Django

O que é?

Framework para construção de aplicações web com Python

Coleção de componentes de software reutilizáveis que tornam mais eficiente o desenvolvimento de novas aplicações.

Conjunto de bibliotecas, que incluem funcionalidades e estruturas para o desenvolvimento de aplicações, a fim de fornecer soluções para o mesmo domínio de problema, permitindo a reutilização do seu código.

Django

Mais algumas vantagens...

gratuito

código aberto

ampla documentação

Django

Onde encontrar o framework...

https://www.djangoproject.com

Django

Preparação do ambiente de desenvolvimento...

VSCode (IDE para desenvolvimento) 

Python (Linguagem) 

Pip (Gerenciador de pacotes) 

Virtual env (responsável por virtualizar um ambiente onde serão instaladas as dependências do projeto) 

Django framework 

Django

Preparação do ambiente de desenvolvimento…

Virtual env (responsável por virtualizar um ambiente onde serão instaladas as dependências do projeto)

Para verificar se o virtualenv está instalado, no terminal

where virtualenv

Se não estiver instalado

pip install virtualenv

Django

Virtual env (responsável por virtualizar um ambiente onde serão instaladas as dependências do projeto)

Para criar o ambiente virtual

Criar uma pasta para o projeto

virtualenv env

Para ativar o ambiente

source env/bin/activate

(MAC)

source env/Scripts/activate

(Windows)

Django

Instalando as dependências...

Django framework 

```
pip install Django==3.0.0
```

Django

Iniciando o projeto...

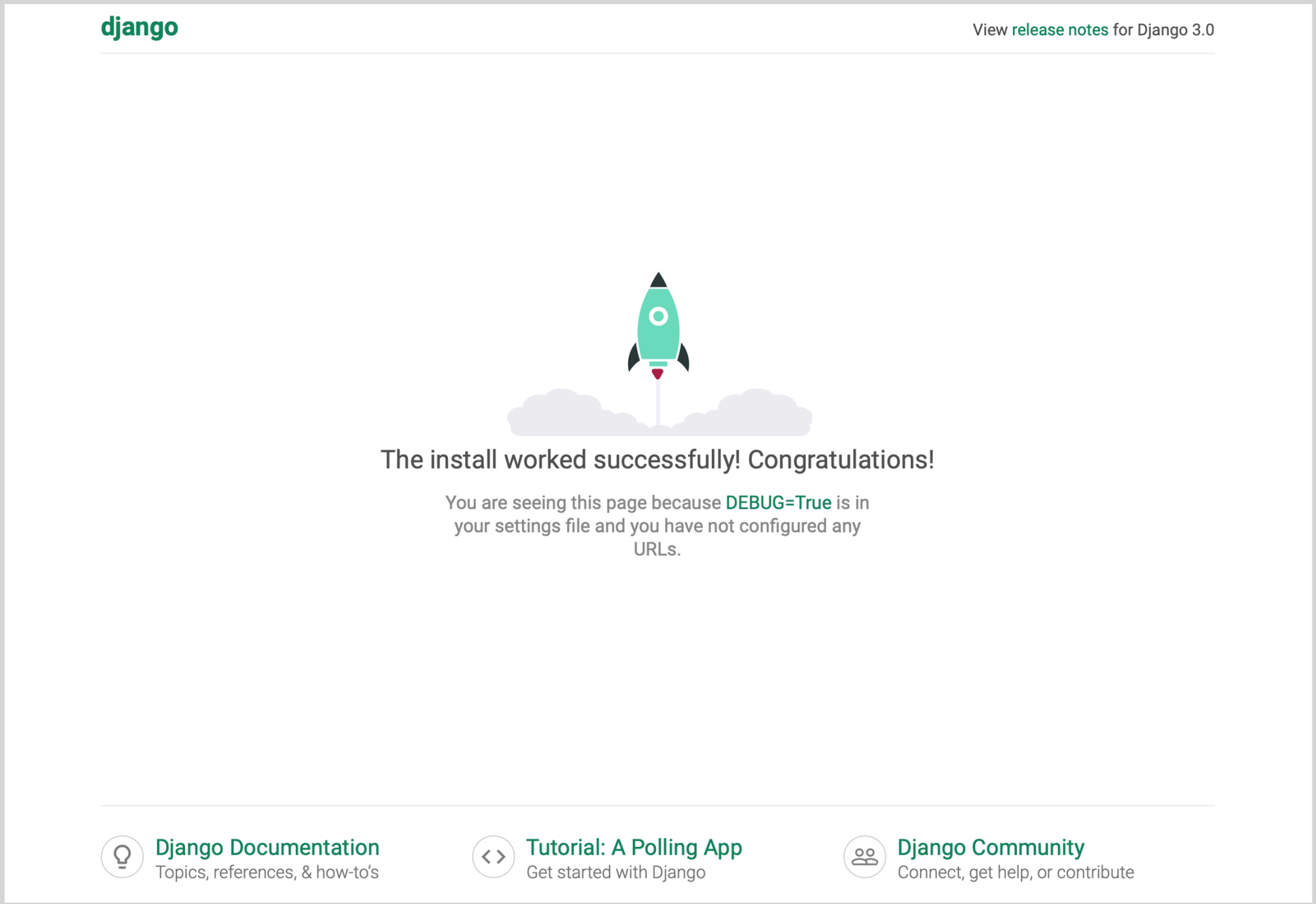
```
django-admin startproject hello .
```

Para inicializar o servidor...

```
python manage.py runserver
```

Django

No browser, use o endereço fornecido para testar se tudo está ok.



Django

Entendendo a estrutura do projeto...

manage.py	utilitário de linha de comando para interagir com um projeto Django
pasta do projeto	pasta que agrupa o pacote Python referente a este projeto (o nome não deve ser alterado)
init.py	indica que esta pasta deve ser reconhecida e tratada como um pacote
settings.py	arquivo de configurações do projeto
urls.py	arquivo de urls do projeto. Para que uma url esteja acessível é preciso declará-la neste arquivo
wsgi.py	arquivo de integração para servidores web que implementam o padrão wsgi
asgi.py	arquivo de integração para servidores web com suporte a comunicação assíncrona

Django

Criando o primeiro aplicativo...

python manage.py startapp index

Cada arquivo criado nessa nova pasta tem a sua responsabilidade e pode ou não representar uma camada na arquitetura do projeto.

É necessário informar para o projeto que a pasta criada é um aplicativo e que a mesma faz parte do projeto.

Django

Criando o primeiro aplicativo...

- Abrir o arquivo settings.py
- Procurar a variável INSTALLED_APPS
- Adicionar no final da lista o nome do aplicativo criado (ou adicionar o novo app numa seção separada dos aplicativos padrão)

```
INSTALLED_APPS += [
  'index',
]
```

Django

Criando o primeiro aplicativo...

- Abrir o arquivo views.py (no aplicativo criado). Essa camada é responsável por encapsular a lógica que recebe e responde as requisições e pode ou não definir comportamentos específicos, bem como buscar informações no banco de dados. Toda view no Django é uma função de retorno que é vinculada a uma url configurada no arquivo de urls. Sendo assim não existe uma url sem que exista uma função de view para ser retornada quando essa url for acessada pelo navegador.
- Apagar os comentários
- Importar HttpResponse

```
from django.http import HttpResponse
```

- Criar uma função

```
def index(request):
    return HttpResponse("Olá Mundo!")
```

- Mapear a função, abrir urls.py
- Apagar os comentários
- Substituir as aspas simples por aspas duplas
- Importar a função que foi escrita

```
from hello.views import index
```

- Registrar a url usando a função path (adicionar)

```
path("", index, name="index"),
```

Django

Criando o primeiro aplicativo...

Abrir o terminal e inicializar o servidor de desenvolvimento

```
python manage.py runserver
```

Django

E como usar um HTML na função de retorno???

Para isso, é necessário usar a função render ao invés da função HttpResponse. Além disso é necessário criar uma pasta para os templates.

return render(request, “html")

IMPORTANTE: a pasta templates deve ser criada na raiz da aplicação e deve ter exatamente este nome: templates

Django

1 - criar a pasta templates na raiz da aplicação (index, neste exemplo)

2 - criar na pasta templates uma pasta para os templates da aplicação

3 - criar um arquivo (index.html, neste exemplo)

4 - abrir o arquivo views.py da aplicação

5 - verificar o import: from django.shortcuts import render (se não estiver, faça o import)

6 - return render(request, “index.html”)