Desenvolvimento Completo de um Blog com Sistema de Comentários Utilizando Django, Bootstrap, JQuery, e AJAX

1. Configuração do Ambiente

1.1 Instalação do Django e Dependências

Certifique-se de ter o Python instalado. Em seguida, crie e ative um ambiente virtual:

```
python -m venv env
source env/bin/activate   # Linux/Mac
env\Scripts\activate      # Windows
```

Instale o Django e outras dependências necessárias:

```
pip install django pillow django-crispy-forms
```

Crie um novo projeto Django:

```
django-admin startproject blog_project
cd blog_project
python manage.py startapp blog
```

Adicione o app ao arquivo settings.py:

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'blog',
    'crispy_forms',  # Para facilitar o uso de formulários
]
CRISPY_TEMPLATE_PACK = 'bootstrap4'
```

1.2 Configuração do Banco de Dados

Configure o banco de dados SQLite padrão em settings.py:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / "db.sqlite3",
    }
}
```

Migre o banco de dados:

```
python manage.py migrate
```

---

2. Modelos (Models)

No arquivo models.py do app blog, defina os modelos para Postagens e Comentários:

```
from django.db import models

class Post(models.Model):
    title = models.CharField(max_length=200)
    content = models.TextField()
    image = models.ImageField(upload_to='post_images/', null=True, blank=True)
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.title

class Comment(models.Model):
    post = models.ForeignKey(Post, on_delete=models.CASCADE, related_name='comments')
    name = models.CharField(max_length=100)
    email = models.EmailField()
    content = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    is_approved = models.BooleanField(default=False)

    def __str__(self):
        return f'Comment by {self.name} on {self.post}'
```

Crie e aplique as migrações:

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

---

3. Interface de Administração

No arquivo admin.py, registre os modelos:

```
from django.contrib import admin
from .models import Post, Comment

@admin.register(Post)
class PostAdmin(admin.ModelAdmin):
    list_display = ['title', 'created_at']

@admin.register(Comment)
class CommentAdmin(admin.ModelAdmin):
    list_display = ['name', 'post', 'is_approved', 'created_at']
    list_filter = ['is_approved']
    actions = ['approve_comments']

    def approve_comments(self, request, queryset):
        queryset.update(is_approved=True)
```

---

4. URLs e Views

4.1 URLs

No arquivo urls.py do app blog, configure as rotas:

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

urlpatterns = [
    path('', views.post_list, name='post_list'),
    path('post/<int:id>/', views.post_detail, name='post_detail'),
    path('post/<int:id>/comment/', views.add_comment, name='add_comment'),
]
```

Adicione as URLs do app ao projeto principal em blog_project/urls.py:

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

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

4.2 Views

No arquivo views.py do app blog:

```
from django.shortcuts import render, get_object_or_404
from django.http import JsonResponse
from .models import Post, Comment
from .forms import CommentForm

def post_list(request):
    posts = Post.objects.all().order_by('-created_at')
    return render(request, 'blog/post_list.html', {'posts': posts})

def post_detail(request, id):
    post = get_object_or_404(Post, id=id)
    comments = post.comments.filter(is_approved=True)
    form = CommentForm()
    return render(request, 'blog/post_detail.html', {'post': post, 'comments': comments, 'form':
form})

def add_comment(request, id):
    if request.method == 'POST':
        post = get_object_or_404(Post, id=id)
        form = CommentForm(request.POST)
        if form.is_valid():
            comment = form.save(commit=False)
            comment.post = post
            comment.save()
            return JsonResponse({'message': 'Comment submitted for moderation.'}, status=200)
    return JsonResponse({'message': 'Invalid request.'}, status=400)
```

---

5. Formulários

Crie o arquivo forms.py e defina o formulário de comentários:

```
from django import forms
from .models import Comment

class CommentForm(forms.ModelForm):
    class Meta:
        model = Comment
        fields = ['name', 'email', 'content']
```

---

6. Templates

Crie a estrutura de pastas para templates (blog/templates/blog/) e adicione os seguintes arquivos:

6.1 Template para Listagem de Posts

post_list.html:

```
{% for post in posts %}
<div class="card mb-3">
    <img src="{{ post.image.url }}" class="card-img-top" alt="{{ post.title }}">
    <div class="card-body">
        <h5 class="card-title">{{ post.title }}</h5>
        <p class="card-text">{{ post.content|truncatewords:30 }}</p>
        <a href="{% url 'post_detail' post.id %}" class="btn btn-primary">Leia mais</a>
    </div>
</div>
{% endfor %}
```

6.2 Template para Detalhes de Post

post_detail.html:

```
<div class="card mb-3">
    <img src="{{ post.image.url }}" class="card-img-top" alt="{{ post.title }}">
    <div class="card-body">
        <h5 class="card-title">{{ post.title }}</h5>
        <p class="card-text">{{ post.content }}</p>
    </div>
</div>

<h3>Comentários</h3>
{% for comment in comments %}
<div class="alert alert-secondary">
    <strong>{{ comment.name }}</strong>: {{ comment.content }}
</div>
{% endfor %}

<h4>Deixe um comentário</h4>
```

```
<form id="comment-form">
  {{ form.as_p }}
  <button type="submit" class="btn btn-primary">Enviar</button>
</form>

<script>
  $('#comment-form').on('submit', function(e) {
    e.preventDefault();
    $.ajax({
      type: 'POST',
      url: "{% url 'add_comment' post.id %}",
      data: $(this).serialize(),
      success: function(response) {
        alert(response.message);
        location.reload();
      },
      error: function() {
        alert('Erro ao enviar o comentário.');
      }
    });
  });
</script>
```

---

7. Frontend: Bootstrap e AJAX

Configure o Bootstrap no projeto adicionando o link no base.html:

```
<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>
```
