供应链平台技术白皮书

信雅达系统工程股份有限公司

2017/5/22

版本历史

版本号	修改人	修改日期	职位	备注
1.0	张晨、孙良良	2017-5	研发总监	

目录内容

1. 概述

1.1 文档目标

本文档概括性的描述了供应链平台系统架构、网络架构、数据交互、接口对接设计、平台安全设计、平台性能设计等。

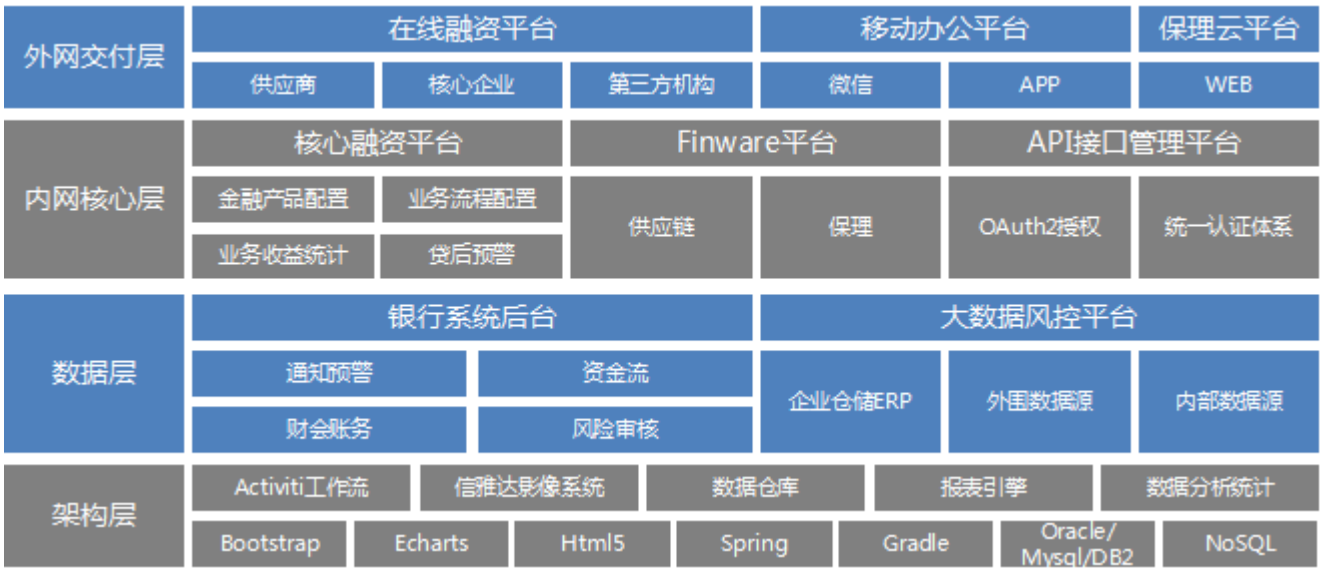
1.2 文档适用人员

下表列出了本文档的适用人员和相关的适用原因。

人员	适用原因
开发人员	在进行开发和概要设计时需要参考本文档。
设计人员	审查源代码和文档以确保其符合技术规范时使用。

2. 平台架构

2.1 系统架构



为满足不同用户的个性化需求，充分考虑到系统的扩展性，我们在系统架构层面进行了精心的设计，对数据、模型及交付领域进行了明确的定义与区分。系统总体上分为架构层、数据层、内网核心层和外网交付层四个层面：

1） 外网交付层：

用户可在互联网访问 web 系统和移动端系统。用户通过使用电脑或移动终端登录系统，创建、审核、办理各种业务流程。

2） 内网核心层：

核心融资平台是银行或企业业务人员在银行或企业内网访问的 web 系统，进行详细的业务查询，流程的变更等操作。Finware 平台、api 接口管理平台用来支撑所有系统快速开发、对接或开放数据接口。

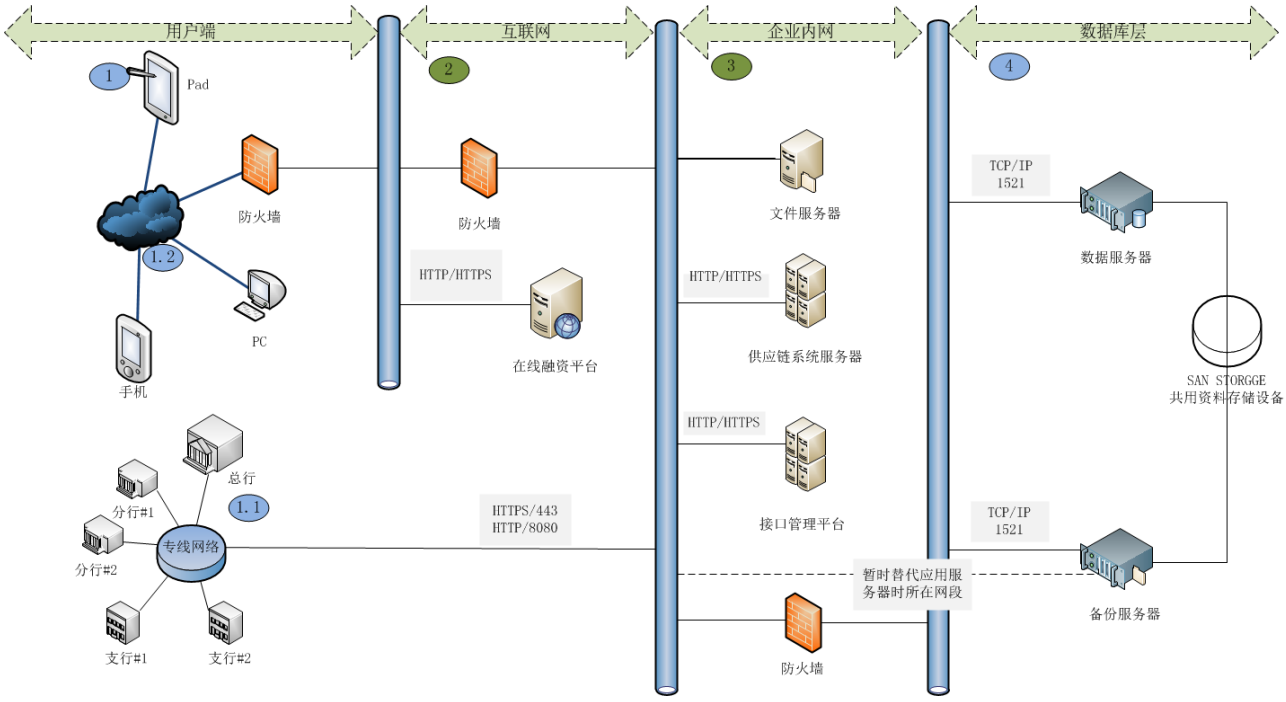
3） 数据层：

数据层用于采集和存储系统数据、业务明细数据、统计数据、数据接口等。亿级以上数据存储于 NoSQL，其他数据存储于关系型数据库。

4） 架构层：

架构提供了支撑其他各层所需的 IT 基础架构、物理服务器、软件与操作系统，以及管理营运等。

2.2 网络架构



■ 供应链平台将整个硬件环境放置于三层网段，用区分网段的方式将各个不同功能的服务器独立在各自的网段，以此增加系统服务器的安全性，网段的划分如下：

- ✧ 最敏感的系统数据——数据库服务器放置在最内部的网段。仅允许应用服务器和备份服务器访问。
- ✧ 供应链系统的服务器放置在应用层的网段，允许行内其他客户端访问。但仅限通过 HTTP 协议访问该服务器。
- ✧ 在线融资平台的服务器放置可连接 Internet 的网段。通过该服务器可以调用供应链系统服务器的服务功能和相应的页面。

■ 在各网段间加入防火墙，仅在防火墙上设置开放各网段必要的和允许访问的端口。从而阻挡具有漏洞的端口暴漏在外，防止外界利用这些端口攻

击本系统。各端口的设置如下：

源主机	目标主机	协议和端口	数据流向
行内客户端主机	供应链系统服务器	HTTP/8080 端口 HTTPS/443 端口	双向
在线融资平台服务器	供应链系统服务器	HTTP/8080 端口 HTTPS/443 端口	双向
供应链系统服务器	数据库服务器	TCP/IP/1521 端口	双向

3. 平台特点

3.1 扩充性

1) **易扩展性的实现方法**

■ 系统功能的扩展性实现方法：系统在设计之初就考虑到功能的扩展性，采用模块化方式加以交易处理流程的组装，针对业务需求的变化，仅需要设定更改逻辑流的调整。或有新的业务需求时，采用模块的方式添加到系统中。实现的方法只需要注册新的模块和新的逻辑流。

■ 系统硬件平台的扩展性实现方法：本系统使用的服务器可分阶段进行扩展，可先对服务器个别硬件进行扩容，比如 CUP、内存、硬盘等进行扩充。当业务量达到一定级数后可增加服务器采用负载均衡的方式扩充系统硬件的承载能力。

2) **升级扩容计划**

■ 当服务器配置无法满足日后业务量的需要时，可提高服务器的配置，比如增加 CUP、内存、硬盘等。

■ 当提高服务器的配置也无法满足业务量的增长时，可考虑增加服务器，并采用负载均衡的方式同步提供服务。

3.2 可靠性

1) **故障恢复能力：**

■ 本系统配置备份服务器，安装部署或更新版本时，同步更新主应用服务器的组件。当应用系统故障后，可立刻替换备份服务器的 IP 地址，暂时替代主应用服务器。采用此方法对应用服务器故障的应急响应时间将可在三分钟内恢复服务。

■ 每日全备份数据或短时间间隔增量备份数据库的方式保证数据库的完整性，当主数据库服务器故障时，可用备份服务器替换数据库服务器，将备份的数据安装在备份服务器的数据库中，暂时替代主数据库服务器。

2) **预防措施：**

■ 系统每日自动做对账处理，确保系统记录的交易与会计核心系统记录的交易保持一致。

■ 建议运营单位每日进行系统日志的检查。检查是否有影响系统运行产生问题的漏洞产生。

■ 建议运营单位每日进行数据库备份文件的检查。关注数据库备份是否按计划产生，以免出现故障时数据丢失。

■ 建议每月更新操作系统漏洞补丁，以免服务器遭受网路的攻击。

3） **数据备份：**

■ 设计每日换日作业、日终作业前进行数据库全备份。

■ 设计短时间间隔自动增量备份，时间间隔一般为 5 分钟备份一次。

■ 具体备份策略请参看后面章节 数据库备份策略。

4） **异常宕机：**

■ 本系统方案设置了备份服务器，当主服务器因为电源等故障无法正常工作时，可立刻接替主服务器继续工作。

3.3 安全性

1） **功能层安控说明**

■ 采用群组设定功能权限、用户关联群组的方式为系统使用者区分各自的权限范围。使用者登录后根据其所在的群组，展现其有权操作的功能菜单项和页面。

■ 当使用者绕过菜单项直接访问其无权操作的功能页面时，系统给出“无权操作该功能”的提示信息并自动导航至系统主页。

■ 在系统使用者管理功能页面，可设定用户的密码复杂度规则、强制修改密码的时间段、登录密码错误尝试的次数以及用户锁定后自动解锁的时间间隔。

■ 上述规则可在系统使用者信息新增时使用系统默认值，亦可单独为每个使用者设置其适合的规则。

■ 用户每次登录请求提交时，无论成功与否都记录该请求发起的使用者账号、发出请求的 IP 地址、发出请求的日期和时间、请求处理的结果，为日

后跟踪审查提供依据。

■ 支持将用户信息，密码验证移至核心系统统一管理和处理。

■ 支持统一管理用户（柜员）签到、签退处理。

■ 用户进入系统后，任何的操作都留做历史记录，包括查询数据的操作都需记录。同时提供《序时报表》，供审查审计所用。

2) **通讯层安控说明**

■ 各分支行的行内客户端主机可通过内部专线网络访问应用服务器。

■ 各分支行的行内客户端主机访问应用服务器允许的协议可选择 HTTP 或 HTTPS。如选择 HTTPS 协议则可将传输的数据加密后传递，更为安全。

■ 网银服务器也可通过内部专线网络访问应用服务器。协议与端口要求与各分支行类似，建议此处连接使用 HTTPS 协议。减少 Internet 上不确定因素的请求发送至应用服务器。

3) **客户应用安控说明**

■ 针对客户操作的不确定性，要求客户每次请求都需要提交验证码。

■ 在客户提交数据后，保护住系统页面，不允许其做任何页面操作，直至提交的流程处理完毕后提示处理结果信息。以此种方式防止客户用户频繁发送请求，增加服务器的负担的情况出现。

4. 技术架构

4.1 前后端交互设计

4.1.1 Client 层

Client 层即客户端，分为 WEB 端、APP 端、微信端以及第三方接口。

●WEB 端由浏览器实现，支持 Internet Explorer、Google Chrome 等浏览器在 PC 上的使用。

●APP 端由 Android/IOS 手机实现，支持在 Android/IOS 手机上的原生应用的展现使用。

●微信端由 HTML5 实现，支持在微信上使用供应链平台的部分功能。

●第三方接口主要是开放给第三方系统访问的部分业务功能接口。

客户端访问供应链平台的敏感数据时，可以在传输层采用 https 协议。为了产品的安全，必要时还会在通讯两端加入消息加密层，为了解决报文被劫持的问题，对客户端访问量进行监控。

各个类型客户端将基于用户名和密码的验证机制。开放给第三方的接口，将使用 DevKey 的形式进行开发者准入的验证。

4.1.2 数据绑定、验证和格式化

供应链平台数据绑定、验证和格式化分前后端，前端为了界面友好操作性，后端为了系统安全，防止恶意绕过前端请求，直接请求后端接口。后端的数据绑定、验证和格式化如下：

客户端数据请求格式如下：

```
{
  "username": "zhangsan",
  "password": "123456",
  "code": "8899"
}
```

使用注解@RequestBody 指定前端的 Json 字符串绑定到自定义的 JavaBean，JavaBean 的属性名和客户端请求属性一一对应，各个属性必须有 set 和 get 方法。

使用注解@Valid 指定需要数据验证的 JavaBean，JavaBean 加上 JSR303 的注解验证即可，验证的错误信息在 ValidationMessages.properties 定义，如下所示：

```
public class UserLoginReqBean extends BaseReqBean {
    /**
     * 用户名
     */
    @NotBlank(message = "{login.username.notblank}")
    private String username;
    /**
     * 密码
     */
    @NotBlank(message = "{login.password.notblank}")
    private String password;
    /**
     * 验证码
     */
    @NotBlank(message = "{login.code.notblank}")
    private String code;
}
```

常用的验证如下：

验证注解	验证的数据类型	说明
@AssertFalse	Boolean,boolean	验证注解的元素值是 false
@AssertTrue	Boolean,boolean	验证注解的元素值是 true
@NotNull	任意类型	验证注解的元素值不是 null
@Null	任意类型	验证注解的元素值是 null
@Min(value=值)	BigDecimal，BigInteger, byte, short, int, long，等任何 Number 或 CharSequence(存储的是数字)子类型	验证注解的元素值大于等于@Min 指定的 value 值
@Max（value=值）	和@Min 要求一样	验证注解的元素值小于等于@Max 指定的 value 值

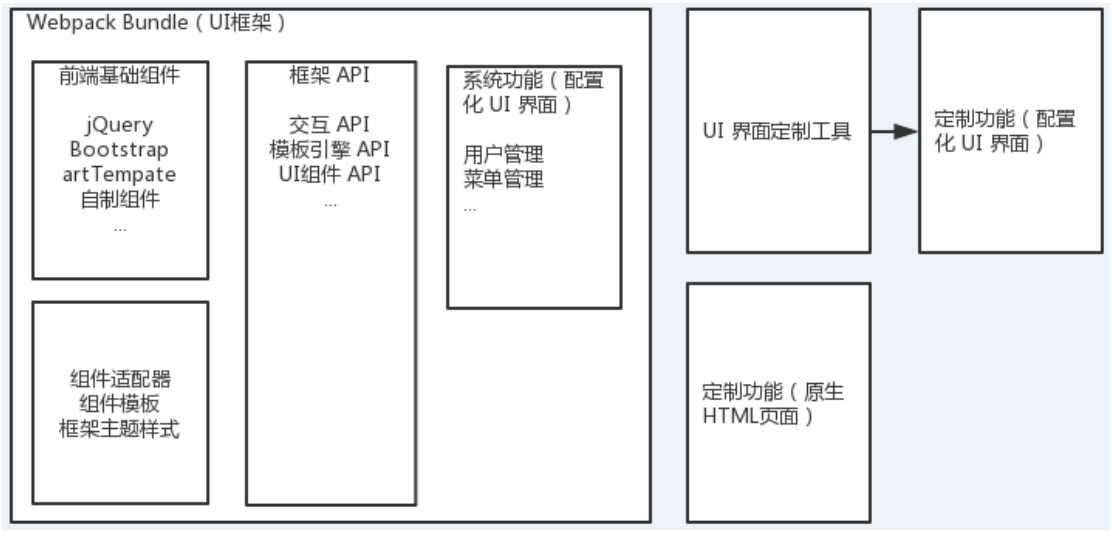
@DecimalMin(value=值)	和@Min 要求一样	验证注解的元素值大于等于@ DecimalMin 指定的 value 值
@DecimalMax(value=值)	和@Min 要求一样	验证注解的元素值小于等于@ DecimalMax 指定的 value 值
@Digits(integer=整数位数, fraction=小数位数)	和@Min 要求一样	验证注解的元素值的整数位数和小数位数上限
@Size(min=下限, max=上限)	字符串、Collection、Map、数组等	验证注解的元素值的在 min 和 max（包含）指定区间之内，如字符长度、集合大小
@Past	java.util.Date, java.util.Calendar; Joda Time 类库的日期类型	验证注解的元素值（日期类型）比当前时间早
@Future	与@Past 要求一样	验证注解的元素值（日期类型）比当前时间晚
@NotBlank	CharSequence 子类型	验证注解的元素值不为空（不为 null、去除首位空格后长度为 0），不同于@NotEmpty，@NotBlank 只应用于字符串且在比较时会去除字符串的首位空格
@Length(min=下限, max=上限)	CharSequence 子类型	验证注解的元素值长度在 min 和 max 区间内
@NotEmpty	CharSequence 子类型、Collection、Map、数组	验证注解的元素值不为 null 且不为空（字符串长度不为 0、集合大小不为 0）
@Range(min=最小值, max=最大值)	BigDecimal,BigInteger,CharSequence, byte, short, int, long 等原子类型和包装类型	验证注解的元素值在最小值和最大值之间
@Email(regexp=正则表达式, flag=标志的模式)	CharSequence 子类型（如 String）	验证注解的元素值是 Email，也可以通过 regexp 和 flag 指定自定义的 email 格式
@Pattern(regexp=正则表达式, flag=标志的模式)	String，任何 CharSequence 的子类型	验证注解的元素值与指定的正则表达式匹配
@Valid	任何非原子类型	指定递归验证关联的对象； 如用户对象中有个地址对象属性，如果想在

		验证用户对象时一起验证地址对象的话，在地址对象上加@Valid 注解即可级联验证

使用注解@ResponseBody 格式化自定义的 JavaBean 为 Json 字符串，JavaBean 的属性名和客户端响应属性一一对应，各个属性必须有 set 和 get 方法。示例如下：

```
{
  "result": 0,
  "resultNote": "操作成功",
  "resultErrorMap": null,
  "userId": "u00001",
  "username": "root",
  "roleId": "ROLE000001",
  "roleName": "系统管理员",
  "roleType": 1,
  "corpId": "",
  "deptId": "",
  "menuList": [
    {
      "menuId": "MENU010000",
      "menuName": "系统功能",
      "menuLevel": 1,
      "parentId": null,
      "menuPath": null,
      "menuOrder": 1,
      "childFlag": true,
      "note": "系统功能",
      "subMenuList": null
    }
  ]
}
```

4.2 前端设计

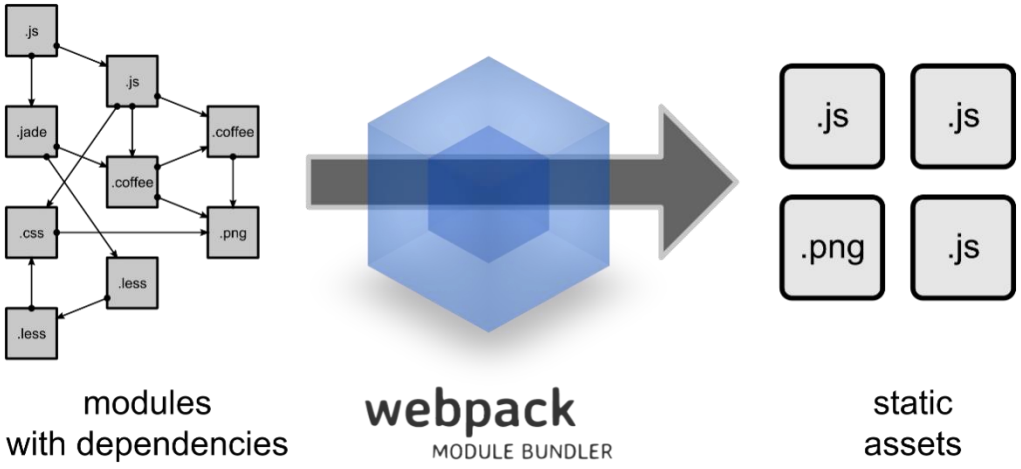


特点：

1) 通过 UI 规范，形成成熟的业务形态，统一操作习惯。
2) 框架规范 API 与模板化管理，达到前端组件实现与业务定制开发解耦，后期组件库升级对业务代码无影响。
3) 基于 SASS 等预处理技术编写的框架主题样式，灵活、便于定制与维护。
4) 框架代码通过 Webpack 技术整合打包，由专业团队统一维护，统一升级，业务开发则更加专注于业务功能的定制。
5) 定制功能可以通过配置化实现，开发效率高。同时也支持原生开发方式，无技术壁垒。

4.2.1 Webpack

Webpack 是一个模块打包器。它将根据模块的依赖关系进行静态分析，然后将这些模块按照指定的规则生成对应的静态资源。



优势：

1) 代码拆分：Webpack 有两种组织模块依赖的方式，同步和异步。异步依赖作为分割点，形成一个新的块。在优化了依赖树后，每一个异步区块都作为一个文件被打包。
2) Loader：Webpack 本身只能处理原生的 JavaScript 模块，但是 loader 转换器可以将各种类型的资源转换成 JavaScript 模块。这样，任何资源都可

以成为 Webpack 可以处理的模块。

3) 智能解析：Webpack 有一个智能解析器，几乎可以处理任何第三方库，无论它们的模块形式是 CommonJS、 AMD 还是普通的 JS 文件。甚至在加载依赖的时候，允许使用动态表达式 require("./templates/" + name + ".jade")。
4) 插件系统：Webpack 还有一个功能丰富的插件系统。大多数内容功能都是基于这个插件系统运行的，还可以开发和使用开源的 Webpack 插件，来满足各式各样的需求。
5) 快速运行：Webpack 使用异步 I/O 和多级缓存提高运行效率，这使得 Webpack 能够以令人难以置信的速度快速增量编译。

4.2.2 Babel

Babel 是一个广泛使用的转码器，可以将 ES6 代码转为 ES5 代码，从而在现有环境执行。

优势：

作为一种语言，JavaScript 在不断发展，新的标准 / 提案和新的特性层出不穷。在得到广泛普及之前，Babel 能够让你提前（甚至数年）使用它们。ES6 标准发布后，前端人员也开发渐渐了解到了 ES6，但是由于兼容性的问题，仍然没有得到广泛的推广。通过使用 Babel，我们是在为明天技术升级做准备。

4.2.3 Bootstrap

简介：由来自 Twitter 的设计师 Mark Otto 和 Jacob Thornton 合作开发的前端框架。基于 HTML、CSS、JAVASCRIPT，它简洁灵活，使得 Web 开发更加快捷。

优势：

1) Twitter 出品：Bootstrap 出自 twitter，大厂出品，并且开源，自然久经考验，减少了测试的工作量。站在巨人的肩膀上，不重复造轮子。
2) 基于 LESS(SASS),丰富的 Mixin:Bootstrap 是目前最好的基于 Less(Sass) 的前端框架，丰富而实用的 Mixin 应该是其最好的地方。

3) 响应式栅格系统：Bootstrap 搭建好了实现 Responsive Design 的基础框架，并且非常容易修改。

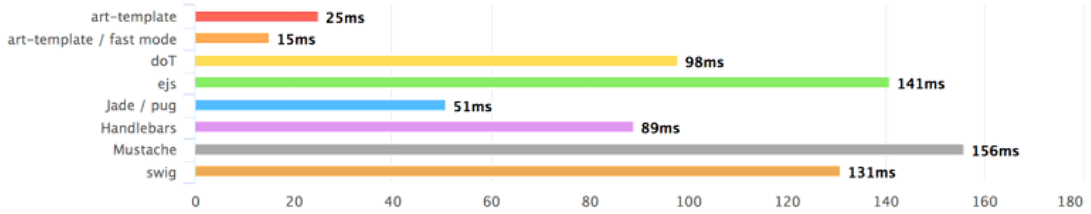
4) 丰富的组件与插件：Bootstrap 的 HTML 组件和 JS 组件非常丰富，并且代码简洁，非常易于修改，你完全可以在其基础之上修改成自己想要的任何样子。这是工作效率的极大提升。另外，由于 Bootstrap 的火爆，又出现了不少围绕 Bootstrap 而开发的插件，如 Font-Awesome 等。

4.2.4 ArtTemplate

一个渲染性能出众模板引擎，无论在 NodeJS 还是在浏览器中都可以运行。原本是这原本是腾讯内部公用组件之一，现已在 MIT、BSD、GPL 协议下开源，人人都可以使用了。

优势：

1) 性能卓越，执行速度通常是 Mustache 与 tmpl 的 20 多倍，拥有接近 JavaScript 渲染极限的的性能



2) 调试友好：语法、运行时错误日志精确到模板所在行；支持支持在模板文件上打断点（Webpack Loader）

3) 支持压缩输出页面中的 HTML、CSS、JS 代码

4) 支持 Express、Koa、Webpack

5) 支持模板继承与子模板

6) 兼容 EJS、Underscore、LoDash 模板语法

7) 模板编译后的代码支持在严格模式下运行

8) 支持 JavaScript 语句与模板语法混合书写

9) 支持自定义模板的语法解析规则

10) 浏览器版本仅 6KB 大小

4.2.5 SASS

简介：一种 CSS 的开发工具，提供了许多便利的写法，大大节省了设计者的时间，使得 CSS 的开发，变得简单和可维护。诞生于 2007 年，是最早也是最成熟的 CSS 预处理器，拥有 ruby 社区的支持和 compass 这一最强大的 css 框架，目前受 LESS 影响，已经进化到了全面兼容 CSS 的 SCSS。

优势：

1) 支持变量、嵌套、逻辑判断等特性

2) 支持局部导入，文件组织管理更方便有条理

3) 支持混合宏，帮助设计师方便地维护复杂的浏览器维护兼容的代码

4.3 后端解耦分层

4.3.1 Controller 层

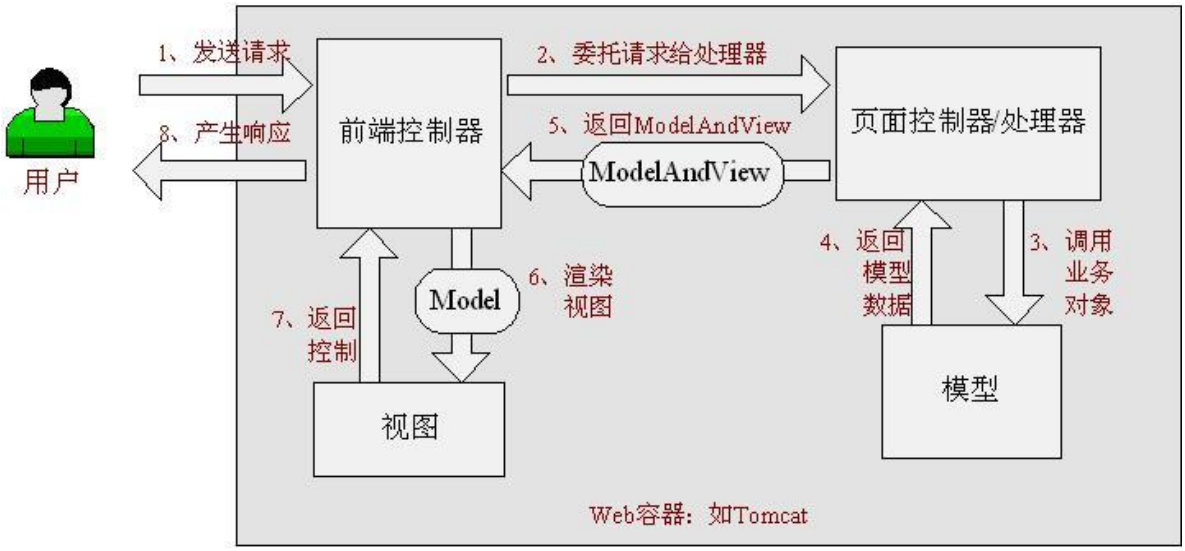
用户一个完整的请求过程如下：

1、首先用户发送请求————>前端控制器，前端控制器根据请求信息（如 URL）来决定选择哪一个页面控制器进行处理并把请求委托给它，即以前的控制器的控制逻辑部分；

2、页面控制器接收到请求后，进行功能处理，首先需要收集和绑定请求参数到一个对象，这个对象在 Spring Web MVC 中叫命令对象，并进行验证，然后将命令对象委托给业务对象进行处理；处理完毕后返回一个 ModelAndView（模型数据和逻辑视图名）；

3、前端控制器收回控制权，然后根据返回的逻辑视图名，选择相应的视图进行渲染，并把模型数据传入以便视图渲染；

4、前端控制器再次收回控制权，将响应返回给用户，至此整个结束。



在上图中，Controller 层由前端控制器和页面控制器/处理器组成。

前端控制器即 DispatcherServlet，提供 Spring MVC 的集中访问点，且负责职责分派给各个页面控制器/处理器。

页面控制器/处理器就是根据业务定义的 Controller，负责数据验证、格式化、数据绑定等，调用业务 Service，返回视图给客户端。在供应链平台中，前后端是通过 Controller 来做数据交互的，为了做到尽可能的解耦分层，采用非侵入式设计，前后端通过 http/json 数据交互（除了特殊页面需要 JSP、或者第三方接口需要 webservice 等服务），这样前端可以实现资源静态化，后台业务流程清晰，代码可以重用。

4.3.2 领域模型

领域模型以面向对象的风格封装了核心业务数据和逻辑，以及其相关的功能，构成了一个以领域为中心的应用程序的心脏。领域模型分为几种类型，其中大部分为 Spring beans。

1) Gateways 供应链开发平台中将会集成诸如短信网关、Push 消息、第三方接口等系统服务。用于支撑系统的扩展性服务。
2) Entities and value objects 他们是普通的旧式 Java 对象（POJO）封装

了状态和行为，并表示业务概念、业务情况的信息，和商业关系，本质上是业务逻辑模型。实体和值对象之间的区别在于，前者具有不同的身份，而后者则没有。如前后端交互的请求对象和响应对象。

3) DAOs 数据访问对象是负责管理实体集合。他们封装了底层数据库操作，并且使用了完全隔离持久性逻辑与业务逻辑的持久化框架 MyBatis。

4) Services 是供应链开发平台的业务逻辑层。他们构建了应用程序的工作流程，指导实体的 DAO，网关和其他服务进行工作。他们是无状态的对象，是业务重点，他们还定义了事务的上下文和边界。

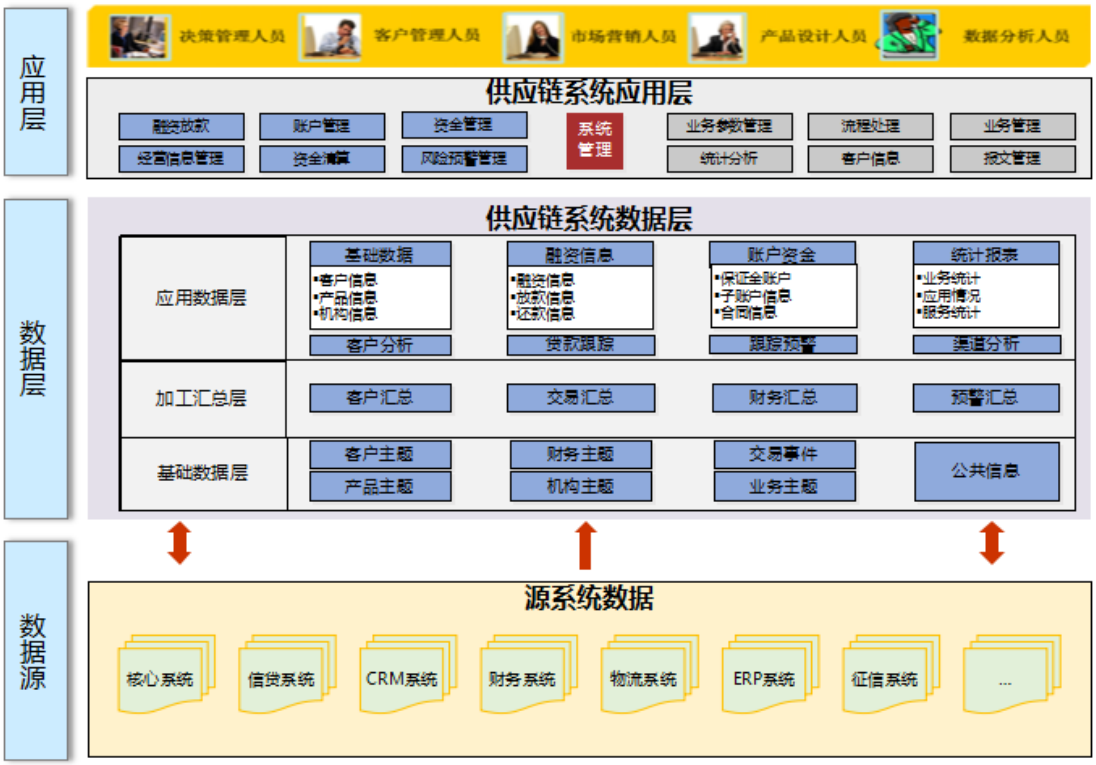
4.3.3 数据层

数据层是数据库或者数据源。数据层分为系统数据、业务数据、统计数据、数据接口。系统数据存储用户、角色、菜单等数据，业务数据即核心企业、供应商、经销商、保理商、买方买方业务相关数据，统计数据则是由存储过程统计业务明细数据而来，数据接口是用来同步第三方数据的（如对接财务系统，企业 ERP 系统等）。

供应链平台数据存储于关系型数据库 Mysql 或 Oracle。对性能、大数据要求时，可能会引入内存数据库 Redis，分布式数据库 Hbase。

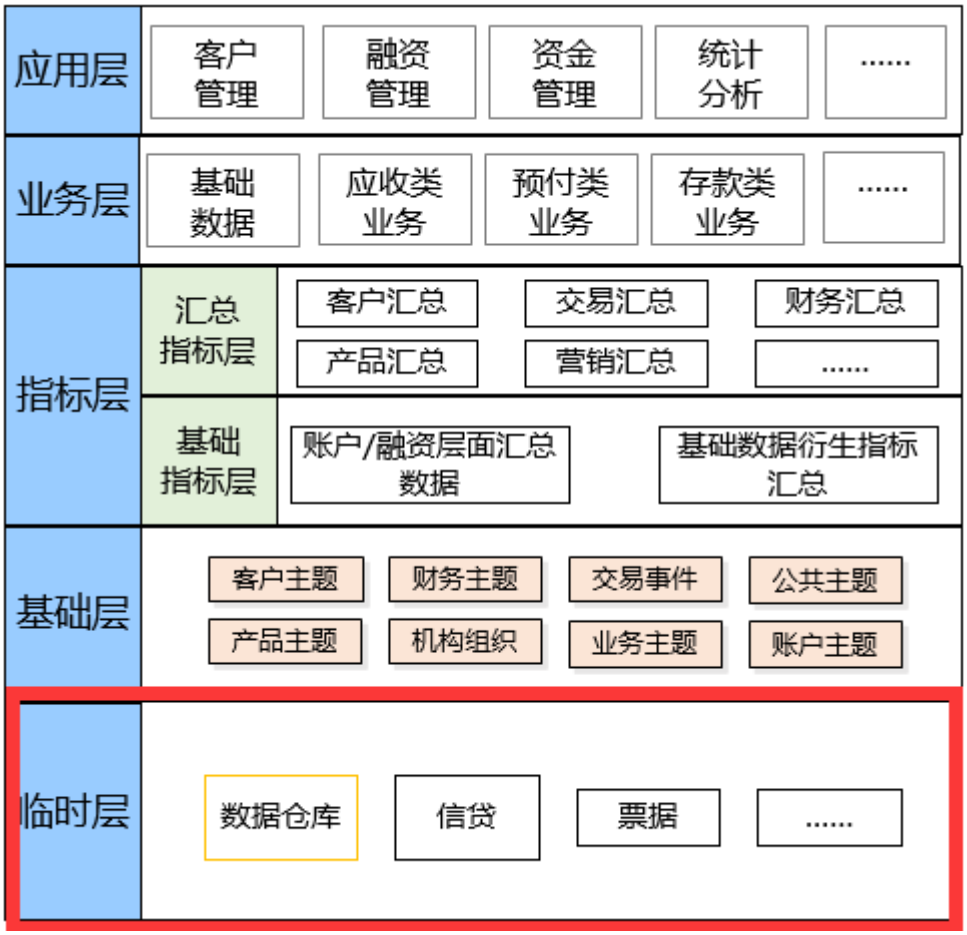
4.4 数据架构

4.4.1 数据架构说明



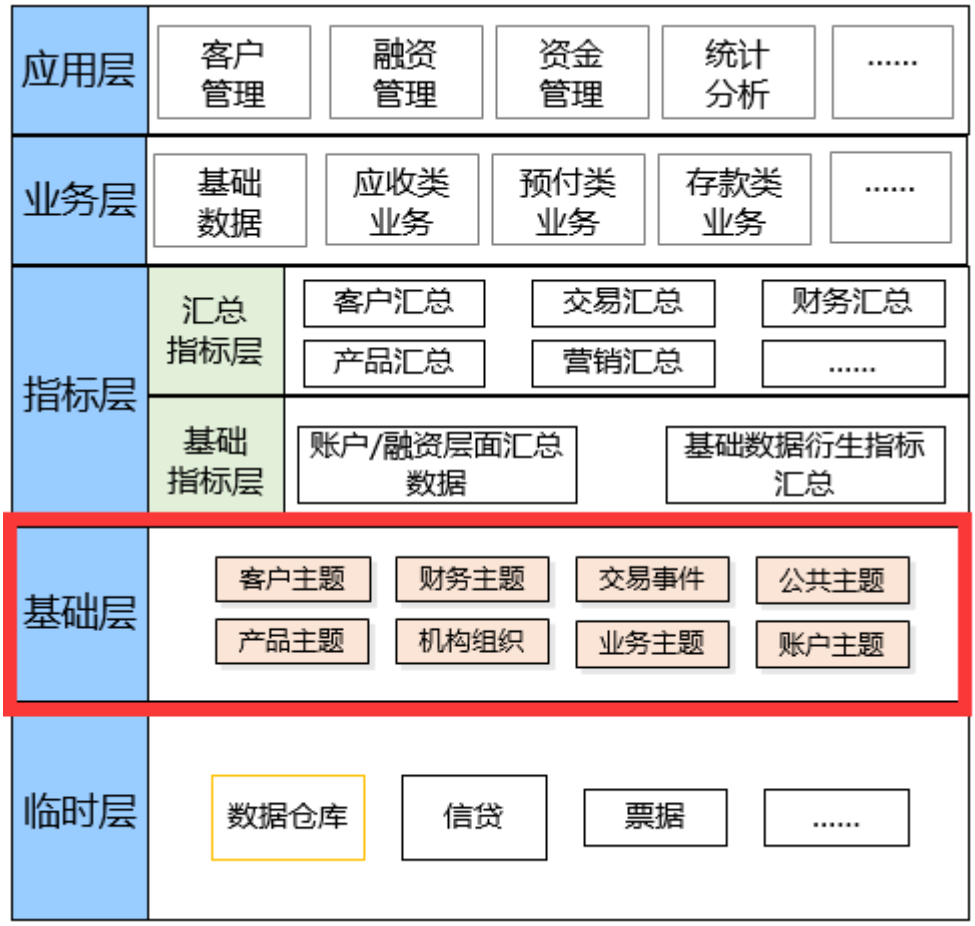
- 通过数据架构，系统分为应用层、数据层、数据源三层架构
- 应用层为系统展现层，负责用户的交互、信息的录入、流程的处理等
- 数据层分为应用数据层、加工汇总层、基础数据层，为系统核心数据处理加工层，用来做基础数据的加工、汇总、分类存放
- 数据源为各系统元数据，不规则，由定时任务统一获取存放至基础数据层。

4.4.2 临时层/缓冲层/接口层



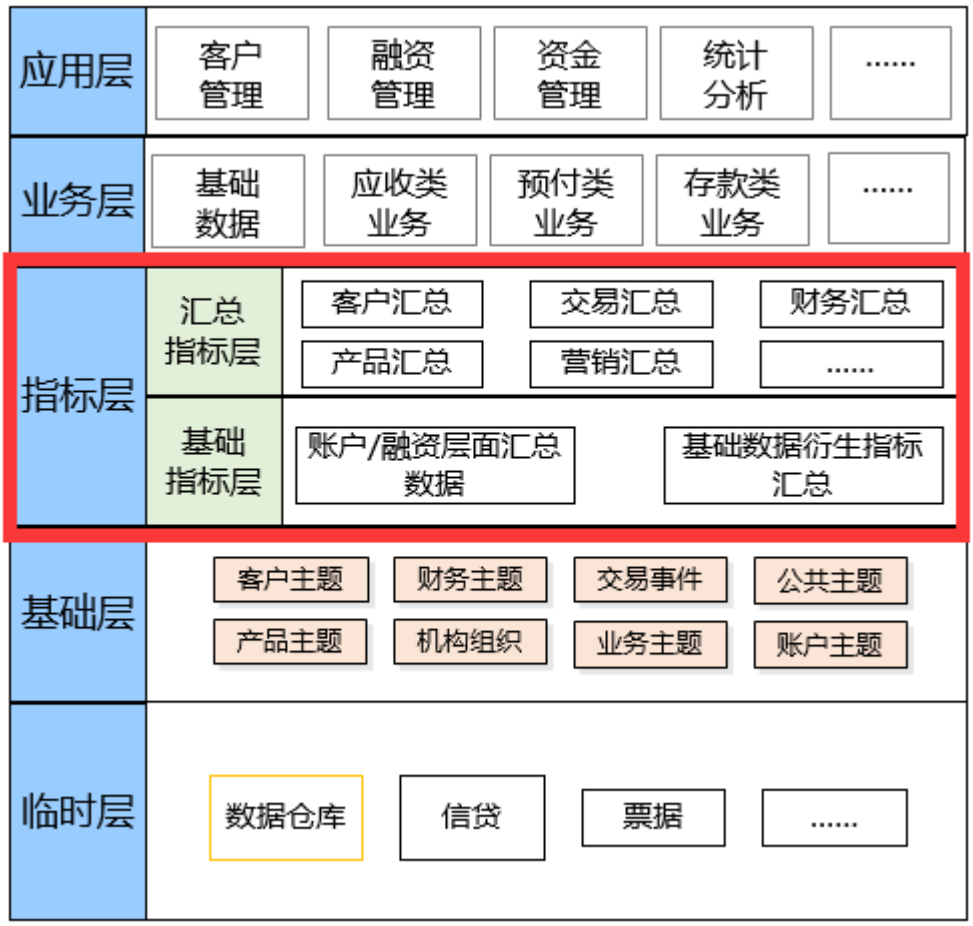
- 职责：接收外系统的数据，缓存存放，客户、账户类数据，存放全量数据
- 建模原则：与来源系统保持一致
- 命名规范：schema 命名或前缀，表命名为来源系统简称_来源系统表名
- 数据存放周期：3-7 天

4.4.3 基础层/事实层



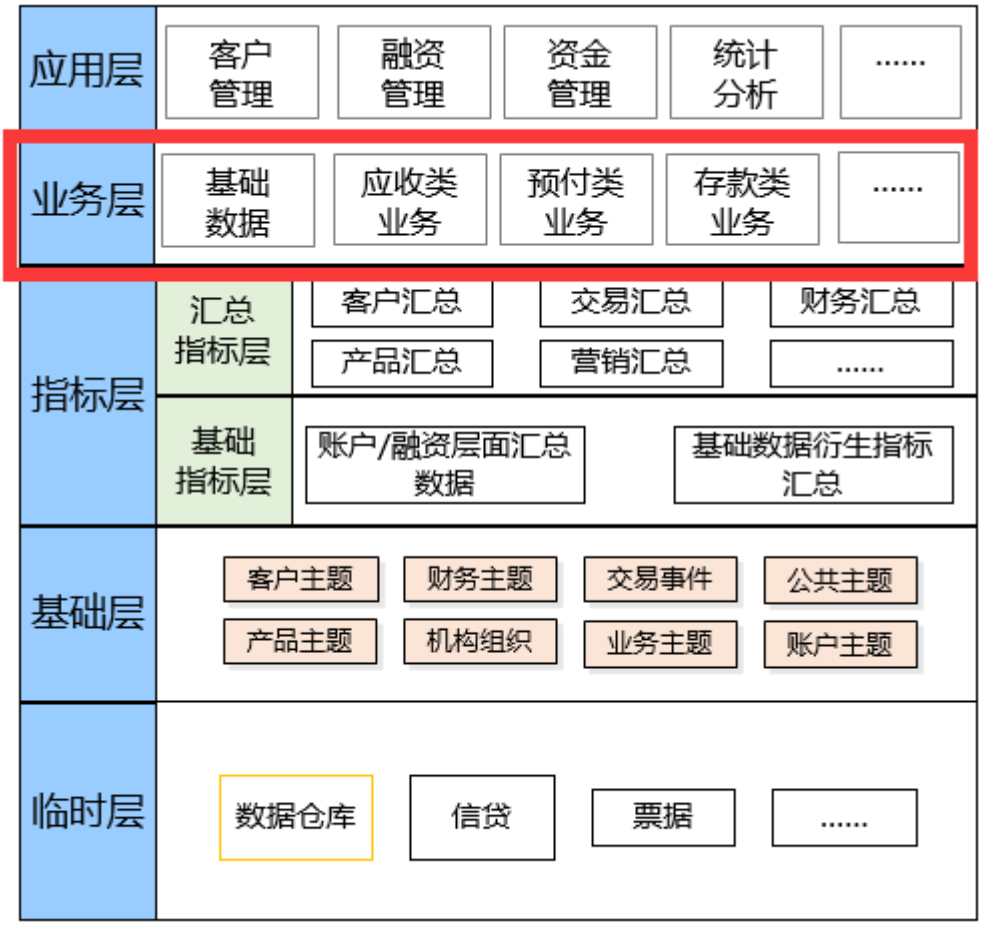
- 职责：
  1. 存放业务相关的细粒度的原始全量数据，不汇总
  2. 完成数据清洗和标准化处理
  3. 该层不产生衍生指标
- 建模原则：
  1. 按主题存放，主题采用行内企业逻辑数据模型的十大主题
  2. 按照三范式建模，适当去范式化
  3. 采用拉链表记录历史变化，适当采用短拉链
- 命名规范：schema 命名或前缀，表命名为主题简称
- 数据存放周期：3 年

4.4.4 指标层/汇总层



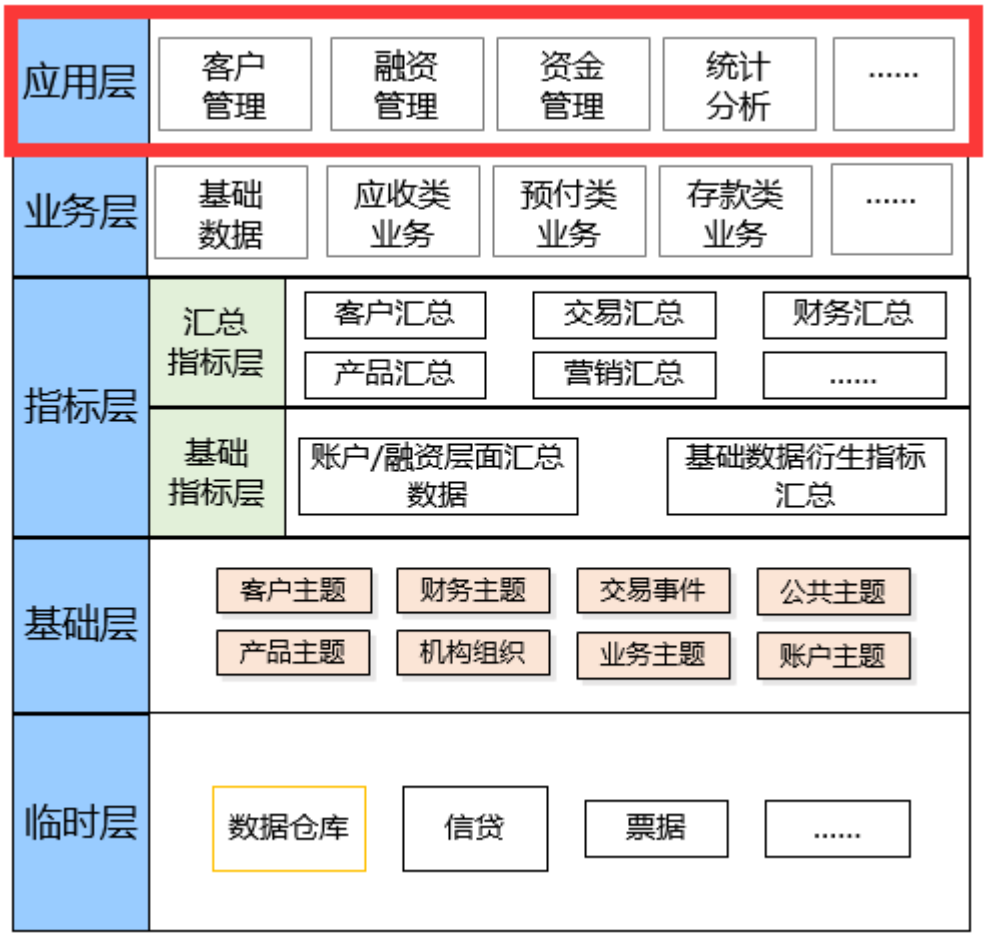
- 职责：
  1. 提供基础业务指标
  2. 明细级别、汇总级别
  3. 基础指标层完成数据的轻量汇总加工
  4. 汇总指标层按照客户、交易等纬度完成指标运算
- 建模原则：
  1. 去范式化建模汇总指标
  2. 宽表建模
  3. 实体由纬度加指标组成
  4. 采用时间切片保存历史数据
- 命名规范：schema 命名或前缀，按角度+纬度+修饰词进行数据库命名
- 数据存放周期：1-3 年

4.4.5 业务层/指标衍生层



- 职责：提供业务属性较高的业务指标
- 建模原则：
  1. 去范式化建模
  2. 宽表建模
  3. 采用时间切片保存历史数据
- 命名规范：schema 命名或前缀，表命名业务类比_*
- 数据存放周期：1-3 年

4.4.6 应用层/数据发布层



- 职责：
  1. 为外部系统提供数据接口服务和直连服务
  2. 按照应用要求加工数据，提供数据接口，外部系统可以直连该层的表
  3. 直接访问业务层、指标层和基础层的数据时需要在应用层创建视图
- 建模原则：
  1. 应用驱动
  2. 充分考虑数据加工的性能
- 命名规范：schema 命名或前缀，按应用进行数据表命名
- 数据存放周期：按应用需要存放

4.5 多租户

1） 业务背景说明

■ 每个核心非银企业/银行都有自己的供应链金融平台，要实现最后一公里的无缝对接，完全依靠平台的技术力量是不够的。这个时候如果把平台的业务能力开放出来，提供足够细粒度的 API，有开发能力的租户就可以自行实现业务对接，并持续更新。随着业务的复杂，越来越多的需求会使平台开发者难以应对，而租户本身对自己的需求的理解是最贴切的，可以利用他们的力量，同时并行开发多个服务，只有这样才能快速丰富供应链金融平台的服务，并确保这些服务的实用性

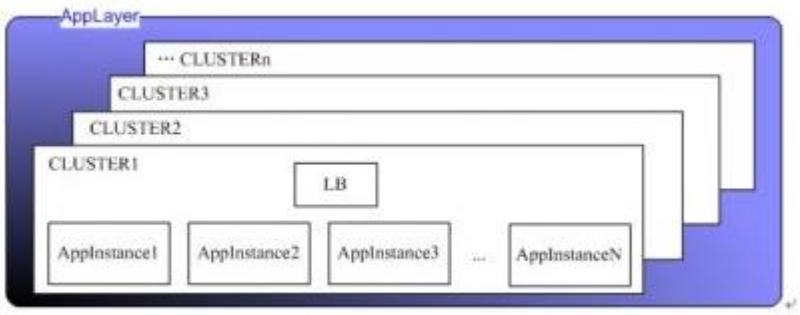
■ 对于供货商/经销商，可以通过 Web、PC 和移动端访问开放系统和其中银行/保理公司提供的融资产品、交易、数据分析类应用；对于银行/保理公司可以通过 web 访问管理和开发融资产品，进行授信审批，贷后跟踪

2） 多租户网络架构

■ 多租户通过 PC/移动端访问互联网在线融资平台，或通过互联网-企业内网方式访问供应链系统管理平台（集群）

■ 总行/分行通过企业专网直接访问供应链系统管理平台（集群）

3） 多租户应用架构



■ 一个应用集群运行着多个应用实例，应用的实例将根据租户注册情况动态创建，根据用户使用情况动态调整和管理资源使用

■ 多租户模型所创建的多个实例为要自动部署

4） 多租户数据架构比较

独立数据库	共享数据库，独立schema	共享数据库及数据表
资源共享度 低 → 高		
隔离度 高 → 底		
复杂度 低 → 高		
占用成本 高 → 低		

5） 多租户数据架构设计

■ 考虑到资源共享方便以及占用成本，系统采用共享数据库表(通用表或公有表)模式进行多租户的数据存放和管理

■ 涉及到多租户的表中存放多个租户的数据，通过租户 ID 来区分和隔离，对于定制化的信息通常采用统一类型(如 varchar)的稀疏列

■ 考虑到存放了所有租户的数据，数据表和数据库的数据量在达到千万级之后，需要使用散列分区技术的大数据库系统

■ 数据备份及导入：由于采用共享数据表的方式，系统会将所有租户的数据一起备份，但是若要为某一个租户或按租户分开进行数据备份，可以根据租户 ID 来取得对应的数据然后再备份导入

4.6 工作流

1）工作流的意义

a） 工作流是指对于工作流程及其各操作步骤之间业务逻辑和规则在计算机中描述和表现。在工作中由一组任务组织起来以完成某个经营活动而形成的业务，在多个参与者或者多个角色之间相互协作完成，根据预定规则在整个过程中自动传递业务的信息、任务和文档之类，这种场景就是工作流正是能够解决的。

b） 使用工作流后，业务将会实现在工作过程中的自动流转、有效传递和智能分配，可以灵便的实现数据整合和数据统计，提升效率质量，消除信息孤岛从而达到降低经营成本加快生产速度。

c） 工作流的核心是工作流引擎，它是驱动流程流动的主要部件，负责解释工作流流程定义，创建并初始化流程实例，控制流程流动的路径，记录流程运行状态，挂起或唤醒流程，终止正在运行的流程，与其他引擎之间通讯等等工作。

2）工作流的功能、特点

a） 工作流的功能主要有：流程定义和导入导出，和客户端 web 之间的交互与协作，流程的管理和监控等

b） 工作流的特点有：

i. 自动与协调：工作当中的自动化，指它自动进行的特征，而不是说没有人的参与。工作流实际上是一个人-电脑协调的混合过程，在一个实际的工作流中，通常总有些步骤是人完成的。

ii. 监察与控制：是工作流系统的重要功能与特征。这不仅包括对正在发生的业务过程（工作流），还包括它的定义或改变（比如 BPR 的过程），可追踪业务进程。

iii. 流程追踪：工作流管理系统是一个真正的“人-机”系统，用户是系统中的基本角色，是直接的任务分派对象，用户可以直接看到电脑针对自己列出的“任务清单”，跟踪每一项任务的状态，或继续一项任务，而不必从一个模块退出，进入另一个模块，搜索相应任务的线索。

3）工作流的使用

a) Activiti 是国内外广泛使用的开源的工作流和业务流程管理平台，它基于 Java 标准的稳定的 BPMN2.0 流程引擎建立可以在有更长久的生命力，它良好的易用性和轻量性可以使用较为方便快速从而有效节约时间，它强大的协作工具可以开发人员、运维人员包括业务人员本身都能够更好的协同工作。

b) Activiti 可以配置流程包括顺序、平行拆分、同步、排他选择、简单合并以及单一审批和比例计算审批等，同时提供可视化流程配置工具，有利于普通业务人员使用。

c) Activiti 包括几个核心部件：引擎 Activiti Engine 用来解释流程定义、控制工作流实例和统计收集数据等，管理系统 Activiti Explorer 提供任务管理、系统管理和数据报表展示等功能，业务流程设计和部署 Activiti Modeler 等。这样即使完全不懂得程序开发的业务人员也能够使用，这恰恰增进了业务人员和开发人员的交流，极大地提高了开发效率，避免因需 求理解的差异导致的返工。

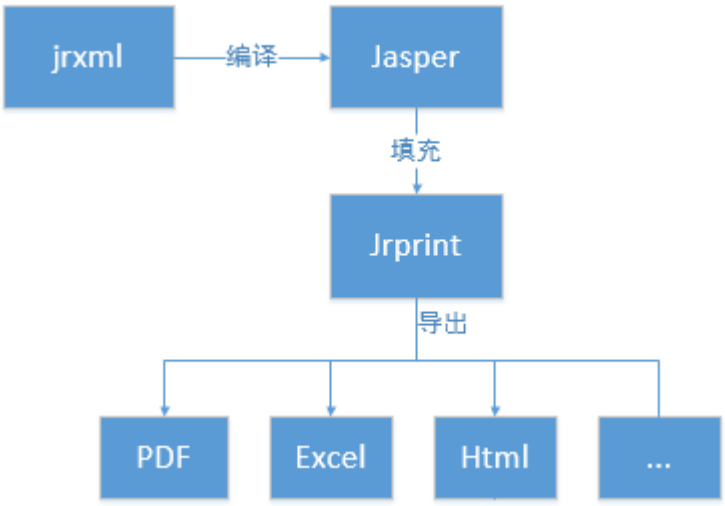
4) 目前流行的工作流引擎有 Activiti 和 jBPM5，而在 jBPM5 发布以前大多数项目、平台都是基于 jBPM3、jBPM4 开发的。下面表格从技术和实际应用上对 Activiti 和 jBPM5 进行比较：

技术组成	Activiti	JBPM5
ORM 框架	Mybatis3	Hibernate3
持久化标准	无	JPA 规范
事务管理	MyBatis 机制/Spring 事务控制	Bitronix，基于 JTA 事务管理
Spring 支持	原生支持 Spring，在流程中可以使用 Spring 代理的 Bean 作为表达式的一部分，且支持 JPA 及事务管理	默认没有提供对 Spring 的支持
设计模式	Command 模式、观察者模式等	
内部服务通讯	Service 间通过 API 调用	基于 Apache Mina 异步通讯
集成接口	SOAP、Mule、RESTful	消息通讯
支持流程格式	BPMN2、xPDL、jPDL 等	目前仅只支持 BPMN2 xml

引擎核心	PVM（流程虚拟机）	Drools
技术前身	jBPM3、jBPM4	Drools Flow
附加工具	提供了基于 Eclipse 插件的流程设计器——Eclipse Designer，提供基于 REST 风格的 Activiti Explorer，可以用来管理仓库、用户、组、启动流程、任务办理等	同样提供 Eclipse 插件和一个 Web 应用管理流程
发布周期	固定每两个月发布一版，其中包括：引擎、Eclipse Designer、Activiti Explorer、REST 应用	JBPM 的发布周期相对来说不太固定，发布内容包括引擎及基于 Eclipse 的设计器

4.7 报表引擎

JasperReport 是一个开源、灵活的报表生成工具，辅以 IReport（一个图形化的报表设计工具），可以在预先设定好格式的报表基础上进行数据填充并可导出各种格式（如 ：PDF、Excel、Html 等）的数据报表，下图为生成报表流程：



示例：应收账款转让通知书面函

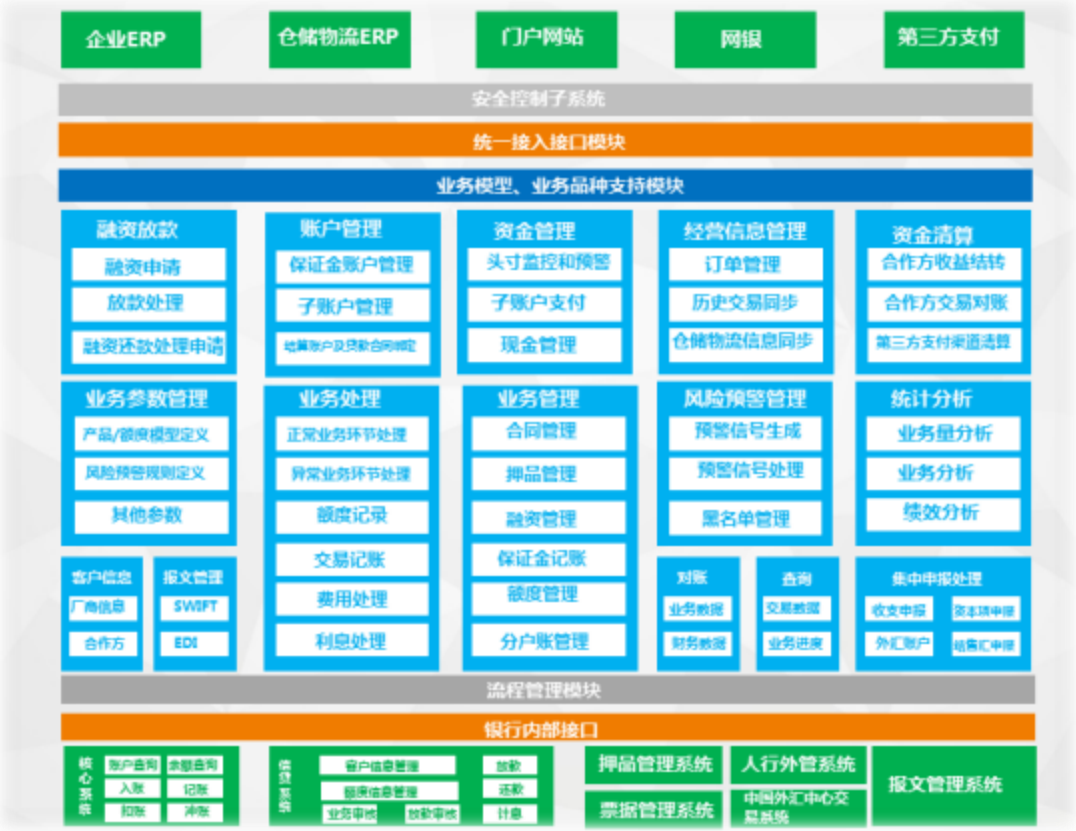
4.8 内存数据库

供应链平台考虑分布式可扩展设计和关系型数据库性能，引入内存数据库 Redis。

1) Redis 是一个 key-value 存储系统，会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件，并且在此基础上实现了 master-slave（主从）同步。
2) session、用户基本信息，token、字典表缓存等数据存放于内存数据库。供应链平台可直接分布式部署，不需要修改程序代码
3) 银行流水号生成、简单 key-value 表设计直接存放于内存数据库，可大大提供后台数据库性能。

4.9 丰富的业务模块

4.9.1 业务系统架构



1） 供应链系统作为供应链融资业务的整体框架，通过串接前、中、后台各系统，实现应收、预付、存货等供应链融资产品的全流程一站式服务。

2） 系统通过统一接入接口，与企业 ERP、物流仓储 ERP、网银、门户网站对接，使供应链上的各类客户可上述渠道统一接入在线供应链系统完成所有操作，平台提供整体服务（融资申请、协议签订、还款等）。

3） 供应链系统中通过行内系统接口，结合贵行现有行内系统的建设与规划情况，整合票据、贷款、保理、国结等业务，协同提供供应链融资的各类服务。

4） 另外系统还提供行内人员操作平台，以响应各类供应链交易的处理申请和进行供应链业务的运维管理工作。

4.9.2业务模块概述

供应链平台功能涵盖预付类业务、存货类业务和应收类业务三大类。详细包含的业务模块如下表（不仅限于以下业务，支持各种定制业务）：

业务大类	业务模块
应收类业务	国内保理
	国际保理
	信保保理
	租赁保理
	担保保理
	应收账款池融资
预付类业务	保兑仓
	先票（款）后货
	特色行业
存货类业务	动态动产质押融资
	静态动产质押融资

1） 国内保理

包括国内标准卖方保理、国内买方保理、买方协议付息保理、租赁保理、保理池、信用保险保理、买断保理、国内双保理。是指卖方（国内供应商）将其与买方（债务人）订立的销售合同所产生的应收账款转让给保理商，由保理商为其提供贸易融资、销售分户账管理、应收账款的催收、信用风险控制与坏账担保等综合性金融服务。

2） 租赁保理

是指出租人与承租人形成融资租赁关系的前提下，出租人将融资租赁合同项下未到期应收租赁款债权转让给银行，由银行作为租赁款债权受让人收取租金，并向出租人提供融资及商业资信调查、应收租赁款管理的综合性金融服务。

3） 信保保理

企业在投保信用保险的前提下，将国内贸易中形成的应收账款转让给银行，

由银行提供应收账款融资、账务管理、账款催收、承担坏账风险等一项或多项的综合金融服务。实质是信保融资模式是银行运用保险增级技术对供应商已投保的应收账款办理的融资。

4） 担保保理

银行受让应收账款，由卖方之外的法人或自然人（第三方）对买方付款向银行作连带责任担保，三方签订“担保协议”，约定如果买方到期未付款或未足额付款，第三方向行方履行担保义务；如果发生争议或第三方拒绝履行担保义务，则由卖方无条件回购应收账款的金融服务。

5） 国际保理

适用于以赊销（O/A）、承兑交单（D/A）等非信用证结算方式的出口贸易，同时，进口保理商提供应收账款的催收及信用风险控制与坏账担保等服务，适用于有应收账款融资、管理需求或优化报表需求的国内出口卖方企业。同时境外买方的商业信用和付款能力符合银行要求。

6） 应收账款池融资

是指将一个或多个国内的不同买方、不同期限和金额的应收账款全部一次性转让给银行、银行根据累积的应收账款余额给予融资。

7） 保兑仓

保兑仓是先票款后货系统的变种，它是在客户交纳一定保证金的前提下，银行贷出全额货款供买方客户向卖方采购，卖方出具全额提单作为授信的抵质押物。随后，买方客户分次向银行提交提货保证金，银行再分次通知卖方向客户发货。卖方就发货不足部分的价值承担向银行的退款责任。

8） 先票（款）后货

是指符合银行条件的经销商（买方）向银行申请承兑银行承兑汇票或者贷款，专项用于向特定的生产商（卖方）购买特定货物的授信业务。生产商、经销商、监管公司、银行通过之间的协议约定各自的权利义务。生产商收到银行承兑汇票或者货款后，以银行为收货人，将货物直接发至银行指定的地点（监管公司仓库），并将货物作为质押担保。经销商以补充保证金的形式或者还款形式向银行申请提

取相应货物，直至对应保证金余额达到银行承兑汇票票面金额或者还完银行这笔贷款，银行则将全部货物移交完毕。

9） 静态动产质押

静态动产质押是动产以及货权抵质押授信业务最基础的系统，它是指客户以自有或第三人合法拥有的动产作为抵质押的授信业务。银行委托第三方物流公司对客户提供的抵质押商品进行监管。抵质押物不允许以货易货，客户必须打款赎货。

10）动态动产质押

它是静态动产质押的延伸系统，指客户以自有或第三人合法拥有的动产为抵质押物得授信业务。银行对客户抵质押的商品价值设定最低限额，允许在限额以上的商品出库，客户可以以货易货。

5. 平台接口规范

5.1 系统接口架构说明

5.1.1 安全性保证

1) 通过互联网端接入，必须保证数据的安全性，防止数据泄漏
2) 采用私钥加密，公钥解密的方式，保证数据安全性
3) 线上交易全部采用电子签名，使用 key 对数据进行加签

5.1.2 可靠性保证

1) 通过互联网传输，必须保证数据可靠性，防止数据被篡改
2) 传输过程中采用 MD5 加密算法对报文进行摘要，保证数据可靠性

5.1.3 兼容性

1) 方案必须兼容银行或金融机构的成熟对接技术
2) 采用 restful api 方式对外提供服务
3) 支持的消息格式：
   - SOAP/XML
   - JSON
   - TXT
   - EXCEL
4) 支持的协议：
   - Socket
   - SFTP
   - https
   - 其他扩展协议

5.1.4 扩展性

1） 为供应链平台以后的规划预留空间

2） 成熟的平台，专业的供应链业务沉淀

5.2 对接案例

1） 供应链平台与会计核心系统的集成

a） 与会计核心系统的接口方式：实时交易通过 esb 系统采用 TCP/IP 协议同步 Socket 的接口方式，日终对账批量文件采用通用文件传输的接口方式。

b） 与会计核心系统的接口定义：

- 上传账务信息：每次交易都在主管放行的时候，将该笔交易的账务上传到会　计核心系统。
- 查询账户余额信息：根据指定的账号查询该账号的余额信息。以便在做扣款交易前查询余额后判断是否可做该交易。
- 查询账户明细：当银行稽核或出现一些异常状况时，需要查询账户的交易明细，根据账号获取该账户在指定时间段内的交易明细。
- 批次上传账务信息：日终时将表外科目的账务和利息提列的账务信息采用通用文件传输方式上传至会计核心系统。
- 批次下载对账信息：每日换日时采用通用文件传输方式从核心服务器上下载前一工作日的对账信息，以便分析供应链金融平台系统中的账务是否与会计核心系统记账的内容是否匹配。
- 批次下载汇率信息：每日换日时采用通用文件传输方式下载各种币别的兑换率。

c） 需要会计核心系统配合的事项：

- 定义各实时交易的交易接口字段；
- 定义每日对账文件的格式。

2) 供应链平台与信贷系统的集成

a) 与信贷系统的接口方式：实时交易通过 esb 系统采用 TCP/IP 协议同步 Socket 的接口方式，日终对账批量文件采用通用文件传输的接口方式。

b) 与信贷系统的接口定义：

- 客户信息的获取：为标准化客户信息，通过客户在行内的唯一编号，从信贷系统获取客户的信息。
- 客户额度信息的获取：根据信贷部门审批的额度编号，从系统获取客户的额度信息并下载保存到供应链平台数据库中。
- 上传交易额度使用状况：当授信拨款放行时，录入授信编号并根据授信编号将额度的使用状况发送到系统，由系统判断额度的动用是否在允许的范围内。
- 授信信息的批次上传：每日【日终】时将授信信息按授信编号汇总存入到电子文档中，采用通用文件传输方式上传到信贷系统指定的服务器目录中。

c) 需要信贷系统配合的事项：

- 定义获取客户信息的报文格式。
- 定义获取客户额度信息的报文格式。
- 定义上传交易额度使用状况的报文格式。
- 定义授信信息批次上传的电子文件格式。

3) 供应链平台与押品管理系统的集成

a) 与押品管理系统的接口方式：实时交易通过 esb 系统采用 TCP/IP 协议同步 Socket 的接口方式，日终对账批量文件采用通用文件传输的接口方式。

b) 与押品管理系统的接口定义：

- 押品登记：供应链平台新增质押品时，调用押品登记接口将押品相关信息传输至该系统中，该系统则返回生成的押品编号。
- 押品删除：根据系统返回押品编号，调用删除接口，删除押品信息。
- 押品修改：根据系统返回押品编号，调用修改接口，修改押品信息。

c) 需要押品管理系统配合的事项：

- 押品登记的报文格式。
- 押品修改的报文格式。
- 押品删除的报文格式。

4) 供应链平台与贷后系统的集成

a) 与贷后系统的接口方式：实时交易通过 esb 系统采用 TCP/IP 协议同步 Socket 的接口方式，日终对账批量文件采用通用文件传输的接口方式。

b) 与贷后系统的接口定义：

- 贷后通知的获取：供应链平台定时从贷后系统获取相关贷后通知。
- 贷后管理：供应链平台在进行贷后管理时，调用贷后管理接口，通知贷后系统登记相关信息。

5) 供应链平台与数据平台的集成

a) 与数据平台的接口方式：实时交易通过 esb 系统采用 TCP/IP 协议同步 Socket 的接口方式，日终对账批量文件采用通用文件传输的接口方式。

b) 与数据平台的接口定义：

- 数据平台信息的获取：获取数据平台的相关数据信息，并依据信息要求做管理和备份。
- 数据平台信息的管理：在供应链平台与数据平台中将储存基于业务交易的批次号，并且同时管理该批次交易所关联的所有相关数据信息的唯一 ID 号码便于后续管理。

6) 供应链平台与影像系统的集成

a) 与影像系统的接口方式：实时交易通过 esb 系统采用 TCP/IP 协议同步 Socket 的接口方式，日终对账批量文件采用通用文件传输的接口方式。

b) 与影像系统的集成方式

供应链平台与影像系统可采用两种集成方式：

- 使用影像系统提供的浏览器插件集成，此种方式可以利用浏览器插件做到对影像文件进行上传、删除、预览、批注，用户体验更高。
- 供应链平台将影像文件传输至供应链平台应用服务器，再通过通用文件传入方式将文件传至影像系统，再通过 ESB 调用添加影像接口

通知影像系统登记并返回影像编号。

c） 与影像系统的接口定义：

- 影像文件的上传：使用通用文件传输方式，将影像文件传输至影像服务器
- 影像添加通知：调用接口，通知影像系统新添加影像文件的路径等相关信息，影像系统返回该文件在影像系统的编号供后续管理。

6. 软硬件配置建议

6.1 物理架构

1. 系统支持集群环境，可随业务量的增长，横向扩展服务器，加上集群技术即可无缝进行硬件扩容。目前阶段考虑性价比，在满足目前业务量和 5 年发展预估的前提下。我们建议先采用 **4 台应用服务器，2 台数据库服务器、1 台备份服务器的配置。**

2. 应用服务器接收客户端请求，将请求封装调用服务器的相应服务，并从数据 库服务器读取数据，对数据进行处理和运算，将结果返回至客户端，或将结果提交到数据库服务器保存。

3. 应用服务器支持横向扩展，可在业务量增加的情况下，利用集群扩展服务器连，主要负责本系统的应用页面的展现和交互数据的展现、收集和接受请求和逻辑运算和数据处理的工作。

4. 数据库服务器也支持利用集群横向扩展，主要负载本系统的数据存储和读取的工作。

5. 备份服务器采用与应用服务器、数据库服务器相同配置，备份应用系统环境和数据库环境。在应用服务器或数据库服务器宕机时，暂时替代运行的服务器进行工作。

6. 在集群的环境下，数据库需连接磁盘阵列，系统的数据保存于磁盘阵列中。

7. 本系统可能会与其他的外围系统进行信息传递或协同工作。与其他外围系统的交互都将由应用服务器向其他外围系统发送请求，或由其他外围系统调用应用服务器的服务。

6.2 软硬件方案

资源名称	硬件配置	软件配置	备注

应用服务器*2	x86 服务器 处理器：至强 E5-2600 v2 * 2 内存：16GB 硬盘：1TB*1，支持热插拔	OS：centOS； 中间件： Weblogic 12	处理所有后台业务逻辑； 处理管理请求； 采用双机负载或热备；
数据库服务器 *2	x86 服务器 处理器：至强 E5-2600 v2 * 2 内存：16GB 硬盘：1TB*2，支持热插拔	OS：centOS； 软件： Mysql 5.0	处理系统数据存储； 如有公共存储可以忽略此配置；
代理服务器	无限制	OS：centOS； 软件：Tengine	处理请求分发和双机热备；
消息中间件	参考现有标准规范	参考现有标准规范	负责与其他系统接口对接；

7. 安全方案

7.1 物理安全架构

物理安全策略主要涉及运行环境、硬件设备、媒体安全三个方面，是对供应链平台开发项目涉及的系统所在环境、所用设备、所涉媒体进行安全保护。这三部分须遵循银行机房设计安全要求。

7.2 网络安全策略

网络安全主要考虑访问安全控制、操作系统、防病毒、防黑客攻击、数据传输加密、网络安全审计、安全响应及应急处理。网络安全主要考虑局域网安全和广域网安全，通过访问控制、操作系统安全、防病毒系统、防火墙系统、安全审计、网络层数据加密、安全响应和处理等实现。

7.2.1 访问安全控制

对于局域网内用户主要通过划分不同网段和 VLAN 的方法控制访问安全，对于数据存储区域应该通过安全审计等控制手段进行访问控制；对于广域网用户应该通过访问控制列表和访问口令进行控制。

7.2.2 安全审计

通过安全审计系统对安全事件进行快速评估并响应。审计评估的目的是根据网络的报警记录、日志信息及其他信息向管理员提供各种问题统计和分析，在审计评估后形成审计报告，达到综合评估网络安全现状的目的。

7.2.3 防病毒系统

现在计算机病毒种类很多，而且危害程度也越来越大，轻者影响系统的运行

性能，重者破坏整个系统数据，使系统陷于瘫痪，所以无论是在客户端还是在服务器端都应该安装安全可靠的软硬件杀病毒、防病毒产品。

7.2.4 防火墙系统

为实现网络之间安全隔离，防止非法攻击对网络中计算机软硬件的损害，采用防火墙系统进行安全隔离，对进出本系统内部网络的外部信息进行过滤，管理进、出系统网络的访问行为，封堵某些禁止的动作，记录通过防火墙的信息内容和活动，对网络攻击进行检测和告警。

7.2.5 数据传输加密

为保证传输数据的安全，对传输数据进行加密。如采用 SSL 加密协议对传输数据加密。

7.2.6 安全响应和处理

通过网络安全扫描系统和网络实时监控预警系统，对系统安全事件及时做出响应和处理。

网络安全扫描系统主要作用是应用黑客攻击技术，以实践性的方法对网络主要部件（包括各类重要应用服务器，数据库，防火墙，交换机，路由器等等）进行安全性扫描分析，发现网络中安全弱点及缺陷，评估其安全风险，并提出专家性的修正意见，使网络管理人员对网络安全状况了如指掌，为修正网络安全策略提供可靠的依据。

网络实时监控预警系统是通过实时分析网络数据流，寻找网络违规模式和未授权的网络访问尝试。当发现网络违规模式和未授权的网络访问时，能根据系统安全策略做出反映，包括实时报警、事件登录、自动阻断通信连接或执行用户自定义的安全策略等。

7.3 操作系统安全策略

操作系统是应用软件和服务运行的公共平台，操作系统安全会影响整个系统

的运行。通过访问用户授权来控制操作系统的安全。使用的操作系统安全标准必须达到C2级标准。

7.4 应用服务器安全策略

为保证应用系统稳定运行，可采用两台服务器，一台做主机一台做备机，将应用分别部署到主机和备机服务器上，两台服务器采用双机热备。当主机服务器出现故障导致应用服务器宕机时，热备软件自动将地址飘移至备机服务器，启动备机服务器。

7.5 应用系统安全策略

7.5.1 用户账户管理

包括用户代码、名称、密码、密码效期、密码验证次数等信息。用户登录时必须键入用户名及口令来证明其是否是合法的用户，只有合法用户才能登录到应用系统。系统支持统一用户认证机制。如果未来行方采用统一用户认证体系，信雅达友田公司积极配合实现统一用户认证，提供统一用户认证接口。如果用户通过统一认证系统登录，在国际结算系统中不进行用户账户检验，只根据用户权限展示用户功能。信雅达友田公司已在其他客户实现了统一用户认证。

7.5.2 用户权限管理

公司系统提供机构管理、用户管理、机构权限、用户权限、角色、角色权限等一系列访问权限控制功能。隶属某机构的用户权限不能超过本机构最大权限。支持角色定义，支持为某一角色分配权限。用户关联某角色时自动继承该角色权限。不同用户可拥有不同权限，用户登录时系统根据用户权限显示不同功能菜单和队列。

7.5.3 电子日志管理

系统建立电子日志，对登录系统的用户任何操作进行登记 ，便于日后审计监督。电子日志包括用户登录日志、交易日志、中间件提供的日志。其中登录日志记录用户代码、用户名称、登录时间等；交易日志详细记录用户在系统中的所有操作。

7.5.4 数据加密

公司系统对于重要的业务数据及密码，系统加密后存储，防止数据泄漏。主要涉及密码加密、影像数据加密、swift 报文数据加密。密码采用 DES 算法加密；影像文件采用 MD5 算法加密；Swift 报文采用 SA2 加密算法，将报文文件加密传输；业务数据传输采用 SSl 加密。

7.5.5 记录加锁机制

支持记录加锁，防止业务数据被多人同时使用及数据库脏读。

7.5.6 风险控制机制

系统提供多种业务风险控制，包括业务授权、高风险业务通过风控模型、额度管理、实时监控等功能。

8. 系统性能设计

8.1 系统性能指标

8.1.1 系统运行性能

1) 系统支持 7×24 小时运行
2) 客户一般操作响应时间不超过 3 秒
3) 批处理不中断正常交易
4) 批处理时间不高于 1 小时
5) 系统整体备份时间不高于 2 小时
6) 系统恢复时间不高于 1.5 小时

8.1.2 系统稳定性

1) 系统可使用率保持在总运行时间的 99% 。
2) 系统连续稳定运行时间不少于 12 个月。
3) 交易成功率：99.999%

8.1.3 系统并发性

系统并发数估算：

假设银行最大在线用户数量为 1000 人。

平均的并发用户数： C = nL/T

即：平均并发数＝总用户数*用户在线时长/总工作时间

并发用户数峰值： C’ ≈ C+3*SQRT（C）

即：峰值并发数＝平均并发数+3*（平均并发数^1/2）

按以上公式，按最大用户数 1000，用户在线时长 4 小时，每天工作 8 小时得出并发用户数为 500，峰值用户约 560。

8.1.4 系统可维护性

公司关于编程有明确的代码规范，其中包括注释要求，我们能够满足贵行的“注释与源代码比例至少为 1:2”要求。

关于系统可维护性信雅达友田公司采用参数化策略维护，提供参数化二次开发，通过配置参数方式实现业务需求，尽量更改后台代码，这种维护方式方便升级和日常维护。

8.2 性能指标实现依据

1) 采用索引技术，建立数据库表的适当的索引，加快查询速度。
2) 采用反范式设计，适当的反范式设计可以减少多表联合查询的情况，有利于优化查询。
3) 采用优化的 SQL 语句，基于合理利用已有的索引、避免使用函数等策略来优化 SQL 语句，往往会使数据查询速递提高至少一个数量级。
4) 采用数据库连接池技术。数据库访问时建立数据库连接需要 1 秒左右，系统存在大量的数据访问，建立数据库连接将极大的降低访问速度。采用数据库链接池技术可以有效解决这一问题。
5) 采用高效的数据库操作技术，充分利用 SQL 高级特性。
6) 采用数据对象缓存技术。对常用数据（如数据参数、机构、产品等）采用缓存技术，避免访问数据库，从而有效提高数据访问性能。
7) 采用高效的算法。在业务数据处理方面采用高效的算法，降低处理的时间复杂性和空间复杂性，加快运算速度。
8) 在大数据量查询方面，一页可以显示的行数以及一次可装入内存的页数都可参数化配置。这种折中数据库访问次数和内存占用的策略，可以根据实际部署环境进行调优，以提高系统的综合性能；
9) 公司提供历史数据归档功能，支持归档条件定义，如设定归档数据时间、归档数据状态、按机构归档等。归档条件定义后系统按照归档条件抽取数据，生成归档文件，并清理已归档记录的相关信息，同时建立归档索引，记录归档内容索引，便于归档后查询。如果想恢复归档数据，可通过归档索引进行

恢复，也可以将归档文件恢复到其他数据库中。

8.3 系统性能说明

支持大数据量数据的快速业务处理和数据交互。并发用户数量要能够满足业务发展需求，不少于 1000 人同时在线作业，超过 1000 人同时在线作业时应有相关机制控制不影响原有作业的处理，针对未来业务量变化需支持在线作业的业务量扩展。对于访问超时需有自动链接释放机制。

1） 系统性能测试报告

a） 操作响应时间如下表:

数据库中发票量（笔）	响应时间
100,000 以下	1 秒以内
100,000 - 500,000	1 - 2 秒
500,000 以上	2 - 3 秒

b） 报表响应时间如下表:

报表类型	响应时间
凭证与传票类报表	2 秒以内
管理类报表	5 秒以内
统计类报表	5 秒以内
通知或回单	2 秒以内

c） 查询响应时间如下表:

数据库中发票量（笔）	响应时间
100,000 以下	1 秒以内
100,000-300,000	2 秒以内
300,000-500,000	2 秒以内
500,000 以上	3 秒以内

d） 批处理所需时间如下表:

数据库中的发票量（笔）	响应时间
10,000	50 秒以内
10,000 - 100,000	6 分钟以内
100,000 - 500,000	18 分钟以内
500,000 以上	30 分钟以内

e） 系统用户承载能力

最大用户数	最大同步用户数
不限	105

2） 系统平均初始化时间(服务器重新启动时间)：32 秒

3） 系统瓶颈：

■ 最费时间和资源的过程为：日终作业。

■ 最费时间和资源的过程的原因：日终需要处理大量的数据，并且因为要根据一定规则将数据缓存在应用服务器端在一个事务环境中一次提交。因此需要占用大量的资源并花费比较多的时间。

8.4 运维管理方案

本系统具有一个后台运营维护管理工具。并为该工具设定了用户权限。运营维护管理工具仅限本系统运营过程中的日志、资源和全行公共参数的维护。其主要包括如下功能：

8.4.1 操作日志查询和监控

系统将记录各用户从登录到退出的所有操作日志，记录的内容包括用户名、IP 地址、发出请求的类型、发出请求的时间等要素。可以实时查询和监控各用户的操作历程。

8.4.2 异常日志查询和监控

系统内部产生的异常将单独记录，在出现未被正常捕获并自动解析处理的异

常时，便于查看和定位、分析异常产生的原因，为解决异常提供依据。

8.4.3 接口日志查询和监控

系统所有与外围系统的报文传递，均单独记录，日志记录报文的类型编号、对方服务器的 IP 地址、使用的端口、报文的发出时间、报文的内容、响应的时间、响应的报文内容等要素。便于监控和调试接口处理。

8.4.4 批量处理日志查询和监控

记录系统提供的批量作业服务运行的启动时间、批量作业各任务的完成状态（正常完成、出现异常）、异常的原因、批量作业的结束时间等信息。运营管理人员可制定《日常运营检查表》，定义批量作业的检查次数和周期，每日在检查时间检查批量作业日志，了解批量作业执行情况，发现异常可及时通知相关人员进行处理。

8.4.5 系统资源监控

监控应用服务器的 CPU、内存、硬盘空间占用情况。可设定不同层级的预警阀值，当出现占用率到达警告阀值时，系统将用黄灯提醒运营人员关注。当占用率到达危险阀值时，系统将用红灯提醒运营人员进行问题的核查和必要的处理。运维人员制定《日常运营检查表》定义系统资源的检查次数和周期，每日在计划时间点检查系统资源占用情况。并加以记录。

8.4.6 运营管理策略

1) 建议使用《日常维护检查表》，在表内制定维护监控服务器的内容（包括各类日志、资源），并制定每日检查的时间，在规定的时间运维专人，使用运维管理工具进行核查，并记录当时的各项检查项的情况，并签名确认。
2) 建议使用 Linux 系统，因 Linux 系统在操作系统层面漏洞较少，可提高服务器操作系统层面的安全性。如果必须要使用 Windows Server 作为操作系统，应停止共享目录服务，不提供共享目录。文件的传输可使用 FTP。

3) 本系统不提供除批处理外的作业调度，批处理使用 CTM 平台，统一调度管理。

4) 系统的运维管理工具提供权限管理的相关功能，可设置账号密码策略，根据参数化设置进行密码复杂度的管理。

5) 变更和投产遵循贵行变更和投产的流程规范：一般采用如下流程

■ 在变更和投产前提交《变更（投产）申请》，并整理《变更（投产）的操作步骤》，准备好变更（投产）的数据或文件。

■ 申请审核通过后，由贵行专人在我司的技术人员陪同下，按《变更（投产）的操作步骤》进行变更（投产）活动。我司技术人员在实际变更（投产）活动中尽量不进行操作，不允许贵行技术人员向我司技术人员透露投产使用的账户和密码信息。

■ 变更（投产）过程中，需按同样步骤更新备份服务器上的应用。

■ 投产完毕填写《变更（投产）报告》记录投产的操作和结果。

6) 如系统需利用负载均衡技术进行横向扩展时，可利用贵行现有负载均衡平台。无需另外建立负载均衡架构，并保持生产环境和测试环境负载均衡的一致性。

7) 每周日对系统进行零级备份。每周一到周六在零级备份的基础上进行增量备份。在备份时，备份介质应不少于两份，备份完成后，应分开保存。备份数据的保存周期可根据情况保存 7-30 天，备份介质在容许范围内可循环使用. 备份采用数据库的代理进行自动处理。运维人员需在计划的时间点检查备份文件是否产生，如果产生则将备份文件复制到磁带或光盘中，甚至可将复制的备份文件运送到异地保存，以做好容灾措施。

8) 应用系统的安装和启停需设立专用账号，并由贵行科技部人员保管。在安装、启停过程中，由贵行人员进行操作，或由贵行人员登录账号后，由我司技术人员进行操作。

9) 系统提供应用系统各类数据的查看视图，原则上无需应用人员直接登录数据库服务器进行查看。仅在版本更新或数据更新时可提供专用账号登录数据库，进行相应的操作。登录的方式仍采用由贵行人员登录账号后，由我司技术人员进行操作

应用系统的数据抽取，系统提供相应的接口或服务，在应用系统层面完成数据的

抽取工作，不提供直接 DBLink 抽取数据的方式。