

django

AULA 07

AUTENTICAÇÃO E CONTROLE DE ACESSO

O QUE VEREMOS HOJE

- 01 AUTENTICAÇÃO NO DJANGO
- 02 COMPONENTES DE AUTENTICAÇÃO
- 03 REGISTRO DE USUÁRIOS
- 04 LOGIN/LOGOUT
- 04 CONTROLE DE ACESSO

AUTENTICAÇÃO NO DJANGO

O Django oferece um sistema completo de autenticação, integrado ao framework, que facilita a implementação de funcionalidades essenciais, como login, logout, registro de usuários e redefinição de senhas.

Ele também permite o controle de acesso a páginas e recursos específicos do projeto, utilizando permissões personalizáveis e grupos de usuários.



COMPONENTES DE AUTENTICAÇÃO

O Django fornece várias ferramentas para gerenciar autenticação de forma eficiente e segura. Alguns dos principais componentes são:

- **User Model:** já vem integrado ao Django e representa as informações dos usuários no banco de dados, como nome, email e senha.
- **Views e URLs de Autenticação:** O Django possui views prontas para login, logout e alteração de senha.
- **Decoradores:** Decoradores como `@login_required` ajudam a proteger views, garantindo que apenas usuários autenticados tenham acesso.
- **Forms de Login e Registro:** O Django oferece formulários prontos, como `AuthenticationForm` e `UserCreationForm`, para criar e autenticar usuários.


CONFIGURAÇÃO INICIAL

Para usar o sistema de autenticação do Django, é necessário realizar algumas configurações iniciais no arquivo `settings.py`.

Certifique-se de que os seguintes aplicativos estão incluídos no projeto:

- **`django.contrib.auth`**: Gerencia usuários, autenticação e permissões.
- **`django.contrib.sessions`**: Controla sessões de usuários autenticados.

Além disso, é necessário configurar o redirecionamento padrão para login e logout.



```
1 LOGIN_REDIRECT_URL = '/'  
2 LOGOUT_REDIRECT_URL = '/login/'
```

REGISTRO DE USUÁRIOS

Para criar a funcionalidade de registro de usuários no Django vamos precisar dos seguintes componentes:

- View
- URL
- Template

Não é necessário o formulário porque o Django já fornece um formulário seguindo a tabela de usuários.

```
1 from django.contrib.auth.forms import UserCreationForm
2 from django.shortcuts import render
3
4 def register(request):
5     if request.method == 'POST':
6         form = UserCreationForm(request.POST)
7         if form.is_valid():
8             form.save()
9         form = UserCreationForm()
10    else:
11        form = UserCreationForm()
12    return render(request, 'registration/register.html', {'form': form})
13
```

REGISTRO DE USUÁRIOS

No template, vamos precisar criar o formulário de forma muito parecida com o cadastro de dados no banco de dados.

Esse template também será no projeto.

E o formulário que será usado é o UserCreationForm do Django que já inclui os campos básicos para registro.

```
1  <body>
2      <div>
3          <h2>Registrar-se</h2>
4          <form method="post">
5              {% csrf_token %}
6              {{ form }}
7              <button type="submit">Registrar</button>
8          </form>
9      </div>
10 </body>
```

REGISTRO DE USUÁRIOS

Por fim, precisamos confirmar a URL de registro no arquivo `urls.py` do projeto.



```
1 from django.urls import path
2 from .views import register
3
4
5 urlpatterns = [
6     path('registro/', register, name='register'),
7 ]
8
```


LOGIN E LOGOUT DO USUÁRIO

O Django fornece views prontas para login e logout, que podem ser configuradas facilmente no arquivo `urls.py`.

- `LoginView`: Renderiza um formulário de login e autentica o usuário.
- `LogoutView`: Encerra a sessão do usuário e o redireciona para a URL configurada em `LOGOUT_REDIRECT_URL`.

```
1 from django.contrib.auth import views as auth_views
2 from django.urls import path
3 from .views import register
4
5
6 urlpatterns = [
7     path('registro/', register, name='register'),
8     path('login/', auth_views.LoginView.as_view(), name='login'),
9     path('logout/', auth_views.LogoutView.as_view(), name='logout'),
10 ]
11
12
```

TEMPLATE DE LOGIN

O Django espera que o template de login esteja localizado em **templates/registration/login.html**.

O template pode ser personalizado para incluir o layout do projeto.

O formulário no template segue a mesma lógica dos demais formulários.



```
1 <div>
2     <h2>Login</h2>
3     <form method="post">
4         {% csrf_token %}
5         {{ form }}
6         <button type="submit">Entrar</button>
7     </form>
8 </div>
```

LOGOUT

O logout não vai exigir template, pois executa apenas a ação de sair do sistema.

Mas ainda é necessário colocar na aplicação a opção de logout. Isso pode ser feito através da tag `<form>`.

```
1  <body>
2    <div>
3      <h2>Eventos</h2>
4      <ul>
5        {% for event in events %}
6          <li>
7            <strong>{{ event.title }}</strong>
8            <p>{{ event.description }}</p>
9            <p>{{ event.date }}</p>
10         </li>
11       {% endfor %}
12     </ul>
13   </div>
14   <div>
15     <form action="{% url 'logout' %}" method="post">
16       {% csrf_token %}
17       <button type="submit">Sair</button>
18     </form>
19   </div>
20 </body>
```

ATIVIDADE PRÁTICA

No App de Tarefas implemente as funcionalidades de registro, login e logout de usuários.

Crie um app separado para colocar essas funcionalidades.

CONTROLE DE ACESSO

Uma vez implementado a funcionalidade de autenticação, podemos implementar o controle de acesso.


Esse controle vai servir para proteger o acesso a páginas quando o usuário não estiver logado.



CONTROLE DE ACESSO

Para impedir o acesso a páginas sem o login, é necessário usar o decorador `@login_required` do Django para proteger a view.

Esse decorador vai servir para verificar se o usuário está logado. Caso esteja, a view será chamada normalmente. E caso não esteja, ele será redirecionado para url do `LOGIN_REDIRECT_URL` no `settings.py`.



```
1 from django.contrib.auth.decorators import login_required
2 from django.shortcuts import render
3 from .models import Event
4
5 @login_required
6 def list_events(request):
7     events = Event.objects.all()
8     return render(request, 'list.html', {'events': events})
9
```

PERSONALIZAÇÃO DAS PÁGINAS

Além do controle de acesso, o Django disponibiliza a personalização das páginas para exibir conteúdos diferentes para o usuário autenticado e o não autenticado.

Nessa opção não é necessário usar o `@login_required`, pois o próprio conteúdo da página vai ser alterado conforme o login.

```
1 <body>
2     {% if user.is_authenticated %}
3         <p>Bem-vindo, {{ user.username }}!</p>
4         <form action="{% url 'logout' %}" method="post">
5             {% csrf_token %}
6             <button type="submit">Sair</button>
7         </form>
8     {% else %}
9         <p>Bem-vindo! Faça login para acessar mais funcionalidades.</p>
10        <a href="{% url 'login' %}">Fazer Login</a>
11    {% endif %}
12 </body>
```

ATIVIDADE PRÁTICA

No App de Tarefas escolha uma das views e adicione o controle de acesso usando o decorador `@login_required` e verifique se o está redirecionando para página de login.

E em outra view, personalize o conteúdo da página para quando o usuário não estiver logado aparecer uma mensagem incentivando o login.