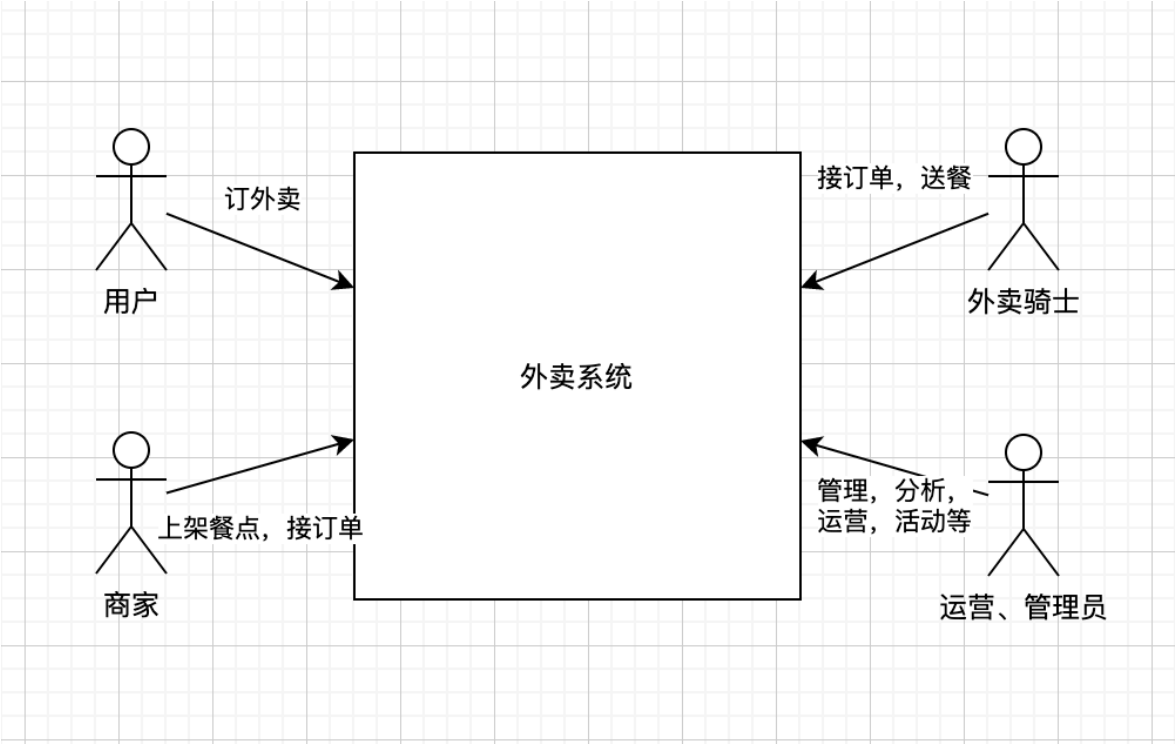
外卖订餐系统架构设计

背景：

为了保障公司的外卖平台业务可以正常运行，需要制作一套外卖订餐的系统，供系统中各个角色进行交互。

系统用例：



如上图所示：系统中主要有商家，用户，外卖骑士，运营和管理员等角色。

其中，商家可以在系统中上线自己的商品和餐点，这些上线的餐点可以用于展示给用户。

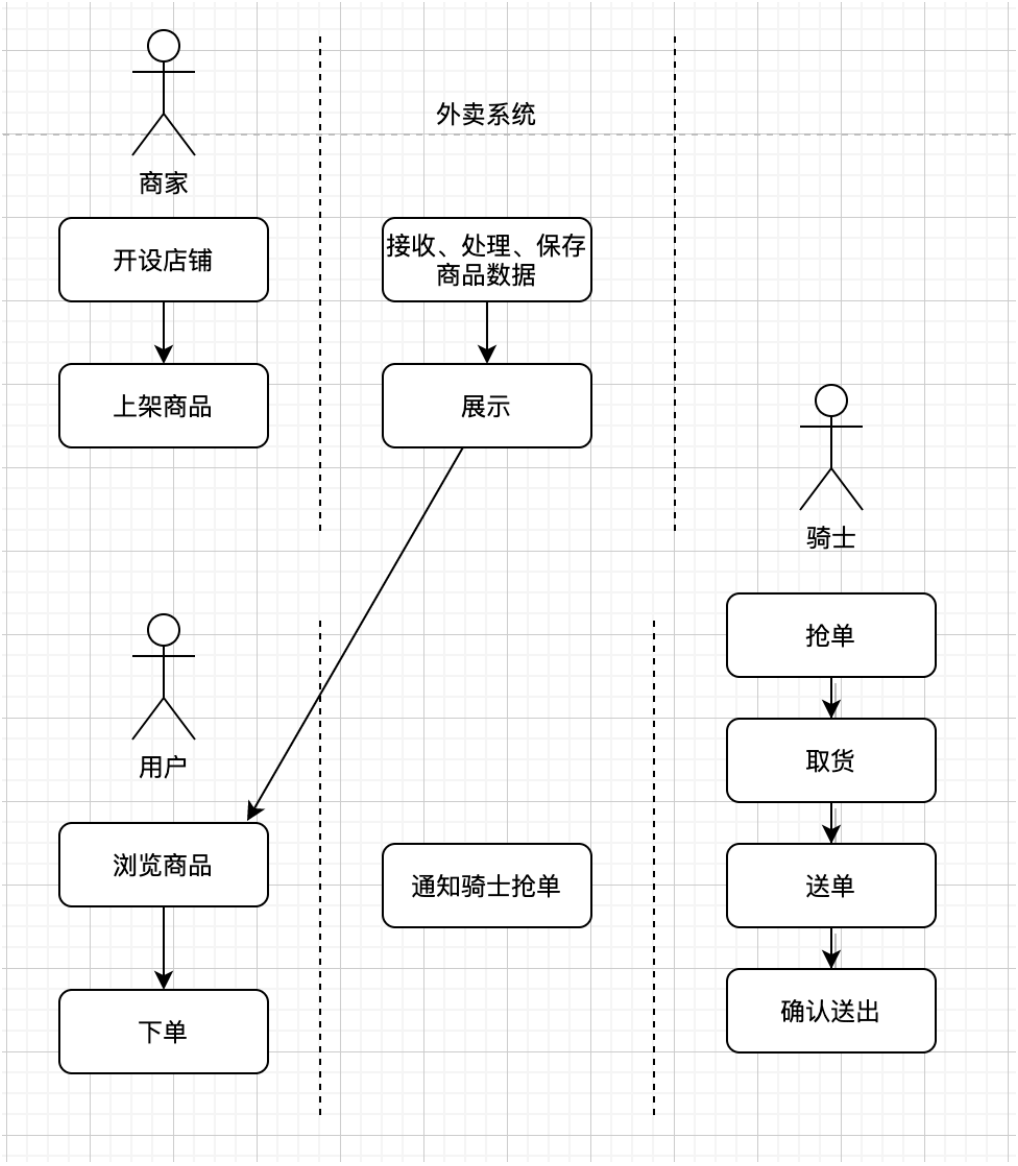
用户进入系统浏览商品和餐点，选中需要的商品下单并完成支付，系统中随即产生一个订单。

系统会针对每个订单通知一定范围内的外卖骑士抢单。

骑士抢到订单后即开始从商家端拿货，送货等行为，成功把商品送到用户手上时，订单即宣告结束。

在整个过程中，运营以及管理人员可以从后台介入，进行一些活动，以及做广告，做分析等商业行为。

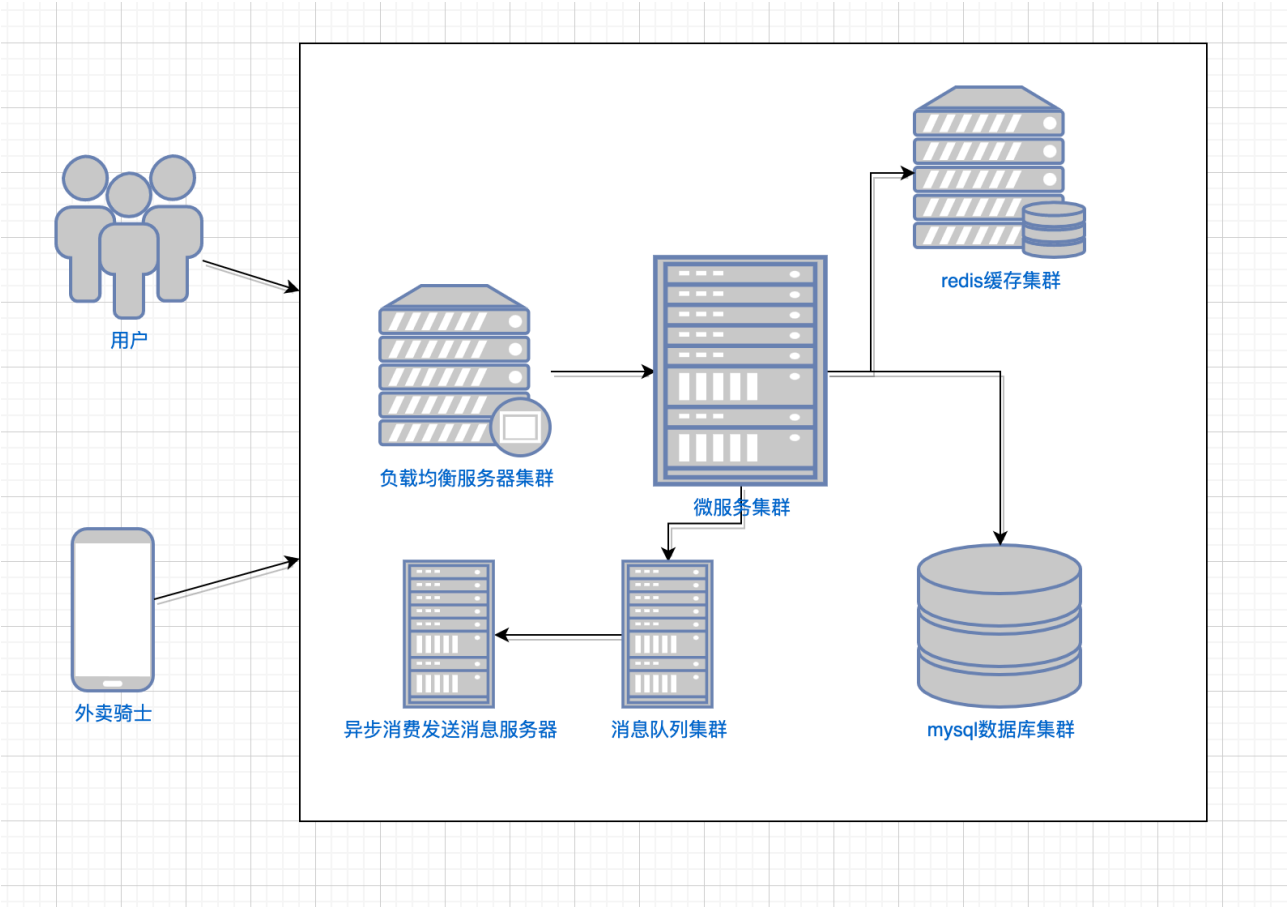
业务抽象以及描述



如上图所示，在整个系统场景中，商家，用户和骑士的业务场景都比较独立，他们从不同的入口进入系统，并各自完成各自的业务流程。

系统作为后台的统筹管理者，将各方有机结合在一起。

系统的部署：



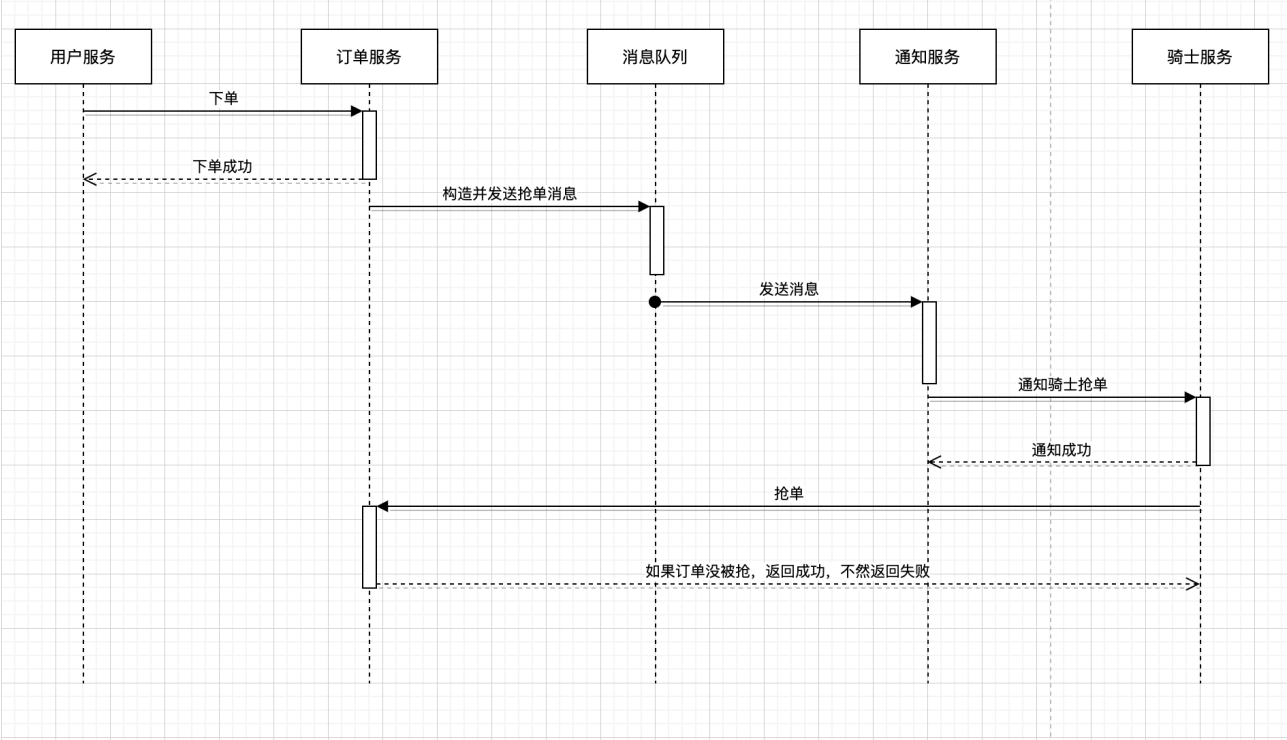
如上图所示，在系统的入口处，由负载均衡服务器集群负责用户，商家和骑士的请求分发。复杂均衡服务器收到请求后通过路由将请求转发给相应的微服务集群，由这些微服务集群来完成业务逻辑。

微服务集群与mysql集群进行交互，用来记录订单的数据和信息。mysql集群需要做主从复制，以防止数据的丢失。如果系统成功发展壮大，后续还要通过分库分表来缓解数据库服务的压力，不过这个工作量比较大，第一期上线先不做。

外卖骑士每隔一定时间上报所处的地点，通过计算更新商家->骑士倒排表，并把信息换存在key value服务器redis集群中，以便于用户下单后迅速找到商家附近的骑士进行订单派送。

用户下单后，微服务集群发送下单的消息到消息队列，比如kafka队列系统中，下游消费系统在收到用户下单的消息后，从redis集群中拉取当前商家附近的骑士信息，并陆续发消息通知。

下单抢单场景系统的交互和时序



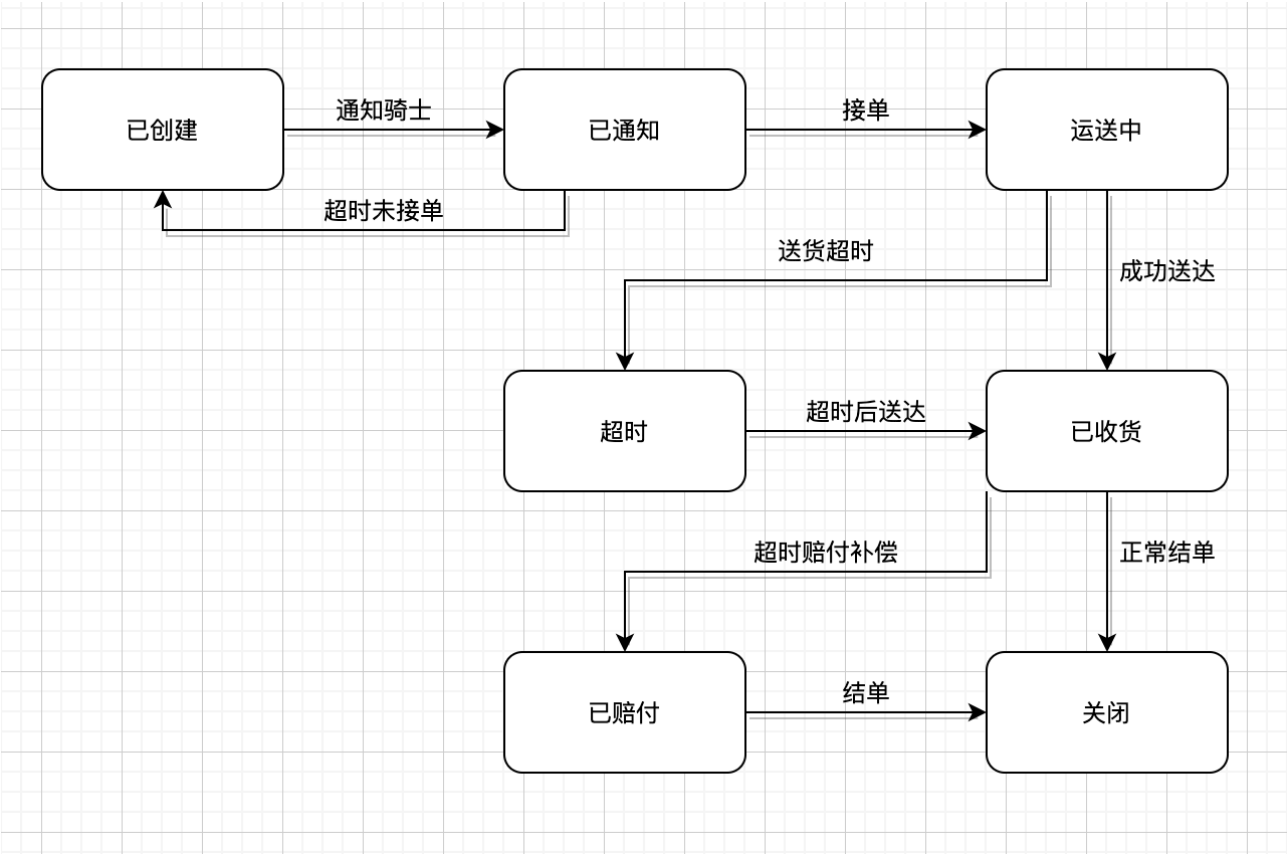
如上图所示，用户服务通过与订单服务的交互完成下单。

订单服务在完成下单时会异步发送一个消息，告知下游用户下了一个订单，并附带订单的信息。

负责通知骑士的服务会订阅这个消息，在收到用户下单的消息后，从redis中获取附近骑士的列表，并发送抢单通知。

骑士在收到抢单通知以后，可以向订单服务发起请求完成抢单。如果抢单成功，则订单的状态推进到下一个阶段

订单的状态流转



如上图所示，一个订单的生命周期起始于用户的下单行为。

用户完成下单后，系统会通知骑士接单，通知成功则状态流转为已通知。如果一段时间内没有骑士阶段，则状态由于超时流转回已创建状态，待再次通知骑士去接单。如果骑士成功接单，则订单状态流转为运送中。

对于运送中的订单，如果在时间限制内成功送达，则流转为已收货，如果没能成功送达，则流转为超时，直到订单最终送到用户手上时再流转为已收货。

视业务逻辑的情况，对于曾经超时的订单可能需要走赔付流程，如果需要走赔付流程，则在完成赔付后进入关闭状态，不然直接进入关闭状态。