AUTENTICAÇÃO COM DJANGO

Criar um sistema de autenticação completo em Django envolve várias etapas, como configurar o ambiente, criar modelos de usuário, lidar com login, logout, registro e proteger páginas com restrições de acesso. Vamos ver como fazer isso de forma didática.

Passo 1: Configuração do Projeto Django

1. **Criar o Projeto Django**: Se você ainda não criou o projeto, inicie criando um com o comando:

   bash
   Copiar código
   django-admin startproject meu_projeto
   cd meu_projeto
   python manage.py startapp autenticacao

2. **Configuração no settings.py**: Certifique-se de que django.contrib.auth e django.contrib.contenttypes estejam em INSTALLED_APPS. Estes pacotes fornecem a base para o sistema de autenticação:

   python
   Copiar código
   INSTALLED_APPS = [
   'django.contrib.auth',
   'django.contrib.contenttypes',
   'autenticacao',
   # outros apps
   ]

3. **Migrar as Tabelas de Autenticação**: O Django já vem com tabelas de autenticação (para usuários, permissões, etc.). Execute o comando abaixo para criar as tabelas no banco de dados:

   bash
   Copiar código
   python manage.py migrate

Passo 2: Criar Views de Login e Logout

Django fornece views prontas para login e logout, que você pode personalizar e utilizar.

1. **Configurar a Rota de Login**: No arquivo urls.py do app autenticacao, adicione as rotas para login e logout.

   python
   Copiar código
   from django.urls import path
   from django.contrib.auth import views as auth_views

```
urlpatterns = [
    path('login/', auth_views.LoginView.as_view(template_name='login.html'),
name='login'),
    path('logout/', auth_views.LogoutView.as_view(), name='logout'),
]
```

- **LoginView**: Responsável por exibir e processar o login.
- **LogoutView**: Faz o logout do usuário, encerrando a sessão.

2. **Criar o Template de Login**: Crie o arquivo templates/login.html para renderizar o formulário de login. No Django, você pode usar a template language para exibir o formulário.

```
html
Copiar código
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <title>Login</title>
</head>
<body>
    <h2>Login</h2>
    <form method="POST">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Entrar</button>
    </form>
</body>
</html>
```

O formulário {{ form.as_p }} é gerado automaticamente com os campos username e password.

Passo 3: Criar View de Registro de Usuário

Agora vamos criar uma view para permitir que novos usuários se registrem no sistema.

1. **View de Registro**: Crie uma view para o registro de novos usuários. No arquivo views.py do app autenticacao, adicione a lógica para processar o formulário de registro.

```
python
Copiar código
from django.shortcuts import render, redirect
from django.contrib.auth.forms import UserCreationForm

def register(request):
    if request.method == 'POST':
        form = UserCreationForm(request.POST)
        if form.is_valid():
            form.save()
```

```
            return redirect('login')  # Redireciona para a página de login após o registro
    else:
        form = UserCreationForm()
    return render(request, 'register.html', {'form': form})
```

2. **Criar o Template de Registro**: Crie o template templates/register.html para exibir o formulário de registro.

   ```
   html
   Copiar código
   <!DOCTYPE html>
   <html lang="pt-br">
   <head>
       <meta charset="UTF-8">
       <title>Registrar-se</title>
   </head>
   <body>
       <h2>Registrar-se</h2>
       <form method="POST">
           {% csrf_token %}
           {{ form.as_p }}
           <button type="submit">Registrar</button>
       </form>
   </body>
   </html>
   ```

3. **Adicionar a Rota de Registro**: No arquivo urls.py, adicione a rota para o registro.

   ```
   python
   Copiar código
   from .views import register

   urlpatterns = [
       path('register/', register, name='register'),
       path('login/', auth_views.LoginView.as_view(template_name='login.html'),
   name='login'),
       path('logout/', auth_views.LogoutView.as_view(), name='logout'),
   ]
   ```

Passo 4: Protegendo Páginas com Autenticação

Agora que você tem login e registro funcionando, vamos proteger algumas páginas para que apenas usuários autenticados possam acessá-las.

1. **Criar uma View Protegida**: Crie uma view que só poderá ser acessada por usuários logados. Use o decorador @login_required para garantir isso.

   ```
   python
   Copiar código
   from django.contrib.auth.decorators import login_required
   from django.shortcuts import render
   ```

```
@login_required
def dashboard(request):
    return render(request, 'dashboard.html')
```

2. **Adicionar a Rota de Dashboard**: No arquivo urls.py, adicione a rota para essa página protegida.

   python
   Copiar código
   ```
   urlpatterns = [
       path('dashboard/', dashboard, name='dashboard'),
       path('login/', auth_views.LoginView.as_view(template_name='login.html'),
   name='login'),
       path('logout/', auth_views.LogoutView.as_view(), name='logout'),
   ]
   ```

3. **Template de Dashboard**: Crie o template templates/dashboard.html para renderizar essa página.

   html
   Copiar código
   ```
   <!DOCTYPE html>
   <html lang="pt-br">
   <head>
       <meta charset="UTF-8">
       <title>Painel</title>
   </head>
   <body>
       <h2>Bem-vindo ao Painel, {{ user.username }}</h2>
       <p>Esta página está protegida e só pode ser acessada por usuários
   logados.</p>
   </body>
   </html>
   ```

4. **Redirecionar Usuários para o Login**: Se um usuário não estiver logado e tentar acessar a página dashboard, ele será redirecionado para a página de login. Para definir a página de redirecionamento após o login ou logout, configure as URLs no settings.py.

   python
   Copiar código
   ```
   LOGIN_REDIRECT_URL = 'dashboard'
   LOGOUT_REDIRECT_URL = 'login'
   LOGIN_URL = 'login'  # Se o usuário tentar acessar uma página protegida sem
   estar logado
   ```

Passo 5: Testar o Sistema de Autenticação

Agora que tudo está configurado, você pode testar o sistema de autenticação. Aqui está o fluxo completo:

1. O usuário acessa /register/ para se registrar.
2. Após o registro, é redirecionado para a página de login.
3. O usuário faz login em /login/.
4. Uma vez logado, ele é redirecionado para o painel /dashboard/, que está protegido e só pode ser acessado por usuários autenticados.
5. O usuário pode fazer logout em /logout/.

Extras: Personalizar Formulários de Autenticação

Se quiser personalizar os formulários, como adicionar novos campos ao registro ou mudar a aparência do formulário de login, você pode criar seus próprios formulários.

Exemplo de Personalização de Formulário de Registro:

python
Copiar código

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

class CustomUserCreationForm(UserCreationForm):
    email = forms.EmailField(required=True)

    class Meta:
        model = User
        fields = ['username', 'email', 'password1', 'password2']
```

Esse formulário personalizado pode ser usado na view de registro.