Apostila: Iniciando Django

Do Zero ao "Olá, Mundo!"

1. Introdução

Bem-vindo ao Django! Esta apostila é o seu guia de "início rápido". Vamos sair do zero absoluto, configurar um ambiente de desenvolvimento e criar sua primeira página web funcional.

O que é o Django?

O Django é um "framework web" de alto nível escrito em Python. Pense nele como uma caixa de ferramentas completa para construir sites.

- **"Alto Nível":** Ele cuida de muitas partes chatas e repetitivas do desenvolvimento web para que você possa focar na lógica do seu site.
- **"Baterias Inclusas":** Diferente de outros frameworks, o Django já vem com quase tudo que você precisa: um painel de administração, sistema de autenticação de usuários, ferramentas para falar com o banco de dados (ORM) e muito mais.

A filosofia do Django é **DRY (Don't Repeat Yourself - Não se Repita)**. Ele foi criado para ajudar desenvolvedores a construir sites complexos e seguros rapidamente.

> **Quem usa Django?** Sites que você provavelmente usa todos os dias, como **Instagram**, **Spotify**, **Pinterest** e até a **NASA**, usam Django para gerenciar seus sistemas robustos.

Nosso Objetivo: Entender o fluxo básico de um pedido web no Django, criando uma página simples que exibe a mensagem "Olá, Mundo!".

2. Módulo 1: Preparação e Configuração (30 min)

Vamos preparar nosso "canteiro de obras".

2.1. Pré-requisitos

Para esta aula, assumimos que você já tem o **Python** (versão 3.8 ou superior) e o **pip** (gerenciador de pacotes do Python) instalados em seu computador.

2.2. O Ambiente Virtual (venv)

Por que isso é crucial? Imagine que você tem dois projetos: um usa a "ferramenta A v1.0" e o outro usa a "ferramenta A v2.0". Se você instalar as duas no seu computador, elas entrarão em conflito.

Um **ambiente virtual** (ou venv) é uma "caixa" isolada para cada projeto. Dentro dela, instalamos todas as dependências (como o Django) sem afetar outros projetos ou o seu sistema.

Passo 1: Crie o ambiente
Abra seu terminal (Prompt de Comando, PowerShell, ou Terminal) e navegue até a pasta onde você quer criar seu projeto (ex: Documentos/Projetos).
Digite o comando:

```
# Este comando cria uma pasta chamada 'venv'
python -m venv venv
```

Passo 2: Ative o ambiente
Você precisa "entrar" nessa caixa para usá-la.

- **No Windows (PowerShell/CMD):**
  Bash
  .\venv\Scripts\activate

- **No Mac/Linux:**
  Bash
  `source venv/bin/activate`

  > **Dica:** Você saberá que funcionou, pois o nome do seu terminal agora terá (venv) no início, assim: (venv) C:\Users\SeuNome\Desktop\meuprojeto>

2.3. Instalando o Django

Agora que estamos *dentro* do venv, podemos instalar o Django com segurança.

`pip install django`

Para verificar se foi instalado, use o comando pip freeze. Ele lista tudo que está instalado *apenas* neste ambiente:

```
(venv) > pip freeze
asgiref==...
Django==...  <-- É isso que queremos ver!
sqlparse==...
...
```

3. Módulo 2: O Esqueleto do Projeto (30 min)

Agora, vamos usar os comandos do Django para construir a "estrutura" do nosso site.

3.1. Criando o "Projeto"

No Django, há uma diferença fundamental entre um **Projeto** e um **App**.

- **Projeto:** É o "site" inteiro, a "casa". Ele contém as configurações gerais, URLs principais, etc.
- **App:** É uma funcionalidade específica do seu site, um "cômodo" da casa (ex: um app de blog, um app de usuários, um app de galeria de fotos).

Vamos criar o **Projeto**:

```
# O 'startproject' cria o projeto
# 'meuprojeto' é o nome que daremos a ele
# O '.' no final é MUITO importante!
django-admin startproject meuprojeto .
```

> **Por que o . no final?** Sem ele, o Django criaria uma pasta meuprojeto dentro de outra pasta meuprojeto (meuprojeto/meuprojeto/). O . diz ao Django: "Crie o projeto *aqui* nesta pasta onde já estou".

3.2. A Estrutura de Arquivos

Se você olhar sua pasta, verá que o Django criou algumas coisas:

```
venv/                   <-- Nosso ambiente virtual
meuprojeto/             <-- A "central de controle" do projeto
    __init__.py
    asgi.py
    settings.py         <-- O "Painel de Controle" (configurações)
    urls.py             <-- O "Mapa" principal do site
    wsgi.py
manage.py               <-- O "Gerente" do nosso site
```

Os dois arquivos mais importantes por agora:

- manage.py: Nosso "gerente de obras". Usamos ele para dar comandos ao Django (como runserver, migrate, etc.).
- meuprojeto/settings.py: O coração das configurações. É aqui que definimos o banco de dados, registramos apps, e muito mais.

3.3. O Primeiro "Win": Testando o Servidor

Vamos ver se tudo funcionou. Use o manage.py para ligar o servidor de desenvolvimento do Django:

```
python manage.py runserver
```

Você verá algo como:

```
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).

You have 18 unapplied migration(s). ... (Ignore isso por enquanto)

Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK (Windows) or CTRL-C (Mac/Linux).
```

Abra seu navegador e acesse **http://127.0.0.1:8000**.

Se você vir a página de "foguete" do Django ("The install worked successfully! Congratulations!"), **parabéns!** Seu esqueleto de projeto está 100% funcional.

Pressione Ctrl+C no terminal para parar o servidor.

---

4. Módulo 3: Criando seu Primeiro "App" (30 min)

Nosso "Projeto" (a casa) está de pé. Agora vamos construir nosso primeiro "App" (um cômodo). Vamos chamá-lo de core (é um nome comum para o app principal de um site).

4.1. O Comando startapp

Verifique se você está com o venv ativo e na mesma pasta que o manage.py.

```
python manage.py startapp core
```

Isso criará uma nova pasta chamada core:

```
venv/
core/                   <-- Nosso novo App!
   __init__.py
   admin.py             (Para o painel admin)
   apps.py
   migrations/          (Para o banco de dados)
   models.py            (Para o banco de dados)
   tests.py
   views.py             <-- É aqui que escrevemos nossa lógica!
meuprojeto/
   ...
manage.py
```

O arquivo mais importante para nós agora é o **core/views.py**.

4.2. Registrando o "App"

Você construiu o "cômodo", mas precisa "avisar a casa" (o Projeto) que ele existe.

1. Abra o arquivo meuprojeto/settings.py.
2. Procure pela lista chamada INSTALLED_APPS.
3. Adicione o nome do seu app ('core') no final da lista.

```
# meuprojeto/settings.py

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'core',  # <-- ADICIONE ESTA LINHA
]
```

Agora o Django oficialmente reconhece seu app.

5. Módulo 4: O Caminho da Requisição (O "Olá, Mundo!") (30 min)

Este é o momento! Vamos entender o fluxo básico do Django para exibir nossa página.

O Fluxo (Simplificado):

1. O usuário acessa uma **URL**.
2. O Django (em urls.py) vê qual **View** (Função Python) é responsável por essa URL.
3. A **View** (em views.py) é executada.
4. A **View** retorna uma **Resposta** (como um HTML) para o usuário.

5.1. Passo 1: A "View" (A Lógica)

A **View** é apenas uma função Python que decide o que o usuário vai ver.

Abra o arquivo **core/views.py** e escreva sua primeira view:

```
# core/views.py

from django.http import HttpResponse

# Uma 'view' é uma função que recebe um 'request' e retorna uma 'response'
def home(request):
    # Vamos retornar a resposta HTTP mais simples: um texto HTML
    return HttpResponse("<h1>Olá, Mundo! Esta é minha primeira página Django!</h1>")
```

5.2. Passo 2: O Roteamento de URLs (O "Mapa")

Nossa view está pronta, mas ninguém consegue acessá-la. Precisamos conectá-la a uma URL. Isso é feito em duas etapas:

Etapa 2.a: Criar o mapa do App

Crie um **novo arquivo** chamado **urls.py** dentro da pasta core (ele não vem por padrão).

```
core/
  ...
  views.py
  urls.py    <-- CRIE ESTE ARQUIVO
```

Dentro de **core/urls.py**, coloque este código:

```
# core/urls.py

from django.urls import path
from . import views  # O '.' importa as 'views' do app atual

urlpatterns = [
  # Quando a URL for a raiz (''), chame a função 'home' de 'views.py'
  path('', views.home, name='home'),
]
```

Etapa 2.b: Conectar o mapa do App ao mapa do Projeto

Agora, precisamos dizer ao mapa principal (do Projeto) para "incluir" o mapa do nosso app.

Abra o arquivo **meuprojeto/urls.py** (o que já existia) e modifique-o:

```
# meuprojeto/urls.py

from django.contrib import admin
from django.urls import path, include  # 1. Importe o 'include'

urlpatterns = [
  path('admin/', admin.site.urls),
  path('', include('core.urls')),  # 2. Adicione esta linha
]
```

O que essa linha path('', include('core.urls')) faz?
Ela diz: "Qualquer URL que chegar no site (representada pelo '') deve ser redirecionada para o arquivo core/urls.py para ser processada."

5.3. O "Win" Final: Teste sua Página!

Vamos ligar o servidor novamente:

```
python manage.py runserver
```

Agora, acesse http://127.0.0.1:8000 no seu navegador.

A página do foguete desapareceu. No lugar dela, você deve ver:

Olá, Mundo! Esta é minha primeira página Django!