Workshop Django

Primeira Página Web

Bem-vinda ao Time!

Este workshop foi desenhado para quem nunca escreveu uma linha de código.

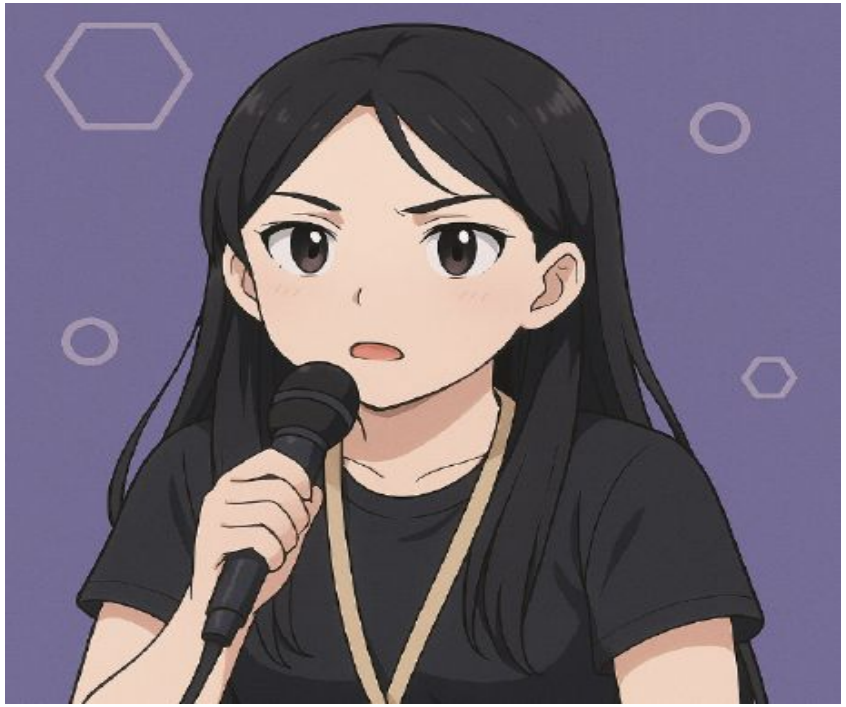
Nossa promessa: Você sairá daqui com um blog funcional na internet.

Não tenha medo de errar. Erros são apenas o computador tentando conversar com você.

Instrutora

Sou **Dáviny Letícia**, formada (ADS) com Pós em Desenvolvimento de aplicações WEB (FAAL) e Cursando Desenvolvimento de Sistema Multiplataforma na Fatec, atuo como Desenvolvedora de Software e Instrutora na ETEC.

www.vidal.press



Nosso Objetivo de Hoje

O Blog Final

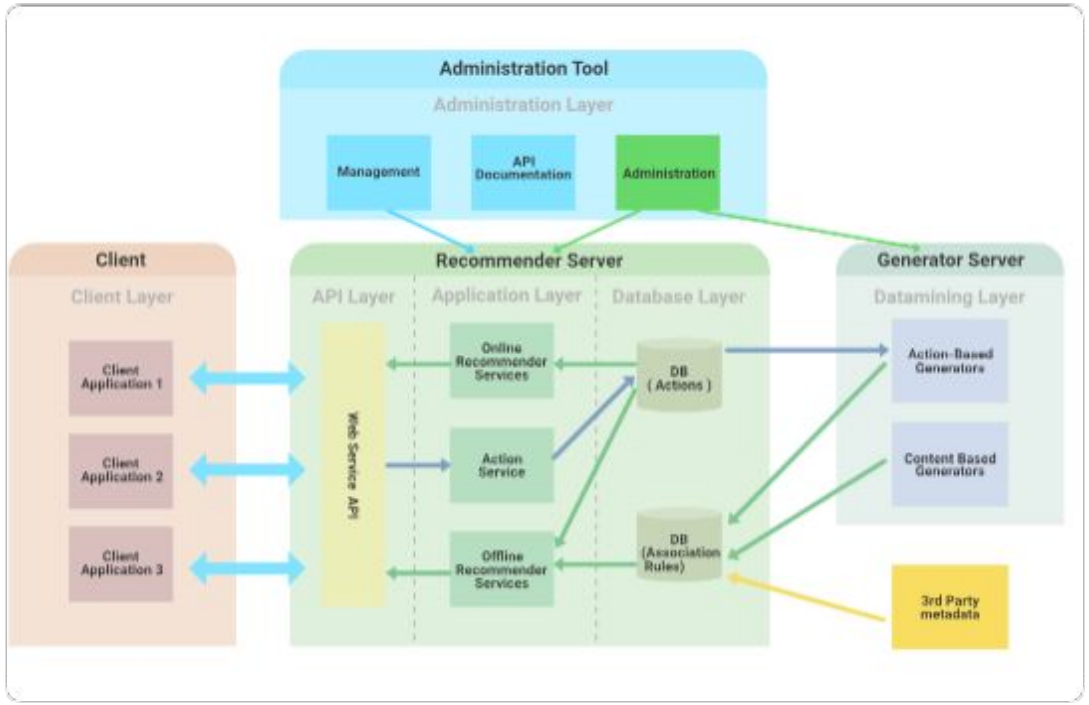
Vamos construir um sistema onde você pode postar textos, imagens, editar conteúdos e ver uma lista de publicações organizada por data.

Como a Web Funciona?

Imagine que você quer ler este blog.

1. Você digita o endereço (URL) no seu navegador.

2. Isso envia uma **Requisição** para um computador em algum lugar do mundo.



O Pedido e a Resposta

O computador que recebe o pedido é chamado de **Servidor**.

O Django é o "chef de cozinha" desse servidor. Ele recebe o pedido, prepara a página e envia de volta como uma **Resposta**.

O Protocolo HTTP

HTTP é a linguagem que os computadores usam para pedir páginas web.

Pense nele como o envelope da carta: ele diz quem envia, quem recebe e o que tem dentro.

A Linha de Comando

Programadores usam janelas pretas (Terminais) para conversar com o computador de forma rápida.

- **Windows:** Prompt de Comando ou PowerShell.
- **Mac/Linux:** Terminal.

Navegando no Terminal

O comando `cd` (Change Directory) serve para entrar em pastas.

O comando `ls` ou `dir` (Windows) lista o que tem dentro da pasta atual.

Criando nossa Pasta

Vamos criar o lugar onde nosso blog vai morar:

```
$ mkdir djangogirls
$ cd djangogirls
```

Verificando o Python

O Django é feito em **Python**. Precisamos ter o Python instalado.

```
$ python --version
```

Editor de Código

Não usamos o Word para programar. Usamos editores como o **VS Code**.

Ele colore o código e nos ajuda a não esquecer parênteses ou vírgulas.

O que é um Virtualenv?

Imagine que o seu computador é uma cozinha. Você quer fazer um bolo (Blog), mas não quer que os ingredientes dele se misturem com os ingredientes de outros doces.

O **Virtualenv** é como um pote isolado onde guardamos apenas as coisas do Django Girls.

Criando o Ambiente

No Windows:

```
$ python -m venv myvenv
```

Isso cria uma pasta chamada myvenv dentro do seu projeto.

Ativando o Ambiente (Windows)

```
$ myvenv\Scripts\activate
```

Você saberá que deu certo quando aparecer um (myvenv) no começo da linha do terminal.

Ativando o Ambiente (Mac/Linux)

```
$ source myvenv/bin/activate
```

Instalando o Django

Com o ambiente ativo, vamos baixar o Django da internet:

```
$ pip install django
```

Iniciando o Projeto

Vamos criar o "esqueleto" do site:

```
$ django-admin startproject mysite .
```

Atenção: Não esqueça o ponto final! Ele diz para criar na pasta onde já estamos.

Por que o ponto final?

O ponto evita que o Django crie uma pasta dentro de outra pasta com o mesmo nome.

Isso deixa seu projeto muito mais organizado e fácil de gerenciar.

Estrutura de Pastas

📁 **djangogirls** (Pasta Raiz)
- 📄 manage.py
- 📁 **mysite**
  - 📄 settings.py
  - 📄 urls.py

Configurações: Idioma

Abra o arquivo mysite/settings.py no VS Code.

Procure por **LANGUAGE_CODE** e mude para:

```
LANGUAGE_CODE = 'pt-br'
```

Configurações: Fuso Horário

No mesmo arquivo, mude o **TIME_ZONE**:

```
TIME_ZONE = 'America/Sao_Paulo'
```

Isso garante que as datas do seu blog fiquem corretas para o Brasil.

Banco de Dados

É o lugar onde guardamos os posts. O Django já vem configurado com o **SQLite**.

Pense nele como uma planilha de Excel gigante que o Django sabe ler e escrever muito bem.

Primeira Migração

Precisamos criar as tabelas básicas do banco:

```
$ python manage.py migrate
```

Você verá vários "OK" no terminal. Parabéns, seu banco de dados está vivo!

Rodando o Servidor

Vamos ligar o site:

```
$ python manage.py runserver
```

Abra **http://127.0.0.1:8000** no navegador. Se vir um foguete, tudo deu certo!

Apps vs Projetos

Projeto

É a casa inteira (mysite).

App

É um cômodo da casa com função específica (ex: blog).

Criando o App do Blog

```
$ python manage.py startapp blog
```

Isso cria uma pasta chamada blog com novos arquivos dentro.

Registrando o App

Abra settings.py e procure por **INSTALLED_APPS**.

Adicione 'blog' na lista:

```
INSTALLED_APPS = [
    ...
    'blog',
]
```

O que é um Modelo?

O **Modelo** é a definição do que um objeto é.

Para o nosso blog, um Post precisa ter: Título, Texto, Data de Criação e Autor.

Criando o Modelo Post

Abra blog/models.py. Vamos escrever nossa primeira classe Python.

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

class Post(models.Model):
    author = models.ForeignKey('auth.User', on_delete=models.CASCADE)
```

Campos do Modelo

```
    title = models.CharField(max_length=200)
    text = models.TextField()
    created_date = models.DateTimeField(default=timezone.now)
    published_date = models.DateTimeField(blank=True, null=True)
```

Avisando o Banco de Dados

O Django precisa saber que criamos um novo modelo.

```
$ python manage.py makemigrations blog
```

Isso cria um arquivo de instruções para o banco de dados.

Aplicando as Mudanças

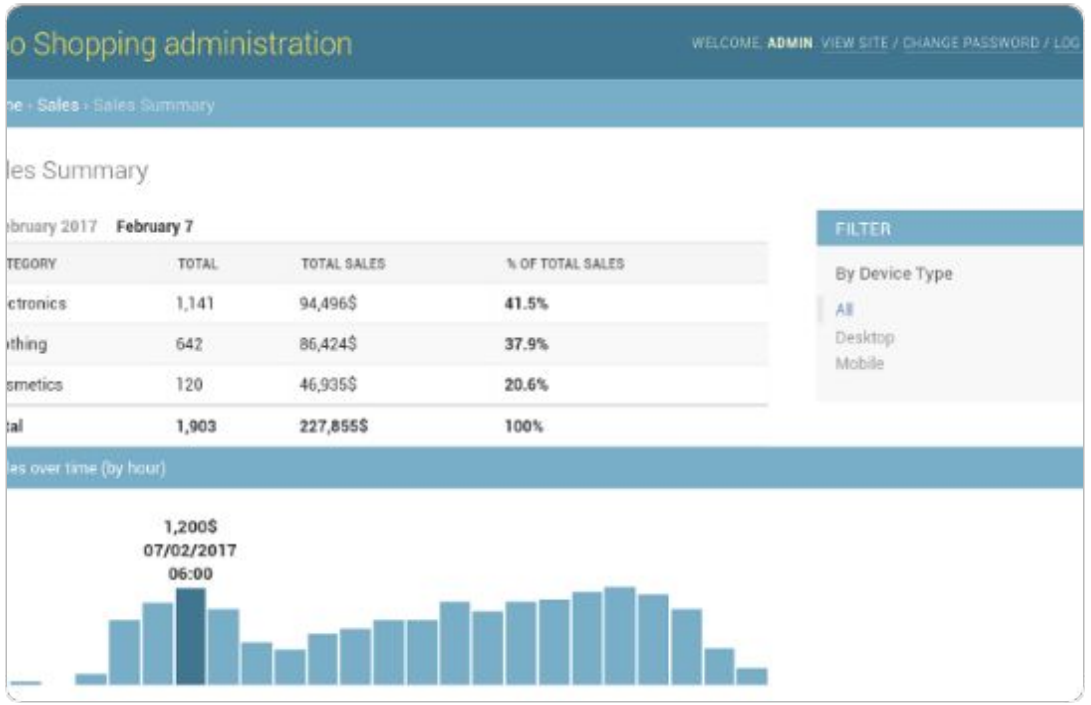
Agora, vamos realmente criar a tabela Post no banco:

```
$ python manage.py migrate blog
```

O Painel de Controle

O Django já traz pronto um site para gerenciar seus dados: o **Django Admin**.

Mas primeiro, precisamos registrar o nosso Post lá.



Registrando no Admin

Abra blog/admin.py e adicione:

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

admin.site.register(Post)
```

Criando seu Usuário

Você precisa de uma conta de administrador:

```
$ python manage.py createsuperuser
```

Siga os passos: escolha nome, e-mail e uma senha que você não vá esquecer!

O que são URLs?

URL é o endereço do seu site. O Django usa o arquivo **urls.py** como um mapa.

Quando alguém pede um endereço, o Django olha o mapa e decide qual código (View) deve rodar.

Configurando URLs

Abra mysite/urls.py. Vamos incluir as URLs do nosso blog:

```
from django.urls import path, include

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

O que é uma View?

A **View** é onde a mágica acontece. É uma função Python que recebe um pedido e decide o que mostrar.

Ela busca dados no Modelo e os entrega para o Template.

Criando a Primeira View

Abra blog/views.py:

```
def post_list(request):
    return render(request, 'blog/post_list.html', {})
```

Esta função vai procurar um arquivo HTML para mostrar na tela.

Introdução ao HTML

HTML é a estrutura do site. Ele define o que é título, o que é parágrafo e o que é link.

Pense nele como o esqueleto do site.

Pasta de Templates

Crie uma pasta chamada **templates** dentro da pasta blog.

Dentro dela, crie outra pasta chamada **blog**.

Por fim, crie o arquivo **post_list.html** lá dentro.

Seu Primeiro HTML

```
Blog da Django Girls
Oi! Funciona!
```

O que é o ORM?

O **ORM** permite que você converse com o banco de dados usando apenas Python.

Em vez de escrever códigos SQL complicados, você usa comandos simples como Post.objects.all().

Buscando Posts na View

Vamos alterar a nossa view para buscar os posts reais:

```
from .models import Post

def post_list(request):
    posts = Post.objects.all()
    return render(request, 'blog/post_list.html', {'posts': posts})
```

Dados Dinâmicos

Diferente de um site estático, um site dinâmico muda conforme os dados no banco mudam.

O Django pega os posts e os "injeta" no HTML automaticamente.

Template Tags: O Loop

Como mostrar vários posts se não sabemos quantos existem?

Usamos o **For Loop**. Ele repete um pedaço de HTML para cada item da nossa lista.

Codando o Loop

```
{% for post in posts %}

    {{ post.title }}
    {{ post.text }}

{% endfor %}
```

CSS: Deixando Bonito

CSS é a maquiagem do site. Ele define cores, tamanhos e fontes.

Sem CSS, a internet pareceria um documento do Word sem formatação.

Arquivos Estáticos

Arquivos que não mudam (como CSS e Imagens) são chamados de **Estáticos**.

No Django, guardamos eles em uma pasta chamada static dentro do app.

Criando o CSS

Crie blog/static/css/blog.css:

```
h1 a { color: #44b78b; text-decoration: none; }
body { font-family: 'Poppins', sans-serif; }
```

Herança de Template

Programadores são preguiçosos. Não queremos repetir o cabeçalho em todas as páginas.

Usamos a **Herança** para criar um arquivo base e apenas mudar o conteúdo do meio.

O arquivo base.html

```
Meu Blog
{% block content %}{% endblock %}
```

Estendendo o Base

No seu post_list.html, você só precisa disso:

```
{% extends 'blog/base.html' %}
{% block content %}
  ... seu loop aqui ...
{% endblock %}
```

Formulários

Até agora, só vimos dados. Mas como o usuário envia dados para nós?

Usamos **Formulários**. Eles permitem que os usuários escrevam novos posts diretamente pelo site.

Criando forms.py

O Django cria formulários automaticamente baseados nos seus modelos!

```
from django import forms
from .models import Post

class PostForm(forms.ModelForm):
    class Meta:
        model = Post
        fields = ('title', 'text',)
```

Segurança: CSRF Token

O Django protege seus formulários contra ataques maliciosos.

Sempre que usar um formulário, você deve incluir a tag:

```
{% csrf_token %}
```

Salvando o Post

Na sua view, você checa se o usuário clicou no botão de enviar:

```
if form.is_valid():
    post = form.save(commit=False)
    post.author = request.user
    post.save()
```

O que é CRUD?

Create (Criar)

Read (Ler/Ver)

Update (Atualizar/Editar)

Delete (Deletar)

Estas são as 4 ações fundamentais de qualquer sistema web.

Página de Detalhes

Como ver apenas um post específico?

Usamos o **ID** (ou PK - Primary Key) dele na URL.

```
path('post//', views.post_detail, name='post_detail')
```

Buscando por PK

Na view, usamos uma função de ajuda para encontrar o post ou mostrar erro 404:

```
post = get_object_or_404(Post, pk=pk)
```

Editando um Post

A edição é quase igual à criação, mas passamos o post que já existe para o formulário:

```
form = PostForm(request.POST, instance=post)
```

O que é Paginação?

Se você tiver 100 posts, não deve mostrar todos de uma vez.

Dividimos em páginas para o site carregar mais rápido e ficar mais organizado.

Paginator na View

```
from django.core.paginator import Paginator

paginator = Paginator(posts, 5) # 5 por página
page = request.GET.get('page')
posts = paginator.get_page(page)
```

Paginação no HTML

```
{% if posts.has_previous %}
  Anterior
{% endif %}
Página {{ posts.number }} de {{ posts.paginator.num_pages }}
```

Revisão: O Caminho

1. Pedido (URL)
2. Lógica (View)
3. Dados (Model)
4. Resposta (Template/HTML)

Conselhos para o Futuro

Não tente decorar tudo. O importante é saber onde procurar.

O Google é o melhor amigo de um programador. Use-o sem **moderação**!

Comunidades

Você não está sozinha. Existem milhares de pessoas dispostas a ajudar.

Seu Próximo Desafio

Mão na Massa!

Tente mudar a cor do cabeçalho no CSS ou adicionar um novo campo ao seu modelo (como "categoria").

Parabéns!

Você completou o Workshop Django

Continue codando e mudando o mundo.

www.vidal.press

Image Sources

https://raw.githubusercontent.com/olasitarska/djangogirls/gh-pages/resources/graphics/logo_square.png

Source: djangogirls.org

https://media.geeksforgeeks.org/wp-content/cdn-uploads/20210204220403/Web-Application-Architecture.png

Source: www.geeksforgeeks.org

https://1000logos.net/wp-content/uploads/2020/08/Python-Logo.png

Source: 1000logos.net

https://upload.wikimedia.org/wikipedia/commons/9/9a/Visual_Studio_Code_1.35_icon.svg

Source: en.wikipedia.org

https://miro.medium.com/1*6-Gd1lZ242CfUT9BkcSniQ.png

Source: medium.com

https://www.sqlitetutorial.net/wp-content/uploads/2021/04/sqlite-tutorial-homepage.svg

Source: www.sqlitetutorial.net

Image Sources

https://hakibenita.com/images/01-how-to-turn-django-admin-into-a-lightweight-dashboard.png

Source: hakibenita.com

https://static.vecteezy.com/system/resources/previews/043/557/249/non_2x/isometric-flat-concept-of-web-page-design-update-vector.jpg

Source: www.vecteezy.com

https://static.vecteezy.com/system/resources/previews/075/828/701/non_2x/startup-rocket-launch-icon-representing-a-new-business-project-innovation-and-fast-growth-for-a-successful-venture-and-market-development-solid-glyph-icon-vector.jpg

Source: www.vecteezy.com