day06 支付相关

今日概要

- 业务支付相关业务
- 项目开发

内容回顾

- 常鑫，上周内容回顾
  - 优惠券
  - 定时任务
  - 优惠券状态
  - 后台管理
- 运营中心（后台管理） 开发了专场、拍品、图片、规格等管理。
  - ModelForm、COS图片上传、定制钩子实现图片上传。
  - 处理上传和修改的操作（类型判断）。
  - datetimepicker时间组件的使用。
  - celery + redis 实现定时任务。
  - 通过js做图片预览 / js FormData 进行文件上传。
- 优惠券
  - 后台
    - 对于优惠券管理。
  - 小程序&api
    - 表结构的设计
    - pass

今日详细

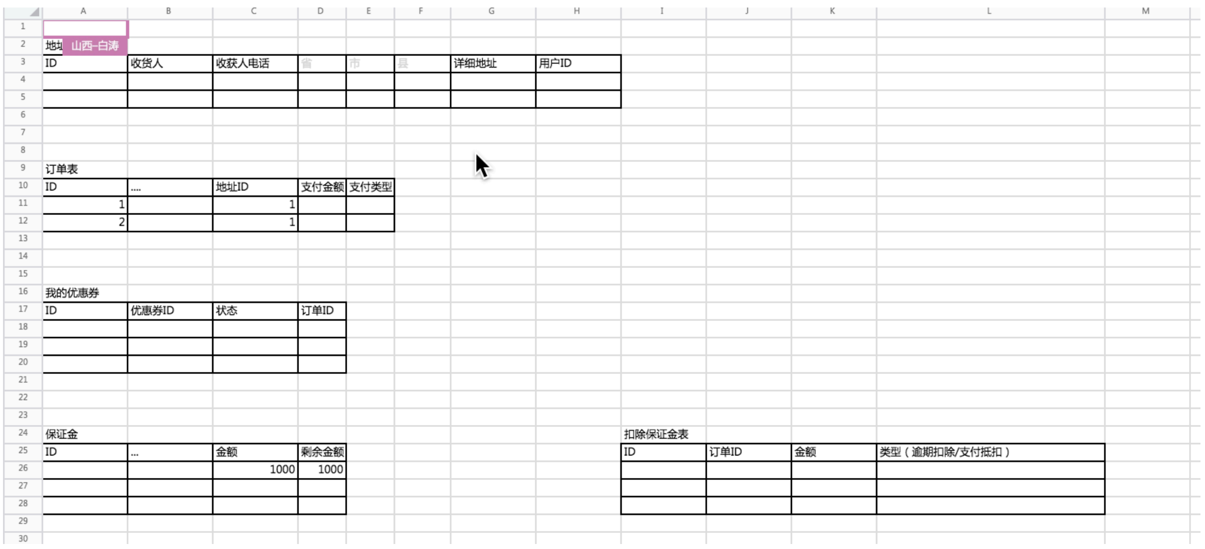
1.支付业务逻辑

- 张达

```
参考淘宝 & 订单去绑定收获地址

- 收获地址和订单关联
- 优惠券
- 价格处理：优惠券、保证金、余额/微信支付
```

- 表结构的设计



地址

ID	收货人	收获人电话	省	市	县	详细地址	用户ID

订单表

ID	....	地址ID	支付金额	支付类型
1		1		
2		1		

我的优惠券

ID	优惠券ID	状态	订单ID

保证金

ID	...	金额	剩余金额
		1000	1000

扣除保证金表

ID	订单ID	金额	类型（逾期扣除/支付抵扣）

任务：根据表同步ORM类（截图发群里）10:25

```python
from django.db import models


# ############################## 动态 ##############################

class UserInfo(models.Model):
    telephone = models.CharField(verbose_name='手机号', max_length=11)
    nickname = models.CharField(verbose_name='昵称', max_length=64)
    avatar = models.CharField(verbose_name='头像', max_length=64, null=True)
    token = models.CharField(verbose_name='用户Token', max_length=64)

    fans_count = models.PositiveIntegerField(verbose_name='粉丝个数',
default=0)
    follow = models.ManyToManyField(verbose_name='关注', to='self', blank=True)

    balance = models.PositiveIntegerField(verbose_name='账户余额',
default=1000)
    session_key = models.CharField(verbose_name='微信会话秘钥', max_length=32)
    openid = models.CharField(verbose_name='微信用户唯一标识', max_length=32)

    def __str__(self):
        return self.nickname


class Topic(models.Model):
    """
    话题
    """
    title = models.CharField(verbose_name='话题', max_length=32)
    count = models.PositiveIntegerField(verbose_name='关注度', default=0)
```

```
class News(models.Model):
    """
    动态
    """
    cover = models.CharField(verbose_name='封面', max_length=128)
    content = models.CharField(verbose_name='内容', max_length=255)
    topic = models.ForeignKey(verbose_name='话题', to='Topic', null=True,
blank=True)
    address = models.CharField(verbose_name='位置', max_length=128, null=True,
blank=True)

    user = models.ForeignKey(verbose_name='发布者', to='UserInfo',
related_name='news')

    favor_count = models.PositiveIntegerField(verbose_name='赞数', default=0)
    # favor = models.ManyToManyField(verbose_name='点赞记录', to='UserInfo',
related_name="news_favor")

    viewer_count = models.PositiveIntegerField(verbose_name='浏览数',
default=0)
    # viewer = models.ManyToManyField(verbose_name='浏览器记录', to='UserInfo',
related_name='news_viewer')

    comment_count = models.PositiveIntegerField(verbose_name='评论数',
default=0)

    create_date = models.DateTimeField(verbose_name='创建时间',
auto_now_add=True)


class ViewerRecord(models.Model):
    """
    浏览器记录
    """
    news = models.ForeignKey(verbose_name='动态', to='News')
    user = models.ForeignKey(verbose_name='用户', to='UserInfo')


class NewsFavorRecord(models.Model):
    """
    动态赞记录表
    """
    news = models.ForeignKey(verbose_name='动态', to='News')
    user = models.ForeignKey(verbose_name='点赞用户', to='UserInfo')


class CommentRecord(models.Model):
    """
    评论记录表
    """
    news = models.ForeignKey(verbose_name='动态', to='News')
    content = models.CharField(verbose_name='评论内容', max_length=255)
    user = models.ForeignKey(verbose_name='评论者', to='UserInfo')
    create_date = models.DateTimeField(verbose_name='评论时间',
auto_now_add=True)

    reply = models.ForeignKey(verbose_name='回复', to='self', null=True,
blank=True)
    depth = models.PositiveIntegerField(verbose_name='评论层级', default=1)

    favor_count = models.PositiveIntegerField(verbose_name='赞数', default=0)

    # 以后方便通过跟评论找到其所有的子孙评论
    root = models.ForeignKey(verbose_name='根评论', to='self', null=True,
blank=True, related_name='descendant')


class CommentFavorRecord(models.Model):
    """
    评论赞记录
    """
    comment = models.ForeignKey(verbose_name='动态', to='CommentRecord')
    user = models.ForeignKey(verbose_name='点赞用户', to='UserInfo')


class NewsDetail(models.Model):
    """
    动态详细
    """
    key = models.CharField(verbose_name='腾讯对象存储中的文件名', max_length=128,
help_text="用于以后在腾讯对象存储中删除")
    cos_path = models.CharField(verbose_name='腾讯对象存储中图片路径',
max_length=128)
    news = models.ForeignKey(verbose_name='动态', to='News')


# ############################### 拍卖 ###############################

class Auction(models.Model):
    """
    拍卖系列
    """
    title = models.CharField(verbose_name='标题', max_length=32)
    status_choices = (
        (1, '未开拍'),
        (2, '预展中'),
        (3, '拍卖中'),
        (4, '已结束')
    )
    status = models.PositiveSmallIntegerField(verbose_name='状态',
choices=status_choices, default=1)

    # cover = models.CharField(verbose_name='封面',max_length=128)
    cover = models.FileField(verbose_name='封面', max_length=128)

    video = models.CharField(verbose_name='预览视频', max_length=128,
null=True, blank=True)

    preview_start_time = models.DateTimeField(verbose_name='预展开始时间')
    preview_end_time = models.DateTimeField(verbose_name='预展结束时间')

    auction_start_time = models.DateTimeField(verbose_name='拍卖开始时间')
    auction_end_time = models.DateTimeField(verbose_name='拍卖结束时间')

    deposit = models.PositiveIntegerField(verbose_name='全场保证金',
default=1000)

    total_price = models.PositiveIntegerField(verbose_name='成交额', null=True,
blank=True)
    goods_count = models.PositiveIntegerField(verbose_name='拍品数量',
default=0)
    bid_count = models.PositiveIntegerField(verbose_name='出价次数', default=0)
    look_count = models.PositiveIntegerField(verbose_name='围观次数',
default=0)
    create_time = models.DateTimeField(verbose_name='创建时间',
auto_now_add=True)

    class Meta:
        verbose_name_plural = '拍卖系列'

    def __str__(self):
        return self.title


class AuctionTask(models.Model):
    """ 定时任务 """
    auction = models.OneToOneField(verbose_name='专场', to='Auction')

    preview_task = models.CharField(verbose_name='Celery预展任务ID',
max_length=64)

    auction_task = models.CharField(verbose_name='Celery拍卖任务ID',
max_length=64)

    auction_end_task = models.CharField(verbose_name='Celery拍卖结束任务ID',
max_length=64)
```

```python
class AuctionItem(models.Model):
    """
    拍卖商品
    """
    auction = models.ForeignKey(verbose_name='拍卖', to='Auction')
    uid = models.CharField(verbose_name='图录号', max_length=12)
    title = models.CharField(verbose_name='拍品名称', max_length=32)
    status_choices = (
        (1, '未开拍'),
        (2, '预展中'),
        (3, '拍卖中'),
        (4, '成交'),
        (5, '流拍'),
        (6, '逾期未支付'),
    )
    status = models.PositiveSmallIntegerField(verbose_name='状态',
choices=status_choices, default=1)

    cover = models.FileField(verbose_name='拍品封面', max_length=128)

    start_price = models.PositiveIntegerField(verbose_name='起拍价')
    deal_price = models.PositiveIntegerField(verbose_name='成交价', null=True,
blank=True)

    reserve_price = models.PositiveIntegerField(verbose_name='参考底价')
    highest_price = models.PositiveIntegerField(verbose_name='参考高价')

    video = models.CharField(verbose_name='预览视频', max_length=128,
null=True, blank=True)
    deposit = models.PositiveIntegerField(verbose_name='单品保证金',
default=100)
    unit = models.PositiveIntegerField(verbose_name='加价幅度', default=100)

    bid_count = models.PositiveIntegerField(verbose_name='出价次数', default=0)
    look_count = models.PositiveIntegerField(verbose_name='围观次数',
default=0)

    class Meta:
        verbose_name_plural = '拍品'

    def __str__(self):
        return self.title


class AuctionItemImage(models.Model):
    """
    拍品详细图
    """
    item = models.ForeignKey(verbose_name='拍品', to='AuctionItem')
    img = models.FileField(verbose_name='详细图', max_length=64)
    carousel = models.BooleanField(verbose_name='是否在轮播中显示',
default=False)
    order = models.FloatField(verbose_name="排序", default=1)

    class Meta:
        verbose_name_plural = '拍品详细图'

    def __str__(self):
        return "{}-{}".format(self.item.title, self.id, )


class AuctionItemDetail(models.Model):
    """
    拍品详细规格
    """
    item = models.ForeignKey(verbose_name='拍品', to='AuctionItem')
    key = models.CharField(verbose_name='项', max_length=16)
    value = models.CharField(verbose_name='值', max_length=32)

    class Meta:
        verbose_name_plural = '拍品规格'


class BrowseRecord(models.Model):
    """
    浏览记录
    """
    item = models.ForeignKey(verbose_name='拍品', to='AuctionItem')
    user = models.ForeignKey(verbose_name='用户', to='UserInfo')


class BidRecord(models.Model):
    """
    出价记录
    """
    status_choices = (
        (1, '竞价'),
        (2, '成交'),
        (3, '逾期未付款'),
    )
    status = models.PositiveSmallIntegerField(verbose_name='状态',
choices=status_choices, default=1)

    item = models.ForeignKey(verbose_name='拍品', to='AuctionItem')
    user = models.ForeignKey(verbose_name='出价人', to='UserInfo')
    price = models.PositiveIntegerField(verbose_name='出价')
```

```
class DepositRecord(models.Model):
    """ 保证金 """
    status_choices = (
        (1, '未支付'),
        (2, '支付成功')
    )
    status = models.PositiveSmallIntegerField(verbose_name='状态',
choices=status_choices, default=1)

    uid = models.CharField(verbose_name='流水号', max_length=64)
    deposit_type_choices = (
        (1, '单品保证金'),
        (2, '全场保证金')
    )
    deposit_type = models.SmallIntegerField(verbose_name='保证金类型',
choices=deposit_type_choices)
    pay_type_choices = (
        (1, '微信'),
        (2, '余额')
    )
    pay_type = models.SmallIntegerField(verbose_name='支付方式',
choices=pay_type_choices)

    amount = models.PositiveIntegerField(verbose_name='金额')  # 200
    balance = models.PositiveIntegerField(verbose_name='余额')  # 0

    user = models.ForeignKey(verbose_name='用户', to='UserInfo')

    # 单品保证金则设置值，全场保证金，则为空
    item = models.ForeignKey(verbose_name='拍品', to='AuctionItem', null=True,
blank=True)

    auction = models.ForeignKey(verbose_name='拍卖', to='Auction')


class Order(models.Model):
    """
    订单，拍卖结束时，执行定时任务处理:
        - 拍得，创建订单。
        - 未拍得，则退款到原账户
    """
    status_choices = (
        (1, '未支付'),
        (2, '待收货'),
        (3, '已完成'),
        (4, '逾期未支付'),
    )
    status = models.PositiveSmallIntegerField(verbose_name='状态',
choices=status_choices)

    uid = models.CharField(verbose_name='流水号', max_length=64)
    user = models.ForeignKey(verbose_name='用户', to='UserInfo')
    item = models.ForeignKey(verbose_name='拍品', to='AuctionItem')
    deposit = models.ForeignKey(verbose_name='保证金', to='DepositRecord')
    price = models.PositiveIntegerField(verbose_name='出价')

    create_date = models.DateTimeField(verbose_name='创建时间',
auto_now_add=True)
    twenty_four_task_id = models.CharField(verbose_name='24小时后定时任务',
max_length=32, null=True, blank=True)

    address = models.ForeignKey(verbose_name='收获地址', to='Address',
null=True, blank=True)

    pay_type_choices = (
        (1, '微信'),
        (2, '余额')
    )
    pay_type = models.SmallIntegerField(verbose_name='支付方式',
choices=pay_type_choices)
    real_price = models.PositiveIntegerField(verbose_name='实际支付金额',
null=True, blank=True)


class DepositRefundRecord(models.Model):
    """ 保证金退款记录 """
    uid = models.CharField(verbose_name='流水号', max_length=64)
    status_choices = (
        (1, "待退款"),
        (2, '退款成功'),
    )
    status = models.PositiveSmallIntegerField(verbose_name='状态',
choices=status_choices)
    deposit = models.ForeignKey(verbose_name='保证金', to='DepositRecord')
    amount = models.PositiveIntegerField(verbose_name='退款金额')


class DepositDeduct(models.Model):
    """ 扣除保证金 """
    order = models.ForeignKey(verbose_name='订单', to='Order')
    amount = models.PositiveIntegerField(verbose_name='金额')

    deduct_type_choices = (
        (1, '逾期扣款'),
        (2, '支付抵扣')
    )
```

```python
    deduct_type = models.SmallIntegerField(verbose_name='扣款类型',
choices=deduct_type_choices, default=1)


class Coupon(models.Model):
    """ 优惠券 """
    status_choices = (
        (1, '未开始'),
        (2, '领取中'),
        (3, '已结束')
    )
    status = models.SmallIntegerField(verbose_name='状态',
choices=status_choices, default=1)
    title = models.CharField(verbose_name='优惠券名称', max_length=32)
    auction = models.ForeignKey(verbose_name='专场', to='Auction')

    money = models.PositiveIntegerField(verbose_name='抵扣金额', default=200)

    count = models.PositiveIntegerField(verbose_name='创建数量', default=100)
    apply_count = models.PositiveIntegerField(verbose_name='已申请数量',
default=0)

    apply_start_date = models.DateTimeField(verbose_name='开始领取时间')
    apply_stop_date = models.DateTimeField(verbose_name='领取结束时间')

    apply_start_task_id = models.CharField(verbose_name='celery任务ID',
max_length=64, null=True, blank=True)
    apply_stop_task_id = models.CharField(verbose_name='celery任务ID',
max_length=64, null=True, blank=True)

    deleted = models.BooleanField(verbose_name='是否删除', default=False)


class UserCoupon(models.Model):
    status_choices = (
        (1, '未使用'),
        (2, '已使用'),
        (3, '已过期')
    )
    status = models.SmallIntegerField(verbose_name='状态',
choices=status_choices, default=1)
    user = models.ForeignKey(verbose_name='用户', to='UserInfo')
    coupon = models.ForeignKey(verbose_name='优惠券', to='Coupon')
    order = models.ForeignKey(verbose_name='订单', to='Order', null=True,
blank=True)


class Address(models.Model):
    """ 地址 """
    name = models.CharField(verbose_name='收货人姓名', max_length=32)
    phone = models.CharField(verbose_name='联系电话', max_length=11)
    detail = models.CharField(verbose_name='收货地址', max_length=255)

    user = models.ForeignKey(verbose_name='用户', to='UserInfo')
```

2.项目开发

2.1 任务：订单列表开发

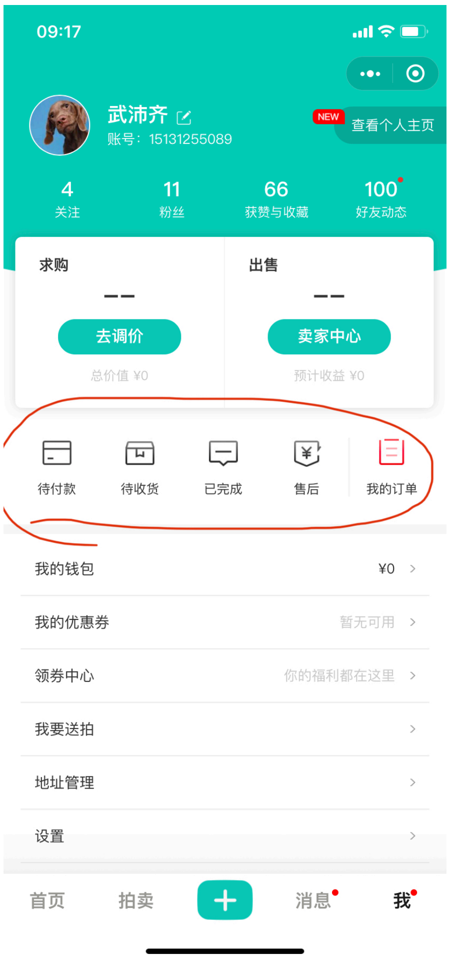
```
在小程序端,home页面编写点击跳转，指向同一个url+通过get传参到我的订单页面。
    url="/pages/order/order?orderType=3"

在小程序端，我的订单页面。
    向后台发送ajax请求获取当前用户所有的订单，并根据订单的状态进行结构化处理。
    {
        1:{
            text:'待支付',
            child:[
                {...},
            ]
        }
        2:
        3:
        4:

    }
    根据跳转过来时传入的参数orderType=3，进行订单的展示。

    点击根据状态切换订单的展示。

参考上节的示例代码。
```


09:17

武沛齐

账号：15131255089

NEW 查看个人主页

4	11	66	100
关注	粉丝	获赞与收藏	好友动态

求购

——

去调价

总价值 ¥0

出售

——

卖家中心

预计收益 ¥0

待付款 待收货 已完成 售后 我的订单

我的钱包 ¥0

我的优惠券 暂无可用

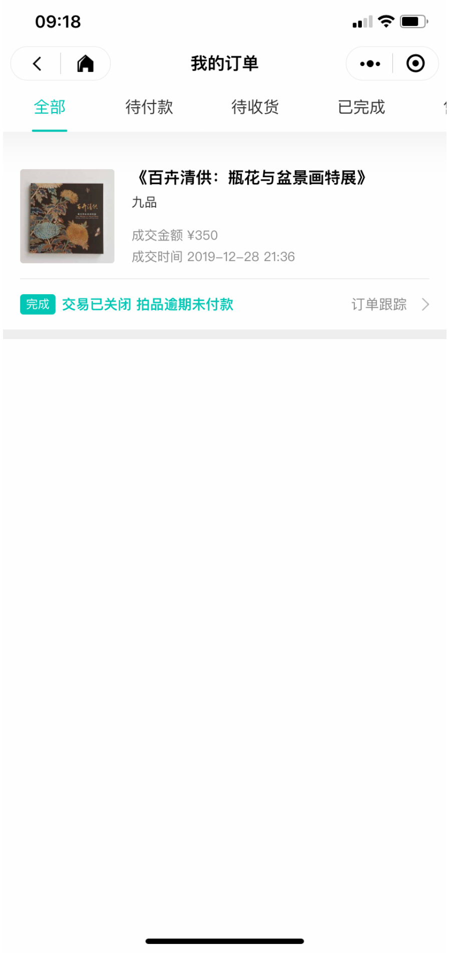
领券中心 你的福利都在这里

我要送拍

地址管理

设置

首页 拍卖 + 消息 我


09:18

我的订单

全部 待付款 待收货 已完成

《百卉清供：瓶花与盆景画特展》

九品

成交金额 ¥350

成交时间 2019-12-28 21:36

完成 交易已关闭 拍品逾期未付款 订单跟踪

2.2 订单列表开发

- API，返回所有的订单。

```
class OrderModelSerializer(serializers.ModelSerializer):
    cover = serializers.CharField(source='item.cover')
    create_date = serializers.DateTimeField(format="%Y-%m-%d %H:%M")
    title = serializers.CharField(source='item.title')

    class Meta:
        model = models.Order
        exclude = ['uid', 'twenty_four_task_id', 'user', 'deposit']

    def text(self,obj):
        # model_to_dict(obj.item,['title','cover'])
        return {
            title:obj.item.title,
            cover:obj.item.cover.name
        }
class OrderView(ListAPIView):
    """ 订单接口 """
    authentication_classes = [UserAuthentication, ]
    # queryset = models.Order.objects.all().order_by('id')
    serializer_class = OrderModelSerializer

    def get_queryset(self):
        return
models.Order.objects.filter(user=self.request.user).order_by('id')

    def list(self, request, *args, **kwargs):
        response = super().list(request, *args, **kwargs)
        if response.status_code != status.HTTP_200_OK:
            return response

        info = OrderedDict()
        for item in models.Order.status_choices:
            info[item[0]] = {'text': item[1], 'child': []}

        for item in response.data:
            info[item['status']]['child'].append(item)
        response.data = info
        return response
```

- 小程序

```
    <navigator class="item" url="/pages/order/order?seleted=1">
      <image src="/static/images/icon/transaction_order1_icon_show.png">
</image>
      <text>未支付</text>
    </navigator>
    <navigator class="item" url="/pages/order/order?seleted=2">
      <image src="/static/images/icon/transaction_order2_icon_show.png">
</image>
      <text>待收货</text>
    </navigator>
    <navigator class="item" url="/pages/order/order?seleted=3">
      <image src="/static/images/icon/transaction_order3_icon_show.png">
</image>
      <text>已完成</text>
    </navigator>
    <navigator class="item" url="/pages/order/order?seleted=4">
      <image src="/static/images/icon/transaction_order5_icon_show.png">
</image>
      <text>逾期未支付</text>
    </navigator>
```

```
<view class="container">
  <view class="header">
    <view class="{{ seleted == index? 'active' : '' }}" wx:for="
{{orderDict}}" wx:key="index" bindtap="changeStatus" data-id="{{index}}">
{{item.text}}</view>
  </view>
  <view class="content">
    <view class="item" wx:for="{{orderDict[seleted].child}}" wx:key="idx"
bindtap="toPay" data-item="{{item}}">
      <view class="cover">
        <image src="{{item.cover}}"></image>
      </view>
      <view class="info">
        <view class="big">
          <text>{{item.title}}</text>
        </view>
        <view class="small">
          <view>成交额：{{item.price}}</view>
          <view>成交时间：{{item.create_date}}</view>
        </view>
      </view>
    </view>
  </view>
</view>
```

```
// pages/order/order.js
var api = require('../../config/api.js')
var app = getApp();

Page({
```

```
/**
 * 页面的初始数据
 */
data: {
  seleted:null,
  orderDict:{}
},
toPay:function(e){
  var order = e.currentTarget.dataset.item;
  if(order.status != 1){
    return
  }
  wx.navigateTo({url: '/pages/pay/pay?orderId=' + order.id});

},
changeStatus: function (e) {
  var seleted = e.currentTarget.dataset.id;
  this.setData({
    seleted: seleted
  })
},
getOrder:function(){
  var userInfo = app.globalData.userInfo;
  wx.request({
    url: api.Order,
    header: {
      Authorization: userInfo ? "token " + userInfo.token : ""
    },
    method: 'GET',
    dataType: 'json',
    responseType: 'text',
    success: (res) => {
      this.setData({
        orderDict: res.data
      })
    },
    fail: function (res) { },
    complete: function (res) { },
  })
},
/**
 * 生命周期函数--监听页面加载
 */
onLoad: function (options) {
  this.setData({
    seleted:options.seleted
  })
  this.getOrder();
},
  onPullDownRefresh: function () {
    this.getOrder();
  },

})
```

2.3 任务：支付

点击未支付的订单，显示相关数据并进行支付。

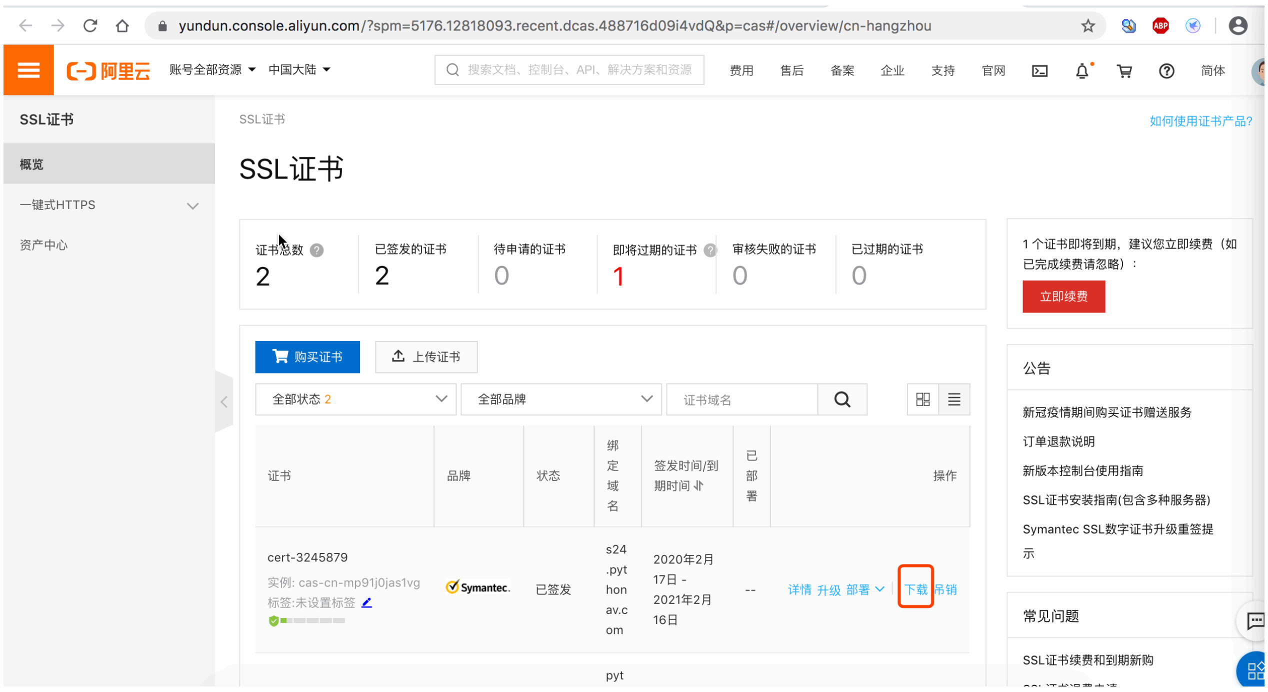
```
- 订单列表页面，点击事件并传入订单ID
- 支付页面，
    接收订单ID，并向后台接口发送request请求获取订单相关信息：拍品、成交、优惠券、保证金、余额
- 立即支付
    用户选择的数据发送API

- API，接收到数据之后进行交易
    成交价 - 优惠券 <= 0
    成交价 - 优惠券 - 保证金 = 0
    成交价 - 优惠券 - 保证金 < 0
        抵扣
        退还(判断如果是专场保证金&还有未支付的订单)
    成交价 - 优惠券 - 保证金 > 0
        余额支付

    操作表结构：
        订单表
        我的优惠券
        抵扣记录
        保证金
        退款
        用户表
```

2.4 任务：https证书

阿里云申请免费的https证书 + 云服务器（最好）


yundun.console.aliyun.com/?spm=5176.12818093.recent.dcas.488716d09i4vdQ&p=cas#/overview/cn-hangzhou

阿里云 账号全部资源 中国大陆 搜索文档、控制台、API、解决方案和资源 费用 售后 备案 企业 支持 官网 简体

SSL证书
概览
一键式HTTPS
资产中心

SSL证书
如何使用证书产品?

SSL证书

证书总数	已签发的证书	待申请的证书	即将过期的证书	审核失败的证书	已过期的证书
2	2	0	1	0	0

1 个证书即将到期，建议您立即续费（如已完成续费请忽略）：
立即续费

购买证书 上传证书
全部状态 2 全部品牌 证书域名

证书	品牌	状态	绑定域名	签发时间/到期时间	已部署	操作
cert-3245879 实例: cas-cn-mp91j0jas1vg 标签:未设置标签	Symantec.	已签发	s24.python av.com	2020年2月17日 - 2021年2月16日	--	详情 升级 部署 下载 吊销
			pyt			

公告
新冠疫情期间购买证书赠送服务
订单退款说明
新版本控制台使用指南
SSL证书安装指南(包含多种服务器)
Symantec SSL数字证书升级重签提示

常见问题
SSL证书续费和到期新购

```
https://common-buy.aliyun.com/?
spm=5176.2020520163.cas.1.3c0956a7fVhtqD&commodityCode=cas#/buy
```

答疑

1.数据库迁移

```
ORM类           migrations目录              数据库
    makemigrations               migrate

开发者正常操作，想要修改数据库修改，不能手动去改数据字段必须通过ORM类+2个命令实现。

情景1：直接去数据库删除了某一列。
    第一步：在ORM类中删除字段
    第二步：执行makemigrations，migrations目录生成配置去删除N字段。
    第三步：执行migrate命令[报错，要去删除N字段，但是数据库中么有这个字段]
            python manage.py migrate --fake

总结：
    a. 三方保持一致，所有行为都是从ORM类开始。
    b. migrate --fake使三方一致。
```