Django Registro de Usuário

Prof. Bruno Gomes

@profbrunogomes

Aula de Hoje

- Django
  - Registro de Usuário

1.

Registro de Usuário

Usuário nativo do Django

Documentação

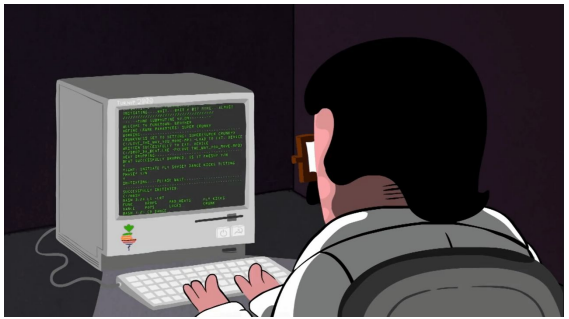
- https://docs.djangoproject.com/en/4.1/topics/auth/default/

Antes de começar

- Se não tiver o código do projeto até o momento, baixar **proj_10_registro_1.zip** da pasta do Drive;

- Extrair no mesmo diretório do Virtualenv, e digitar os comandos:
  - python manage.py migrate
  - python manage.py createsuperuser

- Iniciar servidor e testar os endereços:
  - http://localhost:8000/



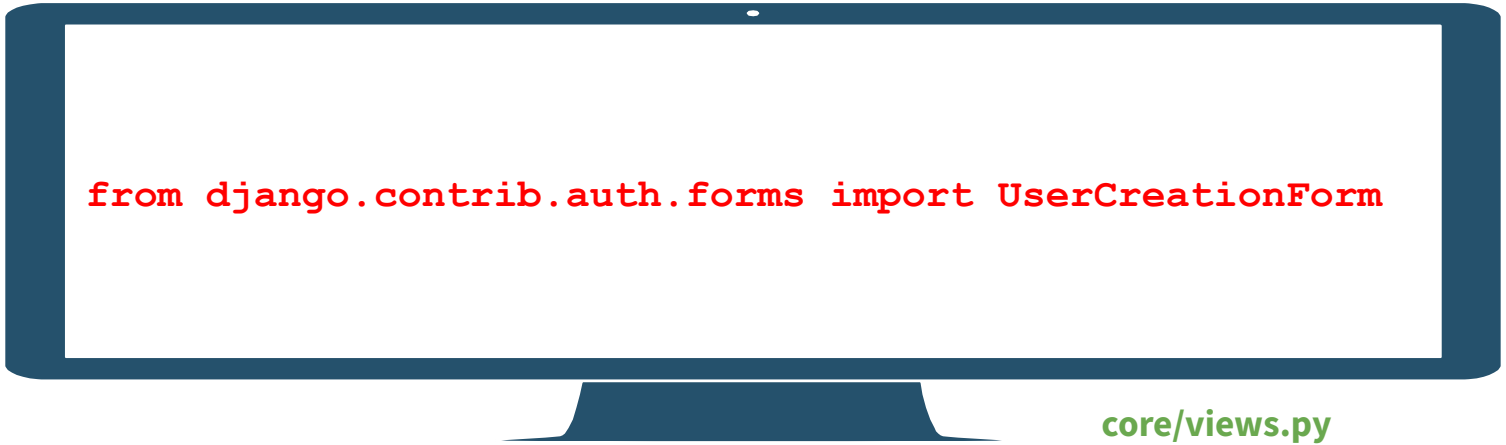
Preparação: Criar a URL para o Registro

» Em **urls.py**, criar a url registro:

proj_10_registro/urls.py

Preparação: View para Registro

» Em views.py, importar as bibliotecas **redirect** e **UserCreationForm**:



core/views.py

Explicação

Atenção


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

- **UserCreationForm**: é um ModelForm que já vem implementado no Django, com 3 campos para o registro de usuário: **username**, **password1** e **password2**.

View para Registro

» Da mesma forma que o cadastro foi criado nas aulas anteriores, deve ser implementado em **views.py** registro utilizando a classe UserCreationForm.

```
def registro(request):
    form = UserCreationForm(request.POST or None)
    if form.is_valid():
        form.save()
        return redirect('login')
    contexto = {
      'form': form
    }
    return render(request, 'registro.html', contexto)
```

core/views.py

Observação

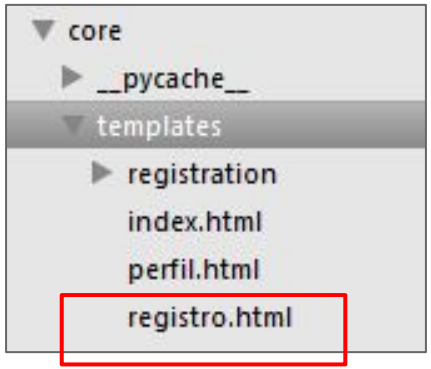
Atenção

» Ao realizar o registro, automaticamente o usuário será redirecionado para a página de login:

```
if form.is_valid():
    form.save()
    return redirect('login')
```

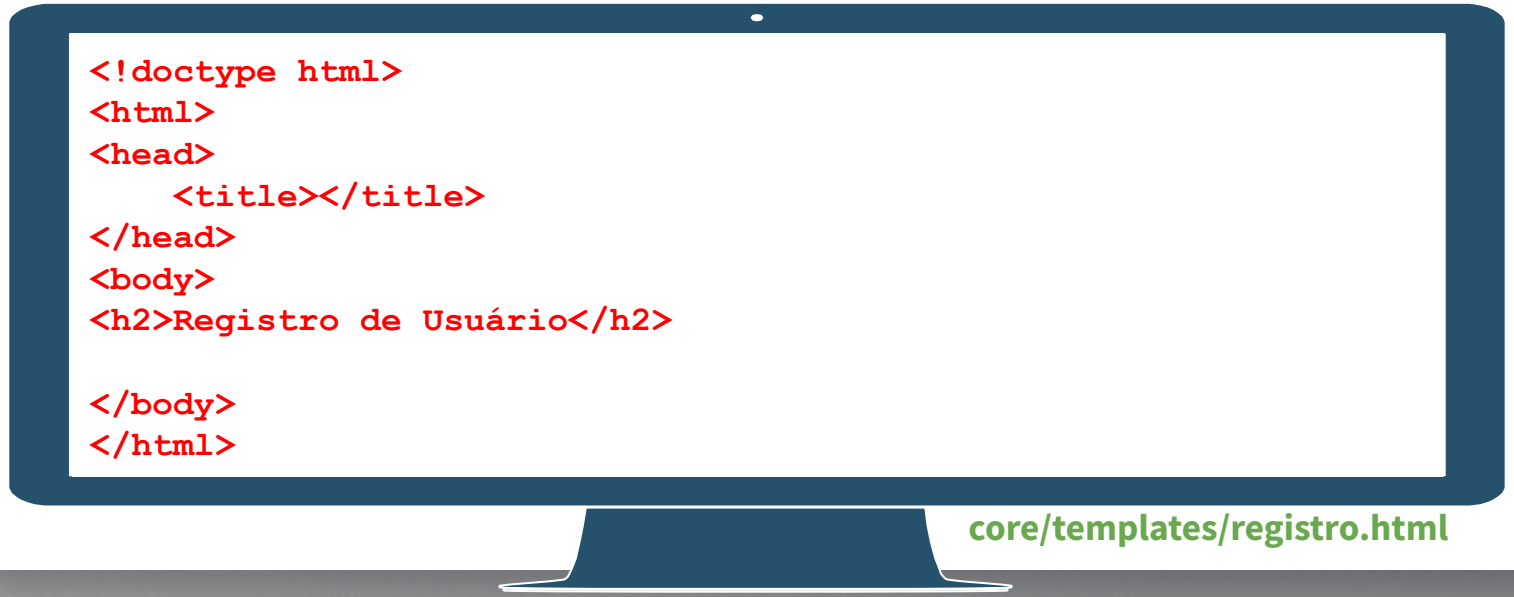
Criar o template registro

» Na pasta templates, criar o template registro.html:



Criar o template registro

» No template **registro.html**, adicionar um HTML básico:



```
<!doctype html>
<html>
<head>
    <title></title>
</head>
<body>
<h2>Registro de Usuário</h2>

</body>
</html>
```

core/templates/registro.html

Criar template Registro

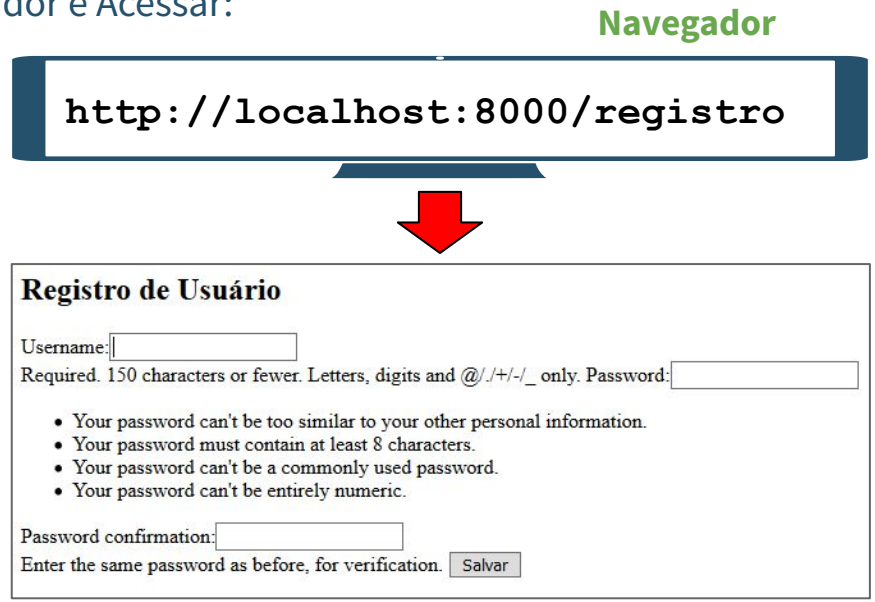
» Ainda no template **registro.html**, adicionar um formulário, e as tags de csrf e do formulário (form):

```
<h2>Registro de Usuário</h2>
<form method="post">
{% csrf_token %}
{{form}}
<input type="submit" value="Cadastrar">
</form>
```

core/templates/registro.html

Testar no Navegador

» Iniciar o servidor e Acessar:



Melhorando a aparência do Formulário

» Ao invés de gerar totalmente o formulário, em **registro.html** substituir {{form}} por:

```
{% csrf_token %}

<p>Usuário: {{ form.username }}</p>
<p>Senha: {{ form.password1 }}</p>
<p>Repetir Senha: {{ form.password2 }}</p>
```

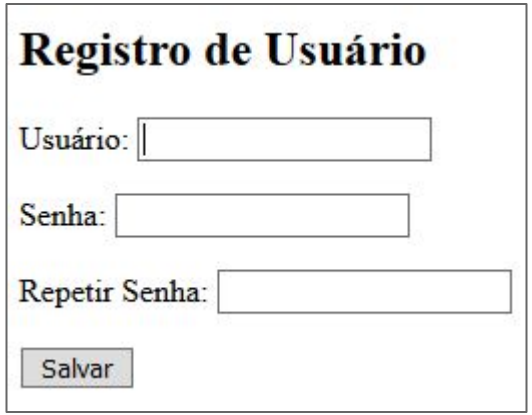
core/templates/registro.html

Testar no Navegador

» Iniciar o servidor e Acessar:

Navegador

`http://localhost:8000/registro`



Link para Cadastro

- No template **index.html**, adicionar um link para o cadastro:
  - Dentro da condição de que o usuário não está cadastrado:

```
{% if not user.is_authenticated %}
<p><a href="{% url 'login' %}">Login</a></p>
<p><a href="{% url 'registro' %}">Cadastro</a></p>
{% endif %}
```

core/templates/index.html

Testar no Navegador

» Iniciar o servidor e Acessar com o usuário não autenticado:

Navegador

http://localhost:8000/

2.

Registro de Usuário

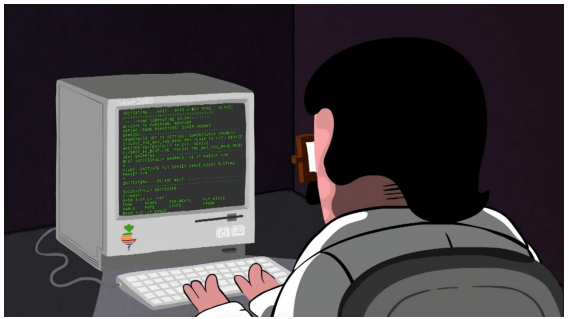
Usuário Definido

Antes de começar

- Se não tiver o código do projeto até o momento, baixar **proj_10_registro_2.zip** da pasta do Drive;

- Extrair no mesmo diretório do Virtualenv, e digitar os comandos:
  - python manage.py migrate
  - python manage.py createsuperuser

- Iniciar servidor e testar os endereços:
  - http://localhost:8000/



Criar forms.py

» Na pasta **core**, criar o arquivo forms.py e adicionar:

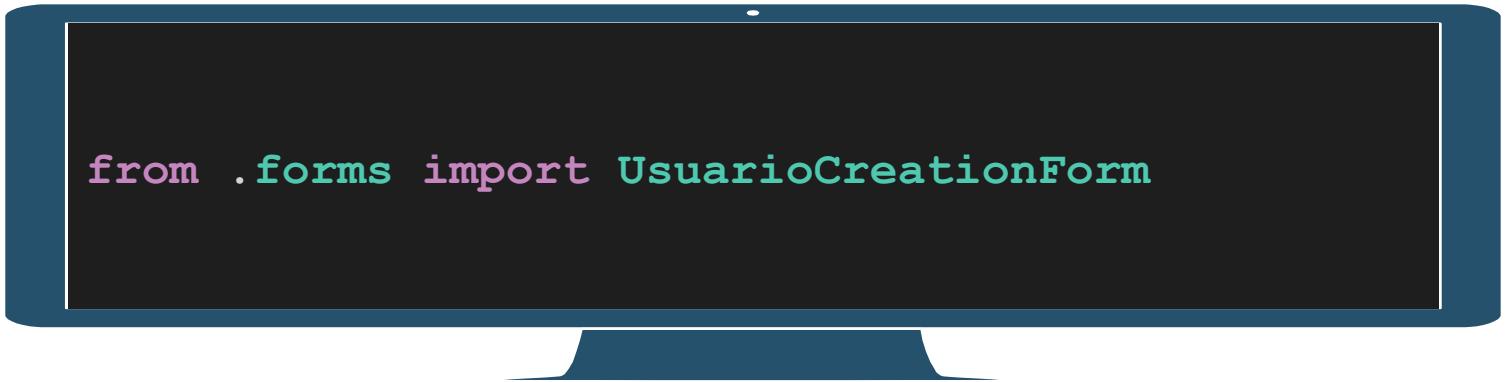
```
from django import forms
from django.contrib.auth.forms import UserCreationForm
from .models import Usuario

class UsuarioCreationForm(UserCreationForm):
    class Meta:
        model = Usuario
        fields = ['username','password1','password2','cpf','email','nome','idade']
```

core/forms.py

Views

» Importar a classe do form criada:



core/views.py

Views

» Implementar o registro usando a classe importada:

```
def registro(request):
    form = UsuarioCreationForm(request.POST or None)
    if form.is_valid():
          form.save()
          return redirect('login')
    contexto = {
        'form': form
    }
    return render(request, 'registro.html', contexto)
```

core/views.py

Criar a rota para a view

» Em **urls.py**, criar a url registro:

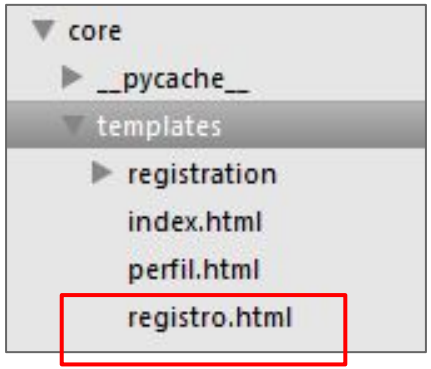
```
from core.views import home, perfil, registro

path('registro/', registro, name='registro'),
```

proj_10_registro/urls.py

Criar o template registro

» Na pasta templates, criar o template **registro.html:**



Criar o template registro

» No template **registro.html**, adicionar um HTML básico:

```
<!doctype html>
<html>
<head>
    <title></title>
</head>
<body>
<h2>Registro de Usuário</h2>

</body>
</html>
```

core/templates/registro.html

Criar template Registro

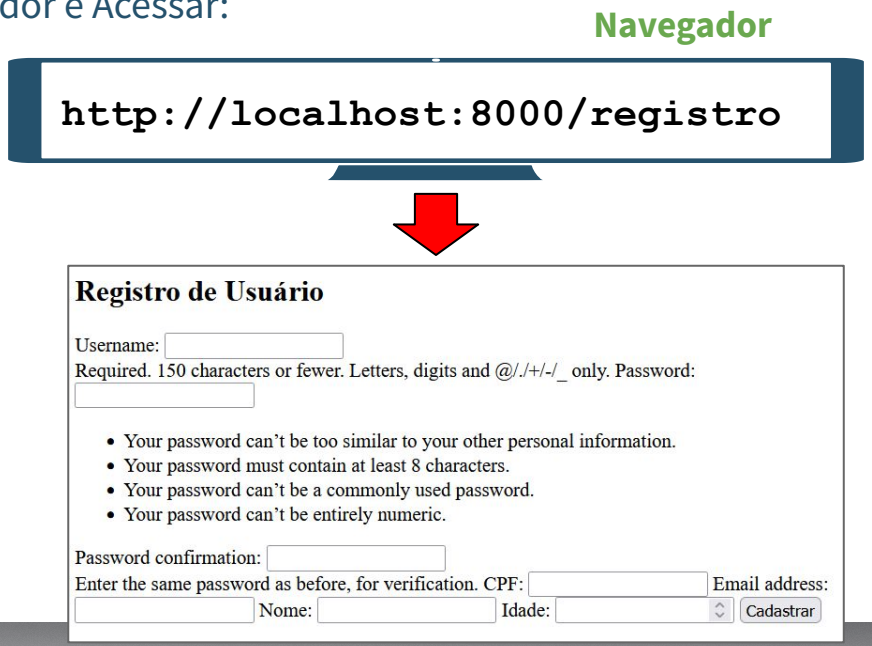
» Ainda no template **registro.html**, adicionar um formulário, e as tags de csrf e do formulário (form):

```
<h2>Registro de Usuário</h2>
<form method="post">
{% csrf_token %}
{{form}}
<input type="submit" value="Cadastrar">
</form>
```

core/templates/registro.html

Testar no Navegador

» Iniciar o servidor e Acessar:



Melhorando a aparência do Formulário

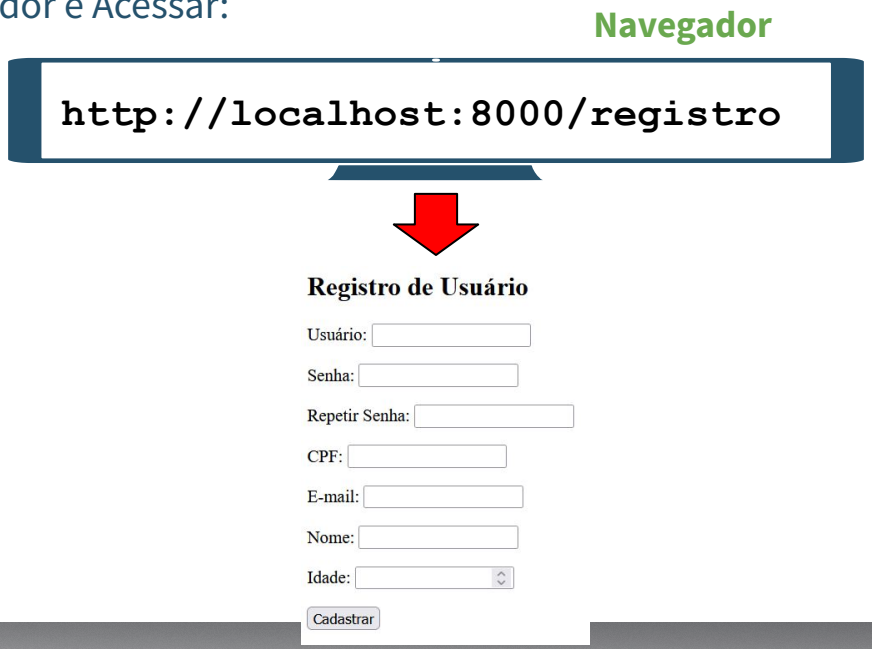
» Ao invés de gerar totalmente o formulário, em **registro.html** substituir {{form}} por:

```
{% csrf_token %}
<p>Usuário: {{ form.username }}</p>
<p>Senha: {{ form.password1 }}</p>
<p>Repetir Senha: {{ form.password2 }}</p>
<p>CPF: {{ form.cpf }}</p>
<p>E-mail: {{ form.email }}</p>
<p>Nome: {{ form.nome }}</p>
<p>Idade: {{ form.idade }}</p>
<input type="submit" value="Cadastrar">
</form>
```

core/templates/registro.html

Testar no Navegador

» Iniciar o servidor e Acessar:



3.

Validação de Senha

Documentação

- https://docs.djangoproject.com/en/4.1/topics/auth/passwords/
- https://docs.djangoproject.com/en/4.1/topics/auth/passwords/#password-validation

Retirando a Validação de Senha

» Em settings, adicionar:

```
AUTH_PASSWORD_VALIDATORS = []
```

proj_10_registro/settings.py

Tipos de Validações

- https://docs.djangoproject.com/en/4.1/topics/auth/passwords/#enabling-password-validation

4.

Edição de Dados

Exibir e Editar Dados do Usuário

» Preparação: em **urls.py**, criar a roda dados (recebendo o parâmetro id do usuário):

```
from core.views import index, perfil, registro, dados

path('dados/<int:id>/', dados, name='dados'),
```

aula9_2/urls.py

Exibir e Editar Dados do Usuário

» Em views.py, implementar a função dados que exibirá os dados do usuário em um formulário (primeiro acesso), e também poderá atualizar os dados (caso ele faça alguma alteração e salve):

core/views.py

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

@login_required
def dados(request, id):
    user = User.objects.get(pk=id)
    form = UserCreationForm(request.POST or None, instance=user)
    if form.is_valid():
       form.save()
       return redirect('perfil')
    contexto = {
       'form': form
    }
    return render(request, 'registro.html', contexto)
```

Criar o link em index

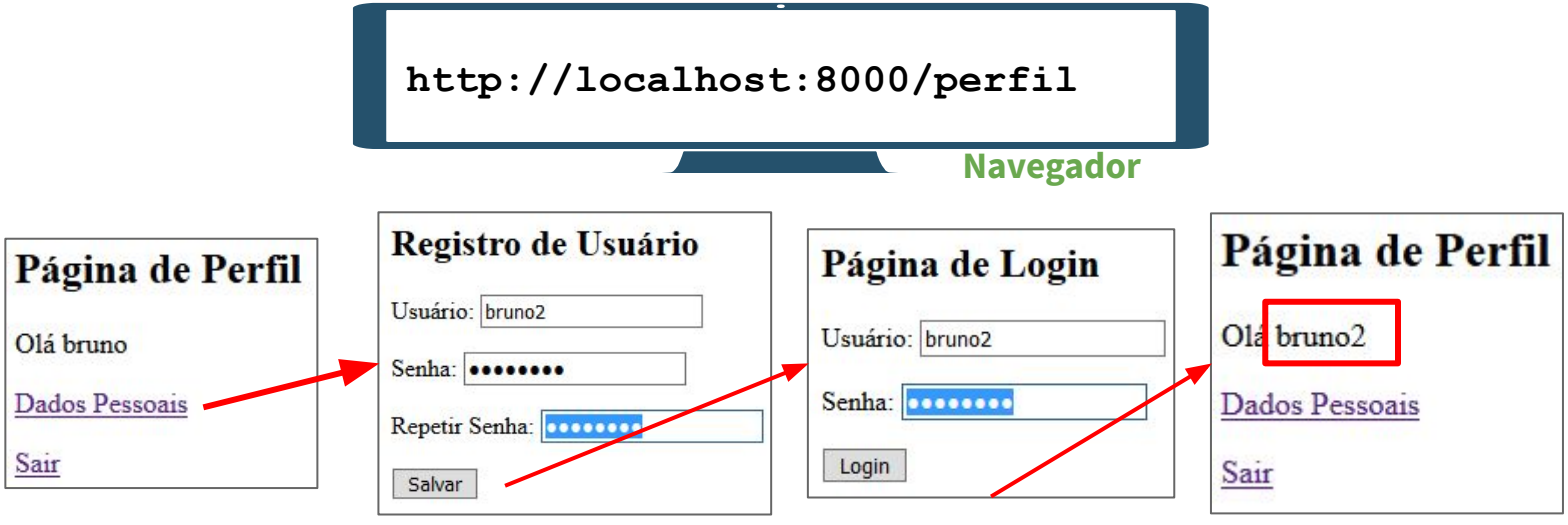
» No template **perfil.html**, adicionar um link para a URL dados:

```
<p>Olá {{ user.username }}</p>
<p><a href="{% url 'dados' user.id %}">Dados
Pessoais</a></p>
```

core/templates/perfil.html

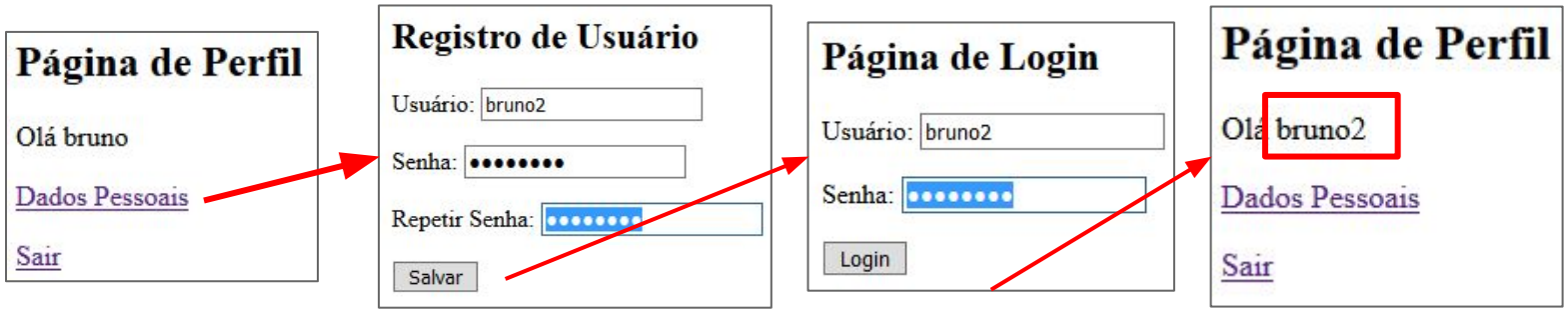
Testar no Navegador

» Iniciar o servidor e Acessar a página de perfil de um usuário autenticado:



Situação

Atenção



- Ao realizar alteração nos dados do usuário, será redirecionado à tela de Login, obrigatoriamente, para que confirme os novos dados e acesse novamente perfil.

Dúvidas?