Fuctura

Desenvolvimento WEB com Django

Módulo 2 - Aula 4

Prof. Jéssica Andrade

Bora Revisar?

DJANGO

1°) Oque é um Model?

2°) Para que serve o construtor :
def __str__(*self*):
**   *return self*.nome?**

3°) Oque é um CharField?

4°) Oque é um DecimalField?

5°) Para que serve um ForeignKey ?

DJANGO

6°) O que é um DateTimeField?

7°) Para que serve o max_length=100?

8°) Para que serve o max_digits=10?

9°) Para que serve o on_delete=models.CASCADE?

10°) Qual o comando para subir um servidor no django?

11°) Como se derruba este servidor?

DJANGO

Sobre os exercícios propostos nas ultimas aulas, sigo no aguardo do link do github de vocês, lembrando que a partir de agora, todos os exercícios serão aceitos apenas via link do github.

Qual o link do linkedin de vocês? Vamos criar conexões?

DJANGO

Oque vamos fazer hoje?

Na aula passada desenvolvemos toda a parte de criação de modelos e tabelas do banco de dados.
Nesta aula vamos continuar com nosso projeto, criando nossas views, que serão como controladores para que assim possamos criar regras de negocio nos nossos apps.
Mas qual a view mais importante e pela qual devemos começar?

Autenticação é o mais importante em um sistema, ele cuida da segurança de quem acessa e de quem tem seus dados acessados.

Na pratica, por que se autenticar é tão importante?

DJANGO

CSRF (Cross-Site Request Forgery) é um tipo de ataque que engana um usuário autenticado em um site para que ele execute ações indesejadas em outro site onde está autenticado. O ataque explora a confiança que um site tem no navegador do usuário.

- **Autenticação**: O usuário faz login em um site (por exemplo, um banco online) e recebe um cookie de sessão que o mantém autenticado.

- **Visita a um Site Malicioso**: Enquanto está autenticado, o usuário visita um site malicioso que contém um código (geralmente JavaScript) que faz uma requisição para o site do banco.

- **Execução da Ação**: O navegador do usuário envia a requisição para o site do banco, incluindo o cookie de sessão. O banco, acreditando que a requisição é legítima, executa a ação (como transferir dinheiro) sem o conhecimento do usuário.

DJANGO


2 O invasor incorpora a solicitação em uma hiperligação e envia-a aos visitantes quem pode estar logado no site
Visitante do site
3 Um visitante clica no link, inadvertidamente envio do pedido para o sítio Web
4 Website valida pedidos e transfere fundos da conta do visitante ao atacante
Atacante
Sítio Web
1 Invasor forja um pedido de transferência de fundos para o site
Neuralegião | www.neuralegion.com

Oque o django nos da pra resolver esse problema?

Uma validação pronta para o csrf!

Podem existir outras validações importantes, mas esta é a que iremos englobar nesta aula!

Vamos la, vamos criar nossas primeiras views

DJANGO

Passo 1: Abra o projeto dentro da sua virtualenv como aprendido anteriormente em aula. Deve ser executado no cmd(windows), no terminal linux ou no bash do git

Windows

- O comando abaixo vai depender do seu windows
- cd venv/Scripts

1. source activate ou activate ou ./activate

- Este pode ser executado em qualquer versão

2. cd ../..

3. cd barbearia_pdv

Linux

1. source venv/bin/activate

3. cd barbearia_pdv

Tudo Inicializado?

Agora vamos criar nossas views de autenticação

DJANGO

- **HTTP (Hypertext Transfer Protocol)** é o protocolo usado para a comunicação na web. Ele define como as mensagens são formatadas e transmitidas entre um cliente (como um navegador) e um servidor (onde os sites estão hospedados).

- Os **métodos HTTP** indicam a ação que o cliente deseja realizar em um recurso no servidor. Aqui estão os métodos mais comuns:
- https://blog.grancursosonline.com.br/arquitetura-rest-metodos-de-requisicoes-http/ (Boa Explicação aprofundada sobre oque é HTTP)

– **GET**:

Descrição: Solicita dados de um servidor.
Uso: Usado para buscar informações, como carregar uma página da web.
Exemplo: Quando você digita uma URL no navegador, ele faz uma requisição GET para obter a página.

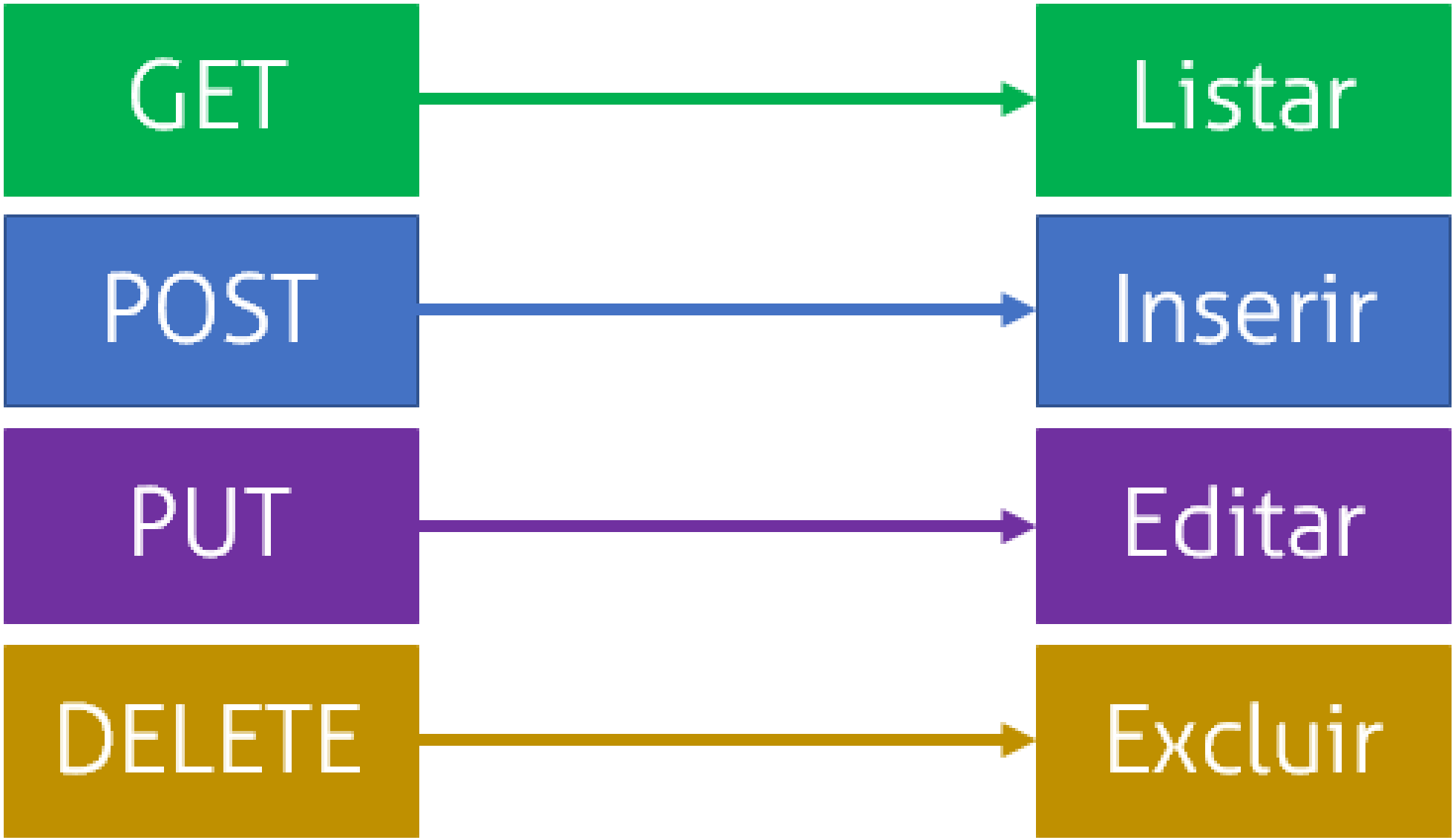
DJANGO

- **POST**:
  Descrição: Envia dados para o servidor.
  Uso: Usado para enviar informações, como ao preencher um formulário (ex: registro ou login).
  Exemplo: Quando você envia um formulário de registro, o navegador faz uma requisição POST com os dados do formulário.

- **PUT**:
  Descrição: Atualiza um recurso existente no servidor.
  Uso: Usado para modificar dados, como atualizar informações de um usuário.
  Exemplo: Se você editar seu perfil em um site, uma requisição PUT pode ser enviada para atualizar suas informações.

DJANGO

- **DELETE**:
  Descrição: Remove um recurso do servidor.
  Uso: Usado para deletar dados, como excluir uma conta ou um item.
  Exemplo: Quando você clica em "Excluir" em um post, uma requisição DELETE pode ser enviada.


GET
Listar
POST
Inserir
PUT
Editar
DELETE
Excluir

Códigos de Status HTTP

- Os códigos de status HTTP são respostas do servidor que indicam o resultado de uma requisição. Aqui estão alguns dos códigos mais comuns:

– **200 OK**:
  Descrição: A requisição foi bem-sucedida e o servidor retornou os dados solicitados.

– **201 Created**:
  Descrição: A requisição foi bem-sucedida e um novo recurso foi criado (geralmente usado com POST).

– **500 Internal Server Error**:
  Descrição: Ocorreu um erro no servidor ao processar a requisição.

- **400 Bad Request**:
  Descrição: A requisição não pôde ser entendida pelo servidor devido a um erro do cliente (ex: dados inválidos).

- **401 Unauthorized**:
  Descrição: O cliente não está autenticado. É necessário fazer login para acessar o recurso.

- **404 Not Found**:
  Descrição: O recurso solicitado não foi encontrado no servidor (ex: uma página que não existe).

HTTP Status Codes

Level 200

200: OK
201: Created
202: Accepted
203: Non-Authoritative Information
204: No content

Level 400

400: Bad Request
401: Unauthorized
403: Forbidden
404: Not Found
409: Conflict

Level 500

500: Internal Server Error
501: Not Implemented
502: Bad Gateway
503: Service Unavailable
504: Gateway Timeout
599: Network Timeout

DJANGO

Agora, dentro do arquivo **urls.py**, que esta na pasta principal do projeto, chamada **barbearia_pdv,** valide se esta igual ao lado, e se não tiver, corrija para que fique igual.

Agora temos algo semelhante ao exemplo da foto ao lado.
Dessa forma conseguimos criar inumeras urls no nosso app filho,e ter diferentes rotas para cada app

➡ **NÃO ESQUEÇA DE SALVAR COM CTRL + S**

DJANGO

Agora, dentro do arquivo **views.py**, que esta na pasta **pdv,** vamos criar nossas views.

1° Passo: Vamos adicionar os imports necessários

views.py

p2 projeto teste > barbearia_pdv > pdv > views.py > login_view

```
1   from django.shortcuts import render, redirect
2   from django.contrib.auth import login as auth_login, authenticate, logout
3   from django.contrib.auth.models import User
4   from .models import Cliente, Servico, Agendamento
5   from .forms import ClienteForm, ServicoForm, AgendamentoForm, UserRegistrationForm
6   from django.contrib.auth.decorators import login_required
7
```

DJANGO

Agora, dentro do arquivo **views.py**, que esta na pasta **pdv,** vamos criar nossas views.

2° Passo: Criação da view de login.

```python
def login_view(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
        user = authenticate(request, username=username, password=password)
        if user is not None:
            auth_login(request, user)
            return redirect('dashboard')  # Redireciona para o dashboard após login
    return render(request, 'pdv/login.html')
```

DJANGO

Agora, dentro do arquivo **views.py**, que esta na pasta **pdv,** vamos criar nossas views.

3° Passo: Criação da view de logout.

```python
def logout_view(request):
    logout(request)
    return redirect('login')  # Redireciona para a página de login após logout
```

DJANGO

Agora, dentro do arquivo **views.py**, que esta na pasta **pdv,** vamos criar nossas views.

4° Passo: Criação da view de register (Cadastro de acesso ao sistema).

```python
def register_view(request):
    if request.method == 'POST':
        form = UserRegistrationForm(request.POST)
        if form.is_valid():
            user = form.save(commit=False)
            user.set_password(form.cleaned_data['password'])
            user.save()
            return redirect('login')
    else:
        form = UserRegistrationForm()
    return render(request, 'pdv/register.html', {'form': form})
```

DJANGO

Agora, dentro do arquivo **views.py**, que esta na pasta **pdv,** vamos criar nossas views.

5° Passo: Criação da view de dashboard

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

DJANGO

6° Passo: Criação da view de criação e listagem de clientes

```python
@login_required
def criar_cliente(request):
    if request.method == 'POST':
        form = ClienteForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('clientes')
    else:
        form = ClienteForm()
    return render(request, 'pdv/criar_cliente.html', {'form': form})

@login_required
def clientes_view(request):
    clientes = Cliente.objects.all()
    return render(request, 'pdv/clientes.html', {'clientes': clientes})
```

DJANGO

7° Passo: Criação da view de criação e listagem de agendamentos

```python
@login_required
def criar_agendamento(request):
    if request.method == 'POST':
        form = AgendamentoForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('agendamentos')
    else:
        form = AgendamentoForm()
    return render(request, 'pdv/criar_agendamento.html', {'form': form})

@login_required
def agendamentos_view(request):
    agendamentos = Agendamento.objects.all()
    return render(request, 'pdv/agendamentos.html', {'agendamentos': agendamentos})
```

DJANGO

8° Passo: Criação da view de criação de serviços

```python
@login_required
def criar_servico(request):
    if request.method == 'POST':
        form = ServicoForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('servicos')
    else:
        form = ServicoForm()
    return render(request, 'pdv/criar_servico.html', {'form': form})
```

DJANGO

7° Passo: Criação da view de Relatórios

```python
@login_required
def relatorios_view(request):
    agendamentos = Agendamento.objects.all()
    total_vendas = sum(agendamento.servico.preco for agendamento in agendamentos)
    return render(request, 'pdv/relatorios.html', {'agendamentos': agendamentos, 'total_vendas':
    total_vendas})
```

DJANGO

Explicando termos importantes sobre nossas views.

- **Método HTTP**: A view verifica se a requisição é do tipo POST para identificar se o usuário enviou um formulário de login, uma prática comum para ações que alteram o estado do servidor.

- **request.POST.get()**: Acessa os dados do formulário. Retorna **None** se a chave não existir, evitando erros.

- **authenticate**: Verifica as credenciais do usuário. Retorna um objeto **User** se válidas; caso contrário, retorna **None**.

- **auth_login**: Cria uma sessão para o usuário autenticado, permitindo que ele permaneça logado em requisições subsequentes.

DJANGO

- **redirect**: Redireciona o usuário para a página do dashboard após o login, melhorando a experiência do usuário.

- **logout**: Encerra a sessão do usuário, removendo suas informações de autenticação para evitar acesso não autorizado.

- **form.is_valid()**: Verifica se os dados do formulário são válidos, retornando **True** se forem.

- **form.save(commit=False)**: Cria uma instância do modelo sem salvá-la imediatamente, permitindo manipulação antes do salvamento.

DJANGO

- **set_password**: Define a senha de forma segura, garantindo que seja armazenada criptografada no banco de dados.

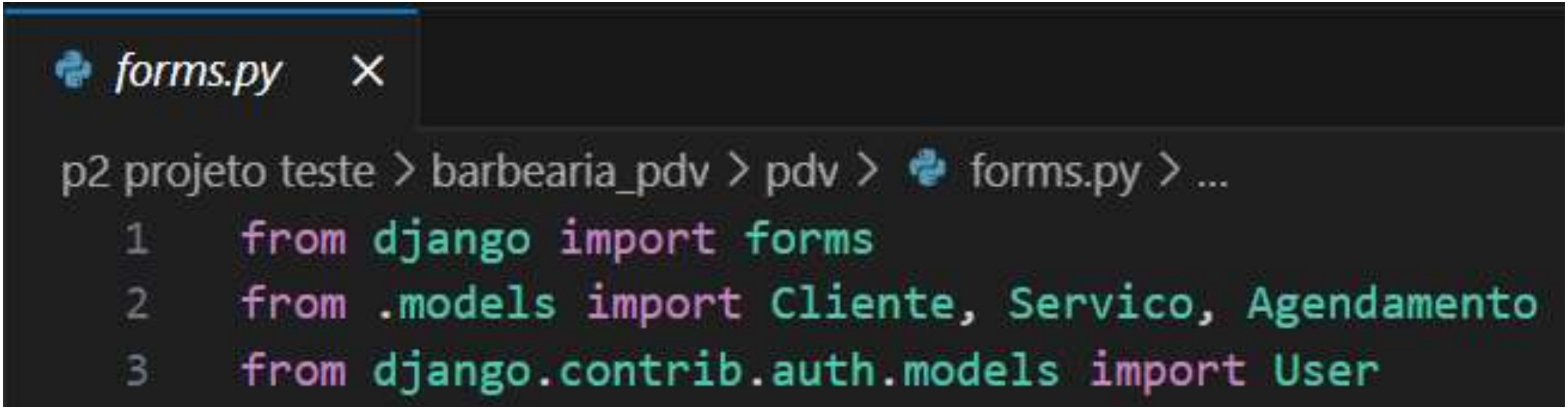
- **@login_required**: Decorador que protege a view, permitindo acesso apenas a usuários autenticados.

- **Cliente.objects.all()**: Consulta que retorna todos os registros da tabela **Cliente**, utilizando o Django ORM para facilitar a interação com o banco de dados.

- **Cálculo de Vendas**: Busca agendamentos e calcula o total de vendas somando os preços dos serviços associados.

DJANGO

Agora vamos criar nosso arquivo **forms.py**. Dentro da pasta **pdv**, crie um arquivo chamado **forms.py** e adicione os imports.



DJANGO

Agora adicione os demais forms no mesmo arquivo respectivamente

```python
class ClienteForm(forms.ModelForm):
    class Meta:
        model = Cliente
        fields = ['nome', 'telefone', 'email']

class ServicoForm(forms.ModelForm):
    class Meta:
        model = Servico
        fields = ['nome', 'preco']

class AgendamentoForm(forms.ModelForm):
    class Meta:
        model = Agendamento
        fields = ['cliente', 'servico', 'data_hora']
```

DJANGO

```python
class UserRegistrationForm(forms.ModelForm):
    password = forms.CharField(widget=forms.PasswordInput)
    password_confirm = forms.CharField(widget=forms.PasswordInput)

    class Meta:
        model = User
        fields = ['username', 'email', 'password']

    def clean(self):
        cleaned_data = super().clean()
        password = cleaned_data.get("password")
        password_confirm = cleaned_data.get("password_confirm")

        if password and password_confirm and password != password_confirm:
            raise forms.ValidationError("As senhas não coincidem.")
```

DJANGO

Explicando termos importantes sobre nossos forms.

- **forms.ModelForm**: É uma classe base que facilita a criação de formulários baseados em modelos. Ela automaticamente gera campos para os atributos do modelo.

- **class Meta**: Esta classe interna é usada para definir metadados sobre o formulário.

- **password e password_confirm**: Esses campos são definidos como **CharField** e usam um widget de entrada de senha (**PasswordInput**), que oculta a senha enquanto o usuário a digita.

- **class Meta**: Define que este formulário está associado ao modelo **User** e inclui os campos **username**, **email** e **password**.

Explicando termos importantes sobre nossas views.

- **clean**: Este método é chamado para validar os dados do formulário. Ele verifica se as senhas fornecidas (**password** e **password_confirm**) são iguais.

- **cleaned_data = super().clean()**: Chama o método **clean** da classe pai para garantir que a validação padrão do Django seja aplicada.

- **if password and password_confirm and password != password_confirm:**: Se as senhas não coincidirem, uma **ValidationError** é levantada, e uma mensagem de erro é exibida.

DJANGO

- Agora vamos adicionar nossas urls para chamar nossas views e encaminhar corretamente. Dentro da pasta **pdv,** crie **SE NÃO EXISTIR** ou edite o arquivo **urls. py,** para que fique da seguinte forma.

urls.py

p2 projeto teste > barbearia_pdv > pdv > urls.py > ...

```python
from django.urls import path
from django.contrib import admin
from .views import (
                    relatorios_view, criar_agendamento,
                    criar_cliente, criar_servico,
                    register_view, logout_view,
                    login_view,
                    dashboard, clientes_view, agendamentos_view,
                    )
```

DJANGO

Ainda no arquivo **urls. py** de dentro da pasta **pdv.**

urlpatterns: Esta é uma lista que contém todas as rotas do seu aplicativo. Cada rota é definida usando a função path.

```
urlpatterns = [
    path('login/', login_view, name='login'),
    path('logout/', logout_view, name='logout'),
    path('register/', register_view, name='register'),
    path('dashboard/', dashboard, name='dashboard'),
    path('clientes/', clientes_view, name='clientes'),
    path('clientes/criar/', criar_cliente, name='criar_cliente'),
    path('servicos/criar/', criar_servico, name='criar_servico'),
    path('agendamentos/', agendamentos_view, name='agendamentos'),
    path('agendamentos/criar/', criar_agendamento, name='criar_agendamento'),
    path('relatorios/', relatorios_view, name='relatorios'),
]
```

DJANGO

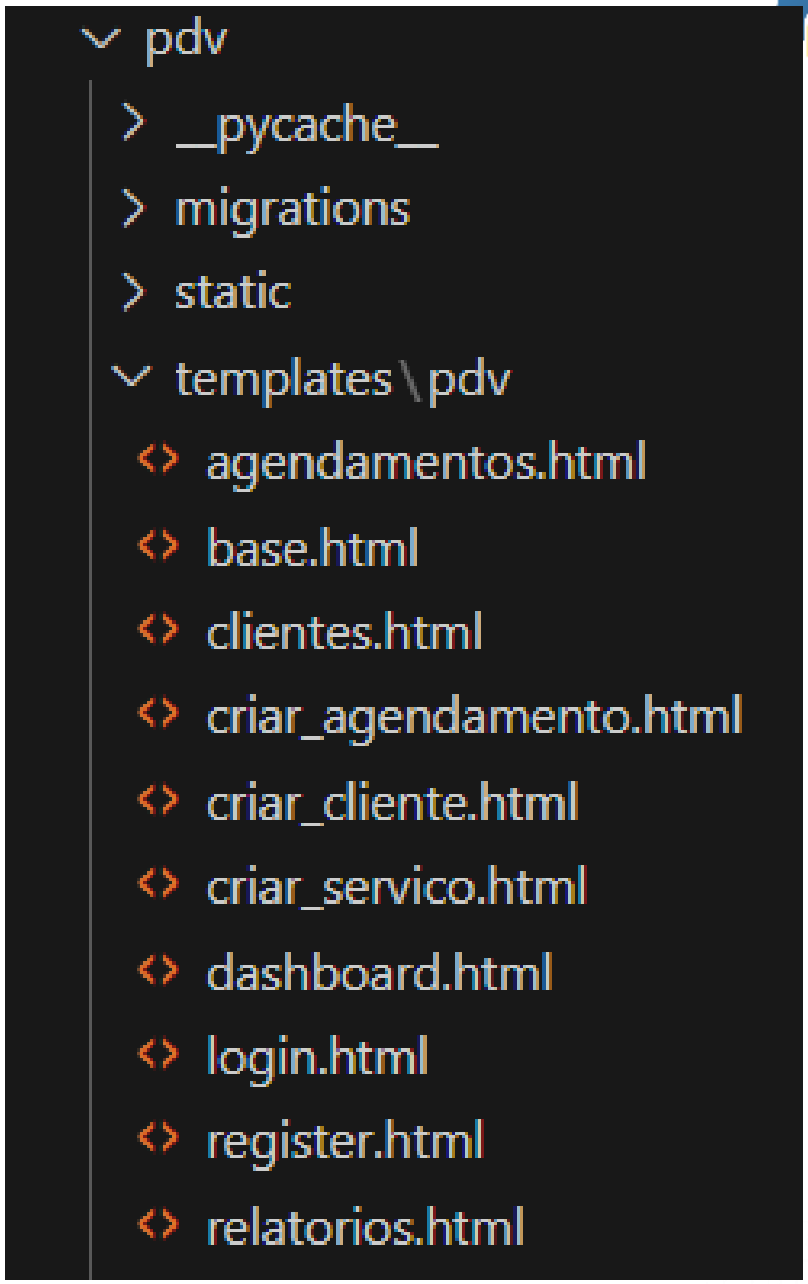
Agora vamos criar e organizar nossos templates. Lembrando que vamos apenas criar os arquivos, na proxima aula vamos realmente implementar nossos frontends.

- 1° Dentro de pdv, cria duas novas pastas, uma chamada templates, outra chamada static.

- 2° Dentro da pasta templates cria uma nova pasta chamada pdv, esta guardará os templates para nosso projeto pdv.

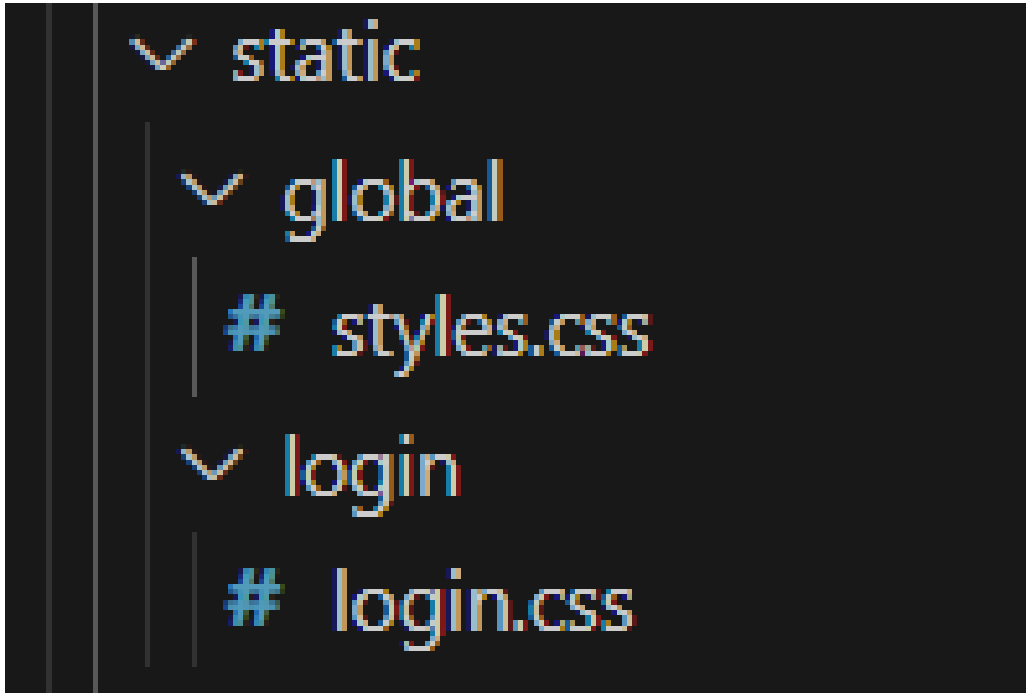
- 3° Agora crie os arquivos .html dentro do diretório pdv que esta dentro de templates, deve ser os mesmo da imagem ao lado.

Confirme se a estrutura do seu projeto esta desta forma, estes são os diretórios e arquivos esperados em pdv



DJANGO

Agora crie duas pastas dentro da pasta static, uma chamada globals, e outra chamada login, e cria os arquivos .css dentro de seus respectivos diretórios como a imagem abaixo



DJANGO

Agora dentro do nosso base.html que esta na pasta templates, vamos criar nosso template pai para os demais templates. Estrutura do nosso HTML.

- **{% load static %}**: Carrega a tag para usar arquivos estáticos do Django.
- **<link rel="stylesheet" type="text/css" href="{% static 'login/login.css' %}">**: Vincula arquivos CSS para estilização da página.
- **{% if user.is_authenticated %}**: Verifica se o usuário está autenticado; se sim, exibe a navegação.
- **<nav>**: Contém links de navegação para diferentes seções do aplicativo, usando a tag **{% url %}** para gerar URLs dinamicamente.
- **<main>**: Define a área principal do conteúdo
- **{% block content %}{% endblock %}**: Permite que outros templates herdem e substituam o conteúdo deste bloco, facilitando a reutilização de layouts.

DJANGO

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>PDV Barbearia</title>
    {% load static %}
    <link rel="stylesheet" type="text/css" href="{% static 'global/styles.css' %}">
    <link rel="stylesheet" type="text/css" href="{% static 'login/login.css' %}">
</head>
<body>
    <header>
        {% if user.is_authenticated %}
            <nav>
                <a href="{% url 'dashboard' %}">Dashboard</a>
                <a href="{% url 'clientes' %}">Clientes</a>
                <a href="{% url 'agendamentos' %}">Agendamentos</a>
                <a href="{% url 'relatorios' %}">Relatórios</a>
                <a href="{% url 'logout' %}">Logout</a>
            </nav>
        {% endif %}
    </header>
    <main>
        {% block content %}{% endblock %}
    </main>
</body>
```

Agora vamos criar o HTML de login, fique livre para costumizar a cor, adicionar logo e outras informações no login. o memso html do login criado na aula anterior. E costumize!

Use sua criatividade e pense fora da caixinha

login.html

p2 projeto teste > barbearia_pdv > pdv > templates > pdv > login.html

```
{% extends 'pdv/base.html' %}
{% block content %}

<div style="display: flex; justify-content: center; align-items: center; height: 100vh;">

    <div style="background-color: white; padding: 40px; border-radius: 8px; box-shadow: 0 2px 10px rgba(0,0,0,0.1);">
     <h2>Login</h2>
    <p>Digite os seus dados de acesso no campo abaixo.</p>
    <form method="post">
        {% csrf_token %}
        <div>
            <label for="username">Usuário:</label>
            <input type="text" id="username" name="username" required class="input-field" placeholder="Digite seu Usuário">
        </div>
        <div>
            <label for="password">Senha</label>
            <input type="password" id="password" class="input-field" name="password" placeholder="Digite sua senha" required>
        </div>
        <p>Ainda não tem uma conta? <a href="{% url 'register' %}">Registre-se aqui</a>.</p>
         <button type="submit" style="background-color: #d5006d; color: white; padding: 10px 20px; border: none;
         border-radius: 5px; cursor: pointer;">Acessar</button>
    </form>
    </div>
</div>

{% endblock %}
```

DJANGO

- Agora adicione o **css** dentro do arquivo **login.css. Livre para costumizações!!!**

login.html | # login.css

p2 projeto teste > barbearia_pdv > pdv > static > login > # login.css > h1

```css
body {
    background-color: #5e5be9; /* Cor de fundo roxa */
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
}

.container {
    background-color: white; /* Fundo da caixa de login */
    border-radius: 8px;
    box-shadow: 0 4px 15px rgba(0, 0, 0, 0.1);
    padding: 20px;
    width: 300px;
}

h1 {
    text-align: center;
    margin-bottom: 10px;
}
```

DJANGO

```css
p {
    text-align: center;
    font-size: 14px;
    margin-bottom: 20px;
}

label {
    margin-bottom: 5px;
    display: block;
    font-weight: bold;
}

input {
    width: 100%;
    padding: 10px;
    margin-bottom: 15px;
    border: 1px solid #ccc;
    border-radius: 4px;
}
```

DJANGO

```css
.forgot-password {
    display: block;
    text-align: center;
    margin-bottom: 15px;
    color: #5e5be9;
    text-decoration: none;
}

.login-button {
    width: 100%;
    padding: 10px;
    background-color: #e91e63; /* Cor do botão */
    color: white;
    border: none;
    border-radius: 4px;
    font-size: 16px;
    cursor: pointer;
}

.login-button:hover {
    background-color: #d81b60; /* Efeito hover */
}
```

DJANGO

- O comando abaixo inicia o servidor, e seve ser executado no CMD, Terminal Linux ou BASH. **Vá em frente e EXECUTE!**
- Lembre-se: **sempre** que formos testar o app, devemos rodar este comando para subir o servidor.
- **NÃO ESQUEÇA DE SALVAR OS ARQUIVOS MODIFICADOS COM CTRL + S**

- **python manage.py runserver**

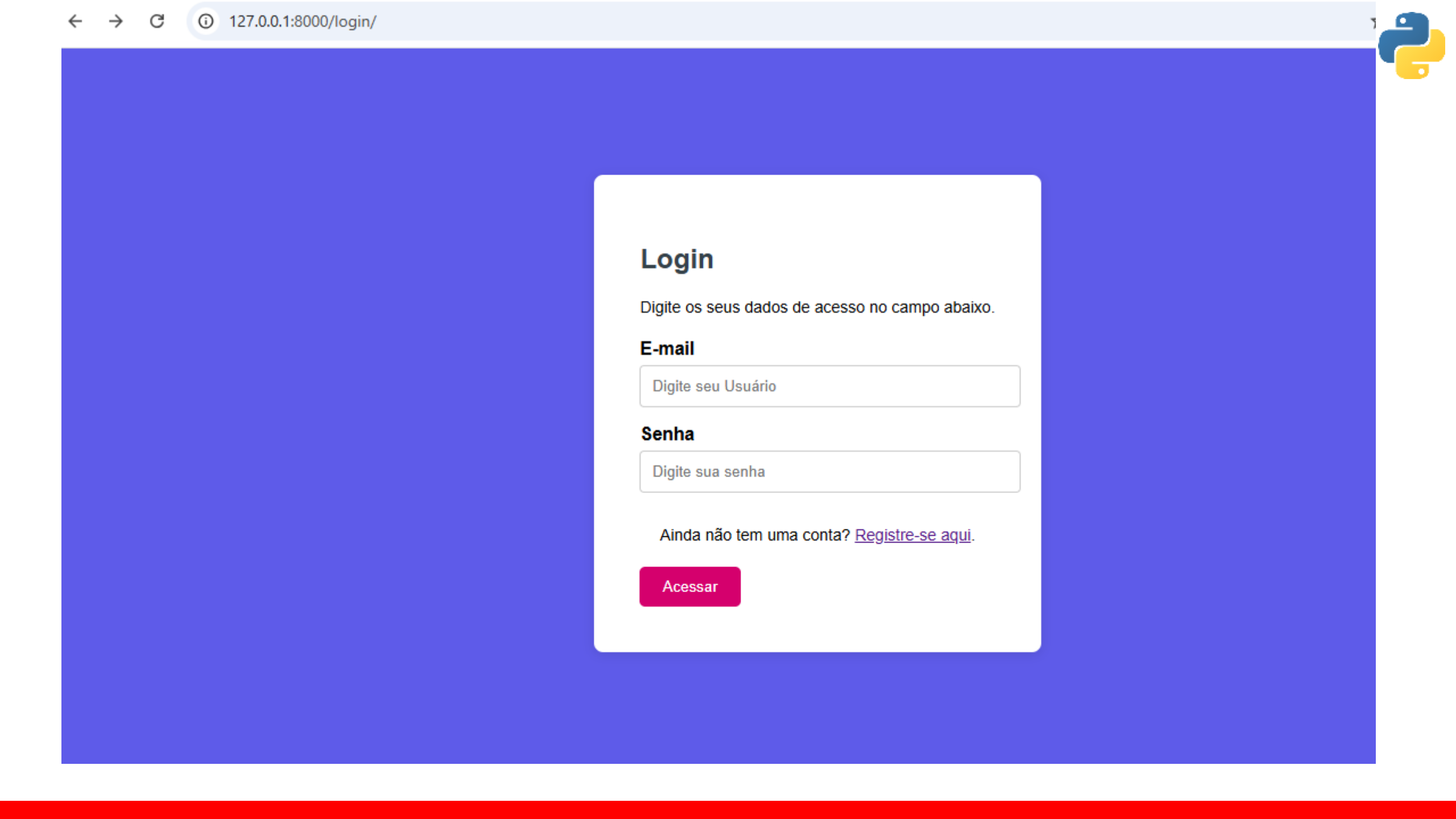
– Execute o comando abaixo no CMD, Terminal ou BASH para subir o servidor

python manage.py runserver

Acesse a URL abaixo no seu navegador

http://127.0.0.1:8000/login/

Oque nós vamos receber?


127.0.0.1:8000/login/
Login
Digite os seus dados de acesso no campo abaixo.
E-mail
Digite seu Usuário
Senha
Digite sua senha
Ainda não tem uma conta? Registre-se aqui.
Acessar

DJANGO

- Use sua senha de acesso ao admin para acessar as rotas protegidas pelo login auth.

- Ao logar, você vai ser redirecionado para a rota dashboard como definimos na nossa view, mas ainda não vai ver nada, para conseguir ver, vamos adicionar a linha abaixo como primeira linha em todo os templates **com excessão de base.html**, desta forma, vamos chamar todo o estilo pai que definimos em base.html, assim como o esqueleto do nosso futuro menu navbar de acesso.

- **{% extends 'pdv/base.html' %}**

Faça login novamente e se tudo der certo você será redirecionado para a pagina de dashboard com os links para as demais rotas herdados a partir de base.

Porém ainda feito e com estilo ruim, vamos resolver isso!!!

Vamos ajustar esses estilos e finalizar nossas demais paginas na próxima aula!

DJANGO

Atividade.

Anexar o link do git do projeto do hotel no exercicio do classroom, considerando tudo o visto em aula.
Com a implementação dos models e views corretos para o hotel.

NÃO ESQUEÇA