**省人力资源市场数据采集系统**

**概要设计说明书**

组长：曾紫飞

组员：孙浩辰 刘洋琳 隋劲丛

# 1.前言

## 1.1系统介绍和目的

省管理部门创建服务机构账号；通过账号登录系统，补充各级服务机构信息，经市管理部门备案后，市管理部门进行审核，全部通过后，报送省级管理部门；市级管理部门审核、汇总各区县公共及经营性人力资源服务机构报送的数据，上报省厅；省厅对数据进行汇总。

省级管理部门创建各级公共就业服务机构及经营性人力资源服务机构账号，人力资源市场用户录入和修改人力资源市场用户详细信息的内容，按照统一规范的模板填写和修改数据表，保存后上报地市、省备案。同时用户可以查询以往调查期人力资源市场数据的状态。

市级管理部门管理各级公共就业服务机构及经营性人力资源服务机构，市级管理部门录入用户名、密码、人力资源市场用户基本信息等内容。可以对录入的人力资源市场用户的信息进行修改、删除。但有上报数据的人力资源市场用户不能删除，对于不合格的信息可以退回，然后把已填写好的报表上报给省级单位。市级管理部门也可以查看以往调查期的报表，或者查看及审核人力资源市场用户上报的建档期和调查期数据，对本市监测点采集数据进行统计汇总并分析。

省级管理部门审核地市上报的数据并汇总上报，对于不合格的数据可以退回。查看所有已备案人力资源市场用户的详细信息，但不可以修改。还可以查看人力资源市场用户的汇总数据并使用图表分析，对全省已创建账号进行条件查询，发布、删除或者查看通知信息。

系统对新增或修改数据调查期设置时限，监控CPU、内存、硬盘等信息和应用系统的一些信息。根据需要建立多种角色，不同角色对应不同功能。并且预定义一些角色。

## 1.2术语定义

|  |  |
| --- | --- |
| **简称/术语** | **说明** |
| **前端** | 前端即网站前台部分，运行在PC端，移动端等浏览器上展现给用户浏览的网页。随着互联网技术的发展，HTML5，CSS3，前端框架的应用，跨平台响应式网页设计能够适应各种屏幕分辨率，完美的动效设计，给用户带来极高的用户体验。 |
| **后端** | 后端是指用于管理网站前端的一系列操作。 |
| **框架（Framework）** | 框架是整个或部分系统的可重用设计，表现为一组抽象构件及构件实例间交互的方法;另一种定义认为，框架是可被应用开发者定制的应用骨架。可以说，一个框架是一个可复用的设计构件，它规定了应用的体系结构，阐明了整个设计、协作构件之间的依赖关系、责任分配和控制流程，表现为一组抽象类以及其实例之间协作的方法，它为构件复用提供了上下文关系。 |

## 1.3参考资料

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **文档名称** | **最后修订时间** | **版本号** | **作者** |
| **1** | Requirements\_Specification\_5.0.pdf | 2018.11.14 | 5.0 | 需求分析标准组 |
| **2** | 阿里巴巴Java开发手册（详尽版）.pdf | 2018.05.20 | 1.4.0 | 阿里巴巴集团技术团队 |
| **3** | 省人力资源市场数据采集系统需求 | 2020.10 | 1.0 | 闫波 |

# 2.系统概述

## 2.1系统目标

开发一个省人力资源市场数据采集系统，运行流程大致如下：省管理部门创建服务机构账号；通过账号登录系统，补充各级服务机构信息，经市管理部门备案后，市管理部门进行审核，全部通过后，报送省级管理部门；市级管理部门审核、汇总各区县公共及经营性人力资源服务机构报送的数据，上报省厅；省厅对数据进行汇总。

## 2.2设计约束

系统软件硬件环境如下：

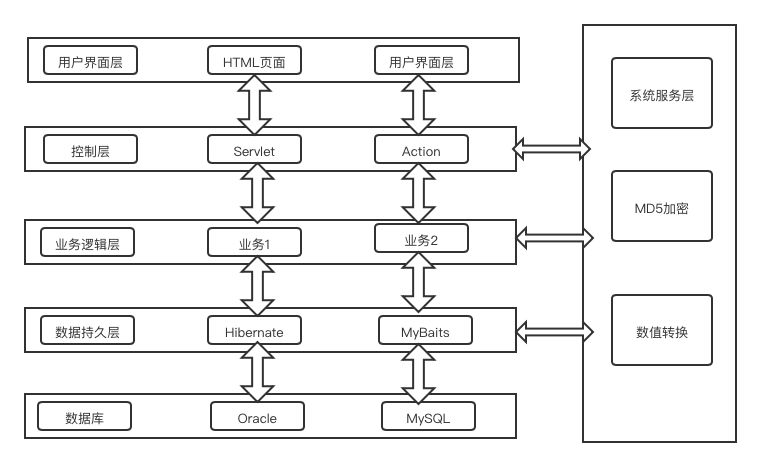
* 防火墙：百兆互联网防火墙，可以利用机房环境的防火墙设备。
* 应用服务器要求如下：CPU主频不低于2.8 GHz 的至强CPU、4GB内存、100G以上硬盘空间、支持Raid5。
* 备份服务器：建议利用托管机房环境的备份设备。

软件环境要求如下：

* + Windows 2003以上操作系统
  + Oracle 10G以上数据库
  + TOMCAT 6.0以上应用中间件
  + 反病毒软件服务器版

要求采用ASP.NET 2.0技术平台，编程语言为C#或VB.NET，后台数据库为MS SQL SERVER 2012，系统架构采用三层以上架构，并且按照微软企业级架构标准进行程序的开发工作，在每个关系到效率和性能的环节中，都先按不同方案进行测试，从中选择最佳方案来实施。

## 2.3系统总体架构



### **2.4功能结构**

2.4.1监测点子系统模块划分

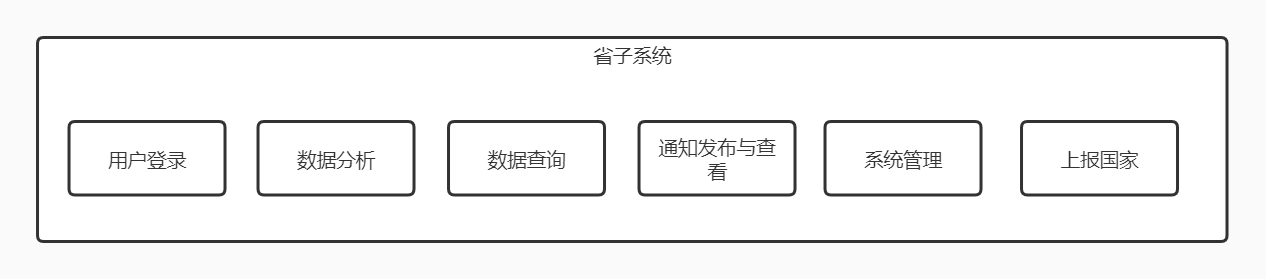


|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **子系统名称** | **模块名称** | **模块功能** |
| **1** | 监测点子系统 | 备案信息 | 保存修改后的人力资源市场用户详细信息 |
| **2** | 数据填报 | 查询在调查期内人力资源市场用户所填报的信息 |
| **3** | 暂时保存人力资源市场用户的信息 |
| **4** | 上报地市、省 |
| **5** | 数据查询 | 查询以往调查期内人力资源市场用户调查期内的数据状态 |

2.4.3市子系统模块划分

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 子系统名称 | 模块名称 | 模块功能 |
| 1 | 市子系统 | 用户管理 | 创建账号并录入用户基本信息 |
| 2 | 将用户信息向省级管理部门上报备案 |
| 3 | 对本市已创建账号进行条件查询 |
| 4 | 对本市监测点采集数据进行分析 |
| 5 | 删除用户 |
| 6 | 数据处理 | 查看数据报表 |
| 7 | 审核数据报表 |
| 8 | 数据汇总 |
| 9 | 通知处理 | 发布通知 |
| 10 | 修改通知 |
| 11 | 删除通知 |
| 12 | 查看某用户发布通知 |

2.4.3省子系统模块划分



|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **子系统名称** | **模块名称** | **模块功能** |
| **1** | 省子系统 | 系统管理 | 新增或修改调查期/上报时限 |
| **2** | 建立省和地市的用户/用户管理 |
| **3** | 根据需要建立多种角色/角色管理 |
| **4** | 查看当前系统工作情况/系统监控 |
| **5** | 数据查询 | 查看用户的汇总数据/数据汇总 |
| **6** | 查看所有已备案用户详细信息/用户备案 |
| **7** | 对全省已创建帐号进行条件查询 |
| **8** | 对查询结果导出 |
| **9** | 数据分析 | 审核上报数据/报表管理 |
| **10** | 显示各市用户的数量和占比/取样分析 |
| **11** | 查看各调查期内各市用户需求人数、需求比重、环比和同比的变化情况/固定报表 |
| **12** | 使用折线图、表格形式对比两个调查期的人力资源市场数据变动情况/图表分析-对比分析 |
| **13** | 使用折线图、表格形式展示多个连续调查期内的需求人数、求职人数的变动情况/图表分析-趋势分析 |
| **14** | 通知发布与查看 | 发布通知 |
| **15** | 浏览通知 |
| **16** | 上报国家 | 上报国家 |
| **17** | 用户登录 | 用户登录 |

# 3.接口设计

## 3.1外部接口设计

无

## 3.2内部接口设计

内部接口主要是设计应用于系统的界面获取和更新数据和资源的接口。为了让系统可以适应更复杂的功能，我们将API设计得尽可能抽象。按API的动词分为如下几类：

1. GET (获得)：获取一个具体的资源或者一个资源列表。
2. POST （创建）： 创建一个新的资源。
3. PUT （更新）：更新一个资源。
4. DELETE （删除）：删除一个资源。

上面4类API表示系统模块间可以按需要相互获取、创建、更新和删除数据和资源。

# 4.数据库设计

## 4.1数据库总体设计

### 4.1.1设计思路与原则

设计思路：根据对象模型的设计、界面原型设计说明书和RESTful API的设计，整理总结出网页需要的资源、有存储需要的的数据结构。分析数据之间的关联后设计出ER图，再设计数据结构对应的存储表。

原则：表的设计尽可能在冗余和数据的效率间取得平衡，即达到尽可能好的范式的规范。

### 4.1.2数据库类型和版本

本系统支持MySQL Community Server 8.0.13

### 4.1.3数据库规范

1. 表名、字段名必须使用小写字母或数字，禁止出现数字开头，禁止两个下划线中间只出现数字。如课程表名称为course、小组话题表名称为group\_topic。
2. 禁用保留字，如 desc、range、match、delayed 等，具体参考MySQL官方保留字。
3. 视图名称均以 ”view\_” 为前缀。存储过程均以 “sp\_” 为前缀。所有表均以自增id为主键，无外键。表达是与否概念的字段，必须使用 is\_xxx 的方式命名，数据类型是 unsigned tinyint （1 表示是，0 表示否）。

说明：任何字段如果为非负数，必须是 unsigned。 注意：POJO 类中的任何布尔类型的变量，都不要加 is 前缀，所以，需要在<resultMap>设置 从 is\_xxx 到 Xxx 的映射关系。数据库表示是与否的值，使用 tinyint 类型，坚持 is\_xxx 的 命名方式是为了明确其取值含义与取值范围。

### 4.1.4表空间规划

MySQL没有真正意义上的表空间管理。MySQL的Innodb包含两种表空间文件模式，默认的共享表空间和每个表分离的独立表空间。一般来说，当数据量很小的时候建议使用共享表空间的管理方式。数据量很大的时候建议使用独立表空间的管理方式。

初步拟定将教师表、学生表、讨论课表放入独立表空间中，其他表放入共享表空间中。

### 4.1.5数据字典定义

人力资源市场用户备案信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型(长度) | 允许为空 | 是否主键 | 注释 |
| marketame | string | NO | PRI | 显示人力资源市场名称 |
| district | string | NO |  | 显示所属地市、市县、区域，不可修改 |
| contact | string | NO |  | 中文、英文 |
| contactphone | string | NO |  | 显示联系人手机 |
| contactnumber | string | NO |  | 格式必须符合（区号）+电话 |
| fax | string | YES |  | 格式必须符合（区号）+电话 |

需求、供求人数信息总表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型(长度) | 允许为空 | 是否主键 | 注释 |
| demandnumber | int | NO |  | 需求人数 |
| jobseekernumber | int | NO |  | 求职人数 |
| rate1 | double | NO |  | 岗位空缺与求职人数比率 |
| rate2 | double | NO |  | 需求比重(%) |
| rate3 | double | NO |  | 求职比重(%) |
| Number1 | int | NO |  | 缺口数 |
| industry | enum | NO |  | 产业（第一、第二、第三产业） |
| occupation | enum | NO |  | 行业（农林渔-国际组织） |
| enterprise | enum | NO |  | 企业类型（事业、机关等） |
| jobtype | enum | NO |  | 职业类型（单位负责人-其他） |
| Job1 | enum | NO |  | 职业（需求大于求职） |
| Job2 | enum | NO |  | 职业（需求小于求职） |
| jobseekertype | enum | NO |  | 求职人员类别 |
| gender | enum | NO |  | 性别 |
| age | enum | NO |  | 年龄 |
| education | enum | NO |  | 文化程度 |
| techrank | enum | NO |  | 技术等级 |

### 4.1.6存储过程和触发器

无触发器

4.1.7数据备份策略

考虑到如下情况，决定采取的备份策略：

1）系统使用频率高，数据更新快。

2）主库或者从库宕机，需要数据的备份。

3) 需要升级数据库或者是需要增加一个从库的时候。

4）人为的DDl或者是DML的语句，导致主从库的数据消失

5）跨机房的灾备，需要备份数据到远端程序。

备份日程安排:

1）中小公司，全量备份每天一次，在业务流量低谷如深夜2点至凌晨6点执行全量备份，备份时会锁表。平时（凌晨6点至深夜2点的时间）每隔1分钟增量备份一次（每隔1分钟对binlog日志做切割然后被分到其他的服务器上，或者本地其他硬盘中）。

2）单台数据库，如和增量。使用rsync（配合定时任务频率大点或者inotify，主从复制），把所有的binlog备份到远程服务器，尽量做主从复制。

3）一主五从，会有一个从库做备份，延迟同步。

部分恢复策略及恢复场景情景：

1）主或者从库宕机（硬件损坏），不需要增量恢复，主库宕机，只需要把其中的一个同步最快的从库提升为主库即可。主库宕机，只要选择更新最快的从库，提升为主库即可。从库宕机，停用该从库，进行正常修复。

2）人为操作数据库SQL语句破坏主库：在数据库主库内部命令行进行误操作，会导致所有的数据库（包括主从库）的数据丢失，例如：在主库执行了drop database test这样的删除语句，这时候所有的从库也会执行这个drop语句，从而导致所有的数据库上的数据库的数据丢失，这样的场景下面需要进行增量恢复的。

3）只有一个主库

只有一个主库的情况下，首先应该做定时的全量备份（每天一次）以及增量备份或者写到网络文件系统中。

## 4.2数据库详细设计

### 4.2.1物理表清单

无

### 4.2.2物理表设计

无

### 4.2.3逻辑表清单

|  |  |  |
| --- | --- | --- |
| 序号 | 名称 | 注释 |
| 1 | 人力资源市场用户信息备案表 | 记录用户基本信息 |
| 2 | 供求总体表 | 记录总体供求信息 |
| 3 | 按产业分组需求人数表 | 记录需求人数 |
| 4 | 按行业分组需求人数表 | 记录需求人数 |
| 5 | 按用人单位性质分组需求人数表 | 记录需求人数 |
| 6 | 按职业分组供求人数表 | 记录供求人数 |
| 7 | 职业大类表 | 记录需求大于求职缺口最大的十个职业 |
| 8 | 职业小类表 | 记录需求小于求职缺口最大的十个职业 |
| 9 | 按求职人员类别分组的求职人数表 | 记录求职人数 |
| 10 | 按性别分组供求人数表 | 记录供求人数 |
| 11 | 按年龄分组供求人数表 | 记录供求人数 |
| 12 | 按职业分组供求人数表 | 记录供求人数 |
| 13 | 按文化程度分组供求人数表 | 记录供求人数 |
| 14 | 按技术等级分组供求人数表 | 记录供求人数 |

### 4.2.4逻辑表设计

见4.1.5

# 5.界面设计

## 5.1界面风格说明

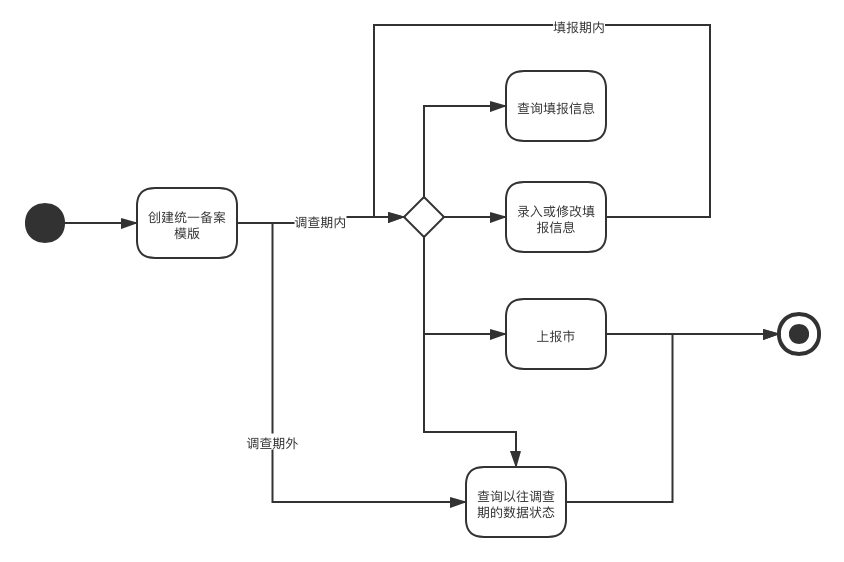
|  |  |
| --- | --- |
| **信息项（输入输出）** | **完全根据标准组需求文件确定的界面信息项。** |
| **布局** | 基本按照标准文件进行布局，但在具体实现时会稍作修改和美化。 |
| **配色** | 主色：  警示色：  状态色： |

\*该版式暂未运用上规定配色，仅用于表示信息项。

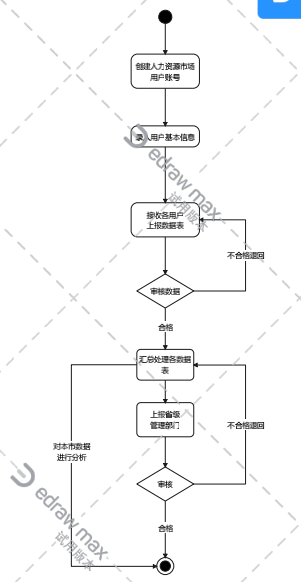
## 5.2界面流程图

### 

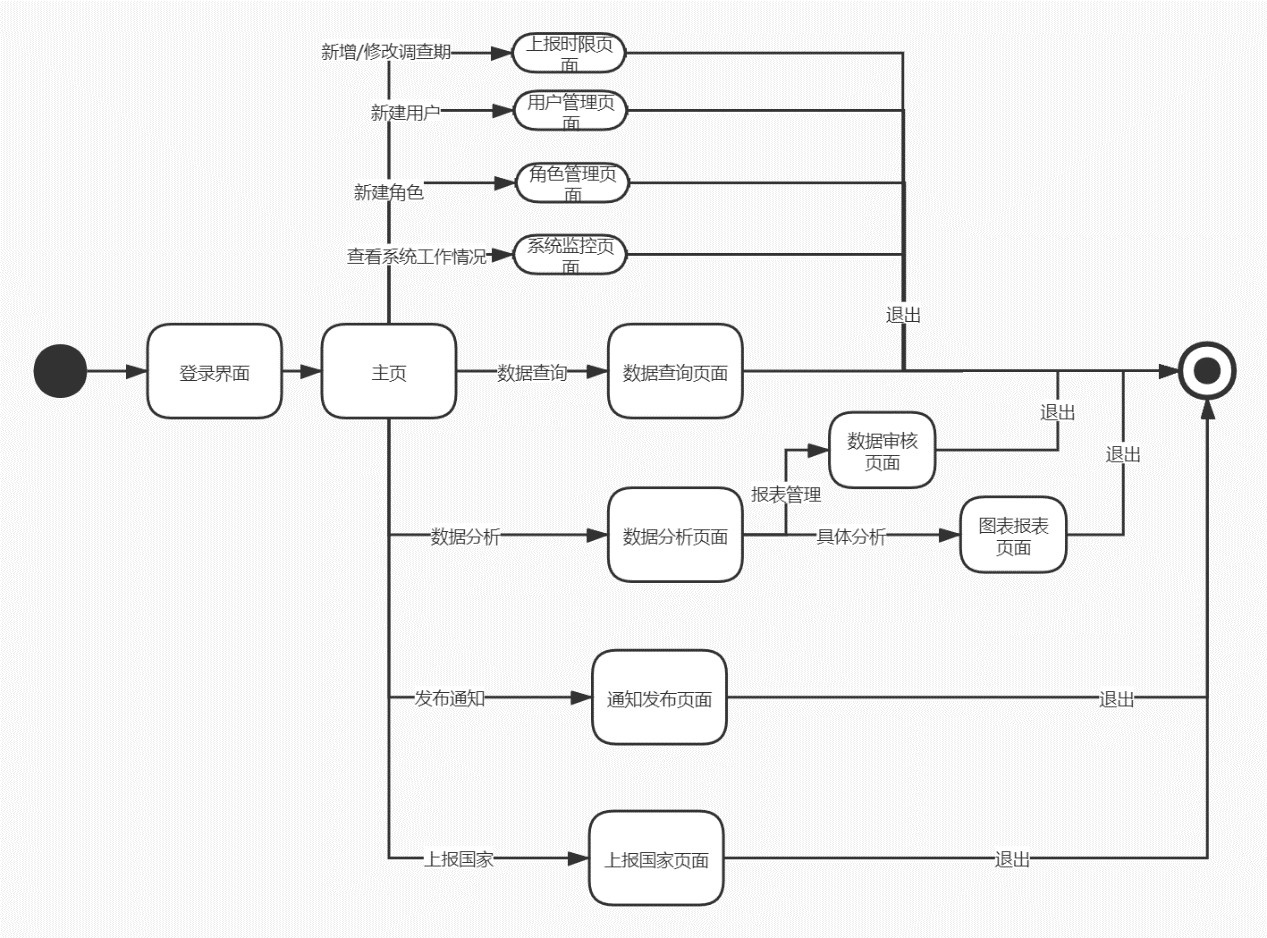
### 5.2.1监测点系统界面流程图



### 5.2.2市系统界面流程图



### 5.2.3省系统界面流程图



# 6.错误处理机制

## 6.1出错信息设计

1. 数据库连接错误。

数据库设置不正确或SQL Sever异常造成的异常，如对应的用户名和密码在数据库内不匹配造成的登录失败、查询相应讨论课结果显示异常等。

发生此类错误，系统提示本次操作失败，提醒用户联系管理员检查数据库，错误信息被记录在系统日志。

1. 输入错误

输入不规范造成的异常，如密码输入错误，新建讨论课信息输入不合法等。

发生此类错误，系统提示重新操作，错误信息被记录在系统日志。

1. 操作错误

操作不规范造成的异常，如用户未登录尝试执行其他需要身份认证的操作等

发生此类错误，系统提示操作不规范，提示规范的操作并中止此操作。

1. 不可预知错误

由未知异常引发的异常。

发生此类错误，系统进行数据库备份，错误信息被记录在系统日志。

## 6.2异常情况处理

1. 数据库连接错误

处理：取消本次操作，提醒用户联系管理员检查数据库。

1. 输入错误

处理：提醒用户输入错误，然后进行再次操作。

1. 操作错误

处理：提示规范的操作并中止此操作。

1. 不可预知错误

处理：进行数据库备份，帮助开发者完善此程序。

# 7.安全机制

## 7.1网络安全设计

1. 由于数据的传输上需要通过网络传输，为了将资料进行保密，需要在网络的传输过程中对数据进行加密。这个工作主要是准备网络包，及解开网络包这两个模块完成，它们各对数据进行加密和解密还原的工作。在加密算法选择上将使用RSA加密算法。具体算法可参照《Computer Network》
2. 通过监测、限制、更改跨越防火墙的数据流，尽可能地对外部屏蔽网络内部的信息、结构和运行状况， 以此来实现网络的安全保护。
3. 系统的硬件，软件及其系统中的数据受到保护，不会因为偶然的或者恶意的原因而遭到破坏，更改和泄露，系统连续可靠正常的运行，网络服务不会中断。
4. 监控与入侵检测系统作为系统端的监控进程,负责接受进入系统的所有信息,并对信息包进行分析和归类,对可能出现的入侵及时发出报警信息,同时如发现有合法用户的非法访问和非法用户的访问,监控系统及时断开访问连接,并进行追踪检查.

## 7.2接口安全设计

外部接口：

1. 用户界面：在用户界面部分，根据需求分析结果，需要一个用户友善界面。服务程序界面要做到操作简，易于管理，出错显示错误提示函数
2. 硬件接口：在输入方面，对鼠标进行输入处理。
3. 软件接口：服务器程序使用对MySQL的接口，进行对数据库的所有访问，做到对数据的保存。在网络软件接口方面，使用一种无差错的传输协议，采用滑动窗口方式对数据进行网络传输及接受，保证了过程的安全性。

内部接口：

内部接口方面，各个模块之间采用函数调用、参数传递、返回值的方式进行信息传递。接口传递的信息以数据结构封装了的数据，保证其准确性及安全性

## 7.3数据安全设计

1. 用户标识和鉴别：每次用户要求进入系统时，由系统核对账号密码，通过鉴定后才提供系统的使用权。
2. 存取控制：通过用户权限定义和合法权检查确保只有合法权限的用户访问数据库，所有未被授权的人员无法存取数据。角色权限固定。部分数据如密码等需要加密存储。
3. 视图机制：为不同的用户定义视图，通过视图机制把要保密的数据对无权存取的用户隐藏起来，从而自动地对数据提供一定程度的安全保护。
4. 防止SQL 注入：加入其他的附加逻辑判断条件，采用正则表达式等

## 7.4应用安全设计

1. 人为故意破坏数据库系统，使其不能恢复

处理：系统日志备份信息

1. 正常数据库收到攻击

处理：系统检查内存是否泄露、网络是否遭受攻击等可行时运行

1. 未经授权越权篡改数据库信息

处理：数据库访问控制机制

1. 身份假冒

处理：提供身份认证与鉴权机制

# 8.运维设计

## 8.1监控设计

1. 进程监控

系统的 CPU使用率，平均负载，内存使用情况，Swap 使用情况，网络接口流量速率，磁盘 I/O 速率，挂载分区的空间使用率以及进程状态等信息，超过设定的阈值则发生警报

1. 网络端口监控

服务器端口号无法访问时发出警报

1. 日志监控

日志内容发生不合法修改时发出警报

1. 数据库监控

数据库内部数据发生不合法修改时发出警报

## 8.2维护设计

1.系统应用程序维护。一旦程序发生问题或业务发生变化，就需要进行程序的修改和调整。

2.数据维护。业务处理对数据的需求是不断发生变化的，除了系统中主体业务数据的定期正常更新外，还有许多数据需要进行不定期的更新，或随环境或业务的变化而进行调整，以及数据内容的增加、数据结构的调整。此外，数据的备份与恢复等，都是数据维护的工作内容。

3.代码维护。随着系统应用范围的扩大，应用环境的变化，系统中的各种代码都需要进行一定程度的增加、修改、删除，以及设置新的代码。

# 9.关键技术点

## 9.1技术要点清单

|  |  |
| --- | --- |
| **前端部分** | **前端页面生成** |
| 前端路由 |
| 前端状态管理 |
| **前后端交互** | Ajax |
| 跨域请求 |
| **后端部分** | MVC |
| ORM |

## 9.2技术要点

### 9.2.1前端页面生成

前端页面生成主要用到了前端模板技术，本项目采用了Vue框架结合Element.ui组件库构建前端用户界面。

Vue.js是一套用于构建用户界面的渐进式框架。与其它大型框架不同的是，Vue被设计为可以自底向上逐层应用。Vue的核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。另一方面，当与现代化的工具链以及各种支持类库结合使用时，Vue 也完全能够为复杂的单页应用提供驱动。

Element UI 是一套采用 Vue 2.0 作为基础框架实现的组件库，一套为开发者、设计师和产品经理准备的基于 Vue 2.0 的组件库，提供了配套设计资源，帮助网站快速成型。

### 9.2.2前端路由

路由是根据不同的 url 地址展示不同的内容或页面。

早期的路由都是后端直接根据 url 来 reload 页面实现的，即后端控制路由。后来页面越来越复杂，服务器压力越来越大，随着 ajax（异步刷新技术） 的出现，页面实现非 reload 就能刷新数据，让前端也可以控制 url 自行管理，前端路由由此而生。

本项目采用的前端路由工具为vue-router。vue-router是Vue.js官方的路由管理器。它和Vue.js的核心深度集成，让构建单页面应用变得易如反掌。包含的功能有：

1. 嵌套的路由/视图表
2. 模块化的、基于组件的路由配置
3. 路由参数、查询、通配符
4. 基于 Vue.js 过渡系统的视图过渡效果
5. 细粒度的导航控制
6. 带有自动激活的 CSS class 的链接
7. HTML5 历史模式或 hash 模式，在 IE9 中自动降级
8. 自定义的滚动条行为

### 9.2.3前端状态管理

本项目使用vuex进行前端页面的状态管理。vuex是专门为vue.js开发的状态管理模式。采用集中式存储管理应用中所有组件的状态，并以响应的规则保证状态以一种可预测的方式发生变化。vuex有5个核心概念，分别为state、getter、mutation、action、module。这五个核心概念都围绕着一个核心，就是store(仓库)，将组件的共享状态抽取出来，存到仓库中，以一个单例模式管理，任何一个组件都可以获取到管理的数据或者触发管理的状态。

vuex中一个store(仓库)包括了行为(actions)、突变(mutations)、状态(state)，完成的vuex的一个流程为：一个组件(components)中的一个方法调用(dispatch)了actions，随后actions提交(commit)了对应的mutations，随后mutations操作了state中的状态数据，当状态改变的时候，组件就会重新渲染，实现了即时的局部更新。

在遇到多个组件共享状态，多个视图依赖于同一个状态，不同视图的行为需要改变同一个状态的时候，就可以用vuex来解决这些问题。

### 9.2.4 Ajax

Ajax 即“Asynchronous Javascript And XML”（异步 JavaScript 和 XML），是指一种创建交互式网页应用的网页开发技术。通过在后台与服务器进行少量数据交换，Ajax 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

本项目使用axios实现Ajax来完成前后台数据流转的功能。Axios是基于promise用于浏览器和node.js的http客户端。支持以下功能：

1. 支持浏览器和node.js
2. 支持promise
3. 能拦截请求和响应
4. 能转换请求和响应数据
5. 能取消请求
6. 自动转换JSON数据
7. 浏览器端支持防止CSRF(跨站请求伪造)

### 9.2.5跨域请求

跨域是指a页面想获取b页面资源，如果a、b页面的协议、域名、端口、子域名不同，所进行的访问行动都是跨域的，而浏览器为了安全问题一般都限制了跨域访问，也就是不允许跨域请求资源。注意：跨域限制访问，其实是浏览器的限制。

本项目使用Nginx响应静态资源（浏览器请求前端界面），并转发请求到后端（请求动态数据）。

### 9.2.6 MVC

MVC全名是Model View Controller，是模型(model)－视图(view)－控制器(controller)的缩写，一种软件设计典范，用一种业务逻辑、数据、界面显示分离的方法组织代码，将业务逻辑聚集到一个部件里面，在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。MVC被独特的发展起来用于映射传统的输入、处理和输出功能在一个逻辑的图形化用户界面的结构中。

本项目使用Spring MVC作为后端框架。SpringMVC是一种基于Java，实现了Web MVC设计模式，请求驱动类型的轻量级Web框架，即使用了MVC架构模式的思想，将Web层进行职责解耦。基于请求驱动指的就是使用请求-响应模型，框架的目的就是帮助我们简化开发，SpringMVC也是要简化我们日常Web开发。

### 9.2.7 ORM

ORM即对象关系映射，是一种程序技术，用于实现面向对象编程语言里不同类型系统的数据之间的转换。从效果上说，它其实是创建了一个可在编程语言里使用的--“虚拟对象数据库”。

本项目使用MyBatis实现数据库与对象之间的映射。MyBatis是一款优秀的持久层框架，它支持定制化SQL、存储过程以及高级映射。MyBatis避免了几乎所有的JDBC代码和手动设置参数以及获取结果集。MyBatis可以使用简单的XML或注解来配置和映射原生信息，将接口和Java的POJOs(Plain Old Java Objects,普通的Java对象)映射成数据库中的记录。