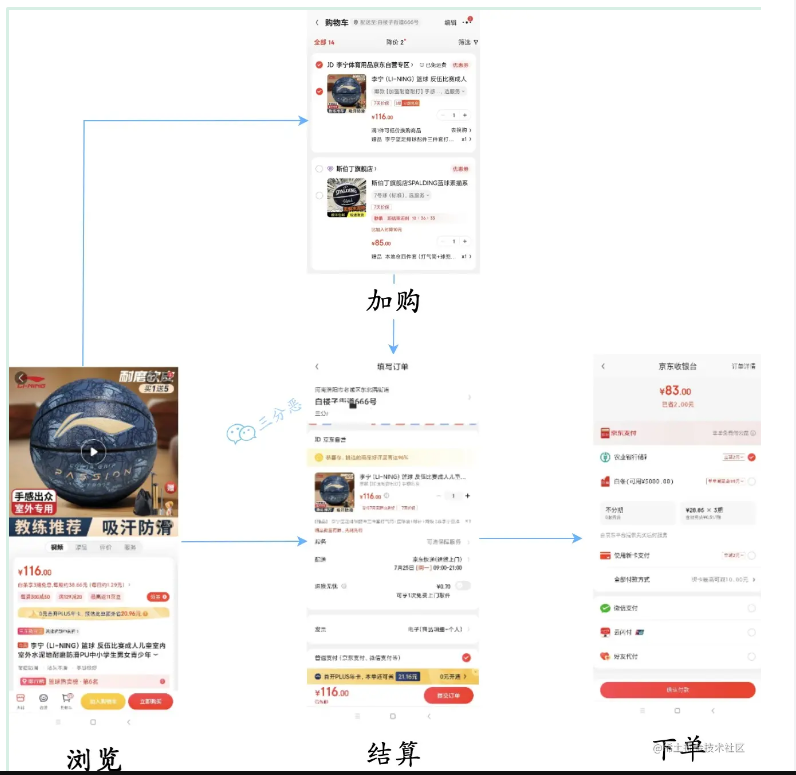
# 用户下单流程

我们从用户浏览商品开始，看看用户下单的简要过程：



1、浏览商品：用户查看商品详情

2、加购/结算：用户可以选择直接购买商品，也可以先加入购物车，用户购买的这一步就是结算

3、确认下单：结算完成，就进入了下单页面，　提交订单，这一步就会生成一个订单，然后进入付款页面。

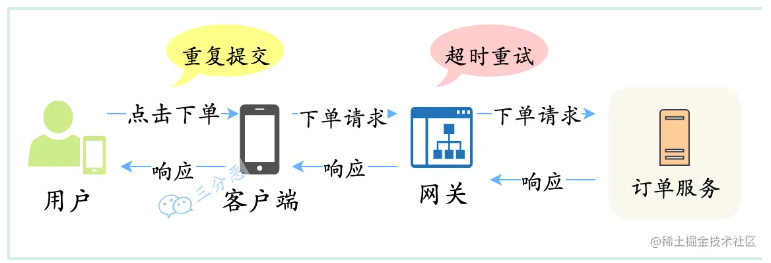
我们可以看到，下单是发生在结算之后，下单之后，会生成唯一的订单号，接下里，客户端需要用这个订单号去完成支付。

# 为什么会重复下单

为什么会重复下单，对于订单服务而言，就是接到了多个下单的请求，原因可能有很多，最常见的是这两种：

1、用户重复提交

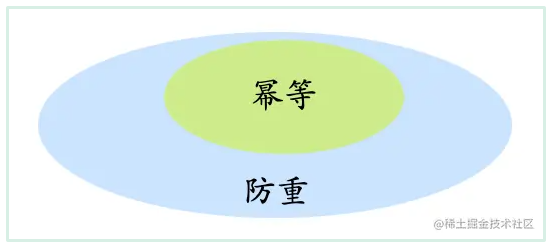
2、网络原因导致的超时重试



# 如何防止重复下单

防止用户提交，最常规的做法，就是客户端点击下单之后，在收到服务端响应之前，按钮置灰。当然防止重复下单，肯定不能只依靠客户端，可能会因为一些网络的抖动，导致仍然有重复的请求到达服务端，所以还是要在服务端做防重/幂等的处理。

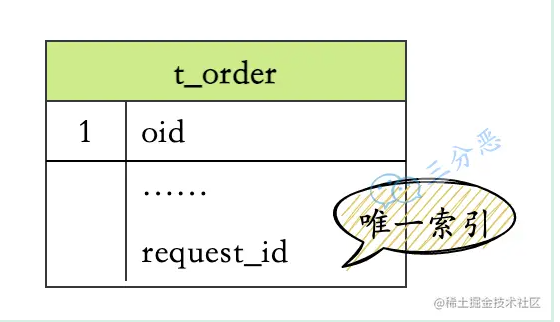
PS：这里额外插入一点我对防重和幂等的理解：防重指的是防止重复提交，幂等指的是多次请求如一次，简单说，就是防重可以给对重复请求抛异常，幂等是对重复的请求响应第一次的结果，在我们讨论的这个场景里，幂等就是响应唯一的订单号。



防重第一步，需要识别请求是否重复，这一步，需要客户端配合实现。大家想一下，下单的时候，服务端怎么去判断这个下单请求是否唯一呢？金额？商品？优惠券？……万一用户就是喜欢，又下了一个一摸一样的单呢？所以，需要客户端在请求下单接口的时候，需要生成一个唯一的请求号：requestId，服务端拿这个请求号，判断是否重复请求。

# 利用数据库实现幂等

可以在订单表t\_order里添加一个字段：requestId，添加唯一索引：



这样一来，如果是重复的请求，在落库的时候就会报错，为了保证幂等性，我们可以catch住这个异常，根据requestId获取订单号，然后向客户端响应订单号。

大概的代码如下：



当然，这里不太好的地方是，拿异常来做了业务判断。

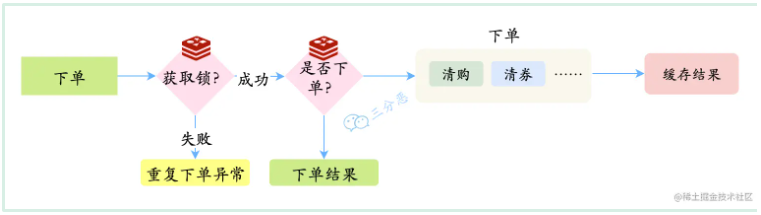
# 利用Redis防重

另外一个办法，就是下单请求的时候要加锁了，通常我们的服务都是集群部署，所以一般都是用Redis实现分布式锁。

大概的逻辑：

1、就是以requestId为维度进行加锁，如果获取锁失败，就抛一个自定义的重复下单异常。

2、如果获取到锁，先check一下，是否已经下单，为了提高性能，下单完成后，也把下单的结果放在Redis缓存里。



大概的代码如下：

文本

描述已自动生成

这里再说明一下：

1、为什么获取不到锁的时候要抛异常呢？

因为下单里面其实还有一些其它的业务流程，比如锁库存、清优惠券……而此时，获取到锁的请求的下单流程还没有结束，下单的结果还获取不到，没法完成响应，也就没办法做幂等。

客户端，也可以根据响应的状态码，进行特殊处理，比如这个异常先不提示，但是允许用户再次点击下单按钮，来提升用户的体验。