lo para acrescentar o bloco de código conforme indicado na próxima imagem.

```
templates > polls > question_form.html > ...
{{ form.as_p }} {% # imprime o objeto form formatado em tags HTML %}
<button type="submit" class="btn btn-success">Salvar</button>
</form>
{% if object %}
<div class="mt-5">
<h3>Alternativas de Pergunta</h3>
<a href="#" class="btn btn-primary mb-5">
+ Incluir Alternativa
</a>
<ol>
{% for choice in object.choice_set.all %}
<li>
{{ choice.choice_text }} &nbsp;
<a href="#" class="link-secondary">
Editar
</a> |
<a href="#" class="link-secondary">
Excluir
</a>
</li>
{% empty %}
<li>Não existem alternativas para a pergunta</li>
{% endfor %}
</ol>
</div>
{% endif %}
</div>
```

(salve o arquivo).

Os links para as ações de criar, editar e excluir serão ajustados após a implementação das respectivas rotas.

Passo 2: Criar a ChoiceCreateView para registrar novas alternativas

Nestes próximos três passos será implementado o CRUD para as alternativas (model Choice). Segue basicamente o mesmo princípio que fizemos para as perguntas (Question). A diferença fundamental é o pré-carregamento da pergunta a qual será associada à alternativa (choice). Outro detalhe é a implementação de um redirecionamento dinâmico para a página de edição da pergunta após incluir uma nova alternativa. Só vamos conseguir observar tudo após a criação dos templates e rotas.

Editar o arquivo **polls/views.py** e implementar a view **ChoiceCreateView** conforme o código indicado na imagem abaixo.

```
polls > views.py > ...  
  
class ChoiceCreateView(LoginRequiredMixin, CreateView):  
    model = Choice  
    template_name = 'polls/choice_form.html'  
    fields = ('choice_text', )  
    success_message = 'Alternativa registrada com sucesso!'  
  
    def dispatch(self, request, *args, **kwargs):  
        self.question = get_object_or_404(Question, pk=self.kwargs.get('pk'))  
        return super(ChoiceCreateView, self).dispatch(request, *args, **kwargs)  
  
    def get_context_data(self, **kwargs):  
        context = super(ChoiceCreateView, self).get_context_data(**kwargs)  
        context['form_title'] = f'Alternativa para:{self.question.question_text}'  
  
        return context  
  
    def form_valid(self, form):  
        form.instance.question = self.question  
        messages.success(self.request, self.success_message)  
        return super(ChoiceCreateView, self).form_valid(form)  
  
    def get_success_url(self, *args, **kwargs):  
        question_id = self.kwargs.get('pk')  
        return reverse_lazy('question_edit', kwargs={'pk': question_id})
```

view generica para criacao de alternativas para uma determinada pergunta pela chave primaria (PK)

(salve o arquivo).

Passo 3: Criar a ChoiceUpdateView para editar alternativas

Editar o arquivo **polls/views.py** e implementar a view **ChoiceUpdateView** conforme o código indicado na imagem abaixo.

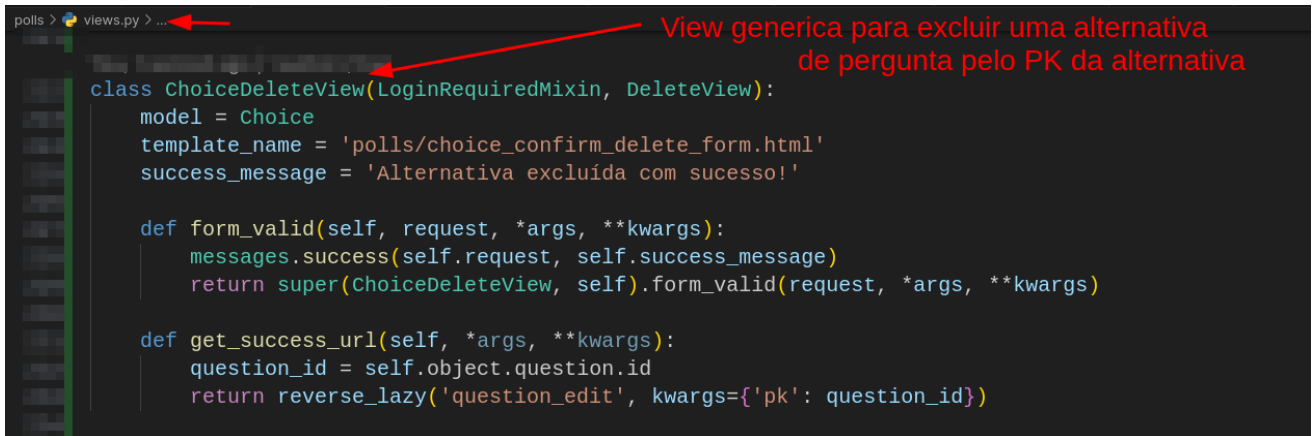
```
polls > views.py > ...  
  
class ChoiceUpdateView(LoginRequiredMixin, UpdateView):  
    model = Choice  
    template_name = 'polls/choice_form.html'  
    fields = ('choice_text', )  
    success_message = 'Alternativa atualizada com sucesso!'  
  
    def get_context_data(self, **kwargs):  
        context = super(ChoiceUpdateView, self).get_context_data(**kwargs)  
        context['form_title'] = 'Editando alternativa'  
  
        return context  
  
    def form_valid(self, request, *args, **kwargs):  
        messages.success(self.request, self.success_message)  
        return super(ChoiceUpdateView, self).form_valid(request, *args, **kwargs)  
  
    def get_success_url(self, *args, **kwargs):  
        question_id = self.object.question.id  
        return reverse_lazy('question_edit', kwargs={'pk': question_id})
```

View generica para editar uma alternativa de pergunta (com PK = question.id)

(salve o arquivo).

Passo 4: Criar a ChoiceDeleteView para excluir alternativas

Editar o arquivo **polls/views.py** e implementar a view **ChoiceDeleteView** conforme o código indicado na imagem abaixo.

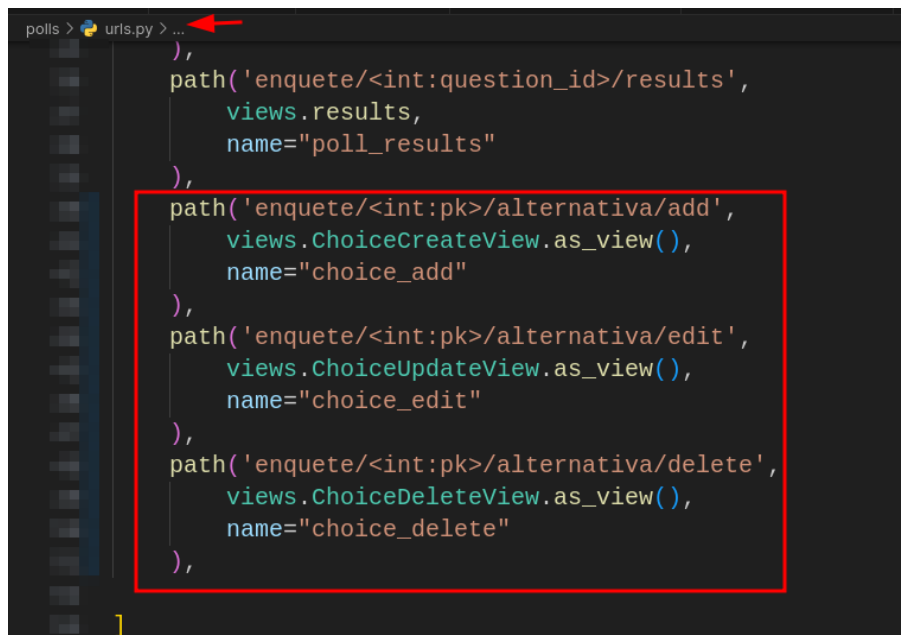


```
polls > polls > views.py > ...  
  
class ChoiceDeleteView(LoginRequiredMixin, DeleteView):  
    model = Choice  
    template_name = 'polls/choice_confirm_delete.html'  
    success_message = 'Alternativa excluída com sucesso!'  
  
    def form_valid(self, request, *args, **kwargs):  
        messages.success(self.request, self.success_message)  
        return super().form_valid(request, *args, **kwargs)  
  
    def get_success_url(self, *args, **kwargs):  
        question_id = self.object.question.id  
        return reverse_lazy('question_edit', kwargs={'pk': question_id})
```

(salve o arquivo).

Passo 5: Criar as rotas para o CRUD de alternativas

Para concluir a implementação vamos criar as respectivas rotas para o CRUD de alternativas. Sendo que a lista de alternativas será exibida na tela de edição da pergunta. Editar o arquivo **polls/urls.py** e modificá-lo para incluir as rotas conforme a imagem abaixo.



```
polls > polls > urls.py > ...  
  
),  
    path('enquete/<int:question_id>/results',  
         views.results,  
         name="poll_results"  
    ),  
    path('enquete/<int:pk>/alternativa/add',  
         views.ChoiceCreateView.as_view(),  
         name="choice_add"  
    ),  
    path('enquete/<int:pk>/alternativa/edit',  
         views.ChoiceUpdateView.as_view(),  
         name="choice_edit"  
    ),  
    path('enquete/<int:pk>/alternativa/delete',  
         views.ChoiceDeleteView.as_view(),  
         name="choice_delete"  
    ),  
]  
]
```

(salve o arquivo).

Passo 6: Preparar os templates para os formulários

Neste passo vamos criar dois templates novos **templates/polls/choice_form.html** e **templates/polls/choice_confirm_delete_form.html** (ambos cópias equivalentes usados para question). E modificar o template **template/polls/question_form.html** (para mostrar as alternativas) da pergunta.

Editar o arquivo **template/polls/question_form.html** e modificá-lo para acrescentar o bloco de código conforme indicado na próxima imagem.

```
templates > polls > question_form.html > ...
{% csrf_token %}
{{ form.as_p }} {%# imprime o objeto form formato em tags HTML #}
<button type="submit" class="btn btn-success">Salvar</button>
</form>
{% if object %}
<div class="mt-5">
  <h3>Alternativas de Pergunta</h3>
  <a href="{% url 'choice_add' pk=object.id %}" class="btn btn-primary mb-5">
    + Incluir Alternativa
  </a>
  <ol>
    {% for choice in object.choice_set.all %}
      <li>
        {{ choice.choice_text }} &nbsp;
        <a href="{% url 'choice_edit' pk=choice.id %}" class="link-secondary">
          Editar
        </a> |
        <a href="{% url 'choice_delete' pk=choice.id %}" class="link-secondary">
          Excluir
        </a>
      </li>
    {% empty %}
      <li>Não existem alternativas para a pergunta</li>
    {% endfor %}
  </ol>
</div>
{% endif %}
</div>
```

(salve o arquivo).

Os demais templates são equivalentes aos usados no CRUD do model Question e podem ser copiados e adaptados conforme a necessidade.

Novo template **templates/polls/choice_form.html**

```
templates > polls > choice_form.html > ...
1 {% extends '_layout1.html' %}
2 {% block content %}
3 <div class="card">
4   <div class="card-header">
5     <h5 class="card-title">{{ form_title }}</h5>
6   </div>
7   <div class="card-body">
8     <form method="post">
9       {% csrf_token %}
10      {{ form.as_p }}
11      <button type="submit" class="btn btn-success">Salvar</button>
12    </form>
13  </div>
14 </div>
15 {% endblock %}
```

(salve o arquivo).

Novo template `templates/polls/choice_confirm_delete_form.html`

```
templates > polls > choice_confirm_delete_form.html > ...  
1 {% extends '_layout1.html' %}  
2  
3 {% block content %}  
4 <div class="card">  
5     <div class="card-header">  
6         <h5 class="card-title">Excluindo a Alternativa Nº {{ object.id }}</h5>  
7     </div>  
8     <div class="card-body">  
9  
10         <form method="post">  
11             {% csrf_token %}  
12             <h6>Você tem certeza que deseja excluir a alternativa abaixo ?</h6>  
13             <br />  
14             <p>"{{ object.choice_text }}"</p>  
15             <br />  
16             <button type="submit" class="btn btn-danger">Confirmar</button>  
17             <a href="{% url 'question_edit' pk=object.question.id %}" class="link-secondary">Cancelar</a>  
18         </form>  
19     </div>  
20 </div>  
21 {% endblock %}
```

(salve os arquivos).

Finalizada a implementação das rotas, execute o servidor de desenvolvimento e acesse a edição de alguma das perguntas cadastradas (cadastre uma nova se necessário).

`python manage.py runserver` (ENTER)

Teste o fluxo completo criando, editando e incluindo alternativos (excluir vai exigir o login). Lembre-se que o acesso às ações de criar, editar e excluir ficou na página de editar a Pergunta (Enquete).