

## Aula 02

### 1. Usuário

#### 1.1. Criando o Aplicativo e os Arquivos

Para que um usuário possa enviar fotos para o site, é necessário que ele seja capaz de se cadastrar e entrar na plataforma. Para gerenciar isso, vamos criar um aplicativo chamado de "usuario".

Seguem os passos:

1. criar o aplicativo;
2. registrar no `/picasa/settings.py`;

```
C:\silver-enigma\picasa\settings.py
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     # meus apps
41     'galeria.apps.GaleriaConfig',
42     'usuario.apps.UsuarioConfig',
43 ]
```

3. criar os arquivos:
  - a. `/usuario/templates/usuario/paginas/login.html`;

```
C:\silver-enigma\usuario\templates\usuario\paginas\login.html
1 {% extends 'usuario/paginas/base.html' %}
2
3 {% block conteudo %}
4 <h1>pagina login.html</h1>
5 <section>
6     <form action="" method="">
7         <div>
8             <label for="nome_login"><b>Nome de Login</b></label>
9             <input type="text" name="nome_login" placeholder="Ex.: Tom Cruise" required>
10         </div>
11         <div>
12             <label for="senha"><b>Senha</b></label>
13             <input type="password" name="senha" placeholder="sua senha" required>
14         </div>
15         <div>
16             <button type="submit">Login</button>
17         </div>
18     </form>
19 </section>
20 {% endblock conteudo %}
21
```

- b. `/usuario/templates/usuario/paginas/cadastro.html`;

```
C:\silver-enigma\usuario\templates\usuario\paginas\cadastro.html
1 {% extends 'base/paginas/base.html' %}
2
3 {% block conteudo %}
4 <h1>pagina login.html</h1>
5 <section>
6     <form action="" method="">
7         <div>
8             <label for="nome_cadastro"><b>Nome completo</b></label>
9             <input type="text" name="nome_cadastro" placeholder="Ex.: Tom Cruise" required>
10         </div>
11         <div>
12             <label for="email"><b>Email</b></label>
13             <input type="email" name="email" placeholder="Ex.: tomcruise@com.com" required>
14         </div>
15         <div>
16             <label for="senha1"><b>Senha</b></label>
17             <input type="password" name="senha1" placeholder="Digite sua senha" required>
18         </div>
19         <div>
20             <label for="senha2"><b>Confirmação de senha</b></label>
21             <input type="password" name="senha2" placeholder="Digite novamente" required>
22         </div>
23         <div>
24             <button type="submit">Criar sua conta</button>
25         </div>
26     </form>
27 </section>
28 {% endblock conteudo %}
```

- criar as funções login e cadastro no arquivo **/usuario/views.py** e chamar seus respectivos arquivos;

```
C:\silver-enigma\usuario\views.py
1 from django.shortcuts import render
2
3 def view_login(request):
4     return render(request, 'usuario/paginas/login.html')
5
6 def view_cadastro(request):
7     return render(request, 'usuario/paginas/cadastro.html')
```

- criar o arquivo **/usuario/urls.py** e chamar as views login e cadastro;

```
C:\silver-enigma\usuario\urls.py
1 from django.urls import path
2 from usuario.views import view_login, view_cadastro
3
4 app_name = 'usuario'
5
6 urlpatterns = [
7     path('login/', view_login, name='login'),
8     path('cadastro', view_cadastro, name='cadastro'),
9 ]
```

- registrar a urls do novo arquivo urls.py no **/picasa/urls.py**;

```
C:\silver-enigma\picasa\urls.py
17 from django.contrib import admin
18 from django.urls import path, include
19 from django.conf import settings
20 from django.conf.urls.static import static
21
22 urlpatterns = [
23     path('admin/', admin.site.urls),
24     path('galeria/', include('galeria.urls')),
25     path('usuario/', include('usuario.urls')),
26 ]
27
28 urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
29
```

Uma vez que todo o processo acima esteja feito, já podemos criar um arquivo ainda não trabalhado.

## 1.2. Arquivo form.py

A classe que será criado nesse arquivo ela tem uma funcionalidade especial. É através dela que montaremos nosso formulário HTML. Dessa forma, se quisermos alterar algo em nosso formulário, basta alterarmos nessa classe. Veja como ela vai ficar:

```
C:\silver-enigma\usuario\forms.py
1 from django import forms
2
3 class LoginForms(forms.Form):
4     nome_login = forms.CharField(
5         label = 'Nome de Login',
6         required = True,
7         max_length = 100,
8         widget = forms.TextInput(
9             attrs = {
10                 'placeholder': 'Ex.: Tom Cruise',
11             }
12         )
13     )
14     senha = forms.CharField(
15         label = 'Senha',
16         required = True,
17         max_length = 70,
18         widget = forms.PasswordInput(
19             attrs = {
20                 'placeholder': 'sua senha',
21             }
22         )
23     )
```

Repare nos campos dele.

O campo nome\_login vai ser do tipo CharField e terá como parâmetros:

- label, que vai ser usado para mostrar o texto ao lado do campo;
- required, para indicar que ele é obrigatório;
- max\_length, para indicar o tamanho máximo de caracteres permitidos;
- widget, com o tipo TextInput indica que aquele campo é do tipo texto e, dentro dele, é usado o atributo attrs para receber um dicionário com os parâmetros de um campo input; nesse caso, está sendo chamado apenas o placeholder, para adicionar um texto modelo ao campo;

Já o campo senha também será do tipo CharField, mas com alguns parâmetros a mais:

- widget, com o tipo PasswordInput indica que aquele campo é do tipo senha, portanto tem que ficar mascarado ao ser digitado qualquer coisa; ele também tem um placeholder definido;

### 1.3. Arquivo views.py

Agora, já podemos atualizar nossa função na views.py. Ela vai criar uma instância do formulário e enviar para a página /usuario/templates/usuario/paginas/login.html.

Veja como ficará:

```
C:\silver-enigma\usuario\views.py
1 from django.shortcuts import render
2 from usuario.forms import LoginForms
3
4 def view_login(request):
5     formulario = LoginForms()
6     return render(request, 'usuario/paginas/login.html', context={'formulario':formulario})
7
8 def view_cadastro(request):
9     return render(request, 'usuario/paginas/cadastro.html')
```

### 1.4. Arquivo login.html

Agora que nossa página HTML está recebendo a variável para criar o formulário, temos que alterar ela para exibir ele, no lugar do antigo.

Veja como vai ficar:

```
C:\silver-enigma\usuario\templates\usuario\paginas\login.html
1 {% extends 'usuario/paginas/base.html' %}
2
3 {% block conteudo %}
4 <h1>pagina login.html</h1>
5 <section>
6     <form action="{% url 'usuario:login' %}" method="POST">
7         {% csrf_token %}
8         {% for campo in formulario.visible_fields %}
9             <div>
10                 <label for="{{ campo.id_for_label }}"><b>{{ campo.label }}</b></label>
11                 {{ campo }}
12             </div>
13         {% endfor %}
14         <div>
15             <button type="submit">Login</button>
16         </div>
17     </form>
18 </section>
19 {% endblock conteudo %}
```

Repare que agora está sendo usada a tag Django csrf\_token. Ela é necessária para que se possa enviar um formulário usando o método POST.

Depois, é usado um for para passar por todos os visible\_fields do formulário e mostrar eles. Dessa forma, se quisermos que nosso formulário tenha mais campos, basta adicionarmos eles à classe LoginForms.

## Atividade

1. Desenvolva a página de cadastro usando como referência a de login.
2. Continue os trabalhos em cima do projeto final.