Backend.APIProject

api

APIproject

Backend.Backend

0001\_initial.py

 Generated by Django 3.2.6 on 2021-08-05 08:28

from django.db import migrations, models

class Migration(migrations.Migration):

    initial = True

    dependencies = [

    ]

    operations = [

        migrations.CreateModel(

            name='Reporte',

            fields=[

                ('id', models.BigAutoField(auto\_created=True, primary\_key=True, serialize=False, verbose\_name='ID')),

                ('id\_contrato', models.CharField(max\_length=50)),

                ('title', models.CharField(max\_length=50)),

                ('fecha', models.DateField()),

                ('lote', models.CharField(max\_length=20)),

                ('parte', models.CharField(max\_length=20)),

                ('cantIns', models.PositiveIntegerField()),

                ('cantRech', models.PositiveIntegerField()),

                ('defecto', models.CharField(max\_length=20)),

                ('cantRet', models.PositiveIntegerField()),

                ('cantAcep', models.PositiveIntegerField()),

            ],

        ),

    ]

settings.py

"""

Django settings for APIproject project.

Generated by 'django-admin startproject' using Django 3.2.6.

For more information on this file, see

https://docs.djangoproject.com/en/3.2/topics/settings/

For the full list of settings and their values, see

https://docs.djangoproject.com/en/3.2/ref/settings/

"""

from pathlib import Path

# Build paths inside the project like this: BASE\_DIR / 'subdir'.

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!

SECRET\_KEY = 'django-insecure-qx#^#7fg#4#@www&b0xk24l^p(l4vg2@-01nyn7kd!a&\*jqo@u'

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True

ALLOWED\_HOSTS = ['192.168.1.143', 'localhost']

# Application definition

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'rest\_framework',

    'api',

]

MIDDLEWARE = [

    'django.middleware.security.SecurityMiddleware',

    'django.contrib.sessions.middleware.SessionMiddleware',

    'django.middleware.common.CommonMiddleware',

    'django.middleware.csrf.CsrfViewMiddleware',

    'django.contrib.auth.middleware.AuthenticationMiddleware',

    'django.contrib.messages.middleware.MessageMiddleware',

    'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

ROOT\_URLCONF = 'APIproject.urls'

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [],

        'APP\_DIRS': True,

        'OPTIONS': {

            'context\_processors': [

                'django.template.context\_processors.debug',

                'django.template.context\_processors.request',

                'django.contrib.auth.context\_processors.auth',

                'django.contrib.messages.context\_processors.messages',

            ],

        },

    },

]

WSGI\_APPLICATION = 'APIproject.wsgi.application'

# Database

# https://docs.djangoproject.com/en/3.2/ref/settings/#databases

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.sqlite3',

        'NAME': BASE\_DIR / 'db.sqlite3',

    }

}

# Password validation

# https://docs.djangoproject.com/en/3.2/ref/settings/#auth-password-validators

AUTH\_PASSWORD\_VALIDATORS = [

    {

        'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

    },

]

# Internationalization

# https://docs.djangoproject.com/en/3.2/topics/i18n/

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_L10N = True

USE\_TZ = True

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/3.2/howto/static-files/

STATIC\_URL = '/static/'

# Default primary key field type

# https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field

DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

Views.py

from django.shortcuts import render, HttpResponse

from .models import Reporte

from .serializers import ReporteSerializer

from django.http import JsonResponse

from rest\_framework.parsers import JSONParser

from django.views.decorators.csrf import csrf\_exempt

import xmlrpc.client

from django import forms

from .forms import LoginForm

url = "https://axhs.odoo.com" #main

db = "axonebackoffice-aigx-main-1924299" #main

models = xmlrpc.client.ServerProxy('{}/xmlrpc/2/object'.format(url))

common = xmlrpc.client.ServerProxy('{}/xmlrpc/2/common'.format(url))

# Create your views here.

@csrf\_exempt

def reporte\_list(request):

    if request.method == 'GET':

        reportes = Reporte.objects.all()

        serializer = ReporteSerializer(reportes, many=True)

        return JsonResponse(serializer.data, safe=False)

    elif request.method == 'POST':

        data = JSONParser().parse(request)

        serializer = ReporteSerializer(data=data)

        if serializer.is\_valid():

            serializer.save()

            return JsonResponse(serializer.data, status=201)

        return JsonResponse(serializer.errors, status=400)

@csrf\_exempt

def reporte\_details(request, pk):

    try:

        reporte = Reporte.objects.get(pk=pk)

    except Reporte.DoesNotExist:

        return HttpResponse(status=404)

    if request.method == 'GET':

        serializer = ReporteSerializer(reporte)

        return JsonResponse(serializer.data)

    elif request.method == 'PUT':

        data = JSONParser().parse(request)

        serializer = ReporteSerializer(reporte, data=data)

        if serializer.is\_valid():

            serializer.save()

            return JsonResponse(serializer.data)

        return JsonResponse(serializer.errors, status=400)

    elif request.method == 'DELETE':

        reporte.delete()

        return HttpResponse(status=204)

@csrf\_exempt

def login(request):

    # username = "jcabrales@aigx.mx"

    # url = "https://axhs.odoo.com" #main

    # db = "axonebackoffice-aigx-main-1924299" #main

    # password = "cf6550d7f13beca8556c3ed77f91ffbe53a0005" #main

    # common = xmlrpc.client.ServerProxy('{}/xmlrpc/2/common'.format(url))

    if request.method == 'POST':

        request.session.flush()

        form = LoginForm(request.POST)

        if form.is\_valid():

            username = form.cleaned\_data['username']

            password = form.cleaned\_data['password']

    elif request.method == 'GET':

        request.session.flush()

        form = LoginForm(request.GET)

        if form.is\_valid():

            username = form.cleaned\_data['username']

            password = form.cleaned\_data['password']

    else:

        return HttpResponse("Wrong Method") #redirect somewhere else

    uid = common.authenticate(db, username, password, {})

    if (uid is not False):

        request.session['uid'] = uid

        request.session['secret'] = password

        request.session['empresa'] = getEmpresa(uid, password)

        print(uid)

        return HttpResponse(uid, status=200)

    return HttpResponse("Wrong Username or Password", status=200)

@csrf\_exempt

def readUser(request):

    uid = request.session.get('uid', False)

    if(uid is not False):

        return HttpResponse(uid)

    return HttpResponse("Not logged In")

def readReporteActividad(request):

    uid = request.session.get('uid', False)

    if(uid is False):

        return HttpResponse("Not logged In")

    else:

        password = request.session.get('secret')

        empresa = "Aceros Trefilados de Precisión, S. de R.L. de C.V. (NUCOR), Christian Merino"

        hola=models.execute\_kw(db, uid, password,

        'x\_reporte\_de\_actividad', 'search\_read',

        [[['x\_studio\_cliente', 'like', empresa]]],

        {'fields': ['x\_studio\_cliente', 'x\_studio\_fecha', 'x\_studio\_\_parte', 'x\_studio\_cantidad\_inspeccionada', 'x\_studio\_cantidad\_aceptada']})

        return HttpResponse(hola)

def getEmpresa(uid, password):

    hola=models.execute\_kw(db, uid, password,

        'x\_reporte\_de\_actividad', 'search\_read',

        [[['x\_studio\_cliente', 'like', empresa]]],

        {'fields': ['x\_studio\_cliente', 'x\_studio\_fecha', 'x\_studio\_\_parte', 'x\_studio\_cantidad\_inspeccionada', 'x\_studio\_cantidad\_aceptada']})

    return hola

Api.Urls.py

from django.urls import path

from .views import reporte\_list, reporte\_details, login, readUser, readReporteActividad

urlpatterns = [

    path('api/reportes/', reporte\_list),

    path('api/reportes/<int:pk>', reporte\_details),

    path('api/login', login),

    path('api/whoami', readUser),

    path('api/reportesActividad', readReporteActividad)

]

Forms.py

from django import forms

class LoginForm(forms.Form):

    username = forms.CharField(label='username', max\_length=100)

    password = forms.CharField(label='password', max\_length=100)

api.models.py

from django.db import models

# Create your models here.

class Reporte(models.Model):

    title = models.CharField(max\_length=50)

    fecha = models.DateField()

    lote = models.CharField(max\_length=20)

    parte = models.CharField(max\_length=20)

    cantIns = models.PositiveIntegerField()

    cantRech = models.PositiveIntegerField()

    defecto = models.CharField(max\_length=20)

    cantRet = models.PositiveIntegerField()

    cantAcep = models.PositiveIntegerField()

    def \_\_str\_\_(self):

        return self.title

APIProject.urls.py

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

    path('admin/', admin.site.urls),

    path('', include('api.urls'))

]

serializers

from rest\_framework import serializers

from .models import Reporte

class ReporteSerializer(serializers.ModelSerializer):

    class Meta:

        model = Reporte

        fields = ['id', 'title', 'fecha', 'lote', 'parte', 'cantIns', 'cantRech', 'defecto', 'cantRet', 'cantAcep']

apps.py

from django.apps import AppConfig

class ApiConfig(AppConfig):

    default\_auto\_field = 'django.db.models.BigAutoField'

    name = 'api'

admin.py

from django.apps import AppConfig

class ApiConfig(AppConfig):

    default\_auto\_field = 'django.db.models.BigAutoField'

    name = 'api'