AchieveIt软件项目管理平台  
软件架构设计说明书

**V1.0**

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ ] 草稿  [√] 正式发布  [ ] 正在修改 | 文件标识： | AT-SSD |
| 当前版本： | 1.0 |
| 作 者： | 宁静、高尚、杨政达、王浩、徐雯蕾 |
| 完成日期： | 2020.03.15 |

版 本 历 史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本/状态 | 作者 | 参与者 | 起止日期 | 备注 |
| v0.1 | 宁静、高尚、杨政达、王浩 | 宁静、高尚、杨政达、王浩 | 2020.03.01-2020.03.08 | 大纲版本 |
| v1.0 | 宁静、高尚、杨政达、王浩、徐雯蕾 | 宁静、高尚、杨政达、王浩、徐雯蕾 | 2020.03.09-2020.03.15 | 正式版 |
|  |  |  |  |  |

目录

[1 概述 3](#_Toc35244387)

[2 设计目标和约束 4](#_Toc35244388)

[3 架构设计 5](#_Toc35244389)

[3.1 总体方案 5](#_Toc35244390)

[3.2 架构说明 7](#_Toc35244391)

[3.2.1 架构图及说明 7](#_Toc35244392)

[3.2.2 架构设计关键点 7](#_Toc35244393)

[3.2.3 高可用性设计 7](#_Toc35244394)

[3.2.4 高性能设计 8](#_Toc35244395)

[3.2.5 可扩展性设计 8](#_Toc35244396)

[3.2.6 安全性设计 8](#_Toc35244397)

[3.2.7 其他设计 8](#_Toc35244398)

[4 部署方案 9](#_Toc35244399)

# 概述

1.1 目的

该文档用以描述软件项目管理平台AchieveIt系统（以下简称“系统”或“本系统”）的整体软件结构设计、模块划分以及各个模块的范围和接口定义。

1.2 范围

本系统的目标是为四月科技有限公司提供能够统一管理公司各个项目的信息和数据，提高管理效率，挖掘潜在价值，支持合理决策的功能。系统范围是支持项目的基本情况、项目成员、项目成员权限、项目所需完成功能、项目状态等信息的管理和查询、能够执行项目立项流程和归档流程。以及提供跟踪评审问题和缺陷，提供项目风险，项目设备，项目成员工时等信息的管理和查询。同时，本系统也会向公司的配置管理系统输出配置访问权限信息以便进行项目资产的访问控制。

1.3 定义、首字母缩写、缩略语

* Vue.js：用于构建用户界面的渐进式[JavaScript](https://baike.baidu.com/item/JavaScript/321142)框架
* AOP：Aspect Oriented Programming，面向切面编程
* IntelliJ Idea：由JetBrains公司开发的Java编程语言的集成开发环境
* VSCode：Visual Studio Code，美国微软公司开发的针对于编写现代 Web 和云应用的跨平台源代码编辑器

1.4 引用

**《**ACHIEVEIT软件项目管理平台原始需求说明》

**《**ACHIEVEIT软件项目管理平台软件需求规格说明书》

1.5 软件架构概述

本系统采用分层模式，其中又分为表示层(Presentation layer)、业务逻辑层(Business Logic Layer)、数据访问层(Data access layer)。

表示层主要是Web前端，基于Vue.js开发，主要负责显示数据和接受用户输入，为网站的系统运行提供交互式操作界面。

表示层和业务逻辑层完全分离，通过axios跨域实现前后端数据通信。

业务逻辑层作为系统架构中体现核心价值的部分，包含领域对象模型、领域实体、业务规则、验证规则、业务流程。数据访问层实现了对数据表的查询，插入，更新，删除等操作，同时包括对象和数据表之间的映射以及对象实体的持久化。

# 设计目标和约束

2.1 性能

1. 网页任意操作响应时间小于一秒
2. 可以承载500人同时进行网页操作
3. 由客户端发往服务器端请求成功率不低于99%

2.2 成本

1. 服务器租借费用低于200元/月
2. 使用开源框架及数据库以降低项目成本

2.3 时间

1. 4月6日前交付1.0版代码
2. 4月22日前完成结项，交付全部项目资产

2.4 可靠性

1. 服务器端使用Logger进行log记录，告警使用log4j排查问题
2. 服务器端每个Entity都需要加toString()方法，便于出现问题时使用Logger排查
3. 每个RESTful接口都必须有对应的API文档
4. 服务器端数据库字段和Entity字段必须相同，保持数据的一致性，避免使用Result反复修改结果集

2.5 安全性

1. 客户端与服务器端通信内容中涉及用户隐私信息时使用post请求并使用MD5加密方法进行加密
2. 服务器端多项数据库操作前，必须在业务逻辑（Service层）的方法前使用@Transactional确保事务能够回滚
3. 服务器端不对数据库中的记录进行删除操作，在每张表新增deleted字段，默认值为False表示该记录未被删除，删除操作将该字段的值设置为True

2.6 合规性

1. 服务器端不使用@Valid注解，保证Entity的简便性
2. 服务器端使用GetMapping，PostMapping等代替RequestMapping，便于代码的理解和确保HTTP请求的正确性
3. 服务器端多表联合查询，使用EntityVo，禁止在Entity中直接添加属性
4. 服务器端Http请求统一返回 {errorCode, message, data} 的对象
5. 后端需进行统一异常处理

2.7 技术性

1. 客户端使用当下最流行的框架之一Vue.js进行开发，使用Vuex进行状态管理
2. 服务器端使用面向切面编程（AOP）技术简洁高效地实现Logger对业务逻辑层每个方法的记录
3. 服务器端使用面向切面编程（AOP）技术简洁高效地实现用户token的身份验证

2.8 兼容性

1. 支持现代浏览器Firefox、Chrome、Safari近期两个版本及Internet Explorer 10+
2. 使用版本控制工具进行项目开发，每位成员将自己的代码commit到各自的分支上，经过review之后才能将代码merge到master分支
3. 不允许轻易变更需求，擅自修改API和数据库表字段，确保项目的版本兼容性

2.9 其他

1. 服务器端数组对象使用序列化操作，将获得的字符串存储在数据库中，同时，可以将字符串通过反序列化操作转化为所需要的对象，从而避免在数据库中创建多余表
2. 客户端使用VSCode，服务器端使用IntelliJ Idea进行作为开发工具进行开发
3. 服务器端使用的JDK版本为1.8，通过maven进行项目的配置

# 架构设计

## 总体方案

本系统采用三层架构设计：

* 表示层：

1. Web前端：基于HTML/HTML5/Vue.js/CSS3开发web前端页面，支持所有兼容ECMAScript5的主流浏览器。展现层和数据层完全分离，通过跨域实现前后端数据通信。
2. 技术/平台：框架基于MVVM库Vue.js，UI采用ElementUI。
3. Restful接口：基于特定业务，采用Restful标准接口，对外提供数据服务。
4. 数据格式：JSON格式的字符串。

* 业务逻辑层：

系统架构中体现核心价值的部分，包含领域对象模型，领域实体，业务规则，验证规则，业务流程。

核心业务基于Spring Boot 架构实现，生成独立的微服务功能单元并自动提供相关配置，其中分为：

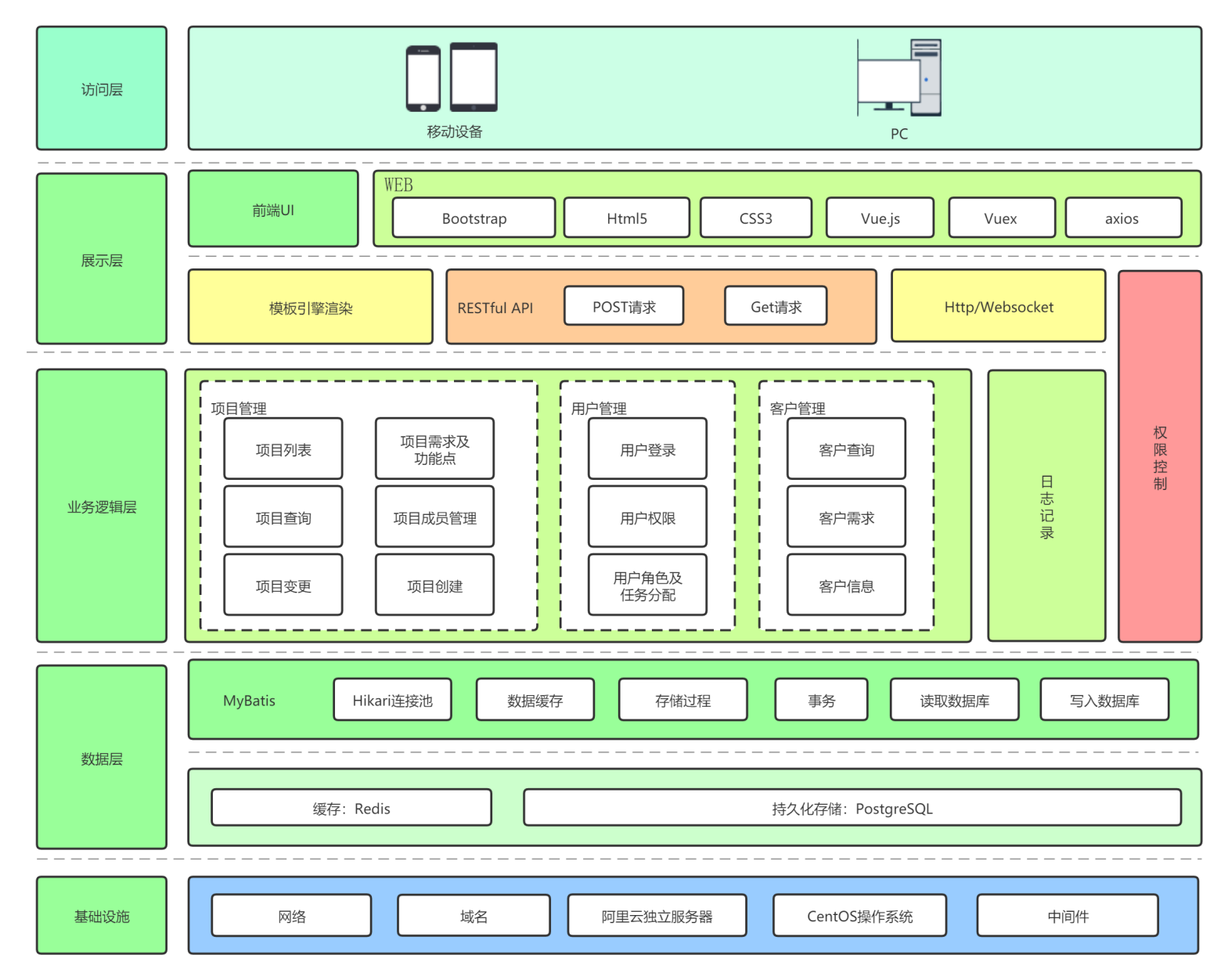
1. Controller层：表示层通过调用Controller层实现与业务逻辑层之间的沟通，同时负责分发用户的请求，并选择恰当的视图以用于显示，同时可以解释用户的输入并映射为模型层可以执行的操作。
2. Service层：Controller层调用Service层，负责主要用来处理业务逻辑，同时调用数据访问层进行相关的数据操作。

* 数据访问层：

1. 实现对数据库的查询，插入，更新，删除等操作。
2. 实现对象和数据表之间的映射以及对象实体的持久化。
3. 数据库事务遵循原子化操作。

## 架构说明

### 架构图及说明



### 架构设计关键点

1. 低耦合架构设计。动静分离，前后分离，能够有效提升系统的可用性
2. 高吞吐设计。连接池技术，缓存技术，负载均衡的服务器前端，提升吞吐量

### 高可用性设计

1. 部署在全球领先的阿里云服务器，提供99.99999%的可靠性
2. 部署多台服务器，负载均衡，避免单点失效
3. 数据库服务器和应用服务器分离，读写分离，保证了数据的可用性
4. 数据库服务器多点备份，提升了可用性
5. 自动监控，运维，在单点失效情况下能够快速恢复服务

### 高性能设计

1. 使用性能优异的Java SpringBoot框架，通过IO优化，有着良好的性能表现
2. 使用NoSql技术存储热点数据，提升了数据吞吐量
3. 多线程并发设计，线程池技术，处理大量请求
4. 使用消息队列，异步处理请求，避免阻塞
5. 动静分离，对于静态请求，使用高性能静态服务器响应，提升负载

### 可扩展性设计

1. 使用阿里云云服务，能够快速拓展集群，应对大负载下场景
2. 微服务设计，将后台服务解耦，提升功能拓展性
3. 通过Restful前后端分离，避免了前后端的耦合
4. 模块化开发，降低模块间耦合度，便于模块调整装卸

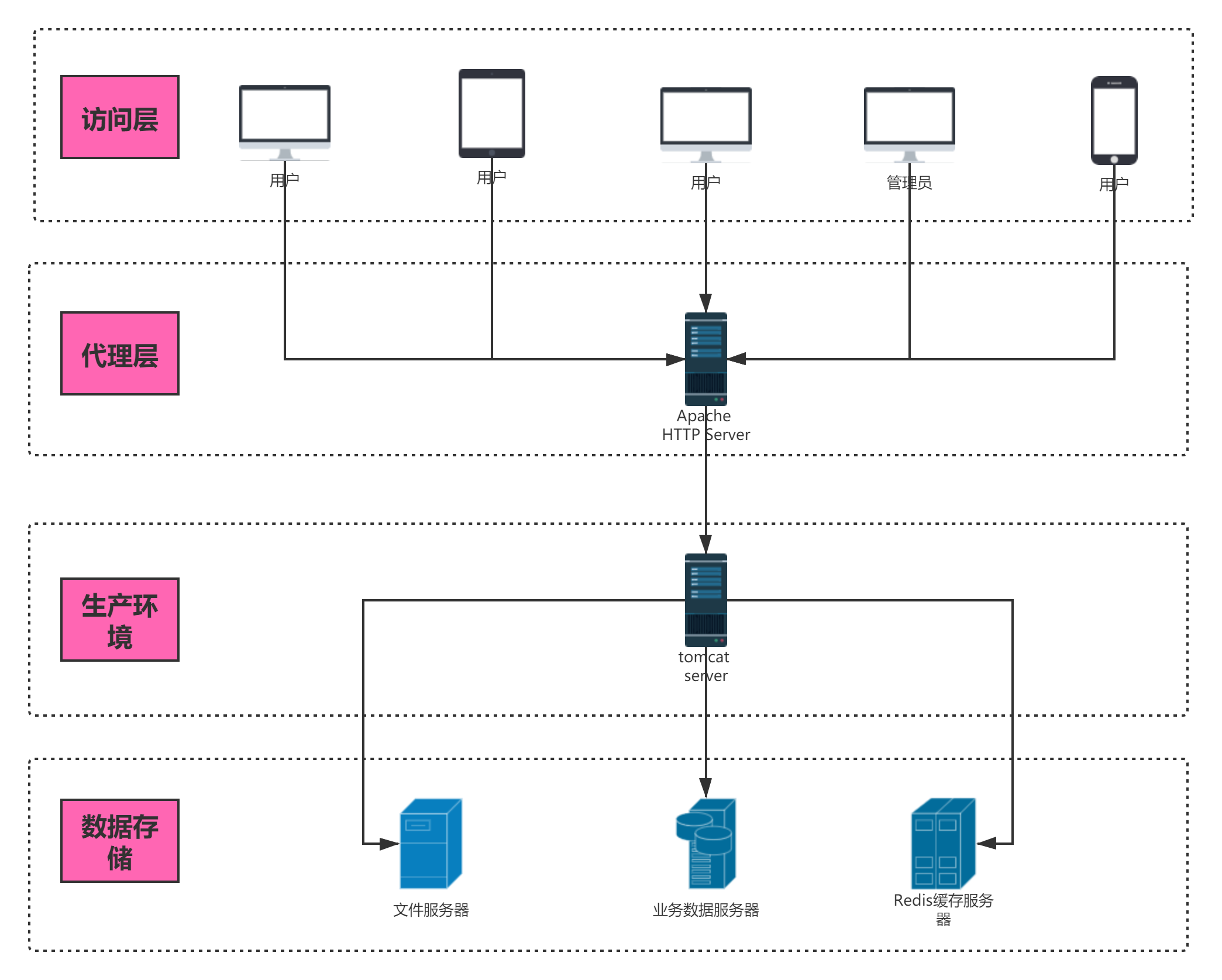
### 安全性设计

1. 通过过滤器，自动检查所有请求，对访问人员进行鉴权，阻止非法请求
2. 通过Token进行鉴权，长时间未访问，Token自动失效
3. 对不同人员进行权限设定，后台逻辑上阻止非权限操作
4. 系统通过阿里云基线功能对漏洞进行检查，防止非法越权
5. 通过Vue框架，前端能够对抵御诸如XSS等漏洞攻击，进一步提升安全性
6. 事务系统能够回滚无效操作，避免数据不一致性情况的发生

### 其他设计

1. 客户端使用HTML5、CSS3、JavaScript，服务器端使用Java开发语言进行开发
2. 产品中立项审批流程功能采用Workflow Engine来实现

# 部署方案



系统部署分为数据存储、生产环境、代理层、访问层四个级层。

1. 数据存储层分割为文件服务器、业务数据服务器、Redis缓存服务器，将三者分离部署在独立的服务器上来提升性能；
2. 代理层使用Apache HTTP Server，具有简单、速度快和性能稳定的优点；
3. 生产环境使用Apache Tomcat进行Java后端代码的部署，提供Servlet容器，具有占用的系统资源小，扩展性好的优点。