ABAIXO ESTÁ O DESENVOLVIMENTO COMPLETO DE UM SISTEMA DE ENQUETES ONLINE PARA ESTATÍSTICAS DAS ELEIÇÕES USANDO DJANGO, COM INTEGRAÇÃO DE BOOTSTRAP, JQUERY E AJAX.

---

1. Configuração do Ambiente

1.1 Instalação do Django e Dependências

Crie um ambiente virtual e instale o Django:

```
python -m venv env
source env/bin/activate   # Linux/Mac
env\Scripts\activate      # Windows
pip install django django-crispy-forms
```

Crie um novo projeto e app Django:

```
django-admin startproject election_poll
cd election_poll
python manage.py startapp poll
```

1.2 Configuração Inicial

Adicione o app `poll` ao `INSTALLED_APPS` no arquivo `settings.py`:

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'poll',
    'crispy_forms',  # Suporte a formulários com Bootstrap
]
CRISPY_TEMPLATE_PACK = 'bootstrap4'
```

Configure o banco de dados SQLite padrão e migre:

```
python manage.py migrate
```

---

2. Modelos (Models)

No arquivo `poll/models.py`, defina os modelos para Enquete e Opções de Resposta:

```
from django.db import models

class Poll(models.Model):
    question = models.CharField(max_length=200)
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.question

class Option(models.Model):
    poll = models.ForeignKey(Poll, on_delete=models.CASCADE,
related_name='options')
    option_text = models.CharField(max_length=100)
    votes = models.IntegerField(default=0)

    def __str__(self):
        return self.option_text
```

Crie e aplique as migrações:

```
python manage.py makemigrations
python manage.py migrate
```

---

3. Interface de Administração

No arquivo `poll/admin.py`, registre os modelos:

```
from django.contrib import admin
from .models import Poll, Option

class OptionInline(admin.TabularInline):
    model = Option
    extra = 2

@admin.register(Poll)
class PollAdmin(admin.ModelAdmin):
    list_display = ['question', 'created_at']
    inlines = [OptionInline]
```

---

4. URLs e Views

4.1 URLs

No arquivo `poll/urls.py`, configure as rotas:

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.poll_list, name='poll_list'),
    path('poll/<int:id>/', views.poll_detail, name='poll_detail'),
    path('poll/<int:id>/vote/', views.vote, name='vote'),
    path('poll/<int:id>/results/', views.poll_results, name='poll_results'),
]
```

Adicione as URLs do app ao arquivo principal `election_poll/urls.py`:

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('poll.urls')),
]
```

4.2 Views

No arquivo `poll/views.py`:

```
from django.shortcuts import render, get_object_or_404
from django.http import JsonResponse
from .models import Poll, Option

def poll_list(request):
    polls = Poll.objects.all().order_by('-created_at')
    return render(request, 'poll/poll_list.html', {'polls': polls})

def poll_detail(request, id):
    poll = get_object_or_404(Poll, id=id)
    return render(request, 'poll/poll_detail.html', {'poll': poll})

def vote(request, id):
    if request.method == 'POST':
        poll = get_object_or_404(Poll, id=id)
        option_id = request.POST.get('option_id')
        option = get_object_or_404(Option, id=option_id)
        option.votes += 1
        option.save()
        return JsonResponse({'message': 'Vote counted.'}, status=200)
    return JsonResponse({'message': 'Invalid request.'}, status=400)

def poll_results(request, id):
    poll = get_object_or_404(Poll, id=id)
    options = poll.options.all()
    return render(request, 'poll/poll_results.html', {'poll': poll, 'options':
options})
```

---

5. Templates

Crie a estrutura de pastas `poll/templates/poll/` e adicione os seguintes arquivos:

5.1 Template de Listagem

`poll_list.html`:

```
<h1>Enquetes Disponíveis</h1>
<div class="list-group">
    {% for poll in polls %}
        <a href="{% url 'poll_detail' poll.id %}" class="list-group-item list-
group-item-action">
            {{ poll.question }}
        </a>
    {% endfor %}
</div>
```

5.2 Template de Detalhes

`poll_detail.html`:

```
<h1>{{ poll.question }}</h1>
<form id="vote-form">
    {% for option in poll.options.all %}
        <div class="form-check">
            <input type="radio" name="option_id" value="{{ option.id }}"
class="form-check-input" id="option{{ option.id }}">
```

```
            <label class="form-check-label"
for="option{{ option.id }}">{{ option.option_text }}</label>
        </div>
    {% endfor %}
    <button type="submit" class="btn btn-primary mt-3">Votar</button>
</form>
<script>
    $('#vote-form').on('submit', function(e) {
        e.preventDefault();
        $.ajax({
            type: 'POST',
            url: "{% url 'vote' poll.id %}",
            data: $(this).serialize(),
            success: function(response) {
                alert(response.message);
                window.location.href = "{% url 'poll_results' poll.id %}";
            },
            error: function() {
                alert('Erro ao registrar o voto.');
            }
        });
    });
</script>
```

5.3 Template de Resultados

`poll_results.html`:

```
<h1>Resultados: {{ poll.question }}</h1>
<ul class="list-group">
    {% for option in options %}
        <li class="list-group-item d-flex justify-content-between align-items-
center">
            {{ option.option_text }}
            <span class="badge bg-primary rounded-pill">{{ option.votes
}}</span>
        </li>
    {% endfor %}
</ul>
<a href="/" class="btn btn-secondary mt-3">Voltar</a>
```

---

6. Controle de Votos Duplicados

Implemente um controle simples de IP para evitar votos duplicados. No modelo `vote`, armazene os IPs dos votantes.

No arquivo `poll/views.py`, modifique a função `vote`:

```
from django.core.cache import cache
import hashlib

def get_client_ip(request):
    x_forwarded_for = request.META.get('HTTP_X_FORWARDED_FOR')
    if x_forwarded_for:
        return x_forwarded_for.split(',')[0]
    return request.META.get('REMOTE_ADDR')

def vote(request, id):
    if request.method == 'POST':
        poll = get_object_or_404(Poll, id=id)
        client_ip = get_client_ip(request)
        unique_id = hashlib.sha256(f'{client_ip}-
{poll.id}'.encode()).hexdigest()
        if cache.get(unique_id):
            return JsonResponse({'message': 'Você já votou nesta enquete.'}, 
status=403)
        option_id = request.POST.get('option_id')
        option = get_object_or_404(Option, id=option_id)
        option.votes += 1
        option.save()
        cache.set(unique_id, True, 3600 * 24)  # 24 horas
        return JsonResponse({'message': 'Vote counted.'}, status=200)
    return JsonResponse({'message': 'Invalid request.'}, status=400)
```

---

7. Frontend: Bootstrap e AJAX

Adicione o Bootstrap no arquivo base:

```
<link 
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" 
rel="stylesheet">
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
```

---

8. Deploy

Configure o projeto para deploy em um serviço como Heroku:

1. Configure o `settings.py` para produção.
2. Configure o `requirements.txt` e o servidor WSGI.
3. Faça o deploy e teste a aplicação.

Seu sistema de enquetes está pronto para uso!