5954024 - Introdução ao Desenvolvimento Web

Python Django

Prof. Dr. Denis M. L. Martins
DCM | FFCLRP | USP

Objetivos de Aprendizagem

- Compreender o papel do Django no desenvolvimento web
- Explicar o padrão arquitetural MVT (Model-View-Template)
- Entender o fluxo de requisição e resposta
- Criar rotas, visualizações e templates básicos.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Python Django

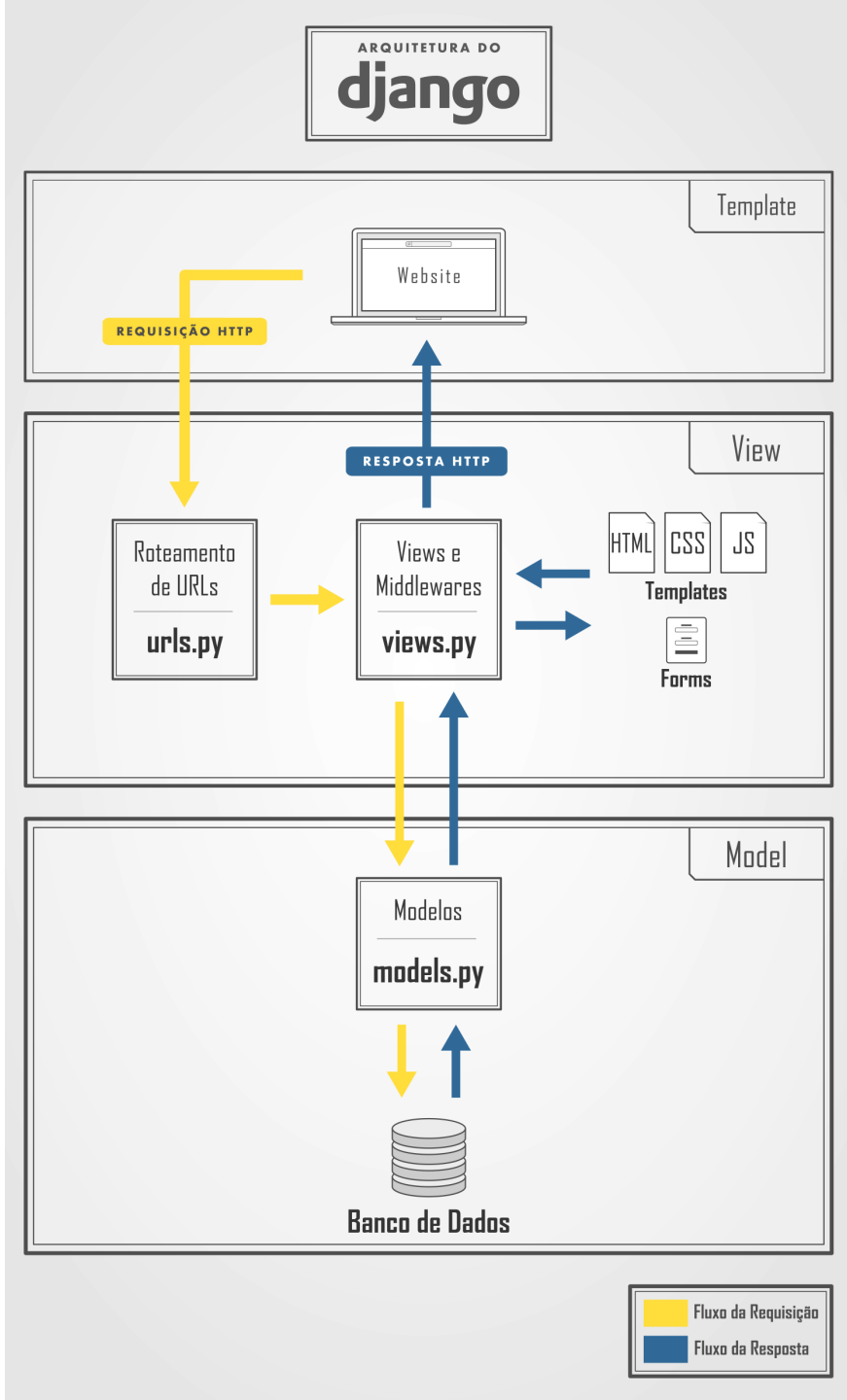
- Framework web de alto nível escrito em Python.
- Segue o princípio: Don't Repeat Yourself (DRY)
- Filosofia "Batteries-included": já vem com autenticação, admin, ORM e segurança integrados.
- Foco em produtividade e segurança.
- Muito utilizado em aplicações reais: Versões anteriores do Instagram e do Bitbucket.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

A Arquitetura MVT (Model-View-Template)

- **Model:** Camada de dados (descrição do banco de dados em Python).
- **View:** Lógica de negócio (processa requisições e retorna respostas).
- **Template:** Camada de apresentação (HTML dinâmico).

> Imagem ao lado: Fluxo de uma requisição por Vinícius Ramos em Python Academy



Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

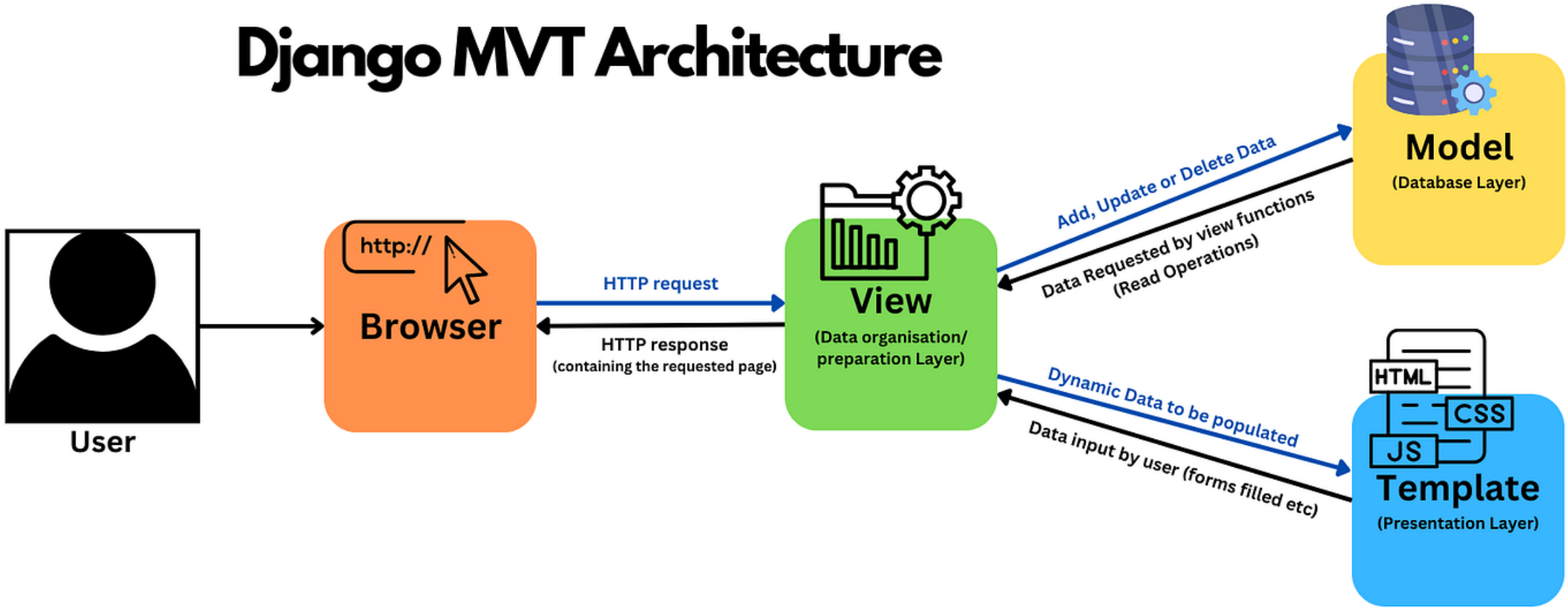


Fig. .1: Arquitetura MVT por PBP Ganjil 24/25.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Configuração Inicial

Criando um ambiente virtual

```
pip install pipenv
pipenv install django
pipenv shell
```

Pipenv é uma ferramenta moderna para gerenciar as bibliotecas (ou "pacotes") de um projeto Python. Ele combina o gerenciamento de pacotes (pip) com a criação de ambientes virtuais.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

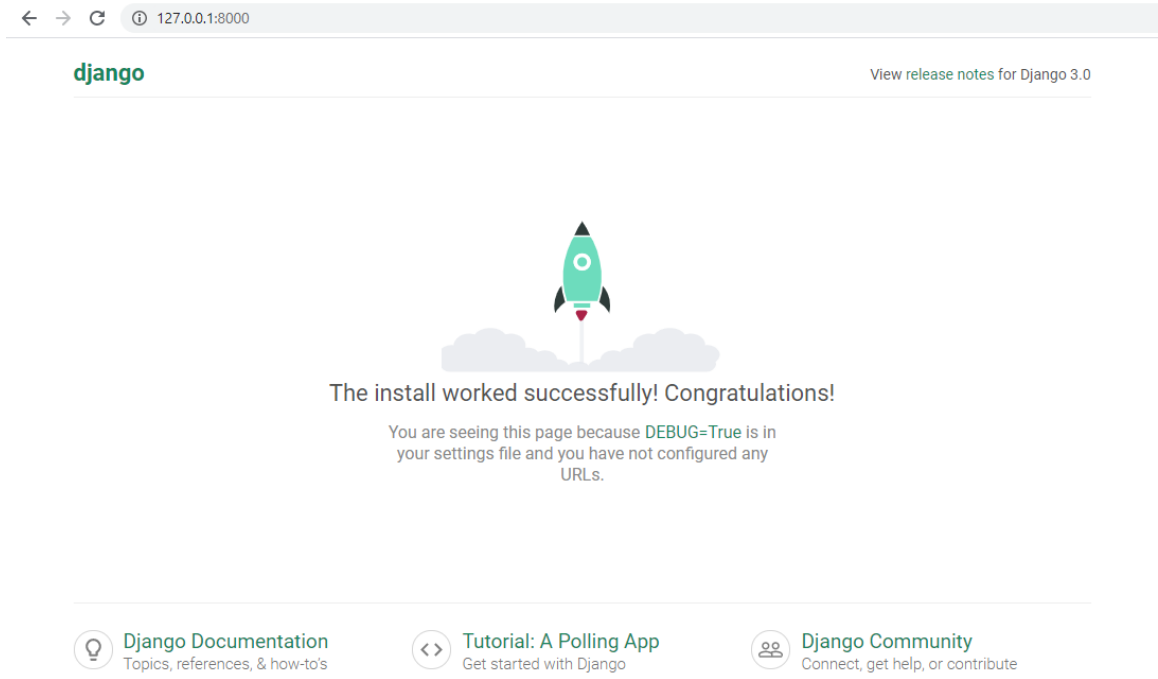
Criando um Projeto Django

```
django-admin startproject myproject
cd myproject
python manage.py runserver
```

- **django-admin startproject**: cria a estrutura base de arquivos.
- **manage.py**: interface de linha de comando para tarefas administrativas.

Acesse a página Web em http://127.0.0.1:8000

> Os comandos acima devem produzir uma página como a mostrada ao lado.



Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

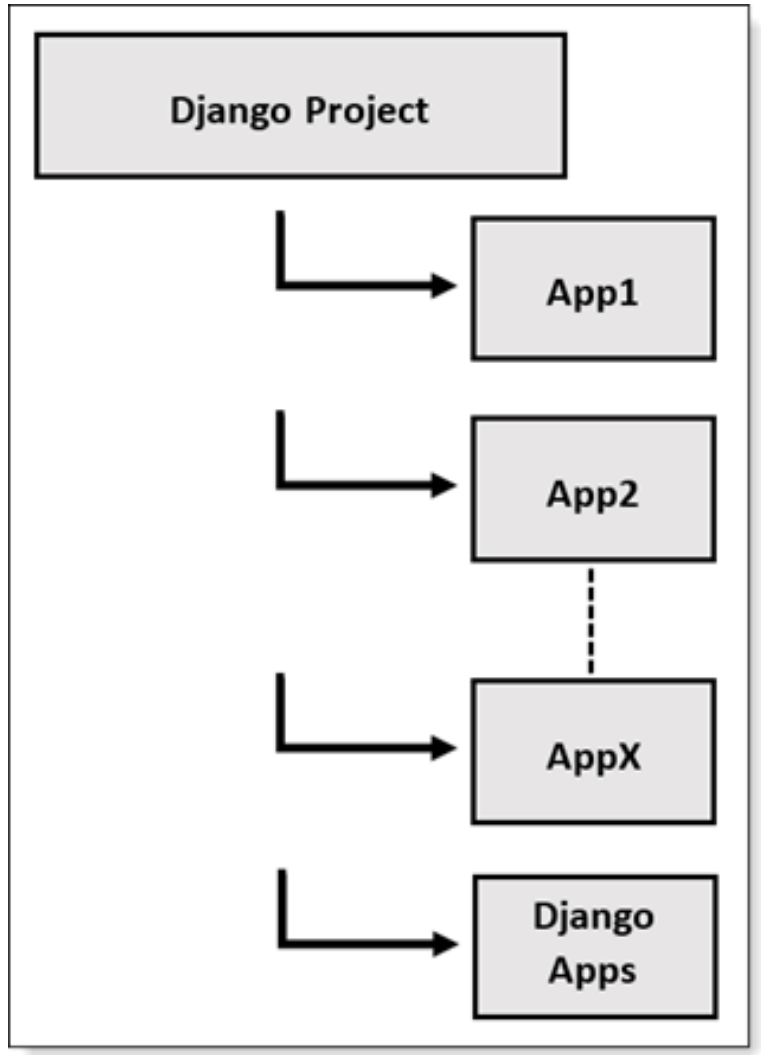
Criando uma App

```
python manage.py startapp myapp
```

Registre no **settings.py**:

```
INSTALLED_APPS = [
    'myapp',
]
```

- **Projeto:** A instalação completa e as configurações globais (ex: um site de notícias).
- **App:** Uma unidade funcional específica dentro do projeto (ex: sistema de comentários, fórum, usuários).



> Imagem ao lado: Um projeto Django com múltiplas Apps por Uday CHANDRAKANT Patkar via ResearchGate

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Criando uma View

Adicione uma view no arquivo **myapp/views.py**:

```
from django.http import HttpResponse

def say_hello(request):
    return HttpResponse('Hello World')
```

Adicione uma nova rota no arquivo **myapp/urls.py**:

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

urlpatterns = [
    path('hello/', views.say_hello)
]
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Conectando a Rota ao Projeto

No arquivo **myproject/urls.py**, adicione a rota:

```
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('myapp/', include('myapp.urls')) # URL para a rota
]
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Templates

Crie um novo diretório **myapp/templates**. No diretório, crie um novo arquivo **hello.html**:

```
cd myapp
mkdir templates
cd templates
touch hello.html
```

Adicione o código abaixo ao arquivo **hello.html**:

```
<h1>Hello World</h1>
```

Mude o código da view em **myapp/views.py** para retornar o template ao invés de uma string:

```
def say_hello(request):
    return render(request, 'hello.html') # returna um objeto HttpResponse
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Templates Dinâmicos

Podemos exibir dados dinamicamente modificando o código no arquivo **myapp/templates/hello.html**:

```
<h1>Hello, {{name}}!</h1>
```

Para passar dados para o template precisamos modifcar **myapp/views.py**:

```
def say_hello(request):
    data = {'name': 'Neo'}
    return render(request, 'hello.html', data)
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Templates Dinâmicos (cont.)

Podemos também adicionar alguma lógica ao template.
Modifique o arquivo **myapp/templates/hello.html**:

```
{% if name %}
<h1>Hello, {{ name }}!</h1>
{% else %}
<h1>Hello, World!</h1>
{% endif %}
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exercício: Criando uma Landing Page

1. Baixe Bootstrap Exemplos.
2. Extraia o arquivo zip e busque pelo diretório **cover**.
3. Copie o conteúdo do diretório **cover** para o diretório **myapp/templates**.
4. Renomeie **cover.html** para **index.html**.
5. Adicione a rota:

```
urlpatterns = [
    path('hello/', views.say_hello),
    path('', views.index) # Nova rota
]
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Criando Modelos (de dados)

Vamos criar um catálogo de produtos. Cada produto é caracterizado por:

Product	Data Type
ID	Auto (created by Django)
Title	Small string
Description	Large String

Adicione uma nova classe **Product** ao arquivo **myapp/models.py**

```
from django.db import models

class Product(models.Model):
    title = models.CharField(max_length=50)
    description = models.TextField()
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Migrações

Agora precisamos notificar ao Django que um novo modelo foi criado:

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

Se as migrações foram feitas corretamente, você deve ver um novo arquivo **myapp/migrations**.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Adicionando Dados

Django usa um banco de dados SQLite por padrão. Podemos adicionar um novo Produto via CLI:

```
python manage.py shell

from myfirstapp.models import Product

product = Product(title="My Book", description="a simple book")
product.save()
```

Podemos realizar consultas aos dados:

```
Product.objects.all()
```

```
Product.objects.get(id=1).title
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exercício

Adicione mais dados ao banco SQLite via CLI.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exibindo dados

Adicione uma nova View ao arquivo **myapp/views.py**:

```
from .models import Product

def get_all_products(request):
    products = Product.objects.all()
    html = ''
    for p in products:
        html += f'<h1>{p.id} - {p.title}<h1><span>{p.description}</span>'

    return HttpResponse(html)
```

Adicione uma nova rota em **myapp/urls.py**:

```
path('products/', views.get_all_products),
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exibindo dados (alternativa)

Como alternativa, crie um novo template **products.html** em **myapp/templates**:

```
{% if products %}
    <ul>
    {% for product in products %}
        <li>{{ product.title }}</li>
    {% endfor %}
    </ul>
{% else %}
    <h1>No products</h1>
{% endif %}
```

E mude a view **myapp/views.py**:

```
def get_produts(request):
    data = {'products': Product.objects.all()}
    return render(request, 'products.html', data)
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Admin dashboard

Crie um novo super usuário

```
python manage.py createsuperuser
```

- Username: admin
- Email: admin@myapp.com
- Password: admin

Para exibir o admin dashboard adicione o código abaixo em **myapp/admin.py**:

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

admin.site.register(Product)
```


Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Considerações Finais

- Django agiliza o desenvolvimento ao oferecer ferramentas nativas para tarefas comuns, como autenticação e interface administrativa.
- **Arquitetura MVT:** Fluxo onde a **URL** mapeia a requisição, a **View** processa a lógica, o **Model** gerencia os dados e o **Template** apresenta a interface final.
- **ORM e Migrações:** Django interage com o banco de dados usando classes Python, garantindo que as mudanças estruturais sejam rastreadas e aplicadas de forma segura via migrações.

Próximos Passos:

- **Aprofundamento em Segurança:** Explore como o Django protege contra ataques como CSRF e como implementar modelos de usuário customizados.
- **Estilização e Frontend:** Integre frameworks como **Bootstrap** ou **Tailwind** aos seus templates para criar interfaces profissionais.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Dúvidas e Discussão

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exercício Prático 1

Objetivo: Praticar a criação de um projeto, uma aplicação, rotas (`urls.py`) e a renderização de templates com herança.
Tarefa: Crie um projeto chamado `meu_site` e uma aplicação chamada `biografia`.
Requisitos:

1. Configure uma página inicial (`/`) que mostre seu nome e uma breve descrição profissional.
2. Crie uma página de contato (`/contato/`) com um texto fictício.
3. **Herança de Templates:** Use um arquivo `base.html` para o cabeçalho (com links de navegação) e rodapé, garantindo que as outras páginas estendam esse arquivo via tag `{% extends %}`.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Exercício Prático 2

Objetivo: Criar um Mini-Blog de Avisos Acadêmicos.

1. Cadastrar avisos (título, autor e mensagem) pelo Admin.
2. Exibir todos os avisos em uma página inicial estilizada com um CSS simples ou Bootstrap.

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Questões para Estudo

- Explique o padrão de arquitetura MVT (Model-View-Template) adotado pelo Django.
- Como funciona o ciclo de requisição e resposta (Request/Response Cycle) no Django?
- O que é o ORM do Django e qual sua principal vantagem para o desenvolvedor? Qual a principal desvantagem introduzida por este mecanismo?

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br

Questões para Estudo (cont.)

Uma aplicação Django foi desenvolvida para gerenciar pedidos de uma loja virtual. Um usuário acessa a URL **/pedidos/**, que exibe todos os pedidos cadastrados no sistema. Explique detalhadamente todo o **fluxo de processamento dessa requisição** no Django, desde o momento em que o usuário digita a URL no navegador até a renderização final da página.

Sua resposta deve incluir obrigatoriamente:

- O papel do arquivo **urls.py**
- A função da *View*
- A interação com o *Model*
- O uso de *Templates*
- O retorno da resposta HTTP ao cliente

Prof. Dr. Denis M. L. Martins | martins.denis@usp.br