Manual Django Rest Framework
http://www.django-rest-framework.org/

Prerrequisitos

- Debe tener configurado y creado el proyecto miblog según el documento django-from-scratch. De no tenerlo, también puede descargar el proyecto finalizado del siguiente repositorio GIT:
- https://github.com/ulima-is2/blog-django
- Tener creado y activado un virtualenvironment.

Instalación

Desde un shell, instalar la librería djangorestframework

```
$ pip install djangorestframework
```

Definición de recursos

Vamos a crear una nueva app (módulo de django) donde se encontrará nuestro API REST

Para esto, desde shell ingresar lo siguiente:

```
$ python manage.py startapp api
```

Donde api es el nombre de la app.

Nuestra primera tarea es poder exponer alguna entidad del modelo como un recurso dentro de nuestro API REST. Para esto, utilizaremos la clase Post de nuestra app posts y la clase Usuario.
A continuación la clase Post:

```
...

class Post(models.Model):
    TECNOLOGIA = 'TE'
    ACTUALIDAD = 'AC'
    POLITICA = 'PO'
    ENTRETENIMIENTO = 'EN'

    CATEGORIAS_CHOICES = (
        (TECNOLOGIA, 'Tecnología'),
        (ACTUALIDAD, 'Actualidad'),
        (POLITICA, 'Política'),
```

```
        (ENTRETENIMIENTO, 'Entretenimiento'),
    )

    titulo = models.CharField(max_length=100)
    contenido = models.CharField(max_length=500)
    fecha_pub = models.DateTimeField('date published')
    usuario_pub = models.ForeignKey(Usuario, on_delete=models.CASCADE)
    categoria = models.CharField(
        max_length=2,
        choices=CATEGORIAS_CHOICES,
        default=ACTUALIDAD,
    )
```

Archivo posts/models.py

Creamos un nuevo módulo python llamado serializers.py en nuestro proyecto especial de API llamado api y codificamos lo siguiente:

```
from posts.models import Usuario, Post
from rest_framework import serializers

class UsuarioSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = Usuario
        fields = ('usuario', 'nombre')

class PostSerializer(serializers.HyperlinkedModelSerializer):
    usuario_pub = serializers.SlugRelatedField(queryset=Usuario.objects.all(), slug_field="nombre")
    class Meta:
        model = Post
        fields = ('titulo', 'contenido', 'fecha_pub', fecha_mod', 'usuario_pub', 'categoria')
```

Archivo api/serializers.py

Como se ve, crea una clase UsuarioSerializer y PostSerializer que definirán la manera como se van a presentar las *representaciones* del *recurso*. En este caso, se mostrarán los campos usuario y nombre para Usuario y titulo, contenido y categoria para Post.

Posteriormente codificamos los views. Para esto modificaremos el archivo views.py del app api creado según lo que se indica a continuación:

```
from rest_framework import viewsets
from serializers import PostSerializer, UsuarioSerializer
from posts.models import Post, Usuario

class UsuarioViewSet(viewsets.ModelViewSet):
    """
    API que permite un usuario ser visualizado y editado
    """
    queryset = Usuario.objects.all()
    serializer_class = UsuarioSerializer

class PostViewSet(viewsets.ModelViewSet):
    """
    API que permite un post ser visualizado y editado
    """
    queryset = Post.objects.all()
    serializer_class = PostSerializer
```

Archivo api/views.py

Definimos las URLs. Para esto creamos un archivo llamado urls.py dentro de nuestra app api

```
from django.conf.urls import url, include
from rest_framework import routers
from views import PostViewSet, UsuarioViewSet

router = routers.DefaultRouter()
router.register(r'posts', PostViewSet);
router.register(r'usuarios', UsuarioViewSet);

urlpatterns = [
    url(r'^', include(router.urls)),
    url('api-auth/', include('rest_framework.urls', namespace='rest_framework'))
]
```

Configuramos en miblog/urls.py la ruta por donde se accederá al servicio.

```
...
urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^posts/', include('posts.urls')),
    url(r'^api/', include('api.urls'))
]
```

Configuramos en miblog/settings.py

```
INSTALLED_APPS = [
    ...
    'rest_framework',
]
```

Para ver el nombre del usuario en vez de Usuario object, nos aseguramos de incluir el método str en la clase Usuario

```
...
    def __str__(self):
        return self.nombre
```

Para agregar seguridad configuramos en miblog/settings.py

```
...
REST_FRAMEWORK = {
   'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.DjangoModelPermissionsOrAnonReadOnly'
    ],
   'DEFAULT_AUTHENTICATION_CLASSES': (
        'rest_framework.authentication.BasicAuthentication',
    )
}
```