Criação de Autenticação de Usuário com Painéis Administrativos em Django

Introdução

Nesta apostila, vamos aprender a criar uma autenticação de usuários em um projeto Django e direcionar os usuários para painéis administrativos distintos com base no tipo de usuário (Administrador ou Cliente). Abordaremos também como criar os formulários de login e registro, e as respectivas páginas de painel administrativo usando HTML e Bootstrap.

1. Estrutura do Projeto

Antes de começarmos a implementação, vamos garantir que temos a estrutura básica do projeto Django configurada corretamente.

1. Crie um novo projeto Django:

   ```
   django-admin startproject painel_administrativo
   cd painel_administrativo
   ```

2. Crie uma aplicação para gerenciar os usuários:

   ```
   python manage.py startapp usuarios
   ```

3. Adicione a aplicação no arquivo settings.py:

   ```
   INSTALLED_APPS = [
      # outras apps
      'usuarios',
   ]
   ```

4. Instale o Bootstrap no projeto para facilitar o design:

   - Você pode incluir o Bootstrap através de uma CDN no arquivo base HTML, como mostrado a seguir, ou utulizar o código abaixo para isntalar no projeto

pip install django-bootstrap4

e no installed_apps

INSTALLED_APPS = [# outras apps... 'bootstrap4',]

no HTML do template basta chamar {% load bootstrap4 %}

---

2. Criação do Modelo de Usuário

Para diferenciar entre um administrador e um cliente, usaremos o campo is_staff para identificar se o usuário é administrador, além de criar um modelo customizado se necessário.

No arquivo models.py da aplicação usuarios, crie o modelo de usuário:

```
from django.db import models
from django.contrib.auth.models import AbstractUser

class Usuario(AbstractUser):
    # Campo para o telefone do usuário (opcional)
    telefone = models.CharField(max_length=15, blank=True, null=True)

    # Campo para o endereço completo (opcional)
    endereco = models.CharField(max_length=255, blank=True, null=True)

    # Campo para data de nascimento (opcional)
    data_nascimento = models.DateField(null=True, blank=True)

    # Campo para armazenar a foto do perfil (opcional)
    foto = models.ImageField(upload_to='fotos_usuarios/', null=True, blank=True)

    # Campo para o tipo de usuário, por exemplo, cliente ou administrador
    tipo_usuario = models.CharField(
        max_length=20, choices=[('admin', 'Administrador'), ('cliente', 'Cliente')],
        default='cliente'
    )

    # Campos de autenticação já inclusos com o AbstractUser: username, email, password, etc.
    # A classe AbstractUser já fornece os seguintes campos:
    # - username
    # - email
    # - password
    # - first_name (primeiro nome)
    # - last_name (sobrenome)
    # - is_active (se o usuário está ativo)
    # - is_staff (se o usuário é staff)
    # - is_superuser (se o usuário é superusuário)
    # - date_joined (data de criação do usuário)
    # - last_login (último login)

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

Em seguida, no arquivo settings.py, defina o modelo de usuário customizado:

```
AUTH_USER_MODEL = 'usuarios.Usuario'
```

---

3. Formulários de Login e Registro

Crie os formulários para login e registro no arquivo forms.py da aplicação usuarios:

Formulário de Login:

```
from django import forms
from django.contrib.auth.forms import AuthenticationForm

class LoginForm(AuthenticationForm):
    username = forms.CharField(widget=forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Nome de Usuário'}))
    password = forms.CharField(widget=forms.PasswordInput(attrs={'class': 'form-control', 'placeholder': 'Senha'}))
```

Formulário de Registro:

```
from django import forms
from django.contrib.auth.forms import UserCreationForm
from .models import Usuario

class RegistroForm(UserCreationForm):
    # Incluindo os campos extras
    telefone = forms.CharField(max_length=15, required=False, label="Telefone")
    endereco = forms.CharField(max_length=255, required=False, label="Endereço")
    data_nascimento = forms.DateField(required=False, label="Data de Nascimento", widget=forms.TextInput(attrs={'type': 'date'}))
    foto = forms.ImageField(required=False, label="Foto")
    tipo_usuario = forms.ChoiceField(
        choices=[('admin', 'Administrador'), ('cliente', 'Cliente')],
        required=True, label="Tipo de Usuário"
    )

    class Meta:
        model = Usuario
        fields = ['username', 'email', 'password1', 'password2', 'telefone', 'endereco', 'data_nascimento', 'foto', 'tipo_usuario']
```

---

4. Criação das Views

Agora, vamos criar as views de login, registro e redirecionamento de acordo com o tipo de usuário (Administrador ou Cliente).

View de Login:

No arquivo views.py, crie a view de login:

```
from django.shortcuts import render, redirect
from django.contrib.auth import login, authenticate

def login_view(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            # Redireciona para o painel correspondente ao tipo de usuário
            if user.tipo_usuario == 'admin':
                return redirect('admin_panel')  # Painel de administrador
            else:
                return redirect('client_panel')  # Painel de cliente
        else:
            # Se a autenticação falhar, redireciona de volta para a página de login
            return redirect('login')  # Redirecionar de volta para a página de login
    return render(request, 'usuarios/login.html')
```

View de Registro:

Para a view de registro, crie a seguinte função:

```
from django.shortcuts import render, redirect
from django.contrib.auth import login
from .forms import RegistroForm
```

```
def registro_view(request):
    if request.method == 'POST':
        form = RegistroForm(request.POST, request.FILES)
        if form.is_valid():
            usuario = form.save()
            login(request, usuario)  # Faz login após o registro
            # Redireciona para o painel dependendo do tipo de usuário
            if usuario.tipo_usuario == 'admin':
                return redirect('admin_panel')  # Direciona para o painel de administrador
            else:
                return redirect('client_panel')  # Direciona para o painel de cliente
    else:
        form = RegistroForm()
    return render(request, 'usuarios/registro.html', {'form': form})
```

---

5. URLs

Adicione as rotas para login, registro e painéis administrativos no arquivo urls.py da aplicação usuarios:

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

urlpatterns = [
    path('login/', views.login_view, name='login'),
    path('registro/', views.registro_view, name='registro'),
    path('admin/', views.admin_panel, name='admin_panel'),
    path('cliente/', views.client_panel, name='client_panel'),
]
```

---

6. Painéis Administrativos

Crie as views para os painéis administrativos de administrador e cliente. Estas views podem ser simples para fins de demonstração.

Painel Administrativo para Administradores:

```
def admin_panel(request):
    return render(request, 'usuarios/admin_panel.html')
```

Painel para Clientes:

```
def client_panel(request):
    return render(request, 'usuarios/client_panel.html')
```

---

7. Templates

Crie os templates para os formulários de login, registro e painéis administrativos.

Template de Login (login.html):

```
{% load static %}
```

```
{% load bootstrap4 %}
<!-- login.html -->
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Login</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
<div class="container">
<h2 class="mt-5">Login</h2>
<form method="POST">
{% csrf_token %}
<div class="mb-3">
<label for="username" class="form-label">Nome de Usuário</label>
<input type="text" class="form-control" id="username" name="username" required>
</div>
<div class="mb-3">
<label for="password" class="form-label">Senha</label>
<input type="password" class="form-control" id="password" name="password" required>
</div>
<button type="submit" class="btn btn-primary">Entrar</button>
</form>
<p class="mt-3">Não tem uma conta? <a href="{% url 'registro' %}">Cadastre-se</a></p>
</div>
</body>
</html>
```

Template de Registro (registro.html):

```
{% load static %}
{% load bootstrap4 %}
<!-- registro.html -->
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Registro</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/
bootstrap.min.css" rel="stylesheet">
</head>
<body>
<div class="container">
<h2 class="mt-5">Cadastro de Usuário</h2>
<form method="POST" enctype="multipart/form-data">
{% csrf_token %}
```

```
<div class="mb-3">
<label for="username" class="form-label">Nome de Usuário</label>
{{ form.username }}
</div>

<div class="mb-3">
<label for="email" class="form-label">Email</label>
{{ form.email }}
</div>

<div class="mb-3">
<label for="password1" class="form-label">Senha</label>
{{ form.password1 }}
</div>

<div class="mb-3">
<label for="password2" class="form-label">Confirmação da Senha</label>
{{ form.password2 }}
</div>

<div class="mb-3">
<label for="telefone" class="form-label">Telefone</label>
{{ form.telefone }}
</div>

<div class="mb-3">
<label for="endereco" class="form-label">Endereço</label>
{{ form.endereco }}
</div>

<div class="mb-3">
<label for="data_nascimento" class="form-label">Data de Nascimento</label>
{{ form.data_nascimento }}
</div>

<div class="mb-3">
<label for="foto" class="form-label">Foto de Perfil</label>
{{ form.foto }}
</div>

<div class="mb-3">
<label for="tipo_usuario" class="form-label">Tipo de Usuário</label>
{{ form.tipo_usuario }}
</div>

<button type="submit" class="btn btn-primary">Registrar</button>
</form>
<p class="mt-3">Já tem uma conta? <a href="{% url 'login' %}">Faça login</a></p>
</div>
</body>
</html>
```

Template de Painel Administrativo para Administrador (admin_panel.html):

```
{% load static %}
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Painel Administrativo</title>
        <link    href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
  <div class="container">
    <h2 class="mt-5">Bem-vindo ao Painel Administrativo</h2>
    <p>Aqui você pode gerenciar o sistema.</p>
  </div>
</body>
</html>
```

Template de Painel para Cliente (client_panel.html):

```
{% load static %}
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Painel do Cliente</title>
        <link    href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
  <div class="container">
    <h2 class="mt-5">Bem-vindo ao Painel do Cliente</h2>
    <p>Aqui você pode acessar suas informações.</p>
  </div>
</body>
</html>
```

---

8. Conclusão

Agora, você tem um sistema de autenticação em Django com redirecionamento para painéis administrativos diferentes, dependendo do tipo de usuário. O painel administrativo para o administrador pode ser expandido para incluir funcionalidades de gerenciamento, enquanto o painel do cliente pode ser destinado à visualização de informações pessoais.