设计文档及工程代码规范

作者：运维开发团队 许俊彦



2018-3-7

招联消费金融有限公司

目录

[目的 2](#_Toc519602451)

[一、概要设计文档 2](#_Toc519602452)

[二、详细设计文档 8](#_Toc519602453)

[三、工程代码设计 13](#_Toc519602454)

## 目的

提高团队内设计文档质量，工程代码质量，提升系统设计效率，提升沟通效率。设计文档在具体项目阶段分为两类：项目初期使用概要设计，项目进入开发使用详细设计，此规范以例子形式对两个阶段所提供的文档进行明确要求。

## 一、概要设计文档

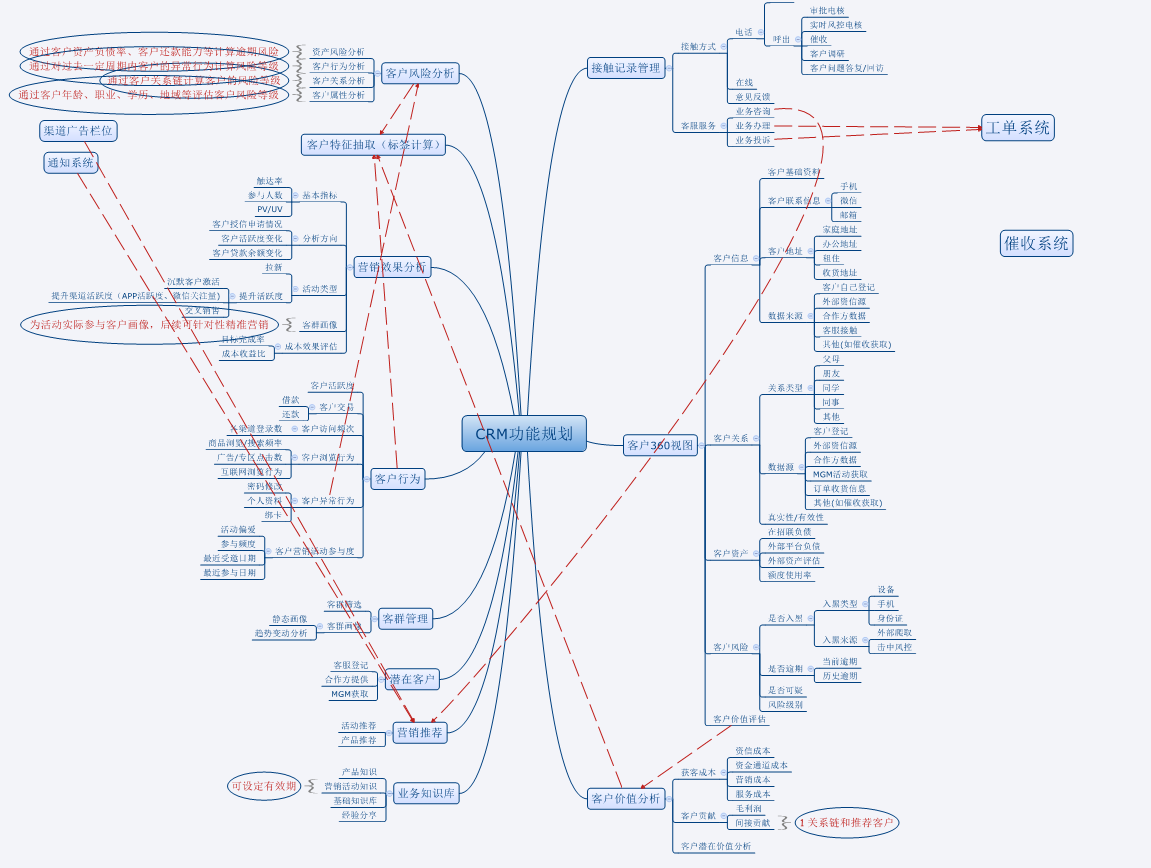
用于项目前期沟通系统或者需求功能的设计，需要包含：

1. **整体设计思路**
2. **系统架构设计**
3. **模块设计**
4. **数据设计（ER或业务对象）**
5. **主要流程设计**

***参考例子***

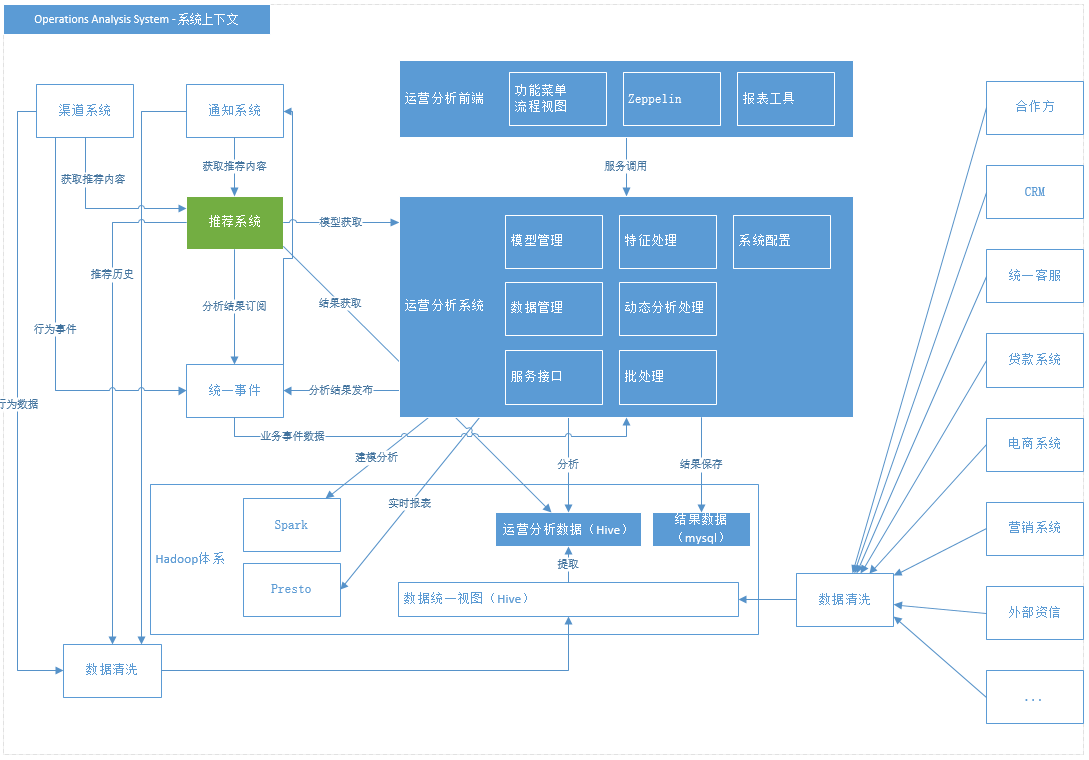
1. ***整体设计思路***

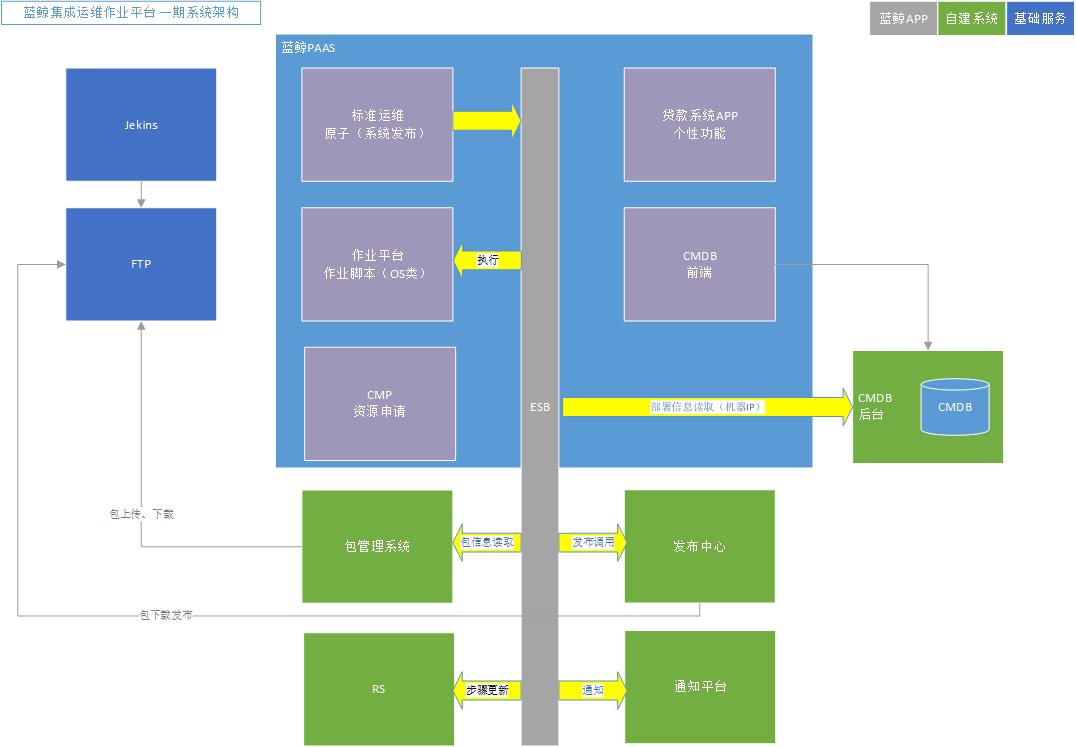
*表达对于整个设计的思路，可通过思维导图整体概况本次设计包含的点，作为后续所有设计的指导*



1. ***系统架构设计***

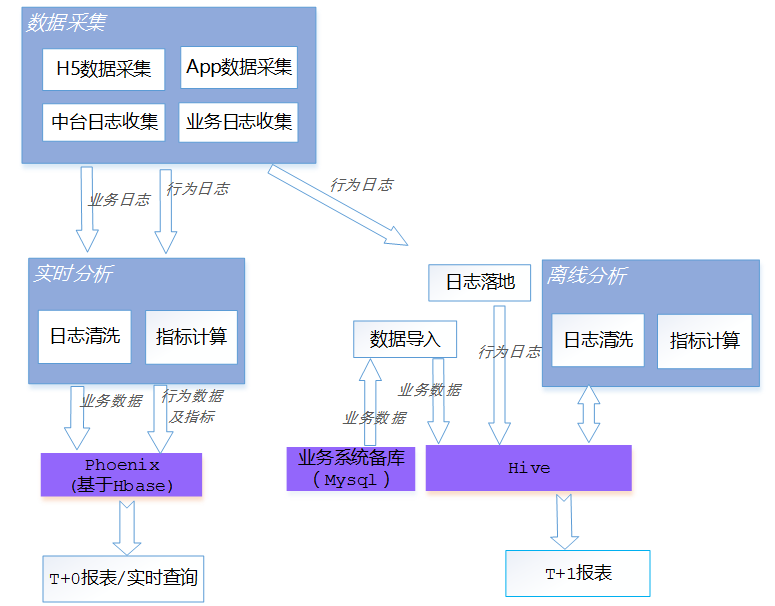
*重点描述各个系统之间的依赖关系，被设计的系统的上下文，对于被设计的系统需细化并突出描述*

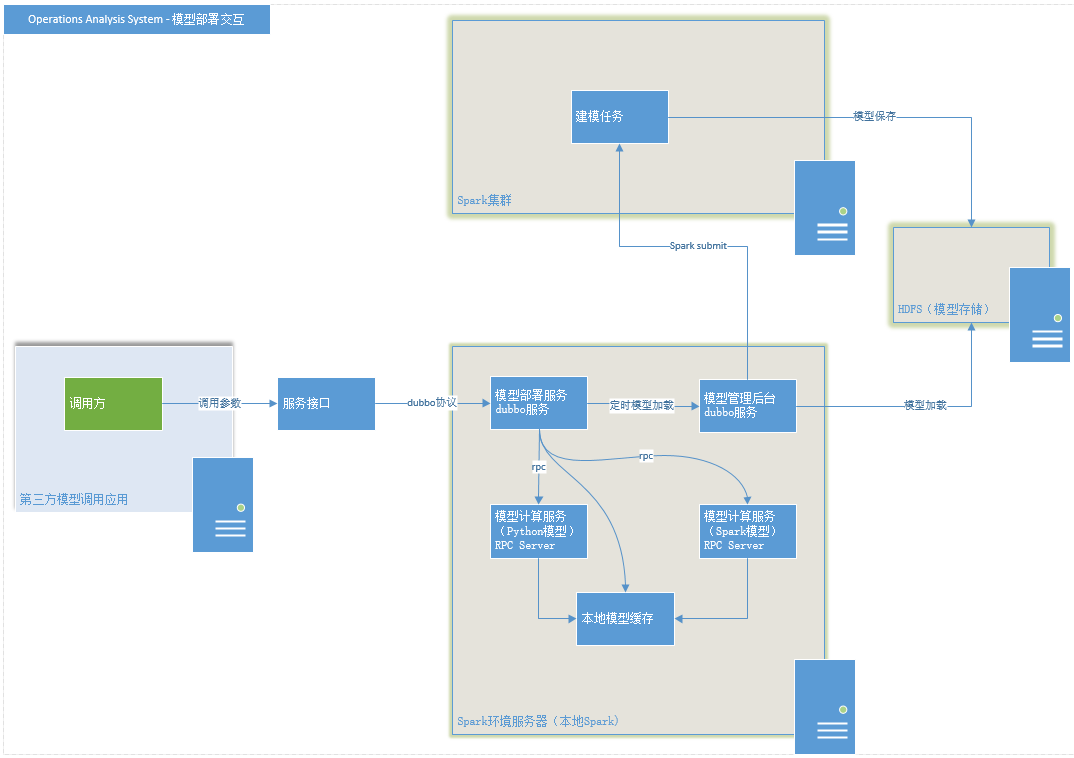




1. ***模块设计***

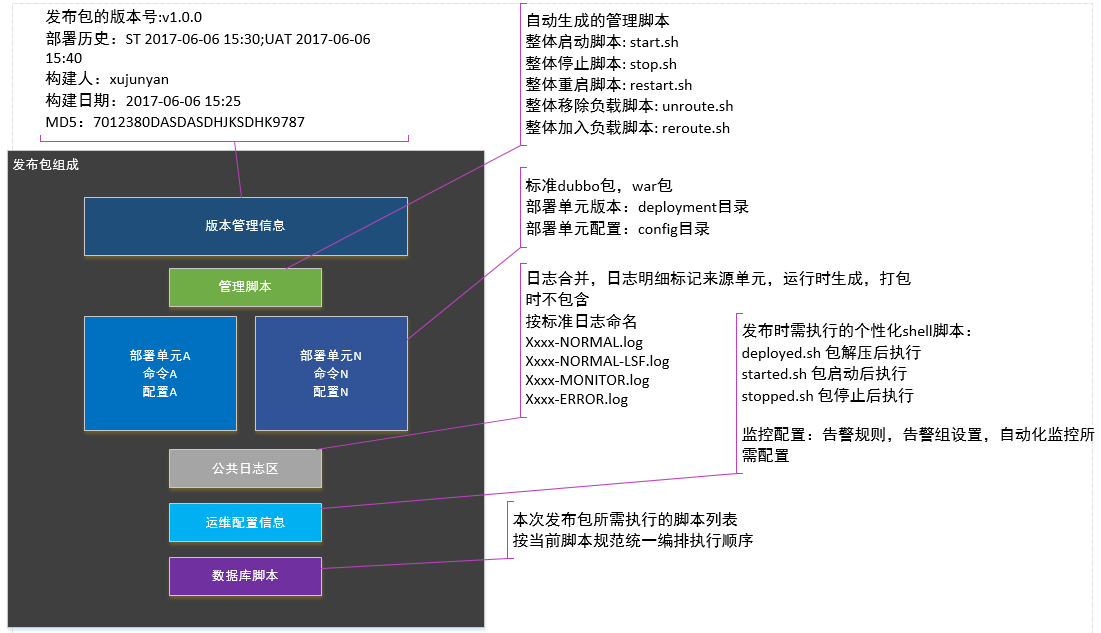
*重点描述被设计的应用的各个模块，各模块功能描述、相互间接口及访问协议、请求响应/数据流转方向*

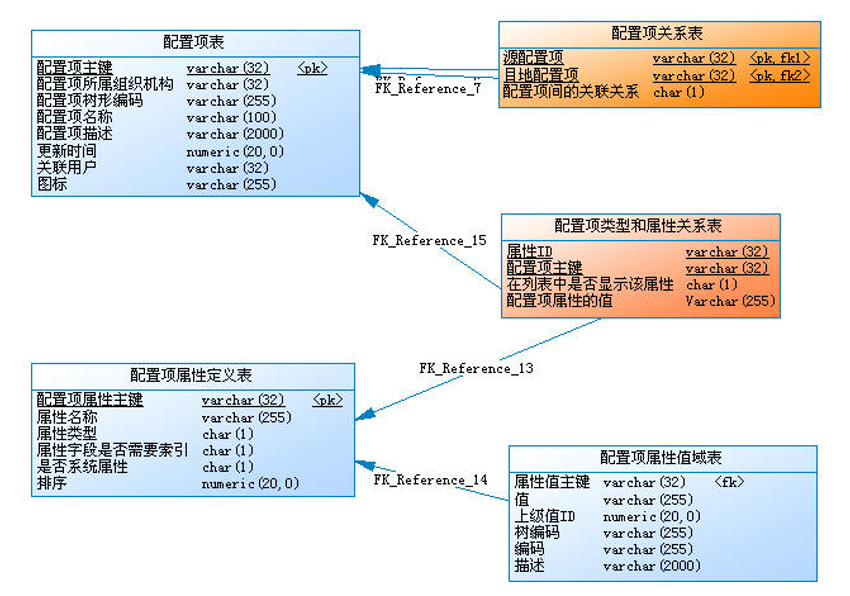




1. ***数据设计***

*重点描述系统用到的主要数据、表或业务对象字段，覆盖各个业务场景下业务对象的定义。可以使用PDM，VISIO ERD等文档表达，在详细设计阶段也需继续更新*

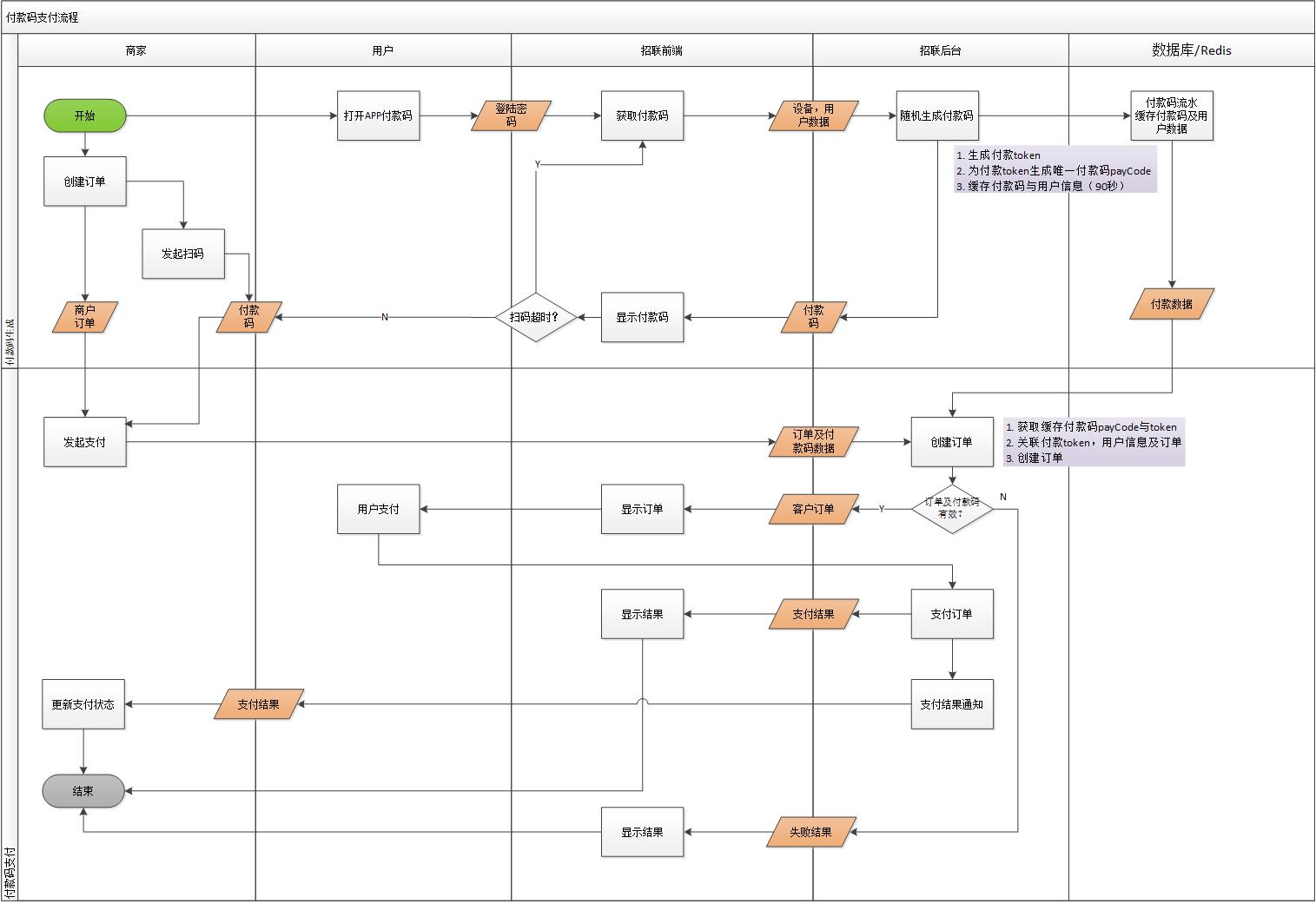


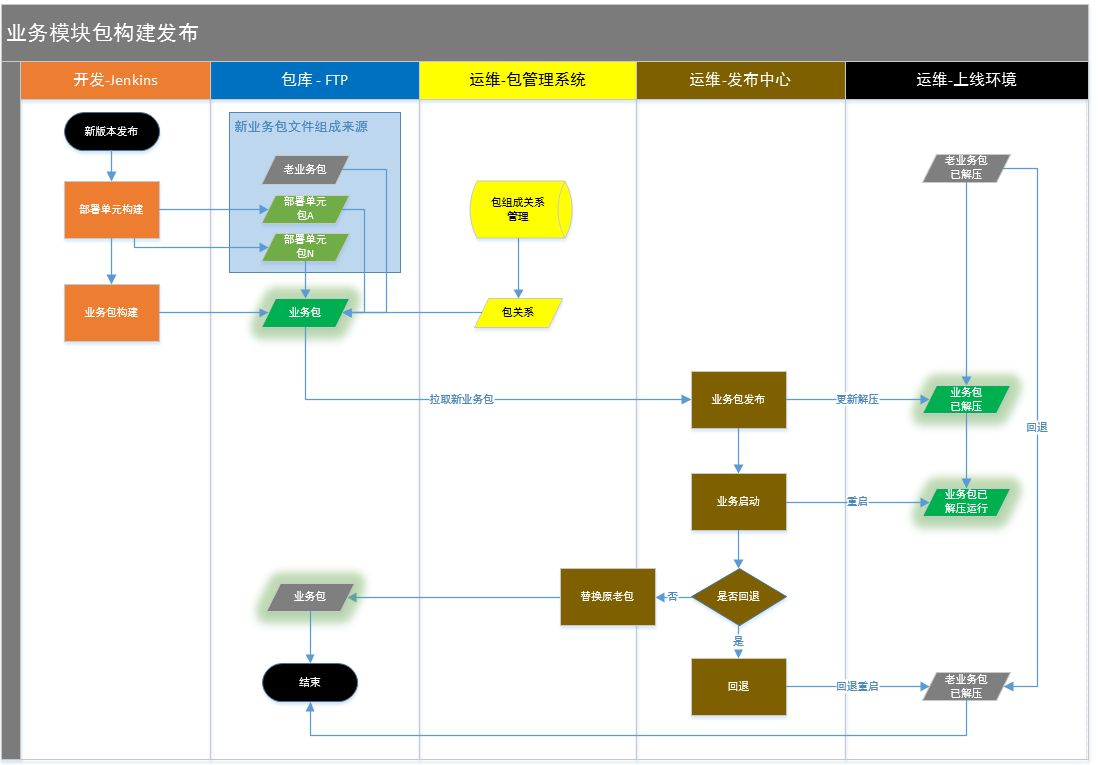


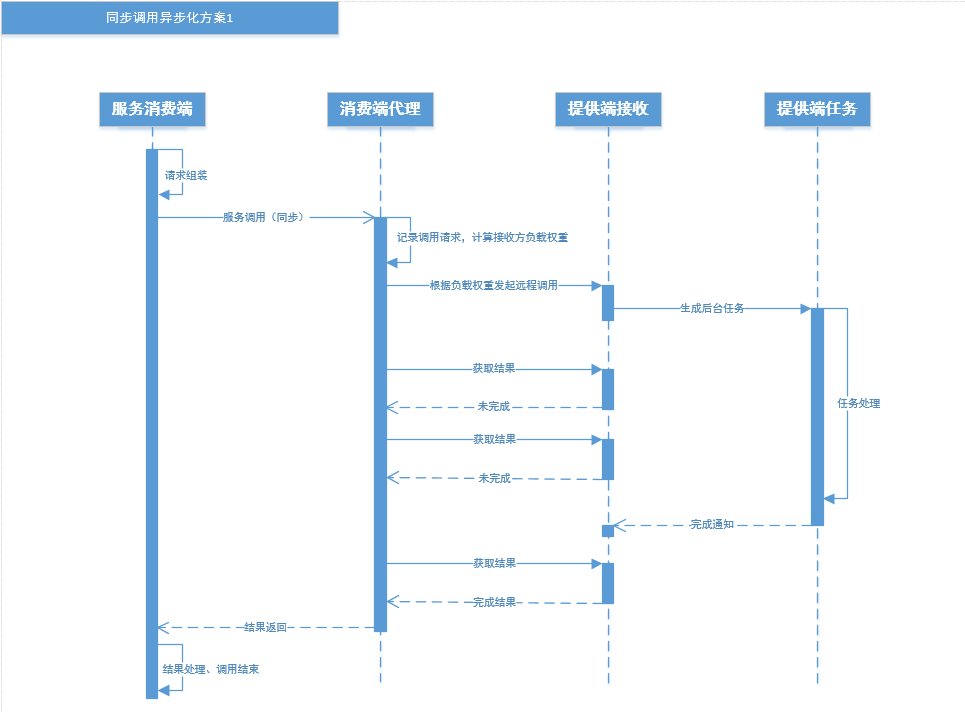


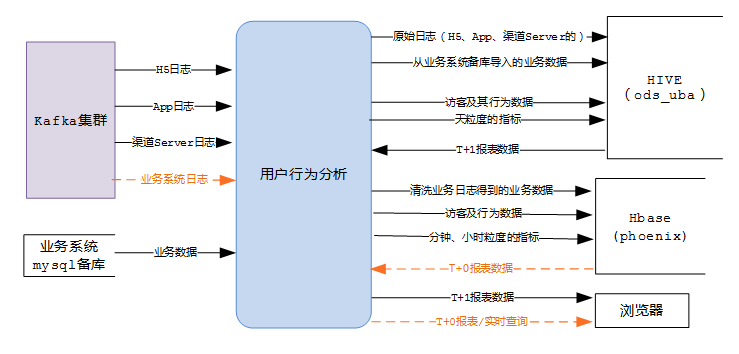
1. ***主要流程设计***

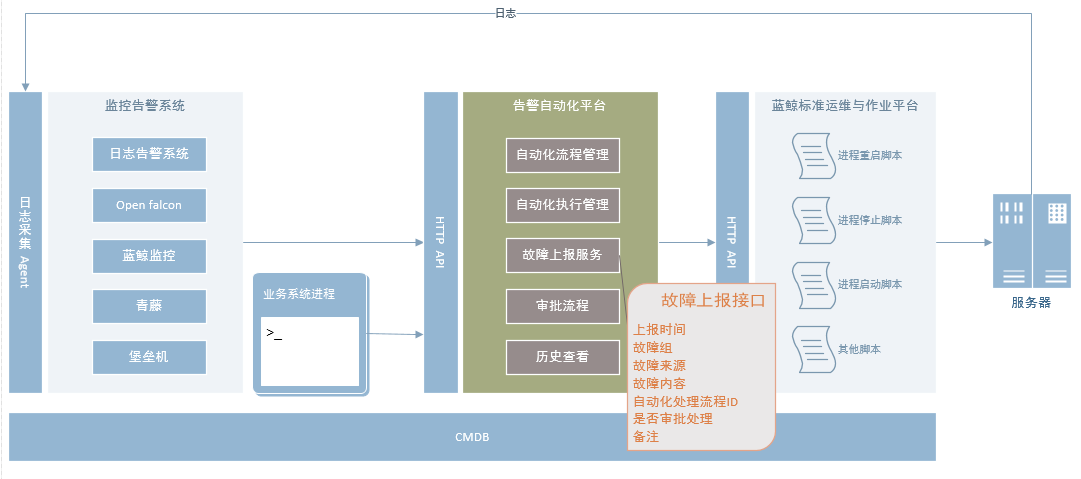
*重点描述系统的关键流程，数据流。使用流程图，时序图。流程节点粒度会比详细设计粗一点，具体流程在详细设计时需补充完整*









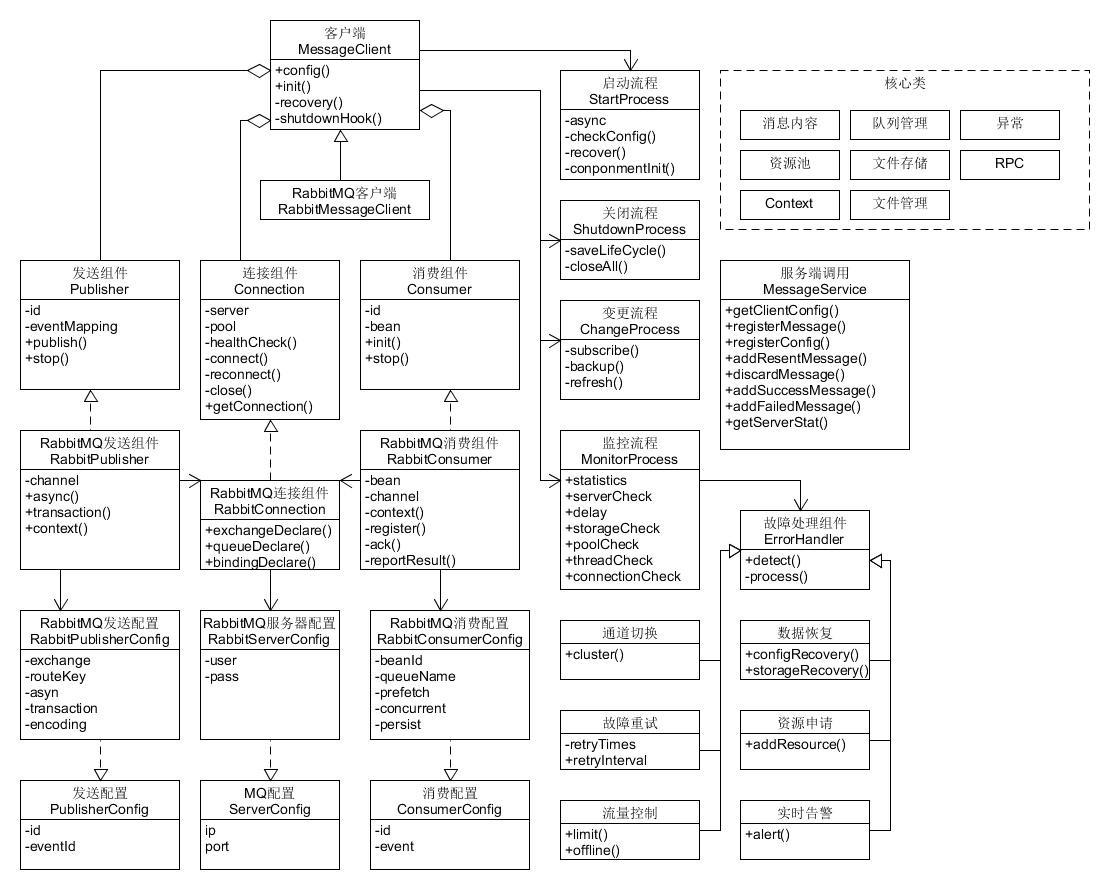


## 二、详细设计文档

用于正式启动开发，运维上线的设计文档，需要描述清楚各个功能设计细节，细化到功能代码层面，主要有：

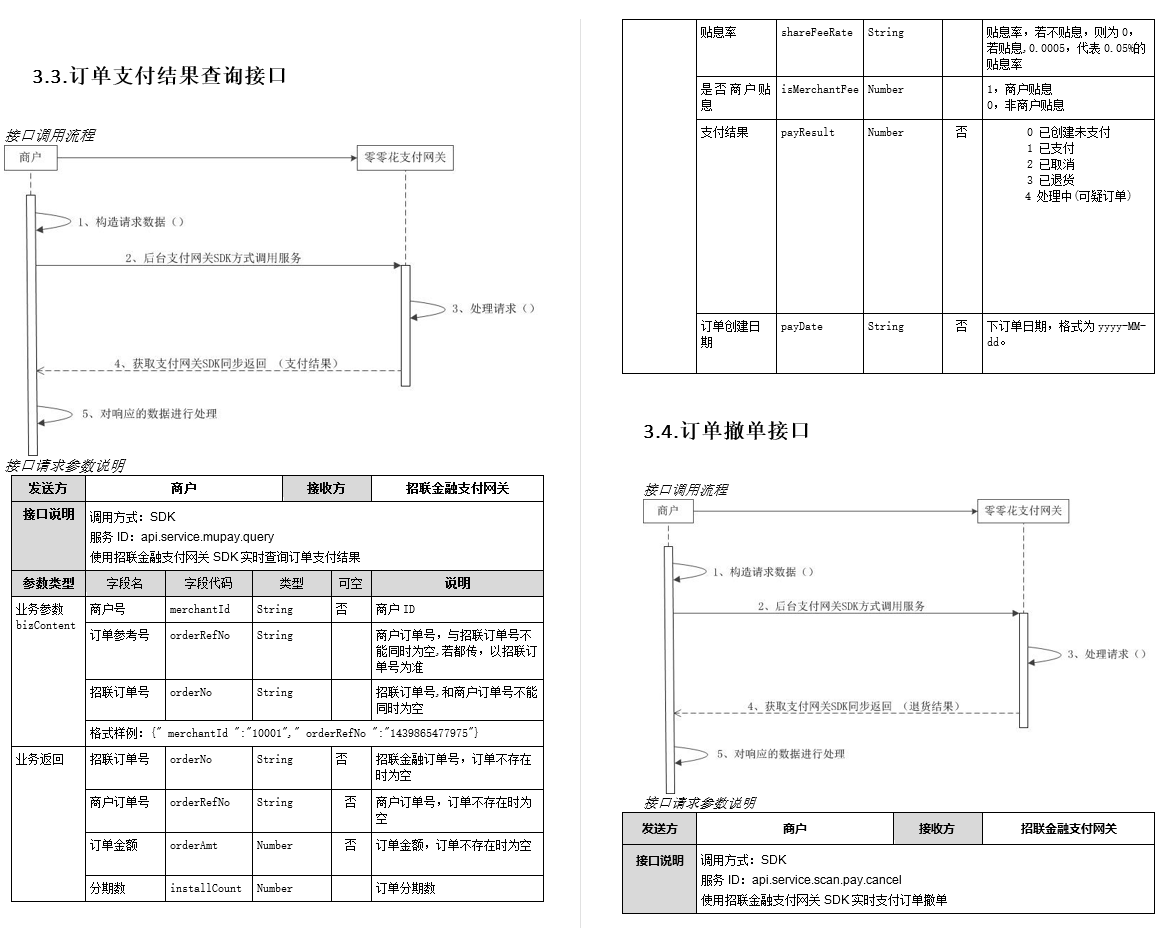
1. **类概要或详细设计**
2. **接口设计**
3. **代码流程设计**
4. **界面设计**
5. **部署架构设计**
6. ***类概要或详细设计***

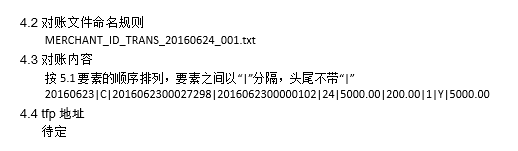
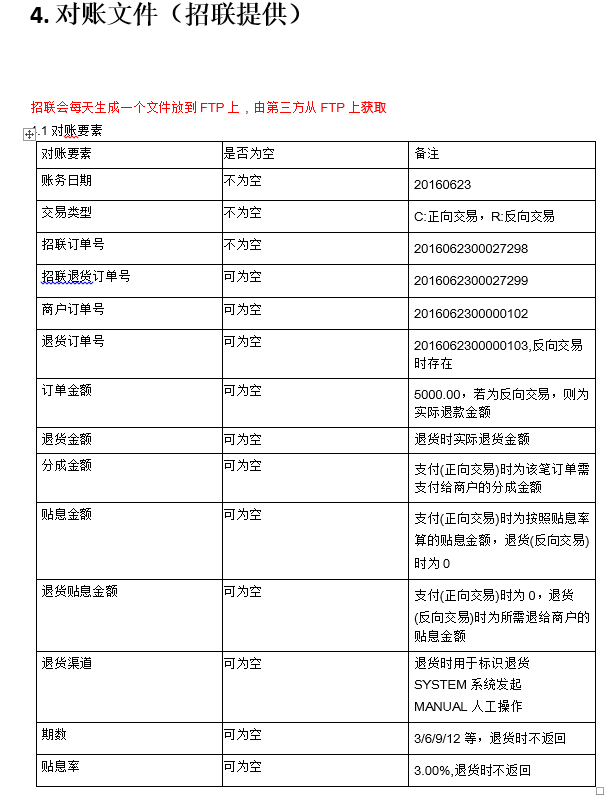
*重点描述开发涉及到的类、接口，描述各个类之间的关系，重点突出核心的方法，代码结构、层次*



1. ***接口设计***

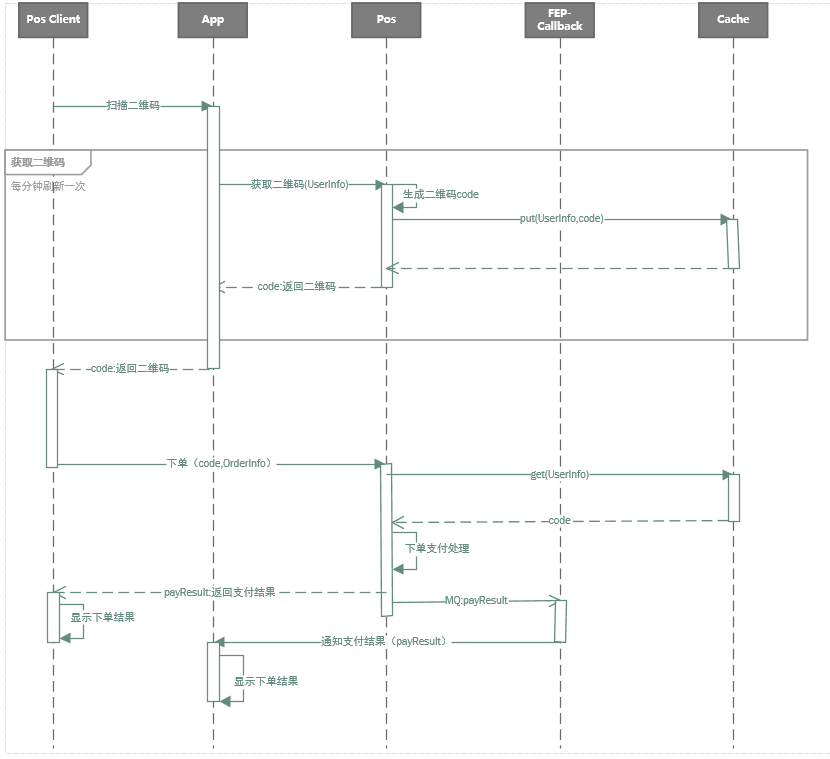
*重点描述对外接口定义，RPC类型的定义入参、返回、异常等，文件类型的定义文件格式，MQ类型的定义MQ消息格式。如果不提供对外接口，可以省略此类文档*

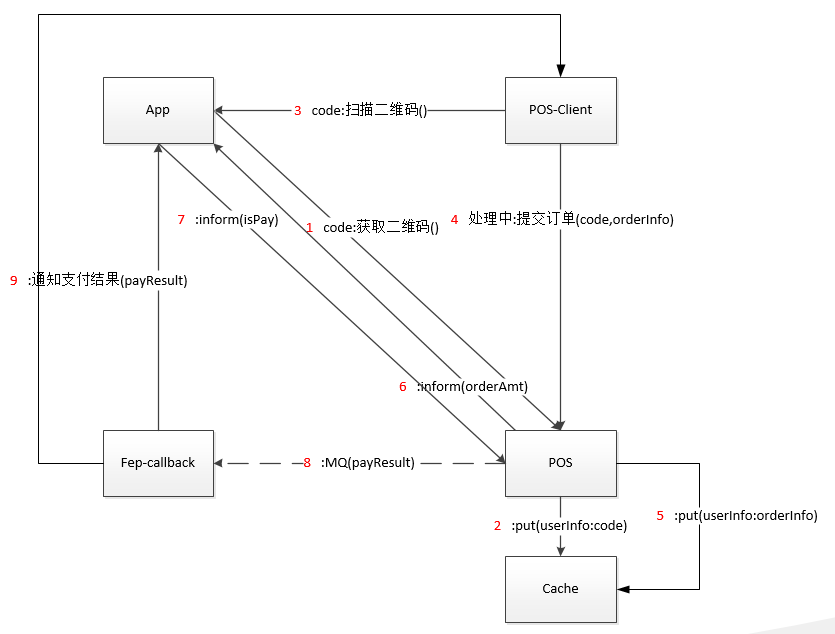




1. ***代码流程设计***

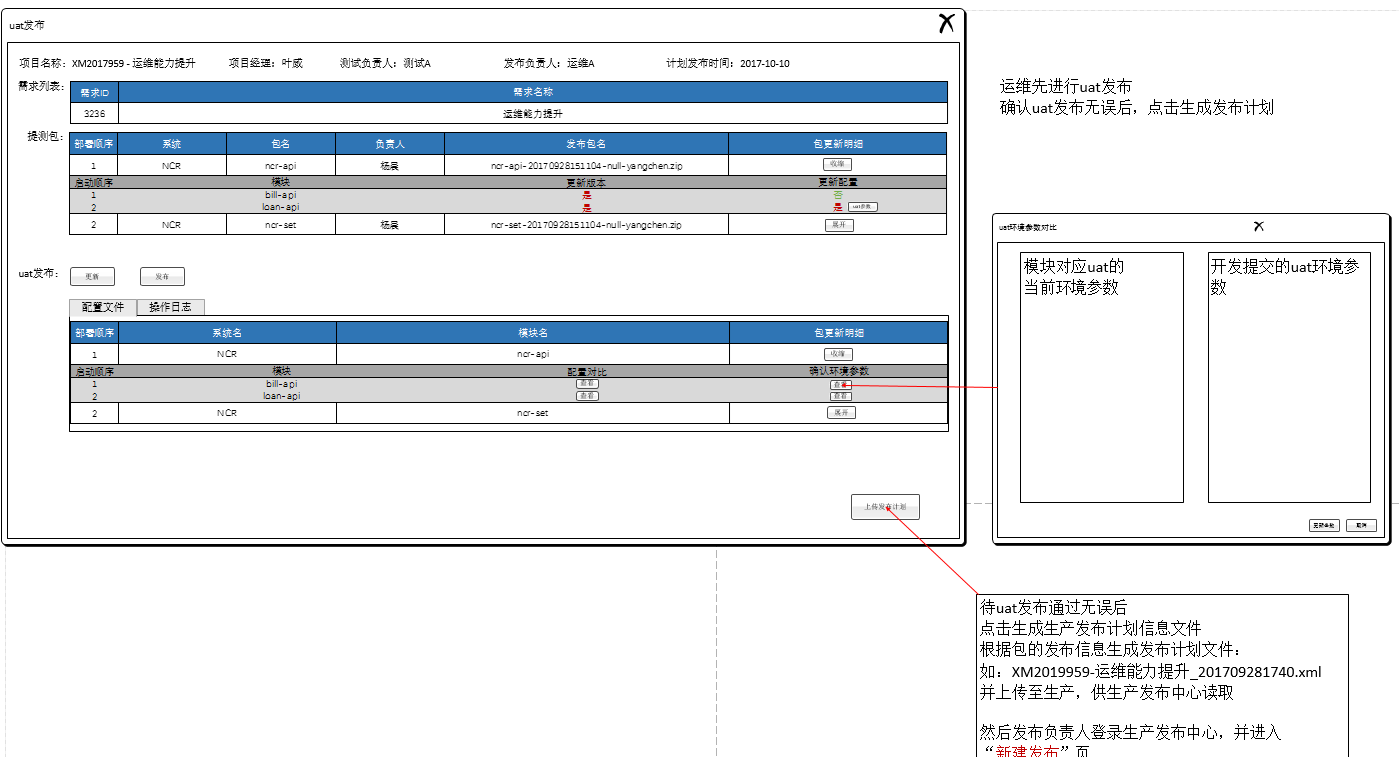
*重点描述模块、代码、类之间调用流程，数据输入、返回及流转状态*

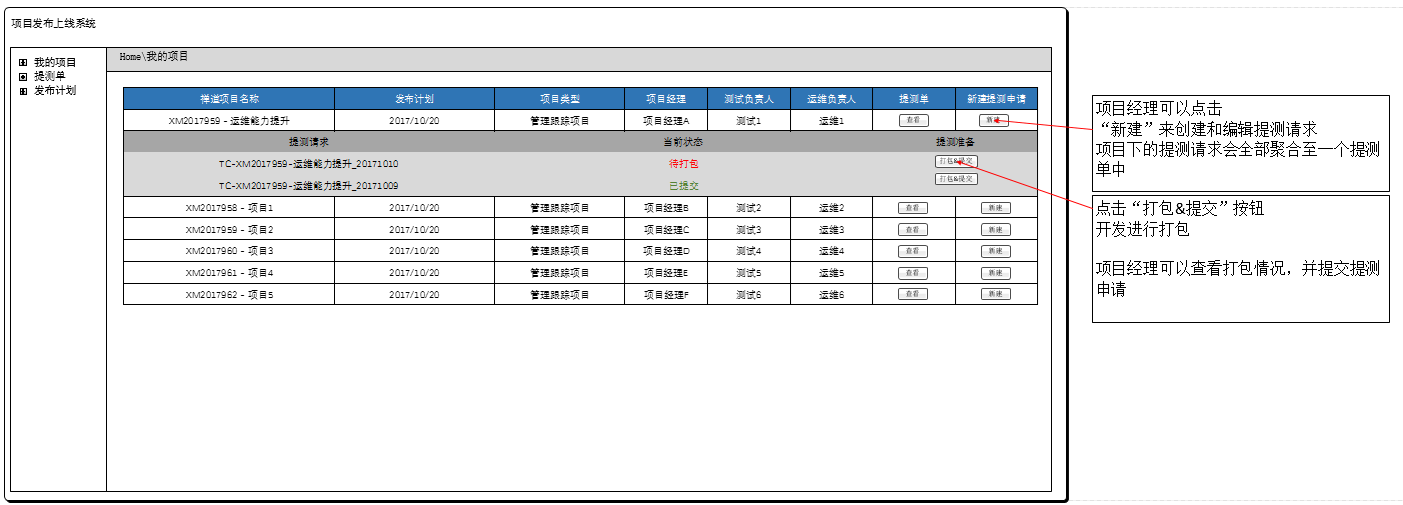




1. ***界面设计***

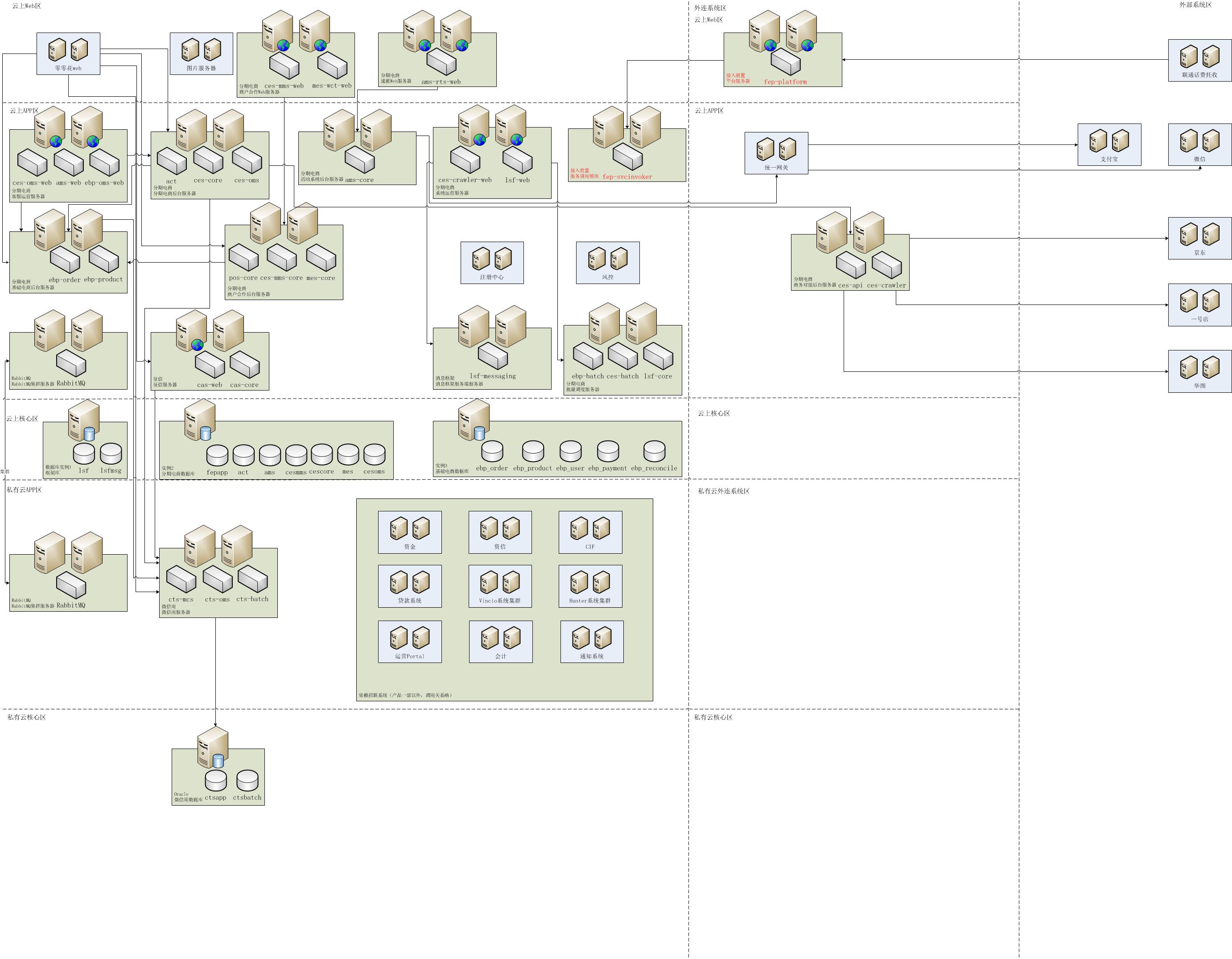
*界面设计重点表达界面操作流程，什么时候显示什么数据，什么时候进行什么操作，通过什么方式展示，同时需考虑用户体验，使用树、tab、颜色等做好数据、操作的分类显示，尽量接近最终开发版本，以免结果偏离太远。*





1. ***部署架构设计***

*系统开发完成后用于运维发布上线，系统使用服务器规划，进行部署架构设计。重点描述用到的服务器、数据库、中间件等，各个模块所在网络区域，负载均衡等*



## 三、工程代码设计

1. ***Maven模块设计***

一般以子系统或模块为单位搭建maven工程，常见一个模块包含以下maven工程

以子系统lsf为例子

lsf-service – 用于定义对外接口、参数、返回

lsf-common – 子系统内共用的对象、工具类代码

lsf-core – 模块后台运行单元，提供服务接口实现，处理数据库事务

lsf-web – 模块前端运行单元，包括前端界面，controller层实现，一般不访问数据库。

1. ***Java工程的类设计***

com.mucfc.xxx.common/utils/enums – 模块内部使用的常用类、工具类、枚举类

com.mucfc.xxx.domain – 系统业务相关的主要对象，主要是对业务的抽象，可以是多种entity聚合

com.mucfc.xxx.entity – 数据库相关的对象，与库表一一对应，多表关联查询可以定义一个新Entity

com.mucfc.xxx.dto – 接口之间传输各种数据的对象，如XXXRequest，XXXResponse等，可以转换成Domain类

com.mucfc.xxx.dao – 数据库操作对象，主要负责Entity的增删改查。

com.mucfc.xxx.controller – HTTP层负责参数接收，负责参数基本类型、数据范围检查，负责返回信息组装及异常处理。一般在这层组装DTO调用Service或Façade层

com.mucfc.xxx.service – 业务处理核心逻辑实现层，数据库相关事务在此层处理，本层不应该存在相互调用，公共的方法需要放在负载或者由更上层Façade层串联起来

com.mucfc.xxx.facade – 面向用户、第三方模块核心处理流程实现，复杂的业务逻辑如需要多个Service组合调用的在此层实现，关注流程调用，也可以处理事务

com.mucfc.xxx.schedule – 调度任务实现层，跟façade类似，主要区别是非同步调用，调用各种Service或Facade关注流程实现。