**SEU PRIMEIRO PROJETO EM DJANGO**

Usando o promp/terminal do seu computador, selecione o local da máquina que será utilizado como repositório do seu código. Após a escolha, cria-se primeiro o ambiente virtual por emprego do comando

python -m venv venv

E em seguida, para usuários Windows, usar o comando abaixo para ativar o ambiente:

venv\Scripts\activate

Caso tenha ativado, você notará que a palavra (venv) estará antes da rota em que seu terminal está posicionado.

Com o ambiente ativo, a próxima etapa consiste em instalar o Django e, para isso, utilize o código abaixo:

pip install django psycopg2-binary

A próxima etapa consiste na criação do seu projeto:

django-admin startproject nomeDoSeuProjeto

e em seguida acesse a pasta do seu projeto por emprego do comando:

cd nomeDoSeuProjeto

-------------------------------------------------------------------------------------------------------

Configurando a aplicação para acessar o banco de dados

No arquivo settings.py busque pela variável DATABASE, a qual terá as configurações de acesso ao banco de dados. Nela temos uma configuração inicial, então a substituímos pela configuração abaixo:

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.postgresql',

        'NAME': 'meubanco',

        'USER': 'meuusuario',

        'PASSWORD': 'minhasenha',

        'HOST': 'localhost',

        'PORT': '5432',

    }

}

Agora, na pasta do seu projeto (nomeDoSeuProjeto) utilize o comando abaixo para criar um aplicativo Django.

python manage.py startapp meuapp

A próxima etapa é disponibilizar o seu meuapp nas configurações e, para isso, acesse o arquivo settings.py e localize a variável INSTALLED\_APPS e insira o meuapp, exemplo abaixo:

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'meuapp',

]

--------------------------------------------------------------------------------------------------------

Sua aplicação está minimamente configurada, agora vamos iniciar a criação dos models que são as classes que espelham tabelas do banco de dados e, para isso, acessamos o ‘meuapp’ e, por emprego do seu editor (IDE), abra o arquivo model.py, abaixo segue um exemplo de como implementar.

from django.db import models

class Cliente(models.Model):

    nome = models.CharField(max\_length=100)

    email = models.EmailField(unique=True)

    idade = models.IntegerField()

    def \_\_str\_\_(self):

        return self.nome

Agora, dentro da pasta nomeDoMeuProjeto, dê o comando para aplicar as migrations, segue abaixo:

python manage.py makemigrations

python manage.py migrate

O Primeiro comando cria arquivos de migração baseados nas mudanças que você fez nos **models** do Django. Já o segundo, com final migrate, **executa** os arquivos de migração e aplica as mudanças no banco de dados.

Próxima etapa consiste em criar um super usuário por emprego do comando:

python manage.py createsuperuser

Após isso será solicitado a inserção do login, email e senha.

A próxima etapa consiste em registrar no Django admin e, para isso, acessamos o admin.py que está na pasta meuapp. Ao acessá-lo utilize as configurações abaixo:

from django.contrib import admin

from .models import Cliente

admin.site.register(Cliente)

Agora, basta executar sua aplicação por emprego do comando abaixo, não se esqueça que o comando deve ser aplicado no terminal e na pasta nomeDoSeuProjeto e não no meu app.

python manage.py runserver

E acessar a rota: http://127.0.0.1:8000/admin

Com isso tivemos uma primeira imersão dentro do framework, essa página consistiu em apenas criar uma aplicação para rodar a interface de super usuário do Django.   
  
Vamos para um segundo exemplo, o qual consiste em listar um conjunto de dados de pessoas salvas no banco de dados PostgreSQL.

--------------------------------------------------------------------------------------------------------

Já com um novo projeto criado, implementamos o nosso app denominado pessoas, por emprego do comando:

python manage.py startapp pessoas

Além disso precisamos instalar o rest framework e, para isso, utilize o comando abaixo:

pip install djangorestframework

Seguimos com a configuração do banco de dados, como supracitado, mas no installed apps, precisamos inserir tanto a nossa aplicação pessoas, quanto o framework, tal como no exemplo abaixo:

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'pessoas',

    'rest\_framework',

]

O Nosso modelo consiste em:

from django.db import models

class Pessoa(models.Model):

    nome = models.CharField(max\_length=100)

    idade = models.IntegerField()

    def \_\_str\_\_(self):

        return self.nome

Agora basta rodar as migrations.

--------------------------------------------------------------------------------------------------------

Com as configurações do projeto já concluídas, iniciamos a implementação do REST e iniciamos criando o arquivo serializers.py dentro da pasta pessoas. O conteúdo desse serializers está disponível abaixo:

from rest\_framework import serializers

from .models import Pessoa

class PessoaSerializer(serializers.ModelSerializer):

    class Meta:

        model = Pessoa

        fields = '\_\_all\_\_'

Agora, no arquivo pessoas/views.py, implementamos o código abaixo:

from django.shortcuts import render

from rest\_framework import viewsets

from .models import Pessoa

from .serializers import PessoaSerializer

class PessoaViewSet(viewsets.ModelViewSet):

    queryset = Pessoa.objects.all()

    serializer\_class = PessoaSerializer

E agora definimos a url (end-point) que disponibilizará os dados e essa configuração deverá ser feita em pessoas/urls.py, segue código abaixo:

from django.urls import path, include

from rest\_framework.routers import DefaultRouter

from .views import PessoaViewSet

router = DefaultRouter()

router.register(r'pessoas', PessoaViewSet)

urlpatterns = [

    path('api/', include(router.urls)),

]

Por fim, também configuramos o arquivo meu\_projeto/urls.py com o seguinte código:

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

    path('admin/', admin.site.urls),

    path('', include('pessoas.urls')),

]

Com essa implementação já seria possível consultar os dados de pessoas usando a url http://localhost:8000/api/pessoas/, mas para isso faça a inserção de dados no banco de dados.

--------------------------------------------------------------------------------------------------------

Agora podemos criar um template para exibir os dados, começando pela implementação do arquivo pessoas/templates/pessoas/lista.html

No arquivo use o exemplo HTML abaixo:

<!DOCTYPE html>

<html lang="pt">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Lista de Pessoas</title>

</head>

<body>

    <h1>Lista de Pessoas</h1>

    <ul>

        {% for pessoa in pessoas %}

            <li>{{ pessoa.nome }} - {{ pessoa.idade }} anos</li>

        {% endfor %}

    </ul>

</body>

</html>

Agora vamos configurar uma nova rota para acessar o template, portanto em pessoas/views.py insira:

from .models import Pessoa

def lista\_pessoas(request):

    pessoas = Pessoa.objects.all()

    return render(request, 'pessoas/lista.html', {'pessoas': pessoas})

O resultado total da implementação nova com a anterior, temos:

from django.shortcuts import render

from rest\_framework import viewsets

from .models import Pessoa

from .serializers import PessoaSerializer

from .models import Pessoa

class PessoaViewSet(viewsets.ModelViewSet):

    queryset = Pessoa.objects.all()

    serializer\_class = PessoaSerializer

def lista\_pessoas(request):

    pessoas = Pessoa.objects.all()

    return render(request, 'pessoas/lista.html', {'pessoas': pessoas})

Agora, para finalizar, editaremos o arquivo pessoas/urls.py inserindo o código abaixo:

from django.urls import path, include

from rest\_framework.routers import DefaultRouter

from .views import PessoaViewSet

from django.urls import path

from .views import lista\_pessoas

router = DefaultRouter()

router.register(r'pessoas', PessoaViewSet)

urlpatterns = [

    path('api/', include(router.urls)),

]

urlpatterns += [

    path('pessoas/', lista\_pessoas, name='lista\_pessoas'),

]

Por fim você poderá inicializar sua aplicação e acessar a URL <http://localhost:8000/pessoas/> em seu navegador.

Esses códigos estão disponíveis em: https://github.com/wbogler/django-desco