# 技术架构

## 系统需求

### 高可用

业务系统的场景是实时的日常使用，系统要具备 7\*24 小时的连续服务的能力。系统在不短于一个月的连续正常负荷运行过程中，不发生系统响应性能下降、响应能力下降、资源占用显著增加等现象。

具备有效的错误诊断和恢复机制，以及主从应急方案，因突发事件导致系统崩溃后以保证有备用系统供切换使用。

系统运行方面，提供异常监控机制，随时监控及报告系统的运行状态。

### 高并发

对于日常操作及月底、年底等操作高峰时段，能够满足大量用户同时在线操作业务的需求，对响应时间、吞吐量、每秒查询数、并发用户数等各项指标要保持在正常水平。

### 安全性

除了保证网络安全、主机安全、数据库安全等基础设施安全外，要保证业务系统的安全、访问可控、数据加密等。

### 高效部署

对交付上线运行的系统，能够高效的部署到生成环境中，对底层环境依赖小，降低部署学习曲线。

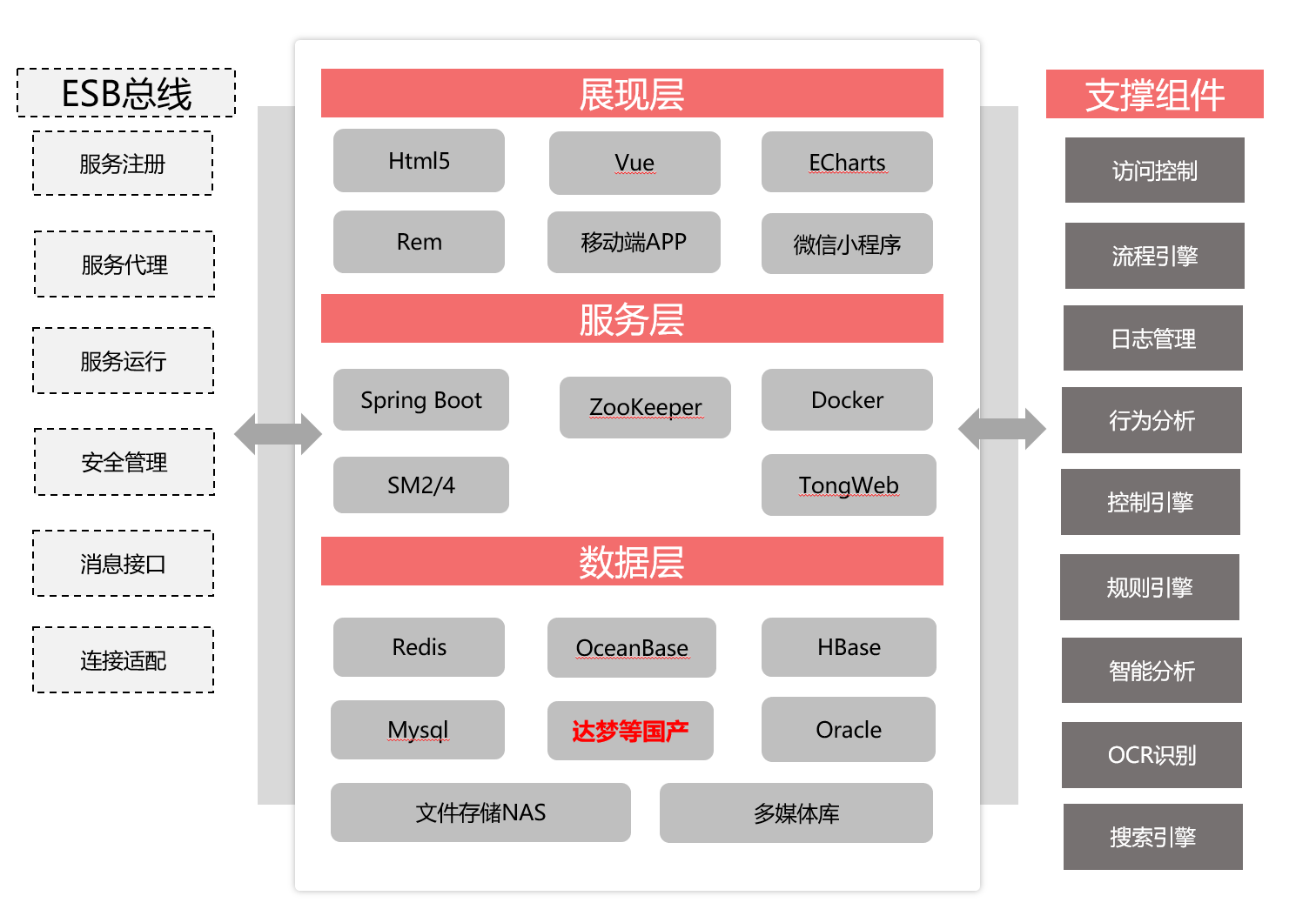
### 降低成本

对服务器及客户端硬件的基础配置要求低，降低硬件采购成本。尽可能使用可靠的开源软件，降低软件采购成本。

### 国产化

对国产数据库、国产中间件、国产操作系统要有较好支持。

## 总体架构



## 技术应用

后端采用广泛使用的Spring Boot微服务框架，以使编码、配置、部署、监控变得更简单。采用Spring Boot 框架的主要考虑因素：

(1) Spirng Boot遵循"习惯优于配置"原则，只需很少的配置就能快速搭建运行环境，能够极大的简化基于Spring MVC的Web应用和REST服务开发。

(2) 微服务架构倡导将功能拆分到离散的服务中，独立地进行部署，Spring Boot能够很方便地将应用打包成独立可运行的JAR包，并内嵌Tomcat、Netty等中间件，无需部署WAR包，可以直接启动运行，这种方式与微服务的理念非常一致。

(3) 直接预置了生成环境下的实用功能，比如度量性能指标、健康检查及安全、日记等扩展配置，方便对运行中应用的状态进行监控。

### 访问控制层

系统通过Web容器发布REST风格的Web API。其控制层是借助于使用Spring MVC架构来实现的。具体的功能有：

1） 发布Web API。本系统使用Spring MVC提供的RestController来实现REST风格Web API的发布。同时，对API的输入、输出，采用JSON数据格式规范，非常便捷。

资源路由管理。控制层对系统发布的Web API进行了规范的资源路由分配、管理，并配合Spring Security对路由进行了权限控制。

2） 身份认证。系统通过配置以及编写身份认证规则以及过滤器，将JWT的认证机制实现到了Spring Security中，包括：token的签发；token的验证；token的过期等。同时通过配置将Spring Security应用到Spring MVC框架中，保护发布的Web API。

3） 权限控制。本系统通过编写过滤器解析HTTP header中的token，可以对任何的Web API访问进行权限控制。解析的token中带有请求的客户端信息，配合资源路由的权限要求，可以限制没有权限的Web API访问。身份认证是基于Token的认证，除了能回避Cookie存在的问题，还有许多额外的优点：

无状态：由于Token已经自包含了所有的用户信息，所以无须再存储Session， 服务端易于扩展。

内容分发：易于实现CDN，将静态资源分布到CDN上，某些CDN服务商提供了基于Token验证的安全服务，服务器仅提供API。

解耦：无须被绑定在一个特定的验证方案，作为独立的会话系统，利于实现分布式。

基于标准：JWT作为Token的标准已经被广泛的接受。主流语言都有相应支持JWT标准的工具包。

### 服务层

服务层是本系统架构的重中之重。因为票据库服务票据存储吞吐量比较大，对服务可靠性要求高，服务层则必须保证高扩展性、高吞吐量、高可用性的业务要求。

为了满足业务要求，子系统的划分以及设计按照以下原则：

1） 所有子系统均可独立运行。每个子系统在部署时均可独立运行，不依赖其他子系统是否存在而启动。

2） 子系统均使用异步＋无阻塞的方式处理业务数据。每个子系统实例只是一个处理数据的过程，不涉及业务的状态。业务的处理结果使用数据层的持久化来保存；业务的上下文状态等，则使用数据层中的缓存技术来保存。

3） 子系统均可运行多份进行负载均衡以及容错，从而保证高吞吐量、高可靠性。

### 数据层

数据层着重考虑的两个指标：1、稳定可靠的持久化方式；2、高效率的数据访问。

在数据持久化方面，本系统选择Mybatis作为关系对象映射中间件。无论底层持久化数据库选择的是Mysql还是Oracle，本系统通过封装数据源并提供数据源会话控制工厂给Mybatis，都能便捷的完成对象关系映射。相对于Hibernate，Mybatis又可以修改sql语句进行性能调优。此外可使用Mybatis的两个插件：通用mapper插件；分页插件，来高效的进行数据层的开发。

### 工作流技术

工作流定义是：工作流是将一组任务组织起来以完成某个经营过程：定义了任务的触发顺序和触发条件，每个任务可以由一个或多个软件系统完成，也可以由一个或一组人完成，还可以由一个或多个人与软件系统协作完。

工作流管理系统的目标：管理工作的流程以确保工作在正确的时间被期望的人员所执行——在自动化进行的业务过程中插入任何的执行和干预。

系统采用了业界广泛使用的开源工作流引擎Activiti， Activiti的特点

1）.数据持久化

Activiti的设计思想是简洁、快速。有过应用开发经验的开发人员都知道应用的瓶颈体现在和数据库交换数据的过程中，针对这一点Activiti选择了使用MyBatis，从而可以通过最优的SQL语句执行Command，仅凭如此就能让引擎在速度上保持最高的性能。

2）引擎Service接口

Activiti引擎提供了七大Service接口，均通过ProcessEngine获取，并且支持链式API编程风格。

3）流程设计器

在jBPM4时代有专门的Eclipse插件可以用来设计jPDL，同样Activiti团队也专门设计了用来设计BPMN 2.0规范的流程设计器—Eclipse Designer。此外还有Signavio公司为Activiti定制的基于Web的Activiti Modeler流程设计器。

4）原生支持Spring

Activiti原生支持Spring，这一点对企业应用来说尤为重要：可以很轻松地进行Spring集成，非常方便管理事务和解析表达式（Expression）。

5）分离运行时与历史数据

Activiti继承自jBPM4，在表结构设计方面也遵循运行时与历史数据的分离，这样的设计可以快速读取运行时数据，仅当需要查询历史数据时再从专门的历史数据表中读取。这种设计方式可以大幅提高数据的存取效率，尤其是当数据日积月累时依然能够快速反应。

### Portal技术

为了更好的完成应用系统集成工作，采用Portal技术，快速创建和装配门户，为员工、合作伙伴和客户提供定制的应用程序、信息和业务流程的集成视图，同时加强了业务策略、业务流程和安全要求。它具备以下特征：

（1） 降低拥有成本，灵活的部署架构促使以最低成本实现门户价值最大化，集成能力可进一步扩展门户，以充分利用目前和未来的应用投资；

（2） 简化生产端到端的门户生命周期管理，可简化生产和管理；开发框架可扩展 Portal进行门户快速开发；智能化管理又简化和分散了门户的组装和管理；

（3） 加速业务解决方案的交付提交目前必需的各种业务服务，同时循序渐进地部署未来所需的业务服务。

## 安全设计

### 应用安全

账号管理

• 支持设置账户策略

• 支持设置密码策略

• 密码保护命令

多种认证机制

• PKI安全体系验证

• 建议购买SSL证书，为商城的机密数据提供加密传输功能，从而确保机密信息的机密性、完整性和不可否认

• 对包含用户密码、证件号码等敏感信息的页面如登录、会员中心、购物车等启用SSL数据传输加密安全通道，防用户隐私遭监视

业务操作控制

• 对于敏感数据，采用加密的方式保存在数据库中

• 采用动态短信密码、图片验证码等方式保证用户账号的安全

• 对于协议支付的支付密码输入框，建议使用密码控件

• 使用预编译语句集 防SQL注入

• 支持跨站点脚本编制保护（基于过滤器,拒绝包含指定为不允许的属性（参数）或字符串的任何用户请求）防

跨站点脚本攻击

• 支持跨站点请求伪造保护(启用跨站点请求伪造保护时，会要求在请求中包含一个名为 authToken 的指定

URL 参数)防请求伪造式网络钓鱼攻击

• 支持URL重定向过滤(启用 URL 重定向过滤时,拒绝所有尝试重定向到未授权站点的请求)防重定向网络钓鱼攻击

日志监控

• 记录查询登录日志

• 记录查询操作日志

• 记录查询授权日志

数据安全

• 敏感数据的加密传输和存储

• 重要业务数据的变更版本记录及恢复

•

### 程序安全

对目前主流的web应用，通过浏览器访问系统，基本上浏览器都支持开发者模式用于调试JavaScript代码。系统在生成环境正式部署运行后，要能对JavaScript代码进行压缩、混淆及加密，避免通过修改运行时JavaScript代码入侵系统，改变系统行为。

后端java代码进行混淆，提高反编译的难度。

### 其他安全

网络安全、主机安全、数据库安全等基础设施安全以及安全管理体系此次不再描述。

## 性能设计

### 分布式微服务

微服务架构（Microservices Architecture）是一种架构风格（Architectural Style）和设计模式，提倡将应用分割成一系列细小的服务，每个服务专注于单一业务功能，运行于独立的进程中，服务之间边界清晰，采用轻量级通信机制（如HTTP/REST）相互沟通、配合来实现完整的应用，满足业务和用户的需求。

简而言之就是服务轻量级化和模块化，可独立部署。其带来的好处包括：解耦合程度更高，屏蔽底层复杂度；技术选型灵活，可方便其他模块调用；易于部署和扩展等。与NodeJs等其他语言相比，Java相对来说实现微服务较为复杂一些，开发一个Web应用需要经历编码－编译打包－部署到Web容器－启动运行四步，但是开源框架Spring Boot的出现，让Java微服务的实现变得很简单，由于内嵌了Web服务器，无需“部署到Web容器”，三步即可实现一个Web微服务。而且由于Spring Boot可以和Spring社区的其他框架进行集成，对于熟悉Spring的开发者来说很容易上手。

使用spring boot 使水平扩展变成了一件非常简单的事情，对于系统使用高峰完全可以容易的复制出多个应用。

### 读写分离

一般应用对数据库而言都是“读多写少”，也就说对数据库读取数据的压力比较大，对写入的实时性要求高，现在系统设计的思路是， 数据库划分为主从库，其中主库负责写入数据，称之为：写库； 其它库都是从库，负责读取数据，称之为：读库；系统设计的要求是：

1） 读库和写库的数据一致；

2） 写数据必须写到写库；

3） 读数据必须到读库；

因为采用了微服务架构，我们设计了对读操作和写操作进行分离，以使日常业务流畅运行。带来的价值：

1）增加了数据的冗余及备份，保证数据的安全。

2）通过将“读”操作和“写”操作分离到不同的数据库服务器上，降低对主服务器的CPU、内存、存储、网络资源的争用；

3）主服务器的增删改进行时，不影响查询服务器的查询，降低阻塞的发生，提高了并发性；

4）在应用程序提交了报表请求、长时间的查询请求时，不会导致长时间的锁表；

### 多级缓存

对于后端程序，系统中的热点数据，进行缓存，避免对数据库的频繁查询，提高性能。

对于前端程序，系统中的热点数据，进行客户端缓存，减少网络请求，减少带宽专业，减少服务器压力。

### 全文检索

结构化和非结构化数据，进行检索，是非常耗时的操作，使用了开源的Elasticsearch引擎实现。

ElasticSearch是一个基于Lucene的搜索服务器。它提供了一个分布式多用户能力的全文搜索引擎，基于RESTful web接口。Elasticsearch是用Java开发的，并作为Apache许可条款下的开放源码发布，是当前流行的企业级搜索引擎。设计用于云计算中，能够达到实时搜索，稳定，可靠，快速，安装使用方便。

### 动静分离

服务端接收来自客户端的请求中，有一部分是静态资源的请求，例如html,css,js和图片资源等等，有一部分是动态数据的请求。因为Tomcat处理静态资源的速度比较慢，所以我们可以考虑把所有静态资源独立开来，交给处理静态资源更快的服务器Nginx处理，而把动态请求交给tomcat处理，这样一来就能更好的提高并发，增强性能。

Nginx配置简洁，轻量级，占用的内存及资源更少。高并发，Nginx处理请求是异步非阻塞的，支持更多的并发连接，在高并发下Nginx 仍能保持低资源低消耗高性能。

## 部署方式

采用数据集中存储的方式 ，统一部署到服务器，用户使用浏览器直接访问服务。

### 虚拟化部署

对于linux服务器，推荐使用Docker进行部署。

Docker 是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的容器中，然后发布到任何流行的 Linux 机器上，也可以实现虚拟化。容器是完全使用沙箱机制，相互之间不会有任何接口（类似 iPhone 的 app）。几乎没有性能开销,可以很容易地在机器和数据中心中运行。最重要的是,他们不依赖于任何语言、框架包括系统。

　　由于其基于LXC的轻量级虚拟化的特点，docker相比KVM之类最明显的特点就是启动快，资源占用小。因此对于构建隔离的标准化的运行环境，构建自动化测试和持续集成环境，以及一切可以横向扩展的应用(尤其是需要快速启停来应对峰谷的web应用)。

### 云部署支持

系统支持部署到阿里云、华为云等云环境。

### 传统部署环境

**系统支持的运行环境:**

|  |  |
| --- | --- |
| 环境类型 | 支持系统 |
| CPU | Intel Xeon CPU E5 2.50GHz及以上 |
| 内存 | 建议16G或以上配置 |
| 数据库服务器操作系统 | Linux、MS Windows 2000 Advanced Server、MS Windows2003 Server、MS Windows2008 Server及以上 |
| 应用服务器操作系统 |
| WEB服务器 |
| 中间件 | Zookeeper、Redis、Tomcat、Nginx、金蝶、东方通、中创等国产中间件 |
| 数据库产品 | Mysql5.6、Mysql5.7、达梦等国产数据库 |
| JDK | 1.8 |

## 系统运行监控

### 底层基础设施监控

对服务器操作系统的IO、内存、网络等运行环境进行资源监控。

### 应用运行状态监控

对系统的吞吐量、响应时间、在线用户等进行性能监控。

## 前端框架

### 前端框架选型

采用目前流行的ES6+Vue2.0+Webpack+Element-UI+vueRouter+NodeJs+Sass技术栈，主要考虑以下因素：

1）性能表现优秀

2）生态丰富

3）国产的优秀框架

4）界面风格一致

5）学习曲线平滑

### 浏览器兼容

IE9及以上浏览器，chrome，firefox等主流浏览器都支持。

### 单页面应用

单页面应用的优势：

1)良好的交互体验

用户不需要重新刷新页面，获取数据也是通过Ajax异步获取，页面显示流畅。

2)减轻服务器压力

服务器只用出数据就可以，不用管展示逻辑和页面合成，吞吐能力会提高几倍；

3)共用一套后端程序代码

不用修改后端程序代码就可以同时用于Web界面、手机、平板等多种客户端；

4)良好的前后端工作分离模式

单页Web应用可以和RESTful规约一起使用，通过REST API提供接口数据，并使用Ajax异步获取，这样有助于分离客户端和服务器端工作。更进一步，可以在客户端也可以分解为静态页面和页面交互两个部分。

### 界面风格

使用一套标准的界面风格，更换皮肤随心所欲。

### 客户端运行环境要求

**客户端计算机配置要求：**

|  |  |
| --- | --- |
| 项目 | 配置 |
| CPU | Core i3 2.40GHz及以上 |
| 内存 | 建议2G或以上配置 |
| 硬盘 | 2G空闲空间 |
| 显示 | 分辨率至少1366x768，32位色或真彩色 |
| 操作系统 | Windows 7或以上版本 |
| 其它 | 主流浏览器如IE 9.0或以上版本、Chrome |