INSTITUTO TECNOLÓGICO DEL PUTUMAYO
MATERIA: PROGRAMACIÓN AVANZADA
SEMESTRE: 9
SESIÓN: 4
DOCENTE: ING. FREDY JACANAMIJOY
11/09/2023

PASOS BÁSICOS: CREACIÓN DE APIREST (DJANGO)

Primero justar versiones de software requeridos en la fecha correspondiente; por ejemplo a 11/09/2023, los requisitos de python y django son:

- Python (3.6, 3.7, 3.8, 3.9, 3.10)
- Django (2.2, 3.0, 3.1, 3.2, 4.0, 4.1)

De lo contrario pueden aparecer conflictos de compatibilidad.

CREACIÓN DE LA API

Para crear una API en Django con el modelo de base de datos que has proporcionado, necesitarás seguir varios pasos. Aquí te proporciono instrucciones generales desde el inicio de un proyecto Django hasta la creación de servicios API utilizando el framework Django REST framework.

****Paso 1: Configurar un entorno virtual****

1. Abre tu terminal y navega hasta el directorio en el que deseas crear tu proyecto Django.

2. Crea un entorno virtual para aislar las dependencias del proyecto:

```
python -m venv nombre_del_entorno
```

3. Activa el entorno virtual:
   - En Windows: `nombre_del_entorno\Scripts\activate`
   - En macOS y Linux: `source nombre_del_entorno/bin/activate`

Paso 2: Instalar Django y Django REST framework

Dentro de tu entorno virtual, instala Django y Django REST framework:

```
pip install django==3.2
```

```
WARNING: You are using pip version 21.2.3; however, version 23.2.1 is available.
You should consider upgrading via the 'D:\PROGRAMACION_PROYECTOS\apirest_django_tareas\entorno_task\Scripts\python.exe -m pip install --upgrade pip' command.
```

Indica actualización necesaria de pip, copiamos el comando que provee (Python.exe -m pip install –upgrade pip)

```
pip install djangorestframework
pip install mysqlclient
pip install pymysql
```

Paso 3: Crear un proyecto Django y una aplicación

1. Crea un nuevo proyecto Django:

```
  django-admin startproject nombre_del_proyecto
```

2. Navega al directorio del proyecto:

```
cd nombre_del_proyecto
```

3. Crea una nueva aplicación dentro del proyecto:
   `python manage.py startapp nombre_de_la_app`

NOTA: Poner las aplicaciones instaladas tal como en el siguiente código previo al paso 4 en settings.py

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'tareas_app', #no coloco carpeta, porque está en la raíz

]
```

Paso 4: Configurar la base de datos

Edita el archivo `settings.py` en tu proyecto para configurar la base de datos. Reemplaza la configuración de `DATABASES` con lo siguiente:

```python
DATABASES = {
    'default': {
     #   'ENGINE': 'django.db.backends.sqlite3',
      #  'NAME': BASE_DIR / 'db.sqlite3',
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'tareas_db',
        'USER': 'root',
        'PASSWORD': '1234',
        'HOST': 'localhost', # Puede variar según tu configuración
        'PORT': '3306', # Puerto de MySQL
        'OPTIONS': {
            'init_command': "SET sql_mode='STRICT_TRANS_TABLES'"
        }
    }
}
```

```

Asegúrate de haber creado previamente la base de datos `nombre_base_de_datos` en MySQL y de tener acceso con el usuario y contraseña proporcionados.
```

INSTITUTO TECNOLÓGICO DEL PUTUMAYO
MATERIA: PROGRAMACIÓN AVANZADA
SEMESTRE: 9
SESIÓN: 4
DOCENTE: ING. FREDY JACANAMIJOY
11/09/2023

Paso 5: Definir el modelo de datos

Edita el archivo `models.py` en la aplicación que creaste (`nombre_de_la_app`) para definir los modelos de datos. Agrega los modelos `Proyecto`, `Usuario` y `Tarea` según la estructura que has proporcionado en tu pregunta.

```python
# models.py
```

```python
# Create your models here.
from django.db import models

class Proyecto(models.Model):
    nombre = models.CharField(max_length=75)
    descripcion = models.TextField()
    fecha_inicio = models.DateField()
    fecha_finalizacion = models.DateField()
class Usuario(models.Model):
    nombre = models.CharField(max_length=100)
    correo_electronico = models.CharField(max_length=50)
    contrasena = models.CharField(max_length=255)
class Tarea(models.Model):
    nombre = models.CharField(max_length=75)
    descripcion = models.TextField()
    fecha_vencimiento = models.DateField()
    estado = models.SmallIntegerField()
    idusuario = models.ForeignKey(Usuario, on_delete=models.CASCADE)
    idproyecto = models.ForeignKey(Proyecto, on_delete=models.CASCADE)
```

NOTA: Ingresar en admin.py el código de importación de las clases según las que hayas creado en la sintaxis anterior. Por ejemplo para este caso sería:

```python
from django.contrib import admin
from .models import Proyecto
from .models import Usuario
from .models import Tarea

# Register your models here.
admin.site.register(Proyecto)
admin.site.register(Usuario)
admin.site.register(Tarea)
```

INSTITUTO TECNOLÓGICO DEL PUTUMAYO
MATERIA: PROGRAMACIÓN AVANZADA
SEMESTRE: 9
SESIÓN: 4
DOCENTE: ING. FREDY JACANAMIJOY
11/09/2023

Paso 6: Crear las migraciones y aplicarlas

Aplica las migraciones:

python manage.py migrate

Verifiar en la APP, carpeta migrations, archivo de nombre: 0001_initial.py

CREAR SÚPER USUARIO

python manage.py createsuperuser

Crear los datos solicitados como nombre de usuario, correo y contraseña.

Genera las migraciones para crear las tablas en la base de datos:

python manage.py makemigrations

Arrancar servidor

python manage.py runserver

rutra: http://localhost:8000/
Verifica el funcionamiento correcto.

http://localhost:8000/admin/
Administra tu APP gráficamente para alimentar la BD

INSTITUTO TECNOLÓGICO DEL PUTUMAYO
MATERIA: PROGRAMACIÓN AVANZADA
SEMESTRE: 9
SESIÓN: 4
DOCENTE: ING. FREDY JACANAMIJOY
11/09/2023

****Paso 7: Crear las vistas y las serializaciones****

Crea vistas y serializaciones para tus modelos en `**views.py**` y `serializers.py` dentro de tu aplicación. Estos archivos definirán cómo se presentarán los datos y cómo se accederán a través de la API.

```python
#from django.shortcuts import render #renderiza plantillas, pero inicialmente no es necesario
from typing import Any
from django import http
from django.http.response import JsonResponse
from django.views import View ##
from .models import Usuario, Tarea, Proyecto
from django.utils.decorators import method_decorator
from django.views.decorators.csrf import csrf_exempt
import json

class UsuarioView(View): #esta es la clase solo para usuario, aplica lo mismo para Tarea y Proyecto

    @method_decorator(csrf_exempt)
    def dispatch(self, request, *args: Any, **kwargs):
        return super().dispatch(request, *args, **kwargs)

    def get(self,request,id=0):
        if(id>0):
            usuarios=list(Usuario.objects.filter(id = id).values())
            if len(usuarios) > 0:
                usuario = usuarios[0]
                datos = {'message': "Succes", 'usuarios': usuarios }
            else:
                datos = {'message': "Usuario no encontrado ..." }
            return JsonResponse(datos)

        else:
            usuarios = list(Usuario.objects.values())
            if len(usuarios) > 0:
                datos = {'messaje':'Succes','usuarios':usuarios}
            else:
                datos = {'messaje': "Usuarios no encontrados..."}
        return JsonResponse(datos)

    def post(self,request):
        #print(request.body)
        jd = json.loads(request.body)
        #print(jd)
        Usuario.objects.create(nombre=jd['nombre'] , correo_electronico = jd['correo_electronico'], contrasena =
jd['contrasena'])
        datos = {'message':"Success"}
        return JsonResponse(datos)

    def put(self,request,id):
        jd = json.loads(request.body)
        usuarios = list(Usuario.objects.filter(id = id).values())
        if len(usuarios) > 0:
            usuario = Usuario.objects.get(id = id)
            usuario.nombre = jd['nombre']
            usuario.correo_electronico = jd['correo_electronico']
            usuario.contrasena = jd['contrasena']
            usuario.save()#para guardar los cambios
            datos = {'message': "Success"}
            return JsonResponse(datos)
        else:
            datos = {'message': "Usuarios no encontrado... "}

    def delete(self,request, id):
        usuarios = list(Usuario.objects.filter(id = id).values())
        if len(usuarios) > 0:
            Usuario.objects.filter(id = id).delete()
            datos = {'message': "Success"}
        else:
            datos = {'message': "Usuario no encontrado ..."}
        return JsonResponse(datos)
```

INSTITUTO TECNOLÓGICO DEL PUTUMAYO
MATERIA: PROGRAMACIÓN AVANZADA
SEMESTRE: 9
SESIÓN: 4
DOCENTE: ING. FREDY JACANAMIJOY
11/09/2023

Paso 8: Configurar las URL

Crea un archivo `urls.py` en tu aplicación y configura las rutas de URL para tus vistas. También, asegúrate de incluir estas rutas en el archivo `urls.py` del proyecto principal.

```
"""tareas URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/3.2/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path, include, re_path  #se agregó include, necesario  para
incluilr lo que tenga tareas_app.urls (ruta: nombre de app y archivo urls.py)
from tareas_app import urls
from tareas_app import views


urlpatterns = [
    path('admin/', admin.site.urls),
    path('tareas_app/',include('tareas_app.urls')), #buscará el archivo urls.py dentro
de la app tareas_app
]
```

Todo lo anterior estás en la urls.py del PROYECTO, pero en el proyecto, pero en la APP existe otro archivo llamado también urls.py (si no existe crearlo), ahí codificamos así:

```
#urls.py APP
from django.urls import path, re_path
from .views import TareaView,ProyectoView, UsuarioView

urlpatterns=[

    path('listar_usuarios/',UsuarioView.as_view(),name='listar_usuarios'),
    path('listar_usuarios/<int:id>',UsuarioView.as_view(),name='listar_usuarios'),
    path('listar_proyectos/',ProyectoView.as_view(),name='listar_proyectos'),
    path('listar_proyectos/',TareaView.as_view(),name='listar_tareas'),

]
```

Paso 9: Ejecutar el servidor de desarrollo

Inicia el servidor de desarrollo para probar tu API:

```
python manage.py runserver
```

Tu API estará disponible en `http://localhost:8000/`.

http://localhost:8000/tareas_app/listar_usuarios/

Paso 10: Prueba y Documentación

Utiliza herramientas como [Django REST framework's browsable API](https://www.django-rest-framework.org/topics/browsable-api/) para probar y documentar tu API.

Estos son los pasos básicos para crear una API en Django con el modelo de base de datos proporcionado. Debes personalizar y agregar lógica adicional según tus necesidades específicas. Además, asegúrate de implementar la autenticación y autorización adecuadas para proteger tu API.

NOTA: Para que no haya error en el migrado y conexiones, instalar mysqlclient:

INSTITUTO TECNOLÓGICO DEL PUTUMAYO
MATERIA: PROGRAMACIÓN AVANZADA
SEMESTRE: 9
SESIÓN: 4
DOCENTE: ING. FREDY JACANAMIJOY
11/09/2023

pip install mysqlclient