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

```python
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    # meus apps
    'galeria.apps.GaleriaConfig',
    'usuario.apps.UsuarioConfig',
]
```

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

C:\silver-enigma\usuario\templates\usuario\paginas\login.html

```html
{% extends 'usuario/paginas/base.html' %}

{% block conteudo %}
<h1>pagina login.html</h1>
<section>
    <form action="" method="">
        <div>
            <label for="nome_login"><b>Nome de Login</b></label>
            <input type="text"  name="nome_login" placeholder="Ex.: Tom Cruise" required>
        </div>
        <div>
            <label for="senha"><b>Senha</b></label>
            <input type="password" name="senha" placeholder="sua senha" required>
        </div>
        <div>
            <button type="submit">Login</button>
        </div>
    </form>
</section>
{% endblock conteudo %}

```

    b. **/usuario/templates/usuario/paginas/cadastro.html**;

C:\silver-enigma\usuario\templates\usuario\paginas\cadastro.html

```html
{% extends 'base/paginas/base.html' %}

{% block conteudo %}
<h1>pagina login.html</h1>
<section>
    <form action="" method="">
        <div>
            <label for="nome_cadastro"><b>Nome completo</b></label>
            <input type="text" name="nome_cadastro" placeholder="Ex.: Tom Cruise" required>
        </div>
        <div>
            <label for="email"><b>Email</b></label>
            <input type="email" name="email" placeholder="Ex.: tomcruise@com.com" required>
        </div>
        <div>
            <label for="senha1"><b>Senha</b></label>
            <input type="password" name="senha1" placeholder="Digite sua senha" required>
        </div>
        <div>
            <label for="senha2"><b>Confirmação de senha</b></label>
            <input type="password" name="senha2" placeholder="Digite novamente" required>
        </div>
        <div>
            <button type="submit">Criar sua conta</button>
        </div>
    </form>
</section>
{% endblock conteudo %}
```

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

```
C:\silver-enigma\usuario\views.py
from django.shortcuts import render

def view_login(request):
    return render(request, 'usuario/paginas/login.html')

def view_cadastro(request):
    return render(request, 'usuario/paginas/cadastro.html')
```

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

```
C:\silver-enigma\usuario\urls.py
from django.urls import path
from usuario.views import view_login, view_cadastro

app_name = 'usuario'

urlpatterns = [
    path('login/', view_login, name='login'),
    path('cadastro', view_cadastro, name='cadastro'),
]
```

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

```
C:\silver-enigma\picasa\urls.py
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('galeria/', include('galeria.urls')),
    path('usuario/', include('usuario.urls')),
]

urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

```

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
from django import forms

class LoginForms(forms.Form):
    nome_login = forms.CharField(
        label = 'Nome de Login',
        required = True,
        max_length = 100,
        widget = forms.TextInput(
            attrs = {
                'placeholder': 'Ex.: Tom Cruise',
            }
        )
    )
    senha = forms.CharField(
        label = 'Senha',
        required = True,
        max_length = 70,
        widget = forms.PasswordInput(
            attrs = {
                'placeholder': 'sua senha',
            }
        )
    )
```

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

```python
from django.shortcuts import render
from usuario.forms import LoginForms

def view_login(request):
    formulario = LoginForms()
    return render(request, 'usuario/paginas/login.html', context={'formulario':formulario})

def view_cadastro(request):
    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

```
{% extends 'usuario/paginas/base.html' %}

{% block conteudo %}
<h1>pagina login.html</h1>
<section>
    <form action="{% url 'usuario:login' %}" method="POST">
        {% csrf_token %}
        {% for campo in formulario.visible_fields %}
        <div>
            <label for="{{ campo.id_for_label }}"><b>{{ campo.label }}</b></label>
            {{ campo }}
        </div>
        {% endfor %}
        <div>
            <button type="submit">Login</button>
        </div>
    </form>
</section>
{% 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.