app 端分享样板间到小程序，小程序打开，app端收到 已查看的通知

caseview=CaseViewRecord.objects.filter(case\_no=no).first()

if caseview:

sales\_user\_id=caseview.sales\_user\_id

PushService.push\_to\_user(sales\_user\_id, CaseVisitMessage(user\_id, case)) #CaseVisitMessage 用户id，和方案详情

在<https://yun.jiandanhome.com/da/wx/case/get_editor_url/> 接口

通过传入的新方案编号判断浏览记录表数据是否存在

如果存在：

取出小程序端用户id，和app端导购id

推送消息，与用户id 相对应的导购id设备

推送消息方法逻辑：

push\_to\_user(导购id，message)

通过导购id 到消息推送设备登记表看有没有记录

如果存在：

取出设备系统版本 和 user\_id

判断设备系统是ios或者android

push\_to\_ios\_user(alias\_id, message)

push\_to\_android\_user(alias\_id, message)

D:\work\yun\apps\push

小米推送：

APP端 将设备信息给服务端，我存下来

APP端 将设备信息给小米服务器

我将要推送的信息给小米服务器

小米服务器将信息给app端

推消息通过别名，或者注册id

相关接口：

存储设备信息

推送app信息给导购sales\_user\_id

def sales\_visit\_user(sales\_user\_id,):

原接口：D:\work\yun\apps\card\views\push\device.py

创建应用成功后，会获得该应用对应的AppId, AppKey, AppSecret。

其中 AppId 和 AppKey 是客户端的身份标识，在客户端SDK初始化时使用； AppSecret是服务器端的身份标识，在使用服务端SDK向客户端发送消息时使用。

settings:

# ============简单搭APP START===========

JDD\_APP\_ANDROID\_PACKAGE\_NAME= 'cy.eju.com.zxzsg'

JDD\_APP\_IOS\_BUNDLE\_ID = 'com.jdj.jdd'

# ============简单搭APP END===========

# ============小米推送 START===========

PUSH\_MODE = 'sandbox'

XMPUSH\_APPID = '2882303761517943463'

XMPUSH\_ANDROID\_KEY = '5291794339463'

XMPUSH\_ANDROID\_SECRET = 'c0cMuZXcYC2T5CryKau7Yg=='

# ============小米推送 END===========

# android key and secret

XMPUSH\_ANDROID\_APPSECRET='c0cMuZXcYC2T5CryKau7Yg=='

# ios key and secret

XMPUSH\_IOS\_APPSECRET='Zb7AmqWKH3FfBxN8MtxEhg=='

在设备登记表新增app\_name 字段：

app\_name=model.IntegerField(default=settings.AppNameEnum.JDM,choices=settings.Appname,verbose\_name=u'app应用名')

settings配置：（默认简单美房，后期有新应用在此扩展）

class AppNameEnum():

JDM=1

JDD=2

Appname=(

(AppNameEnum.JDM,u"简单美房")，

(AppNameEnum.JDD,u"简单搭")

)

在查设备登记表时以user\_id和app\_name 做作为过滤条件，判断是否存在以及区分不同的app应用

class DeviceOSEnum(object):

iOS = 1

Android = 2

# 透传

class PassThroughEnum(object):

ON = 1

OFF = 0

from account.decorators import login\_required

from mconsole.decorators import muser\_required

from push import parameters

@login\_required

@muser\_required

require\_param(parameters.device\_register)

保存接口：

# from apps.push.models import \* #D:\work\yun\apps\push\models\push\_device.py

# @wx\_mini\_required

# @require\_param(parameters.device\_register)

# def device\_save(request):

# user\_id = request.user\_id

# reg\_id = request.params['reg\_id']

# os = request.params['os']

# os\_type = request.params['os\_type']

# os\_version = request.params['os\_version']

# app\_version = request.params['app\_version']

# model = request.params['model']

# app\_name = request.params.get("app\_name", 1)

#

# upfilter={

# "user\_id":user\_id,

# "reg\_id":reg\_id,

# "os":os,

# "os\_type":os\_type,

# "os\_version":os\_version,

# "model":model,

# "app\_version":app\_version

# "app\_name":app\_name

# }

# PushDevice.objects.create(\*\*upfilter)

#

# return JSONResponse()

# 小米推送相关

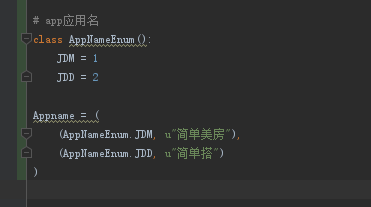
# urlpatterns += [

# url(r'^push/device\_save/$', views.wxmini.device\_save, name='push\_device\_save'),

# ]

建表：





套餐浏览记录表： D:\work\yun\apps\da\models\view\_record.py



error = copy.deepcopy(da\_settings.ERROR["NOT\_EXIST\_ERR"])

error["msg"] = u"自定义分类不存在！"

*return* JSONResponse(error=error)

E:\yun\apps\mconsole\settings.py

*from* mconsole *import* settings