

Django (1/4) – Introdução, Models, Views, Templates e Rotas

Back-End com Python

Conceito de MTV, Instalação e Ambientação Django

Definição de models e migrations

Funções de *view*

Django Template Language (DTL)

Contexto e renderização

Configuração de *URLConf*

Namespaces



PROGRAMA DE
FORMAÇÃO ACELERADA



MINISTÉRIO DA
CIÊNCIA, TECNOLOGIA
E INOVAÇÃO



Roteiro

- Características gerais do Django
- Arquitetura de Desenvolvimento Web Cliente - Servidor
- Conceito de MTV: *Model - Template – View*
- Passo a passo – Ambiente *VS Code*
- Instalação do Django
- Mínimo sistema Django (“Olá, Mundo”)
- Definição de models
- Criação e execução de *migrations*
- *Admin* automático

Roteiro

- Funções de *view*
- *Django Template Language* (DTL)
- Contexto e renderização
- Configuração de *URLConf*
- Namespaces
- Exercícios

Let's Django!



django

OVERVIEW

DOWNLOAD

DOCUMENTATION

NEWS

CODE

ISSUES

COMMUNITY

Documentation

Django documentation ¶

Everything you need to know about Django.

First steps ¶

Are you new to Django or to programming? This is the place to start!

- **From scratch:** [Overview](#) | [Installation](#)
- **Tutorial:** [Part 1: Requests and responses](#) | [Part 2: Models and the admin site](#) | [Part 3: Views and templates](#) | [Part 4: Forms and generic views](#) | [Part 5: Testing](#) | [Part 6: Static files](#) | [Part 7: Customizing the admin site](#) | [Part 8: Adding third-party packages](#)
- **Advanced Tutorials:** [How to write reusable apps](#) | [Writing your first contribution to Django](#)

Getting help ¶

Having trouble? We'd like to help!

- Try the [FAQ](#) – it's got answers to many common questions.
- Looking for specific information? Try the [Index](#), [Module Index](#) or the [detailed table of contents](#).
- Not found anything? See [FAQ: Getting Help](#) for information on getting support and asking questions to the community.



BOLSA
FUTURO
DIGITAL

PROGRAMA DE
FORMAÇÃO ACELERADA



cepedi



Softex

MINISTÉRIO DA
CIÊNCIA, TECNOLOGIA
E INOVAÇÃO

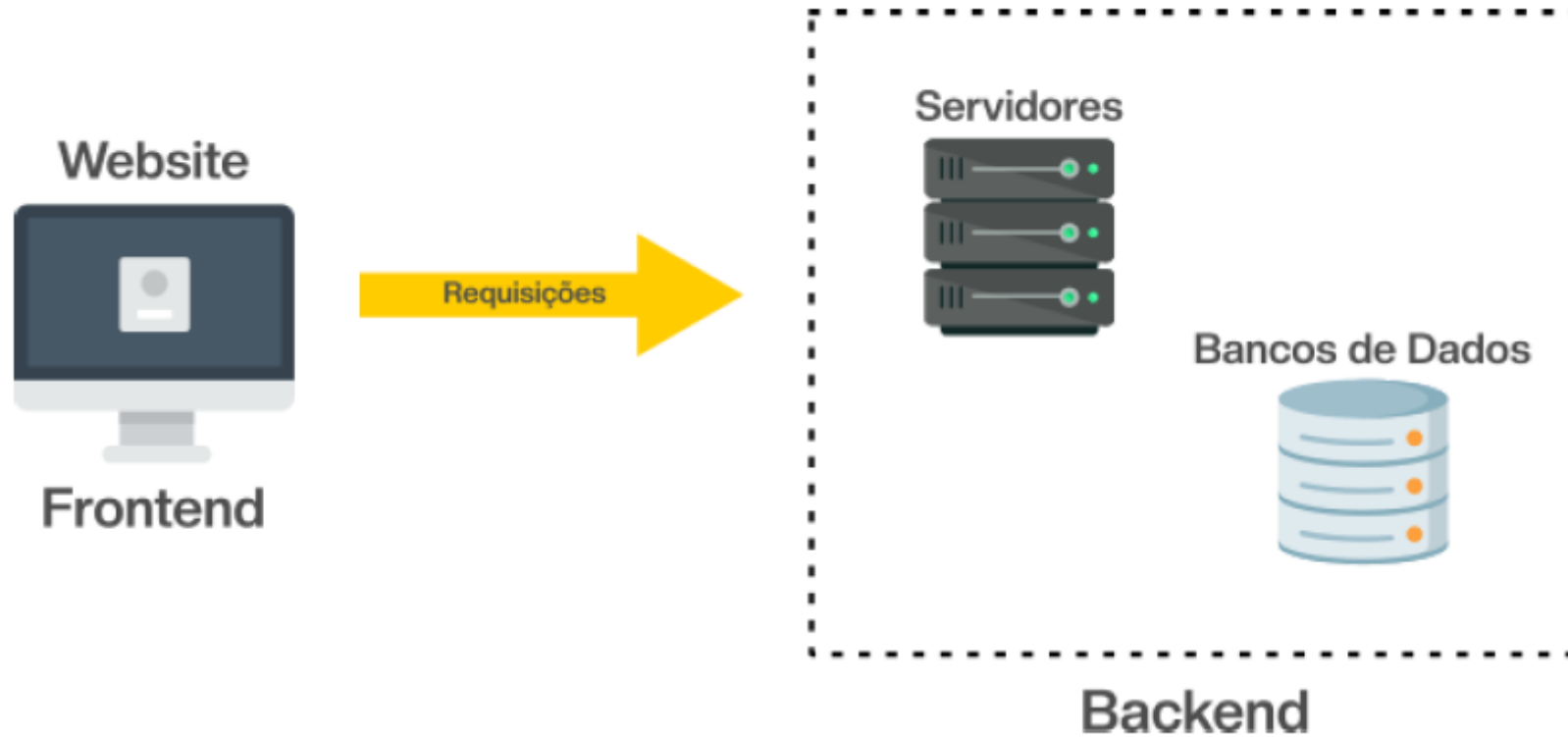


GOVERNO FEDERAL
BRASIL
UNIDO E RECONSTRUÇÃO

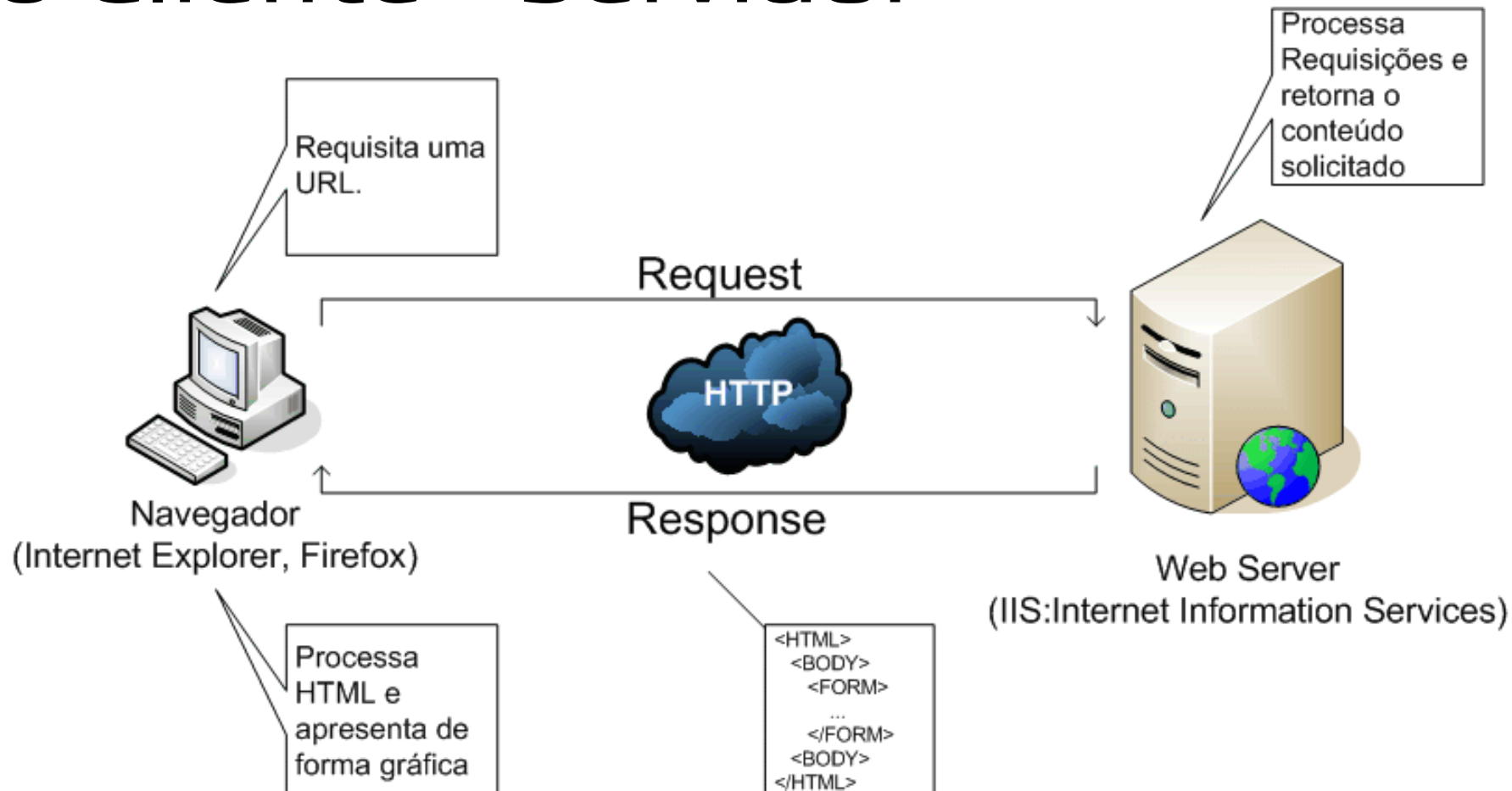
Características gerais do Django

- *Framework* Python de código aberto para desenvolvimento de sistemas web – ênfase no *backend*
- 20+ anos de mercado
- Produtividade DRY (não repetir código)
- ORM (*Object-Relational Mapping*)
- Segurança
- Escalabilidade
- Vasta biblioteca e documentação

Arquitetura de Desenvolvimento Web Cliente - Servidor



Arquitetura de Desenvolvimento Web Cliente - Servidor



BOLSA
FUTURO
DIGITAL

PROGRAMA DE
FORMAÇÃO ACELERADA



CEPEDI



Softex

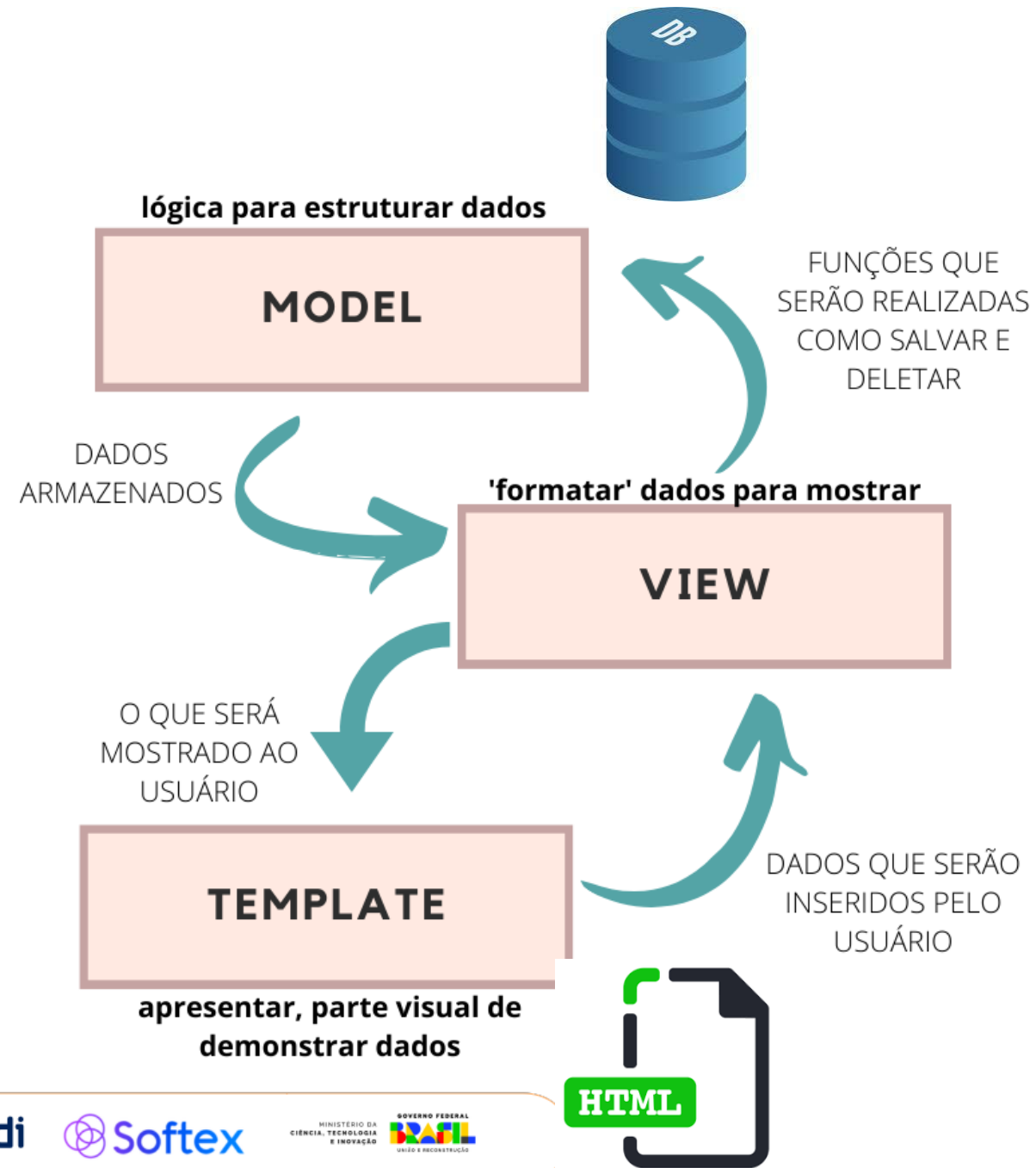
MINISTÉRIO DA
CIÊNCIA, TECNOLOGIA
E INOVAÇÃO



GOVERNO FEDERAL
BRASIL
UNIDO E RECONSTRUÇÃO

Arquitetura Django MTV

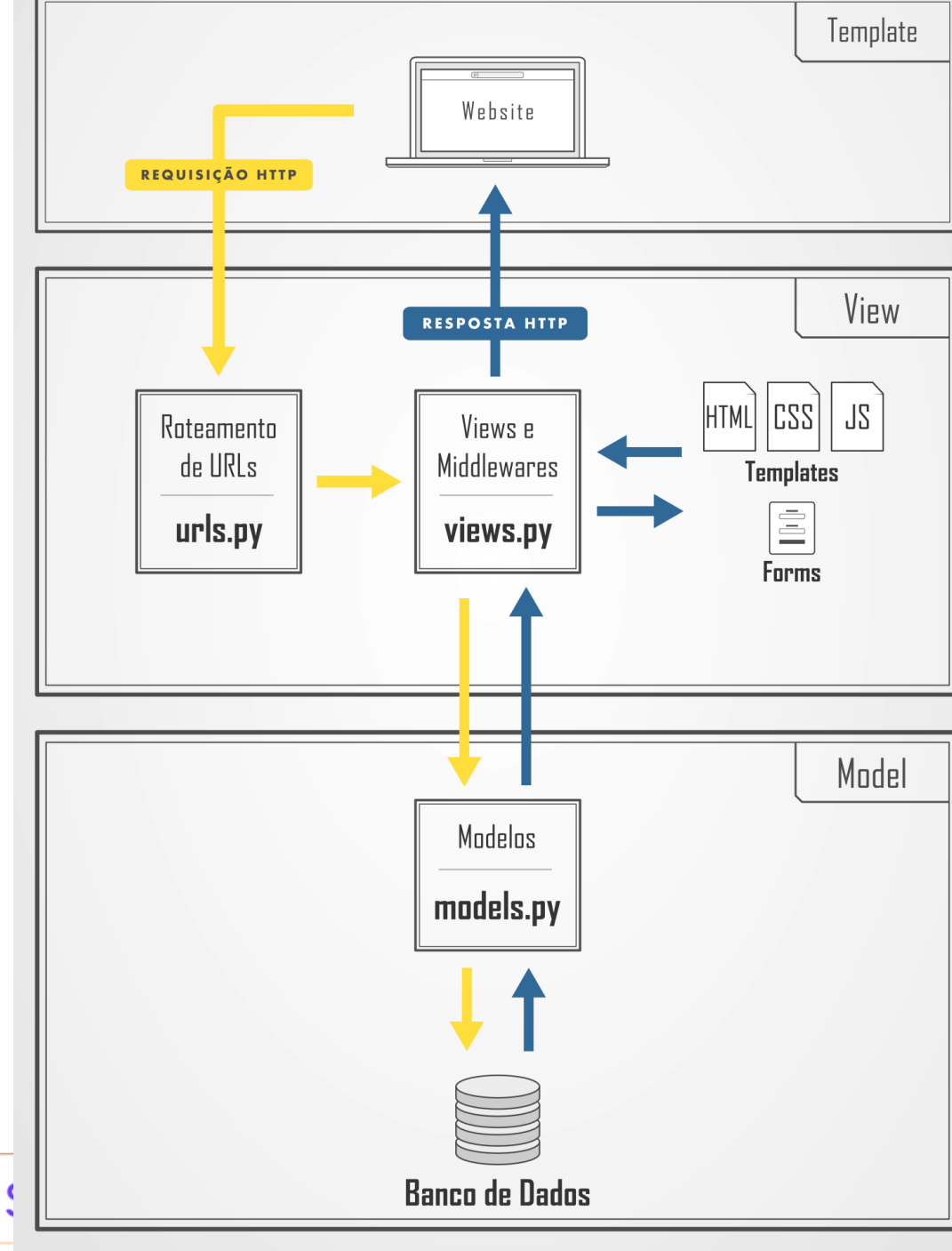
- **M (*Model*)**: Modelagem de entidades e manipulação de dados (Banco de Dados).
- **V (*View*)**: Lógica de processamento e resposta ao Cliente (Regras de negócio).
- **T (*Template*)**: Corresponde à camada de interface com o usuário – páginas *web* (html, css, js).



Arquitetura Django MTV

Fluxo de processamento de Requisição:

- Roteamento de URLs -> View -> Template
- ou*
- Roteamento de URLs -> View -> Model -> Template



Passo a Passo - Ambiente VS Code

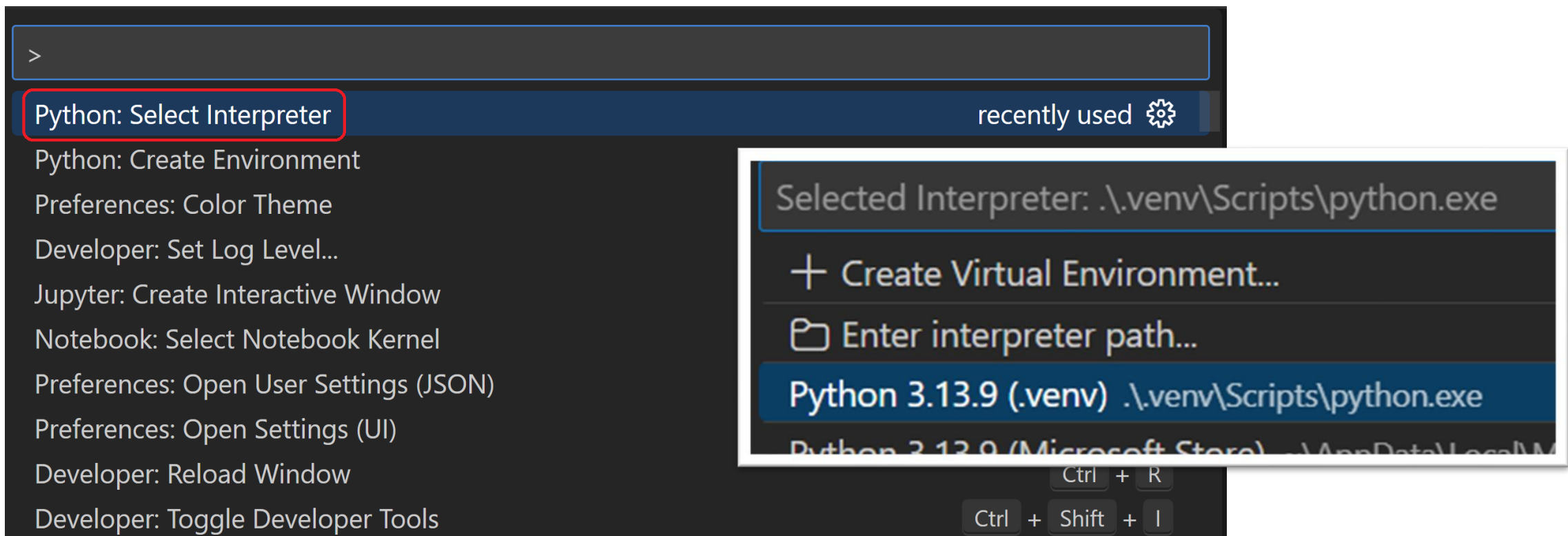
- File -> “Open Folder” (abrir diretório do projeto)
- “Ctrl +J” -> Abrir Terminal
- Recomendação inicial: criação de ambiente virtual Python

```
py -3 -m venv .venv  
.venv\scripts\activate
```

Alternativa (dependendo da variável de ambiente da instalação do Python):
python -m venv .venv

Passo a Passo - Ambiente VS Code

- Se necessário: **Ctrl+Shift+P**



Passo a Passo - Ambiente VS Code

- Atualização do **pip** (“*Pip Installs Packages*”): módulo python responsável por gerenciar bibliotecas e dependências externas

```
python -m pip install --upgrade pip
```

Passo a Passo - Ambiente VS Code

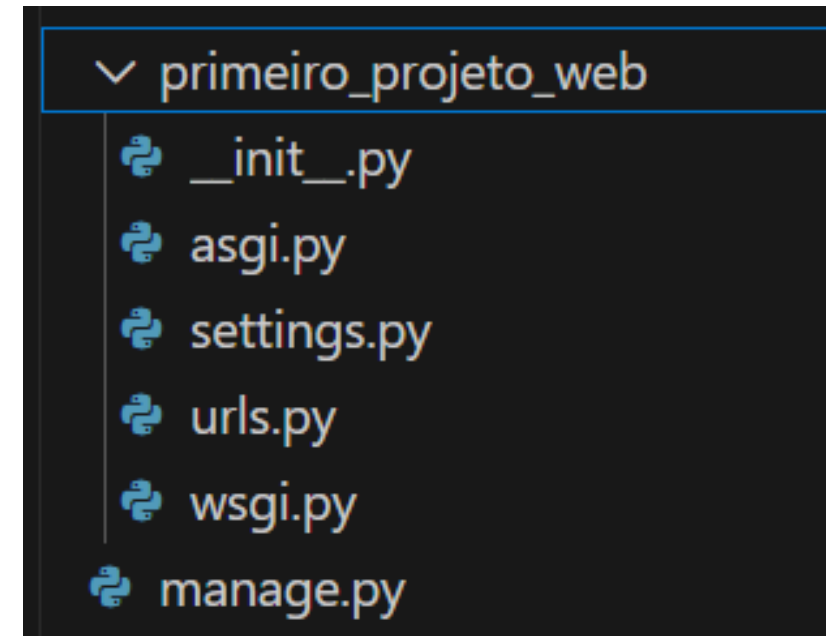
- Instalação do Django

```
pip install django
```

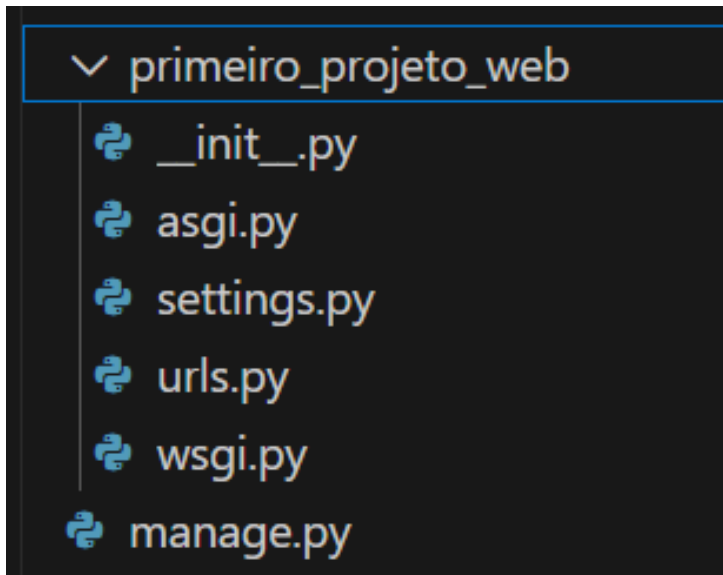
Mínimo sistema Django (“Olá, Mundo”)

- django-admin **startproject**

django-admin **startproject** primeiro_projeto_web .



Mínimo sistema Django (“Olá, Mundo”)



- *asgi.py* e *wsgi.py*: interfaces de comunicação com o servidor
- *settings.py*: configurações gerais do projeto, banco de dados, aplicativos instalados, etc.
- *urls.py*: Gerenciamento e configuração de roteamento de URLs (URLConf).
- *manage.py*: interface de linha de comando para atividades de gerência e manutenção da aplicação.

```
python manage.py <command> [options]
```

Mínimo sistema Django (“Olá, Mundo”)

- Rodando o servidor local

`python manage.py runserver`

acesse no navegador web:
<http://localhost:8000>.

“Ctrl +c” para parar o servidor



The install worked successfully! Congratulations!

View [release notes](#) for Django 6.0

You are seeing this page because `DEBUG=True` is in your settings file and you have not configured any URLs.

Mínimo sistema Django (“Olá, Mundo”)

- Criando um **app**

App: “Uma aplicação Web que faz alguma coisa, por exemplo - um blog, um banco de dados de registros públicos ou um aplicativo de pesquisa. Já um projeto é uma coleção de configurações e apps para um website em particular.”

Um **projeto** pode ter vários **apps** e um **app** pode estar presente em diversos **projetos**.

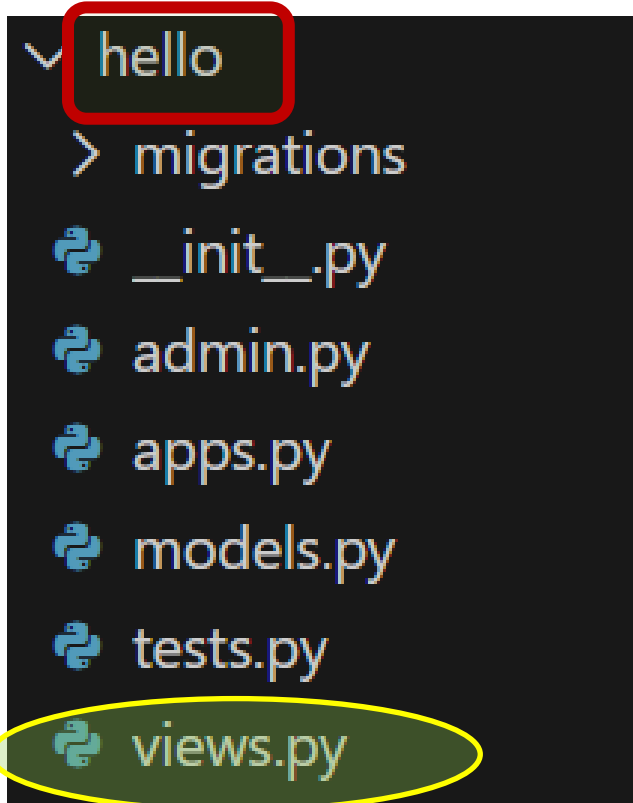
Mínimo sistema Django (“Olá, Mundo”)

- `python manage.py startapp`

`python manage.py startapp hello`

```
✓ hello
  > migrations
  🐍 __init__.py
  🐍 admin.py
  🐍 apps.py
  🐍 models.py
  🐍 tests.py
  🐍 views.py
```

Mínimo sistema Django (“Olá, Mundo”)



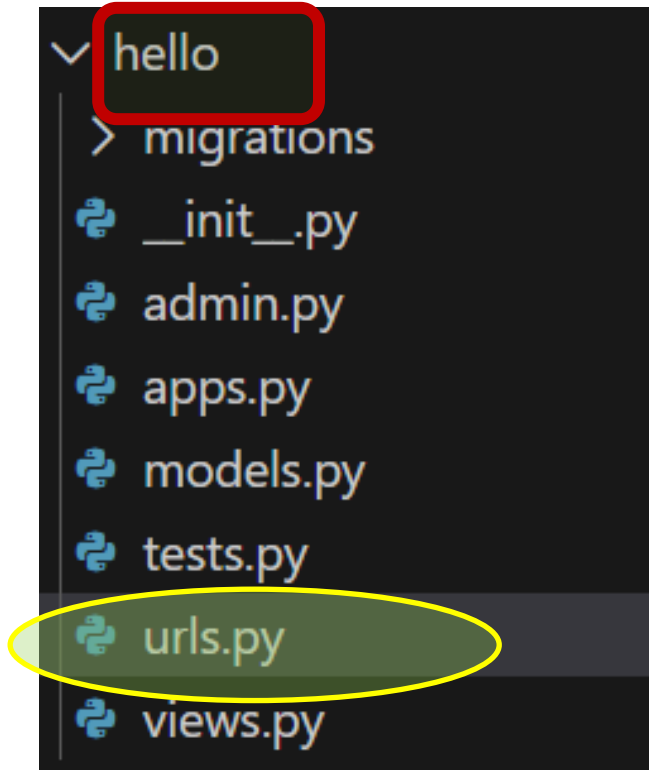
- **views.py**

```
from django.http import HttpResponse

def home(request):
    return HttpResponse("Olá, Mundo!")
```

Mínimo sistema Django (“Olá, Mundo”)

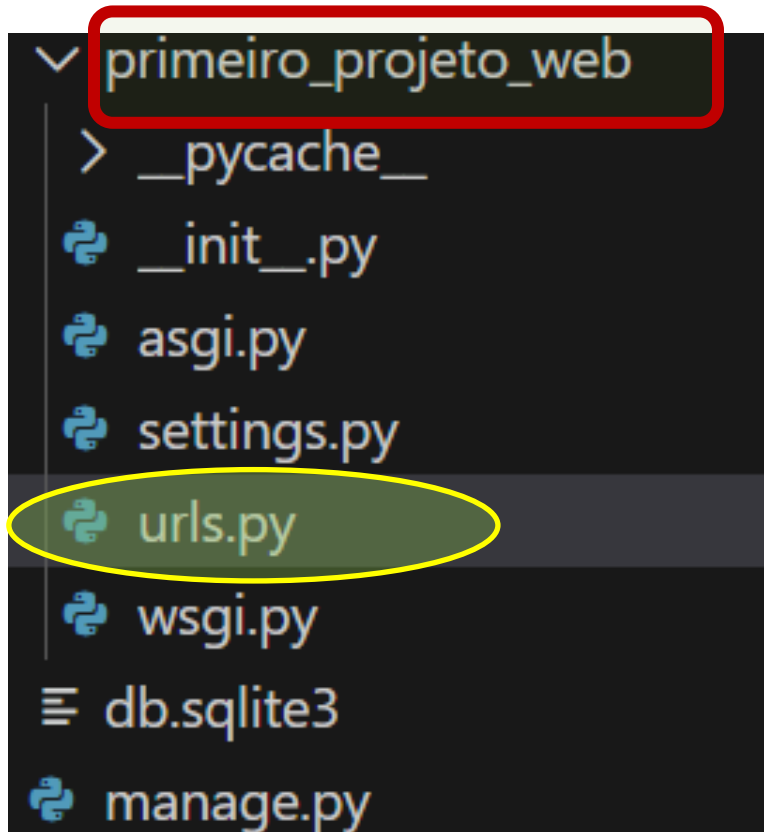
- Crie arquivo “hello/urls.py”



```
from django.urls import path
from hello import views

urlpatterns = [
    path("", views.home, name="home"),
]
```

Mínimo sistema Django (“Olá, Mundo”)



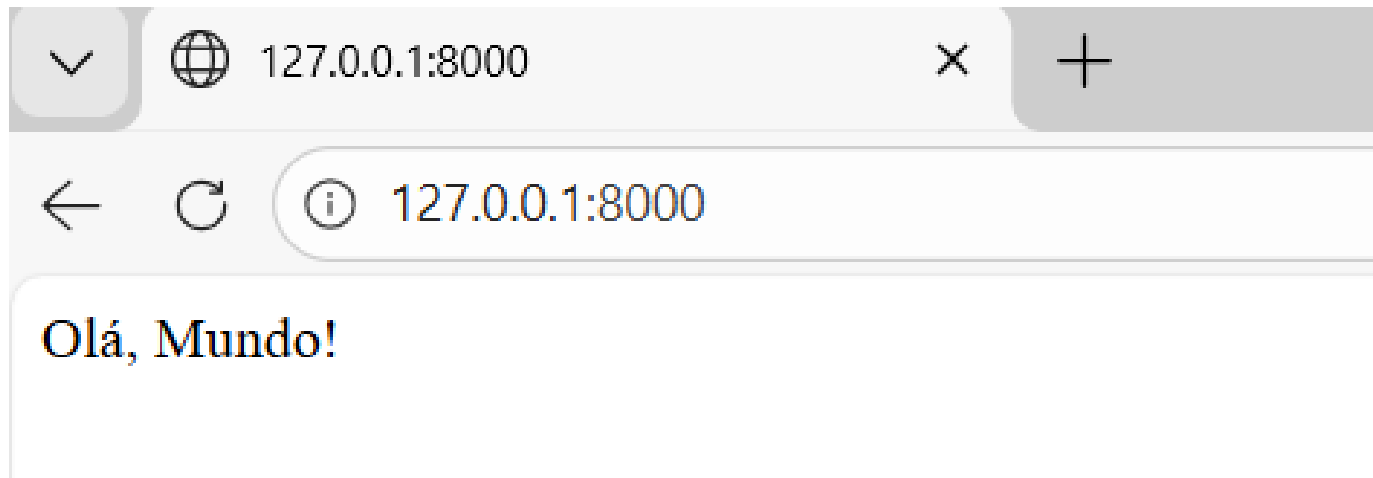
- Inclua na lista de urls do projeto “primeiro_projeto_web/urls.py”

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('', include('hello.urls')),
    path('admin/', admin.site.urls),
]
```

Mínimo sistema Django (“Olá, Mundo”)

- Execute o **servidor** e atualize o **navegador**



Models (Camada de Modelos)

- Definição de entidades
- Mapeamento direto para tabelas do banco de dados
- BD Padrão: SQLite



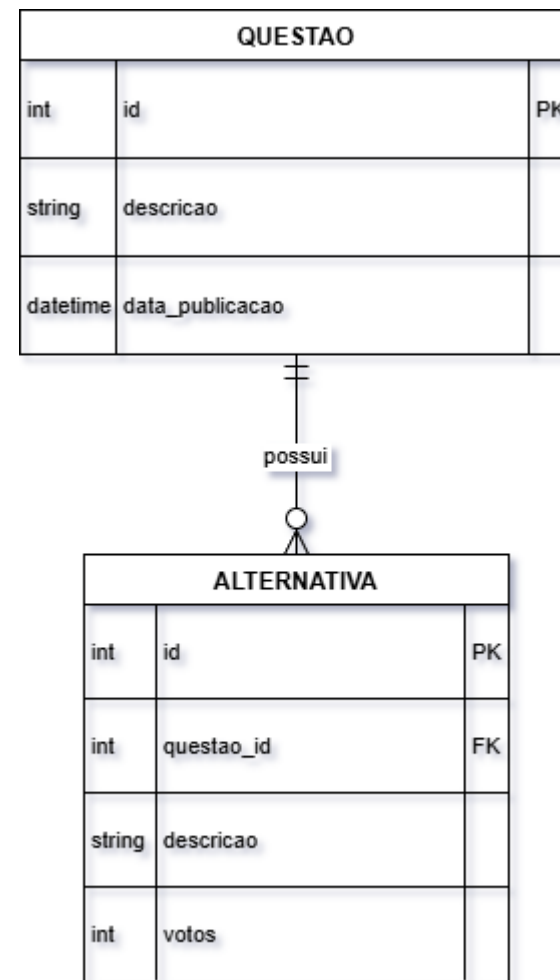
MODEL

settings.py

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.sqlite3',  
        'NAME': BASE_DIR / 'db.sqlite3',  
    }  
}
```

Models (Camada de Modelos)

- Vamos criar um novo App (**enquetes**)



Models (Camada de Modelos)

- <seguir etapas anteriores (**app hello**)>
- incluir nome do app na lista de app instalados (**settings.py**):

```
primeiro_projeto_web > 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     'hello',  
41     'enquetes',  
42 ]
```

```
LANGUAGE_CODE = 'pt-br'  
TIME_ZONE = 'America/Bahia'
```

Models (Camada de Modelos)

- Definir entidades em “*models.py*”
- Cada Classe deve herdar de (*models.Model*)

```
s > models.py > ...  
from django.db import models  
class Questao(models.Model):  
    descricao = models.CharField(max_length=200)  
    data_pub = models.DateTimeField("data de publicação")  
  
class Alternativa(models.Model):  
    questao = models.ForeignKey(Questao, on_delete=models.CASCADE)  
    descricao = models.CharField(max_length=200)  
    votos = models.IntegerField(default=0)
```

Migrations (migração)

*“Migração é a forma do Django de **propagar as alterações** feitas em seu modelo (adição de um novo campo, deleção de um modelo, etc...) ao seu esquema do banco de dados. Elas foram desenvolvidas para serem (na maioria das vezes) **automáticas**, mas cabe a você saber a hora de fazê-las, executá-las e resolver os problemas comuns que possam vir a acontecer”. Django docs*

Migrations (migração)

- Preparação da migração:

python manage.py makemigrations

```
Migrations for 'enquetes':
  enquetes\migrations\0001_initial.py
    + Create model Questao
    + Create model Resposta
```

- Aplicação da migração:

python manage.py migrate

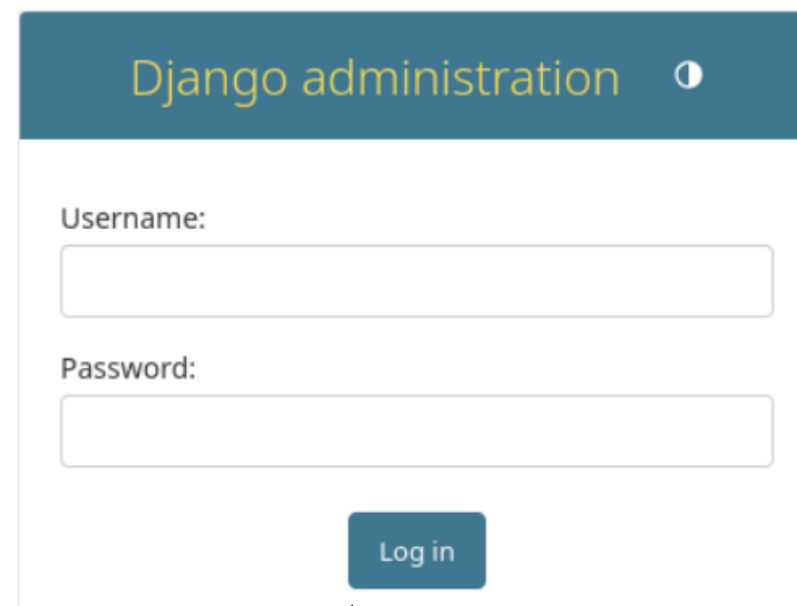
```
Running migrations:
  Applying enquetes.0001_initial... OK
```

Admin automático

- O Django automatiza a criação de interfaces administrativas para os modelos:

- ☒ Create (Adicionar)
- ☒ Read (Visualizar)
- ☒ Update (Alterar)
- ☒ Delete (Excluir)

- Interface unificada **apenas** para os gerentes do sistema administrarem o conteúdo (dados)


A screenshot of the Django administration interface. At the top, there is a dark blue header bar with the text "Django administration" in a light green font, followed by a small circular icon. Below the header, the page has a white background. It features two input fields: "Username:" and "Password:", each with a corresponding text box. Below these fields is a blue button labeled "Log in".

Admin automático

- Criar superuser

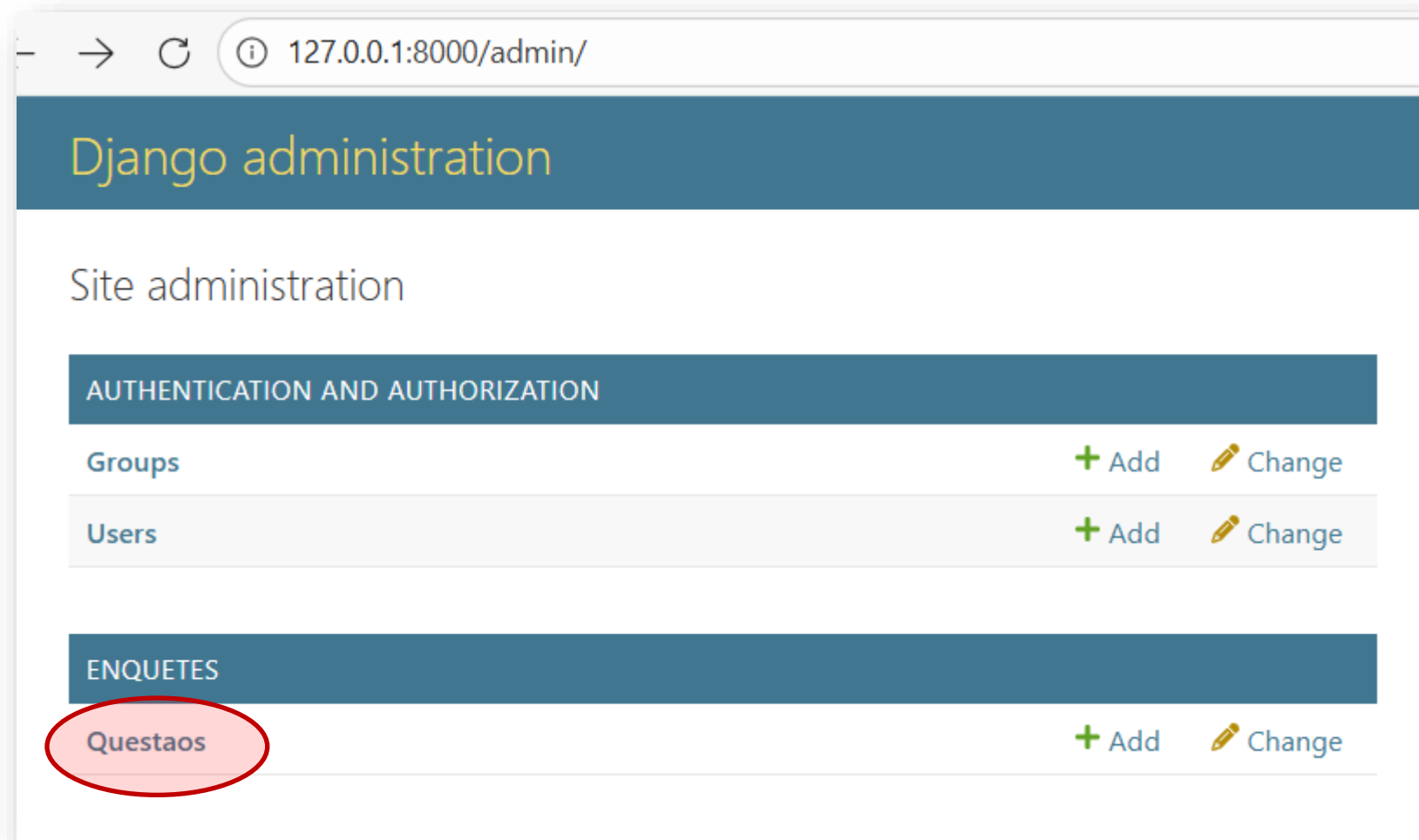
python manage.py createsuperuser

- Registrar modelo(s) em “admin.py”:

```
enquetes >  admin.py
1  from django.contrib import admin
2  from .models import Questao, Alternativa
3
4  admin.site.register(Questao)
5  admin.site.register(Alternativa)
```

Admin automático

- De onde veio “Questaos”??



Models: Sobrescrevendo atributos da classe interna **Meta**

- Opcionalmente, podemos reconfigurar alguns atributos de **metadados** da classe interna Meta (Model), tais como:
 - **db_table**: nome da tabela no banco de dados
 - **ordering**: opções de ordenação
 - **verbose_name** e **verbose_name_plural**: nome do modelo (no singular e no plural) legível para humanos **(esses são os nomes que vão aparecer no Admin!)**

Models: Sobrescrevendo atributos da classe interna **Meta**

```
from django.db import models
from django.utils import timezone

class Questao(models.Model):
    descricao = models.CharField(max_length=200)
    data_pub = models.DateTimeField("data de publicação")

    def __str__(self):
        return self.descricao

    def foi_publicada_recentemente(self):
        agora = timezone.now()
        return agora - timezone.timedelta(days=1) <= self.data_pub <= agora

class Meta:
    verbose_name = "Questão"
    verbose_name_plural = "Questões"
```

Camada View



- Definição e lógica de interação e respostas às URLs
 - **Página “inicial” de questões**: exibe as perguntas mais recentes.
 - Página de “**detalhes**” da pergunta: exibe o texto de uma pergunta, sem os resultados, mas com um formulário para votar.
 - **Ação de voto** – registra um voto em uma alternativa específica de uma dada questão.
 - Página de “**resultados**” da pergunta: exibe os resultados de uma pergunta específica.

Cada view é representada por uma **Função de View** em “views.py”

Funções de View

enquetes >  views.py > ...

```
2
3 def index(request):
4     return HttpResponse("Bem-Vindo, ao Enquetes!")
5
6 def detalhe(request, questao_id):
7     return HttpResponse("Você está vendo a Pergunta %s." % questao_id)
8
9 def voto(request, questao_id):
10    return HttpResponse("Você está votando na Pergunta %s." % questao_id)
11
12 def resultados(request, questao_id):
13    response = "Você está vendo os resultados da Pergunta %s."
14    return HttpResponse(response % questao_id)
```

Funções de View

- Associar cada Função de View a uma URL no arquivo “**urls.py**”.

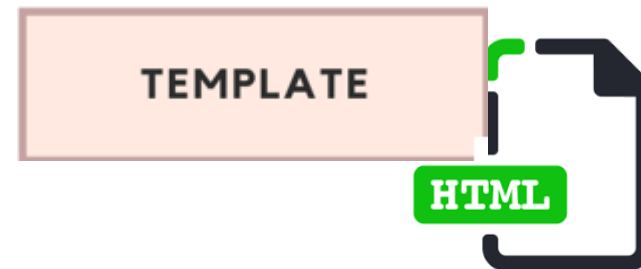
```
urlpatterns = [  
    path("", views.index, name="index"),  
    # ex: /enquetes/5/  
    path("<int:questao_id>/", views.detalhe, name="detalhe"),  
    # ex: /enquetes/5/voto/  
    path("<int:question_id>/vote/", views.voto, name="voto"),  
    # ex: /enquetes/5/resultados/  
    path("<int:question_id>/results/", views.resultados, name="resultados"),  
]
```

Funções de View

- Melhorando a Função de Visualização “index”.

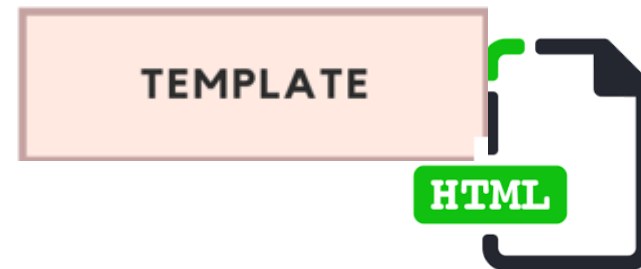
```
def index(request):  
    #return HttpResponse("Bem-Vindo, ao Enquetes!")  
    lista_questoes_recentes = Questao.objects.order_by("-data_pub")[:5]  
    resposta = ", ".join([q.descricao for q in lista_questoes_recentes])  
    return HttpResponse(resposta)
```

Camada Template



- Separação do “*design*” (interface) da lógica e dos dados
- *Django Template Language* (DTL)
 - ✓ Permite a inserção de lógica (variáveis, condicionais, loops, etc) e manipulação de dados dentro do html

Camada Template



- Padrão recomendado: Criar **diretório** no caminho do **app** no seguinte formato:

“enquetes/templates/enquetes/”

* Assim, podemos referenciar o template “**index.html**” da seguinte forma: **enquetes/index.html** e o Django não confundirá com outro “index.html” de outra aplicação

Primeiro template: index.html

```
{% if lista_questoes_recentes %}
    <ul>
    {% for questao in lista_questoes_recentes %}
        <li><a href="/enquetes/{{questao.id}}/">{{ questao.descricao }}</a></li>
    {% endfor %}
    </ul>
{% else %}
    <p>Nenhuma enquete disponível.</p>
{% endif %}
```


Primeiro template: index.html

Versão HTML
completo

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Bem - Vindo, ao Enquetes</title>
</head>
<body>
  <header>
    <h1>Portal de Enquetes</h1>
  </header>
  <main>
    {% if lista_questoes_recentes %}
    <ul>
      {% for questao in lista_questoes_recentes %}
      <li><a href="/enquetes/{{questao.id }}">{{ questao.descricao }}</a></li>
      {% endfor %}
    </ul>
    {% else %}
    <p>Nenhuma enquete disponível.</p>
    {% endif %}
  </main>
  <footer>
    <p>&copy; 2025 Futuro Digital</p>
  </footer>
</body>
</html>
```



Um atalho para a view: *render()*

- Melhorando a Função de Visualização “**index**”. (parte 2)

```
from django.shortcuts import render
def index(request):
    lista_questoes_recentes = Questao.objects.order_by("-data_pub")[:5]
    context = {"lista_questoes_recentes": lista_questoes_recentes}
    return render(request, "enquetes/index.html", context)
```

- Menos código, mais direto, padrão do mercado de desenvolvimento.

Lançando o erro 404 com *get_object_or_404()*

- Esta função “automatiza” a busca por um objeto ou o lançamento da exceção 404

```
def detalhe(request, questao_id):  
    questao = get_object_or_404(Questao, pk=questao_id)  
    return render(request, "enquetes/detalhe.html", {"questao": questao})
```

Mais um template

“detalhe.html”



```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Bem - Vindo, ao Enquetes</title>
  </head>
  <body>
    <header>
      <h1>Portal de Enquetes - Detalhe</h1>
    </header>
    <main>
      <h1>{{ questao.descricao }}</h1>
      <ul>
        {% for resposta in questao.resposta_set.all %}
          <li>{{ resposta.descricao }}</li>
        {% endfor %}
      </ul>
    </main>
    <footer>
      <p>&copy; 2025 Futuro Digital</p>
    </footer>
  </body>
</html>
```

Removendo urls fixas no template

- Utilização da tag de template django `{% url %}`

- “index.html”: substituir

```
<li><a href="/enquetes/{{questao.id}}/">{{ questao.descricao }}</a></li>
```

- Por:

```
<li><a href="{% url 'detalhe' questao.id %}">{{ questao.descricao }}</a></li>
```

* isso permite respeitar o único ponto de alteração/especificação de url (“urls.py”)

Namespaces

- Melhorando a especificação da rota no “**urls.py**”.

```
enquetes > 📄 urls.py > ...  
1  ∨ from django.urls import path  
2    from enquetes import views  
3  
4  app_name = "enquetes"  
5  ∨ urlpatterns = [
```

* o **app_name** diferencia eventuais nomes iguais de templates entre diversos **app** do projeto

- Atualizando no template

```
{% if lista_questoes_recentes %}  
<ul>  
    {% for questao in lista_questoes_recentes %}  
    <li><a href="{% url 'enquetes:detalhe' questao.id %}">{{ questao.descricao }}</a></li>  
    {% endfor %}  
</ul>  
{% else %}  
<p>Nenhuma enquete disponível.</p>  
{% endif %}
```

Extra...

VSCode Extensões úteis - Django

- SQLite (alexcvzz)
 - ✓ Visualização do BD padrão SQLite
- Django (Baptiste Darthenay)
 - ✓ Suporte ao autocompletar do “django-html”
- Django Template Support (junstyle)
 - ✓ Formatação e autocompletar (“shift+alt+f”)
- IntelliSense for CSS class names in HTML (natecorkish)
 - ✓ Formatação e autocompletar html/css/bootstrap

Mais sobre templates html

- Organização de arquivo “base.html”
- Utilizando componentes *bootstrap*
- Tags de controle
 - *{% extends %}*
 - *{% block content %} ... {% endblock %}*
 - *{% load static %}*

templates/enquetes/
"base.html"

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>
      {% block title %}                Meu Projeto Enquetes                {% endblock %}
    </title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet" />
  </head>
  <body>
    <header>
      <nav class="navbar navbar-dark bg-dark">
        <div class="container">
          <a class="navbar-brand" href="{% url 'enquetes:index' %}">Enquetes</a>
        </div>
      </nav>
    </header>
    <main class="container mt-4">
      {% block content %}
      {% endblock %}
    </main>
    <footer class="container mt-5 border-top pt-3">
      <p class="text-muted small">&copy; 2026 - Futuro Digital</p>
    </footer>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
  </body>
</html>
```

```
{% extends 'enquetes/base.html' %}
{% block content %}
    {% if lista_questoes_recentes %}
        <ol class="list-group list-group-numbered">
            {% for questao in lista_questoes_recentes %}
                <li class="list-group-item">
                    <a href="{% url 'enquetes:detalhe' questao.id %}">{{ questao.descricao }}</a>
                </li>
            {% endfor %}
        </ol>
    {% else %}
        <p>Nenhuma enquete disponível ainda.</p>
    {% endif %}
{% endblock %}
```

templates/enquetes/
"index.html"

templates/enquetes/
"detalhe.html"

```
{% extends 'enquetes/base.html' %}
{% block content %}
    <h1>{{ questao.descricao }}</h1>
    <ul>
        {% for resposta in questao.resposta_set.all %}
            <li>{{ resposta.descricao }}</li>
        {% endfor %}
    </ul>
{% endblock %}
```



BOLSA
FUTURO
DIGITAL

PROGRAMA DE
FORMAÇÃO ACELERADA



INSTITUTO
FEDERAL
Rio de Janeiro



cepedi



Softex

MINISTÉRIO DA
CIÊNCIA, TECNOLOGIA
E INOVAÇÃO



GOVERNO FEDERAL
UNIDO E RECONSTRUÇÃO

REFERÊNCIAS COMPLEMENTARES

- Visual Studio Code. **Django Tutorial in Visual Studio Code.**
Disponível em: <https://code.visualstudio.com/docs/python/tutorial-django>.
- Django Software Foundation. **Getting started with Django.**
Disponível em: <https://www.djangoproject.com/start/>.
- W3Schools. **Bootstrap 5 Tutorial.** Disponível em:
<https://www.w3schools.com/bootstrap5/>.