Guia Completo: Estrutura de um Projeto Django para Leigos

Este documento foi criado para explicar, de forma extremamente detalhada e didática, cada elemento de um projeto Django. O objetivo é que mesmo um iniciante absoluto compreenda como tudo funciona. Vamos usar analogias simples, exemplos práticos e diagramas textuais.

# Capítulo 1: Visão Geral do Django

Django é um framework web em Python que segue o padrão MTV (Model-Template-View). Ele ajuda a criar aplicações web de forma rápida, organizada e segura.

Analogia do condomínio:  
• O projeto é o condomínio inteiro (regras gerais, portaria, administração).  
• Cada app é um prédio dentro do condomínio (com sua função específica: usuários, empresas, indicadores, relatórios, etc.).  
• Dentro de cada prédio (app) você tem modelos (o banco de dados), visões (o que acontece quando alguém entra num endereço), templates (as páginas HTML), URLs (endereços que levam às visões), admin (o painel de controle), testes e mais.

# Capítulo 2: Estrutura do Projeto Django

Quando você cria um projeto Django com o comando:  
  
django-admin startproject meu\_projeto  
  
Ele gera uma estrutura básica:  
meu\_projeto/  
 manage.py  
 meu\_projeto/  
 \_\_init\_\_.py  
 settings.py  
 urls.py  
 asgi.py  
 wsgi.py

Explicação dos arquivos principais:  
• manage.py: A 'portaria' do condomínio. É usado para executar comandos (rodar servidor, criar migrações, etc.).  
• settings.py: As regras gerais do condomínio. Configurações globais (idioma, banco de dados, apps instalados).  
• urls.py: O mapa geral de endereços. Define para onde cada URL aponta.  
• wsgi.py/asgi.py: Pontos de entrada para servidores web.

# Capítulo 3: Estrutura de um App Django

Cada app é criado com:  
  
python manage.py startapp nome\_do\_app  
  
Estrutura típica:  
nome\_do\_app/  
 models.py  
 views.py  
 urls.py  
 admin.py  
 tests.py  
 templates/  
 static/

Função de cada arquivo:  
• models.py: Define as tabelas do banco de dados.  
• views.py: Define o que acontece quando alguém acessa uma URL.  
• urls.py: Define os endereços internos do app.  
• admin.py: Configura o painel administrativo.  
• templates/: Contém os arquivos HTML.  
• static/: Contém CSS, JS e imagens.

# Capítulo 4: Banco de Dados e Migrations

Django usa um ORM (Object-Relational Mapper) para interagir com o banco de dados. Por padrão, usa SQLite.  
Quando você cria ou altera modelos, precisa gerar migrações:  
  
python manage.py makemigrations  
python manage.py migrate  
  
Esses comandos criam e aplicam mudanças no banco.

# Capítulo 5: Templates e Arquivos Estáticos

Templates são arquivos HTML que exibem dados dinâmicos usando a linguagem de template do Django.  
Arquivos estáticos (CSS, JS, imagens) ficam na pasta static/.  
Para usar templates, configure TEMPLATES no settings.py e use render nas views.

# Capítulo 6: Como Tudo se Conecta

Fluxo básico:  
1. Usuário acessa uma URL.  
2. urls.py do projeto encaminha para urls.py do app.  
3. A URL chama uma view.  
4. A view acessa modelos (dados) e renderiza um template.  
5. O template exibe os dados para o usuário.

# Capítulo 7: Comandos Básicos do Django

• Criar projeto: django-admin startproject meu\_projeto  
• Criar app: python manage.py startapp meu\_app  
• Rodar servidor: python manage.py runserver  
• Criar migrações: python manage.py makemigrations  
• Aplicar migrações: python manage.py migrate  
• Criar superusuário: python manage.py createsuperuser