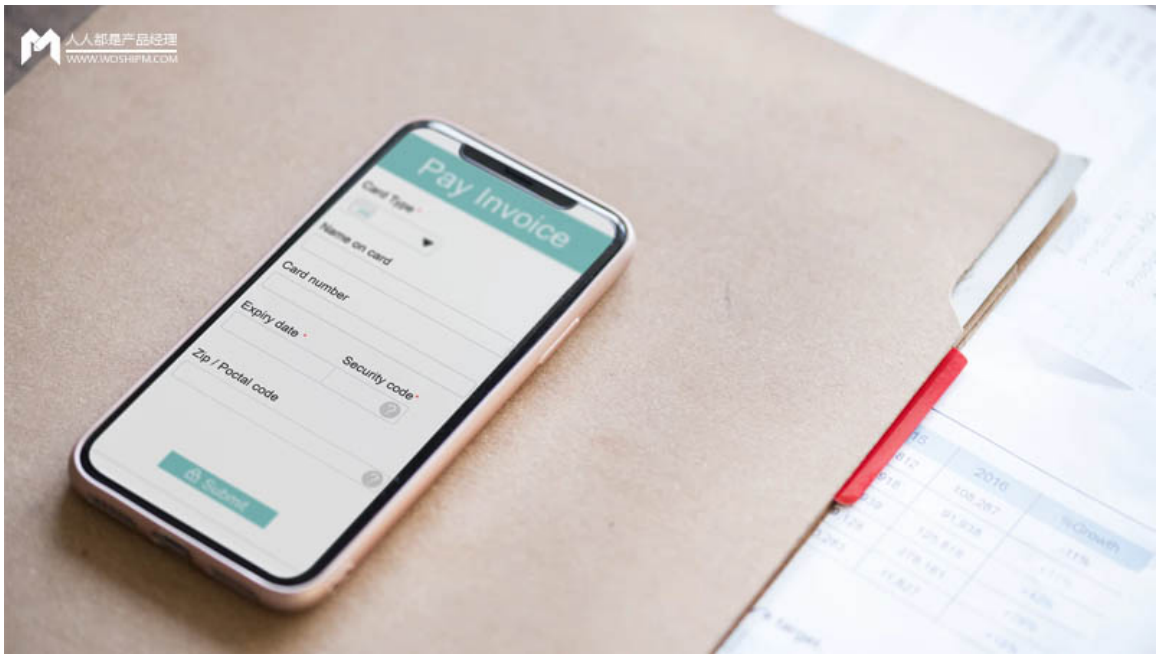
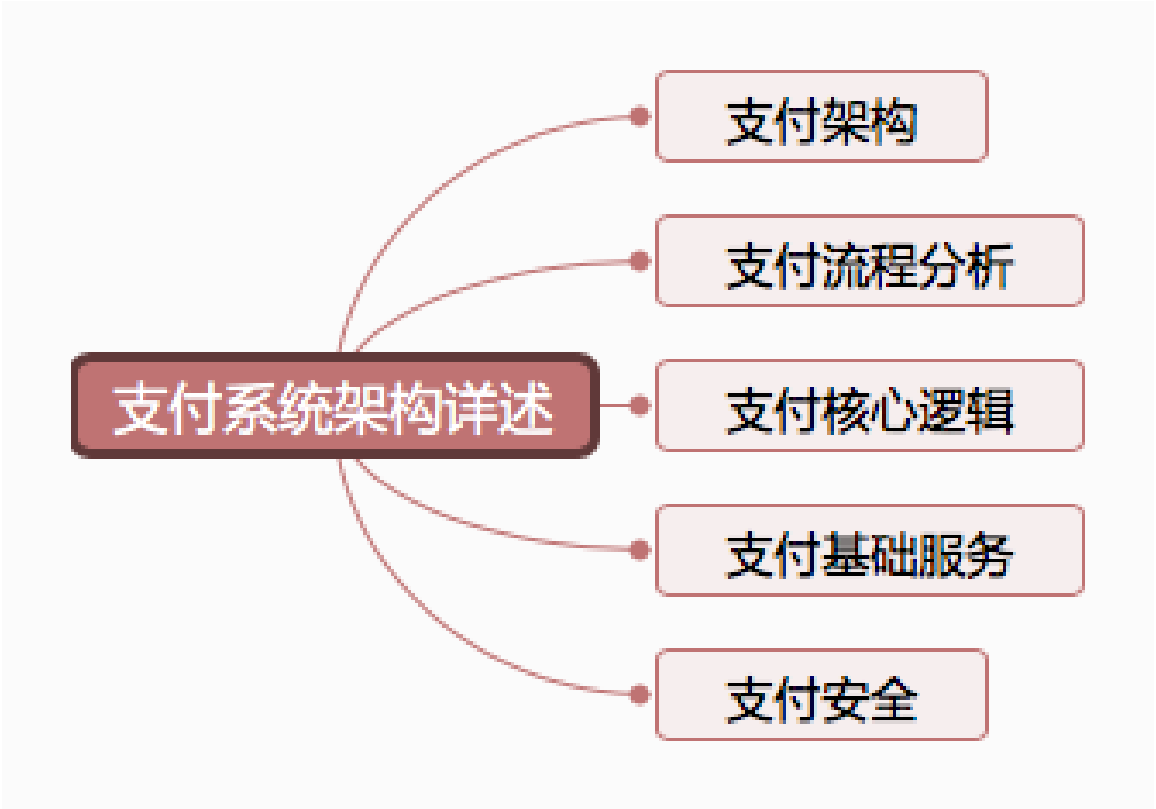
文章主要是从支付架构、支付流程分析、支付核心逻辑、支付基础服务、支付安全五个方面来详细讲述支付系统架构，一起来看看~

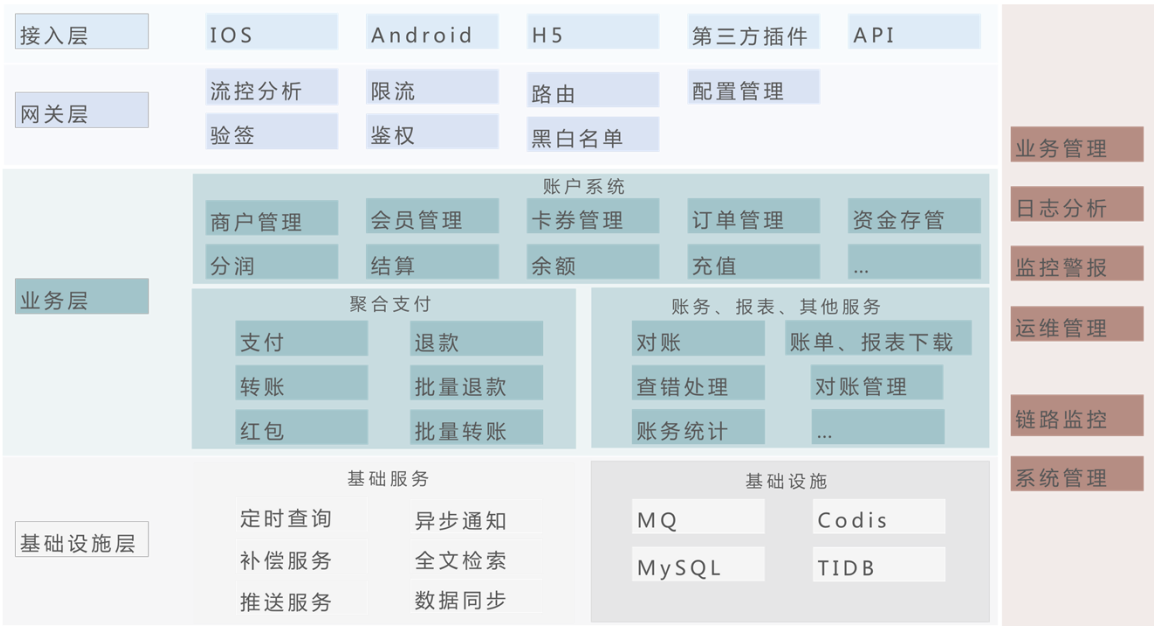


流程图：

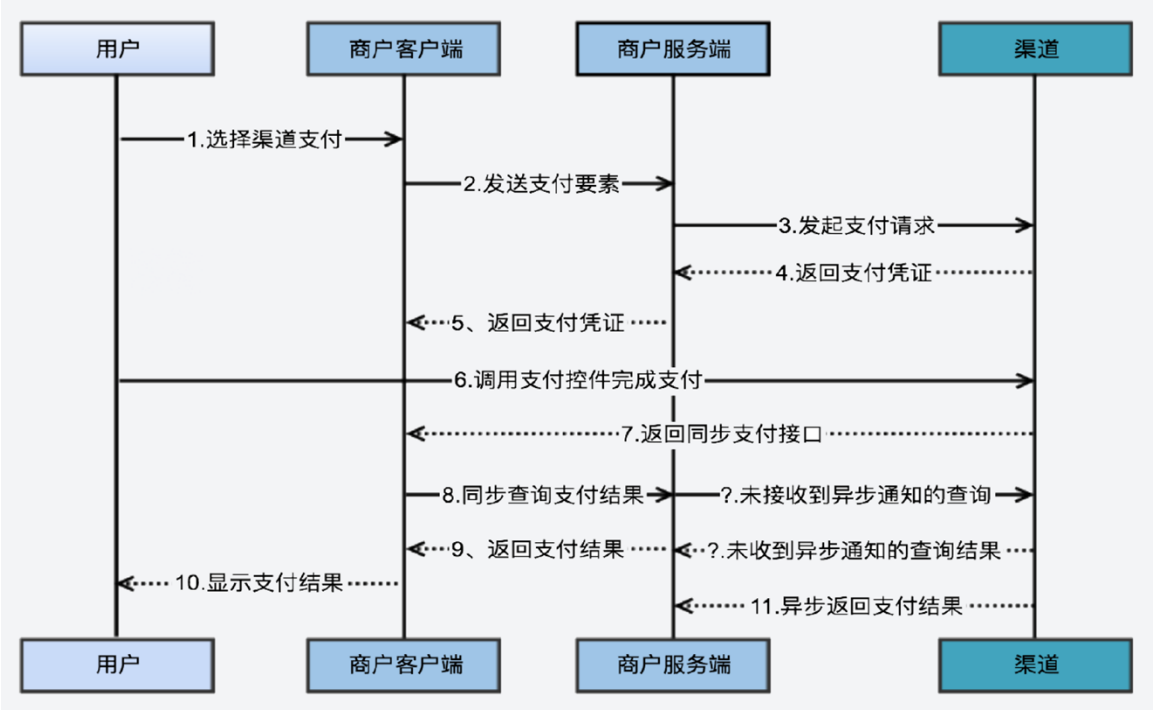


架构的定义：架构一定是基于业务功能来展开的，主要是制定技术规范、框架，指导系统落地，好的架构是需要不断演变和进化而来的。 架构需要关注的基础核心点主要是：安全、稳定、可扩展。 构建架构时需要关注的点：目标客户是谁、主要场景有哪些、流程是怎样的、模型、职责有哪些、边界在哪里以及设计。其中比较难以理解的点是困难及模型这两块。 架构与业务需求的关系：架构的产生来自于业务需求，业务需求进一步抽象形成架构，架构指导后续研发，研发最终成果解决业务需求的问题。整体是一个正向循环的关系。

一、支付架构



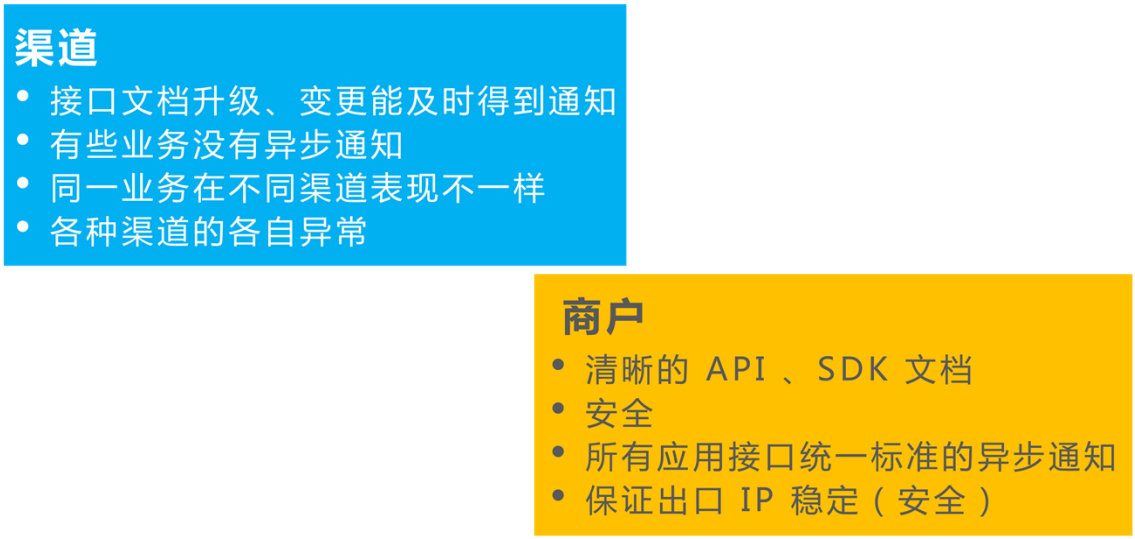
二、支付流程分析



第一步，用户选择支付渠道，进入商户客户端； 第二步，商户客户端发送支付要素，到商户服务端； 第三步，商户服务端发起支付请求到渠道侧（个别渠道如支付宝是不需要此步骤）； 第四步渠道返回支付凭证到商户服务端； 第五步商户服务端返回支付凭证到商户客户端； 第六步，用户调用支付宝控件完成支付。

接下来是重点，第七步一般渠道是采用异步通知方法来通知商户，但是有些企业是在第六步支付完成之后，在 C 端会同步通知支付成功。如果以此结果来判断支付是否成功，其实是不严谨会出问题的，应当调用渠道的支付接口来进行核查，然后以渠道返回的结果为准。

在日常工作中，许多企业在选择第四方服务商或者渠道的时候，会着重关注「并发」这个点，认为并发量需要达到上万级才可以满足日常需求，但实际上这个量级非常大，其实并没有必要的。



若直接对接渠道可能会遇到的问题：

接口文档升级、变更能及时得到通知； 有些业务没有异步通知； 同一业务在不同渠道表现不一样； 各种渠道的各自异常。

商户的要求：

清晰的 API 、SDK 文档； 安全； 所有应用接口统一标准的异步通知； 保证出口 IP 稳定（安全）。

在系统架构设计时需要注意的一些要点：

提供规范的 API、SDK； 安全（通讯安全、数据安全）； 稳定； 异步通知统一； 各渠道的异常； 及时了解渠道接口调整。

API 接口设计规范

> 接口开发前需要先出接口文档，涉及到的内部接口先沟通好配置到 Mo
> 我们对外的 API 都是通过 HTTPS 来访问，支持 `https://api.ping`
> 认证。
> 支持的请求方法包括：HEAD（主要是心跳检测）、GET、POST、PU

1 请求

1.1 请求地址

- 直观能知道改接口功能，使用名词，注意是复数形式
- URL 结尾需要支持包含或不包含 /

1.2 请求头

- Authorization：Bearer sk_live***（HTTP Basic Auth 进行认证）
- Content-Type：支持 JSON（application/json）、Form（applic
    - 聚合支付接口支持上面两中类型
    - 账户系统仅支持 JSON 类型
- Pingplusplus-Request-Timestamp：请求时间（签名使用）
- Pingplusplus-Signature：签名串（目前使用RSA 1024位密钥签

1.3 请求 Body

- JSON 格式

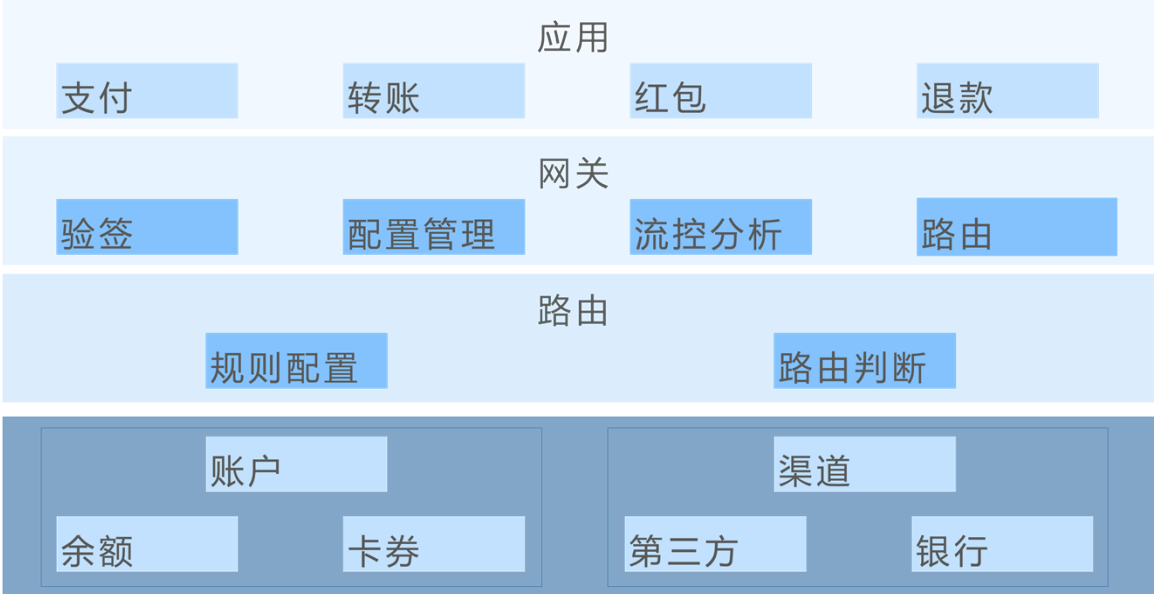
2 响应

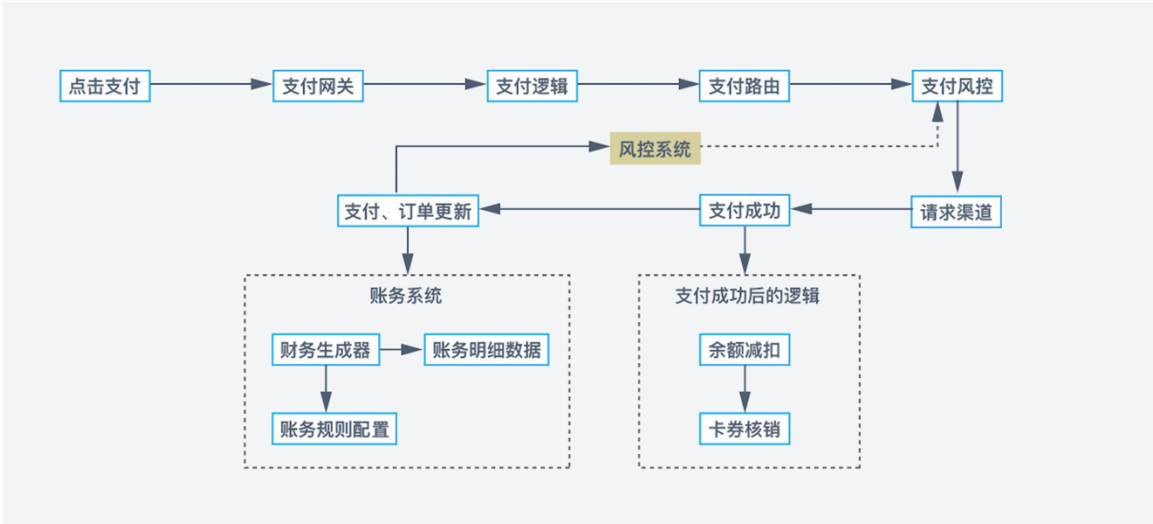
> 外部接口请参考 API 文档，请点击

- 对外 API 接口返回字段说明
    - 非 200 状态码的返回

以上为示例

三、支付核心逻辑

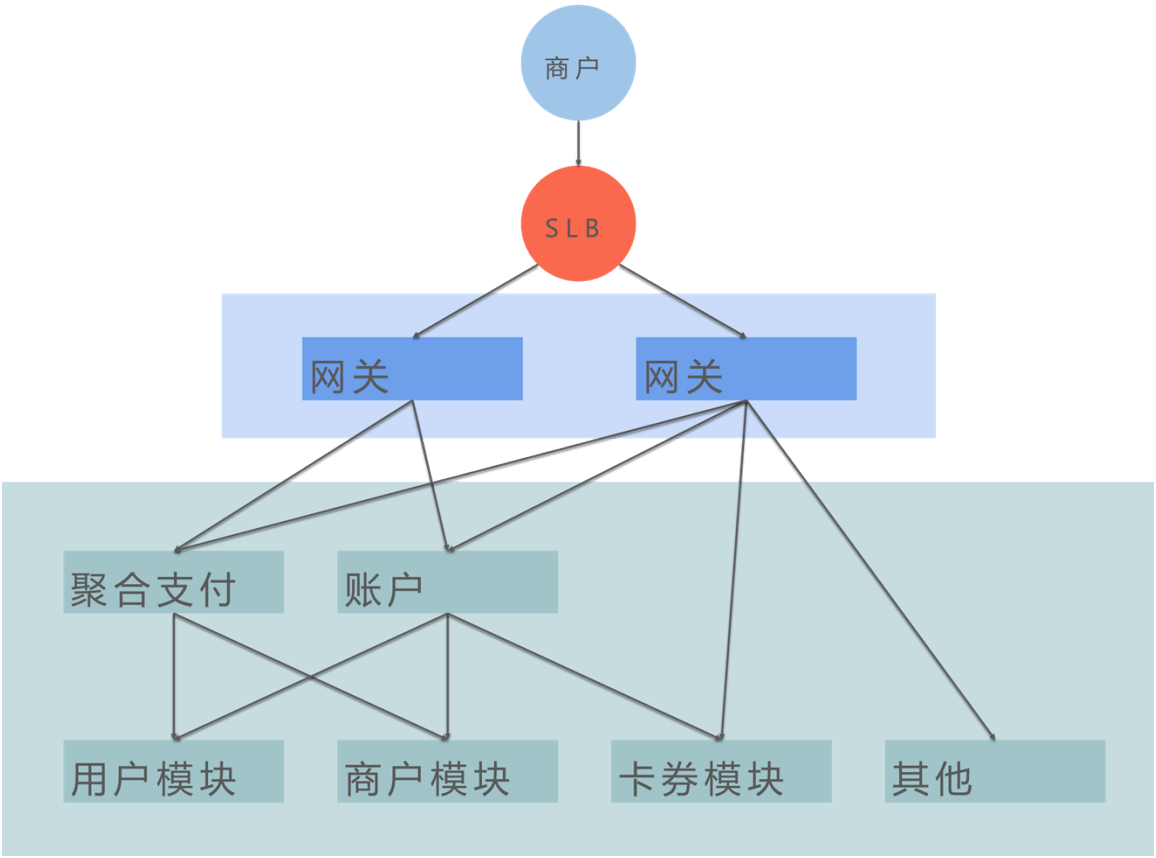




这里讲一下，支付成功之后，我们会把订单信息同步到财务系统，在账务系统里我们设计了诸如转账、汇款等功能，在前期设计时会设计好账务的生成规则，例如；一笔支付的请求会生成多笔账务，对其字段进行区分，这样方便管理和维护。

支付网关

此处特指 API 网关，支付网关的功能：



限流最好不要放到业务流程中做，会影响用户的体验。

支付网关的内容：

唯一的请求入口； 统一的身份认证、签名、加解密、流控； API 调用计费； API 的监控、报警分析； API 发布管理； 熔断； API 聚合； 协议转换。

上述内容除了必要意外，其他不放在业务层做，也是为了更好的用户体验。

支付逻辑

主要是根据请求的参数进行静态检验和业务逻辑校验，避免系统异常。

适配渠道的参数校验：长度、类型、格式； 订单的支付状态：是否支付； 订单的有效期等等。

要点：

一般商户是不需要做支付路由，大部分都是指定了最终的某个支付渠道。

但也有些没有指定了某个最终的渠道，比如银行卡的支付可以选择哪个第三方支付来完成支付，还有微信线上线下的封装，这个时候就涉及到支付路由规则配置。

费率：单笔费率、总额费率、阶梯费率； 营销活动：固定时间单笔优惠、单笔满减、单笔这款、直接补贴； 额度限制：单笔额度、时间范围内总额度； 服务指标：失败率、平均响应时间、异常率、TPS； 特殊配置：特殊要求（比如某渠道能快速结算）。

支付网关的目的——省钱。

支付风控

要点：梳理清楚业务风险，分析风险原因，制定风险防范规则。

拒绝、放行、增强认证

灵活稳定、高性能、准确性

（1）数据来源

内部数据：

用（商）户信息 交易数据 账户数据 黑名单 设备、位置信息 日志数据

外部数据：

第三方购买 央行征信 芝麻信用 合作数据

（2）风控流程

事前：

入网审核 风险评估 单笔限额设置 单日限额设置 频次设置

事中：

实时分析 多维度判断 拒绝 拦截 – 进一步验证– 人工介入 延迟操作（例如用户大额提现，需要时间段进行复核）

事后：

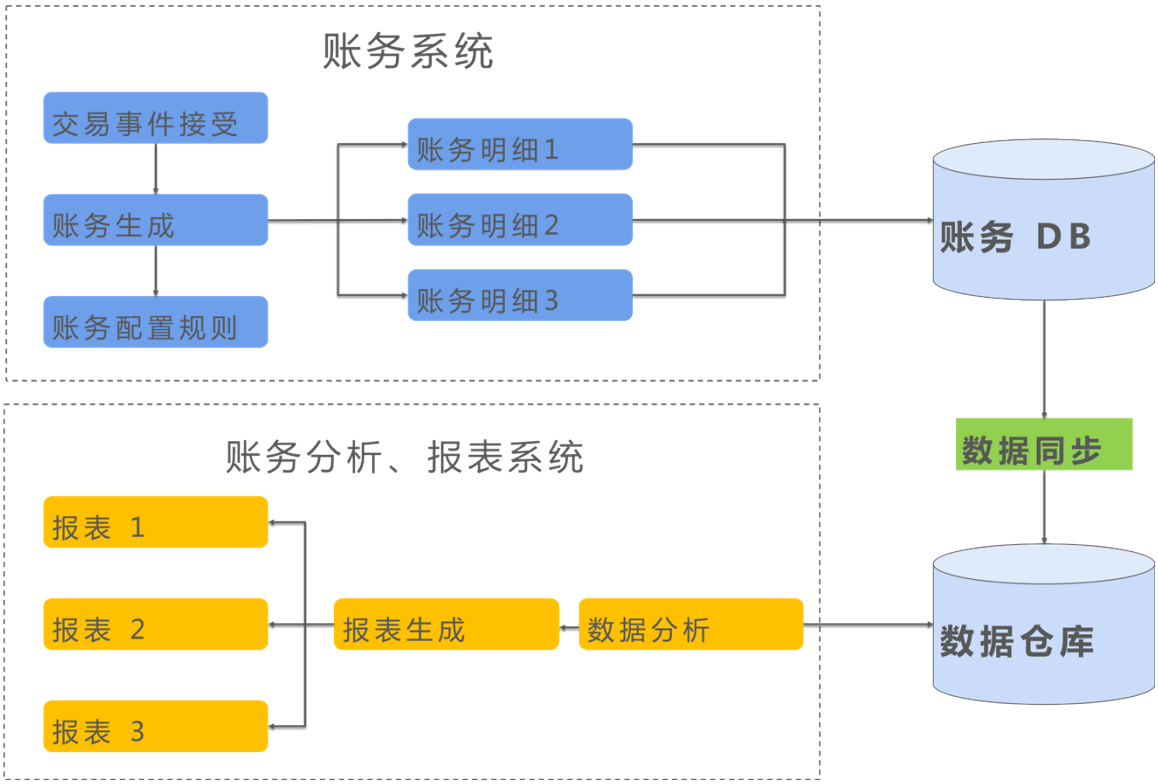
数据分析 巡查、警告 降低评级 升级防范措施 逻辑完善 反馈至事前、事中规则中

账务系统

账务生成 内部对账 原始账单下载 生成标准账单 对账 差错处理

账务生成后首先进行内部对账，一直后进行原始账单下载，再生成标准账单，进行对账之后进行差错处理。

内部流程如图：



订阅交易信息； 根据交易事件查询生成账务的规则。

交易事件：支付、退款、转账等等。

根据规则生成账务明细； 将账务明细落地。

对账流程（实现方式之一）

内部对账：

保证账务和交易信息配对 一条交易信息有多条账务信息

渠道账单下载：

下载； 账单标准化（对账字段统一）； 落地标准化账单。

对账：

账务和标准账单对账； 双向对账； 差错处理。

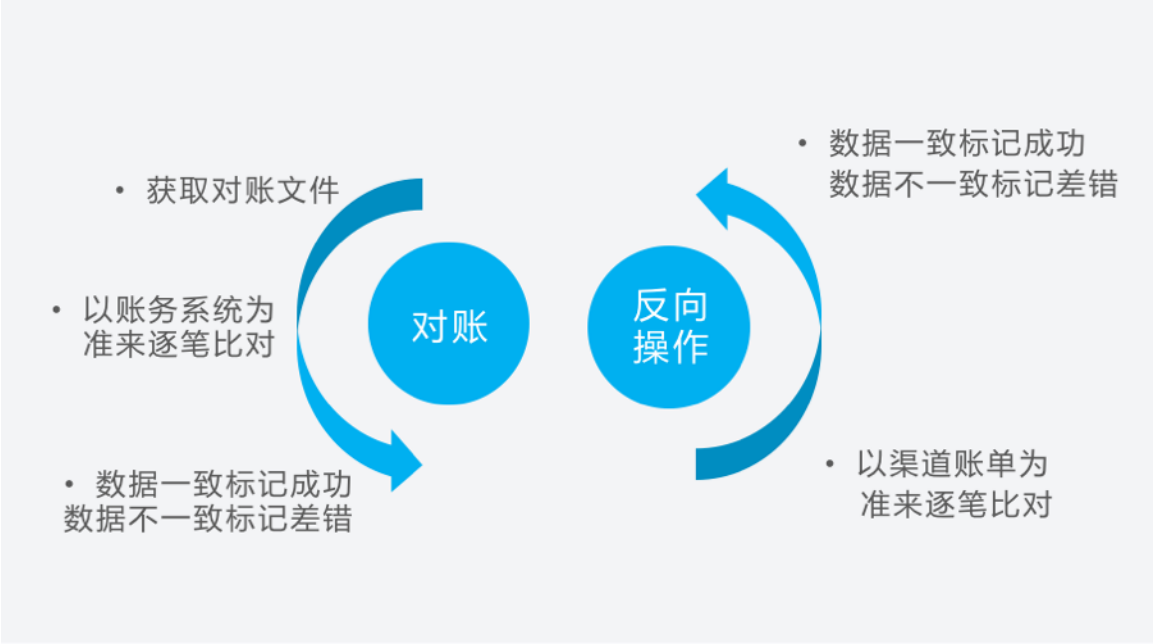
账单下载：

下载方式	根据提供的方式，一般 HTTP
文件格式	CSV、TXT、ZIP 等
下载时间	有些渠道工作日才生成账单，需要一一配置
异常处理	立即重试、递增时间重试

这里提一句，在做异常处理这部分工作时，有的研发朋友写代码时遇到过类似的问题，例如：订单在周末下单，但账单周一才能查询；等到周一时发现查询不到，选择立即重试 + X 分钟后重试就结束了。

这样是不行的，因为银行有的是 8 点之后可以查询到，有的是 9 点之后，所以要选择递增时间重试，然后标记人工处理。

对账：



对账部分：

获取核对文件； 以账务系统为准来逐笔比对（以某个字段为准进行比对）； 数据一致标记成功，数据不一致标记差错。

反向操作：

以渠道账单为准来逐笔比对； 数据一致标记成功，数据不一致标记差错。

差错处理

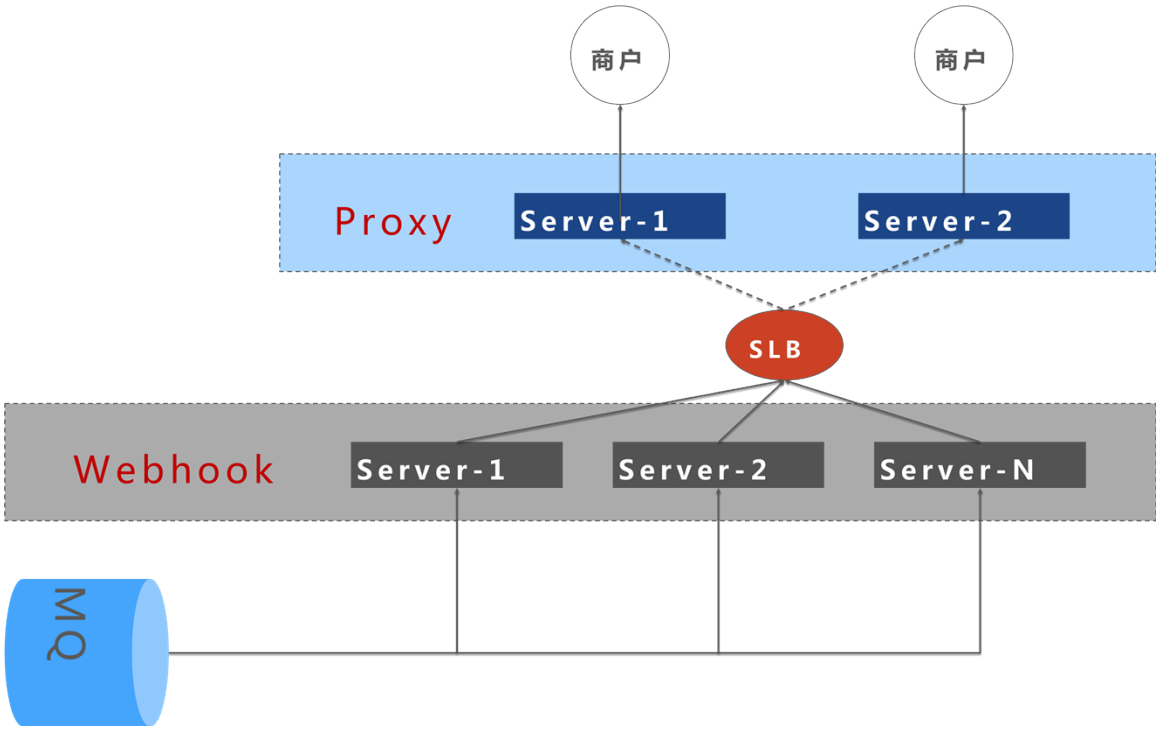
本地丢失：渠道账单的数据未在账务中查找到。 渠道丢失：账务中的数据未在渠道账单中查找到。 数据差错：账务与渠道某些对账字段未能对上。

此处需要注意的是，针对差错都需要向渠道查询每笔订单信息再次确认，同时，有些渠道的交易成功时间本来就是有错误的。一般来说是件不会差错很大，一般出现在跨日交易中，例如：当天交易无账单，先标记为差错，第二天再改正。

四、支付基础服务

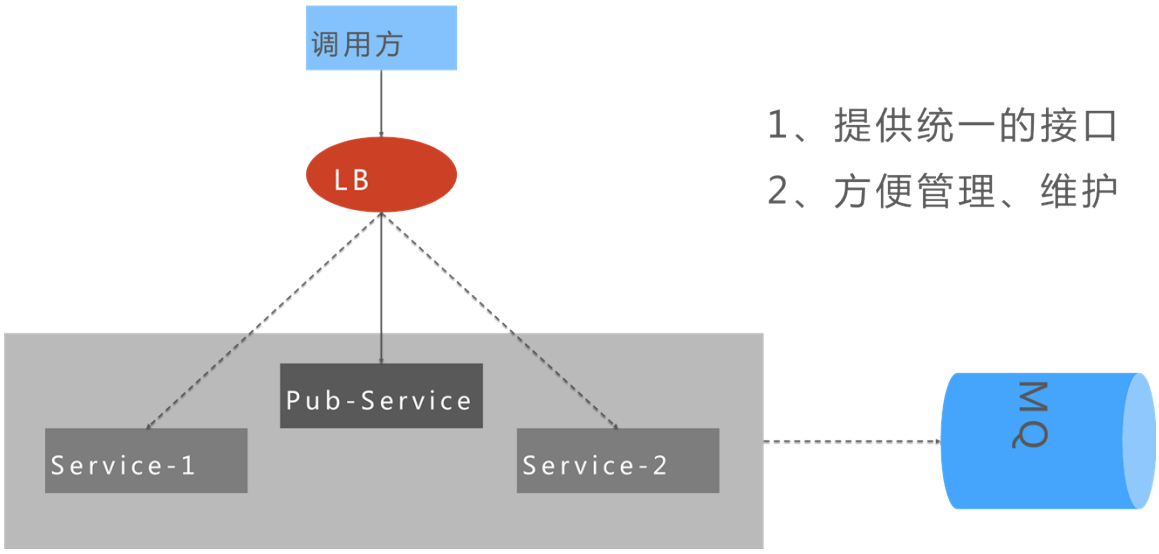
Webhook 公共推送服务 主动查询 补偿 链路监控

Webhook


商户
商户
Proxy
Server-1
Server-2
SLB
Webhook
Server-1
Server-2
Server-N
MQ


Webhook
liveMode和testMode各事件和模拟事件的发送
头信息校验
content-length
x-pingplusplus-signature
Content-Type: application/json
Accept: */*
Connection: keep-alive
Host
证书格式
pkcs #1
pkcs #8
签名验签
正常生成头信息的x-pingplusplus-signature字段值，且验签能通过
API和webhooks_new的交互
新老webhooks交互逻辑
webhooks重发机制
首次立即发送，重试通知时间间隔为"0s","5s","10s","2m","5m","10m","30m","1h","2h","6h","15h"
新webhook重试三次都失败后交给老webhook，时间间隔0s，3s，6s
DB数据校验
dashboard上webhooks补发
redis连接异常
数据量异常
各种类型的接收地址配置
http
https
域名中有中划线，下划线
调API event接口查询event对象

公共推送服务



主动查询

异步延时调用

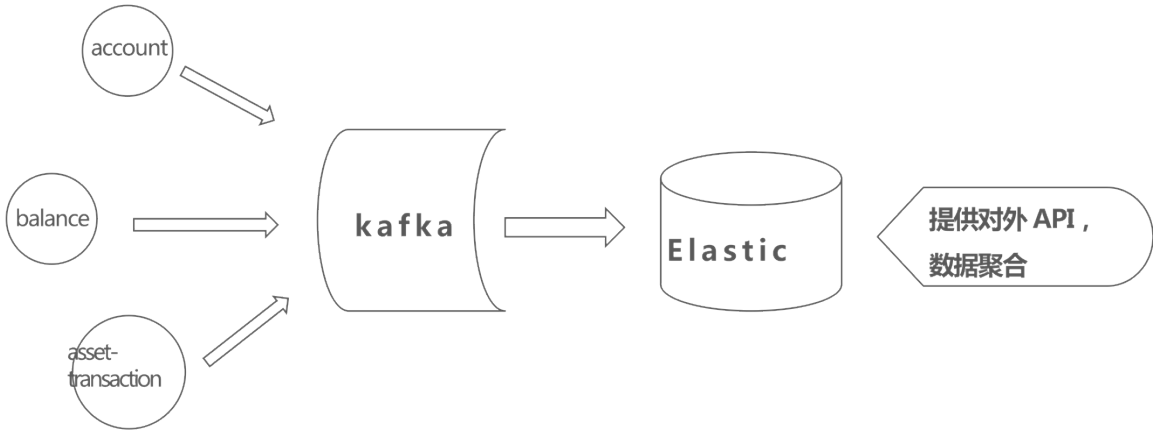
场景：

订单创建成功的时候会向服务推送主动查询信息，如果订单支付成功会通知服务取消后续的主动查询，否则在过期时间点向渠道主动查询订单是否支付目的是避免渠道异步通知服务的异常。 退款创建成功的时候会向服务推送主动查询信息，该服务会在一定的时间范围内多次查询渠道直到有明确的结果返回（有些渠道没有异步通知）。 转账也是类似的逻辑，但某些渠道只提供重试的功能，要注意幂等性。 ……

补偿：

协调保证各模块间数据的一致性； 一般会跟重试、回滚、兜底来协调使用； 使用条件：系统异常、业务异常； 补偿失败报警人工干预。

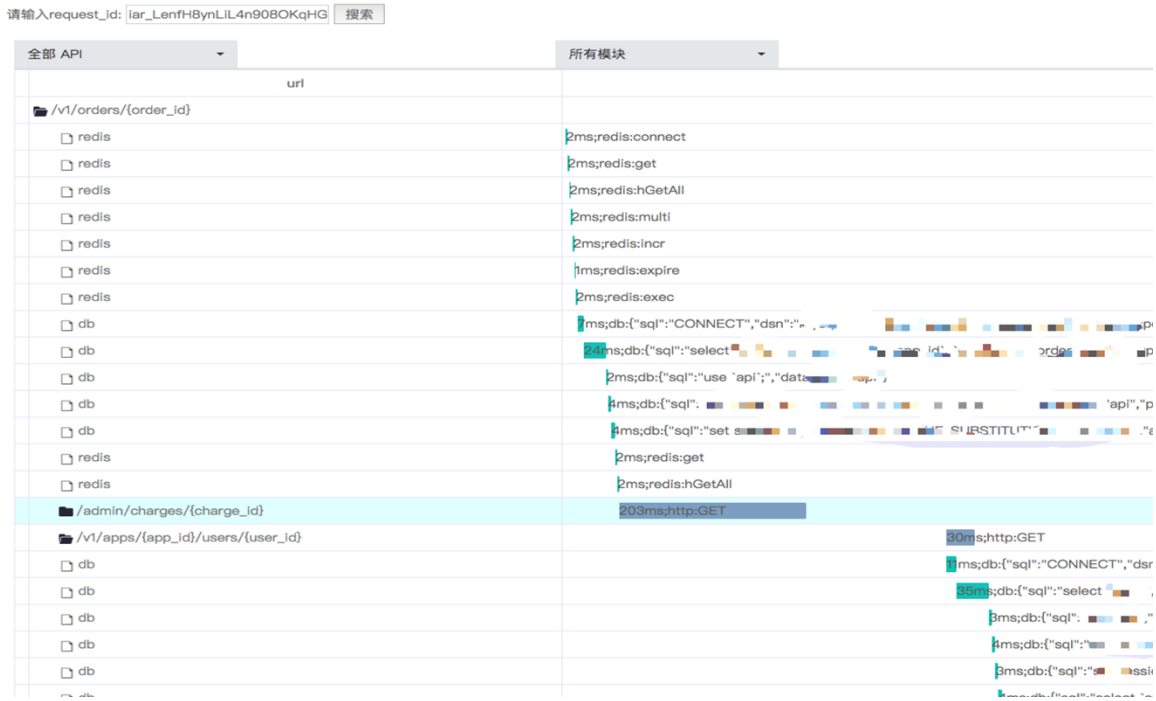
链路监控



展示信息：应用、URL、调用方、调用时间、调用次数、调用失败次数、本地平均耗时、总平均耗时、调用失败平均耗时 、错误率、依赖度。

关注：Cache、SQL、HTTP、TCP

基本信息



依赖度

api_payment|/admin/charges|post

五、支付安全

数据安全

防窃听、防越权防抵赖、防破坏、防篡改、防重放、防泄漏。

使用范围：网络、系统、应用、业务等。

数据安全要点

加密通讯（防窃听） 双向签名（防抵赖、防篡改） 敏感数据加密存储（防泄漏） 密钥管理（通过认证接口获取，只允许加载到内存，不允许直接写入配置文件） 权限控制（防越权-非法访问） 数据的完整性（放篡改- 数据被恶意修改、非法篡改）

其他

内部接口认证。 避免内部代码未经审核发布到托管平台！！！ 数据异常分析。 安全机构合作。

注意点

使用 HTTPS 加密传输； 传输的数据使用签名； 提交的数据是符合规则并且是不存在或者是未支付的； 支付成功以服务端异步通知为准。