END NODE ADD

{% extends "base\_site.html" %}  
{% block content %}  
 <div class="right\_col" role="main">  
 <div class="">  
 <div class="page-title">  
 <div class="title\_left">  
 <h3>Adding End Node</h3>  
 </div>  
 </div>  
 <div class="clearfix"></div>  
 <div class="row">  
 <div class="col-md-12 col-sm-12 col-xs-12">  
 <div class="x\_panel">  
 <div class="x\_title">  
 <h2>End Node Design</h2>  
 <ul class="nav navbar-right panel\_toolbox">  
 </ul>  
 <div class="clearfix"></div>  
 </div>  
 <div class="x\_content">  
 <br />  
 <form method="post" id="demo-form2" data-parsley-validate class="form-horizontal form-label-left">  
 {% csrf\_token %}  
 <div class="form-group">  
 <label class="control-label col-md-3 col-sm-3 col-xs-12" for="first-name">End Node Name <span class="required">\*</span>  
 </label>  
 <div class="col-md-6 col-sm-6 col-xs-12">  
 <input type="text" id="first-name" placeholder="name" name="name" value="{{ name }}" required="required" class="form-control col-md-7 col-xs-12">  
 </div>  
 </div>  
 <div class="form-group">  
 <label class="control-label col-md-3 col-sm-3 col-xs-12" for="last-name">End Node Description <span class="required">\*</span>  
 </label>  
 <div class="col-md-6 col-sm-6 col-xs-12">  
 <input type="text" id="last-name" placeholder="description" name="description" value="{{ description }}" required="required" class="form-control col-md-7 col-xs-12">  
 </div>  
 </div>  
 <div class="form-group">  
 <label class="control-label col-md-3 col-sm-3 col-xs-12">Activation Method</label>  
 <div class="col-md-6 col-sm-6 col-xs-12">  
 <div id="activationmethod" class="btn-group" data-toggle="buttons">  
 <label class="btn btn-default" data-toggle-class="btn-primary" data-toggle-passive-class="btn-default">  
 <input type="radio" name="abp" value="abp"> &nbsp; ABP &nbsp;  
 </label>  
 <label class="btn btn-default" data-toggle-class="btn-primary" data-toggle-passive-class="btn-default">  
 <input type="radio" name="otaa" value="otaa"> OTAA  
 </label>  
 </div>  
 </div>  
 </div>  
 <div class="form-group">  
 <label class="control-label col-md-3 col-sm-3 col-xs-12">Device Type</label>  
 <div class="col-md-6 col-sm-6 col-xs-12">  
 <div id="devicetype" class="btn-group" data-toggle="buttons">  
 <label class="btn btn-default" data-toggle-class="btn-primary" data-toggle-passive-class="btn-default">  
 <input type="radio" name="temphum" value="temphum"> &nbsp; Temperature-Humidity &nbsp;  
 </label>  
 <label class="btn btn-default" data-toggle-class="btn-primary" data-toggle-passive-class="btn-default">  
 <input type="radio" name="energy" value="energy"> Energy  
 </label>  
 <label class="btn btn-default" data-toggle-class="btn-primary" data-toggle-passive-class="btn-default">  
 <input type="radio" name="water" value="water"> Water  
 </label>  
 </div>  
 </div>  
 </div>  
 <div class="form-group">  
 <label class="control-label col-md-3 col-sm-3 col-xs-12" for="first-name">Device EUI <span class="required">\*</span>  
 </label>  
 <div class="col-md-6 col-sm-6 col-xs-12">  
 <input type="text" id="dev\_eui" placeholder="dev\_eui" name="dev\_eui" value="{{ dev\_eui }}" required="required" class="form-control col-md-7 col-xs-12">  
 </div>  
 </div>  
 <div class="form-group">  
 <label class="control-label col-md-3 col-sm-3 col-xs-12" for="first-name">Application EUI <span class="required">\*</span>  
 </label>  
 <div class="col-md-6 col-sm-6 col-xs-12">  
 <input type="text" id="app\_eui" placeholder="app\_eui" name="app\_eui" value="{{ app\_eui }}" required="required" class="form-control col-md-7 col-xs-12">  
 </div>  
 </div>  
 <div class="form-group">  
 <label class="control-label col-md-3 col-sm-3 col-xs-12" for="first-name">End Node Name <span class="required">\*</span>  
 </label>  
 <div class="col-md-6 col-sm-6 col-xs-12">  
 <input type="text" id="dev\_add" placeholder="dev\_add" name="dev\_add" value="{{ dev\_add }}" required="required" class="form-control col-md-7 col-xs-12">  
 </div>  
 </div>  
 <div class="form-group">  
 <label class="control-label col-md-3 col-sm-3 col-xs-12" for="first-name">End Node Name <span class="required">\*</span>  
 </label>  
 <div class="col-md-6 col-sm-6 col-xs-12">  
 <input type="text" id="net\_sess\_key" placeholder="net\_sess\_key" name="net\_sess\_key" value="{{ net\_sess\_key }}" required="required" class="form-control col-md-7 col-xs-12">  
 </div>  
 </div>  
 <div class="form-group">  
 <label class="control-label col-md-3 col-sm-3 col-xs-12" for="first-name">End Node Name <span class="required">\*</span>  
 </label>  
 <div class="col-md-6 col-sm-6 col-xs-12">  
 <input type="text" id="app\_sess\_key" placeholder="app\_sess\_key" name="app\_sess\_key" value="{{ app\_sess\_key }}" required="required" class="form-control col-md-7 col-xs-12">  
 </div>  
 </div>  
 <div class="ln\_solid"></div>  
 <div class="form-group">  
 <div style="" class="col-md-6 col-sm-6 col-xs-12 col-md-offset-3">  
 <button type="submit" class="btn btn-success">Submit</button>  
 </div>  
 </div>  
 </form>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
{% endblock content %}

Gateway vıews

from django.shortcuts import render, HttpResponse, redirect  
from .forms import GatewayForm  
from .models import Gateways, GatewayInternalInformations  
from django.contrib import messages  
from django.contrib.auth.models import User  
import requests, json, dateutil.parser  
from datetime import datetime, timedelta  
  
def ListGateways(request):  
  
 userNames = []  
  
 token = GetTokenFromLoraAPI("yagizhanyakali", "8546Yny")  
  
 gateways = GetAllGateways(token)  
  
 Gateways.objects.all().delete()  
  
 for gateway in gateways:  
 try:  
 gatewayInternal = GatewayInternalInformations.objects.get(gatewayEui = str(gateway[2]))  
 print("Try")  
 gatewayType = gatewayInternal.typeofGatewayInternal  
 addedUser = gatewayInternal.author  
 gatewayModel = Gateways(gatewayName = gateway[0], gatewayDescription = gateway[1], gatewayEui = gateway[2], gatewayCreatedDate = gateway[3], gatewayLastSeen = gateway[4], typeofGateway = gatewayType, author = addedUser)  
  
 except GatewayInternalInformations.DoesNotExist:  
 gatewayModel = Gateways(gatewayName = gateway[0], gatewayDescription = gateway[1], gatewayEui = gateway[2], gatewayCreatedDate = gateway[3], gatewayLastSeen = gateway[4])  
 print("Except")  
 gatewayModel.save()  
   
 gateways = Gateways.objects.all()  
  
 context = {  
 "gateways": gateways  
 }  
   
 return render(request, "gateway\_index.html", context=context)  
  
def GetAllGateways(token):  
  
 gatewayNames =[]  
 gatewayDescriptions = []  
 gatewayEuis = []  
 gatewayCreatedDates = []  
 gatewayLastSeens = []  
  
 url = "http://169.254.1.3:8888/gateways/all/"  
  
 headers = {  
 "Authorization": "Token {}".format(token)  
 }  
  
 response = requests.get(url, headers=headers)  
  
 #print(response.text)  
  
 for gateway in json.loads(response.text)["result"]:  
  
 gatewayNames.append(gateway["name"])  
 gatewayDescriptions.append(gateway["description"])  
 gatewayEuis.append(gateway["id"])  
  
 createdDate = dateutil.parser.parse(gateway["createdAt"])  
 createdDate = datetime.strptime(str(createdDate)[:-13], r'%Y-%m-%d %H:%M:%S')  
 createdDate = createdDate + timedelta(hours=3)  
 gatewayCreatedDates.append(str(createdDate))  
  
 gatewayLastSeens.append(gateway["lastSeenAt"])  
  
 return zip(gatewayNames, gatewayDescriptions, gatewayEuis, gatewayCreatedDates, gatewayLastSeens)  
  
def GetTokenFromLoraAPI(username, password):  
  
 url = "http://169.254.1.3:8888/api-token-auth/"  
  
 data = {  
 "username": username,  
 "password": password  
 }  
  
 response = requests.post(url, data=data)  
  
 token = json.loads(response.text)["token"]  
  
 return token  
  
def AddGateway(request):  
 form = GatewayForm(request.POST or None)  
 if form.is\_valid():  
 typeofGateway = "Indoor"  
 gateway = form.save(commit=False)  
 gateway.author = request.user  
 gatewayIndoor = request.POST.get("indoor", False)  
 gatewayOutdoor = request.POST.get("outdoor", False)  
 if (gatewayIndoor == "indoor" and gatewayOutdoor == False):  
 typeofGateway = "Indoor"  
 if (gatewayOutdoor == "outdoor" and gatewayIndoor == False):  
 typeofGateway = "Outdoor"  
 if (gatewayIndoor == False and gatewayOutdoor == False):  
 typeofGateway = "Indoor"  
 gateway.typeofGateway = typeofGateway   
 gateway.gatewayStatus = "Connected" #will be change with real data...  
 #gateway.save()  
 createGatewayResponse = CreateGateway(gateway.gatewayName, gateway.gatewayDescription, gateway.gatewayEui)  
  
 internalDb = GatewayInternalInformations(gatewayEui = gateway.gatewayEui, typeofGatewayInternal = typeofGateway, author = request.user)  
 internalDb.save()  
  
 if (createGatewayResponse == "OK"):  
 messages.add\_message(request, messages.SUCCESS, "Gateway Added!")  
 else:  
 messages.add\_message(request, messages.ERROR, "Progress Error!")  
 return render(request, "add\_gateway.html")  
 return render(request, "add\_gateway.html")  
  
def CreateGateway(gatewayName, gatewayDescription, gatewayEui):  
  
 url = "http://169.254.1.3:8888/gateways/add/"  
  
 token = GetTokenFromLoraAPI("yagizhanyakali", "8546Yny")  
  
 headers = {  
 "Authorization": "Token {}".format(token),  
 "Content-Type": "application/json"  
 }  
  
 data = {  
 "description": str(gatewayDescription),   
 "gatewayProfileID": "null",   
 "id": str(gatewayEui),   
 "location": {  
 "accuracy": 0,   
 "altitude": 0,   
 "latitude": 0,   
 "longitude": 0,   
 "source": "UNKNOWN"  
 },   
 "name": str(gatewayName).replace(" ", "\_"),   
 "networkServerID": "1",   
 "organizationID": "1"  
 }  
  
 data = json.dumps(data)  
  
 response = requests.post(url, headers=headers, data=data)  
  
 if (response.status\_code == 200):  
 return "OK"  
 else:  
 return "Error"  
  
def UpdateGateway(gatewayName, gatewayDescription, gatewayEui, currentEui):  
  
 url = "http://169.254.1.3:8888/gateways/update/{}/".format(currentEui)  
  
 token = GetTokenFromLoraAPI("yagizhanyakali", "8546Yny")  
  
 headers = {  
 "Authorization": "Token {}".format(token),  
 "Content-Type": "application/json"  
 }  
  
 data = {  
 "description": str(gatewayDescription),  
 "location": {  
 "accuracy": 0,  
 "altitude": 0,  
 "latitude": 0,  
 "longitude": 0,  
 "source": "UNKNOWN"  
 },  
 "name": str(gatewayName)  
 }  
  
 data = json.dumps(data)  
  
 print(data)  
  
 response = requests.post(url, headers=headers, data=data)  
  
 if (response.status\_code == 200):  
 return "OK"  
 else:  
 return "Error"  
  
def EditGateway(request, id):  
  
 gateway = Gateways.objects.get(id = id)  
 gatewayName = gateway.gatewayName  
 gatewayEui = gateway.gatewayEui  
 gatewayDescription = gateway.gatewayDescription  
 gatewayType = gateway.typeofGateway  
  
 currentEui = gatewayEui  
  
 context = {  
 "gatewayName": gatewayName,  
 "gatewayEui": gatewayEui,  
 "gatewayDescription": gatewayDescription,  
 "gatewayType": gatewayType  
 }  
  
 if (request.method == "POST"):  
 typeofGateway = ""  
 gateway = Gateways.objects.get(id = id)  
 gateway.gatewayName = request.POST.get("gatewayName", False)  
 gateway.gatewayEui = request.POST.get("gatewayEui", False)  
 gateway.gatewayDescription = request.POST.get("gatewayDescription", False)  
 gatewayTypeIndoor = request.POST.get("indoor", False)  
 gatewayTypeOutdoor = request.POST.get("outdoor", False)  
 if (gatewayTypeIndoor == "indoor" and gatewayTypeOutdoor == False):  
 typeofGateway = "Indoor"  
 if (gatewayTypeOutdoor == "outdoor" and gatewayTypeIndoor == False):  
 typeofGateway = "Outdoor"  
 if (gatewayTypeIndoor == False and gatewayTypeOutdoor == False):  
 typeofGateway = "Indoor"  
 gateway.typeofGateway = typeofGateway  
  
 print(gateway.gatewayEui)  
  
 try:  
 internalGateway = GatewayInternalInformations.objects.get(gatewayEui = str(gateway.gatewayEui))  
 internalGateway.typeofGatewayInternal = typeofGateway  
 internalGateway.save()  
 except GatewayInternalInformations.DoesNotExist:  
 internalGateway = GatewayInternalInformations(gatewayEui = gateway.gatewayEui, typeofGatewayInternal = typeofGateway)  
 internalGateway.save()  
   
 updateGatewayResponse = UpdateGateway(str(gateway.gatewayName), str(gateway.gatewayDescription), str(gateway.gatewayEui), str(currentEui))  
  
 #gateway.save()  
 return redirect("/gateways/")  
 return render(request, "edit\_gateway.html", context=context)  
  
def DeleteGateway(request, id):  
  
 gateway = Gateways.objects.get(id = id)  
 gatewayName = gateway.gatewayName  
 gatewayEui = gateway.gatewayEui  
 gatewayDescription = gateway.gatewayDescription  
 gatewayId = gateway.id  
  
 context = {  
 "gatewayName": gatewayName,  
 "gatewayEui": gatewayEui,  
 "gatewayDescription": gatewayDescription  
 }  
  
 if(request.method == "POST"):  
  
 print("Delete Progress Started!")  
  
 gateway = Gateways.objects.get(id = gatewayId)  
 gatewayEui = gateway.gatewayEui  
  
 url = "http://169.254.1.3:8888/gateways/delete/{}/".format(gatewayEui)  
  
 token = GetTokenFromLoraAPI("yagizhanyakali", "8546Yny")  
  
 headers = {  
 "Authorization": "Token {}".format(token)  
 }  
  
 response = requests.post(url, headers=headers)  
  
 internalGateway = GatewayInternalInformations.objects.get(gatewayEui = gatewayEui).delete()  
  
 return redirect("http://127.0.0.1:8000/gateways/")  
   
 return render(request, "delete\_gateway.html", context=context)