Framework Django

Disciplina: Backend
Docente: Ma. Karoline Farias

Sumário

- Configurar ambiente Django
- Criar um projeto e um app
- Exibir um Hello World no navegador
- Implementar CRUD completo para modelo Produto (Inserção, Listagem, Edição e Remoção)

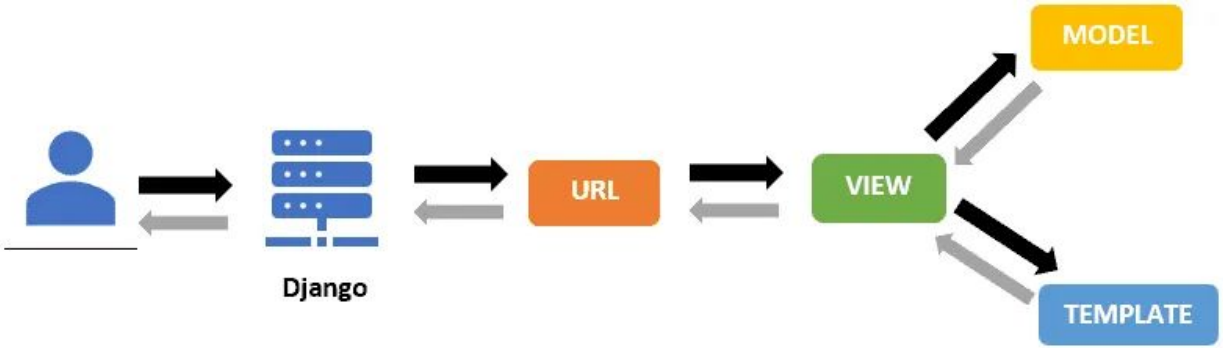
Introdução à Arquitetura do Django

O Django segue o padrão arquitetural **MVT (Model-View-Template)**, uma variação do conhecido padrão **MVC (Model-View-Controller)**. Esta arquitetura permite uma separação clara entre a lógica de negócios, a apresentação de dados e a interação do usuário.

Introdução à Arquitetura do Django

A arquitetura MVT do Django é composta por:

- **Model (Modelo)**: Define a estrutura de dados e a lógica de negócios
- **View (Visão)**: Processa as requisições do usuário e retorna respostas
- **Template (Modelo de Apresentação)**: Define como os dados serão apresentados ao usuário



Introdução à Arquitetura do Django

Além desses componentes, o Django também utiliza:

- **URLs**: Mapeiam as requisições para as views apropriadas
- **Forms**: Gerenciam formulários, validação de dados e conversão de tipos
- **Middleware**: Classes intermediárias que processam requisições e respostas globalmente (autenticação, segurança, logging).
- **Admin**: Interface de administração automática

Fluxo de uma Requisição

1. O navegador faz uma **requisição HTTP para uma URL** específica.
2. O **URL Dispatcher procura um padrão compatível em urls.py** e invoca a view associada.
3. **A view executa lógica de negócio, acessa o banco via ORM** e prepara um contexto de dados.
4. **A view chama render() para aplicar um template**, gerando HTML dinâmico.
5. **O Django retorna a resposta HTTP** ao cliente para exibição no navegador.

Configuração do Ambiente

1. Instalar Python 3.10+
2. Criar e ativar ambiente virtual:

```
python -m venv venv
source venv/bin/activate       # Linux/Mac
venv\\Scripts\\activate      # Windows
```

3. Instalar o Django:

```
pip install django
```

Criando o Projeto e o App

1. Criar projeto Django:

   ```
   django-admin startproject projetoti .
   ```

2. Criar app chamado `hello_world`:

   ```
   python manage.py startapp hello_world
   ```

3. Adicionar `hello_world` em INSTALLED_APPS no **projetoti/settings.py**

   ```
   INSTALLED_APPS = [
       # ... apps padrão
       hello_world,
   ]
   ```

Exibindo "Hello World": Criar View

No arquivo **hello_world/views.py**:

```python
from django.http import HttpResponse

def hello_world(request):
    return HttpResponse('Hello World! Este é meu primeiro view em Django.')
```

Exibindo "Hello World": Configurar URL

Criar arquivo **hello_world/urls.py**:

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

urlpatterns = [
    path('hello/', hello_world, name='hello_world'),
]
```

Exibindo "Hello World": Configurar URL

Criar arquivo **projetoti/urls.py:**

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

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include(hello_world.urls')),

]
```

Testar:

```
python manage.py runserver
```

Acessar http://127.0.0.1:8000/hello/ e verificar mensagem.

Criando o Projeto - Loja de Produtos

1. Criar e ativar ambiente virtual:

```
python -m venv venv
source venv/bin/activate
# Linux/Mac
venv\\Scripts\\activate
# Windows
```

2. Instalar o Django:

```
pip install django
```

3. Criar projeto Django:

```
django-admin startproject
loja_produtos .
```

4. Criar app chamado produtos:

```
python manage.py
startapp produtos
```

5. Adicionar produtos em INSTALLED_APPS no **loja_produtos/settings.py**

```
INSTALLED_APPS = [
    # ... apps padrão
    produtos,
]
```

CRUD para o Modelo Produto: Definindo o Modelo

Em **produtos/models.py**:

```python
from django.db import models

class Produto(models.Model):
    nome = models.CharField(max_length=100)
    descricao = models.TextField(blank=True)
    preco = models.DecimalField(max_digits=8, decimal_places=2)
    estoque = models.PositiveIntegerField()

    def __str__(self):
        return self.nome
```

CRUD para o Modelo Produto: Definindo o Modelo

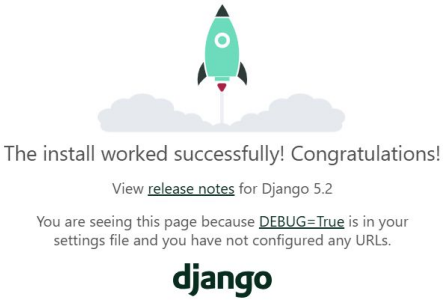
Django oferece vários tipos de campos para definir diferentes tipos de dados:

- **CharField:** Texto de tamanho limitado
- **TextField:** Texto sem limite de tamanho
- **IntegerField:** Números inteiros
- **FloatField:** Números decimais
- **BooleanField:** Valores True/False
- **DateTimeField:** Data e hora
- **ForeignKey:** Relação muitos-para-um
- **ManyToManyField:** Relação muitos-para-muitos
- **OneToOneField:** Relação um-para-um
- **EmailField:** Campo para e-mails com validação
- **URLField:** Campo para URLs com validação
- **FileField/ImageField:** Campos para upload de arquivos/imagens

CRUD para o Modelo Produto: Definindo o Modelo

Criando e Aplicando Migrações

```
python manage.py makemigrations
python manage.py migrate
```



ORM do Django

Exemplos práticos usando o modelo Produto

1. Acessando o Shell

```
python manage.py shell
>>> from produtos.models import Produto
```

2. Criando Registros

```
>>> Produto.objects.create(
...     nome='Caneca', descricao='Caneca de cerâmica branca',
...     preco=29.90, estoque=100
... )
```

ORM do Django

Exemplos práticos usando o modelo Produto

3. Listagem de Objetos

```
>>> Produto.objects.all()
<QuerySet [<Produto: Caneca>, <Produto: Livro>, ...]>
```

4. Filtragem

```
>>> Produto.objects.filter(preco__gt=50)
>>> Produto.objects.get(pk=1)
```

ORM do Django

Exemplos práticos usando o modelo Produto

3. Atualização de Objetos

```
>>> p = Produto.objects.get(pk=1)
>>> p.preco = 24.90
>>> p.save()
```

4. Exclusão de Objetos

```
>>> p.delete()
```

CRUD para o Modelo Produto: Definindo o Modelo

Registrando no Admin (opcional)

Em **produtos/admin.py**:

```
from django.contrib import admin
from .models import Produto

@admin.register(Produto)
class ProdutoAdmin(admin.ModelAdmin):
    list_display = ('nome', 'descricao', 'preco', 'estoque')
```

CRUD para o Modelo Produto: Criando Usuário Admin

Criando super usuário admin

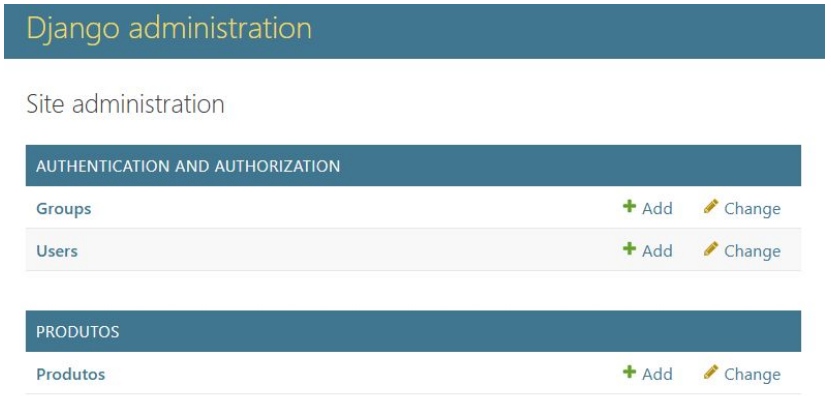
```
python manage.py createsuperuser
```

Depois rodar servidor local

```
python manage.py runserver
```

Acessar:

http://127.0.0.1:8000/admin



CRUD para o Modelo Produto: Formulário com Django Forms

Criar **produtos/forms.py**:

```
from django import forms
from .models import Produto

class ProdutoForm(forms.ModelForm):
    class Meta:
        model = Produto
        fields = ['nome', 'descricao', 'preco', 'estoque']
```

CRUD para o Modelo Produto: Views e URLs do CRUD

Passos para desenvolver uma nova página

1. **Definir o padrão de URL**: em `app/urls.py`, adicione a rota e associe à view.
2. **Implementar a view**: crie função ou classe no `views.py` que processe `request` e retorne `HttpResponse` ou `render()`.
3. **Criar o template**: em `templates/app/`, escreva o HTML usando a sintaxe de template do Django.
4. **Montar o contexto**: na view, passe dados ao template via dicionário.
5. **Testar no navegador**: execute o servidor (`manage.py runserver`) e acesse a URL para validar resultados.

CRUD para o Modelo Produto: Views

Views usando Function-Based Views (FBV)

Em **produtos/views.py**, adicionar:

```
from django.shortcuts import render, get_object_or_404, redirect
from .models import Produto
from .forms import ProdutoForm

def lista_produtos(request):

def cria_produto(request):

def edita_produto(request, pk):

def remove_produto(request, pk):
```

CRUD para o Modelo Produto: Views

Em **produtos/views.py**, adicionar:

```
# Listagem
def lista_produtos(request):
        produtos = Produto.objects.all()
        return render(request, 'produtos/list.html', {'produtos': produtos})
```

CRUD para o Modelo Produto: Views

Em **produtos/views.py**, adicionar:

```
# Criação
def cria_produto(request):
    if request.method == 'POST':
        form = ProdutoForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('lista_produtos')
    else:
        form = ProdutoForm()
    return render(request, 'produtos/form.html', {'form': form})
```

CRUD para o Modelo Produto: Views

Em **produtos/views.py**, adicionar:

```
# Edição
def edita_produto(request, pk):
    produto = get_object_or_404(Produto, pk=pk)
    if request.method == 'POST':
        form = ProdutoForm(request.POST, instance=produto)
        if form.is_valid():
            form.save()
            return redirect('lista_produtos')
    else:
        form = ProdutoForm(instance=produto)
    return render(request, 'produtos/form.html', {'form': form})
```

CRUD para o Modelo Produto: Views

Em **produtos/views.py**, adicionar:

```
# Remoção
def remove_produto(request, pk):
    produto = get_object_or_404(Produto, pk=pk)
    if request.method == 'POST':
        produto.delete()
        return redirect('lista_produtos')
    return render(request, 'produtos/confirm_delete.html', {'produto': produto})
```

CRUD para o Modelo Produto: URLs do CRUD

Em **produtos/urls.py**, substituir por:

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

urlpatterns = [
    path('', views.lista_produtos, name='lista_produtos'),
    path('novo/', views.cria_produto, name='cria_produto'),
    path('edita/<int:pk>/', views.edita_produto, name='edita_produto'),
    path('remove/<int:pk>/', views.remove_produto, name='remove_produto'),
]
```

Django Template Language (DTL)

O Django já inclui por padrão seu próprio motor de templates, baseado em **DTL (Django Template Language)**, que possui sintaxe semelhante ao Jinja, mas com algumas diferenças específicas.

Sintaxe Básica

variáveis

```
{{ produto.nome }}          {# Exibe o atributo nome do objeto produto #}
{{ produto.preco|floatformat:2 }}  {# Aplica filtro para formatar números #}
```

Django Template Language (DTL)

Sintaxe Básica

Tags de Controle:

```
{% if produtos %}
  <ul>
  {% for p in produtos %}
    <li>{{ p.nome }} - R$ {{ p.preco }}</li>
  {% endfor %}
  </ul>
{% else %}
  <p>Nenhum produto encontrado.</p>
{% endif %}
```

Django Template Language (DTL)

Sintaxe Básica

Herança de templates:

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

{% block content %}
  <h1>Produtos</h1>
  {% include 'produtos/partial_list.html' %}
{% endblock %}
```

Templates

Criar pasta **produtos/templates/produtos/** e os arquivos:

- `base.html`
- `list.html`
- `form.html`
- `confirm_delete.html`

Templates: base.html

Criar **templates/produtos/base.html** na pasta do projeto:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <title>Projeto Django Básico</title>
</head>
<body>
  {% block content %}{% endblock %}
</body>
</html>
```

Templates: list.html

Criar **templates/produtos/list.html** na pasta do projeto:

```
{% extends 'produtos/base.html' %}
{% block content %}
<h1>Lista de Produtos</h1>
<a href="{% url 'cria_produto' %}">Novo Produto</a>
<ul>
  {% for p in produtos %}
    <li>
      {{ p.nome }} - R$ {{ p.preco }}
      [<a href="{% url 'edita_produto' p.pk %}">Editar</a>]
      [<a href="{% url 'remove_produto' p.pk %}">Remover</a>]
    </li>
  {% empty %}
    <li>Nenhum produto cadastrado.</li>
  {% endfor %}
</ul>
{% endblock %}
```

Templates: form.html

Criar **templates/produtos/form.html** na pasta do projeto:

```
{% extends 'produtos/base.html' %}
{% block content %}
<h1>{% if form.instance.pk %}Editar{% else %}Novo{% endif %} Produto</h1>
<form method="post">
  {% csrf_token %}
  {{ form.as_p }}
  <button type="submit">Salvar</button>
</form>
{% endblock %}
```

Templates: confirm_delete.html

Criar **templates/produtos/confirm_delete.html** na pasta do projeto:

```
{% extends 'produtos/base.html' %}
{% block content %}
<h1>Remover Produto</h1>
<p>Tem certeza que deseja remover "{{ produto.nome }}"?</p>
<form method="post">
  {% csrf_token %}
  <button type="submit">Confirmar</button>
  <a href="{% url 'lista_produtos' %}">Cancelar</a>
</form>
{% endblock %}
```

Testando a Aplicação

Iniciar servidor:

```
python manage.py runserver
```

Acessar:

- **Lista:** http://127.0.0.1:8000/produtos/
- **Novo:** http://127.0.0.1:8000/produtos/novo/
- **Editar:** http://127.0.0.1:8000/produtos/edita/
- **Remover:** http://127.0.0.1:8000/produtos/remove/

Atividades

Atividade Django Framework 1: https://forms.gle/xMrAvmnfJvKnJWDb7
(Desvendando o Django): https://forms.gle/CZngHe2fGHGqqHAF6