

# Back-End Frameworks

Aula 6 - 24/03/2023



[https://bit.ly/23\\_1\\_backend\\_n](https://bit.ly/23_1_backend_n)



## O que temos para **hoje**?

- Utilizar banco de dados na view e no template;
- Votar em uma enquete;
- Utilizar campos de imagem no banco de dados;
- Continuação do trabalho;



# Utilizar banco de dados na view e no template



## Utilizar banco de dados na view e no template

- Atualmente, o arquivo polls/views.py está assim:

```
1  from django.shortcuts import render
2
3  # Create your views here.
4  from django.http import HttpResponse
5
6
7  def index(request):
8      context = {}
9      return render(request, 'index.html', context)
10
11 def detail(request, question_id):
12     context = {}
13     return render(request, 'detail.html', context)
14
15 def results(request, question_id):
16     context = {}
17     return render(request, 'results.html', context)
18
19 def vote(request, question_id):
20     context = {}
21     return render(request, 'vote.html', context)
```



## Utilizar banco de dados na **view** e no **template**

- No arquivo polls/views.py, modificar o código para ficar assim:

```
1  from django.shortcuts import render
2
3  # Create your views here.
4  from django.http import HttpResponse
5
6  from .models import Question
7
8  def index(request):
9      questoes = Question.objects.all()
10     context = {
11         'questoes_template': questoes
12     }
13     return render(request, 'index.html', context)
```



## Utilizar banco de dados na **view** e no **template**

- No arquivo polls/templates/index.html, modificar o código para ficar assim:

```
1  {% extends "base.html" %}
2  {% load static %}
3  {% block body %}
4  <h1>Questões cadastradas: </h1>
5  {% for questao in questoes_template %}
6  <h2>
7      {{questao.question_text}}
8  </h2>
9  <p>
10     Publicada em: {{questao.pub_date}}
11 </p>
12 <p>
13     Publicada em: {{questao.pub_date | date:"d/m/Y H:i:s"}}
14 </p>
15 <hr>
16 {% endfor %}
17 {% endblock %}
```



## Utilizar banco de dados na **view** e no **template**

○ Como ficou o resultado?



## Utilizar banco de dados na view e no template

### Questões cadastradas:

Acerca de Bootstrap e Javascript, julgue o próximo item.

Publicada em: March 15, 2023, noon

Publicada em: 15/03/2023 12:00:00

O Bootstrap é uma estrutura popular de HTML, CSS e JavaScript para o desenvolvimento de websites responsivos e móveis. São classes do Bootstrap para estilizar botões

Publicada em: March 10, 2023, 5:01 p.m.

Publicada em: 10/03/2023 17:01:17

A maneira correta de criar um layout de grid responsivo usando o Bootstrap é usar a classe

Publicada em: March 15, 2023, 6 p.m.

Publicada em: 15/03/2023 18:00:00





## Utilizar banco de dados na view e no template

- No arquivo polls/templates/index.html, modificar o código para ficar assim:

```
1 {% extends "base.html" %}
2 {% load static %}
3 {% block body %}
4 <h1>Questões cadastradas: </h1>
5 {% for questao in questoes_template %}
6 <h2>
7     {{questao.question_text}}
8 </h2>
9 <p>
10     Publicada em: {{questao.pub_date}}
11 </p>
12 <p>
13     Publicada em: {{questao.pub_date | date:"d/m/Y H:i:s"}}
14 </p>
15 <p>
16     <a href="{% url 'detail' questao.id %}">Ver detalhes</a>
17 </p>
18 <p>
19     <a href="{% url 'results' questao.id %}">Ver resultados</a>
20 </p>
21 <p>
22     <a href="{% url 'vote' questao.id %}">Votar</a>
23 </p>
24 <hr>
25 {% endfor %}
26 {% endblock %}
```



## Utilizar banco de dados na **view** e no **template**

○ Como ficou o resultado?



## Utilizar banco de dados na view e no template

### Questões cadastradas:

Acerca de Bootstrap e Javascript, julgue o próximo item.

Publicada em: March 15, 2023, noon

Publicada em: 15/03/2023 12:00:00

[Ver detalhes](#)

[Ver resultados](#)

[Votar](#)

O Bootstrap é uma estrutura popular de HTML, CSS e JavaScript para o desenvolvimento de websites responsivos e móveis. São classes do Bootstrap para estilizar botões

Publicada em: March 10, 2023, 5:01 p.m.

Publicada em: 10/03/2023 17:01:17

[Ver detalhes](#)

[Ver resultados](#)

[Votar](#)

A maneira correta de criar um layout de grid responsivo usando o Bootstrap é usar a classe

Publicada em: March 15, 2023, 6 p.m.

Publicada em: 15/03/2023 18:00:00

[Ver detalhes](#)

[Ver resultados](#)

[Votar](#)



## Utilizar banco de dados na view e no template

- No arquivo polls/views.py, modificar o código para ficar assim:

```
1  from django.shortcuts import render
2
3  # Create your views here.
4  from django.http import HttpResponse
5
6  from .models import Question, Choice
7
8  def index(request):
9      questoes = Question.objects.all()
10     context = {}
11     context['questoes_template'] = questoes
12
13     return render(request, 'index.html', context)
14
15  def detail(request, question_id):
16      questao = Question.objects.get(id=question_id)
17      opcoes = Choice.objects.filter(question = question_id)
18     context = {
19         'pergunta': questao,
20         'respostas': opcoes
21     }
22     return render(request, 'detail.html', context)
```



## Utilizar banco de dados na **view** e no **template**

- No arquivo polls/templates/detail.html, modificar o código para ficar assim:

```
1 {% extends "base.html" %}
2 {% block body %}
3 <h1>Detalhe da pergunta: {{pergunta.question_text}}</h1>
4 {% for resposta in respostas %}
5 <p>
6     {{resposta.choice_text}}
7 </p>
8 {% endfor %}
9 {% endblock %}
```



## Utilizar banco de dados na **view** e no **template**

☒ Como ficou o resultado?

**Detalhe da pergunta: Acerca de Bootstrap e Javascript, julgue o próximo item.**

Certo

Errado



## Utilizar banco de dados na **view** e no **template**

- Exercício: na página de resultados, mostrar o número de votos de cada resposta de uma pergunta;



## Utilizar banco de dados na **view** e no **template**

- Documentação completa:  
<https://docs.djangoproject.com/pt-br/4.1/topics/db/queries/>





## Utilizar banco de dados na **view** e no **template**

- No arquivo polls/views.py, modificar o código para ficar assim:

```
24  def results(request, question_id):
25      questao = Question.objects.get(id=question_id)
26      opcoes = Choice.objects.filter(question = question_id)
27      context = {}
28      context['pergunta'] = questao,
29      context['respostas'] = opcoes
30      }
31      return render(request, 'results.html', context)
```



## Utilizar banco de dados na **view** e no **template**

- No arquivo polls/templates/results.html, modificar o código para ficar assim:

```
1 {% extends "base.html" %}
2 {% block body %}
3 <h1>Resultado da pergunta: {{pergunta.question_text}}</h1>
4 {% for resposta in respostas %}
5 <p>
6     {{resposta.choice_text}}: {{resposta.votes}} voto(s)
7 </p>
8 {% endfor %}
9 {% endblock %}
```



# Votar em uma enquete



## Votar em uma **enquete**

- No arquivo `polls/views.py`, modificar o código para ficar assim:

```
33  def vote(request, question_id):
34      questao = Question.objects.get(id=question_id)
35      opcoes = Choice.objects.filter(question = question_id)
36      context = {
37          'pergunta': questao,
38          'respostas': opcoes
39      }
40      return render(request, 'vote.html', context)
```



## Votar em uma **enquete**

- No arquivo `polls/templates/vote.html`, modificar o código para ficar assim:

```
1  {% extends "base.html" %}
2  {% block body %}
3  <h1>Vote na pergunta: {{pergunta.question_text}}</h1>
4  <div>
5      <form action="" method="post">
6          {% csrf_token %}
7          {% for resposta in respostas %}
8              <div class="form-check">
9                  <input class="form-check-input" type="radio" name="voto" value="{{ resposta.id }}">
10                 <label class="form-check-label">
11                     {{ resposta.choice_text }}
12                 </label>
13             </div>
14         {% endfor %}
15         <button type="submit" class="btn btn-primary">Enviar</button>
16     </form>
17 </div>
18
19 {% endblock %}
```



## Votar em uma **enquete**

🕒 Como ficou o resultado?

**Vote na pergunta: O Bootstrap é uma estrutura popular de HTML, CSS e JavaScript para o desenvolvimento de websites responsivos e móveis.**  
**São classes do Bootstrap para estilizar botões**

- ☐ button-success, button-info.
- ☐ btn , btn-default.
- ☐ btn-warning, button-danger.
- ☐ button-link, btn-primary.

Enviar



## Votar em uma **enquete**

- No arquivo polls/views.py, modificar o código para ficar assim:

```
33 ∨ def vote(request, question_id):|
34 ∨     if request.method == 'POST':
35         post = request.POST
36         opcao_votada = Choice.objects.get(id=post['voto'])
37         opcao_votada.votes = opcao_votada.votes + 1
38         opcao_votada.save()
39         questao = Question.objects.get(id=question_id)
40         opcoes = Choice.objects.filter(question = question_id)
41 ∨     context = {
42         'pergunta': questao,
43         'respostas': opcoes
44     }
45     return render(request, 'vote.html', context)
```



## Votar em uma **enquete**

- Fazer uma votação e ir na página de resultados;
- Como ficou:





## Votar em uma **enquete**

**Resultado da pergunta: O Bootstrap é uma estrutura popular de HTML, CSS e JavaScript para o desenvolvimento de websites responsivos e móveis. São classes do Bootstrap para estilizar botões**

button-success, button-info.: 1 voto(s)

btn , btn-default.: 0 voto(s)

btn-warning, button-danger.: 1 voto(s)

button-link, btn-primary.: 0 voto(s)



## Votar em uma **enquete**

- No arquivo polls/views.py, modificar o código para ficar assim:

```
34 def vote(request, question_id):
35     if request.method == 'POST':
36         post = request.POST
37         opcao_votada = Choice.objects.get(id=post['voto'])
38         opcao_votada.votes = opcao_votada.votes + 1
39         opcao_votada.save()
40         messages.success(request, 'Voto computado com sucesso.')
41         return redirect("index")
42
43     questao = Question.objects.get(id=question_id)
44     opcoes = Choice.objects.filter(question = question_id)
45     context = {
46         'pergunta': questao,
47         'respostas': opcoes
48     }
49     return render(request, 'vote.html', context)
```



## Votar em uma **enquete**

- No arquivo polls/templates/base.html, modificar o código para ficar assim:

```
<div class="col-12 py-3">
    {% block msg_sucesso %}
        {% if messages %}
            {% for message in messages %}
                <div class="alert alert-success" role="alert">
                    {{ message }}
                </div>
            {% endfor %}
        {% endif %}
    {% endblock %}
</div>
<div class="col-12">
    {% block body %} {% endblock %}
</div>
```



# Utilizar campos de imagem no banco de dados



## Utilizar campos de **imagem** no banco de dados

- No terminal, executar o comando:  
`py -m pip install pillow`



## Utilizar campos de **imagem** no banco de dados

- No arquivo `polls/models.py`, modificar pra ficar assim:

```
4 class Question(models.Model):  
5     question_text = models.CharField(max_length=200)  
6     pub_date = models.DateTimeField('date published')  
7     imagem = models.ImageField(upload_to='uploads/', default="")
```



## Utilizar campos de **imagem** no banco de dados

- No terminal, executar o comando:  
`py manage.py makemigrations`
- Depois:  
`py manage.py migrate`



## Utilizar campos de **imagem** no banco de dados

- Abrir a url:  
<http://127.0.0.1:8000/admin/polls/question/1/change/>
- Como ficou?







## Utilizar campos de **imagem** no banco de dados

**Question object (1)**

**Question text:**

**Date published:**

**Date:**  **Today** | 

**Time:**  **Now** | 

Note: You are 3 hours behind server time.

**Imagem:**  Nenhum arquivo escolhido



## Utilizar campos de **imagem** no banco de dados

- Escolher uma imagem qualquer e enviar nesse campo;
- Voltar para a edição da pergunta;
- Como ficou?




## Utilizar campos de **imagem** no banco de dados


**Question object (1)**

**Question text:**

---

**Date published:**

**Date:**  [Today](#) | 

**Time:**  [Now](#) | 

Note: You are 3 hours behind server time.

---

**Imagem:**

**Currently:** [uploads/bootstrap-logo-shadow.png](#)

**Change:**  Nenhum arquivo escolhido



## Utilizar campos de **imagem** no banco de dados

- Clicar no link da imagem;
- O que acontece?



## Utilizar campos de **imagem** no banco de dados

### Page not found (404)

Request Method: GET

Request URL: `http://127.0.0.1:8000/uploads/bootstrap-logo-shadow.png`

Using the URLconf defined in `mysite.urls`, Django tried these URL patterns, in this order:

1. `polls/`
2. `admin/`

The current path, `uploads/bootstrap-logo-shadow.png`, didn't match any of these.



## Utilizar campos de **imagem** no banco de dados

- No arquivo `mysite/settings.py`, adicionar essas linhas:

```
18 EDIA_URL = "/uploads/"
19 MEDIA_ROOT = BASE_DIR
```



## Utilizar campos de **imagem** no banco de dados

- No arquivo `mysite/urls.py`, modificar para ficar assim:

```
16  ✓ from django.contrib import admin
17    from django.urls import include, path
18    from django.conf import settings
19    from django.conf.urls.static import static
20
21  ✓ urlpatterns = [
22      path('polls/', include('polls.urls')),
23      path('admin/', admin.site.urls),
24  ]
25
26  urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```



## Utilizar campos de **imagem** no banco de dados

- Clicar no link da imagem;
- O que acontece?





## Utilizar campos de **imagem** no banco de dados

- No arquivo polls/templates/detail.html, modificar para ficar assim:

```
1  {% extends "base.html" %}
2  {% block body %}
3  <h1>Detalhe da pergunta: {{pergunta.question_text}}</h1>
4  <p>
5      
6  </p>
7  {% for resposta in respostas %}
8  <p>
9      {{resposta.choice_text}}
10 </p>
11 {% endfor %}
12 {% endblock %}
```



## Utilizar campos de **imagem** no banco de dados

- ⦿ Acessar o detalhe da pergunta que você mandou a resposta;
- ⦿ O que acontece?



## Utilizar campos de **imagem** no banco de dados

- Exercício: repetir a mostra da imagem nos arquivos: results.html e vote.html;



# Continuação do trabalho



## Continuação do **trabalho**

- Na sua tabela "post", adicionar um campo imagem;
- Na página index do seu app blog, você deve listar todas as postagens da seguinte forma:
  - Imagem, nome da postagem e um botão para acessar o detalhe da postagem;



## Continuação do **trabalho**

- Ao clicar no botão, o usuário deve ser direcionado para uma página que será mostrado apenas o conteúdo da postagem selecionada;
- Mostrar os comentários da postagem;
- Você deve adicionar nessa página um formulário para inserção de comentários;



## Continuação do **trabalho**

- Ao inserir um comentário, o usuário deve ser direcionado para a página inicial e deve receber uma mensagem de sucesso;



# Obrigado!

*Alguma pergunta?*

Contato:

☎ 040601692@prof.unama.br