

Pontos turísticos cadastrados funcionarão da seguinte maneira:

Cada ponto turístico terá apenas uma localização;

Esse ponto poderá conter várias atrações;

Esse ponto poderá ter várias avaliações;

Cada avaliação deverá ‘servir’ apenas para um ponto turístico por vez;

Cada atração poderá existir em um único ponto turístico por vez;

Cada localização corresponderá a apenas um ponto turístico;

Um usuário poderá realizar várias avaliações sobre o mesmo ponto turístico;

Uma avaliação poderá corresponder a apenas um usuário;

Cada ‘quadrado’ desses corresponderá a uma API que será criada.

Para iniciar a criação das API temos que executar no terminal o comando python manage.py startapp [nome da api].

No caso mudamos o nome da API pontos t. para core.

Todas as vezes que fizermos alguma alteração temos que aplicar os comandos:

Python manage.py makemigrations

Python manage.py migrate

Para rodar o servidor iremos usar

Python manage.py runserver

Passo a passo:

Criar a APP

Python manage.py startapp [nome da app]

Isso vai criar uma pasta nova com o nome dessa app.

Ir no arquivo settings.py e registrar.

Dentro dessa pasta iremos realizar os passos a seguir:

Definir um modelo para ela (arquivo models.py)

from django.db import models

from django.contrib.auth.models import User

# Create your models here.

#Aqui, ao que entendi, criamos os endpoints que queremos ter no site e, para cada um deles, criamos suas características

class PontoTuristico(models.Model):

    nome = models.CharField(max\_length=255)

    descricao = models.TextField(blank=True, null=True)

    aprovado = models.BooleanField(default=False)

    class Meta:

        ordering = ('nome',) #Aqui ordenamos as categorias em ordem alfabética.

        verbose\_name\_plural = 'Pontos Turísticos'

    def \_\_str\_\_(self):

        return self.nome

Registrar ela no arquivo admin.py

from django.contrib import admin

# Register your models here.

#Aqui, pelo que entendi, registramos as API que criamos, para que elas façam parte do servidor que criamos.

from .models import PontoTuristico

admin.site.register(PontoTuristico)

Depois de fazer isso, dentro da pasta core, ele manda criar uma pasta chamada API e, dentro dela, 3 arquivos, que serão \_\_init\_\_.py serializers.py viewsets.py

A partir desse momento, vamos conseguir criar e modelar as aplicações que estão atreladas ao módulo principal, que serão as atrações, a localização, as avaliações e os usuários.

Vamos começar pelas atrações.

Python manage.py startapp atracoes

Depois disso, vai no arquivo settings.py e escreve o nome dela no campo INSTALLED\_APPS

Agora vamos começar a mexer nos arquivos da pasta atrações.

Vamos primeiro em models.py

from django.db import models

# Create your models here.

class Atracao(models.Model):

    nome = models.CharField(max\_length=150)

    descricao = models.TextField()

    horario\_func = models.TextField()

    idade\_minima = models.IntegerField()

    def \_\_str\_\_(self):

        return self.nome

Nisso criamos as características desse objeto ‘atração’.

Agora vamos no arquivo admin.py, ainda dentro da pasta atrações e vamos registrar esse app

from django.contrib import admin

from .models import Atracao

# Register your models here.

admin.site.register(Atracao)

Agora temos que criar o banco de dados, criar as migrações e migrar. (makemigrations, migrate)

ATENÇÃO que depois de criar o banco de dados, pra alterar, fica dando erro porque as colunas novas ou alteradas não estão registradas no banco de dados. Tem que ficar atento e descobrir, de repente, como alterar o banco de dados depois de sua criação (o que pode ser traumático).

Depois disso, ainda dentro da pasta atrações, vamos criar novamente a pasta API e dentro dela aqueles 3 arquivos criados anteriormente (\_\_init\_\_.py serializers.py viewsets.py).

Agora temos que fazer ela ser importada na aplicação Core.

Vamos em models.py da pasta Core e vamos fazer o import dela.

Em princípio, a app atrações está criada. Agora vamos para a próxima app.

App comentários e avaliações.

Python manage.py startapp comentarios

Registra no arquivo settings.py

Agora, arquivo models.py dentro da pasta comentarios

Criar as características ou atributos que os comentários vão possuir.

from django.db import models

from django.contrib.auth.models import User

# Create your models here.

class Comentario(models.Model):

    usuario = models.ForeignKey(User, on\_delete=models.CASCADE)

    comentario = models.TextField()

    data = models.DateTimeField(auto\_now\_add=True)

    aprovado = models.BooleanField(default=False)

    def \_\_str\_\_(self):

        return self.usuario.first\_name

Após isso, ir no arquivo admin.py da pasta comentarios e registrar.

Copiar e colar a pasta api criada anteriormente (com arquivos sem nada escrito) nesta pasta comentarios, pois todas sempre terão essa pasta api.

Para essa aplicação, foi decidido criar um novo app para as avaliações, separado dos comentários. Vou tentar criar sozinho agora, utilizando meu passo a passo. Senão vejamos!

Deu boa!

Agora vamos iniciar de fato a criação das API (são interfaces que permitem a comunicação).

Não entendi bem... Mas basicamente, na pasta do Projeto, no arquivo urls.py vamos ter isso:

from django.contrib import admin

from django.urls import path

from rest\_framework import routers

from core.api.viewsets import PontoTuristicoViewSet

from django.conf.urls import include

router = routers.DefaultRouter()

router.register('pontoturistico', PontoTuristicoViewSet)

urlpatterns = [

    path('', include(router.urls)),

    path('admin/', admin.site.urls),

]

Na pasta API de cada uma das aplicações, pelo que entendi, vamos ter isso:

Arquivo viewsets.py

from rest\_framework.viewsets import ModelViewSet

from core.models import PontoTuristico

from .serializers import PontoTuristicoSerializer

class PontoTuristicoViewSet(ModelViewSet):

    queryset = PontoTuristico.objects.all()

    serializer\_class = PontoTuristicoSerializer

No arquivo serializers.py

from rest\_framework.serializers import ModelSerializer

from core.models import PontoTuristico

class PontoTuristicoSerializer(ModelSerializer):

    class Meta:

        model = PontoTuristico

        fields = ('id', 'nome', 'descricao') #Não inserir muitos campos pois o serializer vai ficar muito pesado e fica lento de abrir no celular ou outro dispositivo.