



# Python II - Desenvolvendo Aplicações Web

# Python II – Introdução ao curso

**Carga horária:** \*60 horas (15 sábados)

**Início:** 12/04/2025 | **Previsão de termino:** 16/08/2025

**\*Limite de Faltas:** 15 horas (3 sábados)

## Objetivos:

Este curso tem como objetivo capacitar os estudantes e profissionais e estudantes de tecnologia da informação para **desenvolvimento web** com Python utilizando o **framework Django**.



# Apresentação e Expectativas

**Expectativa:**

1 -





**Pasta compartilhada**

<https://drive.google.com/drive/folders/1xlx3096UaE26mCEo0WXKhxLGhXkF09BV?usp=sharing>



**WhatsApp**

<https://chat.whatsapp.com/LIVkOTUbU9GBBHUEm6R5J1>

# Python II – Introdução ao curso

Unidades curriculares	Carga horária
UC1: Desenvolver aplicações web com Python.	60 horas
<b>Carga horária total</b>	<b>60 horas</b>

## Indicadores

1. Desenvolve aplicação web com framework, de acordo com os requisitos do projeto de software.
2. Manipula banco de dados com mapeamento objeto-relacional, conforme requisitos do projeto.
3. Cria serviços web, conforme padrões definidos pela arquitetura REST.
4. Realiza implantação de aplicações web em Python, conforme orientações dos serviços de nuvem.

## Elementos da competência

### CONHECIMENTOS

- Ambiente de desenvolvimento: instalação e configuração de ambientes locais.
- Conceitos de arquiteturas de aplicações web: programação do lado do servidor e do cliente; diferenças entre monólito e micros serviços; protocolo HTTP.
- Controle de versão: utilizar git e github.
- Programação Orientada à objetos: princípios e aplicações.
- Framework web em Python: rotas, renderização de templates, bibliotecas, módulos, modelos e acesso ao banco de dados.
- Banco de dados: relacionais e não relacionais.
- Serviços web: Arquitetura para sistemas distribuídos, API Restful, ferramentas para construção de APIs.
- Ferramentas de deploy: conceito de deploy, finalidade e funcionamento.



# Histórico

**Quem criou?** Van Rossum

**Quando?** O Python foi criado em 1989 e lançado em 1991 como nome de Python

**Com qual motivação?** Precisava de uma linguagem de script que tivesse sintaxe semelhante ao **ABC** e que tivesse acesso às chamadas de sistema;

## Principais Características:

- **Linguagem simples** em relação a outras linguagens como C e C++;
- Tem **suporte** a orientação a objetos;
- Interpretada por uma máquina virtual;
- Tipagem dinamicamente forte;
- Multiplataforma (funciona em sistemas Windows, Linux e Mac);



# Interpretador || Compilador



**CPython** é a implementação principal da linguagem de programação Python, escrita em Linguagem C. CPython é um interpretador de Bytecode.



**PyPy** é um compilador **Just in Time (JIT)**, ele usa uma técnica conhecida como meta-tracing, que é responsável por **transformar um interpretador em um compilador JIT**.



**Jython** é uma implementação da linguagem Python que **gera bytecode** para máquinas Java (JVM - Java Virtual Machine). Com ela é possível fazer o desenvolvimento de aplicações híbridas que unem código em Java e Python.



**IronPython** é uma implementação da linguagem de programação Python escrita em C#, para plataforma.NET e Mono, criada por Jim Hugunin.

# Versões/Documentação

## Download the latest version for Windows

Download Python 3.13.3

Looking for Python with a different OS? Python for [Windows](#),  
[Linux/UNIX](#), [macOS](#), [Other](#)

Want to help test development versions of Python 3.14? [Pre-releases](#),  
[Docker images](#)

<https://www.python.org/downloads/>

### Download

Download these documents

### Docs by version

Python 3.14 (in development)  
Python 3.13 (stable)  
Python 3.12 (security-fixes)  
Python 3.11 (security-fixes)  
Python 3.10 (security-fixes)  
Python 3.9 (security-fixes)  
Python 3.8 (EOL)  
Python 3.7 (EOL)

## Python 3.13.3 documentation

Welcome! This is the official documentation for Python 3.13.3.

### Documentation sections:

#### What's new in Python 3.13?

*Or all "What's new" documents since Python 2.0*

#### Installing Python modules

*Third-party modules and PyPI.org*

#### Tutorial

*Start here: a tour of Python's syntax and features*

#### Distributing Python modules

*Publishing modules for use by other people*

<https://docs.python.org/3.13/>

### Outros links:

<https://www.python.org/dev/peps/pep-0008/>

<https://wiki.python.org.br/GuiaDeEstilo>

<https://www.w3schools.com/python/>



# Para que serve o Python?

## Python

### Data Science

Pandas; NumPy;

Ploty; Matplotlib; Seaborn

Scikit Learn; PyTorch; NLTK; TensorFlow;

Keras; OpenCV

### Automação

### Extração de dados

**Selenium; BeautifulSoup**

Pytest

**PyAutoGui; PyWin32**

### Desenvolvimento

**Requests; Pyodbc**

Flask; Web2Py

**Django, Django API**

Tkinter; PyQt

### Desenvolvimento Game

**PyGame**

PyOpenGL

**Kivy**

### Ethical Hacker

Scapy

Flowgrep

Subrute

# Mão na massa...

## Ambientes utilizando Nuvem:

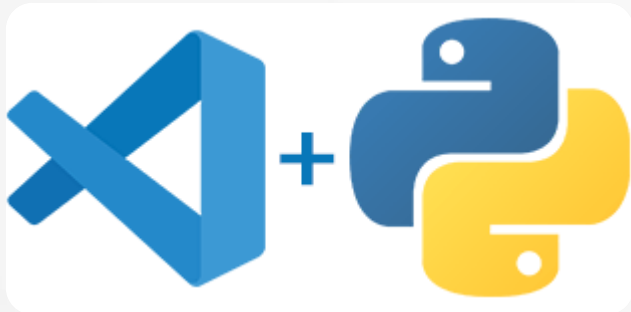
✓ Google Colab: <https://colab.research.google.com/>

✓ Repl.it: <https://replit.com/>

## Ambientes Locais:

✓ Realizar a instalação do **VS Code**  
<https://code.visualstudio.com/download>

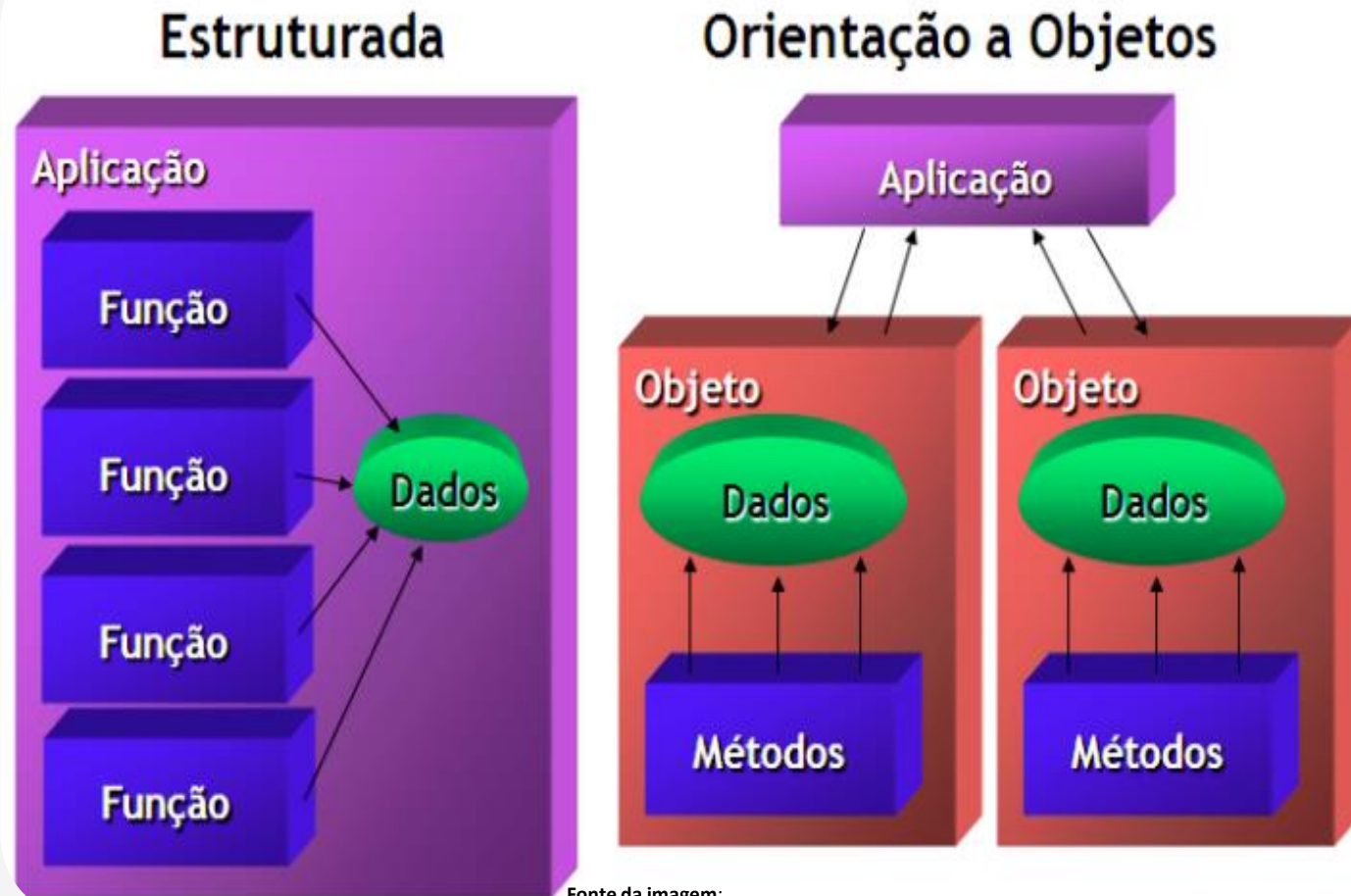
✓ Utilizar o interpretador **Python 3.10**  
<https://ideasdays.com/python/pyth.html>





# **P OO – Programação Orientação a Objetos**

# Estruturada vs. Orientação a Objetos



Fonte da imagem:

<https://docente.ifrn.edu.br/pedrobaesse/disciplinas/programacao-orientada-a-objetos/material-de-aula/aula-02-introducao-a-programacao-orientada-a-objeto>

## Programação

### Estruturada/Procedural

Chamada de procedimentos (ou funções) para manipulação de dados e variáveis;

## Programação

### Orientada a Objetos - POO

Estruturas de CLASSES com comportamentos;



# Paradigmas de programação:

Paradigma de programação é um meio de se classificar as linguagens de programação baseado em suas funcionalidades. As linguagens podem ser classificadas em vários paradigmas. O paradigma de programação fornece a visão que o programador possui sobre a estruturação e execução do programa.

**Mais informações:** [https://pt.wikipedia.org/wiki/Paradigma\\_de\\_programa%C3%A7%C3%A3o](https://pt.wikipedia.org/wiki/Paradigma_de_programa%C3%A7%C3%A3o)

## Programação Estruturada/Procedural/Imperativa

- ☐ Funcional;
- ☐ Lógico;
- ☐ Declarativa;
- ☐ Orientado a eventos;

## Programação Orientada a Objetos (POO)

- ☐ Abstração;
- ☐ Herança/Polimorfismo;
- ☐ Encapsulamento;
- ☐ Composição;
- ☐ Agregação;



# Pilares - POO

A Programação Orientada a Objetos está sedimentada sobre quatro pilares derivados do princípio da abstração, são eles:

- ☐ Encapsulamento;
- ☐ Herança;
- ☐ Composição;
- ☐ Polimorfismo;



<https://materialpublic.imd.ufrn.br/curso/disciplina/2/8/1/4>

# Orientação a Objetos

Provê uma melhor organização do código.  
Contribui para o reaproveitamento do código

**POO -  
Vantagens**

Pode não possuir o mesmo desempenho de códigos estruturados. Seus conceitos são mais difíceis de compreender que os conceitos da programação estruturada.

**POO -  
Desvantagens**

# Estruturada && Orientação a Objetos

- ✓ Na **Programação estruturada** observamos algumas vantagens como um controle mais eficaz quanto ao fluxo de execução do programa e a facilidade de compreender o código quando o mesmo é analisado.
- ✓ Na **POO** temos como vantagens a reutilização de código e a melhor organização do código do programa.

Em ambas os paradigmas existem características bem peculiares nas suas definições, onde dependendo da complexidade da solução a ser implementada uma pode ser mais viável que a outra.

De forma que podemos entender que ao invés de concorrer, **se usadas de maneira inteligente pelos programadores elas se complementam!**



# Linguagens multiparadigma

```
print('Calculadora...\n')

while True:
    op = int(input('1 para inserir valores\n2 para efetuar operações\n3 para sair\nOpção: '))
    if op == 1:
        var1 = float(input('Digite um valor: '))
        var2 = float(input('Digite outro valor: '))
    elif op == 2:
        while True:
            op1 = int(input('\n1 para soma\n2 para subtrair\n3 para multiplicar\n4 para dividir\n5 para sair\nOpção: '))
            if op1 == 1:
                print('Soma: ', var1 + var2)
            elif op1 == 2:
                print('Subtração: ', var1 - var2)
            elif op1 == 3:
                print('Multiplicação: ', var1 * var2)
            elif op1 == 4:
                print('Divisão: ', var1 / var2)
            else:
                break
    else:
        break
```

## Programação Estruturada Utilizando a linguagem **Python**

Fonte:

[https://pt.wikipedia.org/wiki/Linguagem\\_de\\_programa%C3%A7%C3%A3o\\_multiparadigma](https://pt.wikipedia.org/wiki/Linguagem_de_programa%C3%A7%C3%A3o_multiparadigma)



# Linguagens multiparadigma

## Classe - POO

```
class Calculadora:
    def __init__(self):
        self.__var1 = None
        self.__var2 = None

    def lerValores(self):
        self.__var1 = float(input('Digite um valor: '))
        self.__var2 = float(input('Digite outro valor: '))

    def adicao(self):
        soma = self.__var1 + self.__var2
        return soma

    def subtracao(self):
        sub = self.__var1 - self.__var2
        return sub

    def multiplicacao(self):
        mult = self.__var1 * self.__var2
        return mult

    def divisao(self):
        div = self.__var1 / self.__var2
        return div
```

```
from Calculadora import Calculadora

c = Calculadora()
while True:
    op = int(input('1 para setar valores\n2 para operações\n3 para sair\nOpção: '))
    if op == 1:
        c.lerValores()
    elif op == 2:
        while True:
            op1 = int(input('\n1 para soma\n2 para subtração\n3 para multiplicação\n4 para divisão\n5 para sair\nOpção: '))
            if op1 == 1:
                print(c.adicao())
            elif op1 == 2:
                print(c.subtracao())
            elif op1 == 3:
                print(c.multiplicacao())
            elif op1 == 4:
                print(c.divisao())
            else:
                break
    else:
        break
```

## Orientação a Objetos (POO) Utilizando a linguagem Python

Fonte: [https://pt.wikipedia.org/wiki/Linguagem\\_de\\_programa%C3%A7%C3%A3o\\_multiparadigma](https://pt.wikipedia.org/wiki/Linguagem_de_programa%C3%A7%C3%A3o_multiparadigma)



# Classes e Objetos

A *classe* de termos e o *objeto* descrevem o *tipo* de objetos e as *instâncias* de classes, respectivamente. Sendo assim, o ato de criar um objeto é chamado de *instanciação*.

Palavra chave  
do C# para  
criar uma  
classe

C#

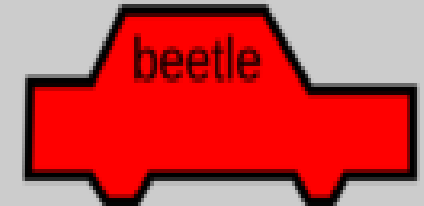
Nome da  
Classe

```
class SampleClass  
{  
}
```

class



objects



# Instanciando uma classe com Python

```
class Produto:
    def __init__(self, descricao, preco):
        self.descricao=descricao
        self.preco=preco
    def verProduto(self):
        print(f'{self.descricao} por apenas R${self.preco:.2f}')
```

```
from classe import Produto
notebook = Produto('Notebook Dell i7', 5800.90)
input('Pressione qualquer tecla para ver a oferta: ')
notebook.verProduto()
print()
```

```
Pressione qualquer tecla para ver a oferta:
Notebook Dell i7 por apenas R$5800.90
```

## Desafio Calculadora:

Fazer uma calculadora que receba **dois valores** e retorne o resultado de **adição**, **subtração**, **multiplicação** ou **divisão** destes dois valores;

# Resposta do desafio Python

```
class Calculadora:
    def __init__(self, valor1, valor2):
        self.valor1=valor1
        self.valor2 =valor2
    def soma(self):
        return self.valor1 + self.valor2
    def sub(self):
        return self.valor1 - self.valor2
    def mult(self):
        return self.valor1 * self.valor2
    def div(self):
        return self.valor1 / self.valor2
```

```
from classe import Calculadora
valor1 = int(input('Informe o 1º valor: '))
valor2 = int(input('Informe o 2º valor: '))
calc = Calculadora(valor1, valor2)
print('Soma: ')
print(calc.soma())
print('---')
print('Subtração: ')
print(calc.sub())
print('---')
print('Multiplicação: ')
print(calc.mult())
print('---')
print('Divisão: ')
print(calc.div())
print('---')
```

```
Informe o 1º valor: 100
Informe o 2º valor: 50
Soma:
150
---
Subtração:
50
---
Multiplicação:
5000
---
Divisão:
2.0
```

# Mão na massa... Encapsulamento com o Python

```
class AssinaturaTv:
    def __init__(self):
        self.canaisDisponiveis=(2, 5, 7, 15, 30, 50)
        self.__canalAtivo = 7
        self.volume=10

    @property
    def canalAtivo(self):
        return(self.__canalAtivo)

    @canalAtivo.setter
    def canalAtivo(self, canal):
        try:
            index=self.canaisDisponiveis.index(canal)
            self.__canalAtivo=canal
            print(f'O canal ativo agora é: {canal}')
        except:
            index=-1
            print(f'O canal {canal} não está disponível')
```

```
from classe import AssinaturaTv
tv = AssinaturaTv()
print(tv)
print('Canais disponíveis')
print(tv.canaisDisponiveis)
print('---')
print('Canal Ativo')
print(tv.canalAtivo)
print('---')
tv.canalAtivo = 15
print(tv.canalAtivo)
print('---')
tv.canalAtivo = 25
print('Canal Ativo')
print(tv.canalAtivo)
print('---')
```

```
Canais disponíveis
(2, 5, 7, 15, 30, 50)
---
Canal Ativo
7
---
O canal ativo agora é: 15
15
---
O canal 25 não está disponível
Canal Ativo
15
```

## Desafio Produto:

Construir uma classe Produto que com atributos **descrição**, **precoCusto** e **precoVenda**. O atributo **precoVenda** deve ser protegido e ter um **método GET e SET** alterando o valor apenas se o novo valor for maior que o **precoCusto**;

# Resposta do desafio

```
class Produto:
    def __init__(self, descricao, precoCusto, precoVenda):
        self.descricao=descricao
        self.precoCusto = precoCusto
        self.__precoVenda=precoVenda
    @property
    def precoVenda(self):
        return(self.__precoVenda)
    @precoVenda.setter
    def precoVenda(self, valor):
        if(valor>=self.precoCusto):
            self.__precoVenda=valor
            print(f'Preço de venda alterado para R${valor}')
        else:
            print('O preço de venda não pode ser alterado.')
```

```
from classe import Produto
notebook = Produto('Notebook Dell i7', 3800, 5800)
print("Mudar preço para 1000")
notebook.precoVenda=100
print("Mudar preço para 5100")
notebook.precoVenda=5100
print('----')
```

```
Mudar preço para 1000
O preço de venda não pode ser alterado.
Mudar preço para 5100
Preço de venda alterado para R$5100
----
```



# Herança com o Python

```
class.py
1 class Animal:
2     def __init__(self, cor, peso):
3         self.cor=cor
4         self.peso=peso
5     def dormir(self):
6         print('Dormirindo...')
7
8 class Passaro(Animal):
9     def __init__(self, cor, peso, bico):
10        super().__init__(cor, peso)
11        self.bico=bico
12    def voar(self):
13        print('Voando...')
14
15 class Papagaio(Passaro):
16     def __init__(self, cor, peso, bico):
17        super().__init__(cor, peso, bico)
18        self.sabeFalar=True
19    def falar(self):
20        print('Falando...')
21
22 class Cachorro(Animal):
23     def __init__(self, cor, peso):
24        super().__init__(cor, peso)
25        self.rabo=True
26    def latir(self):
27        print('Latindo...')
```

```
main.py > ...
1 from classe import *
2 animal=Animal('preto', '10kg')
3 cachorro=Cachorro('caramelo', '15kg')
4 passaro=Passaro('vermelho', '200g', 'curto')
5 papagaio=Papagaio('verde/amarelo', '500g',
6
7 print('Animal')
8 print(animal.cor, animal.peso)
9 animal.dormir()
10 print('---')
11
12 print('Cachorro')
13 print(cachorro.cor, cachorro.peso, cachorro
14 cachorro.dormir()
15 cachorro.latir()
16 print('---')
17
18 print('Passaro')
19 print(passaro.cor, passaro.peso, passaro.bi
20 passaro.dormir()
21 passaro.voar()
22 print('---')
23
24 print('Papagaio')
25 print(papagaio.cor, papagaio.peso, papagaio
26 papagaio.dormir()
27 papagaio.voar()
28 papagaio.falar()
29 print('---')
```

## Exemplo de Composição Agregação

```
class CicloMensal:
    def __init__(self, mes, ano):
        self.mes=mes
        self.ano=ano
        self.lancamentos=[]
    def addLancamento(self, descricao, valor):
        lancamento=Lancamento(descricao, valor)
        self.lancamentos.append(lancamento)

    def calcularTotal(self):
        total=0
        for l in self.lancamentos:
            print(f'{l.descricao} | R${l.valor:.2f}')
            total+=l.valor
        return total

class Lancamento:
    def __init__(self, descricao, valor):
        self.descricao=descricao
        self.valor=valor
```

```
from classe import *
abril=CicloMensal( 'Abril', 22)
abril.addLancamento('Salario', 4000)
abril.addLancamento('Cartão Alimentação', 600)
abril.addLancamento('Trabalho Extra', 2000)
abril.addLancamento('Supermercado', -2000)
abril.addLancamento('Cartão de credits', -1800)
print('-----')
print(f'Lançamentos: {abril.mes}/{abril.ano}')
print('---')
total=abril.calcularTotal()
print('---')
print(f'Total: R${total}')
```

```
Lançamentos: Abril/22
---
Salario | R$4000.00
Cartão Alimentação | R$600.00
Trabalho Extra | R$2000.00
Supermercado | R$-2000.00
Cartão de credits | R$-1800.00
---
Total: R$2800
```

## Desafio CarrinhoCompras:

Construir uma classe **CarrinhoCompras** que tenha como **atributos**: **data da compra**, um objeto **Cliente** composto por **nome e endereço** e o atributo **produtos** onde serão agregado **N objetos da classe produto**. A classe produto deve ter os atributos **descrição e preço**. A classe Carrinho de compras deve ter um método para a somar os preços de todos os produtos agregados e exibir o total.

# Resposta do desafio

```
class CarrinhoCompas:
    def __init__(self, data, nome, endereco):
        self.data=data
        self.cliente=Cliente(nome, endereco)
        self.produtos=[]
    def addProduto(self,descricao, preco):
        produto=Produto(descricao, preco)
        self.produtos.append(produto)
    def calcularTotal(self):
        total=0
        for p in self.produtos:
            print(p.descricao, p.preco)
            total+=p.preco
        return total
```

```
class Cliente:
    def __init__(self, nome, endereco):
        self.nome=nome
        self.endereco=endereco
class Produto:
    def __init__(self, descricao, preco):
        self.descricao=descricao
        self.preco = preco
```

```
from classe import *
data="30/04/2022"
nome="Joaquim da Silva"
endereco="Rua dos Pombos, 000"
carrinho=CarrinhoCompas( data, nome, endereco)
carrinho.addProduto('Notebook Dell i7', 5800)
carrinho.addProduto('SmartPhone', 2800)
carrinho.addProduto('XBox', 3400)
print(f'Carrinho: {carrinho.data}')
print('Cliente',carrinho.cliente.nome)
print('Endereco',carrinho.cliente.endereco)
print('-----')
total=carrinho.calcularTotal()
print(f'Total R${total}')
print('---')
```

```
Carrinho: 30/04/2022
Cliente Joaquim da Silva
Endereco Rua dos Pombos, 000
-----
Notebook Dell i7 5800
SmartPhone 2800
XBox 3400
Total R$12000
```



```

    if not hasattr(self, 'max_length=200): votes = models.IntegerField() def was_published_recently(self): return self.pub_date <= timezone.now() - datetime.timedelta(days=1) def get_object_or_404(self, pk): return get_object_or_404(Poll, pk=pk) def get(self, request, poll_id): p = get_object_or_404(Poll, pk=poll_id) try: selected_choice = p.choice_set.get(pk=request.POST['choice']) except (KeyError, Choice.DoesNotExist): if request.method == 'POST': self.vote(request, poll_id) return HttpResponseRedirect(reverse('polls:detail', args=(p.id,))) else: selected_choice.votes += 1 selected_choice.save() return HttpResponseRedirect(reverse('polls:results', args=(p.id,))) from django.utils import unittest from myapp.models import Animal class AnimalTestCase(unittest.TestCase): def setUp(self): self.objects.create(name="lion", sound="roar") self.cat = Animal.objects.create(name="cat", sound="meow") def test_animals_can_speak(self): self.assertEqual(self.lion.speak(), 'The lion says "roar"') self.assertEqual(self.cat.speak(), 'The cat says "meow"') from django.contrib.syndication.views import FeedDoesNotExist from django.shortcuts import get_object_or_404 class BeatFeed(Feed): description_template = 'feeds/beat_description.html' def get_object(self, request, beat_id): return get_object_or_404(Beat, pk=beat_id) def title(self, obj): return "Chicagocrime.org: Crimes for beat %s" % obj.beat def link(self, obj): return obj.get_absolute_url() def description(self, obj): return "Crimes recently reported in police beat %s" % obj.beat def items(self, obj): return Crime.objects.filter(beat=obj).order_by('-time_date')[:30] from django.utils.html import conditional_escape from django.utils.safestring import mark_safe @register.filter(needs_autoescape=True) def initial_letter_filter(text, autoescape=None): first, other = text[0], text[1:] if autoescape: esc = conditional_escape else: esc = lambda x: x result = '<strong>%s</strong>%s' % (esc(first), esc(other)) return mark_safe(result) from django import template @register.filter(needs_autoescape=True) def do_current_time(parser, token): try: tag_name, format_string = token.split_contents() except ValueError: raise template.TemplateSyntaxError("%r tag requires a single argument" % token.contents.split()[0]) if not (format_string[0] == format_string[-1] and format_string[0] in ('"', "'")): raise template.TemplateSyntaxError("%r tag's argument should be in quotes" % tag_name) return CurrentTimeNode(format_string[1:-1]) from django.conf import settings from django import template import datetime class CurrentTimeNode(template.Node): def __init__(self, format_string): self.format_string = format_string def render(self, context): return datetime.datetime.now().strftime(self.format_string) class CycleNode(Node): def __init__(self, cyclevars): self.cyclevars = cyclevars def render(self, context): if self not in context.render_context: context.render_context[self] = itertools.cycle(self.cyclevars) cycle_iter = context.render_context[self] return cycle_iter.next() class MaleManager(models.Manager): def get_query_set(self): return super(MaleManager, self).get_query_set().filter(sex='M') class FemaleManager(models.Manager): def get_query_set(self): return super(FemaleManager, self).get_query_set().filter(sex='F') class Person(models.Model): first_name = models.CharField(max_length=50) last_name = models.CharField(max_length=50) sex = models.CharField(max_length=1, choices= (('M', 'Male'), ('F', 'Female'))) people = models.Manager() men = MaleManager() women = FemaleManager() from django import forms class ContactForm(forms.Form): subject = forms.CharField(max_length=100) message = forms.CharField() sender = forms.EmailField() cc_myself = forms.BooleanField(required=False) if form.is_valid(): subject = form.cleaned_data['subject'] message = form.cleaned_data['message'] sender = form.cleaned_data['sender'] cc_myself = form.cleaned_data['cc_myself'] recipients = ['info@example.com'] if cc_myself: recipients.append(sender) from django.core.mail import send_mail send_mail(subject, message, sender, recipients) return HttpResponseRedirect('/thanks/') from django.http import HttpResponse from django.template import Context, loader def my_view(request): t = loader.get_template('myapp/template.html') c = Context({'name': 'John'}) return HttpResponse(t.render(c), mimetype="application/xhtml+xml") from django.template.loaders import app_directories class Loader(app_directories.Loader): is_usable = True def load_template(self, template_name, template_dirs=None): source, origin = self.load_template_source(template_name, template_dirs) template = Template(source) return template, origin from django import template from django.template.defaultfilters import string_filter register = template.Library() @register.filter @stringfilter def lower(value): return value.lower() from django.utils.html import conditional_escape from django.utils.safestring import mark_safe @register.filter(needs_autoescape=True) def initial_letter_filter(text, autoescape=None): first, other = text[0], text[1:] if autoescape: esc = conditional_escape else: esc = lambda x: x result = '<strong>%s</strong>%s' % (esc(first), esc(other)) return mark_safe(result) from django import template def do_current_time(parser, token): try: tag_name, format_string = token.split_contents() except ValueError: raise template.TemplateSyntaxError("%r tag requires a single argument" % token.contents.split()[0]) if not (format_string[0] == format_string[-1] and format_string[0] in ('"', "'")): raise template.TemplateSyntaxError("%r tag's argument should be in quotes" % tag_name) return CurrentTimeNode(format_string[1:-1]) from django.conf import settings from django.contrib.auth.models import User, check_password class SettingsBackend(object): supports_inactive_user = False def authenticate(self, username=None, password=None): login_valid = settings.ADMIN_LOGIN == username pwd_valid = check_password(password, settings.ADMIN_PASSWORD) if login_valid and pwd_valid: try: user = User.objects.get(username=username) except User.DoesNotExist: user = User(username=username, password='get from settings.py') user.is_staff = True user.is_superuser = True user.save() return user return None def get_user(self, user_id): try: user = User.objects.get(pk=user_id) except User.DoesNotExist: return None class EntryDetail(models.Model): entry = models.OneToOneField(Entry) details = models.TextField() class Blog(models.Model): name = models.CharField(max_length=100) tagline = models.TextField() def __unicode__(self): return self.name class Author(models.Model): name = models.CharField(max_length=50) email = models.EmailField() def __unicode__(self): return self.name class Entry(models.Model): blog = models.ForeignKey(Blog) headline = models.CharField(max_length=255) body_text = models.TextField() pub_date = models.DateTimeField() authors = models.ManyToManyField(Author) n_comments = models.IntegerField() n_pingbacks = models.IntegerField() rating = models.IntegerField() def __unicode__(self): return self.headline class BookManager(models.Manager): def create_book(title): book = self.create(title=title) return book class Book(models.Model): title = models.CharField(max_length=255) author = models.ForeignKey(Author) def __unicode__(self): return self.title book = Book.objects.create_book("Pride and Prejudice") from django.core.signals import request_finished from django.dispatch import receiver @receiver(request_finished)

```

# Django

- O Django é um framework de aplicativos web gratuito e de código aberto escrito em Python.
- Criado por desenvolvedores experientes, o framework cuida da estrutura da aplicação.
- **Iniciar um projeto:**  
<https://docs.djangoproject.com/pt-br/4.0/intro/tutorial01/>

## Conheça o Django

Django é um framework web Python de alto nível que encoraja o desenvolvimento rápido e design limpo e pragmático. Construído por desenvolvedores experientes, ele resolve grande parte do incômodo do desenvolvimento da Web, para que você possa se concentrar em escrever seu aplicativo sem precisar reinventar a roda. É gratuito e de código aberto.



Ridiculamente rápido.

O Django foi projetado para ajudar os desenvolvedores a levar os aplicativos do conceito à conclusão o mais rápido possível.



Tranquilamente seguro.

O Django leva a segurança a sério e ajuda os desenvolvedores a evitar muitos erros comuns de segurança.



Excessivamente escalável.

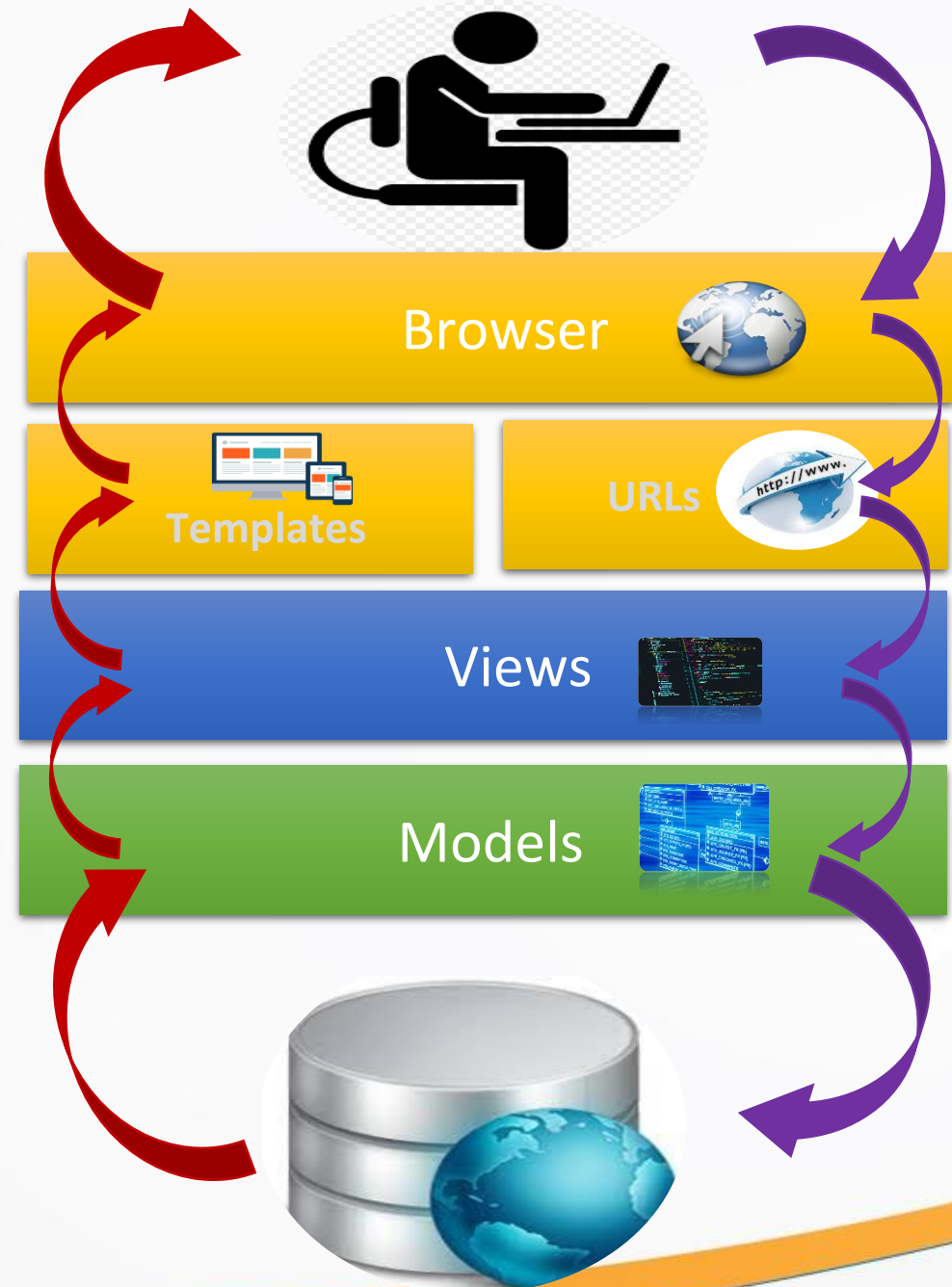
Alguns dos sites mais movimentados da web aproveitam a capacidade do Django de escalar de forma rápida e flexível.



# Django: Estrutura

O Django utiliza um padrão chamado de MTV (similar ao MVC)

- ☐ **M Modelo** ou Regras de negócio;
- ☐ **T Template** Arquivos HTMLs que serão renderizados;
- ☐ **V View** equivalente aos controllers que renderizam os templates.



# Django: Estrutura

## Projeto Vs. Aplicações

- ❑ O Django segue uma estrutura onde um projeto encapsula uma ou mais aplicações, cada uma com a sua funcionalidade específica.
- ❑ As aplicações são plugáveis, de forma que podem ser reaproveitadas em diversos projetos.

### E-Commerce



#### Comando criação de um PROJETO

# No diretório do projeto:

- `django-admin startproject nome_projeto.`

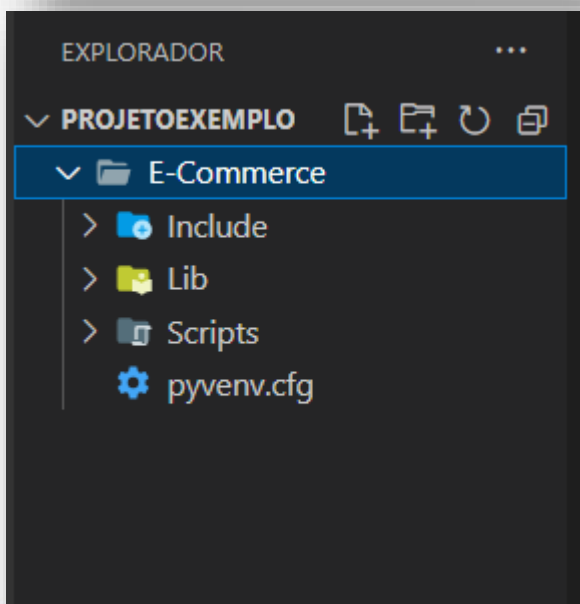
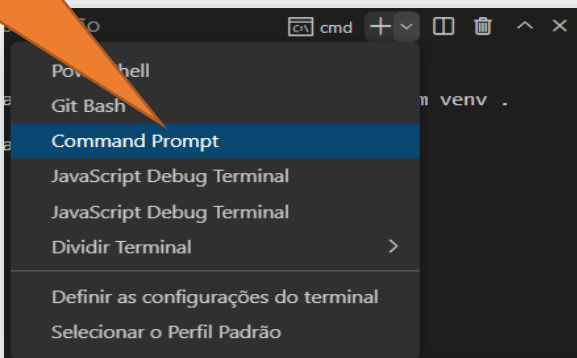
#### Comando criação de uma APLICAÇÃO

# No diretório do projeto:

- `django-admin startapp core`

# Django: Mão na massa

CMD é menos propenso a erros que o PowerShell



## Configuração da VENV

```
PROBLEMAS  SAÍDA  TERMINAL  JUPYTER  CONSOLE

Microsoft Windows [versão 10.0.19044.1826]
(c) Microsoft Corporation. Todos os direitos reservados.

D:\ProjetoExemplo>python -m venv .\E-Commerce

D:\ProjetoExemplo>cd E-Commerce

D:\ProjetoExemplo\E-Commerce>Scripts\activate

(E-Commerce) D:\ProjetoExemplo\E-Commerce>
```

Criação do ambiente virtual VENV

Ativação da VENV

VENV ativada

# Django: Instalação

Instalação do  
Django

PROBLEMAS SAÍDA TERMINAL JUPYTER CONSOLE DE DEPURACÃO

```
(E-Commerce) D:\ProjetoExemplo\E-Commerce>pip install django
```

```
Collecting django
```

```
  Downloading Django-4.0.6-py3-none-any.whl (8.0 MB)
```

```
      ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.0/8.0 MB 10.5 MB/s eta 0:00:00
```

```
Collecting tzdata
```

```
  Downloading tzdata-2022.1-py2.py3-none-any.whl (339 kB)
```

```
      ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 339.5/339.5 KB 10.6 MB/s eta 0:00:00
```

```
Collecting asgiref<4,>=3.4.1
```

```
  Downloading asgiref-3.5.2-py3-none-any.whl (22 kB)
```

```
Collecting sqlparse>=0.2.2
```

```
  Downloading sqlparse-0.4.2-py3-none-any.whl (42 kB)
```

```
      ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 42.3/42.3 KB 2.0 MB/s eta 0:00:00
```

```
Installing collected packages: tzdata, sqlparse, asgiref, django
```

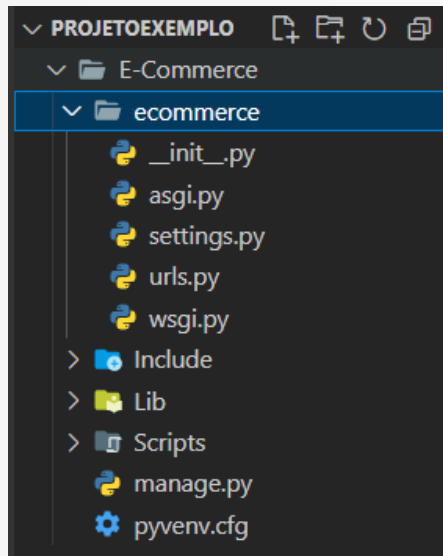
```
Successfully installed asgiref-3.5.2 django-4.0.6 sqlparse-0.4.2 tzdata-2022.1
```

```
20cc622f01111a 1u2f911e6 9281e6t-3.2.5 q19u8o-v.0.0 2d1b9e26-0.v.5 f3q9f9-5055.T  
1u2f9111u6 co11ecf6q b9ck98e2: f3q9f9' 2d1b9e26' 9281e6t' q19u8o
```

```
      ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 45.3/45.3 KB 5.0 MB/s eta 0:00:00
```

```
Dom1to9q1u6 2d1b9e26-0.v.5-b13-uou6-9u1.m11 (45 KB)
```

# Django: Início de um Projeto



Iniciando um projeto Django

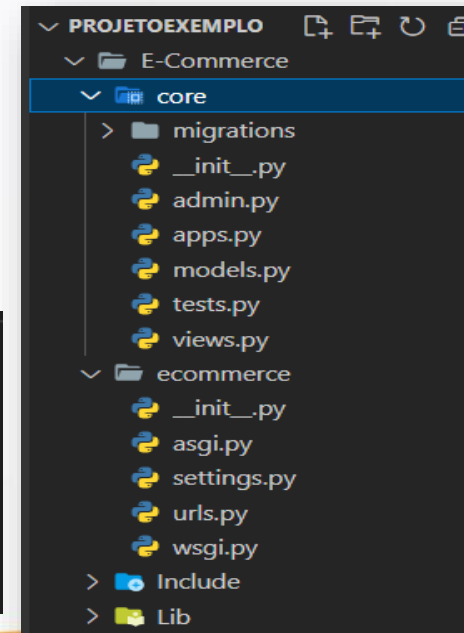
PROBLEMAS SAÍDA TERMINAL JUPYTER CONSOLE DE DEPURACÃO

```
(E-Commerce) D:\ProjetoExemplo\E-Commerce>django-admin startproject ecommerce .  
  
(E-Commerce) D:\ProjetoExemplo\E-Commerce>
```

Iniciando um app Django

PROBLEMAS SAÍDA TERMINAL JUPYTER CONSOLE DE DEPURACÃO

```
(E-Commerce) D:\ProjetoExemplo\E-Commerce>django-admin startapp core  
  
(E-Commerce) D:\ProjetoExemplo\E-Commerce>
```





# Django: Início de um Projeto

Iniciando o  
servidor Django

PROBLEMAS SAÍDA TERMINAL JUPYTER CONSOLE DE DEPURAÇÃO

```
(E-Commerce) D:\ProjetoExemplo\E-Commerce>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
```

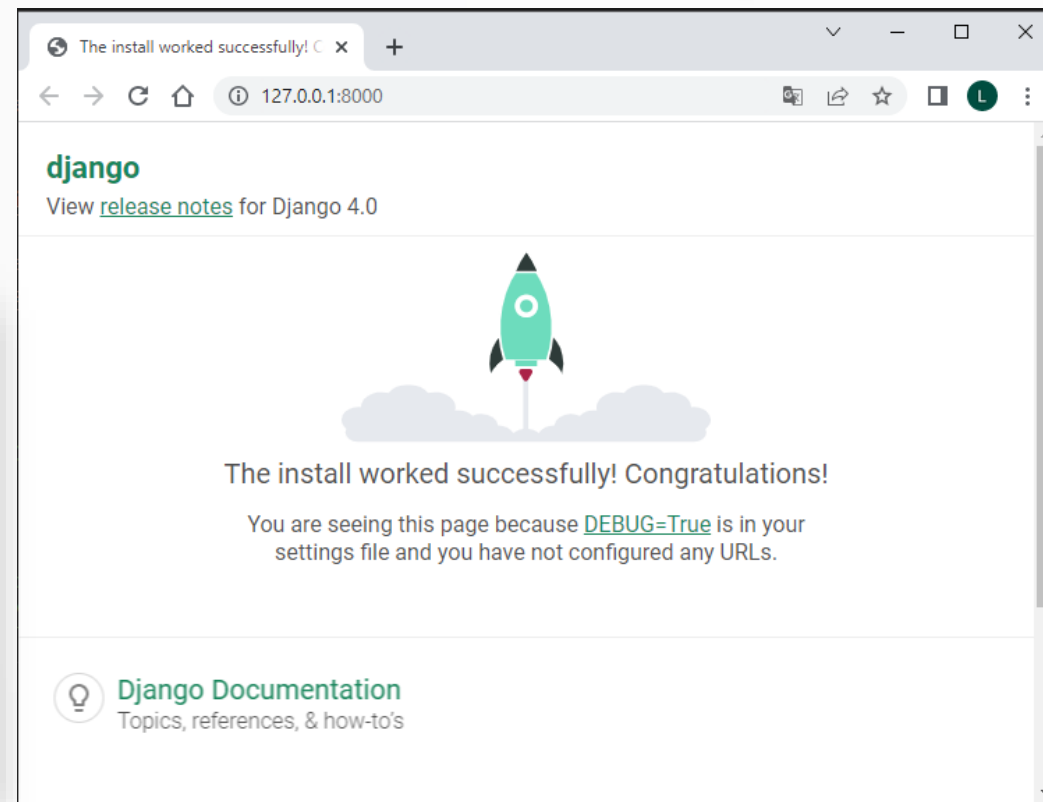
```
System check identified no issues (0 silenced).
```

```
You have 18 unapplied migration(s). Your project may not work properly
Run 'python manage.py migrate' to apply them.
```

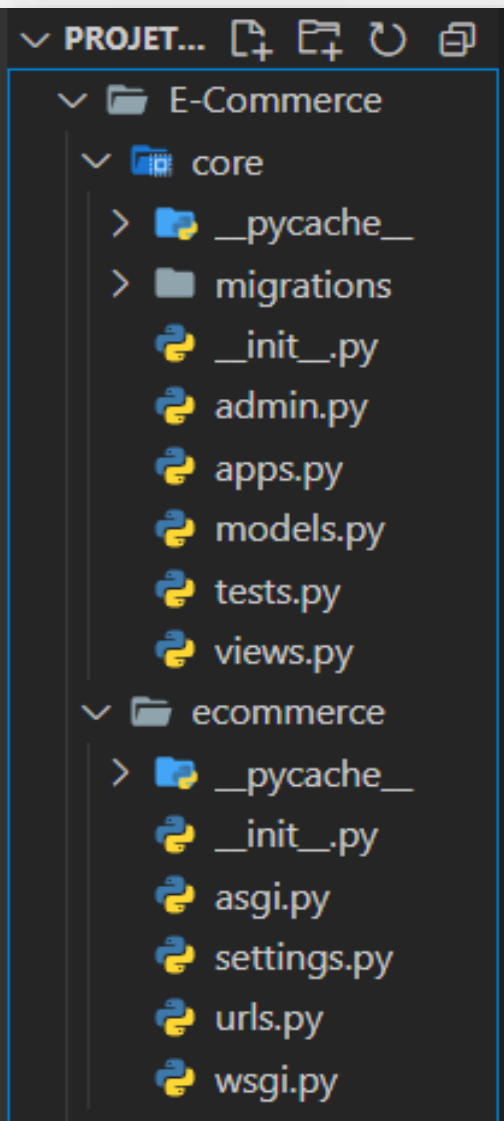
```
July 23, 2022 - 00:29:01
```

```
Django version 4.0.6, using settings 'ecommerce.settings'
```

```
Starting development server at http://127.0.0.1:8000/
```



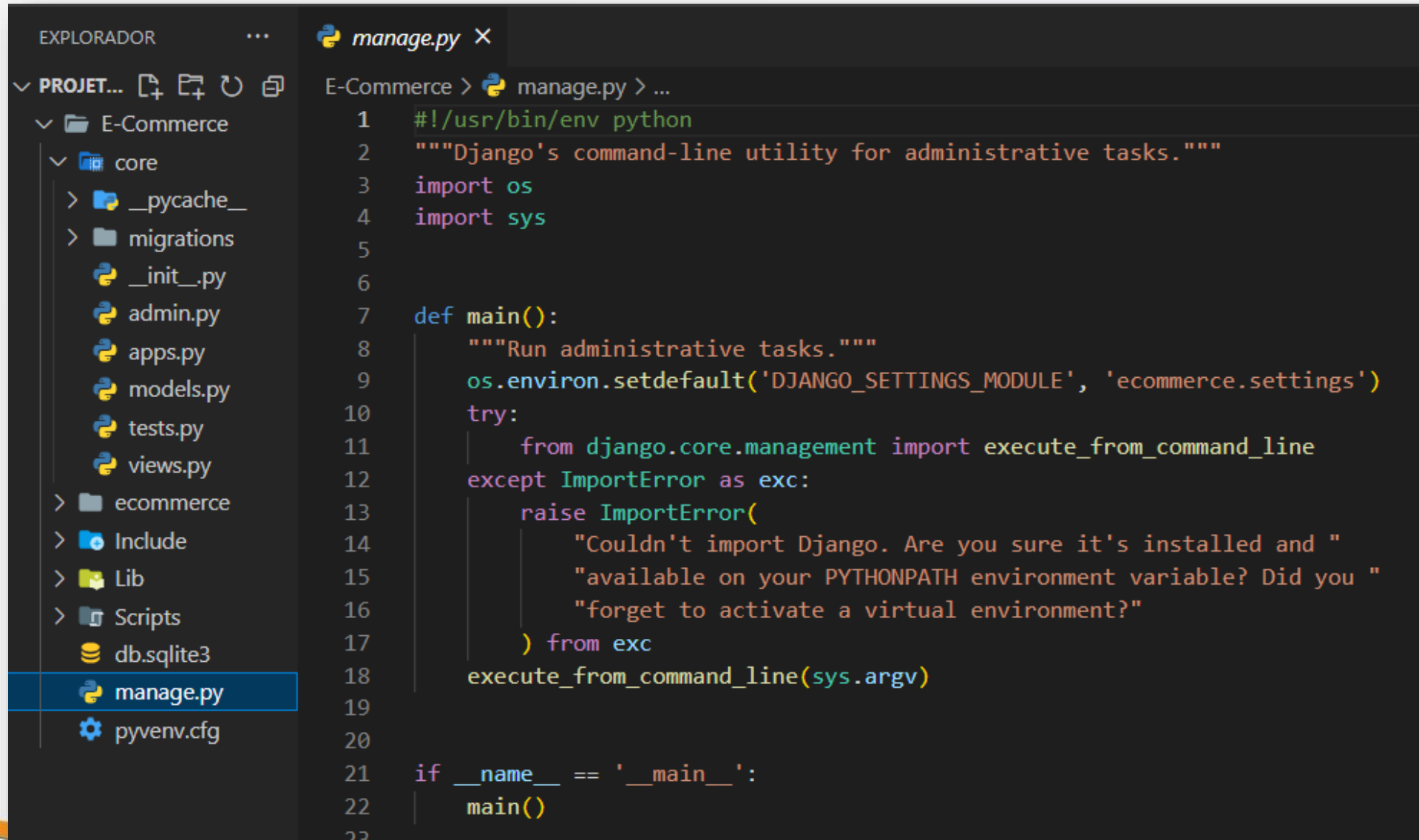
# Django: Estrutura de arquivos



## Principais arquivos

- ❑ **manage.py** – Arquivo responsável por gerenciar todo o projeto. Nele são importadas as bibliotecas do DJANGO, as variáveis de ambiente e as funções necessárias para a execução do projeto;
- ❑ **wsgi.py** – Arquivo responsável por conter as configurações necessárias para a realização de deploy(publicação) do projeto.
- ❑ **settings.py** – Configuração do DJANGO para o projeto. Contem parâmetros, biblioteca e funções importantes para a configuração da aplicação;
- ❑ **urls.py** – Arquivo que armazena as rotas do projeto. Funciona como um índice contendo todos os “lugares” da aplicação.
- ❑ **app/views.py** – Contem funções que renderizam os templates HTMLs definidos nas rotas.
- ❑ **Diretório templates** – Armazena os arquivos HTMLs que serão renderizados pelas rotas.
- ❑ **models.py** – Contem as classes dos models (objetos que mapeiam as tabelas do banco de dados)

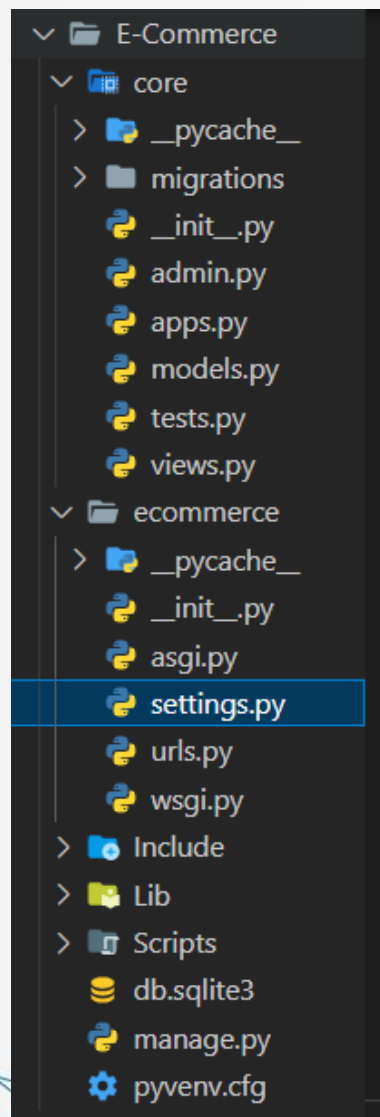
# Django: Estrutura de arquivos



The image shows a code editor with a file explorer on the left and a code editor on the right. The file explorer, titled 'EXPLORADOR', shows a project structure with a folder 'E-Commerce' containing subfolders 'core' and 'ecommerce', and files like 'manage.py' and 'pyenv.cfg'. The code editor shows the contents of 'manage.py', which is a Django management utility script. The script starts with a shebang line, a docstring, imports 'os' and 'sys', defines a 'main()' function, and includes a standard if \_\_name\_\_ == '\_\_main\_\_': guard.

```
1  #!/usr/bin/env python
2  """Django's command-line utility for administrative tasks."""
3  import os
4  import sys
5
6
7  def main():
8      """Run administrative tasks."""
9      os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'ecommerce.settings')
10     try:
11         from django.core.management import execute_from_command_line
12     except ImportError as exc:
13         raise ImportError(
14             "Couldn't import Django. Are you sure it's installed and "
15             "available on your PYTHONPATH environment variable? Did you "
16             "forget to activate a virtual environment?"
17         ) from exc
18     execute_from_command_line(sys.argv)
19
20
21 if __name__ == '__main__':
22     main()
23
```

# Django: Estrutura de arquivos



```
settings.py X
E-Commerce > ecommerce > settings.py > ...

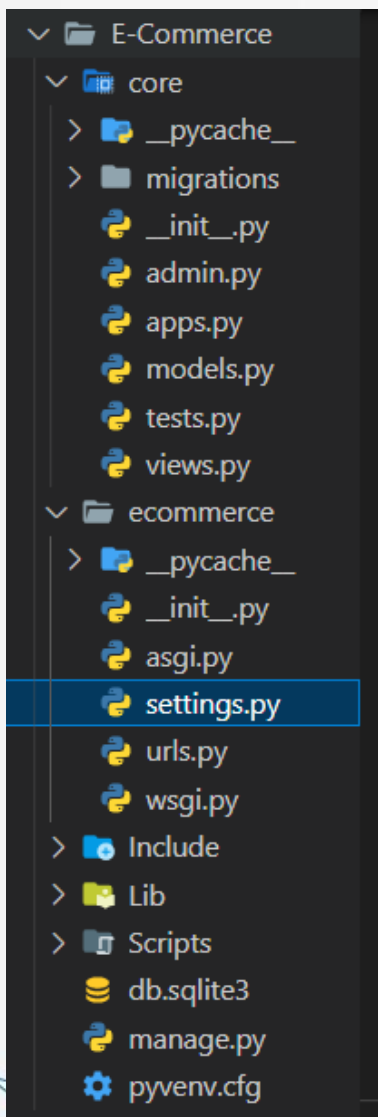
9 For the full list of settings and their values, see the
10 https://docs.djangoproject.com/en/4.0/ref/settings/
11 """
12
13 from pathlib import Path
14
15 # Build paths inside the project like this: BASE_DIR / 'subdir'.
16 BASE_DIR = Path(__file__).resolve().parent.parent
17
18
19 # If you want to allow Django to manage the database via settings - unsuitable for production
20 # See https://docs.djangoproject.com/en/4.0/howto/deployment/checklist/
21
22 # SECURITY WARNING: keep the secret key used in production secret!
23 SECRET_KEY = 'django-insecure-2l9+h8nkoje1l7%%'
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
28 ALLOWED_HOSTS = ['*']
29
30
```

Variável de ambiente  
diretório base do  
projeto.

Define ambiente de  
desenvolvimento ou  
produção.

Permissão de hosts. String  
com a url dos servidores  
permitidos ou '\*' para todos

# Django: Estrutura de arquivos



settings.py X

E-Commerce > ecommerce > settings.py > ...

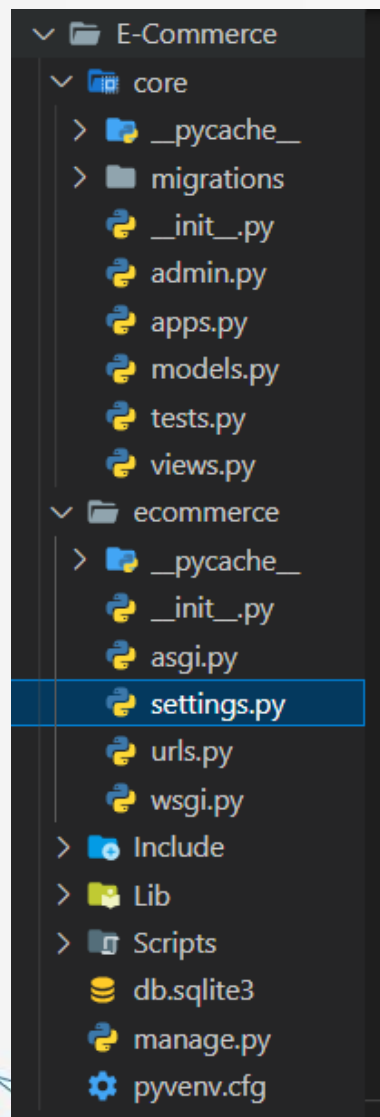
```
54
55  TEMPLATES = [
56      {
57          'BACKEND': 'django.template.backends.django.DjangoTemplates',
58          'DIRS': ['templates'],
59          'APP_DIRS': True,
60          'OPTIONS': {
61              'context_processors': [
62                  'django.template.context_processors.debug',
63                  'django.template.context_processors.request',
64                  'django.contrib.auth.context_processors.auth',
65                  'django.contrib.messages.context_processors.messages',
66              ],
67          },
68      ],
69  ]
70
71  WSGI_APPLICATION = 'ecommerce.wsgi.application'
```

Configura o diretório de templates.

Senac



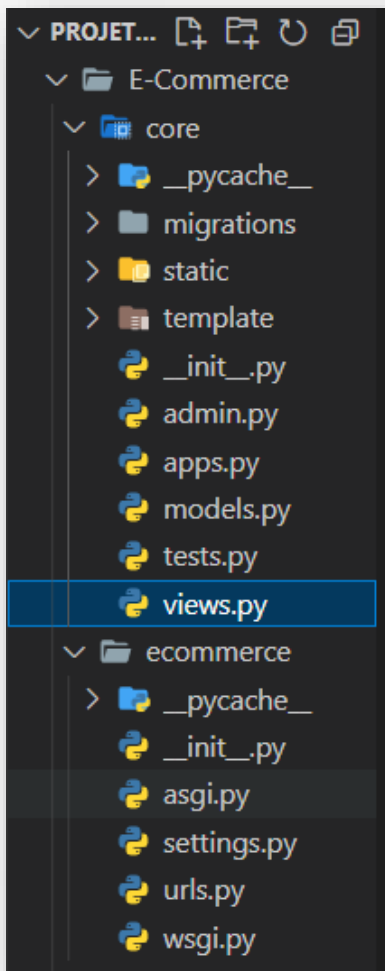
# Django: Estrutura de arquivos



```
settings.py X
E-Commerce > ecommerce > settings.py > ...
104 # Internationalization
105 # https://docs.djangoproject.com/en/4.0/topics/i18n/
106
107 LANGUAGE_CODE = 'pt-br'
108
109 TIME_ZONE = 'America/Sao_Paulo'
110
111 USE_I18N = True
112
113 USE_TZ = True
114
115
116 # Static files (CSS, JavaScript, Images)
117 # https://docs.djangoproject.com/en/4.0/howto/static-files/
118
119 STATIC_URL = 'static/'
120
121 # Default primary key field type
122 # https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field
123
124 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
125
```

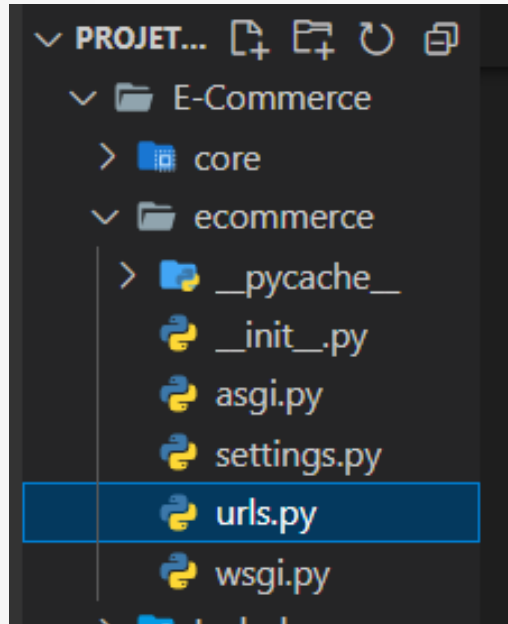
Habilita um idioma  
padrão, caso  
LANGUAGE\_CODE falhar

# Django: Estrutura de arquivos



```
views.py X  
E-Commerce > core > views.py > ...  
1  from django.shortcuts import render  
2  
3  # Create your views here.  
4  def index(request):  
5      return render(request, 'index.html')  
6  
7
```

# Django: Estrutura de arquivos

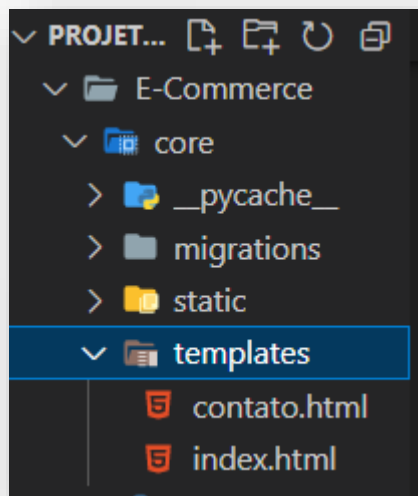


```
urls.py  X
E-Commerce > ecommerce > urls.py > ...
15
16 from django.contrib import admin
17 from django.urls import path
18 #from . import views
19 from core.views import index
20
21 urlpatterns = [
22     path('admin/', admin.site.urls),
23     path('', index)
24 ]
25
```

```
urls.py  X
E-Commerce > ecommerce > urls.py > ...
15
16 from django.contrib import admin
17 from django.urls import path, include
18 #from . import views
19 #from core.views import index
20
21 urlpatterns = [
22     path('admin/', admin.site.urls),
23     path('', include('core.urls'))
24 ]
25
```

# Django: Estrutura de arquivos

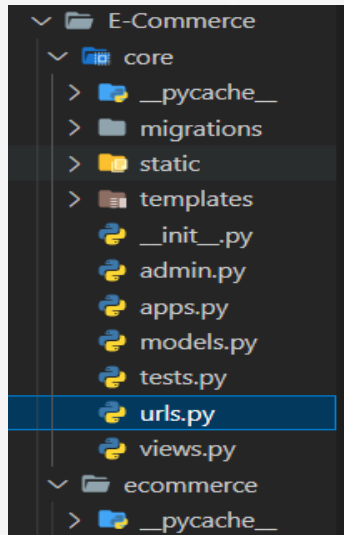
Criar diretório “**templates**”  
e arquivos HTML



```
index.html X
E-Commerce > core > templates > index.html > html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8 </head>
9 <body>
10   <h1>Página Inicial</h1>
11 </body>
12 </html>
```

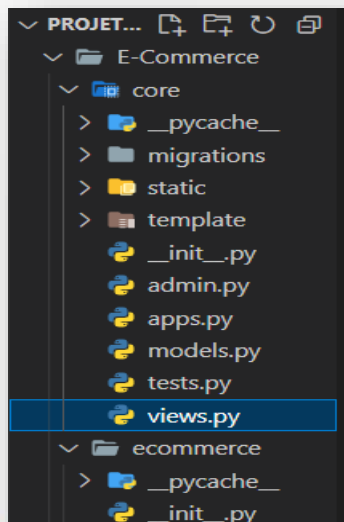
```
contato.html X
E-Commerce > core > templates > contato.html > html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8 </head>
9 <body>
10   <h1>Página contato</h1>
11   <p><strong>E-mail:</strong>{{ email }}</p>
12 </body>
13 </html>
```

# Django: Estrutura de arquivos



## Criar um arquivo urls.py para cada app

```
urls.py X
E-Commerce > core > urls.py > ...
1 from django.urls import path
2 from .views import index, contato #Acrescentar todas as viws do app
3 urlpatterns = [
4     path('', index),
5     path('contato', contato) #Acrescentar o path das views do app
6 ]
```

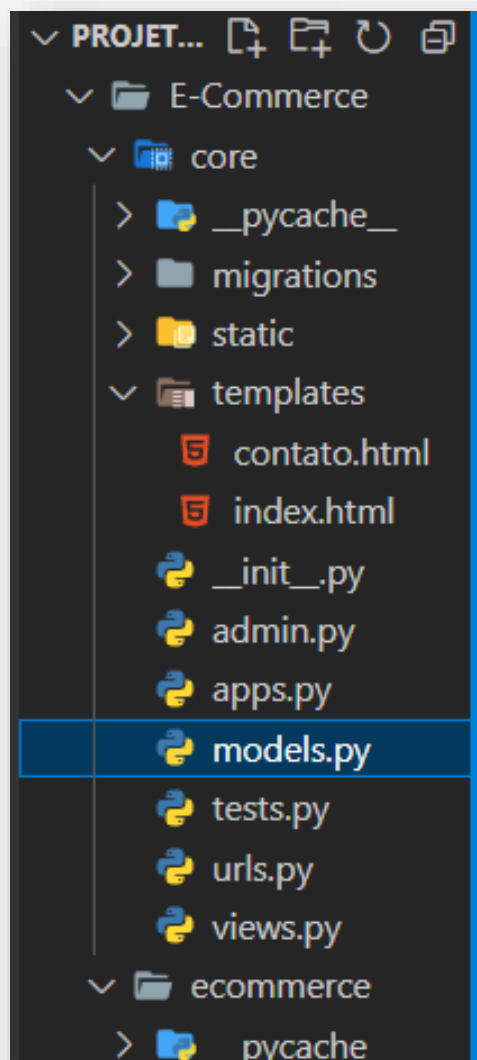


```
views.py X
E-Commerce > core > views.py > ...
2 from django.shortcuts import render
3
4 # Create your views here.
5 def index(request):
6     return render(request, 'index.html')
7
8 def contato(request):
9     context = {'email': 'contato@teste.com'}
10    return render(request, 'contato.html', context)
11
```





# Django: Estrutura de arquivos



```
models.py X  
E-Commerce > core > models.py > ...  
1  from django.db import models  
2  
3  # Create your models here.  
4  class Cliente(models.Model):  
5      nome = models.CharField('Nome', max_length=100)  
6      sobrenome = models.CharField('Sobrenome', max_length=100)  
7      email = models.EmailField('E-mail', max_length=100)  
8      def __str__(self):  
9          return f'{self.nome} {self.sobrenome}'  
10  
11 class Produto(models.Model):  
12     nome = models.CharField('Nome', max_length=100)  
13     preco = models.DecimalField('Preço', decimal_places=2, max_digits=8)  
14     estoque = models.IntegerField('Quantidade em Estoque')  
15     def __str__(self):  
16         return self.nome  
17
```

# Django: Criar e executar migrations

```
PROBLEMAS  SAÍDA  TERMINAL  JUPYTER  CONSOLE DE DEPURACÃO

Watching for file changes with StatReloader
Performing system check...

System check identified no issues (0 silenced).

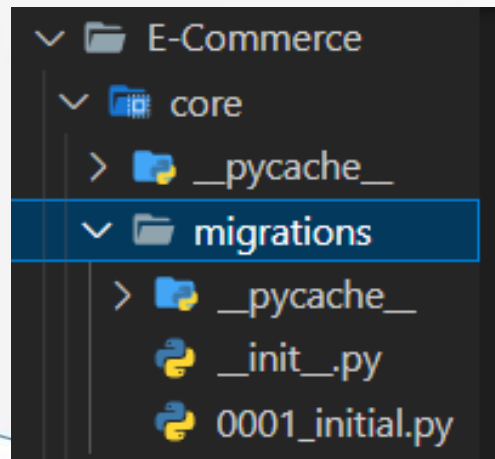
You have 18 unapplied migration(s). Your project may not work properly.
Run 'python manage.py migrate' to apply them.
July 30, 2022 - 00:50:51
Django version 4.0.6, using settings 'ecommerce.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

```
PROBLEMAS  SAÍDA  TERMINAL  JUPYTER  CONSOLE DE DEPURACÃO

D:\ProjetoExemplo\E-Commerce\core\models.py changed, reloading.
(E-Commerce) PS D:\ProjetoExemplo\E-Commerce> python manage.py makemigrations
Migrations for 'core':
  core\migrations\0001_initial.py
    - Create model Cliente
    - Create model Produto
(E-Commerce) PS D:\ProjetoExemplo\E-Commerce> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, core, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying core.0001_initial... OK
```

Comando para  
criar arquivos  
de migrations

Comando executar  
as atualizações de  
migrations



```
0001_initial.py ×
E-Commerce > core > migrations > 0001_initial.py > ...
1  # Generated by Django 4.0.6 on 2022-07-30 03:56
2
3  from django.db import migrations, models
4
5
6  class Migration(migrations.Migration):
7
8      initial = True
9
10     dependencies = [
11     ]
12
13     operations = [
14         migrations.CreateModel(
15             name='Cliente',
16             fields=[
17                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
18                 ('nome', models.CharField(max_length=100, verbose_name='Nome')),
19                 ('sobrenome', models.CharField(max_length=100, verbose_name='Sobrenome')),
20                 ('email', models.EmailField(max_length=100, verbose_name='E-mail')),
21             ],
22         ),
23     ]
```

# Django: Ambiente Administrativo

Comando executar as atualizações de migrations

Acessar | Site de administração

localhost:8000/admin/login/?next=...

### Administração do Django

Usuário:

Senha:

ACESSAR

```
PROBLEMAS  SAÍDA  TERMINAL  JUPYTER  CONSOLE DE DEPURAÇÃO
```

```
(E-Commerce) PS D:\ProjetoExemplo\E-Commerce> python manage.py createsuperuser
Usuário (leave blank to use 'leonardo'): leo
Endereço de email: leo@teste.com
Password:
Password (again):
Superuser created successfully.
(E-Commerce) PS D:\ProjetoExemplo\E-Commerce>
```

Administração do Site | Site de a

localhost:8000/admin/

### Administração do Django

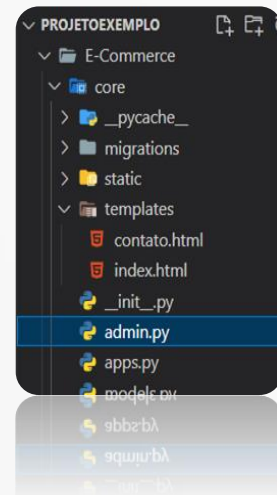
Administração do Site

AUTENTICAÇÃO E AUTORIZAÇÃO

Grupos	+ Adicionar	✎ Modificar
Usuários	+ Adicionar	✎ Modificar

CORE

Clientes	+ Adicionar	✎ Modificar
Produtos	+ Adicionar	✎ Modificar

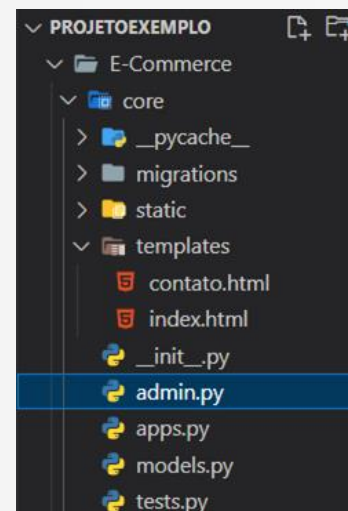


```
admin.py
```

```
E-Commerce > core > admin.py
```

```
1  from django.contrib import admin
2  from .models import Cliente, Produto
3  # Register your models here.
4  admin.site.register(Cliente)
5  admin.site.register(Produto)
6
```

# Django: Ambiente Administrativo



```
admin.py x
E-Commerce > core > admin.py > ...
1  from django.contrib import admin
2  from .models import Cliente, Produto
3  # Register your models here.
4
5  class ClienteAdmin(admin.ModelAdmin):
6      list_display=('nome', 'sobrenome', 'email')
7
8  class ProdutoAdmin(admin.ModelAdmin):
9      list_display=('nome', 'preco', 'estoque')
10
11  admin.site.register(Cliente, ClienteAdmin)
12  admin.site.register(Produto, ProdutoAdmin)
13
```

