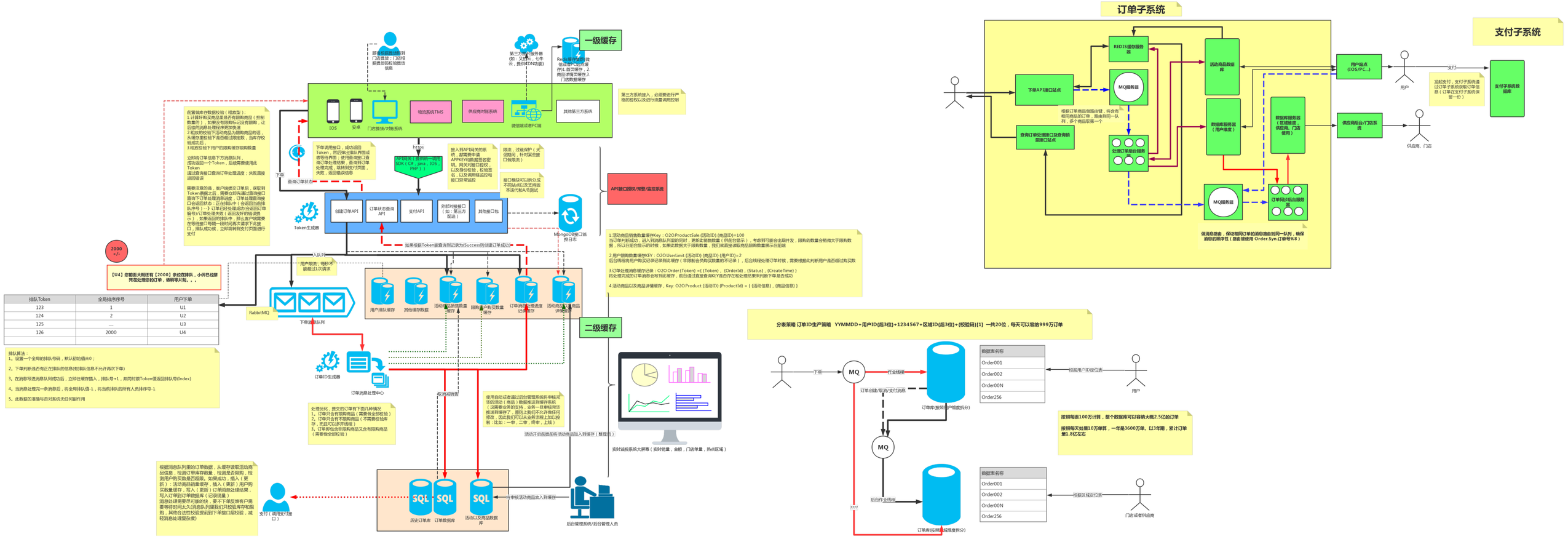
订单子系统
支付子系统
一级缓存
二级缓存
IOS
安卓
门店提货/对账系统
物流系统TMS
供应商对账系统
微信端或者PC端
其他第三方系统
第三方系统接入，必须要进行严格的授权以及进行流量调控限制
https
API网关（Gateway，有SDK（C#，java，IOS，PHP））
下单
查询订单状态
创建订单API
订单状态查询API
支付API
外部对接接口（如：第三方配送）
其他接口恢
Token生成器
MongoDB接口监控日志
API接口授权/限流/监控系统
如果按照Token查询到已录为(Success)则创建订单成功
入队列
RabbitMQ
下单消息队列
用户排队缓存
其他缓存数据
活动商品销售数量缓存
限购量/用户购买数量缓存
订单消息处理进度记录缓存
活动商品/商品评论缓存
订单ID生成器
订单消息处理中心
核销减销量
活动开启前提前将活动商品加入到缓存（整理后）
实时监控系统大屏幕（实时销量，金额，门店单量，热点区域）
将审核活动商品放入到缓存
后台管理系统/后台管理人员
支付（调用支付接口）
SQL
历史订单库
订单数据库
活动以及商品数据库
REDIS缓存服务器
MQ服务器
下单API接口站点
查询订单处理进度接口及查询物流接口站点
处理订单后台服务
活动商品数据库
数据库服务器（用户维度）
数据库服务器（区域维度，供应商、门店使用）
订单同步后台服务器
用户站点（IOS/PC...）
供应商后台/门店系统
支付子系统数据库
用户
供应商、门店
支付
发起支付，支付子系统通过订单子系统获取订单信息（订单在支付子系统保留一份）
根据订单商品做路由键，将含有相同商品的订单，路由到同一队列，多个商品取第一个
做消息路由，保证相同订单的消息路由到同一队列，确保消息的顺序性（路由键使用 Order.Syn.订单号%8）
分表策略 订单ID生产策略 YYMMDD+用户ID(后3位)+1234567+区域ID(后3位)+{校验码}(1) 一共20位，每天可以容纳999万订单
下单
作业线程
订单创建/取消/支付消息
订单库(按用户维度拆分)
数据表名称
Order001
Order002
Order00N
Order256
根据用户ID定位表
用户
MQ
后台作业线程
订单库(按商品/城维度拆分)
根据区域定位表
门店或者供应商
按照每表100万计算，整个数据库可以容纳大概2.5亿的订单
按照每天处单10万单算，一年是3600万单。以3年期，累计订单量1.8亿左右
2000
+/-
【U4】您前面大概还有【2000】余位在排队，小代已经拼死在处理您的订单，请稍等片刻。。。
排队Token
全局排序序号
用户下单
123
1
U1
124
2
U2
125
—
U3
126
2000
U4
排队算法：
1、设置一个全局的排队号码，默认初始值未0；
2、下单判断是否有正在排队的信息(有排队信息不允许再次下单)
3、在消息写进消息队列成功后，立即往缓存插入，排队号+1，并同时跟Token值返回排队号(Index)
4、当消息处理完一条消息后，将全局排队值-1，将当前排队的所有人员排序号-1
5、此数据的准确与否对系统无任何副作用
处理优化，意欠的订单有下面几种情况
1、订单只含有限购商品（需要做全部校验）
2、订单只含有不限购商品（不需要校验库存，而且可以多开线程）
3、订单即包含非限购商品又含有限购商品（需要做全部校验）
根据消息队列里的订单数据，从缓存读取活动商品信息，检测订单库存数量，检测是否限购，检测用户购买数是否超限。如果成功，插入（更新）活动商品销量缓存，插入（更新）用户购买数量缓存，写入（更新）订单消息处理结果，写入订单到订单数据库（记录信息）
消息处理需要尽可能的快，要不下单反馈客户需要等待时间太久(消息队列里我们只校验库存和限购，其他合法性校验提前到下单接口层校验，减轻消息处理复杂度)
使用自动或者通过后台管理系统将审核完毕的活动（商品）数据推送到缓存系统（这需要业务的支持，业务一旦审核完毕推送到缓存了，原则上我们不允许做任何修改，因此我们可以从业务治理上加以控制：比如：一审，二审，终审，上线）
下单前提前将活动商品加入到缓存（整理后）
下单消息队列里的订单数据，从缓存读取活动商品信息，检测订单库存数量，检测是否限购，检测用户购买数是否超限。如果成功，插入（更新）活动商品销量缓存，插入（更新）用户购买数量缓存，写入（更新）订单消息处理结果，写入订单到订单数据库（记录信息）
消息处理需要尽可能的快，要不下单反馈客户需要等待时间太久(消息队列里我们只校验库存和限购，其他合法性校验提前到下单接口层校验，减轻消息处理复杂度)
前置做库存数据校验（粗放型）：
1.计算好购买商品是是否有限购商品（控制数量总）。如果没有限购示记没有限购，让后续的消息处理程序更加快速
2.粗放的校验下活动商品为限购商品的话，从缓存里校验下是否超过限额，当库存校验成功后
3.粗放校验下用户的限购缓存限购数量
立即将订单信息下方消息队列，成功返回一个Token，后续需要使用此Token
通过查询接口查询订单处理进度；失败直接返回错误
需要注意的是，客户端提交订单后，获取到Token需要之后，需要立即先通过查询接口查询下订单处理消息进度，订单处理查询接口会返回状态：正在排队中（会返回当前排队序号）--> 订单已创处理成功(会返回订单编号)/订单处理失败（返回友好的错误提示）。如果返回排队中，那么客户端需要在等待接口每隔一段时间再次请求下述接口，排队成功候，立即跳转到支付页面进行支付
下单，查询订单处理进度，支付子系统接入，以及进行流量调控限制
限流，过载保护（大促期间，针对某些接口做限流）
接入到API网关的系统，都需要申请APPKEY(数据签名密钥，同关对接口授权，以及身份校验，校验签名，以及调用频率监控和接口异常监控
接口模块可以拆分成不同站点以及支持版本迭代和A/B测试
MongoDB接口监控日志
1.活动商品销售数量缓存Key：O2O:ProductSale:{活动ID}:{商品ID}=100
当订单判断成功，进入到消息队列里的同时，更新此销售数量（供前台显示），考虑到可能会出现并发，限购的数量会稍微大于限购数量，所以在前台显示的时候，如果此数量大于限购数量，我们就直接读取商品限购数量展示在前端
2.用户限购数量缓存KEY：O2O:UserLimit:{活动ID}:{商品ID}:{用户ID}=2
后台线程将用户有反记录记录到此缓存（非限制全负责实际数量的不记录），后台线程处理订单时候，需要根据此判断用户是否超过购买数
3.订单处理消息缓存记录：O2O:Order:{Token}={{Token}，{OrderId}，{Status}，{CreateTime}}
将处理完成的订单消息会写到此缓存，前台通过直接查询KEY是否存在和处理结果来判断下单是否成功
4.活动商品以及商品评情缓存，Key: O2O:Product:{活动ID}:{ProductId} = {{活动信息}，{商品信息}}