# 企业间票据撮合平台

# 架构说明书

**时间:2018.11.11**

目录

**[一、引言](#_Toc10093_WPSOffice_Level1)** **[2](#_Toc10093_WPSOffice_Level1)**

[1.1目的](#_Toc6396_WPSOffice_Level2) [3](#_Toc6396_WPSOffice_Level2)

[1.2范围](#_Toc20723_WPSOffice_Level2) [3](#_Toc20723_WPSOffice_Level2)

[1.3参考文献](#_Toc10680_WPSOffice_Level2) [3](#_Toc10680_WPSOffice_Level2)

[1.4名词解释](#_Toc17057_WPSOffice_Level2) [3](#_Toc17057_WPSOffice_Level2)

**[二、软件系统架构设计概述](#_Toc6396_WPSOffice_Level1)** **[3](#_Toc6396_WPSOffice_Level1)**

[2.1软件系统架构设计策略与原则](#_Toc24234_WPSOffice_Level2) [3](#_Toc24234_WPSOffice_Level2)

[2.2关键功能性需求](#_Toc9460_WPSOffice_Level2) [4](#_Toc9460_WPSOffice_Level2)

**[三、软件系统架构设计](#_Toc20723_WPSOffice_Level1)** **[4](#_Toc20723_WPSOffice_Level1)**

[3.1整体架构](#_Toc22231_WPSOffice_Level2) [4](#_Toc22231_WPSOffice_Level2)

[3.2逻辑架构](#_Toc30485_WPSOffice_Level2) [5](#_Toc30485_WPSOffice_Level2)

[3.3接口设计](#_Toc26649_WPSOffice_Level2) [6](#_Toc26649_WPSOffice_Level2)

[3.4数据架构](#_Toc1413_WPSOffice_Level2) [6](#_Toc1413_WPSOffice_Level2)

[3.5技术架构](#_Toc27338_WPSOffice_Level2) [6](#_Toc27338_WPSOffice_Level2)

[3.4部署架构](#_Toc17145_WPSOffice_Level2) [6](#_Toc17145_WPSOffice_Level2)

[3.6非功能需求及解决方案](#_Toc5819_WPSOffice_Level2) [8](#_Toc5819_WPSOffice_Level2)

[3.7日志机制（log）](#_Toc1975_WPSOffice_Level2) [8](#_Toc1975_WPSOffice_Level2)

[3.8实体持久化](#_Toc27372_WPSOffice_Level2) [9](#_Toc27372_WPSOffice_Level2)

[3.9系统层次结构模型](#_Toc28160_WPSOffice_Level2) [9](#_Toc28160_WPSOffice_Level2)

## 一、引言

1.1目的

本文档全面与系统得表述目标软件系统的架构，并通过多种视图从不同角度描述系统的各个主要方面，从而满足客户、设计人员对系统的不同需求。本文档记录并表述了架构师对系统架构方面做出的重要决策；项目经理将根据架构定义的构件结构制定项目的开发计划；设计员根据此进行构件的详细设计；测试人员按照构架设计系统的总体测试框架；另外构架文档还用于指导各构件的实施、集成及测试。

1.2范围

本文档适用于“企业间票据撮合平台”的总体应用框架。

1.3参考文献

## a《**软件架构设计-程序员向架构师转型**》

## b《**一线架构师实践指南**》

1.4名词解释

## 二、软件系统架构设计概述

2.1软件系统架构设计策略与原则

* 抽象原则：各平台(含基础设施、中间件技术服务、各层业务服务等)需要通过合理地抽象，将内部信息、处理与扩展能力聚合成标准的服务于扩展接口，并通过统一的形式提供给使用者，屏蔽内部的实现与运行细节。
* **共享原则**：最大化重用数据、计算资源、业务组件等资产，防止数据、逻辑与技术实现不一致性带来的管理复杂性，避免重复建设成本与管理成本，通过安全机制保证共享资产的合法使用，通过业务分级保障共享资源效益最大化。
* **自治原则**：每一个组件（计算资源、业务组件、信息实体等）具备最大可能的自我完备性，可独立运行、监控、部署、配置与禁用，具备确定的SLA，并与其它组件之间以松散耦合的方式进行协作。当依赖的组件不存在或者无法正常提供服务时，能够以良好的方式降级，且在故障解除后自动恢复。
* **冗余原则**:各组件（计算资源、业务组件、数据等）都必须有充分、合理的冗余实例，保证单一组件实例失效不影响业务正常运行（多活/热备），或可以通过切换备份实例快速恢复（温备/冷备），不会丢失不可恢复的数据。针对不同类型的组件，需要明确定义冗余量与冗余类型。
* **分布原则：**整个系统拆分成职责清晰、粒度恰当、便于管理的组件，各组件（计算资源、业务组件、数据等）可分布部署运行。组件的拆分与分布可以采取复制、根据功能垂直拆分、或根据用户与访问模式水平拆分等形式。

2.2关键功能性需求

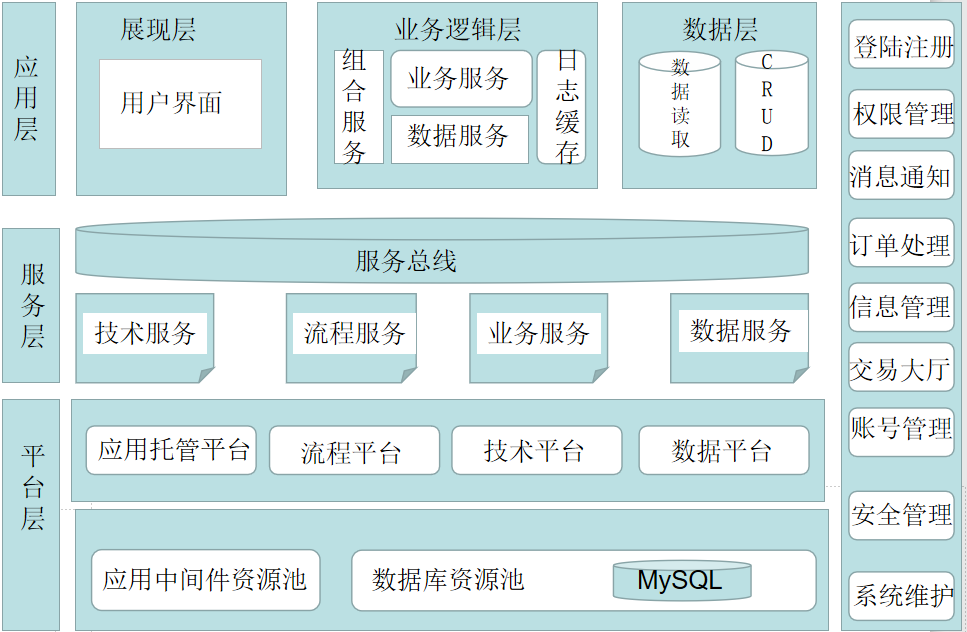
* 跨地域的系统外部用户通过Internet网来使用系统的功能。
* 内部用户、系统管理员在安全性较高的内网中使用系统的功能。
* 消息通知系统是目标系统为了实现相关功能而需要进行协作的一个外部系统，它能够向用户发送email，或者发送短消息。

## 三、软件系统架构设计

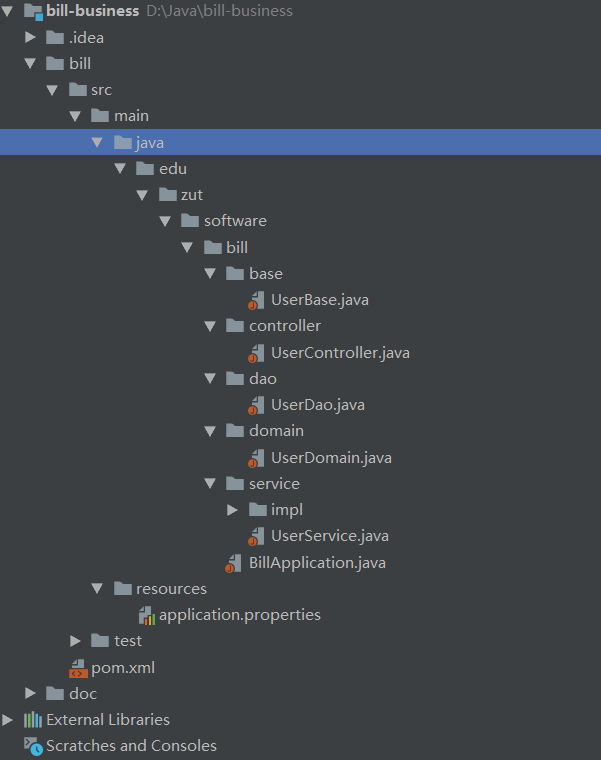
3.1整体架构

3.1.1概述：项目整体分为三个层次应用层、服务层、平台层，应用层包括展现层，提供用户界面，业务逻辑层对业务服务、数据服务等逻辑关系的封装，数据层提供数据源和数据的CRUD的实现。服务层通过服务总线与应用层进行交互提供技术、流程、业务数据等多种服务，平台层对项目的功能进行开发与维护。

项目的模块功能见整体架构图。

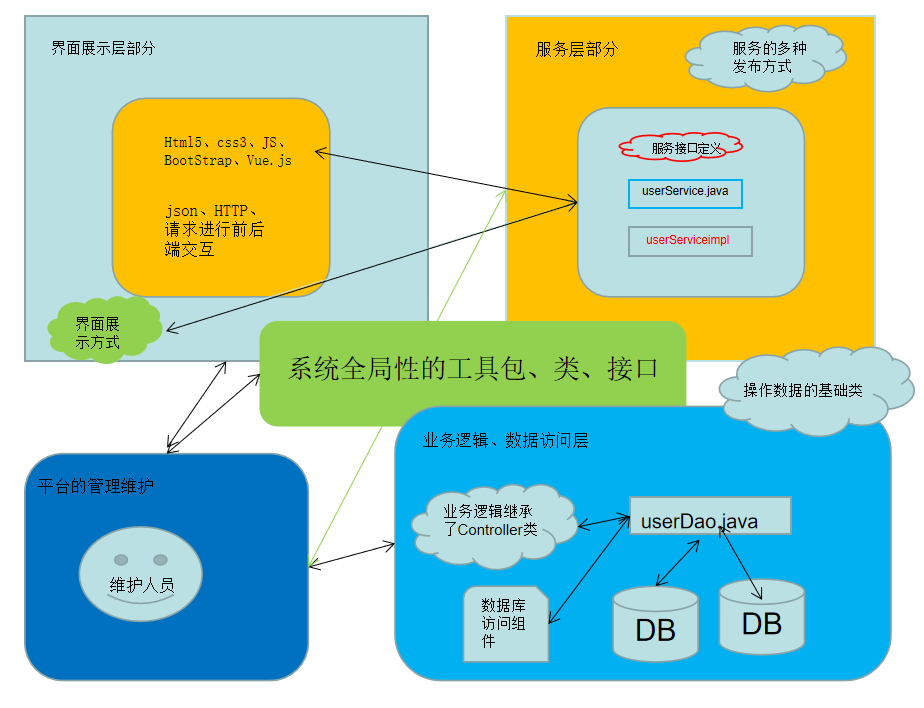


项目目录分层：项目具体分层如图，层次间逻辑关系通过类与类、注解与注解之间进行调用，清晰明了，为后续的测试维护，提供方便。



3.2逻辑架构

3.2.1概述：逻辑架构把系统分为若干个逻辑单元，分别实现模块功能，对系统的开发起到重要作用，并规定这些逻辑单元之间的交互接口和交互机制，识别功能块、规划功能块的接口、明确功能块之间的使用关系和使用机制。



3.2.1概述

逻辑视图从系统内在逻辑结构的角度描述系统的基本结构与动态行为，通常包括分析模型、设计模型、以及数据模型等。

设计模型说明了系统的组成元素、组织架构和关系，并描述了各组成元素的协作以及状态转换关系等。本节将分别在系统层次结构模型中描述系统的层次组织结构；在主要的包和子系统中说明系统的具体组成；并在架构机制中详述系统的各种构架机制；最后在关键的用例实现中通过描述最重要的用例实现，来说明构架的典型协作（动态行为）。

分析模型对等于设计模型，是在更高的抽象层次上定义系统的结构，作为可选项，本文档不再说明。

3.3接口设计

包括系统间的接口设计以及内部功能模块之间的接口设计；

3.4数据架构

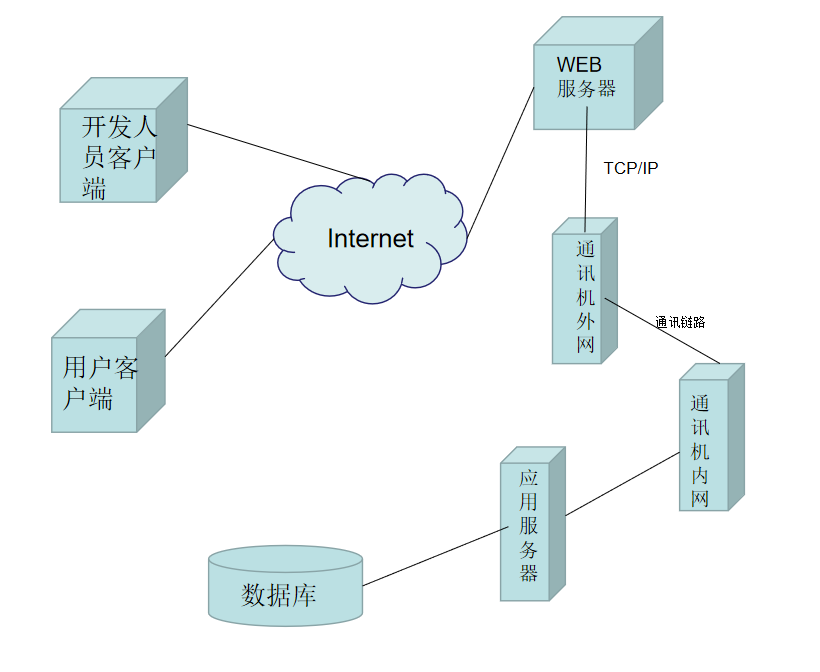
本系统与上下游系统间的数据流关系，以及本系统关键数据表设计、数据管理策略等；

3.5技术架构

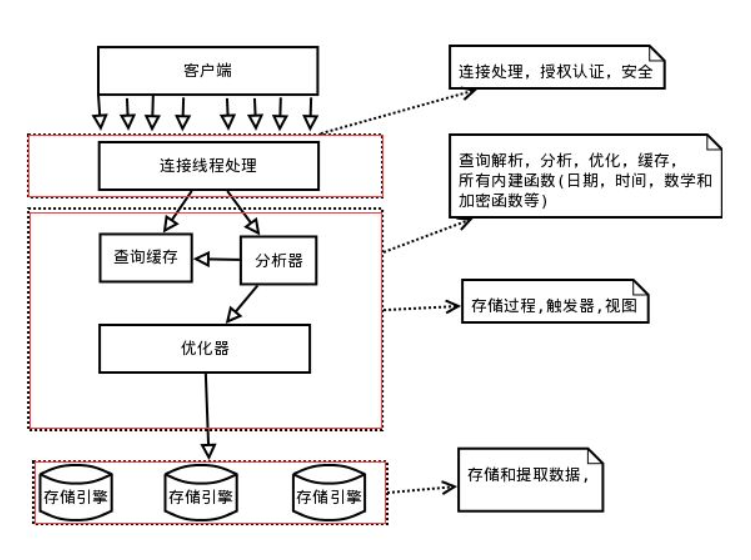
该项目主要使用当前流行前端框架，后端框架，实现架构，前端HTML5+Css3+JavaScript，进行客户端页面的实现，搭配Vue.js、BootStrap进行页面的渲染，从而增强用户的体验效果。后端框架使用SpringBoot搭建，该技术减少了项目结构的冗余，提高代码复用性，前后端完全的分离。减少了配置文件的书写，从而提高开发效率，加速项目的进程。数据库的选用，根据组员们的商讨采用轻量级数据库MySQL，容量也适合该项目（MySQL单表大约在2千万条记录（4G）下能够良好运行，经过数据库的优化后5千万条记录（10G）下运行良好。）同时采用Mybatis与SpringBoot整合操作数据库。需要开发人员有良好的编码能力，熟悉底层原理，从而规避风险，进行开发与维护

3.4部署架构

本图展示了节点以及节点之间的关系组成，系统运行时的结构，展示了硬件的配置及其软件如何部署到网络结构中，用来帮助理解分布式系统，可视化的描绘了系统操作过程，主要考虑如何把软件映射到硬件上。

3.5进程视图

进程视图从系统运行时刻的角度，描述系统划分为进程、线程的结构，及其动态关系。模型主要说明进程、线程的分类，系统构架敏感的主要边界类、控制类对象等在进程、线程中的分布，以及它们之间的创建、交互与消息通讯关系等。



3.6非功能需求及解决方案

非功能性需求：是指软件产品为满足用户业务需求而必须具有且除功能需求以外的特性，包括系统的性能、可靠性、可维护性、可扩充性和对技术和对业务的适应性等。

性能要求：要求系统能满足100个人同时使用，页面反应时间不能超过6秒；

可靠性： 系统能7×24小时连续运行，年非计划宕机时间不能高于8小时。要求能快速的部署，特别是在系统出现故障时，能够快速的切换到备用机。

3.7日志机制（log）

系统日志主要使用SLF4J+log back框架来实现，直接应用，用于记录调试信息，及审计功能



3.8实体持久化

3.9系统层次结构模型

本系统分为 个逻辑层：应用层（Application）、业务逻辑层（service）、业务实体层、资源层、控制层等。

* 应用层：本层主要包括数据库、中间等底层的服务，或者外部系统的直接接口等；这些构件通常不在项目中开发。
* 业务逻辑层：业务服务层以业务服务的方式组织系统中各种业务逻辑，将一些相关的内聚的业务逻辑组织在一起成为一个业务服务，使得业务逻辑可以以业务服务为单位进行重用和替换。
* 业务实体层：本层包含了系统中的各个业务实体，对实体进行持久化的功能也包含在本层。另外业务实体层还包含一些业务里面的常量定义。
* 控制层：Controller是Spring Boot里最基本的组件，他的作用是把用户提交来的请求通过对URL的匹配，分配个不同的接收器，再进行处理，然后向用户返回结果。他的重点就在于如何从HTTP请求中获得信息，提取参数，并分发给不同的处理服务。