# 今日任务

1. 订单系统
2. 支付系统
3. 交易系统
4. 微服务实现

# 订单系统

## 综述

**订单是电商体系的核心，有了订单才有业绩和盈利，才有谈发展、谈梦想的可能性。**

订单中包含商品、 优惠、 用户、 收货信息、 支付信息等一系列的订单实时数据。 通过订单中心，实现对线上订单、 线下订单及第三方订单的管理，支持订单接收、 订单自动合并与拆分、 自动匹配仓库、 库存控制、 自动匹配快递、 结算与支付等订单生命周期中的一系列协同作业。依靠灵活多变的订单产品设计架构，可满足电商企业百万级的订单业务处理需求，提升订单流转的工作效率。在订单生成之后，会随着订单的流转更新状态。 不同业务类型的订单状态，例如机票、 服务订单、 商品服务订单等，和最常见的纯实物商品的订单状态会有所区别。 以实物商品为例，我们来讨论一下订单状态的流转。 **订单状态主要有以下几种类型**。

（1） 待付款：用户刚提交订单，尚未付款，等待用户支付。 **由于待付款状态会锁定库存，所以一般会设置超时自动取消（如0.5小时、 1小时、 24小时等时限） 功能。**

（2） 待发货：用户付款之后，等待商家发货。

（3） 待收货：商家已发货，等待用户收货。

（4） 交易成功：用户确认收货之后，订单已完成交易。

（5） 已取消：付款之前取消订单。 超时未付款或用户取消订单都会产生这种订单状态。

（6） 售后中：用户在付款后发货前申请退款，或商家发货后用户申请退、 换货，都会产生这种订单状态。 订单售后状态又分许多种，后面详述。

（7） 交易关闭：当售后完成后的订单状态。 “已取消”的订单状态可以合并到“交易关闭”中。

**订单状态的正常流转是：①待付款、 ②待发货、 ③待收货、 ④交易成功。** 但订单会有逆向流程，和发生的时间节点及类型相关，情况也很复杂多变。

订单的售后状态主要有以下几种。

（1） 待审核：用户提交退货、 退款申请之后，等待审核的状态。在用户已付款待发货的状态下，订单未推送至仓库或者在仓库拦截发货成功，系统可直接审核通过。 当审核不通过时，回到正常流程中。

（2） 待退货入库：退货申请审核通过，等待用户退货入库。

（3） 待退款：退货入库成功后，等待退款给用户。

（4） 待换货入库：换货申请审核通过，等待用户换货入库。

（5） 换货出库中：换货入库之后，生成换货出库单，订单出库。

（6） 售后成功：当退货、 退款成功或换货成功之后，流转至“售后成功”状态。 退货、 退款的售后成功在主流程下属于“交易关闭”。

在售后管理中，还有一个值得思考的环节：多次售后。 当换货成功之后，在流程上还是允许客户有售后环节的。 那么在产品设计中，就应该考虑允许用户多次发起售后。 另外系统应设置申诉周期，保证商家利益，例如：在发货之后7天自动确认收货，交易成功15天后售后通道关闭。

## 步骤

（1） 在订单过程中进行安全校验，主要是检测用户是否在黑名单上、 用户购买行为是否正常等，当检测到不正常时，终止下单。

（2） 从商品中心获取商品信息（SKU、 规格、 价格等） 。

（3） 从营销中心获取商品、 订单促销信息（优惠券、 促销活动） ，判断是否满足优惠条件，计算出优惠金额。

（4） 在会员中心获取会员权益，例如平台抵扣积分、 折扣条件等。

（5） 在调度中心校验销售层库存，按照调度规则锁定区域库存。

（6） 根据拆单规则（商家、 仓库、 订单类型等） 将订单拆分成若干个子订单，根据运费模板计算运费，根据商品金额、 运费、 优惠金额计算应付金额（实付款） 。

## 内容

用户信息：用户账号、 用户等级。

订单基础信息：父订单与子订单、 订单编号、 订单状态。

收货信息：收货地址、 收货人姓名、 联系电话、 邮编。

商品信息：SKU信息、 规格、 商品数量、 价格、 商品图片、 商家（店铺） 。优惠信息：优惠券、 促销活动、 虚拟币抵扣金额。

支付信息：支付方式、 支付单号、 商品总金额、 实付金额、 运费、虚拟币抵扣金额、 优惠券优惠金额、 总优惠金额。

物流信息：物流公司、 物流单号、 物流状态。

其他信息：发票信息、 下单平台、 分销渠道。

# 支付系统

## 综述

支付系统是指管理支付数据，调用第三方支付平台接口，记录支付信息（对应订单号、 支付金额等） ，支付对账，提供支付服务和资金清算的系统。 大部分电商企业没有支付牌照，自建支付产品的可能性较小，目前都是接入第三方支付服务的方式进行支付。

## 方式

在电商平台中，提供多种支付方式，如**微信支付、 支付宝、 百度钱包、 信用卡、 储蓄卡、 Apple Pay等，主要分为三类支付渠道：第三方平台支付、 快捷支付、 网银支付。** 通常在订单结算页，用户可选择支付方式。

1．第三方平台支付

这种支付方式在电商平台中应用范围最为广泛，如**支付宝、 微信支付**等都是第三方支付平台。 使用时，需要用户已经注册并登录到第三方支付平台，并且账户有余额或者绑定银行卡，电商平台调用第三方支付平台接口完成支付。 这些年支付宝、 微信支付已经成为了电商的主要支付方式。

2．网银支付

网银支付是指用户在支付的时候，需要跳转到银行网银页面来完成支付。 在**网银支付**页面，需要输入用户的卡号和身份信息。 这种支付方式要求用户开通网银，并且跳转出站外，中断当前流程。 目前主要在PC端用，通过直接对接银行或者银联平台来实现。银联可提供大部分银行卡（储蓄卡、 信用卡） 的网银支付和快捷支付服务，能够帮助避免一家家银行对接的麻烦。

3．快捷支付

用户绑定银行卡并开通后，电商平台不留存卡号信息，仅留存Token标识符，后续支付时通过Token标识符、 短信验证码等完成支付，安全性更高。 对于小额度的支付，甚至可以开通小额免密，直接完成支付。 这种方式安全快捷。 快捷支付产品是通过对接银行或者银联提供的快捷支付接口来实现的。 绑定快捷支付就涉及签约、 解约、 支付等环节。

与其他支付渠道不同的是，快捷支付需要绑定用户交易账号，绑定用户ID，记录支付密码（平台） 、 确认是否允许免密支付、 用于支付的手机号等信息。 在用户支付时直接调用Token，简化付款流程。

用户支付时资金从其账户中流出，那流入哪里呢？这就需要电商平台在对接平台开设收款账户，如常见支付宝商家中心、 微信支付商户平台、 中国银联商家中心等。 若用户通过支付宝支付，那资金就流入电商企业的支付宝商户账号中。 电商平台通过支付渠道来完成收款、 退款、对账等流程。

对电商平台来说，所有资金并没有经过电商系统，而是在用户资金账户到平台渠道账户中的流动，并且渠道还要收取费用。

## 记录

**支付中心会记录下交易记录，其与订单记录不同，主要包含支付信息。**

主要包含的信息有交易流水号（一般由第三方平台生成） 、 创建时间、 完成时间、 关联的订单编号、 订单标题、 支付场景、 费用信息（订单总金额、 优惠金额、 实际费用） 、 订单描述。 此外，支付信息还包含以下内容。

（1） 交易渠道信息：记录所使用的交易渠道（支付宝、 微信） ，渠道账户，渠道执行支付的时间、 渠道侧返回的订单号等。 如果有错误发生，还需要记录从渠道接收到的错误信息和错误码。

（2） 支付用户信息：包括用户类型、 用户账号、 渠道账号信息。

（3） 支付状态：包括未支付、 已支付、 已退款、 部分退款、 已取消等。

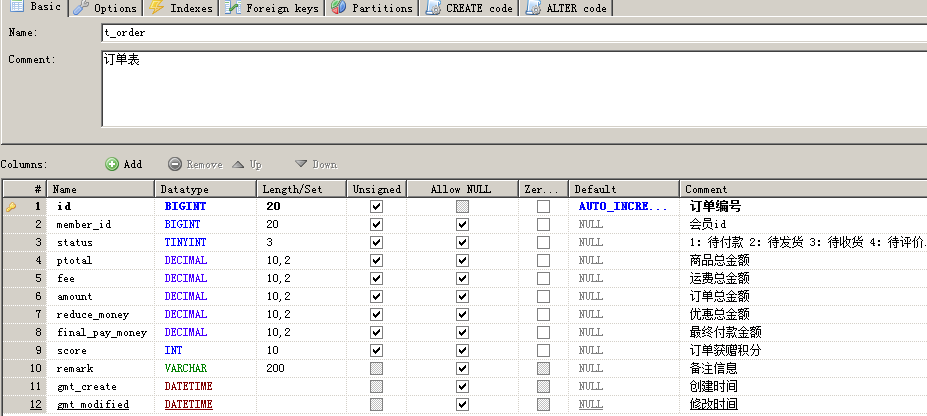
如果有特殊的支付方式，如分期、 货到付款，还会更加复杂。

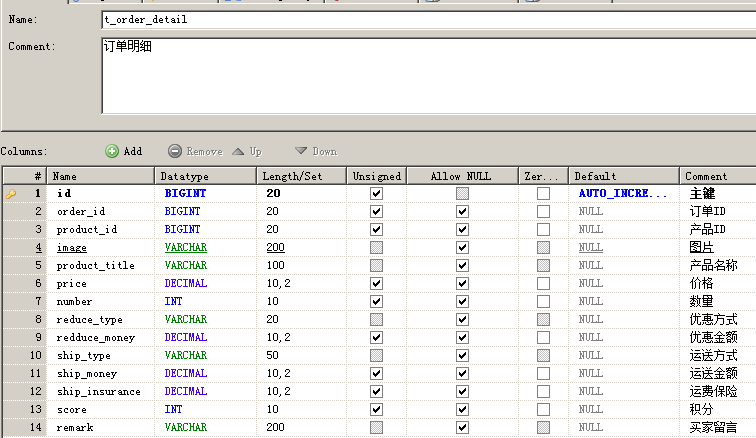
# 交易系统

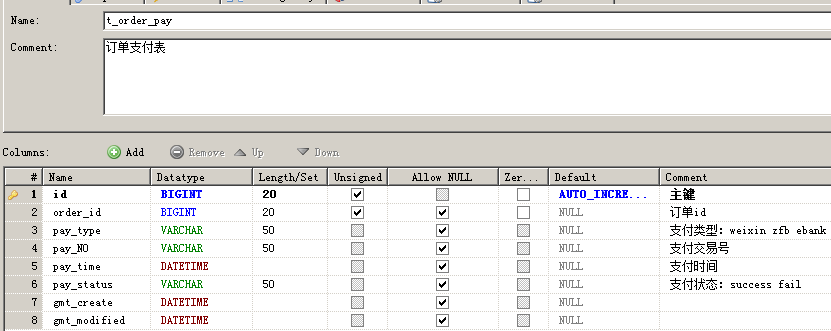
**说明：以上信息是一个真正的大型线上电商平台的订单及支付系统的部分内容，目的更多为了大家理解业务，面试时做到心中有数。我们为了学习的方便性，对内容进行简化，也更便于大家实现功能。**

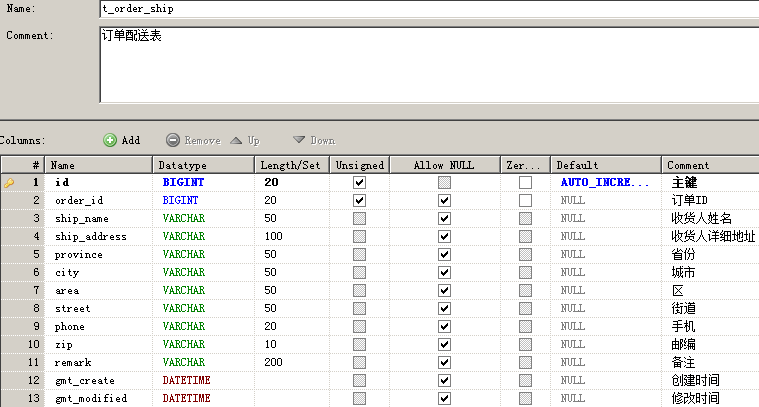
**所以我们将订单和支付2个系统合并成一个交易系统，其实在电商早期这也是很现实的做法。**

## 表设计









# 实现

go8-mall中下单，确认，支付，成功等页面

go8-trade微服务

页面URL

trade restful api

# 下单

|  |
| --- |
| jQuery.ajax({  url:'http://localhost:4040/order/add',  type:'post',  data: JSON.stringify(order),  dataType:'json',  contentType: 'application/json;charset=utf-8',  success:**function**(res){  **if**(res.status == 200 && res.data){  **if**(Cookies.get(GO8.TOKEN\_KEY))  window.location.href = GO8.DN.PAY+"pay.html?orderId="+res.data+"&token="+Cookies.get(GO8.TOKEN\_KEY)  **else**{  window.location.href = GO8.DN.PAY+"pay.html?orderId="+res.data  }  }**else**{  vueOrder.$Message.error(res.message);  }  }  }) |

|  |
| --- |
| @PostMapping("/order/add")  **public** Object add(@RequestBody Order order) {  **try** {  orderService.add(order);  **return** ServiceMessage.*ok*(order.getId());  } **catch** (Exception e) {  e.printStackTrace();  **return** ServiceMessage.*error*();  }  } |

|  |
| --- |
| @Override  **public** **void** add(Order order) {  Date now = **new** Date();  order.setGmtCreate(now);  order.setGmtModified(now);  order.setStatus((**byte**)1);  orderMapper.insert(order);    List<OrderDetail> details = order.getDetails();  **for**(**int** i=0;i<details.size();i++) {  details.get(i).setOrderId(order.getId());  orderDetailMapper.insert(details.get(i));  }    order.getShip().setOrderId(order.getId());  order.getShip().setGmtCreate(now);  order.getShip().setGmtModified(now);  orderShipMapper.insert(order.getShip());  } |

# 支付

|  |
| --- |
| payDone(){  let orderPay = {}  orderPay.orderId = **this**.orderId  orderPay.payType = "weixin"  orderPay.payNo = **new** Date().getTime()  orderPay.payTime = **new** Date().Format("yyyy-MM-dd hh:mm:ss")  console.log(orderPay.payTime)  orderPay.payStatus = "success"  //第三方支付完成后执行，记录支付信息  jQuery.ajax({  url:'http://localhost:4040/order/pay',  type:'post',  data: orderPay,  dataType:'json',  success:**function**(res){  console.log(res)  }  })    //付款成功跳转到成功页面  **if**(orderPay.payStatus == "success"){  window.location.href = GO8.DN.ORDER+"order/paySuccess.html?orderId="+**this**.orderId  }  } |

|  |
| --- |
| //记录支付信息，如果成功付款，修改订单状态  @Override  **public** **void** pay(OrderPay orderPay) {  //调用第三方平台  Date now = **new** Date();  orderPay.setGmtCreate(now);  orderPay.setGmtModified(now);  orderPayMapper.insertSelective(orderPay);    //支付成功后修改订单状态  **if**(orderPay.getPayStatus().equals("success")) {  Order record = **new** Order();  record.setId(orderPay.getOrderId());  record.setStatus((**byte**)2);  orderMapper.updateByPrimaryKeySelective(record);  }  } |

# 支付成功

|  |
| --- |
| jQuery.ajax({  url:'http://localhost:4040/order/'+vuePaySuccess.orderId,  type:'get',  dataType:'json',  async: **true**,  success:**function**(res){  **if**(res.status == 200 && res.data){  console.log("get order info success")  vuePaySuccess.order.finalPayMoney = res.data.finalPayMoney  vuePaySuccess.order.ship.shipName = res.data.ship.shipName  vuePaySuccess.order.ship.shipAddress = res.data.ship.shipAddress  vuePaySuccess.order.ship.province = res.data.ship.province  vuePaySuccess.order.ship.city = res.data.ship.city  vuePaySuccess.order.ship.area = res.data.ship.area  vuePaySuccess.order.ship.street = res.data.ship.street  vuePaySuccess.order.ship.phone = res.data.ship.phone  vuePaySuccess.order.ship.zip = res.data.ship.zip  }  }  }) |

|  |
| --- |
| //清除已付款购物车中商品，并删除服务器端的购物车商品  jQuery.ajax({  url:'http://localhost:4040/order/'+vuePaySuccess.orderId+'/detail',  type:'get',  dataType:'json',  success:**function**(res){  **if**(res.status == 200 && res.data){  console.log("get order details success")  let carts = JSON.parse(Cookies.get(GO8.CART\_KEY))  //delete ids  let ids = []  **for**(let i=0;i<res.data.length;i++){  **for**(let k=0;k<carts.length;k++){  **if**(res.data[i].productId == carts[k].id){  carts.splice(k,1)  **break**  }  }  ids.push(res.data[i].productId)  }  console.log(carts)  Cookies.set(GO8.CART\_KEY,JSON.stringify(carts))  ids = ids.join(",")  console.log(ids)  jQuery.ajax({  url:'http://localhost:8888/cart/sync',  type:'post',  data:{'userId':vueUser.user.id,"ids":ids},  dataType:'json',  success:**function**(res){  **if**(res.status == 200 && res.data){  console.log("cart sync success")  }  }  })  }  }  }) |

|  |
| --- |
| @GetMapping("/order/{id}")  **public** Object getById(@PathVariable **long** id) {  **try** {  Order order = orderService.get(id);  **return** ServiceMessage.*ok*(order);  } **catch** (Exception e) {  e.printStackTrace();  **return** ServiceMessage.*error*();  }  }    @GetMapping("/order/{orderId}/detail")  **public** Object getOrderDetailsByOrderId(@PathVariable **long** orderId) {  **try** {  List<OrderDetail> details = orderService.getDetailsByOrderId(orderId);  **return** ServiceMessage.*ok*(details);  } **catch** (Exception e) {  e.printStackTrace();  **return** ServiceMessage.*error*();  }  } |