**Criando um App com Django - Resumão de Comandos e Conceitos**

**Parte 1**

1. Instale o Django:

*Instalando o Django pelo CMD (win 10)*

python -m pip install Django

1. Escolha o diretório de preferência com o “cd” pela linha de comando do CMD.
2. Inicie o Projeto:

$ django-admin startproject (nomedoprojeto):

O que foi criado no diretório com esse comando?

1. **mysite/ (Externo):**

*O que é*: Container.

*Para que serve*: Armazenar meu projeto.

O nome dele não importa. Tanto faz ser mysite quanto admin123. O Django não se importa com isso.

1. **manage.py :**

* *O que é:* utilitário de linha de comando do Django.
* *Para que serve:* interagir com o projeto e fazer tarefas administrativas nele
* *Diferença do django-admin:* Define o DJANGO\_SETTINGS\_MODULE

1. **mysite/ (interior):**

* *O que é:* Pacote Python para meu projeto
* *Para que serve:* alvo de futuras importações que devem conter o nome deste pacote (p. ex.: mysite.urls) – **recebe importações.**

1. **\_\_init\_\_.py:**

* *O que é:* Arquivo vazio.
* *Para que serve:* Dizer ao Python que o diretório no qual está presente (*mysite/* no caso) **deve ser considerado um pacote Python.**

1. **Settings.py – Configurador do Django:**

* *O que é:* Arquivo Python.
* *Para que serve:* Configurar o projeto.
* *Descrição:* Possui variáveis de módulo representando as configurações do Django. O SQLite é instalado por padrão no Python e, por consequência, no Django.

1. **Urls.py – Índice do Site:**

*O que é:* Arquivo Python

*Para que serve:* armazenar as declarações de URLS que compõem seu projeto Django. Servem como um **índice do seu site** estruturado em Django.

1. **ASGI.py - Integrador de Serviços Web ASGI:**

*O que é: Arquivo Python*

*Para que serve:* Integrar servidores web compatíveis com ASGI. Está ligado com o **deploy** do projeto.

1. **WSGI.py - Integrador de Serviços Web WSGI:**

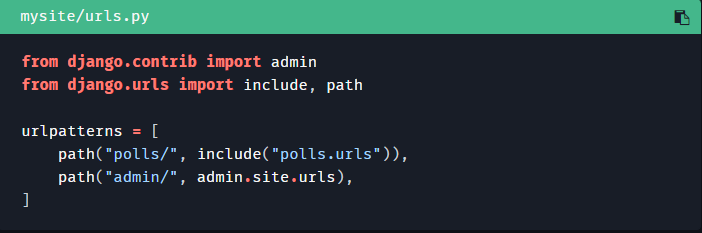
*O que é:* Arquivo Python

*Para que serve:* Da mesma forma que o ASGI.py, integra servidores web compatíveis com WSGI. Está ligado com a **implementação** do projeto.

*Solicitação não requere reinicialização manual do servidor porque faz isso automaticamente, mas adição de arquivos necessita.*

**Aplicação x Projeto**

|  |  |
| --- | --- |
| Aplicação - Unidade | Projeto - Conglomerado |
| * Aplicativo da web com alguma função (blog, banco de dados, registro público, aplicativo de pesquisa, etc.); * Pode estar em vários projetos. | * Coleção de configurações e aplicativos para determinado site; * Pode conter vários aplicativos. |



* **Include:**
* *O que é:* Função
* *Para que serve:* Referenciar URLconfs e recortá-las até o ponto. Facilita o plug entre URLs.
* *Quando usar:* Quando usar URLs com outros padrões.
* **Path:**
* *O que é:* Função
* *Para que serve:* Retornar um elemento através da rota (Route) especificada entre parênteses.
* *Subvidisão:* Possui quatro argumentos:

1. **Obrigatórios:**
2. Route: É uma string que contém uma descrição de URL. Serve para fornecer uma rota devida para uma requisição (GET ou POST) do Django.
3. View: É uma função que processa uma solicitação HTTP, busca os dados necessários no banco de dados, renderiza os dados em uma página HTML usando um modelo HTML e, em seguida, retorna o HTML gerado em uma resposta HTTP para exibir a página ao usuário.
4. **Opcionais:**
5. Kwargs: Argumento que permite passar argumentos adicionais ao método ou função view.
6. Name: Recurso do Dajngo que permite referenciação inequívoca em todos os campos dele. Permite alterações globais nos padrões de URL alterando um só arquivo.

**Parte 2**

1. **Settings: Configuração da Linguagem e Fuso Horário**

LANGUAGE\_CODE = 'pt-br'

TIME\_ZONE = 'GMT-3'

1. **INSTALLED\_APPS:**

*O que é:* Lista de Strings

*Para que serve:* Mostra todas as aplicações ativas na instalação Django  
*Descrição:* Cada string é um **caminho** feito através de pontos (p.ex. django.contrib.admin). Esse **caminho** leva para uma **classe de configuração** ou um **pacote que contem o aplicativo.**

* **Aplicativos padrão do INSTALLED\_APPS:**

1. django.contrib.admin – Site de administração;
2. django.contrib.auth – Sistema de autenticação;
3. django.contrib.contenttypes – Framework para tipos de conteúdo;
4. dango.contrib.sessions – Framework de sessão;
5. django.contrib.messages – Framework de envio de mensagens;
6. django.contrib.staticfiles – Framework que gerencia arquivo estáticos.
7. **FIELD:**

**MIGRATE  
  
Analisa** a configuração do INSTALLED\_APPS e **cria as linhas do bando de dados** de acordo com as configurações do BD especificadas em *mysite/settings.pt* e as migrações de banco de dados enviadas com o aplicativo (ver isso depois).

**MODELS(modelos)**

Fonte única e definitiva de informações sobre seus dados. Objetiva definir os modelos de dados num só lugar e derivar coisas desse lugar de maneira automática. Isso ocorre porque o Django procura estruturar seus elementos de acordo com o **princiípio DRY** que diz que cada conceito distinto e/ou dado deve estar em um, e apenas um, lugar. Redundância é ruim. Normalização é bom.

**MAKEMIGRATIONS:**

Ao executar makemigrations, você está dizendo ao Django que você fez algumas mudanças em seus modelos (neste caso, você fez novas modificações) e que você gostaria que as alterações sejam armazenadas como uma migração.

*O que é:* Classe

*O que faz:* Define tipo primitivo dos dados inseridos nas

variáveis (Integer, Char, Float, Boolean, etc).

1. **Migrações:**

*Como o Django armazena as alterações que faço no*

*meu model? –* Justamente através do comando *migrations.*

Ele pega todas as migrações que ainda não foram aplicadas e

as leva para seu bando de dados

**Mudança nos Modelos: 3 Passos Básicos:**

1. **Mude** seus modelos em *models.py;*
2. **Migre** as alterações:

*python manage.py makemigrations*

1. **Aplique** as alterações:

*python manage.py migrations*

**Uma Aplicação em Vários Projetos**

É possível usar uma mesma aplicação em vários projetos porque ela

não é atrelada a um projeto. Portanto, as aplicações Django são

plugáveis.

**5. Dizer ao Django que a aplicação Polls está instalada:**

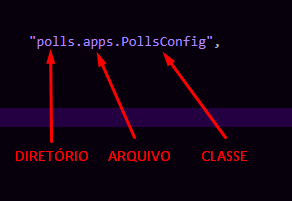
É necessário dizer para o Django que a aplicação polls está instalada.

Como fazemos isso? Através de uma **referenciação**. Precisamos

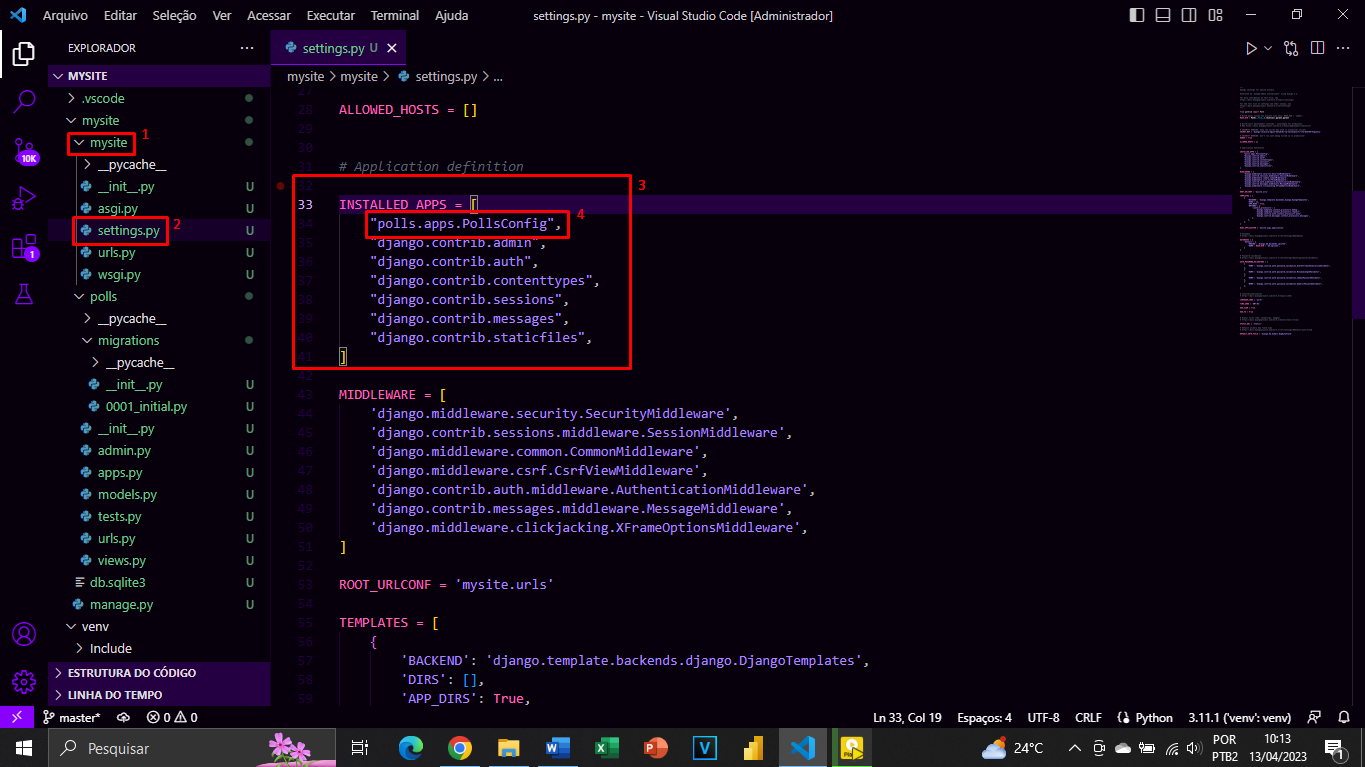
Referenciar em INSTALLED\_APPS a classe PollsConfig que está

presente no arquivo apps que por sua vez está dentro do diretório

Polls.



Esta referenciação está presente na **lista de strings INSTALLED\_APPS**, presente no **arquivo settings** presente no **diretório interno mysite.**



**Filosofias de Projeto**

**Geral**

**Acoplamento Fraco:** *Uma parte não deve “saber” da outra, salvo em casos extremamente necessários*.

O Django segue o princípio do **acoplamento fraco e coesão forte.**

**Quanto menos melhor**

O Django seque a filosofia de usar o mínimo possível de linhas de código, seguindo a característica **introspectiva** do Python.

**Desenvolvimento rápido**

O Django objetiva proporcionar um desenvolvimento rápido em relação a processos lentos e tediosos no desenvolvimento de sites para a internet.

Redundância é ruim. Normalização é bom. O sistema deve, razoavelmente, deduzir o máximo possível do mínimo possível.

**Não repita a si mesmo (Dont Repeat Yourself – DRY)**

**Explítico é Melhor que Implícito**

É um dos 19 princípios do Zen do Python. O código deve ser claro e compreensível sem muitos rodeios ou mágica. Esta falta de clareza deve ser evitada ao máximo, ainda que não seja possível abandonar o uso de termos não muito claros. Isso visa facilitar a vida de desenvolvedores que estão aprendendo novas funcionalidades.

**Consistência**

Consistência deve nortear todos os campos de desenvolvimento do Django, desde níveis mais baixos até mais altos.

**Modelos**

**Explícito Melhor que Implícito:**

Por uma questão de precaução contra erros, o comportamento de determinados campos não devem ser construídos pensando-se somente no nome deles e sim em outros **argumentos nomeados**  ou **no tipo do campo.**

**Incluir toda a Lógica do Domínio Relevante**

Este princípio discorre sobre a necessidade de facilitar-se a leitura de determinado objeto pela descrição de seus dados e comportamentos no campo *models* do Django. Este princípio segue o **Active Record.**

**API de Banco de Dados**

**Eficiência SQL:**

Visa executar comandos SQL o mínimo de vezes possível e otimizar comandos de forma interna através do *save()* para salvamento por linha de comando ou *selected\_related() QuerySet* que faz um bagulho que eu não entendi direito o que é, por exemplo.

**Sintaxe Concisa e Poderosa:**

A API do banco de dados deve ser a mais concisa, rica e expressiva possível, permitindo o máximo possível de acesso aos outros campos do sistema por meio de uma única linha de código. Para ajudar nesse trabalho, a função Join() pode ser usada, pois ela junta strings.

**Usar SQL Nativo**

O framework do Django deve permitir o uso de uma espécie de SQL personalizado, quando necessário.