from django.conf.urls import include, url

from . import views

urlpatterns = [

url(r'^machines/$', views.MachineList.as\_view()),

# url(r'^characteristics/$', views.CharacteristicsList.as\_view()),

url(r'^machines/(?P<family>\w+)/$', views.MachineDetails.as\_view(), {'machineType': '','serial': '','field':'', 'value':''}),

url(r'^machines/(?P<family>\w+)/(?P<machineType>\w+)/$', views.MachineDetails.as\_view(), {'serial':'','field':'', 'value':''}),

url(r'^machines/(?P<family>\w+)/(?P<machineType>\w+)/(?P<serial>[0-9]{4}-[0-9]+)/$', views.MachineDetails.as\_view(), {'field':'', 'value':''}),

url(r'^machines/(?P<family>\w+)/(?P<machineType>\w+)/(?P<serial>[0-9]{4}-[0-9]+)/(?P<field>\w+)/$', views.MachineDetails.as\_view(), {'value':''}),

url(r'^machines/(?P<family>\w+)/(?P<machineType>\w+)/(?P<serial>[0-9]{4}-[0-9]+)/(?P<field>\w+)/(?P<value>\w+)/$', views.MachineDetails.as\_view()),

]

from django.shortcuts import render

from django.template import RequestContext, loader

from django.http import Http404

from M2M.models import machine

from M2M.models import characteristics

from M2M.serializers import MachineSerializer

from M2M.serializers import CharacteristicsSerializer

from rest\_framework.views import APIView

from rest\_framework.response import Response

from rest\_framework import status

class MachineList(APIView):

def get(self, request, format=None):

machines = machine.objects.all()

serializedMachines = MachineSerializer(machines, many = True)

return Response(serializedMachines.data)

def post(self, request, format=None):

serializedMachines=MachineSerializer(data=request.data)

if serializedMachines.is\_valid():

serializedMachines.save()

return Response(serializedMachines.data, status = status.HTTP\_201\_CREATED)

return Response(serializedMachines.errors, status = status.HTTP\_400\_BAD\_REQUEST)

def delete(self, request, format = None):

serial = request.data.get('serial')

machineToDelete = machine.objects.get(serial=serial)

machineToDelete.delete()

return Response(status=status.HTTP\_204\_NO\_CONTENT)

class MachineDetails(APIView):

def get\_object(self, family, machineType, serial):

try:

foundMachine = machine.objects.all()

if family:

foundMachine = foundMachine.filter(family\_\_iexact=family)

if machineType:

foundMachine = foundMachine.filter(machineType\_\_iexact=machineType)

if serial:

foundMachine = foundMachine.filter(serial\_\_exact=serial)

return foundMachine

except machine.DoesNotExist:

raise Http404

def get(self, request, family, machineType, serial, field, value, format = None):

machine = self.get\_object(family, machineType, serial)

if field:

machine = machine[:1].get()

serializedMachine = MachineSerializer(machine)

if not serializedMachine.data.get(field):

return CharacteristicsDetails.as\_view()(request,machine,field,value)

else:

return Response(serializedMachine.data.get(field))

else:

machines = MachineSerializer(machine, many=True)

return Response(machines.data)

def post(self, request, family, machineType, serial, field, value, format=None):

machine = self.get\_object(family, machineType, serial)

machine = machine[:1].get()

return CharacteristicsDetails.as\_view()(request,machine,field,value)

def put(self, request, family, machineType, serial, field, value, format=None):

machine = self.get\_object(family, machineType, serial)

machine = machine[:1].get()

if not field:

serializedMachine = MachineSerializer(machine, data=request.data)

if serializedMachine.is\_valid():

serializedMachine.save()

return Response(serializedMachine.data)

return Response(serializedMachine.errors, status=status.HTTP\_400\_BAD\_REQUEST)

else:

return CharacteristicsDetails.as\_view()(request,machine,field,value)

def delete(self, request, family, machineType, serial, field, value, format=None):

machine = self.get\_object(family, machineType, serial)

if not field:

machine.delete()

return Response(status=status.HTTP\_204\_NO\_CONTENT)

else:

machine = machine[:1].get()

return CharacteristicsDetails.as\_view()(request,machine,field,value)

class CharacteristicsDetails(APIView):

def get(self, request, machine, field, value, format = None):

if field.lower()=='characteristics':

characteristics = machine.characteristics.all()

serializedCharacteristics = CharacteristicsSerializer(characteristics, many=True)

return Response(serializedCharacteristics.data)

else:

characteristics = machine.characteristics.get(characteristicType\_\_iexact=field)

serializedCharacteristics = CharacteristicsSerializer(characteristics)

if value:

return Response(serializedCharacteristics.data.get(value))

else:

return Response(serializedCharacteristics.data)

def post(self, request, machine, field, value, format=None):

request.data['parent'] = machine.serial

serializedCharacteristics=CharacteristicsSerializer(data=request.data)

if serializedCharacteristics.is\_valid():

serializedCharacteristics.save()

return Response(serializedCharacteristics.data, status = status.HTTP\_201\_CREATED)

return Response(serializedCharacteristics.errors, status = status.HTTP\_400\_BAD\_REQUEST)

def put(self, request, machine, field, value, format=None):

characteristics = machine.characteristics.get(characteristicType\_\_iexact=field)

if not value:

serializedCharacteristics = CharacteristicsSerializer(characteristics, data=request.data)

if serializedCharacteristics.is\_valid():

serializedCharacteristics.save()

else:

characteristics.value = request.data.get(value)

characteristics.save()

serializedCharacteristics = CharacteristicsSerializer(characteristics)

return Response(serializedCharacteristics.data)

return Response(serializedCharacteristics.errors, status=status.HTTP\_400\_BAD\_REQUEST)

def delete(self, request, machine, field, value, format=None):

if field.lower()=='characteristics':

characteristics = machine.characteristics.all()

else:

characteristics = machine.characteristics.get(characteristicType\_\_iexact=field)

characteristics.delete()

return Response(status=status.HTTP\_204\_NO\_CONTENT)

# class CharacteristicsList(APIView):

# def get(self, request, format=None):

# characteristics = characteristics.objects.all()

# serializedCharacteristics = CharacteristicsSerializer(characteristics, many = True)

# return Response(serializedCharacteristics.data)

# def post(self, request, format=None):

# serializedCharacteristics=CharacteristicsSerializer(data=request.data)

# if serializedCharacteristics.is\_valid():

# serializedCharacteristics.save()

# return Response(serializedCharacteristics.data, status = status.HTTP\_201\_CREATED)

# return Response(serializedCharacteristics.errors, status = status.HTTP\_400\_BAD\_REQUEST)

# def delete(self, request, format = None):

# serial = request.data.get('serial')

# characteristicsToDelete = characteristics.objects.get(serial=serial)

# characteristicsToDelete.delete()

# return Response(status=status.HTTP\_204\_NO\_CONTENT)