ПРИЛОЖЕНИЕ Г. ТЕКСТ ПРОГРАММЫ

АННОТАЦИЯ

1. Views.py

Скрипт взятия даннных

2. Models.py

Скрип создания таблиц

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В данном разделе документации описан скрипт базы данных программы l_admin.

Поскольку postgresql имеет нативную интеграцию с python в данном приложении будут представлен код отвечающий за работу с бд.

1.Views.py

l_admin/gui/views.py

```
from django.shortcuts import render,redirect
from django.contrib.auth import authenticate, login, logout
from django.http import HttpResponseRedirect, Http404, HttpResponse
from API import models
import paramiko
import time
from datetime import datetime
# Create your views here.
def SingIn(request):
    if request.method == 'GET':
        context = ''
        return render(request, 'Page/SingIn.html', {'context': context})

    elif request.method == 'POST':
        username = request.POST.get('username', '')
        password = request.POST.get('password', '')

        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            return HttpResponseRedirect('/list')
        else:
            context = {'error': 'Wrong credintials'}  # to display error?
            return render(request, 'Page/SingIn.html', {'context': context})
```

```
def List(request,id=None):

machins=models.Machin.objects.filter(user=request.user,user_monitor_permision=True)|
models.Machin.objects.filter(other_monitor_permision=True)|
models.Machin.objects.filter(group__in=
models.Group.objects.filter(users=request.user),group_monitor_permision=True)
    machins=machins.order_by('id')
    if not request.user.is_authenticated: raise Http404
    if request.method == 'GET':
        return List_render(request,machins)
    elif request.method == 'POST':
        if 'id' in request.POST and id==0:
models.Machin.objects.get(id=request.POST['id']).delete()
        elif 'group_filter' in request.POST:

machins=models.Machin.objects.all().filter(machin_group__id=request.POST['group_filte
r'])
            return List_render(request,machins)
        elif id!=None and id!=0:
            machin=models.Machin.objects.get(id=request.POST['id'])
            machin.name=request.POST['name']
            machin.group=models.Group.objects.get(id=request.POST['group'])
            machin.ip=request.POST['ip']
            machin.port=request.POST['port']
            machin.username=request.POST['username']
            if request.POST['password'] != "": machin.password=request.POST['password']
            machin.save()
        elif id==0:
            machin=models.Machin()
            machin.name=request.POST['name']
            machin.group=models.Group.objects.get(id=request.POST['group'])
            machin.ip=request.POST['ip']
            machin.port=request.POST['port']
            machin.username=request.POST['username']
            machin.password=request.POST['password']
            machin.user=request.user
            machin.history_save=bool(request.POST['history_save'])
            machin.history_save=bool(request.POST['history_save'])
            machin.save()
        return List_render(request,machins)
def List_render(request,machins):
    return render(request, 'Page/MachinList.html', {
        "machins": machins,
        "form":models.MachinForm(),
        "machin_groups":models.Machin_Group.objects.all()
        })
def connect_ssh(request,id):
    if not request.user.is_authenticated: raise Http404
    cmdout=""
    usr=request.user
    if request.method == 'POST':
        try:
            cmdout=ssh(id,usr,request.POST['cmdin'])
            print(cmdout)
        except:
            cmdout="ERROR connect"
    elif request.method == 'GET':
        try:
            cmdout=ssh(id,usr,"echo \"l_admin connect\"")
        except:
            cmdout="ERROR connect"
    return render(request, 'Page/Connect.html',{'cmdout':cmdout})
#no render
def ssh(id,usr,cmd):
    machin=models.Machin.objects.get(id=id)
    ssh_ = paramiko.SSHClient()
    ssh_.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    ssh_.connect(str(machin.ip.ip), port=machin.port,
username=machin.username,password=machin.password, timeout=3)
    (stdin, stdout, stderr) = ssh_.exec_command(cmd)
    cmdout=stdout.read().decode("utf-8")
    ssh_.close()
    #log
    fph=f"media/machin/{machin.id}_{machin.name}.txt"
    fpl=f"media/log/{machin.id}_{machin.name}.txt"
    try:
        log=models.Log.objects.get(machin=machin)
    except:
        log=models.Log()
        log.machin=machin
        log.history.name=fph
        log.log.name=fpl
        log.save()
    if machin.history_save:
        f = open(fph, "a")
        f.write("["+datetime.now().strftime("%a, %d %b %Y %H:%M:%S +0000")+"]
"+cmd+"\n")
        f.close()
    if machin.log_save:
        f = open(fpl, "a")
        f.write(f"[{datetime.now().strftime('%a, %d %b %Y %H:%M:%S +0000')}]connect
{usr} to {str(machin.ip.ip)}:{machin.port}\n")
        f.close()
    print(log)
    return cmdout
def Logout(request):
    logout(request)
    return redirect('url-singin')
#StatusCustom
def handler400(request,exception):
    response = render(request, "Page/Status.html", {"status": 400})
    response.status_code = 400
    return response
def handler403(request,exception):
    response = render(request, "Page/Status.html", {"status": 403})
    response.status_code = 403
    return response
def handler404(request,exception):
    response = render(request, "Page/Status.html", {"status": 404})
    response.status_code = 404
    return response
def handler500(request):
    response = render(request, "Page/Status.html", {"status": 500})
    response.status_code = 500
    return response
```

2.Models.py

l_admin/API/models.py — таблицы в бд

```
from django.db import models
from django.contrib.auth.models import AbstractBaseUser, BaseUserManager,
PermissionsMixin
from django.db import models
from django.core.validators import MaxValueValidator, MinValueValidator
from django.utils import timezone
from django.utils.translation import gettext_lazy as _
from colorfield.fields import ColorField
from datetime import timedelta, datetime
from netfields import InetAddressField, NetManager
from django import forms
class UserManager(BaseUserManager):
    def create_user(self, login, history_save=True, password=None,commit=True):
        if not login: raise ValueError(_('Users must have a login'))
        user = self.model(login=login,history_save=history_save)
        user.set_password(password)
        if commit: user.save(using=self._db)
        return user
    def create_superuser(self, login, password):
        user = self.create_user(password=password,login=login,commit=False)
        user.is_staff = True
        user.is_superuser = True
        user.save(using=self._db)
        return user
class User(AbstractBaseUser, PermissionsMixin):
    login = models.CharField(_('login'), max_length=150, blank=True,unique=True)
    is_active = models.BooleanField(_('active'),default=True,help_text=_('Designates whether this user should be treated as active. Unselect this instead of deleting accounts.'),)
    is_staff = models.BooleanField(_('staff status'),default=False,help_text=_('Designates whether the user can log into this admin site.'),)
    is_observer = models.BooleanField(default=False)
    history_save = models.BooleanField(default=True)
    history =models.FileField()
    date_joined = models.DateTimeField(_('date joined'), default=timezone.now)
    objects = UserManager()
    USERNAME_FIELD = 'login'
    def get_full_name(self):
        return self.login
    def __str__(self):
        return self.login
class Group(models.Model):
    users = models.ManyToManyField(User)
    name = models.CharField(max_length=64, unique=True)
    description = models.TextField(max_length=512)
class Machin(models.Model):
    name = models.CharField(max_length=64)
    username = models.CharField(max_length=64)
    password = models.CharField(max_length=64)
    user=models.ForeignKey(User, on_delete=models.PROTECT)
    group=models.ForeignKey(Group, on_delete=models.PROTECT)
    history_save = models.BooleanField(default=True)
    log_save = models.BooleanField(default=True)
    ip = InetAddressField()

port=models.IntegerField(validators=[MaxValueValidator(65535),MinValueValidator(1)])
    objects = NetManager()
    user_monitor_permision=models.BooleanField(default=True)
    user_user_permision=models.BooleanField(default=True)
    group_monitor_permision=models.BooleanField(default=True)
    group_user_permision=models.BooleanField(default=True)
    other_monitor_permision=models.BooleanField(default=True)
    other_user_permision=models.BooleanField(default=True)
    def form(self):
        return MachinForm(instance=self)
    def getip(self):
        ip=str(self.ip.ip)
        return ip
class MachinForm(forms.ModelForm):
    password = forms.CharField(widget=forms.PasswordInput(), required = False)
    group=None
    try:
        group = forms.Select(choices=Group.objects.all().values_list('id', 'name'))
    except:
        group=None
    class Meta:
        model = Machin
        fields = ('__all__')
class Log(models.Model):
    machin =models.ForeignKey(Machin, on_delete=models.CASCADE)
    time_save=models.DateTimeField(auto_now_add=True)
    log =models.FileField(editable=False,auto_created = True)
    history =models.FileField(editable=False,auto_created = True)
class Machin_Group(models.Model):
    machin = models.ManyToManyField(Machin,related_name='machin_group')
    name = models.CharField(max_length=64, unique=True)
class Machine_request_one_comand(models.Model):
    machin_group=models.ForeignKey(Machin_Group, on_delete=models.CASCADE)
    machin=models.ForeignKey(Machin, on_delete=models.CASCADE)
    user=models.ForeignKey(User, on_delete=models.SET_NULL, null=True)
    color = ColorField()
    request_file_path=models.FileField(editable=False)
```